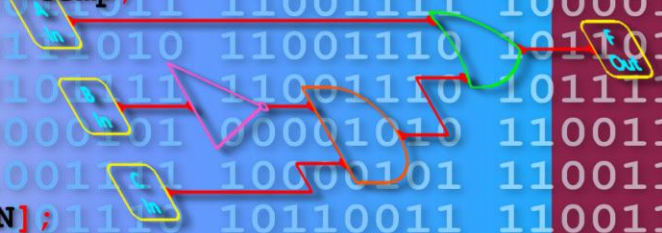
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΠΟΛΙΤΙΣΜΟΥ

ΠΛΗΡΟΦΟΡΙΚΗ ΚΑΙ ΕΠΙΣΤΗΜΗ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Γ ΛΥΚΕΙΟΥ

Σημειώσεις και ασκήσεις

```
#include <iostream>
#include<string>
using namespacestd;
struct students{
string name;
int grade;
};
void insertionSort(students stu[],int S){
  students temp;
int j;
for (int i=1; i<S; i++){
      temp = stu[i];
      j = i - 1;
 while(j>=0&& stu[j].grade<temp.grade
 || (stu[j].grade==temp.grade
          && stu[j].name>temp.name)) {
          stu[j+1] = stu[j];
          j--;
 }
      stu[j+1] = temp;
  }
}
int main(){
int N;
cin>> N;
  students st[N];
for(int i=0; i<N; i++)
cin>> st[i].name >> st[i].grade;
  insertionSort(st,N);
for(int i=0; i<N; i++)
cout<<st[i].name <<""<< st[i].grade <<endl;
  return0;
}
```



ΠΑΙΔΑΓΩΓΙΚΟ ΙΝΣΤΙΤΟΥΤΟ ΚΥΠΡΟΥ
ΥΠΗΡΕΣΙΑ ΑΝΑΠΤΥΞΗΣ ΠΡΟΓΡΑΜΜΑΤΩΝ

Πληροφορική και Επιστήμη Ηλεκτρονικών Υπολογιστών

Γ΄ Λυκείου

Σημειώσεις και ασκήσεις

Συγγραφή: Η Ομάδα Υποστήριξης του Αναλυτικού Προγράμματος και των Συμβούλων Καθηγητών Πληροφορικής και Επιστήμης Ηλεκτρονικών Υπολογιστών:

Παύλος Παυλικκάς
Παναγιώτης Ηρακλέους

Ηλεκτρονική σελίδωση: Πόλα Μάκκουλα
Καθηγήτρια Πληροφορικής και Επιστήμης Ηλεκτρονικών Υπολογιστών

Εποπτεία: Μάριος Μιλτιάδου
Μιχάλης Τορτούρης
Σωκράτης Μυλωνάς
Επιθεωρητές Πληροφορικής και Επιστήμης Ηλεκτρονικών Υπολογιστών

Γλωσσική επιμέλεια: Παναγιώτης Πετρίδης
Λειτουργός Υπηρεσίας Ανάπτυξης Προγραμμάτων

Σχεδιασμός εξωφύλλου: Σωκράτης Μυλωνάς
Επιθεωρητής Πληροφορικής και Επιστήμης Ηλεκτρονικών Υπολογιστών

Επιμέλεια έκδοσης: Μαρίνα Άστρα Ιωάννου
Λειτουργός Υπηρεσίας Ανάπτυξης Προγραμμάτων

Συντονισμός έκδοσης: Χρίστος Παρπούνας
Συντονιστής Υπηρεσίας Ανάπτυξης Προγραμμάτων

Α' Έκδοση 2017

Β' Έκδοση 2018

Εκτύπωση: Printco Manufacturing & Trading Ltd

© ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΠΟΛΙΤΙΣΜΟΥ
ΠΑΙΔΑΓΩΓΙΚΟ ΙΝΣΤΙΤΟΥΤΟ ΚΥΠΡΟΥ
ΥΠΗΡΕΣΙΑ ΑΝΑΠΤΥΞΗΣ ΠΡΟΓΡΑΜΜΑΤΩΝ

ISBN: 978-9963-0-54-164-5

Στο εξώφυλλο χρησιμοποιήθηκε ανακυκλωμένο χαρτί σε ποσοστό τουλάχιστον 50%, προερχόμενο από διαχείριση απορριμμάτων χαρτιού. Το υπόλοιπο ποσοστό προέρχεται από υπεύθυνη διαχείριση δασών.

Πρόλογος

Με ιδιαίτερη χαρά και ικανοποίηση προλογίζω το έντυπο υλικό για την υποστήριξη της διδασκαλίας του μαθήματος της Πληροφορικής και της Επιστήμης των Ηλεκτρονικών Υπολογιστών στη Γ' Λυκείου.

Μέσα από την προσπάθεια διαμόρφωσης ενός νέου αναλυτικού προγράμματος, δρομολογήθηκαν και εφαρμόζονται αλλαγές με στόχο την αναβάθμιση και τον εκσυγχρονισμό του μαθήματος τόσο στο Γυμνάσιο όσο και στο Λύκειο. Το έντυπο αυτό υλικό δημιουργήθηκε ακριβώς, για να υποστηρίξει τη διδασκαλία και τη μάθηση στο πλαίσιο του ριζικά αναθεωρημένου σε περιεχόμενο και μεθοδολογική προσέγγιση μαθήματος της Πληροφορικής και της Επιστήμης Ηλεκτρονικών Υπολογιστών στη Γ' Λυκείου.

Η δομή του μαθησιακού υλικού βασίζεται στις επτά ενότητες του Αναλυτικού Προγράμματος, καλύπτοντας το φάσμα των γνώσεων, των ικανοτήτων, των δεξιοτήτων, των στάσεων και των αξιών που προκύπτουν από τον γενικό σκοπό του μαθήματος, να προετοιμάσει δηλαδή τους/τις μαθητές/τριες για την ένταξή τους στην Κοινωνία της Πληροφορίας. Τα εφόδια αυτά θα τους επιτρέψουν την υπεύθυνη, την ενσυνείδητη, την ασφαλή, την αποδοτική και τη δημιουργική χρήση σύγχρονων τεχνολογιών της Πληροφορικής και της Επιστήμης των Ηλεκτρονικών Υπολογιστών στον επαγγελματικό τομέα που θα επιλέξουν, στον κλάδο σπουδών που πιθανό να ακολουθήσουν αλλά και στην καθημερινότητά τους.

Το διδακτικό βιβλίο *Σημειώσεις και Ασκήσεις* δεν αποτελεί διδακτικό εγχειρίδιο αλλά υλικό αναφοράς, το οποίο περιέχει παραδείγματα, επεξηγήσεις και επιπρόσθετες πληροφορίες και χρησιμοποιείται ως πηγή άντλησης πληροφοριών και είναι στενά συνδεδεμένο με το Αναλυτικό Πρόγραμμα. Στην αρχή του κάθε κεφαλαίου διατυπώνονται οι διδακτικοί στόχοι, βασισμένοι στους δείκτες επιτυχίας και το περιεχόμενο αναπτύχθηκε με βάση τους δείκτες επάρκειας. Στο τέλος κάθε κεφαλαίου περιλαμβάνονται ποικίλες δραστηριότητες με διαφοροποιημένο δείκτη δυσκολίας, ώστε να είναι δυνατή η προσαρμογή του μαθήματος στις ικανότητες, στις απαιτήσεις και στις ανάγκες όλων των μαθητών/τριών. Υιοθετούνται σύγχρονα διδακτικά μοντέλα, που στηρίζονται στην προώθηση εξελιγμένων μεθόδων οικοδόμησης της γνώσης και συστηματικών τρόπων προσέγγισης για την επίλυση προβλημάτων, ώστε οι μαθητές/τριες να προετοιμάζονται για να λειτουργήσουν στο συνεχώς μεταβαλλόμενο περιβάλλον των τεχνολογιών της πληροφορίας και των επικοινωνιών.

Ευχαριστώ θερμά όλους τους συντελεστές της προσπάθειας αυτής, ιδιαίτερα την ομάδα υποστήριξης της εφαρμογής του Αναλυτικού Προγράμματος και των συμβούλων καθηγητών της Πληροφορικής και της Επιστήμης των Ηλεκτρονικών Υπολογιστών για τη δημιουργία του έντυπου υλικού αλλά και του ηλεκτρονικού υλικού που το συνοδεύει, του οικείους επιθεωρητές, τους λειτουργούς της Υπηρεσίας Ανάπτυξης Προγραμμάτων και ιδιαίτερα τους επιστημονικούς συνεργάτες, τον Δρα Χρύση Γεωργίου (Τμήμα Πληροφορικής του Πανεπιστημίου Κύπρου), τη Δρα Χαρούλα Αγγελή (Τμήμα Επιστημών της Αγωγής του Πανεπιστημίου Κύπρου) και τον Δρα Ανδρέα Ανδρέου (Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Ηλεκτρονικών Υπολογιστών και Πληροφορικής του Τεχνολογικού Πανεπιστημίου Κύπρου) για τη συμβολή τους στην ανάπτυξη του υλικού αυτού.

Δρ Κυπριανός Δ. Λούης

Διευθυντής Μέσης Εκπαίδευσης

ΠΕΡΙΕΧΟΜΕΝΑ

Εισαγωγή

Όταν τον Μάρτιο του 2009 ξεκίνησε η προσπάθεια για εκπόνηση του Νέου Αναλυτικού Προγράμματος του μαθήματος της Πληροφορικής και της Επιστήμης των Ηλεκτρονικών Υπολογιστών, τέθηκε ως προτεραιότητα η θεώρηση του μαθήματος μέσα από το πρίσμα των τριών πυλώνων της εκπαιδευτικής μεταρρύθμισης. Μέσα από το πρίσμα, δηλαδή, της απόκτησης ενός επαρκούς σώματος γνώσεων από τους μαθητές και τις μαθήτριες, την καλλιέργεια αξιών και την εκδήλωση συμπεριφορών που να συνάδουν με τη σύγχρονη έννοια της δημοκρατικής πολιτότητας και την καλλιέργεια κομβικών ικανοτήτων και δεξιοτήτων από τους μαθητές και τις μαθήτριες, ώστε να μπορούν να λειτουργούν στη διαμορφούμενη κοινωνία του 21ου αιώνα.

Στην επιτροπή για την εκπόνηση του Αναλυτικού Προγράμματος τότε συμμετείχαν ο ακαδημαϊκός Ανδρέας Ανδρέου (Αναπληρωτής Καθηγητής στο Τεχνολογικό Πανεπιστήμιο Κύπρου), οι επιθεωρητές Πληροφορικής Μάριος Μιλτιάδου και Μιχάλης Τορτούρης και οι καθηγητές Πληροφορικής Νικόλαος Ζάγγουλος, Ιωάννης Ιωάννου, Μηνάς Καραολής, Θεόδουλος Κωνσταντίνου, Μαρία Μαρδά, Σωκράτης Μυλωνάς και Παναγιώτης Παπέττας. Παρόλο που είχε προταθεί ένα ολοκληρωμένο αναλυτικό πρόγραμμα από την Α' Γυμνασίου μέχρι και την Γ' Λυκείου, σε πρώτη φάση εφαρμόστηκε μόνο μέχρι την Γ' Γυμνασίου.

Το 2014 η διαδικασία εφαρμογής του Νέου Αναλυτικού Προγράμματος αναστάλθηκε για ένα έτος με σκοπό την αξιολόγησή του. Η ανεξάρτητη επιτροπή αξιολόγησης η οποία εξέτασε το περιεχόμενο του Νέου Αναλυτικού Προγράμματος για το μάθημα της Πληροφορικής και της Επιστήμης των Ηλεκτρονικών Υπολογιστών δεν εντόπισε παραλείψεις και επισήμανε ότι αυτό ήταν βασισμένο σε διεθνή πρότυπα και ότι ήταν επιστημονικά επαρκές.

Τον Φεβρουάριο του 2015 διορίστηκε νέα επιτροπή για τη βελτίωση του Νέου Αναλυτικού Προγράμματος και την αναδιατύπωση του με συγκεκριμένο τρόπο (με τη μορφή Δεικτών Επιτυχίας και Δεικτών Επάρκειας), καθώς και την ετοιμασία Αναλυτικού Προγράμματος για τα μαθήματα που θα διδάσκονται στο Λύκειο, λαμβάνοντας υπόψη και τις παραμέτρους του Νέου Ωρολογίου Προγράμματος. Στην επιτροπή αυτή συμμετείχαν οι ακαδημαϊκοί Δρ Χρύσης Γεωργίου (Αναπληρωτής Καθηγητής, Τμήμα Πληροφορικής του Πανεπιστημίου Κύπρου), Δρ Χαρούλα Αγγελή (Αναπληρώτρια Καθηγήτρια, Τμήμα Επιστημών της Αγωγής του Πανεπιστημίου Κύπρου) και Δρ Ανδρέας Ανδρέου (Αναπληρωτής Καθηγητής, Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Ηλεκτρονικών Υπολογιστών και Πληροφορικής του Τεχνολογικού Πανεπιστημίου Κύπρου), οι Επιθεωρητές Πληροφορικής Μάριος Μιλτιάδου και Μιχάλης Τορτούρης, οι Βοηθοί Διευθυντές Πληροφορικής Σωκράτης Μυλωνάς και Νικόλαος Ζάγγουλος και οι καθηγητές/τριες Πληροφορικής Παύλος Παυλικκάς, Πόλα Μάκκουλα, Παναγιώτης Ηρακλέους, Κωνσταντίνος Σωφρονίου, Μιλτιάδης Χαριλάου, Ξένιος Ξενοφώντος, Σοφία Καζέλη, Ηλίας Θεοδώρου, Δημήτρης Μαυροβουνιώτης, Δημήτρης Χατζηπαντελής, Νικόλας Στρατής και Θωμάς Ιωάννου.

Η διατύπωση του Αναλυτικού Προγράμματος για τη Β' και τη Γ' Λυκείου ολοκληρώθηκε τον Ιούλιο του 2016. Παράλληλα αναπτύχθηκε και το υποστηρικτικό υλικό (έντυπο και ηλεκτρονικό), το οποίο έχετε στα χέρια σας.

Σε επίπεδο περιεχομένου, το Νέο Αναλυτικό Πρόγραμμα περιέχει σημαντικές αλλαγές από το αντίστοιχο παλαιότερο, αλλά δεν περιέχει μεγάλο αριθμό νέων θεμάτων. Τόσο το Νέο Αναλυτικό Πρόγραμμα όσο και το υποστηρικτικό υλικό είναι οργανωμένα με βάση επτά ενότητες, οι οποίες αναπτύσσονται παράλληλα από τη μία τάξη στην επόμενη και αντικατοπτρίζουν βασικούς θεματικούς άξονες της Πληροφορικής και Επιστήμης των Ηλεκτρονικών Υπολογιστών:

- Βασικές Έννοιες
- Το Υλικό/Αρχιτεκτονική Υπολογιστών
- Λειτουργικά Συστήματα
- Λογισμικό Εφαρμογών
- Δίκτυα και Διαδίκτυο
- Βάσεις Δεδομένων και Ανάπτυξη Πληροφοριακών Συστημάτων
- Αλγοριθμική Σκέψη, Προγραμματισμός και Σύγχρονες Εφαρμογές Πληροφορικής.

Στα μαθήματα της Β' και Γ' Λυκείου υπάρχει εξειδίκευση στα προσφερόμενα μαθήματα. Στο μάθημα *Πληροφορική και Επιστήμη Ηλεκτρονικών Υπολογιστών* δίνεται έμφαση στις Βασικές Έννοιες, στην Αρχιτεκτονική Υπολογιστών, στα Λειτουργικά Συστήματα, στις Βάσεις Δεδομένων και στην Ανάπτυξη Πληροφοριακών Συστημάτων και, κυρίως, στην Αλγοριθμική Σκέψη, στον Προγραμματισμό και στις Σύγχρονες Εφαρμογές Πληροφορικής, όπου παρατηρούνται οι κύριες αλλαγές, με την εισαγωγή της γλώσσας προγραμματισμού C++ και τις ενότητες για ανάπτυξη παιγνιδιών και εφαρμογών για φορητές συσκευές. Το Λογισμικό Εφαρμογών και τα Δίκτυα Ηλεκτρονικών Υπολογιστών αναπτύσσονται σε άλλα εξειδικευμένα μαθήματα του κλάδου της Πληροφορικής και της Επιστήμης των Ηλεκτρονικών Υπολογιστών.

Οι ενότητες αυτές αναπτύσσονται σταδιακά μέσα από διαβαθμισμένους στόχους/Δείκτες Επιτυχίας από τη μία τάξη στην επόμενη, παρέχοντας επιπρόσθετες ευκαιρίες στους/στις μαθητές/τριες που δεν μπόρεσαν να τους κατακτήσουν σε μία τάξη να το πράξουν σε επόμενη, με παράλληλη εμβάθυνση. Σημαντικό στοιχείο της νέας προσέγγισης είναι και η αλληλεξάρτηση των ενοτήτων, αλλά και η συγκεκριμενοποίηση και ο περιορισμός των εννοιών που θα πρέπει να γνωρίζουν οι μαθητές/τριες, με έμφαση στην εφαρμογή τους στην πράξη.

Οι σημειώσεις και οι δραστηριότητες που ακολουθούν ετοιμάστηκαν από τους υποστηρικτές της εφαρμογής του Νέου Αναλυτικού Προγράμματος και από τους συμβούλους της Πληροφορικής και της Επιστήμης των Ηλεκτρονικών Υπολογιστών. Μέσα από αυτές παρέχεται καθοδήγηση για το εύρος και το βάθος των γνώσεων και των δεξιοτήτων που χρειάζεται να αναπτύξουν οι μαθητές/τριες. Τονίζεται ότι οι σημειώσεις δεν αποτελούν διδακτικό εγχειρίδιο, αλλά αποτελούν σημείο αναφοράς περιλαμβάνοντας γνώσεις, επεξήγηση και παραδείγματα. Οι καθηγητές/τριες ενθαρρύνονται να στηρίξουν τη διδασκαλία τους στις σημειώσεις αυτές, αποφεύγοντας τη συστηματική αποστήθιση και ανάκληση ορισμών, αλλά με στόχο την εκμάθηση των βασικών γνώσεων και την εφαρμογή τους από τους/τις μαθητές/τριες.

Οι δραστηριότητες στο τέλος κάθε κεφαλαίου είναι διαβαθμισμένες, ώστε να δίνεται η ευκαιρία εφαρμογής της νέας γνώσης σε όλους/ες τους/τις μαθητές/τριες ανάλογα με τον βαθμό ετοιμότητάς τους, αρχίζοντας από το απλό και προχωρώντας στο σύνθετο. Μέσα από αυτές παρέχεται καθοδήγηση τόσο για το εύρος και το βάθος των γνώσεων και των δεξιοτήτων που χρειάζεται να αναπτύξουν οι μαθητές/τριες όσο και για τις προτεινόμενες μεθοδολογικές προσεγγίσεις. Έμφαση δίνεται στην πρακτική εφαρμογή των γνώσεων και των δεξιοτήτων και στη χρήση εργαλείων και λογισμικού από τους/τις μαθητές/τριες στους ηλεκτρονικούς υπολογιστές, προάγοντας μαθητοκεντρικές προσεγγίσεις και περιορίζοντας τη μετωπική διδασκαλία. Οι καθηγητές/τριες ενθαρρύνονται να τα προσαρμόσουν ή και να δημιουργήσουν και άλλες δραστηριότητες, οι οποίες να ανταποκρίνονται καλύτερα, τόσο στα ιδιαίτερα χαρακτηριστικά των μαθητών/τριών τους όσο και στη δική τους διδακτική προσέγγιση.

ΕΝΟΤΗΤΑ Γ1 Βασικές Έννοιες της Πληροφορικής και της Επιστήμης Ηλεκτρονικών Υπολογιστών

Γ1.1 Δυαδικό Σύστημα Αρίθμησης

Τι θα μάθουμε σε αυτό το κεφάλαιο:

- Να αναφέρουμε τον τρόπο αναπαράστασης των δεδομένων (δυαδικό σύστημα)
- Να μετατρέπουμε έναν αριθμό από το δεκαδικό σύστημα στο δυαδικό
- Να υπολογίζουμε το συμπλήρωμα ενός δυαδικού αριθμού ως προς 1 και ως προς 2
- Να αναπαριστούμε προσημασμένους δυαδικούς αριθμούς
- Να υπολογίζουμε το άθροισμα και τη διαφορά 2 δυαδικών αριθμών.

1. Δυαδικοί αριθμοί

Οι αριθμοί που χρησιμοποιούμε στην καθημερινή ζωή έχουν ως βάση τον αριθμό 10 (decimal numbers). Ένας αριθμός με βάση το 10 (δεκαδικός) όπως το 259, παριστάνει μία ποσότητα ίση με 2 εκατοντάδες (2×10^2), 5 δεκάδες (5×10^1) και 9 μονάδες (9×10^0). Αντίστοιχα, ένας δυαδικός αριθμός όπως το 1101, παριστάνει μία ποσότητα ίση με 1 οκτάδα (1×2^3), 1 τετράδα (1×2^2), 0 δυάδες (0×2^1) και 1 μονάδα (1×2^0). Άρα $(1101)_2=(13)_{10}$.

Αντίστοιχα, ο δυαδικός αριθμός 1010.01 είναι ίσος με $1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2}$. Άρα $(1010.01)_2=(10.25)_{10}$.

1.1 Παραδείγματα μετατροπής από δυαδικό σε δεκαδικό

$(10111)_2=(23)_{10}$

$(1010100)_2=(84)_{10}$

$(111.11)_2=(7.75)_{10}$

$(10000.101)_2=(16.625)_{10}$

Όπως έχουμε δει, η μετατροπή ενός δυαδικού αριθμού σε δεκαδικό γίνεται με την ανάπτυξη του αριθμού σε μία σειρά δυνάμεων του 2. Τι γίνεται όμως όταν θέλουμε να μετατρέψουμε έναν δεκαδικό αριθμό σε δυαδικό; Για να το πετύχουμε αυτό, ακολουθούμε τον παρακάτω αλγόριθμο:

Βήμα 1: Διαιρούμε τον αριθμό με το 2.

Βήμα 2: Σημειώνουμε το υπόλοιπο και διαιρούμε το πηλίκο που προέκυψε με τον αριθμό 2.

Βήμα 3: Επαναλαμβάνουμε το βήμα 2 για όσο το πηλίκο είναι μεγαλύτερο από το 0.

Βήμα 4: Ο δυαδικός αριθμός προκύπτει από τα υπόλοιπα των διαιρέσεων, **βάζοντας τα σε αντίστροφη σειρά**.

Στο πιο κάτω παράδειγμα θα μετατρέψουμε τον αριθμό 23 σε δυαδικό (10111).

1.2 Παράδειγμα μετατροπής από δεκαδικό σε δυαδικό

Αριθμός	Πηλίκο	Υπόλοιπο
23/2 =	11	1
11/2 =	5	1
5/2 =	2	1
2/2 =	1	0
1/2 =	0	1

Στην περίπτωση που θέλουμε να μετατρέψουμε το κλασματικό μέρος ενός αριθμού από δεκαδικό σε δυαδικό, εφαρμόζουμε τον πιο κάτω αλγόριθμο.

Βήμα 1: Πολλαπλασιάζουμε το κλασματικό μέρος με το 2.

Βήμα 2: Αν το αποτέλεσμα του πολλαπλασιασμού είναι μεγαλύτερο του 1, τότε το bit του αριθμού θα είναι 1, διαφορετικά θα είναι 0.

Βήμα 3: Πολλαπλασιάζουμε με το 2 το κλασματικό μέρος του προηγούμενου αποτελέσματος.

Βήμα 4: Αν το αποτέλεσμα του πολλαπλασιασμού είναι μεγαλύτερο του 1, τότε το bit του αριθμού θα είναι 1, διαφορετικά θα είναι 0.

Βήμα 5: Επαναλαμβάνουμε τα βήματα 3 και 4 μέχρι να βρούμε κλασματικό μέρος 0 ή να πετύχουμε την επιθυμητή ακρίβεια.

Στο παρακάτω παράδειγμα θα μετατρέψουμε τον αριθμό 0.6875 σε δυαδικό. Αν εφαρμόσουμε τον αλγόριθμο προκύπτει ότι $\mathbf{(0.6875)_{10}=(0.1011)_2}$.

1.3 Παράδειγμα μετατροπής από δεκαδικό σε δυαδικό

Αριθμός	Γινόμενο	Bit
0.6875 * 2 =	1.375	1
0.375 * 2 =	0.75	0
0.75 * 2 =	1.5	1
0.5 * 2 =	1	1

Στο παράδειγμα 1.4 θα μετατρέψουμε τον αριθμό 12.45. Εφαρμόζουμε τον αλγόριθμο για το ακέραιο μέρος που θα μας δώσει την μετατροπή 1100 και τον αλγόριθμο για το κλασματικό μέρος που θα μας δώσει την μετατροπή 0111001. Παρατηρούμε ότι η κλασματική μετατροπή θα μπορούσε να συνεχιστεί και ότι η μετατροπή μάς δίνει μία προσέγγιση του αριθμού 0.45. Εφαρμόζοντας τον αλγόριθμο, προκύπτει ότι $\mathbf{(12.45)_{10}=(1100.0111001)_2}$.

1.4 Παράδειγμα μετατροπής δεκαδικού αριθμού με υποδιαστολή σε δυαδικό

Ακέραιο Μέρος

Αριθμός	Πηλίκο	Υπόλοιπο
12/2 =	6	0
6/2 =	3	0
3/2 =	1	1
1/2 =	0	1

Δεκαδικό Μέρος

Αριθμός	Γινόμενο	Bit
0.45 * 2=	0.9	0
0.9 * 2 =	1.8	1
0.8 * 2 =	1.6	1
0.6 * 2 =	1.2	1
0.2 * 2 =	0.4	0
0.4 * 2 =	0.8	0
0.8 * 2 =	1.6	1

2. Συμπληρώματα (Complements)

Τα συμπληρώματα χρησιμοποιούνται στους ηλεκτρονικούς υπολογιστές για την απλοποίηση της πράξης της αφαίρεσης και για τις λογικές πράξεις. Υπάρχουν δύο τύποι συμπληρώματος για κάθε αριθμητικό σύστημα: το συμπλήρωμα ως προς τη βάση και το συμπλήρωμα ως προς τη μειωμένη βάση κατά 1. Στην περίπτωση των δυαδικών αριθμών θα έχουμε το συμπλήρωμα ως προς 2 (βάση) και το συμπλήρωμα ως προς 1 (μειωμένη βάση κατά 1).

2.1 Συμπλήρωμα ως προς 1

Το συμπλήρωμα ως προς 1 ενός δυαδικού αριθμού N με k ψηφία, προκύπτει αν αφαιρέσουμε από τον αριθμό 2^k-1 τον αριθμό N. Αν για παράδειγμα N=101, τότε 2^k-1 = 111, άρα 111-101=010. Όταν αφαιρούμε δυαδικά ψηφία από το 1, θα έχουμε ως αποτέλεσμα είτε 1-0=1 είτε 1-1=0.

1.5 Παράδειγμα συμπληρώματος ως προς 1 για 8-bit δυαδικούς αριθμούς

Το συμπλήρωμα ως προς 1 του 00011001 είναι 11100110

Το συμπλήρωμα ως προς 1 του 11010000 είναι 00101111

Το συμπλήρωμα ως προς 1 του 10101010 είναι 01010101

Το συμπλήρωμα ως προς 1 του 00111100 είναι 11000011

Παρατηρούμε ότι το συμπλήρωμα ενός δυαδικού αριθμού ως προς 1, προκύπτει αν μετατρέψουμε τα ψηφία που έχουν τιμή 1 σε 0 και αυτά που έχουν τιμή 0 σε 1.

2.2 Συμπλήρωμα ως προς 2

Το συμπλήρωμα ως προς 2 ενός δυαδικού αριθμού N με k ψηφία προκύπτει αν αφαιρέσουμε από τον αριθμό 2^k τον αριθμό N, για N ≠ 0 και αφαιρούμε 0 για N = 0. Αν, για παράδειγμα, N=10100 τότε 2^k= 100000, άρα 100000-10100=01100. Το συμπλήρωμα ως προς 2 μπορεί να σχηματιστεί αν αφήσουμε όλα τα λιγότερα σημαντικά ψηφία 0 (το λιγότερο σημαντικό ψηφίο ενός αριθμού είναι αυτό που βρίσκεται δεξιότερα) και το πρώτο ψηφίο 1, αναλλοίωτα, και μετατρέψουμε τα υπόλοιπα ψηφία που έχουν τιμή 1 σε 0 και αυτά που έχουν τιμή 0 σε 1.

1.6 Παράδειγμα συμπληρώματος ως προς 2 για 8-bit δυαδικούς αριθμούς

Το συμπλήρωμα ως προς 2 του 00011001 είναι 11100111

Το συμπλήρωμα ως προς 2 του 11010000 είναι 00110000

Το συμπλήρωμα ως προς 2 του 10101010 είναι 01010110

Το συμπλήρωμα ως προς 2 του 00111100 είναι 11000100

2.3 Αριθμητική Πρόσθεση 2 δυαδικών αριθμών

Η πρόσθεση 2 δυαδικών δεν διαφέρει πολύ από την πρόσθεση δεκαδικών αριθμών. Η διαφορά είναι ότι έχουμε κρατούμενο όταν συμπληρωθεί μία δυάδα και όχι όταν συμπληρωθεί μία δεκάδα. Αν έχουμε, για παράδειγμα, να προσθέσουμε τους αριθμούς 00001011 (ο αριθμός 11 στο δεκαδικό) και 00000011 (ο αριθμός 3 στο δεκαδικό), θα πάρουμε τον αριθμό 00001110 (ο αριθμός 14 στο δεκαδικό).

1.7 Παραδείγματα πρόσθεσης δυαδικών αριθμών

```
  10001011      00001111      10000000      11101111
+ 00101010    + 00000001    + 10000000    + 11111111
  ========      ========      ========      ========
  10110101      00010000     1 00000000    1 11101110
```

Στα δύο τελευταία παραδείγματα έχουμε υπερχείλιση του τελευταίου ψηφίου. Αυτό σημαίνει ότι το αποτέλεσμα της πρόσθεσης δεν χωρά σε έναν δυαδικό αριθμό μεγέθους 8 ψηφίων.

2.4 Αριθμητική Αφαίρεση 2 δυαδικών αριθμών

Στην περίπτωση που θέλουμε να αφαιρέσουμε 2 δυαδικούς αριθμούς Χ και Υ όπου Χ>=Υ, τότε χρησιμοποιούμε τον ίδιο τρόπο με τον οποίο κάνουμε αφαίρεση στο δεκαδικό σύστημα, χρησιμοποιώντας την έννοια του δανεικού (borrow).

1.8 Παραδείγματα αφαίρεσης δυαδικών αριθμών Χ-Υ για Χ>=Υ

```
  01100100          00001010          10000001
- 00101000        - 00000111        - 00011101
  ========          ========          ========
  00111100          00000011          01100100
```

Μία εναλλακτική λύση, που είναι πιο αποτελεσματική στην υλοποίηση της με ψηφιακά κυκλώματα, είναι η μέθοδος που χρησιμοποιεί συμπληρώματα.

Έστω ότι έχουμε τους δυαδικούς αριθμούς Χ=01010100 και Υ=01000011 και θέλουμε να υπολογίσουμε το αποτέλεσμα της πράξης Χ – Υ.

2.4.1 Αφαίρεση (Χ-Υ) με συμπλήρωμα ως προς 2

(α) Υπολογίζουμε το συμπλήρωμα ως προς 2 του αφαιρετέου Υ (10111101).

(β) Προσθέτουμε τον αριθμό Χ με το συμπλήρωμα του αριθμού Υ:

```
   01010100
+  10111101
   ========
 1 00010001
```

(γ) Αγνοούμε την υπερχείλιση του τελευταίου ψηφίου.

Επαληθεύουμε ότι 01010100 $(84)_{10}$ – 01000011 $(67)_{10}$ = 00010001 $(17)_{10}$.

2.4.2 Αφαίρεση (Χ-Υ) με συμπλήρωμα ως προς 1

(α) Υπολογίζουμε το συμπλήρωμα ως προς 1 του αφαιρετέου Υ (10111100).

(β) Προσθέτουμε τον αριθμό Χ με το συμπλήρωμα του αριθμού Υ:

```
   01010100
+  10111100
   ========
 1 00010000
```

(γ) Αγνοούμε την υπερχείλιση του τελευταίου ψηφίου.

(δ) Προσθέτουμε στο αποτέλεσμα τον αριθμό 1.

```
   00010000
+  00000001
   ========
   00010001
```

2.4.3 Αφαίρεση (Υ-Χ) με συμπλήρωμα ως προς 2

(α) Υπολογίζουμε το συμπλήρωμα ως προς 2 του αφαιρετέου Χ (10101100).

(β) Προσθέτουμε τον αριθμό Υ με το συμπλήρωμα του αριθμού Χ:

```
   01000011
 + 10101100
   ========
   11101111
```

(γ) Το αποτέλεσμα που προκύπτει θα είναι ένας αρνητικός αριθμός, οπότε προσθέτουμε στην αρχή το πρόσημο – και βρίσκουμε το συμπλήρωμα ως προς 2 του αποτελέσματος (ο αριθμός 11101111 θα μετατραπεί σε -00010001).

2.4.4 Αφαίρεση (Υ-Χ) με συμπλήρωμα ως προς 1

(α) Υπολογίζουμε το συμπλήρωμα ως προς 1 του αφαιρετέου Χ (10101011).

(β) Προσθέτουμε το Υ με το συμπλήρωμα του Χ:

```
   01000011
 + 10101011
   ========
   11101110
```

(γ) Το αποτέλεσμα που προκύπτει θα είναι ένας αρνητικός αριθμός, οπότε προσθέτουμε στην αρχή το πρόσημο – και βρίσκουμε το συμπλήρωμα ως προς 1 του αποτελέσματος (ο αριθμός 11101110 θα μετατραπεί σε -00010001).

3. Δυαδικοί αριθμοί με πρόσημο

Οι θετικοί ακέραιοι αριθμοί μπορούν να γραφτούν ως αριθμοί χωρίς πρόσημο. Για να γράψουμε όμως έναν αρνητικό δυαδικό αριθμό, χρειαζόμαστε έναν κοινό συμβολισμό. Αυτό μπορεί να γίνει με την υιοθέτηση του συμβολισμού, όπου το πρώτο ψηφίο από τα αριστερά δηλώνει το πρόσημο (0 για +, 1 για –). Η παράσταση αυτή ονομάζεται παράσταση προσημασμένου μέτρου (signed-magnitude conversion). Σύμφωνα με αυτή την παράσταση, ο αριθμός $(00000011)_2$ παρουσιάζει τον αριθμό $(+3)_{10}$, ενώ ο αριθμός $(10000011)_2$ παρουσιάζει τον αριθμό $(-3)_{10}$.

Ένας άλλος τρόπος αναπαράστασης αρνητικών αριθμών είναι με τη χρήση συμπληρώματος, που ονομάζεται παράσταση προσημασμένου συμπληρώματος (signed-complement conversion). Στην παράσταση προσημασμένου συμπληρώματος μπορούν να χρησιμοποιηθούν τόσο το συμπλήρωμα ενός δυαδικού ως προς 1 όσο και το συμπλήρωμα ενός δυαδικού αριθμού ως προς 2. Συνήθως, χρησιμοποιείται το συμπλήρωμα ως προς 2.

Για παράδειγμα, για τον αριθμό $(-3)_{10}$ η αναπαράσταση προσημασμένου συμπληρώματος ως προς 1 θα είναι 11111100, ενώ η αναπαράσταση ως προς 2 θα είναι 111111101. Να σημειώσουμε ότι οι θετικοί αριθμοί δεν αλλάζουν. Για παράδειγμα, ο αριθμός $(+3)_{10}$ στην παράσταση προσημασμένου συμπληρώματος είτε ως προς 1 είτε ως προς 2 παραμένει 00000011.

Βιβλιογραφία

1. Mano, M. M., Kime, C. R. (2014). *Logic and computer design fundamentals*. London: Pearson.
2. Κοζύρης, Ν. (2005). *Σημειώσεις στα Συστήματα Αρίθμησης – Δυαδική Παράσταση Αριθμών*. Αθήνα: Εθνικό Μετσόβιο Πολυτεχνείο.

Ασκήσεις Κεφαλαίου

Άσκηση 1.1

Να μετατρέψετε τους πιο κάτω δεκαδικούς αριθμούς σε δυαδικούς.

(α) 19
(β) 48
(γ) 254
(δ) 5.90
(ε) 0.05

Άσκηση 1.2

Να μετατρέψετε τους πιο κάτω δυαδικούς αριθμούς σε δεκαδικούς.

(α) 01010111
(β) 10010001
(γ) 10000001
(δ) 00110010.01
(ε) 01100100.11

Άσκηση 1.3

Να υπολογίσετε τα συμπληρώματα ως προς 1 και ως προς 2 των πιο κάτω δυαδικών αριθμών.

(α) 00110010
(β) 10001000
(γ) 00000000
(δ) 11111111
(ε) 10101010

Άσκηση 1.4

Αν A=00010110, B=01001011 και C=00101100, να υπολογίσετε το αποτέλεσμα των πράξεων A+B, B+C, B-C και C-A.

Άσκηση 1.5

Να αντιστοιχίσετε τα συμπληρώματα ως προς 2 της αριστερής στήλης, με τους δεκαδικούς αριθμούς της δεξιάς στήλης.

(α) 01100011	(1) 111
(β) 01111111	(2) 14
(γ) 00001000	(3) 99
(δ) 01101111	(4) 8
(ε) 00001110	(5) 127

ΕΝΟΤΗΤΑ Γ2 Το Υλικό / Αρχιτεκτονική Ηλεκτρονικών Υπολογιστών

Γ2.1 Στοιχεία Αρχιτεκτονικής

Τι θα μάθουμε σε αυτό το κεφάλαιο:

- Να απαριθμούμε τις βασικές λειτουργίες της άλγεβρας Boole
- Να αναφέρουμε τα αξιώματα της άλγεβρας Boole
- Να υπολογίζουμε το αποτέλεσμα βασικών λογικών πράξεων (λογικό άθροισμα, λογικό γινόμενο, λογική αντιστροφή) σε λογικές παραστάσεις δεδομένων των τιμών των μεταβλητών της
- Να αναφέρουμε και να εφαρμόζουμε τα θεωρήματα της άλγεβρας Boole σε λογικές παραστάσεις
- Να γράφουμε τον πίνακα αλήθειας βασικών πράξεων AND, OR και NOT
- Να διακρίνουμε και να γράφουμε τη λογική έκφραση ή τη λογική συνάρτηση που αντιστοιχεί σε μία περιγραφή ή ένα πρόβλημα
- Να αναγνωρίζουμε τη σχηματική αναπαράσταση των λογικών πυλών AND, OR, NOT, NAND, NOR, XOR και XNOR
- Να αποδίδουμε τον πίνακα αλήθειας του κάθε είδους λογικής πύλης
- Να διαχωρίζουμε τα σήματα/ μεταβλητές εισόδου, και εξόδου
- Να υπολογίζουμε το αποτέλεσμα στην έξοδο μίας λογικής πύλης για συγκεκριμένες τιμές στα σήματα εισόδου
- Να ακολουθούμε τη διαδρομή ενός σήματος μέσα από ένα λογικό κύκλωμα και να υπολογίζουμε την τιμή του, όταν εξέρχεται από κάθε λογική πύλη
- Να υπολογίζουμε το αποτέλεσμα στις εξόδους ενός λογικού κυκλώματος για συγκεκριμένες τιμές στα σήματα εισόδου
- Να σχεδιάζουμε τον πίνακα αλήθειας για ένα λογικό κύκλωμα
- Να αναλύουμε ένα πρόβλημα, για να προσδιορίζουμε το πλήθος των εισόδων και των εξόδων του λογικού κυκλώματος που το επιλύει
- Να αναλύουμε ένα πρόβλημα, για να σχεδιάζουμε τον πίνακα αλήθειας του λογικού κυκλώματος που το επιλύει
- Να σχεδιάζουμε ένα λογικό κύκλωμα το οποίο να αντιστοιχεί στη λύση του προβλήματος (στο χαρτί ή με τη χρήση κατάλληλου εργαλείου για δημιουργία λογικών κυκλωμάτων)
- Να γράφουμε την αλγεβρική συνάρτηση που αντιστοιχεί σε ένα λογικό κύκλωμα
- Να απλοποιούμε μία λογική συνάρτηση, χρησιμοποιώντας αλγεβρικούς κανόνες (θεωρήματα) της άλγεβρας Boole
- Να διατυπώνουμε μία λογική συνάρτηση ως άθροισμα ελάχιστων όρων
- Να διατυπώνουμε μία λογική συνάρτηση ως γινόμενο μέγιστων όρων
- Να δημιουργούμε έναν χάρτη Karnaugh ή έναν πίνακα αληθείας ως άθροισμα ελαχιστόρων και αντίστροφα
- Να συμπληρώνουμε έναν χάρτη Karnaugh, όταν είναι γνωστός ο πίνακας αληθείας
- Να προσθέτουμε δύο ακέραιους δυαδικούς αριθμούς και να καταγράφουμε το αποτέλεσμα
- Να διακρίνουμε τα βασικά λογικά κυκλώματα για εκτέλεση πρόσθεσης δυαδικών αριθμών
- Να σχεδιάζουμε το λογικό κύκλωμα ενός Ημιαθροιστή και ενός Πλήρους Αθροιστή
- Να συνθέτουμε κυκλώματα αθροιστών για εκτέλεση πρόσθεσης δύο αριθμών με 2 μέχρι 8 δυαδικά ψηφία.

1. Άλγεβρα Boole (Boolean Algebra)

Η εύκολη σχεδίαση πολύπλοκων ψηφιακών συστημάτων οφείλεται στη δυνατότητα που έχουμε να παραστήσουμε τις λειτουργίες τους με συμβολικό τρόπο. Η δυνατότητα αυτή προέρχεται από την αντιστοιχία που έχουν τα βασικά στοιχεία ενός ψηφιακού κυκλώματος (Λογικές Πύλες) με τις μαθηματικές-λογικές πράξεις ενός αλγεβρικού συστήματος που ονομάζεται «αλγεβρα Boole». Οι αρχές αυτής της άλγεβρας μάς επιτρέπουν να σχεδιάζουμε τα κυκλώματα γρήγορα και χωρίς λάθη. Για να μάθουμε λοιπόν τους κανόνες της σχεδίασης ψηφιακών κυκλωμάτων (ψηφιακή σχεδίαση), πρέπει πρώτα να κατανοήσουμε τις πράξεις και τις ιδιότητες της άλγεβρας Boole, η οποία έχει καταξιωθεί ως το βασικό εργαλείο για την ανάλυση και σχεδίαση κάθε ψηφιακού κυκλώματος.

Η άλγεβρα Boole ορίζεται ως μία αλγεβρική δομή, δηλαδή ένα μαθηματικό σύστημα, το οποίο περιέχει τα εξής:

(α) Ένα σύνολο από στοιχεία που έχουν κοινές ιδιότητες. Π.χ. Ένα σύνολο Α, που αποτελείται από πλήθος στοιχείων μπορεί να οριστεί περικλείοντας τα στοιχεία του μέσα σε άγκιστρα: Α = {1, 2, 3, 4}.

(β) Ένα σύνολο από πράξεις. Μία πράξη είναι ένας κανόνας που δηλώνει πως όταν συνδυάσουμε τα στοιχεία x και y του συνόλου Α, θα προκύψει το στοιχείο z, π.χ. η πράξη της πρόσθεσης δηλώνει πως όταν συνδυάσω το 3 με το 4, θα πάρω το 7.

(γ) Ένα σύνολο από αξιώματα. Αποτελούν τις βασικές παραδοχές του συστήματος από τις οποίες προκύπτουν όλες οι άλλες ιδιότητες της άλγεβρας, που ονομάζονται θεωρήματα.

Από την εποχή του Boole, έχουν δημιουργηθεί διάφορες παραλλαγές της άλγεβρας αυτής. Η σημαντικότερη παραλλαγή της η οποία στηρίζεται στην αξιωματική θεμελίωση της άλγεβρας του Boole, προτάθηκε από τον Huntington.

2. Ορισμός άλγεβρας Boole

Η άλγεβρα Boole ορίζεται, σύμφωνα με τη θεμελίωση του Huntington, ως μία αλγεβρική δομή Α:

```
A = { A, +, *, ', 0, 1 }
```

όπου:

(α) Το Α είναι ένα σύνολο στοιχείων που περιέχει δύο τουλάχιστον στοιχεία

(β) Τα σύμβολα (+, *, ') δηλώνουν πράξεις επάνω στα στοιχεία του Α όπου:

- η πράξη + ονομάζεται λογικό άθροισμα (OR)
- η πράξη * ονομάζεται λογικό γινόμενο (AND)
- η πράξη ' ονομάζεται λογική αντιστροφή (NOT)

(γ) Τα 0 και 1 παριστάνουν ειδικά στοιχεία του Α που ονομάζονται σταθερές του συστήματος.

Οι πράξεις του αθροίσματος (+), του γινομένου (*) και της αντιστροφής (') ορίζονται σχετικά με το σύνολο Α={0,1}, από τον ακόλουθο πίνακα αντιστοιχίας ο οποίος ονομάζεται Πίνακας Αλήθειας:

A	B	OR A + B	AND A * B	NOT A'
0	0	0	0	1
0	1	1	0	1
1	0	1	0	0
1	1	1	1	0

(δ) Για κάθε στοιχείο a, b, c του συνόλου Α, οι πιο πάνω πράξεις ικανοποιούν τα αξιώματα της άλγεβρας Boole (Huntington).

3. Αξιώματα της άλγεβρας Boole (Huntington)

A0	**a, b ∈ A και a ≠ b, το σύνολο A έχει δύο τουλάχιστον στοιχεία**

Σύμφωνα με το αξίωμα A0, το σύνολο A πρέπει να περιέχει τουλάχιστον δύο στοιχεία. Στα στοιχεία του A συμπεριλαμβάνονται οι σταθερές 0 και 1.

A1	**a + b ∈ A, a * b ∈ A, Αξίωμα της κλειστότητας**

Σύμφωνα με το αξίωμα A1, οι πράξεις του λογικού αθροίσματος και του λογικού γινομένου είναι κλειστές, δηλαδή οι πράξεις αυτές για κάθε ζεύγος στοιχείων a, b που ανήκει στο A αντιστοιχούν πάντοτε με ένα στοιχείο c, το οποίο επίσης ανήκει στο σύνολο A.

A2	**a + b = b + a, a * b = b * a, Αξίωμα της αντιμεταθετικής ιδιότητας**

Σύμφωνα με το αξίωμα A2, το αποτέλεσμα των πράξεων είναι το ίδιο είτε στο στοιχείο a προστεθεί (ή πολλαπλασιαστεί) το b είτε στο στοιχείο b προστεθεί (ή πολλαπλασιαστεί) το a.

A3	**a + (b * c) = (a + b) * (a + c), a * (b + c) = (a * b) + (a * c)** **Αξίωμα της επιμεριστικής ιδιότητας**

Το αξίωμα A3 δηλώνει ότι ισχύει ο επιμερισμός όχι μόνο του γινομένου στο άθροισμα, a * (b + c), αλλά επιπλέον το άθροισμα επιμερίζεται στο γινόμενο, a + (b * c). Αυτό δεν συμβαίνει στην άλγεβρα των αριθμών και πρέπει να μας κάνει προσεκτικούς στις πράξεις με την άλγεβρα Boole.

A4	**a + 0 = a, a * 1 = a, Αξίωμα των ουδέτερων στοιχείων**

Το αξίωμα A4 δηλώνει ότι στις πράξεις του λογικού αθροίσματος και του λογικού γινομένου υπάρχουν δύο ειδικά στοιχεία του συνόλου A, το 0 και το 1, αντίστοιχα, τα οποία ονομάζονται ουδέτερα στοιχεία. Όσες φορές προσθέτουμε το 0 ή πολλαπλασιάζουμε με το 1, το αποτέλεσμα δεν αλλάζει.

A5	**a + a' = 1, a * a' = 0, Αξίωμα του συμπληρώματος**

Το αξίωμα A5 δηλώνει ότι για κάθε στοιχείο a του A, υπάρχει πάντοτε ένα στοιχείο, που ονομάζεται συμπλήρωμα ή αντίστροφο του a και συμβολίζεται σαν a', το οποίο σε συνδυασμό με το a δίνει σαν αποτέλεσμα τη σταθερά 1 ή 0. Να σημειωθεί ότι το a' δεν είναι αντίθετο του a, δηλαδή a + a' ≠ 0.

Συγκρίνοντας την άλγεβρα Boole με την άλγεβρα των αριθμών, παρατηρούμε ότι έχουν αρκετές ομοιότητες. Η άλγεβρα Boole, όμως, διαφέρει σε ορισμένα βασικά σημεία από την άλγεβρα των φυσικών αριθμών, τα οποία θα πρέπει να επισημάνουμε:

(α) Δεν έχει αντίστροφες πράξεις, π.χ. δεν υπάρχει πράξη αντίστοιχη της αφαίρεσης ή της διαίρεσης, επομένως δεν μπορούμε να μεταφέρουμε στοιχεία από το ένα μέλος μίας εξίσωσης στο άλλο με τον τρόπο που γίνεται στην άλγεβρα των αριθμών.

(β) Ισχύει ο επιμερισμός του αθροίσματος στον πολλαπλασιασμό (αξίωμα A3), δηλαδή a + (b * c) = (a + b) * (a + c), κάτι που δεν ισχύει στην άλγεβρα των αριθμών.

(γ) Δεν υπάρχουν αντίθετα στοιχεία, επομένως π.χ., αν ένα στοιχείο εμφανίζεται και στα δύο μέλη μίας εξίσωσης δεν μπορούμε να το απαλείψουμε, δηλαδή αν a+c=b+c, δεν συνεπάγεται ότι a = b.

4. Αρχή του Δυϊσμού

Η Αρχή του Δυϊσμού (Duality principle) δηλώνει πως:

Μία σχέση Α ονομάζεται δυϊκή (dual) μίας άλλης Β, εάν η Α προκύπτει από τη Β:

(α) με αμοιβαία εναλλαγή των πράξεων του γινομένου και του αθροίσματος.

(β) με αμοιβαία εναλλαγή των σταθερών 0 και 1.

Για παράδειγμα, αν ισχύει η έκφραση a + 1 = 1, τότε ισχύει και η έκφραση a * 0 = 0.

Η αρχή του δυϊσμού στην άλγεβρα Boole οδηγεί στο εξής θεώρημα:

Θεώρημα του Δυϊσμού: Αν μία σχέση της άλγεβρας Boole είναι αληθής, τότε θα είναι αληθής και η δυϊκή της.

Π.χ. αν (a + a) * (b' + 1) = a, τότε θα αληθεύει και η σχέση (a * a) + (b' * 0) = a

5. Λογική Έκφραση

Μία έκφραση της μορφής F = a + (b + c) * a + a * b + (b + c) * b, η οποία:

(α) Παραθέτει στη σειρά σύμβολα λογικών μεταβλητών, δηλαδή στοιχεία που παίρνουν τιμές από το σύνολο Α, π.χ. a, b, c και

(β) μεταξύ των μεταβλητών καθορίζει ορισμένες λογικές πράξεις

αποτελεί μία λογική έκφραση (boolean expression).

Η λογική έκφραση είναι ο τρόπος με τον οποίο διατυπώνονται τα θεωρήματα και οι σχέσεις της άλγεβρας Boole. Όταν υπολογίζουμε μία λογική έκφραση, πρέπει να ακολουθούμε την προτεραιότητα των τελεστών, που είναι εξ' ορισμού: παρενθέσεις, αντιστροφή, γινόμενο και τελευταία η πρόσθεση.

Π.χ. η έκφραση a * b + a * c ισοδυναμεί με την (a * b) + (a * c).

Παράδειγμα 2.1

Να υπολογιστούν οι τιμές των ακόλουθων λογικών εκφράσεων για τις τιμές των μεταβλητών a=0, b=1 και c=1:

(α) F = a + b * c

F = 0 + (1 * 1)	- προτεραιότητα πράξεων
F = 1 * 1	- αξίωμα Α4 (ουδέτερων στοιχείων)
F = 1	- αξίωμα Α4 (ουδέτερων στοιχείων)

(β) F = (a + b) * (a + c)

F = a + (b * c) - αξίωμα Α3 (επιμεριστική ιδιότητα)

F = 0 + (1 * 1)

F = 1 * 1 - αξίωμα Α4 (ουδέτερων στοιχείων)

F = 1 - αξίωμα Α4 (ουδέτερων στοιχείων)

(γ) Δίνεται η λογική έκφραση F = (a + b) * c

Να βρείτε η δυϊκή της F (Fd) και να υπολογίσετε τις τιμές και των δύο.

Η δυϊκή της F είναι Fd = (a * b) + c (Θεώρημα του Δυϊσμού)

F = (0 + 1) * 1

F = (1) * 1 - αξίωμα Α4 (ουδέτερων στοιχείων)

F = 1 - αξίωμα Α4 (ουδέτερων στοιχείων)

Fd = (0 * 1) + 1

Fd = (0) + 1 - αξίωμα Α4 (ουδέτερων στοιχείων)

Fd = 1 - αξίωμα Α4 (ουδέτερων στοιχείων)

6. Βασικά θεωρήματα της άλγεβρας Boole

Από τον ορισμό και τα αξιώματα της άλγεβρας Boole παράγονται θεωρήματα, τα σπουδαιότερα από τα οποία παρουσιάζονται πιο κάτω:

Θ1	**a * a = a**	**a + a = a**
Θ2	**a * 0 = 0**	**a + 1 = 1**
Θ3	**a + (a * b) = a**	**a * (a + b) = a**
Θ4	**(a + b) + c = a + (b + c)**	**(a * b) * c = a * (b * c)**
Θ5	**(a ') ' = a**	
Θ6	**(a + b)' = a' * b'**	**(a * b)' = a' + b'**
Θ7	**a + a' * b = a + b**	**a * (a' + b) = a * b**

Τα θεωρήματα Θ1 μέχρι Θ7, αν και δεν είναι τα μοναδικά, αποτελούν βασικά θεωρήματα της άλγεβρας Boole και η εξοικείωση με αυτά είναι πολύ σημαντική. Οι αποδείξεις των πιο πάνω θεωρημάτων βασίζονται στα αξιώματα Α0-Α5, αλλά δεν εμπίπτουν στους στόχους του μαθήματος και δεν παρατίθενται.

Παράδειγμα 2.2

(α) Να απλοποιήσετε τη λογική έκφραση F = (x'+xyz')+(x'+xyz')(x+x'y'z)

Σημείωση: Αντί x*y*z μπορούμε να γράψουμε xyz.

Λύση:

	= (**x'+xyz'**)+(**x'+xyz'**)(x+x'y'z)	
	= **x'**+**x**yz'	Θ3
	= x'+yz'	Θ7

(β) Να αποδείξετε ότι (abcd)' = a'+b'+c'+d'

Λύση:

a'+b'+c'+d'	= (**a'+b'**) + (**c'+d'**)	
	= **(ab)'**+**(cd)'**	Θ6
	= [**(ab)*(cd)**]'	Θ6
	= (abcd)'	

(γ) Να αποδείξετε ότι ab+a'c = ab+a'c+bc

Λύση:

ab+a'c	= (ab+**abc**) + (a'c+**a'bc**)	Θ3
	= ab+a'c+**a**bc+**a'**bc	A2
	= ab+a'c+**(a+a')** bc	A3
	= ab+a'c+**1 * bc**	A5
	= ab+a'c+bc	A4

(δ) Να βρείτε το συμπλήρωμα της: F = xy+wxyz'+x'y

Λύση:

F'	= [xy(**1+wz'**)+x'y]'	A3
	= (x**y**+x'**y**)'	Θ2
	= [y(**x+x'**)]'	A3
	= (**y * 1**)'	A5
	= y'	A4

5. Να χρησιμοποιήσετε πίνακα αλήθειας, για να αποδείξετε το θεώρημα Θ6, δηλαδή,

(a + b)' = a' * b'

Λύση:

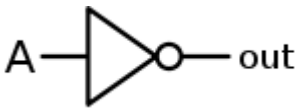
Θα δημιουργήσουμε έναν πίνακα αλήθειας, ο οποίος θα περιέχει όλους τους συνδυασμούς τιμών των μεταβλητών a και b και για κάθε συνδυασμό θα συγκρίνουμε το αποτέλεσμα που προκύπτει από κάθε μέλος της παράστασης:

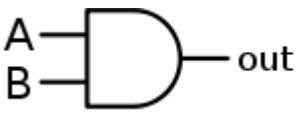
a	b	(a + b)'	a' * b'
0	0	1	1
0	1	0	0
1	0	0	0
1	1	0	0

Παρατηρούμε ότι για κάθε σειρά του πίνακα και τα δύο μέλη έχουν την ίδια τιμή, επομένως το θεώρημα Θ6 αληθεύει.

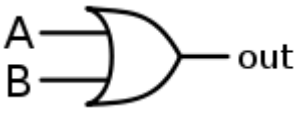
7. Λογικές Πύλες

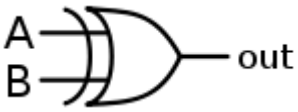
Οι λογικές πύλες είναι τα βασικά δομικά στοιχεία, τα οποία χρησιμοποιούνται για να κατασκευάσουμε λογικά κυκλώματα. Είναι έτοιμοι ψηφιακοί σχεδιασμοί, οι οποίοι υλοποιούν τις βασικές λογικές πράξεις. Στην έξοδο της πύλης παράγεται ένα σήμα, του οποίου η τιμή είναι το αποτέλεσμα της επεξεργασίας, σύμφωνα με τις παρούσες τιμές των σημάτων εισόδου και το είδος της πύλης.

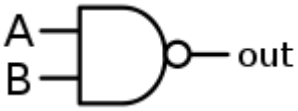
<table>
<tr>
<td>Πύλη NOT</td>
<td> 
</td>
<td>Η λειτουργία της είναι η αντιστροφή του λογικού σήματος της εισόδου.

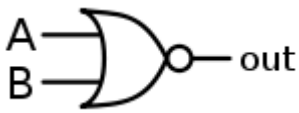
Y = A'</td>
<td>
<table>
<tr><th>Είσοδος</th><th>Έξοδος</th></tr>
<tr><td>A</td><td>OXI A</td></tr>
<tr><td>0</td><td>1</td></tr>
<tr><td>1</td><td>0</td></tr>
</table>
</td>
</tr>
<tr>
<td>Πύλη AND</td>
<td> 
</td>
<td>Η πύλη AND εκτελεί τη λογική πράξη AND (ΚΑΙ) μεταξύ των εισόδων της.

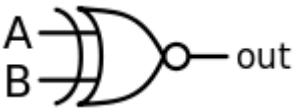
Y = A * B</td>
<td>
<table>
<tr><th colspan="2">Είσοδοι</th><th>Έξοδος</th></tr>
<tr><td>A</td><td>B</td><td>A ΚΑΙ B</td></tr>
<tr><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>1</td></tr>
</table>
</td>
</tr>
</table>

<table>
<tr>
<td>Πύλη OR</td>
<td> 
</td>
<td>Η πύλη OR εκτελεί τη λογική πράξη OR (Η΄) μεταξύ των εισόδων της.

Y = A + B</td>
<td>
<table>
<tr><th colspan="2">Είσοδοι</th><th>Έξοδος</th></tr>
<tr><td>A</td><td>B</td><td>A H' B</td></tr>
<tr><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>1</td></tr>
</table>
</td>
</tr>
<tr>
<td>Πύλη XOR</td>
<td> 
</td>
<td>Η πύλη XOR εκτελεί τη λογική πράξη XOR (ΑΠΟΚΛΕΙΣΤΙΚΟ Η') μεταξύ των εισόδων της.

Y = A ⊕ B
= A * B' + A' * B</td>
<td>
<table>
<tr><th colspan="2">Είσοδοι</th><th>Έξοδος</th></tr>
<tr><td>A</td><td>B</td><td>A XOR B</td></tr>
<tr><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>0</td></tr>
</table>
</td>
</tr>
<tr>
<td>Πύλη NAND</td>
<td> 
</td>
<td>Η πύλη NAND (ΟΧΙ-ΚΑΙ) δίνει την αντίθετη έξοδο από την AND.

Y = (A * B)'</td>
<td>
<table>
<tr><th colspan="2">Είσοδοι</th><th>Έξοδος</th></tr>
<tr><td>A</td><td>B</td><td>A NAND B</td></tr>
<tr><td>0</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>0</td></tr>
</table>
</td>
</tr>
<tr>
<td>Πύλη NOR</td>
<td> 
</td>
<td>Η πύλη NOR (ΟΧΙ-Η') δίνει την αντίθετη έξοδο από την OR.

Y = (A + B)'</td>
<td>
<table>
<tr><th colspan="2">Είσοδοι</th><th>Έξοδος</th></tr>
<tr><td>A</td><td>B</td><td>A NOR B</td></tr>
<tr><td>0</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>0</td></tr>
</table>
</td>
</tr>
<tr>
<td>Πύλη XNOR</td>
<td> 
</td>
<td>Η πύλη XNOR δίνει την αντίθετη έξοδο από την XOR, δηλαδή δίνει λογικό 1 όταν οι δύο είσοδοι είναι στην ίδια λογική στάθμη.

Y = A * B + A' * B'
= (A ⊕ B)'
= A ¤ B</td>
<td>
<table>
<tr><th colspan="2">Είσοδοι</th><th>Έξοδος</th></tr>
<tr><td>A</td><td>B</td><td>A XNOR B</td></tr>
<tr><td>0</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>1</td></tr>
</table>
</td>
</tr>
</table>

Ας δούμε κάποια παραδείγματα μετατροπής λεκτικών προτάσεων σε λογικές εκφράσεις, χρησιμοποιώντας τις λογικές πύλες.

2.3 Παραδείγματα μετατροπής λεκτικών προτάσεων

Για τα παρακάτω παραδείγματα θεωρούμε ότι έχουμε τρεις λογικές μεταβλητές (boolean) X, Y και Z και η λεκτική πρόταση είναι **αληθής**. Δηλαδή, η λεκτική πρόταση με οποιονδήποτε πιθανό συνδυασμό θα έχει πάντα αποτέλεσμα **true (1).**

1. Και οι 3 μεταβλητές είναι αληθείς: **X AND Y AND Z (**1 AND 1 AND 1 = 1**)**
2. Τουλάχιστον μία μεταβλητή είναι αληθής: **X OR Y OR Z (**1 OR 0 OR 0 = 1**)**
3. Δεν είναι καμία μεταβλητή αληθής: **X NAND Y NAND Z** (0 NAND 0 NAND 0 = 1)
4. Οι μεταβλητές X και Y είναι αληθείς ή η μεταβλητή Z είναι αληθής: **X AND Y OR Z** (1 AND 1 OR 1 = 1)
5. Και οι 3 μεταβλητές είναι ψευδείς: **X NOR Y NOR Z** (0 NOR 0 NOR 0 = 1)
6. Και οι 3 μεταβλητές είναι είτε ψευδείς είτε αληθείς: **X XNOR Y XNOR Z**
7. Δεν είναι και οι 3 μεταβλητές ψευδείς, ούτε είναι και οι 3 μεταβλητές αληθείς: **X XOR Y XOR Z**

8. Λογικές Συναρτήσεις και Λογικά Κυκλώματα

Μία λογική συνάρτηση (boolean function) αποτελείται από μία δυαδική μεταβλητή, το σύμβολο '=' και μία λογική έκφραση που αποτελείται από λογικές μεταβλητές, τις σταθερές 0, 1, παρενθέσεις και λογικούς τελεστές. Η λογική έκφραση ορίζει την σχέση μεταξύ των δυαδικών μεταβλητών και η συνάρτηση, για ορισμένες τιμές των μεταβλητών, παίρνει τιμή 1 (`true`), ή 0 (`false`).

Κάθε λογική συνάρτηση μπορεί να μετατραπεί σε λογικό κύκλωμα. Η διαδικασία σχεδίασης ξεκινά με τη λεκτική περιγραφή της λειτουργίας του προς σχεδίαση συνδυαστικού κυκλώματος και καταλήγει σε ένα λογικό κύκλωμα. Για τη σχεδίαση ενός συνδυαστικού λογικού κυκλώματος ακολουθούμε τα εξής βήματα:

(α) Από την περιγραφή της λειτουργίας του κυκλώματος που θέλουμε να σχεδιάσουμε, προσδιορίζουμε το πλήθος των εισόδων και το πλήθος των εξόδων και δίνουμε κατάλληλα ονόματα σε αυτές.

(β) Συμπληρώνουμε τον πίνακα αλήθειας.

(γ) Σχεδιάζουμε το λογικό κύκλωμα.

Πιο κάτω παρουσιάζονται αναλυτικά τα βήματα της διαδικασίας μετατροπής από λογική συνάρτηση σε λογικό κύκλωμα:

Έστω ότι μας δίνεται η πιο κάτω λογική συνάρτηση:

F = A + B' * C

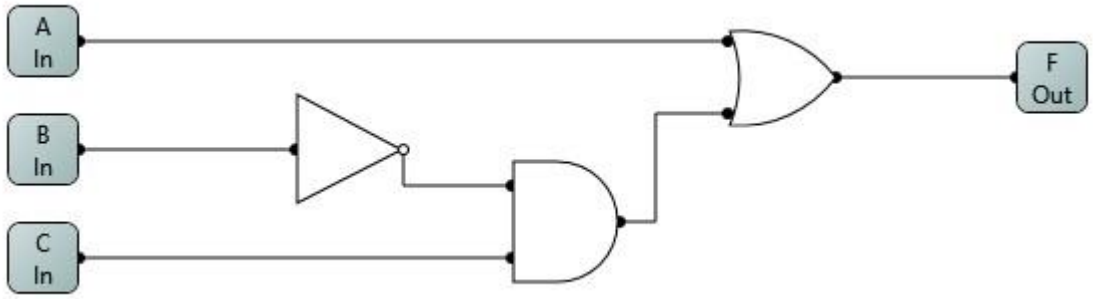
- Η τιμή της F θα είναι 1, όταν ο όρος A = 1 ή ο όρος B' * C =1.
- Το B' * C = 1 όταν το B' = 1 (B = 0) και το C = 1.

Ο πίνακας αλήθειας μίας συνάρτησης έχει 2^N περιπτώσεις εισόδου, όπου N το πλήθος των εισόδων. Στο παράδειγμα έχουμε N=3 (A, B, C), άρα θα έχουμε 2^3=8 περιπτώσεις.

Δημιουργούμε τον **πίνακα αλήθειας** της συνάρτησης.

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Σχεδιάζουμε το **λογικό κύκλωμα** με το εργαλείο Logic Circuit.



Κύκλωμα 1: Κύκλωμα συνάρτησης F = A + B' * C

Παράδειγμα 2.4

Έστω ότι έχουμε 3 σήματα εισόδου (A, B, C). Να γράψετε τη συνάρτηση, να δημιουργήσετε τον πίνακα αλήθειας της και να σχεδιάσετε το κύκλωμα, έτσι ώστε η παρακάτω λογική πρόταση να είναι αληθής.

«Οι είσοδοι A και B δεν έχουν την ίδια τιμή ή η είσοδος C είναι αληθής».

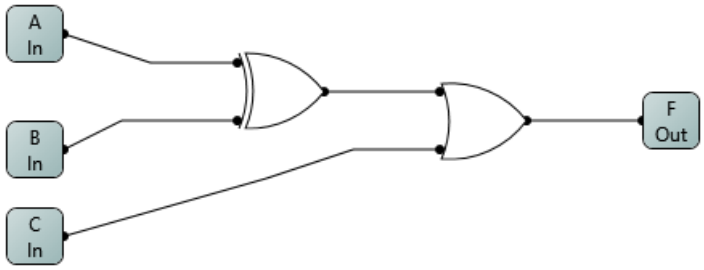
Η πύλη που δέχεται αντίθετες τιμές στις δύο εισόδους για να επαληθευτεί, είναι η πύλη XOR. Για να συνδυάσουμε και την είσοδο C θα χρησιμοποιήσουμε μία πύλη OR. Η συνάρτηση της πιο πάνω λογικής πρότασης θα έχει ως εξής:

Συνάρτηση: $F = A \oplus B + C$

Πίνακας Αλήθειας

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

Λογικό Κύκλωμα

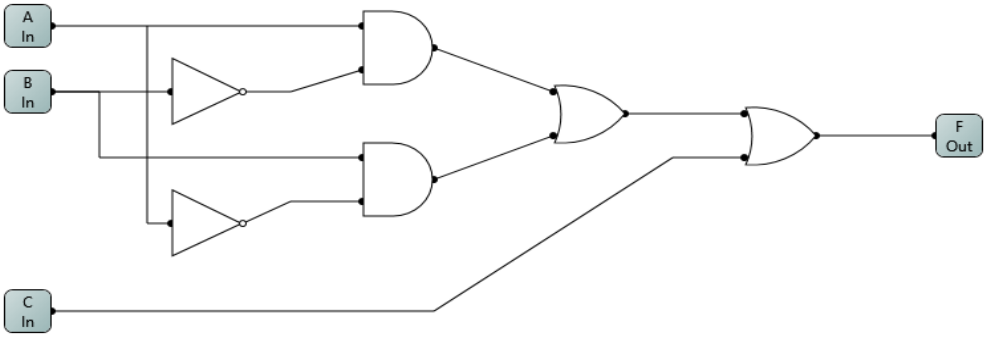


Κύκλωμα 2: Κύκλωμα συνάρτησης F = A ⊕ B + C

Η συνάρτηση του πιο πάνω παραδείγματος θα μπορούσε να διατυπωθεί και με άλλο τρόπο, αντικαθιστώντας την πύλη XOR με πύλες AND, OR και NOT.

F = A * B' + A' * B + C

Αυτό θα έχει ως αποτέλεσμα να αλλάξει και ο σχεδιασμός του κυκλώματος.



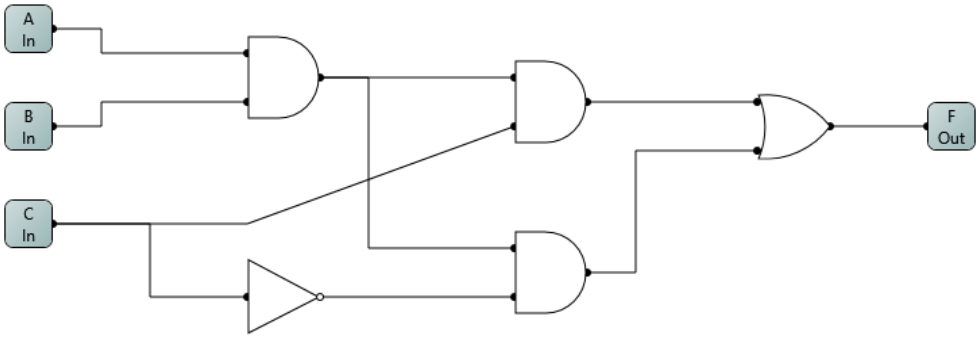
Κύκλωμα 3: Κύκλωμα παραδείγματος 2.4 με πύλες AND, OR και NOT

Παρόλο, όμως, που έχουν αλλάξει τόσο το κύκλωμα όσο και η συνάρτηση, ο πίνακας αλήθειας παραμένει ο ίδιος. Μία συνάρτηση που πρόκειται να υλοποιηθεί ως κύκλωμα, κρίνεται αναγκαίο να απλοποιηθεί, ώστε να προκύψει ένα κύκλωμα με μικρότερο αριθμό πυλών και συνδέσεων.

Η απλοποίηση γίνεται είτε αλγεβρικά είτε χρησιμοποιώντας τους χάρτες Karnaugh. Η αλγεβρική απλοποίηση μίας συνάρτησης απαιτεί ιδιαίτερη εξοικείωση με την άλγεβρα Boole και είναι δύσκολη για σύνθετες εκφράσεις.

Παράδειγμα 2.5

Δίνεται η λογική συνάρτηση F = A*B*C + A*B*C'. Το κύκλωμα της συνάρτησης εμφανίζεται πιο κάτω. Να απλοποιήσετε τη συνάρτηση.

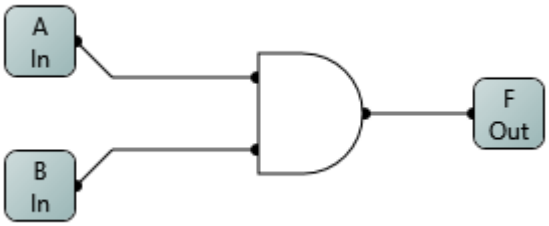


Κύκλωμα 4: Κύκλωμα συνάρτησης F = A*B*C + A*B*C' πριν από την απλοποίηση

Χρησιμοποιώντας τα αξιώματα και τα θεωρήματα της άλγεβρας Boole, μπορούμε να απλοποιήσουμε τη συνάρτηση F, ως ακολούθως.

A * B * C + A * B * C'	= A * B * (C + C')	- A3
	= A * B * 1	- A5
	= A * B	- A4

Το απλοποιημένο κύκλωμα της συνάρτησης φαίνεται πιο κάτω.



Κύκλωμα 5: Κύκλωμα συνάρτησης μετά την απλοποίηση

9. Ελαχιστόροι και Μεγιστόροι

Με τον όρο ελαχιστόρος (minterm) εννοούμε κάθε πιθανό συνδυασμό που μπορεί να σχηματιστεί, χρησιμοποιώντας τις εισόδους της συνάρτησης και το λογικό ΚΑΙ. Για παράδειγμα, αν έχουμε τις εισόδους X και Y, οι ελαχιστόροι που μπορούν να σχηματιστούν είναι οι XY, X'Y, XY' και X'Y'. Αντιστοίχως, με τον όρο μεγιστόρος (maxterm) εννοούμε κάθε πιθανό συνδυασμό που μπορεί να σχηματιστεί από τις εισόδους και το λογικό Ή. Για παράδειγμα, αν έχουμε τις εισόδους X και Y, θα έχουμε τους μεγιστόρους X+Y, X'+Y, X+Y' και X'+Y'. Οι ελαχιστόροι συμβολίζονται με το γράμμα m και οι μεγιστόροι με το γράμμα M. Στον παρακάτω πίνακα μπορείτε να δείτε τους ελαχιστόρους και τους μεγιστόρους για τρεις μεταβλητές (x, y, z).

			Ελαχιστόροι		Μεγιστόροι	
x	y	z	Όρος	Ονομασία	Όρος	Ονομασία
0	0	0	x′y′z′	m_0	x+y+z	M_0
0	0	1	x′y′z	m_1	x+y+z′	M_1
0	1	0	x′yz′	m_2	x+y′+z	M_2
0	1	1	x′yz	m_3	x+y′+z′	M_3
1	0	0	xy′z′	m_4	x′+y+z	M_4
1	0	1	xy′z	m_5	x′+y+z′	M_5
1	1	0	xyz′	m_6	x′+y′+z	M_6
1	1	1	xyz	m_7	x′+y′+z′	M_7

Παρατηρούμε ότι ο κάθε ελαχιστόρος αποτελεί το συμπλήρωμα του αντίστοιχου μεγιστόρου και αντιστρόφως.

Μία συνάρτηση μπορεί να εκφραστεί ως το άθροισμα των ελαχιστόρων. Αν έχουμε, για παράδειγμα, τη συνάρτηση f=x'y'z+xy'z'+xyz, μπορεί να γραφτεί ως $f=m_1+m_4+m_7$. Επίσης, χρησιμοποιείται ο συμβολισμός f=Σ(1, 4, 7). Η συνάρτηση μπορεί να γραφτεί και ως το γινόμενο των μεγιστόρων που δεν χρησιμοποιούνται στο άθροισμα των ελαχιστόρων. Οπότε θα έχουμε $f=M_0M_2M_3M_5M_6$. Χρησιμοποιείται ο συμβολισμός f=Π(0, 2, 3, 5, 6).

Παράδειγμα 2.6

Να γράψετε τη συνάρτηση f = w'x'y'z' + wxy'z + wxyz + wxy'z' ως άθροισμα ελαχιστόρων και γινόμενο μεγιστόρων.

Με βάση τον πίνακα τριών μεταβλητών, μπορούμε να δημιουργήσουμε τον πίνακα για τέσσερις μεταβλητές. Για τέσσερις μεταβλητές (w, x, y, z), θα έχουμε 16 ελαχιστόρους και 16 μεγιστόρους.

				Ελαχιστόροι		Μεγιστόροι	
w	x	y	z	Όρος	Ονομασία	Όρος	Ονομασία
0	0	0	0	w′x′y′z′	m_0	w+x+y+z	M_0
0	0	0	1	w′x′y′z	m_1	w+x+y+z′	M_1
0	0	1	0	w′x′yz′	m_2	w+x+y′+z	M_2
0	0	1	1	w′x′yz	m_3	w+x+y′+z′	M_3
0	1	0	0	w′xy′z′	m_4	w+x′+y+z	M_4
0	1	0	1	w′xy′z	m_5	w+x′+y+z′	M_5
0	1	1	0	w′xyz′	m_6	w+x′+y′+z	M_6
0	1	1	1	w′xyz	m_7	w+x′+y′+z′	M_7
1	0	0	0	wx′y′z′	m_8	w′+x+y+z	M_8
1	0	0	1	wx′y′z	m_9	w′+x+y+z′	M_9
1	0	1	0	wx′yz′	m_{10}	w′+x+y′+z	M_{10}
1	0	1	1	wx′yz	m_{11}	w′+x+y′+z′	M_{11}
1	1	0	0	wxy′z′	m_{12}	w′+x′+y+z	M_{12}
1	1	0	1	wxy′z	m_{13}	w′+x′+y+z′	M_{13}
1	1	1	0	wxyz′	m_{14}	w′+x′+y′+z	M_{14}
1	1	1	1	wxyz	m_{15}	w′+x′+y′+z′	M_{15}

Η συνάρτηση θα εκφραστεί ως το άθροισμα των ελαχιστόρων f=Σ(0,13,15,12) και ως το γινόμενο των μεγιστόρων f=Π(1,2,3,4,5,6,7,8,9,10,11,14).

Παράδειγμα 2.7

Να γράψετε τη συνάρτηση f = a + b'c ως άθροισμα ελαχιστόρων και γινόμενο μεγιστόρων.

Η συνάρτηση έχει 3 μεταβλητές (a, b, c). Από τον πρώτο όρο a λείπουν οι όροι b και c.

Καταρχάς προσθέτουμε τον όρο b. Άρα θα έχουμε:

a(b+b') = ab+ab' - A5

Στη συνέχεια προσθέτουμε τον όρο c:

ab(c+c')+ab'(c+c') = abc + abc' + ab'c + ab'c' - A5

Από τον όρο b'c λείπει ο όρος a. Άρα θα έχουμε:

b'c(a+a') = ab'c + a'b'c - A5

Έχοντας προσθέσει όλους τους όρους, η συνάρτηση τώρα αποτελείται από τους όρους:

f = abc + abc' + ab'c + ab'c' + ab'c + a'b'c

Παρατηρούμε ότι ο όρος ab'c εμφανίζεται δύο φορές. Σύμφωνα με το θεώρημα 1 (a+a=a), μπορούμε να απαλείψουμε τον έναν από αυτούς.

Οι όροι της συνάρτησης έχουν διαμορφωθεί ως:

f = abc + abc' + ab'c + ab'c' + a'b'c

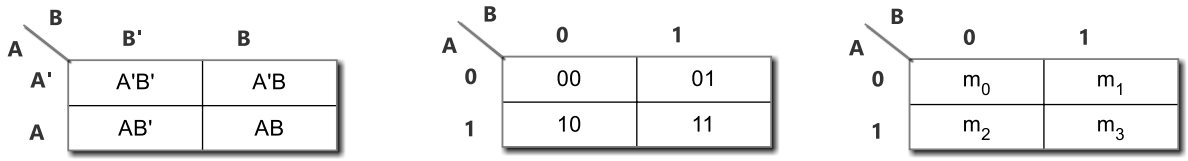
Το άθροισμα των ελαχιστόρων θα είναι f = Σ (1,4,5,6,7) και το γινόμενο των μεγιστόρων θα είναι f = Π (0,2,3).

10. Χάρτες Karnaugh

Σκοπός ενός χάρτη Karnaugh είναι η απλοποίηση μίας λογικής συνάρτησης. Όπως είδαμε πιο πάνω, η απλοποίηση με τη χρήση της άλγεβρας Boole είναι αρκετά δύσκολη, ιδιαίτερα για πολύπλοκες συναρτήσεις. Ο χάρτης Karnaugh αναπαριστά έναν κύβο n διαστάσεων, διατηρώντας τη σχέση «γειτονικότητας» μεταξύ των κορυφών του. Σε κάθε χάρτη Karnaugh υπάρχουν τόσα τετράγωνα όσοι οι δυνατοί συνδυασμοί τιμών των μεταβλητών του. Κάθε τετράγωνο αντιστοιχεί σε έναν ελαχιστόρο, δηλαδή αν υπάρχουν k μεταβλητές, σχηματίζονται 2^k τετράγωνα και αντιστοίχως ελαχιστόροι. Αν ο ελαχιστόρος επαληθεύει τη συνάρτηση, βάζουμε 1 στο αντίστοιχο τετράγωνο, αν όχι, βάζουμε 0. Ομάδες, δηλαδή τετράγωνα ή ορθογώνια 2 ή 4 ή 8 ή 16 επαληθευμένων ελαχιστόρων στον χάρτη, υποδεικνύουν αθροίσματα ελαχιστόρων στα οποία 1 ή 2 ή 3 ή 4 μεταβλητές, αντίστοιχα, παίρνουν όλες τις δυνατές τιμές.

10.1 Χάρτης Karnaugh δύο μεταβλητών

Ένας χάρτης Karnaugh δύο μεταβλητών θα έχει 4 ελαχιστόρους. Αν ονομάσουμε τις μεταβλητές A και B, τότε οι ελαχιστόροι που θα έχουμε θα είναι οι A'B', A'B, AB' και AB. Αν αντικαταστήσουμε τις μεταβλητές με το ψηφίο 1 και το συμπλήρωμα τους με το ψηφίο 0, τότε οι ελαχιστόροι γράφονται ως 00, 01, 10 και 11. Η θέση των ελαχιστόρων σε έναν χάρτη Karnaugh δύο μεταβλητών παρουσιάζεται στην εικόνα 1.



Εικόνα 1: Η θέση των ελαχιστόρων σε έναν χάρτη Karnaugh δύο μεταβλητών

Όπως αναφέραμε και πιο πάνω, όπου υπάρχουν οι ελαχιστόροι βάζουμε τον αριθμό 1, ενώ όπου δεν υπάρχουν βάζουμε τον αριθμό 0.

Για παράδειγμα, αν έχουμε τη συνάρτηση **F = AB + A'B'**, ο χάρτης Karnaugh της συνάρτησης θα είναι:

A \ B	0	1
0	1	0
1	0	1

Η συνάρτηση αποτελείται από τους ελαχιστόρους AB και A'B' (δηλαδή τους ελαχιστόρους m_0 και m_3), οπότε βάζουμε στις αντίστοιχες θέσεις τον αριθμό 1 και στις υπόλοιπες θέσεις τον αριθμό 0.

10.2 Απλοποίηση συνάρτησης με χάρτη Karnaugh

Τα βήματα που πρέπει να ακολουθήσουμε είναι:

(α) Να ομαδοποιήσουμε τους όρους με τιμή 1. Αυτό σημαίνει να φτιάξουμε ομάδες που περιλαμβάνουν γειτονικούς όρους με τιμή 1. Για να είναι δύο όροι με τιμή 1 γειτονικοί, πρέπει να έχουν μία κοινή πλευρά μεταξύ τους. Γειτονικοί όροι θεωρούνται και οι όροι που βρίσκονται στις γωνίες. Γενικά, για να είναι δύο ή περισσότεροι όροι γειτονικοί, θα πρέπει να έχουν τουλάχιστον μία μεταβλητή που δεν μεταβάλλεται

(β) Επιλέγουμε ομάδες 2 ή 4 ή 8 ή 16 ελαχιστόρων, με γειτονικούς όρους που έχουν τιμή 1. Στόχος μας είναι να επιλέξουμε λίγες και όσο το δυνατόν μεγαλύτερες ομάδες. Όσο πιο μεγάλη είναι η ομάδα, τόσο πιο πολλές μεταβλητές απλοποιούνται

(γ) Ένα κελί με τιμή 1 μπορεί να είναι ομαδοποιημένο σε περισσότερες από μία ομάδες. Αν για κάποιο κελί με τιμή 1 δεν υπάρχουν γειτονικά κελιά με τιμή 1, τότε αποτελεί από μόνο του μία ομάδα

(δ) Μία ομάδα πρέπει να έχει τουλάχιστον ένα κελί με τιμή 1 που δεν ανήκει σε άλλη ομάδα

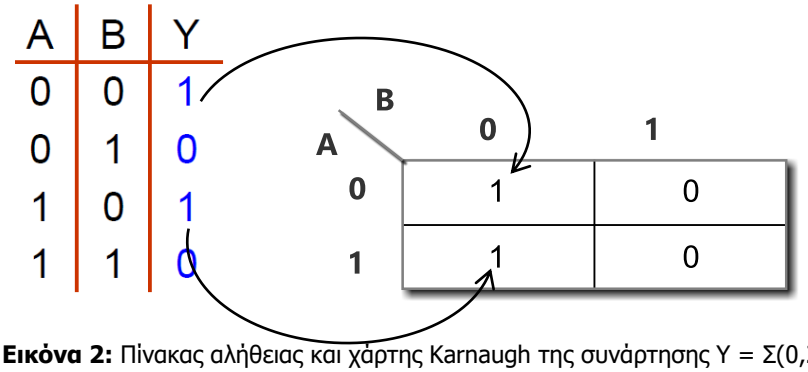
(ε) Σε κάθε ομάδα απλοποιούνται οι μεταβλητές που αλλάζουν τιμή, δηλαδή, οι οποίες σε ένα κελί εμφανίζονται σε κανονική μορφή και στο άλλο εμφανίζονται σε συμπληρωματική μορφή. Παραμένουν όσοι όροι διατηρούν την τιμή 1, σε όλα τα κελιά της ομάδας

(στ) Η τελική μορφή της συνάρτησης θα αποτελεί το άθροισμα των όρων που προκύπτουν από τις ομάδες.

Παράδειγμα 2.8

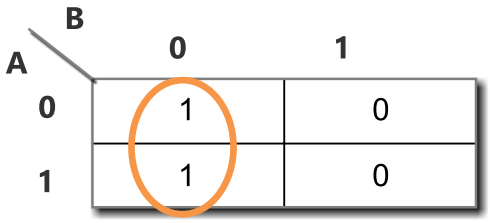
Να απλοποιηθεί η συνάρτηση Y = A'B' + AB'.

Ο πίνακας αλήθειας και ο χάρτης Karnaugh για τη συνάρτηση Y = Σ(0,2) παρουσιάζονται στην εικόνα 2.



Εικόνα 2: Πίνακας αλήθειας και χάρτης Karnaugh της συνάρτησης Y = Σ(0,2)

Ομαδοποιούμε τους γειτονικούς όρους πάνω στον χάρτη.



Βλέπουμε ότι η μόνη μεταβλητή που παραμένει σταθερή στην ομαδοποίηση είναι το B'. Επομένως, Y = B'.

Μπορούμε να επαληθεύσουμε αλγεβρικά τη συνάρτηση ως εξής:

Y = A'B' + AB'

= (A + A') * B' - A3

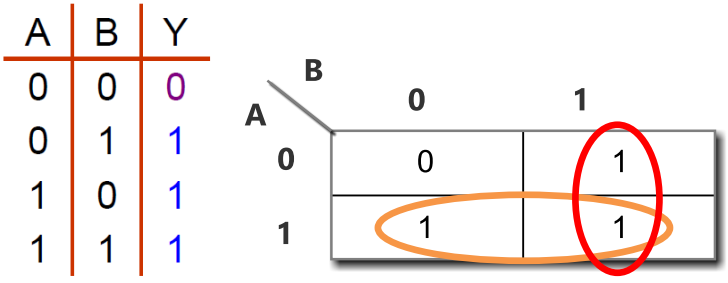
= 1 * B' - A5

= B' - A4

Παράδειγμα 2.9

Να απλοποιηθεί η συνάρτηση Y = A'B + AB' + AB.

Ο πίνακας αλήθειας και ο χάρτης Karnaugh για τη συνάρτηση Y = Σ(1,2,3) παρουσιάζονται στην εικόνα 3.



Εικόνα 3: Πίνακας αλήθειας και χάρτης Karnaugh της συνάρτησης Y = Σ(1,2,3)

Με βάση την ομαδοποίηση, η συνάρτηση απλοποιείται σε Σ(1,2,3) = A + B.

Μπορούμε να επαληθεύσουμε αλγεβρικά τη συνάρτηση ως εξής:

Y = A'B + AB' + AB

= A * (B' + B) + A'B - A3

= A + A'B - A5, A4

= A + B - Θ7

10.3 Χάρτης Karnaugh τριών μεταβλητών

Ένας χάρτης Karnaugh τριών μεταβλητών έχει 8 ελαχιστόρους. Η θέση των ελαχιστόρων σε έναν χάρτη Karnaugh τριών μεταβλητών, παρουσιάζεται στην εικόνα 4.

A \ BC	00	01	11	10
0	000	001	011	010
1	100	101	111	110

A \ BC	00	01	11	10
0	m_0	m_1	m_3	m_2
1	m_4	m_5	m_7	m_6

Εικόνα 4: Η θέση των ελαχιστόρων σε έναν χάρτη Karnaugh τριών μεταβλητών

Παράδειγμα 2.10

Να απλοποιηθεί η συνάρτηση F = A'B'C' + A'B'C.

Έχουμε τη συνάρτηση F = A'B'C' + A'B'C = Σ(0,1). Ο χάρτης Karnaugh και η ομαδοποίηση των όρων φαίνεται στην εικόνα 5.

A \ BC	00	01	11	10
0	1	1	0	0
1	0	0	0	0

Εικόνα 5: Ο χάρτης Karnaugh της συνάρτησης F = Σ(0,1)

Με βάση την ομαδοποίηση, βλέπουμε ότι οι μεταβλητές οι οποίες παραμένουν σταθερές είναι το A' και B'.

Άρα, Σ(0,1) = A' B'

Μπορούμε να επαληθεύσουμε αλγεβρικά τη συνάρτηση ως εξής:

F = A'B'C' + A'B'C

= A'B' * (C + C') - A3

= A'B' - A5, A4

Παράδειγμα 2.11

Να σχεδιαστεί λογικό κύκλωμα, το οποίο να υλοποιεί τη συνάρτηση πλειοψηφίας τριών εισόδων. Το κύκλωμα να αναγνωρίζει, δηλαδή, πότε έχουμε περισσότερα 1 από 0 στις τρεις εισόδους.

Από την περιγραφή της λειτουργίας του κυκλώματος φαίνεται ότι έχουμε τρεις εισόδους. Επιπλέον, απαιτείται μία έξοδος αφού το πλήθος των εισόδων είναι περιττός αριθμός (3) και, επομένως, είτε θα έχουμε περισσότερα 1 είτε περισσότερα 0 (αποκλείεται να έχουμε ίσο πλήθος από 1 και 0, όταν έχουμε τρεις εισόδους). Η έξοδος θα παίρνει τιμή 1 όταν έχουμε περισσότερα 1 από 0 στις τρεις εισόδους, αλλιώς θα παίρνει τιμή 0.

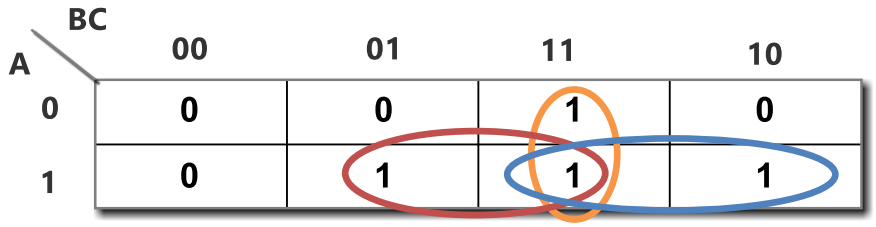
Δημιουργούμε τον πίνακα αλήθειας:

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Με βάση τον πίνακα αλήθειας, μπορούμε να δημιουργήσουμε τη συνάρτηση εξόδου:

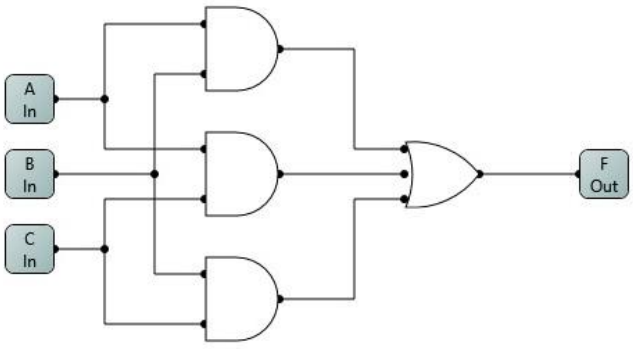
F = A'BC + AB'C + ABC' + ABC

Απλοποιούμε την πιο πάνω συνάρτηση με τη χρήση χάρτη Karnaugh τριών μεταβλητών (εικόνα 6).



Εικόνα 6: Χάρτης Karnaugh παραδείγματος 2.11

Με βάση τον χάρτη, μπορούμε να απλοποιήσουμε τη συνάρτηση σε F = **AB** + **AC** + **BC** (με αντίστοιχο χρώμα η ομάδα που έχει απλοποιηθεί). Τέλος, σχεδιάζουμε το λογικό κύκλωμα.



Κύκλωμα 6: Κύκλωμα υλοποίησης παραδείγματος 2.11

10.4 Χάρτης Karnaugh τεσσάρων μεταβλητών

Ένας χάρτης Karnaugh τεσσάρων μεταβλητών έχει 16 ελαχιστόρους. Η θέση των ελαχιστόρων σε έναν χάρτη Karnaugh τεσσάρων μεταβλητών, παρουσιάζεται στην εικόνα 7.

AB \ CD	00	01	11	10
00	0000	0001	0011	0010
01	0100	0101	0111	0110
11	1100	1101	1111	1110
10	1000	1001	1011	1010

AB \ CD	00	01	11	10
00	m_0	m_1	m_3	m_2
01	m_4	m_5	m_7	m_6
11	m_{12}	m_{13}	m_{15}	m_{14}
10	m_8	m_9	m_{11}	m_{10}

Εικόνα 7: Η θέση των ελαχιστόρων σε έναν χάρτη Karnaugh τεσσάρων μεταβλητών

Παράδειγμα 2.12

Να απλοποιήσετε τη συνάρτηση F = ABCD + A'B'C'D' + A'B'C'D + A'BC'D' + A'BC'D.

Ο χάρτης Karnaugh της συνάρτησης και οι ομαδοποιήσεις που μπορούν να γίνουν είναι οι πιο κάτω (εικόνα 8).

AB \ CD	00	01	11	10
00	1	1	0	0
01	1	1	0	0
11	0	0	1	0
10	0	0	0	0

Εικόνα 8: Χάρτης Karnaugh για το παράδειγμα 2.12

Παρατηρούμε ότι ο ελαχιστόρος m15 δεν μπορεί να ομαδοποιηθεί, γιατί δεν υπάρχουν γειτονικά κελιά με τιμή 1. Άρα, θα αποτελείται και από τις τέσσερις μεταβλητές, **ABCD**.

Η απλοποίηση που προκύπτει από την ομαδοποίηση των ελαχιστόρων m0, m1, m4, m5 (A'B'C'D'+A'B'C'D+A'BC'D'+A'BC'D) είναι **A'C'**, αφού αυτοί είναι οι δύο όροι που δεν μεταβάλλονται και διατηρούν την τιμή 1 σε όλα τα κελιά της ομάδας.

Η απλοποιημένη συνάρτηση είναι F = **A'C'** + **ABCD**.

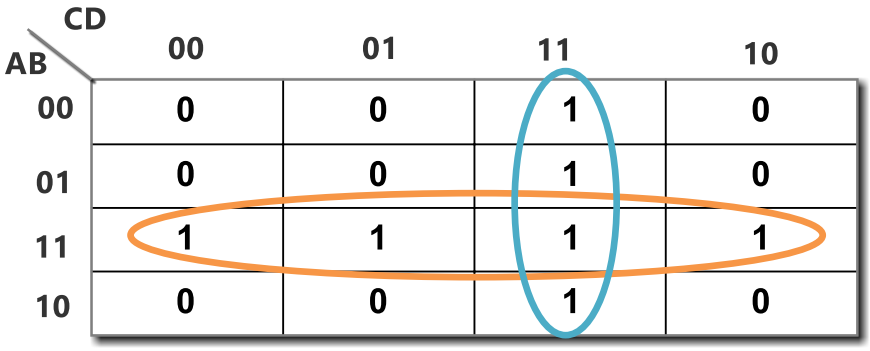
Σε χάρτη Karnaugh τεσσάρων μεταβλητών ισχύουν τα εξής:

- Μία δεκαεξάδα οδηγεί σε απλοποίηση με τιμή 1
- Μία οκτάδα οδηγεί σε όρο με μία μεταβλητή (απλοποιούνται τρεις μεταβλητές)
- Μία τετράδα οδηγεί σε όρο με δύο μεταβλητές (απλοποιούνται δύο μεταβλητές)
- Μία δυάδα οδηγεί σε όρο με τρεις μεταβλητές (απλοποιείται μία μεταβλητή)

Παράδειγμα 2.13

Να απλοποιήσετε τη συνάρτηση F(A,B,C,D) = Σ(3,7,11,12,13,14,15).

Ο χάρτης Karnaugh της συνάρτησης και οι ομαδοποιήσεις που μπορούν να γίνουν είναι οι πιο κάτω (εικόνα 9).



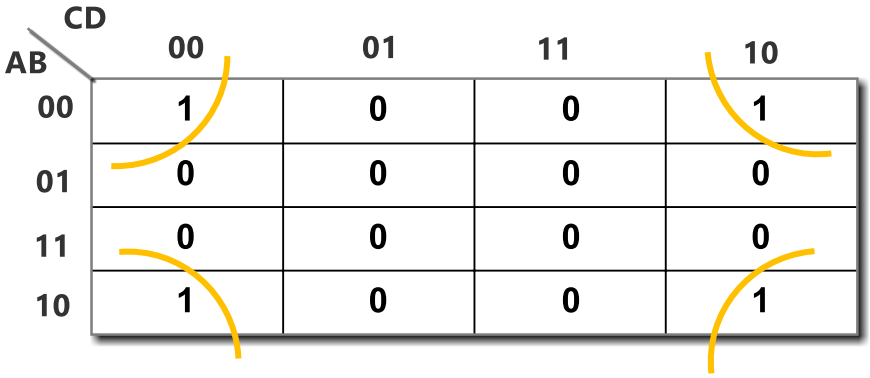
Εικόνα 9: Χάρτης Karnaugh για το παράδειγμα 2.13

Η απλοποιημένη συνάρτηση είναι F = **AB** + **CD**

Παράδειγμα 2.14

Να απλοποιήσετε τη συνάρτηση F(A,B,C,D) = Σ(0, 2, 8, 10).

Ο χάρτης Karnaugh της συνάρτησης και οι ομαδοποιήσεις που μπορούν να γίνουν είναι οι πιο κάτω (εικόνα 10). Παρατηρούμε ότι η ομαδοποίηση γίνεται για γωνιακούς όρους.



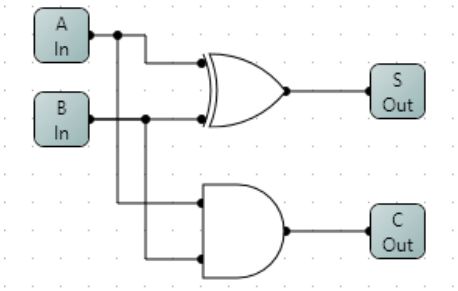
Εικόνα 10: Χάρτης Karnaugh για το παράδειγμα 2.14

Η απλοποιημένη συνάρτηση είναι **F = B' D'**.

11. Το κύκλωμα του Ημιαθροιστή (Half Adder)

Το κύκλωμα θέλουμε να δέχεται ως είσοδο δύο δυαδικά ψηφία (A και B) και να επιστρέφει στην έξοδο άλλα δύο δυαδικά ψηφία. Το άθροισμα της πρόσθεσης των A και B (συμβολίζεται με S) και το τυχόν κρατούμενο που προκύπτει από την πρόσθεση των A και B (συμβολίζεται με C). Οι τιμές που παίρνει το άθροισμα και το κρατούμενο για κάθε συνδυασμό τιμών των A και B εμφανίζονται στον πιο κάτω πίνακα:

Πράξη	A	B	S	C
0 + 0 = 0 (0 κρατούμενο)	0	0	0	0
0 + 1 = 1 (0 κρατούμενο)	0	1	1	0
1 + 0 = 1 (0 κρατούμενο)	1	0	1	0
1 + 1 = 0 (1 κρατούμενο)	1	1	0	1



Κύκλωμα 7: Ημιαθροιστής

Παρατηρούμε ότι ο πίνακας αλήθειας του αθροίσματος S αντιστοιχεί στη λειτουργία της πύλης XOR (αφού παράγει 1, όταν παίρνει διαφορετικές εισόδους A και B), ενώ ο πίνακας αλήθειας του κρατούμενου C αντιστοιχεί στη λειτουργία της πύλης AND (αφού παράγει 1 μόνο, όταν και οι δύο είσοδοι A και B είναι 1). Το κύκλωμα του Ημιαθροιστή παρουσιάζεται στο σχήμα αριστερά (κύκλωμα 7).

12. Το κύκλωμα του Πλήρους Αθροιστή (Full Adder)

Κατά την πρόσθεση δύο δυαδικών αριθμών προσθέτουμε τρία δυαδικά ψηφία. Τα δύο είναι τα ψηφία μίας τάξης και το τρίτο είναι το κρατούμενο από την πρόσθεση των ψηφίων της προηγούμενης, λιγότερο σημαντικής τάξης. Εξαιρείται βέβαια η πρόσθεση των λιγότερο σημαντικών ψηφίων, κατά την οποία δεν υπάρχει κρατούμενο.

Ο Πλήρης Αθροιστής πρέπει να παίρνει ως είσοδο τρία δυαδικά ψηφία (A, B και C για το κρατούμενο) και να παράγει στην έξοδο δύο δυαδικά ψηφία: το άθροισμα της πρόσθεσης A+B+C (S) και το τυχόν κρατούμενο που προκύπτει από την πρόσθεση αυτή (outC). Όπως και πριν, σχεδιάζουμε τον πίνακα αλήθειας του κυκλώματος με βάση τη συμπεριφορά που θέλουμε να έχει σε κάθε περίπτωση.

Αφού έχουμε 3 εισόδους, ο πίνακας αλήθειας του Πλήρους Αθροιστή θα έχει οκτώ (2^3) γραμμές.

A	B	C	S	outC
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1

Σχεδιάζουμε το κύκλωμα του Πλήρους Αθροιστή σε μορφή αθροίσματος γινομένων. Για να πετύχουμε το απλοποιημένο κύκλωμα, δημιουργούμε τους χάρτες Karnaugh για τις εξόδους S και outC (εικόνα 11).

Έξοδος S

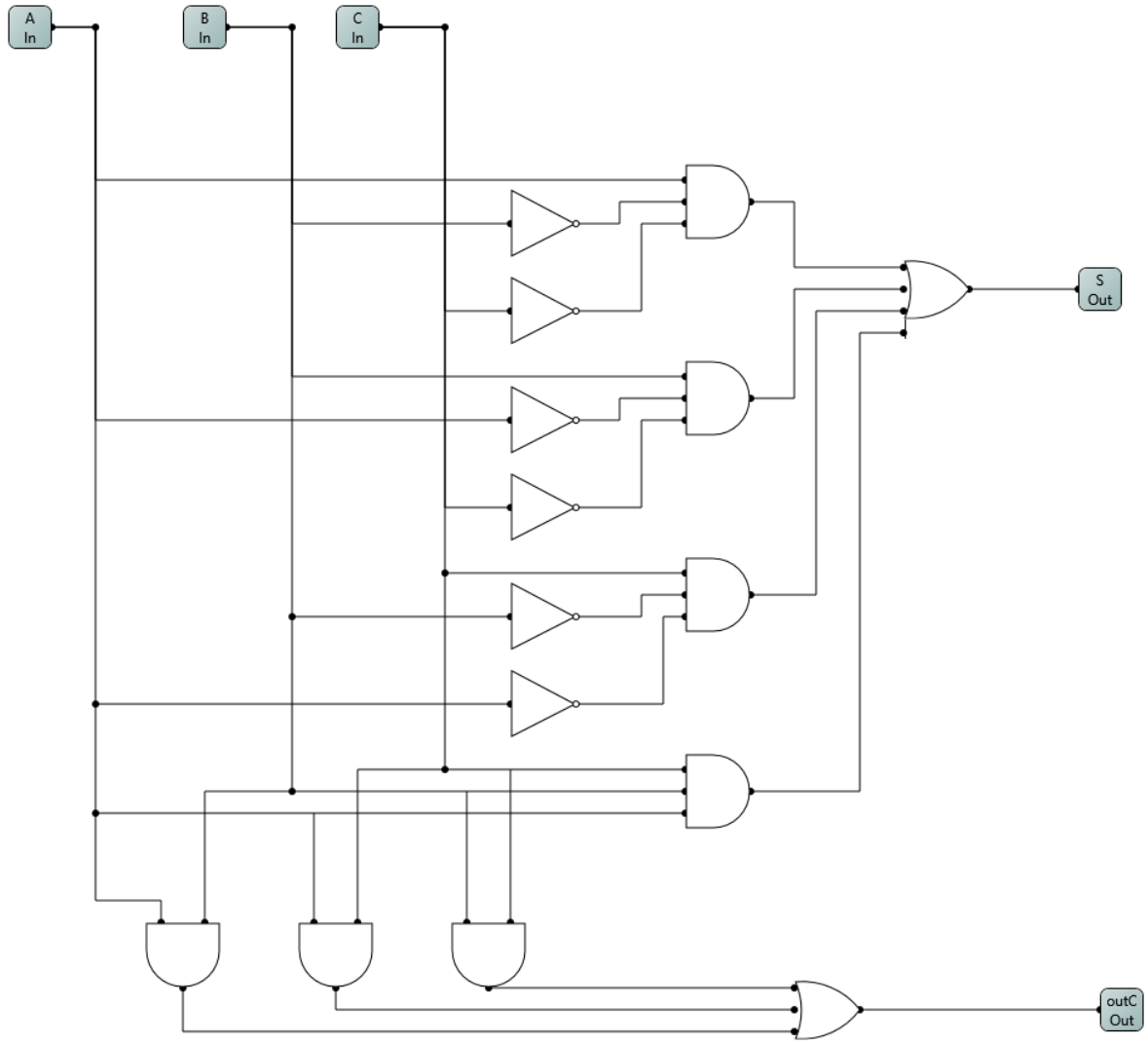
A \ BC	00	01	11	10
0	0	1	0	1
1	1	0	1	0

Έξοδος outC

A \ BC	00	01	11	10
0	0	0	1	0
1	0	1	1	1

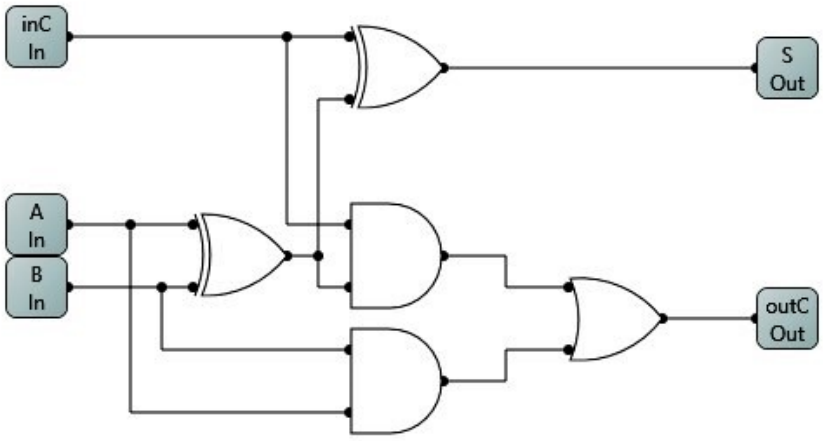
Εικόνα 11: Χάρτες Karnaugh για τις εξόδους S και outC ενός Πλήρους Αθροιστή

Οι ελαχιστοποιημένες συναρτήσεις που προκύπτουν είναι οι πιο κάτω:
S = AB'C' + A'B'C + ABC + A'BC' και outC = AC + BC + AB.
Το κύκλωμα του Πλήρους Αθροιστή σε μορφή αθροίσματος γινομένων φαίνεται πιο κάτω, στο κύκλωμα 8.



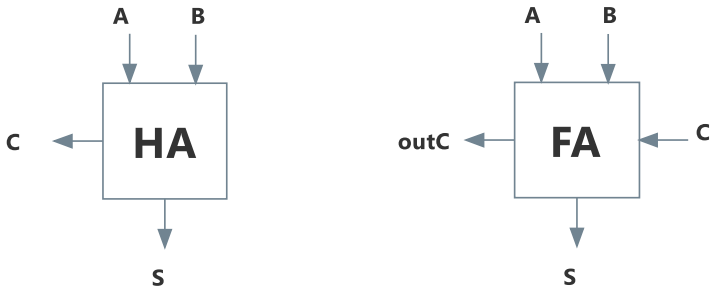
Κύκλωμα 8: Πλήρης Αθροιστής σε μορφή αθροίσματος γινομένων

Μία εναλλακτική υλοποίηση του κυκλώματος του Πλήρους Αθροιστή δίνεται πιο κάτω. Σε αυτό το κύκλωμα ο Πλήρης Αθροιστής δημιουργείται από δύο Ημιαθροιστές και μία πύλη OR.



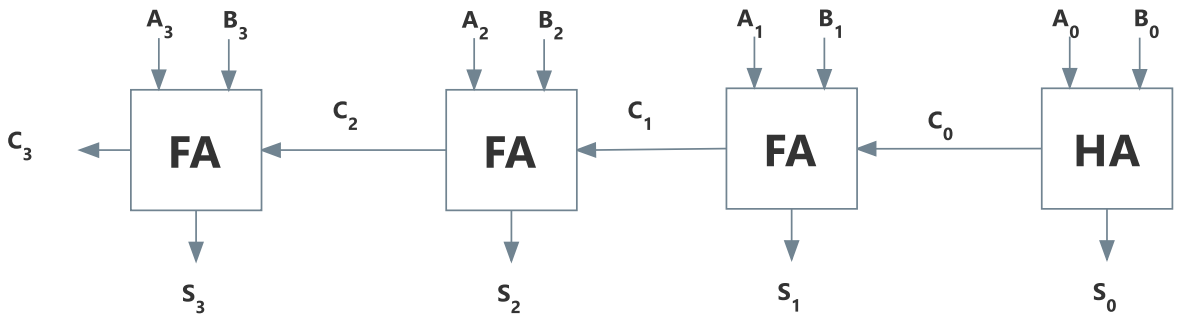
Κύκλωμα 9: Πλήρης Αθροιστής με δύο Ημιαθροιστές και μία πύλη OR

Τόσο ο Ημιαθροιστής όσο και ο Πλήρης Αθροιστής αποτελούν δομικά στοιχεία για πολυπλοκότερα κυκλώματα. Για να είναι ευκολότερη η χρήση τους, χρησιμοποιούμε τα μπλοκ διαγράμματα που φαίνονται στην εικόνα 12.



Εικόνα 12 Μπλοκ διαγράμματα για Ημιαθροιστή (HA) και Πλήρη Αθροιστή (FA)

Ένα παράδειγμα χρήσης των κυκλωμάτων του Ημιαθροιστή και του Πλήρους Αθροιστή είναι το κύκλωμα του αθροιστή δύο αριθμών των 4 bit. Ο Δυαδικός Αθροιστής μπορεί να κατασκευαστεί αν συνδέσουμε διαδοχικά έναν Ημιαθροιστή και τρεις Πλήρεις Αθροιστές. Όπως φαίνεται στο κύκλωμα 10, το κρατούμενο εξόδου του προηγούμενου αθροιστή συνδέεται με το κρατούμενο εισόδου του επόμενου.



Κύκλωμα 10: Δυαδικός Αθροιστής 4 bit

Έστω ότι έχουμε τους αριθμούς Α=1011 και Β=1010. Οι τιμές των κρατουμένων (C_i) και του αποτελέσματος (S_i) παρουσιάζονται στον πιο κάτω πίνακα.

i	A_i	B_i	C_i	S_i
0	1	0	0	1
1	1	1	1	0
2	0	0	0	1
3	1	1	1	0

Το αποτέλεσμα θα είναι ο αριθμός:

C_3	S_3	S_2	S_1	S_0
1	0	1	0	1

Βιβλιογραφία

1. Mano, M. M., Kime, C. R. (2014). *Logic and computer design fundamentals*. London: Pearson.
2. Wakerly, J. F. (2000). *Digital Design Principles and Practice*. London: Prentice Hall.
3. Λιναρδής, Π. (2001). *Ψηφιακή Σχεδίαση*. Πάτρα: Ελληνικό Ανοικτό Πανεπιστήμιο.
4. Ρουμελιώτης, Μ. (2012). *Ψηφιακή σχεδίαση*: Αρχές και εφαρμογές. Θεσσαλονίκη: Τζιόλα.

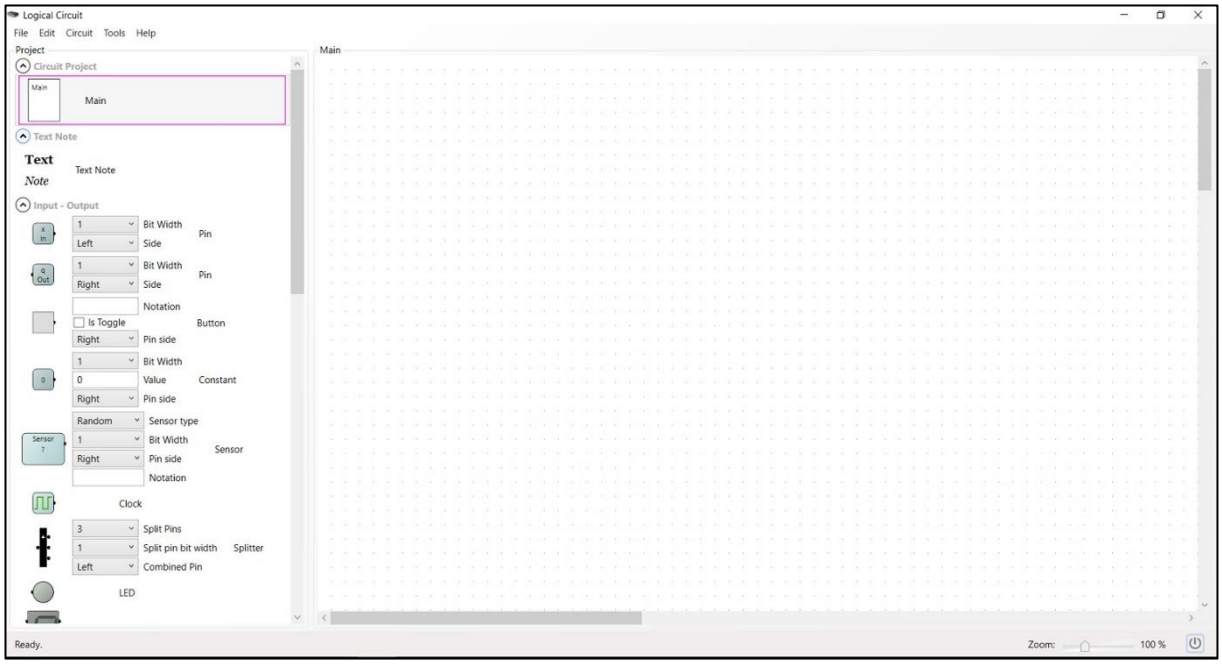
Γ2.2 Εγχειρίδιο χρήσεως Logic Circuit

Τι θα μάθουμε σήμερα:

- Να σχεδιάζουμε λογικά κυκλώματα με τη χρήση του εργαλείου Logic Circuit
- Να δημιουργούμε τον πίνακα αλήθειας για ένα λογικό κύκλωμα με τη χρήση του εργαλείου Logic Circuit.

1. Εισαγωγή

Το **Logic Circuit**© είναι ένα δωρεάν, ανοικτού κώδικα, εκπαιδευτικό λογισμικό, το οποίο επιτρέπει τον σχεδιασμό και την προσομοίωση ψηφιακών, λογικών κυκλωμάτων. Είναι εύκολο στη χρήση και το περιβάλλον διεπαφής χρήστη μοιάζει με τα πρότυπα του λογισμικού **ALGO**. Διαθέτει πληθώρα αντικειμένων τα οποία μπορούμε να χρησιμοποιήσουμε, για να δημιουργήσουμε και να αποθηκεύσουμε δικά μας λογικά κυκλώματα. Υπάρχει επιλογή εμφάνισης του περιβάλλοντος διεπαφής στην ελληνική γλώσσα. Μπορούμε να κατεβάσουμε το εργαλείο από την ιστοσελίδα http://www.logiccircuit.org/download.html.



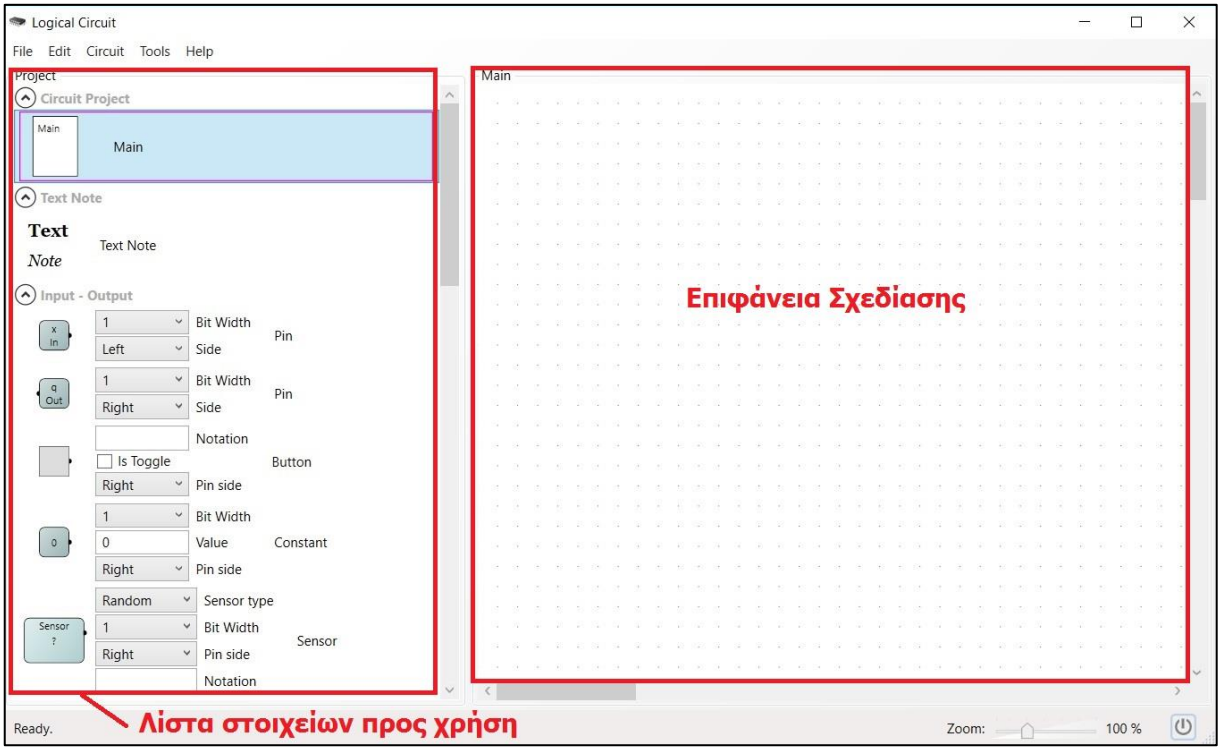
Εικόνα 1: Το περιβάλλον εργασίας του Logic Circuit

Εκτός από τη σχεδίαση του λογικού κυκλώματος, το εργαλείο επιτρέπει και τη δημιουργία του πίνακα αλήθειας, ώστε να μπορούμε επαληθεύσουμε τις τιμές.

2. Δημιουργία Λογικού Κυκλώματος

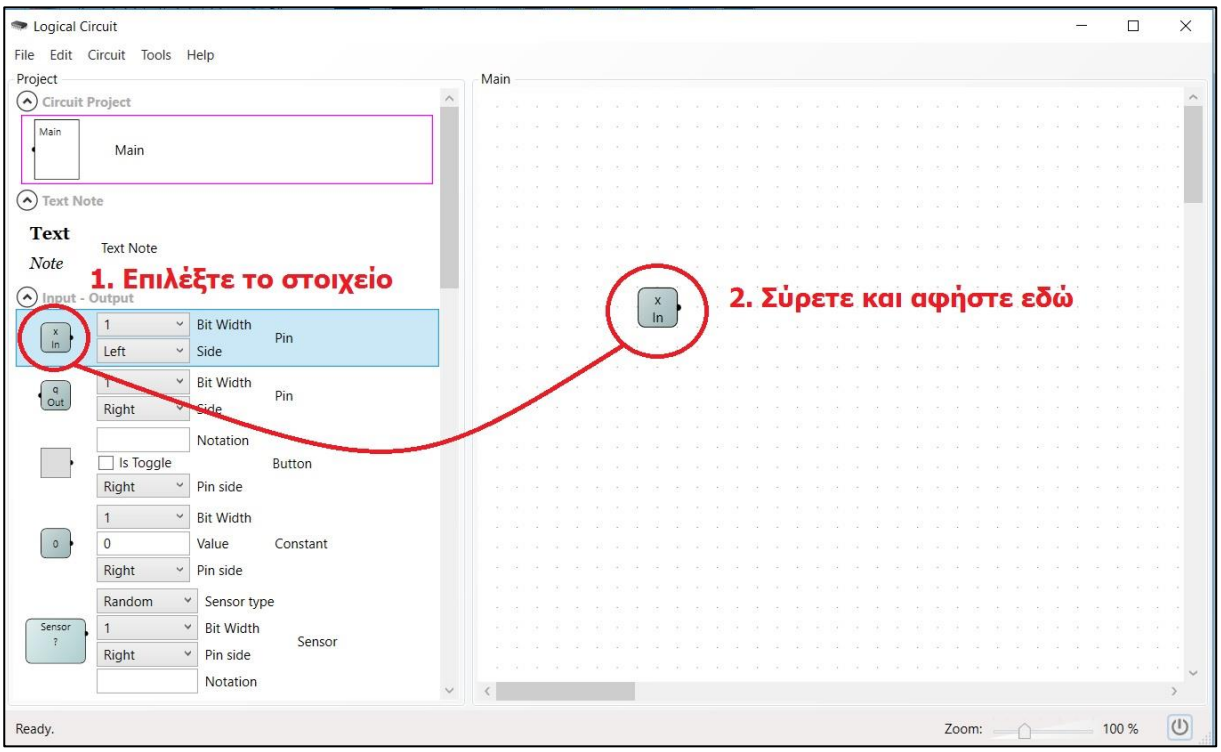
Ας υποθέσουμε ότι μας έχει δοθεί η λογική συνάρτηση **F = A + B' * C** και μας έχει ζητηθεί να την αναπαραστήσουμε σε λογικό κύκλωμα με τη χρήση του εργαλείου **Logic Circuit**© και να δημιουργήσουμε τον πίνακα αλήθειας. Από τη λογική συνάρτηση προσδιορίζουμε το πλήθος των εισόδων (**A, B, C**) και των εξόδων (**F**) και δίνουμε κατάλληλα ονόματα σε αυτές. Επίσης, αναγνωρίζουμε τις λογικές πύλες (**AND, NOT, OR**) οι οποίες θα χρησιμοποιηθούν με την κατάλληλη σειρά προτεραιότητας. Πιο κάτω εμφανίζονται αναλυτικά τα βήματα σχεδιασμού του λογικού κυκλώματος της πιο πάνω λογικής συνάρτησης:

3. Ενεργοποίηση του εργαλείου Logic Circuit©



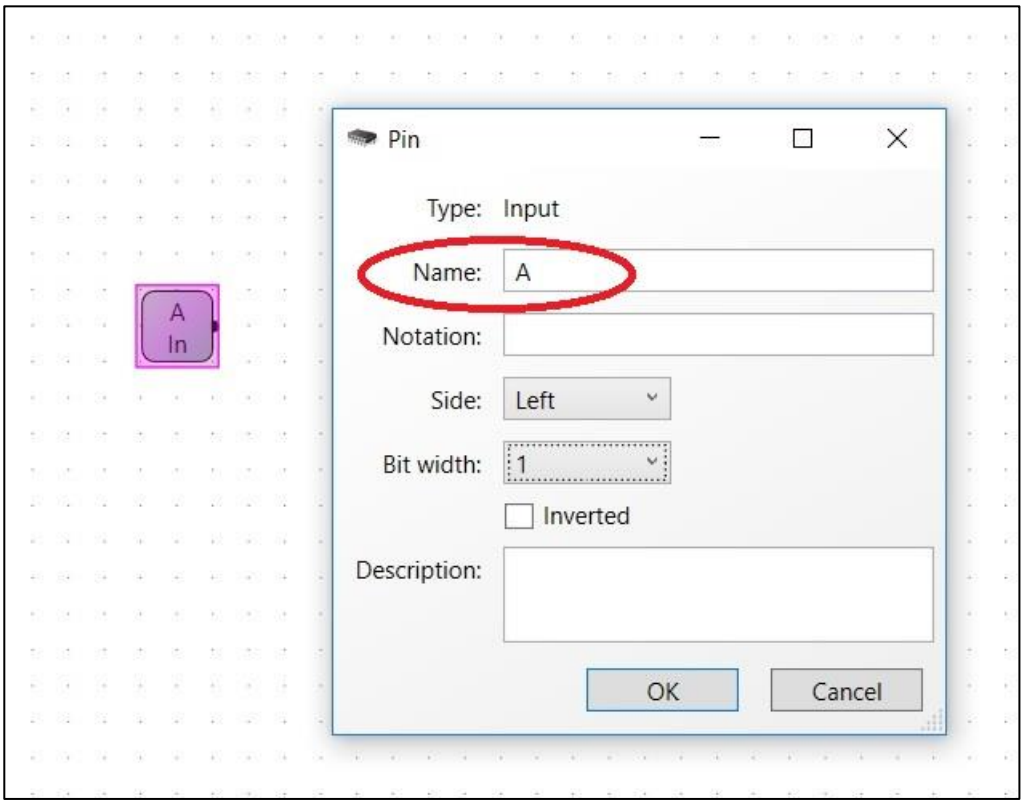
Εικόνα 2: Λίστα στοιχείων και επιφάνεια σχεδίασης

- Επιλέγουμε και τοποθετούμε το πρώτο στοιχείο. Μπορούμε να μετακινήσουμε το στοιχείο, αφού το επιλέξουμε, οπουδήποτε πάνω στην επιφάνεια σχεδίασης.



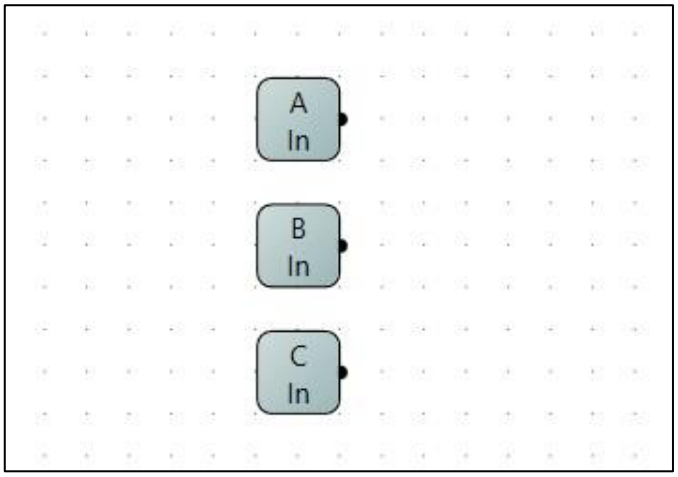
Εικόνα 3: Εισαγωγή αντικειμένου

- Πατούμε διπλό κλικ στο στοιχείο που έχουμε εισαγάγει, ώστε να μπορέσουμε να το μορφοποιήσουμε. Δίνουμε την τιμή **A** στο στοιχείο εισόδου.



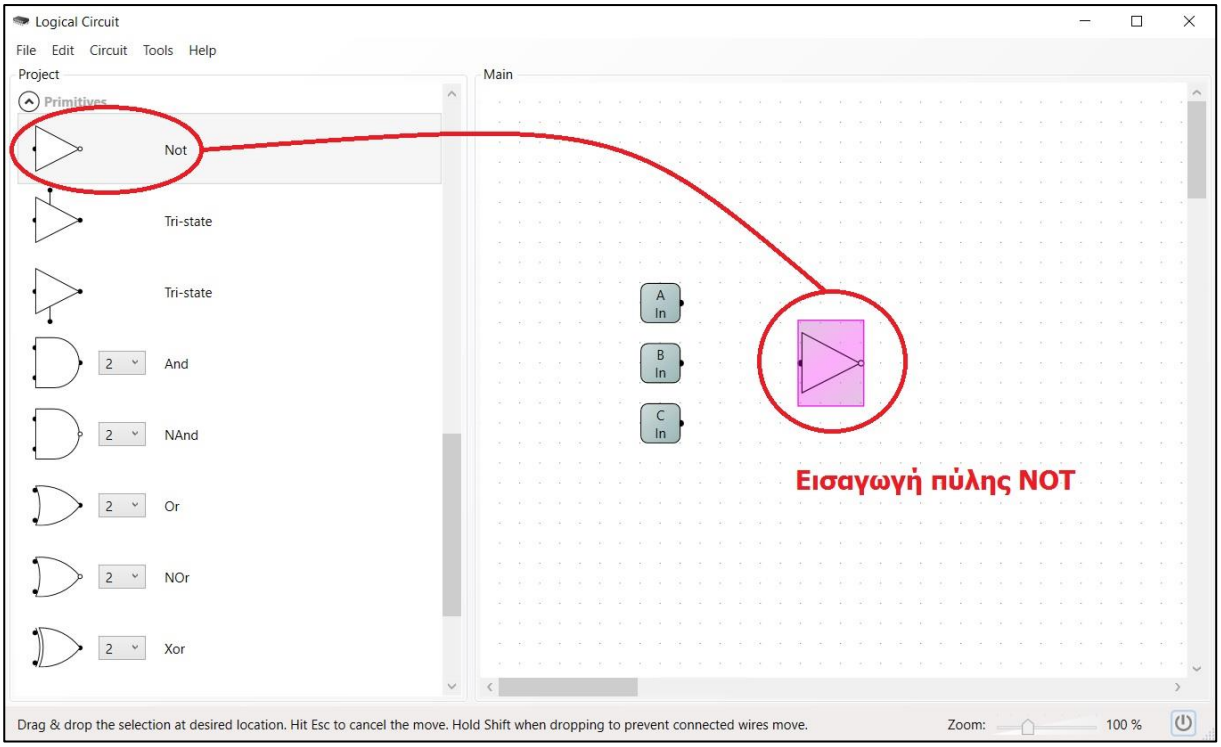
Εικόνα 4: Αλλαγή παραμέτρων στοιχείου εισόδου

- Εισάγουμε τα επόμενα δύο στοιχεία εισόδου. Μορφοποιούμε τις παραμέτρους των στοιχείων.



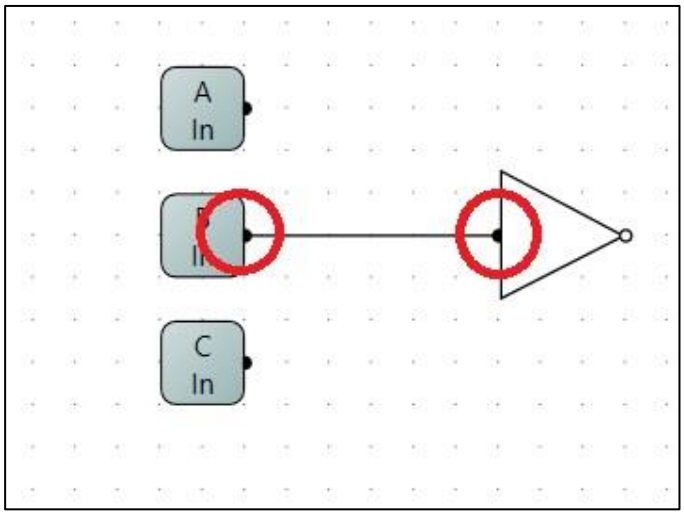
Εικόνα 5: Εισαγωγή στοιχείων εισόδου

- Προσθέτουμε μία πύλη **NOT**.



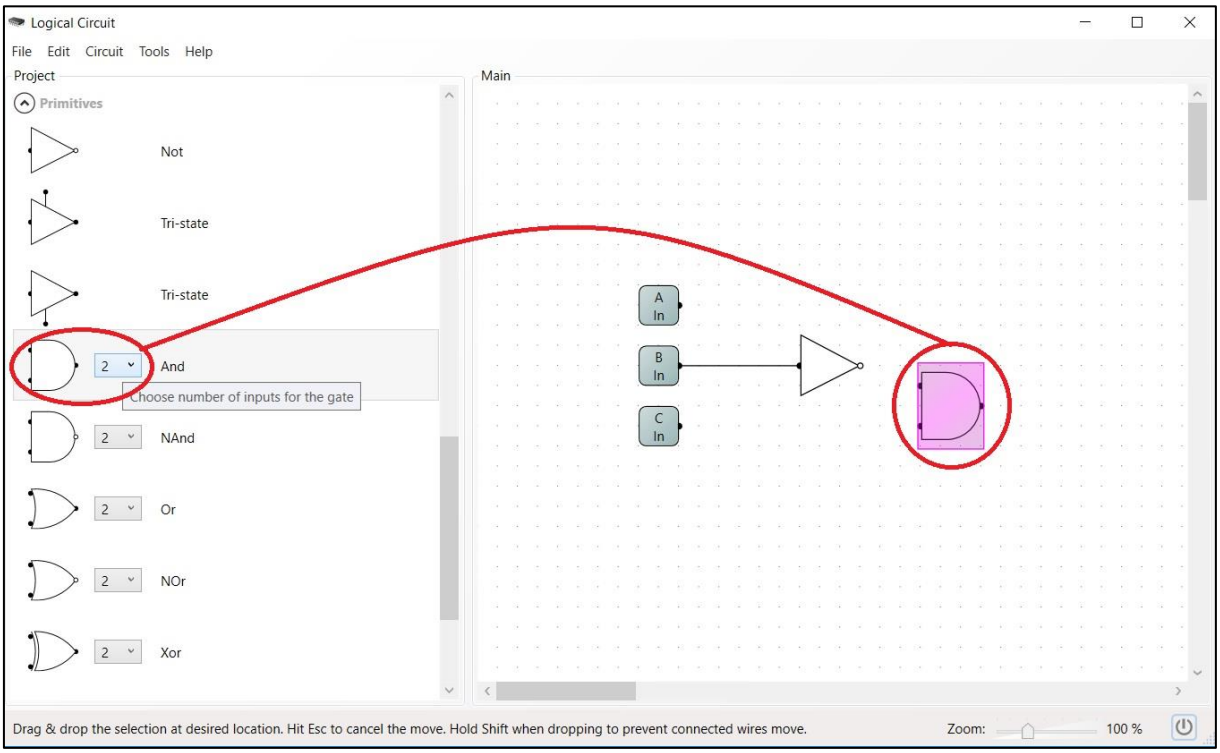
Εικόνα 6: Προσθήκη πύλης NOT

- Ενώνουμε το στοιχείο εισόδου **B** με την πύλη **NOT**. Για να γίνει αυτό, πατούμε πάνω στην κουκκίδα του στοιχείου εισόδου και ενώνουμε με την κουκκίδα της πύλης **NOT**.



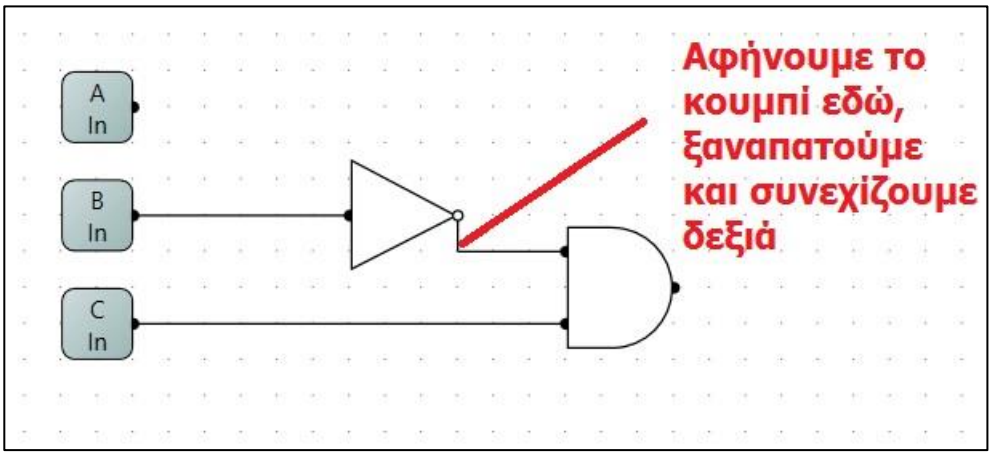
Εικόνα 7: Ενώνουμε τα δύο αντικείμενα

- Προσθέτουμε μία πύλη **AND** δύο εισόδων. Ο αριθμός δίπλα από το αντικείμενο καθορίζει τις εισόδους που θα έχει η λογική πύλη.



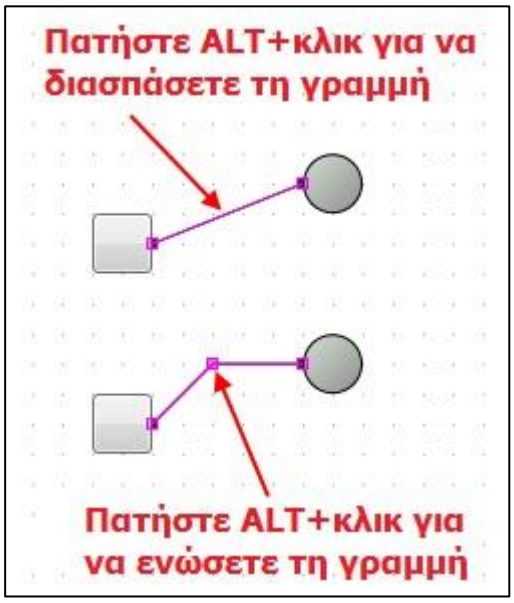
Εικόνα 8: Προσθήκη πύλης AND

- Ενώνουμε το στοιχείο εισόδου **C** και την πύλη **NOT** με την πύλη **AND**. Για να βάλουμε κλίση στη γραμμή, αφήνουμε το κουμπί του ποντικιού στο σημείο που θα εφαρμόσουμε την κλίση και συνεχίζουμε προς την άλλη κατεύθυνση.



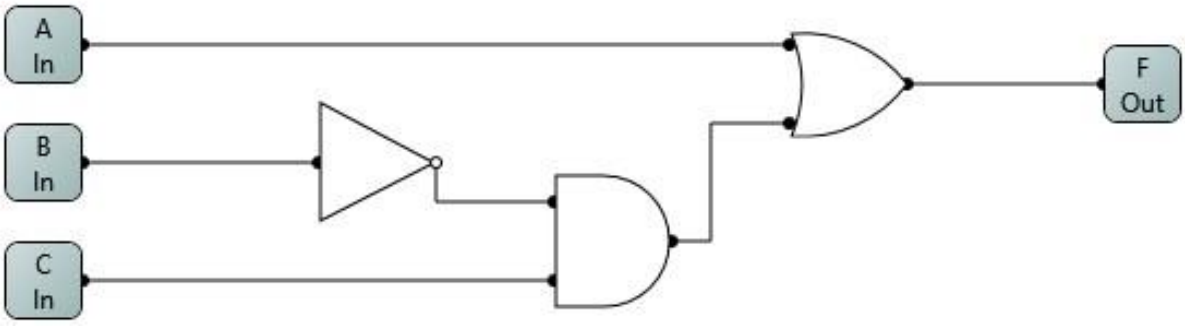
Εικόνα 9: Εφαρμόζουμε κλίση σε γραμμή

- Εναλλακτικά, μπορούμε να διασπάσουμε ή να ενώσουμε μία γραμμή χρησιμοποιώντας το πλήκτρο **ALT** όπως πιο κάτω:



Εικόνα 10: Συνδυασμός ALT+κλικ

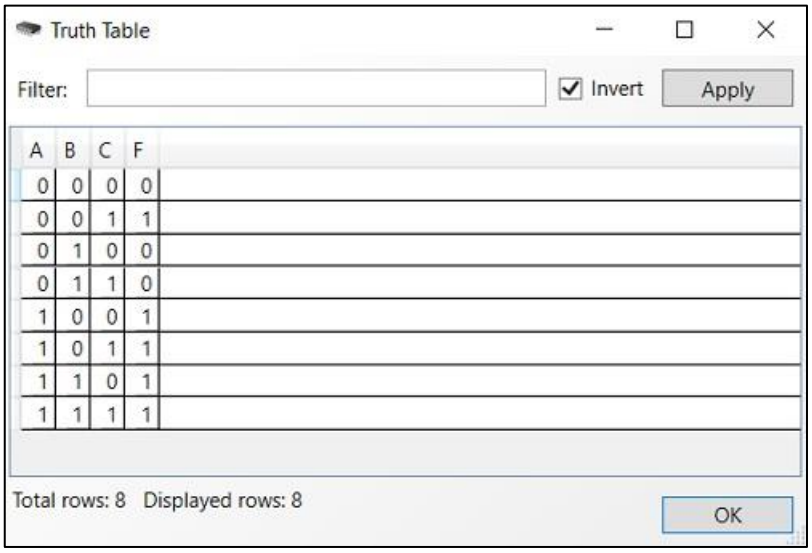
- Ολοκληρώνουμε το λογικό κύκλωμα όπως βλέπετε στην πιο κάτω εικόνα, προσθέτοντας τη λογική πύλη **OR** και το στοιχείο εξόδου **F**.



Εικόνα 11: Λογικό κύκλωμα συνάρτησης F = A + B' * C

4. Δημιουργία Πίνακα Αλήθειας

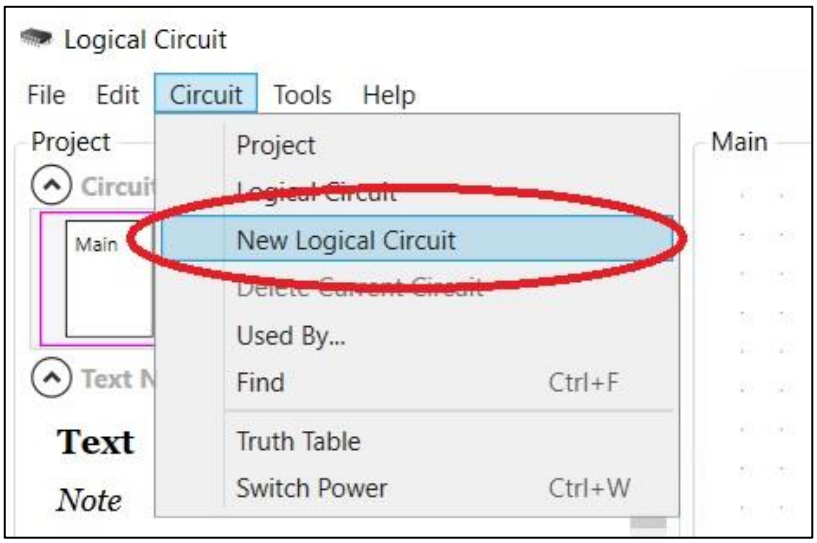
- Για να δημιουργήσουμε τον πίνακα αλήθειας για το λογικό κύκλωμα που έχουμε σχεδιάσει, επιλέγουμε από το μενού **Circuit -> Truth Table**.



Εικόνα 12: Πίνακας Αλήθειας της συνάρτησης F = A + B' * C

5. Δημιουργία Σύνθετων Κυκλωμάτων

- Για να δημιουργήσουμε πιο σύνθετα κυκλώματα, μπορούμε να χρησιμοποιήσουμε από το μενού την επιλογή **Circuit -> New Logical Circuit**.



Εικόνα 13: Δημιουργία σύνθετου κυκλώματος

- Για να μορφοποιήσουμε τις παραμέτρους του κυκλώματος, επιλέγουμε:

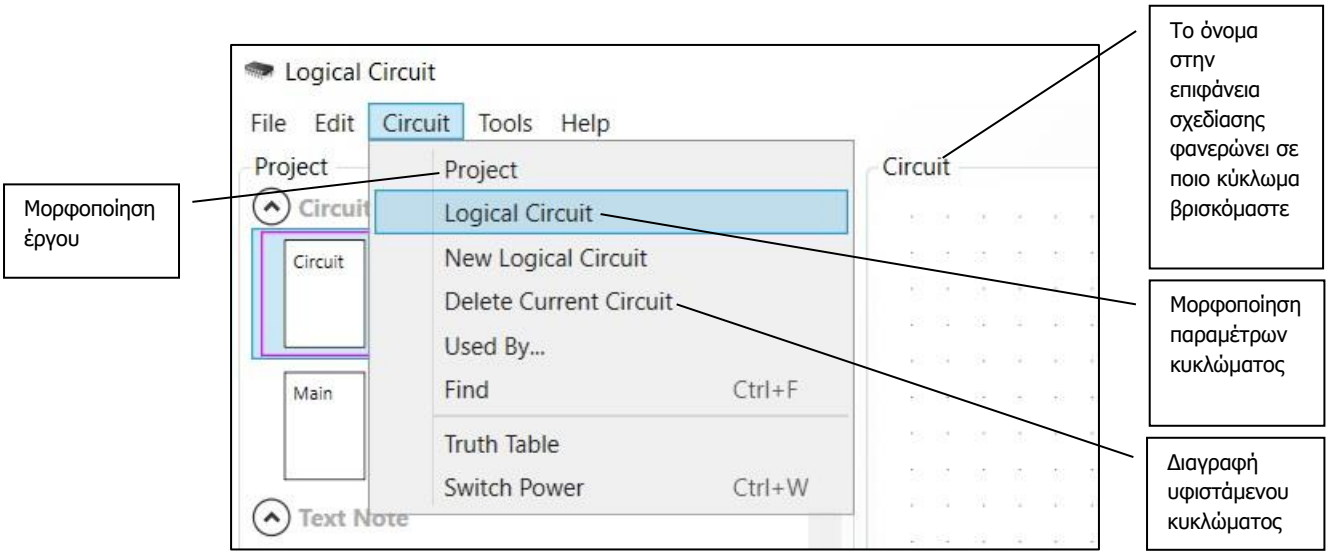
  Circuit -> Logical Circuit.

- Για να δημιουργήσουμε νέο κύκλωμα, επιλέγουμε:

  Circuit -> New Logical Circuit.

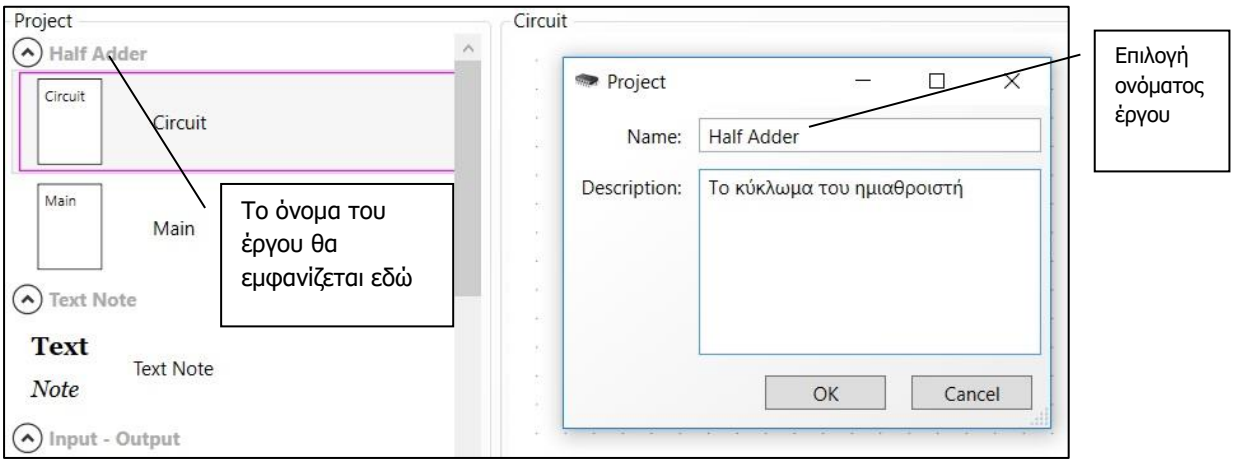
- Για να διαγράψουμε το κύκλωμα, επιλέγουμε:

  Circuit -> Delete Current Circuit.



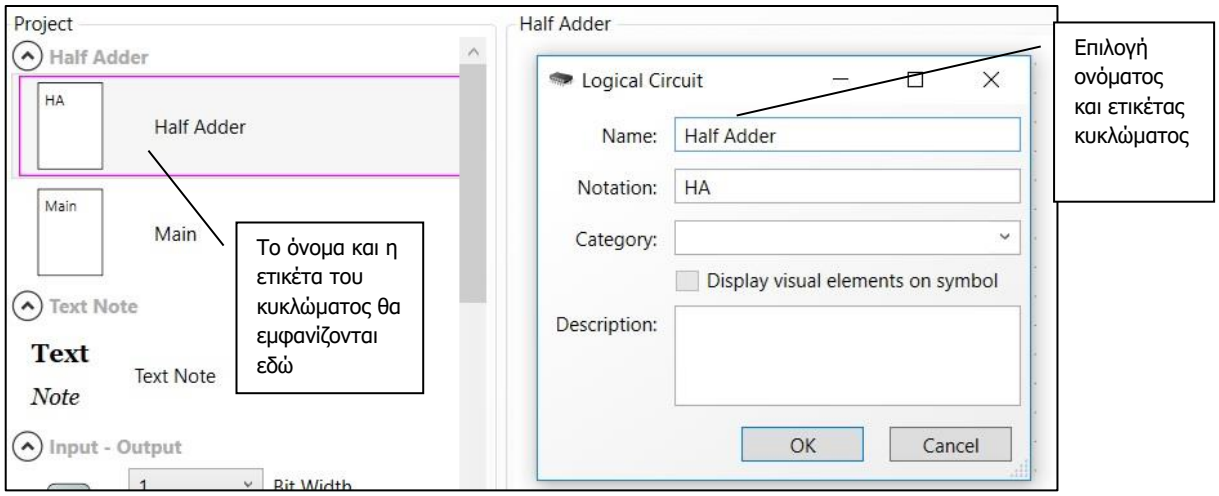
Εικόνα 14: Επιλογές μορφοποίησης κυκλώματος

- Με την επιλογή **Project** μπορούμε να μορφοποιήσουμε τις παραμέτρους του έργου.



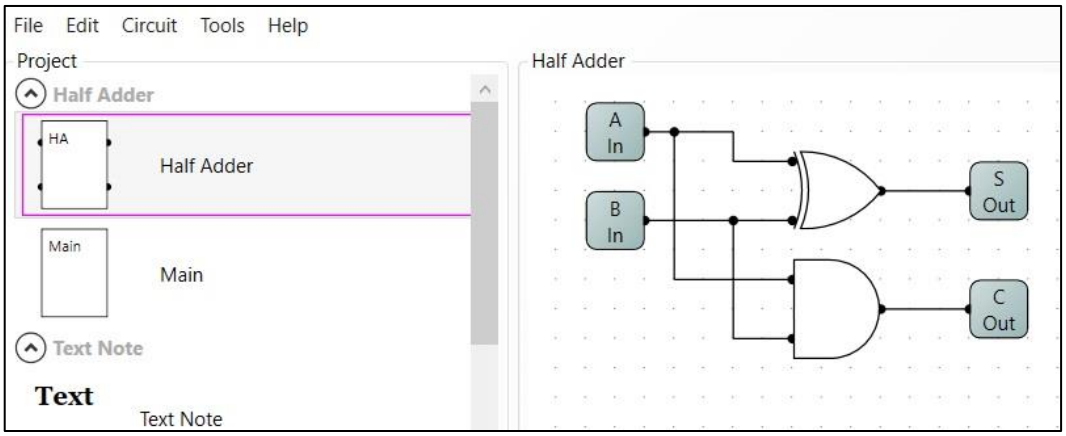
Εικόνα 15: Μορφοποίηση παραμέτρων έργου

- Με την επιλογή **Logical Circuit** μπορούμε να μορφοποιήσουμε τις παραμέτρους του κυκλώματος.



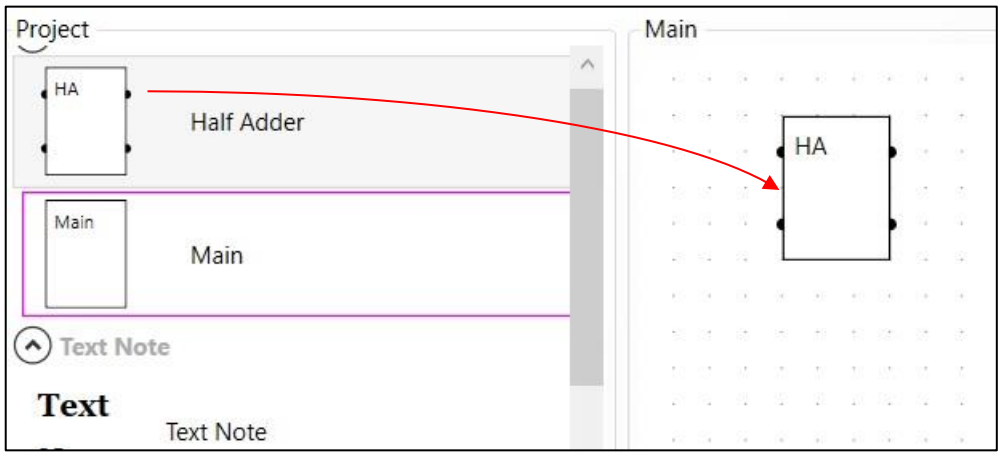
Εικόνα 16: Μορφοποίηση παραμέτρων κυκλώματος

- Σχεδιάζουμε το λογικό κύκλωμα του Ημιαθροιστή (Half-Adder).



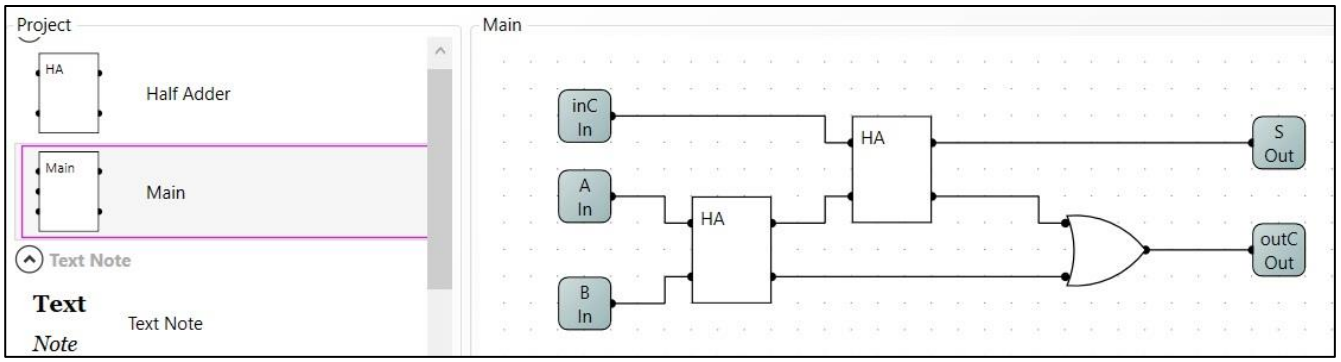
Εικόνα 17: Το λογικό κύκλωμα του Ημιαθροιστή

- Αν μεταφερθούμε στην κύρια σχεδιαστική επιφάνεια (Main), μπορούμε να χρησιμοποιήσουμε το λογικό κύκλωμα που έχουμε δημιουργήσει.



Εικόνα 18: Εισαγωγή κυκλώματος σε άλλο κύκλωμα

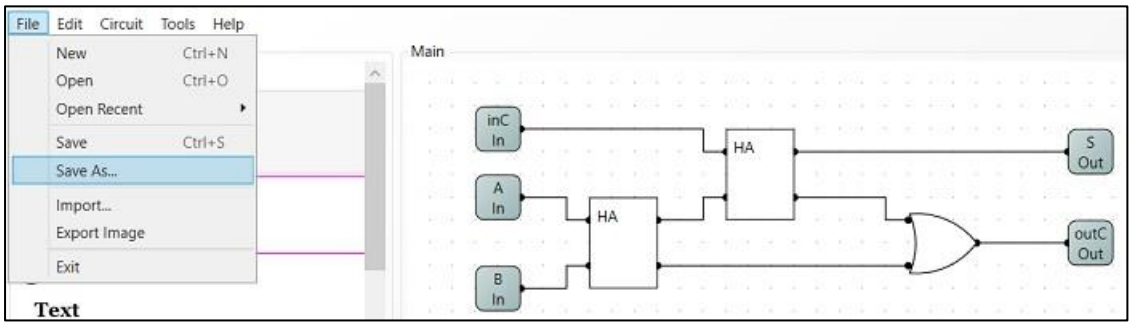
- Με τη χρήση δύο σύνθετων κυκλωμάτων Ημιαθροιστών μπορούμε να δημιουργήσουμε έναν Πλήρη Αθροιστή όπως πιο κάτω.



Εικόνα 19: Κύκλωμα Πλήρους Αθροιστή με δύο Ημιαθροιστές

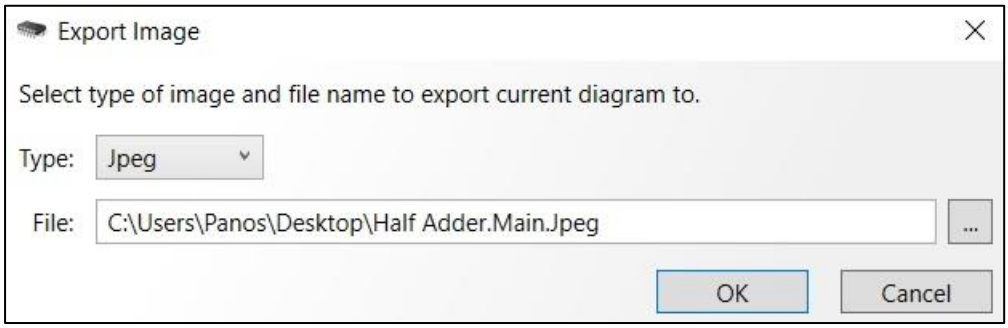
6. Αποθήκευση και Εξαγωγή Κυκλωμάτων

- Με τις επιλογές Save / Save As, μπορούμε να αποθηκεύσουμε το κύκλωμα. Όλα τα αρχεία στο **Logic Circuit**© αποθηκεύονται με την κατάληξη `CircuitProject`.



Εικόνα 20: Αποθήκευση κυκλώματος

- Με τη επιλογή Export Image, μπορούμε να εξαγάγουμε το κύκλωμα που έχουμε δημιουργήσει σε εικόνα. Το **Logic Circuit**© υποστηρίζει όλες τις γνωστές μορφοποιήσεις εικόνας.



Εικόνα 21: Εξαγωγή κυκλώματος σε εικόνα

Ασκήσεις Κεφαλαίου

Άσκηση 1.1

Να γράψετε δίπλα από κάθε δήλωση Σωστό ή Λάθος, ανάλογα.

(α) Η συνάρτηση F = x' + y + z έχει τέσσερις μεταβλητές.

(β) Αν η συνάρτηση που αναπαριστά ένα λογικό κύκλωμα έχει τρεις μεταβλητές, αυτές μπορούν να συνδυαστούν για να σχηματίσουν συνολικά $3^2 = 9$ ελαχιστόρους (minterms).

(γ) Οι χάρτες Karnaugh χρησιμοποιούνται για να απλοποιήσουμε τους όρους μίας λογικής συνάρτησης.

(δ) Η λογική συνάρτηση είναι ο τρόπος με τον οποίο διατυπώνονται τα θεωρήματα και οι σχέσεις της άλγεβρας Boole.

Άσκηση 1.2

Να συμπληρώσετε τους πιο κάτω πίνακες αλήθειας.

(α)

A	B	F = A + B’
0	0	
0	1	
1	0	
1	1	

(β)

A	B	F = A * B + A’
0	0	
0	1	
1	0	
1	1	

Άσκηση 1.3

Να υπολογίσετε τις τιμές των ακόλουθων εκφράσεων για τις τιμές των μεταβλητών a=0, b=1 και c=1.

(α) F = a + bc

(β) F = (a + b)(a + c’)

(γ) F = ab + bc + ac

Άσκηση 1.4

Να συμπληρώσετε τον ακόλουθο πίνακα αλήθειας.

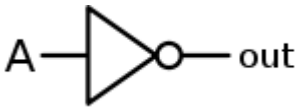
X	Y	X AND Y	X OR Y	X NAND Y	X NOR Y
True	True				
True	False				
False	True				
False	False				

Άσκηση 1.5

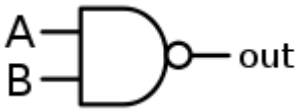
Να βρείτε σε ποιες λογικές πύλες αντιστοιχούν τα πιο κάτω σχήματα.

Α.

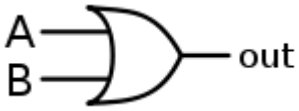
Β.



Γ.



Δ.



Άσκηση 1.6

Να βρείτε σε ποιες λογικές πύλες αντιστοιχούν οι πιο κάτω πίνακες αλήθειας.

Α.

Είσοδοι		Έξοδος
A	**B**	**A ? B**
0	0	1
0	1	1
1	0	1
1	1	0

Β.

Είσοδοι		Έξοδος
A	**B**	**A ? B**
0	0	0
0	1	1
1	0	1
1	1	1

Γ.

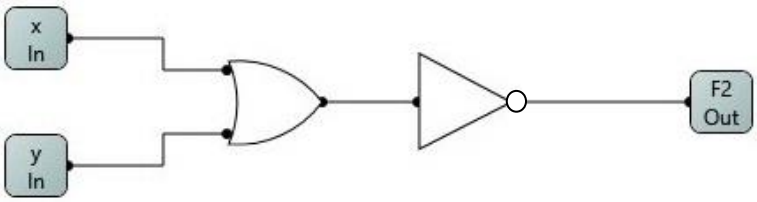
Είσοδοι		Έξοδος
A	**B**	**A ? B**
0	0	0
0	1	0
1	0	0
1	1	1

Δ.

Είσοδοι		Έξοδος
A	**B**	**A ? B**
0	0	1
0	1	0
1	0	0
1	1	0

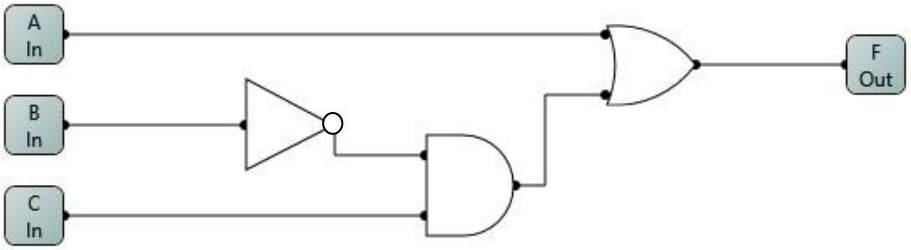
Άσκηση 1.7

Να δημιουργήσετε τον πίνακα αλήθειας για το πιο κάτω λογικό κύκλωμα.



Άσκηση 1.8

Σας δίνεται το πιο κάτω λογικό κύκλωμα:

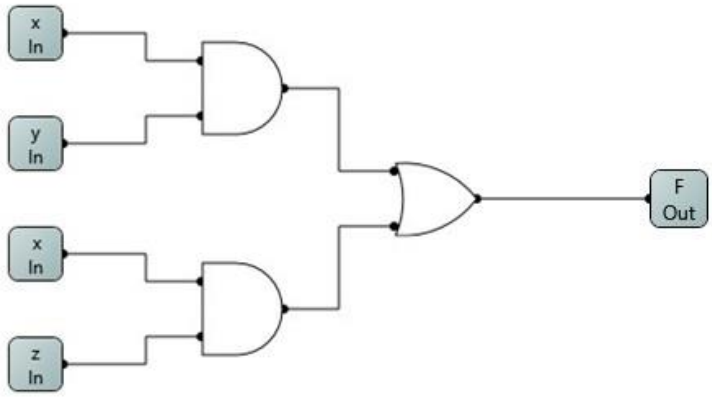


(α) Να βρείτε τη συνάρτηση του πιο πάνω λογικού κυκλώματος.

(β) Να βρείτε την τιμή της F, για αρχικές τιμές εισόδου A = 1, B = 1 και C = 1.

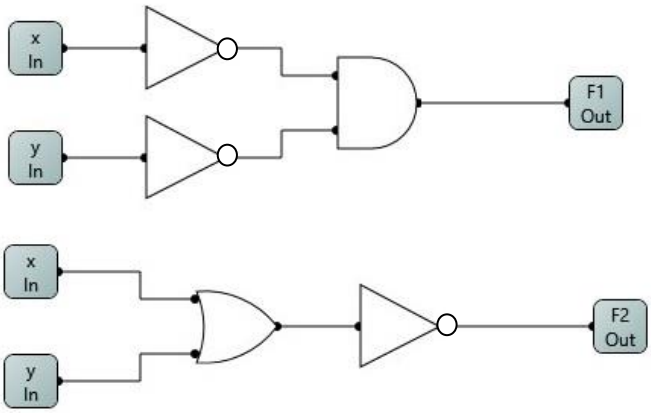
Άσκηση 1.9

Να βρείτε τη συνάρτηση που απεικονίζει το πιο κάτω λογικό κύκλωμα.



Άσκηση 1.10

Να δημιουργήσετε τους πίνακες αλήθειας για τα πιο κάτω λογικά κυκλώματα.



Άσκηση 1.11

Να σχεδιάσετε λογικό κύκλωμα, το οποίο να απεικονίζει τη συνάρτηση:

F (x, y, z) = (x + y) * z

Άσκηση 1.12

Σας δίνεται η συνάρτηση F = (a * b + c)'

(α) Να δημιουργήσετε τον πίνακα αλήθειας.

(β) Να σχεδιάσετε το λογικό κύκλωμα.

Άσκηση 1.13

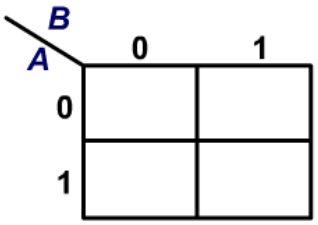
Να σχεδιάσετε λογικό κύκλωμα, το οποίο να απεικονίζει τη συνάρτηση:

F (X, Y, Z) = X'Y + XZ

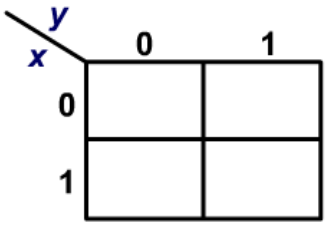
Άσκηση 1.14

Να συμπληρώσετε τους χάρτες Karnaugh για τις πιο κάτω λογικές συναρτήσεις.

(α) F = AB



(β) F = x + y



Άσκηση 1.15

Να συμπληρωθούν οι ελαχιστόροι (m) και οι μεγιστόροι (M) στον πιο κάτω πίνακα.

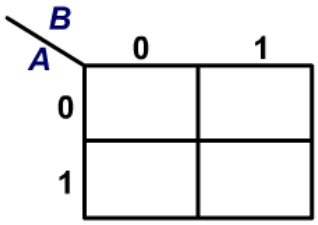
		Ελαχιστόροι		Μεγιστόροι	
A	**B**	**Όρος**	**Ονομασία**	**Όρος**	**Ονομασία**
0	0	A'B'	m_0	A + B	M_0
0	1				
1	0				
1	1				

Άσκηση 1.16

Να συμπληρώσετε τους χάρτες Karnaugh δύο μεταβλητών, οι οποίοι αντιστοιχούν στους πιο κάτω πίνακες αλήθειας.

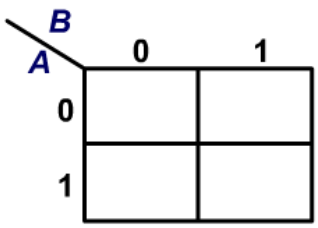
A.

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0



B.

A	B	Y
0	0	0
0	1	1
1	0	0
1	1	1



Άσκηση 1.17

Να βρείτε τις συναρτήσεις αθροισμάτων των ελαχιστόρων και γινομένων των μεγιστόρων, για τους πιο κάτω χάρτες Karnaugh δύο μεταβλητών.

(α)

A \ B	0	1
0	1	0
1	0	1

F (A, B) = Σ (_____)
F (A, B) = Π (_____)

(β)

A \ B	0	1
0	1	1
1	1	1

F (A, B) = Σ (_____)
F (A, B) = Π (_____)

(γ)

A \ B	0	1
0	0	0
1	0	1

F (A, B) = Σ (_____)
F (A, B) = Π (_____)

Άσκηση 1.18

Να ομαδοποιήσετε τους γειτονικούς όρους στους πιο κάτω χάρτες Karnaugh δύο μεταβλητών και να βρείτε τις συναρτήσεις που θα προκύψουν μετά την απλοποίηση.

(α) F = _____

A \ B	0	1
0	1	1
1	0	0

(β) F = _____

A \ B	0	1
0	1	1
1	0	1

(γ) F = _____

A \ B	0	1
0	0	1
1	0	1

Άσκηση 1.19

Σας δίνεται η συνάρτηση F = A′B′ + A′B + AB. Να σχεδιάσετε τον χάρτη Karnaugh, να ομαδοποιήσετε τους γειτονικούς όρους στον πίνακα και να απλοποιήσετε τη συνάρτηση. Να σχεδιάσετε το λογικό κύκλωμα που θα προκύψει μετά την απλοποίηση.

Άσκηση 1.20

Να συμπληρώσετε τις συναρτήσεις αθροισμάτων των ελαχιστόρων, με βάση τους πιο κάτω χάρτες Karnaugh τριών μεταβλητών.

x \ yz	00	01	11	10
0	0	1	0	1
1	0	1	0	1

(α) F(x, y, z) = Σ (_____)

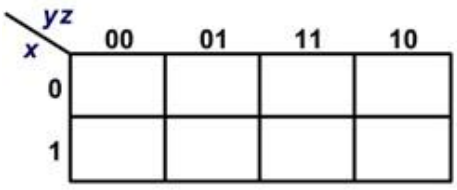
x \ yz	00	01	11	10
0	1	1	0	1
1	1	0	1	1

(β) F(x, y, z) = Σ (_____)

Άσκηση 1.21

Να συμπληρώσετε τον χάρτη Karnaugh τριών μεταβλητών, με βάση τον πιο κάτω πίνακα αλήθειας.

x	y	z	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0



Άσκηση 1.22

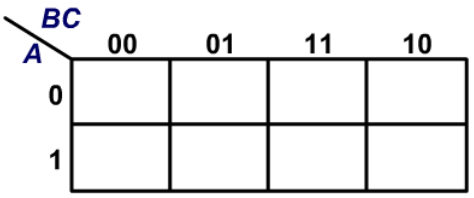
Να βρείτε τις συναρτήσεις για τις μεταβλητές F1 και F2, οι οποίες αντιστοιχούν στον πιο κάτω πίνακα αλήθειας.

x	y	z	F1	F2
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Άσκηση 1.23

Να συμπληρώσετε τον χάρτη Karnaugh, με βάση την πιο κάτω λογική συνάρτηση και να τον χρησιμοποιήσετε ώστε να απλοποιηθεί η συνάρτηση.

F(A,B,C) = A'C + A'B + AB'C + BC



Άσκηση 1.24

Να επιλέξετε τις συναρτήσεις που απλοποιούν, στον μέγιστο δυνατό βαθμό, τους πιο κάτω χάρτες Karnaugh.

(α)

x \ yz	00	01	11	10
0	0	0	1	0
1	1	0	1	1

(α) F (x, y, z) = x'y'z' + yz + xy

(β) F (x, y, z) = xz' + yz

(γ) F (x, y, z) = xz + y'z

(β)

x \ yz	00	01	11	10
0	1	0	0	1
1	1	1	0	1

(α) F (x, y, z) = x'y'z' + xy' + xz'

(β) F (x, y, z) = y'z' + xy' + yz'

(γ) F (x, y, z) = z' + xy'

Άσκηση 1.25

Να ομαδοποιήσετε τους γειτονικούς όρους στους πιο κάτω χάρτες Karnaugh τριών μεταβλητών και να βρείτε τις συναρτήσεις που θα προκύψουν μετά την απλοποίηση.

(α)

x \ yz	00	01	11	10
0	0	0	1	0
1	0	0	1	1

(β)

x \ yz	00	01	11	10
0	0	1	1	0
1	0	1	1	0

Άσκηση 1.26

Να συμπληρώσετε τον χάρτη Karnaugh με βάση την πιο κάτω λογική συνάρτηση και να τον χρησιμοποιήσετε ώστε να απλοποιηθεί η συνάρτηση.

F(x,y,z) = x'y'z' + xy'z' + x'yz' + xyz'

x \ yz	00	01	11	10
0				
1				

Άσκηση 1.27

Να συμπληρώσετε τις συναρτήσεις αθροισμάτων των ελαχιστόρων, με βάση τους πιο κάτω χάρτες Karnaugh.

AB \ CD	00	01	11	10
00	0	1	0	0
01	1	0	0	1
11	0	0	1	0
10	0	1	0	0

AB \ CD	00	01	11	10
00	1	1	0	1
01	1	1	1	1
11	0	0	1	0
10	1	1	0	1

(α) F(A, B, C, D) = Σ (_____) (β) F(A, B, C, D) = Σ (_____)

Άσκηση 1.28

Να συμπληρώσετε τον χάρτη Karnaugh, ο οποίος αντιστοιχεί στον πιο κάτω πίνακα αλήθειας.

A	B	C	D	F
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1

AB \ CD	00	01	11	10
00				
01				
11				
10				

Άσκηση 1.29

Σας δίνεται η συνάρτηση F = A'B'C'D' + A'BC'D' + ABC' + AB'CD + AB'CD'. Να συμπληρώσετε τον χάρτη Karnaugh και να τον χρησιμοποιήσετε ώστε να απλοποιηθεί η συνάρτηση.

AB \ CD	00	01	11	10
00				
01				
11				
10				

Άσκηση 1.30

Να συμπληρώσετε τον χάρτη Karnaugh με βάση την πιο κάτω συνάρτηση αθροίσματος των ελαχιστόρων και να βρείτε τη συνάρτηση που θα προκύψει μετά την απλοποίηση.

$$F(A,B,C,D) = \sum (0,1,2,5,8,9,10)$$

AB \ CD	00	01	11	10
00				
01				
11				
10				

Άσκηση 1.31

Να συμπληρώσετε τον χάρτη Karnaugh με βάση την πιο κάτω συνάρτηση αθροίσματος των ελαχιστόρων, να βρείτε τη συνάρτηση γινομένου των μεγιστόρων και τη συνάρτηση που θα προκύψει μετά την απλοποίηση.

$$F(A,B,C,D) = \sum (2,3,7,9,10,12,15)$$

AB \ CD	00	01	11	10
00				
01				
11				
10				

Άσκηση 1.32

Σας δίνεται η συνάρτηση F = A'B'CD' + A'BCD' + AB'CD' + ABC'D' + ABC'D + ABCD' + ABCD

(α) Να δημιουργήσετε τον πίνακα αλήθειας, να συμπληρώσετε τον χάρτη Karnaugh και να απλοποιήσετε τη συνάρτηση.

(β) Να σχεδιάσετε το λογικό κύκλωμα που θα προκύψει μετά την απλοποίηση.

Άσκηση 1.33

Να σχεδιάσετε λογικό κύκλωμα, το οποίο να ελέγχει τη λειτουργία ενός φωτεινού σηματοδότη ρύθμισης της κυκλοφορίας αυτοκινήτων. Το κύκλωμα αυτό θα πρέπει να ανιχνεύει οποιονδήποτε λανθασμένο συνδυασμό. Να σημειωθεί ότι σε κανονική λειτουργία μόνο ένα από τα τρία φώτα του σηματοδότη (πράσινο, πορτοκαλί ή κόκκινο) είναι αναμμένο και οποιοσδήποτε άλλος συνδυασμός αποτελεί σφάλμα λειτουργίας.

Άσκηση 1.34

Να αναπαραστήσετε με χάρτη Karnaugh την πιο κάτω λογική συνάρτηση τεσσάρων μεταβλητών.

Y (A, B, C, D) = ABCD + A'BC + C'D'

Άσκηση 1.35

Έστω ότι κάποια επιχείρηση θέτει τους ακόλουθους όρους για την πρόσληψη ενός υπαλλήλου, δηλαδή ότι ο υποψήφιος πρέπει να είναι:

(α) άνδρας, παντρεμένος και κάτω των 40 ετών ή

(β) άνδρας, ανύπανδρος και κάτω των 40 ή

(γ) γυναίκα, παντρεμένη και κάτω των 40 ή

(δ) γυναίκα, ανύπανδρη και άνω των 40.

Να βρείτε ισοδύναμους όρους οι οποίοι να είναι διατυπωμένοι με πιο απλό τρόπο και να διατυπώσετε το πρόβλημα σε αλγεβρική μορφή.

Άσκηση 1.36

Πέντε φίλοι, ο Άρης, ο Βαγγέλης, ο Γιώργος, ο Δημήτρης και ο Ευαγόρας θέλουν να πάνε εκδρομή, αλλά θέτουν κάποιες προϋποθέσεις για τη συμμετοχή τους σε αυτή. Η εκδρομή θα γίνει, αν πληρούνται όλες οι ακόλουθες προϋποθέσεις:

- ή ο Α ή ο Β ή και οι δύο πρέπει να πάνε,
- ή ο Γ ή ο Ε αλλά όχι και οι δύο πρέπει να πάνε,
- ή ο Α και ο Γ θα πάνε μαζί ή κανείς τους,
- αν πάει ο Δ, τότε πρέπει να πάει και ο Ε,
- αν πάει ο Β, τότε πρέπει να πάνε μαζί ο Α και ο Δ.

Ζητούνται:

(α) Να απλοποιηθεί το σύνολο των προϋποθέσεων.

(β) Να βρεθεί αν θα γίνει οπωσδήποτε η εκδρομή.

(γ) Να διατυπωθούν απλούστερες προϋποθέσεις, κάτω από τις οποίες θα γινόταν η εκδρομή.

Άσκηση 1.37

Στην άρση βαρών, μία προσπάθεια ενός αθλητή θεωρείται έγκυρη, όταν την αποδέχονται τουλάχιστον δύο από τους τρεις κριτές (Α, Β, C) και άκυρη, σε αντίθετη περίπτωση. Να σχεδιάσετε λογικό κύκλωμα, χρησιμοποιώντας πύλες NOR και πύλες OR, το οποίο να δίνει έξοδο (Χ) ίση με λογικό 1, όταν μία προσπάθεια θεωρείται άκυρη. Να συμβολίσετε με λογικό 1 την αποδοχή της προσπάθειας από έναν κριτή και με λογικό 0 τη μη αποδοχή της.

Άσκηση 1.38

Το σύστημα ασφαλείας του χρηματοκιβωτίου μίας τράπεζας έχει δύο πόρτες, την εξωτερική (X) και την εσωτερική (S), που οδηγεί στον χώρο που φυλάγεται ο χρυσός της τράπεζας. Οι δύο αυτές πόρτες έχουν ηλεκτρονικές κλειδαριές με θέσεις για τρία κλειδιά, τα οποία έχουν:

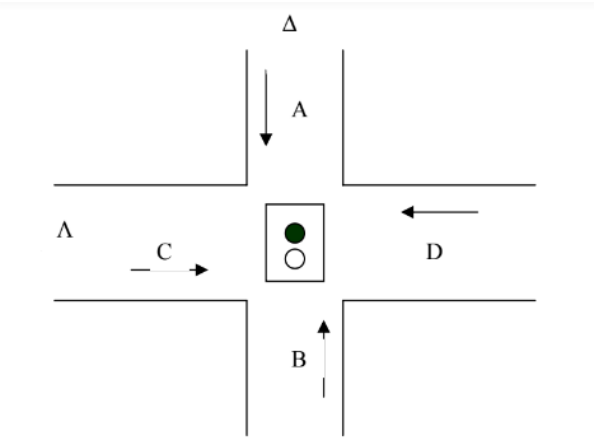
- Ο Διευθυντής (D), ο Υποδιευθυντής (Y) και ο Ταμίας (T).
- Η εξωτερική πόρτα ανοίγει με τα κλειδιά οποιωνδήποτε δύο από τους παραπάνω υπαλλήλους.
- Η εσωτερική πόρτα ανοίγει μόνο με τα κλειδιά και των τριών υπαλλήλων.
- Να συμβολίσετε με λογικό 1 την παρουσία ενός υπαλλήλου με το κλειδί και με λογικό 0, το αντίθετο. Να συμβολίσετε με λογικό 1 την κατάσταση μία πόρτα να είναι ανοιχτή και με λογικό 0 την κατάσταση να είναι κλειστή.

(α) Αν το σύστημα ασφαλείας υλοποιείται με ένα λογικό κύκλωμα, να καθορίσετε τις εισόδους και τις εξόδους του και να δημιουργήσετε τον πίνακα αλήθειας.

(β) Να απλοποιήσετε το κύκλωμα, χρησιμοποιώντας χάρτη Karnaugh και να το σχεδιάσετε με τις κατάλληλες λογικές πύλες.

Άσκηση 1.39

Το παρακάτω σχήμα δείχνει τη διασταύρωση μίας λεωφόρου Λ με έναν δρόμο Δ. Στις γραμμές A, B, C, D έχουν τοποθετηθεί ανιχνευτές οχημάτων που δίνουν έξοδο 0, όταν δεν είναι κατειλημμένες για 50 μέτρα πριν από τη διασταύρωση και 1, όταν συμβαίνει το αντίθετο.



Το σύστημα ελέγχου της κυκλοφορίας λειτουργεί με τον ακόλουθο τρόπο:

Το φανάρι που επιτρέπει την κυκλοφορία στη λεωφόρο Λ γίνεται πράσινο και το φανάρι στον δρόμο Δ γίνεται κόκκινο, όταν συμβαίνει τουλάχιστον μία από τις παρακάτω περιπτώσεις:

- Και οι δύο γραμμές C και D είναι κατειλημμένες.
- Δεν υπάρχουν οχήματα στις γραμμές A, B, C και D.
- Μία τουλάχιστον από τις γραμμές C, D είναι κατειλημμένη, αλλά οι γραμμές A, B δεν είναι και οι δύο κατειλημμένες.
- Σε οποιαδήποτε άλλη περίπτωση, το φανάρι που επιτρέπει την κυκλοφορία στη λεωφόρο Λ είναι κόκκινο και το φανάρι που επιτρέπει την κυκλοφορία στον δρόμο Δ είναι πράσινο.

(α) Με βάση την παραπάνω λογική, να δημιουργηθεί ο πίνακας αλήθειας και ο αντίστοιχος χάρτης Karnaugh.

(β) Να σχεδιάσετε λογικό κύκλωμα για την υλοποίηση του πιο πάνω προβλήματος.

Άσκηση 1.40

Να απαντήσετε τις πιο κάτω ερωτήσεις:

(α) Πόσες εισόδους έχει ένας Ημιαθροιστής;

(β) Πόσες και ποιες εξόδους έχει ένας Ημιαθροιστής;

(γ) Πόσες εισόδους έχει ένας Πλήρης Αθροιστής;

(δ) Πόσες και ποιες εξόδους έχει ένας Πλήρης Αθροιστής;

ΕΝΟΤΗΤΑ Γ7 Αλγοριθμική Σκέψη, Προγραμματισμός και Σύγχρονες Εφαρμογές Πληροφορικής

Οι ασκήσεις της ενότητας που σημειώνονται με το **λογότυπο του HackerRank**, έχουν αναρτηθεί στον πιο κάτω σύνδεσμο, όπου μπορείτε να υποβάλετε τις λύσεις σας:

 https://www.hackerrank.com/c-lyceum

Οι περισσότερες από τις υπόλοιπες ασκήσεις της ενότητας, αλλά και επιπρόσθετες ασκήσεις, βρίσκονται στους συνδέσμους που εμφανίζονται στην επικεφαλίδα των ασκήσεων κάθε κεφαλαίου, οι οποίοι παρουσιάζονται συνοπτικά πιο κάτω:

Γ7.1 Επανάληψη Β' Λυκείου	https://www.hackerrank.com/g71
Γ7.2 Συμβολοσειρές	https://www.hackerrank.com/g72
Γ7.3 Αρχεία	(δεν υποστηρίζονται από την πλατφόρμα)
Γ7.4 Συναρτήσεις	https://www.hackerrank.com/g74
Γ7.5 Αλγόριθμοι Αναζήτησης	https://www.hackerrank.com/g75
Γ7.6 Αλγόριθμοι Ταξινόμησης	https://www.hackerrank.com/g76
Γ7.7 Δισδιάστατοι Πίνακες	https://www.hackerrank.com/g77
Γ7.8 Εγγραφές	https://www.hackerrank.com/g78

Γ7.1 Επανάληψη ύλης Β' Λυκείου

Τι θα μάθουμε σε αυτό το κεφάλαιο:

- Να χρησιμοποιούμε τις απαραίτητες μεταβλητές/σταθερές με κατάλληλους τύπους δεδομένων, εντολές εισόδου και εξόδου, για να δημιουργήσουμε απλά προγράμματα
- Να χρησιμοποιούμε τελεστές και συναρτήσεις από βιβλιοθήκες για την παράσταση εκφράσεων και την επεξεργασία δεδομένων
- Να ακολουθούμε τα βήματα του κύκλου ανάπτυξης, για να λύνουμε σύνθετα προβλήματα που χρειάζονται δομή διακλάδωσης ή και επανάληψης
- Να χρησιμοποιούμε τους πίνακες για την επίλυση προβλημάτων.

1. Εισαγωγή

Σε αυτό το κεφάλαιο θα θυμηθούμε όλα όσα έχουμε μάθει κατά τη διάρκεια της προηγούμενης χρονιάς.

2. Απλά προγράμματα

Παρακάτω βλέπουμε ένα απλό πρόγραμμα στη C++ που υπολογίζει το άθροισμα 2 ακέραιων αριθμών.

```
#include<iostream>
using namespace std;
int main(){
    int a, b, sum;
    cin >> a >> b;
    sum = a + b;
    cout << "SUM=" << sum;
return 0;
}
```

Αναλύοντας τις εντολές του προγράμματος, καταλήγουμε στα παρακάτω.

```
#include<iostream>
```

Η παραπάνω εντολή δίνει οδηγία στο πρόγραμμα να συμπεριλάβει τη βιβλιοθήκη `<iostream>`. Η βιβλιοθήκη `<iostream>` μας επιτρέπει να χρησιμοποιήσουμε το πληκτρολόγιο και την οθόνη για είσοδο δεδομένων και έξοδο πληροφοριών. Γενικά, η δήλωση `#include <filename>` μάς επιτρέπει να δηλώσουμε τη χρήση βιβλιοθηκών, όπως η `iomanip` και η `cmath`.

```
int a, b, sum;
```

Στο πρόγραμμα έχουμε δηλώσει 3 μεταβλητές (a, b, sum) ακέραιου τύπου.

Στη C++ υπάρχουν οι ακόλουθοι τύποι δεδομένων:

Τύπος	Διάκριση	Bytes	Εμβέλεια
char	Χαρακτήρας	1	-128 έως 127
short	ακέραιος αριθμός	2	-32,768 έως 32,767
int	ακέραιος αριθμός	4	-2,147,483,648 έως 2,147,483,647
long long	ακέραιος αριθμός	8	$(-2^{63}+1)$ έως $(+2^{63}+1)$

float	πραγματικός αριθμός	4	1.2E-38 έως 3.4E+38
double	πραγματικός αριθμός	8	2.2E-308 έως 1.8E+308
long double	πραγματικός αριθμός	16	3.4E-4932 έως 1.2E+4932
bool	λογικές τιμές	1	true ή false

```
cin >> a >> b;
```

Η εντολή `cin` χρησιμοποιείται για την είσοδο δεδομένων από το πληκτρολόγιο. Μπορούμε να διαβάσουμε πολλές μεταβλητές, βάζοντας μπροστά από κάθε μεταβλητή τον τελεστή >>.

```
cout << "SUM=" << sum;
```

Η εντολή `cout` χρησιμοποιείται για την εμφάνιση πληροφοριών στην οθόνη. Μεταξύ των διπλών εισαγωγικών μπορούμε να συμπεριλάβουμε οποιαδήποτε ακολουθία λατινικών χαρακτήρων, συμβόλων κ.λπ.

2.1 Δήλωση Μεταβλητών και Σταθερών

Η δήλωση μεταβλητών γίνεται οπουδήποτε μέσα στην κύρια συνάρτηση (main) του προγράμματος. Η αρχικοποίηση μίας μεταβλητής μπορεί να γίνει κατά τη δήλωσή της.

```
int a;
int x,y,z;
int num = 10;
int add2 = 10 + 20;
int kpak = 4, jpak = kpak + 2;
double ratio = 25.56;
bool check = false;
char alpha = 'A', beta = 'B';
```

Οι σταθερές δηλώνονται είτε όπως οι μεταβλητές, με προσθήκη της δεσμευμένης λέξης `const` στην αρχή, είτε με την οδηγία `#define`. Κανένας από τους δύο τρόπους δεν επιτρέπει να αλλάξει η τιμή μετά την αρχικοποίηση. Η συντακτική ορθότητα μίας δήλωσης `const` ελέγχεται αμέσως, ενώ για το `#define` ελέγχεται μόνο αφού γίνει η αντικατάσταση. Αν υπάρχει λάθος, θα εμφανίζεται στη γραμμή του προγράμματος και όχι στη δήλωση του `#define`, σε αντίθεση με το `const`.

Παράδειγμα 1.1

Ο όγκος ενός κυλίνδρου δίνεται από τον τύπο $V = h * \pi * R^2$. Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται το ύψος (h) και την ακτίνα (R) του κυλίνδρου και να τυπώνει τον όγκο (V).

```
#include<iostream>
#define p 3.14              // Δήλωση σταθεράς με define
using namespace std;
int main(){
    float R, V, h;
    cout << "R=";
```

```
    cin >> R;
    cout << "h=";
    cin >> h;
    V = h * p * R * R;
    cout << "V=" << V;
return 0;
}
```

(Code: C7_1_example1.cpp)

2.2 Μορφοποίηση εξόδου

Κατά την προηγούμενη χρονιά μάθαμε 3 μορφοποιήσεις για την έξοδο.

2.2.1 Αλλαγή γραμμής

```
cout << "C++ is easy!" << endl;
```

Η λέξη `endl` είναι ένας χειριστής, ο οποίος εισάγει μία αλλαγή γραμμής (end line).

2.2.2 Καθορισμός μεγέθους διαστήματος εκτύπωσης (setw)

Η εντολή αυτή βρίσκεται στη βιβλιοθήκη `<iomanip>`. Η εντολή `setw` δέχεται ως παράμετρο έναν ακέραιο αριθμό που καθορίζει το διάστημα μέσα στο οποίο θα γίνει η εκτύπωση.

Το παρακάτω παράδειγμα θα τυπώσει τη φράση **Long_live_and_prosper**, όπου το σύμβολο _ αντιστοιχεί σε ένα κενό διάστημα (space).

```
#include<iostream>
#include<iomanip>
using namespace std;
int main(){
  cout<<"Long"<<setw(5)<<"live"<<setw(4)<<"and"<<setw(8)<<"prosper";
return 0;
}
```

2.2.3 Ακρίβεια δεκαδικών ψηφίων (setprecision)

Πολλές φορές υπάρχει η ανάγκη να καθορίσουμε τον αριθμό των δεκαδικών ψηφίων που θα εμφανίζονται μετά το αποτέλεσμα μίας πράξης. Η εντολή `setprecision` δέχεται ως παράμετρο έναν ακέραιο αριθμό που καθορίζει τον αριθμό των δεκαδικών ψηφίων που θα τυπωθούν. Να σημειώσουμε ότι γίνεται στρογγυλοποίηση στον αριθμό των ψηφίων που θα καθορίσει η `setprecision`. Τέλος, είναι απαραίτητο να γράψουμε την εντολή `fixed` πριν από την εντολή `setprecision`, αν θέλουμε να έχουμε ακριβώς τον αριθμό των ψηφίων που καθορίζει η εντολή. Τόσο η εντολή `setprecision` όσο και η εντολή `fixed` συμπεριλαμβάνονται στη βιβλιοθήκη `<iomanip>`.

Παράδειγμα 1.2

Ο ΦΠΑ για τα προϊόντα πολυτελείας είναι 23% της αρχικής τιμής. Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται το αρχικό ποσό και να τυπώνει την τελική τιμή με ακρίβεια 2 δεκαδικών ψηφίων.

```
#include<iostream>
#include<iomanip>
using namespace std;
int main(){
    const float fpa = 0.23;      // Δήλωση σταθεράς με const
    float cost;
    cin >> cost;
    cout << fixed << setprecision(2) << cost + cost * fpa;
return 0;
}
```

(Code: C7_1_example2.cpp)

2.3 Τελεστές, Όρια και Συναρτήσεις

Οι πέντε αριθμητικοί τελεστές που υποστηρίζονται από τη C++ είναι:

+	πρόσθεση
-	αφαίρεση
*	πολλαπλασιασμός
/	διαίρεση
%	υπόλοιπο ακέραιας διαίρεσης

Η σειρά με την οποία γίνονται οι πράξεις αλλάζει με τη χρήση παρενθέσεων. Είναι καλύτερα να χρησιμοποιούμε παρενθέσεις, γιατί κάνουν πιο σωστή τη σύνταξη του κώδικα και μας προφυλάσσουν από τυχόν λάθη. Η σειρά με την οποία ενεργούν οι τελεστές είναι:

( )	παρενθέσεις
*, /, %	πολλαπλασιασμός, διαίρεση, υπόλοιπο ακέραιας διαίρεσης
+, -	πρόσθεση, αφαίρεση

Όταν οι τελεστές έχουν την ίδια προτεραιότητα, τότε οι πράξεις εκτελούνται με τη σειρά, από αριστερά προς δεξιά.

Οι τελεστές προσαρμογής τύπου (type casting) μάς επιτρέπουν να μετατρέψουμε δεδομένα ενός τύπου σε δεδομένα διαφορετικού τύπου.

Παράδειγμα 1.3

Η στιγμιαία ταχύτητα ενός σώματος υπολογίζεται από τον τύπο U=X/T, όπου X η μετατόπιση και T ο χρόνος. Αν X και T είναι ακέραιοι αριθμοί, να υπολογίσετε την ταχύτητα και να την παρουσιάσετε με ακρίβεια 3 δεκαδικών ψηφίων.

```
#include<iostream>
#include<iomanip>
using namespace std;
int main(){
    float U;
    int X, T;
```

```
    cin >> X;
    cin >> T;
    U = float(X) / T;
    cout << "Speed= " << fixed << setprecision(3) << U;
return 0;
}
```

(Code: C7_1_example3.cpp)

Μία άλλη κατηγορία τελεστών είναι οι τελεστές αύξησης/μείωσης (++/--). Οι τελεστές μπορούν να χρησιμοποιηθούν ως πρόθεμα ή ως επίθεμα.

Υπενθυμίζεται, επίσης, ότι τα παρακάτω είναι ισοδύναμα:

```
a++; a += 1; a = a + 1;        // η τιμή του a αυξάνεται κατά ένα
b *= a; b = b * a;             // το b παίρνει την τιμή του γινομένου a*b
x /= y; x = x / y;             // το x παίρνει την τιμή του πηλίκου x/y
```

```
#include<iostream>
using namespace std;
int main(){
    int a, b, c, d;
    a = 4;
    b = ++a;
    cout << a << " " << b << endl;          // 5 5
    c = 4;
    d = c++;
    cout << c << " " << d << endl;          // 5 4
return 0;
}
```

Η βιβλιοθήκη <climits> καθορίζει ως σταθερές (INT_MAX, INT_MIN), τα αριθμητικά όρια για μεταβλητές ακέραιου τύπου.

```
#include <iostream>
#include <climits>                  // Integer limits
using namespace std;
int main() {
   cout << INT_MAX << endl;         // 2147483647
   cout << INT_MIN << endl;         // -2147483647
return 0;
}
```

Στη C++, όπως και στις πλείστες γλώσσες προγραμματισμού, υπάρχουν υλοποιημένες συναρτήσεις που μπορεί να χρησιμοποιήσει ο προγραμματιστής. Ο αριθμός των συναρτήσεων είναι πολύ μεγάλος και δεν είναι απαραίτητο να τις γνωρίζουμε όλες. Μπορείτε να βρείτε σχετικό κατάλογο στη σελίδα http://www.cplusplus.com/. Η χρήση της κάθε συνάρτησης απαιτεί και τη δήλωση της βιβλιοθήκης στην οποία ανήκει.

Παρακάτω παρουσιάζουμε χρήσιμες συναρτήσεις που υπάρχουν στη βιβλιοθήκη <cmath>.

Συνάρτηση	Χρήση	Παράμετροι	Παράδειγμα
sqrt(x)	Επιστρέφει την τετραγωνική ρίζα του αριθμού x. Η επιστρεφόμενη τιμή είναι δεκαδικός αριθμός.	Ένας θετικός αριθμός (ακέραιος ή δεκαδικός)	sqrt(9)=3.0
abs(x)	Επιστρέφει την απόλυτη τιμή του αριθμού x. Η επιστρεφόμενη τιμή εξαρτάται από τον τύπο του αριθμού x.	Ένας αριθμός (ακέραιος ή δεκαδικός)	abs(-2)=2 abs(-3.1)=3.1
pow(x,y)	Επιστρέφει το αποτέλεσμα της δύναμης x^y. Η επιστρεφόμενη τιμή είναι δεκαδικός αριθμός.	Δύο δεκαδικοί αριθμοί	pow(2.0,3.0)=8.0 pow(2.0,-1.0)=0.5
trunc(x)	Επιστρέφει το ακέραιο μέρος του αριθμού x σε δεκαδική μορφή, αγνοώντας το δεκαδικό μέρος του.	Ένας δεκαδικός αριθμός	trunc(1.7)=1.0 trunc(-2.1)=-2.0
round(x)	Επιστρέφει το ακέραιο μέρος του αριθμού x σε δεκαδική μορφή, στρογγυλοποιημένο στην πλησιέστερη τιμή.	Ένας δεκαδικός αριθμός	round(1.5)=2.0 round(1.4)=1.0

Παράδειγμα 1.4

Να δημιουργήσετε πρόγραμμα, το οποίο να διαβάζει έναν δεκαδικό αριθμό και να εμφανίζει το δεκαδικό του μέρος στην οθόνη.

Παράδειγμα εισόδου

```
56.78
```

Παράδειγμα εξόδου

```
0.78
```

```
#include<iostream>
#include<cmath>              // Για να γίνει χρήση της trunc
using namespace std;
int main(){
    double N;
    cin >> N;
    cout << N - trunc(N);   // Εναλλακτικά:  cout << N - (int) N;
return 0;
}
```

(Code: C7_1_example4.cpp)

3. Δομή Διακλάδωσης

Στη δομή διακλάδωσης το πρόγραμμα εκτελεί μία εντολή ή μία ομάδα εντολών, με βάση το αποτέλεσμα μίας λογικής έκφρασης.

Η εντολή που χρησιμοποιούμε για να πάρει μία απόφαση το πρόγραμμά μας, είναι η εντολή `if`. Η σύνταξη της εντολής έχει ως ακολούθως:

```
if (λογική έκφραση) {
     εντολές που εκτελούνται, αν ισχύει η λογική έκφραση
}
else {
     εντολές που εκτελούνται, αν δεν ισχύει η λογική έκφραση
}
```

Παρατηρήσεις:

- Μπορεί να έχουμε την εντολή `if` χωρίς το `else`. Δηλαδή, σε περίπτωση που δεν ισχύει η λογική έκφραση να μην εκτελείται άλλη εντολή.
- Τα άγκιστρα μπορούν να παραλείπονται, αν έχουμε μόνο μία εντολή.
- Προκειμένου να συγκρίνουμε δύο αριθμητικές παραστάσεις ως προς την τιμή τους, χρησιμοποιούμε τους συγκριτικούς τελεστές:

==	ίσο
!=	διάφορο / άνισο
>	μεγαλύτερο
<	μικρότερο
>=	μεγαλύτερο ή ίσο
<=	μικρότερο ή ίσο

Προσοχή στον έλεγχο της ισότητας. Ο τελεστής ισότητας είναι το == και όχι το =.

Παράδειγμα 1.5

Ένας ωρομίσθιος υπάλληλος, για κάθε ώρα που δουλεύει, πληρώνεται με 15 ευρώ. Αν το συνολικό ποσό που θα εισπράξει ξεπερνά τα 1000 ευρώ, θα του επιβληθεί φορολογία 13%. Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται τις ώρες που δούλεψε ο υπάλληλος μέσα σε μία εβδομάδα και να εμφανίζει τον εβδομαδιαίο μισθό του.

```
#include<iostream>
#define tax 0.13
#define amount 15
using namespace std;

int main(){
    int h;
    float salary;
    cin >> h;
    salary = h * amount;
```

```
    if(salary>1000)
        salary -= salary * tax;
    cout << salary;
return 0;
}
```

(Code: C7_1_example5.cpp)

Μπορούμε να δημιουργήσουμε πιο σύνθετες λογικές προτάσεις, χρησιμοποιώντας τους λογικούς τελεστές AND (&&), OR (||) και NOT (!).

Ο πίνακας αλήθειας των λογικών τελεστών είναι ο ακόλουθος:

X	Y	X && Y	X || Y	!X
True	True	True	True	False
True	False	False	True	False
False	True	False	True	True
False	False	False	False	True

Η προτεραιότητα των τελεστών είναι (ξεκινώντας από τη μεγαλύτερη):

1. ! (NOT)
2. && (AND)
3. || (OR)

Παράδειγμα 1.6

Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται ένα γράμμα του λατινικού αλφαβήτου και αν το γράμμα είναι το 'Y' ή το 'y', να τυπώνει το μήνυμα «Yes», αν το γράμμα είναι το 'N' ή το 'n', να τυπώνει το μήνυμα «No», αλλιώς να τυπώνει το μήνυμα «Wrong Input».

```
#include<iostream>
using namespace std;

int main(){
    char c;
    cin >> c;
    if(c=='Y' || c=='y')
        cout << "Yes";
    else
        if(c=='N' || c=='n')
            cout << "No";
        else
            cout << "Wrong Input";
return 0;
}
```

(Code: C7_1_example6.cpp)

4. Περιπτωσιακή Δομή – Η εντολή switch

Με την εντολή `switch` μπορούμε να εκτελέσουμε μία ομάδα εντολών, αναλόγως της τιμής που θα έχει η μεταβλητή ελέγχου. Η διαφορά με το `if` είναι ότι δεν εξετάζει κάποια λογική έκφραση, αλλά τις τιμές που μπορεί να πάρει μία μεταβλητή ελέγχου.

Γενικά, η εντολή `switch` δεν χρησιμοποιείται ευρέως, παρά μόνο σε ειδικές περιπτώσεις.

Σύνταξη Εντολής

```
switch(μεταβλητή ελέγχου){
    case περίπτωση 1:
       εντολές;
       break;
    case περίπτωση 2:
       εντολές;
       break;
    // Μπορείτε να έχετε όσες περιπτώσεις θέλετε
    default:
       εντολές;
}
```

Επεξήγηση

Η μεταβλητή ελέγχου πρέπει να είναι διακριτού τύπου (`integer`, `char`).

Όταν βρεθεί περίπτωση που θα έχει την ίδια τιμή με την μεταβλητή ελέγχου, τότε εκτελούνται οι εντολές που βρίσκονται από κάτω, μέχρι να βρεθεί η εντολή `break`. Όταν εκτελεστεί η εντολή `break`, η `switch` τερματίζεται.

Αν θέλετε να εκτελεστούν οπωσδήποτε κάποιες εντολές μέσα στη `switch`, τότε πρέπει να τις βάλετε μετά το `default`. Οι εντολές αυτές θα εκτελεστούν, εφόσον δεν βρεθεί καμιά περίπτωση που να είναι ίση με την τιμή της μεταβλητής ελέγχου.

Παράδειγμα 1.7

Σας δίνεται ο πιο κάτω πίνακας που περιλαμβάνει τον κωδικό και την τιμή ενός προϊόντος.

Κωδικός Προϊόντος	Τιμή
1	€ 4.00
2, 3, 4	€ 4.50
5, 10, 15	€ 5.00
7, 9	€ 2.00
12	€ 1.50

Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται τον κωδικό (code) και την ποσότητα ενός προϊόντος (qty) και να τυπώνει το συνολικό ποσό (total), με ακρίβεια δύο δεκαδικών ψηφίων.

Παράδειγμα εισόδου

```
3 10
```

Παράδειγμα εξόδου

```
45.00
```

```
#include<iostream>
#include<iomanip>
using namespace std;

int main(){
    int code, qty;
    float price, total;
    cin >> code >> qty;
    switch (code){
        case 1:
                price = 4.0;
                break;
        case 2:
        case 3:
        case 4:
                price = 4.5;
                break;
        case 5:
        case 10:
        case 15:
                price = 5.0;
                break;
        case 7:
        case 9:
                price = 2.0;
                break;
        case 12:
                price = 1.5;
                break;
        default:
               cout <<  "Wrong input" << endl;
    }
    total = price * qty;
    cout << fixed << setprecision(2) << total;
return 0;
}
```

(Code: C7_1_example7.cpp)

5. Δομή Επανάληψης

Δομές επανάληψης ή βρόχοι (loops) ονομάζονται μέρη του κώδικα, που εκτελούνται περισσότερες από μία φορές, ανάλογα με τη συνθήκη που έχουμε δηλώσει.

Οι βρόχοι αναγκάζουν ένα τμήμα κώδικα να επαναλαμβάνεται. Η επανάληψη συνεχίζεται για όσο μία λογική συνθήκη είναι αληθής. Όταν η συνθήκη γίνει ψευδής, ο βρόχος τελειώνει και ο έλεγχος του προγράμματος συνεχίζεται παρακάτω. Στη C++ υπάρχουν τρία είδη βρόχων: ο βρόχος `for`, ο βρόχος `while` και ο βρόχος `do`.

5.1 Ο βρόχος for

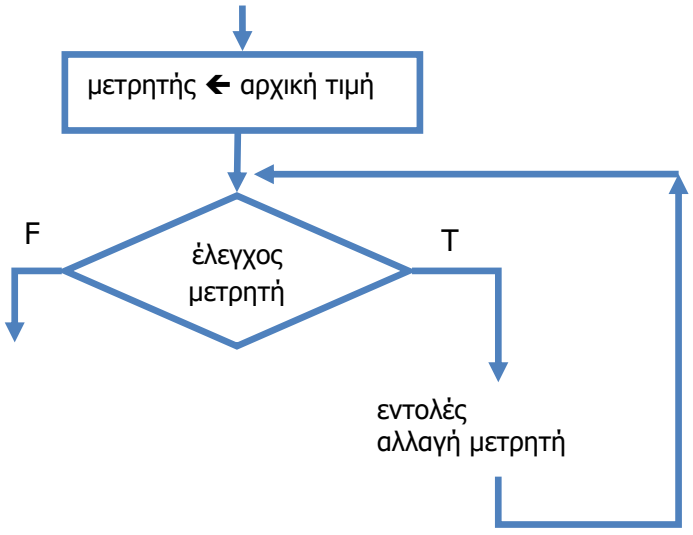
Ο βρόχος `for` είναι η πιο απλή επαναληπτική δήλωση, καθώς μας επιτρέπει να δημιουργούμε επαναληπτικές δομές με την ελάχιστη δυνατή χρήση κώδικα. Ο βρόχος `for` χρησιμοποιείται όταν γνωρίζουμε από πριν τον ακριβή αριθμό των επαναλήψεων. Μετά τη δεσμευμένη λέξη `for` ακολουθούν μέσα σε παρένθεση οι οδηγίες εκτέλεσης του βρόχου. Μία μεταβλητή παίζει τον ρόλο του μετρητή των επαναλήψεων. Έτσι, η μεταβλητή-μετρητής λαμβάνει αρχική τιμή. Αυτή η εντολή εκτελείται μόνο την πρώτη φορά, στην αρχή του βρόχου. Ακολουθεί μία λογική συνθήκη. Όσο η λογική συνθήκη είναι αληθής, ο βρόχος θα επαναλαμβάνει τις εντολές που περιέχει. Όταν η λογική συνθήκη γίνει ψευδής, ο έλεγχος του προγράμματος συνεχίζει με το υπόλοιπο πρόγραμμα εκτός βρόχου. Η εξέταση της λογικής συνθήκης γίνεται πριν ο έλεγχος του προγράμματος εισέλθει στον βρόχο. Η τελευταία παράμετρος μέσα στις παρενθέσεις είναι η μεταβολή του μετρητή.

Ο μετρητής μας μπορεί να είναι ακέραιος, δεκαδικός ή χαρακτήρας. Μπορούμε, επίσης, να δηλώσουμε και να αρχικοποιήσουμε τον μετρητή μέσα στον βρόχο.

```
for (μετρητής = αρχική τιμή; συνθήκη; μεταβολή μετρητή){
      εντολές;
}
```

Τα άγκιστρα μπορούν να παραλείπονται αν έχουμε μόνο μία εντολή.

Το λογικό διάγραμμα του βρόχου `for` είναι το παρακάτω.



Παράδειγμα 1.8

Να δημιουργήσετε πρόγραμμα, το οποίο να ζητά έναν ακέραιο αριθμό Ν και να βρίσκει και να εμφανίζει στην οθόνη το άθροισμα όλων των αριθμών από το 1 μέχρι το Ν. Για παράδειγμα, αν ο χρήστης δώσει τον αριθμό 10, το πρόγραμμα θα βρίσκει το άθροισμα:

```
S = 1+2+3+4+5+6+7+8+9+10 = 55
```

```
#include<iostream>
using namespace std;

int main(){
    int N, sum = 0;
    cin >> N;
    for(int i=1; i<=N; i++)
        sum += i;
    cout << sum;
return 0;
}
```

(Code: C7_1_example8.cpp)

Σε πολλές περιπτώσεις μπορούμε να χρησιμοποιήσουμε επαναληπτικές εντολές μέσα σε άλλες επαναληπτικές εντολές. Στην περίπτωση αυτή, για κάθε επανάληψη της εξωτερικής εντολής, η εσωτερική θα εκτελέσει όλες τις δικές τις επαναλήψεις.

Παράδειγμα 1.9

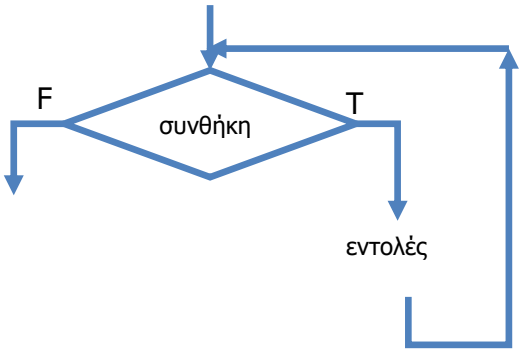
Να δημιουργήσετε πρόγραμμα, το οποίο να τυπώνει τον πίνακα προπαίδειας (9 x 9).

```
#include<iostream>
using namespace std;

int main(){
    int i, j;
    for(i=1; i<=9; i++){
        for(j=1; j<=9; j++){
            cout << i * j << " ";
        }
        cout << endl;
    }
return 0;
}
```

(Code: C7_1_example9.cpp)

5.2 Ο βρόχος while

Ο βρόχος `while` μάς επιτρέπει να δημιουργήσουμε δομές επανάληψης, όταν δεν ξέρουμε τον ακριβή αριθμό των επαναλήψεων. Για να επαναλάβει τις εντολές του ο βρόχος `while`, ελέγχει τη λογική συνθήκη και εάν αυτή είναι αληθής, τότε επαναλαμβάνει τις εντολές που του έχουμε ορίσει. Όταν εκτελέσει όλες τις εντολές του, ο έλεγχος μεταβιβάζεται πάλι στην αρχή και επανεξετάζει τη λογική συνθήκη. Όσο η λογική συνθήκη παραμένει αληθής, ο βρόχος θα επαναλαμβάνεται. Όταν η λογική συνθήκη γίνει ψευδής, τότε ο έλεγχος του προγράμματος θα συνεχίσει με το υπόλοιπο πρόγραμμα μετά από αυτόν. Το λογικό διάγραμμα του βρόχου `while` είναι το ακόλουθο:




```
while (συνθήκη){
      εντολές;
}
```

Τα άγκιστρα μπορούν να παραλείπονται αν έχουμε μόνο μία εντολή.

Παράδειγμα 1.10

Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται δύο ακέραιους θετικούς αριθμούς και να εμφανίζει το ελάχιστο κοινό πολλαπλάσιό τους.

```
#include<iostream>
using namespace std;

int main(){
    int a, b, ekp;
    cin >> a >> b;
    ekp = a;
    while (ekp%a!=0 || ekp%b!=0)
         ekp++;
    cout << ekp;
return 0;
}
```

(Code: C7_1_example10.cpp)

5.3 Ο βρόχος do

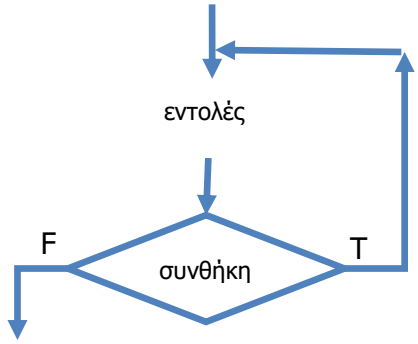
Ο βρόχος do χρησιμοποιείται όταν θέλουμε να διασφαλίσουμε ότι οι εντολές θα εκτελεστούν τουλάχιστον μία φορά, ανεξάρτητα από την τιμή της συνθήκης. Στο βρόχο do ο έλεγχος της συνθήκης γίνεται στο τέλος του βρόχου.

```
do {
      εντολές;
} while (συνθήκη);
```

Πρώτα, η δεσμευμένη λέξη do σηματοδοτεί την έναρξη του βρόχου. Έπειτα, ακολουθεί το σώμα του βρόχου. Στο τέλος, υπάρχει η λέξη while με τη λογική συνθήκη που ελέγχει μέσα σε παρενθέσεις. Προσοχή στο ερωτηματικό που υπάρχει μετά τη συνθήκη, το οποίο σηματοδοτεί το τέλος της εντολής. Όσο η λογική συνθήκη είναι αληθής, ο βρόχος θα εκτελεί

τις εντολές του. Όταν η λογική συνθήκη γίνει ψευδής, ο έλεγχος του προγράμματος θα συνεχίσει παρακάτω. Όπως βλέπετε, ο έλεγχος της λογικής συνθήκης με τη δεσμευμένη λέξη `while` γίνεται στο τέλος του βρόχου. Αυτό εξασφαλίζει την εκτέλεση του βρόχου τουλάχιστον μία φορά, ακόμα και όταν η λογική συνθήκη είναι ψευδής.

Το λογικό διάγραμμα για τον βρόχο `do` είναι το παρακάτω.



Παράδειγμα 1.11

Ένα ασανσέρ έχει ανώτατο όριο βάρους τα 600 κιλά ή τα 8 άτομα. Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται το βάρος κάθε ατόμου που μπαίνει, μέχρι που να μην μπορεί να σηκώσει άλλο βάρος ή να φτάσει το όριο των 8 ατόμων, οπόταν και θα τυπώνει το συνολικό βάρος και τον αριθμό των ατόμων που έχουν εισέλθει.

```
#include<iostream>
using namespace std;

int main(){
    int weight, people = 0, total = 0;
    cin >> weight;
    do {
         total += weight;
         people++;
         cin >> weight;
    } while(people!=8 && total+weight<=600);
    cout << "Total=" << total << endl;
    cout << "Persons=" << people << endl;
return 0;
}
```

(Code: C7_1_example11.cpp)

6. Έλεγχος Προγράμματος

Ο έλεγχος του προγράμματος είναι μία πολύ σημαντική διαδικασία. Θα μπορούσαμε να τον χωρίσουμε σε 2 κατηγορίες: α) συντακτικός, β) λογικός. Ο συντακτικός έλεγχος μπορεί να πραγματοποιηθεί εύκολα με τη βοήθεια του μεταγλωττιστή (Code::Blocks) της γλώσσας που χρησιμοποιούμε. Συνηθισμένα συντακτικά λάθη στη C++ είναι η μη δήλωση μεταβλητών και βιβλιοθηκών και η απουσία του ερωτηματικού ';' στο τέλος μίας εντολής. Ο λογικός έλεγχος

είναι πιο δύσκολος και τα λάθη δεν εντοπίζονται εύκολα. Θα μπορούσαμε να χωρίσουμε τον λογικό έλεγχο σε δύο βήματα: α) επιλογή εισόδων που θα καλύπτουν όλες τις πιθανές περιπτώσεις, β) έλεγχος αποτελεσμάτων.

Ας δοκιμάσουμε να πραγματοποιήσουμε λογικό έλεγχο στο πρόγραμμα για τον υπολογισμό της δύναμης, όταν ο εκθέτης και η βάση είναι θετικοί αριθμοί.

- Επιλογή εισόδων
  - Επιλέγουμε εισόδους για μη ακραίες περιπτώσεις, δηλαδή για τη συνηθέστερη λειτουργία του προγράμματος, π.χ. b=3 e=4
  - Επιλέγουμε εισόδους για τις ακραίες περιπτώσεις. Στην περίπτωσή μας έχουμε τα εξής:
    - b = θετικός ακέραιος, e = 0
    - b = 0, e = θετικός ακέραιος
    - b = 0, e = 0
    - b=μεγάλος ακέραιος (π.χ. 10,000), e=μεγάλος ακέραιος (π.χ. 10,000)

Αφού επιλέξαμε τις εισόδους, πρέπει να ελέγξουμε ότι για τις συγκεκριμένες εισόδους το πρόγραμμα δίνει το σωστό αποτέλεσμα. Υπάρχουν διάφοροι τρόποι να γίνει αυτό, όπως:

(α) με έναν αναλυτικό πίνακα προκαταρκτικής εκτέλεσης που παρουσιάζει όλα τα βήματα, τις τιμές των μεταβλητών και των αποφάσεων του προγράμματος

(β) με έλεγχο του τελικού αποτελέσματος (αν γνωρίζουμε από πριν ποιο είναι αυτό)

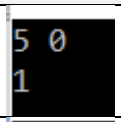
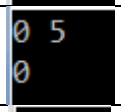
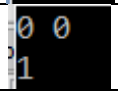
(γ) με την εκτύπωση ενδιάμεσων αποτελεσμάτων (η μέθοδος αυτή είναι πολύ χρήσιμη, όταν εντοπίσουμε κάποιο λάθος).

Θα παρουσιάσουμε και τους τρεις τρόπους (οι οποίοι δεν είναι οι μοναδικοί) πιο κάτω.

6.2 Προκαταρκτική εκτέλεση με αναλυτικό πίνακα. Είσοδος: b=3, e=4.

Μεταβλητές				Αποφάσεις		Παρουσίαση
b	**e**	**i**	**result**	**i<=e**	**T/F**	
3	4		1			81
		1	3	1<=4	T	
		2	9	2<=4	T	
		3	27	3<=4	T	
		4	81	4<=4	T	
		5		5<=4	F	

6.3 Έλεγχος τελικού αποτελέσματος.

b	e	result
5	0	
0	5	
0	0	

6.4 Εκτύπωση ενδιάμεσων αποτελεσμάτων

Αν δοκιμάσετε να υπολογίσετε τη δύναμη 10^{15}, θα πάρετε λανθασμένο αποτέλεσμα, όπως φαίνεται στην πιο κάτω εικόνα.

```
10 15
-1530494976
```

Το σωστό αποτέλεσμα είναι 1,000,000,000,000,000. Ο λόγος που εμφανίζεται το λάθος είναι επειδή γίνεται υπερχείλιση του αριθμού, δηλαδή το αποτέλεσμα δεν χωράει να αποθηκευτεί μέσα στο result που είναι τύπου `integer`. Για να δούμε πού έγινε η υπερχείλιση, εκτυπώνουμε το αποτέλεσμα μετά από κάθε πολλαπλασιασμό:

```
for (int i = 1; i <= e; i++){
      result = result * b;
      cout << "For i=" << i << " result=" << result << endl;
   }
```

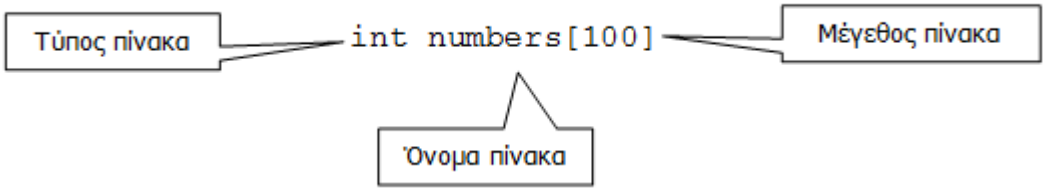
```
10 15
For i=1 result=10
For i=2 result=100
For i=3 result=1000
For i=4 result=10000
For i=5 result=100000
For i=6 result=1000000
For i=7 result=10000000
For i=8 result=100000000
For i=9 result=1000000000
For i=10 result=1410065408
For i=11 result=1215752192
For i=12 result=-727379968
For i=13 result=1316134912
For i=14 result=276447232
For i=15 result=-1530494976
```

Παρατηρούμε ότι για i>=10 δεν εμφανίζει το σωστό αποτέλεσμα λόγω υπερχείλισης. Άρα, το πρόγραμμα που φτιάξαμε έχει ένα όριο στο μέγεθος του αποτελέσματος.

7. Πίνακες

Οι πίνακες μάς δίνουν τη δυνατότητα να ομαδοποιήσουμε δεδομένα του ιδίου τύπου. Αναφορά στον πίνακα γίνεται μέσω ενός ονόματος (αναγνωριστικού), όπως στις μεταβλητές, ενώ για πρόσβαση στα δεδομένα ενός πίνακα χρησιμοποιείται το όνομα και ένας αριθμός (δείκτης - index), ο οποίος καθορίζει τη θέση του στοιχείου μέσα στον πίνακα.

Ένας πίνακας θα πρέπει να δηλωθεί πριν χρησιμοποιηθεί μέσα σε ένα πρόγραμμα. Για τον ορισμό ενός πίνακα παρέχουμε τρία στοιχεία. Το όνομα του πίνακα, τον τύπο δεδομένων των στοιχείων του και το πλήθος των θέσεων του πίνακα.



Αν γνωρίζουμε εκ των προτέρων τις τιμές που θα πάρουν τα στοιχεία του πίνακα, η απόδοση τιμών μπορεί να γίνει ως εξής:

```
int grades[4] = {16, 18, 15, 20};
```

Η δήλωση του μεγέθους μέσα στις αγκύλες δεν είναι απαραίτητη όταν αποδίδουμε αρχικές τιμές, αφού ο μεταγλωττιστής μπορεί να καταμετρήσει τις τιμές μέσα στα άγκιστρα. Αν δώσουμε το μέγεθος του πίνακα και το πλήθος των αρχικών τιμών μέσα στα άγκιστρα είναι μικρότερο, δεν θα υπάρχει πρόβλημα, καθώς στα στοιχεία του πίνακα που απομένουν θα αποδοθεί ως αρχική τιμή το 0. Αν όμως το πλήθος των αρχικών τιμών μέσα στα άγκιστρα είναι μεγαλύτερο από το μέγεθος που δηλώθηκε, τότε υπάρχει σφάλμα. Ας δούμε κάποια παραδείγματα:

```
int primes[] = {2,3,5,7,11};       // Δεν χρειάζεται να δηλώνουμε το μέγεθος
char characters[3] = {'a', 'b', 'c', 'd'};  // Σφάλμα! Περισσότερα στοιχεία
int a[5] = {12,5,4};                        // a → {12, 5, 4, 0, 0}
```

Παράδειγμα 1.12

Να δημιουργήσετε πρόγραμμα, το οποίο να γεμίζει έναν μονοδιάστατο πίνακα ακεραίων 100 θέσεων, τοποθετώντας την τιμή 0 στις περιττές θέσεις και την τιμή 1 στις ζυγές θέσεις. Στη συνέχεια, να εκτυπώνει όλα τα στοιχεία του πίνακα σε μία γραμμή.

```
#include<iostream>
#define N 100
using namespace std;
int A[N];          // Η δήλωση ενός πίνακα ακεραίων εκτός της κύριας
                   // συνάρτησης (main), αρχικοποιεί όλα τα στοιχεία
int main(){        // του πίνακα με την τιμή 0
    for(int i=0; i<N; i+=2)
        A[i] = 1;
    for(int i=0; i<N; i++)
        cout << A[i] << " ";
return 0;
}
```

(Code: C7_1_example12.cpp)

Παράδειγμα 1.13

Να δημιουργήσετε πρόγραμμα, το οποίο να διαβάζει 15 ονόματα και να τα αποθηκεύει σε έναν πίνακα. Στη συνέχεια, να ζητά από τον χρήστη να δώσει έναν αριθμό από το 0 μέχρι το 14 και να επιστρέφει το όνομα που βρίσκεται στη θέση αυτή.

```
#include<iostream>
#include<string>
#define N 15
using namespace std;

int main(){
    string names[N];
```

```
    int pos;
    for(int i=0; i<N; i++){
        cout << "Name: ";
        cin >> names[i];
    }
    cout << "Position:";
    cin >> pos;
    cout << names[pos];
return 0;
}
```

(Code: C7_1_example13.cpp)

Παράδειγμα 1.14

Σε έναν διαγωνισμό μαθηματικών λαμβάνουν μέρος 10 μαθητές. Τα ονόματα και οι βαθμοί αποθηκεύονται στους πίνακες onomata και vathmoi, αντίστοιχα. Να δημιουργήσετε πρόγραμμα, το οποίο να τυπώνει τα ονόματα των μαθητών που πέτυχαν βαθμολογία μεγαλύτερη του μέσου όρου των βαθμών όλων των μαθητών.

```
#include<iostream>
#include<string>
#define N 10
using namespace std;

int main(){
    string names[N];
    int grades[N], sum=0, i;
    float mo;
    for(i=0; i<N; i++){
        cin >> names[i] >> grades[i];
        sum += grades[i];
    }
    mo = float(sum)/N;
    for(i=0; i<N; i++)
        if(grades[i]>mo)
            cout << names[i] << endl;
return 0;
}
```

(Code: C7_1_example14.cpp)

Ασκήσεις Κεφαλαίου - http://www.hackerrank.com/g71

Άσκηση 1.1

Ένα γυμναστήριο το οποίο χρεώνει συνδρομή €50 ανά μήνα, προσφέρει έκπτωση 25% στο κόστος της συνδρομής, σε πελάτες που επιθυμούν συνδρομή διάρκειας μεγαλύτερης ή ίσης των έξι μηνών ή σε πελάτες με ηλικία άνω των 60 ετών. Να σχεδιάσετε λογικό διάγραμμα, το οποίο να δέχεται τη διάρκεια της συνδρομής και την ηλικία του πελάτη και, στη συνέχεια, αν ο πελάτης δικαιούται έκπτωση, να υπολογίζει και να εμφανίζει το ποσό της συνδρομής που πρέπει να πληρώσει, διαφορετικά να εμφανίζει το μήνυμα «Καμία έκπτωση» και το ποσό της συνδρομής.

Άσκηση 1.2

Οι μεταβλητές πραγματικού τύπου x, y και z, έχουν τις ακόλουθες τιμές:

```
float x = 5.5, y = 6.0, z = 7.5;
```

Να γράψετε το αποτέλεσμα των πιο κάτω εντολών της C++:

```
a. cout << ((1>abs(x-z)) && (y>x));
b. cout << ((round(x)>=y) || (y!=6) || !(x>z) && (trunc(z)<y));
```

Άσκηση 1.3

Οι μεταβλητές ακέραιου τύπου x, y και z, έχουν τις ακόλουθες τιμές:

```
int x = 2, y = -4, z = 8;
```

Να γράψετε την τιμή που θα πάρει η μεταβλητή check, λογικού τύπου (boolean), στις πιο κάτω εντολές της C++:

```
a. check = (y<x) && (z<x) || (z%x==0);
b. check = !((z-x)==0) && (z==4*x) || (y>z);
c. check = (2*z%x==z) && !(2*y+z+2%2==0);
```

Άσκηση 1.4

Σας δίνονται τρεις ακέραιοι αριθμοί a, b, c. Να γράψετε τη λογική έκφραση για κάθε μία από τις παρακάτω περιπτώσεις:

(α) Είναι ο αριθμός a ο μεγαλύτερος αριθμός από τους τρεις;

(β) Είναι ο αριθμός b μέσα στο διάστημα a μέχρι c, συμπεριλαμβανομένων;

(γ) Είναι κάποιος από τους τρεις αριθμούς άρτιος (ζυγός);

Άσκηση 1.5

Αν X = 10, Y = 5 και Z = - 2, να γράψετε το αποτέλεσμα των πιο κάτω λογικών εκφράσεων:

(α) (X<Y) && (Z>Y)	(β) (X==Y) || (X!=Z)
(γ) (Y<0) || (Z>0)	(δ) (X>Y) || (X>Z) && (Y>X)
(ε) !(X==(2*Y)) && !(Z==2)	(στ) !(X!=Y) || !(Y==Z)
(ζ) (X<0) && (Y>0) || (Z>0)	(η) (Y==(X/2)) && (Z>0)

Άσκηση 1.6

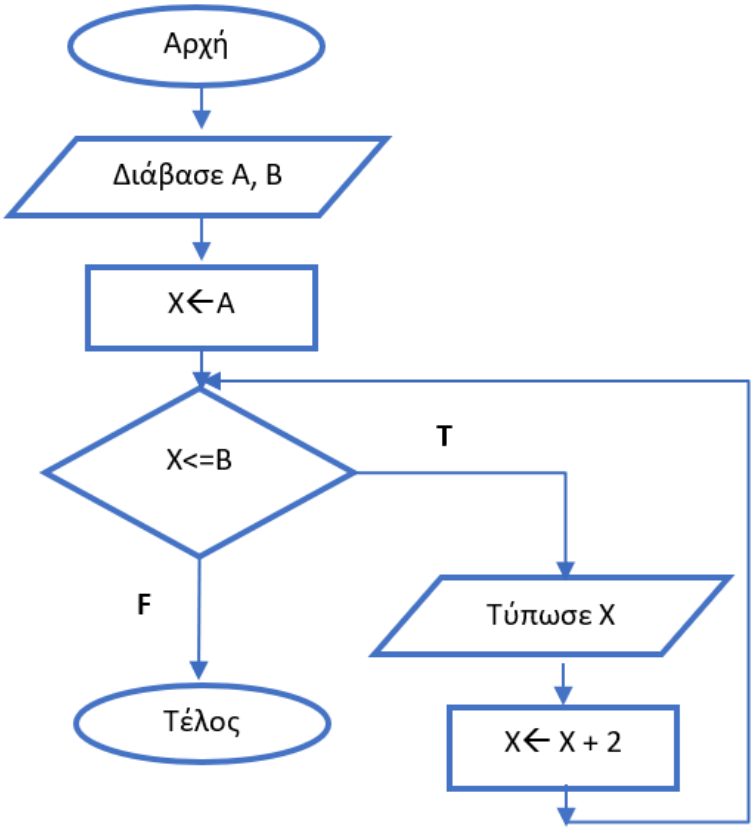
Να σχεδιάσετε λογικό διάγραμμα, το οποίο να δέχεται έναν ακέραιο Ν και να εμφανίζει τον επόμενο περιττό και τον προηγούμενο ζυγό αριθμό του Ν. Π.χ. αν δοθεί ο αριθμός 521, θα εμφανιστούν οι αριθμοί 523 και 520.

Άσκηση 1.7

Να σχεδιάσετε λογικό διάγραμμα, το οποίο να δέχεται έναν ακέραιο Ν (1<=N<=18) και να επιστρέφει το πλήθος των διψήφιων ακεραίων, των οποίων τα ψηφία έχουν άθροισμα Ν. Για παράδειγμα, για Ν = 11, το πλήθος θα είναι 8 (29, 38, 47, 56, 65, 74, 83, 92).

Άσκηση 1.8

Να μετατρέψετε το πιο κάτω λογικό διάγραμμα σε πρόγραμμα στη C++. Με τη χρήση προκαταρκτικής εκτέλεσης, να παρουσιάσετε τα αποτελέσματα για Α = 20 και Β = 30.



Άσκηση 1.9

Να σχεδιάσετε λογικό διάγραμμα, το οποίο να δέχεται τα ονόματα και τις βαθμολογίες 25 μαθητών και να υπολογίζει:

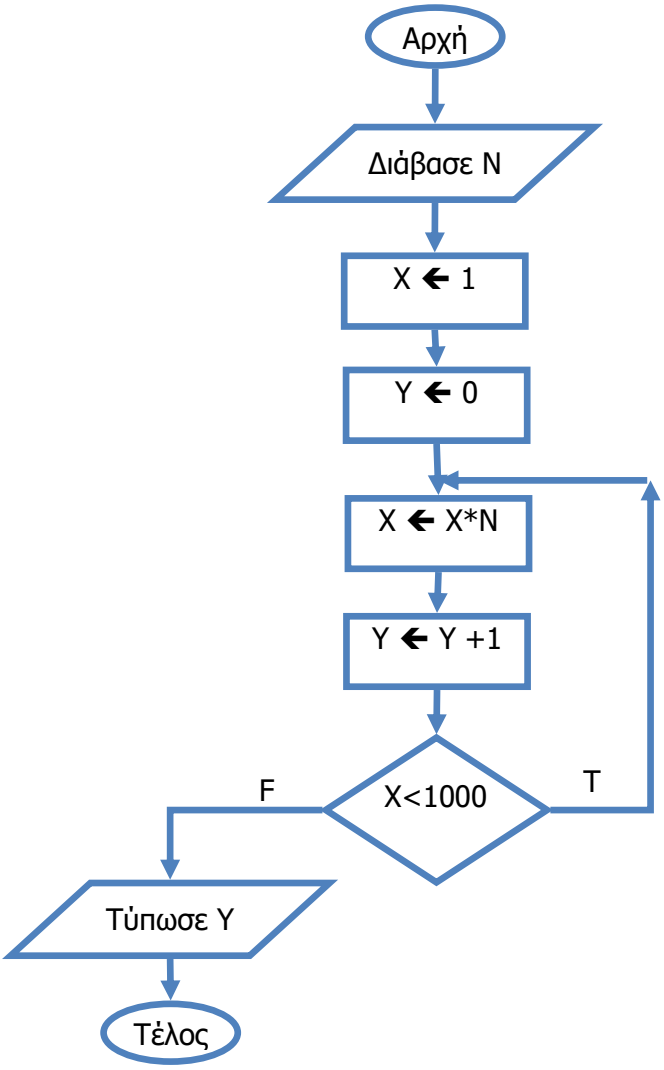
(α) Το πλήθος των μαθητών με βαθμολογία μεγαλύτερη από το 49

(β) Το όνομα του μαθητή με τη μεγαλύτερη βαθμολογία.

Οι βαθμολογίες των μαθητών θα είναι ακέραιοι αριθμοί στο διάστημα [1…100]. Να θεωρήσετε ότι δεν θα υπάρχουν δύο μαθητές με την ίδια βαθμολογία.

Άσκηση 1.10

Να μετατρέψετε το πιο κάτω λογικό διάγραμμα σε πρόγραμμα στη C++ και να γράψετε τον πίνακα προκαταρκτικής εκτέλεσης για N = 5.



Άσκηση 1.11

Με τη χρήση προκαταρκτικής εκτέλεσης, να παρουσιάσετε τα αποτελέσματα του πιο κάτω προγράμματος.

```
#include<iostream>
using namespace std;
int main(){
    int p = 1;
    for(int i=1; i<=5; i++)
        p *= i;
    cout << p;
return 0;
}
```

Άσκηση 1.12

Με τη χρήση προκαταρκτικής εκτέλεσης, να παρουσιάσετε τα αποτελέσματα του πιο κάτω προγράμματος, χρησιμοποιώντας τους αριθμούς 3, 21, 40 και 7.

```
#include<iostream>
using namespace std;
#define N 4
int main(){
    int i, num, counter = 0;
    cout << "Δώσε " << N << " αριθμούς:"<< endl;
    for(i=1; i<=N; i++){
        cin >> num;
        if(num%2 == 0)
            counter++;
    }
    cout<<"Πλήθος άρτιων: " << counter;
return 0;
}
```

Άσκηση 1.13

Με τη χρήση προκαταρκτικής εκτέλεσης, να παρουσιάσετε τα αποτελέσματα του πιο κάτω προγράμματος.

```
#include<iostream>
using namespace std;
int main(){
    int arr[6];
    int i;
    for(i=0; i<6; i++){
        arr[i] = (i + 1) % 3;
        cout << arr[i] << " ";
    }
return 0;
}
```

Άσκηση 1.14

Η απόσταση μεταξύ δύο σημείων δίνεται από τον τύπο

$$d = \sqrt{(X2 - X1)^2 + (Y2 - Y1)^2}$$

όπου X1, Y1 και X2, Y2 οι συντεταγμένες 2 σημείων. Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται τις συντεταγμένες των 2 σημείων και να υπολογίζει την απόσταση d.

Το αποτέλεσμα να εμφανίζεται με ακρίβεια 2 δεκαδικών ψηφιών.

Σημείωση: Να γίνει χρήση των έτοιμων συναρτήσεων `sqrt` και `pow` της βιβλιοθήκης `cmath`.

Παράδειγμα εισόδου

```
1 1 0 0
```

Παράδειγμα εξόδου

```
1.41
```

Άσκηση 1.15

Το πιο κάτω πρόγραμμα περιγράφει τον υπολογισμό του μέσου όρου των ψηφίων των μονάδων δύο ακέραιων αριθμών. Στο πρόγραμμα υπάρχουν 4 συντακτικά / λογικά λάθη. Να τα εντοπίσετε και να τα διορθώσετε.

```
#include<iostream>
using namespace std;
int main(){
    int a, b, m1, m2;
    float mesos;
    cin >> a,b;
    m1 = a%10;
    m2 = a%10;
    mesos = (m1+m2)/2;
    cout >> mesos;
return 0;
}
```

Παράδειγμα εισόδου

```
123
56
```

Παράδειγμα εξόδου

```
4.5
```

Άσκηση 1.16

Στο εστιατόριο της Αριάδνης, κάθε πελάτης πληρώνει επιπλέον 5% της αξίας του λογαριασμού, για φιλανθρωπικούς σκοπούς. Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται το ποσό του λογαριασμού και να υπολογίζει το ποσό που θα πληρωθεί για φιλανθρωπικούς σκοπούς.

Σημείωση: Το ποσοστό 5% να δηλωθεί ως σταθερά.

Παράδειγμα εισόδου

```
22
```

Παράδειγμα εξόδου

```
1.10
```

Άσκηση 1.17

Σας δίνονται 2 ακέραιοι θετικοί αριθμοί Α και Β ($0<A,B<=2^{40}$). Να δημιουργήσετε πρόγραμμα, το οποίο να υπολογίζει το άθροισμα των ψηφίων των δεκάδων τους.

Παράδειγμα εισόδου

```
12348889 456778
```

Παράδειγμα εξόδου

```
15
```

Άσκηση 1.18

Ο Θησέας παίζει με τα κουβαδάκια του στην άμμο. Έχει καταφέρει να μαζέψει Ν κιλά άμμου. Θέλει να φτιάξει πύργους με την άμμο που έχει μαζέψει και έχει στη διάθεσή του κουβαδάκια των 5 κιλών, των 3 κιλών και του ενός κιλού. Ο Θησέας θέλει να χρησιμοποιήσει όσο το δυνατό λιγότερους κουβάδες, για να μαζέψει όλη την άμμο. Να δημιουργήσετε πρόγραμμα, το οποίο να υπολογίζει το πλήθος των κουβάδων που θα χρειαστεί.

Παράδειγμα εισόδου

```
29
```

Παράδειγμα εξόδου

```
7
```

Άσκηση 1.19

Η βουλή των αντιπροσώπων έχει ψηφίσει τον ακόλουθο νόμο για στέρηση της άδειας κυκλοφορίας παρανομούντων οδηγών. Η στέρηση άδειας θα ισχύει για έναν χρόνο, είτε όταν ο οδηγός ξεπεράσει τους 12 βαθμούς ποινής είτε αν υπάρχει καταγγελία για οδήγηση υπό την επήρεια αλκοόλ. Σε περίπτωση που ο οδηγός ξεπεράσει τους 12 βαθμούς και υπάρχει ταυτόχρονα καταγγελία για οδήγηση υπό την επήρεια αλκοόλ, η στέρηση άδειας θα ισχύει για δύο χρόνια.

Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται τους βαθμούς ποινής (`integer`), καθώς και τον έλεγχο καταγγελίας (`boolean`) για οδήγηση υπό την επήρεια αλκοόλ (δίνεται ο αριθμός 1 - true στην περίπτωση που υπάρχει καταγγελία και ο αριθμός 0 - false στην περίπτωση που δεν υπάρχει). Το πρόγραμμα να εμφανίζει ένα από τα ακόλουθα μηνύματα: «Νομοταγής οδηγός», «Ένας χρόνος στέρηση άδειας», «Δύο χρόνια στέρηση άδειας».

Άσκηση 1.20

Ο Θεόδωρος και ο Ραφαήλ είναι τερματοφύλακες στην ομάδα LAG FC. Ο προπονητής δεν μπορεί να αποφασίσει ποιος από τους δύο θα ξεκινά ως βασικός, αφού είναι και οι 2 πολύ καλοί τερματοφύλακες. Τελικά, αποφάσισε ότι θα παίζει βασικός αυτός που έχει δεχτεί τα λιγότερα τέρματα ανά λεπτά συμμετοχής. Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται τα τέρματα και τα λεπτά συμμετοχής για τον καθένα (πρώτα του Θεόδωρου και μετά του Ραφαήλ) και να τυπώνει το γράμμα «Τ», αν πρέπει να ξεκινήσει βασικός ο Θεόδωρος, ή το γράμμα «R», αν θα ξεκινά βασικός ο Ραφαήλ.

Παράδειγμα εισόδου

```
5 180
7 160
```

Παράδειγμα εξόδου

```
T
```

Άσκηση 1.21

Η Γεωργία εργάζεται σε κέντρο ελέγχου παραγωγής σοκολάτας. Το μηχάνημα συσκευασίας ενεργοποιείται από 4 σήματα ελέγχου (A, B, C, D). Το μηχάνημα μπαίνει σε λειτουργία μόνο για συγκεκριμένους συνδυασμούς των σημάτων ελέγχου. Οι συνδυασμοί αυτοί φαίνονται στον πιο κάτω χάρτη Karnaugh.

AB \ CD	00	01	11	10
00	1	0	0	1
01	0	1	1	0
11	0	1	1	0
10	1	0	0	1

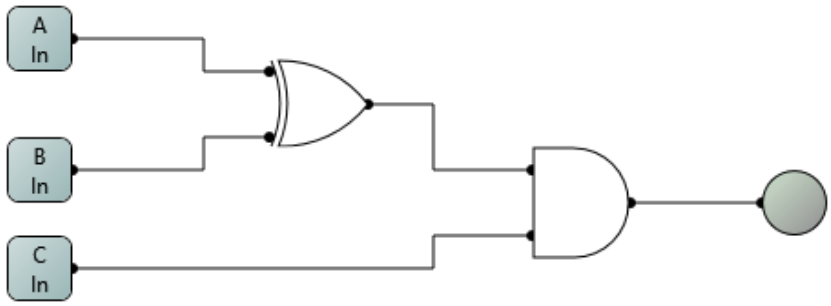
Η Γεωργία παρατήρησε ότι μπορεί να παρακολουθεί το μηχάνημα συσκευασίας, εξετάζοντας μόνο 2 από τα σήματα ελέγχου. Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται ΜΟΝΟ τις 2 εισόδους που είναι απαραίτητες για τον έλεγχο της μηχανής και να τυπώνει τη λέξη «ON», σε περίπτωση που η μηχανή είναι σε λειτουργία, διαφορετικά, να τυπώνει τη λέξη «OFF».

Σημείωση: Να γίνει ομαδοποίηση των ελαχιστόρων και να απλοποιηθεί η συνάρτηση με τη βοήθεια του χάρτη Karnaugh.

Άσκηση 1.22

Η Στυλιανή είναι υπεύθυνη του κέντρου έρευνας και διάσωσης. Πρόσφατα έχουν εγκατασταθεί 2 μανόμετρα για την μέτρηση της ατμοσφαιρικής πίεσης και ένα ηλεκτρονικό θερμόμετρο για τη μέτρηση της θερμοκρασίας. Και τα 3 όργανα στέλνουν στην κεντρική κονσόλα ένα σήμα που δηλώνει τη λειτουργία τους. Απουσία του σήματος σημαίνει ότι κάποιο όργανο δεν βρίσκεται σε λειτουργία. Τα μανόμετρα λειτουργούν ως εξής: πρέπει να υπάρχει πάντα ένα μανόμετρο σε λειτουργία, ποτέ όμως και τα 2 ταυτόχρονα. Μέσα στο κέντρο ανάβει μία πράσινη λυχνία, όταν το σύστημα των τριών οργάνων δουλεύει σωστά. Το κύκλωμα που ανάβει τη λυχνία είναι το ακόλουθο: όπου A και B είναι τα σήματα από τα μανόμετρα και C το σήμα από το θερμόμετρο. Ο προϊστάμενος της Στυλιανής θέλει να βλέπει αυτή την ένδειξη στον υπολογιστή του.

Βοηθήστε τη Στυλιανή να δημιουργήσει πρόγραμμα, το οποίο να υλοποιεί το πιο πάνω κύκλωμα και να εμφανίζει το μήνυμα «OK», στην περίπτωση που το σύστημα δουλεύει σωστά, διαφορετικά, να τυπώνει το μήνυμα «manometer», όταν υπάρχει πρόβλημα στα μανόμετρα, και «thermometer», όταν υπάρχει πρόβλημα στο θερμόμετρο. Σε περίπτωση που υπάρχει πρόβλημα και στα 2, να τυπώνει το μήνυμα «manometer thermometer».



Παράδειγμα εισόδου 1

```
1 0 1
```

Παράδειγμα εξόδου 1

```
OK
```

Παράδειγμα εισόδου 2

```
0 0 1
```

Παράδειγμα εξόδου 2

```
manometer
```

Παράδειγμα εισόδου 3

```
1 1 0
```

Παράδειγμα εξόδου 3

```
manometer thermometer
```

Άσκηση 1.23

Μία βιβλιοθήκη έχει βιβλία που είναι διαθέσιμα για δανεισμό για μία εβδομάδα, για έναν μήνα και βιβλία που δεν δανείζονται. Το κάθε βιβλίο έχει έναν αριθμό μητρώου που είναι μοναδικός. Ο αριθμός μητρώου είναι ακέραιος. Το γινόμενο των δύο τελευταίων ψηφίων του αριθμού μητρώου καθορίζει τον τύπο δανεισμού του, με βάση τον πιο κάτω πίνακα.

Γινόμενο 2 τελευταίων ψηφίων αριθμού μητρώου	Είδος δανεισμού
< 10	week
>=10	month
0	never

Να σχεδιάσετε λογικό διάγραμμα και να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται τον αριθμό μητρώου και να τυπώνει το είδος δανεισμού.

Παράδειγμα εισόδου

```
123
```

Παράδειγμα εξόδου

```
week
```

Άσκηση 1.24

Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται 2 δεκαδικούς αριθμούς και έναν μαθηματικό τελεστή (+, -, *, /) και να τυπώνει το αποτέλεσμα της πράξης με ακρίβεια 2

δεκαδικών ψηφιών. Σε περίπτωση που η μαθηματική πράξη είναι αδύνατη, να τυπώνεται το μήνυμα «Error».

Παράδειγμα εισόδου

```
1.1 2.1 +
```

Παράδειγμα εξόδου

```
3.20
```

Άσκηση 1.25

Ο Δημήτρης είναι πρόεδρος της εταιρείας «Happy Penguin». Η εταιρεία αντιμετωπίζει σοβαρά οικονομικά προβλήματα και ο Δημήτρης αποφάσισε να αποκόπτονται από τον μισθό των υπαλλήλων 2 ποσά. Το πρώτο αφορά στην ενίσχυση του αποθεματικού και το δεύτερο αφορά στην αποπληρωμή δανείου. Οι αποκοπές γίνονται με βάση τον πιο κάτω πίνακα. Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται τον μισθό ενός υπαλλήλου, να τυπώνει τα ποσά των 2 κρατήσεων και τον καθαρό μισθό του υπαλλήλου.

Να προσθέσετε, πριν από τις βιβλιοθήκες, ως σχόλια (comments) στο πρόγραμμα, το όνομά σας και το τμήμα σας ως εξής:

```
// Name:  John Doe
// Class: C4
```

Μισθός	Κράτηση 1	Κράτηση 2
<=800	2%	2.5%
801-1200	2.5%	3%
>1200	3%	4%

Παράδειγμα εισόδου

```
1000.00
```

Παράδειγμα εξόδου

```
25.00
30.00
945.00
```

Άσκηση 1.26

Να μετατρέψετε την πιο κάτω ένθετη δομή διακλάδωσης σε περιπτωσιακή δομή (switch).

```
#include<iostream>
using namespace std;

int main(){
    int epid;
    cin >> epid;
    if (epid>=1 && epid<=3)
       cout << "ΜΕΤΡΙΑ" << endl;
    else if (epid>=4 && epid<=7)
       cout << "ΚΑΛΗ" << endl;
```

```
    else if (epid>=8 && epid<=10)
       cout << "ΑΡΙΣΤΗ" << endl;
    else
       cout << "Λανθασμένη τιμή" << endl;
  return 0;
}
```

Άσκηση 1.27

Να μετατρέψετε την πιο κάτω περιπτωσιακή δομή (switch) σε ένθετη δομή διακλάδωσης.

```
#include<iostream>
#include<cmath>
using namespace std;

int main(){
      int num;
      cin >> num;
      switch (num){
          case 2:
          case 5:    cout << pow(num,2) << endl;
                     break;
          case 8:
          case 10:   cout << pow(num,3) << endl;
                     break;
          case 11:
          case 14:   cout << num << endl;
                     break;
          default:   cout << "Λανθασμένη τιμή" << endl;
     }
  return 0;
}
```

Άσκηση 1.28

Το παιχνίδι που σκαρφίστηκαν ο Μάκης και ο Τάκης έχει ως εξής: Κάθε ένας από τους δύο παίκτες ρίχνει τρία ζάρια, μόνο μία φορά.

Μπορούν να σκοράρουν με έναν από τους εξής τέσσερις διαφορετικούς τρόπους:

(α) Αν έρθουν και τα τρία ζάρια 6, κερδίζουν 50 πόντους

(β) Αν έρθουν τρία ζάρια τα ίδια, κερδίζουν 30 πόντους

(γ) Αν έρθουν δύο ζάρια τα ίδια, κερδίζουν 20 πόντους

(δ) Αν δεν έρθει κανένα από τα πιο πάνω, κερδίζουν το άθροισμα των τριών ζαριών.

Αν σας δοθούν οι ζαριές και των δύο παικτών, πρώτα του Μάκη και μετά του Τάκη, να υπολογίσετε ποια θα είναι η βαθμολογία του νικητή. Νικητής θεωρείται αυτός που θα πετύχει την πιο ψηλή βαθμολογία.

Παράδειγμα εισόδου

```
1 1 1
6 6 6
```

Παράδειγμα εξόδου

```
50
```

Επεξήγηση: Ο Μάκης κερδίζει 30 πόντους, αφού έχει φέρει τρία ίδια ζάρια, αλλά ο Τάκης έχει φέρει τρία εξάρια, άρα με 50 πόντους, είναι ο νικητής.

Άσκηση 1.29

Να ξαναγράψετε το πιο κάτω πρόγραμμα, χρησιμοποιώντας τις δομές while και for (Να θεωρήσετε ότι θα δοθεί x>0).

```
#include<iostream>
using namespace std;
int main(){
    int x, sum = 0, i = 0, num;
    cin >> x;
    do {
         cin >> num;
         sum += num;
         i++;
     } while(i < x);
    cout << sum;
return 0;
}
```

Άσκηση 1.30

Να ξαναγράψετε το πιο κάτω πρόγραμμα, χρησιμοποιώντας τις δομές while και do (Να θεωρήσετε ότι θα δοθεί x>0).

```
#include<iostream>
using namespace std;
int main(){
    int x, sum = 0;
    cin >> x;
    for(int i=0; i<=x; i+=2)
        sum += i;
    cout << sum;
return 0;
}
```

Άσκηση 1.31

Το παρακάτω πρόγραμμα δημιουργήθηκε για να υπολογίζει τον μέγιστο κοινό διαιρέτη 2 ακέραιων αριθμών. Στο πρόγραμμα υπάρχουν 4 λογικά/συντακτικά λάθη. Να τα διορθώσετε και να γράψετε ξανά το πρόγραμμα.

```
#include<iostream>
using namespace std;
int main(){
    int a, b;
    cin >> a, b;
    mkd = b;
    while(mkd%b!=0 || a%mkd!=0)
        mkd++;
    cout << mkd;
return 0;
}
```

Άσκηση 1.32

Ο δήμος αποφάσισε να κατασκευάσει νέο αμφιθέατρο. Στην πρώτη σειρά θα τοποθετηθούν 50 καθίσματα, ενώ σε κάθε επόμενη σειρά θα προστίθενται 6 καθίσματα. Το κόστος κάθε καθίσματος είναι 40 ευρώ, ενώ ο προϋπολογισμός για τα καθίσματα είναι Ν (2,000<N<=50,000). Να δημιουργήσετε πρόγραμμα και να σχεδιάσετε λογικό διάγραμμα, το οποίο να δέχεται έναν ακέραιο Ν και να υπολογίζει:

(α) Το πλήθος των σειρών του αμφιθεάτρου

(β) Τη συνολική χωρητικότητα του αμφιθεάτρου

(γ) Το ποσό που περίσσεψε (αν υπάρχει) από την κατασκευή.

Σημείωση: όταν το κόστος κατασκευής μίας σειράς υπερβαίνει το διαθέσιμο ποσό που υπολείπεται, η σειρά αυτή δεν θα κατασκευάζεται.

Παράδειγμα εισόδου

```
20000
```

Παράδειγμα εξόδου

```
7
476
960
```

Άσκηση 1.33

Ένα μικρό χωρίο τροφοδοτείται με νερό από μία δεξαμενή, χωρητικότητας 400 κυβικών τόνων. Ο μηχανισμός τροφοδοσίας λειτουργεί ως εξής: Από τα μεσάνυκτα μέχρι τις 7 το πρωί γίνεται μόνο εισροή (είσοδος) νερού στη δεξαμενή, ενώ την υπόλοιπη μέρα γίνεται μόνο εκροή (έξοδος). Σε περίπτωση που η εισροή νερού ξεπεράσει τη διαθέσιμη χωρητικότητα της δεξαμενής, τότε το επιπλέον νερό διανέμεται με ένα ειδικό σύστημα στα χωράφια των κατοίκων του χωριού.

Αν η δεξαμενή στην αρχή είναι άδεια, να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται την εισροή και εκροή νερού σε κυβικούς τόνους για 20 μέρες και να τυπώνει:

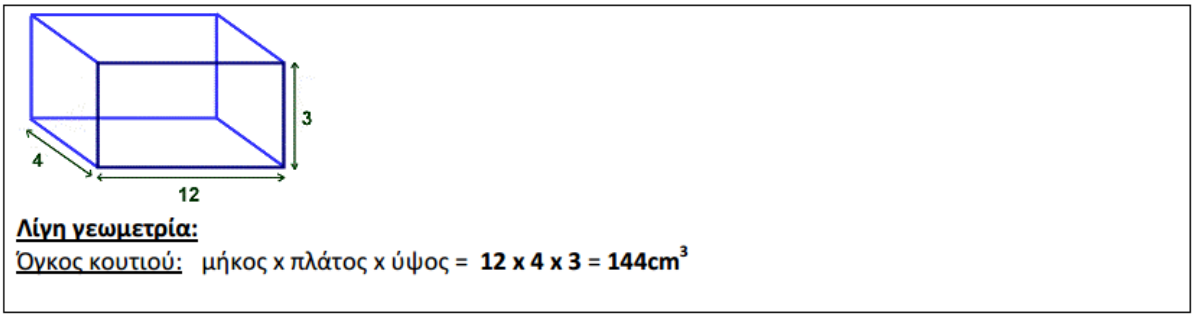
(α) Τον αριθμό των κυβικών τόνων που υπάρχουν μέσα στη δεξαμενή μετά το τέλος της 20ής ημέρας

(β) Τον αριθμό των κυβικών τόνων νερού που κατέληξαν στα χωράφια λόγω υπερχείλισης.

Σημείωση: Δεν υπάρχει περίπτωση να αδειάσει ποτέ τελείως η δεξαμενή.

Άσκηση 1.34

Αν γνωρίζουμε τον όγκο V (1<=V<=10^7) ενός κουτιού, να δημιουργήσετε πρόγραμμα, το οποίο να υπολογίζει το πλήθος των πιθανών διαστάσεών του.



Π.χ. Αν ο όγκος του κουτιού είναι 10 cm^3, τότε οι πιθανές διαστάσεις του κουτιού μπορεί να είναι:

`1 x 1 x 10`

`1 x 2 x 5`

Δηλαδή 2 πιθανοί τρόποι.

Σημείωση: Οι διαστάσεις `10 x 1 x 1` και `5 x 2 x 1`, όπου εναλλάσσονται οι τιμές, δεν πρέπει να υπολογίζονται.

Παράδειγμα εισόδου

```
10
```

Παράδειγμα εξόδου

```
2
```

Άσκηση 1.35

Η Ευρωπαϊκή Υπηρεσία Πολιτικής Αεροπορίας θέλει να συλλέξει πληροφορίες για όλα τα αεροσκάφη που είναι εγγεγραμμένα στους καταλόγους των κρατών μελών της Ευρωπαϊκής Ένωσης. Για τον λόγο αυτό σάς ζητά να δημιουργήσετε πρόγραμμα, το οποίο να:

(α) ζητά τον κωδικό εγγραφής, το ποσό για το οποίο είναι ασφαλισμένο το αεροπλάνο και τη χωρητικότητά του σε επιβάτες. Ο κωδικός εγγραφής είναι μία συμβολοσειρά μεγέθους πέντε χαρακτήρων, εκ των οποίων οι δύο πρώτοι δηλώνουν τη χώρα προέλευσης. Τα αεροπλάνα που είναι εγγεγραμμένα στην Κύπρο ξεκινούν με τους χαρακτήρες 5B (π.χ. «5BCIC»). Η επανάληψη σταματά, όταν δοθεί η συμβολοσειρά «AAAAA»

(β) τυπώνει τον συνολικό αριθμό επιβατών που μπορούν να μεταφέρουν όλα τα εγγεγραμμένα αεροπλάνα

(γ) τυπώνει τους κωδικούς του 1ου και του 2ου αεροπλάνου με το μεγαλύτερο ποσό ασφάλισης

(δ) τυπώνει τον μέσο όρο του ποσού ασφάλισης όλων των αεροπλάνων

(ε) τυπώνει το πλήθος των αεροσκαφών που είναι εγγεγραμμένα στην Κύπρο.

Σημείωση: Θα δοθούν στοιχεία για τουλάχιστον 2 αεροπλάνα.

Άσκηση 1.36

Να δημιουργήσετε πρόγραμμα, το οποίο να:

(α) ζητά από τον χρήστη το πλήθος των μαθητών ενός σχολείου και, ακολούθως, να ζητά για κάθε μαθητή: όνομα, φύλο (Α για αγόρι και Κ για κορίτσι), αριθμό δικαιολογημένων απουσιών, αριθμό αδικαιολόγητων απουσιών και αριθμό απουσιών στο ΔΔΚ. Η επανάληψη να τερματίζει είτε όταν δοθούν τα στοιχεία για όλους τους μαθητές του σχολείου είτε όταν δοθεί ως όνομα μαθητή η λέξη «end». Να θεωρήσετε ότι όλα τα στοιχεία, εκτός από τον κωδικό φύλου, δίνονται σωστά και δεν χρειάζεται κανένας έλεγχος. Στην περίπτωση του κωδικού φύλου, πρέπει να γίνεται έλεγχος αν δόθηκε σωστά (Α ή Κ). Αν δοθεί λάθος, να ενημερώνεται σχετικά ο χρήστης και το πρόγραμμα να του ζητά να δώσει ξανά τον κωδικό φύλου

(β) εμφανίζει το όνομα του κάθε μαθητή και το σύνολο των απουσιών του (δικαιολογημένες, αδικαιολόγητες και ΔΔΚ)

(γ) βρίσκει και να τυπώνει το όνομα του μαθητή που έχει τις περισσότερες απουσίες στο ΔΔΚ. Σε περίπτωση ύπαρξης δύο ή περισσότερων τέτοιων μαθητών, να τυπώνει τον τελευταίο από αυτούς

(δ) υπολογίζει και να εμφανίζει ξεχωριστά τον αριθμό των αγοριών και τον αριθμό των κοριτσιών που έχουν από 40 μέχρι 50 (συμπεριλαμβανομένων) αδικαιολόγητες απουσίες και κινδυνεύουν να μείνουν στάσιμοι

(ε) υπολογίζει και να τυπώνει τον αριθμό των μαθητών που έμειναν στάσιμοι. Στάσιμος μένει ο μαθητής με περισσότερες από 50 αδικαιολόγητες απουσίες ή περισσότερες από 160 απουσίες στο σύνολο (δικαιολογημένες και αδικαιολόγητες)

(στ) υπολογίζει και να τυπώνει τον μέσο όρο των αδικαιολόγητων απουσιών των μαθητών που καταχωρίσθηκαν.

Το πρόγραμμα πρέπει να εμφανίζει στην οθόνη τα κατάλληλα μηνύματα για την εισαγωγή των δεδομένων και την εξαγωγή των αποτελεσμάτων.

(Παγκύπριες Εξετάσεις 2015)

Άσκηση 1.37

Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται 10 ακέραιους αριθμούς και να τους αποθηκεύει στον πίνακα Α. Στη συνέχεια, να τυπώνει το πλήθος των αριθμών του πίνακα που είναι μεγαλύτεροι από τον τελευταίο αριθμό που δόθηκε.

Άσκηση 1.38

Η Μαρία είναι η πρόεδρος της εταιρείας παρασκευής μελιού «Z Honey». Η εταιρεία μαζεύει το μέλι από μία κυψέλη, μόνο αν η προηγούμενη στη σειρά κυψέλη δεν έχει καθόλου μέλι. Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται το πλήθος των κυψελών (N) και την ποσότητα μελιού που έχει η κάθε κυψέλη και να τυπώνει τη συνολική ποσότητα μελιού που θα μαζέψει η εταιρεία.

Παράδειγμα εισόδου

```
7
0 3 1 0 4 0 2
```

Παράδειγμα εξόδου

```
9
```

Άσκηση 1.39

Σε έναν πίνακα 30 θέσεων αποθηκεύεται σε κάθε θέση ένας από τους χαρακτήρες A, B και C. Να δημιουργήσετε πρόγραμμα, το οποίο να:

(α) διαβάζει 30 χαρακτήρες και να τους αποθηκεύει στον πίνακα xar. Να γίνεται έλεγχος αν δίνονται σωστά οι χαρακτήρες

(β) βρίσκει και να τυπώνει το πλήθος των εμφανίσεων του χαρακτήρα B μέσα στον πίνακα

(γ) αντικαθιστά τον χαρακτήρα C με τον χαρακτήρα D και να τυπώνει τον πίνακα (το ένα στοιχείο κάτω από το άλλο).

Άσκηση 1.40

Να δημιουργήσετε πρόγραμμα, το οποίο να:

(α) δέχεται 30 ακέραιους αριθμούς (>99) και να τους αποθηκεύει μέσα στον πίνακα N. Να γίνεται έλεγχος αν δίνονται σωστά οι αριθμοί

(β) ελέγχει για κάθε αριθμό αν είναι αριθμός ariks. Αριθμός ariks ορίζεται ο αριθμός που το άθροισμα του πιο σημαντικού (αριστερότερου) και του λιγότερου σημαντικού ψηφίου (δεξιότερου) ισούται με το άθροισμα των υπόλοιπων ψηφίων (π.χ. 121, 1335). Αν είναι αριθμός ariks, τότε αποθηκεύεται στον παράλληλο πίνακα P η τιμή `true`, διαφορετικά αποθηκεύεται η τιμή `false`

(γ) τυπώνει τους 2 πίνακες, τον έναν δίπλα από τον άλλο.

Άσκηση 1.41

Να δημιουργήσετε πρόγραμμα, το οποίο να:

(α) δέχεται τα ονόματα και τη φωτεινότητα 100 αστεριών. Η φωτεινότητα είναι ένας δεκαδικός αριθμός μέσα στο διάστημα, [-10.0…10.0]. Σε περίπτωση που δοθεί λανθασμένη φωτεινότητα, να εμφανίζεται σχετικό μήνυμα σφάλματος και να ζητά από τον χρήστη να την ξαναδώσει

(β) τυπώνει τα ονόματα των αστεριών με τη μεγαλύτερη φωτεινότητα. Σε περίπτωση που υπάρχουν περισσότερα από ένα αστέρια με μέγιστη φωτεινότητα, να τυπώνονται όλα.

Άσκηση 1.42

Σε έναν διαγωνισμό Tetris συμμετέχουν 10 διαγωνιζόμενοι. Κάθε ένας από αυτούς δικαιούται να παίξει 2 φορές και κάθε φορά παίρνει βαθμολογία από το 1 μέχρι το 10, συμπεριλαμβανομένων. Η τελική βαθμολογία του είναι ο μέσος όρος (πραγματικός αριθμός) των δύο προσπαθειών του. Να δημιουργήσετε πρόγραμμα, το οποίο να:

(α) δέχεται τα ονόματα και τις βαθμολογίες και να τα καταχωρίζει στους πίνακες Names, B1 και B2, αντίστοιχα. Να γίνεται έλεγχος αν δίνονται σωστά οι βαθμολογίες

(β) συμπληρώνει τον πίνακα Total με την τελική βαθμολογία κάθε διαγωνιζόμενου

(γ) βρίσκει και να τυπώνει το όνομα του διαγωνιζόμενου με τη μέγιστη τελική βαθμολογία. Σε περίπτωση ισοβαθμίας, να τυπώνονται όλα τα ονόματα

(δ) υπολογίζει και να τυπώνει το πλήθος των διαγωνιζομένων που πέτυχαν βαθμολογία μεγαλύτερη του μέσου όρου των τελικών βαθμολογιών.

Άσκηση 1.43

Να δημιουργήσετε πρόγραμμα, το οποίο να:

(α) δέχεται τα ονόματα και το ύψος 100 ατόμων και να τα καταχωρίζει στους πίνακες names και height αντίστοιχα. Το ύψος είναι ένας δεκαδικός αριθμός μέσα στο διάστημα [1.40…2.30]. Σε περίπτωση που δοθεί λανθασμένο ύψος, να εμφανίζεται σχετικό μήνυμα σφάλματος και να ζητά από τον χρήστη να το ξαναδώσει

(β) τυπώνει το όνομα του ατόμου με το μεγαλύτερο ύψος. Σε περίπτωση που υπάρχουν περισσότερα από ένα άτομα με μέγιστο ύψος, να τυπώνονται όλα.

Άσκηση 1.44

Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται στην πρώτη γραμμή έναν ακέραιο αριθμό N (0<N<=100) και στη δεύτερη γραμμή N ακέραιους αριθμούς. Το πρόγραμμα να τυπώνει όλες τις θέσεις του πίνακα, όπου εμφανίζεται το ελάχιστο στοιχείο του πίνακα και στην επόμενη γραμμή όλες τις θέσεις του πίνακα, όπου εμφανίζεται το μέγιστο στοιχείο του πίνακα.

Παράδειγμα εισόδου

```
10
3 9 2 9 7 8 1 1 9 5
```

Παράδειγμα εξόδου

```
6 7
1 3 8
```

Άσκηση 1.45

Στον πίνακα Α αποθηκεύονται 10 χαρακτήρες. Όλοι οι χαρακτήρες είναι κεφαλαία γράμματα του λατινικού αλφαβήτου. Να δημιουργήσετε πρόγραμμα, το οποίο να τυπώνει το πλήθος των εμφανίσεων του κάθε χαρακτήρα, όπως φαίνεται στο πιο κάτω παράδειγμα.

Σημείωση: Οι χαρακτήρες να εμφανίζονται με αλφαβητική σειρά.

Παράδειγμα εισόδου

```
B F A A F C B B B B
```

Παράδειγμα εξόδου

```
A:2
B:5
C:1
F:2
```

Άσκηση 1.46

Να δημιουργήσετε το πρόγραμμα, το οποίο να:

(α) δέχεται 100 ακέραιους αριθμούς και να τους αποθηκεύει στον πίνακα Nums

(β) εμφανίζει το πλήθος για τους θετικούς, τους αρνητικούς και τα μηδενικά

(γ) δημιουργεί τον πίνακα Filter ως εξής: Αν το Nums[i] είναι θετικός αριθμός που δεν διαιρείται με το 3, τότε Filter[i]=0, διαφορετικά Filter[i]=1

(δ) τυπώνει όλους τους πίνακες με τις κατάλληλες κεφαλίδες.

Άσκηση 1.47

Η ζαριά με άθροισμα 7, όταν ρίξουμε 2 ζάρια, είναι η πιο συνηθισμένη ζαριά. Στο παιχνίδι Lucky Seven, ένας παίκτης προσπαθεί να φέρει όσες περισσότερες, συνεχόμενες ζαριές με άθροισμα 7 μπορεί. Όποιος το κατορθώσει βγαίνει νικητής. Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται το πλήθος των ζαριών ενός παίκτη, τις ζαριές που έχει ρίξει και να εμφανίζει στην οθόνη τις περισσότερες, διαδοχικές φορές που έφερε άθροισμα 7. Να γίνεται έλεγχος αν δίνονται σωστά οι ζαριές (1<=Z1,Z2<=6).

Παράδειγμα εισόδου

```
8
6 1
4 3
5 1
1 2
2 2
3 5
2 5
4 4
```

Παράδειγμα εξόδου

```
2
```

Ασκήσεις Εμπλουτισμού

Άσκηση 1.48

Ο Σπανός έχει επιστρέψει θριαμβευτής στο χωριό του, αφού κατάφερε να νικήσει τους σαράντα δράκους. Σαν ανταμοιβή, ο βασιλιάς υποσχέθηκε να δίνει στον Σπανό ένα χρυσό νόμισμα κάθε μέρα. Ο Σπανός, όμως, χρειάζεται άμεσα Ν χρυσά νομίσματα, για να βοηθήσει τους συγχωριανούς του να επιδιορθώσουν τις ζημιές, τις οποίες προξένησαν στο χωριό τους οι δράκοι και δεν μπορεί να περιμένει για μεγάλο χρονικό διάστημα.

Για να τον βοηθήσει, ο μάγος του χωριού έχει δώσει στον Σπανό ένα μαγικό φίλτρο που του επιτρέπει να διπλασιάζει τα χρυσά νομίσματα που έχει στην κατοχή του όσες φορές θέλει. Αν, για παράδειγμα, ο Σπανός χρειάζεται 4 νομίσματα, τότε θα χρειαστεί μόνο μία μέρα, καθώς θα πάρει ένα χρυσό νόμισμα από τον βασιλιά και θα χρησιμοποιήσει το μαγικό φίλτρο δύο φορές για να τα κάνει αρχικά δύο και στη συνέχεια, τέσσερα χρυσά νομίσματα. Προσοχή όμως! Ο Σπανός δεν μπορεί να υπερβεί τον αριθμό των χρυσών νομισμάτων που χρειάζεται, καθώς θα κατηγορηθεί ότι προσπαθεί να πλουτίσει εις βάρος των συγχωριανών του. Να βρείτε πόσες μέρες θα χρειαστεί ο Σπανός, για να έχει στη κατοχή του Ν χρυσά νομίσματα ακριβώς.

Δεδομένα εισόδου

- Ένας ακέραιος Τ (1<=Τ<=100,000), το πλήθος των αριθμών που ακολουθούν.
- Τ ακέραιοι αριθμοί Ν (0<=Ν<=1,000,000,000) που υποδεικνύουν τον ακριβή αριθμό των χρυσών νομισμάτων που πρέπει να μαζέψει ο Σπανός.

Δεδομένα εξόδου

Για κάθε έναν από τους αριθμούς Ν να εμφανίζεται ένας ακέραιος αριθμός, ο ελάχιστος αριθμός ημερών που θα χρειαστεί ο Σπανός για να μαζέψει ακριβώς Ν χρυσά νομίσματα.

Παράδειγμα εισόδου

```
3
4
5
100
```

Παράδειγμα εξόδου

```
1
2
3
```

Επεξήγηση:

Για να μαζέψει 4 νομίσματα ο Σπανός, παίρνει 1 νόμισμα από τον βασιλιά (1) και χρησιμοποιεί το μαγικό φίλτρο δύο φορές (2, 4), άρα θα χρειαστεί μόνο μία μέρα. Για να μαζέψει 5 χρυσά νομίσματα ο Σπανός, παίρνει 1 νόμισμα από τον βασιλιά την πρώτη μέρα (1), χρησιμοποιεί το μαγικό φίλτρο δύο φορές (2, 4), μετά επισκέπτεται και την επόμενη μέρα τον βασιλιά, για να πάρει ακόμα ένα (5) και να έχει έτσι 5 συνολικά. Για να μαζέψει 100 χρυσά νομίσματα, ο Σπανός παίρνει 1 νόμισμα την πρώτη μέρα (1), χρησιμοποιεί το μαγικό φίλτρο μία φορά (2), παίρνει ακόμη 1 νόμισμα τη δεύτερη μέρα (3), χρησιμοποιεί το μαγικό φίλτρο τρεις φορές (6,

12, 24), παίρνει ακόμη 1 νόμισμα την τρίτη μέρα (25) και χρησιμοποιεί το μαγικό φίλτρο ακόμα δύο φορές (50, 100), για να πάρει 100 νομίσματα ακριβώς.

Άσκηση 1.49

Στο σπίτι της Αριάδνης και του Θησέα οι σοκολάτες αποθηκεύονται σε ένα ψυγείο με ηλεκτρονική κλειδαριά. Ο κωδικός της κλειδαριάς αποτελείται από Ν θετικούς ακέραιους αριθμούς (N>0). Ο Θησέας έχει ανακαλύψει τον κωδικό, αλλά θέλει να παιδέψει λίγο την αδελφή του. Αποφάσισε να της δώσει τον κωδικό της κλειδαριάς κωδικοποιημένο ως εξής: Θα δίνει μία ακολουθία Μ αριθμών (2<M<100) μέσα στην οποία είναι κρυμμένοι οι αριθμοί του κωδικού. Ο κωδικοί αριθμοί θα βρίσκονται στις θέσεις που ισαπέχουν από 2 πρώτους αριθμούς, έναν στα αριστερά και έναν στα δεξιά.

Για παράδειγμα, αν έχουμε την ακολουθία 4, 1, 2, 4, 8, 9, 5, ο κωδικός αριθμός είναι το 8, καθώς απέχει το ίδιο (2 θέσεις) τόσο από το 2 όσο και από το 5.

Να βοηθήσετε την Αριάδνη να δημιουργήσει πρόγραμμα, το οποίο θα της αποκαλύπτει τους κωδικούς αριθμούς της κλειδαριάς.

Σημείωση: Πρώτος αριθμός καλείται ένας ακέραιος θετικός αριθμός μεγαλύτερος του 1, που διαιρείται ακριβώς μόνο από τον εαυτό του και τον αριθμό 1.

Δεδομένα εισόδου

- Ένας ακέραιος αριθμός Μ (2<M<100), το πλήθος των αριθμών που ακολουθούν.
- Μία ακολουθία Μ θετικών ακέραιων αριθμών.

Δεδομένα εξόδου

Οι κωδικοί αριθμοί της κλειδαριάς. Θα υπάρχει πάντα τουλάχιστον ένας. Σε περίπτωση που υπάρχουν περισσότεροι, τότε να τυπώνονται ο ένας κάτω από τον άλλο, με βάση τη σειρά εμφάνισής τους μέσα στην ακολουθία.

Σημείωση: Αν ένας αριθμός βρίσκεται μεταξύ δύο ή περισσοτέρων ζευγαριών πρώτων αριθμών, θα τυπωθεί μόνο μία φορά.

Παράδειγμα εισόδου

```
10
12 3 100 5 21 9 8 7 6 4
```

Παράδειγμα εξόδου

```
100
21
9
```

Επεξήγηση

Με υπογράμμιση φαίνονται οι πρώτοι αριθμοί της ακολουθίας: 12 3 100 5 21 9 8 7 6 4

Ο αριθμός 100 απέχει την ίδια απόσταση τόσο από τον αριθμό 3 όσο και από τον αριθμό 5. Ο αριθμός 21 απέχει την ίδια απόσταση τόσο από τον αριθμό 3 όσο και από τον αριθμό 7. Ο αριθμός 9 απέχει την ίδια απόσταση τόσο από τον αριθμό 5 όσο και από τον αριθμό 7.

Άσκηση 1.50

Η Αριάδνη έχει μάθει στο σχολείο ότι το τετράγωνο ενός αριθμού προκύπτει όταν πολλαπλασιάσουμε τον αριθμό με τον εαυτό του. Παρατήρησε, επίσης, ότι κάποιοι αριθμοί μπορούν να γραφτούν ως άθροισμα δύο τετραγώνων φυσικών αριθμών. Για παράδειγμα, ο αριθμός 5 μπορεί να γραφτεί ως άθροισμα του 1^2+2^2. Κάποιοι αριθμοί, μάλιστα, μπορούν να γραφτούν με περισσότερους από έναν τρόπους (π.χ. $25=0^2+5^2=3^2+4^2$), ενώ κάποιοι άλλοι, όπως ο αριθμός 12, δεν μπορούν να γραφτούν ως άθροισμα τετραγώνων. Η Αριάδνη σάς ζητά να τη βοηθήσετε, ώστε να δημιουργήσει ένα πρόγραμμα, το οποίο να δέχεται κάποιους φυσικούς αριθμούς και να βρίσκει με πόσους διαφορετικούς τρόπους ο καθένας από αυτούς μπορεί να γραφτεί ως άθροισμα δύο τετραγώνων αριθμών.

Δεδομένα εισόδου

- Ένας θετικός ακέραιος αριθμός T (1<=T<=1000) που αντιστοιχεί στο πλήθος των αριθμών που θα ακολουθήσουν.
- Ακολουθούν T γραμμές και κάθε μία περιέχει έναν φυσικό αριθμό N (0<=N<=1,000,000,000).

Δεδομένα εξόδου

Θα εμφανίζονται T γραμμές. Κάθε μία από αυτές θα περιέχει έναν ακέραιο αριθμό, ο οποίος θα αντιστοιχεί στο πλήθος των τρόπων με τους οποίους ο αριθμός N που βρίσκεται στη συγκεκριμένη γραμμή, μπορεί να γραφτεί ως άθροισμα δύο τετραγώνων αριθμών.

Παράδειγμα εισόδου

```
3
17
42
25
```

Παράδειγμα εξόδου

```
1
0
2
```

Επεξήγηση: Ελέγχουμε 3 αριθμούς. Ο αριθμός 17 μπορεί να γραφτεί μόνο με 1 τρόπο ($1^2 + 4^2$), ο αριθμός 42 με κανέναν τρόπο και ο αριθμός 25 με 2 τρόπους ($3^2 + 4^2$ και $0^2 + 5^2$).

Γ7.2 Συμβολοσειρές (Strings)

Τι θα μάθουμε σε αυτό το κεφάλαιο:
- Να δηλώνουμε μία συμβολοσειρά ως αντικείμενο, χρησιμοποιώντας την κατάλληλη βιβλιοθήκη
- Να χρησιμοποιούμε μία συμβολοσειρά ως πίνακα χαρακτήρων
- Να χρησιμοποιούμε έτοιμες συναρτήσεις για τον χειρισμό συμβολοσειρών
- Να χρησιμοποιούμε συμβολοσειρές για την επίλυση προβλημάτων.

1. Εισαγωγή

Στη C++ δεν υπάρχει τύπος δεδομένων για τη δήλωση μίας συμβολοσειράς, όπως υπάρχει για ακέραιους αριθμούς, χαρακτήρες κ.λπ. Υπάρχει, όμως, η δυνατότητα να δηλώσουμε μία συμβολοσειρά ως αντικείμενο, χρησιμοποιώντας τη βιβλιοθήκη `<string>`. Επειδή οι συμβολοσειρές είναι αντικείμενα και όχι τύποι δεδομένων, συνεπάγεται ότι θα έχουν ιδιότητες και έτοιμες συναρτήσεις. Μετά τη δήλωση ενός αντικειμένου τύπου `string`, μπορούμε να το χειριστούμε ως πίνακα χαρακτήρων.

2. Δήλωση και χρήση συμβολοσειράς

Η δήλωση μίας συμβολοσειράς γίνεται με τον ίδιο τρόπο που δηλώνουμε μεταβλητές, χρησιμοποιώντας το αναγνωριστικό `string`. Απαραίτητη προϋπόθεση είναι η δήλωση της βιβλιοθήκης `<string>`.

```
#include<iostream>
#include<string>
using namespace std;

int main(){
    string onoma;
    cout << "What is your name?" << endl;
    cin >> onoma;
    cout << "Hello " << onoma;
return 0;
}
```

Μπορούμε να αρχικοποιήσουμε μία συμβολοσειρά κατά τη δήλωσή της. Μπορούμε, επίσης, να τη χρησιμοποιήσουμε μέσα σε μία συνθήκη. Να σημειωθεί ότι οι συμβολοσειρές (`strings`) δηλώνονται με διπλά εισαγωγικά ("), σε αντίθεση με τις μεταβλητές τύπου `char`, οι οποίες δηλώνονται με μονά εισαγωγικά (').

Παράδειγμα 2.1

Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται το όνομα ενός χρώματος και να τυπώνει τη λέξη «Primary», σε περίπτωση που το χρώμα είναι κόκκινο, κίτρινο ή μπλε, διαφορετικά να τυπώνει «Not Primary».

```
#include<iostream>
#include<string>
using namespace std;
```

```
int main(){
    string color;
    cin >> color;
    if(color=="red" || color=="yellow" || color=="blue")
        cout << "Primary";
    else
        cout << "Not Primary";
return 0;
}
```

(Code: C7_2_example1.cpp)

Όπως αναφέραμε πιο πάνω, μία συμβολοσειρά είναι ουσιαστικά ένας πίνακας από χαρακτήρες και μπορούμε να έχουμε πρόσβαση σε οποιοδήποτε στοιχείο της, αναφέροντας τη θέση του. Ο πρώτος χαρακτήρας της συμβολοσειράς βρίσκεται στη θέση 0. Το παρακάτω πρόγραμμα τυπώνει τους 2 πρώτους χαρακτήρες της συμβολοσειράς test η οποία έχει αρχικοποιηθεί με τη λέξη «hello».

```
#include<iostream>
#include<string>
using namespace std;
int main(){
    string test = "Hello";
    cout << test[0] << test[1];        // Τυπώνει τους χαρακτήρες He
return 0;
}
```

Παράδειγμα 2.2

Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται 2 συμβολοσειρές και να τυπώνει τον αριθμό 1, αν ξεκινούν με τον ίδιο χαρακτήρα, διαφορετικά να τυπώνει τον αριθμό 0.

```
#include<iostream>
#include<string>
using namespace std;
int main(){
    string a, b;
    cin >> a >> b;
    if(a[0]==b[0])          // Στη θέση 0 είναι ο πρώτος χαρακτήρας
        cout << "1";        // της κάθε συμβολοσειράς
    else
        cout << "0";
return 0;
}
```

(Code: C7_2_example2.cpp)

3. Συναρτήσεις της βιβλιοθήκης string

Για τις ανάγκες τους μαθήματος, θα χρησιμοποιήσουμε τις συναρτήσεις `size()`, `clear()`, `empty()` και `getline()`. Η συνάρτηση `size()` επιστρέφει το μέγεθος της συμβολοσειράς, η συνάρτηση `clear()` διαγράφει το περιεχόμενο μίας συμβολοσειράς, η συνάρτηση `empty()` ελέγχει αν η συμβολοσειρά είναι άδεια και η `getline()` διαβάζει ολόκληρη τη γραμμή.

Παράδειγμα 2.3

Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται μία συμβολοσειρά και να τυπώνει το κάθε γράμμα της σε διαφορετική γραμμή.

```
#include<iostream>
#include<string>
using namespace std;
int main(){
    string a;
    cin >> a;
    for(int i=0; i<a.size(); i++)
        cout << a[i] << endl;
return 0;
}
```

(Code: C7_2_example3.cpp)

Η εντολή `a.size()` επιστέφει το μέγεθος της συμβολοσειράς. Για παράδειγμα, αν στο πιο πάνω πρόγραμμα δοθεί η συμβολοσειρά `"Hello"`, η εντολή `a.size()` θα επιστρέψει 5.

Παράδειγμα 2.4

Δίνεται μία συμβολοσειρά που περιέχει πεζούς και κεφαλαίους χαρακτήρες του λατινικού αλφαβήτου. Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται τη συμβολοσειρά και να δημιουργεί μία νέα συμβολοσειρά που να περιέχει μόνο τα κεφαλαία γράμματα της συμβολοσειράς που δόθηκε. Σε περίπτωση που δεν βρεθούν κεφαλαία γράμματα, να τυπώνεται το μήνυμα «No capital letters found».

Σημείωση: Στον κώδικα ASCII, οι κεφαλαίοι λατινικοί χαρακτήρες αναπαρίστανται από το 65 (A) μέχρι και το 90 (Z), ενώ οι πεζοί χαρακτήρες από το 97 (a) μέχρι το 122 (z).

```
#include<iostream>
#include<string>
using namespace std;
int main(){
  string st1, st2;
  st2.clear();          // Διαγράφει το περιεχόμενο της st2
  cin >> st1;
  for(int i=0; i<st1.size(); i++)
     if(st1[i]>=65 && st1[i]<=90){ // ή (st1[i]>='A' && st1[i]<='Z')
        st2 += st1[i]; // Προσθέτει τον χαρακτήρα st1[i] στην st2
      }
```

```
// Αν η st2 είναι άδεια η συνάρτηση st2.empty() θα είναι true
   if(st2.empty())
      cout << "No capital letters found";
   else
      cout << st2;
return 0;
}
```

(Code: C7_2_example4.cpp)

Παράδειγμα 2.5

Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται μία συμβολοσειρά με κεφαλαίους λατινικούς χαρακτήρες και να εμφανίζει τον χαρακτήρα με τις περισσότερες εμφανίσεις.

```
#include <iostream>
#include <string>
using namespace std;

int cnt[26];     // Πίνακας εμφανίσεων κάθε χαρακτήρα. Τα στοιχεία
                 // του πίνακα, εφόσον έχει δηλωθεί εκτός της κύριας
int main(){      // συνάρτησης, αρχικοποιούνται με 0
 string st1;
 cin >> st1;
 for (int i=0; i<st1.size(); i++)
       cnt[st1[i]-'A']++;        // Π.χ. για τον χαρακτήρα C (67)
 int maxt = 0, pos = -1;         // θα αυξηθούν οι εμφανίσεις στη
 for (int i=0; i<26; i++)        // θέση 2 (67-65) του πίνακα cnt
     if (cnt[i]>maxt) {          // Εύρεση μέγιστου στοιχείου
        maxt = cnt[i];
        pos = i;            // Αποθηκεύουμε τη θέση του χαρακτήρα
     }                      // με τις περισσότερες εμφανίσεις
 cout << char (65 + pos);   // Μετατροπή ακεραίου σε χαρακτήρα
return 0;                  // Για πεζούς χαρακτήρες το αντίστοιχο
}                          // θα ήταν char (97 + pos)
```

(Code: C7_2_example5.cpp)

Τέλος, η συνάρτηση `getline` διαβάζει μία συμβολοσειρά μέχρι να συναντήσει αλλαγή γραμμής. Αν έχουμε τις πιο κάτω εντολές και δώσουμε από το πληκτρολόγιο τη φράση «Hello world», το πρόγραμμα θα τυπώσει μόνο τη λέξη «Hello». Ο λόγος είναι ότι η εντολή `cin` θεωρεί το κενό διάστημα (space) ως διαχωριστικό χαρακτήρα για τις συμβολοσειρές.

```
string st1;
cin >> st1;                     // Input: Hello world
cout << st1 << endl;            // Output: Hello
cout << st1.size() << endl;     // 5
```

Αν θέλουμε το πρόγραμμά μας να αποθηκεύσει στη συμβολοσειρά `st1` ολόκληρη τη φράση, πρέπει να χρησιμοποιήσουμε τη συνάρτηση `getline`.

```
string st1;
getline(cin, st1);              // Input: Hello world
cout << st1 << endl;            // Output: Hello world
cout << st1.size() << endl;     // 11
```

Η συνάρτηση `getline` δέχεται 2 παραμέτρους. Η πρώτη παράμετρος αφορά στην μέθοδο εισαγωγής της συμβολοσειράς. Στο παράδειγμά μας, η μέθοδος αυτή είναι το πληκτρολόγιο (`cin`). Η δεύτερη παράμετρος αφορά στο αντικείμενο όπου θα αποθηκευτεί η συμβολοσειρά, η οποία θα διαβαστεί (`st1`).

Σημείωση: Η εντολή `cin` αγνοεί τον χαρακτήρα αλλαγής γραμμής. Για να χρησιμοποιηθεί πριν από την εντολή `getline` μπορούμε να κάνουμε το εξής:

```
int num;
strign st1, temp;
cin >> num;
getline(cin, temp);   // Διαβάζει τον χαρακτήρα αλλαγής γραμμής
getline(cin, st1);    // Αποθηκεύει τη φράση στη συμβολοσειρά st1
```

4. Πίνακες συμβολοσειρών

Μπορούμε να δηλώσουμε πίνακες με συμβολοσειρές με τον ίδιο τρόπο που θα δηλώναμε έναν πίνακα ακεραίων. Για παράδειγμα, η πιο κάτω εντολή δημιουργεί έναν πίνακα 10 συμβολοσειρών:

```
string st[10];
```

Παράδειγμα 2.6

Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται 5 συμβολοσειρές και να τις αποθηκεύει στον πίνακα Α. Στη συνέχεια, να δημιουργεί και να εμφανίζει μία συμβολοσειρά που να περιέχει τον πρώτο χαρακτήρα από την κάθε συμβολοσειρά.

```
#include<iostream>
#include<string>
using namespace std;
int main(){
    string A[5], st;
    for(int i=0; i<5; i++)
        cin >> A[i];
    for(int i=0; i<5; i++)
        st += A[i][0];   // Προσθήκη 1ου χαρακτήρα στη συμβολοσειρά
    cout<<st;
return 0;
}
```

(Code: C7_2_example6.cpp)

Ασκήσεις Κεφαλαίου - www.hackerrank.com/g72

Άσκηση 2.1

Σας δίδονται δύο συμβολοσειρές a και b. Να δημιουργήσετε πρόγραμμα, το οποίο να εμφανίζει κατά σειρά σε κάθε γραμμή:

(α) Το μέγεθος της κάθε συμβολοσειράς

(β) Το αποτέλεσμα της ένωσης των συμβολοσειρών

(γ) Τις δύο συμβολοσειρές, αφού ανταλλάξετε τον πρώτο χαρακτήρα της μίας συμβολοσειράς με την άλλη.

Παράδειγμα εισόδου

```
abcdefg
fyi
```

Παράδειγμα εξόδου

```
7 3
abcdefgfyi
fbcdefg ayi
```

Άσκηση 2.2

Να δημιουργήσετε πρόγραμμα, το οποίο να διαβάζει 10 ονόματα και να εμφανίζει στην οθόνη το όνομα με τους περισσότερους χαρακτήρες.

Άσκηση 2.3

Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται μία λέξη και έναν χαρακτήρα. Αν ο χαρακτήρας υπάρχει μέσα στη λέξη, να τυπώνεται η θέση της πρώτης εμφάνισής του, διαφορετικά να τυπώνεται ο αριθμός -1.

Παράδειγμα εισόδου

```
lannister n
```

Παράδειγμα εξόδου

```
2
```

Άσκηση 2.4

Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται μία λέξη και να αντικαθιστά τον χαρακτήρα 'a' (αν υπάρχει μέσα στη λέξη) με τον χαρακτήρα 'b'.

Παράδειγμα εισόδου

```
abcd
```

Παράδειγμα εξόδου

```
bbcd
```

Άσκηση 2.5

Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται μία λέξη και να μετατοπίζει τους χαρακτήρες της κατά έναν προς τα δεξιά (ο τελευταίος χαρακτήρας θα μεταφερθεί στην πρώτη θέση).

Παράδειγμα εισόδου

```
abcd
```

Παράδειγμα εξόδου

```
dabc
```

Άσκηση 2.6

Η Κωνσταντίνα παρακολουθεί στο σχολείο το μάθημα των δικτύων. Ο τρόπος με τον οποίο δημιουργεί τους κωδικούς για τους εξυπηρετητές είναι ο εξής: παίρνει 2 λέξεις με μέγεθος μεγαλύτερο από 3 χαρακτήρες και ενώνει τους 3 πρώτους χαρακτήρες της πρώτης λέξης, με τους 3 τελευταίους χαρακτήρες της δεύτερης λέξης. Να δημιουργήσετε πρόγραμμα, το οποίο να βοηθά την Κωνσταντίνα να δημιουργήσει τους κωδικούς της.

Παράδειγμα εισόδου

```
abra katabra
```

Παράδειγμα εξόδου

```
abrbra
```

Άσκηση 2.7

Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται μία συμβολοσειρά και να τυπώνει τον i-οστό χαρακτήρα i φορές. Να θεωρήσετε ότι ο πρώτος χαρακτήρας βρίσκεται στη θέση 1.

Παράδειγμα εισόδου

```
abdf
```

Παράδειγμα εξόδου

```
abbdddffff
```

Άσκηση 2.8

Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται τα ονόματα για άγνωστο πλήθος μαθητών και να εμφανίζει το πλήθος των ονομάτων που ξεκινούν με το γράμμα Ν και τελειώνουν με το γράμμα S. Το πρόγραμμα να σταματά, όταν δοθεί η λέξη «STOP».

Παράδειγμα εισόδου

```
NIKOS
STELIOS
NONTAS
NTINA
STOP
```

Παράδειγμα εξόδου

```
2
```

Άσκηση 2.9

Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται μία συμβολοσειρά που αποτελείται από πεζούς, λατινικούς χαρακτήρες και να τυπώνει τον χαρακτήρα με τις περισσότερες εμφανίσεις. Σε περίπτωση που υπάρχουν περισσότεροι από ένας χαρακτήρες με μέγιστο πλήθος εμφανίσεων, να τυπώνεται αυτός που προηγείται αλφαβητικά.

Παράδειγμα εισόδου

```
zavarazovoro
```

Παράδειγμα εξόδου

```
a
```

Άσκηση 2.10

Δίνεται μία μαθηματική έκφραση που περιέχει μέσα παρενθέσεις οι οποίες όμως μπορεί να μην είναι ισορροπημένες. Να δημιουργήσετε πρόγραμμα, το οποίο να διαβάζει τη μαθηματική έκφραση και να τυπώνει την τιμή `true` σε περίπτωση που οι παρενθέσεις είναι ισορροπημένες, διαφορετικά να τυπώνει την τιμή `false`.

Παράδειγμα εισόδου 1

```
1+(2*3)
```

Παράδειγμα εξόδου 1

```
true
```

Παράδειγμα εισόδου 2

```
a+)(b-c)
```

Παράδειγμα εξόδου 2

```
false
```

Άσκηση 2.11

Μία λέξη ονομάζεται παλίνδρομη, όταν μπορεί να διαβαστεί με τον ίδιο τρόπο τόσο από αριστερά όσο και από δεξιά. Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται μία λέξη και να τυπώνει την τιμή `true`, όταν η λέξη είναι παλίνδρομη, διαφορετικά να τυπώνει την τιμή `false`.

Παράδειγμα εισόδου 1

```
anna
```

Παράδειγμα εξόδου 1

```
true
```

Παράδειγμα εισόδου 2

```
abc
```

Παράδειγμα εξόδου 2

```
false
```

Άσκηση 2.12

Στο σύστημα διοίκησης ενός σχολείου, οι απουσίες ενός μαθητή σημειώνονται με το γράμμα Α και οι παρουσίες με το γράμμα Ρ. Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται μία

συμβολοσειρά με τις παρουσίες και τις απουσίες ενός μαθητή και να εμφανίζει τη λέξη «PASS», αν ο μαθητής έχει παρευρεθεί στο 75% των μαθημάτων του, διαφορετικά να εμφανίζει «FAIL».

Παράδειγμα εισόδου

```
APAAPA
```

Παράδειγμα εξόδου

```
FAIL
```

Άσκηση 2.13

Στο τμήμα πληροφορικής S21 υπάρχουν 18 μαθητές. Η Μαρία, η οποία είναι μαθήτρια του τμήματος, αποφάσισε να αναθέσει ένα πρόβλημα στους συμμαθητές της. Τους ζήτησε να δημιουργήσουν πρόγραμμα το οποίο να δέχεται το επίθετο του κάθε μαθητή και στο τέλος να τυπώνει τα επίθετα που ξεκινούν με το αρχικό γράμμα, το οποίο εμφανίζεται τις περισσότερες φορές. Αν, για παράδειγμα, τα περισσότερα επίθετα ξεκινούν με το γράμμα Α, θα εμφανίζονται μόνο αυτά τα επίθετα.

Άσκηση 2.14

Για να κωδικοποιήσουν τους κωδικούς πρόσβασης οι φοιτητές ενός πανεπιστημίου εφάρμοσαν ένα εναλλακτικό αλφάβητο όπου: A➔C, B➔D, C➔E …, Y➔A, Z➔B. Να δημιουργήσετε πρόγραμμα, το οποίο να διαβάζει έναν κωδικό μέχρι 50 χαρακτήρες και να εμφανίζει την κωδικοποιημένη λέξη στην οθόνη. Οι κωδικοί θα αποτελούνται από κεφαλαία γράμματα του λατινικού αλφαβήτου.

Παράδειγμα εισόδου

```
AWAY
```

Παράδειγμα εξόδου

```
CYCA
```

Άσκηση 2.15

Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται μία πρόταση και μία λέξη και να τυπώνει το πλήθος των εμφανίσεων της λέξης μέσα στην πρόταση. Για να διαβαστεί η πρόταση να γίνει χρήση της εντολής `getline`.

Παράδειγμα εισόδου

```
to be or not be a better member not a wannabe
be
```

Παράδειγμα εξόδου

```
5
```

Άσκηση 2.16

Η συμβολοσειρά «AAAAABCCCCCCDDDDCC» αποτελείται μόνο από χαρακτήρες του λατινικού αλφαβήτου και το μήκος της είναι 18 χαρακτήρες. Βάσει του μεγέθους της, αν αποθηκευτεί με αυτή τη μορφή, θα χρειαστούν 18 bytes. Για να περιορίσουμε τη χρήση μνήμης, εφαρμόζουμε την εξής μέθοδο συμπίεσης: όταν υπάρχουν περισσότεροι από έναν συνεχόμενοι ίδιοι χαρακτήρες, αυτοί θα διαγράφονται και θα αντικαθίστανται με έναν αριθμό

ο οποίος θα υποδεικνύει το πλήθος τους, ακολουθούμενος από ένα αντίγραφο του χαρακτήρα. Για παράδειγμα, η πιο πάνω συμβολοσειρά θα μετατραπεί σε «5AB6C4D2C», η οποία έχει μήκος μόλις 9 χαρακτήρες. Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται μία συμβολοσειρά και να τυπώνει τη συμπιεσμένη συμβολοσειρά.

Παράδειγμα εισόδου

```
AAABCDDAA
```

Παράδειγμα εξόδου

```
3ABC2D2A
```

Άσκηση 2.17

Κατά τη διάρκεια ενός πειράματος στο μάθημα Χημείας, ο Παντελής έχει στη διάθεσή του N (2<=N<100) χημικές ενώσεις σε στερεά μορφή. Κάθε χημική ένωση αποτελείται από K χημικά στοιχεία με τη μορφή χαρακτήρων από το a μέχρι το z. Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται N χημικές ενώσεις και να εμφανίζει το πλήθος των κοινών χημικών τους στοιχείων.

Παράδειγμα εισόδου

```
3
abcdde
baccd
eeabg
```

Παράδειγμα εξόδου

```
2
```

Επεξήγηση: Τα κοινά χημικά στοιχεία που εμφανίζονται τουλάχιστον μία φορά και στις 3 ενώσεις είναι τα 'a' και 'b'.

Άσκηση 2.18

Η Αριάδνη έχει κρύψει τον κωδικό της μέσα σε μία συμβολοσειρά. Δεν είναι σίγουρη, όμως, αν το έκανε σωστά. Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται τον κωδικό και τη συμβολοσειρά και να τυπώνει τις θέσεις των χαρακτήρων του κωδικού που εμφανίζονται μέσα στη συμβολοσειρά. Αν ο κωδικός δεν υπάρχει μέσα στη συμβολοσειρά, να τυπώνεται ο αριθμός -1. Οι χαρακτήρες του κωδικού πρέπει να εμφανίζονται μέσα στη συμβολοσειρά με τη σωστή σειρά, αλλά όχι κατά ανάγκη ο ένας δίπλα από τον άλλο. Τόσο ο κωδικός όσο και η συμβολοσειρά αποτελούνται από κεφαλαία γράμματα του λατινικού αλφαβήτου.

Παράδειγμα εισόδου

```
STOP
ABDSTKLOTP
```

Παράδειγμα εξόδου

```
3 4 7 9
```

Άσκηση 2.19

Ο Χρήστος έχει γράψει μία σειρά από λέξεις, ακολουθώντας το πρότυπο cameCase. Σε αυτό το πρότυπο γράφουμε όλες τις λέξεις ενωμένες μεταξύ τους. Το πρώτο γράμμα της πρώτης λέξη γράφεται με μικρό (πεζό) χαρακτήρα, ενώ για όλες τις υπόλοιπες το πρώτο γράμμα

γράφεται με κεφαλαίο. Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται τη συμβολοσειρά που έχει γράψει ο Χρήστος και να εμφανίζει το πλήθος των λέξεων που υπάρχουν σε αυτή.

Παράδειγμα εισόδου

```
oursIsTheFury
```

Παράδειγμα εξόδου

```
4
```

Άσκηση 2.20

Σας δίνεται μία συμβολοσειρά S, μεγέθους μέχρι και 100 χαρακτήρες. Να δημιουργήσετε πρόγραμμα, το οποίο να τυπώνει όλα τα διγράμματα (συμβολοσειρά με δύο χαρακτήρες), καθώς και το πλήθος τους μέσα στη συμβολοσειρά. Η εκτύπωση γίνεται με τη σειρά εμφάνισης των διγραμμάτων μέσα στη συμβολοσειρά.

Παράδειγμα εισόδου

```
abbaabbc
```

Παράδειγμα εξόδου

```
ab:2
bb:2
ba:1
aa:1
bc:1
```

Άσκηση 2.21

Ο Τάσος εργάζεται σε μία διαδικτυακή εταιρεία στην οποία οι πελάτες πρέπει να συνδεθούν με το email τους. Ο προϊστάμενος τού έχει αναθέσει να δημιουργήσει πρόγραμμα το οποίο να δέχεται N (0<N<=100) emails και να ελέγχει την ορθότητά τους. Ένα email χωρίζεται από το σύμβολο @ σε 2 μέρη (username@domain). Οι κανόνες που έχει η εταιρεία για τα emails είναι οι ακόλουθοι:

(α) Ο χαρακτήρας @ υπάρχει μόνο μία φορά.

(β) Επιτρέπονται μόνο οι πεζοί χαρακτήρες, οι τελείες και οι αριθμοί.

(γ) Το username ξεκινά πάντα με πεζό χαρακτήρα.

(δ) Στο domain υπάρχει πάντα μία τελεία, που δεν είναι όμως ο τελευταίος χαρακτήρας.

(ε) Το μέγεθος του username και του domain είναι μεγαλύτερο από 5 χαρακτήρες.

Σε περίπτωση που το email είναι έγκυρο, τυπώνεται η λέξη «valid», διαφορετικά τυπώνεται η λέξη «invalid».

Παράδειγμα εισόδου

```
3
tasse#gmail.com
tasse1@lag.com
1tasse@yahoo.com
```

Παράδειγμα εξόδου

```
invalid
valid
invalid
```

Άσκηση 2.22

Παντόγραμμα καλούμε μία φράση (όσο γίνεται πιο σύντομη), η οποία περιλαμβάνει όλα ανεξαιρέτως τα γράμματα του αλφαβήτου. Τα παντογράμματα, εκτός από την ψυχαγωγική τους αξία, είναι χρήσιμα στην τυπογραφία και τον σχεδιασμό γραμματοσειρών, αφού παρουσιάζουν όλα τα γράμματα. Για παράδειγμα, η φράση «The quick brown fox jumps over the lazy dog» είναι ένα παντόγραμμα. Να δημιουργήσετε πρόγραμμα, το οποίο να υπολογίζει αν μία φράση είναι παντόγραμμα και να τυπώνει τον αριθμό 1, διαφορετικά να τυπώνει τον αριθμό 0.

Παράδειγμα εισόδου

```
The quick brown fox jumps over the lazy dog
```

Παράδειγμα εξόδου

```
1
```

Ασκήσεις Εμπλουτισμού

Άσκηση 2.23

Το DNA αποτελείται από τέσσερα νουκλεοτίδια (αδενίνη, γουανίνη, κυτοσίνη και θυμίνη). Στο συγκεκριμένο πρόβλημα δίνεται μία συμβολοσειρά DNA, την οποία θα πρέπει να μετατρέψετε σε περιοδική.

Μία συμβολοσειρά μήκους N καλείται περιοδική, με περίοδο p, αν ο i-οστός χαρακτήρας είναι ίσος με τον (i+p)-οστό χαρακτήρα για κάθε χαρακτήρα από τον πρώτο έως τον (N-p)-οστό. Για παράδειγμα, οι συμβολοσειρές «CATCATC», «CATCAT», «ACTAC» και «ACT» είναι όλες περιοδικές με περίοδο 3.

Σας δίνεται μία συμβολοσειρά S και ένας ακέραιος K. Να δημιουργήσετε πρόγραμμα, το οποίο να υπολογίζει τον ελάχιστο αριθμό αντικαταστάσεων που είναι απαραίτητες, ώστε η τελική συμβολοσειρά να είναι περιοδική, με περίοδο μικρότερη ή ίση του K. Κάθε αντικατάσταση αποτελείται από την αλλαγή ενός μόνο χαρακτήρα από ένα γράμμα σε ένα οποιοδήποτε άλλο.

Δεδομένα εισόδου

Στην πρώτη γραμμή βρίσκονται δύο ακέραιοι αριθμοί N (1<=N<=1000) και K (1<=K<=N). Η δεύτερη γραμμή αποτελείται από N χαρακτήρες (A, G, C ή T).

Δεδομένα εξόδου

Η έξοδος αποτελείται από έναν μόνο αριθμό, το πλήθος των ελάχιστων αντικαταστάσεων που απαιτούνται για τη μετατροπή της συμβολοσειράς σε περιοδική.

Παράδειγμα εισόδου

```
7 2
ACGTGCA
```

Παράδειγμα εξόδου

```
3
```

Επεξήγηση: Με τρεις αντικαταστάσεις η συμβολοσειρά γίνεται AC**ACA**CA, η οποία είναι περιοδική με περίοδο 2.

Άσκηση 2.24

Ο Μάριος περνάει την ώρα του δημιουργώντας παιχνίδια αντί να διαβάζει για το σχολείο. Οι κανόνες του παιχνιδιού είναι οι ακόλουθοι:

- Στην αρχή διαλέγει στην τύχη μία λέξη από το σχολικό βιβλίο. Ύστερα χωρίζει τη λέξη σε δύο αυθαίρετα σημεία, ώστε να πάρει τρεις ξεχωριστές λέξεις.
- Στη συνέχεια, αντιστρέφει τη σειρά των γραμμάτων στην κάθε μία από αυτές τις τρεις λέξεις (αλλάζει το πρώτο με το τελευταίο, το δεύτερο με το προτελευταίο κ.ο.κ.).
- Στο τέλος, ξαναενώνει τις τρείς λέξεις στην ίδια σειρά με την οποία ήταν αρχικά.

Ο στόχος του παιχνιδιού είναι να βρει μία όσο το δυνατόν μικρότερη λεξικογραφικά λέξη που να μπορεί να δημιουργηθεί από αυτή την διαδικασία.
Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται μία συμβολοσειρά και να εμφανίζει τη μικρότερη λεξικογραφικά λέξη που μπορεί να προκύψει.

Δεδομένα εισόδου

Η λέξη που διάλεξε ο Μάριος από το βιβλίο. Θα αποτελείται από 3 έως 50 λατινικούς χαρακτήρες.

Δεδομένα εξόδου

Η μικρότερη λεξικογραφικά λέξη που μπορεί να προκύψει, σε μία γραμμή.

Παράδειγμα εισόδου 1

```
dcbagfekjih
```

Παράδειγμα εξόδου 1

```
abcdefghijk
```

Παράδειγμα εισόδου 2

```
mobitel
```

Παράδειγμα εξόδου 2

```
bometil
```

Παράδειγμα εισόδου 3

```
anakonda
```

Παράδειγμα εξόδου 3

```
aanadnok
```

Άσκηση 2.25

«Συμμετρική» συμβολοσειρά καλείται μία συμβολοσειρά S, της οποίας ο κάθε χαρακτήρας εμφανίζεται τόσες φορές όσες και ο κάθε ένας από τους υπόλοιπους χαρακτήρες. Για παράδειγμα, η συμβολοσειρά S1="xxyyzz" είναι συμμετρική, ενώ η συμβολοσειρά S2="aabbcd", όχι. Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται μία συμβολοσειρά και να εμφανίζει τη λέξη «YES», αν η συμβολοσειρά είναι συμμετρική, ή τη λέξη «NO», αν δεν είναι.

Σε περίπτωση που η συμβολοσειρά δεν είναι συμμετρική, το πρόγραμμα να εντοπίζει αν η συμβολοσειρά μπορεί να μετατραπεί σε συμμετρική, αφαιρώντας μόνο έναν χαρακτήρα από αυτή. Σε τέτοια περίπτωση, να εμφανίζεται ο αριθμός 1, αλλιώς να εμφανίζεται ο αριθμός 0. Να θεωρήσετε ότι όλοι οι χαρακτήρες της συμβολοσειράς θα είναι πεζοί χαρακτήρες του λατινικού αλφαβήτου.

Παράδειγμα εισόδου

```
dccdefe
```

Παράδειγμα εξόδου

```
NO
1
```

Επεξήγηση: Η συμβολοσειρά δεν είναι συμμετρική, όμως, αφαιρώντας τον χαρακτήρα 'f' μετατρέπεται σε συμμετρική.

Άσκηση 2.26

Για να κρυπτογραφήσει την ηλεκτρονική αλληλογραφία που στέλνει στους φίλους του, ο Παντελής θέλει να εφαρμόσει την εξής μέθοδο. Για κάθε χαρακτήρα του λατινικού αλφαβήτου που θέλει θα χρησιμοποιήσει, θα βάλει στη θέση του τον χαρακτήρα ο οποίος βρίσκεται Χ θέσεις μετά από αυτόν στο αλφάβητο. Για παράδειγμα, αν βάλει τον χαρακτήρα 'm' και το Χ=5, το πρόγραμμα θα τον μετατρέψει σε 'r'. Αν βάλει τον χαρακτήρα 'W' και το Χ=4, το πρόγραμμα θα τον μετατρέψει σε 'A'. Να δημιουργήσετε πρόγραμμα, το οποίο να διαβάζει μία πρόταση που αποτελείται από κεφαλαίους και πεζούς χαρακτήρες του λατινικού αλφαβήτου, αλλά και σημεία της στίξης και, ακολούθως, να διαβάζει τον ακέραιο Χ και να εφαρμόζει την πιο πάνω διαδικασία κρυπτογράφησης.

Παράδειγμα εισόδου

```
Hello my friend. How are you?
8
```

Παράδειγμα εξόδου

```
Pmttw ug nzqmvl. Pwe izm gwc?
```

Επεξήγηση: Η πρόταση κρυπτογραφείται, βάζοντας στη θέση του κάθε χαρακτήρα τον χαρακτήρα ο οποίος βρίσκεται 8 θέσεις μετά στο αλφάβητο. Τα σημεία της στίξης δεν υφίστανται καμία αλλαγή και παραμένουν ως έχουν.

Γ7.3 Αρχεία (Files)

Τι θα μάθουμε σε αυτό το κεφάλαιο:

- Να χρησιμοποιούμε την κατάλληλη βιβλιοθήκη για την προσπέλαση αρχείων
- Να ανοίγουμε ένα αρχείο και να διαβάζουμε το περιεχόμενό του
- Να δημιουργούμε ένα αρχείο και να γράφουμε πληροφορίες σε αυτό
- Να ανοίγουμε ένα αρχείο και να προσθέτουμε πληροφορίες σε αυτό
- Να ελέγχουμε για το τέλος ενός αρχείου
- Να κλείνουμε ένα αρχείο
- Να δημιουργούμε προγράμματα που θα χρησιμοποιούν αρχεία.

1. Εισαγωγή

Μέχρι στιγμής, η είσοδος δεδομένων στα προγράμματά μας γινόταν από το πληκτρολόγιο και η έξοδος πληροφοριών γινόταν στην οθόνη. Η χρήση των εντολών `cin/cout` δεν είναι, όμως, πολύ βολική, όταν έχουμε να διαχειριστούμε μεγάλο όγκο δεδομένων. Σε τέτοιες περιπτώσεις ενδείκνυται η χρήση αρχείων. Μπορούμε να χρησιμοποιήσουμε τα αρχεία, για να διαβάσουμε δεδομένα και να γράψουμε σε αυτά πληροφορίες. Η βιβλιοθήκη που περιέχει τις εντολές που αφορούν στα αρχεία είναι η `<fstream>`.

2. Ανάγνωση και εγγραφή σε αρχείο

Η εντολή που χρησιμοποιείται για την ανάγνωση ενός αρχείου είναι η `ifstream`. Με την εντολή `ifstream` δημιουργούμε μία ροή (`stream`) ανάγνωσης προς ένα εξωτερικό αρχείο. Αντίστοιχα, με την εντολή `ofstream` δημιουργούμε μία ροή εγγραφής προς ένα εξωτερικό αρχείο.

Παράδειγμα 3.1

Να δημιουργήσετε πρόγραμμα, το οποίο να διαβάζει δύο ακέραιους αριθμούς από το αρχείο `input.txt` και να τυπώνει το άθροισμά τους στο αρχείο `output.txt`.

```
#include<fstream> //βιβλιοθήκη για ifstream, ofstream
using namespace std;

int main(){
// Δημιουργία ροής ανάγνωσης για το αρχείο input.txt με όνομα fin
    ifstream fin("input.txt");
// Δημιουργία ροής εγγραφής για το αρχείο output.txt με όνομα fout
    ofstream fout("output.txt");
    int a, b, c;
    fin >> a >> b;
    c = a + b;
    fout << c << endl;
    fin.close();            // Τερματισμός ροής ανάγνωσης
    fout.close();           // Τερματισμός ροής εγγραφής
return 0;
}
```

(Code: C7_3_example1.cpp)

Το αρχείο input.txt **πρέπει να βρίσκεται στον ίδιο φάκελο με το πρόγραμμα**, ενώ το αρχείο output.txt δεν είναι απαραίτητο να προϋπάρχει, αφού θα δημιουργηθεί από το πρόγραμμα στον ίδιο φάκελο. Είναι σημαντικό να αναφέρουμε ότι το όνομα των ροών (fin, fout) μπορεί να είναι οτιδήποτε, ό,τι όμως δηλωθεί ως ροή θα πρέπει να χρησιμοποιηθεί και στη συνέχεια με τους τελεστές >>, για ανάγνωση και <<, για εγγραφή.

Παράδειγμα 3.2

Να δημιουργήσετε πρόγραμμα, το οποίο να διαβάζει 100 ακέραιους αριθμούς από το αρχείο in.txt και να τυπώνει στο αρχείο out.txt όσους είναι μεγαλύτεροι από τον μέσο όρο τους.

```
#include<fstream>
using namespace std;

int main(){
    ifstream infile("in.txt");          // Ροή ανάγνωσης - infile
    ofstream outfile("out.txt");        // Ροή εγγραφής - outfile
    int i, A[100], sum = 0;
    float mo;
    for(i=0; i<100; i++){
        infile >> A[i];
        sum += A[i];                    // Άθροισμα στοιχείων
    }
    mo = sum / 100.0;
    for(i=0; i<100; i++)
        if(A[i]>mo)
            outfile << A[i] << endl;    // Εμφάνιση στοιχείων
    infile.close();
    outfile.close();
return 0;
}
```

(Code: C7_3_example2.cpp)

Ήδη αναφέραμε ότι τα ονόματα των ροών μπορεί να διαφέρουν από πρόγραμμα σε πρόγραμμα. Μπορείτε να το παρατηρήσετε στα δύο πιο πάνω παραδείγματα.

3. Εγγραφή στο τέλος ενός αρχείου

Η C++ μάς δίνει τη δυνατότητα να γράψουμε στο τέλος ενός αρχείου (append). Αν, για παράδειγμα, θέλουμε να γράψουμε κάτι στο τέλος τους αρχείου file.txt, θα πρέπει να δημιουργήσουμε μία ροή όπως πιο κάτω.

```
ofstream myout("file.txt", (ios::out | ios::app));
```

Παράδειγμα 3.3

Να δημιουργήσετε πρόγραμμα, το οποίο να διαβάζει πέντε ονόματα από το αρχείο names.txt και να τα τυπώνει στο τέλος του αρχείου append.txt.

```
#include<fstream>
#include<string>
using namespace std;

int main(){
    ifstream fin("names.txt");
    // Δημιουργία ροής εγγραφής στο τέλος του αρχείου (append)
    ofstream fout("append.txt",(ios::out|ios::app));
    string name;
    for(int i=0; i<5; i++){
        fin >> name;
        fout << name << endl;
    }
    fin.close();
    fout.close();
return 0;
}
```

(Code: C7_3_example3.cpp)

Επεξήγηση παραδείγματος

Αν τα δύο αρχεία στο πιο πάνω παράδειγμα, είχαν αρχικά τη μορφή:

names.txt

```
Mark
Mason
Mindy
Mike
Matt
```

append.txt

```
John
Jamie
Jason
Jonathan
```

Με την προσθήκη των ονομάτων από το αρχείο names.txt, το αρχείο append.txt θα γίνει:

```
John
Jamie
Jason
Jonathan
Mark
Mason
Mindy
Mike
Matt
```

4. Έλεγχος για το τέλος ενός αρχείου

Πολλές φορές δεν γνωρίζουμε το πλήθος των στοιχείων ενός αρχείου. Για να το υπολογίσουμε, μπορούμε να διαβάσουμε όλα τα στοιχεία του αρχείου μέχρι να φτάσουμε στο τέλος του. Αυτό μπορεί να γίνει με τη χρήση της συνάρτησης `eof()`, η οποία επιστρέφει την τιμή `true` όταν φτάσουμε στο τέλος του αρχείου.

Παράδειγμα 3.4

Να δημιουργήσετε πρόγραμμα, το οποίο να βρίσκει τον μέσο όρο ηλικίας για άγνωστο πλήθος ατόμων που υπάρχουν στο αρχείο `persons.txt`. Το αρχείο θα περιέχει δύο στήλες με τα ονόματα και τις ηλικίες N (2<=N<=1000) ατόμων. Το πρόγραμμα να αποθηκεύει τα στοιχεία σε κατάλληλους πίνακες και να τυπώνει το αποτέλεσμα στην οθόνη με ένα δεκαδικό ψηφίο.

Παράδειγμα εισόδου (persons.txt)

```
James   45
Johnson 76
Jeremy  58
Jacobs  51
```

Παράδειγμα εξόδου (οθόνη)

```
57.5
```

```
#include<fstream>
#include<iostream>
#include<iomanip>
#include<string>
using namespace std;
int main(){
    int sum = 0, i = 0;
    string names[1000];          // Οι πίνακες δηλώνονται με βάση το
    int ages[1000];              // μέγιστο δυνατό πλήθος (1000)
    double avg;
    ifstream fin("persons.txt");
    while(!fin.eof()){           // Έλεγχος για το τέλος του αρχείου
        fin >> names[i] >> ages[i];
        sum += ages[i];
        i++;                     // Αύξηση πλήθους ατόμων
    }
    avg = (double) sum / i;
    cout << fixed << setprecision(1) << avg << endl;
    fin.close();
return 0;
}
```

(Code: C7_3_example4.cpp)

Ασκήσεις Κεφαλαίου

Τα αρχεία των ασκήσεων βρίσκονται στο επιπρόσθετο υλικό

Άσκηση 3.1

Να δημιουργήσετε πρόγραμμα, το οποίο να διαβάζει 10 ακέραιους αριθμούς από το αρχείο in.txt και να τυπώνει στο αρχείο out1.txt τον μεγαλύτερο από αυτούς.

Άσκηση 3.2

Να δημιουργήσετε πρόγραμμα, το οποίο να διαβάζει 10 ακέραιους αριθμός από το αρχείο in.txt και να τυπώνει στο αρχείο out2.txt τον μικρότερο από αυτούς.

Άσκηση 3.3

Στο αρχείο ask3_in.txt υπάρχουν 3 θετικοί ακέραιοι τριψήφιοι αριθμοί. Να δημιουργήσετε πρόγραμμα, το οποίο να τυπώνει στο αρχείο ask3_out.txt τον αριθμό που προκύπτει από τις εκατοντάδες του πρώτου, τις δεκάδες του δεύτερου και τις μονάδες του τρίτου αριθμού.

Παράδειγμα εισόδου (ask3_in.txt)

```
310 921 145
```

Παράδειγμα εξόδου (ask3_out.txt)

```
325
```

Άσκηση 3.4

Στο αρχείο lines.txt υπάρχει ένα κείμενο 15 γραμμών. Να δημιουργήσετε πρόγραμμα, το οποίο να το αντιγράφει στο αρχείο out4.txt και να το προσθέτει στο τέλος (append) του αρχείου ask4.txt.

Άσκηση 3.5

Να δημιουργήσετε πρόγραμμα, το οποίο να υπολογίζει το άθροισμα των αριθμών που υπάρχουν στο αρχείο 5in1.txt και στο αρχείο 5in2.txt. Το πλήθος των αριθμών στο κάθε αρχείο δεν είναι γνωστό. Το αποτέλεσμα (ένας ακέραιος) να τυπωθεί στο αρχείο sum.txt.

Άσκηση 3.6

Τα ονόματα και οι ηλικίες των κατοίκων ενός χωριού είναι αποθηκευμένα στο αρχείο citizens.txt. Να δημιουργήσετε πρόγραμμα, το οποίο να τυπώνει στο αρχείο elder.txt τα ονόματα των κατοίκων που έχουν ξεπεράσει το 65ο έτος ηλικίας.

Παράδειγμα εισόδου (citizens.txt)

```
30 Mitsos
70 Marikkou
20 Kokos
```

Παράδειγμα εξόδου (elder.txt)

```
Marikkou
```

Άσκηση 3.7

Στο αρχείο students.txt βρίσκονται οι αριθμοί μητρώου και τα 2 διαγωνίσματα των μαθητών ενός τμήματος. Να δημιουργήσετε πρόγραμμα, το οποίο να διαβάζει τα στοιχεία από το αρχείο students.txt και να τυπώνει στην οθόνη τους αριθμούς μητρώου των μαθητών που η διαφορά των 2 διαγωνισμάτων είναι μεγαλύτερη από 4 μονάδες.

Παράδειγμα εισόδου (students.txt)

```
1001 14 16
1002 19 11
1003 19 19
1004 13 20
```

Παράδειγμα εξόδου (οθόνη)

```
1002
1004
```

Άσκηση 3.8

Να δημιουργήσετε πρόγραμμα, το οποίο να διαβάζει 100 αριθμούς από το αρχείο nums.txt και να τυπώνει τους άρτιους στο αρχείο even.txt και τους περιττούς στο αρχείο odd.txt.

Άσκηση 3.9

Στο αρχείο names.txt υπάρχει ένας ακέραιος αριθμός Ν ακολουθούμενος από Ν ονόματα. Να δημιουργήσετε πρόγραμμα, το οποίο να διαβάζει τα ονόματα και να αποθηκεύει στο αρχείο c.txt τα ονόματα που ξεκινούν από το γράμμα C.

Παράδειγμα εισόδου (names.txt)

```
4
Cynthia
John
Mary
Cecil
```

Παράδειγμα εξόδου (c.txt)

```
Cynthia
Cecil
```

Άσκηση 3.10

Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται από το πληκτρολόγιο έναν ακέραιο αριθμό Ν και έναν χαρακτήρα και να τυπώνει στο αρχείο square.txt ένα τετράγωνο σχήμα διαστάσεων Ν x Ν, με τον χαρακτήρα που έχει δοθεί.

Παράδειγμα εισόδου (πληκτρολόγιο)

```
3 +
```

Παράδειγμα εξόδου (square.txt)

```
+++
+++
+++
```

Άσκηση 3.11

Να δημιουργήσετε πρόγραμμα, το οποίο να διαβάζει δύο αρχεία (file1.txt, file2.txt), να τα συγκρίνει και να εμφανίζει στην οθόνη τα μηνύματα «Files are the same», αν τα δύο αρχεία είναι τα ίδια, ή «Files are the not the same», στην αντίθετη περίπτωση.

Άσκηση 3.12

Να δημιουργήσετε πρόγραμμα, το οποίο να διαβάζει, αρχικά, το αρχείο file1.txt, να προσθέτει τα περιεχόμενα του αρχείου file1.txt στο τέλος (append) του αρχείου file2.txt και, τέλος, να τυπώνει όλα τα περιεχόμενα του αρχείου file2.txt στο αρχείο file3.txt.

Άσκηση 3.13

Να δημιουργήσετε πρόγραμμα, το οποίο να διαβάζει μία συμβολοσειρά από το πληκτρολόγιο και ένα αρχείο (lines.txt). Το πρόγραμμα να αναζητά τη συμβολοσειρά μέσα στο αρχείο και αν την εντοπίσει, να εμφανίζει στην οθόνη το πλήθος των εμφανίσεών της μέσα σε αυτό, αλλιώς, αν δεν εντοπιστεί, να εμφανίζει τον αριθμό -1.

Άσκηση 3.14

Ένας φίλος σάς έχει στείλει τον κωδικό του υπολογιστή του στο ηλεκτρονικό ταχυδρομείο. Για σκοπούς ασφαλείας έχει κρύψει τον κωδικό μέσα στο αρχείο secret.txt. Ο κωδικός αποτελείται από λατινικά γράμματα (κεφαλαία και πεζά) και ψηφία από το 0 μέχρι το 9, συμπεριλαμβανομένων. Να δημιουργήσετε πρόγραμμα, το οποίο να διαβάζει το αρχείο secret.txt και να τυπώνει στο αρχείο password.txt τον κωδικό του φίλου σας.

Άσκηση 3.15

Στο αρχείο primary.txt είναι αποθηκευμένοι όλοι οι μονοψήφιοι πρώτοι αριθμοί. Ένας αριθμός καλείται πρώτος, αν διαιρείται μόνο από τον αριθμό 1 και τον εαυτό του. Να δημιουργήσετε πρόγραμμα, το οποίο να προσθέτει στο τέλος (append) του αρχείου primary.txt, όλους τους διψήφιους πρώτους αριθμούς.

Γ7.4 Συναρτήσεις (Functions)

Τι θα μάθουμε σε αυτό το κεφάλαιο:

- Να αναγνωρίζουμε προβλήματα που χρειάζονται για την επίλυσή τους υποπρογράμματα / συναρτήσεις (functions)
- Να αναφέρουμε τι είναι υποπρόγραμμα/συνάρτηση και ποιον σκοπό εξυπηρετεί σε ένα πρόγραμμα
- Να αναγνωρίζουμε τη δομή της επικεφαλίδας μίας συνάρτησης και να εντοπίζουμε τα μέρη της (όνομα/αναγνωριστικό, τυπικές παράμετροι, επιστρεφόμενη τιμή)
- Να αναγνωρίζουμε τη δομή του σώματος μίας συνάρτησης και να εντοπίζουμε τα μέρη της (τοπικές μεταβλητές, αρχή και τέλος, επιστροφή τιμής)
- Να αναγνωρίζουμε τη δομή κλήσης μίας συνάρτησης (πραγματικές παράμετροι, αντιστοίχιση με τις τυπικές παραμέτρους και επιστρεφόμενη τιμή)
- Να διακρίνουμε τους δύο τρόπους ορισμού μίας συνάρτησης (μαζί με το σώμα, πριν την κλήση της και με δήλωση προτύπου)
- Να αναφέρουμε τον ρόλο της εντολής επιστροφής τιμής (return)
- Να αναγνωρίζουμε την ύπαρξη συναρτήσεων που δεν επιστρέφουν τιμή (π.χ. τύπου void)
- Να συντάσσουμε την επικεφαλίδας μίας συνάρτησης (επιλογή τύπου επιστρεφόμενης τιμής, αναγνωριστικού και αναγνωριστικών και τύπων δεδομένων των τυπικών παραμέτρων)
- Να συντάσσουν το σώμα μίας συνάρτησης (καθορισμός τοπικών μεταβλητών, εντολών και επιστροφή τιμής)
- Να αναγνωρίζουμε τον ρόλο των καθολικών μεταβλητών (global variables)
- Να εντοπίζουμε το πεδίο εφαρμογής (scope) μίας μεταβλητής
- Να διακρίνουμε πότε μία παράμετρος αποστέλλεται με τιμή (value parameter) και πότε με αναφορά (reference parameter, π.χ. &a)
- Να χρησιμοποιούμε παραμέτρους εξ αναφοράς για «επιστροφή» πολλαπλών τιμών
- Να χρησιμοποιούμε παραμέτρους τύπου πίνακα (array) σε μία συνάρτηση και να επισημαίνουμε τη διαφορά σε σύγκριση με τις απλές μεταβλητές
- Να αναγνωρίζουμε από την περιγραφή του προβλήματος τις τυπικές παραμέτρους και τον αντίστοιχο τύπο τους που πρέπει να δέχεται μία συνάρτηση αλλά και τον τύπο του αποτελέσματος που επιστρέφεται
- Να αναγνωρίζουμε από την περιγραφή του προβλήματος αν σε μία συνάρτηση χρειάζονται τοπικές μεταβλητές και ποιών τύπων
- Να ακολουθούμε κατάλληλη στρατηγική για επίλυση ενός σύνθετου προβλήματος (π.χ. διάσπαση σε απλούστερα μέρη, επίλυση των μερών, σύνθεση και έλεγχος, μέθοδος της σταδιακής βελτίωσης – stepwise refinement) με χρήση συναρτήσεων
- Να καθορίζουμε το πρόβλημα με ακρίβεια, συγκεκριμένα: να εντοπίζουμε/ διακρίνουμε τα Δεδομένα, τις Πληροφορίες και την Επεξεργασία
- Να σχεδιάζουμε τον τρόπο επίλυσης του προβλήματος
- Να επιλέγουμε να διασπάσουμε το πρόβλημα με χρήση κατάλληλων συναρτήσεων και δομών (επανάληψης ή και διακλάδωσης) για επίλυση του προβλήματος
- Να υλοποιούμε τον σχεδιασμό τους σε πρόγραμμα, με τη χρήση του προγραμματιστικού περιβάλλοντος, ώστε να επιλυθεί το πρόβλημα
- Να επιλέγουμε κατάλληλα δεδομένα και στρατηγική για έλεγχο των συναρτήσεων τους και ολόκληρου του προγράμματος
- Να ελέγχουμε την ορθότητα της λύσης του προβλήματος, χρησιμοποιώντας τη μέθοδο της προκαταρκτικής εκτέλεσης ή και άλλες μεθόδους για επαλήθευση
- Να μελετούμε έτοιμο πρόγραμμα, το οποίο περιλαμβάνει συναρτήσεις και να εντοπίζουμε βασικά μέρη του, τα οποία συνδέονται με πτυχές του προβλήματος που επιλύει
- Να προσθέτουμε επεξηγηματικά σχόλια σε ένα πρόγραμμα
- Να συμπληρώνουμε ένα έτοιμο πρόγραμμα από το οποίο λείπουν δηλώσεις συναρτήσεων ή και κλήσεις συναρτήσεων
- Να εντοπίζουμε και να αναγνωρίζουμε σε ένα πρόγραμμα πρότυπα σχεδίασης και στρατηγικές (design patterns, τμήματα κώδικα)
- Να χρησιμοποιούμε πρότυπα σχεδίασης και στρατηγικές που εντοπίσαμε ως εργαλεία επίλυσης προβλημάτων σε νέα προγράμματά τους.

1. Εισαγωγή

Τμηματικός προγραμματισμός (Modular programming) είναι μία τεχνική ανάπτυξης προγραμμάτων, της οποίας κύριο χαρακτηριστικό είναι η διάσπαση ενός προβλήματος σε πιο απλά τμήματα προγραμμάτων, έτσι ώστε να διευκολυνθεί η αποσφαλμάτωση και να μειωθεί η έκταση του κώδικά. Αυτά τα τμήματα κώδικά καλούνται υποπρογράμματα. Υποπρόγραμμα θεωρείται ένα αυτόνομο κομμάτι κώδικα, το οποίο εκτελεί ένα συγκεκριμένο έργο και ορίζεται ξεχωριστά από την κύρια συνάρτηση (main). Η σωστή εφαρμογή του τμηματικού προγραμματισμού διασφαλίζει τη σωστή και απλούστερη δημιουργία προγραμμάτων. Επίσης, διευκολύνει την κατανόηση του τρόπου λειτουργίας ενός προγράμματος.

2. Εφαρμογή τμηματικού προγραμματισμού σε πρόγραμμα

Έχουμε το εξής πρόβλημα:

Ένας καθηγητής θέλει να δημιουργήσει πρόγραμμα το οποίο να μπορεί να αποθηκεύει τα ονόματα και τους βαθμούς των τετραμήνων για κάθε ένα από τα τμήματα που διδάσκει και να μπορεί να εμφανίζει στην οθόνη τον γενικό τους βαθμό. Για να επαναλάβει όλες τις πιο πάνω διαδικασίες για κάθε ένα από τα τμήματα, ο καθηγητής θα χρειαστεί να επαναλάβει αρκετές φορές κάποιες από τις εντολές του ίδιου κώδικα. Διαπιστώνουμε ότι το πρόγραμμα μπορεί να χωριστεί σε τρία διαφορετικά «κομμάτια» κώδικα:

(α) Καταχώριση δεδομένων
(β) Επεξεργασία δεδομένων
(γ) Εμφάνιση αποτελεσμάτων (πληροφοριών)

Για κάθε ένα από τα πιο πάνω τμήματα μπορεί να υλοποιηθεί ένα υποπρόγραμμα το οποίο να εκπληρώνει τη συγκεκριμένη εργασία αυτόνομα και σε συνεργασία με τα υπόλοιπα υποπρογράμματα να δίνει τα επιθυμητά αποτελέσματα. Για κάθε τμήμα, ο καθηγητής θα καλεί τα υποπρογράμματα με μία μόνο εντολή και θα αποφύγει να επαναλάβει όλες τις εντολές για να υπολογίσει τους γενικούς βαθμούς για κάθε τμήμα.

3. Ορισμός και κλήση συναρτήσεων

Η συνάρτηση (function) είναι ένας τύπος υποπρογράμματος που υπολογίζει και επιστρέφει **μόνο μία τιμή** στην κύρια συνάρτηση (main). Ας δούμε ένα παράδειγμα το οποίο χρησιμοποιεί μία συνάρτηση, για να επιστρέφει το άθροισμα δύο αριθμών. Η συνάρτηση ορίζεται ξεχωριστά από την κύρια συνάρτηση (main) ως εξής:

```
#include <iostream>
using namespace std;

int add_2_nums(int a, int b){          // Ορισμός συνάρτησης
      return a + b;                    // Επιστροφή αθροίσματος
}                                      // Τέλος σώματος συνάρτησης

int main() {                           // Κύρια συνάρτηση (main)
      cout << add_2_nums(65,97);       // Κλήση συνάρτησης add_2_nums
return 0;                              // μέσα στην κύρια συνάρτηση
}                                      // Τέλος κύριας συνάρτησης
```

Για κάθε συνάρτηση πρέπει να ορίζουμε τα εξής:

(α) **Τύπος επιστρεφόμενης τιμής**: Η συνάρτηση μπορεί να επιστρέψει όλους τους βασικούς τύπους που γνωρίζουμε (`int`, `double`, `char`, `boolean`), ακόμα και συμβολοσειρές (`string`). **Αν η συνάρτηση δεν θα επιστρέφει κάποια τιμή, ορίζεται με τύπο** `void`. Αν επιστρέφει κάποια τιμή, αυτή θα εμφανίζεται μετά τη λέξη `return`.

(β) **Όνομα συνάρτησης**: Ισχύουν οι ίδιοι κανόνες ονοματολογίας με τις μεταβλητές.

(γ) **Παράμετροι**: Μεταβλητές που επιτρέπουν το πέρασμα της τιμής τους από ένα τμήμα προγράμματος σε άλλο. Μία συνάρτηση μπορεί να έχει μηδέν ή περισσότερες παραμέτρους.

Στο πιο πάνω παράδειγμα η επικεφαλίδα της συνάρτησης είναι η εξής:

```
int add_2_nums(int a, int b){
```

Από την επικεφαλίδα συμπεραίνουμε τα εξής:

- Η συνάρτηση θα επιστρέφει έναν ακέραιο αριθμό γιατί είναι τύπου `integer`
- Το όνομα της συνάρτησης είναι `add_2_nums`
- Η συνάρτηση δέχεται δύο παραμέτρους (a, b) ως ακέραιες τιμές

Η κλήση της συνάρτησης στο πιο πάνω παράδειγμα γίνεται στην κύρια συνάρτηση ως εξής:

```
cout << add_2_nums(65,97);        // Αποτέλεσμα: 162
```

Όταν ο μεταγλωττιστής συναντήσει το όνομα μίας συνάρτησης μέσα στην κύρια συνάρτηση (main), εντοπίζει την επικεφαλίδα της, εκτελεί όλες τις εντολές που περιλαμβάνονται στο σώμα της συνάρτησης, επιστρέφει την τιμή που πρέπει και συνεχίζει με την κανονική ροή του προγράμματος. Η συνάρτηση, όταν καλείται μέσα στην κύρια συνάρτηση (main), πρέπει να έχει το ίδιο όνομα, το ίδιο πλήθος και τον ίδιο τύπο παραμέτρων.

Κάθε συνάρτηση βασικού τύπου χρησιμοποιεί τη δεσμευμένη λέξη `return`, για να επιστρέψει μία τιμή στην κύρια συνάρτηση (main). Με την εκτέλεση της εντολής `return` η συνάρτηση τερματίζει. Όταν έχουμε περισσότερες εντολές `return` μέσα στη συνάρτηση, θα εκτελεστεί μόνο μία από αυτές. Ας δούμε ένα παράδειγμα:

Παράδειγμα 4.1

Να δημιουργήσετε πρόγραμμα, το οποίο να καλεί μία συνάρτηση η οποία να δέχεται παραμετρικά δύο ακέραιους αριθμούς και να επιστρέφει στην κύρια συνάρτηση (main) τον μεγαλύτερο από τους δύο. Να θεωρήσετε ότι οι δύο ακέραιοι που θα δοθούν δεν θα είναι ίσοι.

```
#include <iostream>
using namespace std;

int max_num(int a, int b) {       // Ορισμός συνάρτησης
  if (a>b)                        // Αν a>b η συνάρτηση θα επιστρέψει
     return a;                    // το a και θα τερματίσει,
return b;                         // αλλιώς θα επιστρέψει το b. Το
}                                 // else μπορούμε να το παραβλέψουμε
```

```
int main() {                              // Κύρια συνάρτηση (main)
      int num1, num2;
      cin >> num1 >> num2;                // Καταχώριση τιμών
      cout << max_num(num1, num2);        // Εμφάνιση μέγιστου
return 0;
}                                         // Τέλος κύριας συνάρτησης
```

(Code: C7_4_example1.cpp)

4. Εναλλακτικός ορισμός συναρτήσεων

Εναλλακτικά, μπορούμε να δηλώσουμε μόνο το πρότυπο της συνάρτησης πριν από την κύρια συνάρτηση (main) και ο ορισμός της συνάρτησης να ακολουθήσει μετά το τέλος του προγράμματος. Αυτό μάς επιτρέπει να χρησιμοποιήσουμε τη συνάρτηση στο πρόγραμμα πριν ακόμα αυτή οριστεί. Ας δούμε το παράδειγμα 4.1 με τη χρήση του εναλλακτικού ορισμού:

```
#include <iostream>
using namespace std;

int max_num(int a, int b);          // Πρότυπο συνάρτησης

int main() {                        // Κύρια συνάρτηση (main)
      int num1, num2;
      cin >> num1 >> num2;                // Καταχώριση τιμών
      cout << max_num(num1, num2);        // Εμφάνιση μέγιστου
return 0;
}                                   // Τέλος κύριας συνάρτησης

int max_num(int a, int b){          // Ορισμός συνάρτησης max_num
  if (a>b)                          // Σώμα συνάρτησης
      return a;
return b;
}                                   // Τέλος συνάρτησης max_num
```

5. Τοπικές (local) και καθολικές (global) μεταβλητές

Μέσα σε μία συνάρτηση μπορούμε να δηλώσουμε τοπικές μεταβλητές. Οι τοπικές μεταβλητές μπορούν να χρησιμοποιηθούν μέσα σε μία δηλωμένη συνάρτηση, όμως δεν είναι προσβάσιμες από την κύρια συνάρτηση (main). Αντίθετα, μία καθολική μεταβλητή η οποία δηλώνεται εκτός της κύριας συνάρτησης, μπορεί να είναι προσβάσιμη από την κύρια συνάρτηση αλλά και από όσες συναρτήσεις ορίζονται σε αυτήν. Ας δούμε ένα παράδειγμα:

```
// Το πρόγραμμα αυτό βρίσκει το συνολικό άθροισμα όλων των ψηφίων
// για άγνωστο πλήθος ακεραίων, καθώς και το πλήθος των ακεραίων.
// Κάνει χρήση συνάρτησης η οποία υπολογίζει το άθροισμα των ψηφίων
// ενός ακεραίου. Η είσοδος τερματίζει όταν δοθεί ο αριθμός 0
```

```
#include <iostream>
using namespace std;
int cnt = 0;          // Δήλωση καθολικής μεταβλητής - μετρητή
int sum_of_digits(int x) {            // Ορισμός συνάρτησης
    int temp = x, sum = 0;            // Τοπικές μεταβλητές
    while (temp>0){                   // Εύρεση αθροίσματος
        sum += temp%10;               // ψηφίων ακεραίου
        temp /= 10;
     }
    cnt++;            // Αύξηση μετρητή μέσα στη συνάρτηση
return sum;           // Επιστροφή αθροίσματος ψηφίων ακεραίου
}
int main() {               // Κύρια συνάρτηση (main)
     int num, total = 0;   // Τοπικές μεταβλητές κύριας συνάρτησης
     cin >> num;
     while (num!=0){       // Η είσοδος τερματίζει όταν δοθεί το 0
        total += sum_of_digits(num); // Εύρεση συνολικού αθροίσματος
        cin >> num;
      }
  cout << total << " " << cnt;        // Εμφάνιση αποτελεσμάτων
return 0;
}
```

Παράδειγμα 4.2

Να δημιουργήσετε πρόγραμμα, το οποίο να υπολογίζει την τιμή της εξίσωσης x=y!+z! **Σημείωση:** Παραγοντικό (!) ενός αριθμού Ν ορίζεται το γινόμενο 1*2*3*…(N-1)*N.

Π.χ. 5!=1*2*3*4*5=120. Το πρόγραμμα να υλοποιηθεί αρχικά χωρίς τη χρήση συνάρτησης.

Λύση 1: Χωρίς τη χρήση συνάρτησης

```
#include <iostream>
using namespace std;
int main() {                          // Κύρια συνάρτηση (main)
  int y, z, y_par = 1, z_par = 1;
  cin >> y >> z;
  for (int i=1; i<=y; i++)
        y_par *= i;                   // Υπολογισμός y!
  for (int j=1; j<=z; j++)
        z_par *= j;                   // Υπολογιαμός z!
  int x = y_par + z_par;              // Υπολογισμός x
  cout << x << endl;                  // Εμφάνιση αποτελέσματος
return 0;
}
```

Λύση 2: Με τη χρήση συνάρτησης

```
#include <iostream>
using namespace std;

int factorial (int N) {                  // Ορισμός συνάρτησης
  int ans = 1;                           // Τοπική μεταβλητή
  for (int i=1; i<=N; i++)
       ans *= i;
return ans;                              // Επιστροφή αποτελέσματος
}

int main() {                             // Κύρια συνάρτηση (main)
  int y, z;
  cin >> y >> z;
  int x = factorial(y) + factorial(z);   // Υπολογισμός x = y! + z!
  cout << x << endl;                     // Εμφάνιση αποτελέσματος
return 0;
}
```

(Code: C7_4_example2.cpp)

6. Παράμετροι τιμών (by value) και παράμετροι αναφοράς (by reference)

Οι συναρτήσεις με τη χρήση της εντολής `return` μπορούν να επιστρέψουν μόνο μία τιμή. Σε περίπτωση που θέλουμε η συνάρτησή μας να «επηρεάσει» περισσότερες από μία μεταβλητές, χρησιμοποιούμε την κλήση παραμέτρων με αναφορά. Με αυτό τον τρόπο περνάμε στη συνάρτηση τη διεύθυνση της μεταβλητής, με τη χρήση του τελεστή **&** και όχι την τιμή της.

6.1 Χρήση του τελεστή &

```
int x = 5;            // Δηλώνουμε μία μεταβλητή με το όνομα x
int &y = x;           // Η y δηλώνεται σαν μεταβλητή αναφοράς της x
cout << y << endl;    // 5 – Η τιμή της y είναι η ίδια με της x
cout << &y << endl;   // 0x69fee8 (διεύθυνση μνήμης, ίδια με την &x)
y  = 10;              // Αλλαγή της τιμής της y θα αλλάξει και την x
cout << x << endl;    // 10 – Η τιμή της x έχει αλλάξει
```

Η θέση του τελεστή δεν παίζει κανέναν ρόλο. Όλες οι πιο κάτω εντολές είναι έγκυρες:

```
int x = 5;       // Δηλώνουμε μία μεταβλητή με όνομα x και τιμή 5
int &y1 = x;     // Μεταβλητές αναφοράς της μεταβλητής x
int& y2 = x;
int & y3 = x;
```

Οι y1, y2, y3 δηλώνουν μεταβλητές αναφοράς για τη μεταβλητή x.

Ας δούμε ένα παράδειγμα με τη χρήση παραμέτρων αναφοράς:

Παράδειγμα 4.3

```
#include<iostream>
using namespace std;

// Συνάρτηση τύπου void. Δεν επιστρέφει τιμή – δεν υπάρχει return
void byvalue(int x){              // Παράμετρος τιμής
      x = x + 1;                  // Αλλαγή τιμής παραμέτρου
      cout << x << endl;          // Εντολή εξόδου
}
// Συνάρτηση τύπου void. Δεν επιστρέφει τιμή – δεν υπάρχει return
void byreference(int &y){   // Παράμετρος αναφοράς με τη χρήση του &
      y = y + 1;            // Η τιμή της μεταβλητής θα αλλάξει
      cout << y << endl;    // και μέσα στην κύρια συνάρτηση
}

int main() {                // Κύρια συνάρτηση (main)
  int num = 5;
  byvalue(num);                              // 6
  cout << "number=" << num << endl;          // number=5
  byreference(num);                          // 6
  cout << "number=" << num << endl;          // number=6
return 0;
}
```

(Code: C7_4_example3.cpp)

Παράδειγμα 4.4

Να δημιουργήσετε πρόγραμμα, το οποίο να καλεί μία συνάρτηση με το όνομα swap, η οποία να δέχεται δύο παραμέτρους αναφοράς και να αντιμεταθέτει τις τιμές τους. Το πρόγραμμα αρχικά θα διαβάζει δύο ακέραιους αριθμούς και θα τους αποθηκεύει στις μεταβλητές x και y. Στη συνέχεια, θα καλεί τη συνάρτηση swap η οποία θα δίνει στη μεταβλητή x την τιμή της μεταβλητής y και στην μεταβλητή y την τιμή της μεταβλητής x. Τέλος, να εμφανίζει τις τιμές των μεταβλητών στην οθόνη.

Παράδειγμα εισόδου

```
13 19
```

Παράδειγμα εξόδου

```
x=19
y=13
```

```
#include<iostream>
using namespace std;
// Συνάρτηση τύπου void
void swap(int &a, int &b){       // Παράμετροι αναφοράς (&)
      int temp;                  // Τοπική μεταβλητή temp
      temp = a;                  // Τεχνική αντιμετάθεσης δύο τιμών
      a = b;                     // με τη χρήση μίας επιπρόσθετης
      b = temp;                  // μεταβλητής
}
int main(){                      // Κύρια συνάρτηση (main)
      int x, y;
      cin >> x >> y;             // Καταχώριση δεδομένων
      swap(x, y);                // Κλήση συνάρτησης swap
      cout << "x=" << x << endl; // Εμφάνιση αποτελεσμάτων
      cout << "y=" << y << endl;
return 0;
}
```

(Code: C7_4_example4.cpp)

Παράδειγμα 4.5

Να δημιουργήσετε πρόγραμμα, το οποίο να καλεί μία συνάρτηση με το όνομα rectangle, η οποία να δέχεται παραμετρικά το μήκος και το πλάτος ενός ορθογωνίου και να επιστρέφει με παραμέτρους αναφοράς στην κύρια συνάρτηση (main), το εμβαδόν και την περίμετρό του.

```
#include<iostream>
using namespace std;

// Συνάρτηση για υπολογισμό εμβαδού και περιμέτρου ορθογωνίου
void rectangle(int h, int w, int &E, int &P){
      E = h * w;
      P = (h + w) * 2;
}

int main(){                                  // Κύρια συνάρτηση (main)
      int mikos, platos, emv, per;           // Δήλωση μεταβλητών
      cin >> mikos >> platos;                // Καταχώριση δεδομένων
      rectangle(mikos, platos, emv, per);    // Κλήση συνάρτησης
      cout << "Emvadon=" << emv << endl;     // Εμφάνιση αποτελεσμάτων
      cout << "Perimetros=" << per << endl;
return 0;
}
```

(Code: C7_4_example5.cpp)

Παράδειγμα 4.6

Με τη χρήση προκαταρκτικής εκτέλεσης, να παρουσιάσετε τα αποτελέσματα του πιο κάτω προγράμματος.

```
#include <iostream>
using namespace std;

int check(int a, int &b){        // a – τιμής, b – αναφοράς (&)
 b += 2;
     if (a>b)
         return a;
return a + b;
}
int main(){
  int x = 3, y = 2;
  if (x>y)
      cout << check(x, y) << endl;
  else
      cout << check(y, x) << endl;

  cout << x << " " << y << endl;

return 0;
}
```

(Code: C7_4_example6.cpp)

Κύρια συνάρτηση (main)

Μεταβλητές		Αποφάσεις		Παρουσίαση
x	y	x>y	T/F	
3	2	3>2	T	
	4			7
				3□4

Συνάρτηση check

Τυπικές Παράμετροι Τιμών	Τυπικές Παράμετροι Αναφοράς	Αποφάσεις		Επιστρέφει
a	b	a>b	T/F	
3	2			
	4	3>4	F	7

7. Πίνακες και συναρτήσεις

Ένας πίνακας αποτελείται από περισσότερες από μία τιμές, οπότε για να περάσουμε έναν πίνακα σε μία συνάρτηση ως παράμετρο, το πρότυπο της συνάρτησης θα περιλαμβάνει τον πίνακα ως παράμετρο αναφοράς (δηλαδή θα περάσουμε τη διεύθυνση του πίνακα). **Στις παραμέτρους αναφοράς με πίνακες δεν χρησιμοποιείται ο τελεστής &.** Επιπρόσθετα, το μέγεθος του πίνακα δεν ορίζεται παραμετρικά στο πρότυπο της συνάρτησης, ενώ στην κλήση της συνάρτησης δηλώνουμε μόνο το όνομα του πίνακα.

Παράδειγμα 4.7

Να δημιουργήσετε πρόγραμμα, το οποίο να χρησιμοποιεί τις συναρτήσεις fill για να γεμίζει και τη συνάρτηση print για να εμφανίζει έναν μονοδιάστατο πίνακα ακεραίων μεγέθους 20 θέσεων.

```
#include<iostream>
#define N 20                  // Σταθερά για το μέγεθος του πίνακα
using namespace std;

void fill(int arr[]);         // Πρότυπο συνάρτησης fill
void print(int arr[]);        // Πρότυπο συνάρτησης print

int main(){                   // Κύρια συνάρτηση (main)
   int A[N];                  // Δήλωση πίνακα ακεραίων 20 θέσεων
   fill(A);                   // Καταχώριση στοιχείων
   print(A);                  // Εμφάνιση στοιχείων
return 0;
}                             // Τέλος κύριας συνάρτησης
void fill(int arr[]){         // Ορισμός συνάρτησης fill
   for(int i=0; i<N; i++)
       cin >> arr[i];
}                             // Τέλος συνάρτησης fill
void print(int arr[]){        // Ορισμός συνάρτησης print
   for(int i=0; i<N; i++)
       cout << arr[i] << " ";
}                             // Τέλος συνάρτησης print
```

(Code: C7_4_example7.cpp)

Παράδειγμα 4.8

Να δημιουργήσετε πρόγραμμα, το οποίο να υλοποιεί και να καλεί:

(α) τη συνάρτηση read για να διαβάσει και να καταχωρίσει τους βαθμούς και τα ονόματα 25 μαθητών σε πίνακες τους οποίους θα δέχεται παραμετρικά

(β) τη συνάρτηση average για να υπολογίσει και να εμφανίσει τον μέσο όρο των βαθμών. Η συνάρτηση θα δέχεται παραμετρικά τον πίνακα των βαθμών και θα επιστρέφει τον μέσο όρο στην κύρια συνάρτηση (main), όπου και θα εμφανίζεται με ένα δεκαδικό ψηφίο

(γ) τη συνάρτηση max_grade για να βρει το όνομα του μαθητή με τον ψηλότερο βαθμό. Η συνάρτηση θα δέχεται παραμετρικά τους πίνακες των βαθμών και των ονομάτων και θα επιστρέφει στην κύρια συνάρτηση (main) το όνομα του μαθητή με τον υψηλότερο βαθμό. Να θεωρήσετε ότι δεν θα υπάρχει ισοβαθμία βαθμών.

```
#include<iostream>
#include<iomanip>
#include<string>
#define N 25
using namespace std;

void read(int grades[], string names[]){    // Συνάρτηση read
    for(int i=0; i<N; i++)
        cin >> grades[i] >> names[i];
}
double average(int grades[]){               // Συνάρτηση average
    int sum = 0;                            // Τοπική μεταβλητή sum
    for(int i=0; i<N; i++)
        sum += grades[i];
return (double) sum / N;
}
string max_grade(int grades[],string names[]){ //Συνάρτηση max_grade
  int maxg = grades[0];     // Τοπικές μεταβλητές τις οποίες
  string maxn = names[0];   // αρχικοποιούμε με τα πρώτα στοιχεία
    for(int i=1; i<N; i++){      // Υπολογισμός μέγιστου βαθμού
        if(grades[i] > maxg){
           maxg = grades[i];     // Μέγιστος βαθμός
           maxn = names[i];      // Όνομα μαθητή με μέγιστο βαθμό
        }
    }
return maxn;                     // Επιστροφή ονόματος
}

int main(){                      // Κύρια συνάρτηση (main)
  int vathmoi[N];                // Δήλωση πινάκων
  string onomata[N];
  read(vathmoi, onomata);        // Κλήση συναρτήσεων
  cout<<"Avg="<<fixed<<setprecision(1)<<average(vathmoi)<<endl;
  cout<<"Best student="<<max_grade(vathmoi,onomata)<<endl;
return 0;
}
```

(Code: C7_4_example8.cpp)

8. Αρχεία και συναρτήσεις

Αν θέλουμε να χρησιμοποιήσουμε αρχεία σε συνδυασμό με συναρτήσεις, τότε οι εντολές που χρησιμοποιούμε για δημιουργία ροής ανάγνωσης (ifstream) και ροής εγγραφής (ofstream) μπορούν να δηλωθούν εκτός της κύριας συνάρτησης. Με αυτό τον τρόπο, οι εντολές αυτές θα είναι προσβάσιμες από όσες συναρτήσεις χρησιμοποιήσουμε στο πρόγραμμα.

Παράδειγμα 4.9

Να δημιουργήσετε πρόγραμμα, το οποίο να:

(α) χρησιμοποιεί τη συνάρτηση fill η οποία να δέχεται ως παράμετρο έναν πίνακα ακεραίων και να τον γεμίζει με τα περιεχόμενα του αρχείου numbers.txt. Το αρχείο numbers.txt θα αποτελείται από N (1<=N<=100), ακέραιους αριθμούς στο διάστημα [1…10000]

(β) χρησιμοποιεί τη συνάρτηση square η οποία να δέχεται παραμετρικά έναν ακέραιο αριθμό και να επιστρέφει `true`, αν ο αριθμός είναι τέλειο τετράγωνο ενός ακεραίου (π.χ. 16, 49), αλλιώς να επιστρέφει `false`

(γ) χρησιμοποιεί τη συνάρτηση print η οποία να καταχωρίζει στο αρχείο squares.txt τα στοιχεία του πίνακα, τα οποία είναι τέλεια τετράγωνα ακέραιων αριθμών.

Παράδειγμα εισόδου (numbers.txt)

```
13 19 25 41 82 16 49 54 112 90
```

Παράδειγμα εξόδου (squares.txt)

```
25 16 49
```

```
#include<fstream>                  // Για τη χρήση αρχείων
#include<cmath>                    // Για τη χρήση της sqrt
using namespace std;
// Οι ροές δηλώνονται εκτός της κύριας συνάρτησης main
ifstream fin("numbers.txt");       // Δημιουργία ροής ανάγνωσης
ofstream fout("squares.txt");      // Δημιουργία ροής εγγραφής
int cnt = 0;     // Καθολική μεταβλητή για το πλήθος των ακεραίων

void fill(int nums[]){       // Συνάρτηση καταχώρισης ακεραίων
    while(!fin.eof()){       // στον πίνακα, από το αρχείο numbers.txt
       fin >> nums[cnt];     // Διαβάζει και καταχωρίζει έναν ακέραιο
       cnt++;                // Αύξηση του μετρητή
    }
}
bool square(int n){   // Συνάρτηση για εντοπισμό τέλειου τετραγώνου
   int x = (int)(sqrt(n) + 0.5);
   if (x*x == n)
       return true;   // Επιστρέφει true αν έχουμε τέλειο τετράγωνο
return false;
}
```

```
void print(int nums[]){     // Συνάρτηση εγγραφής στο squares.txt
  for(int i=0; i<cnt; i++)
      if(square(nums[i]))   // Κλήση συνάρτησης εντοπισμού
         fout << nums[i] << " ";
}

int main() {                // Κύρια συνάρτηση (main)

   int numbers[100];        // Καθορισμός μεγέθους με βάση το μέγιστο
   fill(numbers);           // Συνάρτηση ανάγνωσης αρχείου
   print(numbers);          // Συνάρτηση εγγραφής σε αρχείο η οποία
                            // καλεί εσωτερικά τη συνάρτηση square
   fin.close();             // Κλείσιμο ροής ανάγνωσης
   fout.close();            // Κλείσιμο ροής εγγραφής

return 0;                   // Τέλος κύριας συνάρτησης
}
```

(Code: C7_4_example9.cpp)

Ασκήσεις Κεφαλαίου - www.hackerrank.com/g74

Άσκηση 4.1

Να συμπληρώσετε τις εντολές για την πιο κάτω συνάρτηση, η οποία να εμφανίζει το μήνυμα «Hello world» στην οθόνη.

```
void say_hello( ){



}
```

Άσκηση 4.2

Να συμπληρώσετε τις εντολές για την πιο κάτω συνάρτηση, η οποία να εμφανίζει το μήνυμα «Hello world» στην οθόνη, όσες φορές θα υποδεικνύει η παράμετρος x που θα δοθεί.

```
void say_hello_x_times(int x){




}
```

Άσκηση 4.3

Να συμπληρώσετε μέσα στην κύρια συνάρτηση (main), την κλήση της συνάρτησης FirstLetter, η οποία τυπώνει τον πρώτο χαρακτήρα μίας συμβολοσειράς.

```
#include <iostream>
#include <string>
using namespace std;

char FirstLetter(string st){
    return st[0];
}

int main(){
    string s;
    char c;
    cin >> s;
    ________________________
    cout << c;
return 0;
}
```

Άσκηση 4.4

Να συμπληρώσετε τις παραμέτρους της συνάρτησης AverageLastDigit που υπολογίζει τον μέσο όρο των ψηφίων των μονάδων 3 ακεραίων αριθμών.

```
#include<iostream>
using namespace std;
float AverageLastDigit(_____________________________){
    int d1, d2, d3, sum;
    d1 = x % 10;
    d2 = y % 10;
    d3 = z % 10;
    sum = d1 + d2 + d3;
return sum / 3.0;
}
int main(){
    int a, b, c;
    cin >> a >> b >> c;
    cout << AverageLastDigit(a, b, c);
return 0;
}
```

Άσκηση 4.5

Να συμπληρώσετε την επικεφαλίδα της συνάρτησης (A) που χρησιμοποιείται στην κύρια συνάρτηση (main) για να γεμίσει τον πίνακα arr, καθώς και την επικεφαλίδα της συνάρτησης (B) που χρησιμοποιείται για τον υπολογισμό του αθροίσματος των στοιχείων του πίνακα A.

```
#include<iostream>
#define N 30
using namespace std;
_____________________________ {       // A
    int sum = 0, i;
    for(i=0; i<N; i++)
        sum += b[i];
 return sum;
}
_____________________________ {       // B
    int i;
    for(i=0; i<N; i++){
        cout << "arr[" << i << "]:";
        cin >> b[i];
    }
}
```

```
int main(){
    int arr[N];
    fillArray(arr);
    cout << "Sum of array = " << sumArray(arr);
return 0;
}
```

Άσκηση 4.6

Με τη χρήση προκαταρκτικής εκτέλεσης, να παρουσιάσετε τα αποτελέσματα της πιο κάτω συνάρτησης, αν δοθεί ως παράμετρος ο αριθμός 100.

```
int hundreds (int x){
  if (x/100 > 1)
     return 1;
  if (x%100 == 1)
     return 0;
return -1;
}
```

Άσκηση 4.7

Με τη χρήση προκαταρκτικής εκτέλεσης, να παρουσιάσετε τα αποτελέσματα του πιο κάτω προγράμματος.

```
#include<iostream>
using namespace std;

int f1(int x, int &y){
    int s;
    s = x + y;
    x = s / 10;
    y = s % 10;
    cout << x << endl << y << endl;
return x+y;
}
int main(){
    int a = 11, b = 12, c;
    c = f1(a, b);
    cout << a << endl;
    cout << b << endl;
    cout << c << endl;
return 0;
}
```

Άσκηση 4.8

Με τη χρήση προκαταρκτικής εκτέλεσης, να παρουσιάσετε τα αποτελέσματα του πιο κάτω προγράμματος.

```
#include <iostream>
using namespace std;

void change(int a, int &b, int &c){
     a += 4;
     b += 5;
     c += 6;
     cout << a << " " << b << " " << c << endl;
}
int main(){
  int x = 2, y = 1, z = 3;
   change(z, y, x);
   cout << x << " " << y << " " << z << endl;
   change(y, x, z);
   cout << x << " " << y << " " << z << endl;
return 0;
}
```

Άσκηση 4.9

Το πιο κάτω πρόγραμμα, το οποίο υπολογίζει το πλήθος των άρτιων (ζυγών) αριθμών ενός πίνακα 10 ακεραίων, έχει 4 λάθη. Να τα εντοπίσετε και να τα διορθώσετε.

```
#include<iostream>
#define N 10
using namespace std;
int even(int B[]);

int main(){
    int A[N];
    cout << "Even numbers:" << even(A[]) << endl;
return 0;
}
void fillArray(int B[]){
    int i;
    for(i=0; i<N; i++){
        cout << "A[" << i << "]:";
        cin >> A[i];
    }
}
```

```
int even(int B[]){
    int i, p = 0;
    fillArray(A);
    for(i=0; i<N; i++)
        if(B[i]%2==0)
            p++;
return p;
}
```

Άσκηση 4.10

Να δημιουργήσετε πρόγραμμα, το οποίο να καλεί μία συνάρτηση με το όνομα next, η οποία να δέχεται παραμετρικά έναν ακέραιο αριθμό και να επιστρέφει τον επόμενό του.

Πρότυπο συνάρτησης: `int next(int x);`

Παράδειγμα εισόδου

```
7
```

Παράδειγμα εξόδου

```
8
```

Άσκηση 4.11

Να δημιουργήσετε πρόγραμμα, το οποίο να καλεί μία συνάρτηση με το όνομα prod, η οποία να δέχεται παραμετρικά δύο πραγματικούς αριθμούς και να επιστρέφει το γινόμενό τους, το οποίο να εμφανίζεται στην οθόνη από την κύρια συνάρτηση (main), με δύο δεκαδικά ψηφία.

Πρότυπο συνάρτησης: `float prod(float a, float b);`

Παράδειγμα εισόδου

```
2.3 6.8
```

Παράδειγμα εξόδου

```
15.64
```

Άσκηση 4.12

Να δημιουργήσετε πρόγραμμα, το οποίο να καλεί μία συνάρτηση με το όνομα caps, η οποία να δέχεται παραμετρικά έναν χαρακτήρα και να επιστρέφει `true`, αν ο χαρακτήρας είναι κεφαλαίο γράμμα, αλλιώς να επιστρέφει `false`.

Πρότυπο συνάρτησης: `bool caps(char letter);`

Άσκηση 4.13

Να δημιουργήσετε πρόγραμμα, το οποίο να καλεί μία συνάρτηση με το όνομα mod, η οποία να δέχεται παραμετρικά δύο ακεραίους αριθμούς και να επιστρέφει το υπόλοιπο της διαίρεσης του μεγαλύτερου με τον μικρότερο αριθμό. Οι αριθμοί μπορούν να δοθούν με οποιαδήποτε σειρά.

Πρότυπο συνάρτησης: `int mod(int a, int b);`

Άσκηση 4.14

Να δημιουργήσετε πρόγραμμα, το οποίο να καλεί μία συνάρτηση με το όνομα power, η οποία να δέχεται παραμετρικά δύο ακέραιους αριθμούς (x, y) και να επιστρέφει τη δύναμη x^y.

Πρότυπο συνάρτησης: `int power(int x, int y);`

Άσκηση 4.15

Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται έναν ακέραιο αριθμό και χρησιμοποιώντας τη συνάρτηση prime να ελέγχει αν ο αριθμός είναι πρώτος ή όχι. Πρώτος αριθμός θεωρείται αυτός που έχει ως διαιρέτες μόνο τον εαυτό του και τον αριθμό ένα, π.χ. οι αριθμοί 3, 5, 11, 17. Η συνάρτηση θα επιστρέφει `true` αν ο αριθμός είναι πρώτος, αλλιώς θα επιστρέφει `false`.

Πρότυπο συνάρτησης: `bool prime(int n);`

Άσκηση 4.16

Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται έναν ακέραιο αριθμό και έναν χαρακτήρα και να χρησιμοποιεί τη συνάρτηση triangle, για να τυπώσει ένα ορθογώνιο τρίγωνο που θα έχει N γραμμές και θα σχηματίζεται από τον χαρακτήρα C.

Πρότυπο συνάρτησης: `void triangle(int n, char c);`

Παράδειγμα εισόδου

```
3 @
```

Παράδειγμα εξόδου

```
@
@@
@@@
```

Άσκηση 4.17

Να δημιουργήσετε πρόγραμμα, το οποίο να χρησιμοποιεί τη συνάρτηση rect, η οποία να δέχεται παραμετρικά το μήκος και το πλάτος ενός ορθογωνίου, να υπολογίζει και να επιστρέφει, ως παραμέτρους αναφοράς, το εμβαδόν και την περίμετρο του ορθογωνίου.

Πρότυπο συνάρτησης: `void rect(double w,double l,double &e,double &p);`

Άσκηση 4.18

Τέλειος καλείται ένας ακέραιος αριθμός όταν το άθροισμα των θετικών διαιρετών του, εκτός του ιδίου, είναι ίσο με τον αριθμό. Για παράδειγμα, ο αριθμός 28 είναι τέλειος αριθμός. Οι διαιρέτες του 28 είναι οι 1, 2, 4, 7, 14 και το άθροισμα αυτών είναι ίσο με 28 (1+2+4+7+14=28). Να δημιουργήσετε πρόγραμμα, το οποίο να καλεί τη συνάρτηση perfect, η οποία να δέχεται παραμετρικά έναν ακέραιο θετικό αριθμό και να επιστρέφει το μήνυμα «perfect» σε περίπτωση που ο αριθμός είναι τέλειος, ενώ, σε αντίθετη περίπτωση, να επιστρέφει το μήνυμα «not perfect».

Άσκηση 4.19

Να δημιουργήσετε πρόγραμμα, το οποίο να καλεί μία συνάρτηση με το όνομα pyramid, η οποία να δέχεται το ύψος h (1<h<30) μίας πυραμίδας και να τυπώνει το κατάλληλο σχήμα, όπως το πιο κάτω παράδειγμα.

Παράδειγμα εισόδου

```
6
```

Παράδειγμα εξόδου

```
     *
    ***
   *****
  *******
 *********
***********
```

Άσκηση 4.20

Να δημιουργήσετε πρόγραμμα, το οποίο να καλεί μία συνάρτηση με το όνομα positive, η οποία να δέχεται παραμετρικά έναν πίνακα ακεραίων αριθμών πέντε θέσεων και να επιστρέφει `true`, αν όλα τα στοιχεία του πίνακα είναι θετικοί ακέραιοι αριθμοί, αλλιώς να επιστρέφει `false`. Η καταχώριση των στοιχείων θα γίνεται στην κύρια συνάρτηση (main).

Παράδειγμα εισόδου

```
5 6 8 11 -12
```

Παράδειγμα εξόδου

```
false
```

Άσκηση 4.21

Να δημιουργήσετε πρόγραμμα, το οποίο να διαβάζει 5 συμβολοσειρές από το αρχείο in.txt και να χρησιμοποιεί τη συνάρτηση `string reverseString(string st)`, για να αντιστρέψει κάθε μία από τις συμβολοσειρές. Οι ανεστραμμένες συμβολοσειρές να τυπώνονται στο αρχείο out.txt.

Παράδειγμα εισόδου (in.txt)

```
abc
pacman
1234
A
madam
```

Παράδειγμα εξόδου (out.txt)

```
cba
namcap
4321
A
madam
```

Άσκηση 4.22

Να δημιουργήσετε πρόγραμμα, το οποίο να καλεί μία συνάρτηση με όνομα search, η οποία να δέχεται παραμετρικά μία συμβολοσειρά και έναν χαρακτήρα και να επιστρέφει το πλήθος των εμφανίσεων του χαρακτήρα μέσα στη συμβολοσειρά.

Παράδειγμα εισόδου

```
abracadabra a
```

Παράδειγμα εξόδου

```
5
```

Άσκηση 4.23

Να δημιουργήσετε πρόγραμμα, το οποίο να καλεί μία συνάρτηση με όνομα fill, η οποία να δέχεται παραμετρικά έναν πίνακα χαρακτήρων και έναν ακέραιο αριθμό (το μέγεθος του πίνακα). Το πρόγραμμα θα δέχεται τα μεγέθη για τρεις πίνακες και θα καλεί τη συνάρτηση fill τρεις φορές, για να γεμίσει με χαρακτήρες και τους τρεις πίνακες. Το πρόγραμμα να εμφανίζει στην οθόνη τον χαρακτήρα της τελευταίας θέσης του κάθε πίνακα.

Παράδειγμα εισόδου

```
3 4 5
A B C
D E F G
H I J K L
```

Παράδειγμα εξόδου

```
C
G
L
```

Άσκηση 4.24

Να δημιουργήσετε πρόγραμμα, το οποίο να καλεί μία συνάρτηση με όνομα min3, η οποία να δέχεται παραμετρικά τρεις ακέραιους αριθμούς και να επιστρέφει τον μικρότερο από τους τρεις.

Παράδειγμα εισόδου

```
12 5 8
```

Παράδειγμα εξόδου

```
5
```

Άσκηση 4.25

Να δημιουργήσετε πρόγραμμα, το οποίο να καλεί μία συνάρτηση με το όνομα shift_right, η οποία να δέχεται παραμετρικά έναν πίνακα ακεραίων έξι θέσεων και έναν ακέραιο αριθμό N (0<=N<=6). Η συνάρτηση να μετακινεί κάθε στοιχείο του πίνακα N θέσεις δεξιά. Η καταχώριση των στοιχείων για τον πίνακα θα γίνεται στην κύρια συνάρτηση (main).

Παράδειγμα εισόδου

```
15 26 32 44 51 68
3
```

Παράδειγμα εξόδου

```
44 51 68 15 26 32
```

Άσκηση 4.26

Να δημιουργήσετε πρόγραμμα, το οποίο να καλεί μία συνάρτηση με όνομα palindrome, η οποία να δέχεται παραμετρικά μία συμβολοσειρά και να επιστρέφει `true`, αν η συμβολοσειρά είναι παλίνδρομη, αλλιώς να επιστρέφει `false`. Παλίνδρομη καλείται η λέξη που διαβάζεται το ίδιο και από δεξιά προς αριστερά, όπως: eve, madam, deified.

Παράδειγμα εισόδου

```
madam
```

Παράδειγμα εξόδου

```
true
```

Άσκηση 4.27

Στη Φλουρεντζούπολη, ο δήμαρχος έχει διατάξει να γίνει άμεσα απογραφή του πληθυσμού. Για τον λόγο αυτό έχει απευθυνθεί στο γραφείο προγραμματισμού του δήμου του και έχει ζητήσει να δημιουργηθεί πρόγραμμα το οποίο να:

(α) χρησιμοποιεί τη συνάρτηση FillData, η οποία να δέχεται τις ηλικίες των 8921 κατοίκων της πόλης, από το αρχείο data.txt και να τις αποθηκεύει στον πίνακα ages

(β) χρησιμοποιεί τη συνάρτηση AboveAverage, για να επιστρέψει στην κύρια συνάρτηση (main) το πλήθος των κατοίκων, των οποίων η ηλικία τους είναι μεγαλύτερη από τον μέσο όρο ηλικίας των κατοίκων της Φλουρεντζούπολης. Το αποτέλεσμα να εμφανίζεται στην οθόνη

(γ) χρησιμοποιεί τη συνάρτηση Voters, για να επιστρέψει στην κύρια συνάρτηση (main) το πλήθος των ψηφοφόρων. Ψηφοφόρος θεωρείται κάποιος που είναι άνω των 17 ετών. Το αποτέλεσμα να εμφανίζεται στην οθόνη.

Άσκηση 4.28

Ο Μυριάνθης είναι ιδιοκτήτης της γνωστής αλυσίδας εστιατορίων «Το εκλεκτό κουπέπι», με 3 εστιατόρια σε όλη την Κύπρο. Ο Μυριάνθης θέλει να δημιουργήσει πρόγραμμα το οποίο να:

(α) χρησιμοποιεί τη συνάρτηση FillData, η οποία να δέχεται τις εισπράξεις για κάθε ένα από τα 3 εστιατόρια, για μία εβδομάδα και να τις αποθηκεύει στους πίνακες R1, R2 και R3. Οι εισπράξεις είναι πραγματικοί αριθμοί και βρίσκονται στο αρχείο in.txt

(β) χρησιμοποιεί τη συνάρτηση Calculate, η οποία να υπολογίζει το σύνολο των εισπράξεων για κάθε μέρα και να το αποθηκεύει στον πίνακα Total

(γ) χρησιμοποιεί τη συνάρτηση Print, για να καταχωρίσει τα στοιχεία του πίνακα Total μέσα στο αρχείο out.txt.

Άσκηση 4.29

Ο διάσημος περιβαλλοντολόγος δρ Julius εργάζεται στον μετεωρολογικό σταθμό Amundsen–Scott South Pole Station που βρίσκεται στον Νότιο Πόλο. Καθημερινά γίνονται μετρήσεις της θερμοκρασίας. Ο δρ Julius θέλει να δημιουργήσει πρόγραμμα το οποίο να:

(α) δέχεται τις θερμοκρασίες για έναν χρόνο (365 μέρες) και να τις αποθηκεύει στον πίνακα Τ. Οι θερμοκρασίες είναι πραγματικοί αριθμοί και βρίσκονται αποθηκευμένοι στο αρχείο temp.txt. Η λειτουργία αυτή πρέπει να εκτελείται από τη συνάρτηση Temperature

(β) χρησιμοποιεί τη συνάρτηση MaxT, για να καταχωρίσει στο αρχείο max.txt τη μέγιστη θερμοκρασία και τις μέρες που έχει καταγραφεί η μέγιστη θερμοκρασία

(γ) χρησιμοποιεί τη συνάρτηση Avg5, για να τυπώσει στο αρχείο average.txt τους μέσους όρους των θερμοκρασιών ανά 5 ημέρες. Δηλαδή θα τυπώσει τον μέσο όρο για τις μέρες 1 μέχρι και την 5, τον μέσο όρο για τις μέρες 6 μέχρι και την 10 κ.λπ.

Ασκήσεις Εμπλουτισμού

Άσκηση 4.30

Να δημιουργήσετε πρόγραμμα, το οποίο να καλεί μία συνάρτηση με το όνομα sumofdigits, η οποία να δέχεται ως παραμέτρους δύο μονοψήφιους ακέραιους αριθμούς a και b (1<=a, b<=9) και να επιστρέφει το άθροισμα όλων των διψήφιων αριθμών που σχηματίζονται με τα ψηφία από το a μέχρι και το b, συμπεριλαμβανομένων. Για παράδειγμα, η συνάρτηση sumofdigits(1,3) θα επιστρέφει τον αριθμό 198, δηλαδή το άθροισμα των αριθμών 11, 12, 13, 21, 22, 23, 31, 32, 33.

Παράδειγμα εισόδου

```
1 3
```

Παράδειγμα εξόδου

```
198
```

Άσκηση 4.31

Να δημιουργήσετε πρόγραμμα, το οποίο να καλεί μία συνάρτηση με το όνομα permutate, η οποία να δέχεται παραμετρικά μία συμβολοσειρά και το μέγεθός της και να εμφανίζει στην οθόνη όλες τις συμβολοσειρές που δημιουργούνται, αναδιατάσσοντας τους χαρακτήρες της συμβολοσειράς με όλους τους πιθανούς τρόπους.

Παράδειγμα εισόδου

```
abc
```

Παράδειγμα εξόδου

```
bac
bca
cba
cab
acb
abc
```

Άσκηση 4.32

Ο Παντελής αποφάσισε να ψάξει στο χωράφι του για φυσικό αέριο. Έχει εξασφαλίσει από την τράπεζα ένα αρχικό ποσό Μ (10,000<=M<=100,000), με το οποίο μπορεί να καλύψει τα έξοδα για τα πρώτα 100 μέτρα γεώτρησης που ανέρχονται στις €5,000. Το 101ο μέτρο θα κοστίζει επιπλέον €2.00 και για κάθε επιπρόσθετο μέτρο το κόστος θα αυξάνεται κατά 1%. Οι προκαταρκτικές μελέτες, όμως, έχουν δείξει ότι αν δεν βρεθεί φυσικό αέριο σε βάθος 500 μέτρων δεν υπάρχει λόγος να συνεχιστεί η γεώτρηση.

Να δημιουργήσετε πρόγραμμα, το οποίο να καλεί μία συνάρτηση με το όνομα drill, η οποία να δέχεται ως παράμετρο τιμής το αρχικό ποσό που έχει στη διάθεσή του ο Παντελής και να επιστρέφει ως παραμέτρους αναφοράς στην κύρια συνάρτηση (main):

(α) το κόστος για γεώτρηση βάθους 500 μέτρων, το οποίο να εμφανίζεται με δύο δεκαδικά ψηφία

(β) το μέγιστο βάθος σε μέτρα που μπορεί να φτάσει η γεώτρηση, με βάση το αρχικό διαθέσιμο ποσό.

Παράδειγμα εισόδου

```
30000
```

Παράδειγμα εξόδου

```
15504.82
586
```

Άσκηση 4.33

Να δημιουργήσετε πρόγραμμα, το οποίο να καλεί μία συνάρτηση με το όνομα find, η οποία να δέχεται παραμετρικά έναν πίνακα ακεραίων και το μέγεθός του και να επιστρέφει το μέγεθος της μέγιστης αύξουσας υπακολουθίας του πίνακα. Όπως υποδηλώνει το όνομά της, η μέγιστη αύξουσα υπακολουθία είναι η μεγαλύτερη δυνατή υπακολουθία αριθμών του πίνακα σε αύξουσα σειρά, όπου οι αριθμοί δεν είναι απαραίτητα συνεχόμενοι.

Παράδειγμα εισόδου

```
8
10 23 9 34 27 51 41 62
```

Παράδειγμα εξόδου

```
5
```

Επεξήγηση: Η μέγιστη αύξουσα υπακολουθία είναι η {10, 23, 34, 51, 62}, μεγέθους 5.

Γ7.5 Αλγόριθμοι Αναζήτησης (Searching Algorithms)

Τι θα μάθουμε σε αυτό το κεφάλαιο:

- Να ορίζουμε τι είναι αναζήτηση δεδομένων
- Να αναφέρουμε γνωστούς αλγόριθμους αναζήτησης
- Να περιγράφουμε σύντομα τι είναι η χρονική πολυπλοκότητα (execution time complexity) ενός αλγόριθμου ως συνάρτηση του μεγέθους των δεδομένων εισόδου, n, με βάση τον συμβολισμό *O*
- Να υπολογίζουμε πρακτικά τη χρονική πολυπλοκότητα απλών αλγορίθμων, αναγνωρίζοντας και μετρώντας τον αριθμό των εμφωλιασμένων βρόγχων τους, μέχρι $O(n^3)$
- Να αναγνωρίζουμε τον αλγόριθμο της μεθόδου της σειριακής αναζήτησης (sequential search) και τον αντίστοιχο κώδικα για υλοποίησή του σε προβλήματα που χρησιμοποιούν μονοδιάστατους πίνακες
- Να αναγνωρίζουμε τον αλγόριθμο της μεθόδου της δυαδικής αναζήτησης (binary search) και τον αντίστοιχο κώδικα για υλοποίησή του σε προβλήματα που χρησιμοποιούν μονοδιάστατους πίνακες
- Να υπολογίζουμε γενικά τη χρονική πολυπλοκότητα των δύο αυτών αλγορίθμων, μετρώντας τους βρόγχους που τους αποτελούν και τον αριθμό των επαναλήψεων
- Να συγκρίνουμε την ταχύτητα εκτέλεσης των δύο πιο πάνω αλγορίθμων αναζήτησης, μέσω αριθμητικών παραδειγμάτων, με βάση το θεωρητικό υπόβαθρο της χρονικής πολυπλοκότητάς τους, αλλά και επιβεβαιώνοντας τα αποτελέσματα πρακτικά, με εκτέλεση των αντίστοιχων προγραμμάτων στον ηλεκτρονικό υπολογιστή σε πραγματικό χρόνο με διάφορα μεγέθη εισόδου, n
- Να γνωρίζουμε παραλλαγές των αλγορίθμων αναζήτησης.

1. Εισαγωγή

Τα προβλήματα αναζήτησης είναι πολύ κοινά, τόσο στην καθημερινή ζωή όσο και στην επιστήμη της Πληροφορικής. Αλγόριθμος αναζήτησης θεωρείται ένας αλγόριθμος, ο οποίος προσπαθεί να εντοπίσει ένα στοιχείο με συγκεκριμένες ιδιότητες, μέσα σε μία συλλογή από στοιχεία του ίδιου τύπου. Στο κεφάλαιο αυτό, θα ασχοληθούμε με την αναζήτηση δεδομένων σε πίνακες και θα αναλύσουμε τους αλγόριθμους της σειριακής (sequential) και της δυαδικής (binary) αναζήτησης.

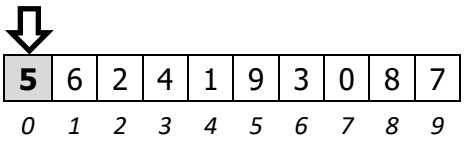
2. Σειριακή αναζήτηση (Sequential search)

Η σειριακή (γραμμική) αναζήτηση είναι ο ευκολότερος τρόπος αναζήτησης στοιχείων. Η σειριακή αναζήτηση, ξεκινώντας από το πρώτο στοιχείο του πίνακα, ελέγχει ένα-ένα τα στοιχεία μέχρι να εντοπίσει αυτό που ψάχνει ή να φτάσει στο τελευταίο στοιχείο του πίνακα. Γίνονται, δηλαδή, διαδοχικές συγκρίσεις των στοιχείων του πίνακα με το στοιχείο προς αναζήτηση. Ο μέγιστος αριθμός των συγκρίσεων στη σειριακή αναζήτηση είναι ίσος με N, όπου N είναι το πλήθος των στοιχείων του πίνακα. Αυτό συμβαίνει όταν το στοιχείο που αναζητούμε βρίσκεται στην τελευταία θέση ή δεν υπάρχει στον πίνακα. Πιο κάτω βλέπουμε τη συνάρτηση της σειριακής αναζήτησης. Η συνάρτηση είναι τύπου `boolean` και επιστρέφει `true`, αν εντοπίσει το στοιχείο που ψάχνει, αλλιώς επιστρέφει `false`.

```
bool sequential_search(int arr[], int N, int target){
      for (int i=0; i<N; i++)
            if (target == arr[i])
                  return true;          // Το στοιχείο βρέθηκε
return false;                           // Το στοιχείο δεν βρέθηκε
}
```

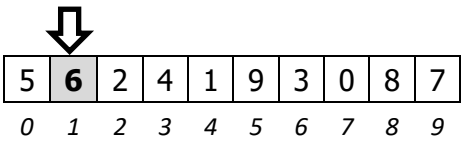
Παράδειγμα

Ας υποθέσουμε ότι έχουμε τον πίνακα **arr = {5, 6, 2, 4, 1, 9, 3, 0, 8, 7}** μεγέθους **10** και ότι ψάχνουμε για το στοιχείο **4**. Αρχικά, η συνάρτηση ξεκινά από το πρώτο στοιχείο στη θέση **0** του πίνακα και ελέγχει για ισότητα με το στοιχείο που ψάχνουμε.



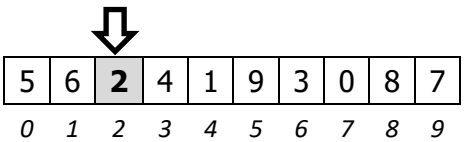

```
target = 4
arr[0] = 5
```

Η συνάρτηση ελέγχει το στοιχείο στη θέση **1**. Εφόσον δεν ισούται με το στοιχείο που ψάχνει, θα συνεχίσει την αναζήτηση.



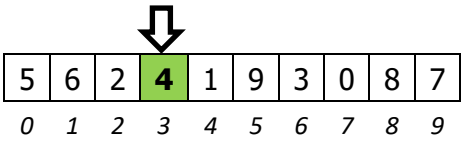

```
target = 4
arr[1] = 6
```

Η συνάρτηση ελέγχει το στοιχείο στη θέση **2**. Εφόσον δεν ισούται με το στοιχείο που ψάχνει, θα συνεχίσει την αναζήτηση.




```
target = 4
arr[2] = 2
```

Το στοιχείο που ψάχνουμε ισούται με το στοιχείο που βρίσκεται στη θέση **3**, η συνάρτηση επιστρέφει `true`.




```
target = 4
arr[3] = 4
```

Αν θέλουμε η συνάρτηση να επιστρέφει τη θέση του στοιχείου στον πίνακα, τότε θα αλλάξουμε τον τύπο της συνάρτησης σε `integer`. Η συνάρτηση θα επιστρέφει τη θέση του στοιχείου, αν αυτό βρεθεί στον πίνακα, αλλιώς θα επιστρέφει την τιμή **-1**, το οποίο σημαίνει ότι το στοιχείο δεν βρέθηκε.

```
int sequential_search(int arr[], int N, int target){
      for (int i=0; i<N; i++)
            if (target == arr[i])
                  return i;              // Το στοιχείο βρέθηκε
return -1;                               // Το στοιχείο δεν βρέθηκε
}
```

Σε περίπτωση που το στοιχείο που ψάχνουμε εμφανίζεται περισσότερες φορές στον πίνακα, η πιο πάνω συνάρτηση θα επιστρέψει μόνο τη θέση της πρώτης του εμφάνισης. Αν θέλουμε η συνάρτηση να επιστρέφει τη θέση της τελευταίας του εμφάνισης, πρέπει να αποθηκεύουμε κάθε φορά τη θέση της πιο πρόσφατης εμφάνισης και να την αλλάζουμε σε κάθε νέα εμφάνιση. Η συνάρτηση θα μετατραπεί όπως πιο κάτω:

```
int sequential_search(int arr[], int N, int target){
      int pos = -1;
      for (int i=0; i<N; i++)
            if (target == arr[i])
                  pos = i;
return pos;                          // Η θέση της τελευταίας εμφάνισης,
}                                    // ή -1 αν το στοιχείο δεν βρεθεί
```

Παράδειγμα 5.1

Να δημιουργήσετε πρόγραμμα, το οποίο να καλεί μία συνάρτηση σειριακής αναζήτησης, η οποία να βρίσκει και να εμφανίζει στην οθόνη το πλήθος των εμφανίσεων ενός χαρακτήρα, μέσα σε έναν πίνακα χαρακτήρων. Το πρόγραμμα, αρχικά, θα διαβάζει το πλήθος N (1<=N<=1000) των χαρακτήρων και, ακολούθως, θα διαβάζει τους χαρακτήρες και θα τους αποθηκεύει σε πίνακα. Στη συνέχεια, θα διαβάζει τον χαρακτήρα τον οποίο θα αναζητήσει και θα καλεί τη συνάρτηση.

Παράδειγμα εισόδου

```
13
C B K J H A C B T Y U B V
B
```

Παράδειγμα εξόδου

```
3
```

```
#include <iostream>
using namespace std;
#define MAXN 1000

int sequential_search(char arr[], int N, char target){
 int total = 0;              // Τοπική μεταβλητή υπολογισμού πλήθους
 for (int i=0; i<N; i++)
        if (target == arr[i])         // Σε κάθε εμφάνιση αυξάνουμε
                total++;              // το πλήθος εμφανίσεων
```

```
 return total;                    // Επιστροφή πλήθους εμφανίσεων
}

int main(){
 char characters[MAXN], item;     // Ο πίνακας δηλώνεται με το
                                  // μέγιστο δυνατό πλήθος χαρακτήρων
 int N;
 cin >> N;

 for (int i=0; i<N; i++)
    cin >> characters[i];         // Εισαγωγή χαρακτήρων

 cin >> item;                     // Χαρακτήρας προς αναζήτηση

 // Καλούμε τη συνάρτηση η οποία επιστρέφει το πλήθος των εμφανίσεων
 cout << sequential_search(characters, N, item) << endl;

return 0;
}
```

(Code: C7_5_example1.cpp)

3. Χρονική Πολυπλοκότητα (Time Complexity)

Η χρονική πολυπλοκότητα της σειριακής αναζήτησης είναι O(N). Αυτό σημαίνει ότι, στη χειρότερη περίπτωση, ο αλγόριθμος για να εκτελεστεί θα χρειαστεί N βήματα. Έχουμε ήδη αναφέρει ότι η χειρότερη περίπτωση που μπορεί να προκύψει με τη σειριακή αναζήτηση, είναι όταν το στοιχείο που ψάχνουμε βρίσκεται στην τελευταία θέση (N-1), ή δεν υπάρχει στον πίνακα. Σε αυτές τις δύο περιπτώσεις, πρέπει να εξετάσουμε ένα-ένα όλα τα στοιχεία του πίνακα (το πλήθος των στοιχείων του οποίου είναι N).

Η χρονική πολυπλοκότητα ενός αλγόριθμου αφορά στο μέγιστο κόστος εκτέλεσής του. Το κόστος αυτό μπορεί να μετρηθεί υπολογίζοντας το πλήθος των βασικών πράξεων που πρέπει να εκτελέσει ο αλγόριθμος, στη χειρότερη περίπτωση. Βασικές πράξεις θεωρούνται:

(α) ανάθεση τιμής (Π.χ. a = 10;)

(β) σύγκριση μεταξύ μεταβλητών (Π.χ. if (a==b))

(γ) εκτέλεση πράξεων (Π.χ. a = b + 5;)

Ουσιαστικά, όταν αναφερόμαστε στη χρονική πολυπλοκότητα ενός αλγόριθμου, αναφερόμαστε στην ταχύτητά του για να εκτελέσει μία συγκεκριμένη εργασία. Η πολυπλοκότητα συμβολίζεται με το γράμμα O και τις παραμέτρους που επηρεάζουν την ταχύτητα, όπως: O(1), O(N), $O(N^2)$, O(log N), κ.λπ.

4. Ανάλυση χρονικής πολυπλοκότητας

Για να κατανοήσουμε καλύτερα τις κατηγορίες της χρονικής πολυπλοκότητας, ας δούμε το ακόλουθο παράδειγμα:

Ο καθηγητής μίας τάξης προσπαθεί να βρει ποιος/ποια από τους συνολικά Ν μαθητές/μαθήτριες έχει στην κατοχή του το απουσιολόγιο της τάξης, κάνοντας ερωτήσεις στους μαθητές και μαθήτριες. Το πλήθος των ερωτήσεων θα καθορίσει, σε κάθε περίπτωση, και την πολυπλοκότητα. Ο καθηγητής στην προσπάθειά του να εντοπίσει το απουσιολόγιο μπορεί να εφαρμόσει τις ακόλουθες μεθόδους:

Α. O(1) Σταθερή (constant): Ο αλγόριθμος χρειάζεται τον ίδιο χρόνο ανεξάρτητα από τα δεδομένα που επεξεργάζεται.

- Ο καθηγητής ρωτά όλη την τάξη: «Ποιος ή ποια έχει στην κατοχή του/της το απουσιολόγιο;». Ο καθηγητής κάνει μόνο μία ερώτηση.

Κώδικας:

```
int ans = 1;
```

Β. O(N) Γραμμική (linear): Ο χρόνος εκτέλεσης έχει γραμμική σχέση με τα δεδομένα που επεξεργάζεται.

- Ο καθηγητής ρωτάει έναν-έναν τους μαθητές και τις μαθήτριες της τάξης αν έχουν στην κατοχή τους το απουσιολόγιο. Στη χειρότερη περίπτωση, θα πρέπει να τους ρωτήσει όλους, δηλαδή, Ν ερωτήσεις.

Κώδικας:

```
int ans = 0;
for (int i = 0; i < N; i++){
    ans++;
}
```

Γ. O(N^2) Τετραγωνική (quadratic): Ο χρόνος εκτέλεσης έχει τετραγωνική σχέση με τα δεδομένα που επεξεργάζεται.

- Ο καθηγητής ρωτάει τον μαθητή που κάθεται στο πρώτο θρανίο: «Έχει ο Αντρέας το απουσιολόγιο; Όχι; Έχει η Βίκυ το απουσιολόγιο;...» και συνεχίζει να τον ρωτάει για όλους τους υπόλοιπους μαθητές της τάξης. Αν ο πρώτος μαθητής δεν γνωρίζει, τότε θα προχωρήσει στον επόμενο και θα ρωτήσει για όλους τους υπόλοιπους κ.ο.κ. Στη χειρότερη περίπτωση θα χρειαστεί να κάνει N^2 ερωτήσεις.

Κώδικας:

```
int ans = 0;
for (int i = 0; i < N; i++){
   for (int j = 0; j < N; j++){
        ans++;
   }
}
```

Δ. O($\log_2$ N) Λογαριθμική (logarithmic): Ο χρόνος εκτέλεσης έχει λογαριθμική σχέση με τα δεδομένα που επεξεργάζεται.

- Ο καθηγητής χωρίζει την τάξη σε δύο μέρη (αριστερό, δεξί) και ρωτάει: «Το απουσιολόγιο βρίσκεται στο αριστερό ή στο δεξί μέρος της τάξης;» και αφού του απαντήσουν, χωρίζει το μέρος της τάξης που βρίσκεται το απουσιολόγιο σε άλλα δύο μέρη κ.ο.κ. Στη χειρότερη περίπτωση, θα χρειαστεί να κάνει $\log_2$ N ερωτήσεις.

Κώδικας:

```
int ans = 0;
while (N > 0) {
    N /= 2;
    ans++;
}
```

Παράδειγμα 5.2

Να βρείτε τη χρονική πολυπλοκότητα καθώς και το αποτέλεσμα του πιο κάτω κώδικα:

```
int N = 5, steps = 0;
  for (int i = 1; i <= N; i++)
        for (int j = N; j >= 1; j--)
             for (int k = 1; k <= N; k++)
                  steps++;
  cout << steps << endl;
```

Λύση: Έχουμε τρεις εμφωλευμένους βρόχους for. Άρα, τα συνολικά βήματα υπολογίζονται ως εξής: N * N * N = N^3, το οποίο συμβολίζεται με O(N^3) και καλείται Κυβική (cubic) πολυπλοκότητα. Το αποτέλεσμα του πιο πάνω κώδικα είναι 125.

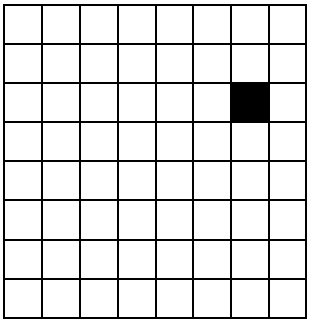
5. Στρατηγική Διαίρει και Βασίλευε (Divide and Conquer)

Στην κατηγορία αυτή οι αλγόριθμοι, αρχικά, διασπούν το πρόβλημα σε μικρότερα προβλήματα. Στη συνέχεια, επιλύουν ανεξάρτητα και διαδοχικά κάθε ένα από τα υποπροβλήματα και, τέλος, συνδυάζουν τις επιμέρους λύσεις με τέτοιο τρόπο, ώστε να σχηματιστεί η λύση του αρχικού προβλήματος. Ας δούμε ένα παράδειγμα:

Να γεμίσετε έναν πίνακα διαστάσεων **2^nx2^n** με τρόμινος, έτσι ώστε να μη μείνουν κενά τετράγωνα. Ο πιο κάτω πίνακας διαστάσεων **8x8** περιέχει και ένα μη προσβάσιμο τετράγωνο.

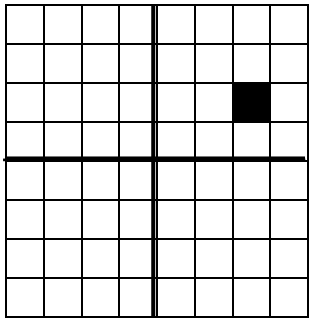
τρόμινο

πίνακας 8x8

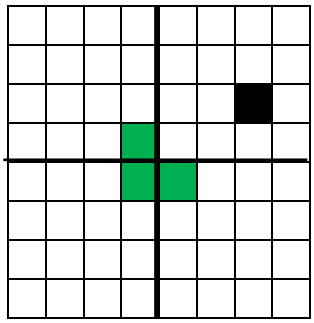


Λύση

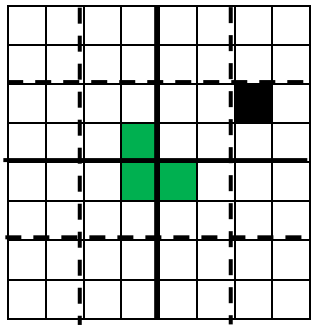
Χωρίζουμε τον αρχικό πίνακα σε τέσσερα ίσα μέρη μεγέθους **4x4**.



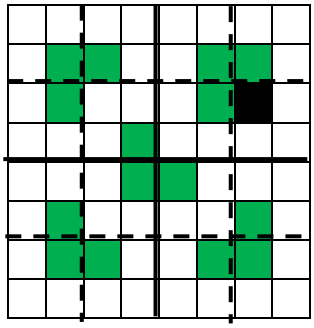
Τοποθετούμε ένα τρόμινο με τέτοιο τρόπο, έτσι ώστε κάθε ένας από τους υπόλοιπους υποπίνακες μεγέθους **4x4** να έχει και αυτός ένα μη προσβάσιμο τετράγωνο.



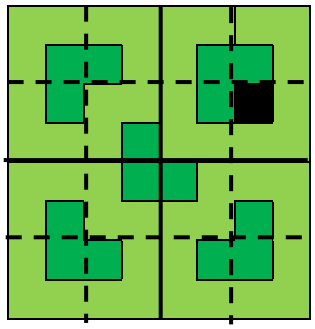
Χωρίζουμε κάθε έναν από τους υποπίνακες σε τέσσερα ίσα μέρη, μεγέθους **2x2**.



Τοποθετούμε ένα τρόμινο με τέτοιο τρόπο, έτσι ώστε κάθε ένας από τους υπόλοιπους υποπίνακες, μεγέθους **2x2**, να έχει και αυτός ένα μη προσβάσιμο τετράγωνο.



Συμπληρώνουμε με τρόμινος κάθε έναν από τους υποπίνακες, μεγέθους **2x2**, και έχουμε το τελικό αποτέλεσμα.



6. Δυαδική αναζήτηση (Binary search)

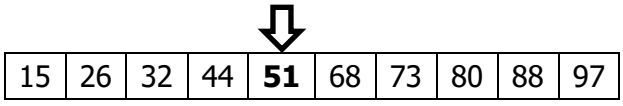
Η δυαδική αναζήτηση ανήκει στην κατηγορία αλγορίθμων «Διαίρει και Βασίλευε» (Divide and Conquer). Η δυαδική αναζήτηση μπορεί να χρησιμοποιηθεί μόνο αν ο πίνακας είναι **ήδη ταξινομημένος**. Ο αλγόριθμος ξεκινά εξετάζοντας το «μεσαίο» στοιχείο του πίνακα. Το μεσαίο στοιχείο καθορίζεται βρίσκοντας τον μέσο όρο των θέσεων του πρώτου και του τελευταίου στοιχείου του πίνακα. Αν η τιμή του ζητούμενου στοιχείου είναι μικρότερη από την τιμή του μεσαίου στοιχείου, τότε θα συνεχίσουμε το ψάξιμο στο «αριστερό» μέρος του πίνακα αγνοώντας το «δεξί». Διαφορετικά, αν η τιμή του ζητούμενου στοιχείου είναι μεγαλύτερη από την τιμή του μεσαίου στοιχείου, τότε θα συνεχίσουμε το ψάξιμο στο «δεξί» μέρος του πίνακα, αγνοώντας το «αριστερό» μέρος.

Πιο κάτω βλέπουμε τη συνάρτηση της δυαδικής αναζήτησης. Η συνάρτηση είναι τύπου `boolean` και επιστρέφει `true` αν εντοπίσει το στοιχείο που ψάχνει, αλλιώς επιστρέφει `false`. Όταν το στοιχείο first γίνει μεγαλύτερο από το στοιχείο last, συνεπάγεται ότι το στοιχείο που αναζητούμε δεν βρέθηκε.

```
bool binary_search(int arr[], int N, int target){

    int first = 0, last = N - 1, mid;
    while (first <= last) {
        mid = (first + last) / 2;      // Εύρεση μεσαίου στοιχείου
        if (target == arr[mid])
            return true;               // Το στοιχείο βρέθηκε
        else if (target < arr[mid])
            last = mid - 1;            // Ψάξε στο «αριστερό» μέρος
        else
            first = mid + 1;           // Ψάξε στο «δεξί» μέρος
     }

return false;                          // Το στοιχείο δεν βρέθηκε
}
```

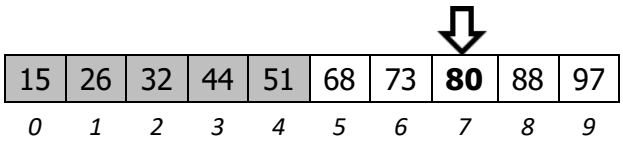
Παράδειγμα

Ας υποθέσουμε ότι έχουμε τον ταξινομημένο πίνακα **arr = {15, 26, 32, 44, 51, 68, 73, 80, 88, 97}** μεγέθους **10** και ότι ψάχνουμε για το στοιχείο **73**. Αρχικά, βρίσκουμε το μεσαίο στοιχείο του πίνακα, υπολογίζοντας τον μέσο όρο των θέσεων του πρώτου και τελευταίου στοιχείου και ελέγχουμε για ισότητα με το στοιχείο που ψάχνουμε:



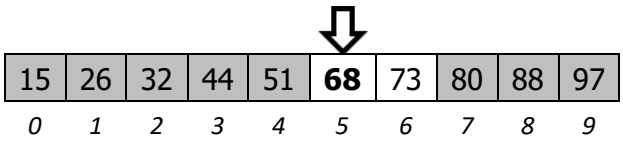

```
first = 0
last = 9
mid = 4
arr[mid] = 51
target = 73
```

Αφού το μεσαίο στοιχείο είναι μικρότερο από το ζητούμενο στοιχείο (**51<73**), θα συνεχίσουμε την αναζήτηση στο «δεξί» μέρος του πίνακα. Το μεσαίο στοιχείο θα είναι το στοιχείο στη θέση **7**:



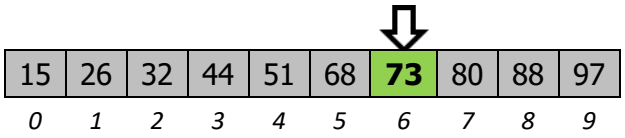

```
first = 5
last = 9
mid = 7
arr[mid] = 80
target = 73
```

Αφού το μεσαίο στοιχείο είναι μεγαλύτερο από το ζητούμενο στοιχείο (**80>73**), θα συνεχίσουμε την αναζήτηση στο «αριστερό» μέρος του πίνακα. Το μεσαίο στοιχείο θα είναι το στοιχείο στη θέση **5**:




```
first = 5
last = 6
mid = 5
arr[mid] = 68
target = 73
```

Αφού το μεσαίο στοιχείο είναι μικρότερο από το ζητούμενο στοιχείο (**68<73**), θα συνεχίσουμε την αναζήτηση στο «δεξί» μέρος του πίνακα. Το μεσαίο στοιχείο είναι το στοιχείο στη θέση **6**, το οποίο είναι το στοιχείο που ψάχνουμε:




```
first = 6
last = 6
mid = 6
arr[mid] = 73
target = 73
```

Αν θέλουμε η συνάρτηση να επιστρέφει τη θέση του στοιχείου στον πίνακα, τότε θα αλλάξουμε τον τύπο της συνάρτησης σε `integer`. Η συνάρτηση θα επιστρέφει τη θέση του στοιχείου, αν αυτό βρεθεί στον πίνακα, αλλιώς θα επιστρέφει την τιμή **-1**, που σημαίνει ότι το στοιχείο δεν βρέθηκε (last<first).

```
int binary_search(int arr[], int N, int target){
    int first = 0, last = N - 1, mid;
    while (first <= last) {
        mid = (first + last) / 2;      // Εύρεση μεσαίου στοιχείου
        if (target == arr[mid])
            return mid;                // Το στοιχείο βρέθηκε
        else if (target < arr[mid])
            last = mid - 1;            // Ψάξε στο «αριστερό» μέρος
        else
            first = mid + 1;           // Ψάξε στο «δεξί» μέρος
     }

return -1;                             // Το στοιχείο δεν βρέθηκε
}
```

Η χρονική πολυπλοκότητα της δυαδικής αναζήτησης είναι $O(\log_2 N)$. Αυτό σημαίνει ότι, στη χειρότερη περίπτωση, ο αλγόριθμος για να εκτελεστεί θα χρειαστεί $\log_2 N$ βήματα.

Απόδειξη

Έστω ότι έχουμε έναν πίνακα N στοιχείων.

- Αρχικά υπάρχουν N στοιχεία για την αναζήτηση.
- Μετά το πρώτο βήμα, αφού χωρίζουμε τον πίνακα στα δύο, θα απομείνουν $\leq N/2$ στοιχεία για αναζήτηση.
- Μετά το δεύτερο βήμα απομένουν $\leq N/4 = N/2^2$ στοιχεία.
- Μετά το τρίτο βήμα απομένουν $\leq N/8 = N/2^3$ στοιχεία.
- Μετά το τέταρτο βήμα απομένουν $\leq N/16 = N/2^4$ στοιχεία.
- Μετά το βήμα k απομένουν $\leq N/2^k$ στοιχεία.
- Όταν απομείνουν ≤ 1 στοιχεία η διαδικασία τερματίζει.

- Επομένως: $N/2^k \leq 1 \rightarrow k = O(\log_2 N)$.
- Άρα το πολύ σε $k = O(\log_2 N)$ βήματα η διαδικασία τερματίζει.
- Κάθε βήμα της δυαδικής αναζήτησης απαιτεί χρόνο O(1) και, επομένως, η συνολική πολυπλοκότητα χειρότερης περίπτωσης είναι: **$O(\log_2 N)$**.

Π.χ. για N=100, δηλαδή σε πίνακα 100 στοιχείων, η δυαδική αναζήτηση θα χρειαστεί το πολύ: **$\log_2(100) = 6.6438561898 \approx 7$ βήματα.**

Παράδειγμα 5.3

Να δημιουργήσετε πρόγραμμα, το οποίο να καλεί μία συνάρτηση δυαδικής αναζήτησης, η οποία να βρίσκει και να εμφανίζει στην οθόνη τη θέση ενός δεκαδικού αριθμού μέσα σε έναν ταξινομημένο πίνακα δεκαδικών. Το πρόγραμμα, αρχικά, θα διαβάζει το πλήθος N (1<=N<=1000) των αριθμών και, ακολούθως, θα διαβάζει τους αριθμούς και θα τους αποθηκεύει σε πίνακα. Στη συνέχεια, θα διαβάζει τον δεκαδικό τον οποίο θα αναζητήσει και θα καλεί τη συνάρτηση. Αν ο αριθμός προς αναζήτηση δεν βρίσκεται στον πίνακα, η συνάρτηση θα επιστρέφει -1.

Παράδειγμα εισόδου

```
12
5.1 6.2 8.3 10.4 13.5 14.6 18.7 24.8 34.9 55.4 65.1 85.2
55.4
```

Παράδειγμα εξόδου

```
9
```

```
#include <iostream>
using namespace std;
#define MAXN 1000

// Συνάρτηση δυαδικής αναζήτησης
int binary_search(double arr[], int N, double target){
    int first = 0, last = N - 1, mid;
    while (first <= last) {
        mid = (first + last) / 2;      // Εύρεση μεσαίου στοιχείου
        if (target == arr[mid])
            return mid;                // Το στοιχείο βρέθηκε
        else if (target < arr[mid])
            last = mid - 1;            // Ψάξε στο «αριστερό» μέρος
        else
            first = mid + 1;           // Ψάξε στο «δεξί» μέρος
     }
return -1;                             // Το στοιχείο δεν βρέθηκε
}
```

```
int main(){
 double numbers[MAXN], item;     // Ο πίνακας δηλώνεται με το
 int N;                          // μέγιστο δυνατό πλήθος αριθμών
 cin >> N;

 for (int i=0; i<N; i++)         // Εισαγωγή αριθμών
    cin >> numbers[i];

 cin >> item;                    // Εισαγωγή αριθμού προς αναζήτηση
 cout << binary_search(numbers, N, item) << endl;

return 0;
}
```

(Code: C7_5_example3.cpp)

Παράδειγμα 5.4

Να δημιουργήσετε πρόγραμμα, το οποίο να καλεί μία συνάρτηση δυαδικής αναζήτησης, η οποία να βρίσκει και να εμφανίζει στην οθόνη τη θέση της πρώτης εμφάνισης ενός ακέραιου, μέσα σε έναν ταξινομημένο πίνακα ακεραίων. Το πρόγραμμα, αρχικά, θα διαβάζει το πλήθος N (1<=N<=1000) των ακεραίων και, ακολούθως, θα διαβάζει τους ακέραιους και θα τους αποθηκεύει σε πίνακα. Στη συνέχεια, θα διαβάζει τον ακέραιο τον οποίο θα αναζητήσει και θα καλεί τη συνάρτηση. Να θεωρήσετε ότι ο ακέραιος προς αναζήτηση θα εμφανίζεται τουλάχιστον μία φορά στον πίνακα.

Παράδειγμα εισόδου

```
12
2 2 2 2 3 3 3 3 4 4 4 4
2
```

Παράδειγμα εξόδου

```
0
```

```
#include <iostream>
using namespace std;
#define MAXN 1000
// Συνάρτηση εύρεσης θέσης πρώτης εμφάνισης στοιχείου σε πίνακα
int binary_search_first(int arr[], int N, int target) {
   int first = 0, last = N - 1, mid, pos = -1;
   while (first <= last) {
     mid = (first + last) / 2;   // Εύρεση μεσαίου στοιχείου
     if (arr[mid] == target) {
        pos = mid;      // Το στοιχείο βρέθηκε, αποθηκεύουμε τη θέση
       last = mid - 1; // του, αλλά συνεχίζουμε να ψάχνουμε στο
     }                 // «αριστερό μέρος» του πίνακα
```

```
     else if (target < arr[mid])
        last = mid - 1;                  // Ψάξε στο «αριστερό» μέρος
     else
        first = mid + 1;                 // Ψάξε στο «δεξί» μέρος
   }
return pos;
}

int main(){
 int numbers[MAXN], item, N;     // Ο πίνακας δηλώνεται με το
 cin >> N;                       // μέγιστο δυνατό πλήθος ακεραίων

 for (int i=0; i<N; i++)
    cin >> numbers[i];

 cin >> item;
 cout << binary_search_first(numbers, N, item) << endl;

return 0;
}
```

(Code: C7_5_example4.cpp)

Εναλλακτική προσέγγιση

```
#include <iostream>
using namespace std;
#define MAXN 1000

// Συνάρτηση εύρεσης θέσης πρώτης εμφάνισης στοιχείου σε πίνακα
// Αρχικοποιούμε το first με -1 γιατί το στοιχείο που ψάχνουμε
// μπορεί να βρίσκεται στην πρώτη θέση του πίνακα

int binary_search_first(int arr[], int N, int target) {
 int first = -1, last = N - 1, mid;
    while (first < last - 1) {
        mid = (first + last) / 2;      // Εύρεση μεσαίου στοιχείου
        if (arr[mid] >= target)        // Ακόμα και αν βρεθεί
            last = mid;                // το στοιχείο, συνεχίζουμε
        else                           // το ψάξιμο αποθηκεύοντας την
            first = mid;               // θέση του στοιχείου
    }
return last;
}
```

```
int main(){
 int numbers[MAXN], item, N;
 cin >> N;
 for (int i=0; i<N; i++)
      cin >> numbers[i];
 cin >> item;
 cout << binary_search_first(numbers, N, item) << endl;

return 0;
}
```

(Code: C7_5_example5.cpp)

Επεξήγηση παραδείγματος

Ο πιο πάνω αλγόριθμος δουλεύει ως εξής:

- Αρχικοποιούμε τη θέση first με -1, γιατί το στοιχείο που ψάχνουμε μπορεί να βρίσκεται στην πρώτη θέση (θέση 0).
- Ακόμα και αν το στοιχείο που ψάχνουμε βρεθεί, ο αλγόριθμος δεν σταματά, αλλά συνεχίζει μέχρι να βρεθεί η πρώτη του εμφάνιση στον πίνακα.
- Ο αλγόριθμος τερματίζει όταν απομείνουν μόνο δύο γειτονικά στοιχεία για έλεγχο και ισχύει ότι first=last-1.
- Ο αλγόριθμος επιστρέφει τη θέση last.

Στο πιο πάνω παράδειγμα ο αλγόριθμος θα εκτελεστεί ως εξής:

Βήμα 1: first = -1, last = 11, mid = 5, arr[mid] = 3

2	2	2	2	3	3	3	3	4	4	4	4
0	1	2	3	4	5	6	7	8	9	10	11

Βήμα 2: first = -1, last = 5, mid = 2, arr[mid] = 2

2	2	2	2	3	3	3	3	4	4	4	4
0	1	2	3	4	5	6	7	8	9	10	11

Βήμα 3: first = -1, last = 2, mid = 0, arr[mid] = 2

2	2	2	2	3	3	3	3	4	4	4	4
0	1	2	3	4	5	6	7	8	9	10	11

Για first=-1 και last=0 η συνθήκη first<last-1 (-1<-1) είναι ψευδής και ο βρόχος τερματίζει, επιστρέφοντας τη θέση 0.

2	2	2	2	3	3	3	3	4	4	4	4
0	1	2	3	4	5	6	7	8	9	10	11

Η εύρεση της θέσης της τελευταίας εμφάνισης ενός στοιχείου με τη χρήση δυαδικής αναζήτησης αφήνεται ως άσκηση (**5.20**).

Ασκήσεις Κεφαλαίου - www.hackerrank.com/g75

Άσκηση 5.1

Να συμπληρώσετε τις εντολές για την πιο κάτω συνάρτηση σειριακής αναζήτησης, η οποία να επιστρέφει τη θέση του στοιχείου στον πίνακα ή να επιστρέφει -1 αν το στοιχείο δεν βρεθεί.

```
int sequential_search(int arr[], int N, int target){




}
```

Άσκηση 5.2

Να συμπληρώσετε τις εντολές για την πιο κάτω συνάρτηση δυαδικής αναζήτησης, η οποία να επιστρέφει τη θέση του στοιχείου στον πίνακα ή να επιστρέφει -1 αν το στοιχείο δεν βρεθεί.

```
int binary_search(string arr[], int N, string target){









}
```

Άσκηση 5.3

Να δείξετε τα βήματα του αλγόριθμου της σειριακής αναζήτησης στον πιο κάτω πίνακα, μεγέθους 20, αν αναζητούμε το στοιχείο 8. Σε κάθε βήμα να δείξετε τη θέση και το στοιχείο που βρισκόμαστε, καθώς και το στοιχείο προς αναζήτηση. Σας δίνεται το πρώτο βήμα:

`Βήμα 1: i=0, arr[i]=1, target =8`

1	3	5	7	**8**	9	11	12	13	15	16	18	20	22	23	24	27	28	29	30

Άσκηση 5.4

Να δείξετε τα βήματα του αλγόριθμου της δυαδικής αναζήτησης στον πιο κάτω πίνακα, μεγέθους 20, αν αναζητούμε το στοιχείο 13. Σε κάθε βήμα να δείξετε τη θέση του πρώτου, του τελευταίου, του μεσαίου στοιχείου, το στοιχείο που βρίσκεται στη μεσαία θέση και το στοιχείο προς αναζήτηση. Σας δίνεται το πρώτο βήμα:

`Βήμα 1: first=0, last=19, mid=9, arr[mid]=15, target=13`

1	3	5	7	8	9	11	12	**13**	15	16	18	20	22	23	24	27	28	29	30

Άσκηση 5.5

Να δημιουργήσετε πρόγραμμα, το οποίο, αρχικά, να διαβάζει έναν ακέραιο αριθμό Ν (1<=N<=100). Ακολούθως, να διαβάζει Ν δεκαδικούς αριθμούς και να τους αποθηκεύει σε πίνακα. Στη συνέχεια, να καλεί μία συνάρτηση σειριακής αναζήτησης, η οποία να δέχεται παραμετρικά έναν πίνακα δεκαδικών, το πλήθος των στοιχείων του πίνακα και ένα στοιχείο προς αναζήτηση (δεκαδικός). Η συνάρτηση να επιστρέφει τη θέση του στοιχείου προς αναζήτηση στον πίνακα ή να επιστρέφει -1, σε περίπτωση που το στοιχείο δεν εντοπιστεί. **Η συνάρτηση να τερματίζει, όταν βρεθεί το στοιχείο ή όταν φτάσει στο τελευταίο στοιχείο του πίνακα χωρίς να το εντοπίσει**. Να θεωρήσετε ότι το στοιχείο προς αναζήτηση, εφόσον εμφανίζεται στον πίνακα, εμφανίζεται μόνο μία φορά.

Άσκηση 5.6

Να δημιουργήσετε πρόγραμμα, το οποίο, αρχικά, να διαβάζει έναν ακέραιο αριθμό Ν (1<=N<=100). Ακολούθως, να διαβάζει Ν ακέραιους, ταξινομημένους σε αύξουσα σειρά και να τους αποθηκεύει σε πίνακα. Στη συνέχεια, να καλεί μία συνάρτηση σειριακής αναζήτησης, η οποία να δέχεται παραμετρικά έναν πίνακα ακεραίων, το πλήθος των στοιχείων του πίνακα και ένα στοιχείο προς αναζήτηση (ακέραιος). Η συνάρτηση να επιστρέφει τη θέση του στοιχείου προς αναζήτηση στον πίνακα ή να επιστρέφει -1, σε περίπτωση που το στοιχείο δεν βρεθεί. **Η συνάρτηση να τερματίζει, όταν βρεθεί το στοιχείο ή όταν βρεθεί στοιχείο μεγαλύτερο από το στοιχείο προς αναζήτηση (το στοιχείο δεν υπάρχει στον πίνακα)**. Να θεωρήσετε ότι το στοιχείο προς αναζήτηση, εφόσον εμφανίζεται στον πίνακα, εμφανίζεται μόνο μία φορά.

Άσκηση 5.7

Να δημιουργήσετε πρόγραμμα, το οποίο αρχικά να διαβάζει έναν ακέραιο αριθμό Ν (1<=N<=50). Ακολούθως, να διαβάζει Ν χαρακτήρες στο πεδίο τιμών [A…Z] και να τους αποθηκεύει σε πίνακα. Στη συνέχεια, να καλεί μία συνάρτηση σειριακής αναζήτησης, η οποία να δέχεται παραμετρικά έναν πίνακα χαρακτήρων, το πλήθος των στοιχείων του πίνακα και ένα στοιχείο προς αναζήτηση (χαρακτήρας). Η συνάρτηση να επιστρέφει τη θέση της τελευταίας εμφάνισης του στοιχείου προς αναζήτηση στον πίνακα ή να επιστρέφει -1, σε περίπτωση που το στοιχείο δεν βρεθεί. Να θεωρήσετε ότι το στοιχείο προς αναζήτηση, εφόσον εμφανίζεται στον πίνακα, εμφανίζεται μία ή περισσότερες φορές.

Άσκηση 5.8

Μέσα στο αρχείο askisi5_8.txt υπάρχει ένας ακέραιος αριθμός Ν (1<=N<=1000), ακολουθούμενος από Ν ακέραιους σε τυχαία σειρά. Να δημιουργήσετε πρόγραμμα, το οποίο να διαβάζει τους αριθμούς από το αρχείο και να τους αποθηκεύει σε πίνακα. Ακολούθως, να δέχεται έναν ακέραιο αριθμό Χ προς αναζήτηση από το πληκτρολόγιο, να χρησιμοποιεί μία συνάρτηση σειριακής αναζήτησης και να εμφανίζει στην οθόνη την λέξη «found» σε περίπτωση που ο αριθμός εντοπιστεί στον πίνακα ή «not found» στην αντίθετη περίπτωση.

Άσκηση 5.9

Να δημιουργήσετε πρόγραμμα, το οποίο να διαβάζει ένα όνομα από το πληκτρολόγιο και να το αναζητά, με τη χρήση συνάρτησης σειριακής αναζήτησης, στο αρχείο names.txt. Αν το όνομα βρεθεί στο αρχείο, το πρόγραμμα να επιστρέφει τη θέση του ονόματος, αλλιώς να

επιστρέφει το μήνυμα «not found». Να θεωρηθεί ότι δεν γνωρίζουμε το πλήθος των ονομάτων που υπάρχουν στο αρχείο names.txt.

Άσκηση 5.10

Να υλοποιήσετε την άσκηση **5.9** με τη χρήση συνάρτησης δυαδικής αναζήτησης. Να θεωρήσετε ότι το αρχείο μπορεί να περιέχει μέχρι 10,000 ονόματα.

Άσκηση 5.11

Να δημιουργήσετε πρόγραμμα, το οποίο να διαβάζει αρχικά έναν ακέραιο αριθμό Ν. Ακολούθως, να διαβάζει Ν (1<=N<=50) χαρακτήρες στο πεδίο τιμών [A…Z] και να τους αποθηκεύει σε πίνακα. Στη συνέχεια, να καλεί μία συνάρτηση αναζήτησης, η οποία θα δέχεται παραμετρικά έναν πίνακα χαρακτήρων, το πλήθος των στοιχείων του πίνακα και ένα στοιχείο προς αναζήτηση (χαρακτήρας). Η συνάρτηση θα επιστρέφει το πλήθος των εμφανίσεων αποκλειστικά δύο διαδοχικών χαρακτήρων προς αναζήτηση στον πίνακα.

Παράδειγμα εισόδου

```
12
ABCDEEEGRREE
E
```

Παράδειγμα εξόδου

```
1
```

Επεξήγηση: οι διαδοχικοί χαρακτήρες ΕΕ εμφανίζονται μία φορά στον πίνακα. Η συμβολοσειρά ΕΕΕ με τρεις διαδοχικούς χαρακτήρες δεν λαμβάνεται υπόψη.

Άσκηση 5.12

Ο Παντελής ψάχνει την κάρτα με τον αριθμό 58 σε μία σειρά από Ν (1<=N<10,000) ταξινομημένες κάρτες, με την εξής μέθοδο:

1. Αναποδογυρίζει τη μεσαία κάρτα (αν δεν υπάρχει μεσαίο στοιχείο, επιλέγει αυτό που βρίσκεται πιο κοντά στη μέση).

2. Το 58 είναι μεγαλύτερο από το 39, οπόταν γνωρίζει ότι πρέπει να ψάξει στα δεξιά της κάρτας με τον αριθμό 39. Επιλέγει τη μεσαία κάρτα της υπόλοιπης σειράς, που βρίσκεται δεξιά της κάρτας με τον αριθμό 39.

3. Αφού το 58 είναι μικρότερο του 81, τότε θα ψάξει στα αριστερά του.

4. Αφού το 58 είναι μεγαλύτερο του 41, θα αναποδογυρίσει την κάρτα που βρίσκεται στα δεξιά του, που είναι η κάρτα με τον αριθμό 58 που ψάχνει.

Ο Παντελής, για να βρει την κάρτα με τον αριθμό 58 ανάμεσα σε δέκα κάρτες, χρειάστηκε να αναποδογυρίσει τέσσερις κάρτες. Ο Ανδρέας ακολουθεί μία πιο απλή μέθοδο. Ξεκινώντας από την κάρτα που βρίσκεται στα αριστερά, αναποδογυρίζει μία-μία τις κάρτες μέχρι να βρει την κάρτα που ψάχνει. Άρα, για να βρει την κάρτα με τον αριθμό 58, θα πρέπει να αναποδογυρίσει επτά συνολικά κάρτες.

Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται το πλήθος και τους αριθμούς των καρτών, καθώς και τον αριθμό της κάρτας που ψάχνουν ο Παντελής και ο Ανδρέας. Το πρόγραμμα να εμφανίζει το πλήθος των καρτών που αναποδογύρισε ο Παντελής και το πλήθος των καρτών που αναποδογύρισε ο Ανδρέας στην προσπάθεια τους να βρουν την κάρτα. Να σημειωθεί ότι η κάρτα που ψάχνουν μπορεί να μην υπάρχει ανάμεσα στο σύνολο των καρτών.

Παράδειγμα εισόδου

```
8
5 33 53 64 70 78 79 87
87
```

Παράδειγμα εξόδου

```
4
8
```

Άσκηση 5.13

Αν σας δοθούν ένας ακέραιος αριθμός N (1<=N<=100), ακολουθούμενος από N συμβολοσειρές ταξινομημένες σε αλφαβητική σειρά και μία συμβολοσειρά S, Να δημιουργήσετε πρόγραμμα, το οποίο να χρησιμοποιεί μία συνάρτηση δυαδικής αναζήτησης, η οποία να αναζητά τη συμβολοσειρά S ανάμεσα στις N συμβολοσειρές. Το πρόγραμμα να εμφανίζει τη θέση της συμβολοσειράς στον πίνακα ή το μήνυμα «not found», σε περίπτωση που η συμβολοσειρά S δεν βρεθεί.

Άσκηση 5.14

Αν σας δοθούν ένας ακέραιος αριθμός N (1<=N<=100), ακολουθούμενος από N ακέραιους **ταξινομημένους σε φθίνουσα σειρά** και ένας ακέραιος X, Να δημιουργήσετε πρόγραμμα, το οποίο να χρησιμοποιεί μία συνάρτηση δυαδικής αναζήτησης, η οποία να αναζητά τον αριθμό X ανάμεσα στους N ακεραίους. Το πρόγραμμα να εμφανίζει τη θέση του αριθμού στον πίνακα ή το μήνυμα «not found», σε περίπτωση που ο αριθμός X δεν βρεθεί.

Άσκηση 5.15

Να δημιουργήσετε πρόγραμμα, το οποίο να καλεί μία συνάρτηση σειριακής αναζήτησης, η οποία να βρίσκει πόσες φορές εμφανίζεται ένα όνομα σε έναν πίνακα ονομάτων. Το πρόγραμμα να δέχεται, αρχικά, έναν ακέραιο αριθμό N (1<=N<=100), ακολουθούμενο από N ονόματα σε τυχαία σειρά, τα οποία θα αποθηκεύει στον πίνακα. Ακολούθως, το πρόγραμμα να δέχεται ένα όνομα προς αναζήτηση και στη συνέχεια να καλεί τη συνάρτηση, η οποία να επιστρέφει το πλήθος των εμφανίσεων του ονόματος στον πίνακα. Να θεωρήσετε ότι το όνομα θα εμφανίζεται τουλάχιστον μία φορά στον πίνακα.

Άσκηση 5.16

Να τροποποιήσετε την άσκηση **5.15**, ώστε το πρόγραμμα να εμφανίζει, στην ίδια γραμμή, τις θέσεις των εμφανίσεων του ονόματος προς αναζήτηση στον πίνακα.

Άσκηση 5.17

Ο Χρίστος εργάζεται στο τμήμα προσωπικού της εταιρείας «TIF Advogates», η οποία εργοδοτεί 12 υπαλλήλους. Για κάθε υπάλληλο έχει καταχωρημένο τον αριθμό μητρώου και το email του. Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται, αρχικά, τον αριθμό μητρώου και το email κάθε υπαλλήλου και να τα καταχωρίζει στους πίνακες ΑΜ και emails, αντίστοιχα. Στη συνέχεια, να δέχεται τον αριθμό μητρώου ενός υπαλλήλου και να τυπώνει το email του. Σε περίπτωση που δεν βρεθεί ο αριθμός μητρώου τότε να εμφανίζεται ο αριθμός -1.

Παράδειγμα εισόδου

```
24 elenap@tifa.com
102 petros1@tifa.com
31 papgio@tifa.com
190 georgio@tifa.com
82 vassos@tifa.com
123 chris@tifa.com
47 frixost@tifa.com
791 fayt@tifa.com
33 jinar@tifa.com
77 taspet@tifa.com
57 leont@tifa.com
62 kleio@tifa.com
82
```

Παράδειγμα εξόδου

```
vassos@tifa.com
```

Άσκηση 5.18

Να δημιουργήσετε πρόγραμμα, το οποίο να διαβάζει, αρχικά, έναν ακέραιο αριθμό N (1<=N<=100) και, ακολούθως, N ακέραιους τους οποίους θα αποθηκεύει σε πίνακα. Στη συνέχεια, να καλεί μία συνάρτηση σειριακής αναζήτησης η οποία θα δέχεται παραμετρικά έναν πίνακα ακεραίων, το πλήθος των στοιχείων του πίνακα και δύο ακέραιους προς αναζήτηση. Αν η συνάρτηση εντοπίσει και τα δύο στοιχεία στον πίνακα, να εμφανίζει το μήνυμα «both items found», ενώ αν βρεθεί μόνο το ένα, να εμφανίζει το μήνυμα «item X

was found». Σε περίπτωση που κανένα στοιχείο δεν βρεθεί στον πίνακα να εμφανίζεται ο αριθμός -1.

Άσκηση 5.19

Αν σας δοθούν, αρχικά, N (1<=N<=100) ακέραιοι αριθμοί σε αύξουσα σειρά και, ακολούθως, Q (1<=Q<=10) ακέραιοι αριθμοί προς αναζήτηση, Να δημιουργήσετε πρόγραμμα, το οποίο να καλεί μία συνάρτηση δυαδικής αναζήτησης, η οποία να ψάχνει για κάθε έναν από τους Q ακέραιους ανάμεσα στους N ακέραιους. Αν ένας ακέραιος βρεθεί στον πίνακα, να εμφανίζεται η λέξη «YES», ενώ αν δεν βρεθεί να εμφανίζεται η λέξη «NO».

Παράδειγμα εισόδου

```
5 2
13 24 45 67 71
45
85
```

Παράδειγμα εξόδου

```
YES
NO
```

Επεξήγηση: ο αριθμός 45 εμφανίζεται ανάμεσα στους ακεραίους, ενώ ο αριθμός 85 όχι.

Άσκηση 5.20

Να δημιουργήσετε πρόγραμμα, το οποίο να καλεί μία συνάρτηση δυαδικής αναζήτησης, η οποία να βρίσκει και να εμφανίζει στην οθόνη τη θέση της τελευταίας εμφάνισης ενός ακέραιου αριθμού μέσα σε έναν ταξινομημένο, σε αύξουσα σειρά, πίνακα ακεραίων. Το πρόγραμμα, αρχικά, να διαβάζει το πλήθος N (1<=N<=100) των ακεραίων και, ακολούθως, να διαβάζει τους ακέραιους και να τους αποθηκεύει σε πίνακα. Στη συνέχεια, να διαβάζει τον ακέραιο προς αναζήτηση και να καλεί τη συνάρτηση. Να θεωρήσετε ότι ο ακέραιος προς αναζήτηση θα εμφανίζεται τουλάχιστον μία φορά στον πίνακα.

Παράδειγμα εισόδου

```
12
5 5 5 5 13 13 13 14 14 15 15 15
13
```

Παράδειγμα εξόδου

```
6
```

Άσκηση 5.21

Να δημιουργήσετε πρόγραμμα, το οποίο να βρίσκει το πλήθος των εμφανίσεων ενός ακεραίου αριθμού, μέσα σε ένα ταξινομημένο σε αύξουσα σειρά πίνακα ακεραίων, με τη χρήση δυαδικής αναζήτησης. Το πρόγραμμα, αρχικά, να διαβάζει το πλήθος N (1<=N<=100) των ακεραίων και, ακολούθως, να διαβάζει τους ακέραιους και να τους αποθηκεύει σε πίνακα. Στη συνέχεια, να διαβάζει τον ακέραιο προς αναζήτηση. Να θεωρήσετε ότι ο ακέραιος προς αναζήτηση θα εμφανίζεται τουλάχιστον μία φορά στον πίνακα.

Σημείωση: Να χρησιμοποιήσετε δύο συναρτήσεις δυαδικής αναζήτησης, οι οποίες να βρίσκουν τη θέση της πρώτης και της τελευταίας εμφάνισης του ακέραιου, αντίστοιχα.

Παράδειγμα εισόδου

```
12
5 5 5 5 13 13 13 14 14 15 15 15
15
```

Παράδειγμα εξόδου

```
3
```

Άσκηση 5.22

Ο Αντώνης είναι διαχειριστής των στοιχείων των πελατών του γνωστού ηλεκτρονικού βιβλιοπωλείου «Foraris Books». Το βιβλιοπωλείο έχει τα τηλέφωνα 1,000,000 πελατών, τα οποία είναι ταξινομημένα σε αύξουσα σειρά στο αρχείο phones.txt. Ο Αντώνης θέλει να δημιουργήσει πρόγραμμα το οποίο να διαβάζει από το αρχείο customers.txt τα τηλέφωνα 10 ατόμων και να εμφανίζει στην οθόνη τη λέξη «YES» στην περίπτωση που ένας αριθμός τηλεφώνου ανήκει σε πελάτη του βιβλιοπωλείου, διαφορετικά να εμφανίζει τη λέξη «NO».

Άσκηση 5.23

Να δημιουργήσετε πρόγραμμα, το οποίο να:

(α) διαβάζει 100 ακέραιους αριθμούς από το αρχείο inA.txt και να βρίσκει τον αριθμό με τις περισσότερες εμφανίσεις

(β) διαβάζει 1,000,000 ταξινομημένους σε αύξουσα σειρά ακέραιους από το αρχείο inB.txt και να εμφανίζει το πλήθος των εμφανίσεων του αριθμού που βρέθηκε στο ερώτημα (α).

Άσκηση 5.24

Να δημιουργήσετε πρόγραμμα, το οποίο να βρίσκει τον αμέσως μεγαλύτερο ακέραιο αριθμό από έναν ακέραιο Κ, μέσα σε έναν ταξινομημένο πίνακα ακεραίων, με τη χρήση συνάρτησης δυαδικής αναζήτησης. Το πρόγραμμα, αρχικά, να διαβάζει το πλήθος Ν (1<=N<=100) των ακεραίων και, ακολούθως, να διαβάζει τους ακέραιους και να τους αποθηκεύει σε πίνακα. Στη συνέχεια, να διαβάζει τον ακέραιο Κ. Να θεωρήσετε ότι ο ακέραιος Κ θα είναι μικρότερος από το τελευταίο στοιχείο του πίνακα και μεγαλύτερος ή ίσος από το πρώτο στοιχείο.

Παράδειγμα εισόδου

```
10
15 26 32 44 51 68 73 80 88 97
70
```

Παράδειγμα εξόδου

```
73
```

Άσκηση 5.25

Να δημιουργήσετε πρόγραμμα, το οποίο να βοηθά τον υπολογιστή να μαντέψει τον αριθμό που σκέφτεστε, από το 1 μέχρι το Ν (1<=N<=10,000). Ο υπολογιστής θα δέχεται, αρχικά, έναν ακέραιο Ν, θα χρησιμοποιεί τα βήματα της δυαδικής αναζήτησης, για να βρίσκει κάθε φορά το μεσαίο στοιχείο και θα ζητά από εσάς να τον ενημερώσετε αν ο αριθμός είναι μεγαλύτερος ή μικρότερος από το μεσαίο στοιχείο. Ο αλγόριθμός θα συνεχίσει να «μαντεύει»

μέχρι να βρει τον αριθμό που σκέφτεστε και θα σας ενημερώσει για το πλήθος των προσπαθειών που έχει κάνει. Με έντονη γραφή εμφανίζονται τα δεδομένα εισόδου.

Παράδειγμα εκτέλεσης (για τον αριθμό 400)

```
Input N: 1000
Think of a number between 1 and 1000...
Is it number 500? (bigger/smaller/yes): smaller
Is it number 250? (bigger/smaller/yes): bigger
Is it number 375? (bigger/smaller/yes): bigger
Is it number 437? (bigger/smaller/yes): smaller
Is it number 406? (bigger/smaller/yes): smaller
Is it number 390? (bigger/smaller/yes): bigger
Is it number 398? (bigger/smaller/yes): bigger
Is it number 402? (bigger/smaller/yes): smaller
Is it number 400? (bigger/smaller/yes): yes
Found it after 9 tries!
```

Άσκηση 5.26

Να δημιουργήσετε πρόγραμμα, το οποίο να αναζητά μέσα σε έναν πίνακα Α, όλα τα στοιχεία ενός πίνακα Β. Το πρόγραμμα θα διαβάζει τα πλήθη N, M (1<=N, M<=100) και τα στοιχεία των δύο πινάκων και αν όλα τα στοιχεία του πίνακα Β εμφανίζονται μέσα στον πίνακα Α, το πρόγραμμα να εμφανίζει το μήνυμα «All items found», αλλιώς να εμφανίζει το μήνυμα «Some not found».

Παράδειγμα εισόδου

```
15
12 24 4 54 44 31 78 173 83 82 97 14 23 99 61
5
31 23 12 61 4
```

Παράδειγμα εξόδου

```
All items found
```

Άσκηση 5.27

Να βρείτε τη χρονική πολυπλοκότητα της πιο κάτω συνάρτησης. Να συμπληρώσετε τον πίνακα προκαταρκτικής εκτέλεσης και να βρείτε το αποτέλεσμα, αν δοθεί αρχικά N = 12345.

```
int my_function(int N) {
  int ans = 0;
      while (N > 0){
            ans += N % 10;
            N /= 10;
      }
return ans;
}
```

Άσκηση 5.28

Να βρείτε τη χρονική πολυπλοκότητα των συναρτήσεων f1, f2 και f3.

```
#include<iostream>
#include<cmath>
using namespace std;
int f1(){
    int x, y;
    cin >> x;
    if(x>0)
        y = x % 10;
    else
        y = abs(x);
return y;
}
void f2(int N){
    for(int i=0; i<N; i++){
        for(int j=0; j<N; j++)
            cout << i * j << " ";
        cout << endl;
    }
}
int f3(int x, int y){
    int s = 0;
    for(int i=1; i<=x; i++)
        s += i;
    for(int j=1; j<=y; j++)
        s += j;
return s;
}
int main(){
    cout << f1() << endl << f2(4);
    cout << f3(3, 4);
return 0;
}
```

Άσκηση 5.29

Να δημιουργήσετε πρόγραμμα, το οποίο να χρησιμοποιεί δύο συναρτήσεις αναζήτησης (σειριακής και δυαδικής), για να αναζητήσει έναν ακέραιο ανάμεσα στους αριθμούς που βρίσκονται σε ένα αρχείο. Το αρχείο θα περιέχει μέχρι 20,000 ακέραιους αριθμούς, ταξινομημένους σε αύξουσα σειρά. Οι συναρτήσεις να επιστρέφουν το πλήθος των συγκρίσεων που θα χρειαστεί η κάθε μία για να εντοπίσει τον αριθμό. Να γίνει χρήση των αρχείων 1000s.txt, 5000s.txt, 20000s.txt και unsorted.txt (μη ταξινομημένοι ακέραιοι).

Παράδειγμα εισόδου (αρχείο και πληκτρολόγιο)

```
16 29 31 32 49 65
49
```

Παράδειγμα εξόδου (οθόνη)

```
Found with sequential search. Steps: 5
Found with binary search. Steps: 2
```

Άσκηση 5.30

Να δημιουργήσετε πρόγραμμα, το οποίο να καλεί τον χρήστη να μαντέψει έναν αριθμό από το 1 μέχρι το 1000. Αν ο χρήστης δώσει μεγαλύτερο ή μικρότερο αριθμό, να εμφανίζει τα μηνύματα «Smaller» ή «Bigger», αντίστοιχα. Η διαδικασία αυτή συνεχίζεται μέχρι ο χρήστης να βρει τον ζητούμενο αριθμό. Μόλις βρει τον αριθμό ο χρήστης, θα ενημερώνεται για το πλήθος των προσπαθειών που χρειάστηκε και το πρόγραμμα θα τον ρωτά αν θέλει να συνεχίσει το παιχνίδι. Ο χρήστης με το που θα τερματίσει τη λειτουργία του προγράμματος, θα ενημερώνεται για το προσωπικό του ρεκόρ, δηλαδή τον ελάχιστο αριθμό προσπαθειών που χρειάστηκε για να βρει τον αριθμό σε όλα τα παιχνίδια.

Να χρησιμοποιήσετε την πιο κάτω συνάρτηση η οποία παράγει έναν τυχαίο αριθμό στο διάστημα 1 μέχρι 1000. Αυτός θα είναι ο αριθμός ο οποίος θα «σκέφτεται» ο υπολογιστής και θα πρέπει εσείς να μαντέψετε:

```
#include <iostream>
#include <cstdlib>
#include <time.h>
using namespace std;

int generate_number() {
    srand(time(NULL));
    int random = rand()%1000+1;
return random;
}
```

Παράδειγμα εκτέλεσης (με έντονη γραφή τα δεδομένα του χρήστη)

```
Guess the number I am thinking (1-1000): 500
Bigger
750
Smaller
625
Bigger
685
Smaller
655
Smaller
640
Bigger
647
Smaller
643
Bigger
645
Bigger
```

```
646
You found it. Tries: 10
Do you want to continue (y/n): n
Your record is: 10
```

Ασκήσεις Εμπλουτισμού

Άσκηση 5.31

Να δημιουργήσετε πρόγραμμα, το οποίο να βρίσκει την τετραγωνική ρίζα ενός ακέραιου θετικού αριθμού N (1<=N<=10,000), χωρίς τη χρήση της συνάρτησης `sqrt`. Το πρόβλημα μπορεί να λυθεί εφαρμόζοντας τον αλγόριθμο της δυαδικής αναζήτησης στο διάστημα [1…N], ως εξής:

Αρχικά, θα ορίσουμε: `low=1.0, hi=N, diff=abs(hi-low)` και `EPS=0.0000000001`. Για όσο `diff>=EPS`, ορίζουμε `mid=(low+hi)/2.0` και ελέγχουμε κάθε φορά αν `mid*mid>N`, οπόταν θα ορίσουμε το `hi=mid`, αλλιώς θα ορίσουμε το `low=mid`. Σε κάθε επανάληψη θα υπολογίζουμε ξανά το `diff=abs(hi-low)`. Το αποτέλεσμα να εμφανίζεται με πέντε δεκαδικά ψηφία.

Παράδειγμα εισόδου

```
5
```

Παράδειγμα εξόδου

```
2.23607
```

Άσκηση 5.32

Σε ένα κέντρο εκπαίδευσης νεοσυλλέκτων οι φαντάροι, την ώρα της πρωινής αναφοράς, στοιχίζονται στις γραμμές τους κατά ύψος, από τον πιο κοντό στον πιο ψηλό. Η πρωινή αναφορά κάθε λόχου μπορεί να διαρκέσει αρκετή ώρα, έτσι οι φαντάροι που τελειώνουν τα καθήκοντά τους (σκοπιά, θαλαμοφυλακή, μαγειρεία κ.λπ.) υποχρεούνται να παρουσιαστούν. Ο κάθε φαντάρος βρίσκει τη θέση του στη γραμμή, εντοπίζοντας τον λίγο κοντύτερο και τον λίγο ψηλότερο από αυτόν φαντάρο και εισέρχεται στη γραμμή μεταξύ τους.

Αν σας δοθούν τα ύψη των N φαντάρων (1<=N<=100), να βρείτε μεταξύ ποιων φαντάρων θα εισέλθουν στη γραμμή οι φαντάροι που καθυστέρησαν. Αν δεν υπάρχει άλλος φαντάρος πριν ή μετά, να εμφανίζεται ο χαρακτήρας 'X'. Να θεωρήσετε ότι πάντοτε θα συγκρίνετε τα ύψη της αρχικής διάταξης των φαντάρων και ότι δεν θα υπάρχουν δύο φαντάροι με το ίδιο ύψος.

Παράδειγμα εισόδου

```
4
2 4 5 7
5
1 3 6 8 10
```

Παράδειγμα εξόδου

```
X 2
2 4
5 7
7 X
7 X
```

Επεξήγηση παραδείγματος: Έχουμε 4 φαντάρους που στέκονται στη σειρά με τα ακόλουθα ύψη: 2, 4, 5, 7. Έχουμε 5 φαντάρους που καθυστέρησαν και θέλουν να σταθούν στη σειρά. Ο φαντάρος με ύψος 1 θα σταθεί στη σειρά μπροστά από τον φαντάρο με ύψος 2. Ο φαντάρος με ύψος 3 θα σταθεί στη σειρά μεταξύ των φαντάρων με ύψη 2 και 4. Ο φαντάρος με ύψος 6 θα σταθεί στη σειρά μεταξύ των φαντάρων με ύψη 5 και 7. Οι φαντάροι με ύψη 8 και 10 θα σταθούν στη σειρά μετά τον φαντάρο με ύψος 7.

Άσκηση 5.33

Ο Παντελής έχει μία εξίσωση της μορφής $f(x)=x^a-bx-c$. Θέλει να βρει, μέσα σε ένα πεδίο τιμών μεταξύ L και H, ποια είναι η μικρότερη τιμή του x για την οποία η f(x) θα είναι μεγαλύτερη από έναν ακέραιο K. Για παράδειγμα, αν a=2, b=3, c=4, L=0, H=20, και K=10, τότε η απάντηση θα είναι 6.

$f(5)=5^2-3*5-4=6$, (6<10)
$f(6)=6^2-3*6-4=14$, (OK)
$f(7)=7^2-3*7-4=24$, (24>10)

Παράδειγμα εισόδου

```
2 3 4 0 20 10
```

Παράδειγμα εξόδου

```
6
```

Άσκηση 5.34

Ο Batman, ο Superman και ο Captain America ψάχνουν να βρουν μία ομάδα κακοποιών, οι οποίοι έχουν κρυφτεί σε ένα κτήριο της 5ης λεωφόρου της Νέας Υόρκης. Για να μην ψάχνουν στα ίδια σημεία, οι ήρωες έχουν συμφωνήσει να ψάξουν τα κτήρια ως εξής:

- Ο Batman θα ξεκινήσει το ψάξιμο από το πρώτο κτήριο που βρίσκεται στα αριστερά και θα μετακινείται ένα-ένα κτήριο προς τα δεξιά.
- Ο Captain America θα ξεκινήσει το ψάξιμο από το πρώτο κτήριο που βρίσκεται στα δεξιά και θα μετακινείται ένα-ένα κτήριο προς τα αριστερά.
- Ο Superman, επειδή μπορεί να πετάει, θα εφαρμόσει τον αλγόριθμο της δυαδικής αναζήτησης, για να ψάξει τα κτήρια. Θα ψάξει, δηλαδή, πρώτα το κτήριο που βρίσκεται στο μέσο της 5ης λεωφόρου και θα κινηθεί, αναλόγως, αριστερά ή δεξιά.

Αν σας δοθούν N (1<=N<=1000) ακέραιοι αριθμοί ταξινομημένοι σε αύξουσα σειρά οι οποίοι υποδεικνύουν τους αριθμούς των κτηρίων και, στη συνέχεια, ένας ακέραιος X που υποδεικνύει το κτήριο που έχουν καταφύγει οι κακοποιοί, Να δημιουργήσετε πρόγραμμα, το οποίο να εμφανίζει το όνομα του ήρωα που θα καταφέρει να εντοπίσει πρώτος τους

κακοποιούς, ψάχνοντας τα λιγότερα κτήρια. Αν κάποιοι ήρωες βρουν ταυτόχρονα τους κακοποιούς, να εμφανίζονται τα ονόματά τους με αλφαβητική σειρά.

Παράδειγμα εισόδου

```
9
20 30 40 50 60 100 200 500 1000
500
```

Παράδειγμα εξόδου

```
Captain America
```

Άσκηση 5.35

Ο Παντελής ψάχνει επειγόντως να βρει ένα βιβλίο μέσα σε μία τεράστια βιβλιοθήκη. Η βιβλιοθήκη αποτελείται από μία σειρά από C ράφια. Σε κάθε ράφι υπάρχουν X βιβλία. Ευτυχώς, ο Παντελής έχει μαζί του κάποιους φίλους του για να τον βοηθήσουν. Θα χωρίσει τη βιβλιοθήκη σε N τμήματα για κάθε έναν από τους N φίλους του και κάθε ένας από αυτούς θα ψάξει στο δικό του τμήμα, χωρίς να επεμβαίνει στο γειτονικό τμήμα που ψάχνει ο διπλανός του. Ο Παντελής πρέπει να βρει τρόπο να ελαχιστοποιήσει τον αριθμό των βιβλίων που πρέπει να ψάξει κάποιος από τους φίλους του, στη χειρότερη περίπτωση, ώστε να μην τους ταλαιπωρήσει. Ας δούμε ένα παράδειγμα:

Η βιβλιοθήκη αποτελείται από C=9 ράφια, ο Παντελής έχει μαζί του N=3 φίλους του και το πλήθος των βιβλίων σε κάθε ράφι είναι: `10 20 30 40 50 60 70 80 90`

Ο Παντελής θα πρέπει να χωρίσει τα ράφια σε τρία τμήματα όπως πιο κάτω:

`10 20 30 40 50 | 60 70 | 80 90`

Ο φίλος του Παντελή στο πρώτο τμήμα από αριστερά θα πρέπει να ψάξει ανάμεσα σε 150 βιβλία (10+20+30+40+50). Ο φίλος του που θα ψάξει στο δεύτερο τμήμα, θα ψάξει ανάμεσα σε 130 βιβλία (60+70) και ο τρίτος ανάμεσα σε 170 βιβλία (80+90). Κανένας άλλος διαχωρισμός δεν δίνει μικρότερο αριθμό βιβλίων.

Να δημιουργήσετε πρόγραμμα, το οποίο θα δέχεται έναν ακέραιο C (1<=C<=1000) που υποδεικνύει το πλήθος των ραφιών, ακολουθούμενο από C ακέραιους που υποδεικνύουν το πλήθος των βιβλίων σε κάθε ράφι. Τέλος, το πρόγραμμα να δέχεται έναν ακέραιο N (1<=N<=100) που υποδεικνύει το πλήθος των φίλων του Παντελή και να επιστρέφει έναν ακέραιο M, τον ελάχιστο αριθμό βιβλίων που κάποιος φίλος του Παντελή πρέπει να ψάξει στη χειρότερη περίπτωση.

Παράδειγμα εισόδου

```
9
10 20 30 40 50 60 70 80 90
5
```

Παράδειγμα εξόδου

```
110
```

Επεξήγηση: Ο ιδανικός τρόπος για να χωριστεί η βιβλιοθήκη εμφανίζεται πιο κάτω. Στην χειρότερη περίπτωση, θα χρειαστεί να ψάξει κάποιος ανάμεσα σε 110 βιβλία (50+60).

`10 20 30 40 | 50 60 | 70 | 80 | 90`

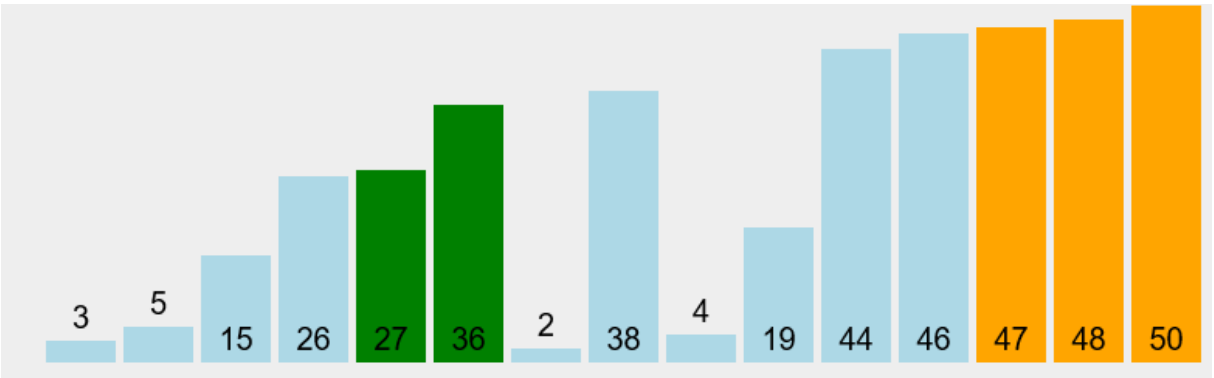
Γ7.6 Αλγόριθμοι Ταξινόμησης (Sorting Algorithms)

Τι θα μάθουμε σε αυτό το κεφάλαιο:

- Να ορίζουμε τι είναι η ταξινόμηση δεδομένων
- Να διακρίνουμε την ταξινόμηση σε αύξουσα σειρά από την ταξινόμηση σε φθίνουσα σειρά
- Να αναφέρουμε γνωστούς αλγόριθμους ταξινόμησης
- Να περιγράφουμε σύντομα τι είναι η χρονική πολυπλοκότητα (execution time complexity) ενός αλγόριθμου ως συνάρτηση του μεγέθους των δεδομένων εισόδου, n, με βάση τον συμβολισμό *O*
- Να υπολογίζουμε πρακτικά τη χρονική πολυπλοκότητα απλών αλγορίθμων, αναγνωρίζοντας και μετρώντας τον αριθμό των εμφωλιασμένων βρόγχων τους, μέχρι $O(n^3)$
- Να αναγνωρίζουμε τον αλγόριθμο της μεθόδου ταξινόμησης της φυσαλίδας (bubble sort) και τον αντίστοιχο κώδικα για υλοποίησή του σε προβλήματα που χρησιμοποιούν μονοδιάστατους πίνακες
- Να υλοποιούμε τον αλγόριθμο της μεθόδου ταξινόμησης της φυσαλίδας (bubble sort) για ταξινόμηση σε αύξουσα και σε φθίνουσα σειρά για πίνακες διάφορων τύπων δεδομένων
- Να αναγνωρίζουμε τον αλγόριθμο της μεθόδου ταξινόμησης της εισαγωγής (insertion sort) και τον αντίστοιχο κώδικα για υλοποίησή του σε προβλήματα που χρησιμοποιούν μονοδιάστατους πίνακες
- Να υλοποιούμε τον αλγόριθμο της μεθόδου ταξινόμησης της εισαγωγής (insertion sort) για ταξινόμηση σε αύξουσα και σε φθίνουσα σειρά για πίνακες διάφορων τύπων δεδομένων
- Να υπολογίζουμε γενικά τη χρονική πολυπλοκότητα των δύο αυτών αλγορίθμων, μετρώντας τους βρόγχους που τους αποτελούν
- Να συγκρίνουμε την ταχύτητα εκτέλεσης των δύο πιο πάνω αλγορίθμων ταξινόμησης, μέσω αριθμητικών παραδειγμάτων με βάση το θεωρητικό υπόβαθρο της χρονικής πολυπλοκότητάς τους αλλά και επιβεβαιώνοντας τα αποτελέσματα πρακτικά με εκτέλεση των αντίστοιχων προγραμμάτων στον ηλεκτρονικό υπολογιστή, σε πραγματικό χρόνο, με διάφορα μεγέθη εισόδου, n
- Να γνωρίζουμε παραλλαγές των αλγορίθμων ταξινόμησης (π.χ. με στοιχεία άλλου τύπου δεδομένων, ταξινόμηση σε φθίνουσα σειρά κ.λπ.).

1. Εισαγωγή

Με τον όρο ταξινόμηση εννοούμε τη διάταξη (permutation) των στοιχείων ενός συνόλου σε μία συγκεκριμένη σειρά, αύξουσα ή φθίνουσα. Αποτελεί προγραμματιστική τεχνική, η οποία χρησιμοποιείται πολύ συχνά, καθώς διευκολύνεται η αναζήτηση και ο εντοπισμός στοιχείων του αντίστοιχου συνόλου. Στο κεφάλαιο αυτό θα ασχοληθούμε με δύο αλγόριθμους ταξινόμησης οι οποίοι χρησιμοποιούν τεχνικές ανταλλαγής, τον αλγόριθμο ταξινόμησης φυσαλίδας (bubble sort) και τον αλγόριθμο ταξινόμησης με εισαγωγή (insertion sort). Ένα εργαλείο οπτικοποίησης, το οποίο αναπαριστά τον τρόπο λειτουργίας αυτών των αλγόριθμων, μπορείτε να βρείτε στην ιστοσελίδα http://visualgo.net/sorting.



Εικόνα 1: Εργαλείο Visualgo

2. Αλγόριθμος φυσαλίδας (Bubble sort)

Ο αλγόριθμος φυσαλίδας (bubble sort) είναι ένας από τους πιο γνωστούς αλλά και λιγότερο αποδοτικούς αλγόριθμους ταξινόμησης. Το όνομα του αλγόριθμου προέρχεται από τον τρόπο ταξινόμησης: τα μεγαλύτερα στοιχεία κατευθύνονται προς το τέλος, όπως οι φυσαλίδες που αναδύονται στην επιφάνεια. Ο αλγόριθμος βασίζεται στην αρχή της σύγκρισης και αντιμετάθεσης δύο γειτονικών στοιχείων, μέχρι να διαταχθούν όλα τα στοιχεία του πίνακα. Ξεκινώντας από τα πρώτα δύο στοιχεία, ο αλγόριθμος διασχίζει κάθε φορά όλα τα στοιχεία του πίνακα και θα χρειαστεί να κάνει αρκετές διασχίσεις μέχρι την τελική διάταξη των στοιχείων.

Πιο κάτω βλέπουμε τη συνάρτηση του αλγόριθμου φυσαλίδας, η οποία είναι τύπου void και ταξινομεί τα στοιχεία του πίνακα σε αύξουσα σειρά. Η υλοποίηση γίνεται ως εξής:

```
// Συνάρτηση αλγόριθμου ταξινόμησης φυσαλίδας, όπου arr είναι ο
// πίνακας και N είναι το πλήθος των στοιχείων του πίνακα
void bubblesort(int arr[], int N){

 bool sorted;               // Μεταβλητή λογικού τύπου
 int temp;                  // Βοηθητική μεταβλητή αντιμετάθεσης

 do{
  sorted = true;
  for (int i=0; i<N-1; i++){
     if (arr[i]>arr[i+1]){      // Αν το επόμενο στοιχείο είναι
           temp = arr[i];       // μικρότερο, τότε αντιμεταθέτουμε
           arr[i] = arr[i+1];   // τα στοιχεία με τη χρήση της
           arr[i+1] = temp;     // βοηθητικής μεταβλητής (temp)
           sorted = false;      // Αλλαγή τιμής της μεταβλητής
        }                       // σημαίνει ότι ο πίνακας δεν είναι
      }                         // ταξινομημένος
  } while (sorted==false);
}                               // Τέλος συνάρτησης
```

Στον πιο πάνω κώδικα, χρησιμοποιούμε μία μεταβλητή λογικού τύπου (`sorted`), η οποία θα σηματοδοτεί αν ο πίνακας είναι ταξινομημένος ή όχι. Πριν από κάθε διάσχιση του πίνακα, αρχικοποιούμε αυτή τη μεταβλητή με κάποια συγκεκριμένη τιμή (`sorted=true`). Αν κατά την σύγκριση δύο τιμών διαπιστωθεί ότι χρειάζεται να αντιμετατεθούν, τότε αλλάζουμε την τιμή της μεταβλητής (`sorted=false`). Μετά το τέλος της διάσχισης, ελέγχουμε την τιμή της μεταβλητής. Αν η μεταβλητή διατηρεί την αρχική τιμή της (`true`), αυτό σημαίνει ότι δεν έχει γίνει κάποια αντιμετάθεση τιμών, επομένως ο πίνακας είναι ταξινομημένος και δεν χρειάζεται να γίνει άλλη διάσχιση. Αν η μεταβλητή έχει αλλάξει τιμή (`false`), σημαίνει ότι κάποιες τιμές έχουν αντιμετατεθεί, επομένως ο πίνακας δεν είναι ταξινομημένος και απαιτείται και άλλη διάσχιση.

Παράδειγμα

Ας υποθέσουμε ότι έχουμε τον πίνακα **arr = {5, 6, 2, 4, 1, 9, 3, 0, 8, 7}** μεγέθους **10** και ότι θέλουμε να ταξινομηθεί σε **αύξουσα σειρά**. Στην πρώτη διάσχιση ο αλγόριθμος συγκρίνει τα δύο πρώτα στοιχεία του πίνακα (5, 6). Αφού το 5 είναι μικρότερο του 6, τα στοιχεία δεν θα αντιμετατεθούν:

5	**6**	2	4	1	9	3	0	8	7
0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*

Ο αλγόριθμος συνεχίζει συγκρίνοντας τα επόμενα δύο στοιχεία (6, 2). Αφού το 6 είναι μεγαλύτερο του 2, τα στοιχεία θα αντιμετατεθούν:

5	**2**	**6**	4	1	9	3	0	8	7
0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*

Ο αλγόριθμος συνεχίζει συγκρίνοντας τα επόμενα δύο στοιχεία (6, 4). Αφού το 6 είναι μεγαλύτερο του 4, τα στοιχεία θα αντιμετατεθούν:

5	2	**4**	**6**	1	9	3	0	8	7
0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*

Ο αλγόριθμος συνεχίζει συγκρίνοντας τα επόμενα δύο στοιχεία (6, 1). Αφού το 6 είναι μεγαλύτερο του 1, τα στοιχεία θα αντιμετατεθούν:

5	2	4	**1**	**6**	9	3	0	8	7
0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*

Ο αλγόριθμος συνεχίζει συγκρίνοντας τα επόμενα δύο στοιχεία (6, 9). Αφού το 6 είναι μικρότερο του 9, τα στοιχεία δεν θα αντιμετατεθούν:

5	2	4	1	**6**	**9**	3	0	8	7
0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*

Ο αλγόριθμος συνεχίζει συγκρίνοντας τα επόμενα δύο στοιχεία (9, 3). Αφού το 9 είναι μεγαλύτερο του 3, τα στοιχεία θα αντιμετατεθούν:

5	2	4	1	6	**3**	**9**	0	8	7
0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*

Ο αλγόριθμος συνεχίζει συγκρίνοντας τα επόμενα δύο στοιχεία (9, 0). Αφού το 9 είναι μεγαλύτερο του 0, τα στοιχεία θα αντιμετατεθούν:

5	2	4	1	6	**3**	**0**	**9**	8	7
0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*

Ο αλγόριθμος συνεχίζει συγκρίνοντας τα επόμενα δύο στοιχεία (9, 8). Αφού το 9 είναι μεγαλύτερο του 8, τα στοιχεία θα αντιμετατεθούν:

5	2	4	1	6	3	0	**8**	**9**	7
0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*

Ο αλγόριθμος συνεχίζει συγκρίνοντας τα επόμενα δύο στοιχεία (9, 7). Αφού το 9 είναι μεγαλύτερο του 7, τα στοιχεία θα αντιμετατεθούν:

5	2	4	1	6	3	0	8	**7**	**9**
0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*

Μετά το τέλος της πρώτη διάσχισης, παρατηρούμε ότι το μεγαλύτερο στοιχείο του πίνακα έχει τοποθετηθεί στην τελευταία θέση:

5	2	4	1	6	3	0	8	7	**9**
0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*

Μετά το τέλος της δεύτερης διάσχισης ο πίνακας θα έχει την εξής μορφή:

2	4	1	5	3	0	6	7	**8**	**9**
0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*

Μετά το τέλος της τρίτης διάσχισης ο πίνακας θα έχει την εξής μορφή:

2	1	4	3	0	5	6	**7**	**8**	**9**
0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*

Μετά το τέλος της τέταρτης διάσχισης ο πίνακας θα έχει την εξής μορφή:

1	2	3	0	4	5	**6**	**7**	**8**	**9**
0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*

Μετά το τέλος της πέμπτης διάσχισης ο πίνακας θα έχει την εξής μορφή:

1	2	0	3	4	**5**	**6**	**7**	**8**	**9**
0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*

Μετά το τέλος της έκτης διάσχισης ο πίνακας θα έχει την εξής μορφή:

1	0	2	3	**4**	**5**	**6**	**7**	**8**	**9**
0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*

Μετά το τέλος της έβδομης διάσχισης ο πίνακας θα έχει την εξής μορφή:

0	1	2	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*

Με την όγδοη διάσχιση διαπιστώνουμε ότι ο πίνακας είναι πλέον ταξινομημένος και δεν απαιτούνται άλλες διασχίσεις.

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*

3. Βελτιστοποίηση αλγόριθμου

Στην πιο πάνω οπτικοποίηση παρατηρούμε ότι μετά από μία διάσχιση τα εκάστοτε μεγαλύτερα στοιχεία τοποθετούνται στις τελευταίες θέσεις του πίνακα. Συμπεραίνουμε ότι σε κάθε επόμενη διάσχιση μπορούμε να ελέγχουμε μία λιγότερη θέση του πίνακα, ώστε να μην συγκρίνουμε ξανά τα ήδη ταξινομημένα στοιχεία. Αυτό υλοποιείται με τη σταδιακή μείωση, μετά από κάθε διάσχιση, της τιμής του μεγέθους του πίνακα μέσα στον βρόχο do. Έτσι, βελτιώνουμε, σε κάποιο βαθμό, την απόδοση του αλγόριθμου.

```
void bubblesort(int arr[], int N){
 bool sorted;
 int temp, S = N;                  // Το S είναι η μεταβλητή που θα
                                   // βάλουμε στη θέση του πλήθους
 do{                               // των στοιχείων του πίνακα
  sorted = true;                   // Μεταβλητή λογικού τύπου
  for (int i=0; i<S-1; i++){       // Η διάσχιση δεν θα γίνεται
                                   // κάθε φορά σε ολόκληρο τον πίνακα
      if (arr[i]>arr[i+1]){        // Αν το επόμενο στοιχείο είναι
           temp = arr[i];          // μικρότερο, τότε αντιμεταθέτουμε
           arr[i] = arr[i+1];      // τα στοιχεία με τη χρήση της
           arr[i+1] = temp;        // βοηθητικής μεταβλητής (temp)
           sorted = false;         // Αλλαγή τιμής της μεταβλητής
         }                         // σημαίνει ότι ο πίνακας δεν είναι
       }                           // ταξινομημένος
      S--;                         // Μειώνουμε το πεδίο διάσχισης
  } while (sorted==false);
}                                  // Τέλος συνάρτησης
```

Παράδειγμα 6.1

Να δημιουργήσετε πρόγραμμα, το οποίο να καλεί μία συνάρτηση ταξινόμησης φυσαλίδας, η οποία να ταξινομεί σε αύξουσα σειρά έναν πίνακα ακεραίων. Το πρόγραμμα, αρχικά, θα διαβάζει το πλήθος N (1<=N<=1000) των ακεραίων και, ακολούθως, θα διαβάζει τους ακέραιους και θα τους αποθηκεύει σε πίνακα. Στη συνέχεια, θα καλεί τη συνάρτηση η οποία θα ταξινομεί τον πίνακα και θα εμφανίζει τα στοιχεία του πίνακα μετά από κάθε αντιμετάθεση, καθώς και την τελική τους διάταξη, όπως στο παράδειγμα εξόδου πιο κάτω.

Παράδειγμα εισόδου

```
10
5 6 2 4 1 9 3 0 8 7
```

Παράδειγμα εξόδου

```
Pass = 1:
5 2 6 4 1 9 3 0 8 7
5 2 4 6 1 9 3 0 8 7
5 2 4 1 6 9 3 0 8 7
5 2 4 1 6 3 9 0 8 7
5 2 4 1 6 3 0 9 8 7
5 2 4 1 6 3 0 8 9 7
5 2 4 1 6 3 0 8 7 9
Pass = 2:
2 5 4 1 6 3 0 8 7 9
2 4 5 1 6 3 0 8 7 9
2 4 1 5 6 3 0 8 7 9
2 4 1 5 3 6 0 8 7 9
```

```
2 4 1 5 3 0 6 8 7 9
2 4 1 5 3 0 6 7 8 9
Pass = 3:
2 1 4 5 3 0 6 7 8 9
2 1 4 3 5 0 6 7 8 9
2 1 4 3 0 5 6 7 8 9
Pass = 4:
1 2 4 3 0 5 6 7 8 9
1 2 3 4 0 5 6 7 8 9
1 2 3 0 4 5 6 7 8 9
Pass = 5:
1 2 0 3 4 5 6 7 8 9
Pass = 6:
1 0 2 3 4 5 6 7 8 9
Pass = 7:
0 1 2 3 4 5 6 7 8 9
Pass = 8:
final: 0 1 2 3 4 5 6 7 8 9
```

```
#include <iostream>
using namespace std;
#define MAXN 1000

void bubblesort(int arr[], int N){
 bool sorted;
 int temp, cnt = 0;

do{
 sorted = true;
 cout << "Pass = " << ++cnt << ":" << endl; // Αριθμός διάσχισης
 for (int i=0; i<N-1; i++){
     if (arr[i]>arr[i+1]){
           temp = arr[i];
           arr[i] = arr[i+1];
           arr[i+1] = temp;
           sorted = false;
           for (int i=0; i<N; i++)          // Εμφάνιση στοιχείων
               cout << arr[i] << " ";
           cout << endl;
       }
    }
 } while (sorted==false);
}
```

```
int main(){
  int numbers[MAXN], N;                 // Ο πίνακας δηλώνεται με το
  cin >> N;                             // μέγιστο δυνατό πλήθος
  for (int i=0; i<N; i++)               // Εισαγωγή στοιχείων
     cin >> numbers[i];

 bubblesort(numbers, N);                // Κλήση συνάρτησης
 cout << "final: ";
 for (int i=0; i<N; i++)                // Εμφάνιση τελικής διάταξης
     cout << numbers[i] << " ";
return 0;
}
```

(Code: C7_6_example1.cpp)

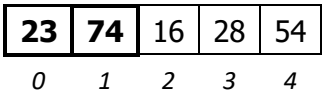
4. Εναλλακτική προσέγγιση

Για την υλοποίηση του αλγόριθμου ταξινόμησης φυσαλίδας υπάρχουν πολλές διαφορετικές προσεγγίσεις. Μία από αυτές παρουσιάζεται πιο κάτω, η οποία όμως δεν συγκρίνει διαδοχικά στοιχεία του πίνακα. Η συγκεκριμένη υλοποίηση κάνει χρήση δύο βρόχων for (δύο μεταβλητές-δείκτες) και ξεκινώντας από το πρώτο στοιχείο, το συγκρίνει με όλα τα υπόλοιπα στοιχεία του πίνακα. Όταν φτάσει στο τέλος του πίνακα, τότε προχωρά στο δεύτερο στοιχείο για να το συγκρίνει με όλα τα υπόλοιπα, εκτός από το πρώτο κ.ο.κ. Η πιο κάτω συνάρτηση δεν ελέγχει αν ο πίνακας έχει ταξινομηθεί και τερματίζει πάντα όταν συγκριθεί το προτελευταίο με το τελευταίο στοιχείο.

```
// Συνάρτηση ταξινόμησης φυσαλίδας με δύο βρόχους for

void bubblesort(int arr[], int N){

  int temp;
  for (int i=0; i<N-1; i++){
       for (int j=i+1; j<N; j++){
           if (arr[i]>arr[j]){    // Αν το στοιχείο στη θέση j είναι
              temp = arr[i];      // μικρότερο, τότε αντιμεταθέτουμε
              arr[i] = arr[j];    // τα στοιχεία με τη χρήση της
              arr[j] = temp;      // βοηθητικής μεταβλητής (temp)
           }
       }
    }
}                                 // Τέλος συνάρτησης
```

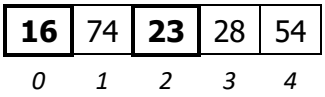
Παράδειγμα

Ας υποθέσουμε ότι έχουμε τον πίνακα **arr = {23, 74, 16, 28, 54}** μεγέθους **5** και ότι θέλουμε να ταξινομηθεί σε **αύξουσα σειρά**. Στην πρώτη διάσχιση ο αλγόριθμος συγκρίνει το πρώτο στοιχείο με το δεύτερο στοιχείο του πίνακα (23, 74). Αφού το 23 είναι μικρότερο του 74, τα στοιχεία δεν θα αντιμετατεθούν:



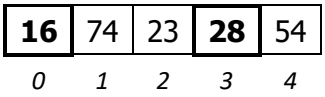

```
i = 0, j = 1
```

Ο αλγόριθμος συνεχίζει συγκρίνοντας το πρώτο στοιχείο με το τρίτο (23, 16). Αφού το 23 είναι μεγαλύτερο του 16, τα στοιχεία θα αντιμετατεθούν:



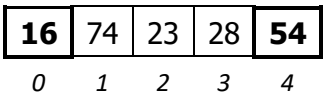

```
i = 0, j = 2
```

Ο αλγόριθμος συνεχίζει συγκρίνοντας το πρώτο στοιχείο με το τέταρτο (16, 28). Αφού το 16 είναι μικρότερο του 28, τα στοιχεία δεν θα αντιμετατεθούν:



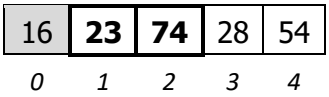

```
i = 0, j = 3
```

Ο αλγόριθμος συνεχίζει συγκρίνοντας το πρώτο στοιχείο με το πέμπτο (16, 54). Αφού το 16 είναι μικρότερο του 54, τα στοιχεία δεν θα αντιμετατεθούν:



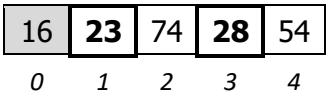

```
i = 0, j = 4
```

Παρατηρούμε ότι το μικρότερο στοιχείο του πίνακα έχει τοποθετηθεί στην πρώτη θέση. Ο αλγόριθμος προχωρεί στη δεύτερη διάσχιση, για να συγκρίνει το δεύτερο στοιχείο του πίνακα με όλα τα υπόλοιπα στοιχεία. Ο αλγόριθμος συγκρίνει το δεύτερο στοιχείο με το τρίτο στοιχείο (74, 23). Αφού το 74 είναι μεγαλύτερο του 23, τα στοιχεία θα αντιμετατεθούν:



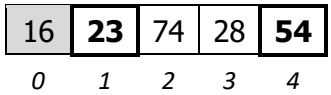

```
i = 1, j = 2
```

Ο αλγόριθμος συγκρίνει το δεύτερο στοιχείο με το τέταρτο (23, 28). Αφού το 23 είναι μικρότερο του 28, τα στοιχεία δεν θα αντιμετατεθούν:



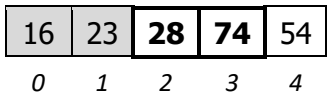

```
i = 1, j = 3
```

Ο αλγόριθμος συγκρίνει το δεύτερο στοιχείο με το πέμπτο (23, 54). Αφού το 23 είναι μικρότερο του 54, τα στοιχεία δεν θα αντιμετατεθούν:



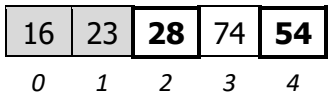

```
i = 1, j = 4
```

Ο αλγόριθμος συγκρίνει το τρίτο στοιχείο με το τέταρτο (74, 28). Αφού το 74 είναι μεγαλύτερο του 28, τα στοιχεία θα αντιμετατεθούν:



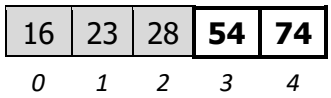

```
i = 2, j = 3
```

Ο αλγόριθμος συγκρίνει το τρίτο στοιχείο με το πέμπτο (28, 54). Αφού το 28 είναι μικρότερο του 54, τα στοιχεία δεν θα αντιμετατεθούν:



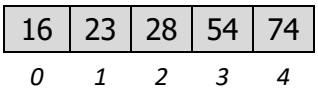

```
i = 2, j = 4
```

Ο αλγόριθμος συγκρίνει το τέταρτο στοιχείο με το πέμπτο (74, 54). Αφού το 74 είναι μεγαλύτερο του 54, τα στοιχεία θα αντιμετατεθούν.




```
i = 3, j = 4
```

Ο πίνακας έχει ταξινομηθεί:



5. Σύγκριση απόδοσης συναρτήσεων ταξινόμησης φυσαλίδας

Για να συγκρίνουμε την απόδοση των δύο συναρτήσεων, χρησιμοποιούμε τον πίνακα **arr = {2, 1, 3, 4, 5, 6, 7, 8}**, ο οποίος μπορεί να ταξινομηθεί με μία μόνο αντιμετάθεση. Δημιουργούμε πρόγραμμα (`C7_6_example2.cpp` – **βρίσκεται στο επιπρόσθετο υλικό**) το οποίο καλεί τις δύο συναρτήσεις, παρουσιάζει τον πίνακα μετά από κάθε αντιμετάθεση και εμφανίζει το συνολικό πλήθος «βημάτων» κάθε συνάρτησης. «Βήμα» θεωρούμε κάθε εκτέλεση του εσωτερικού βρόχου.

Παράδειγμα εισόδου 1 (bubblesort με μεταβλητή λογικού τύπου)

```
8
2 1 3 4 5 6 7 8
```

Παράδειγμα εξόδου 1

```
Pass = 1:
1 2 3 4 5 6 7 8
Pass = 2:
steps: 13
final: 1 2 3 4 5 6 7 8
```

Παράδειγμα εισόδου 2 (bubblesort με δύο βρόχους for)

```
8
2 1 3 4 5 6 7 8
```

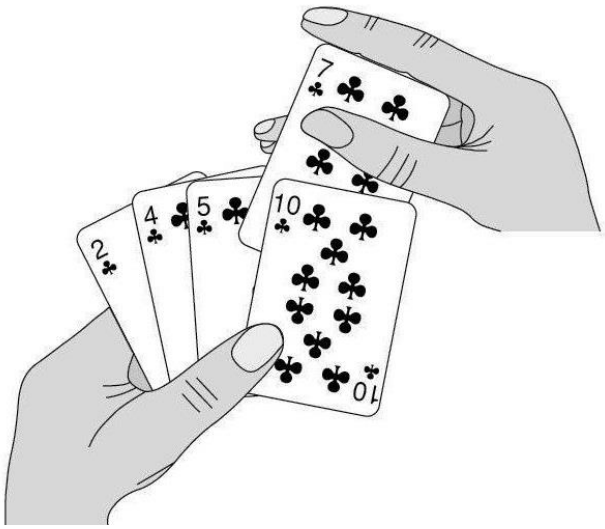
Παράδειγμα εξόδου 2

```
Pass = 1:
1 2 3 4 5 6 7 8
Pass = 2:
Pass = 3:
Pass = 4:
Pass = 5:
Pass = 6:
Pass = 7:
steps: 28
final: 1 2 3 4 5 6 7 8
```

Όπως βλέπουμε πιο πάνω, η συνάρτηση η οποία κάνει χρήση της μεταβλητής λογικού τύπου, ταξινομεί τον πίνακα με μία μόνο διάσχιση και με τη δεύτερη διάσχιση το επιβεβαιώνει. Θα χρειαστεί συνολικά 13 «βήματα» (7 στην πρώτη και 6 στη δεύτερη διάσχιση). Η συνάρτηση, η οποία κάνει χρήση δύο βρόχων for, ταξινομεί τον πίνακα με την πρώτη διάσχιση, όμως συνεχίζει να κάνει διασχίσεις. Θα χρειαστεί συνολικά 28 «βήματα» (7 στην πρώτη, 6 στη δεύτερη, 5 στην τρίτη, 4 στην τέταρτη, 3 στην πέμπτη, 2 στην έκτη και 1 στην έβδομη διάσχιση).

6. Αλγόριθμος ταξινόμησης με εισαγωγή (Insertion sort)

Η ταξινόμηση με εισαγωγή (Insertion sort) λειτουργεί ως εξής: Ξεκινώντας από το πρώτο στοιχείο στα αριστερά, παίρνουμε κάθε στοιχείο και το τοποθετούμε στη σωστή θέση, στον ταξινομημένο πίνακα, αριστερά του στοιχείου στο οποίο βρισκόμαστε. Αρχικά, θεωρούμε ότι το πρώτο στοιχείο βρίσκεται στη σωστή θέση, αφού δεν υπάρχουν άλλα στοιχεία στα αριστερά του για να το συγκρίνουμε. Μπορούμε να φανταστούμε ότι ο αλγόριθμος μοιάζει με την προσπάθειά μας να ταξινομήσουμε ένα «φύλλο» που μας έχουν μοιράσει σε νέα παρτίδα.



Εικόνα 2: Ταξινόμηση με εισαγωγή

Πιο κάτω βλέπουμε τη συνάρτηση του αλγόριθμου ταξινόμηση εισαγωγής, η οποία είναι τύπου void και ταξινομεί τα στοιχεία του πίνακα σε αύξουσα σειρά.

```
void insertionSort(int arr[], int N){

 int temp;                        // Βοηθητική μεταβλητή
 int j;                           // Θέση στοιχείου
 for (int i=1; i<N; i++){         // i=1 γιατί ξεκινούμε από το
                                  // δεύτερο στοιχείο
   temp = arr[i];                 // Αποθηκεύουμε το στοιχείο
   j = i - 1;                     // Ψάχνουμε στα αριστερά

   while(j>=0 && arr[j]>temp){ // Όσο βρίσκουμε μεγαλύτερα στοιχεία
           arr[j+1] = arr[j];  // αντιγράφουμε στη θέση δεξιά
           j--;                // και μετακινούμαστε αριστερά
       }                       // Έξοδος από το while loop
                               // σημαίνει ότι φτάσαμε στην αρχή ή
                               // συναντήσαμε μικρότερο στοιχείο
   arr[j+1] = temp;            // Αντιγράφουμε το στοιχείο
  }                            // στη σωστή θέση, δεξιά του
}                              // μικρότερου στοιχείου
```

Παράδειγμα

Ας υποθέσουμε ότι έχουμε τον πίνακα **arr = {5, 6, 2, 4, 1, 9, 3, 0, 8, 7}** μεγέθους **10** και ότι θέλουμε να ταξινομηθεί σε **αύξουσα σειρά**.

5	6	2	4	1	9	3	0	8	7
0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*

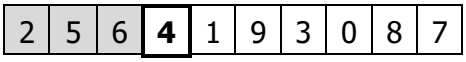
Θεωρούμε ότι το πρώτο στοιχείο (5) είναι ήδη ταξινομημένο, καθώς δεν έχουμε άλλα στοιχεία στα αριστερά του για να το συγκρίνουμε.

5	**6**	2	4	1	9	3	0	8	7
0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*

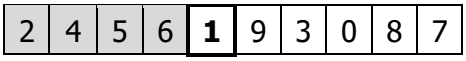
Συγκρίνουμε το δεύτερο στοιχείο (6) με το στοιχείο που βρίσκεται στα αριστερά του (5). Αφού το 6 είναι μεγαλύτερο του 5, δεν το μετακινούμε.

5	6	**2**	4	1	9	3	0	8	7
0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*

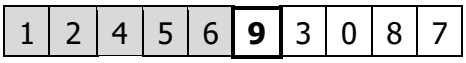
Συγκρίνουμε το τρίτο στοιχείο (2) με τα στοιχεία που βρίσκονται στα αριστερά του μέχρι να βρεθεί μικρότερο στοιχείο (5,6) ή να φτάσουμε στην αρχή του πίνακα. Αφού το 2 είναι μικρότερο του 5 και του 6, το μετακινούμε στη σωστή θέση, στην αρχή δηλαδή του πίνακα, πριν από το 5. Τα στοιχεία 5 και 6 μετακινούνται μία θέση δεξιά.



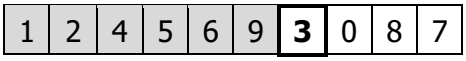
Συγκρίνουμε το τέταρτο στοιχείο (4) με τα στοιχεία που βρίσκονται στα αριστερά του μέχρι να βρεθεί μικρότερο στοιχείο (2,5,6). Αφού το 4 είναι μικρότερο του 5 και του 6, αλλά όχι του 2, το μετακινούμε στη σωστή θέση, μετά το 2 και πριν από το 5. Τα στοιχεία 5 και 6 μετακινούνται μία θέση δεξιά.



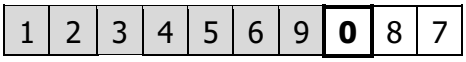
Συγκρίνουμε το πέμπτο στοιχείο (1) με τα στοιχεία που βρίσκονται στα αριστερά του μέχρι να βρεθεί μικρότερο στοιχείο (2,4,5,6). Αφού το 1 είναι μικρότερο από όλα τα στοιχεία, το μετακινούμε στη σωστή θέση, πριν από το 2. Τα στοιχεία 2, 4, 5 και 6 μετακινούνται μία θέση δεξιά.



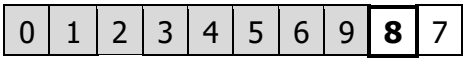
Συγκρίνουμε το έκτο στοιχείο (9) με το στοιχείο που βρίσκεται στα αριστερά του (6). Αφού το 9 είναι μεγαλύτερο, δεν το μετακινούμε.



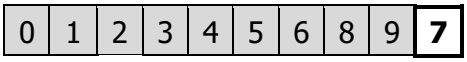
Συγκρίνουμε το έβδομο στοιχείο (3) με τα στοιχεία που βρίσκονται στα αριστερά του μέχρι να βρεθεί μικρότερο στοιχείο (2,4,5,6,9). Μετακινούμε το στοιχείο 3 στη σωστή θέση, μετά το 2 και πριν από το 4. Τα στοιχεία 4, 5, 6, 9 μετακινούνται μία θέση δεξιά.



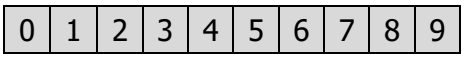
Συγκρίνουμε το όγδοο στοιχείο (0) με τα στοιχεία που βρίσκονται στα αριστερά του μέχρι να βρεθεί μικρότερο στοιχείο (1,2,3,4,5,6,9), ή να φτάσουμε στην αρχή του πίνακα. Το στοιχείο 0 τοποθετείται στην αρχή. Τα στοιχεία 1,2,3,4,5,6,9 μετακινούνται μία θέση δεξιά.



Συγκρίνουμε το ένατο στοιχείο (8) με τα στοιχεία που βρίσκονται στα αριστερά του μέχρι να βρεθεί μικρότερο στοιχείο (6,9). Το στοιχείο 8 τοποθετείται μετά το 6 και πριν το 9. Το στοιχείο 9 μετακινείται μία θέση δεξιά.



Συγκρίνουμε το δέκατο στοιχείο (7) με τα στοιχεία που βρίσκονται στα αριστερά του μέχρι να βρεθεί μικρότερο στοιχείο (6,8,9). Το στοιχείο 7 τοποθετείται μετά το 6 και πριν το 8. Τα στοιχεία 8 και 9 μετακινούνται μία θέση δεξιά.



Ο πίνακας έχει ταξινομηθεί.

7. Μετακίνηση στοιχείων

Ας δούμε πιο αναλυτικά πώς γίνεται η μετακίνηση των στοιχείων μέσα στον πίνακα. Στο πιο πάνω παράδειγμα είδαμε ότι για να ταξινομήσουμε το στοιχείο 4, πρέπει να το συγκρίνουμε με τα στοιχεία που βρίσκονται στα αριστερά του μέχρι να βρεθεί μικρότερο στοιχείο ή να φτάσουμε στην αρχή του πίνακα:

2	5	6	**4**	1	9	3	0	8	7
0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*

Αποθηκεύουμε την τιμή του 4 σε μία βοηθητική μεταβλητή (temp) και ξεκινούμε τις συγκρίσεις από το πρώτο στοιχείο στα αριστερά του (6).

Αφού 6>4, αντιγράφουμε την τιμή του 6 στη θέση που βρίσκεται στα δεξιά του. Συγκρίνουμε το 4 με το επόμενο στοιχείο (5).

2	5	6	**6**	1	9	3	0	8	7
0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*

Αφού 5>4, αντιγράφουμε την τιμή του 5 στη θέση που βρίσκεται στα δεξιά του. Συγκρίνουμε το 4 με το επόμενο στοιχείο (2).

2	5	**5**	6	1	9	3	0	8	7
0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*

Αφού 2<4, αντιγράφουμε την τιμή της temp (4) στη θέση που βρίσκεται στα δεξιά του 2. Το στοιχείο 4 έχει τοποθετηθεί στην σωστή θέση.

2	**4**	5	6	1	9	3	0	8	7
0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*

Παράδειγμα 6.3

Να δημιουργήσετε πρόγραμμα, το οποίο να καλεί μία συνάρτηση ταξινόμησης με εισαγωγή, η οποία να ταξινομεί σε φθίνουσα σειρά έναν πίνακα ακεραίων. Το πρόγραμμα, αρχικά, θα διαβάζει το πλήθος N (1<=N<=1000) των ακεραίων και, ακολούθως, θα διαβάζει τους ακέραιους και θα τους αποθηκεύει σε κατάλληλο πίνακα. Στη συνέχεια, θα καλεί τη συνάρτηση η οποία θα ταξινομεί τον πίνακα και θα εμφανίζει την τελική τους διάταξη, όπως στο παράδειγμα εξόδου πιο κάτω.

Παράδειγμα εισόδου

```
10
5 6 2 4 1 9 3 0 8 7
```

Παράδειγμα εξόδου

```
9 8 7 6 5 4 3 2 1 0
```

```
#include <iostream>
using namespace std;
#define MAXN 1000
void insertionSort(int arr[], int N){
 int temp;
 int j;
```

```
 for (int i=1; i<N; i++){
      temp = arr[i];
      j = i - 1;
      while (j>=0 && arr[j]<temp){     // Φθίνουσα σειρά
            arr[j+1] = arr[j];
            j--;
        }
      arr[j+1] = temp;
   }
}

int main(){
   int numbers[MAXN], N;           // Ο πίνακας δηλώνεται με το
   cin >> N;                       // μέγιστο δυνατό πλήθος ακεραίων
   for (int i=0; i<N; i++)         // Εισαγωγή στοιχείων
      cin >> numbers[i];
 insertionSort(numbers, N);        // Κλήση συνάρτησης
 for (int i=0; i<N; i++)
      cout << numbers[i] << " ";   // Εμφάνιση στοιχείων
return 0;
}
```

(Code: C7_6_example3.cpp)

8. Ανάλυση χρονικής πολυπλοκότητας αλγόριθμων ταξινόμησης

Ο αλγόριθμος ταξινόμησης φυσαλίδας έχει χρονική πολυπλοκότητα $O(N^2)$ σε όλες τις περιπτώσεις (καλύτερη και χειρότερη). Ειδικότερα, όταν γίνεται χρήση της συνάρτησης με δύο βρόχους for, το εξωτερικό for εκτελείται N-1 φορές και για κάθε i-οστή επανάληψη του εξωτερικού βρόχου, ο εσωτερικός θα εκτελείται N-i-1 φορές. Σε κάθε εκτέλεση του εσωτερικού βρόχου έχουμε μία σύγκριση στοιχείων, οπόταν το συνολικό πλήθος των συγκρίσεων είναι περίπου $N^2/2$.

Ο αλγόριθμος ταξινόμησης με εισαγωγή έχει, επίσης, χρονική πολυπλοκότητα $O(N^2)$, αλλά είναι πιο γρήγορος από τον αλγόριθμο ταξινόμησης φυσαλίδας. Ο βρόχος for εκτελείται N-1 φορές και ο βρόχος while εξαρτάται από τη θέση που πρέπει να πάρει το στοιχείο στη θέση j, στον ταξινομημένο υποπίνακα A[0…j-1]. Στην καλύτερη περίπτωση, το στοιχείο στη θέση j θα παραμείνει στη θέση του. Όταν ο πίνακας είναι ήδη ταξινομημένος, αυτό θα συμβαίνει σε κάθε εκτέλεση του βρόχου for. Ο βρόχος while δεν θα εκτελεστεί καμία φορά και ο βρόχος for θα κάνει μία μόνο σύγκριση μεταξύ στοιχείων. Επομένως, στην καλύτερη περίπτωση, που ο πίνακας είναι ήδη ταξινομημένος, το πλήθος των συγκρίσεων που πραγματοποιεί ο αλγόριθμος μεταξύ στοιχείων είναι N-1.

Στη χειρότερη περίπτωση, το στοιχείο A[j] είναι μικρότερο από όλα τα στοιχεία του ταξινομημένου υποπίνακα A[0…j-1] και πρέπει να τοποθετηθεί στην πρώτη θέση. Αυτό συμβαίνει όταν ο πίνακας είναι ταξινομημένος σε φθίνουσα σειρά και ο αλγόριθμος πρέπει να τον ταξινομήσει σε αύξουσα σειρά. Ο βρόχος while εκτελείται συνολικά j-1 φορές και,

επομένως, στη χειρότερη περίπτωση, ο αριθμός των συγκρίσεων του αλγόριθμου είναι περίπου $N^2/2$.

8.1 Χρονική σύγκριση αλγόριθμων ταξινόμησης

Για σκοπούς σύγκρισης των αλγόριθμων παραθέτουμε τον πιο κάτω πίνακα ο οποίος εμφανίζει τα αποτελέσματα εκτέλεσης του πιο κάτω κώδικα, σε επεξεργαστή **Intel® Core™ i5-6200U,** στα **2.40 GHz** και **8 GB RAM**. Χρησιμοποιώντας το πιο κάτω τμήμα κώδικα, έχουμε τη δυνατότητα να υπολογίσουμε τον χρόνο εκτέλεσης των αλγόριθμων για σκοπούς σύγκρισης:

```
// Για να χρησιμοποιήσουμε τις χρονικές συναρτήσεις, πρέπει να
// συμπεριλάβουμε τη βιβλιοθήκη <ctime>
time_t t1 = clock();              // Ξεκινούμε το χρονόμετρο
insertionSort(numbers, N);        // Καλούμε τη συνάρτηση
time_t t2 = clock() - t1;         // Σταματούμε το χρονόμετρο
double total_time = ((double)t2)/CLOCKS_PER_SEC;
cout << total_time;
```

(Code: C7_6_example4.cpp)

Πλήθος Στοιχείων (N)	Bubble Sort (με δύο for loops)	Bubble Sort (με boolean)	Insertion Sort
1,000	0.000	0.000	0.000
5,000	0.078	0.059	0.032
20,000	1.423	1.302	0.312
50,000	4.419	4.387	1.417
100,000	18.015	17.794	7.632
1,000,000	386.193	377.322	82.841

Πίνακας 6.1: Χρόνοι αλγόριθμων ταξινόμησης (σε δευτερόλεπτα)

Από τον πιο πάνω πίνακα φαίνεται ότι οι χρόνοι των δύο συναρτήσεων της ταξινόμησης φυσαλίδας δεν έχουν μεγάλες διαφορές. Η συνάρτηση με τη μεταβλητή λογικού τύπου εμφανίζεται να έχει ελαφρώς καλύτερους χρόνους ταξινόμησης από τη συνάρτηση με τους δύο βρόχους for και η διαφορά φαίνεται να αυξάνεται, σε μικρό βαθμό, για μεγαλύτερα πλήθη στοιχείων. Ξεκάθαρα φαίνεται, επίσης, η διαφορά στους χρόνους της συνάρτησης ταξινόμησης με εισαγωγή, η οποία παρά το γεγονός ότι έχει την ίδια χρονική πολυπλοκότητα με την ταξινόμηση φυσαλίδας, εντούτοις οι ενδεικτικοί χρόνοι φανερώνουν πως ο αλγόριθμος ταξινόμησης με εισαγωγή είναι πολύ πιο αποδοτικός.

Σημείωση: Οι πιο πάνω χρόνοι είναι ενδεικτικοί και ενδέχεται να διαφέρουν από εκτέλεση σε εκτέλεση, καθώς και από υπολογιστή σε υπολογιστή.

Ασκήσεις Κεφαλαίου - www.hackerrank.com/g76

Άσκηση 6.1

Να συμπληρώσετε τις εντολές για την πιο κάτω συνάρτηση ταξινόμησης φυσαλίδας, η οποία ταξινομεί τον πίνακα ακεραίων που θα δοθεί ως παράμετρος, σε φθίνουσα σειρά.

```
void bubblesort(int arr[], int N){
 bool sorted;
 int temp;
 do{









   } while
}
```

Άσκηση 6.2

Να συμπληρώσετε τις εντολές για την πιο κάτω συνάρτηση ταξινόμησης με εισαγωγή, η οποία ταξινομεί τον πίνακα χαρακτήρων που θα δοθεί ως παράμετρος, σε φθίνουσα σειρά.

```
void insertionSort(char arr[], int N){
 char temp;
 int j;
 for (int i=1; i<N; i++){









   }
}
```

Άσκηση 6.3

Να παρουσιάσετε τα αποτελέσματα της ταξινόμησης σε αύξουσα σειρά του πιο κάτω πίνακα, με τη χρήση του αλγόριθμου ταξινόμησης με εισαγωγή. Για κάθε διάσχιση να παρουσιάζετε ολόκληρο τον πίνακα.

25	16	14	13	20	15	26	17
0	*1*	*2*	*3*	*4*	*5*	*6*	*7*

Άσκηση 6.4

Να παρουσιάσετε τα αποτελέσματα της ταξινόμησης σε αύξουσα σειρά του πιο κάτω πίνακα, με τη χρήση του αλγόριθμου ταξινόμησης φυσαλίδας. Η συνάρτηση να συγκρίνει κάθε φορά διαδοχικά στοιχεία και να κάνει χρήση μίας λογικής μεταβλητής. Για κάθε αντιμετάθεση στοιχείου να παρουσιάζετε ολόκληρο τον πίνακα.

16	6	4	14	5
0	*1*	*2*	*3*	*4*

Άσκηση 6.5

Να παρουσιάσετε τα αποτελέσματα της ταξινόμησης σε φθίνουσα σειρά του πιο κάτω πίνακα, με τη χρήση του αλγόριθμου ταξινόμησης φυσαλίδας. Η συνάρτηση να κάνει χρήση δύο βρόχων for. Για κάθε αντιμετάθεση στοιχείου να παρουσιάζετε ολόκληρο τον πίνακα.

15	61	42	24	53
0	*1*	*2*	*3*	*4*

Άσκηση 6.6

Να δημιουργήσετε πρόγραμμα, το οποίο να διαβάζει 10 ακέραιους αριθμούς, να τους τοποθετεί σε πίνακα και να εμφανίζει τους 3 μεγαλύτερους σε αύξουσα σειρά.

Παράδειγμα εισόδου

```
24 66 12 45 61 77 89 108 15 44
```

Παράδειγμα εξόδου

```
77 89 108
```

Άσκηση 6.7

Να δημιουργήσετε πρόγραμμα, το οποίο να διαβάζει 20 ακέραιους αριθμούς, να τους τοποθετεί σε πίνακα και να εμφανίζει τους 5 μεγαλύτερους και τους 5 μικρότερους, όπως στο πιο κάτω παράδειγμα.

Παράδειγμα εισόδου

```
2 4 7 87 65 6 45 89 7 5 54 32 11 12 23 43 46 59 87 54
```

Παράδειγμα εξόδου

```
89  87  87  65  59
2   4   5   6   7
```

Άσκηση 6.8

Έστω ότι ο Α είναι μονοδιάστατος πίνακας ακεραίων με 10 στοιχεία. Να δημιουργήσετε πρόγραμμα, το οποίο να δημιουργεί και να παρουσιάζει έναν δεύτερο πίνακα Β, ο οποίος να περιέχει τα στοιχεία του πίνακα Α, έχοντας όμως τα μηδενικά στοιχεία μαζεμένα στο τέλος του. Δηλαδή, μόνο τα μηδενικά στοιχεία θα ταξινομούνται και όχι ο υπόλοιπος πίνακας.

Για παράδειγμα, αν ο πίνακας Α είναι της μορφής:

```
A = [1, 0, 3, 7, 0, 0, 6, 4, 0, 9]
```

τότε ο πίνακας Β θα πρέπει να εμφανίζεται ως:

```
B = [1, 3, 7, 6, 4, 9, 0, 0, 0, 0]
```

Άσκηση 6.9

Να δημιουργήσετε πρόγραμμα, το οποίο να διαβάζει έναν ακέραιο αριθμό Ν (1<=N<=15) ακολουθούμενο από Ν χαρακτήρες. Το πρόγραμμα να ταξινομεί σε αλφαβητική σειρά τους χαρακτήρες, κάνοντας χρήση μίας συνάρτησης ταξινόμησης με εισαγωγή και να τους εμφανίζει στην οθόνη.

Παράδειγμα εισόδου

```
4
G P T E
```

Παράδειγμα εξόδου

```
E G P T
```

Άσκηση 6.10

Να δημιουργήσετε πρόγραμμα, το οποίο να διαβάζει έναν ακέραιο αριθμό Ν (1<=N<=15) ακολουθούμενο από Ν πραγματικούς αριθμούς. Αν ο μέσος όρος των αριθμών είναι μικρότερος του δέκα, τότε να τυπώνει τους αριθμούς σε αύξουσα σειρά, αλλιώς να τους τυπώνει σε φθίνουσα σειρά.

Παράδειγμα εισόδου

```
5
1.2 1.3 0.9 1.1 1.4
```

Παράδειγμα εξόδου

```
0.9 1.1 1.2 1.3 1.4
```

Άσκηση 6.11

Αν σας δοθούν ένας ακέραιος αριθμός Ν (1<=N<=20), ακολουθούμενος από Ν ακεραίους, Να δημιουργήσετε πρόγραμμα, το οποίο θα καλεί μία συνάρτηση ταξινόμησης με εισαγωγή, η οποία να εμφανίζει μετά από κάθε διάσχιση, τον πίνακα ακεραίων μέχρι την τελική του ταξινόμηση σε αύξουσα σειρά.

Παράδειγμα εισόδου

```
6
1 4 3 5 6 2
```

Παράδειγμα εξόδου

```
1 4 3 5 6 2
1 3 4 5 6 2
1 3 4 5 6 2
1 3 4 5 6 2
1 2 3 4 5 6
```

Άσκηση 6.12

Ο Παντελής έχει στη διάθεσή του W (1<=W<=1000) ευρώ. Θέλει να αγοράσει όσο το δυνατόν περισσότερα δώρα για τις γιορτές. Αν έχει να επιλέξει μεταξύ N (1<=N<=100) δώρων, να βρείτε ποιος είναι ο μέγιστος αριθμός δώρων που μπορεί να αγοράσει. Να δημιουργήσετε πρόγραμμα, το οποίο να διαβάζει, στην πρώτη γραμμή, δύο ακέραιους αριθμούς, το W και το N και, στη συνέχεια, N ακέραιους αριθμούς που αντιστοιχούν στην τιμή του κάθε δώρου. Το πρόγραμμα να επιστρέφει το μέγιστο πλήθος από δώρα που μπορεί να αγοράσει ο Παντελής με τα χρήματα που έχει στην κατοχή του.

Παράδειγμα εισόδου

```
100 4
56 23 14 38
```

Παράδειγμα εξόδου

```
3
```

Επεξήγηση: Ο Παντελής μπορεί να αγοράσει τα δώρα αξίας 14, 23 και 38 ευρώ.

Άσκηση 6.13

Να δημιουργήσετε πρόγραμμα, το οποίο να διαβάζει αρχικά έναν ακέραιο N (1<=N<=50) και, στη συνέχεια, N ακεραίους, σε τυχαία σειρά. Ακολούθως, το πρόγραμμα να:

(α) αποθηκεύει τους ακεραίους σε κατάλληλο πίνακα

(β) καλεί μία συνάρτηση ταξινόμησης η οποία να ταξινομεί τον πίνακα σε αύξουσα σειρά

(γ) καλεί μία συνάρτηση δυαδικής αναζήτησης η οποία να δέχεται παραμετρικά ένα στοιχείο προς αναζήτηση, να αναζητά το στοιχείο στον πίνακα και να επιστρέφει τη θέση του στοιχείου στον ταξινομημένο πίνακα, ή τον αριθμό -1 αν το στοιχείο δεν βρεθεί

(δ) εμφανίζει τα μηνύματα «Number found at index …», ή «Number not found», αντίστοιχα.

Παράδειγμα εισόδου

```
5
65 32 41 83 78
41
```

Παράδειγμα εξόδου

```
Number found at index 1
```

Επεξήγηση: Ο αριθμός 41, αν ο πίνακας ταξινομηθεί σε αύξουσα σειρά, θα τοποθετηθεί στη θέση 1.

Άσκηση 6.14

Να δημιουργήσετε πρόγραμμα, το οποίο να διαβάζει δέκα ονόματα ατόμων και τις ηλικίες τους, να τους ταξινομεί σε αύξουσα σειρά από τον μικρότερο σε ηλικία μέχρι τον μεγαλύτερο και να τους εμφανίζει στην οθόνη με δεξιά στοίχιση 20 χαρακτήρων, όπως βλέπετε πιο κάτω:

```
             Douglas                  19
            Andersen                  21
              Irving                  23
               James                  25
               Young                  26
               Damon                  28
                Love                  29
              Peters                  30
             Wiggins                  31
              Jordan                  43
```

Άσκηση 6.15

Να τροποποιήσετε το πρόγραμμα της άσκησης **6.14**, έτσι ώστε αν δύο άτομα έχουν την ίδια ηλικία, τότε να εμφανίζει πρώτα το όνομα αυτού που προηγείται αλφαβητικά, όπως στο παράδειγμα πιο κάτω:

```
              Adams                   23
               Bale                   23
```

Άσκηση 6.16

Να δημιουργήσετε πρόγραμμα, το οποίο να διαβάζει αρχικά έναν περιττό ακέραιο Ν (1<=N<=101) και, στη συνέχεια, Ν ακεραίους. Ανάμεσα τους υπάρχει ένας και μοναδικός ακέραιος, ο οποίος δεν επαναλαμβάνεται. Το πρόγραμμα να τον βρίσκει και να τον εμφανίζει στην οθόνη.

Παράδειγμα εισόδου

```
5
14 6 8 14 6
```

Παράδειγμα εξόδου

```
8
```

Άσκηση 6.17

Να δημιουργήσετε πρόγραμμα, το οποίο να χρησιμοποιεί δύο συναρτήσεις ταξινόμησης (φυσαλίδας και εισαγωγής), για να ταξινομήσει τους αριθμούς που βρίσκονται σε ένα αρχείο, σε αύξουσα σειρά. Το αρχείο θα περιέχει μέχρι 20,000 ακέραιους αριθμούς. Οι συναρτήσεις να καταμετρούν το πλήθος των «βημάτων» που θα χρειαστεί η κάθε μία για να ταξινομήσει τους αριθμούς. «Βήμα» θεωρούμε κάθε εκτέλεση του εσωτερικού βρόχου επανάληψης της κάθε συνάρτησης. Να γίνει χρήση των αρχείων 1000.txt, 5000.txt, 20000.txt, nearly.txt (σχεδόν ταξινομημένοι ακέραιοι) και reverse.txt (σε φθίνουσα σειρά).

Παράδειγμα εισόδου (αρχείο)

```
13 24 37 38 49 40
```

Παράδειγμα εξόδου (οθόνη)

```
Bubble Sort: 9 Insertion Sort: 1
```

Άσκηση 6.18

Ο Παντελής εργάζεται εδώ και χρόνια στο Γραφείο Εξυπηρέτησης του Πολίτη. Τα καθήκοντά του περιλαμβάνουν την επίδοση ενός αριθμού σε κάθε δημότη, από τους N (1<=N<=1000) συνολικά, που μπαίνει στο γραφείο, ώστε να τηρείται η σειρά προτεραιότητας. Για να διευκολύνει τους δημότες, ο Παντελής θέλει να τους τοποθετεί στα καθίσματα στον χώρο αναμονής, με βάση τον αριθμό που τους έχει δώσει. Για παράδειγμα, όποιος έχει τον αριθμό 1 θα καθίσει στο πρώτο κάθισμα αριστερά, ο αριθμός 2 στο αμέσως επόμενο κ.ο.κ. Ως συνήθως, οι δημότες κάθονται σε διαφορετική σειρά από αυτή με την οποία πρόκειται να εξυπηρετηθούν. Ο Παντελής θέλει να γνωρίζει πόσες αλλαγές θέσεων πρέπει να κάνει, ώστε να καθίσουν οι δημότες ταξινομημένοι με τη σειρά προτεραιότητας. Λόγω περιορισμένου χώρου οι αλλαγές θέσεων γίνονται μόνο μεταξύ δύο ατόμων που κάθονται δίπλα ο ένας στον άλλο.

Παράδειγμα εισόδου

```
4
3 4 1 2
```

Παράδειγμα εξόδου

```
4
```

Επεξήγηση: Οι 4 αλλαγές είναι οι εξής: (3 **1 4** 2), (3 1 **2 4**), (**1 3** 2 4), (1 **2 3** 4).

Άσκηση 6.19

Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται έναν ακέραιο αριθμό N (1<=N<=100), στη συνέχεια N ακέραιους αριθμούς και να τους ταξινομεί σε αύξουσα σειρά, με βάση το ψηφίο των μονάδων τους. Αν το ψηφίο των μονάδων των δύο αριθμών είναι το ίδιο, τότε να εμφανίζεται πρώτος ο μικρότερος από τους δύο αριθμούς.

Παράδειγμα εισόδου

```
4
17 8 119 200
```

Παράδειγμα εξόδου

```
200 17 8 119
```

Άσκηση 6.20

Να δημιουργήσετε πρόγραμμα, το οποίο να χρησιμοποιεί μία συνάρτηση ταξινόμησης με εισαγωγή και να ταξινομεί έναν πίνακα ακεραίων σε αύξουσα σειρά, με βάση το υπόλοιπο της διαίρεσης των στοιχείων του πίνακα με έναν ακέραιο K. Για παράδειγμα, αν στον πίνακα υπάρχουν τα στοιχεία a και b, αν a%K < b%K τότε το στοιχείο a θα τοποθετηθεί πριν από το στοιχείο b. Το πρόγραμμα να δέχεται αρχικά έναν ακέραιο N (1<=N<=1000) και έναν ακέραιο K (1<=K<=10) και, ακολούθως, N ακέραιους. Το πρόγραμμα να εμφανίζει τους αριθμούς ταξινομημένους με τον τρόπο που αναφέρεται πιο πάνω.

Παράδειγμα εισόδου

```
5 6
12 18 17 65 46
```

Παράδειγμα εξόδου

```
12 18 46 17 65
```

Επεξήγηση: Τα υπόλοιπα της διαίρεσης των στοιχείων του πίνακα με τον αριθμό K=6, είναι 12%6=0, 18%6=0 17%6=5, 65%6=5, 46%6=4, άρα θα τοποθετηθούν όπως πιο πάνω.

Άσκηση 6.21

Ένας μαθητής θέλει να βρει τη μέγιστη διαφορά ύψους που υπάρχει μεταξύ δύο μαθητών στο τμήμα του. Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται, αρχικά, έναν ακέραιο αριθμό N (1<=N<=50), στη συνέχεια N πραγματικούς αριθμούς (τα ύψη των μαθητών) και να εμφανίζει τη μέγιστη διαφορά ύψους μεταξύ δύο μαθητών με δύο δεκαδικά ψηφία.

Παράδειγμα εισόδου

```
4
1.82 1.58 1.76 1.67
```

Παράδειγμα εξόδου

```
0.24
```

Άσκηση 6.22

Ο Παντελής θέλει να ταξινομήσει κάποια κιβώτια με βάση το βάρος τους. Τα κιβώτια είναι χωρισμένα σε T (1<=T<=10) σειρές και κάθε σειρά έχει N (1<=N<=100) κιβώτια. Ο Παντελής θέλει να ταξινομήσει τα κιβώτια κάθε σειράς από το βαρύτερο μέχρι το ελαφρύτερο. Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται, αρχικά, έναν ακέραιο T (πλήθος σειρών) και για κάθε σειρά να δέχεται έναν ακέραιο N (πλήθος κιβωτίων της σειράς). Ακολούθως, το πρόγραμμα να δέχεται N ακέραιους που αντιστοιχούν στο βάρος κάθε κιβωτίου της σειράς. Το πρόγραμμα να επιστρέφει τις σειρές ταξινομημένες σε φθίνουσα σειρά με βάση το βάρος τους.

Παράδειγμα εισόδου

```
3
4
56 23 43 38
2
17 22
3
56 57 58
```

Παράδειγμα εξόδου

```
56 43 38 23
22 17
58 57 56
```

Άσκηση 6.23

Έχουμε δύο πίνακες ακεραίων μεγέθους N και M (1<=N,M<=1000) στοιχείων, αντίστοιχα, τους οποίους θέλουμε να συγχωνεύσουμε και να ταξινομήσουμε σε φθίνουσα σειρά. Να δημιουργήσετε πρόγραμμα, το οποίο να διαβάζει έναν ακέραιο N και, στη συνέχεια, N ακέραιους (τα στοιχεία του πρώτου πίνακα). Ακολούθως, να διαβάζει έναν ακέραιο M και, στη συνέχεια, M ακέραιους (τα στοιχεία του δεύτερου πίνακα). Το πρόγραμμα θα εμφανίζει έναν πίνακα με N+M στοιχεία, ταξινομημένα σε φθίνουσα σειρά.

Παράδειγμα εισόδου

```
4
16 44 33 27
5
51 68 12 4 19
```

Παράδειγμα εξόδου

```
68 51 44 33 27 19 16 12 4
```

Άσκηση 6.24

Η Αριάδνη ετοιμάζεται να κτίσει το καινούριο της παλάτι στην Κνωσσό. Το παλάτι θα είναι επενδυμένο με πέτρα. Λόγω της ιδιαίτερης αρχιτεκτονικής του παλατιού, θα πρέπει κάποια κομμάτια πέτρας να μεταφερθούν άμεσα. Αυτά τα κομμάτια έχουν βάρος 1. Να δημιουργήσετε πρόγραμμα, το οποίο θα εμφανίζει τη σωστή σειρά μεταφοράς των πετρών. Τα βάρη με την ενδεικτική τιμή 1 αντιστοιχούν σε αυτά τα ξεχωριστά κομμάτια πέτρας, τα οποία πρέπει να μεταφερθούν με τη δεδομένη σειρά εμφάνισης. Τα υπόλοιπα κομμάτια θα μεταφερθούν με αύξουσα σειρά με βάση το βάρος τους. Το πρόγραμμα να διαβάζει το πλήθος των πετρών N (1<=N<=1000) και, ακολούθως, N ακέραιους, οι οποίοι αντιστοιχούν στο βάρος κάθε πέτρας. Οι πέτρες με βάρος 1 πρέπει να μεταφερθούν με βάση τη σειρά εμφάνισής τους.

Παράδειγμα εισόδου

```
10
1740 532 9000 1 8500 4000 5000 2120 1 8999
```

Παράδειγμα εξόδου

```
532 1740 2120 1 4000 5000 8500 8999 1 9000
```

Άσκηση 6.25

Να δημιουργήσετε πρόγραμμα, το οποίο δέχεται N (1<=N<=1000) κλάσματα με τη μορφή δύο θετικών ακεραίων και να εμφανίζει τα τρία μεγαλύτερα κλάσματα στην ίδια μορφή με την μορφή εισόδου. Το πρόγραμμα θα διαβάζει αρχικά έναν ακέραιο N. Στις επόμενες N γραμμές θα εμφανίζονται ζεύγη ακεραίων, οι αριθμητές και οι παρονομαστές των κλασμάτων. Το πρόγραμμα θα εμφανίζει τα 3 μεγαλύτερα κλάσματα σε φθίνουσα σειρά, ένα σε κάθε γραμμή. Σε περίπτωση που δύο κλάσματα είναι ίσα, δίνεται προτεραιότητα σε εκείνο με τον μικρότερο αριθμητή.

Παράδειγμα εισόδου

```
5
1 2
```

```
1 4
5 17
2 4
2 9
```

Παράδειγμα εξόδου

```
1 2
2 4
5 17
```

Ασκήσεις Εμπλουτισμού

Άσκηση 6.26

Η τάξη του Παντελή ψηφίζει για τις καθιερωμένες εκλογές μαθητικού συμβουλίου. Ο Παντελής θέλει να ξέρει το ποσοστό που έλαβαν οι μαθητές και οι μαθήτριες της τάξης του. Έχει καταγράψει όλες τις ψήφους στον πίνακα και θέλει να εμφανίζονται τα ονόματα των υποψηφίων με αλφαβητική σειρά και δίπλα το ποσοστό ψήφων που έλαβε το κάθε όνομα. Να δημιουργήσετε πρόγραμμα, το οποίο θα βοηθήσει τον Παντελή. Το πρόγραμμα θα διαβάζει αρχικά έναν ακέραιο Ν (1<=N<=1000), ακολουθούμενο από Ν ψήφους. Το πρόγραμμα να εμφανίζει τα τελικά αποτελέσματα σε αλφαβητική σειρά με βάση τα ονόματα και δίπλα το ποσοστό που εξασφάλισε ο κάθε υποψήφιος.

Παράδειγμα εισόδου

```
18
Andreas
Nikos
Marios
Andreas
Kostas
Maria
Maria
Andri
Maria
Andreas
Giorgos
Pantelis
Marios
Andreas
Kyriakos
Pavlos
Maria
Andri
```

Παράδειγμα εξόδου

```
Andreas 22%
Andri 11%
```

```
Giorgos 6%
Kostas 6%
Kyriakos 6%
Maria 22%
Marios 11%
Nikos 6%
Pantelis 6%
Pavlos 6%
```

Άσκηση 6.27

Ο Παντελής θα πάει φέτος στον χορό των τελειοφοίτων του σχολείου του. Η παράδοση του χορού λέει πως τα αγόρια πρέπει να χορέψουν με ένα κοντύτερο κορίτσι και μπορούν να χορέψουν μόνο με ένα κορίτσι κατά τη διάρκεια της βραδιάς. Να βρείτε πόσα αγόρια θα καταφέρουν να χορέψουν με κορίτσι φέτος. Το πρόγραμμα θα διαβάζει στην πρώτη γραμμή δύο ακέραιους B και G (1<=B,G<=100), το πλήθος των αγοριών και το πλήθος των κοριτσιών, αντίστοιχα. Στη δεύτερη γραμμή, θα εμφανίζονται B ακέραιοι αριθμοί, τα ύψη των αγοριών και στην τρίτη γραμμή G ακέραιοι αριθμοί, τα ύψη των κοριτσιών. Το πρόγραμμα να εμφανίζει έναν ακέραιο, το πλήθος των αγοριών που θα βρουν ντάμα για τον χορό φέτος.

Παράδειγμα εισόδου

```
3 3
3 8 5
2 4 9
```

Παράδειγμα εξόδου

```
2
```

Επεξήγηση: Το αγόρι με ύψος 3 μπορεί να χορέψει με το κορίτσι με ύψος 2 και το αγόρι με ύψος 5 μπορεί να χορέψει με το κορίτσι με ύψος 4. Το αγόρι με ύψος 8 δεν μπορεί να χορέψει με το κορίτσι με ύψος 9, άρα έχουμε δύο αγόρια συνολικά.

Άσκηση 6.28

Να δημιουργήσετε πρόγραμμα, το οποίο να διαβάζει αρχικά έναν ακέραιο N (1<=N<=1000), ακολουθούμενο από N χαρακτήρες, τους οποίους να αποθηκεύει σε πίνακα. Στη συνέχεια, να διαβάζει έναν ακέραιο Q, ακολουθούμενο από Q χαρακτήρες από αυτούς που υπάρχουν ήδη στον πίνακα. Για κάθε έναν από τους Q χαρακτήρες να εμφανίζεται το πλήθος των χαρακτήρων του πίνακα, οι οποίοι προηγούνται αλφαβητικά του χαρακτήρα αυτού, όπως στο παράδειγμα πιο κάτω.

Παράδειγμα εισόδου

```
5
f c r m x
3
x
c
r
```

Παράδειγμα εξόδου

```
4
0
3
```

Επεξήγηση: Δίδονται αρχικά πέντε χαρακτήρες (f, c, r, m, x). Ακολουθούν τρεις χαρακτήρες. Για τον χαρακτήρα x υπάρχουν 4 χαρακτήρες στον πίνακα, οι οποίοι προηγούνται αλφαβητικά (f, c, r, m). Για τον χαρακτήρα c δεν υπάρχει χαρακτήρας που να προηγείται, ενώ για τον χαρακτήρα r υπάρχουν 3 χαρακτήρες του πίνακα που προηγούνται (f, c, m).

Άσκηση 6.29

Σε ένα δάσος υπάρχουν N (1<=N<=1000) δέντρα και M (1<=M<=1000) ξυλοκόποι. Ένα δέντρο έχει διάμετρο D και μπορεί να το κόψει ένας ξυλοκόπος με ύψος H μόνο αν D<=H. Ένας ξυλοκόπος μπορεί να κόψει μόνο ένα δέντρο. Αν σας δοθούν ένας πίνακας με τις διαμέτρους των δέντρων και ένας πίνακας με τα ύψη των ξυλοκόπων, να βρείτε αν είναι κατορθωτό να κοπούν όλα τα δέντρα και ποιο είναι το ελάχιστο συνολικό ύψος ξυλοκόπων που απαιτείται, για να γίνει αυτό. Αν δεν είναι δυνατόν να κοπούν όλα τα δέντρα, να εμφανίζεται ο αριθμός -1, αλλιώς να εμφανίζεται το ελάχιστο συνολικό ύψος που πρέπει να έχουν οι ξυλοκόποι, για να κοπούν όλα τα δέντρα.

Παράδειγμα εισόδου

```
5
34 13 46 99 78
6
17 21 64 40 99 100
```

Παράδειγμα εξόδου

```
320
```

Επεξήγηση: Στο δάσος υπάρχουν 5 δέντρα (34, 13, 46, 99, 78) και 6 ξυλοκόποι (17, 21, 64, 40, 99, 100). Ο ξυλοκόπος με ύψος 17 κόβει το δέντρο με διάμετρο 13. Ο ξυλοκόπος με ύψος 21 δεν μπορεί να κόψει δέντρο. Ο ξυλοκόπος με ύψος 40 κόβει το δέντρο με διάμετρο 34. Ο ξυλοκόπος με ύψος 64 κόβει το δέντρο με διάμετρο 46. Ο ξυλοκόπος με ύψος 99 κόβει το δέντρο με διάμετρο 78. Τέλος, ο ξυλοκόπος με ύψος 100 κόβει το δέντρο με διάμετρο 99. Συνολικό ύψος = 17+40+64+99+100 = 320.

Άσκηση 6.30

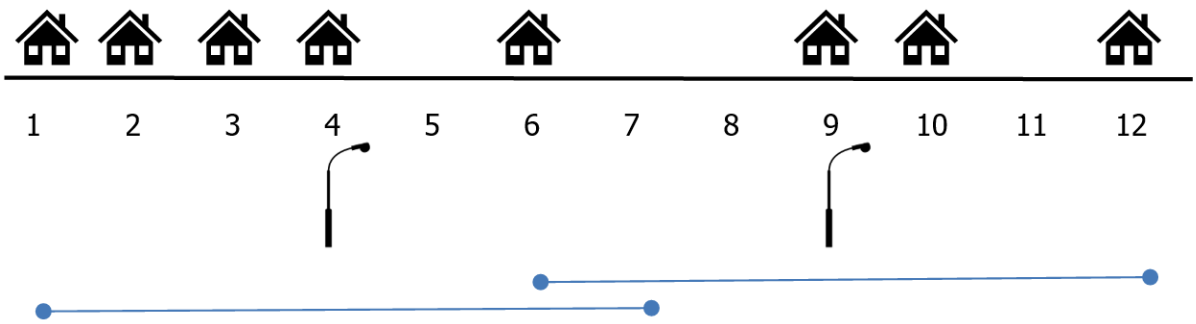
Ο δήμος Λευκωσίας έχει αποφασίσει να εγκαταστήσει καινούριες λάμπες φωτισμού, σε όλες τις κύριες οδούς της πόλης. Η εγκατάσταση των λαμπών θα γίνει με τέτοιο τρόπο ώστε:

- Να γίνει χρήση όσον το δυνατόν λιγότερων λαμπών φωτισμού.
- Κάθε σπίτι οποιασδήποτε οδού να φωτίζεται το βράδυ.
- Η εγκατάσταση μίας λάμπας πρέπει να γίνει σε θέση όπου υπάρχει σπίτι.

Για παράδειγμα, αν σε μία οδό έχουμε 8 σπίτια και η εμβέλεια φωτισμού κάθε λάμπας είναι 3, τότε για το πιο κάτω παράδειγμα θα χρειαστούμε μόνο 2 λάμπες στις θέσεις 4 και 9, όπως βλέπετε στην πιο κάτω εικόνα:

Παράδειγμα

```
8 3
1 3 4 6 2 10 9 12
```



Στο πιο πάνω παράδειγμα, οι λάμπες μπορούν να εγκατασταθούν και με άλλους τρόπους, οι οποίοι δίνουν ως αποτέλεσμα 2. Π.χ. (1, 9), (2, 9), (3, 9), (4, 10) και (4, 12).

Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται το πλήθος και τους αριθμούς των σπιτιών μίας οδού και να εμφανίζει τον ελάχιστο αριθμό από λάμπες που θα χρειαστούν, για να φωτιστούν όλα τα σπίτια.

Δεδομένα εισόδου

- Στην πρώτη γραμμή θα εμφανίζονται δύο ακέραιοι αριθμοί: το N (1<=N<=10,000) που υποδεικνύει το πλήθος των σπιτιών και E (1<=E<=10,000) που υποδεικνύει την εμβέλεια φωτισμού κάθε μίας από τις λάμπες.
- Στην επόμενη γραμμή θα έχουμε N ακέραιους X (1<=X<=10,000) που υποδεικνύουν τον αριθμό του κάθε σπιτιού.

Δεδομένα εξόδου

Ένας ακέραιος K, ο ελάχιστος αριθμός από λάμπες που θα χρειαστούν, για να φωτίζονται όλα τα σπίτια της οδού το βράδυ.

Παράδειγμα εισόδου

```
8 3
1 3 4 6 2 10 9 12
```

Παράδειγμα εισόδου

```
2
```

Γ7.7 Δισδιάστατοι Πίνακες (2D - Arrays)

Τι θα μάθουμε σε αυτό το κεφάλαιο:

- Να αναγνωρίζουμε προβλήματα που χρειάζονται για την επίλυσή τους μονοδιάστατους και δισδιάστατους πίνακες (1-D arrays και 2-D arrays)
- Να εντοπίζουμε ποιοι πίνακες (arrays) χρειάζονται με βάση τις ανάγκες του αλγορίθμου/προγράμματος
- Να δίνουμε κατάλληλο όνομα (αναγνωριστικό) σε έναν πίνακα και να καθορίζουμε τον αριθμό των σειρών (rows) και των στηλών (columns) που χρειάζεται να περιέχει
- Να επιλέγουμε κατάλληλο τύπο δεδομένων για έναν πίνακα
- Να δηλώνουμε πίνακες (arrays) με βάση τις ανάγκες του προγράμματος και να αρχικοποιούμε τους πίνακες όπου χρειάζεται
- Να αναφερόμαστε σε στοιχεία του πίνακα, χρησιμοποιώντας κατάλληλους δείκτες
- Να χρησιμοποιούμε μία βασική τεχνική για είσοδο/εισαγωγή (εκχώρηση τιμής) στα στοιχεία ενός δισδιάστατου πίνακα με χρήση δύο εμφωλιασμένων βρόγχων for
- Να χρησιμοποιούμε μία βασική τεχνική για έξοδο (εμφάνιση) των τιμών των στοιχείων ενός δισδιάστατου πίνακα με χρήση δύο εμφωλιασμένων βρόγχων for
- Να χρησιμοποιούμε κατάλληλους βρόγχους και δομές διακλάδωσης για επεξεργασία στοιχείων ενός πίνακα σύμφωνα με την περιγραφή συγκεκριμένου προβλήματος
- Να χρησιμοποιούμε κατάλληλους βρόγχους και δομές διακλάδωσης για επεξεργασία συγκεκριμένης σειράς ή στήλης, σύμφωνα με την περιγραφή συγκεκριμένου προβλήματος
- Να ορίζουμε τι είναι τετραγωνικός πίνακας και να αναγνωρίζουμε την κύρια και τη δευτερεύουσα διαγώνιό του
- Να χρησιμοποιούμε κατάλληλους βρόγχους και δομές διακλάδωσης για επεξεργασία στοιχείων τετραγωνικού πίνακα που έχουν σχέση με την κύρια και δευτερεύουσα διαγώνιο αλλά και στοιχείων που βρίσκονται πάνω και κάτω από την κύρια και δευτερεύουσα διαγώνιο
- Να ορίζουμε και να χρησιμοποιούμε παράλληλους δισδιάστατους και μονοδιάστατους παράλληλους πίνακες σε ένα πρόγραμμα
- Να καθορίζουμε το πρόβλημα με ακρίβεια, συγκεκριμένα: να εντοπίζουμε/διακρίνουμε τα Δεδομένα, τις Πληροφορίες και την Επεξεργασία
- Να αποφασίζουμε εάν χρειάζονται πίνακες για την επίλυση του προβλήματος και να τους καθορίζουμε
- Να σχεδιάζουμε τον τρόπο επίλυσης του προβλήματος
- Να επιλέγουμε κατάλληλες δομές (επανάληψης ή και διακλάδωσης) ανάλογα με τις δυνατότητες, τους περιορισμούς και τα χαρακτηριστικά της για επίλυση του προβλήματος
- Να υλοποιούμε τον σχεδιασμό τους σε πρόγραμμα με τη χρήση του προγραμματιστικού περιβάλλοντος, ώστε να επιλυθεί το πρόβλημα
- Να επιλέγουμε κατάλληλα δεδομένα και στρατηγική για έλεγχο του προγράμματος
- Να ελέγχουμε την ορθότητα της λύσης του προβλήματος, χρησιμοποιώντας τη μέθοδο της προκαταρκτικής εκτέλεσης ή και άλλες μεθόδους για επαλήθευση
- Να μελετούμε έτοιμο πρόγραμμα το οποίο περιλαμβάνει μονοδιάστατους και δισδιάστατους πίνακες και να εντοπίζουμε βασικά μέρη του τα οποία συνδέονται με πτυχές του προβλήματος που επιλύει
- Να προσθέτουμε επεξηγηματικά σχόλια σε ένα πρόγραμμα
- Να συμπληρώνουμε ένα έτοιμο πρόγραμμα με μονοδιάστατους και δισδιάστατους πίνακες και δομή/δομές διακλάδωσης και επανάληψης, ώστε να αποτελεί λύση ενός διαφοροποιημένου προβλήματος
- Να εντοπίζουμε και να αναγνωρίζουμε σε ένα πρόγραμμα πρότυπα σχεδίασης και στρατηγικές (design patterns, τμήματα κώδικα)
- Να χρησιμοποιούμε πρότυπα σχεδίασης και στρατηγικές που εντοπίσαμε ως εργαλεία επίλυσης προβλημάτων σε νέα προγράμματά τους

1. Εισαγωγή

Με τους δισδιάστατους πίνακες (2D arrays) έχουμε τη δυνατότητα να αποθηκεύσουμε στοιχεία ίδιου τύπου σε ένα ορθογώνιο πλέγμα κελιών, το οποίο αποτελείται από πολλές γραμμές και πολλές στήλες. Κάθε στοιχείο του πίνακα χαρακτηρίζεται από δύο μεταβλητές-δείκτες. Έναν δείκτη που υποδεικνύει τη γραμμή (στο πρώτο ζεύγος αγκυλών) και έναν που

υποδεικνύει τη στήλη που βρίσκεται το στοιχείο (στο δεύτερο ζεύγος αγκυλών). Η πιο κάτω δήλωση δεσμεύει χώρο στη μνήμη, ώστε να αποθηκευτούν δεκαπέντε ακέραιοι, χωρισμένοι σε πέντε γραμμές (rows) και τρεις στήλες (columns):

```
int arr[5][3];
```

Η αρίθμηση και των δύο δεικτών ξεκινά **από το μηδέν**. Μπορούμε να φανταστούμε τον δισδιάστατο πίνακα όπως πιο κάτω:

	στήλη j=0	στήλη j=1	στήλη j=2
γραμμή i=0	arr[0][0]	arr[0][1]	arr[0][2]
γραμμή i=1	arr[1][0]	arr[1][1]	arr[1][2]
γραμμή i=2	arr[2][0]	arr[2][1]	arr[2][2]
γραμμή i=3	arr[3][0]	arr[3][1]	arr[3][2]
γραμμή i=4	arr[4][0]	arr[4][1]	arr[4][2]

Μπορούμε να καταχωρίσουμε τιμές στα στοιχεία του πίνακα όπως ακριβώς σε μία μεταβλητή:

```
arr[0][0] = 17;
arr[1][2] = 5;
arr[4][3] = arr[1][2] + 19;
```

Σε έναν δισδιάστατο πίνακα με όνομα `arr`, διαστάσεων `N x M`, ισχύει ότι:

- Το στοιχείο στην πάνω αριστερή γωνία του πίνακα είναι το στοιχείο `arr[0][0]`
- Το στοιχείο στην πάνω δεξιά γωνία του πίνακα είναι το στοιχείο `arr[0][M-1]`
- Το στοιχείο στην κάτω αριστερή γωνία του πίνακα είναι το στοιχείο `arr[N-1][0]`
- Το στοιχείο στην κάτω δεξιά γωνία του πίνακα είναι το στοιχείο `arr[N-1][M-1]`

2. Διαγώνιοι του πίνακα

Ένας τετραγωνικός πίνακας διαστάσεων N x N έχει δύο διαγώνιους. Η κύρια διαγώνιος του πίνακα (**όπου `i = j`**) εμφανίζεται πιο κάτω με κίτρινο χρώμα:

[0][0]	[0][1]	[0][2]	[0][3]	[0][4]
[1][0]	[1][1]	[1][2]	[1][3]	[1][4]
[2][0]	[2][1]	[2][2]	[2][3]	[2][4]
[3][0]	[3][1]	[3][2]	[3][3]	[3][4]
[4][0]	[4][1]	[4][2]	[4][3]	[4][4]

Για τα στοιχεία του πίνακα που βρίσκονται πάνω από την κύρια διαγώνιο ισχύει ότι `j>i`, ενώ για τα στοιχεία του πίνακα που βρίσκονται κάτω από την κύρια διαγώνιο ισχύει ότι `i>j`.

Η δευτερεύουσα διαγώνιος του πίνακα (**όπου `i = N-j-1`**), εμφανίζεται πιο κάτω με πράσινο χρώμα:

[0][0]	[0][1]	[0][2]	[0][3]	[0][4]
[1][0]	[1][1]	[1][2]	[1][3]	[1][4]
[2][0]	[2][1]	[2][2]	[2][3]	[2][4]
[3][0]	[3][1]	[3][2]	[3][3]	[3][4]
[4][0]	[4][1]	[4][2]	[4][3]	[4][4]

3. Απόδοση αρχικών τιμών σε δισδιάστατο πίνακα

Για να δώσουμε αρχικές τιμές σε έναν δισδιάστατο πίνακα, θα πρέπει κάθε γραμμή να περικλείεται από άγκιστρα. Αν δεν υπάρχουν τιμές για κάποια στοιχεία, τότε τα στοιχεία αυτά παίρνουν τιμή μηδέν. Η πιο κάτω εντολή δημιουργεί τον παρακάτω πίνακα ακεραίων:

```
int arr[3][3] = { {1,2,3},{4,5,6},{7,8,9} };
```

	0	1	2
0	1	2	3
1	4	5	6
2	7	8	9

Αν αλλάξουμε τις διαστάσεις του πίνακα ακεραίων χωρίς να δώσουμε αρχικές τιμές, τότε αυτές θα πάρουν αρχική τιμή μηδέν:

```
int arr[4][4] = { {1,2,3},{4,5,6},{7,8,9} };
```

	0	1	2	3
0	1	2	3	0
1	4	5	6	0
2	7	8	9	0
3	0	0	0	0

4. Εισαγωγή στοιχείων σε δισδιάστατο πίνακα

Για την εισαγωγή στοιχείων σε πίνακα μπορούμε να χρησιμοποιήσουμε δύο εμφωλευμένες δομές επανάληψης `for`, μαζί με δύο μεταβλητές-δείκτες. Μπορεί να γίνει χρήση συνάρτησης και ένας δισδιάστατος πίνακας να δοθεί ως **παράμετρος αναφοράς** στη συνάρτηση, τον οποίο η συνάρτηση θα γεμίζει. Οι δισδιάστατοι πίνακες, όπως και οι μονοδιάστατοι, όταν περνούν παραμετρικά σε μία συνάρτηση, δίδονται πάντοτε ως παράμετροι αναφοράς και **δεν χρειάζεται να γίνει χρήση του τελεστή &.** Η μόνη διάσταση που απαιτείται είναι αυτή

των στηλών, η οποία **πρέπει να είναι σταθερά**, επιτρέπεται όμως να δοθούν και οι δύο διαστάσεις παραμετρικά.

Παράδειγμα 7.1

```
#include <iostream>
using namespace std;
#define ROWS 3        // Δηλώνουμε το πλήθος των γραμμών ως σταθερά
#define COLS 4        // Δηλώνουμε το πλήθος των στηλών ως σταθερά

void fill_array(int arr[][COLS]){   // Ο πίνακας περνά ως παράμετρος
    for (int i=0; i<ROWS; i++)      // αναφοράς δηλώνοντας μόνο το
        for (int j=0; j<COLS; j++) // πλήθος των στηλών
             cin >> arr[i][j];
}

int main(){
     int my_array[ROWS][COLS];        // Δήλωση πίνακα
     fill_array(my_array);            // Κλήση συνάρτησης εισόδου
return 0;
}
```

(Code: C7_7_example1.cpp)

5. Εμφάνιση στοιχείων δισδιάστατου πίνακα

Για την εμφάνιση των στοιχείων ενός δισδιάστατου πίνακα στην οθόνη χρησιμοποιούμε παρόμοιες εμφωλευμένες δομές επανάληψης με την εισαγωγή στοιχείων, αυτή τη φορά με χρήση εντολής εξόδου. Μετά από κάθε εσωτερικό βρόχο προχωρούμε στην επόμενη γραμμή, ώστε να ξεχωρίσουμε τα στοιχεία των γραμμών. Επιπρόσθετα, μπορεί να γίνει χρήση της συνάρτησης setw, ώστε να εμφανίζεται ομοιόμορφη, δεξιά στοίχιση των στοιχείων στην οθόνη.

Παράδειγμα 7.2

```
#include <iostream>
#include <iomanip>    // Για τη χρήση της setw
using namespace std;
#define ROWS 3        // Δηλώνουμε το πλήθος των γραμμών ως σταθερά
#define COLS 4        // Δηλώνουμε το πλήθος των στηλών ως σταθερά

void fill_array(int arr[][COLS]){  // Ο πίνακας περνά ως παράμετρος
    for (int i=0; i<ROWS; i++)
        for (int j=0; j<COLS; j++)
            cin >> arr[i][j];
}
```

```
void print_array(int arr[][COLS]){ // Ο πίνακας περνά ως παράμετρος
    for (int i=0; i<ROWS; i++){
        for (int j=0; j<COLS; j++){
            cout << setw(8) << arr[i][j];   // Εφαρμόζουμε δεξιά
        }                                   // στοίχιση 8 χαρακτήρων
        cout << endl;                       // Αλλαγή γραμμής
    }
}

int main(){
     int my_array[ROWS][COLS];         // Δήλωση πίνακα
     fill_array(my_array);             // Κλήση συνάρτησης εισόδου
     print_array(my_array);            // Κλήση συνάρτησης εμφάνισης
return 0;
}
```

(Code: C7_7_example2.cpp)

6. Εντοπισμός μέγιστου/ελάχιστου στοιχείου γραμμής/στήλης

Για τον εντοπισμό του μέγιστου στοιχείου μίας στήλης ενός δισδιάστατου πίνακα θετικών ακεραίων, μπορούμε να χρησιμοποιήσουμε παρόμοια τεχνική με τους μονοδιάστατους πίνακες. Δηλώνουμε μία μεταβλητή (maxV) η οποία θα έχει αρχικά τιμή μηδέν και ελέγχουμε ένα-ένα όλα τα στοιχεία της στήλης. Κάθε φορά που εντοπίζουμε ένα μεγαλύτερο στοιχείο από την μεταβλητή maxV, βάζουμε την τιμή του στοιχείου αυτού ως τιμή της μεταβλητής maxV.

Για την υλοποίηση του πιο πάνω κάνουμε χρήση μίας συνάρτησης (find_col_max) που δέχεται παραμετρικά τον πίνακα και τον αριθμό της στήλης και μας επιστρέφει το μέγιστο στοιχείο της στήλης αυτής.

Παράδειγμα 7.3

```
#include <iostream>
using namespace std;
#define ROWS 6
#define COLS 8

void fill_array(int arr[][COLS]){
    for (int i=0; i<ROWS; i++)
        for (int j=0; j<COLS; j++)
            cin >> arr[i][j];
}

int find_col_max(int arr[][COLS], int col){
     int maxV = 0;              // Μεταβλητή υπολογισμού μέγιστου
                                // στοιχείου. Αρχικοποιείται με 0
```

```
for (int i=0; i<ROWS; i++){
           if (arr[i][col] > maxV)     // Ο αριθμός της στήλης
                maxV = arr[i][col];   // δίνεται παραμετρικά
        }
 return maxV;                          // Επιστροφή μέγιστου στοιχείου
}

int main(){

      int col_max;
      int my_array[ROWS][COLS];         // Δήλωση πίνακα
      fill_array(my_array);             // Κλήση συνάρτησης εισόδου

      cout << "Give column (0-7): ";
      cin >> col_max;                   // Δέχεται τη στήλη (0-7)

      // Καλεί τη συνάρτηση η οποία επιστρέφει το μέγιστο στοιχείο
      cout << find_col_max(my_array, col_max) << endl;

return 0;
}
```

(Code: C7_7_example3.cpp)

Η εύρεση του ελάχιστου στοιχείου μίας γραμμής του πίνακα, αφήνεται ως άσκηση (7.17).

7. Συνδυασμός δισδιάστατων πινάκων με παράλληλους πίνακες

Πολλές φορές μπορεί να χρειαστεί να συνδυάσουμε δεδομένα διαφορετικών τύπων. Σε τέτοια περίπτωση μπορούμε να χρησιμοποιήσουμε παράλληλους μονοδιάστατους πίνακες ως προέκταση των δισδιάστατων, για να υπολογίσουμε τα επιθυμητά αποτελέσματα.

Παράδειγμα 7.4

Οι 150 μαθητές της Α' τάξης ενός γυμνασίου της Κύπρου εξετάστηκαν σε ένα διαγώνισμα γενικών γνώσεων. Το διαγώνισμα περιλάμβανε 20 ερωτήσεις πολλαπλής επιλογής και για κάθε ερώτηση δίνονταν 5 επιλογές (A, B, C, D και E). Να δημιουργήσετε πρόγραμμα, το οποίο να:

(α) ζητά από τον χρήστη τις σωστές απαντήσεις για τις 20 ερωτήσεις που δόθηκαν στο διαγώνισμα και να τις καταχωρεί στον μονοδιάστατο πίνακα Lyseis

(β) ζητά από τον χρήστη τις απαντήσεις που έδωσαν οι μαθητές στο διαγώνισμα και να τις καταχωρεί σε έναν δισδιάστατο πίνακα 150 γραμμών και 20 στηλών με το όνομα Apantiseis, ο οποίος είναι παράλληλος με τον πίνακα Lyseis. Κάθε γραμμή του πίνακα Apantiseis θα αντιστοιχεί με τις 20 απαντήσεις ενός μαθητή στο διαγώνισμα

Πίνακας Lyseis

B	D	C	D		A	A	B

Λύση του διαγωνίσματος (οι σωστές απαντήσεις)

Πίνακας Apantiseis

A	A	C	A		A	C	A
B	D	B	D		C	B	D
A	D	D	D		D	D	A
E	E	C			E	A	B
B	A	B			A	A	B

Π.χ. η $3^{η}$ γραμμή του πίνακα περιέχει τις απαντήσεις του $3^{ου}$ μαθητή στο διαγώνισμα

(γ) υπολογίζει τον συνολικό βαθμό του κάθε μαθητή στο διαγώνισμα, δεδομένου ότι η κάθε σωστή απάντηση βαθμολογείται με 5 μονάδες. Οι βαθμοί όλων των μαθητών να καταχωρούνται σε μονοδιάστατο πίνακα με το όνομα Vathmoi, ο οποίος είναι παράλληλος με τον πίνακα Apantiseis

(δ) ταξινομεί σε φθίνουσα σειρά τα στοιχεία του πίνακα Vathmoi και, ακολούθως, να τα τυπώνει

(ε) ζητά από τον χρήστη τον αριθμό μίας συγκεκριμένης ερώτησης. Ακολούθως, να καλεί μία συνάρτηση με το όνομα erotisi, η οποία να δέχεται παραμετρικά τους πίνακες Lyseis και Apantiseis, καθώς και τον αριθμό της ερώτησης τον οποίο έδωσε ο χρήστης. Η συνάρτηση να υπολογίζει και να επιστρέφει στην κύρια συνάρτηση (main) τον αριθμό των μαθητών που απάντησαν σωστά σε αυτή την ερώτηση.

```
π.χ. Give question number (1-20): 18
This question was answered correctly by 73 students.
```

Το πρόγραμμα πρέπει να εμφανίζει στην οθόνη τα κατάλληλα μηνύματα για την εισαγωγή των δεδομένων και την εξαγωγή των αποτελεσμάτων. Να θεωρήσετε ότι τα δεδομένα δίνονται σωστά και δεν χρειάζεται έλεγχος.

(Παγκύπριες Εξετάσεις 2010)

```
#include <iostream>
using namespace std;
#define Q 20
#define N 150
int erotisi(char sol[], char ans[][Q], int quest){
    int correct = 0;
     for (int i=0; i<N; i++)
        if (sol[quest]==ans[i][quest])
            correct++;
return correct;
}
void insertionSort(int A[], int n){
    int temp;
```

```
    int j;
    for (int i=1; i<n; i++){
            temp = A[i];
            j = i - 1;
            while (j>=0 && A[j]<temp){
                A[j+1] = A[j];
                j--;
              }
        A[j+1] = temp;
    }
}
int main(){
 char Lyseis[Q];
 char Apantiseis[N][Q];
 int Vathmoi[N], question;

//Αρχικοποίηση πίνακα βαθμών
 for (int i=0; i<N; i++)
    Vathmoi[i]=0;

// Είσοδος σωστών απαντήσεων
 cout << "Input correct answers: " << endl;
    for (int i=0; i<Q; i++)
        cin >> Lyseis[i];

// Είσοδος απαντήσεων μαθητών
 cout << "Input student answers: " << endl;
 for (int i=0; i<N; i++)
    for (int j=0; j<Q; j++){
        cin >> Apantiseis[i][j];
// Υπολογισμός βαθμών +5 σε κάθε σωστή απάντηση
        if (Apantiseis[i][j] == Lyseis[j])
            Vathmoi[i] += 5;
     }
// Ταξινόμηση πίνακα βαθμών
 insertionSort(Vathmoi, N);
// Εμφάνιση βαθμών
 cout << "Student Grades"<< endl;
    for (int i=0; i<N; i++)
        cout << Vathmoi[i] << " ";
 cout << endl;
```

```
// Σωστές απαντήσεις συγκεκριμένης ερώτησης
 cout << "Give question number (1-20): ";
 cin >> question;
 cout << "This question was answered correctly by ";
 cout << erotisi(Lyseis,Apantiseis,question-1)<<" students."<<endl;
return 0;
}
```

(Code: C7_7_example4.cpp)

Παράδειγμα 7.5

Να δημιουργήσετε πρόγραμμα, το οποίο να αποθηκεύει τον πίνακα προπαίδειας, από το 1 ως το 10, σε δισδιάστατο πίνακα και να τον εμφανίζει στην οθόνη. Να εφαρμοστεί δεξιά στοίχιση τεσσάρων χαρακτήρων.

```
#include <iostream>
#include <iomanip>
using namespace std;
#define N 10          // Δηλώνουμε τις διαστάσεις ως σταθερά

void fill_array(int arr[][N]){  // Ο πίνακας περνά ως παράμετρος
    for (int i=0; i<N; i++)
        for (int j=0; j<N; j++)
            arr[i][j] = (i+1)*(j+1);   // Αυξάνουμε κατά έναν τους
}                                      // δείκτες για να ξεκινήσουμε
                                       // από τον αριθμό ένα
void print_array(int arr[][N]){ // Ο πίνακας περνά ως παράμετρος
    for (int i=0; i<N; i++){
        for (int j=0; j<N; j++){
            cout << setw(4) << arr[i][j];   // Εφαρμόζουμε δεξιά
        }                                   // στοίχιση 4 χαρακτήρων
        cout << endl;                       // Αλλαγή γραμμής
    }
}

int main(){

     int my_array[N][N];              // Δήλωση πίνακα
     fill_array(my_array);            // Κλήση συνάρτησης εισόδου
     print_array(my_array);           // Κλήση συνάρτησης εμφάνισης

return 0;
}
```

(Code: C7_7_example5.cpp)

Παράδειγμα 7.6

Να δημιουργήσετε πρόγραμμα, το οποίο να:

(α) διαβάζει τα στοιχεία ενός δισδιάστατου πίνακα ακεραίων με το όνομα numbers, που αποτελείται από 4 γραμμές και 6 στήλες

(β) υπολογίζει και να εμφανίζει το άθροισμα των στοιχείων κάθε γραμμής

(γ) υπολογίζει και να εμφανίζει το άθροισμα των στοιχείων κάθε στήλης

```
#include <iostream>
using namespace std;

// Η δήλωση των πινάκων ακεραίων εκτός της κύριας συνάρτησης
// μας επιτρέπει να αρχικοποιήσουμε όλα τα στοιχεία τους με τιμή 0
int sum_of_rows[4];                    // Πίνακας αθροισμάτων γραμμών
int sum_of_cols[6];                    // Πίνακας αθροισμάτων στηλών

int main(){

 int numbers[4][6];
 for (int i=0; i<4; i++){
      for (int j=0; j<6; j++){
            cin >> numbers[i][j];
            sum_of_rows[i] += numbers[i][j];  // Το ίδιο στοιχείο
            sum_of_cols[j] += numbers[i][j];  // υπολογίζεται τόσο στο
            }                                 // άθροισμα της γραμμής
      }                                       // όσο και της στήλης
 for (int i=0; i<4; i++)
      cout << sum_of_rows[i] << " ";
 cout << endl;
 for (int j=0; j<6; j++)
      cout << sum_of_cols[j] << " ";
return 0;
}
```

(Code: C7_7_example6.cpp)

Παράδειγμα 7.7

Ανάστροφος (ΑΤ) ενός πίνακα Α καλείται ο πίνακας που προκύπτει από τον Α, αν οι γραμμές γίνουν στήλες και οι στήλες γραμμές, με την ίδια ακολουθία (δηλαδή, η πρώτη γραμμή να γίνει πρώτη στήλη, η δεύτερη γραμμή να γίνει δεύτερη στήλη κ.ο.κ.). Να δημιουργήσετε πρόγραμμα, το οποίο να διαβάζει τις διαστάσεις ενός πίνακα N x M (2<=N,M<=100), καθώς και τα στοιχεία του πίνακα που ακολουθούν. Στη συνεχεία, να γεμίζει και να εμφανίζει στην οθόνη τον ανάστροφο του πίνακα, με δεξιά στοίχιση 5 χαρακτήρων.

Παράδειγμα εισόδου

```
3 4
1 2 3 4
5 6 7 8
9 10 11 12
```

Παράδειγμα εξόδου

```
    1    5    9
    2    6   10
    3    7   11
    4    8   12
```

```
#include <iostream>
#include <iomanip>
using namespace std;
#define MAXT 100

int main(){

 int A[MAXT][MAXT], AT[MAXT][ MAXT], N, M;

 cin >> N >> M;                  // Διαβάζουμε τις διαστάσεις του Α

 for (int i=0; i<N; i++)
      for (int j=0; j<M; j++){
           cin >> A[i][j];       // Το στοιχείο τοποθετείται στις
           AT[j][i] = A[i][j];   // αντίστοιχες διαστάσεις του
       }                         // ανάστροφου πίνακα ΑΤ

 for (int i=0; i<M; i++){
       for (int j=0; j<N; j++)
            cout << setw(5) << AT[i][j];
   cout << endl;
 }

return 0;
}
```

(Code: C7_7_example7.cpp)

Παράδειγμα 7.8

Σε ένα ηλεκτρονικό παιχνίδι, ένα υποβρύχιο επικοινωνεί με το αρχηγείο του ναυτικού της χώρας του με κωδικοποιημένα μηνύματα σε μορφή τετραγωνικού πίνακα 10 x 10, του οποίου τα κελιά περιέχουν τους χαρακτήρες Α και Β. Η αντίπαλη χώρα έχει υποκλέψει ένα από τα μηνύματα αυτά και θέλει να το αποκωδικοποιήσει, γιατί φοβάται πιθανή πολεμική σύρραξη. Η πληροφορία που πήρε από τις μυστικές υπηρεσίες είναι:

Αν το πλήθος των χαρακτήρων Α που έχει η κύρια διαγώνιος είναι ίσο με 5 και το πλήθος των χαρακτήρων Β στα κελιά που βρίσκονται πάνω από την κύρια διαγώνιο είναι περισσότερα από το πλήθος των χαρακτήρων Α που βρίσκονται στα κελιά κάτω από την κύρια διαγώνιο, τότε η χώρα θα εμπλακεί σε πόλεμο, διαφορετικά δεν υπάρχει κίνδυνος πολεμικής σύρραξης.

Παράδειγμα

code

A	A	B	B	A	B	A	A	A	A
A	A	B	A	B	A	A	A	A	A
B	A	B	B	A	B	B	B	B	B
A	B	A	A	A	A	A	A	A	A
A	B	B	A	B	A	A	B	B	B
B	A	A	B	A	A	B	A	A	A
A	A	B	A	A	A	B	A	B	B
B	B	B	A	B	B	A	B	B	A
B	A	A	B	A	A	A	A	B	B
A	B	B	A	A	B	A	B	B	A

Στοιχεία κύριας διαγωνίου

- Πλήθος χαρακτήρων Α κύριας διαγώνιου = 5
- Πλήθος χαρακτήρων Β που βρίσκονται πάνω από την κύρια διαγώνιο = 19
- Πλήθος χαρακτήρων Α που βρίσκονται κάτω από την κύρια διαγώνιο = 26

Να δημιουργήσετε πρόγραμμα, το οποίο να εισάγει τους χαρακτήρες Α και Β σε έναν τετραγωνικό πίνακα 10Χ10 με το όνομα code. Ακολούθως, αφού τον αποκωδικοποιήσει όπως έχει περιγραφεί πιο πάνω, να τυπώνει το μήνυμα «Danger», αν οι δύο χώρες θα εμπλακούν σε πόλεμο και το μήνυμα «No Danger», αν το πιο πάνω σενάριο δεν ισχύει. Να υποθέσετε ότι η εισαγωγή των χαρακτήρων Α και Β γίνεται σωστά και δεν χρειάζεται έλεγχος.

Με βάση το πιο πάνω παράδειγμα, το μήνυμα που θα εμφανίζεται είναι «No Danger».

(Παγκύπριες Εξετάσεις 2016)

```
#include <iostream>
using namespace std;

int main(){
 char code[10][10];          // Δηλώνουμε τον πίνακα code (10 x 10)
 int sum_on = 0, sum_over = 0, sum_below = 0;
```

```
 for (int i=0; i<10; i++)
      for (int j=0; j<10; j++){
            cin >> code[i][j];
            if (code[i][j] == 'A'){     // Αν ο χαρακτήρας είναι το Α
                  if (i == j)           // και βρίσκεται στη
                        sum_on++;       // διαγώνιο, αύξησε το sum_on
                  else
                        if (i > j)      // Αν ο χαρακτήρας είναι το Α
                           sum_below++; // και βρίσκεται κάτω από τη
                 }                      // διαγώνιο, η τιμή του i θα
            else                        // είναι μεγαλύτερη του j
                  if (i < j)            // Αν ο χαρακτήρας είναι το Β
                        sum_over++;     // και βρίσκεται πάνω από
                                        // τη διαγώνιο, η τιμή του i
      }                                 // θα είναι μικρότερη του j
 if (sum_over > sum_below && sum_on == 5)
      cout << "Danger";
 else
      cout << "No Danger";

return 0;
}
```

(Code: C7_7_example8.cpp)

Παράδειγμα 7.9

Στο παιχνίδι «Ναρκαλιευτής», όταν εντοπιστεί μία νάρκη, τα προσκείμενα (γειτονικά) σε αυτή σημεία (8 συνολικά αν είναι εντός ορίων) παίρνουν τιμή 1, υποδεικνύοντας την παρουσία της νάρκης. Να δημιουργήσετε πρόγραμμα, το οποίο να διαβάζει, αρχικά, τις διαστάσεις N x M (2<=N,M<=100) του πεδίου και, ακολούθως, να διαβάζει τους χαρακτήρες που το αποτελούν. Στο πεδίο θα υπάρχει μόνο μία νάρκη, η οποία θα συμβολίζεται με τον χαρακτήρα 'X'. Ο χαρακτήρας '.' θα συμβολίζει τα υπόλοιπα κενά σημεία. Το πρόγραμμα πρέπει να εντοπίζει τη νάρκη και να βάζει στα προσκείμενα σημεία που βρίσκονται εντός ορίων τον αριθμό 1. Το τελικό πεδίο να εμφανίζεται στην οθόνη.

Παράδειγμα εισόδου

```
7 5
. . . . .
. . . . .
. . . . .
. . . X .
. . . . .
. . . . .
. . . . .
```

Παράδειγμα εξόδου

```
. . . . .
. . . . .
. . 1 1 1
. . 1 X 1
. . 1 1 1
. . . . .
. . . . .
```

```
#include <iostream>
using namespace std;
#define MAXT 100

int main() {

 int ni[8]= {-1, -1, -1, 1, 1, 1, 0, 0};    // Οι θέσεις των οκτώ
 int nj[8]= {0, -1, 1, 0, 1, -1, -1, 1};    // γειτονικών σημείων

 char field[MAXT][ MAXT];
 int n, m;
 cin >> n >> m;

 for (int i=0; i<n; i++)
     for (int j=0; j<m; j++)
          cin >> field[i][j];

 for (int i=0; i<n; i++){
     for (int j=0; j<m; j++){
          if (field[i][j]=='X'){        // Αν εντοπίσουμε νάρκη,
              for (int k=0; k<8; k++){  // ελέγχουμε τα οκτώ
                                        // γειτονικά σημεία

           if((field[i+ni[k]][j+nj[k]]=='.') &&    // Αν είναι κενό
                  (i+ni[k]>=0)&&(j+nj[k]>=0) &&    // και βρίσκεται
                  (i+ni[k]<n)&&(j+nj[k]<m))        // εντός ορίων,
                    field[i+ni[k]][j+nj[k]]='1';  // βάλε τιμή 1
              }
          }
     }
 }
```

```
for (int i=0; i<n; i++){                    // Εμφάνιση πεδίου
      for (int j=0; j<m; j++)
            cout << field[i][j] << " ";
      cout << endl;
 }
 return 0;
}
```

(Code: C7_7_example9.cpp)

Ασκήσεις Κεφαλαίου - www.hackerrank.com/g77

Άσκηση 7.1

Δίνεται ο πιο κάτω πίνακας με το όνομα matrix:

10	20	30	40	50
60	70	80	90	100
110	120	130	140	150
160	170	180	190	200

(α) Να δώσετε την εντολή δήλωσης ενός κενού πίνακα matrix.

(β) Ποιο είναι το στοιχείο `matrix[2][3];`

(γ) Ποιο είναι το αποτέλεσμα της εντολής: `cout << matrix[3][2] / matrix[1][0];`

(δ) Να συμπληρώσετε τον πιο κάτω κώδικα, ο οποίος γεμίζει με τους αριθμούς που βλέπετε πιο πάνω, τον πίνακα matrix.

```
int num = 10;
for (int i = ___ ; i < ___ ; i++){
      for (int j =___ ; j < ___ ; j++){
            matrix[i][j] = ____________ ;
            num = ____________ ;
      }
}
```

Άσκηση 7.2

Να βρείτε τις τιμές των στοιχείων του πίνακα M μετά από κάθε εντολή που βλέπετε στα δεξιά. Να θεωρήσετε ότι κάθε εντολή θα είναι για τον αρχικό πίνακα:

21	9	22
4	11	-26
12	16	20
-2	42	4
35	18	2

(α) `M[0][2] = M[3][2] + M[4][1];`

(β) `M[2][2] = M[2][1] + M[0][0];`

(γ) `M[3][0] = M[3][1] - M[1][0];`

(δ) `M[1][2] = M[1][1] - M[2][2];`

(ε) `M[3][3] = M[4][2] - M[2][0];`

Άσκηση 7.3

Να δημιουργήσετε πρόγραμμα, το οποίο να γεμίζει και να εμφανίζει τον πιο κάτω πίνακα στην οθόνη.

```
1 1 1 1 1 1 1 1 1
2 2 2 2 2 2 2 2 2
3 3 3 3 3 3 3 3 3
4 4 4 4 4 4 4 4 4
5 5 5 5 5 5 5 5 5
6 6 6 6 6 6 6 6 6
7 7 7 7 7 7 7 7 7
8 8 8 8 8 8 8 8 8
9 9 9 9 9 9 9 9 9
```

Άσκηση 7.4

Να μετατρέψετε το πρόγραμμα της άσκησης **7.3**, ώστε να γεμίζει και να εμφανίζει τον πιο κάτω πίνακα στην οθόνη.

```
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
```

Άσκηση 7.5

Ποιο θα είναι το αποτέλεσμα του πιο κάτω κώδικα; Να συμπληρώσετε τον πίνακα test που βλέπετε πιο κάτω:

Πίνακας test


```
int m = 16;
for (int i = 0; i < 3; i++){
      for (int j = 0; j < 4; j++){
            test[i][j] = (m + 2) % 3;
            m += 4;
      }
}
```

Άσκηση 7.6

Να δημιουργήσετε πρόγραμμα, το οποίο να καλεί μία συνάρτηση με το όνομα fill_array η οποία θα δέχεται παραμετρικά έναν δισδιάστατο πίνακα χαρακτήρων, διαστάσεων 3 x 4 και θα τον γεμίζει με στοιχεία που θα δίνει ο χρήστης από το πληκτρολόγιο.

Άσκηση 7.7

Στο πρόγραμμα που δημιουργήσατε στην άσκηση **7.6** να προσθέσετε και μία συνάρτηση με το όνομα print_array, η οποία να δέχεται παραμετρικά έναν δισδιάστατο πίνακα χαρακτήρων, διαστάσεων 3 x 4 και να τον εμφανίζει στην οθόνη σε τρεις γραμμές και τέσσερις στήλες, με δεξιά στοίχιση τριών χαρακτήρων.

Άσκηση 7.8

Να δημιουργήσετε πρόγραμμα, το οποίο να δηλώνει έναν δισδιάστατο πίνακα ακεραίων, διαστάσεων 6 x 8 και να καταχωρίζει σε όλα τα στοιχεία του την τιμή δέκα (10).

Άσκηση 7.9

Να δημιουργήσετε πρόγραμμα, το οποίο να δηλώνει έναν δισδιάστατο πίνακα ακεραίων, διαστάσεων 10 x 10 και να καταχωρίζει σε αυτόν τους αριθμούς από το 1 μέχρι το 100.

Άσκηση 7.10

Να δημιουργήσετε πρόγραμμα, το οποίο να γεμίζει, αρχικά, έναν μονοδιάστατο πίνακα ακεραίων arr1[100] με τους αριθμούς από το 101 μέχρι το 200 και να τον αντιγράφει στον δισδιάστατο πίνακα ακεραίων arr2[10][10]. Δεν θα δίνονται δεδομένα από τον χρήστη.

Άσκηση 7.11

Να δημιουργήσετε πρόγραμμα, το οποίο να γεμίζει, αρχικά, έναν δισδιάστατο πίνακα ακεραίων arr1[10][10] με τους αριθμούς από το 200 μέχρι το 101 και να τον αντιγράφει στον μονοδιάστατο πίνακα ακεραίων arr2[100]. Δεν θα δίνονται δεδομένα από τον χρήστη.

Άσκηση 7.12

Να συμπληρώστε τα στοιχεία του πίνακα B, με βάση τον πιο κάτω κώδικα.

Πίνακας A

1	0	0
1	1	1
0	2	0

Πίνακας B


```
for (int i = 0; i < 3; i++)
      for (int j = 0; j < 3; j++)
            B[i][j] = A[A[j][i]][A[i][j]];
```

Άσκηση 7.13

Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται 16 ακέραιους αριθμούς και να:

(α) τους αποθηκεύει στον πίνακα Α, διαστάσεων 4 x 4

(β) βρίσκει και να εμφανίζει τον μεγαλύτερο αριθμό της κάθε γραμμής και να τον αποθηκεύει στον μονοδιάστατο πίνακα maxR, μεγέθους 4

(γ) βρίσκει και να εμφανίζει το πλήθος των γραμμών, οι οποίες έχουν μόνο θετικούς αριθμούς.

Άσκηση 7.14

Να δημιουργήσετε πρόγραμμα, το οποίο να ελέγχει αν ένας τετραγωνικός πίνακας διαστάσεων 3x3 είναι έγκυρος πίνακας για το παιχνίδι SUDOKU. Για να είναι έγκυρος ένας πίνακας, πρέπει:

(α) να περιέχει μόνο τους αριθμούς από το 1 μέχρι το 9 και

(β) ο κάθε αριθμός να εμφανίζεται μόνο μία φορά

Το πρόγραμμα να διαβάζει έναν πίνακα SUDOKU, διαστάσεων 3x3 και να ενημερώνει τον χρήστη αν ο πίνακας είναι έγκυρος (Valid) ή όχι (Not valid).

Παράδειγμα εισόδου

```
4 5 6
9 8 7
1 3 2
```

Παράδειγμα εξόδου

```
Valid
```

Άσκηση 7.15

Να δημιουργήσετε πρόγραμμα, το οποίο να:

(α) διαβάζει τα στοιχεία ενός δισδιάστατου πίνακα ακεραίων, διαστάσεων 5 x 5

(β) υπολογίζει και να εμφανίζει το άθροισμα κάθε γραμμής του πίνακα

(γ) βρίσκει το μέγιστο άθροισμα και την γραμμή στην οποία βρίσκεται.

Παράδειγμα εισόδου

```
1 2 3 4 5
6 7 8 9 8
7 6 5 4 3
2 1 2 3 4
5 6 7 8 7
```

Παράδειγμα εξόδου

```
sums: 15 38 25 12 33
max: 38 line: 1
```

Άσκηση 7.16

Να δημιουργήσετε πρόγραμμα, το οποίο αρχικά να δέχεται έναν περιττό ακέραιο N (3<=N<=99), που αντιστοιχεί στις διαστάσεις N x N ενός δισδιάστατου πίνακα. Στη

συνέχεια, να δέχεται τα στοιχεία του πίνακα, να υπολογίζει και να εμφανίζει το άθροισμα των στοιχείων της κύριας διαγωνίου και το άθροισμα των στοιχείων της δευτερεύουσας διαγωνίου.

Παράδειγμα εισόδου

```
3
6 4 5
1 2 3
7 8 9
```

Παράδειγμα εξόδου

```
17 14
```

Άσκηση 7.17

Να δημιουργήσετε πρόγραμμα, το οποίο να χρησιμοποιεί:

(α) συνάρτηση με το όνομα fill, για να διαβάζει τα στοιχεία ενός δισδιάστατου πίνακα ακεραίων, διαστάσεων 6 x 5

(β) συνάρτηση με το όνομα find_min_row, η οποία να δέχεται παραμετρικά τον πίνακα και τον αριθμό μίας γραμμής (0-5) και να επιστρέφει το ελάχιστο στοιχείο της γραμμής αυτής.

Παράδειγμα εισόδου

```
1 2 3 4 5
6 7 8 9 8
7 6 5 4 3
2 1 2 3 4
5 6 7 8 7
1 4 2 6 7
1
```

Παράδειγμα εξόδου

```
6
```

Άσκηση 7.18

Στο πρωτάθλημα ποδοσφαίρου λαμβάνουν μέρος 14 ομάδες. Κάθε ομάδα παίρνει 3 βαθμούς για κάθε νίκη, 1 βαθμό για κάθε ισοπαλία και 0 βαθμούς για κάθε ήττα.

Να δημιουργήσετε πρόγραμμα, το οποίο να:

(α) δέχεται τα ονόματα των ομάδων και να τα καταχωρίζει στον μονοδιάστατο πίνακα teams. Επίσης, να καταχωρίζει στον δισδιάστατο πίνακα matches τις νίκες, τις ισοπαλίες και τις ήττες που έχει κάθε ομάδα

(β) χρησιμοποιεί τη συνάρτηση calculate, για να υπολογίσει τη συνολική βαθμολογία της κάθε ομάδας και να την καταχωρίζει στον πίνακα total

(γ) χρησιμοποιεί τη συνάρτηση sort, για να ταξινομήσει και να τυπώσει τα ονόματα των ομάδων και τις βαθμολογίες τους σε φθίνουσα σειρά, με βάση τη βαθμολογία τους, με δεξιά στοίχιση 20 και 10 χαρακτήρων, αντίστοιχα.

Άσκηση 7.19

Ο Παντελής έχει στην κατοχή του έναν χάρτη θησαυρού. Με το σημείο 'X' πάνω στον χάρτη εμφανίζονται τα σημεία που πιθανόν να είναι κρυμμένος ο θησαυρός. Ο Παντελής θέλει να ξέρει ποιες είναι οι συντεταγμένες αυτών των σημείων. Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται, αρχικά, δύο ακέραιους αριθμούς N και M (2<=N,M<=100), που αντιστοιχούν στις διαστάσεις του ορθογώνιου πλέγματος σημείων που αντιπροσωπεύει τον χάρτη. Στη συνέχεια, να διαβάζει τους χαρακτήρες του πλέγματος, όπου τα πιθανά σημεία που είναι κρυμμένος ο θησαυρός θα εμφανίζονται με 'X' και τα υπόλοιπα με '.' (τελεία). Δεν θα υπάρχουν άλλοι χαρακτήρες. Στο χάρτη θα υπάρχει τουλάχιστον ένας χαρακτήρας 'X'. Το πρόγραμμα να εμφανίζει τις συντεταγμένες των σημείων που εμφανίζονται με 'X' πάνω στον χάρτη, όπως στο παράδειγμα εξόδου πιο κάτω.

Παράδειγμα εισόδου

```
4 5
..X..
.....
.....
...X.
```

Παράδειγμα εξόδου

```
(0,2)
(3,3)
```

Άσκηση 7.20

Να δημιουργήσετε πρόγραμμα, το οποίο να ελέγχει αν έχουμε νικητή στο παιχνίδι τρίλιζα. Το πρόγραμμα να δέχεται τα σημεία του παιχνιδιού και να εμφανίζει τα κατάλληλα μηνύματα («X wins», «O wins» ή «Draw»).

Παράδειγμα εισόδου 1

```
X X O
X X O
O O X
```

Παράδειγμα εξόδου 1

```
X wins
```

Παράδειγμα εισόδου 2

```
X O O
O X X
O X O
```

Παράδειγμα εξόδου 2

```
Draw
```

Άσκηση 7.21

Ο Παντελής παίζει το παιχνίδι «Ναυμαχία» με έναν φίλο του. Στόχος του παιχνιδιού είναι να βυθίσει όλα τα πλοία του αντιπάλου, προτού βυθίσει αυτός τα δικά του. Το παιχνίδι παίζεται σε ένα πλέγμα διαστάσεων N x N και ο Παντελής μπορεί να κάνει M προβλέψεις σχετικά με

τις θέσεις των πλοίων του αντιπάλου. Αν ο Παντελής καταφέρει να βρει και να βυθίσει όλα τα πλοία του αντιπάλου, είναι ο νικητής του παιχνιδιού.

Να δημιουργήσετε πρόγραμμα, το οποίο να διαβάζει με τη σειρά:

- έναν ακέραιο αριθμό N (2<=N<=100), που υποδεικνύει τις διαστάσεις N x N του πλέγματος που παίζεται το παιχνίδι
- ένα πλέγμα με τους χαρακτήρες '.' για τα σημεία χωρίς πλοίο ή τον χαρακτήρα 'S' για τα πλοία του αντιπάλου. Κάποια πλοία μπορεί να εφάπτονται
- έναν ακέραιο αριθμό M (1<=M<=20), που υποδεικνύει το πλήθος των προβλέψεων του Παντελή για τις θέσεις των πλοίων του αντιπάλου
- M ζεύγη συντεταγμένων (x, y) για κάθε θέση που έχει προβλέψει ο Παντελής.

Το πρόγραμμα να εμφανίζει τη λέξη «victory», αν ο Παντελής έχει βυθίσει όλα τα πλοία του φίλου του, ή, σε αντίθετή περίπτωση, να εμφανίζει το αρχικό πλέγμα μετά τις προβλέψεις, όπου:

- Η πρόβλεψη που έχει κάνει ο Παντελής και δεν έχει βρει πλοίο εμφανίζεται με τον χαρακτήρα 'M' (miss).
- Η πρόβλεψη που έχει κάνει ο Παντελής και έχει βρει πλοίο εμφανίζεται με τον χαρακτήρα 'H' (hit).

Παράδειγμα εισόδου

```
3
...
..S
..S
3
0 2
1 2
2 0
```

Παράδειγμα εξόδου

```
..M
..H
M.S
```

Επεξήγηση: Στο αρχικό πλέγμα υπάρχουν δύο πλοία και ο Παντελής κάνει τρεις προβλέψεις, από τις οποίες μόνο η μία (1, 2) πετυχαίνει ένα πλοίο.

Άσκηση 7.22

Το παιχνίδι Sudoku για έναν πίνακα 9 x 9 ακολουθεί τους εξής κανόνες:

(α) Χρησιμοποιεί τους αριθμούς 1 μέχρι 9.

(β) Σε κάθε γραμμή και σε κάθε στήλη οι αριθμοί πρέπει να είναι μοναδικοί.

(γ) Σε κάθε υποπίνακα 3 x 3 οι αριθμοί πρέπει να είναι μοναδικοί.

Να δημιουργήσετε πρόγραμμα, το οποίο να διαβάζει έναν πίνακα Sudoku από το αρχείο sudoku.txt και να εμφανίζει στην οθόνη τη λέξη «correct», αν είναι σωστό, ή τη λέξη «wrong», αν είναι λάθος.

Παράδειγμα εισόδου

```
2 3 7 6 1 8 4 9 5
9 5 1 7 3 4 6 2 8
4 8 6 2 5 9 3 1 7
7 1 9 4 2 5 8 3 6
6 4 3 1 8 7 2 5 9
5 2 8 3 9 6 1 7 4
8 7 2 5 6 3 9 4 1
3 6 5 9 4 1 7 8 2
1 9 4 8 7 2 5 6 3
```

Παράδειγμα εξόδου

```
correct
```

Άσκηση 7.23

Στον διαγωνισμό PISA, που έγινε τον Μάρτιο του 2012, συμμετείχαν 49 σχολεία. Ο αριθμός συμμετοχής για κάθε σχολείο ήταν 80 μαθητές στη γραπτή εξέταση και 35 μαθητές στην εξέταση στον Ηλεκτρονικό Υπολογιστή. Τα ονόματα των σχολείων καταχωρίζονται σε έναν μονοδιάστατο πίνακα με το όνομα schools και οι βαθμολογίες σε δύο δισδιάστατους πίνακες που είναι παράλληλοι με τον πίνακα schools. Στον πρώτο πίνακα με το όνομα grapta (49 γραμμές, 80 στήλες) καταχωρίζονται τα αποτελέσματα της γραπτής εξέτασης. Στον δεύτερο πίνακα με το όνομα electro (49 γραμμές, 35 στήλες) καταχωρίζονται τα αποτελέσματα της εξέτασης στον υπολογιστή για κάθε σχολείο.

Να δημιουργήσετε πρόγραμμα, το οποίο να:

(α) ζητά από τον χρήστη τα ονόματα των 49 σχολείων και να τα καταχωρίζει στον πίνακα schools, καθώς επίσης και τις βαθμολογίες των μαθητών και να τις καταχωρίζει στους πίνακες grapta και electro. Να θεωρήσετε ότι όλα τα στοιχεία δίνονται σωστά και δεν χρειάζεται οποιοσδήποτε έλεγχος

(β) υπολογίζει τους μέσους όρους των βαθμολογιών που πήρε το κάθε σχολείο τόσο στη γραπτή όσο και στην εξέταση στον Η/Υ και να τους καταχωρίζει σε μονοδιάστατους παράλληλους πίνακες, με τα ονόματα mesosg και mesose, αντίστοιχα

(γ) βρίσκει και να τυπώνει το σχολείο με το μεγαλύτερο άθροισμα των 2 μέσων όρων των εξετάσεων (να θεωρήσετε ότι δεν θα υπάρξει ισοβαθμία)

(δ) υπολογίζει και να τυπώνει πόσοι μαθητές κάθε σχολείου έχουν πάρει βαθμολογία μεγαλύτερη από τον μέσο όρο του σχολείου τους στην εξέταση στους Η/Υ.

(Παγκύπριες Εξετάσεις 2012)

Άσκηση 7.24

Στην παγκόσμια Ολυμπιάδα Πληροφορικής συμμετέχουν 10 χώρες, μεταξύ των οποίων και η Κύπρος. Κάθε χώρα συμμετέχει με 3 διαγωνιζόμενους. Τα ονόματα των 10 χωρών θα καταχωριστούν σε έναν μονοδιάστατο πίνακα με το όνομα country και οι βαθμοί των διαγωνιζομένων της κάθε χώρας θα καταχωριστούν σε έναν άλλο δισδιάστατο πίνακα πραγματικών αριθμών, ο οποίος έχει το όνομα results και είναι παράλληλος με τον πρώτο

πίνακα. Ο πίνακας results έχει τέσσερις στήλες, από τις οποίες οι τρεις πρώτες αντιστοιχούν με τις βαθμολογίες των διαγωνιζομένων της κάθε χώρας και η τέταρτη με τις συνολικές τους βαθμολογίες, οι οποίες θα υπολογίζονται από το πρόγραμμα.

Να δημιουργήσετε πρόγραμμα, το οποίο να:

(α) ζητά από τον χρήστη το όνομα της χώρας και τις βαθμολογίες που πήραν οι διαγωνιζόμενοί της και να τοποθετεί τις πληροφορίες αυτές στις κατάλληλες θέσεις των δύο πινάκων

(β) βρίσκει και να παρουσιάζει την υψηλότερη βαθμολογία που επιτεύχθηκε μεταξύ όλων των διαγωνιζομένων

(γ) υπολογίζει τη συνολική βαθμολογία των διαγωνιζομένων κάθε χώρας (αθροίζοντας τις τρεις βαθμολογίες) και να την τοποθετεί στην κατάλληλη θέση του πίνακα results

(δ) υπολογίζει και να παρουσιάζει το πλήθος των χωρών που πέτυχαν συνολική βαθμολογία μεγαλύτερη από 250

(ε) βρίσκει και να παρουσιάζει τη συνολική βαθμολογία που πέτυχε η Κύπρος (να γίνει αναζήτηση με το όνομα CYPRUS).

(Παγκύπριες Εξετάσεις 2005)

Άσκηση 7.25

Το παγκόσμιο πρωτάθλημα ποδοσφαίρου 2018 έχει ανατεθεί, μετά από ψηφοφορία της εκτελεστικής επιτροπής της FIFA, στη Ρωσία. Μέσα στο πλαίσιο διοργάνωσης του μεγάλου αυτού γεγονότος, η κυβέρνηση της χώρας έχει προκηρύξει διαγωνισμό για αναβάθμιση των υφιστάμενων αθλητικών εγκαταστάσεων αλλά και για τη δημιουργία νέων. Για την υλοποίηση του έργου η διοργανώτρια χώρα ζήτησε από κάθε αρχιτεκτονικό γραφείο τέσσερις (4) διαφορετικές αρχιτεκτονικές προτάσεις. Στο διαγωνισμό αυτό έλαβαν μέρος τριάντα (30) αρχιτεκτονικά γραφεία.

Να δημιουργήσετε πρόγραμμα, το οποίο να:

(α) καταχωρίζει στον μονοδιάστατο πίνακα names το όνομα του κάθε αρχιτεκτονικού γραφείου και στον παράλληλο δισδιάστατο πίνακα vathm τις βαθμολογίες (από 1 μέχρι 10) που έχει πάρει το κάθε αρχιτεκτονικό γραφείο για τις τέσσερις (4) διαφορετικές προτάσεις, με τις οποίες έλαβε μέρος. Να θεωρήσετε ότι κανένα αρχιτεκτονικό γραφείο δεν έχει δύο ή περισσότερες προτάσεις με την ίδια βαθμολογία και όλα τα στοιχεία δίνονται σωστά και δεν χρειάζεται οποιοσδήποτε έλεγχος

names

DesignArch
BuildArch
ModernArch
.
.
FFArch
ArtArch

vathm

6	8	5	9
3	4	8	7
5	9	4	10
.	.	.	.
.	.	.	.
5	10	6	7
9	8	4	5

Η **1η γραμμή** του πίνακα **vathm** αντιπροσωπεύει τις βαθμολογίες του αρχιτεκτονικού γραφείου **DesignArch** για τις τέσσερις (4) αρχιτεκτονικές προτάσεις που έχει προτείνει. π.χ. η βαθμολογία της 1ης αρχιτεκτονικής πρότασης είναι 6, της 2ης 8, της 3ης 5 και της 4ης 9.

(β) υπολογίζει και να παρουσιάζει το πλήθος των αρχιτεκτονικών γραφείων που έχουν πάρει σε μία πρότασή τους βαθμολογία 10

(γ) βρίσκει και να καταχωρίζει σε έναν παράλληλο δισδιάστατο πίνακα 30 γραμμών και 2 στηλών, με το όνομα protasi, τον αριθμό της αρχιτεκτονικής πρότασης που αντιστοιχεί στην υψηλότερη βαθμολογία για κάθε αρχιτεκτονικό γραφείο (στην 1η στήλη) και τη βαθμολογία που πήρε σε αυτή την πρόταση (στη 2η στήλη). Για παράδειγμα, για το αρχιτεκτονικό γραφείο BuildArch, στο κελί protasi[1][0] καταχωρίζει τον αριθμό 3 και στο κελί protasi[1][1] καταχωρίζει τη βαθμολογία 8

(δ) χρησιμοποιεί τη συνάρτηση pmax5, η οποία να λαμβάνει ως παράμετρο από την κύρια συνάρτηση (main) τον πίνακα protasi, για να υπολογίζει και να επιστρέφει στην κύρια συνάρτηση (main) το πλήθος των αρχιτεκτονικών γραφείων που η υψηλότερη τους βαθμολογία είναι κάτω από 5. Το πλήθος αυτό να τυπώνεται στην κύρια συνάρτηση (main) του προγράμματος.

(Παγκύπριες Εξετάσεις 2016)

Άσκηση 7.26

Το χωράφι με τις λεμονιές του Παντελή έχει προσβληθεί από την ασθένεια «Ψευδόκοκκος». Όταν μία λεμονιά προσβληθεί, τα γειτονικά δέντρα μολύνονται και αυτά. Ο Παντελής έχει σημειώσει τις σειρές και τα μολυσμένα δέντρα της κάθε σειράς. Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται τις διαστάσεις του χωραφιού και τις σειρές που έχουν προσβληθεί από την ασθένεια και να επιστρέφει πόσες λεμονιές μέσα στο χωράφι έχουν μείνει απρόσβλητες από την ασθένεια.

Δεδομένα εισόδου

- Στην πρώτη γραμμή θα εμφανίζονται τρεις ακέραιοι: N, το μήκος του χωραφιού, M, το πλάτος του χωραφιού (2<=N,M<=100) και K (1<=K<=50), οι σειρές που έχουν προσβληθεί από την ασθένεια.
- Σε κάθε μία από τις επόμενες K γραμμές θα εμφανίζονται:
  - Ένας χαρακτήρας (R ή C), αν πρόκειται για γραμμή (R) ή στήλη (C).
  - Ένας ακέραιος Z (0<=Z<N,M), ο αριθμός της γραμμής ή της στήλης.
  - Δύο ακέραιοι a και b (0<=a, b<N,M) που υποδεικνύουν ότι οι λεμονιές από το a μέχρι το b, συμπεριλαμβανομένων, έχουν προσβληθεί από την ασθένεια.

Να σημειωθεί ότι κάποια από τα δεδομένα μπορεί να επικαλύπτονται. Δηλαδή ο Παντελής μπορεί να έχει σημειώσει δύο φορές το ίδιο δέντρο ως μολυσμένο.

Δεδομένα εξόδου

Ένας ακέραιος αριθμός, οι λεμονιές του χωραφιού που δεν έχουν μολυνθεί από την ασθένεια.

Παράδειγμα εισόδου

```
5 7 4
R 0 1 4
R 3 0 6
C 5 1 3
C 6 0 4
```

Παράδειγμα εξόδου

```
18
```

Επεξήγηση: Ο πιο κάτω πίνακας παρουσιάζει το χωράφι του Παντελή. Με 'Χ' σημειώνονται τα μολυσμένα δέντρα.

	X	X	X	X		X
					X	X
					X	X
X	X	X	X	X	X	X
						X

Άσκηση 7.27

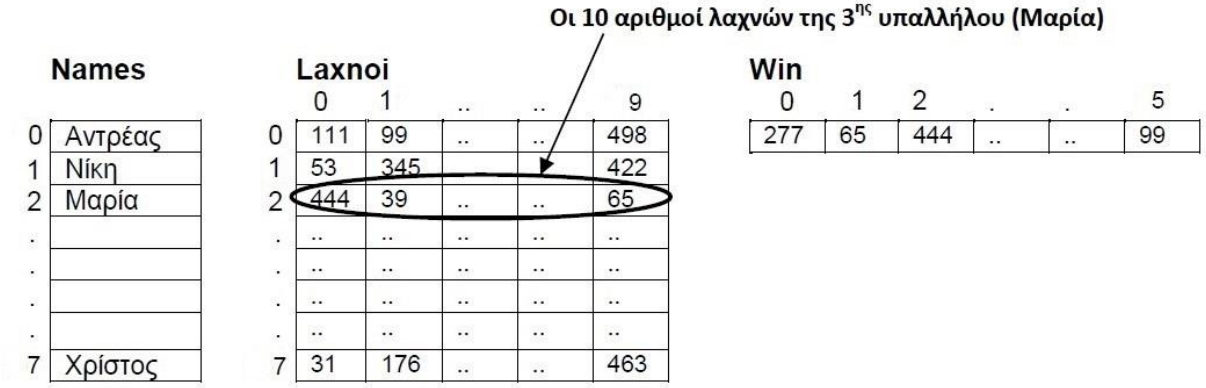
Μια εταιρεία απασχολεί 8 υπαλλήλους. Κάθε Πάσχα εκδίδει 80 λαχνούς αριθμημένους από το 1 μέχρι και το 500 και δίνει 10 λαχνούς στον/στην κάθε υπάλληλο. Μετά από κλήρωση, 6 τυχεροί λαχνοί κερδίζουν από 100 ευρώ ο κάθε ένας.

Να δημιουργήσετε πρόγραμμα, το οποίο να:

(α) καταχωρίζει στον μονοδιάστατο πίνακα Names τα ονόματα των υπαλλήλων και στον παράλληλο δισδιάστατο πίνακα Laxnoi τους αριθμούς των λαχνών που πήρε ο/η κάθε υπάλληλος. Να θεωρήσετε ότι όλα τα στοιχεία δίνονται σωστά και δεν χρειάζεται οποιοσδήποτε έλεγχος

(β) καταχωρίζει στον μονοδιάστατο πίνακα Win τους 6 τυχερούς αριθμούς. Να γίνεται έλεγχος ότι οι τυχεροί αριθμοί δίνονται σωστά (1 – 500). Σε διαφορετική περίπτωση, να παρουσιάζεται στην οθόνη το μήνυμα «Λάθος λαχνός» και να ζητείται ξανά αριθμός τυχερού λαχνού

Για παράδειγμα, αν στην 3η θέση του πίνακα Names καταχωρισθεί το όνομα της υπαλλήλου Μαρίας, τότε στην 3η γραμμή του πίνακα Laxnoi καταχωρίζονται οι αριθμοί των λαχνών που πήρε η υπάλληλος αυτή. Επίσης, βλέπουμε ότι η Μαρία κερδίζει το ελάχιστο 200 ευρώ (100 ευρώ με τον λαχνό 444 και 100 ευρώ με τον λαχνό 65)



(γ) ζητά τον αριθμό ενός λαχνού και, στη συνέχεια, να βρίσκει και να τυπώνει το όνομα του/της κατόχου του. Αν ο λαχνός δεν βρεθεί, να εμφανίζει το μήνυμα «Ο λαχνός δεν βρέθηκε»

(δ) υπολογίζει το συνολικό ποσό που κέρδισε ο/η κάθε υπάλληλος και να το καταχωρίζει στον μονοδιάστατο πίνακα Totals, ο οποίος είναι παράλληλος με τους πίνακες Names και Laxnoi

(ε) ταξινομεί με τη χρήση αλγόριθμου φυσαλίδας, σε φθίνουσα σειρά, τους πίνακες Totals και Names με βάση τον πίνακα Totals και να εμφανίζει στην οθόνη τα στοιχεία των δύο πινάκων με δεξιά στοίχιση 20 χαρακτήρων.

Το πρόγραμμα πρέπει να εμφανίζει στην οθόνη τα κατάλληλα μηνύματα για την εισαγωγή των δεδομένων και την εξαγωγή των αποτελεσμάτων.

(Παγκύπριες Εξετάσεις 2014)

Άσκηση 7.28

Σε ένα μεταπτυχιακό πρόγραμμα, 20 φοιτητές διδάσκονται 10 διαφορετικά μαθήματα. Στο τέλος του προγράμματος, παίρνουν βαθμό για το κάθε μάθημα. Τα ονόματα των 20 φοιτητών καταχωρίζονται σε έναν μονοδιάστατο πίνακα με το όνομα onoma. Σε έναν άλλο παράλληλο πίνακα δύο διαστάσεων με 20 γραμμές και 11 στήλες, με το όνομα vathmoi, καταχωρίζονται στις πρώτες 10 στήλες οι βαθμοί (δεκαδικοί αριθμοί) που πήρε ο κάθε φοιτητής στο κάθε ένα από τα 10 μαθήματα.

Να δημιουργήσετε πρόγραμμα, το οποίο:

(α) i. Να καταχωρίζει στον μονοδιάστατο πίνακα onoma το όνομα του κάθε φοιτητή και στις 10 πρώτες στήλες του παράλληλου δισδιάστατου πίνακα vathmoi, τον βαθμό (0.0 – 10.0) που έχει πάρει σε κάθε μάθημα. Να γίνεται έλεγχος αν ο βαθμός έχει δοθεί εκτός ορίων. Σε αυτή την περίπτωση να παρουσιάζεται στην οθόνη το μήνυμα «Wrong Input» και να ζητείται να δοθεί ξανά ο σωστός βαθμός

ii. Να υπολογίζει και να καταχωρίζει στην 11η στήλη του πίνακα vathmoi τον γενικό βαθμό (μέσος όρος βαθμολογίας) του κάθε φοιτητή

Παράδειγμα:

	onoma
0	Ανδρέας Χ.
1	Κώστας Ν.
2	Μαρία Κ.
	. .
	Γιάννης Κ.
19	Άννα Ζ.

	vathmoi						
	0	1	2	3	...		10
0	9.1	8.3	10.0	9.0	...	10.0	9.83
1	2.5	9.1	5.0	8.0	...	7.8	6.89
2	5.2	4.0	7.7	6.5	...	9.0	8.46
	. .	. .	. .	. .	. .	. .	. .
	3.7	7.5	4.0	9.3	...	4.7	7.1
19	1.0	2.8	4.1	6.0	...	3.8	4.88

← π.χ. στην **1η γραμμή** του πίνακα **vathmoi,** στις **πρώτες 10 θέσεις** είναι οι βαθμολογίες του φοιτητή **Ανδρέα Χ.** για τα 10 του μαθήματα.
Στην **11η** θέση είναι ο γενικός βαθμός του **Ανδρέα Χ.**

(β) Σε έναν άλλο δισδιάστατο πίνακα (20 γραμμών και 2 στηλών) με το όνομα results, παράλληλο με τους άλλους δύο, να υπολογίζει και να καταχωρίζει στην πρώτη στήλη τον αριθμό των μαθημάτων που πέρασε ο φοιτητής (βαθμός >= 5.0) και στη δεύτερη στήλη, τον αριθμό των μαθημάτων που δεν πέρασε (βαθμός < 5.0)

(γ) Να βρίσκει και να τυπώνει τα ονόματα των φοιτητών που πήραν την πιο υψηλή βαθμολογία στο πρώτο μάθημα (μπορεί να υπάρχουν περισσότεροι του ενός)

(δ) Να χρησιμοποιεί τη συνάρτηση found, η οποία θα λαμβάνει ως παραμέτρους από την κύρια συνάρτηση (main) τους πίνακες onoma και results. Ακολούθως, να εντοπίζει και να επιστρέφει (χρησιμοποιώντας τις κατάλληλες παραμέτρους αναφοράς) στην κύρια συνάρτηση (main) το όνομα του φοιτητή που δεν πέρασε τα περισσότερα μαθήματα, καθώς και τον αριθμό των μαθημάτων που δεν πέρασε. Τα

δύο αυτά στοιχεία να τυπώνονται στην κύρια συνάρτηση (main). Να θεωρήσετε ότι μόνο ένας φοιτητής δεν πέρασε τα περισσότερα μαθήματα σε σχέση με τους υπόλοιπους

(ε) Να βρίσκει και να τυπώνει τον αριθμό του μαθήματος με τους περισσότερους αριστεύσαντες φοιτητές (με βαθμό μεγαλύτερο ή ίσο του 8.5), καθώς και το πλήθος των φοιτητών που αρίστευσαν. Να θεωρήσετε ότι υπάρχει μόνο ένα τέτοιο μάθημα.

Το πρόγραμμα πρέπει να εμφανίζει στην οθόνη τα κατάλληλα μηνύματα για την εισαγωγή των δεδομένων και την εξαγωγή των αποτελεσμάτων.

(Παγκύπριες Εξετάσεις 2017)

Ασκήσεις Εμπλουτισμού

Άσκηση 7.29

«Το Οικόπεδο 12, γνωστό και ως "Αφροδίτη", είναι μία διερευνητική άδεια γεώτρησης που βρίσκεται στα ανοικτά της νότιας ακτής της Κύπρου και στη θαλάσσια Αποκλειστική Οικονομική Ζώνη (ΑΟΖ) της χώρας. Βρίσκεται 34 χιλιόμετρα δυτικά του κοιτάσματος φυσικού αερίου Λεβιάθαν, του μεγαλύτερου κοιτάσματος φυσικού αερίου που ανακαλύφθηκε την τελευταία δεκαετία».

Έχουμε έναν χάρτη διαστάσεων N x N του πιο πάνω οικοπέδου. Κάθε σημείο του χάρτη εμφανίζεται με ένα ψηφίο d (1<=d<=9), που αντιστοιχεί στο βάθος του συγκεκριμένου σημείου σε χιλιάδες πόδια. Η πλατφόρμα «Όμηρος» έχει ξεκινήσει να διερευνά πιθανά σημεία εξόρυξης. Για να θεωρηθεί κάποιο σημείο σαν πιθανό σημείο εξόρυξης, πρέπει να μην βρίσκεται στα όρια του χάρτη και να έχει μεγαλύτερο βάθος από όλα τα προσκείμενα σημεία (τέσσερα συνολικά σε Ανατολή, Δύση, Βορρά και Νότο). Προσκείμενο σημείο θεωρείται ένα σημείο με το οποίο το υπό διερεύνηση σημείο εφάπτεται σε μία πλευρά.

Να δημιουργήσετε πρόγραμμα, το οποίο να βρίσκει όλα τα πιθανά σημεία εξόρυξης και να τα σημειώνει στον χάρτη με τον χαρακτήρα 'X'.

Δεδομένα εισόδου

- Ένας ακέραιος αριθμός N (1<=N<=100) που αντιστοιχεί στις διαστάσεις N x N του χάρτη του οικοπέδου.
- Ακολουθούν N γραμμές με N ψηφία d (1<=d<=9) σε κάθε γραμμή, που αντιστοιχούν στο βάθος του κάθε σημείου.

Δεδομένα εξόδου

Θα εμφανίζονται N γραμμές με N χαρακτήρες σε κάθε γραμμή, όπου κάθε σημείο εξόρυξης θα έχει αντικατασταθεί με τον χαρακτήρα 'X'.

Παράδειγμα εισόδου

```
4
1112
1912
1472
1234
```

Παράδειγμα εξόδου

```
1112
1X12
14X2
1234
```

Άσκηση 7.30

Δύο ανθρακωρύχοι έχουν παγιδευτεί σε ένα ορυχείο μετά από μία κατολίσθηση. Για καλή τους τύχη έχουν ανακαλύψει ένα άθικτο αγωγό μεταφοράς υλικού, ο οποίος, με βάση τα σχέδια του ορυχείου, μπορεί να τους οδηγήσει στην έξοδο. Ο αγωγός όμως είναι πολύ μικρός για να χωρέσει και τους δύο. Συμφώνησαν, λοιπόν, ο ένας να μπει στον αγωγό και ο άλλος να του δίνει οδηγίες μέσω ασυρμάτου, για να βρει την έξοδο και να καλέσει βοήθεια.

Αν σας δίνονται οι διαστάσεις του ορυχείου, ο χάρτης όπου εμφανίζονται κατά σειρά: η θέση των ανθρακωρύχων, ο αγωγός και η έξοδος, να βρείτε τις οδηγίες που πρέπει να δώσει μέσω ασυρμάτου ο ένας ανθρακωρύχος στον άλλο, για να βρει την έξοδο. Οι οδηγίες να δοθούν με βάση τα τέσσερα σημεία του ορίζοντα που μπορεί να κινηθεί ο ανθρακωρύχος μέσα στον αγωγό. Να θεωρήσετε ότι πάντα θα υπάρχει έξοδος.

Δεδομένα εισόδου

- Δύο ακέραιοι αριθμοί N, M (1<=N, M<=50) που αντιστοιχούν στις διαστάσεις N x M του ορυχείου.
- Ο χάρτης του ορυχείου όπου:
  - '*' είναι η θέση των ανθρακωρύχων
  - 'X' είναι η έξοδος
  - '.' είναι ο αγωγός μεταφοράς
  - '#' είναι μη προσβάσιμο σημείο λόγω κατολίσθησης

Δεδομένα εξόδου

Οι οδηγίες (N, E, S, W) για να κινηθεί μέσα στον αγωγό ο ανθρακωρύχος μέχρι να βρει την έξοδο. N-Βορράς (πάνω), E-Ανατολή (δεξιά), S-Νότος (κάτω), W-Δύση (αριστερά).

Παράδειγμα εισόδου

```
4 4
*.##
#..#
##.#
X..#
```

Παράδειγμα εξόδου

```
ESESSWW
```

Άσκηση 7.31

Έστω ότι έχουμε μία σκακιέρα (8 x 8) και πάνω σε αυτή υπάρχουν N βασίλισσες και ένας βασιλιάς. Υπάρχει τουλάχιστον μία βασίλισσα που απειλεί τον βασιλιά. Να βοηθήσετε τον βασιλιά να βρει τον τρόπο διαφυγής (πάντα υπάρχει ένας τουλάχιστον τρόπος) από τις βασίλισσες. Ο βασιλιάς δικαιούται μόνο μία κίνηση. Σε περίπτωση που υπάρχουν

περισσότερες από μία θέσεις στις οποίες μπορεί να διαφύγει ο βασιλιάς, να εμφανίζονται όλες σε αύξουσα σειρά, με βάση τις γραμμές και μετά με βάση τις στήλες.

Κανόνες σκακιού

Ο βασιλιάς κινείται οριζόντια, κάθετα και διαγώνια, σε όποιο γειτονικό τετράγωνο δεν απειλείται από κομμάτι αντιπάλου.

Η βασίλισσα μπορεί να κινηθεί σε ευθεία γραμμή προς οποιαδήποτε κατεύθυνση, οριζόντια, κάθετα ή διαγώνια.

Δεδομένα εισόδου

- Γραμμή 1: Ένας αριθμός N (0<N<=42) που αντιστοιχεί στο πλήθος των βασιλισσών.
- Γραμμές 2…N+1: Ζευγάρια αριθμών X, Y (0<=X,Y<=7) που αντιστοιχούν στις θέσεις των βασιλισσών πάνω στη σκακιέρα.
- Γραμμή N+2: Ένα ζευγάρι αριθμών A, B (0<=A, B<=7) που αντιστοιχεί στη θέση του βασιλιά πάνω στη σκακιέρα.

Δεδομένα εξόδου

Οι θέσεις στις οποίες μπορεί να διαφύγει ο βασιλιάς. Η κάθε θέση είναι ένα ζεύγος αριθμών, το οποίο δηλώνει τη θέση που θα πρέπει να μετακινηθεί ο βασιλιάς. Τα αποτελέσματα παρουσιάζονται ταξινομημένα σε αύξουσα σειρά, με βάση τη γραμμή και μετά με βάση τη στήλη.

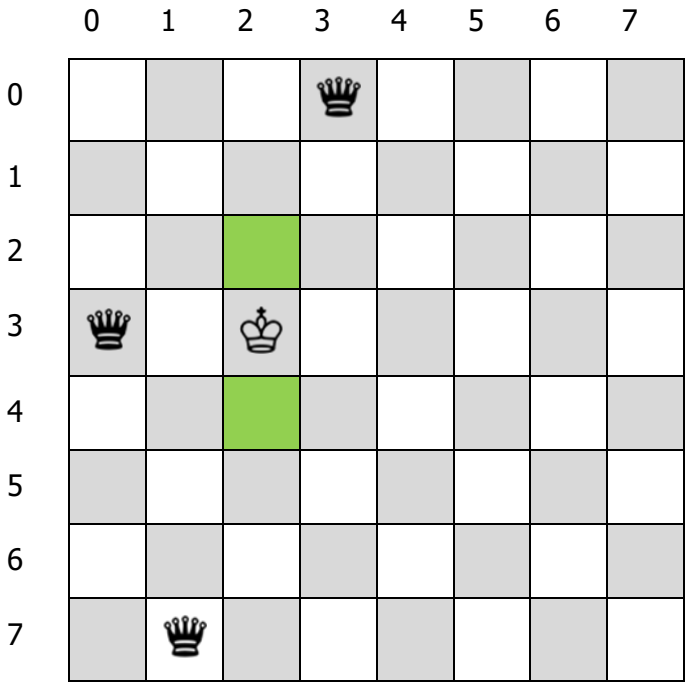
Παράδειγμα εισόδου

```
3
3 0
0 3
7 1
3 2
```

Παράδειγμα εξόδου

```
2 2
4 2
```

Επεξήγηση: Με πράσινο χρώμα εμφανίζονται οι θέσεις που μπορεί να διαφύγει ο βασιλιάς.



Άσκηση 7.32

Ο Ναρκαλιευτής (Minesweeper) είναι ένα απλό παιχνίδι μνήμης και λογικής και ένα από τα δημοφιλέστερα παιχνίδια όλων των εποχών.

Σκοπός του παιχνιδιού είναι να βρείτε τα κενά τετράγωνα και να αποφύγετε αυτά που κρύβουν νάρκες. Το παιχνίδι, όταν πατήσετε ένα κενό τετράγωνο, εμφανίζει έναν αριθμό που φανερώνει πόσες νάρκες πρόσκεινται σε αυτό (8 συνολικά, αν το επιτρέπουν τα όρια του πλέγματος).

Το πιο κάτω πλέγμα περιέχει δύο νάρκες, οι οποίες αντιπροσωπεύονται με τον χαρακτήρα '*'.

```
*...
....
.*..
....
```

Αν το ίδιο πλέγμα το παρουσιάσουμε λυμένο ως προς τις νάρκες, θα έχουμε το αποτέλεσμα που βλέπετε πιο κάτω:

```
*100
2210
1*10
1110
```

Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται τις διαστάσεις του πλέγματος (2<=N,M<=64) και τους χαρακτήρες που το αποτελούν. Τα άδεια τετράγωνα θα εμφανίζονται με τον χαρακτήρα '.', ενώ αυτά που περιέχουν νάρκες με τον χαρακτήρα '*'. Το πρόγραμμα να εμφανίζει το αρχικό πλέγμα, όπου οι χαρακτήρες '.' θα έχουν αντικατασταθεί με το πλήθος των ναρκών που πρόσκεινται στο τετράγωνο αυτό. Αν δεν υπάρχει προσκείμενη νάρκη, θα εμφανίζεται ο αριθμός 0.

Παράδειγμα εισόδου

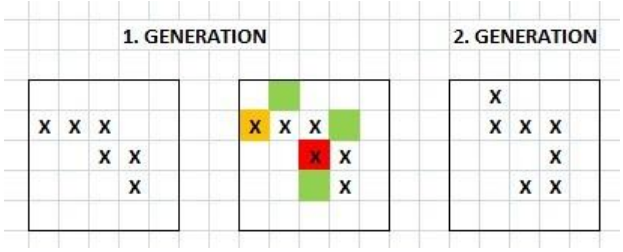
```
4 5
.....
.*.*.
.*.*.
.....
```

Παράδειγμα εξόδου

```
11211
2*4*2
2*4*2
11211
```

Άσκηση 7.33

Το παιχνίδι της ζωής αποτελείται από μία συλλογή (γενιά) κυττάρων τα οποία συμβολίζονται με X στο πλέγμα. Κάθε κύτταρο γειτονεύει με τα οκτώ κελιά που πρόσκεινται σε αυτό:



Υπάρχουν τέσσερις κανόνες που καθορίζουν το πότε γεννιέται ένα νέο μέλος (στα άδεια κελιά) και πότε πεθαίνει ή αναπαράγεται κάποιο ζωντανό:

1. Κάθε ζωντανό κελί με έναν ή κανέναν ζωντανό γείτονα θα πεθάνει στην επόμενη γενιά, ως απομονωμένο (κίτρινο).
2. Κάθε ζωντανό κελί με δύο ή τρεις ζωντανούς γείτονες θα παραμένει ζωντανό και στην επόμενη γενιά.
3. Κάθε ζωντανό κελί με τέσσερις ή περισσότερους γείτονες θα πεθάνει στην επόμενη γενιά, λόγω ανταγωνισμού-υπερπληθυσμού (κόκκινο).
4. Αν υπάρχει ένα κενό κελί με ακριβώς τρεις γείτονες, τότε στην επόμενη γενιά θα γεννιέται εκεί ένα νέο μέλος (πράσινο).

Δεδομένα εισόδου

- Μία γραμμή με τους ακεραίους N (2<=N<=100) και K (1<=K<=10). Όπου N είναι οι διαστάσεις N x N του πλέγματος, ενώ K είναι ο αριθμός των επόμενων γενεών που πρέπει να εμφανίζει το πρόγραμμα.
- Ένας πίνακας N x N με τους χαρακτήρες 'X' και '.'. Όπου 'X' ένα ζωντανό κύτταρο.

Δεδομένα εξόδου

K πίνακες διαστάσεων N x N που παρουσιάζουν τις επόμενες γενιές.

Παράδειγμα εισόδου

```
3 2
. . X
X . X
. X X
```

Παράδειγμα εξόδου

```
GENERATION 1
. X .
. . X
. X X

GENERATION 2
. . .
. . X
. X X
```

Γ7.8 Εγγραφές (Structures)

Τι θα μάθουμε σε αυτό το κεφάλαιο:

- Να αναφέρουμε τι είναι η εγγραφή (structure) και ποια τα βασικά χαρακτηριστικά της
- Να αναγνωρίζουμε προβλήματα που μπορούν να επιλυθούν με χρήση εγγραφών
- Να εντοπίζουμε ποιες εγγραφές (και ποια πεδία πρέπει να έχει η κάθε μία) χρειάζονται με βάση τις ανάγκες του αλγόριθμου/προγράμματος
- Να δίνουμε κατάλληλο όνομα (αναγνωριστικό) σε έναν τύπο εγγραφής και να καθορίζουμε τα αναγνωριστικά και τους τύπους δεδομένων των πεδίων της
- Να δηλώνουμε μεταβλητές τύπου εγγραφής και να τις αρχικοποιούμε όπου χρειάζεται
- Να αναφερόμαστε σε συγκεκριμένα μέλη/πεδία της εγγραφής (εγγραφή**.**πεδίο)
- Να χρησιμοποιούμε σε πρόγραμμα εγγραφές ως παραμέτρους σε συναρτήσεις
- Να δηλώνουμε πίνακες όπου τα στοιχεία είναι τύπου εγγραφής και διαχειρίζονται τα στοιχεία του πίνακα
- Να συγκρίνουμε πίνακες και εγγραφές, καθώς και παράλληλους πίνακες με πίνακες από εγγραφές ως προς τις ομοιότητες και τις διαφορές τους
- Να μελετούμε ένα πρόβλημα και να αποφασίζουμε κατά πόσο επιλύεται με τη χρήση εγγραφών ή πινάκων από εγγραφές
- Να καθορίζουμε το πρόβλημα με ακρίβεια, συγκεκριμένα: να εντοπίζουμε/διακρίνουμε τα Δεδομένα, τις Πληροφορίες και την Επεξεργασία
- Να αποφασίζουμε εάν χρειάζονται εγγραφές ή πίνακες από εγγραφές για την επίλυση του προβλήματος και να τις καθορίζουμε
- Να σχεδιάζουμε τον τρόπο επίλυσης του προβλήματος
- Να επιλέγουμε κατάλληλες δομές (επανάληψης ή και διακλάδωσης), ανάλογα με τις δυνατότητες, τους περιορισμούς και τα χαρακτηριστικά τους για επίλυση του προβλήματος
- Να υλοποιούμε τον σχεδιασμό τους σε πρόγραμμα με τη χρήση του προγραμματιστικού περιβάλλοντος, ώστε να επιλυθεί το πρόβλημα
- Να επιλέγουμε κατάλληλα δεδομένα και στρατηγική για έλεγχο του προγράμματος
- Να ελέγχουμε την ορθότητα της λύσης του προβλήματος, χρησιμοποιώντας τη μέθοδο της προκαταρκτικής εκτέλεσης ή και άλλες μεθόδους για επαλήθευση
- Να μελετούμε έτοιμο πρόγραμμα το οποίο περιλαμβάνει εγγραφές ή πίνακες από εγγραφές και να εντοπίζουμε βασικά μέρη του, τα οποία συνδέονται με πτυχές του προβλήματος που επιλύει
- Να προσθέτουμε επεξηγηματικά σχόλια σε ένα πρόγραμμα
- Να συμπληρώνουμε ένα έτοιμο πρόγραμμα με εγγραφές ή πίνακες από εγγραφές και κατάλληλες δομές και εντολές, ώστε να αποτελεί λύση ενός διαφοροποιημένου προβλήματος
- Να δημιουργούμε/τροποποιούμε προγράμματα για ταξινόμηση και αναζήτηση, ώστε να χρησιμοποιούμε πίνακα με εγγραφές και η ταξινόμηση/αναζήτηση να γίνεται με βάση συγκεκριμένο πεδίο των εγγραφών, χρησιμοποιώντας αλγόριθμους που ήδη μελετήθηκαν
- Να εντοπίζουμε και να αναγνωρίζουμε σε ένα πρόγραμμα πρότυπα σχεδίασης και στρατηγικές (design patterns, τμήματα κώδικα)
- Να χρησιμοποιούμε πρότυπα σχεδίασης και στρατηγικές που εντοπίσαμε ως εργαλεία επίλυσης προβλημάτων σε νέα προγράμματά μας.

1. Εισαγωγή

Σε αυτό το κεφάλαιο θα ασχοληθούμε με τις εγγραφές (structures). Εγγραφές ονομάζουμε συλλογές στοιχείων, οι οποίες μας δίνουν τη δυνατότητα να ομαδοποιήσουμε στοιχεία διαφορετικών τύπων. Για παράδειγμα, θα χρησιμοποιήσουμε εγγραφές όταν θέλουμε να αποθηκεύσουμε πληροφορίες για τους μαθητές ενός σχολείου, όπου ο κάθε μαθητής θα μας δίνει: αριθμό μητρώου (ακέραιος), όνομα (συμβολοσειρά), επίθετο (συμβολοσειρά), διεύθυνση (συμβολοσειρά) και ηλικία (ακέραιος). Τα στοιχεία που ανήκουν σε μία εγγραφή καλούνται μέλη. Σε αντίθεση με τους πίνακες, κάθε μέλος της εγγραφής μπορεί να έχει το δικό του αναγνωριστικό. Τα μέλη μπορούν να ανήκουν στους βασικούς τύπους δεδομένων (`integer`, `float`, `double`, `boolean`, `char`), να είναι συμβολοσειρές (`string`), πίνακες ή ακόμη και άλλες εγγραφές.

2. Ορισμός εγγραφής

Μία εγγραφή ορίζεται ως εξής:

(α) Ξεκινά με τη δεσμευμένη λέξη `struct` και ακολουθεί το αναγνωριστικό της εγγραφής.

(β) Τα άγκιστρα οριοθετούν τα μέλη της εγγραφής.

(γ) Το ερωτηματικό στο τέλος ορίζει το τέλος της εγγραφής.

```
struct <όνομα εγγραφής> {
 <τύπος 1ου μέλους> <όνομα 1ου μέλους>;
 <τύπος 2ου μέλους> <όνομα 2ου μέλους>;
 ...
 <τύπος Ν-οστού μέλους> <όνομα Ν-οστού μέλους>;
};
```

Αν δύο ή περισσότερα μέλη έχουν τον ίδιο τύπο, τότε μπορούμε να τα ορίσουμε μαζί:

```
<τύπος μέλους> <όνομα 1ου>, <όνομα 2ου>;
```

Για παράδειγμα, θέλουμε να αποθηκεύσουμε πληροφορίες για τα προϊόντα μίας υπεραγοράς. Για κάθε προϊόν έχουμε τις εξής πληροφορίες: κωδικός προϊόντος (ακέραιος), ποσότητα (ακέραιος), περιγραφή (συμβολοσειρά) και τιμή (πραγματικός). Η εγγραφή θα δηλωθεί όπως πιο κάτω, με το όνομα product:

```
struct product {            // Ξεκινούμε με τη δεσμευμένη λέξη struct
  int code, quantity;       // Πρώτο/Δεύτερο μέλος: Ακέραιοι
  string description;       // Τρίτο μέλος: Συμβολοσειρά
  double price;             // Τέταρτο μέλος: Πραγματικός αριθμός
};                          // Τέλος εγγραφής
```

3. Δήλωση μεταβλητών εγγραφής

Για να δηλώσουμε μεταβλητές τύπου εγγραφής, μπορούμε να κάνουμε τα εξής:

(α) Για μία μεταβλητή:

```
product apple;
```

(β) Για πλήθος μεταβλητών:

```
product apple, banana, orange;
```

(γ) Για δήλωση εγγραφής και μεταβλητών ταυτόχρονα:

```
struct product {
  int code, quantity;
  string description;
  double price;
} apple, banana, orange;
```

4. Απόδοση αρχικών τιμών εγγραφής

Η απόδοση αρχικών τιμών στα μέλη μίας εγγραφής γίνεται ως εξής:

```
struct product {
  int code, quantity;
  string description;
  double price;
} apple, banana, orange;

int main() {
 apple  = {101, 50, "ambrosia", 1.99};
 banana = {102, 40, "chiquita", 2.70};
 orange = {103, "clementine", 1.60};     // ΣΦΑΛΜΑ. Λιγότερα μέλη
 orange = {103,60,"clementine",1.60,0}; // ΣΦΑΛΜΑ. Περισσότερα μέλη
return 0;
}
```

Θα πρέπει να δοθεί προσοχή όταν ορίζουμε αρχικές τιμές, καθώς το πλήθος των τιμών μέσα στα άγκιστρα πρέπει να αντιστοιχεί με το πλήθος των μελών της εγγραφής. Σε αντίθεση με τους πίνακες, στις εγγραφές αν ορίσουμε λιγότερες ή περισσότερες τιμές, θα έχουμε σφάλμα. Η σειρά με την οποία ορίζουμε τις αρχικές τιμές πρέπει να είναι η ίδια με την σειρά των μελών στη δήλωση της εγγραφής.

5. Αναφορά στα μέλη μίας εγγραφής

Για αναφορά στα μέλη μίας εγγραφής χρησιμοποιούμε την πιο κάτω γενική εντολή, όπου μετά το όνομα της εγγραφής ακολουθεί μία τελεία (.) και το όνομα του μέλους της εγγραφής στο οποίο αναφερόμαστε:

```
<όνομα μεταβλητής εγγραφής>.<όνομα μέλους εγγραφής>
```

Έτσι, όταν θέλουμε να διαβάσουμε την τιμή ενός μέλους μίας εγγραφής:

```
 cin >> apple.code;
 cin >> apple.quantity;
 cin >> apple.description;
 cin >> apple.price;
```

Ενώ αν θέλουμε να εκχωρήσουμε τιμές:

```
 orange.code = 103;
 orange.quantity = 60;
 orange.description = "clementine";
 orange.price = 1.60;
```

Ένα μέλος της εγγραφής μπορεί να είναι πίνακας. Αν θέλουμε, για παράδειγμα, στην εγγραφή για έναν μαθητή να μπορούμε να αποθηκεύσουμε τους βαθμούς τεσσάρων διαγωνισμάτων, θα το κάνουμε ως εξής:

```
struct student {            // Ορισμός εγγραφής student
  int student_number;
  string student_name;
  double test[4];           // Μέλος: πίνακας τεσσάρων στοιχείων
};
```

Αν υπάρχει μέλος της εγγραφής, το οποίο είναι πίνακας, η εκχώρηση τιμών και η απόδοση αρχικών τιμών μπορεί να γίνει ως εξής:

```
struct student {            // Ορισμός εγγραφής student
  int student_number;
  string student_name;
  double test[4];           // Μέλος: πίνακας τεσσάρων στοιχείων
};

int main(){
  // Μαθητής 1: με εκχώρηση τιμών
  student st1;
  st1.student_number = 1001;
  st1.student_name = "Jones";
  st1.test[0]=13; st1.test[1]=15; st1.test[2]=16; st1.test[3]=15;

  // Μαθητής 2: με απόδοση αρχικών τιμών (αρχικοποίηση)
  student st2 = { 1002, "Vick", {12, 14, 16, 19} };
return 0;
}
```

6. Πίνακες εγγραφών

Αν θέλουμε να αποθηκεύσουμε περισσότερα από ένα στοιχεία, μπορούμε να ορίσουμε έναν πίνακα εγγραφών. Για παράδειγμα, αν θέλουμε να αποθηκεύσουμε τις πληροφορίες τριών προϊόντων, θα το κάνουμε ως εξής:

```
struct product {                // Ορισμός εγγραφής product
   int code, quantity;          // Μέλη εγγραφής τύπου integer
   string description;          // Μέλος εγγραφής τύπου string
   double price;                // Μέλος εγγραφής τύπου double
} fruit[3];      // Δέσμευση μνήμης για τρεις εγγραφές τύπου product
                 // σε πίνακα εγγραφών τριών θέσεων
```

Αν θέλουμε, μαζί με τη δήλωση του πίνακα, να ορίσουμε και τις αρχικές τιμές στα μέλη των στοιχείων του, εφόσον γνωρίζουμε εκ των προτέρων τις τιμές τους, μπορεί να γίνει ως εξής:

```
product fruit[3] = {  {101, 50, "ambrosia", 1.99 },
                      {102, 40, "chiquita", 2.70 },
                      {103, 60, "clementine", 1.60 }  };
```

Αν επιθυμούμε η καταχώριση των στοιχείων του πίνακα εγγραφών να γίνει από τον χρήστη μέσω πληκτρολογίου, μπορούμε να χρησιμοποιήσουμε δομή επανάληψης. Η αναφορά στα στοιχεία του πίνακα, όπως και στις μεταβλητές τύπου εγγραφής, γίνεται με τη χρήση μίας τελείας μεταξύ του στοιχείου του πίνακα και του μέλους του ως ακολούθως:

```
for (int i=0; i<3; i++){
      cin >> fruits[i].code >> fruits[i].quantity;
      cin >> fruits[i].description >> fruits[i].price;
}
```

Παράδειγμα 8.1

Να δημιουργήσετε πρόγραμμα, το οποίο να ορίζει μία εγγραφή με το όνομα student, η οποία να περιέχει τα ακόλουθα μέλη:

- Αριθμός μητρώου (integer)
- Επώνυμο (string)
- Τηλέφωνο (integer)
- Εκτοπισμένος (boolean) – θα δοθεί η τιμή 1 (true) ή η τιμή 0 (false)

Να ορίσετε έναν πίνακα εγγραφών 20 θέσεων τύπου student, με το όνομα B3. Το πρόγραμμα να δέχεται τα στοιχεία για τους 20 μαθητές από το πληκτρολόγιο.

```
#include <iostream>
#include <string>
using namespace std;

struct student {            // Ορισμός εγγραφής student
      int am;               // Μέλη εγγραφής
      string name;
      int tel;
      bool ekt;
};                          // Τέλος εγγραφής

int main(){                 // Κύρια συνάρτηση (main)
      student B3[20];       // Δήλωση πίνακα εγγραφών 20 θέσεων
      for (int i=0; i<20; i++)
            cin >> B3[i].am >> B3[i].name >> B3[i].tel >> B3[i].ekt;
return 0;
}
```

(Code: C7_8_example1.cpp)

Παράδειγμα 8.2

Να δημιουργήσετε πρόγραμμα, το οποίο να ορίζει μία εγγραφή με το όνομα shape, η οποία να περιέχει τα μέλη height (double) και width (double). Το πρόγραμμα να διαβάζει και να αποθηκεύει τις διαστάσεις δέκα ορθογωνίων και να παρουσιάζει στην οθόνη το εμβαδόν και την περίμετρο για κάθε ένα από τα ορθογώνια.

```
#include <iostream>
using namespace std;

struct shape {              // Ορισμός εγγραφής shape
      double height;
      double width;
} rectangle[10];            // Πίνακας εγγραφών τύπου shape

int main(){
  for (int i=0; i<10; i++)       // Καταχώριση στοιχείων
      cin >> rectangle[i].height >> rectangle[i].width;
  for (int i=0; i<10; i++){      // Υπολογισμός εμβαδών-περιμέτρων
      cout << "Rectangle " << i+1 << endl;
      cout << "E: " << rectangle[i].height*rectangle[i].width << " ";
      cout << "P: " << (rectangle[i].height+rectangle[i].width)*2;
      cout << endl << endl;
 }
return 0;
}
```

(Code: C7_8_example2.cpp)

7. Ένθετες εγγραφές

Μία εγγραφή μπορεί να χρησιμοποιηθεί ως μέλος άλλης εγγραφής. Η πρόσβαση στα μέλη της ένθετης εγγραφής γίνεται ιεραρχικά, με τη χρήση επιπρόσθετης τελείας, όπως θα δείτε στο πιο κάτω παράδειγμα. Μπορούμε να ορίσουμε όσες ένθετες εγγραφές επιθυμούμε και δεν υπάρχει περιορισμός στο βάθος ένθεσης που μπορούμε να φτάσουμε.

Παράδειγμα 8.3

```
#include<iostream>
using namespace std;

struct employee {               // Δηλώνεται πρώτα η ένθετη εγγραφή
      int id;
      double salary;
};
```

```
struct company {                    // Ακολούθως δηλώνεται η εγγραφή
      employee manager;             // που περιέχει την ένθετη εγγραφή
      int years;
};

int main(){
  company abc;
  abc.manager.id = 778899;          // Ιεραρχική ανάθεση τιμής
  abc.manager.salary = 80000;       // με τη χρήση 2ης τελείας
  abc.years= 12;
return 0;
}
```

(Code: C7_8_example3.cpp)

Παράδειγμα 8.4

Να δημιουργήσετε πρόγραμμα, το οποίο να ορίζει μία εγγραφή με το όνομα member, η οποία να έχει ως μέλη τον κωδικό (integer), το email (string) και μία ένθετη δομή (date), με μέλη την ημέρα (integer), τον μήνα (integer) και τη χρονολογία γεννήσεως (integer). Το πρόγραμμα να δέχεται τα δεδομένα για 50 άτομα και να εμφανίζει το email όσων έχουν γεννηθεί στις 5 Σεπτεμβρίου του 2005.

```
#include <iostream>
#include <string>
using namespace std;

struct date {          // Ορισμός εγγραφής date
   int dd;
   int mm;
   int yy;
 };

struct member {        // Ορισμός εγγραφής member
   int code;
   string email;
   date dob;           // Ένθετη εγγραφή τύπου date
} mem[50];             // Δήλωση πίνακα εγγραφών member

int main(){
  for (int i=0; i<50; i++){// Καταχώριση στοιχείων
      cin >> mem[i].code >> mem[i].email;
      cin >> mem[i].dob.dd >> mem[i].dob.mm >> mem[i].dob.yy;
 }
```

```
for (int i=0; i<50; i++)          // Έλεγχος ημερομηνίας 5/9/2005
    if (mem[i].dob.dd==5 && mem[i].dob.mm==9 && mem[i].dob.yy==2005)
           cout << mem[i].email << endl;
return 0;
}
```

(Code: C7_8_example4.cpp)

8. Εγγραφές και συναρτήσεις

Οι εγγραφές μπορούν να δοθούν παραμετρικά σε συναρτήσεις όπως και οι μεταβλητές. Μία εγγραφή μπορεί να δοθεί ως παράμετρος αναφοράς, αλλά και ως παράμετρος τιμής.

Παράδειγμα 8.5

```
#include <iostream>
#include <string>
using namespace std;
struct book {               // Ορισμός εγγραφής book
  string title;
  string author;
  string isbn;
  int quantity;
};
// Καταχώριση στοιχείων
void readBook(book &bk) { // Παράμετρος αναφοράς (&)
 getline(cin, bk.title);  // Η getline() διαβάζει ολόκληρη τη γραμμή
 getline(cin, bk.author); // π.χ. Edgar Allan Poe
 getline(cin, bk.isbn);
 cin >> bk.quantity;
}
// Εμφάνιση στοιχείων
void printBook(book bk) {  // Παράμετρος τιμής
 cout << "Book title : "    << bk.title << endl;
 cout << "Book author : "   << bk.author << endl;
 cout << "Book isbn : "     << bk.isbn << endl;
 cout << "Book quantity : " << bk.quantity << endl;
}
int main( ){
  book book1;          // Δήλωση μεταβλητής τύπου εγγραφής book
  readBook(book1);     // Κλήση συνάρτησης readBook (καταχώριση)
  printBook(book1);    // Κλήση συνάρτησης printBook (εμφάνιση)
return 0;
}
```

(Code: C7_8_example5.cpp)

Παράδειγμα 8.6

Έστω ότι διαχειρίζεστε μία ιστοσελίδα και θέλετε να υπολογίσετε τα έσοδα από τις διαφημίσεις. Να ορίσετε μία εγγραφή (`advertising`) που να κρατά δεδομένα για το πλήθος των διαφημίσεων (`integer`), το ποσοστό διαφημίσεων που έχει επιλεγεί για προβολή (`double`) και το ποσό που κερδίζετε από κάθε διαφήμιση που προβλήθηκε (`double`). Να δημιουργήσετε πρόγραμμα, το οποίο να διαβάζει τα δεδομένα για τα μέλη της εγγραφής και να καλεί μία συνάρτηση (`calculate`), η οποία να δέχεται παραμετρικά μία εγγραφή advertising και να επιστρέφει τα συνολικά κέρδη από τις διαφημίσεις, που θα εμφανίζονται με δύο δεκαδικά ψηφία.

Παράδειγμα εισόδου

```
60 25.0 2.50
```

Παράδειγμα εξόδου

```
37.50
```

```
#include <iostream>
#include <iomanip>
using namespace std;

struct advertising {        // Ορισμός εγγραφής advertising
  int total;
  double pct;
  double amount;
};

// Υπολογισμός κερδών
double calculate(advertising ad){      // Παράμετρος τιμής
       return (ad.total * ad.pct / 100 * ad.amount);
}

int main(){

  advertising ads;          // Δήλωση μεταβλητής τύπου εγγραφής ads
  cin >> ads.total;
  cin >> ads.pct;
  cin >> ads.amount;
  cout << fixed << setprecision(2) << calculate(ads) << endl;

return 0;
}
```

(Code: C7_8_example6.cpp)

9. Αναζήτηση και ταξινόμηση εγγραφών

Είδαμε ότι με τη χρήση εγγραφών μπορούμε εύκολα να ομαδοποιήσουμε στοιχεία ανόμοιων τύπων. Το γεγονός αυτό καθιστά ευκολότερο το έργο της αναζήτησης συγκεκριμένων στοιχείων ή της ταξινόμησης αυτών σε σειρά. Ειδικότερα στην ταξινόμηση, η χρήση εγγραφών αντί πολλαπλών παράλληλων πινάκων μειώνει κατά πολύ την έκταση του κώδικα. Ας δούμε παραδείγματα:

Παράδειγμα 8.7

Το Υπουργείο Παιδείας και Πολιτισμού έχει καταγεγραμμένα τα στοιχεία (κωδικός, όνομα, τηλέφωνο, επαρχία) όλων των σχολείων της Μέσης Εκπαίδευσης της Κύπρου, στο αρχείο schools.txt. Στο αρχείο αυτό υπάρχουν το πολύ 200 σχολεία. Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται τον κωδικό ενός σχολείου και να τυπώνει στην οθόνη, σε μία γραμμή, τα στοιχεία του. Σε περίπτωση που ο κωδικός του σχολείου δεν υπάρχει στο αρχείο, τότε να τυπώνει το μήνυμα «School not found».

```
#include<iostream>
#include<fstream>
#include<string>
using namespace std;

struct school {                  // Ορισμός εγγραφής school
  int code;
  string name;
  int tel;
  string city;
};

// Συνάρτηση σειριακής αναζήτησης
int find_school(school sch[], int key, int N){
    for (int i=0; i<N; i++)
           if (key == sch[i].code)
                return i;
return -1;
}

int main(){
  school sch[200];               // Πίνακας 200 εγγραφών
  int target, cnt = 0;
  ifstream fin("schools.txt");   // Ορισμός αρχείου εισόδου
  while(!fin.eof()){             // Καταχώριση στοιχείων εγγραφής
    fin >> sch[cnt].code;
    fin >> sch[cnt].name;
    fin >> sch[cnt].tel;
```

```
    fin >> sch[cnt].city;
    cnt++;                          // Επόμενη εγγραφή
 }

  cin >> target;                    // Κωδικός προς αναζήτηση
  int pos = find_school(sch, target, cnt);
  if(pos == -1)                     // Το σχολείο δεν βρέθηκε
     cout << "School not found" << endl;
  else{                             // Εμφάνιση στοιχείων
     cout << sch[pos].code << " " << sch[pos].name << " ";
     cout << sch[pos].tel << " " << sch[pos].city << endl;
 }
fin.close();                        // Κλείσιμο αρχείου
return 0;
}
```

(Code: C7_8_example7.cpp)

Παράδειγμα 8.8

Να δημιουργήσετε πρόγραμμα, το οποίο να ορίζει μία εγγραφή με το όνομα student, η οποία να περιέχει τα εξής μέλη: όνομα (sting), βαθμός Α' τετράμηνου (integer), βαθμός Β' τετράμηνου (integer) και τελικός βαθμός (integer). Ο τελικός βαθμός είναι ο μέσος όρος των δύο βαθμών, στρογγυλοποιημένος προς τα πάνω (π.χ. αν έχουμε βαθμούς τετραμήνων 16 και 17, ο τελικός βαθμός θα είναι 17).

Το πρόγραμμα να διαβάζει δεδομένα από το αρχείο students.txt (όνομα, βαθμός Α, βαθμός Β) για 20 μαθητές, να υπολογίζει τον τελικό βαθμό για κάθε μαθητή και να εμφανίζει στο αρχείο results.txt το όνομα και τον τελικό βαθμό, με δεξιά στοίχιση δέκα χαρακτήρων, ταξινομημένους σε φθίνουσα σειρά με βάση τον τελικό βαθμό.

```
#include <fstream>
#include <string>
#include <iomanip>          // Για τη χρήση της setw
#include <cmath>            // Για τη χρήση της round
using namespace std;

#define N 20                // Σταθερά για το πλήθος των μαθητών

struct student {            // Ορισμός εγγραφής student
   string name;
   int t1;
   int t2;
   int F;
} A3[N];                    // Δήλωση πίνακα εγγραφών εκτός
                            // της κύριας συνάρτησης
```

```
// Συνάρτηση ταξινόμησης εγγραφών με εισαγωγή. Η συνάρτηση δεν έχει
// παράμετρους αφού ο πίνακας εγγραφών έχει δηλωθεί εκτός της κύριας
// συνάρτησης και είναι προσβάσιμος από τη συνάρτηση ταξινόμησης

 void sort_struct( ) {
   student temp;      // Μεταβλητή τύπου εγγραφής
   int j;
   for (int i=1; i<N; i++){
            temp = A3[i];
            j = i - 1;
            while (j>=0 && A3[j].F<temp.F){
                A3[j+1] = A3[j];
                j--;
             }
        A3[j+1] = temp;
       }
    }

int main(){

 ifstream fin("students.txt");          // Δημιουργία ροής ανάγνωσης
 ofstream fout("results.txt");          // Δημιουργία ροής εγγραφής
 for (int i=0; i<N; i++){               // Καταχώριση / Υπολογισμός
     fin >> A3[i].name >> A3[i].t1 >> A3[i].t2;
     A3[i].F = round((A3[i].t1 + A3[i].t2) / 2.0);
 }

 sort_struct( );                        // Ταξινόμηση εγγραφών
 for (int i=0; i<N; i++)                // Εμφάνιση στοιχείων
      fout << setw(10) << A3[i].name << setw(10) << A3[i].F<<endl;

 fin.close();                           // Κλείσιμο αρχείων
 fout.close();

return 0;
}
```

(Code: C7_8_example8.cpp)

Ασκήσεις Κεφαλαίου - https://www.hackerrank.com/g78

Άσκηση 8.1

Να συμπληρώσετε τον πιο κάτω κώδικα, ο οποίος θα δηλώνει μία εγγραφή με το όνομα product, με τα πιο κάτω μέλη:

- Κωδικός (ακέραιος)
- Περιγραφή (συμβολοσειρά)
- Τιμή (πραγματικός)
- Διαθέσιμο (λογική μεταβλητή)

```
struct product {




};
```

Άσκηση 8.2

Να συμπληρώσετε τον πιο κάτω κώδικα, ο οποίος θα δηλώνει μία εγγραφή με το όνομα vehicle, με τα πιο κάτω μέλη. Στο τέλος της εγγραφής να δηλωθεί ένας πίνακας εγγραφών 100 στοιχείων, με το όνομα cars:

- Αριθμός πινακίδας (συμβολοσειρά)
- Ένδειξη χιλιομετρητή (πραγματικός)
- Τιμή (πραγματικός)
- Μοντέλο (συμβολοσειρά)

```
struct vehicle {




}
```

Άσκηση 8.3

Να συμπληρώσετε τον πιο κάτω κώδικα, ο οποίος θα δηλώνει μία εγγραφή με το όνομα account, με τα πιο κάτω μέλη:

- Αριθμός λογαριασμού (συμβολοσειρά)
- Διαθέσιμο ποσό (πραγματικός)
- Αριθμός ταυτότητας πελάτη (ακέραιος)
- Κατάθεση - Ένθετη εγγραφή τύπου last_transaction με τα εξής μέλη: κωδικός συναλλαγής (ακέραιος), ημερομηνία συναλλαγής (συμβολοσειρά)

```
struct last_transaction {



};

struct account {




};
```

Άσκηση 8.4

Να υλοποιήσετε μία εγγραφή με το όνομα student, η οποία θα περιέχει τα εξής μέλη:

- id (ακέραιος)
- first_name (συμβολοσειρά)
- last_name (συμβολοσειρά)
- grade (ακέραιος)

Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται τα στοιχεία από το πληκτρολόγιο, για έναν μόνο μαθητή, με τη σειρά που εμφανίζονται πιο πάνω, να τα αποθηκεύει σε μία μεταβλητή εγγραφής και να τα εμφανίζει σε μία γραμμή.

Παράδειγμα εισόδου

```
1001
John
Adams
15
```

Παράδειγμα εξόδου

```
1001 John Adams 15
```

Άσκηση 8.5

Να υλοποιήσετε μία εγγραφή με το όνομα employee, η οποία να περιέχει τα πιο κάτω μέλη:

- Κωδικός υπαλλήλου (ακέραιος)
- Όνομα υπαλλήλου (συμβολοσειρά)
- Επίθετο υπαλλήλου (συμβολοσειρά)
- Μισθός (πραγματικός)

Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται στοιχεία από το πληκτρολόγιο για 10 υπαλλήλους με τη σειρά που εμφανίζονται πιο πάνω, να τα αποθηκεύει σε πίνακα εγγραφών και να εμφανίζει τα στοιχεία κάθε υπαλλήλου σε ξεχωριστή γραμμή.

Άσκηση 8.6

Να υλοποιήσετε μία εγγραφή με το όνομα schools, με τα εξής μέλη:

- Κωδικός σχολείου (ακέραιος)
- Όνομα σχολείου (συμβολοσειρά)
- Πόλη (συμβολοσειρά)
- Τηλέφωνο (ακέραιος)

Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται στοιχεία από το πληκτρολόγιο, για 20 σχολεία, με τη σειρά που εμφανίζονται πιο πάνω και να τα αποθηκεύει σε πίνακα εγγραφών. Ακολούθως, να εμφανίζει τα στοιχεία για όλα τα σχολεία της Λευκωσίας (Lefkosia ή Nicosia), σε ξεχωριστή γραμμή για κάθε σχολείο.

Άσκηση 8.7

Να υλοποιήσετε μία εγγραφή με το όνομα members, με τα εξής μέλη:

- Κωδικός μέλους (ακέραιος)
- Όνομα μέλους (συμβολοσειρά)
- Τηλέφωνο (ακέραιος)
- Συναλλαγή (Ένθετη εγγραφή με δύο μέλη: κωδικός συναλλαγής (ακέραιος) και ποσό συναλλαγής (πραγματικός)

Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται στοιχεία από το πληκτρολόγιο για 5 μέλη, με τη σειρά που εμφανίζονται πιο πάνω και να τα αποθηκεύει σε πίνακα εγγραφών. Ακολούθως, να εμφανίζει τους κωδικούς των μελών με συναλλαγές άνω των 100€.

Άσκηση 8.8

Να υλοποιήσετε μία εγγραφή με το όνομα projects, με τα εξής μέλη:

- Κωδικός έργου (ακέραιος)
- Όνομα έργου (συμβολοσειρά)
- Ημερομηνία παράδοσης (συμβολοσειρά)
- Προϋπολογισμός (πραγματικός)

Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται στοιχεία για 15 έργα από το πληκτρολόγιο, με τη σειρά που εμφανίζονται πιο πάνω, να τα αποθηκεύει σε πίνακα εγγραφών και, ακολούθως, να εμφανίζει τα στοιχεία για τα έργα, με προϋπολογισμό μεταξύ 10,000.00, και 20,000.00, συμπεριλαμβανομένων, σε ξεχωριστή γραμμή για κάθε έργο.

Άσκηση 8.9

Να υλοποιήσετε μία εγγραφή με το όνομα books, με τα εξής μέλη:

- Κωδικός βιβλίου (ακέραιος)
- Όνομα βιβλίου (συμβολοσειρά)
- Επίθετο συγγραφέα (συμβολοσειρά)
- Έτος έκδοσης (ακέραιος)

Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται στοιχεία για 10 βιβλία από το πληκτρολόγιο και να τα αποθηκεύει σε πίνακα εγγραφών. Το πρόγραμμα να εμφανίζει τα στοιχεία των βιβλίων που έχουν γραφτεί από τον Tolkien και έχουν εκδοθεί μετά το 2010.

Άσκηση 8.10

Να υλοποιήσετε μία εγγραφή με το όνομα phones, με τα εξής μέλη:

- Κωδικός τηλεφώνου (ακέραιος)
- Μοντέλο τηλεφώνου (συμβολοσειρά)
- Τιμή (πραγματικός)
- Έτος κυκλοφορίας (ακέραιος)

Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται τα στοιχεία 20 τηλεφώνων από το πληκτρολόγιο και να τα αποθηκεύει σε πίνακα εγγραφών. Το πρόγραμμα θα πρέπει, ακολούθως, να:

(α) εμφανίζει στην οθόνη τα στοιχεία των τηλεφώνων τα οποία έχουν κυκλοφορήσει μεταξύ των ετών 2014 και 2016, συμπεριλαμβανομένων

(β) εμφανίζει στην οθόνη τα στοιχεία των τηλεφώνων των οποίων η τιμή είναι μεγαλύτερη από 500€ και το όνομα του μοντέλου ξεκινά από 'S'.

Άσκηση 8.11

Να βρείτε τέσσερα λάθη στο πιο κάτω πρόγραμμα.

```
#include<iostream>
using namespace std;

struct company {
  int code;
  double budget;
  int employees;
}

int main(){

  company soft1 = {101, 20000.00, 123, 15};
  company soft2;
  soft2 code = 35;
  cin >> company.code;

return 0;
}
```

Άσκηση 8.12

Στο πλαίσιο της συμφωνίας με την εταιρεία Microsoft, παρέχεται το δικαίωμα σε όλους τους μαθητές της δημόσιας εκπαίδευσης να χρησιμοποιούν δωρεάν το λογισμικό office365 σε πέντε συσκευές προσωπικών ηλεκτρονικών υπολογιστών και σε πέντε έξυπνες συσκευές τους. Για να μπορούν οι μαθητές να κάνουν χρήση των αδειών αυτών, θα πρέπει να δημιουργηθούν λογαριασμοί πρόσβασης στη συγκεκριμένη πλατφόρμα. Για κάθε μαθητή πρέπει να καταχωρίζονται τα εξής στοιχεία:

- Κωδικός σχολείου (Π.χ. 1234)
- Όνομα σχολείου (Π.χ. Palouriotissa)
- Όνομα χρήστη / Email μαθητή (Π.χ. student001@moec.ac.cy)
- Κωδικός χρήστη (Π.χ. PS!!1234_001)

Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται το πλήθος (N) των μαθητών, να δηλώνει μία εγγραφή με το όνομα user και να αποθηκεύει τα πιο πάνω στοιχεία σε πίνακα εγγραφών. Επιπρόσθετα, να δέχεται το όνομα ενός χρήστη και να τον αναζητά στον κατάλογο. Αν το όνομα του χρήστη βρεθεί, να εμφανίζει στην οθόνη όλα τα στοιχεία του, αλλιώς να εμφανίζει το μήνυμα «Not found». Να θεωρήσετε ότι δεν θα δοθούν στοιχεία για περισσότερους από 300 μαθητές.

Παράδειγμα εισόδου

```
3
1234
Palouriotissa
student001@moec.ac.cy
PS!!1234_001
1235
Palouriotissa
student002@moec.ac.cy
PS!!1234_002
1236
Palouriotissa
student003@moec.ac.cy
PS!!1234_003
student001@moec.ac.cy
```

Παράδειγμα εξόδου

```
School Code: 1234
School Name: Palouriotissa
User Name: student001@moec.ac.cy
User Password: PS!!1234_001
```

Άσκηση 8.13

Σας δίνεται το αρχείο players.txt με τις εξής πληροφορίες: κωδικός, επίθετο, ημερομηνία γέννησης και ποσοστό ευστοχίας. Πιο κάτω βλέπετε δείγμα του αρχείου:

```
101     James     11/09/1985     53.17
102     Scott     12/10/1983     17.14
103     Sherman   28/11/1979     21.89
105     Brady     06/04/1977     34.67
105     Jackson   09/08/1979     24.87
106     Manning   07/09/1976     37.75
108     Curry     12/10/1980     51.34
...
```

Να δημιουργήσετε πρόγραμμα, το οποίο να:

(α) δηλώνει μία εγγραφή με το όνομα players, η οποία θα περιέχει τα κατάλληλα μέλη για να αποθηκεύσει τις πληροφορίες που εμφανίζονται στο αρχείο

(β) διαβάζει όλα τα στοιχεία από το αρχείο players.txt και να τα αποθηκεύει σε πίνακα εγγραφών. Να θεωρήσετε ότι ο μέγιστος αριθμός παικτών που μπορεί να περιέχει το αρχείο είναι 100

(γ) εμφανίζει στην οθόνη τα ονόματα των παικτών με ποσοστό ευστοχίας πάνω από 50.

Άσκηση 8.14

Σε μία υπεραγορά τα προϊόντα καταχωρίζονται σε μία βάση δεδομένων. Συγκεκριμένα, καταχωρίζονται τα εξής στοιχεία:

- Κωδικός προϊόντος (ακέραιος)
- Ποσότητα (ακέραιος)
- Τιμή (πραγματικός)
- Περιγραφή (συμβολοσειρά)

Να δημιουργήσετε πρόγραμμα, το οποίο να:

(α) ζητά από τον χρήστη έναν ακέραιο αριθμό N (1<=N<=50), το πλήθος των προϊόντων

(α) δηλώνει μία εγγραφή με το όνομα product και να αποθηκεύει τα στοιχεία των προϊόντων σε πίνακα εγγραφών

(β) χρησιμοποιεί μία συνάρτηση με το όνομα calculate, η οποία να δέχεται ως παράμετρο μία μεταβλητή τύπου product και να υπολογίζει το κόστος για ένα προϊόν της υπεραγοράς (κόστος = ποσότητα x τιμή)

(γ) εμφανίζει το συνολικό κόστος για όλα τα προϊόντα της υπεραγοράς, με δύο δεκαδικά ψηφία.

Παράδειγμα εισόδου

```
2
101
2
2.99
banana
102
1
1.99
apple
```

Παράδειγμα εξόδου

```
7.97
```

Επεξήγηση: 2 * 2.99 + 1 * 1.99 = 7.97

Άσκηση 8.15

Να δημιουργήσετε πρόγραμμα, το οποίο να:

(α) δηλώνει μία εγγραφή με το όνομα fraction, η οποία θα περιέχει τα εξής μέλη:

- Αριθμητής (ακέραιος)
- Παρονομαστής (ακέραιος)

(β) ορίζει δύο μεταβλητές τύπου fraction για τις οποίες το πρόγραμμα θα δέχεται τα δεδομένα εισόδου από το πληκτρολόγιο

(γ) χρησιμοποιεί μία συνάρτηση με το όνομα multiply, η οποία να δέχεται ως παραμέτρους δύο μεταβλητές τύπου fraction, να υπολογίζει και να εμφανίζει το γινόμενο των δύο κλασμάτων με τρία δεκαδικά ψηφία.

Παράδειγμα εισόδου

```
1 4
1 2
```

Παράδειγμα εξόδου

```
0.125
```

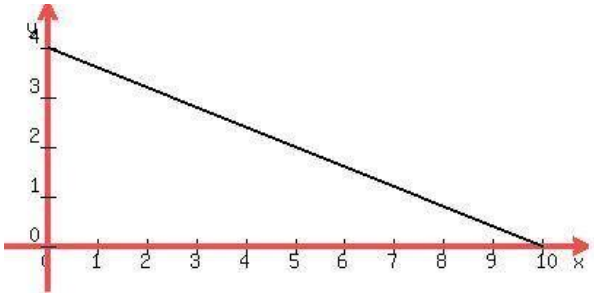
Επεξήγηση: (1/4) * (1/2) = (1/8) = 0.125

Άσκηση 8.16

Για να υπολογίσουμε το μήκος μίας ευθείας σε ένα πλέγμα, εφαρμόζουμε τον εξής τύπο:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Δίδονται δύο σημεία τα οποία δείχνουν πού ξεκινά (x1, y1) και πού καταλήγει (x2, y2) η ευθεία, όπως βλέπουμε στην πιο κάτω εικόνα:



Χρησιμοποιώντας τον τύπο, μπορούμε να υπολογίσουμε ότι:

```
d = sqrt ((10-0)² + (0-4)²)  = 10.77
```

Να δημιουργήσετε πρόγραμμα, το οποίο να:

(α) υλοποιεί μία εγγραφή με το όνομα point, η οποία να έχει ως μέλη δύο ακέραιους αριθμούς (x, y)

(β) υλοποιεί μία εγγραφή με το όνομα line, η οποία να έχει ως μέλη δύο ένθετες δομές point, τα σημεία τα οποία ξεκινά (start) και καταλήγει η ευθεία (finish)

(γ) χρησιμοποιεί μία συνάρτηση με το όνομα calculate, η οποία να δέχεται ως παράμετρο μία μεταβλητή τύπου line και να επιστρέφει το μήκος της ευθείας, το οποίο θα εμφανίζεται με δύο δεκαδικά ψηφία.

Παράδειγμα εισόδου

```
0 4
10 0
```

Παράδειγμα εξόδου

```
10.77
```

Άσκηση 8.17

Να δημιουργήσετε πρόγραμμα, το οποίο να:

(α) υλοποιεί μία εγγραφή με το όνομα players με τα ακόλουθα μέλη:

- Αριθμός παίκτη (ακέραιος)
- Όνομα παίκτη (συμβολοσειρά)
- Σύνολο πόντων (πραγματικός)

(β) καταχωρίζει τα στοιχεία για 20 παίκτες, σε πίνακα εγγραφών τύπου players

(γ) χρησιμοποιεί μία συνάρτηση με όνομα sort_struct, η οποία να δέχεται παραμετρικά τον πίνακα εγγραφών και να τον ταξινομεί σε φθίνουσα σειρά, με βάση το σύνολο των πόντων

(δ) χρησιμοποιεί μία συνάρτηση με το όνομα print_struct, η οποία να δέχεται παραμετρικά τον πίνακα με τα στοιχεία των εγγραφών και να εμφανίζει τα ταξινομημένα στοιχεία των παικτών στην οθόνη, όπως πιο κάτω:

```
10 Brooks 132.5
6 Jones 123.7
8 Henry 120.2
...
```

Άσκηση 8.18

Να δημιουργήσετε πρόγραμμα, το οποίο να:

(α) υλοποιεί μία εγγραφή με το όνομα members με τα ακόλουθα μέλη:

- Κωδικός μέλους (ακέραιος)
- Όνομα μέλους (συμβολοσειρά)
- Ηλικία (ακέραιος)

(β) καταχωρίζει τα στοιχεία για 10 μέλη σε έναν πίνακα εγγραφών τύπου members

(γ) χρησιμοποιεί μία συνάρτηση με όνομα sort_members, η οποία να δέχεται παραμετρικά τον πίνακα εγγραφών με τα στοιχεία των μελών και να τον ταξινομεί σε αύξουσα σειρά, με βάση την ηλικία

(δ) χρησιμοποιεί μία συνάρτηση με το όνομα print_members, η οποία να δέχεται παραμετρικά τον πίνακα με τα στοιχεία των μελών και να εμφανίζει τα ταξινομημένα στοιχεία των μελών στην οθόνη, όπως πιο κάτω:

```
101 Adams 15
103 Bond 18
107 Bale 18
102 Kim 20
...
```

Άσκηση 8.19

Να τροποποιήσετε τη συνάρτηση sort_members της άσκησης **8.18**, ώστε αν δύο μέλη έχουν την ίδια ηλικία, τότε να εμφανίζεται πρώτα το μέλος του οποίου το όνομα προηγείται αλφαβητικά, όπως πιο κάτω:

```
101 Adams 15
107 Bale 18
103 Bond 18
102 Kim 20
```

Άσκηση 8.20

Να προσθέσετε μία συνάρτηση με όνομα edit στην άσκηση **8.18**, η οποία να δέχεται παραμετρικά τον πίνακα εγγραφών και δύο ακέραιες τιμές (κωδικός μέλους και νέα τιμή για την ηλικία). Η συνάρτηση να αναζητά τον κωδικό μέλους ανάμεσα στον πίνακα εγγραφών και αν τον εντοπίσει να αλλάζει την υφιστάμενη ηλικία της εγγραφής, με την νέα τιμή που δίνεται στη συνάρτηση παραμετρικά. Αν ο κωδικός μέλους δεν εντοπιστεί, να εμφανίζεται μήνυμα σφάλματος.

Π.χ. Αν δοθεί η εντολή edit (107, 28), τα στοιχεία που εμφανίζονται πιο πάνω θα μετατραπούν σε: `107 Bale 28`

Άσκηση 8.21

Αν σας δοθούν δύο ενδείξεις ενός χρονομέτρου, να βρείτε τον συνολικό χρόνο που έχει παρέλθει σε δευτερόλεπτα από την πρώτη μέχρι τη δεύτερη ένδειξη.

Να δημιουργήσετε πρόγραμμα, το οποίο να:

(α) υλοποιεί μία εγγραφή με το όνομα times, με μέλη τρεις ακέραιους αριθμούς (hours, minutes, seconds)

(β) υλοποιεί μία εγγραφή με το όνομα timer, με μέλη δύο ένθετες εγγραφές times, την αρχική ένδειξη (start) και την τελική ένδειξη (stop)

(γ) χρησιμοποιεί μία συνάρτηση με το όνομα calculate, η οποία να δέχεται ως παράμετρο μία μεταβλητή τύπου timer και να επιστρέφει τον συνολικό αριθμό δευτερολέπτων που έχουν περάσει από την αρχική μέχρι την τελική ένδειξη του χρονομέτρου.

Παράδειγμα εισόδου

```
1 20 50
1 21 10
```

Παράδειγμα εξόδου

```
20
```

Άσκηση 8.22

Για τους Ν υπαλλήλους μίας εταιρείας θέλουμε να καταχωρίζονται τα εξής στοιχεία:

- Επίθετο (συμβολοσειρά)
- Όνομα (συμβολοσειρά)
- Ηλικία (ακέραιος)

- Φύλο (χαρακτήρας: M/F)
- Μισθός των τελευταίων τριών ετών (πίνακας πραγματικών αριθμών με την τιμή στη θέση 0 να είναι η προ τριετίας)

Να δημιουργήσετε πρόγραμμα, το οποίο να:

(α) ζητά από τον χρήστη έναν ακέραιο αριθμό N (1<=N<=50), το πλήθος των υπαλλήλων

(β) δηλώνει μία εγγραφή με το όνομα employee και να αποθηκεύει τα στοιχεία των υπαλλήλων σε πίνακα εγγραφών

(γ) χρησιμοποιεί τη συνάρτηση average, η οποία να υπολογίζει και να εμφανίζει τον μέσο όρο των μισθών όλων των υπαλλήλων, για κάθε ένα από τα τρία τελευταία χρόνια (π.χ. τρεις τιμές για το 2013, 2014 και 2015)

(δ) χρησιμοποιεί την συνάρτηση max_name, η οποία να εμφανίζει το επίθετο και την ηλικία της γυναίκας, η οποία είχε τον μεγαλύτερο μισθό το τελευταίο έτος.

Παράδειγμα εισόδου

```
4
Jones James 43 M 40000.00 42000.00 45000.00
Riley Mary 46 F 50000.00 55000.00 58000.00
Peters John 53 M 76000.00 78000.00 81000.00
Miles Anna 48 F 56000.00 58000.00 62000.00
```

Παράδειγμα εξόδου

```
Year1: 55500.00 Year2: 58250.00 Year3: 61500.00
Miles 48
```

Άσκηση 8.23

Για τις N νέες μετοχές οι οποίες εισήλθαν το 2016 στο Χρηματιστήριο Αξιών Κύπρου (ΧΑΚ), θέλουμε να αποθηκεύονται τα ακόλουθα στοιχεία:

- Το όνομα της μετοχής (συμβολοσειρά).
- Τις τιμές κλεισίματος της μετοχής (πίνακας πραγματικών αριθμών) των τελευταίων επτά ημερών.
- Το πλήθος των συναλλαγών της μετοχής για την τρέχουσα ημέρα (ακέραιος).

Να δημιουργήσετε πρόγραμμα, το οποίο να:

(α) ζητά από τον χρήστη έναν ακέραιο αριθμό N (1<=N<=100), το πλήθος των μετοχών

(β) να δηλώνει μία εγγραφή με το όνομα stocks και να αποθηκεύει τα στοιχεία των μετοχών σε πίνακα εγγραφών

(γ) χρησιμοποιεί την συνάρτηση min_share, η οποία να εμφανίζει το όνομα και το πλήθος συναλλαγών της μετοχής με το μικρότερο πλήθος συναλλαγών

(δ) εμφανίζει το όνομα της μετοχής, της οποίας η τιμή κλεισίματος είχε τη μέγιστη αύξηση σε μία ημέρα της εβδομάδας, σε σχέση με την προηγούμενη ημέρα.

Παράδειγμα εισόδου

```
5
kronos 2.33 3.56 4.56 6.78 3.12 4.56 6.78 12
zeus 3.45 5.67 6.89 4.18 6.42 5.96 3.54 13
mars 3.43 5.56 6.78 3.12 4.45 6.45 4.65 14
pluto 1.43 13.56 4.36 4.72 2.15 4.16 7.71 15
venus 3.21 3.22 3.45 3.56 3.58 3.78 3.87 16
```

Παράδειγμα εξόδου

```
kronos 12
pluto
```

Άσκηση 8.24

Σε ένα συνεργείο αυτοκινήτων για κάθε αυτοκίνητο καταχωρίζονται τα εξής στοιχεία:

- Αριθμός πινακίδας (Π.χ. ΚΑΡ161)
- Κατασκευαστής (Π.χ. OPEL)
- Χρονολογία κατασκευής (Π.χ. 2003)
- Ημερομηνία επισκευής: η ημερομηνία θα είναι πάντα της μορφής dd/mm/yyyy (Π.χ. 12/11/2016 ή 09/05/2003)

Τα στοιχεία των αυτοκινήτων θα βρίσκονται στο αρχείο cars.txt. Κάθε γραμμή του αρχείου θα περιέχει τα στοιχεία ενός αυτοκινήτου ως εξής:

```
KAP161 OPEL 2003 12/11/2016
```

Να δημιουργήσετε πρόγραμμα, το οποίο να:

(α) δηλώνει μία εγγραφή με όνομα vehicles, η οποία να περιέχει τα κατάλληλα μέλη για να αποθηκεύει τις πληροφορίες που εμφανίζονται στο αρχείο

(β) διαβάζει τα στοιχεία των αυτοκινήτων από το αρχείο και να τα καταχωρίζει σε πίνακα εγγραφών. Να θεωρήσετε ότι ο μέγιστος αριθμός αυτοκινήτων που μπορεί να περιέχει το αρχείο είναι 100

(γ) καλεί μία συνάρτηση με όνομα year_total, η οποία να δέχεται παραμετρικά τη χρονολογία κατασκευής και να επιστρέφει το πλήθος των αυτοκινήτων που κατασκευάστηκαν τη συγκεκριμένη χρονιά

(δ) καλεί μία συνάρτηση με το όνομα month_end, η οποία να εμφανίζει στην οθόνη τα στοιχεία των αυτοκινήτων που επισκευάστηκαν στις 30 ή 31 οποιουδήποτε μήνα.

Παράδειγμα εισόδου (Στοιχεία: cars.txt και Χρονολογία: πληκτρολόγιο)

```
KAP161 OPEL 2003 12/11/2016
EEM330 MAZDA 2004 30/04/2015
LAF501 MERCEDES 2006 18/06/2005
KTM334 RENAULT 2007 31/08/2006
2006
```

Παράδειγμα εξόδου (οθόνη)

```
1
EEM330 MAZDA 2004 30/04/2015
KTM334 RENAULT 2007 31/08/2006
```

Άσκηση 8.25

Σε ένα γυμναστήριο, όταν κάνει εγγραφή ένα νέο μέλος, καταχωρίζονται για αυτόν/αυτή τα εξής στοιχεία:

- Κωδικός μέλους (Π.χ. 1719)
- Επίθετο μέλους (Π.χ. Jones)
- Όνομα μέλους (Π.χ. Tom)
- Χαρακτηριστικά: Τρεις τιμές για ηλικία, ύψος και βάρος (Π.χ. 42, 1.76, 83)

Να δημιουργήσετε πρόγραμμα, το οποίο να:

(α) δηλώνει μία εγγραφή με το όνομα gym_members, η οποία να περιέχει τα κατάλληλα μέλη για να αποθηκεύει τις πληροφορίες που εμφανίζονται πιο πάνω

(β) διαβάζει τα στοιχεία των μελών από το πληκτρολόγιο και να τα καταχωρίζει σε πίνακα εγγραφών. Για τα χαρακτηριστικά να χρησιμοποιήσετε μία ένθετη εγγραφή με τρία μέλη, με το όνομα chars. Το πρόγραμμα να δέχεται στοιχεία μέχρι να δοθεί ως κωδικός μέλους ο αριθμός μηδέν (0) ή να δοθούν στοιχεία για διακόσια (200) μέλη

(γ) καλεί μία συνάρτηση με το όνομα find_limits, η οποία να δέχεται παραμετρικά δύο ακέραιους αριθμούς (όριο ηλικίας και όριο βάρους) και να επιστρέφει το πλήθος των μελών που ξεπερνούν και τα δύο όρια. Π.χ. αν δοθούν οι τιμές (40, 80), η συνάρτηση θα επιστρέφει το πλήθος των μελών που έχουν ηλικία άνω των 40 ετών και βάρος άνω των 80 κιλών.

(δ) καλεί μία συνάρτηση με το όνομα sort_weight, η οποία να ταξινομεί τα μέλη σε αύξουσα σειρά, με βάση το βάρος τους

(ε) εμφανίζει το επίθετο, το ύψος και το βάρος όλων των μελών στην οθόνη.

Παράδειγμα εισόδου

```
1719 Jones Tom 42 1.76 83
1720 Franco James 39 1.85 86
1721 Jill Natalie 28 1.76 74
1722 Pitt Greg 31 1.91 89
1723 Ryan Nick 32 1.82 76
1724 Mason Matt 43 1.79 82
0
40 80
```

Παράδειγμα εξόδου

```
2
Jill 1.76 74
Ryan 1.82 76
Mason 1.79 82
Jones 1.76 83
Franco 1.85 86
Pitt 1.91 89
```

Ασκήσεις Εμπλουτισμού

Άσκηση 8.26

Για τους Ν πελάτες μίας τράπεζας θέλουμε να καταχωρίζονται τα εξής στοιχεία:

- Αριθμός ταυτότητας (ακέραιος)
- Όνομα (συμβολοσειρά)
- Επίθετο (συμβολοσειρά)
- Ημερομηνία γέννησης (συμβολοσειρά)
- Στοιχεία λογαριασμού: αριθμός λογαριασμού (ακέραιος) και υπόλοιπο λογαριασμού (πραγματικός).

Να δημιουργήσετε πρόγραμμα, το οποίο να:

(α) δηλώνει μία εγγραφή με το όνομα bank_cust, η οποία να περιέχει τα κατάλληλα μέλη για να αποθηκεύει τις πληροφορίες που εμφανίζονται πιο πάνω

(β) διαβάζει τα πιο πάνω στοιχεία από το αρχείο customers.txt και να τα καταχωρίζει σε πίνακα εγγραφών. Να θεωρήσετε ότι ο μέγιστος αριθμός πελατών που μπορεί να περιέχει το αρχείο είναι 100. Για τα στοιχεία λογαριασμού να γίνει χρήση ένθετης εγγραφής

(γ) χρησιμοποιεί τη συνάρτηση transaction, η οποία να δέχεται παραμετρικά τρεις τιμές: τον αριθμό λογαριασμού, έναν ακέραιο αριθμό Τ (0 ή 1) και έναν πραγματικό αριθμό V. Ο αριθμός Τ υποδεικνύει το είδος της συναλλαγής (0 – ανάληψη και 1 – κατάθεση), ενώ ο αριθμός V υποδεικνύει το ποσό της συναλλαγής. Να σημειωθεί πως για να γίνει ανάληψη, πρέπει να υπάρχει το διαθέσιμο ποσό. Η συνάρτηση να ενημερώνει το υπόλοιπο λογαριασμού κάθε πελάτη. Αν ο αριθμός λογαριασμού δεν εντοπιστεί, να εμφανίζεται μήνυμα σφάλματος

(δ) χρησιμοποιεί την συνάρτηση sort_balances, η οποία να ταξινομεί τους πελάτες σε φθίνουσα σειρά, με βάση το υπόλοιπο του λογαριασμού τους

(ε) εμφανίζει τους αριθμούς ταυτότητας και τα υπόλοιπα λογαριασμού όλων των πελατών στην οθόνη.

Παράδειγμα εισόδου (Πελάτες: customers.txt και Συναλλαγή: πληκτρολόγιο)

```
1234 James Jones 19/03/1985 101 456.75
1267 Mike Mason 23/10/1981 103 1098.45
1345 George Henman 05/05/1986 108 890.65
2345 William Beeman 03/02/1987 106 678.45
3456 Will Smith 02/05/1979 110 1134.78
110 1 500.00
```

Παράδειγμα εξόδου (οθόνη)

```
3456 1634.78
1267 1098.45
1345 890.65
2345 678.45
1234 456.75
```

Άσκηση 8.27

Ο Παντελής θέλει να δημιουργήσει πρόγραμμα για να διαχειρίζεται τις επαφές του στον υπολογιστή. Τα στοιχεία που επιθυμεί να διαχειρίζεται για κάθε μία από τις επαφές είναι τα εξής: όνομα επαφής (συμβολοσειρά) και αριθμός τηλεφώνου επαφής (ακέραιος). Το πρόγραμμα πρέπει να εμφανίζει στην οθόνη τις εξής επιλογές:

1. **Προσθήκη επαφής:** Με αυτή την επιλογή θα μπορεί να προσθέσει μία νέα επαφή.

2. **Αναζήτηση επαφής:** Με αυτή την επιλογή θα μπορεί να εμφανίσει τα στοιχεία μίας επαφής στην οθόνη. Η αναζήτηση να γίνεται με βάση τον αριθμό τηλεφώνου. Αν δεν υπάρχει η επαφή, να εμφανίζεται μήνυμα σφάλματος.

3. **Ταξινόμηση επαφών:** Με αυτή την επιλογή οι επαφές να ταξινομούνται αλφαβητικά με βάση τα ονόματα.

4. **Έξοδος:** Τερματισμός προγράμματος.

Να δημιουργήσετε πρόγραμμα, το οποίο να υλοποιεί το πιο πάνω. Το πρόγραμμα να μπορεί να αποθηκεύει στοιχεία για μέχρι 200 επαφές.

Σημείωση: Να δηλώσετε μία καθολική (public) μεταβλητή, η οποία να αποθηκεύει το πλήθος των επαφών. Αρχικά, η τιμή θα είναι μηδέν και με την προσθήκη μίας επαφής θα αυξάνεται κατά ένα.

Άσκηση 8.28

Ένα πανεπιστήμιο θέλει να υλοποιήσει ένα πρόγραμμα για την αποθήκευση και διαχείριση πληροφοριών σχετικά με τις βαθμολογίες των φοιτητών στα μαθήματα που εγγράφονται. Τα στοιχεία που επιθυμούν να διαχειρίζονται είναι τα εξής:

- Κωδικός φοιτητή (ακέραιος)
- Επώνυμο φοιτητή (συμβολοσειρά)
- Κωδικός μαθήματος (ακέραιος)
- Βαθμός (ακέραιος)

Οι λειτουργίες που θα υποστηρίζονται από το πρόγραμμα θα εμφανίζονται μέσω ενός βασικού μενού στην οθόνη του χρήστη και θα είναι οι εξής:

1. **Εισαγωγή νέας εγγραφής:** Ο χρήστης θα εισάγει τα στοιχεία μίας εγγραφής με την παραπάνω σειρά.

2. **Εύρεση βαθμολογίας:** Ο χρήστης θα εισάγει τον κωδικό του φοιτητή και τον κωδικό του μαθήματος. Στην οθόνη θα εμφανίζεται το επώνυμο του φοιτητή, ο κωδικός του, ο κωδικός του μαθήματος και ο βαθμός που έχει πάρει ο φοιτητής στο συγκεκριμένο μάθημα. Αν ο κωδικός του φοιτητή ή του μαθήματος δεν εντοπιστούν, θα εμφανίζεται μήνυμα σφάλματος.

3. **Αλλαγή στοιχείων:** Με αυτή την επιλογή θα ζητούνται από τον χρήστη ο κωδικός του φοιτητή και ο κωδικός του μαθήματος. Με τα στοιχεία αυτά το πρόγραμμα θα εντοπίζει τη σχετική εγγραφή (αν δεν υπάρχει, θα εμφανίζεται μήνυμα σφάλματος), θα την εμφανίζει στην οθόνη και, στη συνέχεια, θα ρωτά τον χρήστη αν θέλει να προχωρήσει ή όχι στην αλλαγή του βαθμού του μαθήματος. Σε περίπτωση που ο χρήστης επιθυμεί να αλλάξει τον βαθμό, το σύστημα θα του ζητήσει τη νέα τιμή του βαθμού και θα αντικαταστήσει την παλιά.

4. **Εμφάνιση εγγραφών:** Με αυτή την επιλογή θα εμφανίζονται στην οθόνη όλες οι εγγραφές που βρίσκονται στο σύστημα, μία σε κάθε γραμμή. Στο τέλος, θα εμφανίζεται και το πλήθος των εγγραφών.
5. **Έξοδος:** Τερματισμός προγράμματος.

Να δημιουργήσετε πρόγραμμα, το οποίο να υλοποιεί το πιο πάνω. Να θεωρήσετε ότι το σύστημα θα μπορεί να αποθηκεύει στοιχεία για μέχρι 1000 φοιτητές.

Άσκηση 8.29

Μία εταιρεία θέλει να αποθηκεύει τα πιο κάτω στοιχεία για κάθε ένα από τα έργα τα οποία έχει αναλάβει να διεκπεραιώσει:

- Κωδικός έργου: Ακέραιος αριθμός, π.χ. 101
- Περιγραφή: Συμβολοσειρά, π.χ. balance
- Προϋπολογισμός: Πραγματικός αριθμός, π.χ. 50,000.00
- Πλήθος υπαλλήλων έργου: Ακέραιος αριθμός (Αρχικά θα είναι 0 και θα αυξάνεται με την προσθήκη ενός υπαλλήλου).
- Υπάλληλοι: Τρεις τιμές για κωδικό υπάλληλου (ακέραιος), επίθετο (συμβολοσειρά) και ετήσιο μισθό (πραγματικός), π.χ. 101, James, 20,000.00.

Να δημιουργήσετε πρόγραμμα, το οποίο να διαβάζει τα στοιχεία των έργων από το αρχείο με όνομα projects.txt, τα στοιχεία των υπαλλήλων από το αρχείο employees.txt και να τα καταχωρίζει σε πίνακα εγγραφών έργων. Για τους υπαλλήλους, να χρησιμοποιήσετε μία ένθετη εγγραφή με τρία μέλη. Το πρόγραμμα να μπορεί να δέχεται στοιχεία για τουλάχιστον 50 έργα. Σε κάθε έργο μπορούν να εργάζονται μέχρι 20 υπάλληλοι και κάθε έργο θα έχει τουλάχιστον έναν υπάλληλο. Σε κάθε γραμμή του αρχείου employees.txt, πριν από τα στοιχεία ενός υπαλλήλου, θα εμφανίζεται ο κωδικός του έργου στο οποίο εργάζεται ο υπάλληλος. Το πρόγραμμα θα πρέπει να τοποθετεί τον κάθε υπάλληλο στο αντίστοιχο έργο. Δείγματα των δύο αρχείων εμφανίζονται πιο κάτω:

`projects.txt`

```
101 Marketing 100000.00
102 Advertising 100000.00
103 Copyrights 50000.00
```

`employees.txt`

```
101 1 Jones 10000.00
101 2 Michael 20000.00
102 3 Compton 30000.00
103 4 Jennings 40000.00
103 5 Adams 45000.00
103 6 Abraham 45000.00
102 7 Rose 35000.00
101 8 Brown 25000.00
```

Επιπρόσθετα το πρόγραμμα να:

(α) καλεί μία συνάρτηση με το όνομα check_balance, η οποία να δέχεται παραμετρικά έναν πίνακα εγγραφών έργων και έναν ακέραιο αριθμό που να αντιπροσωπεύει τον κωδικό ενός έργου. Για το έργο του οποίου δόθηκε παραμετρικά ο κωδικός, η

συνάρτηση να υπολογίζει αν ο προϋπολογισμός του είναι μεγαλύτερος ή ίσος από τους συνολικούς μισθούς των υπαλλήλων που εργάζονται σε αυτό και να επιστρέφει `true`, αν αυτό ισχύει, διαφορετικά, να επιστρέφει `false`. Π.χ. αν δοθούν τα αρχεία που εμφανίζονται πιο πάνω και ως κωδικός έργου δοθεί η τιμή 103, η συνάρτηση check_balance θα επιστρέψει `false` γιατί 40,000+45,000+45,000 > 50,000

(β) καλεί μία συνάρτηση με το όνομα print, η οποία να δέχεται παραμετρικά έναν πίνακα εγγραφών έργων και να εμφανίζει για κάθε έργο, στην πρώτη γραμμή, τον κωδικό του έργου, την περιγραφή, τον προϋπολογισμό και το πλήθος των υπαλλήλων που εργάζονται σε αυτό και στις επόμενες γραμμές να εμφανίζει τα στοιχεία των υπαλλήλων που εργάζονται στο συγκεκριμένο έργο. Π.χ. αν δοθούν τα αρχεία που εμφανίζονται πιο πάνω, η συνάρτηση print θα εμφανίσει:

```
101 Marketing 100000.00 3
1 Jones 10000.00
2 Michael 20000.00
8 Brown 25000.00

102 Advertising 100000.00 2
3 Compton 30000.00
7 Rose 35000.00

103 Copyrights 50000.00 3
4 Jennings 40000.00
5 Adams 45000.00
6 Abraham 45000.00
```

Άσκηση 8.30

Να υλοποιήσετε μία επιπρόσθετη συνάρτηση για την άσκηση **8.29** με το όνομα sort_emp, η οποία να ταξινομεί τους υπαλλήλους κάθε έργου, με βάση τον μισθό τους σε φθίνουσα σειρά. Αν δύο υπάλληλοι έχουν τον ίδιο μισθό, να εμφανίζεται πρώτος αυτός του οποίου το επίθετο προηγείται αλφαβητικά. Το παράδειγμα εξόδου της άσκησης **8.29** θα μετατραπεί, δηλαδή, ως εξής:

```
101 Marketing 100000.00 3
8 Brown 25000.00
2 Michael 20000.00
1 Jones 10000.00

102 Advertising 100000.00 2
7 Rose 35000.00
3 Compton 30000.00

103 Copyrights 50000.00 3
6 Abraham 45000.00
5 Adams 45000.00
4 Jennings 40000.00
```

Γ7.9 Αφηρημένοι τύποι δεδομένων (Abstract data types)

Τι θα μάθουμε σε αυτό το κεφάλαιο:

- Να ορίζουμε τι αποτελεί αφηρημένο τύπο δεδομένων (δομές δεδομένων με συγκεκριμένη συμπεριφορά/λειτουργίες)
- Να περιγράφουμε τη δομή δεδομένων στοίβα (stack) και τις βασικές λειτουργίες της, push() και pop()
- Να δημιουργούμε πρόγραμμα με υλοποίηση και χρήση απλής στοίβας με μονοδιάστατο πίνακα και συναρτήσεις για τις βασικές λειτουργίες της
- Να περιγράφουμε τη δομή δεδομένων ουρά (queue) και τις βασικές λειτουργίες της, enqueue() και dequeue()
- Να δημιουργούμε πρόγραμμα με υλοποίηση και χρήση απλής ουράς με μονοδιάστατο πίνακα και συναρτήσεις για τις βασικές λειτουργίες της
- Να συγκρίνουμε τη στοίβα με την ουρά και να αναφέρουμε τις ομοιότητες και τις διαφορές τους
- Να προετοιμάζουμε ένα πρόγραμμα για χρήση συγκεκριμένης βιβλιοθήκης ενσωματωμένων (έτοιμων) δομών για υλοποίηση στοίβας και ουράς
- Να αναφέρουμε το όνομα και τον ρόλο των επιπρόσθετων λειτουργιών ενσωματωμένων δομών (π.χ. empty(), size()) που παρέχονται για τις ενσωματωμένες δομές
- Να αναγνωρίζουμε από την περιγραφή του προβλήματος αν σε ένα πρόβλημα χρειάζεται η χρήση στοίβας ή ουράς
- Να καθορίζουμε το πρόβλημα με ακρίβεια, συγκεκριμένα: να εντοπίζουμε/διακρίνουμε τα Δεδομένα, τις Πληροφορίες και την Επεξεργασία
- Να σχεδιάζουμε τον τρόπο επίλυσης του προβλήματος
- Να επιλέγουμε να διασπάσουμε το πρόβλημα με χρήση στοίβας ή ουράς για επίλυση του προβλήματος
- Να υλοποιούμε τον σχεδιασμό τους σε πρόγραμμα με τη χρήση του προγραμματιστικού περιβάλλοντος, ώστε να επιλυθεί το πρόβλημα
- Να επιλέγουμε κατάλληλα δεδομένα και στρατηγική για έλεγχο των συναρτήσεών τους και ολόκληρου του προγράμματος
- Να ελέγχουμε την ορθότητα της λύσης του προβλήματος, χρησιμοποιώντας τη μέθοδο της προκαταρκτικής εκτέλεσης ή και άλλες μεθόδους για επαλήθευση
- Να μελετούμε έτοιμο πρόγραμμα το οποίο να περιλαμβάνει συναρτήσεις και να εντοπίζουμε βασικά μέρη του τα οποία συνδέονται με πτυχές του προβλήματος που επιλύει
- Να προσθέτουμε επεξηγηματικά σχόλια σε ένα πρόγραμμα
- Να συμπληρώνουμε ένα έτοιμο πρόγραμμα από το οποίο λείπουν δηλώσεις συναρτήσεων ή και κλήσεις συναρτήσεων
- Να εντοπίζουμε και να αναγνωρίζουμε σε ένα πρόγραμμα πρότυπα σχεδίασης και στρατηγικές (design patterns, τμήματα κώδικα)
- Να χρησιμοποιούμε πρότυπα σχεδίασης και στρατηγικές που εντοπίσαμε ως εργαλεία επίλυσης προβλημάτων σε νέα προγράμματά μας.

1. Εισαγωγή

Αφηρημένοι Τύποι Δεδομένων (Abstract Data Types) είναι σύνολα στοιχείων, τα οποία διαθέτουν μία συλλογή λειτουργιών (ενεργειών) που εφαρμόζονται πάνω στα στοιχεία των συνόλων τους. Για να υλοποιήσουμε έναν αφηρημένο τύπο δεδομένων, χρησιμοποιούμε μία δομή δεδομένων. Οι δομές δεδομένων μπορούν να θεωρηθούν ως εργαλεία, τα οποία χρησιμοποιεί ο υπολογιστής για να αποθηκεύει δεδομένα με δομημένο τρόπο, ώστε να μπορεί να τα χρησιμοποιεί αποδοτικά. Οι δομές δεδομένων θεωρούνται βασικοί οργανωτικοί συντελεστές στον σχεδιασμό λογισμικού, σε μεθόδους σχεδιασμού και σε γλώσσες προγραμματισμού.

Μία δομή δεδομένων περιλαμβάνει:

(α) Ένα σύνολο στοιχείων, τα οποία επιδέχονται επεξεργασία μέσω των λειτουργιών που υποστηρίζονται από τον αφηρημένο τύπο δεδομένων που υλοποιεί η δομή.

(β) Μία δομή αποθήκευσης των δεδομένων (π.χ. έναν πίνακα, μία στοίβα, μία ουρά, κ.λπ.).

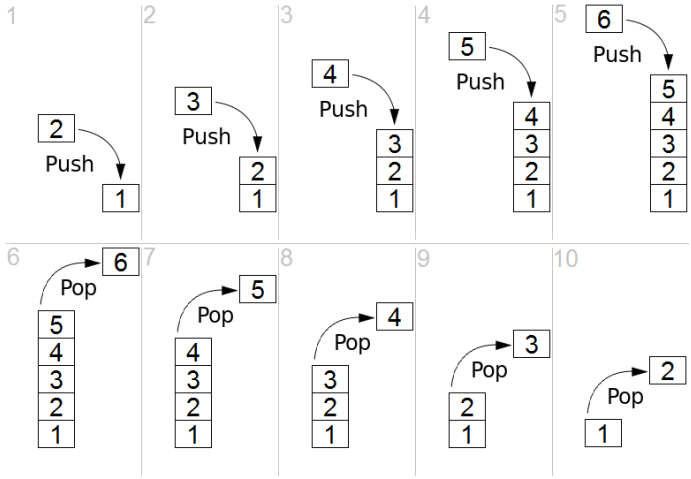
(γ) Ένα σύνολο από ορισμούς συναρτήσεων, που η κάθε συνάρτηση αντιστοιχεί σε μία από τις λειτουργίες της δομής.

Μερικές από τις βασικές λειτουργίες που υποστηρίζει μία δομή δεδομένων είναι οι εξής: προσπέλαση, αναζήτηση, εισαγωγή, διαγραφή, ταξινόμηση, αντιγραφή κ.λπ.

2. Αφηρημένος τύπος δεδομένων - Στοίβα (Stack)

Η στοίβα (stack) είναι μία συλλογή στοιχείων, στην οποία τα νέα στοιχεία μπορούν να προστεθούν και να αφαιρεθούν μόνο από τη μία άκρη της (κορυφή της στοίβας). Κλασικό παράδειγμα στοίβας αποτελεί μία στοίβα από πιάτα, που κάθε νέο πιάτο τοποθετείται στην κορυφή της, ενώ η εξαγωγή ενός πιάτου από τη στοίβα πραγματοποιείται κάθε φορά επίσης από την ίδια κορυφή. Άλλο αντίστοιχο παράδειγμα, είναι η είσοδος αυτοκινήτων σε χώρο στάθμευσης με τη μορφή διαδρόμου. Τα αυτοκίνητα εισέρχονται ένα-ένα και το τελευταίο αυτοκίνητο που μπήκε θα είναι το πρώτο που θα βγει, ώστε να μπορέσουν να βγουν και τα υπόλοιπα.

Μία στοίβα αναφέρεται και ως μία δομή τύπου LIFO (Last-In-First-Out), για να δηλώνεται έτσι η βασική της ιδιότητα, ότι το στοιχείο που θα προστεθεί τελευταίο στη στοίβα θα αφαιρεθεί πρώτο ή, αντίστοιχα, το πρώτο στοιχείο που θα προστεθεί στη στοίβα αναγκαστικά θα πρέπει να αφαιρεθεί τελευταίο.



3. Βασικές λειτουργίες σε στοίβες

Οι βασικές λειτουργίες (ενέργειες) που ορίζονται για τον τύπο της στοίβας είναι οι εξής:

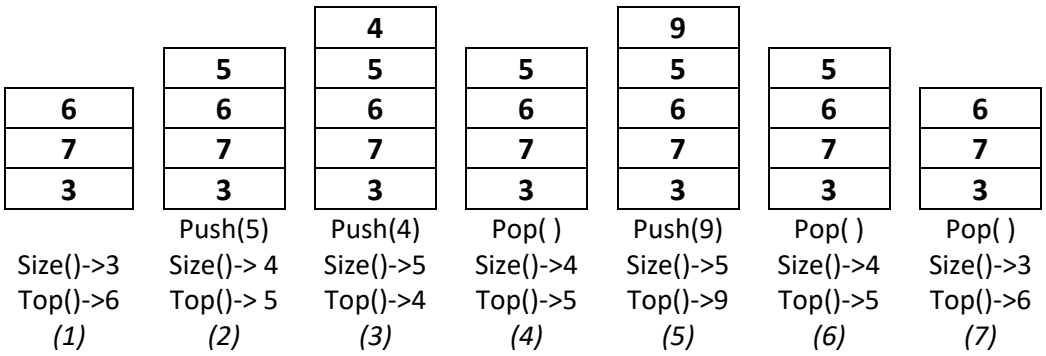
1. Δημιουργία στοίβας: Με την λειτουργία αυτή δημιουργείται μία κενή στοίβα S, η οποία δεν περιέχει, αρχικά, κανένα στοιχείο.
2. Εισαγωγή στοιχείου σε στοίβα - **Push(X)**: Το στοιχείο X τοποθετείται στην κορυφή (top) της στοίβας S ως τελευταίο στοιχείο.
3. Εξαγωγή στοιχείου από στοίβα - **Pop( )**: Το στοιχείο που βρίσκεται στην κορυφή της στοίβας S αφαιρείται από αυτή.

4. Έλεγχος κενής στοίβας - **Empty( )**: Επιστρέφει την τιμή `true`, αν η στοίβα S δεν περιέχει κανένα στοιχείο, ενώ στην αντίθετη περίπτωση επιστρέφει την τιμή `false`.

Επιπρόσθετα από τις βασικές λειτουργίες, ορίζονται και οι παρακάτω:

5. Κορυφή της στοίβας - **Top( )**: Το στοιχείο που βρίσκεται στην κορυφή της στοίβας S επιστρέφεται, χωρίς όμως να αφαιρεθεί από αυτή.
6. Μέγεθος της στοίβας - **Size( )**: Επιστρέφει το πλήθος των στοιχείων της στοίβας.

Πιο κάτω βλέπετε παραστατικά τις λειτουργίες μίας στοίβας:



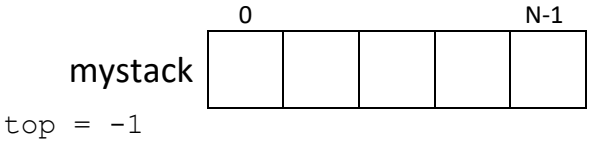
1. Αρχικά, η στοίβα περιέχει τρία στοιχεία (**3, 7, 6**). Η εντολή **Size** θα επιστρέψει τον αριθμό **3**, που είναι το πλήθος των στοιχείων της στοίβας και η εντολή **Top** θα επιστρέψει το στοιχείο **6**, το οποίο είναι το στοιχείο που βρίσκεται στην κορυφή της στοίβας.
2. Με την εντολή **Push(5)** προσθέτουμε το στοιχείο **5** στη στοίβα. Η εντολή **Size** επιστρέφει **4**, γιατί το πλήθος των στοιχείων έχει αυξηθεί κατά ένα. Η εντολή **Top** επιστρέφει το στοιχείο **5**, το οποίο είναι το στοιχείο που βρίσκεται στην κορυφή της στοίβας.
3. Με την εντολή **Push(4)** προσθέτουμε το στοιχείο **4** στη στοίβα. Η εντολή **Size** επιστρέφει **5**, γιατί το πλήθος των στοιχείων έχει αυξηθεί κατά ένα. Η εντολή **Top** επιστρέφει το στοιχείο **4**, το οποίο είναι το στοιχείο που βρίσκεται στην κορυφή της στοίβας.
4. Με την εντολή **Pop** αφαιρούμε από τη στοίβα το στοιχείο **4**, το οποίο βρίσκεται στην κορυφή της στοίβας. Η εντολή **Size** επιστρέφει **4**, γιατί το πλήθος των στοιχείων έχει μειωθεί κατά ένα. Η εντολή **Top** επιστρέφει το στοιχείο **5**, το οποίο είναι το στοιχείο που βρίσκεται στην κορυφή της στοίβας.
5. Με την εντολή **Push(9)** προσθέτουμε το στοιχείο **9** στη στοίβα. Η εντολή **Size** επιστρέφει **5**, γιατί το πλήθος των στοιχείων έχει αυξηθεί κατά ένα. Η εντολή **Top** επιστρέφει το στοιχείο **9**, το οποίο είναι το στοιχείο που βρίσκεται στην κορυφή της στοίβας.
6. Με την εντολή **Pop** αφαιρούμε από τη στοίβα το στοιχείο **9**, το οποίο βρίσκεται στην κορυφή της στοίβας. Η εντολή **Size** επιστρέφει **4**, γιατί το πλήθος των στοιχείων έχει μειωθεί κατά ένα. Η εντολή **Top** επιστρέφει το στοιχείο **5**, το οποίο είναι το στοιχείο που βρίσκεται στην κορυφή της στοίβας.
7. Με την εντολή **Pop** αφαιρούμε από τη στοίβα το στοιχείο **5**, το οποίο βρίσκεται στην κορυφή της στοίβας. Η εντολή **Size** επιστρέφει **3**, γιατί το πλήθος των στοιχείων έχει μειωθεί κατά ένα. Η εντολή **Top** επιστρέφει το στοιχείο **6**, το οποίο είναι το στοιχείο που βρίσκεται στην κορυφή της στοίβας.

4. Υλοποίηση στοίβας με τη χρήση πίνακα και συναρτήσεων

Μπορούμε να υλοποιήσουμε τις λειτουργίες μίας στοίβας με τη χρήση ενός πίνακα συγκεκριμένου μεγέθους. Μία μεταβλητή τύπου ακέραιου θα έχει τον ρόλο του δείκτη, ο οποίος θα μας δείχνει το στοιχείο του πίνακα που βρίσκεται στην κορυφή (top) της στοίβας. Αρχικά, όταν δηλωθεί ο πίνακας, ο δείκτης αυτός θα έχει τιμή **-1**.

4.1 Δημιουργία στοίβας

Η κενή στοίβα που δημιουργείται μπορεί να αναπαρασταθεί με έναν «άδειο» πίνακα και την τιμή του δείκτη **top** να είναι αρχικά **-1**. Στο συγκεκριμένο παράδειγμα δίνουμε μέγεθος **N=5** στον πίνακα.




```
int mystack[5];
int top = -1
```

4.2 Εισαγωγή στοιχείου στη στοίβα

Η λειτουργία εισαγωγής (προσθήκης) ενός στοιχείου στη στοίβα γίνεται με την αύξηση της μεταβλητής-δείκτη top κατά ένα και την τοποθέτηση του στοιχείου στη θέση του πίνακα που δείχνει η μεταβλητή top. Στην περίπτωση που ο χώρος της στοίβας έχει εξαντληθεί από προηγούμενες εισαγωγές στοιχείων, δημιουργείται υπερχείλιση στοίβας την οποία πρέπει να χειριστούμε. Η συνάρτηση Push (item) τύπου void, υλοποιείται ως εξής:

```
void Push (int arr[], int X, int N) {
if (top == N-1) {      // Έλεγχος αν η στοίβα έχει γεμίσει
      cout << "Stack is full" << endl;
  }
else {
      top++;           // Αυξάνουμε τη θέση του δείκτη κατά ένα
      arr[top] = X;    // Τοποθετούμε το στοιχείο X στη θέση top
  }
}
```

4.3 Εξαγωγή στοιχείου από τη στοίβα

Η λειτουργία εξαγωγής (αφαίρεσης) ενός στοιχείου από τη στοίβα γίνεται με τη μείωση του δείκτη top κατά ένα. Αν κατά την λειτουργία εξαγωγής στοιχείου η στοίβα βρεθεί άδεια, δημιουργείται κατάσταση εξαίρεσης (άδεια στοίβα). Η συνάρτηση Pop τύπου void υλοποιείται ως εξής:

```
void Pop ( ) {
if (Empty()) {         // Έλεγχος αν η στοίβα είναι άδεια
      cout << "Stack is empty" << endl;
  }
else {
      top--;           // Μειώνουμε τη θέση του δείκτη κατά ένα
```

```
  }                    // Το στοιχείο που βρισκόταν στη θέση top έχει
}                      // πλέον «αφαιρεθεί» από τη στοίβα
```

4.4 Έλεγχος κενής στοίβας

Στη λειτουργία ελέγχου κενής στοίβας σε περίπτωση που η στοίβα δεν έχει κανένα στοιχείο, είτε μετά από διαδοχικές εξαγωγές στοιχείων είτε μετά τη δημιουργία νέας στοίβας, ο δείκτης top θα πρέπει να έχει την τιμή -1. Η συνάρτηση θα ελέγχει την τιμή του δείκτη και αν η μεταβλητή top είναι ίση με -1, τότε θα επιστρέφει `true` που σημαίνει ότι η στοίβα είναι κενή, αλλιώς θα επιστρέφει `false`. Η συνάρτηση Empty τύπου `boolean` υλοποιείται ως εξής:

```
bool Empty ( ) {
if (top == -1)         // Αν ο δείκτης top είναι ίσος με -1
      return true;     // η στοίβα είναι άδεια, επέστρεψε true
else
      return false;    // Η στοίβα δεν είναι άδεια, επέστρεψε false
}
```

4.5 Επιστροφή στοιχείου κορυφής

Η λειτουργία εμφάνισης του στοιχείου κορυφής της στοίβας επιτρέπει την πρόσβαση στο στοιχείο αυτό, χωρίς όμως να το αφαιρεί από τη στοίβα και χωρίς να αλλάζει την τιμή του δείκτη. Η συνάρτηση Top τύπου `integer` υλοποιείται ως εξής:

```
int Top (int arr[ ]) {
    return arr[top];        // Επιστρέφει το στοιχείο της κορυφής
}
```

4.6 Εμφάνιση μεγέθους της στοίβας

Η λειτουργία εμφάνισης του μεγέθους της στοίβας εμφανίζει το πλήθος των στοιχείων που βρίσκονται μέσα σε αυτή. Η συνάρτηση Size τύπου `integer` υλοποιείται ως εξής:

```
int Size ( ) {
    return top + 1;         // Επιστρέφει το μέγεθος της στοίβας
}
```

5. Υλοποίηση στοίβας σε πρόγραμμα

Ας υλοποιήσουμε ένα πρόγραμμα το οποίο να συνδυάζει και να χρησιμοποιεί όλες τις πιο πάνω συναρτήσεις. Το αρχικό μέγεθος του πίνακα θα οριστεί ως σταθερά.

```
#include<iostream>
using namespace std;
#define ARR_SIZE 5          // Σταθερά για το μέγεθος της στοίβας

int top = -1;               // Καθολική μεταβλητή για τον δείκτη

void Push (int arr[], int X, int N) {
 if (top == N-1) {
     cout << "Stack is full" << endl;
```

```
  }
 else {
      top++;
      arr[top] = X;
      cout << "Pushed item " << X << endl;
  }
}

bool Empty ( ) {
 if (top == -1)
      return true;
 else
      return false;
}

void Pop ( ) {
 if (Empty())
      cout << "Stack is empty" << endl;

else
      top--;
}

int Size ( ) {
      return top + 1;
}

int Top (int arr[]) {
      return arr[top];
}
int main(){
  int mystack[ARR_SIZE];              // Δημιουργία στοίβας
  Push(mystack, 10, ARR_SIZE);        // Pushed item 10
  Push(mystack, 20, ARR_SIZE);        // Pushed item 20
  Push(mystack, 30, ARR_SIZE);        // Pushed item 30
  Push(mystack, 40, ARR_SIZE);        // Pushed item 40
  Push(mystack, 50, ARR_SIZE);        // Pushed item 50
  Push(mystack, 60, ARR_SIZE);        // Stack is full
  Pop( );
  cout << "Size of stack: " << Size( ) << endl;  // Size of stack: 4
  Pop( );
```

```
  Pop( );
  cout<<"Top of stack: "<< Top(mystack) << endl; // Top of stack: 20
  cout<<"Size of stack: "<< Size( ) << endl;     // Size of stack: 2
  Pop( );
  cout<<"Top of stack: " << Top(mystack) << endl;// Top of stack: 10
  cout<<"Size of stack: " << Size( ) << endl;    // Size of stack: 1
  Pop( );
  Pop( );                                        // Stack is empty
return 0;
}
```

(Code: C7_9_example1.cpp)

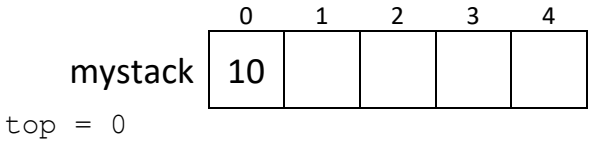
Το αποτέλεσμα του πιο πάνω προγράμματος εμφανίζεται πιο κάτω:

```
Pushed item 10
Pushed item 20
Pushed item 30
Pushed item 40
Pushed item 50
Stack is full
Size of stack: 4
Top of stack: 20
Size of stack: 2
Top of stack: 10
Size of stack: 1
Stack is empty
```

Ακολουθεί μία αναπαράσταση των εντολών του πιο πάνω προγράμματος:

Αρχικά δίνεται εντολή εισαγωγής, η οποία εισάγει το στοιχείο 10 στη στοίβα:

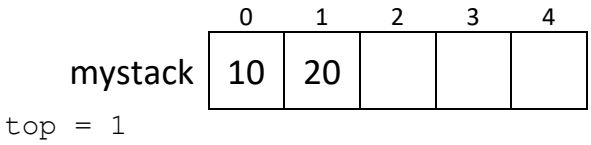
```
Push(mystack, 10, ARR_SIZE);
```




```
-> Pushed item 10
```

Ακολούθως, δίνεται εντολή εισαγωγής, η οποία εισάγει το στοιχείο 20 στη στοίβα:

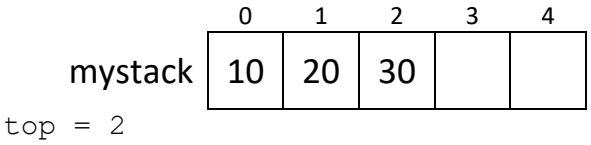
```
Push(mystack, 20, ARR_SIZE);
```




```
-> Pushed item 20
```

Ακολούθως, δίνεται εντολή εισαγωγής, η οποία εισάγει το στοιχείο 30 στη στοίβα:

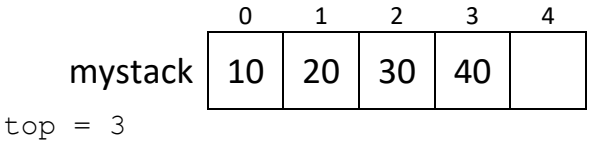
```
Push(mystack, 30, ARR_SIZE);
```




```
-> Pushed item 30
```

Ακολούθως, δίνεται εντολή εισαγωγής, η οποία εισάγει το στοιχείο 40 στη στοίβα:

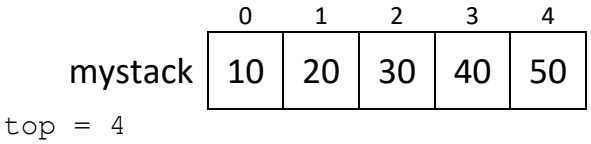
```
Push(mystack, 40, ARR_SIZE);
```




```
-> Pushed item 40
```

Ακολούθως, δίνεται εντολή εισαγωγής, η οποία εισάγει το στοιχείο 50 στη στοίβα:

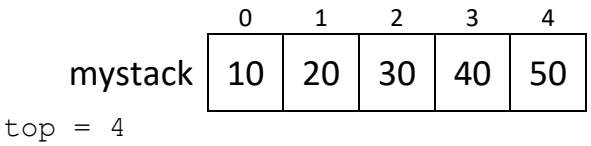
```
Push(mystack, 50, ARR_SIZE);
```




```
-> Pushed item 50
```

Ακολούθως, δίνεται εντολή εισαγωγής, η οποία προσπαθεί να εισαγάγει το στοιχείο 60 στη στοίβα. Η στοίβα όμως είναι γεμάτη. Το πρόγραμμα επιστρέφει το μήνυμα «Stack is full» και δεν προσθέτει το στοιχείο 60 στη στοίβα.

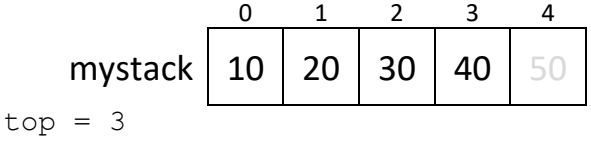
```
Push(mystack, 60, ARR_SIZE);
```




```
-> Stack is full
```

Ακολούθως, δίνεται εντολή εξαγωγής, η οποία «αφαιρεί» το στοιχείο 50 από τη στοίβα. Ο δείκτης top μειώνεται κατά ένα.

```
Pop( );
```



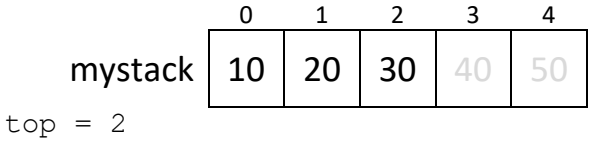
Ακολουθεί εντολή εξόδου, η οποία εμφανίζει το μέγεθος της στοίβας:

```
cout << "Size of stack: " << Size( ) << endl;
```

```
-> Size of stack: 4          // Το πλήθος των στοιχείων είναι 4
```

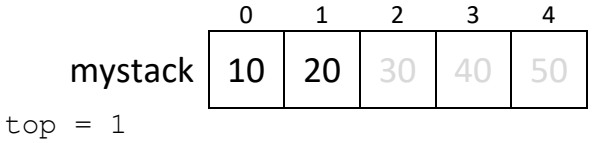
Ακολούθως, δίνεται εντολή εξαγωγής, η οποία «αφαιρεί» το στοιχείο 40 από τη στοίβα. Ο δείκτης top μειώνεται κατά ένα.

```
Pop( );
```



Ακολούθως, δίνεται εντολή εξαγωγής, η οποία «αφαιρεί» το στοιχείο 30 από τη στοίβα. Ο δείκτης top μειώνεται κατά ένα.

```
Pop( );
```



Ακολουθεί εντολή εξόδου, η οποία εμφανίζει το στοιχείο που βρίσκεται στην κορυφή της στοίβας:

```
cout << "Top of stack: " << Top(mystack) << endl;
```

```
-> Top of stack: 20          // Το στοιχείο στην κορυφή είναι το 20
```

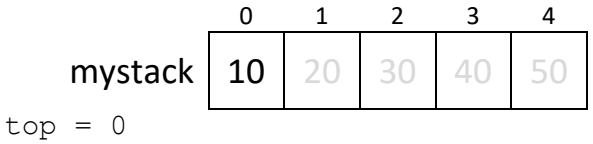
Ακολουθεί εντολή εξόδου, η οποία εμφανίζει το μέγεθος της στοίβας:

```
cout << "Size of stack: " << Size( ) << endl;
```

```
-> Size of stack: 2          // Το πλήθος των στοιχείων είναι 2
```

Ακολούθως, δίνεται εντολή εξαγωγής, η οποία «αφαιρεί» το στοιχείο 20 από τη στοίβα. Ο δείκτης top μειώνεται κατά ένα.

```
Pop( );
```



Ακολουθεί εντολή εξόδου, η οποία εμφανίζει το στοιχείο που βρίσκεται στην κορυφή της στοίβας:

```
cout << "Top of stack: " << Top(mystack) << endl;
```

```
-> Top of stack: 10          // Το στοιχείο στην κορυφή είναι το 10
```

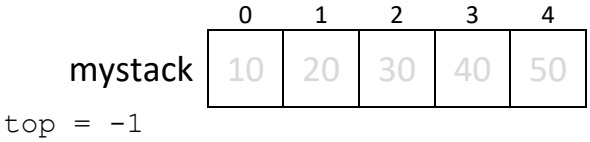
Ακολουθεί εντολή εξόδου, η οποία εμφανίζει το μέγεθος της στοίβας:

```
cout << "Size of stack: " << Size( ) << endl;
```

```
-> Size of stack: 1     // Το πλήθος των στοιχείων είναι 1
```

Ακολούθως, δίνεται εντολή εξαγωγής, η οποία «αφαιρεί» το στοιχείο 10 από τη στοίβα. Ο δείκτης top μειώνεται κατά ένα. Η στοίβα πλέον είναι κενή.

```
Pop( );
```



Ακολούθως, δίνεται εντολή εξαγωγής, η οποία προσπαθεί να αφαιρέσει στοιχείο από τη στοίβα. Η στοίβα όμως είναι άδεια. Το πρόγραμμα επιστρέφει το μήνυμα «Stack is empty».

```
Pop ( )
```

```
-> Stack is empty
```

6. Χρήση στοίβας μέσω της βιβλιοθήκης STL (Standard Template Library)

Η πρότυπη βιβλιοθήκη (STL) ορίζει ένα σύνολο από περιέχοντες (containers), μεταξύ των οποίων και τη στοίβα. Οι περιέχοντες είναι ένας τρόπος οργάνωσης μίας συλλογής αντικειμένων στη μνήμη του υπολογιστή, οι οποίοι περιέχουν ενσωματωμένες τις βασικές λειτουργίες κάθε δομής που υλοποιούν. Η πιο κάτω δήλωση μάς παρέχει έναν περιέχοντα (container) τύπου στοίβας:

```
#include <stack>
```

Με τη χρήση της πιο πάνω επικεφαλίδας, μπορούμε να δηλώσουμε μία στοίβα οποιουδήποτε βασικού τύπου και να έχουμε πρόσβαση στις μεθόδους που αποτελούν τις βασικές της λειτουργίες. Υπενθυμίζεται ότι ισχύει η αρχή LIFO (το στοιχείο που θα προστεθεί τελευταίο στη στοίβα θα αφαιρεθεί πρώτο) και ότι έχουμε πρόσβαση μόνο στο στοιχείο που βρίσκεται στην κορυφή.

6.1 Δήλωση στοίβας

Για να δηλώσουμε μία στοίβα, πρέπει να ορίσουμε τον τύπο και το αναγνωριστικό της. Αρχικά, η στοίβα που δηλώνουμε έχει μέγεθος 0. Καθώς θα προσθέτουμε στοιχεία, το μέγεθός της στοίβας θα προσαρμόζεται ανάλογα με τα στοιχεία που θα περιέχει.

```
stack<int> my_stack;   // δηλώνει μία στοίβα ακεραίων
stack<char> S1, S2;    // δηλώνει δύο στοίβες χαρακτήρων
stack<string> words    // δηλώνει μία στοίβα συμβολοσειρών
```

6.2 Μέθοδοι στοίβας

Οι βασικές λειτουργίες της στοίβας παρέχονται μέσα από τις ενσωματωμένες συναρτήσεις που προσφέρονται και είναι οι εξής:

push(item)	Με τη συνάρτηση push γίνεται εισαγωγή ενός στοιχείου στην κορυφή της στοίβας.
pop( )	Με τη συνάρτηση pop γίνεται εξαγωγή ενός στοιχείου από την κορυφή της στοίβας.
top( )	Η συνάρτηση top επιστρέφει το στοιχείο που βρίσκεται στην κορυφή της στοίβας, χωρίς να το αφαιρέσει.
empty( )	Η συνάρτηση empty επιστρέφει true, αν η στοίβα είναι άδεια, αλλιώς επιστρέφει false.
size( )	Η συνάρτηση size επιστρέφει το πλήθος των στοιχείων της στοίβας.

6.3 Κλήση συναρτήσεων

Για να χρησιμοποιήσουμε τις πιο πάνω συναρτήσεις, βάζουμε πρώτα το αναγνωριστικό που δηλώσαμε, ακολουθούμενο από μία τελεία και το όνομα της συνάρτησης την οποία θέλουμε να καλέσουμε. Ας δούμε ένα παράδειγμα:

```
#include<iostream>
#include<stack>
using namespace std;
int main(){
 stack<char> mystack;                    // Δήλωση στοίβας χαρακτήρων
 mystack.push('A');                      // Εισάγουμε το στοιχείο A
 mystack.push('B');                      // Εισάγουμε το στοιχείο B
 mystack.push('C');                      // Εισάγουμε το στοιχείο C
 cout << mystack.top() << endl;          // Στοιχείο κορυφής -> C
 mystack.pop( );                         // Εξαγωγή στοιχείου κορυφής
 cout << mystack.size() << endl;         // Μέγεθος στοίβας -> 2
return 0;
}
```

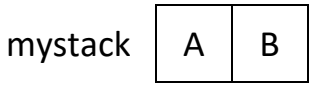
(Code: C7_9_example2.cpp)

Ακολουθεί μία αναπαράσταση των εντολών του πιο πάνω προγράμματος:

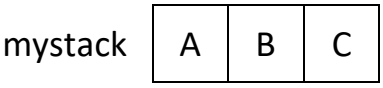
Αρχικά, η εντολή `stack<char> mystack;` δηλώνει μία στοίβα χαρακτήρων με όνομα `mystack`. Η στοίβα έχει μέγεθος 0.

Ακολούθως, η εντολή `mystack.push('A');` εισάγει στη στοίβα το στοιχείο A. Η στοίβα έχει τώρα μέγεθος 1.

Ακολούθως, η εντολή `mystack.push('B');` εισάγει στη στοίβα το στοιχείο B. Η στοίβα έχει τώρα μέγεθος 2.



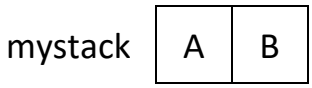
Ακολούθως, η εντολή `mystack.push('C');` εισάγει στη στοίβα το στοιχείο C. Η στοίβα έχει τώρα μέγεθος 3.



Ακολουθεί η εντολή εξόδου `cout << mystack.top() << endl;` η οποία εμφανίζει το στοιχείο που βρίσκεται στην κορυφή της στοίβας:

```
-> C        // Το στοιχείο στην κορυφή είναι το C
```

Ακολούθως, η εντολή `mystack.pop( );` αφαιρεί από τη στοίβα το στοιχείο C. Η στοίβα έχει τώρα μέγεθος 2.



Ακολουθεί η εντολή εξόδου `cout << mystack.size() << endl;` η οποία εμφανίζει το μέγεθος της στοίβας:

```
-> 2        // Το πλήθος των στοιχείων της στοίβας είναι 2
```

6.4 Εμφάνιση όλων των στοιχείων μίας στοίβας

Σε περίπτωση που θέλουμε να εμφανίσουμε όλα τα στοιχεία μία στοίβας, μπορούμε να χρησιμοποιήσουμε την πιο κάτω μέθοδο. Η μέθοδος λειτουργεί ως εξής:

(α) Αρχικά, εισάγουμε στη στοίβα τα στοιχεία που θέλουμε.

(β) Εμφανίζουμε το στοιχείο της κορυφής.

(γ) Αφαιρούμε το στοιχείο της κορυφής.

(δ) Συνεχίζουμε μέχρι να αδειάσει η στοίβα.

Μπορούμε να χρησιμοποιήσουμε τη συνάρτηση empty() για τον έλεγχο. Τα στοιχεία της στοίβας θα εμφανίζονται με την αντίθετη σειρά από αυτή που είχαν δοθεί.

Δείτε το πιο κάτω παράδειγμα:

```
#include<iostream>
#include<stack>
using namespace std;
int main(){
 stack<int> S;              // Δήλωση στοίβας ακεραίων
 for (int i=1; i<=5; i++)
     S.push(i*10);          // Η στοίβα περιέχει: {10,20,30,40,50}
 cout << S.top() << endl;       // 50
 while(!S.empty()){             // Για όσο η στοίβα δεν είναι άδεια
      cout << S.top() << " ";   // εμφάνισε το στοιχείο της κορυφής
      S.pop();                  // και αφαίρεσέ το από τη στοίβα
```

```
   }                          // 50 40 30 20 10
return 0;
}
```

(Code: C7_9_example3.cpp)

Σημαντική διαφορά της στοίβας που δημιουργούμε μέσω της βιβλιοθήκης STL σε σχέση με τη δημιουργία στοίβας με δικές μας συναρτήσεις, είναι ότι αν προσπαθήσουμε να αφαιρέσουμε στοιχείο από άδεια στοίβα, χωρίς να κάνουμε έλεγχο, δεν θα εμφανιστεί κάποιο μήνυμα σφάλματος. Για παράδειγμα, δείτε το πιο κάτω πρόγραμμα:

```
#include<iostream>
#include<stack>
using namespace std;
int main(){
 stack<int> S;              // Δήλωση στοίβας ακεραίων
 for (int i=1; i<=5; i++)
     S.push(i*10);          // Η στοίβα περιέχει: {10,20,30,40,50}
 for (int j=1; j<=10; j++) // Εκτελούμε περισσότερα pop από τα
     S.pop();               // στοιχεία που περιέχει η στοίβα
return 0;                   // Το πρόγραμμα εκτελείται κανονικά
}
```

Μία περίπτωση σφάλματος που μπορούμε να συναντήσουμε στην υλοποίηση στοίβας μέσω της βιβλιοθήκης STL, είναι αν προσπαθήσουμε να έχουμε πρόσβαση στο στοιχείο της κορυφής (top) μίας άδειας στοίβας. Για παράδειγμα, στο πιο κάτω πρόγραμμα προσπαθούμε να εμφανίσουμε το στοιχείο της κορυφής, όταν πλέον έχουμε αδειάσει τη στοίβα. Συνιστάται, λοιπόν, να γίνονται πάντοτε οι απαραίτητοι έλεγχοι, όταν χειριζόμαστε στοίβες.

```
#include<iostream>
#include<stack>
using namespace std;

int main(){
 stack<int> S;              // Δήλωση στοίβας ακεραίων
 for (int i=1; i<=5; i++)
     S.push(i*10);          // Η στοίβα περιέχει: {10,20,30,40,50}
 cout << S.top() << endl;   // 50
 for (int j=1; j<=10; j++) // Εκτελούμε περισσότερα pop από τα
     S.pop();               // στοιχεία που περιέχει η στοίβα
 cout << S.top() << endl;   // ΣΦΑΛΜΑ. Η στοίβα είναι κενή
return 0;
}
```

Παράδειγμα 9.4

Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται άγνωστο αριθμό λέξεων. Το πρόγραμμα να σταματά να δέχεται λέξεις, όταν δοθεί η λέξη «stop». Στη συνέχεια, να τυπώνει τις λέξεις με αντίστροφη σειρά από αυτή που είχαν δοθεί.

Προσέγγιση: Θα διαβάσουμε μία-μία τις λέξεις και θα τις εισαγάγουμε σε μία στοίβα. Αφαιρώντας και εμφανίζοντας τις λέξεις, η τελευταία που δόθηκε θα είναι και η πρώτη που θα εμφανιστεί.

```
#include<iostream>
#include<stack>
#include<string>
using namespace std;

int main(){
 stack <string> words;      // Δήλωση στοίβας συμβολοσειρών
 string st;                 // Μεταβλητή τύπου string
 cin >> st;

 while(st!="stop") {        // Διαβάζουμε λέξεις μέχρι να δοθεί stop
    words.push(st);         // Εισάγουμε τη λέξη στη στοίβα
    cin >> st;              // Διαβάζουμε την επόμενη λέξη
 }
 while(!words.empty()){           // Μέχρι να αδειάσει η στοίβα,
    cout << words.top() << endl; // εμφανίζουμε τη λέξη που
    words.pop();                 // βρίσκεται στην κορυφή και την
 }                                // αφαιρούμε από τη στοίβα
return 0;
}
```

(Code: C7_9_example4.cpp)

Παράδειγμα 9.5

Να δημιουργήσετε πρόγραμμα, το οποίο, με τη χρήση στοίβας, να μετατρέπει έναν αριθμό από το δεκαδικό στο δυαδικό σύστημα αρίθμησης. Π.χ. ο αριθμός 35 μετατρέπεται στον αριθμό 100011.

Προσέγγιση: Ένας αριθμός του δεκαδικού μετατρέπεται στο δυαδικό με τον εξής τρόπο: Διαιρούμε τον αριθμό με το 2 και βρίσκουμε το αποτέλεσμα και το υπόλοιπο της διαίρεσης. Εξακολουθούμε να διαιρούμε το αποτέλεσμα της διαίρεσης με το 2 μέχρι το αποτέλεσμα να γίνει 0 και σε κάθε διαίρεση αποθηκεύουμε το υπόλοιπο της σε μία στοίβα.

```
#include<iostream>
#include<stack>
using namespace std;

int main(){
 stack <int> binary;        // Δήλωση στοίβας ακεραίων
 int num;
 cin >> num;
 while(num > 0) {           // Όσο ο αριθμός είναι μεγαλύτερος του 0,
    binary.push(num % 2);   // εισάγουμε το υπόλοιπο num % 2,
    num /= 2;               // διαιρούμε με το 2 και αποθηκεύουμε
 }
```

```
 while(!binary.empty()){          // Μέχρι να αδειάσει η στοίβα,
    cout << binary.top();         // εμφανίζουμε το στοιχείο που
    binary.pop();                 // βρίσκεται στην κορυφή και το
 }                                // αφαιρούμε από τη στοίβα
return 0;
}
```

(Code: C7_9_example5.cpp)

Παράδειγμα 9.6

Οι παρενθέσεις ( ) είναι σημαντικά στοιχεία των λογικών προτάσεων. Όλες οι παρενθέσεις πρέπει να είναι ισορροπημένες, δηλαδή για κάθε παρένθεση που ανοίγει (αριστερή) πρέπει να υπάρχει και μία που να κλείνει (δεξιά). Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται μία λογική πρόταση και να γίνεται έλεγχος αν είναι ισορροπημένη σε σχέση με το πλήθος των παραθέσεων που περιέχει. Το πρόγραμμα να εμφανίζει τα μηνύματα «Balanced», αν η λογική πρόταση περιέχει ισορροπημένο αριθμό παρενθέσεων, αλλιώς να εμφανίζει το μήνυμα «Not balanced».

Π.χ. η πρόταση `((A || B) && C)` είναι ισορροπημένη ενώ η `(!(Z && Y || X)))`, δεν είναι.

Προσέγγιση: Για να επιλύσουμε το πιο πάνω πρόβλημα, θα χρησιμοποιήσουμε μία στοίβα. Κάθε φορά που βρίσκουμε μία αριστερή παρένθεση θα την εισαγάγουμε στη στοίβα, ενώ κάθε φορά που βρίσκουμε μία δεξιά παρένθεση θα ελέγχουμε πρώτα αν η στοίβα είναι άδεια. Αν η στοίβα είναι άδεια, σημαίνει ότι έχουμε βρει περισσότερες δεξιές παρενθέσεις απ' ό,τι αριστερές και η λογική μας πρόταση δεν είναι ισορροπημένη. Αν η στοίβα δεν είναι κενή, τότε θα αφαιρούμε μία αριστερή παρένθεση από τη στοίβα. Όταν φτάσουμε στο τέλος της λογικής πρότασης, αν το μέγεθος της στοίβας μας είναι 0, τότε έχουμε ισορροπημένη πρόταση, ενώ σε αντίθετη περίπτωση η λογική πρόταση δεν είναι ισορροπημένη.

```
#include<iostream>
#include<stack>
#include<string>
using namespace std;

// Συνάρτηση που ελέγχει αν μία λογική έκφραση έχει ισορροπημένο
// αριθμό παρνθέσεων
bool check_if_balanced(string st){
  stack <char> par;               // Δήλωση στοίβας χαρακτήρων
  for (int i=0; i<st.size(); i++){
    if (st[i]=='(')               // Αν βρούμε αριστερή παρένθεση
        par.push(st[i]);          // την εισάγουμε στη στοίβα
    if (st[i]==')'){              // Αν βρούμε δεξιά παρένθεση
          if (!par.empty())       // και αν η στοίβα δεν είναι κενή,
              par.pop();          // τότε αφαιρούμε ένα στοιχείο
          else
              return false;       // Αλλιώς, αν η στοίβα είναι κενη
```

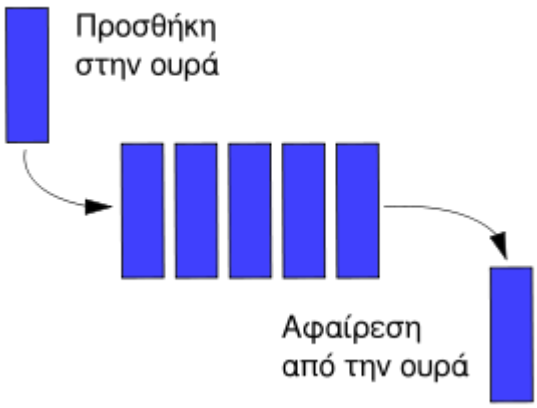
```
      }                              // επιστρέφουμε false. Η πρόταση
   }                                 // δεν είναι ισορροπημένη
 return par.empty();                 // Αν η στοίβα είναι κενή θα
}                                    // επιστρέψει true, αλλιώς θα
                                     // επιστρέψει false
// Κύρια συνάρτηση (main)
int main(){

 string exp;
 cin >> exp;

// Αν η συνάρτηση επιστρέψει true η πρόταση είναι ισορροπημένη
 if(check_if_balanced(exp))
           cout << "Balanced" << endl;
     else
           cout << "Not balanced" << endl;
return 0;
}
```

(Code: C7_9_example6.cpp)

7. Αφηρημένος τύπος δεδομένων - Ουρά (Queue)

Η ουρά (queue) είναι μία συλλογή στοιχείων, στην οποία τα νέα στοιχεία μπορούν να προστεθούν από τη πίσω άκρη και να αφαιρεθούν μόνο από τη μπροστινή άκρη της. Ένα παράδειγμα ουράς αποτελεί μία ουρά από πελάτες σε ένα ταμείο τράπεζας, όπου κάθε νέος πελάτης παίρνει θέση στο τέλος της ουράς, ενώ ο πελάτης που εξυπηρετείται αποχωρεί από το μπροστινό μέρος της ουράς.

Μία ουρά αναφέρεται και ως μία δομή τύπου FIFO (First-In-First-Out), για να δηλώνεται έτσι η βασική της ιδιότητα, ότι το στοιχείο που θα προστεθεί πρώτο στην ουρά θα αφαιρεθεί πρώτο ή, αντίστοιχα, το τελευταίο στοιχείο που θα προστεθεί στην ουρά, αναγκαστικά θα πρέπει να αφαιρεθεί τελευταίο.



8. Βασικές λειτουργίες σε ουρές

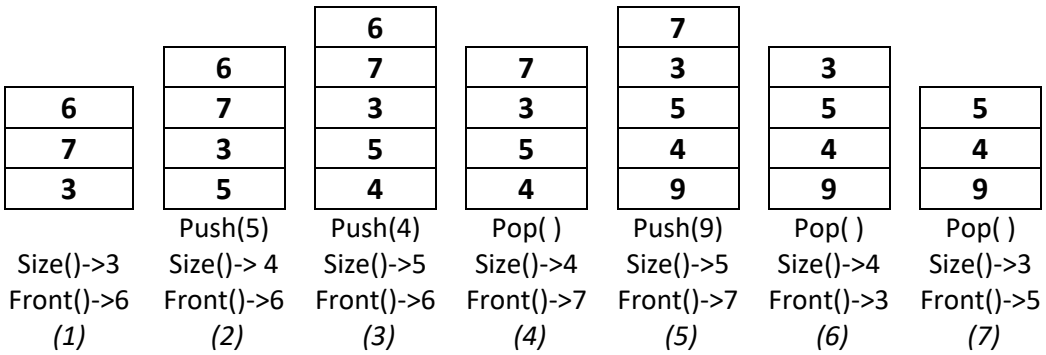
Οι βασικές λειτουργίες (ενέργειες) που ορίζονται για τον τύπο της ουράς είναι οι εξής:

1. Δημιουργία ουράς: Με τη λειτουργία αυτή δημιουργείται μία κενή ουρά Q, η οποία δεν περιέχει, αρχικά, κανένα στοιχείο.
2. Εισαγωγή στοιχείου σε ουρά - **Push(X)**: Το στοιχείο X τοποθετείται στο πίσω μέρος (back) της ουράς Q ως τελευταίο στοιχείο.
3. Εξαγωγή στοιχείου από ουρά - **Pop( )**: Το στοιχείο που βρίσκεται στην πρώτη θέση της ουράς Q αφαιρείται από αυτή.
4. Έλεγχος κενής ουράς - **Empty( )**: Επιστρέφει την τιμή `true`, αν η ουρά Q δεν περιέχει κανένα στοιχείο, ενώ στην αντίθετη περίπτωση επιστρέφει την τιμή `false`.

Επιπρόσθετα από τις βασικές λειτουργίες, ορίζονται και οι παρακάτω:

5. Πρώτο στοιχείο ουράς - **Front( )**: Επιστρέφει το πρώτο στοιχείο της ουράς, χωρίς όμως να το αφαιρεί από αυτή.
6. Μέγεθος της ουράς - **Size( )**: Επιστρέφει το πλήθος των στοιχείων της ουράς.

Πιο κάτω βλέπετε παραστατικά τις λειτουργίες μίας ουράς:



1. Αρχικά, η ουρά περιέχει τρία στοιχεία (**3, 7, 6**). Η εντολή **Size** θα επιστρέψει τον αριθμό **3** που είναι το πλήθος των στοιχείων της ουράς και η εντολή **Front** θα επιστρέψει το στοιχείο **6**, το οποίο είναι το στοιχείο που βρίσκεται στην πρώτη θέση της ουράς.
2. Με την εντολή **Push(5)** προσθέτουμε το στοιχείο **5** στην τελευταία θέση της ουράς. Η εντολή **Size** επιστρέφει **4**, γιατί το πλήθος των στοιχείων έχει αυξηθεί κατά ένα. Η εντολή **Front** επιστρέφει το στοιχείο **6**, το οποίο είναι το στοιχείο που βρίσκεται στην πρώτη θέση της ουράς.
3. Με την εντολή **Push(4)** προσθέτουμε το στοιχείο **4** στην τελευταία θέση της ουράς. Η εντολή **Size** επιστρέφει **5**, γιατί το πλήθος των στοιχείων έχει αυξηθεί κατά ένα. Η εντολή **Front** επιστρέφει το στοιχείο **6**, το οποίο είναι το στοιχείο που βρίσκεται στην πρώτη θέση της ουράς.
4. Με την εντολή **Pop** αφαιρούμε από την ουρά το στοιχείο **6**, το οποίο βρίσκεται στην πρώτη θέση της ουράς. Η εντολή **Size** επιστρέφει **4**, γιατί το πλήθος των στοιχείων έχει μειωθεί κατά ένα. Η εντολή **Front** επιστρέφει το στοιχείο **7**, το οποίο είναι το στοιχείο που βρίσκεται στην πρώτη θέση της ουράς.
5. Με την εντολή **Push(9)** προσθέτουμε το στοιχείο **9** στην τελευταία θέση της ουράς. Η εντολή **Size** επιστρέφει **5**, γιατί το πλήθος των στοιχείων έχει αυξηθεί

κατά ένα. Η εντολή **Front** επιστρέφει το στοιχείο **7**, το οποίο είναι το στοιχείο που βρίσκεται στην πρώτη θέση της ουράς.

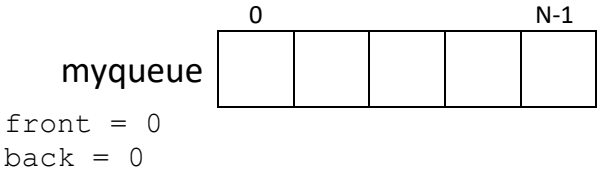
6. Με την εντολή **Pop** αφαιρούμε από την ουρά το στοιχείο **7**, το οποίο βρίσκεται στην πρώτη θέση της ουράς. Η εντολή **Size** επιστρέφει **4**, γιατί το πλήθος των στοιχείων έχει μειωθεί κατά ένα. Η εντολή **Front** επιστρέφει το στοιχείο **3**, το οποίο είναι το στοιχείο που βρίσκεται στην πρώτη θέση της ουράς.
7. Με την εντολή **Pop** αφαιρούμε από την ουρά το στοιχείο **3**, το οποίο βρίσκεται στην πρώτη θέση της ουράς. Η εντολή **Size** επιστρέφει **3**, γιατί το πλήθος των στοιχείων έχει μειωθεί κατά ένα. Η εντολή **Front** επιστρέφει το στοιχείο **5**, το οποίο είναι το στοιχείο που βρίσκεται στην πρώτη θέση της ουράς.

9. Υλοποίηση ουράς με τη χρήση πίνακα και συναρτήσεων

Μπορούμε να υλοποιήσουμε τις λειτουργίες μίας ουράς με τη χρήση ενός πίνακα συγκεκριμένου μεγέθους. Χρησιμοποιούμε δύο ακέραιους αριθμούς (δείκτες), από τους οποίους ο ένας υποδεικνύει το τελευταίο στοιχείο (back) της ουράς και ο άλλος υποδεικνύει το πρώτο στοιχείο της ουράς (front).

9.1 Δημιουργία ουράς

Η κενή ουρά που δημιουργείται μπορεί να αναπαρασταθεί με έναν «άδειο» πίνακα. Κατά την δημιουργία της ουράς, οι τιμές των δεικτών front και back είναι μηδέν. Στο συγκεκριμένο παράδειγμα δίνουμε μέγεθος **N=5** στον πίνακα.




```
int myqueue[5];
int front = 0, back = 0;
```

9.2 Εισαγωγή στοιχείου στην ουρά

Η λειτουργία εισαγωγής (προσθήκης) ενός στοιχείου στην ουρά γίνεται με την τοποθέτηση του στοιχείου στη θέση του πίνακα που υποδεικνύει ο δείκτης back. Στη συνέχεια, αυξάνουμε την τιμή του δείκτη αυτού κατά ένα. Στην περίπτωση που ο χώρος της ουράς έχει εξαντληθεί από προηγούμενες εισαγωγές στοιχείων, δημιουργείται υπερχείλιση ουράς, την οποία πρέπει να χειριστούμε. Η συνάρτηση Push (item) τύπου `void` υλοποιείται ως εξής:

```
void Push (int arr[], int X, int N) {
if (back == N) {       // Έλεγχος αν η ουρά έχει γεμίσει
      cout << "Queue is full" << endl;
  }
else {
      arr[back] = X;   // Τοποθετούμε το στοιχείο X στη θέση back
      back++;          // Αυξάνουμε τη θέση του δείκτη κατά ένα
  }
}
```

9.3 Εξαγωγή στοιχείου από την ουρά

Η λειτουργία εξαγωγής (αφαίρεσης) ενός στοιχείου από την ουρά γίνεται με την αύξηση του δείκτη front κατά ένα. Εάν κατά τη λειτουργία εξαγωγής στοιχείου η ουρά βρεθεί άδεια, δημιουργείται κατάσταση εξαίρεσης (άδεια ουρά). Η συνάρτηση Pop τύπου `void` υλοποιείται ως εξής:

```
void Pop (int arr[ ]) {
if (Empty()) {          // Έλεγχος αν η ουρά είναι άδεια
      cout << "Queue is empty" << endl;
  }
else {
      arr[front] = 0;  // Βάζουμε 0 γιατί το στοιχείο «αφαιρέθηκε»
      front++;         // Αυξάνουμε τη θέση του δείκτη κατά ένα
 }
}
```

9.4 Έλεγχος κενής ουράς

Στη λειτουργία ελέγχου κενής ουράς, σε περίπτωση που η ουρά δεν έχει κανένα στοιχείο, είτε μετά από διαδοχικές εξαγωγές στοιχείων είτε μετά τη δημιουργία νέας ουράς, οι δείκτες front και back θα πρέπει να έχουν την ίδια τιμή. Η συνάρτηση θα ελέγχει αν είναι ίσοι οι δείκτες και θα επιστρέφει `true`, που σημαίνει ότι η ουρά είναι κενή, αλλιώς θα επιστρέφει `false`. Η συνάρτηση Empty τύπου `boolean` υλοποιείται ως εξής:

```
bool Empty ( ) {
      return (front == back);      // Αν οι δείκτες είναι ίσοι
}                                  // τότε η ουρά είναι κενή
```

9.5 Επιστροφή μπροστινού στοιχείου

Η λειτουργία εμφάνισης του μπροστινού στοιχείου της ουράς επιτρέπει την πρόσβαση στο στοιχείο αυτό, χωρίς όμως να το αφαιρεί από την ουρά και χωρίς να αλλάζει την τιμή του δείκτη front. Η συνάρτηση Front τύπου `integer` υλοποιείται ως εξής:

```
int Front (int arr[]) {
      return arr[front];          // Επιστρέφει το πρώτο στοιχείο
}
```

9.6 Εμφάνιση μεγέθους της ουράς

Η λειτουργία εμφάνισης του μεγέθους της ουράς εμφανίζει το πλήθος των στοιχείων που βρίσκονται μέσα σε αυτή. Η συνάρτηση Size τύπου `integer` υλοποιείται ως εξής:

```
int Size ( ) {
      return (back - front);      // Επιστρέφει το μέγεθος της ουράς
}
```

10. Υλοποίηση ουράς σε πρόγραμμα

Ας υλοποιήσουμε ένα πρόγραμμα το οποίο να συνδυάζει και να χρησιμοποιεί όλες τις πιο πάνω συναρτήσεις. Το αρχικό μέγεθος του πίνακα θα οριστεί ως σταθερά.

```
#include <iostream>
using namespace std;
#define ARR_SIZE 5                // Μέγεθος ουράς
int front=0, back=0;              // Καθολικές μεταβλητές

void Push(int arr[], int X, int N) {
    if(back == N)
        cout << "Queue is full" << endl;
    else {
        arr[back] = X;
        back++;
        cout << "Pushed item " << X << endl;
    }
}

bool Empty( ) {
     return (front == back);
}
void Pop(int arr[]) {
    if(Empty())
        cout << "Queue is empty" << endl;
    else {
        arr[front] = 0;
        front++;
    }
}
int Front(int arr[]) {
    return arr[front];
}
int Size() {
    return (back - front);
}

int main() {
    int myqueue[ARR_SIZE];        // Δημιουργία ουράς

 for(int num = 10; num<=15; num++)
     Push(myqueue,num, ARR_SIZE);

 cout << "Front: " << Front(myqueue) <<" Size: "<< Size( ) << endl;
 Pop(myqueue);
```

```
 cout << "Front: " << Front(myqueue) <<" Size: "<< Size( ) << endl;
 Pop(myqueue);
 cout << "Front: " << Front(myqueue) <<" Size: "<< Size( ) << endl;

 for(int i=front; i<back; i++)
      cout <<  myqueue[i] << " ";
return 0;
}
```

(Code: C7_9_example7.cpp)

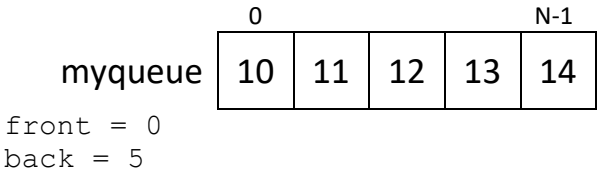
Το αποτέλεσμα του πιο πάνω προγράμματος εμφανίζεται πιο κάτω:

```
Pushed item 10
Pushed item 11
Pushed item 12
Pushed item 13
Pushed item 14
Queue is full
Front: 10 Size: 5
Front: 11 Size: 4
Front: 12 Size: 3
12 13 14
```

Ακολουθεί μία αναπαράσταση των εντολών του πιο πάνω προγράμματος:

Αρχικά, μέσω μίας δομής επανάληψης, δίνονται διαδοχικά εντολές εισαγωγής, οι οποίες εισάγουν τα στοιχεία 10, 11, 12, 13, 14 στην ουρά. Το στοιχείο 15 δεν μπορεί να εισαχθεί, γατί η ουρά είναι γεμάτη. Το πρόγραμμα επιστρέφει το μήνυμα «Queue is full» και δεν προσθέτει το στοιχείο 15 στην ουρά. Ο δείκτης front έχει τιμή 0 και ο δείκτης back έχει τιμή 5.

```
Push(myqueue, num, ARR_SIZE);
```

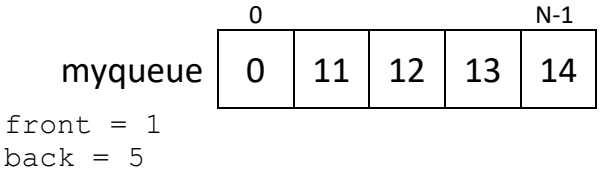


Ακολουθεί εντολή εξόδου, η οποία εμφανίζει το στοιχείο που βρίσκεται στην πρώτη θέση της ουράς, καθώς και το πλήθος των στοιχείων της:

```
cout << "Front: " << Front(myqueue) <<" Size: "<< Size( ) << endl;

-> Front: 10 Size: 5    // Το πρώτο στοιχείο είναι το 10 και το
                        // πλήθος των στοιχείων της ουράς είναι 5
```

Ακολούθως, δίνεται εντολή εξαγωγής, η οποία αφαιρεί το στοιχείο 10 που βρίσκεται στην πρώτη θέση (front). Το στοιχείο στη θέση front παίρνει τιμή μηδέν («αφαιρείται») και ο δείκτης front αυξάνεται κατά 1:

```
Pop(myqueue);
```

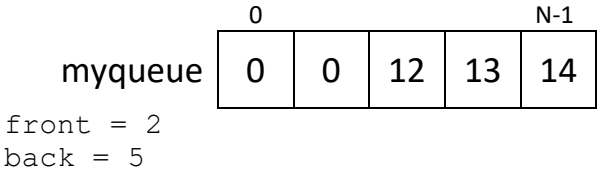


Ακολουθεί εντολή εξόδου, η οποία εμφανίζει το στοιχείο που βρίσκεται στην πρώτη θέση της ουράς, καθώς και το πλήθος των στοιχείων της:

```
cout << "Front: " << Front(myqueue) <<" Size: "<< Size( ) << endl;

-> Front: 11 Size: 4    // Το πρώτο στοιχείο είναι το 11 και το
                        // πλήθος των στοιχείων της ουράς είναι 4
```

Ακολούθως δίνεται εντολή εξαγωγής, η οποία αφαιρεί το στοιχείο 11 που βρίσκεται στην πρώτη θέση (front). Το στοιχείο στη θέση front παίρνει τιμή μηδέν («αφαιρείται») και ο δείκτης front αυξάνεται κατά 1:

```
Pop(myqueue);
```



Ακολουθεί εντολή εξόδου, η οποία εμφανίζει το στοιχείο που βρίσκεται στην πρώτη θέση της ουράς, καθώς και το πλήθος των στοιχείων της:

```
cout << "Front: " << Front(myqueue) <<" Size: "<< Size( ) << endl;

-> Front: 12 Size: 3    // Το πρώτο στοιχείο είναι το 12 και το
                        // πλήθος των στοιχείων της ουράς είναι 3
```

Ακολουθεί μία δομή επανάληψης, η οποία εμφανίζει όλα τα στοιχεία της ουράς:

```
for(int i=front; i<back; i++)
      cout <<  myqueue[i] << " ";

-> 12 13 14             // Τα στοιχεία εμφανίζονται με τη σειρά που
                        // πρόκειται να εξέλθουν: 12 13 14
```

11. Χρήση ουράς μέσω της βιβλιοθήκης STL (Standard Template Library)

Όπως και για τις στοίβες, η πρότυπη βιβλιοθήκη (STL) ορίζει ως περιέχοντα (container) και την ουρά. Οι περιέχοντες, όπως έχει ήδη αναφερθεί, είναι ένας τρόπος οργάνωσης μίας συλλογής αντικειμένων στη μνήμη του υπολογιστή, οι οποίοι περιέχουν ενσωματωμένες τις βασικές λειτουργίες κάθε δομής που υλοποιούν.

Η πιο κάτω δήλωση μάς παρέχει ένα περιέχοντα (container) τύπου ουράς:

```
#include <queue>
```

Με τη χρήση της πιο πάνω επικεφαλίδας, μπορούμε να δηλώσουμε μία ουρά οποιουδήποτε βασικού τύπου και να έχουμε πρόσβαση στις μεθόδους που αποτελούν τις βασικές της λειτουργίες. Υπενθυμίζεται ότι ισχύει η αρχή FIFO (το στοιχείο που θα προστεθεί πρώτο στην ουρά θα αφαιρεθεί πρώτο) και ότι έχουμε πρόσβαση στα στοιχεία που βρίσκονται στην πρώτη και στην τελευταία θέση.

11.1 Δήλωση ουράς

Για να δηλώσουμε μία ουρά, πρέπει να ορίσουμε τον τύπο και το αναγνωριστικό της. Αρχικά, η ουρά που δηλώνουμε έχει μέγεθος 0. Καθώς θα προσθέτουμε στοιχεία, το μέγεθος της ουράς θα προσαρμόζεται ανάλογα με τα στοιχεία που θα περιέχει.

```
queue<int> my_queue;        // δηλώνει μία ουρά ακεραίων
queue<char> Q1;             // δηλώνει μία ουρά χαρακτήρων
queue<string> names         // δηλώνει μία ουρά συμβολοσειρών
```

11.2 Μέθοδοι ουράς

Οι βασικές λειτουργίες της ουράς παρέχονται μέσα από τις ενσωματωμένες συναρτήσεις που προσφέρονται και είναι οι εξής:

push(item)	Με τη συνάρτηση push γίνεται εισαγωγή ενός στοιχείου στο πίσω μέρος (back) της ουράς.
pop( )	Με τη συνάρτηση pop γίνεται εξαγωγή ενός στοιχείου από την πρώτη θέση (front) της ουράς.
front( )	Η συνάρτηση front επιστρέφει το στοιχείο που βρίσκεται στην πρώτη θέση της ουράς, χωρίς να το αφαιρέσει.
back( )	Η συνάρτηση back επιστρέφει το στοιχείο που βρίσκεται στην τελευταία θέση της ουράς, χωρίς να το αφαιρέσει.
empty( )	Η συνάρτηση empty επιστρέφει true, αν η ουρά είναι άδεια, αλλιώς επιστρέφει false.
size( )	Η συνάρτηση size επιστρέφει το πλήθος των στοιχείων της ουράς.

11.3 Κλήση συναρτήσεων

Για να χρησιμοποιήσουμε τις πιο πάνω συναρτήσεις, βάζουμε πρώτα το αναγνωριστικό που δηλώσαμε, ακολουθούμενο από μία τελεία και το όνομα της συνάρτησης την οποία θέλουμε να καλέσουμε.

Ας δούμε παραδείγματα:

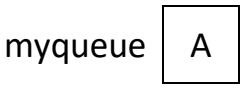
```
#include<iostream>
#include<queue>
using namespace std;
int main(){

 queue<char> myqueue;                  // Δήλωση ουράς χαρακτήρων
 myqueue.push('A');                    // Εισάγουμε το στοιχείο A
 myqueue.push('B');                    // Εισάγουμε το στοιχείο B
 myqueue.push('C');                    // Εισάγουμε το στοιχείο C
 myqueue.push('D');                    // Εισάγουμε το στοιχείο D
 myqueue.pop( );                       // Εξαγωγή πρώτου στοιχείου
 cout << myqueue.front() << endl;      // Πρώτο στοιχείο -> B
 cout << myqueue.back() << endl;       // Τελευταίο στοιχείο -> D
 cout << myqueue.size() << endl;       // Μέγεθος ουράς -> 3
return 0;
}
```

(Code: C7_9_example8.cpp)

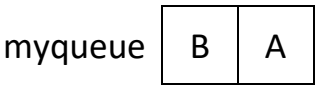
Ακολουθεί μία αναπαράσταση των εντολών του πιο πάνω προγράμματος:

Αρχικά, η εντολή `queue<char> myqueue;` δηλώνει μία ουρά χαρακτήρων με όνομα `myqueue`. Η ουρά, αρχικά, έχει μέγεθος 0.

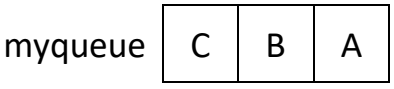
Ακολούθως, η εντολή `myqueue.push('A');` εισάγει στην ουρά το στοιχείο A. Η ουρά έχει τώρα μέγεθος 1.



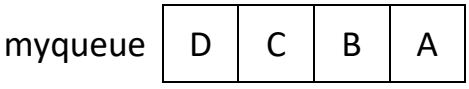
Ακολούθως, η εντολή `myqueue.push('B');` εισάγει στην ουρά το στοιχείο B. Η ουρά έχει τώρα μέγεθος 2. Το στοιχείο B εισέρχεται στην τελευταία θέση (αριστερότερη), ενώ το στοιχείο A βρίσκεται στην πρώτη θέση (δεξιότερη).



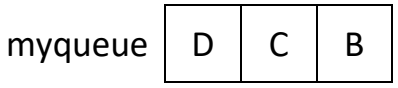
Ακολούθως, η εντολή `myqueue.push('C');` εισάγει στην ουρά το στοιχείο C. Η ουρά έχει τώρα μέγεθος 3. Το στοιχείο C εισέρχεται στην τελευταία θέση (αριστερότερη), ενώ το στοιχείο 'A' βρίσκεται στην πρώτη θέση (δεξιότερη).



Ακολούθως, η εντολή `myqueue.push('D');` εισάγει στην ουρά το στοιχείο D. Η ουρά έχει τώρα μέγεθος 4. Το στοιχείο D εισέρχεται στην τελευταία θέση (αριστερότερη), ενώ το στοιχείο A βρίσκεται στην πρώτη θέση (δεξιότερη).



Ακολούθως, η εντολή `myqueue.pop();` αφαιρεί από την ουρά το στοιχείο Α, το οποίο βρισκόταν στην πρώτη θέση. Η ουρά έχει τώρα μέγεθος 3.



Ακολουθεί η εντολή εξόδου `cout << myqueue.front() << endl;` η οποία εμφανίζει το στοιχείο που βρίσκεται στην πρώτη θέση της ουράς:

```
-> B        // Το στοιχείο στην πρώτη θέση της ουράς είναι το B
```

Ακολουθεί η εντολή εξόδου `cout << myqueue.back() << endl;` η οποία εμφανίζει το στοιχείο που βρίσκεται στην τελευταία θέση της ουράς:

```
-> D        // Το στοιχείο στην τελευταία θέση της ουράς είναι το D
```

Ακολουθεί η εντολή εξόδου `cout << myqueue.size() << endl;` η οποία εμφανίζει το μέγεθος της ουράς:

```
-> 3        // Το πλήθος των στοιχείων της ουράς είναι 3
```

11.4 Εμφάνιση όλων των στοιχείων μίας ουράς

Σε περίπτωση που θέλουμε να εμφανίσουμε όλα τα στοιχεία μία ουράς, μπορούμε να χρησιμοποιήσουμε παρόμοια μέθοδο όπως και στις στοίβες. Μπορούμε να χρησιμοποιήσουμε τη συνάρτηση empty() για τον έλεγχο. Τα στοιχεία της ουράς θα εμφανίζονται με τη ίδια σειρά που είχαν δοθεί.

Δείτε το πιο κάτω παράδειγμα:

```
#include<iostream>
#include<queue>
using namespace std;

int main(){
 queue<int> Q;                    // Δήλωση ουράς ακεραίων

 for (int i=1; i<=5; i++)
     Q.push(i*10);                // Η ουρά περιέχει:{50,40,30,20,10}

 cout << Q.front() << endl;       // 10

 while(!Q.empty()){               // Για όσο η ουρά δεν είναι άδεια
      cout << Q.front() << " ";   // εμφάνισε το μπροστινό στοιχείο
      Q.pop();                    // και αφαίρεσέ το από την ουρά
   }                              // 10 20 30 40 50
return 0;
}
```

(Code: C7_9_example9.cpp)

Παράδειγμα 9.10

Να δημιουργήσετε τον πίνακα προκαταρκτικής εκτέλεσης για το πιο κάτω πρόγραμμα.

```
#include<iostream>
#include<queue>
using namespace std;

int main(){
  queue<char> Q;
  Q.push('a');
  Q.push('b');
  Q.push('c');
  Q.push('A');
  Q.push('B');
  Q.push('C');
  cout << Q.front() << " " << Q.back() << endl;
  Q.pop();
  Q.pop();
  while(!Q.empty()){
      cout << Q.size() << " " << Q.front() << endl;
      Q.pop();
  }
return 0;
}
```

(Code: C7_9_example10.cpp)

Λειτουργία	Ουρά	size	front	back	empty	Παρουσίαση
push(a)	a	1	a	a		
push(b)	b a	2	a	b		
push(c)	c b a	3	a	c		
push(A)	A c b a	4	a	A		
push(B)	B A c b a	5	a	B		
push(C)	C B A c b a	6	a	C		
		6	a	C		a C
pop()	C B A c b	5	b	C		
pop()	C B A c	4	c	C		
		4	c	C	F	4 c
pop()	C B A	3	A	C		
		3	A	C	F	3 A
pop()	C B	2	B	C		
		2	B	C	F	2 B
pop()	C	1	C	C		
		1	C	C	F	1 C
pop()		0			T	

Παράδειγμα 9.11

Δίνονται αγνώστου πλήθους χαρακτήρες από το πληκτρολόγιο, οι οποίοι μπορεί να είναι γράμματα ή αστερίσκοι ('*'). Αν κάθε γράμμα ερμηνεύεται ως μία διαδικασία εισαγωγής (push) και ένας αστερίσκος ως μία διαδικασία εξαγωγής από μία ουρά, να βρείτε ποια θα είναι η τελική σειρά των χαρακτήρων, όταν εφαρμόσουμε τις πιο πάνω διαδικασίες σε μία άδεια αρχικά ουρά.

Προσοχή: μπορεί να δοθούν περισσότερες εντολές εξαγωγής από τα στοιχεία που περιέχει η ουρά, οπόταν προτού γίνει η εξαγωγή, να γίνεται έλεγχος αν η ουρά είναι άδεια.

Προσέγγιση: Διαβάζουμε από το πληκτρολόγιο έναν-έναν τους χαρακτήρες (`while (cin >> ch)`). Για κάθε χαρακτήρα θα καλούμε τη συνάρτηση push, ενώ για κάθε αστερίσκο, αφού βεβαιωθούμε ότι η ουρά δεν είναι άδεια, θα καλούμε τη συνάρτηση pop. Όταν διαβάσουμε όλους χαρακτήρες, εμφανίζουμε τα περιεχόμενα της ουράς.

Σημείωση: για να τερματιστεί η είσοδος των χαρακτήρων, πατάμε τα πλήκτρα CTRL+Z.

```
#include<iostream>
#include<queue>
using namespace std;

int main(){
 queue<char> Q;              // Δήλωση ουράς χαρακτήρων
 char ch;
 while (cin >> ch){
  if (ch=='*')               // Αν διαβάσουμε έναν αστερίσκο
     if (!Q.empty())         // ελέγχουμε για κενή ουρά προτού
       Q.pop();              // να αφαιρέσουμε στοιχείο
  else
       Q.push(ch);           // Εισάγουμε στην ουρά όλους τους
 }                           // υπόλοιπους χαρακτήρες
 while (!Q.empty()){         // Εμφανίζουμε τους χαρακτήρες
     cout << Q.front();      // που έχουν απομείνει στην ουρά
     Q.pop();
 }
return 0;
}
```

(Code: C7_9_example11.cpp)

Παράδειγμα εισόδου

```
A B C * D E F * * G
```

Παράδειγμα εξόδου

```
DEFG
```

Παράδειγμα 9.12

Μία μηχανή προώθησης αναψυκτικών λειτουργεί ως εξής: όταν ο περιπτεράς τοποθετήσει ένα νέο αναψυκτικό, αυτό τοποθετείται στην πρώτη θέση και διατίθεται άμεσα προς κατανάλωση. Το αναψυκτικό, δηλαδή, το οποίο ο περιπτεράς τοποθέτησε πρώτο χρονικά στο μηχάνημα, θα είναι διαθέσιμο προς κατανάλωση τελευταίο. Στο μηχάνημα έχουν τοποθετηθεί Ν αναψυκτικά. Καθώς τα αναψυκτικά προσφέρονται δωρεάν, έχουν κάνει ουρά Μ πελάτες για να τα πάρουν. Οι πελάτες εξυπηρετούνται με τη σειρά που καταφθάνουν στο περίπτερο, αυτός δηλαδή που έφτασε πρώτος, θα εξυπηρετηθεί πρώτος. Να δημιουργήσετε πρόγραμμα, το οποίο να διαβάζει αρχικά το πλήθος (Ν) και τα ονόματα των αναψυκτικών και στη συνέχεια το πλήθος (Μ) και τα ονόματα των πελατών. Ακολούθως, να εμφανίζει με ποια σειρά θα πάρουν τα αναψυκτικά οι πελάτες. Αν ένας πελάτης δεν πάρει αναψυκτικό, ή ένα αναψυκτικό παραμείνει στο μηχάνημα, στη θέση του να εμφανίζεται ο χαρακτήρας 'Χ'.

Παράδειγμα εισόδου

```
4
Fanta Sprite Coke Pepsi
6
Andy Brian Cameron Dave Eddie Freddie
```

Παράδειγμα εξόδου

```
Andy Pepsi
Brian Coke
Cameron Sprite
Dave Fanta
Eddie X
Freddie X
```

Προσέγγιση: Τοποθετούμε τα αναψυκτικά σε μία στοίβα και τους πελάτες σε μία ουρά. Όσο υπάρχουν και στις δύο στοιχεία, αντιστοιχούμε το στοιχείο που βρίσκεται στην κορυφή της στοίβας, με το στοιχείο που βρίσκεται στην πρώτη θέση της ουράς και, στη συνέχεια, αφαιρούμε τα στοιχεία. Αν αδειάσει πρώτα η στοίβα, βάζουμε 'Χ' στη θέση των αναψυκτικών, ενώ αν αδειάσει πρώτα η ουρά, βάζουμε 'Χ' στη θέση των πελατών.

```
#include <iostream>
#include <stack>
#include <queue>
using namespace std;

int main(){
stack<string> drinks;       // Στοίβα με αναψυκτικά
queue<string> people;       // Ουρά με πελάτες

int N, M;
string st;
cin >> N;
while(N--){                 // Είσοδος Ν αναψυκτικών
  cin >> st;
```

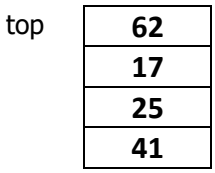
```
  drinks.push(st);
}
cin >> M;
while(M--){                  // Είσοδος Μ πελατών
  cin >> st;
  people.push(st);
 }
while(!people.empty() || !drinks.empty()){
// Υπάρχουν στοιχεία και στη στοίβα και στην ουρά
    if (!people.empty() && !drinks.empty()) {
        cout << people.front() << " " << drinks.top() << endl;
        people.pop();
        drinks.pop();
       }
    else
       if (!people.empty()) {    // Υπάρχουν στοιχεία μόνο στην ουρά
           cout << people.front() << " X" << endl;
           people.pop();
    }
       else {    // Υπάρχουν στοιχεία μόνο στη στοίβα
           cout << "X " << drinks.top() << endl;
           drinks.pop();
       }
 }
return 0;
}
```

(Code: C7_9_example12.cpp)

Ασκήσεις Κεφαλαίου

Άσκηση 9.1

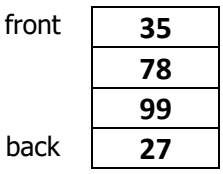
Σας δίνεται η πιο κάτω στοίβα με το όνομα S1. Ποια θα είναι η τελική μορφή της στοίβας, όταν εκτελεστούν με τη σειρά όλες οι πιο κάτω εντολές;




```
S1.push(78);
S1.pop();
S1.pop();
S1.push(36);
S1.push(85);
```

Άσκηση 9.2

Σας δίνεται η πιο κάτω ουρά με το όνομα Q1. Ποια θα είναι η τελική μορφή της ουράς, όταν εκτελεστούν με τη σειρά όλες οι πιο κάτω εντολές;



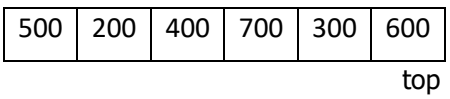

```
Q1.push(54);
Q1.pop();
Q1.push(51);
Q1.pop();
Q1.push(42);
```

Άσκηση 9.3

Δίνεται το παρακάτω τμήμα κώδικα:

```
mystack.pop();
mystack.push(100);
mystack.push(900);
mystack.pop();
mystack.push(800);
```

Αν η στοίβα mystack περιέχει, αρχικά, τα πιο κάτω στοιχεία, ποια θα είναι τα περιεχόμενα της στοίβας μετά την εκτέλεση του παραπάνω τμήματος κώδικα;



Άσκηση 9.4

Δίνεται το παρακάτω τμήμα κώδικα:

```
myqueue.push(10);
myqueue.pop();
myqueue.push(58);
myqueue.pop();
myqueue.push(39);
myqueue.pop();
myqueue.push(91);
```

Αν η ουρά myqueue περιέχει, αρχικά, τα πιο κάτω στοιχεία, ποια θα είναι τα περιεχόμενα της ουράς μετά την εκτέλεση του παραπάνω τμήματος κώδικα;

16	13	14	18	21	19

back front

Άσκηση 9.5

Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται άγνωστο πλήθος ακεραίων αριθμών από το πληκτρολόγιο, να τους αποθηκεύει σε μία στοίβα και να εμφανίζει το πλήθος τους στην οθόνη. Η είσοδος δεδομένων θα τερματίζει όταν δοθεί ο αριθμός 0.

Άσκηση 9.6

Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται άγνωστο πλήθος συμβολοσειρών από το πληκτρολόγιο, να τις αποθηκεύει σε μία ουρά και να εμφανίζει τη συμβολοσειρά που βρίσκεται στην πρώτη θέση (front) της ουράς. Η είσοδος δεδομένων θα τερματίζει όταν δοθεί η συμβολοσειρά «quit».

Άσκηση 9.7

Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται άγνωστο πλήθος χαρακτήρων από το πληκτρολόγιο, να τους αποθηκεύει σε μία στοίβα και να τους εμφανίζει στην οθόνη σε αντίθετη σειρά από αυτήν που δόθηκαν. Η είσοδος δεδομένων θα τερματίζει όταν δοθεί ο χαρακτήρας 'Q'.

Άσκηση 9.8

Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται άγνωστο πλήθος ακεραίων αριθμών από το πληκτρολόγιο, να τους αποθηκεύει σε μία στοίβα και να εμφανίζει τους τρεις τελευταίους αριθμούς σε αντίθετη σειρά από αυτήν που δόθηκαν. Η είσοδος δεδομένων θα τερματίζει όταν δοθεί ο αριθμός 0.

Παράδειγμα εισόδου

```
41 15 26 51 88 39 28 40 0
```

Παράδειγμα εξόδου

```
40 28 39
```

Άσκηση 9.9

Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται άγνωστο πλήθος ακεραίων αριθμών από το πληκτρολόγιο, να τους αποθηκεύει σε μία ουρά και να εμφανίζει τους τρεις πρώτους αριθμούς με τη σειρά που δόθηκαν. Η είσοδος δεδομένων θα τερματίζει όταν δοθεί ο αριθμός 0.

Παράδειγμα εισόδου

```
41 15 26 51 88 12 7 39 28 40 0
```

Παράδειγμα εξόδου

```
41 15 26
```

Άσκηση 9.10

Να δημιουργήσετε πρόγραμμα, το οποίο να διαβάζει ονόματα από το πληκτρολόγιο μέχρι να δοθεί ως όνομα η λέξη «stop». Τα ονόματα θα τοποθετηθούν με τη σειρά που δίνονται σε μία ουρά, για να εξυπηρετηθούν σε ένα ταμείο. Το πρόγραμμα να εμφανίζει το όνομα του ατόμου που θα εξυπηρετηθεί πρώτο και το όνομα του ατόμου που θα εξυπηρετηθεί τελευταίο.

Άσκηση 9.11

Να δημιουργήσετε πρόγραμμα, το οποίο να διαβάζει ακέραιους αριθμούς από το πληκτρολόγιο, μέχρι να δοθεί ο αριθμός -1. Οι αριθμοί θα τοποθετηθούν με τη σειρά που δίνονται σε μία στοίβα. Ακολούθως, τα περιεχόμενα της στοίβας να αφαιρούνται ένα-ένα και μόνο οι μονοψήφιοι αριθμοί να καταχωρίζονται σε μία ουρά, μέχρι η στοίβα να αδειάσει. Το πρόγραμμα να εμφανίζει το τελικό πλήθος των στοιχείων της ουράς.

Άσκηση 9.12

Να δημιουργήσετε τον πίνακα προκαταρκτικής εκτέλεσης για το πιο κάτω πρόγραμμα.

```
#include <iostream>
#include <stack>
using namespace std;

int main(){
  stack <int> S;
  S.push(10);
  S.push(20);
  S.push(30);
  S.pop();
  S.push(40);
  S.push(50);
  S.pop();
  S.push(60);
  cout << S.top() << endl;
  cout << S.size() << endl;

  while(!S.empty()) {
      cout << S.top() << " ";
      S.pop();
   }
return 0;
}
```

Άσκηση 9.13

Να δημιουργήσετε τον πίνακα προκαταρκτικής εκτέλεσης για το πιο κάτω πρόγραμμα.

```
#include <iostream>
#include <queue>
using namespace std;

int main(){
  queue <double> Q;
  Q.push(34.62);
  Q.push(53.78);
  Q.push(66.55);
  Q.pop();
  Q.push(45.23);
  Q.push(36.98);
  Q.push(13.76);
  cout << Q.front() << endl;
  cout << Q.back() << endl;
  Q.pop();

  while(!Q.empty()) {
      cout << Q.size() << " ";
      cout << Q.front() << endl;
      Q.pop();
   }

return 0;
}
```

Άσκηση 9.14

Ένας θετικός ακέραιος αριθμός σημαίνει push (εισαγωγή στοιχείου), ένας αρνητικός ακέραιος αριθμός σημαίνει pop (εξαγωγή στοιχείου) και ο αριθμός μηδέν σηματοδοτεί το τέλος των δεδομένων εισόδου. Να βρείτε ποια θα είναι η τελική σειρά των στοιχείων, όταν καταχωρίσουμε μία ακολουθία αριθμών σε μία άδεια αρχικά στοίβα. Να θεωρήσετε ότι δεν θα δοθεί αρνητικός αριθμός, αν η στοίβα είναι άδεια.

Παράδειγμα εισόδου

```
4 5 6 -1 8 9 -2 0
```

Παράδειγμα εξόδου

```
8 5 4
```

Άσκηση 9.15

Ένας θετικός ακέραιος αριθμός σημαίνει push (εισαγωγή στοιχείου), ένας αρνητικός ακέραιος αριθμός σημαίνει pop (εξαγωγή στοιχείου) και ο αριθμός μηδέν σηματοδοτεί το τέλος των δεδομένων εισόδου. Να βρείτε ποια θα είναι η τελική σειρά των στοιχείων, όταν καταχωρίσουμε μία ακολουθία αριθμών σε μία άδεια αρχικά ουρά. Να θεωρήσετε ότι μπορεί να δοθούν αρνητικοί αριθμοί, όταν η ουρά είναι άδεια, οπόταν θα χρειαστεί να κάνετε έλεγχο πριν να γίνει εξαγωγή στοιχείου.

Παράδειγμα εισόδου

```
4 -1 -3 -2 3 5 6 -1 8 9 -2 0
```

Παράδειγμα εξόδου

```
6 8 9
```

Άσκηση 9.16

Ένας υπόγειος χώρος στάθμευσης έχει τη μορφή ενός στενού διαδρόμου και διαθέτει μία μόνο είσοδο. Κάθε φορά που εισέρχεται ένα αυτοκίνητο, αυτό τοποθετείται στο τέλος του διαδρόμου και το επόμενο αυτοκίνητο τοποθετείται ακριβώς πίσω του. Για να μπορεί, δηλαδή, να εξέλθει το αυτοκίνητο που στάθμευσε πρώτο, πρέπει να εξέλθουν όλα τα αυτοκίνητα που μπήκαν στον χώρο στάθμευσης μετά από αυτό. Για κάθε αυτοκίνητο που θέλει να εισέλθει στον χώρο στάθμευσης θα εμφανίζεται η λέξη «enter» και, ακολούθως, τα νούμερα εγγραφής του αυτοκινήτου. Για κάθε αυτοκίνητο που θα εξέρχεται θα εμφανίζεται η λέξη «exit». Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται, αρχικά, έναν ακέραιο Ν, που υποδεικνύει το πλήθος των εντολών που θα ακολουθήσουν. Σε κάθε εντολή, ένα αυτοκίνητο είτε θα εισέρχεται (enter) είτε θα εξέρχεται (exit) από τον χώρο στάθμευσης.

Το πρόγραμμα να εμφανίζει τα νούμερα εγγραφής των αυτοκινήτων που θα βρίσκονται στον χώρο στάθμευσης, αφού εκτελεστούν όλες οι εντολές, με τη σειρά που αυτά θα εξέλθουν από τον χώρο στάθμευσης. Να θεωρήσετε ότι δεν θα δοθεί εντολή «exit», αν δεν υπάρχουν αυτοκίνητα στον χώρο στάθμευσης.

Παράδειγμα εισόδου

```
8
enter EEM330
enter KAP161
enter KMB633
exit
enter LAF501
enter KWG145
exit
enter LAB211
```

Παράδειγμα εξόδου

```
LAB211
LAF501
KAP161
EEM330
```

Άσκηση 9.17

Οι μαθητές ενός σχολείου στέκονται υπομονετικά σε μία ουρά, για να εξυπηρετηθούν. Ο πρώτος μαθητής που εισέρχεται στην ουρά θα είναι και ο πρώτος που θα εξυπηρετηθεί. Η εντολή E A1 3456 σημαίνει ότι ο μαθητής του A1 με αριθμό μητρώου 3456 εισέρχεται στην ουρά, ενώ η εντολή D σημαίνει ότι ο μαθητής που βρίσκεται στην πρώτη θέση της ουράς έχει εξυπηρετηθεί και φεύγει. Να δημιουργήσετε πρόγραμμα, το οποίο να δέχεται έναν ακέραιο N, που υποδεικνύει το πλήθος των εντολών που θα ακολουθήσουν. Για κάθε εντολή τύπου D να εμφανίζεται το τμήμα και ο αριθμός μητρώου του μαθητή που έχει εξυπηρετηθεί. Να θεωρήσετε ότι δεν θα δοθεί εντολή D, αν δεν υπάρχουν μαθητές στην ουρά.

Σημείωση: Να δηλωθεί μία εγγραφή student με μέλη τα στοιχεία του μαθητή και να υλοποιηθεί μία ουρά τύπου student.

Παράδειγμα εισόδου

```
10
E A1 3456
E A2 3567
E A3 3678
D
E B3 4567
D
E C4 8767
E C4 8887
D
D
```

Παράδειγμα εξόδου

```
A1 3456
A2 3567
A3 3678
B3 4567
```

Άσκηση 9.18

Να υλοποιήσετε τις λειτουργίες μίας ουράς ακέραιων αριθμών ως εξής:

- Η ουρά θα δέχεται αριθμούς στο πεδίο τιμών [1…9999]. Αν δοθεί οποιοσδήποτε άλλος αριθμός, δεν θα καταχωρείται στην ουρά.
- Αν δοθεί εντολή εξόδου και ο αριθμός που βρίσκεται στην μπροστινή θέση της ουράς είναι διψήφιος ή τετραψήφιος, τότε θα εξέρχεται από την ουρά και θα εμφανίζεται στην οθόνη.
- Αν δοθεί εντολή εξόδου και ο αριθμός που βρίσκεται στην μπροστινή θέση της ουράς είναι μονοψήφιος ή τριψήφιος θα εξέρχεται από το μπροστινό μέρος της ουράς και θα εισέρχεται ξανά από το πίσω μέρος, χωρίς να εμφανίζεται στην οθόνη.

Να δημιουργήσετε πρόγραμμα, το οποίο να διαβάζει αρχικά έναν ακέραιο N, που υποδεικνύει το πλήθος των εντολών που θα ακολουθήσουν. Οι εντολές θα είναι δύο ειδών: E x για να καταχωριστεί ο ακέραιος x στην ουρά και D για να εξέλθει ο ακέραιος αριθμός που έχει σειρά από την ουρά. Για κάθε μία από τις εντολές τύπου D το πρόγραμμα θα εμφανίζει τον ακέραιο

αριθμό που εξέρχεται από την ουρά, αν αυτό είναι δυνατόν. Να θεωρήσετε ότι δεν θα δοθεί εντολή D, αν δεν υπάρχουν ακέραιοι στην ουρά.

Παράδειγμα εισόδου

```
10
E 345
E 5
D
E 5678
E 23
D
D
E 234
D
D
```

Παράδειγμα εξόδου

```
5678
23
```

Επεξήγηση: Η εντολή E 345 προσθέτει το στοιχείο 345 στην ουρά (345). Η εντολή E 5 προσθέτει το στοιχείο 5 στην ουρά (5, 345). Η εντολή D αφαιρεί το στοιχείο 345 από την ουρά και το προσθέτει στο τέλος της, χωρίς να το εμφανίσει στην οθόνη (345, 5). Η εντολή E 5678 προσθέτει το στοιχείο 5678 στην ουρά (5678, 345, 5). Η εντολή E 23 προσθέτει το στοιχείο 23 στην ουρά (23, 5678, 345, 5). Η εντολή D αφαιρεί το στοιχείο 5 από την ουρά και το προσθέτει στο τέλος της, χωρίς να το εμφανίσει στην οθόνη (5, 23, 5678, 345). Η εντολή D αφαιρεί το στοιχείο 345 από την ουρά και το προσθέτει στο τέλος της, χωρίς να το εμφανίσει στην οθόνη (345, 5, 23, 5678). Η εντολή E 234 προσθέτει το στοιχείο 234 στην ουρά (234, 345, 5, 23, 5678). Η εντολή D αφαιρεί το στοιχείο 5678 από την ουρά και το εμφανίζει στην οθόνη (234, 345, 5, 23). Η εντολή D αφαιρεί το στοιχείο 23 από την ουρά και το εμφανίζει στην οθόνη. Στην ουρά παραμένουν οι αριθμοί 234, 345 και 5.

Άσκηση 9.19

Να υλοποιήσετε τις λειτουργίες μίας στοίβας ακεραίων ως εξής:

- Ένα στοιχείο (item1) μπορεί να αφαιρεθεί από την κορυφή της στοίβας μόνο αν είναι μεγαλύτερο από το στοιχείο που το ακολουθεί (item2). Σε τέτοια περίπτωση το στοιχείο της κορυφής (item1) αφαιρείται και το δεύτερο στοιχείο παίρνει τη θέση του στην κορυφή.
- Σε περίπτωση που το στοιχείο της κορυφής είναι μικρότερο από το στοιχείο που το ακολουθεί, τότε θα αφαιρεθεί πρώτο το δεύτερο στοιχείο. Η διαδικασία για να επιτευχθεί αυτό έχει ως ακολούθως: Το στοιχείο κορυφής εξέρχεται της στοίβας και αποθηκεύεται προσωρινά. Το δεύτερο στοιχείο γίνεται στοιχείο κορυφής και εξέρχεται της στοίβας. Το πρώτο στοιχείο (item1) εισέρχεται στη στοίβα ξανά και θα γίνει ξανά νέος έλεγχος με το επόμενο δεύτερο στοιχείο.

Να δημιουργήσετε πρόγραμμα, το οποίο να διαβάζει αρχικά έναν ακέραιο N, που υποδεικνύει το πλήθος των εντολών που θα ακολουθήσουν. Οι εντολές θα είναι δύο ειδών: E x για να καταχωριστεί ο ακέραιος x στη στοίβα και D για να εξέλθει ο ακέραιος που έχει σειρά από τη στοίβα. Για κάθε μία από τις εντολές τύπου D το πρόγραμμα θα εμφανίζει τον ακέραιο που

εξέρχεται από τη στοίβα. Να θεωρήσετε ότι δεν θα δοθεί εντολή D, αν δεν υπάρχουν ακέραιοι στη στοίβα.

Παράδειγμα εισόδου

```
10
E 7
E 5
D
E 8
E 2
D
D
E 4
D
D
```

Παράδειγμα εξόδου

```
7
8
5
4
2
```

Άσκηση 9.20

Να υλοποιήσετε τις λειτουργίες μίας ουράς ακέραιων αριθμών, χρησιμοποιώντας δύο στοίβες ως εξής:

- Η πρώτη στοίβα (input) θα χρησιμοποιηθεί για τη διαδικασία εισόδου. Κάθε στοιχείο που θα εισέρχεται, θα εισαχθεί στη στοίβα αυτή.
- Η δεύτερη στοίβα (output) θα χρησιμοποιηθεί ως στοίβα εξόδου. Κάθε στοιχείο που θα εξέλθει θα το κάνει από αυτή τη στοίβα.
- Αν δοθεί εντολή εισόδου, θα εκτελείται διαδικασία εισόδου στην πρώτη στοίβα.
- Αν δοθεί εντολή εξόδου, πρέπει να γίνεται ο εξής έλεγχος:
    - Αν και οι δύο στοίβες είναι κενές, να εμφανίζεται το μήνυμα «Nothing to dequeue».
    - Αν μόνο η δεύτερη στοίβα είναι κενή, όλα τα στοιχεία της πρώτης στοίβας θα προστίθενται στη δεύτερη στοίβα. Η πρώτη στοίβα θα αδειάζει.
    - Αν η δεύτερη στοίβα δεν είναι κενή, θα εκτελείται διαδικασία εξόδου.

Να δημιουργήσετε πρόγραμμα, το οποίο να διαβάζει αρχικά έναν ακέραιο αριθμό N, που υποδεικνύει το πλήθος των εντολών που θα ακολουθήσουν. Οι εντολές θα είναι δύο ειδών: `enqueue x` για να καταχωρηθεί ο ακέραιος x στην «ουρά» και `dequeue` για να εξέλθει ο ακέραιος που έχει σειρά από την «ουρά». Για κάθε μία από τις εντολές το πρόγραμμα θα εμφανίζει τα μηνύματα «Item x enters» ή «Item x exits», αντίστοιχα.

Παράδειγμα εισόδου

```
9
enqueue 40
dequeue
```

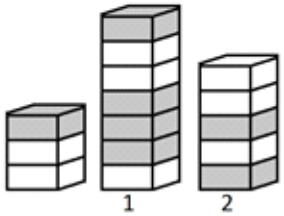
```
enqueue 30
enqueue 20
dequeue
enqueue 10
dequeue
dequeue
dequeue
```

Παράδειγμα εξόδου

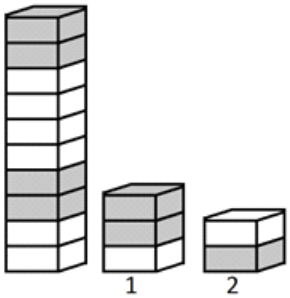
```
Item 40 enters
Item 40 exits
Item 30 enters
Item 20 enters
Item 30 exits
Item 10 enters
Item 20 exits
Item 10 exits
Nothing to dequeue
```

Άσκηση 9.21

Ένα ρομπότ, το οποίο αποτελείται από έναν μηχανικό βραχίονα, μπορεί να πάρει ένα κιβώτιο από την αριστερότερη στοίβα και να το τοποθετήσει είτε στη στοίβα 1 είτε στη στοίβα 2. Μπορούμε να προγραμματίσουμε το ρομπότ, δίνοντάς του μία ακολουθία αριθμών, η οποία να αποτελείται από τους αριθμούς 1 και 2 μόνο. Κάθε αριθμός δηλώνει σε ποια στοίβα πρέπει να τοποθετηθεί το κιβώτιο. Για παράδειγμα, το ρομπότ μόλις εκτέλεσε το ακόλουθο πρόγραμμα: 2 1 2 1 1 2 1 και οι στοίβες έχουν την παρακάτω μορφή:



που σημαίνει ότι οι αρχικές θέσεις των κιβωτίων ήταν οι εξής:



Δεδομένα εισόδου

- Ένας ακέραιος N, το πλήθος των αριθμών του προγράμματος.
- Μία ακολουθία από N αριθμούς (1 ή 2), το πρόγραμμα που εκτέλεσε το ρομπότ
- Μία συμβολοσειρά με τους χαρακτήρες 'B' (μαύρο κιβώτιο) και 'W' (άσπρο κιβώτιο), η τελική μορφή της αρχικής στοίβας

- Μία συμβολοσειρά με τους χαρακτήρες 'B' και 'W', η τελική μορφή της στοίβας 1
- Μία συμβολοσειρά με τους χαρακτήρες 'B' και 'W', η τελική μορφή της στοίβας 2

Δεδομένα εξόδου

Τρεις συμβολοσειρές με τους χαρακτήρες 'B' και 'W', η αρχική διάταξη των κιβωτίων.

Παράδειγμα εισόδου

```
7
2 1 2 1 1 2 1
WWB
WBBBWWB
BWBWW
```

Παράδειγμα εξόδου

```
BBWWWWBBWW
BBW
WB
```

Άσκηση 9.22

Η Αριάδνη έχει πάρει το πλοίο για να πάει να συναντήσει τον φίλο της τον Κυριάκο. Για να περάσει την ώρα της στη διαδρομή, παρατηρεί ποια αυτοκίνητα μπαίνουν και πόσα αυτοκίνητα φεύγουν σε κάθε λιμάνι που κάνει στάση το πλοίο. Αν στο συγκεκριμένο πλοίο το πρώτο αυτοκίνητο που βγαίνει κάθε φορά είναι αυτό που μπήκε τελευταίο, να βοηθήσετε την Αριάδνη να βρει ποια αυτοκίνητα θα είναι στο πλοίο, όταν αυτό θα φτάσει στον προορισμό του.

Δεδομένα εισόδου

- Η πρώτη γραμμή περιέχει έναν αριθμό N, το πλήθος των λιμανιών που συναντά η Αριάδνη πριν από τον τελικό της προορισμό.
- Κάθε μία από τις N επόμενες γραμμές αντιστοιχεί σε κάθε λιμάνι.
- Για κάθε λιμάνι θα υπάρχουν 2 ακέραιοι, ο αριθμός των αυτοκινήτων που βγήκαν από το πλοίο (X) και ο αριθμός των αυτοκινήτων που μπήκαν στο πλοίο (Y). Ακολουθούν Y λέξεις, που αντιστοιχούν στο μοντέλο κάθε αυτοκινήτου που μπαίνει στο πλοίο, με τη σειρά που αυτά μπήκαν.

Δεδομένα εισόδου

Θα εμφανίζεται μία λίστα με τα αυτοκίνητα που υπάρχουν στο πλοίο, με πρώτο αυτό που μπήκε τελευταίο. Θα εμφανίζεται ένα αυτοκίνητο σε κάθε γραμμή.

Παράδειγμα εισόδου

```
3
0 4 Audi Ford Fiat Toyota
2 3 BMW Suzuki Honda
4 2 Mercedes Citroen
```

Παράδειγμα εξόδου

```
Citroen
Mercedes
Audi
```

Επεξήγηση: Στο πρώτο λιμάνι μπαίνουν τα αυτοκίνητα Audi, Ford, Fiat, Toyota. Στο δεύτερο λιμάνι κατεβαίνουν τα Fiat και Toyota και ανεβαίνουν τα BMW, Suzuki και Honda. Στο τρίτο και τελευταίο λιμάνι κατεβαίνουν τα Ford, BMW, Suzuki και Honda και ανεβαίνουν τα Mercedes και Citroen. Άρα, στο πλοίο έμειναν τα αυτοκίνητα Audi, Mercedes και Citroen.

Άσκηση 9.23

Μία αερογραμμή τοποθετεί τους επιβάτες της πρώτης θέσης και τους επιβάτες της οικονομικής θέσης σε δύο ουρές, για να εξυπηρετηθούν το συντομότερο δυνατό. Όταν καταφθάνουν οι επιβάτες, δίνουν το εισιτήριό τους, στο οποίο αναγράφονται είτε ο χαρακτήρας 'Β' για τους επιβάτες της πρώτης θέσης είτε ο χαρακτήρας 'Ε' για τους επιβάτες της οικονομικής θέσης. Μαζί με τον χαρακτήρα αναγράφεται και ο κωδικός εισιτηρίου του επιβάτη.

Καθώς υπάρχει μόνο ένα σημείο για παράδοση αποσκευών, για κάθε τρεις επιβάτες της πρώτης θέσης η υπάλληλος εξυπηρετεί έναν επιβάτη της οικονομικής θέσης. Όταν όλοι οι επιβάτες της πρώτης θέσης εξυπηρετηθούν, τότε μπορούν να εξυπηρετηθούν συνεχόμενα οι επιβάτες της οικονομικής θέσης. Ο χρόνος εξυπηρέτησης για κάθε επιβάτη της πρώτης θέσης είναι δύο λεπτά και για κάθε επιβάτη της οικονομικής θέσης είναι πέντε λεπτά.

Να δημιουργήσετε πρόγραμμα, το οποίο να διαβάζει, αρχικά, το πλήθος των επιβατών (N) και, ακολούθως, τον χαρακτήρα και τον κωδικό εισιτηρίου του κάθε επιβάτη. Με βάση τον χαρακτήρα (Β ή Ε), θα τοποθετούνται στην κατάλληλη ουρά, ώστε αυτός που μπήκε πρώτος να εξυπηρετηθεί πρώτος. Το πρόγραμμα να εμφανίζει τους κωδικούς εισιτηρίων των επιβατών με τη σειρά που εξυπηρετήθηκαν και τον συνολικό χρόνο που θα χρειαστεί για να εξυπηρετηθούν όλοι οι επιβάτες.

Παράδειγμα εισόδου

```
7
B 101
E 102
B 113
B 123
E 134
B 178
B 290
```

Παράδειγμα εξόδου

```
101 113 123 102 178 290 134
20
```

Επεξήγηση: Εξυπηρετούνται πρώτα οι επιβάτες με κωδικό 101, 112, 123 της πρώτης θέσης. Μετά, ο επιβάτης με κωδικό 102 της οικονομικής και οι επιβάτες με κωδικούς 178 και 290 της πρώτης θέσης. Τελευταίος ο επιβάτης με κωδικό 124 της οικονομικής. Συνολικός χρόνος εξυπηρέτησης: (5*2)+(2*5)=20 λεπτά.

Άσκηση 9.24

Ο Παντελής παίζει με τους φίλους του το γνωστό παιχνίδι «Πάρε-Δώσε». Το παιχνίδι αυτό παίζεται με μία μπάλα και οι κανόνες είναι απλοί. Τα παιδιά στέκονται σε κύκλο και αρχικά ένας παίκτης κρατά την μπάλα. Κάθε παίκτης μπορεί είτε να επιστρέψει την μπάλα σε αυτόν που του πάσαρε (B) είτε να πασάρει σε έναν οποιονδήποτε άλλο παίκτη μέσα στον κύκλο (P).

Για παράδειγμα, ας υποθέσουμε ότι κρατά την μπάλα ο παίκτης με τον αριθμό 5.

- `P 23` – Ο παίκτης με αριθμό 5 πασάρει στον παίκτη με αριθμό 23
- `P 4` – Ο παίκτης με αριθμό 23 πασάρει στον παίκτη με αριθμό 4
- `B` – Ο παίκτης με αριθμό 4 πασάρει πίσω στον παίκτη με αριθμό 23
- `P 19` – Ο παίκτης με αριθμό 23 πασάρει στον παίκτη με αριθμό 19
- `B` – Ο παίκτης με αριθμό 19 πασάρει πίσω στον παίκτη με αριθμό 23

Άρα ο παίκτης που έχει στην κατοχή του την μπάλα, μετά από πέντε πάσες, θα είναι ο παίκτης με τον αριθμό 23.

Δεδομένα εισόδου

- Η πρώτη γραμμή περιέχει δύο θετικούς ακεραίους. Το N, οι συνολικές πάσες και P, ο παίκτης που έχει αρχικά την μπάλα στην κατοχή του.
- Στις επόμενες N γραμμές θα εμφανίζεται το είδος της πάσας ως εξής:
  - P id, ο παίκτης που κρατά την μπάλα πασάρει στον παίκτη με αριθμό id.
  - B, ο παίκτης που κρατά την μπάλα την επιστρέφει σε αυτόν που του πάσαρε.

Δεδομένα εισόδου

Θα εμφανίζεται ο αριθμός του παίκτη που έχει την μπάλα στην κατοχή του, μετά την τελευταία πάσα.

Παράδειγμα εισόδου

```
5 17
P 19
P 5
P 10
B
B
```

Παράδειγμα εξόδου

```
10
```

Επεξήγηση: Ο παίκτης με αριθμό 17 έχει αρχικά στην κατοχή του την μπάλα και πασάρει στον παίκτη με αριθμό 19 (P 19). Αυτός πασάρει στον παίκτη με αριθμό 5 (P 5), ο οποίος πασάρει στον παίκτη με αριθμό 10 (P 10). Ο παίκτης με αριθμό 10 επιστρέφει την μπάλα πίσω στον παίκτη με αριθμό 5 (B) και αυτός την επιστρέφει ξανά στον παίκτη με αριθμό 10 (B). Άρα, μετά από πέντε πάσες, ο παίκτης που θα έχει στην κατοχή του την μπάλα θα είναι αυτός με τον αριθμό 10.

Ασκήσεις Εμπλουτισμού

Άσκηση 9.25

Ο διακομιστής μίας εταιρείας όταν παραλάβει μία ακολουθία από διαδικασίες, επιλέγει ο ίδιος την ιδανική σειρά εκτέλεσής τους, με στόχο τη διεκπεραίωση όλων των διαδικασιών στον ελάχιστο δυνατό χρόνο. Ο διακομιστής μπορεί να εκτελεί μόνο μία διαδικασία κάθε φορά και αυτή που εκτελείται είναι η διαδικασία που βρίσκεται στην πρώτη θέση της ουράς αναμονής.

Για παράδειγμα: Έχουμε 3 διαδικασίες με αριθμούς 1, 2, 3 οι οποίες έχουν την εξής σειρά παραλαβής: 3, 2, 1. Ο διακομιστής επιλέγει να τις εκτελέσει με την εξής σειρά: 1, 3, 2. Κάθε διεκπεραίωση και κάθε μετακίνηση διαδικασίας παίρνει ένα δευτερόλεπτο. Άρα θα έχουμε τις πιο κάτω εναλλαγές:

- `3 2 1` - η αρχική ουρά με τις διαδικασίες προς εκτέλεση. Ιδανική σειρά: 1, 3, 2.
- `2 1 3` - η διαδικασία 3 αφαιρείται από την αρχή και εισέρχεται στο τέλος (1 δ.)
- `1 3 2` - η διαδικασία 2 αφαιρείται από την αρχή και εισέρχεται στο τέλος (1 δ.)
- `3 2` - Η διαδικασία 1 εκτελείται (1 δ.)
- `2` - Η διαδικασία 3 εκτελείται (1 δ.)
- Η διαδικασία 2 εκτελείται (1 δ.). Η ουρά είναι πλέον άδεια.

Ο συνολικός χρόνος εκτέλεσης θα είναι 5 δευτερόλεπτα.

Δεδομένα εισόδου

- Στην πρώτη γραμμή θα εμφανίζεται ένας ακέραιος Ν, το πλήθος των διαδικασιών.
- Στη δεύτερη γραμμή θα εμφανίζεται η σειρά παραλαβής των διαδικασιών.
- Στην τρίτη γραμμή θα εμφανίζεται η ιδανική σειρά εκτέλεσης των διαδικασιών.

Δεδομένα εισόδου

Ο ελάχιστος, συνολικός χρόνος εκτέλεσης όλων των διαδικασιών.

Παράδειγμα εισόδου

```
3
3 2 1
1 3 2
```

Παράδειγμα εξόδου

```
5
```

Άσκηση 9.26

Στο εστιατόριο που δουλεύει ο Παντελής, οι πελάτες εξυπηρετούνται μόνοι τους με έναν ιδιαίτερο τρόπο. Ο πελάτης που στέκεται πρώτος στην ουρά μπορεί να πάρει το πιάτο που έχει ετοιμαστεί από τον σεφ και να πληρώσει το αντίστοιχο ποσό, αλλιώς, αν δεν υπάρχει έτοιμο φαγητό, θα χάσει τη σειρά του. Τα πιάτα που ετοιμάζονται από τον σεφ μπαίνουν σε μία σειρά και ο πελάτης μπορεί να πάρει μόνο αυτό που βρίσκεται στην πρώτη θέση της σειράς των φαγητών, το οποίο είναι αυτό που ετοιμάστηκε πρώτο. Να δημιουργήσετε πρόγραμμα, το οποίο να εμφανίζει το ποσό που πρέπει να πληρώσει ο κάθε πελάτης που εξυπηρετήθηκε, ή αν ο πελάτης δεν πήρε φαγητό να εμφανίζεται η φράση «no service».

Δεδομένα εισόδου

- Στην πρώτη γραμμή εμφανίζεται ένας ακέραιος αριθμός N, ο αριθμός των γραμμών που ακολουθούν.
- Στις επόμενες N γραμμές θα εμφανίζονται είτε:
  - ο αριθμός 1, που υποδεικνύει ότι έχει φτάσει νέος πελάτης και στέκεται στην ουρά είτε
  - ο αριθμός 2, ακολουθούμενος από το κόστος K του φαγητού σε ευρώ, αν ο σεφ έχει ετοιμάσει ένα πιάτο.

Δεδομένα εισόδου

Για κάθε πελάτη που εξυπηρετήθηκε θα εμφανίζεται το ποσό που πρέπει να πληρώσει, με δύο δεκαδικά ψηφία, ή αν δεν έχει πάρει φαγητό, θα εμφανίζεται η φράση «no service».

Παράδειγμα εισόδου

```
6
1
2 5.50
2 7.75
2 9.99
1
1
```

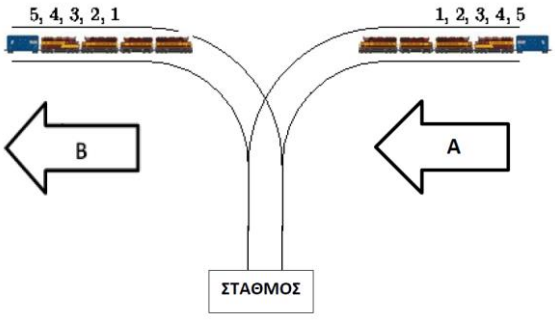
Παράδειγμα εξόδου

```
no service
5.50
7.75
```

Επεξήγηση: Ο πρώτος πελάτης που καταφθάνει δεν θα εξυπηρετηθεί, γιατί δεν υπάρχει έτοιμο φαγητό, άρα θα χάσει τη σειρά του. Ο σεφ τοποθετεί τρία φαγητά στη σειρά, τα οποία κοστίζουν 5.50, 7.75 και 9.99 ευρώ, αντίστοιχα. Ο δεύτερος πελάτης που καταφθάνει θα πληρώσει 5.00 ευρώ, όσα στοιχίζει το πιάτο που βρίσκεται στην πρώτη θέση. Στην πρώτη θέση, τώρα, θα μετακινηθεί το πιάτο που στοιχίζει 7.75 ευρώ. Ο τρίτος πελάτης που καταφθάνει θα πληρώσει 7.75 ευρώ, όσα στοιχίζει το πιάτο που βρίσκεται στην πρώτη θέση.

Άσκηση 9.27

Ο Παντελής είναι σταθμάρχης ενός πολύ παλιού σταθμού τρένου. Όπως βλέπετε στην πιο κάτω εικόνα, υπάρχει μόνο μία σιδηροδρομική γραμμή για να εισέλθει ένα τρένο στον σταθμό.



Ένα τρένο που εισέρχεται στον σταθμό από τη γραμμή Α πρέπει να εξέλθει από τη γραμμή Β. Κάθε βαγόνι του τρένου μπορεί να αποκολληθεί από την αμαξοστοιχία προτού εισέλθει στον σταθμό και να εξέλθει από τη γραμμή Β. Ένα βαγόνι που μπαίνει στον σταθμό δεν μπορεί να επιστρέψει πίσω στη γραμμή Α και όταν βγει από τον σταθμό, μέσω της γραμμής Β, δεν μπορεί να επιστρέψει πίσω στον σταθμό. Κάθε τρένο μπορεί να έχει μέχρι 100 βαγόνια, τα οποία εισέρχονται στον σταθμό σε αύξουσα σειρά 1, 2, 3,…N. Οι μηχανοδηγοί κάποιες φορές ζητούν από τον Παντελή να αλλάξει τη σειρά των βαγονιών, προτού αυτά εισέλθουν στον σταθμό. Ο Παντελής θέλει να ξέρει αν είναι εφικτή μία τέτοια αλλαγή. Αν γνωρίζετε το πλήθος των βαγονιών για κάθε τρένο και τη διάταξη (ή τις διατάξεις) με την οποία επιθυμεί ο μηχανοδηγός να εξέλθει το τρένο από τον σταθμό, να βρείτε αν είναι εφικτή η κάθε διάταξη.

Δεδομένα εισόδου

- Ένας ακέραιος αριθμός N, το πλήθος των βαγονιών.
- Ακολουθεί άγνωστος αριθμός Χ διατάξεων που επιθυμούν οι μηχανοδηγοί.
- Ο αριθμός 0 που υποδεικνύει το τέλος των δεδομένων εισόδου.

Δεδομένα εισόδου

Χ γραμμές με τη λέξη «Yes», αν είναι εφικτή η αλλαγή ή «No», αν δεν είναι.

Παράδειγμα εισόδου

```
5
1 2 3 5 4
5 4 3 2 1
5 4 1 2 3
0
```

Παράδειγμα εξόδου

```
Yes
Yes
No
```

Επεξήγηση: Το τρένο στη γραμμή Α έχει την εξής σειρά βαγονιών: 1, 2, 3, 4, 5. Η πρώτη διάταξη που θέλει ο μηχανοδηγός είναι η εξής: 1, 2, 3, 5, 4. Το βαγόνι 1 αποκολλάται από το υπόλοιπο τρένο, μπαίνει στον σταθμό και βγαίνει στη γραμμή Β. Ακολούθως, το βαγόνι 2 κάνει το ίδιο πράγμα και ακολουθεί το βαγόνι 3. Το βαγόνι 4 θα μείνει μέσα στον σταθμό, επιτρέποντας στο βαγόνι 5 να εισέλθει και να εξέλθει από τον σταθμό. Άρα, η πρώτη διάταξη είναι εφικτή.

Η δεύτερη διάταξη είναι η εξής: 5, 4, 3, 2, 1. Τα βαγόνια εισέρχονται όλα στον σταθμό και εξέρχεται πρώτο αυτό που μπήκε τελευταίο (βαγόνι 5). Ακολουθεί το βαγόνι 4, μετά το βαγόνι 3 κ.ο.κ., άρα και η δεύτερη διάταξη είναι εφικτή.

Η τρίτη διάταξη είναι η εξής: 5, 4, 1, 2, 3. Το βαγόνι 1 αποκολλάται από το υπόλοιπο τρένο και εισέρχεται στον σταθμό. Ακολούθως, το βαγόνι 2, μετά το 3, το 4 και τέλος το 5. Το βαγόνι 5 εξέρχεται πρώτο (αφού μπήκε τελευταίο), ακολουθεί το βαγόνι 4, αλλά μετά το βαγόνι 1 δεν μπορεί να εξέλθει, γιατί το εμποδίζουν το βαγόνι 2 και το βαγόνι 3. Άρα, η τρίτη διάταξη δεν είναι εφικτή.

Άσκηση 9.28

Στον Βόρειο Πόλο ο Άγιος Βασίλης και οι βοηθοί του, τα ξωτικά, έχουν πολλή δουλειά φέτος. Τα δώρα ετοιμάζονται πάνω σε δύο ιμάντες και τα ξωτικά στέκονται δίπλα από κάθε ιμάντα. Όταν ένα ξωτικό φτάνει στο εργαστήριο για να πιάσει δουλεία, ο Άγιος Βασίλης το στέλνει να σταθεί στην αρχή ενός ιμάντα (τα άλλα ξωτικά μετακινούνται, για να κάνουν χώρο στο νέο ξωτικό). Το πρώτο ξωτικό που θα πάει στη δουλειά θα τοποθετηθεί στον ιμάντα 1, το δεύτερο στον ιμάντα 2, το τρίτο στον ιμάντα 1, το τέταρτο στον ιμάντα 2 κ.λπ.

Ο Άγιος Βασίλης θέλει να είναι δίκαιος με τα ξωτικά, γι' αυτό θα σχολνάει πρώτο αυτός που έφτασε στο εργαστήριο πρώτο. Επειδή, όμως, θέλει να υπάρχει μία τάξη στο εργαστήριο, αυτός που θα σχολάσει πρέπει να βρίσκεται στην αρχή του ιμάντα. Για να είναι αυτό εφικτό, ο Άγιος Βασίλης θα πρέπει να κάνει κάποιες μετακινήσεις. Δυστυχώς για αυτόν, η μοναδική μετακίνηση που μπορεί να κάνει, χωρίς να επηρεαστεί η παραγωγή, είναι να μεταφέρει ένα ξωτικό από την αρχή του ενός ιμάντα στην αρχή του άλλου.

Να δημιουργήσετε πρόγραμμα, το οποίο θα βοηθήσει τον Άγιο Βασίλη να απαντά στα πιο κάτω ερωτήματα:

'1': Ένα νέο ξωτικό έρχεται να πιάσει δουλειά. Το πρόγραμμα τυπώνει τον ιμάντα στον οποίο θα τοποθετηθεί (1 ή 2).

'2': Ένα ξωτικό θα σχολάσει. Το πρόγραμμα τυπώνει τις μετακινήσεις, αν υπάρχουν, και τον ιμάντα από τον οποίο θα φύγει το ξωτικό. Η μετακίνηση '1 2' σημαίνει ότι το ξωτικό που βρίσκεται στην αρχή του ιμάντα 1 θα μετακινηθεί στην αρχή του ιμάντα 2, ενώ η μετακίνηση '2 1' σημαίνει ότι το ξωτικό που βρίσκεται στην αρχή του ιμάντα 2 θα μετακινηθεί στην αρχή του ιμάντα 1.

Δεδομένα εισόδου

- Η πρώτη γραμμή περιέχει έναν ακέραιο αριθμό N, το πλήθος των ξωτικών.
- Η δεύτερη γραμμή περιέχει μία συμβολοσειρά μεγέθους 2N με τους χαρακτήρες '1' και '2'. Θα υπάρχει ίσος αριθμός χαρακτήρων '1' και '2'. Δηλαδή, όσα ξωτικά πιάσουν δουλειά στο τέλος θα σχολάσουν.

Δεδομένα εισόδου

Κάθε γραμμή της εξόδου περιέχει την απάντηση στα ερωτήματα. Οι μετακινήσεις που θα γίνουν θα πρέπει να είναι οι ελάχιστες.

Παράδειγμα εισόδου

```
3
111222
```

Παράδειγμα εξόδου

```
1
2
1
1 2
1
2 1
2
1
```

Επεξήγηση: Έχουμε συνολικά τρία ξωτικά. Το 1ο ξωτικό που θα πιάσει δουλειά θα πάει στην αρχή του ιμάντα 1, το 2ο ξωτικό στην αρχή του ιμάντα 2 και το 3ο ξωτικό στην αρχή του ιμάντα 1. Όταν το 1ο ξωτικό θα πρέπει να σχολάσει, θα μετακινήσουμε το 3ο ξωτικό από τον ιμάντα 1 στον ιμάντα 2, οπότε το 1ο ξωτικό θα είναι στην αρχή του ιμάντα 1 και θα μπορεί να σχολάσει. Όταν το 2ο ξωτικό θα πρέπει να σχολάσει, θα μετακινήσουμε το 3ο ξωτικό από τον ιμάντα 2 στον ιμάντα 1, οπότε το 2ο ξωτικό θα είναι στην αρχή του ιμάντα 2 και θα μπορεί να σχολάσει. Τέλος, το τρίτο ξωτικό βρίσκεται στην αρχή του ιμάντα 1 και θα μπορεί να σχολάσει.

Άσκηση 9.29

Μία μαθηματική εξίσωση μπορεί να γραφτεί με δύο τρόπους:

```
Infix: (5 * 3) + 7
```

```
Postfix: 5 3 * 7 +
```

Με τη μορφή postfix οι παρενθέσεις δεν εμφανίζονται και το σύμβολο της πράξης εμφανίζεται μετά από τους αριθμούς.

Ο αλγόριθμος για υπολογισμό μίας postfix εξίσωσης έχει ως εξής:

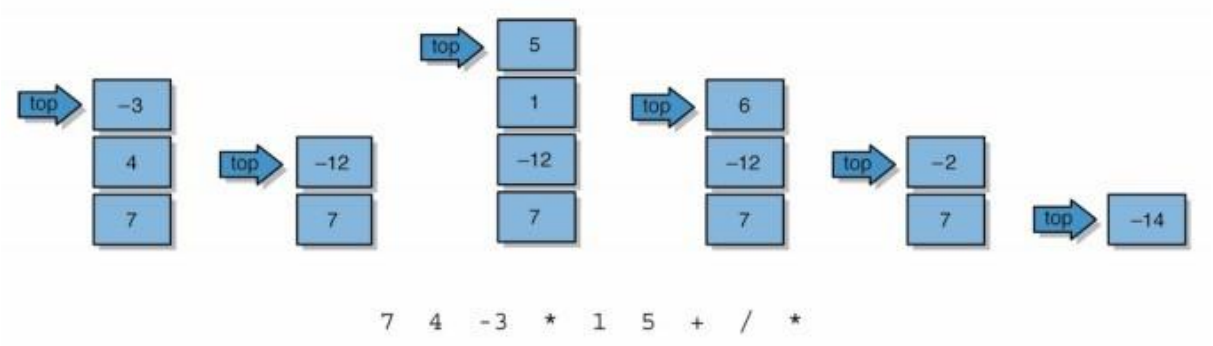
(α) Ξεκινώντας από τα αριστερά, κάντε έλεγχο για αριθμό ή σύμβολο.

(β) Αν είναι αριθμός, τοποθετήστε τον σε μία στοίβα.

(γ) Αν είναι σύμβολο πράξης, εφαρμόστε δύο pop στη στοίβα, για να πάρετε τους δύο τελευταίους αριθμούς, κάντε την πράξη και τοποθετήστε το αποτέλεσμα πίσω στη στοίβα.

(δ) Στο τέλος, θα παραμείνει ένας αριθμός στη στοίβα, ο οποίος θα είναι το αποτέλεσμα της εξίσωσης.

Παράδειγμα



Παράδειγμα εισόδου

```
7 4 -3 * 1 5 + / *
```

Παράδειγμα εξόδου

```
-14
```

Σημείωση: Για να μπορέσουμε να μετατρέψουμε τα δεδομένα μίας συμβολοσειράς σε ακέραιους αριθμούς, μπορούμε να χρησιμοποιήσουμε παράλληλα τις συναρτήσεις c_str και atoi της βιβλιοθήκης algorithm. Οι συναρτήσεις αυτές όταν χρησιμοποιηθούν παράλληλα,

μετατρέπουν, αρχικά, έναν χαρακτήρα σε `string` και, ακολούθως, το `string` σε ακέραιο. Μπορείτε να δείτε το αποτέλεσμα της χρήσης των δύο αυτών συναρτήσεων, εκτελώντας τον πιο κάτω κώδικα, τον οποίο μπορείτε να εφαρμόσετε και για την επίλυση της άσκησης.

```
#include <iostream>
#include <algorithm>
using namespace std;

int main(){
    string st;
    int tmp;
    while (cin >> st){
        if(st=="+")
            cout  << st << endl;
        else if (st=="-")
            cout << st << endl;
        else if (st=="/")
            cout << st << endl;
        else if (st=="*")
            cout << st << endl;
        else {
            tmp = atoi(st.c_str());
            cout << "number: " << tmp << endl;
        }
    }

return 0;
}
```

Άσκηση 9.30

Να προσομοιώσετε μία ουρά σε ένα ταμείο τράπεζας. Κάθε πελάτης έχει διαφορετικό χρόνο άφιξης από τον προηγούμενο πελάτη. Ο χρόνος εξυπηρέτησης κάθε πελάτη μπορεί, επίσης, να διαφέρει, ανάλογα με το είδος της συναλλαγής που επιθυμεί. Ο χρόνος κατά τον οποίο ένας πελάτης φεύγει από το ταμείο είναι ίσος με τον χρόνο άφιξης, συν τον χρόνο αναμονής στην ουρά, συν τον χρόνο εξυπηρέτησης. Να βρείτε τη χρονική στιγμή που εξέρχεται της τράπεζας ο τελευταίος πελάτης και τον μέσο χρόνο αναμονής των πελατών στην ουρά.

Να υλοποιήσετε:

- Μία ουρά πελατών (ταμείο).
- Μία εγγραφή για να φυλάει δεδομένα για μέχρι 100 πελάτες, με μέλη τον χρόνο άφιξης [1...500] και τον χρόνο εξυπηρέτησης [1...50] των πελατών.

Δεδομένα εισόδου

- Στην πρώτη γραμμή θα εμφανίζονται δύο ακέραιοι. Το N, το πλήθος των πελατών και το T, ο μέγιστος χρόνος προσομοίωσης.
- Στις επόμενες N γραμμές θα εμφανίζονται: ο χρόνος άφιξης t1 και ο χρόνος εξυπηρέτησης t2 για κάθε έναν από τους πελάτες.

Δεδομένα εισόδου

- Στην πρώτη γραμμή θα εμφανίζεται η χρονική στιγμή που εξέρχεται της τράπεζας ο τελευταίος πελάτης.
- Στη δεύτερη γραμμή θα εμφανίζεται ο μέσος χρόνος αναμονής των πελατών, με δύο δεκαδικά ψηφία.

Παράδειγμα εισόδου

```
4 20
1 5
2 3
3 4
7 5
```

Παράδειγμα εξόδου

```
18
4.00
```

Επεξήγηση: Έχουμε συνολικά 4 πελάτες και μέγιστο χρόνο προσομοίωσης 20 λεπτών. Ο πρώτος πελάτης, με χρόνο άφιξης 1 και χρόνο εξυπηρέτησης 5, θα εξυπηρετηθεί μέχρι τη χρονική στιγμή 6. Χρόνος αναμονής 0. Ο δεύτερος πελάτης, με χρόνο άφιξης 2 και χρόνο εξυπηρέτησης 3, θα εξυπηρετηθεί από τη χρονική στιγμή 6 μέχρι τη χρονική στιγμή 9. Χρόνος αναμονής 4. Ο τρίτος πελάτης, με χρόνο άφιξης 3 και χρόνο εξυπηρέτησης 4, θα εξυπηρετηθεί από τη χρονική στιγμή 9 μέχρι τη χρονική στιγμή 13. Χρόνος αναμονής 6. Ο τέταρτος πελάτης, με χρόνο άφιξης 7 και χρόνο εξυπηρέτησης 5 θα εξυπηρετηθεί από τη χρονική στιγμή 13 μέχρι τη χρονική στιγμή 18. Χρόνος αναμονής 6. Άρα η χρονική στιγμή που θα εξυπηρετηθούν όλοι οι πελάτες είναι 18. Ο μέσος χρόνος αναμονής των πελατών θα είναι (0+4+6+6)/4=4.00.

Άσκηση 9.31

Να υλοποιηθεί σε πρόγραμμα ο αφηρημένος τύπος μίας ουράς συμβολοσειρών, ο οποίος να υποστηρίζει τις παρακάτω συναρτήσεις/λειτουργίες:

(α) `void Enqueue(string X)` – Το στοιχείο X καταχωρίζεται στο τέλος της ουράς

(β) `void Dequeue( )` – Το στοιχείο στην αρχή της ουράς (αν υπάρχει) αφαιρείται

(γ) `bool Empty( )` – Επιστρέφει `true` αν η ουρά είναι άδεια, αλλιώς επιστρέφει `false`

(δ) `string Front( )` – Επιστρέφει το στοιχείο που βρίσκεται στην αρχή της ουράς

(ε) `string Back( )` – Επιστρέφει το στοιχείο που βρίσκεται στο τέλος της ουράς

(στ)`int Size( )` – Επιστρέφει το μέγεθος της ουράς

Να δηλώσετε ως σταθερά το μέγιστο μέγεθος της ουράς (50) και ως καθολικές μεταβλητές τα εξής:

- Έναν πίνακα συμβολοσειρών μεγέθους 50 θέσεων.
- Δύο ακέραιους δείκτες (first, last), οι οποίοι θα έχουν αρχικά τιμή 0.

Το πρόγραμμα να εμφανίζει το πιο κάτω μενού στην οθόνη και ανάλογα με την επιλογή του χρήστη, να εμφανίζει τα αναμενόμενα αποτελέσματα.

Οι επιλογές από το 1 μέχρι το 5 να καλούν τις αντίστοιχες συναρτήσεις, η επιλογή 6 να εμφανίζει στην οθόνη όλα τα στοιχεία της ουράς, το ένα κάτω από το άλλο και η επιλογή 7 να τερματίζει το πρόγραμμα.

Να γίνονται οι απαραίτητοι έλεγχοι ώστε:

(α) αν η ουρά γεμίσει να μην μπορεί να καταχωριστεί στοιχείο και ο χρήστης να ενημερώνεται με το μήνυμα «Queue is full»

(β) αν η ουρά αδειάσει να μην μπορεί να αφαιρεθεί στοιχείο και ο χρήστης να ενημερώνεται με το μήνυμα «Queue is empty».

```
1. Enqueue item
2. Dequeue item
3. Front item
4. Back item
5. Size of queue
6. Print queue
7. Exit
Enter choice (1-7):
```

Να θεωρήσετε ότι ο χρήστης θα δώσει σωστή επιλογή (1-7) και δεν χρειάζεται έλεγχος.

Βιβλιογραφία

1. Halim, S. and Halim, F. (2013). *Competitive Programming 3*. Singapore: Lulu.com.
2. Skiena, S. (2012). *The Algorithm Design Manual*. London: Springer.
3. Stroustrup, B. (2015). *The C++ Programming Language*. New Jersey: Addison-Wesley.
4. Stroustrup, B. (2015). *Programming: Principles and Practice Using C++*. New Jersey: Addison-Wesley.

ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΠΟΛΙΤΙΣΜΟΥ

ΠΛΗΡΟΦΟΡΙΚΗ

ΣΗΜΕΙΩΣΕΙΣ

Σχεδίαση Λογικών Διαγραμμάτων

Προκαταρκτική Εκτέλεση Λογικού Διαγράμματος και Προγράμματος

Σεπτέμβριος 2017

ΠΕΡΙΕΧΟΜΕΝΑ

Συνάδελφοι,

Οι σημειώσεις αυτές έχουν γραφτεί για να καλύψουν τις ανάγκες του μαθήματος της Πληροφορικής σε Γυμνάσιο και Λύκειο. Έχουν ως στόχο να μας βοηθήσουν να χρησιμοποιήσουμε όλοι ομοιόμορφες μεθόδους στον σχεδιασμό λογικών διαγραμμάτων, αλλά και στην προκαταρκτική εκτέλεση των προγραμμάτων ή και των λογικών διαγραμμάτων. Οι σημειώσεις αποτελούνται από δύο ενότητες με παραδείγματα.
Η πρώτη ενότητα αφορά στον σχεδιασμό των λογικών διαγραμμάτων, διαδικασία αναγκαία στην ανάπτυξη των προγραμμάτων. Τα σύμβολα που χρησιμοποιήθηκαν ακολουθούν το πρότυπο ANSI - Flowchart Symbols.

Η δεύτερη ενότητα αφορά στην προκαταρκτική εκτέλεση προγραμμάτων ή/και λογικών διαγραμμάτων.

Κατά την προκαταρκτική εκτέλεση θα πρέπει να δίνεται ιδιαίτερη έμφαση στην πιστή εκτέλεση όλων των εντολών του προγράμματος ή και του λογικού διαγράμματος. Θα πρέπει να καταγράφονται αναλυτικά όλες οι σταθερές, οι μεταβλητές, οι συνθήκες και οι αντίστοιχες τιμές τους, ώστε να βοηθηθούν οι μαθητές στην κατανόηση της ροής και των αλλαγών των δεδομένων στο πρόγραμμα ή και στο λογικό διάγραμμα.

Στην προσπάθειά μας να συμβαδίσουμε με τα υφιστάμενα εγχειρίδια του Γυμνασίου και του Λυκείου, χρησιμοποιήσαμε μεθοδολογία από διάφορες έγκυρες πηγές (βλέπε βιβλιογραφία), χωρίς να χάνεται η συνοχή και η ομοιομορφία. Βέβαια, δεν αποκλείονται άλλες προσεγγίσεις παρουσίασης των λογικών διαγραμμάτων και προκαταρκτικών εκτελέσεων, νοουμένου ότι είναι επιστημονικά αποδεκτές.

Συγγραφική ομάδα:
Μιλτιάδου Μάριος, Β.Δ.
Σχίζα Ιουλία, Β.Δ.
Σωτικόπουλος Κωνσταντίνος, Β.Δ.
Τορτούρης Μιχάλης, Β.Δ.

Εποπτεία:
Χατζησάββας Γιώργος, ΕΜΕ

Αναθεώρηση για την C++:
Άκης Συκοπετρίτης, Καθηγητής Πληροφορικής
Κωνσταντίνος Σωφρονίου, Σύμβουλος Πληροφορικής

Λογικά Διαγράμματα

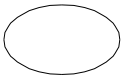
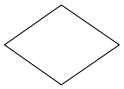
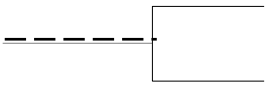
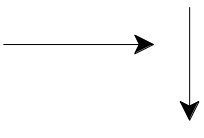
Δομημένος Προγραμματισμός

Ο δομημένος προγραμματισμός είναι η τεχνική που διαιρεί το πρόγραμμα σε λογικά τμήματα, χρησιμοποιώντας βασικές δομές. Η τεχνική αυτή μάς βοηθά στην κωδικοποίηση, στον έλεγχο και στη συντήρηση ενός προγράμματος. Στην παραγωγή λογισμικού, ο δομημένος προγραμματισμός χρησιμοποιείται ως βασική αρχή προγραμματισμού. Βασικός στόχος του δομημένου προγραμματισμού είναι η παραγωγή προγραμμάτων που έχουν μια καθορισμένη μορφή και για αυτό γίνονται πιο εύκολα κατανοητά, όχι μόνο από τον δημιουργό τους αλλά και από άλλους προγραμματιστές, οι οποίοι στο μέλλον θα ασχοληθούν με τα προγράμματα αυτά.

Ο δομημένος προγραμματισμός χρησιμοποιεί μόνο τρεις βασικές δομές ελέγχου, που είναι:

1. Η Ακολουθιακή Δομή (Sequence)
2. Η Δομή Διακλάδωσης (Selection) και η επέκτασή της, η Περιπτωσιακή Δομή (Switch)
3. Η Δομή Επανάληψης (Iteration)

Σύμβολα Λογικών Διαγραμμάτων

Σύμβολο	Επεξήγηση
	Αρχή - Τέλος Κυρίως Προγράμματος Είσοδος - Έξοδος Υποπρογραμμάτων
	Επεξεργασία
	Είσοδος Δεδομένων – Έξοδος Αποτελεσμάτων
	Απόφαση
	Υποπρόγραμμα (=Συνάρτηση)
	Σύνδεσμος
	Σχόλιο
	Ροή

ANSI Flowchart symbols (Capron, H. L., Perron, J. D., p. 520)

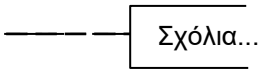
Εισηγήσεις για Ομοιόμορφο Σχεδιασμό Λογικών Διαγραμμάτων

1. Το λογικό διάγραμμα πρέπει να αντιστοιχεί με το πρόγραμμα. Αντιστοιχία λογικού διαγράμματος και προγράμματος σημαίνει ότι:
   i. Οι τεχνικές και οι δομές προγραμματισμού πρέπει να είναι οι ίδιες στο λογικό διάγραμμα και στο πρόγραμμα. Για παράδειγμα, αν χρησιμοποιείται η δομή if-else στο λογικό διάγραμμα, θα πρέπει να αναμένεται η ίδια δομή και στο πρόγραμμα.
   ii. Οι μεταβλητές πρέπει να είναι οι ίδιες στο λογικό διάγραμμα και στο πρόγραμμα.
2. Τα μηνύματα εισόδου και εξόδου στο λογικό διάγραμμα δεν είναι υποχρεωτικά. Το πρόγραμμα, όμως, πρέπει να περιλαμβάνει τα κατάλληλα μηνύματα εισόδου και εξόδου (είτε το ζητά η άσκηση είτε όχι, στο πλαίσιο του καλού προγραμματισμού).
3. Οι μεταβλητές που χρησιμοποιούνται σε λογικό διάγραμμα πρέπει να γράφονται με λατινικούς χαρακτήρες. Οι οδηγίες μπορούν να είναι στα Ελληνικά (διάβασε, τύπωσε, κ.τ.λ.).
4. Στις αποφάσεις στο λογικό διάγραμμα μπορούν να χρησιμοποιηθούν τα εξής: Αληθής – Ψευδής ή Α – Ψ ή Ναι – Όχι ή Ν – Ο ή True – False ή T – F ή Yes – No ή Y - N.
5. Ως σύμβολο εκχώρησης τιμής σε μεταβλητή συστήνεται όπως χρησιμοποιείται το βέλος (←) για ομοιομορφία, χωρίς όμως να απαγορεύονται άλλα σύμβολα όπως το =.
6. Σε περίπτωση μαθηματικών πράξεων στο λογικό διάγραμμα, μπορούν να χρησιμοποιηθούν είτε μαθηματικά σύμβολα (όπως $\frac{a+b+c}{3}, \sqrt{a-b}, x^2, \leq$, κ.τ.λ.), είτε σύμβολα που, συνήθως, χρησιμοποιούνται σε γλώσσες προγραμματισμού, όπως: * για πολλαπλασιασμό, / για διαίρεση, MOD, %, DIV, <=, >=, κ.τ.λ.
7. Σε περίπτωση εκτύπωσης (πραγματικών ή και ακεραίων αριθμών, χαρακτήρων και ακολουθίας χαρακτήρων) σε λογικό διάγραμμα, δεν χρειάζονται οι δείκτες πλάτους εκτύπωσης. Στο αντίστοιχο πρόγραμμα, όμως, συστήνεται όπως χρησιμοποιούνται στο πλαίσιο του καλού προγραμματισμού. Για παράδειγμα:

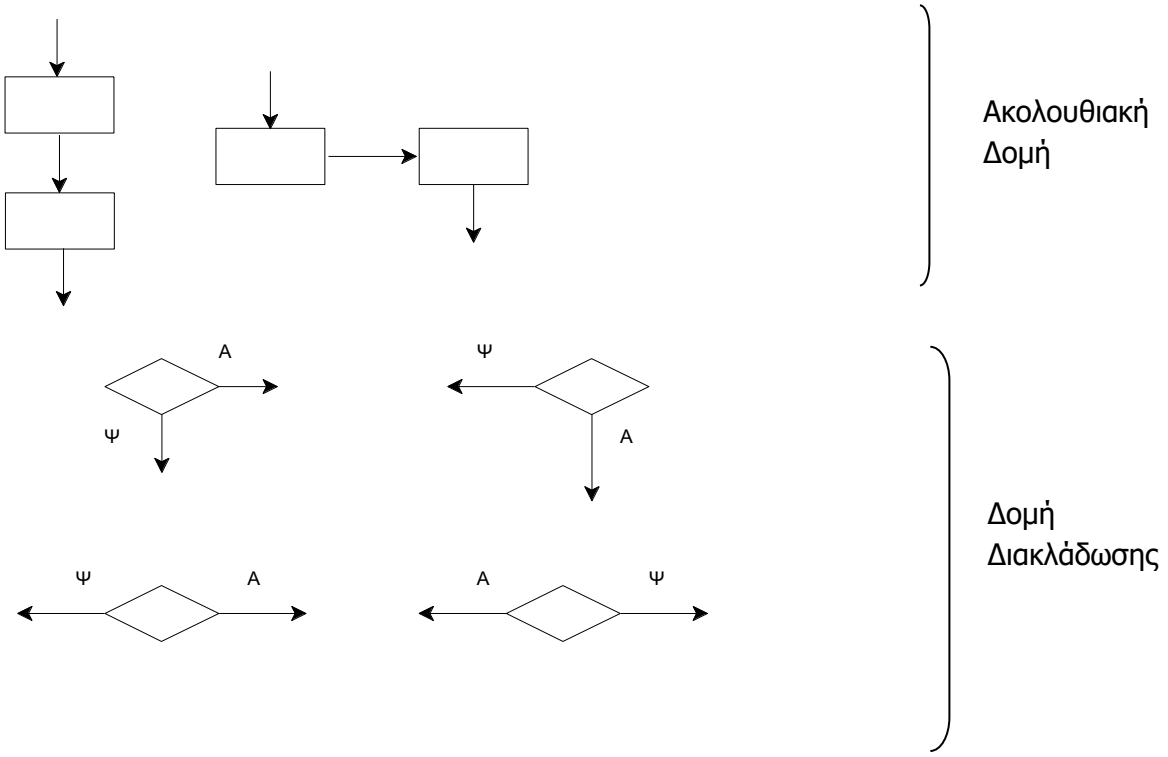
   Pascal:
   ```
   writeln(A:6:2, B:8, 'Onoma':8);
   ```

   C++:
   ```
   cout<<fixed<<setw(8)<<setprecision(2)<<A<<setw(8)<<B<<setw(8)<<"Onoma";
   ```

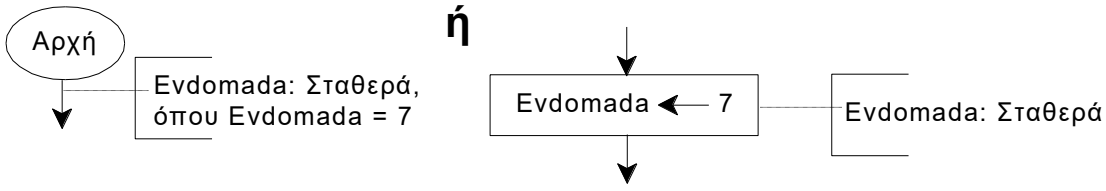
8. Στο λογικό διάγραμμα μπορούν να γραφτούν σχόλια, χρησιμοποιώντας το πιο κάτω σύμβολο:



9. Η ροή στο λογικό διάγραμμα υποδεικνύεται από τα βέλη. Τα πιο κάτω σχήματα είναι αποδεκτά:

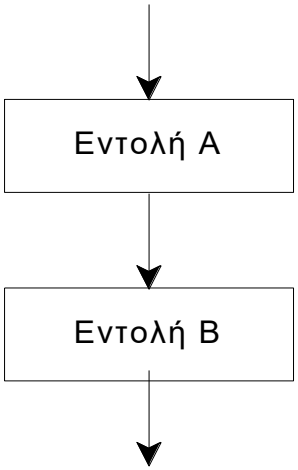


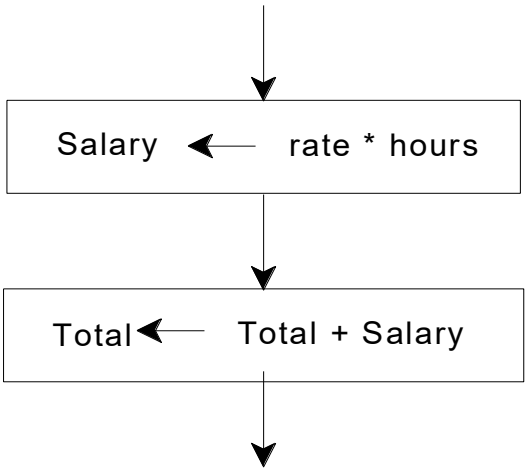
10. Οι σταθερές πρέπει να διαφοροποιούνται από τις μεταβλητές με τη χρήση σχολίου. Για παράδειγμα:



Δομές Προγραμματισμού

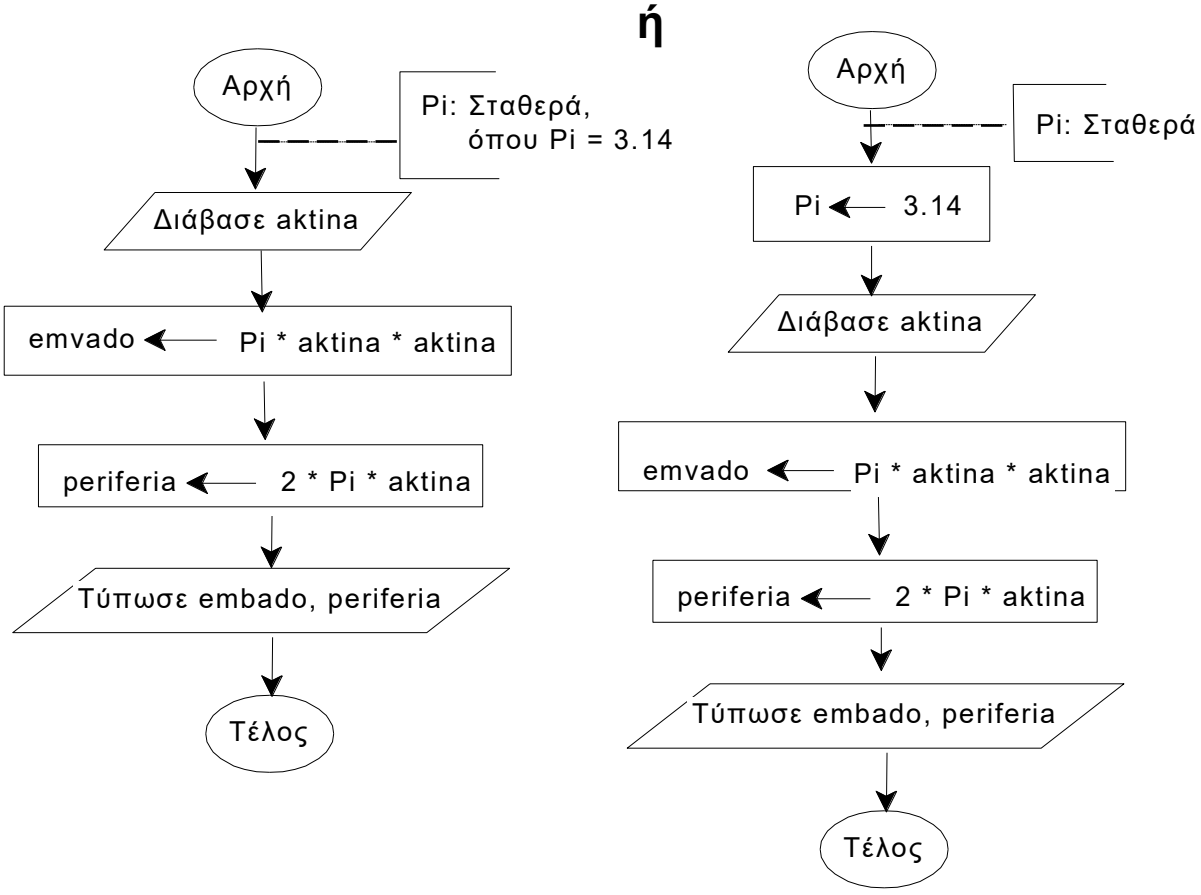
(α) Ακολουθιακή δομή



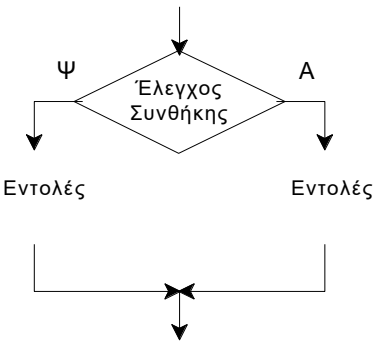


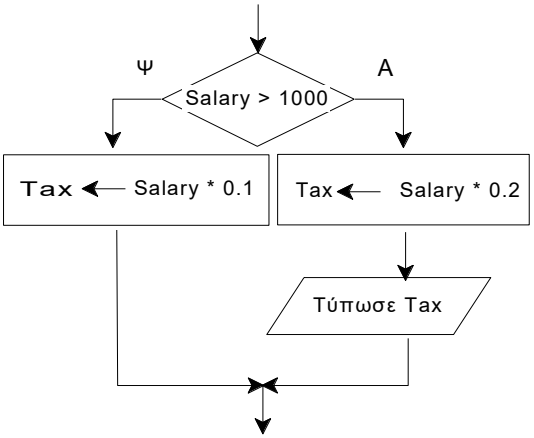
Παράδειγμα Ακολουθιακής Δομής

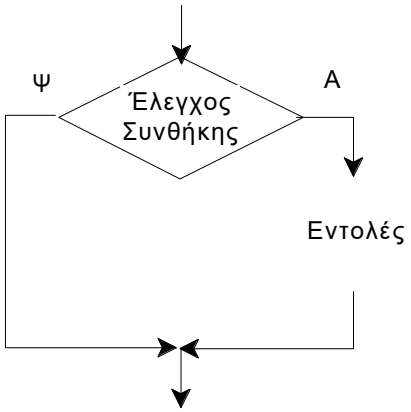
Να σχεδιάσετε λογικό διάγραμμα, το οποίο να ζητά την ακτίνα του κύκλου και να υπολογίζει και να παρουσιάζει το εμβαδόν και την περιφέρειά του.
(Σημείωση: Εμβαδόν Κύκλου = πr^2 , Περιφέρεια Κύκλου = $2\pi r$, $\pi = 3.14$)

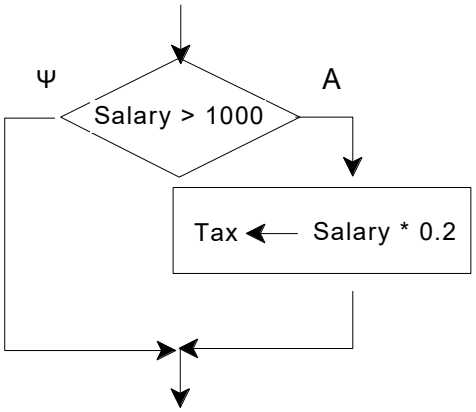


(β) Δομή Διακλάδωσης

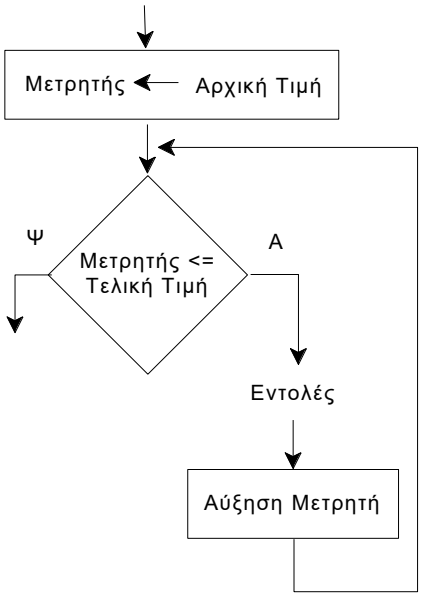


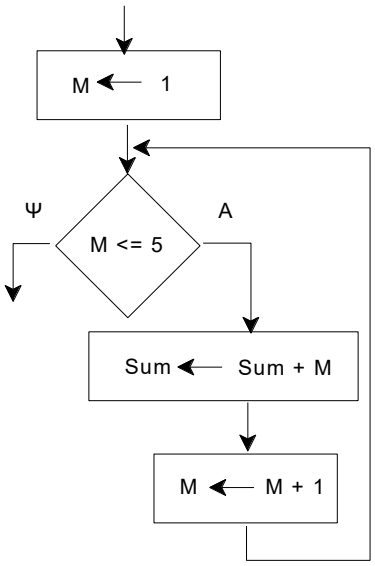


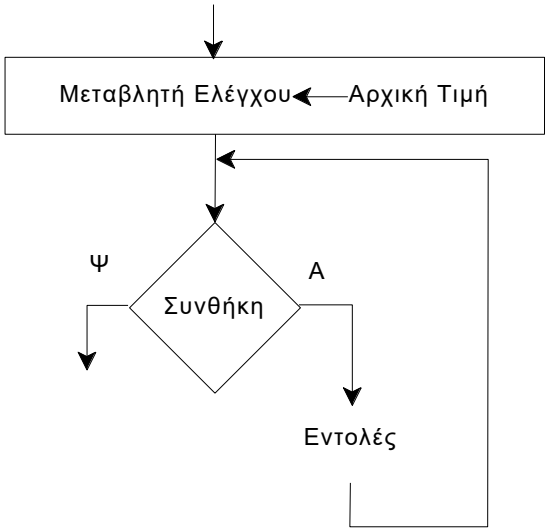


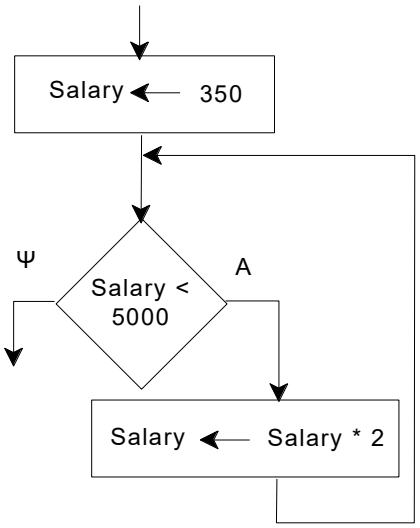


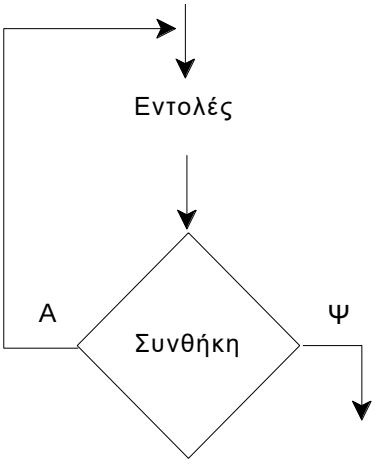
(γ) Επαναληπτική Δομή

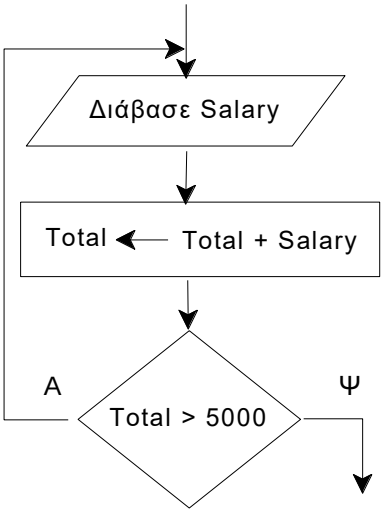






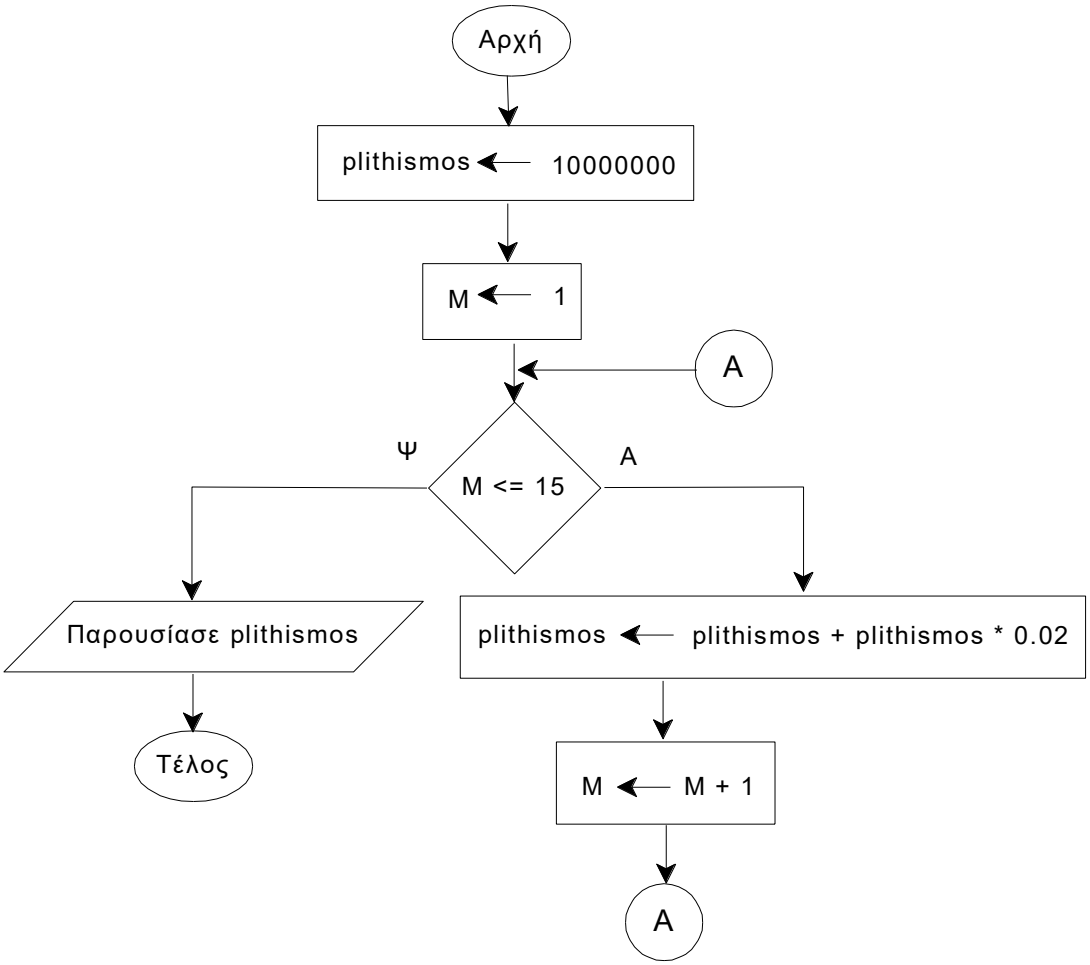




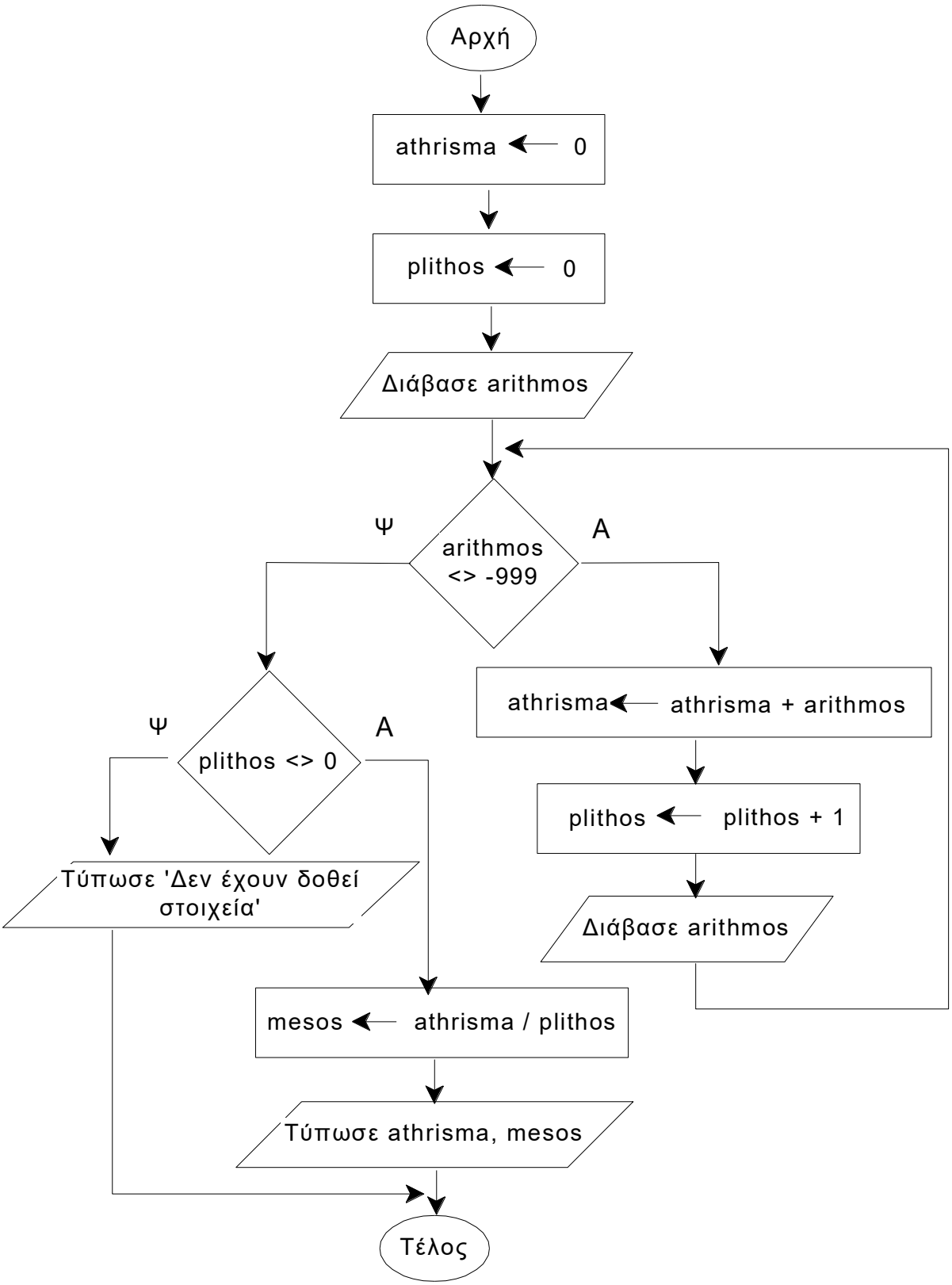


Παραδείγματα Επαναληπτικής Δομής

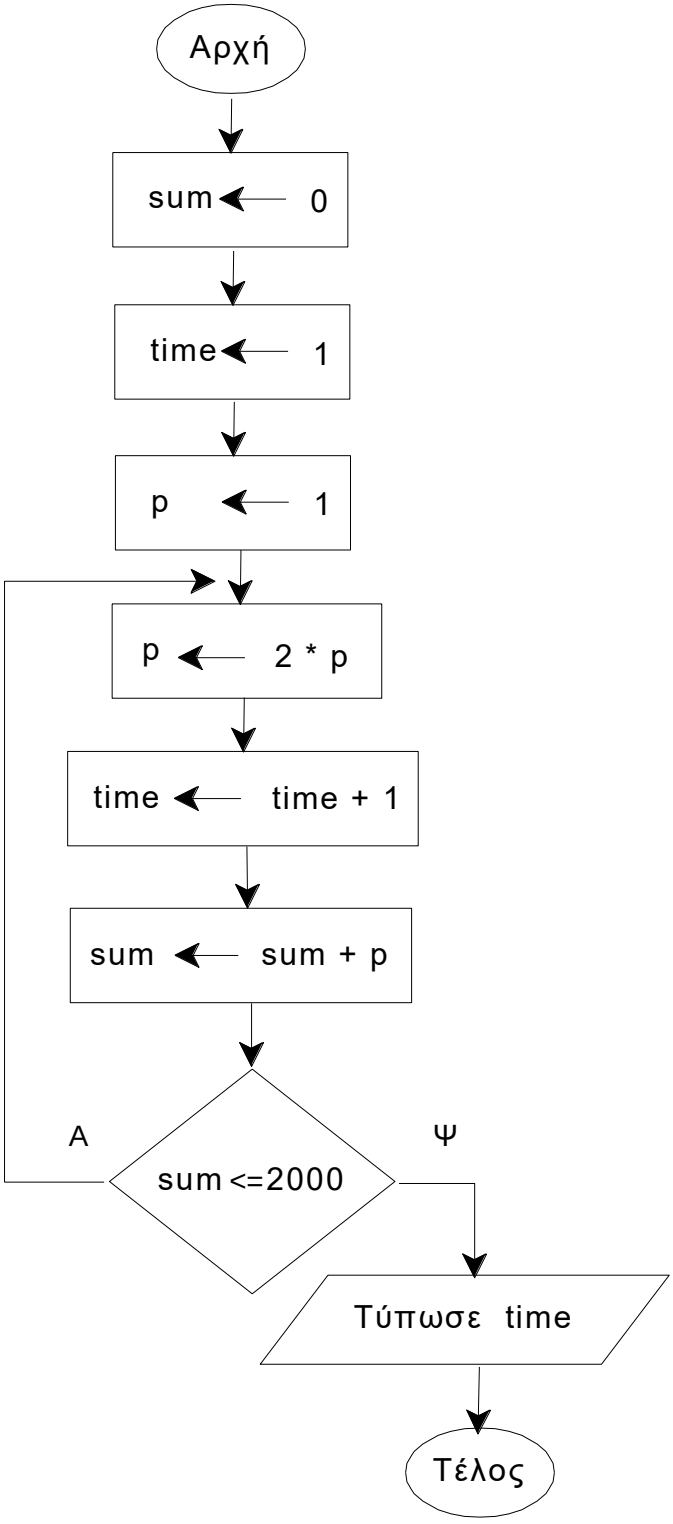
1. Ο πληθυσμός μιας χώρας είναι 10 εκατομμύρια και παρουσιάζει ετήσια αύξηση 2%. Να σχεδιάσετε λογικό διάγραμμα, το οποίο να υπολογίζει και να παρουσιάζει τον πληθυσμό της χώρας μετά από 15 χρόνια.



2. Να σχεδιάσετε λογικό διάγραμμα, το οποίο να δέχεται ένα πλήθος θετικών ακεραίων αριθμών και να υπολογίζει και να παρουσιάζει το άθροισμά τους και τον μέσο όρο τους. Η διαδικασία να τερματίζεται με την εισαγωγή αρνητικού αριθμού.



3. Ο πατέρας του Γιάννη δίνει στο παιδί του διπλάσιο ποσό από τον προηγούμενο μήνα. Να σχεδιάσετε λογικό διάγραμμα το οποίο να υπολογίζει και να παρουσιάζει το μικρότερο χρονικό διάστημα, κατά το οποίο ο Γιάννης θα πάρει ποσό που να υπερβαίνει τις 2000 ευρώ. Να υποθέσετε ότι ο Γιάννης πήρε την πρώτη φορά 1 ευρώ.



Προκαταρκτική Εκτέλεση

Κατά την προκαταρκτική εκτέλεση, ο μαθητής εκτελεί βήμα προς βήμα όλες τις εντολές του προγράμματος ή του λογικού διαγράμματος, όπως θα τις εκτελούσε ο ηλεκτρονικός υπολογιστής. Σε κάθε βήμα σημειώνει τις τιμές των μεταβλητών ή των αποφάσεων που αλλάζουν. Για τον σκοπό αυτό, χρησιμοποιείται, συνήθως, ένας πίνακας στον οποίο καταχωρούνται:

i. Οι τιμές των σταθερών (εάν υπάρχουν).
ii. Οι αρχικές τιμές των μεταβλητών, καθώς και οι τιμές τους μετά από κάθε αλλαγή.
iii. Οι αποφάσεις και οι τιμές τους όταν αλλάζουν.
iv. Οι δείκτες και το περιεχόμενο των πινάκων.
v. Η παρουσίαση των μηνυμάτων και των αποτελεσμάτων του προγράμματος ή του λογικού διαγράμματος. Η παρουσίαση θεωρείται ανεξάρτητη από το κύριο πρόγραμμα ή τα υποπρογράμματα (συνάρτηση ή διαδικασία, αν υπάρχουν) και είναι ενιαία (δηλαδή, όπως θα εμφανιζόταν στην οθόνη του Η.Υ. μετά την εκτέλεση του προγράμματος).

Για σκοπούς εξοικονόμησης χώρου τα βήματα της προκαταρκτικής εκτέλεσης δεν είναι αναγκαίο να παρουσιάζονται οπωσδήποτε σε ξεχωριστές γραμμές (βλέπε παραδείγματα).

Δομές Προγραμματισμού

(α) Ακολουθιακή Δομή

Να γίνει η προκαταρκτική εκτέλεση του πιο κάτω προγράμματος με τιμές εισόδου: 5.5, 8.0

```
// Τρίγωνο
#include <iostream>
#include <iomanip>
#include <cmath>
using namespace std;
int main() {
    float katheti1, katheti2, ypoteinousa, perimetros, emvadon;
    cout << "Δώσε το μήκος της κάθετης 1: ";
    cin >> katheti1;
    cout << "Δώσε το μήκος της κάθετης 2: ";
    cin >> katheti2;
    ypoteinousa = sqrt( pow( katheti1, 2.0 ) + pow( katheti2, 2.0 ) );
    perimetros = katheti1 + katheti2 + ypoteinousa ;
    emvadon = katheti1 * katheti2 / 2;
    cout << "Κάθετη 1: " << fixed << setprecision(2) << katheti1 << endl;
    cout << " Κάθετη 2: " << fixed << setprecision(2) << katheti2 << endl;
    cout << "Υποτείνουσα: " << fixed << setprecision(2) << ypoteinousa << endl;
    cout << "Περίμετρος: " << fixed << setprecision(2) << perimetros << endl;
    cout << "Εμβαδόν: " << fixed << setprecision(2) << emvadon << endl;
    return 0;
}
```

Μεταβλητές					Παρουσίαση
katheti1	katheti2	ypoteinousa	perimetros	emvadon	
5.5					Δώσε το μήκος της κάθετης 1:
	8.0				Δώσε το μήκος της κάθετης 2:
		9.71	23.21	22.00	Κάθετη 1: 5.50
					Κάθετη 2: 8.00
					Υποτείνουσα: 9.71
					Περίμετρος: 23.21
					Εμβαδόν: 22.00

(β) Δομή Διακλάδωσης

Παράδειγμα 1: Να γίνει η προκαταρκτική εκτέλεση του πιο κάτω προγράμματος με τιμές εισόδου: 2000, 100

```
// Rent_A_Car
#include <iostream>
#include <iomanip>
using namespace std;

int main() {
    int kivismos;
    float timi, poso, ekptosi;
    ekptosi = 0.0;
    cout << "Δώσε τον κυβισμό: ";
    cin >> kivismos;
    cout << "Δώσε την αρχική τιμή: ";
    cin >> timi;
    if ( kivismos >= 1500 )
        ekptosi = timi * 10/100;
    poso = timi - ekptosi;
    cout << "Ο κυβισμός είναι: " << kivismos << endl;
    cout << "Η αρχική τιμή είναι: " << fixed << setprecision(2) << timi << endl;
    cout << "Η έκπτωση είναι: " << fixed << setprecision(2) << ekptosi << endl;
    cout << "Η τιμή πληρωμής είναι: " << fixed << setprecision(2) << poso << endl;
    return 0;
}
```

Μεταβλητές				Απόφαση		Παρουσίαση
ekptosi	kivismos	timi	poso	kivismos>=1500	Α/Ψ	Δώσε τον κυβισμό:
0.0						Δώσε την αρχική τιμή:
	2000	100		2000>=1500	Α	Ο κυβισμός είναι: 2000
10.0			90.0			Η αρχική τιμή είναι: 100.00
						Η έκπτωση είναι: 10.00
						Η τιμή πληρωμής είναι: 90.00

Παράδειγμα 2: Να γίνει η προκαταρκτική εκτέλεση του πιο κάτω προγράμματος με τιμές εισόδου για την περίπτωση 1: M, 45 και για την περίπτωση 2: F, 28 .

```
// Insurance
#include <iostream>
using namespace std;
int main() {
    int age, asfalistra;
    char sex;
    cout << "Δώσε το φύλο: ";
    cin >> sex;
    cout << "Δώσε την ηλικία: ";
    cin >> age;
    if ( sex == 'M' )
        if ( age < 25 )
            asfalistra = 300;
        else
            asfalistra = 250;
    else  if ( age < 21 )
asfalistra = 200;
    else
asfalistra = 150;
    cout << "Ασφάλιστρα: " << asfalistra;
    return 0;
}
```

Περίπτωση 1

Μεταβλητές			Απόφαση				Παρουσίαση
age	asfalistra	sex	sex=='M'	Α/Ψ	age<25	Α/Ψ	Δώσε το φύλο: Δώσε την ηλικία: Ασφάλιστρα: 250
		M					
45							
	250		M==M	Α	45<25	Ψ	

Περίπτωση 2

Μεταβλητές			Απόφαση				Παρουσίαση
age	asfalistra	sex	sex=='M'	Α/Ψ	age<21	Α/Ψ	Δώσε το φύλο: Δώσε την ηλικία: Ασφάλιστρα: 150
		F					
28							
	150		F==M	Ψ	28<21	Ψ	

(γ) Επαναληπτική Δομή

Παράδειγμα 1: Να γίνει η προκαταρκτική εκτέλεση του πιο κάτω προγράμματος.

```
// Average
#include <iostream>
#include <iomanip>
using namespace std;
int main() {
    int no, sum, metritis;
    float mesos;
    sum = 0;
    metritis = 0;
    for ( no=10; no<=15; no++ )  {
        sum = sum + no;
        metritis = metritis + 1;
    }
    mesos = (float) sum / metritis;
    cout << "Ο μέσος όρος είναι: " << fixed << setprecision(2) << mesos;
    return 0;
}
```

Μεταβλητές				Απόφαση		Παρουσίαση
no	sum	metritis	mesos	no<=15	Α/Ψ	Ο μέσος όρος είναι: 12.50
10	0	0		10<=15	Α	
	10	1				
11	21	2		11<=15	Α	
12	33	3		12<=15	Α	
13	46	4		13<=15	Α	
14	60	5		14<=15	Α	
15	75	6		15<=15	Α	
16				16<=15	Ψ	
			12.50			

Παράδειγμα 2: Να γίνει η προκαταρκτική εκτέλεση του πιο κάτω προγράμματος.

```
// Sum_Of_Odd_Nos;
#include <iostream>
using namespace std;
int main() {
    int arithmos, sum;
    sum = 0;
    arithmos = 1;
    while ( arithmos <= 11 ) {
        sum = sum + arithmos;
        arithmos = arithmos + 2;
    }
    cout << "Το άθροισμα των μονών αριθμών (1-11) είναι: " << sum;
    return 0;
}
```

Μεταβλητές		Απόφαση		Παρουσίαση
arithmos	sum	arithmos<=11	Α/Ψ	Το άθροισμα των μονών αριθμών (1-11) είναι: 36
1	0	1<=11	Α	
	1			
3	4	3<=11	Α	
5	9	5<=11	Α	
7	16	7<=11	Α	
9	25	9<=11	Α	
11	36	11<=11	Α	
13		13<=11	Ψ	

Παράδειγμα 3: Να γίνει η προκαταρκτική εκτέλεση του πιο κάτω προγράμματος.

```
// Sum_Of_Odd_Nos;
#include <iostream>
using namespace std;
int main() {
    int arithmos, sum;
    sum = 0;
    arithmos = 1;
    do {
        sum = sum + arithmos;
        arithmos = arithmos + 2;
    } while ( arithmos <= 11);
    cout << "Το άθροισμα των μονών αριθμών (1-11) είναι: " << sum;
    return 0;
}
```

Μεταβλητές		Απόφαση		Παρουσίαση
arithmos	sum	arithmos<=11	Α/Ψ	Το άθροισμα των μονών αριθμών (1-11) είναι: 36
1	0			
3	1	3<=11	Α	
5	4	5<=11	Α	
7	9	7<=11	Α	
9	16	9<=11	Α	
11	25	11<=11	Α	
13	36	13<=11	Ψ	

Υποπρογράμματα - Συναρτήσεις που υπολογίζουν και επιστρέφουν μόνο μία τιμή στην κύρια συνάρτηση (με τη χρήση της εντολής return)

Παράδειγμα 1: Να γίνει η προκαταρκτική εκτέλεση του πιο κάτω προγράμματος με τιμές εισόδου: 10.5, 12.5.

```
// Demo1;
#include <iostream>
#include <iomanip>
using namespace std;

float calculate ( float m, float p) {       // συνάρτηση calculate
      float emvadon, perimetros;
      emvadon = p * m;
      perimetros = 2.0 * (m + p);
      cout << "Το εμβαδόν είναι: " << fixed << setprecision(2) << emvadon << endl;
      return perimetros;
}

int main() {      // κύριο πρόγραμμα
     float mikos, platos, perim;
     cout << "Δώσε το μήκος: ";
     cin >> mikos;
     cout << "Δώσε το πλάτος: ";
     cin >> platos;
     perim = calculate ( mikos, platos );
     cout << "Η περίμετρος είναι: " << fixed << setprecision(2) << perim << endl;
     return 0;
}
```

Κύρια συνάρτηση (main)

Μεταβλητές			Παρουσίαση
mikos	**platos**	**perim**	
10.5	12.5	46	Δώσε το μήκος: Δώσε το πλάτος: Το εμβαδόν είναι: 131.25 Η περίμετρος είναι: 46.00

Συνάρτηση calculate

Τυπικές παράμετροι		Τοπικές μεταβλητές		Επιστρέφει
m	**p**	**emvadon**	**perimetros**	
10.5	12.5	131.25	46	46

Παράδειγμα 2: Να γίνει η προκαταρκτική εκτέλεση του πιο κάτω προγράμματος με τιμές εισόδου: 5, 10.

```
// Μεγαλύτερος
#include <iostream>
using namespace std;

int bigger( int num1, int num2 ) {
    if ( num1 > num2 )
        return num1;
    else
        return num2;
}
int main() {
   int arithmos1, arithmos2;
   cout << "Δώσε δύο ακέραιους αριθμούς: ";
   cin >> arithmos1 >> arithmos2;
   cout << "Ο μεγαλύτερος είναι: " << bigger( arithmos1, arithmos2) << endl;
   return 0;
}
```

Κύρια συνάρτηση (main)

Μεταβλητές		Παρουσίαση
arithmos1	arithmos2	
5	10	Δώσε δύο ακέραιους αριθμούς: Ο μεγαλύτερος είναι: 10

Συνάρτηση bigger

Τυπικές παράμετροι		Απόφαση		Επιστρέφει
num1	num2	num1>num2	Α/Ψ	
5	10	5>10	Ψ	10

Υποπρογράμματα – Συναρτήσεις που **έχουν τυπικές παραμέτρους αναφοράς** και **δεν έχουν την εντολή return**

Παράδειγμα 1: Να γίνει η προκαταρκτική εκτέλεση του πιο κάτω προγράμματος με τιμές εισόδου: 10, 12, 3.

```
// Vol_Per;
#include <iostream>
using namespace std;

void perivol( int length, int width, int height, int &perimeter, int &volume ) {
      perimeter = length + 2 * width * height;
      volume = length * width * height;
}
int main() {
      int len, wid, heit, perim, vol;
      cout << "Δώσε το μήκος: ";
      cin >> len;
      cout << "Δώσε το πλάτος: ";
      cin >> wid;
      cout << "Δώσε το ύψος: ";
      cin >> heit;
      perivol ( len, wid, heit, perim, vol);
      cout << "Η περίμετρος είναι: " << perim << endl ;
      cout << "Ο όγκος είναι: " << vol << endl;
      return 0;
}
```

Κύρια συνάρτηση (main)

Μεταβλητές					Παρουσίαση
len	**wid**	**heit**	**perim**	**vol**	Δώσε το μήκος: Δώσε το πλάτος: Δώσε το ύψος: Η περίμετρος είναι: 82 Ο όγκος είναι: 360
10	12	3	82	360	

Συνάρτηση perivol

Τυπικές Παράμετροι Τιμών			Τυπικές Παράμετροι Αναφοράς	
length	**width**	**height**	**perimeter**	**volume**
10	12	3	82	360

Υποπρογράμματα – Συναρτήσεις που έχουν **και τυπικές παραμέτρους αναφοράς και την εντολή return.**

Παράδειγμα 4.6 – σελ. βιβλίου 139
(Έχει τυπικές παραμέτρους αναφοράς ΚΑΙ την εντολή return).

```
#include <iostream>
using namespace std;

int check(int a, int &b){
 b += 2;
if (a>b)
    return a;
return a+b;
}

int main(){
  int x = 3, y = 2;
  if (x>y)
      cout << check(x,y) << endl;
  else
      cout << check(y,x) << endl;
  cout << x << " " << y << endl;
return 0;
}
```

Κύρια συνάρτηση (main)

Μεταβλητές		Αποφάσεις		Παρουσίαση
x	y	x>y	T/F	
3	2	3>2	T	
	4			7
				3□4

Συνάρτηση check

Τυπικές Παράμετροι Τιμών	Τυπικές Παράμετροι Αναφοράς	Αποφάσεις		Επιστρέφει
a	b	a>b	T/F	
3	2			
	4	3>4	F	7

Υποπρογράμματα – Συναρτήσεις που δεν έχουν ούτε τυπικές παραμέτρους αναφοράς ούτε και την εντολή return, αλλά το **αποτέλεσμα τυπώνεται εντός της συνάρτησης**.

Παράδειγμα 1:

```
#include <iostream>
using namespace std;

void check(int a, int b){
if (a>b)
  cout<<"a="<<a<<endl;
else
  cout<<"b="<<b<<endl;
}
int main(){
  int x = 3, y = 2;
  if (x>y)
     check(x,y);
  else
     check(y,x);
  cout<<"x="<<x<<" "<<"y="<<y;
return 0;
}
```

Κύρια συνάρτηση (main)

Μεταβλητές		Αποφάσεις		Παρουσίαση
x	y	x>y	T/F	
3	2	3>2	T	a=3
				x=3 y=2

Συνάρτηση check

Παράμετροι		Αποφάσεις	
a	b	a>b	T/F
3	2	3>2	T

Παράδειγμα 2: Να γίνει η προκαταρκτική εκτέλεση του πιο κάτω προγράμματος με τιμές εισόδου: 10, 12, 3.

```
// Vol_Per;
#include <iostream>
using namespace std;

void perivol( int length, int width, int height ) {
      int perimeter, volume;
      perimeter = length + 2 * width * height;
      volume = length * width * height;
      cout << "Η περίμετρος είναι: " << perimeter << endl;
      cout << "Ο όγκος είναι: "<< volume << endl;
}

int main() {
      int len, wid, heit;
      cout << "Δώσε το μήκος: ";
      cin >> len;
      cout << "Δώσε το πλάτος: ";
      cin >> wid;
      cout << "Δώσε το ύψος: ";
      cin >> heit;
      perivol( len, wid, heit);
      return 0;
}
```

Κύρια συνάρτηση (main)

Μεταβλητές			Παρουσίαση
len	wid	heit	
10	12	3	Δώσε το μήκος:
			Δώσε το πλάτος:
			Δώσε το ύψος:
			Η περίμετρος είναι: 82
			Ο όγκος είναι: 360

Συνάρτηση perivol

Τυπικές Παράμετροι Τιμών			Τοπικές μεταβλητές	
length	width	height	perimeter	volume
10	12	3	82	360

Μονοδιάστατοι Πίνακες

Παράδειγμα 1: Να γίνει η προκαταρκτική εκτέλεση του πιο κάτω προγράμματος με τιμές εισόδου: 10, 20, 30, 40, 50 .

```
// Πίνακες
#include <iostream>
using namespace std;

const int LOW=0, HIGH=5;

int main() {
   int i;
   int lista[ HIGH ];
   for ( i=LOW; i<HIGH; i++ ) {
         cout << "Δώσε τον αριθμό: ";
         cin >> lista[ i ];
   }
   cout << "Αντίστροφα οι αριθμοί είναι: " << endl;
   for ( i = HIGH-1; i>=LOW; i-- )
         cout << lista[ i ] << endl;
   return 0;
}
```

Σταθερές		Μεταβλητές			Απόφαση			Παρουσίαση
LOW	HIGH	i	Πίνακας lista		i<5	i>=0	Α/Ψ	
			Δείκτης	Περιεχόμενο				
0	5	0	0	10	0<5		Α	Δώσε τον αριθμό:
		1	1	20	1<5		Α	Δώσε τον αριθμό:
		2	2	30	2<5		Α	Δώσε τον αριθμό:
		3	3	40	3<5		Α	Δώσε τον αριθμό:
		4	4	50	4<5		Α	Δώσε τον αριθμό:
		5			5<5		Ψ	Αντίστροφα οι αριθμοί είναι:
		4	4			4>=0	Α	50
		3	3			3>=0	Α	40
		2	2			2>=0	Α	30
		1	1			1>=0	Α	20
		0	0			0>=0	Α	10
		-1				-1>=0	Ψ	

Παράδειγμα 2: Να γίνει η προκαταρκτική εκτέλεση του πιο κάτω προγράμματος.

```
// Πίνακες2
#include <iostream>
using namespace std;

int main() {
      int i, a[ 6 ];
      for ( i = 0; i< 5; i++)
            a[ i ] = (i+1) * 3;
      for ( i = 0; i< 4; i++)
            a[ i+1 ] = (i+1) * a[ i ];
      for ( i = 0; i< 5; i++ )
            cout << a[ i ] << endl;
      return 0;
}
```

Μεταβλητές			Απόφαση				Παρουσίαση
i	Πίνακας a		i <5	i<4	i<5	Α/Ψ	
	Δείκτης	Περιεχόμενο					
0	0	3	0<5			Α	
1	1	6	1<5			Α	3
2	2	9	2<5			Α	3
3	3	12	3<5			Α	6
4	4	15	4<5			Α	18
5			5<5			Ψ	72
0	1	3		0<4		Α	
1	2	6		1<4		Α	
2	3	18		2<4		Α	
3	4	72		3<4		Α	
4				4<4		Ψ	
0	0				0<5	Α	
1	1				1<5	Α	
2	2				2<5	Α	
3	3				3<5	Α	
4	4				4<5	Α	
5					5<5	Ψ	

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. "Ανάπτυξη εφαρμογών σε προγραμματιστικό περιβάλλον". Γ' Λυκείου Τεχνολογική Κατεύθυνση – Βιβλίο καθηγητή

Αντωνάκος Νίκος, Βογιατζής Ιωάννης, Κατοπόδης Ιωάννης, Πατριαρχέας Κυριάκος. Υπουργείο Παιδείας και Θρησκευμάτων – Παιδαγωγικό Ινστιτούτο. Αθήνα 1999.

2. "Ανάπτυξη εφαρμογών σε προγραμματιστικό περιβάλλον". Γ' Λυκείου Τεχνολογική Κατεύθυνση – Τετράδιο μαθητή.

Αντωνάκος Νίκος, Βογιατζής Ιωάννης, Κατοπόδης Ιωάννης, Πατριαρχέας Κυριάκος. Υπουργείο Παιδείας και Θρησκευμάτων – Παιδαγωγικό Ινστιτούτο. Αθήνα 1999.

3. "Ανάπτυξη εφαρμογών σε προγραμματιστικό περιβάλλον". Γ' Λυκείου Τεχνολογική Κατεύθυνση – Βιβλίο μαθητή.

Αντωνάκος Νίκος, Βογιατζής Ιωάννης, Κατοπόδης Ιωάννης, Πατριαρχέας Κυριάκος. Υπουργείο Παιδείας και Θρησκευμάτων – Παιδαγωγικό Ινστιτούτο. Αθήνα 1999.

4. "Προγραμματισμός υπολογιστών με Visual Basic". Βιβλίο μαθητή. Τομέας ηλεκτρονικών 2ος κύκλος.

Βουτυράς Γεώργιος, Βιδιαδάκης Ανδρέας, Ματζάκος Πέτρος, Σκουρλάς Χρήστος. Οργανισμός Εκδόσεων Διδακτικών Βιβλίων, Αθήνα.

5. "Computer programming in the Pascal Language". By Neal Golden, Antonio M. Lopez, Jr. Harcourt Brace Jovanovich, Publishers.
6. "Pascal – The software fundamentals of computer science". By Richard A, Meyers. Prentice-Hall International Editions.
7. "Software Engineering. A Practitioner's Approach" .Third Edition. By Roger S. Pressman. McGraw Hill International Edition – 1992.
8. "Computing: An Introduction to Procedures and Procedure – Followers". By Fred M. Tonge and Julian Feldman. McGraw-Hill Book Company.
9. "Computers and Information Systems". Third edition. By H. L Capron and John D. Perron. The Benjamin / Cummings Publishing Company, Inc.
10. "FORTRAN a structured, disciplined style" . By Gordon B. Davis, Thomas R. Hoffmann. McGraw – Hill Company.
11. "Computer programming in the Basic Language". Third Edition. By Neal Golden. Harcourt Brace Jovanovich, Publishers.
12. "Computer programming in the Basic Language". Teachers manual and resource guide. Third Edition. By Neal Golden. Harcourt Brace Jovanovich, Publishers.
13. "Basic for students using the IBM PC". Second edition By Michael

Trombetta. Addison – Wesley Publishing Company.

14. “Microsoft Basic using Modular Structure”. Third Edition. By: Julia Case Bradley. WCB –1991.
15. “Structured VAX Basic and Basic-Plus”. Second Edition. By: Teglovic Jr. and Kenneth D. Douglas. IRWIN-1987.
16. ”Understanding Computer Science for Advanced Level”. Fourth Edition. By: R. Bradley. Stanley Thornes Publishers Ltd 1999.

ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΠΟΛΙΤΙΣΜΟΥ
ΥΠΗΡΕΣΙΑ ΕΞΕΤΑΣΕΩΝ

ΠΑΓΚΥΠΡΙΕΣ ΕΞΕΤΑΣΕΙΣ 2018

Μάθημα: ΠΛΗΡΟΦΟΡΙΚΗ (15)

Ημερομηνία και ώρα εξέτασης: 04/06/2018

08:00 – 11:00

ΤΟ ΕΞΕΤΑΣΤΙΚΟ ΔΟΚΙΜΙΟ ΑΠΟΤΕΛΕΙΤΑΙ ΑΠΟ ΔΕΚΑΤΡΕΙΣ (13) ΣΕΛΙΔΕΣ.

ΟΔΗΓΙΕΣ

- Να απαντήσετε σε όλες τις ερωτήσεις.
- Το εξεταστικό δοκίμιο αποτελείται από τρία μέρη Α', Β' και Γ'.
- Το μέρος Α΄ αποτελείται από έξι (6) ερωτήσεις και κάθε ερώτηση βαθμολογείται με πέντε (5) μονάδες.
- Το μέρος Β΄ αποτελείται από τέσσερις (4) ερωτήσεις και κάθε ερώτηση βαθμολογείται με δέκα (10) μονάδες.
- Το μέρος Γ΄ αποτελείται από δύο (2) ερωτήσεις και κάθε ερώτηση βαθμολογείται με δεκαπέντε (15) μονάδες.
- Επιτρέπεται η χρήση μη προγραμματιζόμενης υπολογιστικής μηχανής.
- Οι μοναδικές βιβλιοθήκες που επιτρέπονται στη δημιουργία προγραμμάτων, είναι οι <iostream>, <fstream>, <string>, <iomanip>, <climits> και <cmath>.
- Η έκδοση της γλώσσας προγραμματισμού C++ που μπορεί να χρησιμοποιήσει ο υποψήφιος είναι η C++98 (ISO/IEC 14882:1998). Οποιεσδήποτε επεκτάσεις (extensions) παρέχονται από κάποιους μεταγλωττιστές (compilers) δεν μπορούν να χρησιμοποιηθούν.

Τα σύμβολα των Λογικών Διαγραμμάτων και των Λογικών Κυκλωμάτων, καθώς και το λεκτικό περιεχόμενό τους μπορούν να γίνουν με μολύβι.

ΜΕΡΟΣ Α'

ΑΣΚΗΣΗ 1:

Ένα σχολείο θα πραγματοποιήσει εκδρομή για τους μαθητές του χρησιμοποιώντας λεωφορεία. Η τιμή ενοικιάσεως **για κάθε** λεωφορείο είναι **50 ευρώ**. Αν χρησιμοποιηθούν **περισσότερα από 15 λεωφορεία** δίνεται έκπτωση **20% πάνω** στη **συνολική τιμή**, διαφορετικά δίνεται έκπτωση **10%**.

Να σχεδιάσετε **λογικό διάγραμμα**, το οποίο:

(α) Να δέχεται τον **αριθμό των λεωφορείων** που θα χρησιμοποιηθούν για να πραγματοποιηθεί η εκδρομή.

(Βαθμός 1)

(β) Να υπολογίζει το **ποσό** της **έκπτωσης** καθώς και το **τελικό ποσό** (τελικό ποσό = συνολική τιμή - έκπτωση) που πρέπει να πληρώσει το σχολείο.

(Βαθμοί 3)

(γ) Να τυπώνει το **ποσό** της **έκπτωσης** καθώς και το **τελικό ποσό** που πρέπει να πληρώσει το σχολείο, όπως έχουν υπολογιστεί στο ερώτημα (β).

(Βαθμός 1)

ΑΣΚΗΣΗ 2:

Δίνονται ο **δεκαδικός** αριθμός **Α=36** και οι **δυαδικοί** αριθμοί **Β=01010101** και **Γ=01010011.**

(α) Να δείξετε ότι η αντίστοιχη τιμή του **δεκαδικού αριθμού Α** στο δυαδικό σύστημα είναι **$(100100)_2$, σημειώνοντας τα βήματα** που ακολουθήσατε για να φτάσετε στο συγκεκριμένο αποτέλεσμα.

(Βαθμοί 2)

(β) Να δείξετε ότι η αντίστοιχη τιμή του **δυαδικού αριθμού Β** στο δεκαδικό σύστημα είναι **$(85)_{10}$, σημειώνοντας τα βήματα** που ακολουθήσατε για να φτάσετε στο συγκεκριμένο αποτέλεσμα.

(Βαθμοί 2)

(γ) Να γράψετε το **συμπλήρωμα ως προς 2** του δυαδικού αριθμού **Γ**.

(Βαθμός 1)

ΑΣΚΗΣΗ 3:

(α) Να γράψετε το αποτέλεσμα της πιο κάτω εντολής, η οποία είναι γραμμένη στη γλώσσα προγραμματισμού C++, **σημειώνοντας τα βήματα** που ακολουθήσατε για να φτάσετε στο συγκεκριμένο αποτέλεσμα:

```
cout<<trunc(-6.5+pow(2,2))+abs(round(-sqrt(67)));
```

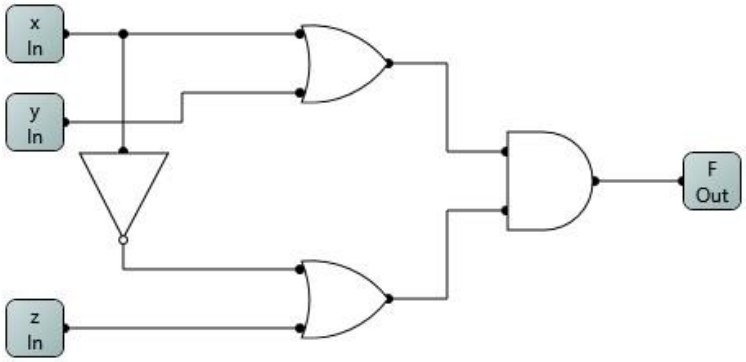
(Βαθμοί 2)

(β) Οι μεταβλητές a, b και c είναι τύπου integer και έχουν τις ακόλουθες τιμές: a = -3, b = 2 και c = 4. Να γράψετε τις τιμές που θα έχουν οι λογικές μεταβλητές x και y (boolean), όταν εκτελεστούν οι πιο κάτω εντολές στη γλώσσα προγραμματισμού C++ :

```
i)  x=(b!=(3*pow(a,2)-1))&&((abs(a)+trunc((float)b/c))==b);
ii) y=!((b+c)>=a) || (sqrt(b*8)<=round(a+7.3));
```

(Βαθμοί 2)

(γ) Να γράψετε τη **λογική συνάρτηση F(x,y,z)** που αντιστοιχεί στο πιο κάτω λογικό κύκλωμα:



(Βαθμός 1)

ΑΣΚΗΣΗ 4:

(α) Να μετατρέψετε τις πιο κάτω **λεκτικές προτάσεις** στις αντίστοιχες **λογικές εκφράσεις** στη γλώσσα προγραμματισμού C++:

i) Η μεταβλητή **grade** να είναι μεταξύ του **1** και του **100** **συμπεριλαμβανομένων**.

ii) Η μεταβλητή **z** να είναι **ζυγός αριθμός** και να **μην ισούται** με **8**.

(Βαθμός 1)

(β) Το πιο κάτω πρόγραμμα στη γλώσσα προγραμματισμού C++ δέχεται **10 ακέραιους αριθμούς** και υπολογίζει και παρουσιάζει τους αριθμούς που είναι **πολλαπλάσιοι του 5** καθώς και το **πλήθος** των αριθμών αυτών.
Στο πρόγραμμα υπάρχουν λογικά ή/και συντακτικά λάθη. Να γράψετε στο τετράδιο απαντήσεών σας **τέσσερα (4)** από αυτά, αναφέροντας τον αριθμό της γραμμής στην οποία εμφανίζεται το κάθε λάθος μαζί με τη διορθωμένη εντολή.

```
/*1*/  #include<iostream>
/*2*/  using namespace std;
/*3*/  int main(){
/*4*/  int i=0,count=0;
/*5*/  do {
/*6*/    cout<<"Δώσε ένα ακέραιο αριθμό:";
/*7*/    cin>>num;
 /*8*/   if (num%5=0){
 /*9*/    count++;
 /*10*/   cout<<"Ο αριθμός "<<num<<" είναι πολλαπλάσιο του 5";
 /*11*/  }
 /*12*/  i++;
 /*13*/   } while (i>10);
 /*14*/  cin<<count<<endl;
 /*15*/ return 0;
 /*16*/ }
```

(Βαθμοί 2)

(γ) Δίνεται το πιο κάτω τμήμα προγράμματος στη γλώσσα προγραμματισμού C++. Χωρίς να αλλοιωθεί η λογική του, να γράψετε το αντίστοιχο τμήμα προγράμματος χρησιμοποιώντας την περιπτωσιακή δομή **switch**.

```
if (flag=='Z')
   cout<<"ΖΕΣΤΟ"<<endl;
else if (flag=='X')
   cout<<"ΧΛΙΑΡΟ"<<endl;
else if (flag=='K')
   cout<<"ΚΡΥΟ"<<endl;
else
   cout<<"Λανθασμένη τιμή"<<endl;
```

(Βαθμοί 2)

ΑΣΚΗΣΗ 5:

Δίνεται το πιο κάτω πρόγραμμα στη γλώσσα προγραμματισμού C++.

```
/*1*/  #include<iostream>
/*2*/  #include<string>
/*3*/  using namespace std;
/*4*/  int main(){
/*5*/       string st1,st2;
/*6*/       int s1,s2;
/*7*/       getline(cin,st1);
/*8*/       cout<<st1<<endl;
/*9*/       cin>>st2;
/*10*/      cout<<st2<<endl;
/*11*/      s1=st1.size();
/*12*/      s2=st2.size();
/*13*/      cout<<s1<<endl;
/*14*/      cout<<s2<<endl;
/*15*/      [ A ]
/*16*/  return 0;
/*17*/  }
```

Ας υποθέσουμε ότι για τις συμβολοσειρές **st1** και **st2,** εισάγονται από το πληκτρολόγιο οι φράσεις **"Εξετάσεις Πληροφορικής"** και **"Προγραμματισμός στη C++"** αντίστοιχα**.**

(α) Να γράψετε στο τετράδιο απαντήσεών σας τα αποτελέσματα των εντολών που βρίσκονται στις γραμμές **8** και **10** του πιο πάνω προγράμματος.

(Βαθμοί 2)

(β) Να γράψετε στο τετράδιο απαντήσεών σας τα αποτελέσματα των εντολών που βρίσκονται στις γραμμές **13** και **14** του πιο πάνω προγράμματος.

(Βαθμοί 2)

(γ) Να γράψετε στο τετράδιο απαντήσεών σας την εντολή που πρέπει να τοποθετηθεί στη **θέση Α (γραμμή 15),** ώστε να **τυπωθούν** οι **τρεις (3)** πρώτοι χαρακτήρες της γραμματοσειράς **st1.**

(Βαθμός 1)

ΑΣΚΗΣΗ 6:

Δίνεται το πιο κάτω πρόγραμμα στη γλώσσα προγραμματισμού C++.

```
#include<iostream>
#include<cmath>
#include<iomanip>
using namespace std;
```

Α

```
{
     float ipot;
     ipot=sqrt(pow(a,2)+pow(b,2));
     return ipot;
}

void evperim(int a,int b,float ipot,float &evad,float &perim){
     evad=a*b/2;
     perim=a+b+ipot;
}
```

Β

```
int main(){
 int plevra1,plevra2;
 float ipotin,evadon,perimeter;
 string message;
 cout<<"Δώσε τις κάθετες πλευρές του τριγώνου:";
 cin>>plevra1>>plevra2;
 ipotin=ipotinousa(plevra1,plevra2);
```

Γ

```
 cout<<"Υποτείνουσα: "<<fixed<<setprecision(2)<<ipotin<<endl;
 cout<<"Εμβαδόν: "<<fixed<<setprecision(2)<<evadon<<endl;
 cout<<"Περίμετρος: "<<fixed<<setprecision(2)<<perimeter;
 message=check(evadon,perimeter);
 cout<<message;
return 0;
}
```

(α) Να γράψετε στο τετράδιο απαντήσεών σας την **επικεφαλίδα** της συνάρτησης **ipotinousa**, που πρέπει να τοποθετηθεί στη **θέση Α,** έτσι ώστε η συνάρτηση να δέχεται τις **δύο κάθετες πλευρές** ενός ορθογωνίου τριγώνου και να υπολογίζει και να επιστρέφει στην κύρια συνάρτηση (main) την **υποτείνουσά** του.

(Βαθμός 1)

(β) Να γράψετε στο τετράδιο απαντήσεών σας την **εντολή** που πρέπει να τοποθετηθεί στη **θέση Γ**, η οποία **καλεί** τη συνάρτηση **evperim** που δέχεται τις **δύο κάθετες πλευρές** και την **υποτείνουσα** του ορθογωνίου τριγώνου και υπολογίζει και επιστρέφει στην κύρια συνάρτηση (main) το **εμβαδό** και την **περίμετρό** του.

(Βαθμός 1)

(γ) Να γράψετε στο τετράδιο απαντήσεών σας τη συνάρτηση **check**, που πρέπει να τοποθετηθεί στη **θέση Β**, η οποία δέχεται το **εμβαδόν** και την **περίμετρο** του ορθογωνίου τριγώνου και επιστρέφει στην κύρια συνάρτηση (main) το μήνυμα **«Είναι ίσα»** όταν το εμβαδόν **είναι ίσο** με την περίμετρο, διαφορετικά επιστρέφει το μήνυμα **«Άνισα»**

(Βαθμοί 3)

ΤΕΛΟΣ Α' ΜΕΡΟΥΣ
ΑΚΟΛΟΥΘΕΙ ΤΟ ΜΕΡΟΣ Β'

ΜΕΡΟΣ Β'

ΑΣΚΗΣΗ 7:

Δίνεται η πιο κάτω λογική συνάρτηση:

$$F\ (A,B,C) = A'B'C' + AB'C + ABC + A'BC'$$

(α) Να δημιουργήσετε τον **πίνακα αληθείας** της συνάρτησης.

(Βαθμοί 3)

(β) Να σχεδιάσετε τον **χάρτη Karnaugh** που αντιστοιχεί στη συνάρτηση.

(Βαθμοί 3)

(γ) Να **απλοποιήσετε** τη συνάρτηση (στην πιο απλή της μορφή), με τη χρήση **χάρτη Karnaugh** και να σχεδιάσετε το **λογικό κύκλωμα** που θα προκύψει μετά την απλοποίηση.

(Βαθμοί 4)

ΑΣΚΗΣΗ 8:

Να γράψετε πρόγραμμα στη γλώσσα προγραμματισμού C++, το οποίο:

(α) Να δημιουργεί μια εγγραφή με το όνομα **mathitis,** η οποία να περιλαμβάνει τα μέλη: το **όνομα** του μαθητή **(string)**, το **τμήμα** του μαθητή (**string**), τον **αριθμό** των **απουσιών** του (**integer**), καθώς και το αν είναι **άπορος** ή όχι (**boolean**). Στη συνέχεια, να ορίζει ένα μονοδιάστατο πίνακα εγγραφών **25** θέσεων τύπου **mathitis** με το όνομα **s_class**.

(Βαθμοί 3)

(β) Να δέχεται τα πιο πάνω στοιχεία για **25** μαθητές από το πληκτρολόγιο και να τα αποθηκεύει στον μονοδιάστατο πίνακα εγγραφών **s_class**, όπως αυτός έχει οριστεί στο ερώτημα (α).

(Βαθμοί 2)

(γ) Να υπολογίζει και να τυπώνει:

(i) τον **μέσο όρο** των απουσιών των **άπορων** μαθητών. Ο μέσος όρος να τυπώνεται με **2 δεκαδικά ψηφία**. Να θεωρήσετε ότι υπάρχει τουλάχιστον ένας άπορος μαθητής με απουσίες.

(ii) τα **ονόματα** των **άπορων** μαθητών του **Β5**. Στην περίπτωση που δεν βρέθηκε κανένας άπορος μαθητής να τυπώνεται το μήνυμα **«Δεν υπάρχουν άποροι μαθητές στο Β5»**.

(Βαθμοί 5)

Το πρόγραμμα πρέπει να εμφανίζει στην οθόνη τα κατάλληλα μηνύματα για την εισαγωγή των δεδομένων και την εξαγωγή των αποτελεσμάτων σύμφωνα με τα πιο κάτω παραδείγματα.

Παράδειγμα Εισόδου 1 (πληκτρολόγιο) (για 4 μαθητές μόνο)

```
Μαρίνα Β3 15 1
Γιώργος Β5 20 1
Μάριος Β5 40 1
Μαρία Β2 20 0
```

Παράδειγμα Εξόδου 1 (οθόνη) (για 4 μαθητές μόνο)

```
Δώσε στοιχεία μαθητή: Όνομα, τμήμα, αριθμό απουσιών, άπορος:
Μέσος όρος απουσιών άπορων μαθητών:25.00
Γιώργος
Μάριος
```

Παράδειγμα Εισόδου 2 (πληκτρολόγιο) **(για 4 μαθητές μόνο)**
Μαρίνα Β3 15 1 Γιώργος Β5 17 0 Μάριος Β5 40 0 Μαρία Β2 25 1
Παράδειγμα Εξόδου 2 (οθόνη) **(για 4 μαθητές μόνο)**
Δώσε στοιχεία μαθητή: Όνομα, τμήμα, αριθμό απουσιών, άπορος: Μέσος όρος απουσιών άπορων μαθητών:20.00 Δεν υπάρχουν άποροι μαθητές στο Β5

ΑΣΚΗΣΗ 9:

Ο κωδικός που ελέγχει την πρόσβαση της κύριας εισόδου ενός κτιρίου είναι 16-ψήφιος και αποτελείται από μονοψήφιους αριθμούς από το 0 μέχρι και το 9 συμπεριλαμβανομένων, σε μορφή τετραγωνικού πίνακα 4 γραμμών και 4 στηλών. Να γράψετε πρόγραμμα στη γλώσσα προγραμματισμού C++, το οποίο:

(α) Να δέχεται **16 ακέραιους θετικούς αριθμούς** από το **0** μέχρι και το **9** συμπεριλαμβανομένων και να τους καταχωρίζει σε τετραγωνικό πίνακα **4 γραμμών** και **4 στηλών** με το όνομα **code.** Να θεωρήσετε ότι τα δεδομένα δίνονται σωστά και δεν χρειάζεται οποιοσδήποτε έλεγχος.

(Βαθμοί 3)

(β) Να δηλώνει και να χρησιμοποιεί τη συνάρτηση **pzeros,** η οποία να δέχεται ως παράμετρο τον δισδιάστατο πίνακα **code** και να επιστρέφει στην κύρια συνάρτηση (main): i) τον **συνολικό αριθμό** των μηδενικών (πλήθος) που υπάρχουν στην κύρια και στη δευτερεύουσα διαγώνιο του πίνακα, και ii) τον **αριθμό των υπόλοιπων μηδενικών** του πίνακα (δηλαδή αυτών που **δεν ανήκουν** ούτε στην κύρια ούτε και στη δευτερεύουσα διαγώνιο του πίνακα).

(Βαθμοί 5)

(γ) Στην περίπτωση που ο **συνολικός αριθμός** των μηδενικών που **ανήκουν** στην κύρια και στη δευτερεύουσα διαγώνιο του πίνακα είναι **ίσος** με το πλήθος των υπόλοιπων μηδενικών που υπάρχουν στον πίνακα, να εμφανίζει στην οθόνη το μήνυμα **«Ελεύθερη πρόσβαση»** διαφορετικά να εμφανίζει στην οθόνη το μήνυμα **«Λάθος κωδικός».**

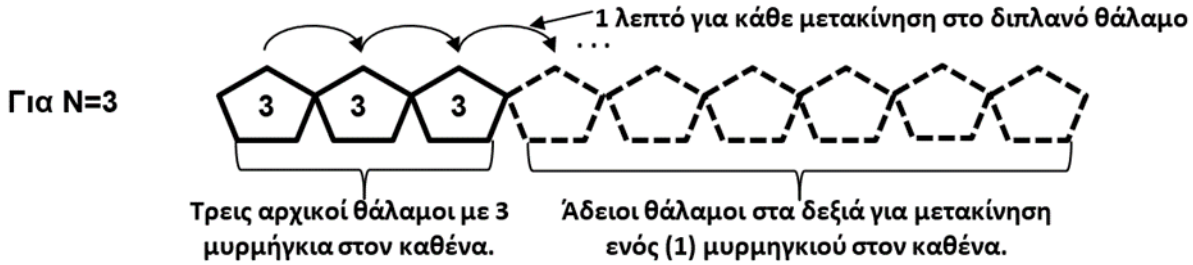
(Βαθμοί 2)

Το πρόγραμμα πρέπει να εμφανίζει στην οθόνη τα κατάλληλα μηνύματα για την εισαγωγή των δεδομένων και την εξαγωγή των αποτελεσμάτων σύμφωνα με τα πιο κάτω παραδείγματα.

Παράδειγμα Εισόδου 1 (πληκτρολόγιο)	**Παράδειγμα Εξόδου 1(οθόνη)**
4 2 5 0 0 0 1 8 6 3 0 0 2 4 0 9	Δώσε 16 ακέραιους αριθμούς από 0 μέχρι 9: Ελεύθερη πρόσβαση
Παράδειγμα Εισόδου 2 (πληκτρολόγιο)	**Παράδειγμα Εξόδου 2(οθόνη)**
2 6 0 0 3 0 5 6 2 9 0 1 4 0 9 7	Δώσε 16 ακέραιους αριθμούς από 0 μέχρι 9: Λάθος κωδικός

ΑΣΚΗΣΗ 10:

Μια φωλιά μυρμηγκιών αποτελείται από ένα αριθμό διαδοχικών θαλάμων. Αρχικά, σε κάθε ένα από τους Ν πρώτους θαλάμους υπάρχουν Ν μυρμήγκια, ενώ οι υπόλοιποι θάλαμοι είναι άδειοι. Θεωρούμε ότι ο συνολικός αριθμός των θαλάμων είναι ίσος με τον συνολικό αριθμό των μυρμηγκιών της φωλιάς. Στόχος των μυρμηγκιών είναι να μετακινηθούν έτσι ώστε, τελικά, να βρίσκεται ένα μυρμήγκι σε κάθε θάλαμο. Κάθε μυρμήγκι μπορεί σε κάθε ένα του βήμα να μετακινηθεί μόνο στον διπλανό **δεξιά** θάλαμο. Επίσης, ένα μυρμήγκι χρειάζεται ένα λεπτό για να μετακινηθεί από θάλαμο σε θάλαμο και κάθε λεπτό μετακινείται μόνο ένα μυρμήγκι. Σημειώνεται ότι κατά τη μετακίνηση των μυρμηγκιών κάθε θάλαμος μπορεί να φιλοξενήσει και περισσότερα από Ν μυρμήγκια.



Να γράψετε πρόγραμμα στη γλώσσα προγραμματισμού C++, το οποίο να δέχεται ένα θετικό ακέραιο αριθμό **Ν** μεταξύ του **2** και του **20 συμπεριλαμβανομένων (2<=N<=20)**. Να θεωρήσετε ότι τα δεδομένα δίνονται σωστά και δεν χρειάζεται οποιοσδήποτε έλεγχος. Ακολούθως να υπολογίζει και να τυπώνει:

(α) Τον **αριθμό των μυρμηγκιών** που βρίσκονται συνολικά στους αρχικούς θαλάμους.

(Βαθμοί 2)

(β) Τον **αριθμό των άδειων θαλάμων** που χρειάζονται τα μυρμήγκια για να μετακινηθούν έτσι ώστε να βρίσκεται ένα μυρμήγκι σε κάθε θάλαμο.

(Βαθμοί 3)

(γ) Τον **συνολικό χρόνο** που χρειάζονται τα μυρμήγκια για να μετακινηθούν έτσι ώστε σε κάθε θάλαμο να βρίσκεται μόνο ένα μυρμήγκι.

(Βαθμοί 5)

Το πρόγραμμα πρέπει να εμφανίζει στην οθόνη τα κατάλληλα μηνύματα για την εισαγωγή των δεδομένων και την εξαγωγή των αποτελεσμάτων σύμφωνα με τα πιο κάτω παραδείγματα.

Παράδειγμα Εισόδου 1 (πληκτρολόγιο) 3	**Παράδειγμα Εξόδου 1 (οθόνη)** Δώσε ένα ακέραιο αριθμό (N): Αριθμός μυρμηγκιών:9 Αριθμός άδειων θαλάμων:6 Συνολικός χρόνος (λεπτά):27
Παράδειγμα Εισόδου 2 (πληκτρολόγιο) 4	**Παράδειγμα Εξόδου 2 (οθόνη)** Δώσε ένα ακέραιο αριθμό (N): Αριθμός μυρμηγκιών:16 Αριθμός άδειων θαλάμων:12 Συνολικός χρόνος (λεπτά):96

ΤΕΛΟΣ Β' ΜΕΡΟΥΣ
ΑΚΟΛΟΥΘΕΙ ΤΟ ΜΕΡΟΣ Γ'

ΜΕΡΟΣ Γ'

ΑΣΚΗΣΗ 11:

Ο Δήμος Λευκωσίας για λόγους εξωραϊσμού του χώρου της τάφρου που βρίσκεται περιμετρικά της Πλατείας Ελευθερίας έχει προκηρύξει διαγωνισμό προσφορών για αγορά φυτών από διάφορα φυτώρια της Κύπρου. Στον διαγωνισμό αυτό έλαβαν μέρος **20** διαφορετικά **φυτώρια** από ολόκληρη την Κύπρο με **5 διαφορετικά είδη φυτών** το κάθε φυτώριο. Τα ονόματα των 20 φυτωρίων καταχωρίζονται σε ένα **μονοδιάστατο** πίνακα με το όνομα **names.** Οι τιμές για τα 5 διαφορετικά είδη φυτών καταχωρίζονται σε ένα άλλο πίνακα **πραγματικών αριθμών δύο διαστάσεων** με το όνομα **prices** που είναι παράλληλος με τον πρώτο πίνακα.

Παράδειγμα:

	names
0	DFlowerShop
1	TGarden
2	ModGarden
.	.
.	.
18	ModernPlants
19	FlowerArt

	prices				
	0	1	2	3	4
0	8.30	12.00	10.50	7.25	6.50
1	9.20	11.50	11.00	6.75	5.90
2	8.30	11.20	10.80	6.60	6.50
.	.	.	.	.	.
.	.	.	.	.	.
18	9.10	11.10	11.30	7.75	7.90
19	8.30	10.20	10.80	6.80	6.00

Η **1η γραμμή (γραμμή με δείκτη 0)** του πίνακα **prices** αντιπροσωπεύει τις τιμές για 5 διαφορετικά είδη φυτών που έχει δώσει το φυτώριο **DFlowerShop** π.χ. η τιμή για το 1ο είδος φυτού είναι €8.30, του 2ου €12.00, του 3ου €10.50, του 4ου €7.25 και του 5ου €6.50.

Να γράψετε πρόγραμμα στη γλώσσα προγραμματισμού C++, το οποίο:

α) Να καταχωρίζει στον μονοδιάστατο πίνακα **names** το όνομα του κάθε φυτώριου και στον παράλληλο δισδιάστατο πίνακα **prices** τις τιμές για το κάθε ένα από τα 5 διαφορετικά είδη φυτών. Να θεωρήσετε ότι όλα τα στοιχεία δίνονται σωστά και δεν χρειάζεται οποιοσδήποτε έλεγχος.

(Βαθμοί 3)

β) Να υπολογίζει τη **συνολική τιμή** και για τα 5 διαφορετικά είδη φυτών του **κάθε φυτωρίου** και να την τοποθετεί σε ένα άλλο παράλληλο μονοδιάστατο πίνακα με το όνομα **totals**.

(Βαθμοί 3)

γ) Να χρησιμοποιεί τη **συνάρτηση m10,** η οποία θα λαμβάνει ως παράμετρο από την κύρια συνάρτηση (main) τον πίνακα **totals**, και να υπολογίζει και να επιστρέφει στην κύρια συνάρτηση (main) το **πλήθος** των φυτωρίων που ο **μέσος όρος** των τιμών που έχουν προσφέρει και για τα 5 διαφορετικά είδη φυτών είναι **μικρότερος** από **10.** Το πλήθος αυτό να τυπώνεται στο κυρίως μέρος του προγράμματος.

(Βαθμοί 4)

δ) Να ταξινομεί τους πίνακες **names** και **totals** σε **φθίνουσα** σειρά με βάση τις συνολικές τιμές του πίνακα **totals**. Η ταξινόμηση να γίνει με τη χρήση του αλγόριθμου **εισαγωγής** (**insertion sort**). Ακολούθως, να εμφανίζει στην οθόνη τα **ονόματα** των **τριών (3) φυτωρίων** που έχουν δώσει τη **χαμηλότερη συνολική τιμή** και για τα 5 διαφορετικά είδη φυτών. Να θεωρήσετε ότι υπάρχουν μόνο τρία φυτώρια που έδωσαν τις τρεις (3) χαμηλότερες συνολικές τιμές.

(Βαθμοί 5)

Το πρόγραμμα πρέπει να εμφανίζει στην οθόνη τα κατάλληλα μηνύματα για την εισαγωγή των δεδομένων και την εξαγωγή των αποτελεσμάτων σύμφωνα με τα πιο κάτω παραδείγματα.

Παράδειγμα Εισόδου (για 5 φυτώρια και 5 διαφορετικά είδη φυτών)
(πληκτρολόγιο)

```
DFlowerShop 8.30 12.00 10.50 7.25 6.50
TGarden 9.20 11.50 11.00 6.75 5.90
ModGarden 8.30 11.20 10.80 6.60 6.50
DesignPlants 9.00 11.30 12.50 9.50 8.60
FlowerVision 8.20 10.30 10.90 6.40 6.00
```

Παράδειγμα Εξόδου (για 5 φυτώρια και 5 διαφορετικά είδη φυτών)
(οθόνη)

```
Δώσε το όνομα κάθε φυτωρίου και τις τιμές για το κάθε ένα από
τα 5 διαφορετικά είδη φυτών:
Πλήθος φυτωρίων με μέσο όρο τιμών <10:4
Τα 3 φυτώρια με την χαμηλότερη συνολική τιμή...
TGarden
ModGarden
FlowerVision
```

ΑΣΚΗΣΗ 12:

Η εταιρεία πώλησης μεταχειρισμένων αυτοκινήτων **CarSalesLtd**, στην προσπάθειά της για καλύτερο έλεγχο και συντονισμό των εργασιών της, σας ζητά να δημιουργήσετε ένα πρόγραμμα στη γλώσσα προγραμματισμού C++, το οποίο:

(α) Να δημιουργεί μια εγγραφή με το όνομα **car,** η οποία να περιλαμβάνει τα πιο κάτω μέλη:

- Μάρκα (string)
- Αριθμός εγγραφής (string)
- Χρώμα (string)
- Τιμή (integer)

Στη συνέχεια να διαβάζει τα στοιχεία (μάρκα, αριθμό εγγραφής, χρώμα και τιμή) για τα **100** αυτοκίνητα που διαθέτει προς πώληση από το αρχείο **askisi12IN.txt** και τα στοιχεία αυτά να τα αποθηκεύει σ' ένα πίνακα εγγραφών τύπου **car**, με το όνομα **carSt**.

(Βαθμοί 4)

(β) Να διαβάζει από το αρχείο **afxisi.txt** τον **αριθμό εγγραφής** και το **ποσό** της **αύξησης** της **τιμής** για το κάθε αυτοκίνητο. Το αρχείο αυτό περιέχει μόνο τα αυτοκίνητα που θα έχουν αύξηση στην αρχική τους τιμή, δηλαδή υπάρχει περίπτωση, η τιμή κάποιων αυτοκινήτων που υπάρχουν στο αρχείο **askisi12IN.txt** να μην αυξηθεί. Το ποσό αύξησης της τιμής για το κάθε αυτοκίνητο θα προστίθεται στην αρχική τιμή. Στη συνέχεια, όλα τα στοιχεία και των 100 αυτοκινήτων να καταχωρίζονται στο αρχείο **askisi12OUT.txt**. Να θεωρήσετε ότι στο αρχείο **afxisi.txt** δεν μπορούν να υπάρξουν καταχωρήσεις για άλλα αυτοκίνητα εκτός από τα 100 αυτοκίνητα που έχουν ήδη καταχωρηθεί στο αρχείο **askisi12IN.txt.**

(Βαθμοί 4)

(γ) Να υπολογίζει και να εμφανίζει στην οθόνη την **τιμή** και το **χρώμα** του **ακριβότερου** αυτοκινήτου που διαθέτει η εταιρεία. Να θεωρήσετε ότι υπάρχει μόνο ένα τέτοιο αυτοκίνητο.

(Βαθμοί 3)

(δ) Να δέχεται τον **αριθμό εγγραφής** ενός αυτοκινήτου και να εντοπίζει και να εμφανίζει στην οθόνη τη **μάρκα** του αυτοκινήτου. Σε περίπτωση που δεν βρεθεί αυτοκίνητο με τον συγκεκριμένο αριθμό εγγραφής, να εμφανίζεται το μήνυμα **«Δεν υπάρχει τέτοιο αυτοκίνητο»**. Η αναζήτηση να γίνεται με τη χρήση του αλγόριθμου **σειριακής** αναζήτησης **(sequential search)**. Σημειώνεται ότι ο αριθμός εγγραφής είναι μοναδικός.

(Βαθμοί 4)

Το πρόγραμμα πρέπει να εμφανίζει στην οθόνη τα κατάλληλα μηνύματα για την εισαγωγή των δεδομένων και την εξαγωγή των αποτελεσμάτων σύμφωνα με τα πιο κάτω παραδείγματα.

Παράδειγμα Εισόδου **(για 10 αυτοκίνητα μόνο)**

(από αρχείο askisi12IN.txt)

```
HONDA KNP231 WHITE 3250
TOYOTA KME134 BLACK 4200
BMW MYR654 RED 10200
VW LKR343 GREY 7900
VOLVO MOE899 BLUE 9400
MAZDA KNP805 YELLOW 6750
MERCEDES KLL248 WHITE 12500
HONDA KRZ378 BLUE 10500
NISSAN MYN555 BLACK 11600
BMW KNR111 GREY 9900
```

(από αρχείο afxisi.txt)

```
MYR654 200
MOE899 150
KRZ378 100
KNR111 300
KLL248 500
```

(από πληκτρολόγιο)

```
KME134
```

Παράδειγμα Εξόδου **(για 10 αυτοκίνητα μόνο)**

(στο αρχείο askisi12OUT.txt)

```
HONDA KNP231 WHITE 3250
TOYOTA KME134 BLACK 4200
BMW MYR654 RED 10400
VW LKR343 GREY 7900
VOLVO MOE899 BLUE 9550
MAZDA KNP805 YELLOW 6750
MERCEDES KLL248 WHITE 13000
HONDA KRZ378 BLUE 10600
NISSAN MYN555 BLACK 11600
BMW KNR111 GREY 10200
```

(στη οθόνη)

```
Τιμή και χρώμα ακριβότερου αυτοκινήτου:13000 WHITE
Δώσε αριθμό εγγραφής αυτοκινήτου:
TOYOTA
```

ΤΕΛΟΣ ΕΞΕΤΑΣΤΙΚΟΥ ΔΟΚΙΜΙΟΥ

ΤΥΠΟΛΟΓΙΟ ΣΥΝΑΡΤΗΣΕΩΝ ΣΤΗ ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ C++

ΣΥΝΑΡΤΗΣΕΙΣ ΤΗΣ ΒΙΒΛΙΟΘΗΚΗΣ <cmath>		
Συνάρτηση	**Χρήση**	**Παράμετροι**
sqrt(x)	Επιστρέφει την **τετραγωνική ρίζα** του αριθμού x. Η επιστρεφόμενη τιμή είναι πραγματικός αριθμός.	Ένας θετικός αριθμός (ακέραιος ή πραγματικός)
abs(x)	Επιστρέφει την **απόλυτη τιμή** του αριθμού x. Η επιστρεφόμενη τιμή εξαρτάται από τον τύπο του αριθμού x.	Ένας αριθμός (ακέραιος ή πραγματικός)
pow(x,y)	Επιστρέφει το **αποτέλεσμα** της **δύναμης x^y**. Η επιστρεφόμενη τιμή είναι πραγματικός αριθμός.	Δύο πραγματικοί αριθμοί
trunc(x)	Επιστρέφει το **ακέραιο μέρος** του αριθμού x σε πραγματική μορφή, **αγνοώντας το δεκαδικό μέρος του**.	Ένας πραγματικός αριθμός
round(x)	Επιστρέφει το **ακέραιο μέρος** του αριθμού x σε πραγματική μορφή, **στρογγυλοποιημένο** στην **πλησιέστερη τιμή.**	Ένας πραγματικός αριθμός
ΣΥΝΑΡΤΗΣΕΙΣ ΤΗΣ ΒΙΒΛΙΟΘΗΚΗΣ <string>		
size()	**Επιστρέφει** το **μέγεθος** μιας συμβολοσειράς. Η επιστρεφόμενη τιμή είναι **ακέραιος αριθμός** που συμβολίζει από πόσα bytes αποτελείται μια συμβολοσειρά.	Καμία παράμετρος
clear()	**Διαγράφει** το **περιεχόμενο** μιας συμβολοσειράς. Δεν επιστρέφει τίποτα.	Καμία παράμετρος
empty()	**Ελέγχει** αν μια συμβολοσειρά είναι **άδεια**. Η επιστρεφόμενη τιμή είναι τύπου **Boolean**.	Καμία παράμετρος
getline(x,y)	**Αποθηκεύει ολόκληρη** μια συμβολοσειρά που μπορεί να εισαχθεί από το πληκτρολόγιο ή από αρχείο (x) στο αντικείμενο y.	Η 1η παράμετρος (x) αφορά στη μέθοδο εισαγωγής της συμβολοσειράς (π.χ. από το πληκτρολόγιο ή από αρχείο) και η 2η παράμετρος (y) αφορά στο αντικείμενο στο οποίο θα αποθηκευτεί η συμβολοσειρά, η οποία έχει διαβαστεί αρχικά.
ΣΤΑΘΕΡΕΣ ΤΗΣ ΒΙΒΛΙΟΘΗΚΗΣ <climits>		
INT_MAX	**Μέγιστο αριθμητικό** όριο μεταβλητής ή σταθεράς τύπου **integer.** Η ακριβής αριθμητική τιμή της είναι **32767 (**στα **2 bytes)** ή **2147483647 (**στα **4 bytes)**	
INT_MIN	**Ελάχιστο αριθμητικό** όριο μεταβλητής ή σταθεράς τύπου **integer.** Η ακριβής αριθμητική τιμή της είναι **-32767 (**στα **2 bytes)** ή **-2147483647 (**στα **4 bytes)**	

ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΠΟΛΙΤΙΣΜΟΥ
ΥΠΗΡΕΣΙΑ ΕΞΕΤΑΣΕΩΝ

ΠΑΓΚΥΠΡΙΕΣ ΕΞΕΤΑΣΕΙΣ

ΔΕΙΓΜΑ ΕΞΕΤΑΣΤΙΚΟΥ ΔΟΚΙΜΙΟΥ

Μάθημα: ΠΛΗΡΟΦΟΡΙΚΗ

Ημερομηνία και ώρα εξέτασης:

ΟΔΗΓΙΕΣ

- Να απαντήσετε σε όλες τις ερωτήσεις.
- Το εξεταστικό δοκίμιο αποτελείται από τρία μέρη Α', Β' και Γ'.
- Το μέρος Α΄ αποτελείται από έξι (6) ερωτήσεις και κάθε ερώτηση βαθμολογείται με πέντε (5) μονάδες.
- Το μέρος Β΄ αποτελείται από τέσσερις (4) ερωτήσεις και κάθε ερώτηση βαθμολογείται με δέκα (10) μονάδες.
- Το μέρος Γ΄ αποτελείται από δύο (2) ερωτήσεις και κάθε ερώτηση βαθμολογείται με δεκαπέντε (15) μονάδες.
- Επιτρέπεται η χρήση μη προγραμματιζόμενης υπολογιστικής μηχανής.
- Οι μοναδικές βιβλιοθήκες που επιτρέπονται στη δημιουργία προγραμμάτων, είναι οι <iostream>, <fstream>, <string>, <iomanip>, <climits> και <cmath>.
- Η έκδοση της γλώσσας προγραμματισμού C++ που μπορεί να χρησιμοποιήσει ο υποψήφιος είναι η C++98 (ISO/IEC 14882:1998). Οποιεσδήποτε επεκτάσεις (extensions) παρέχονται από κάποιους μεταγλωττιστές (compilers) δεν μπορούν να χρησιμοποιηθούν.

Τα σύμβολα των Λογικών Διαγραμμάτων και των Λογικών Κυκλωμάτων, καθώς και το λεκτικό περιεχόμενό τους μπορούν να γίνουν με μολύβι.

ΜΕΡΟΣ Α'

Άσκηση 1

Ένα κατάστημα πώλησης ηλεκτρονικών υπολογιστών διοργανώνει «Σαββατοκύριακο Προσφορών» για τα προϊόντα που πωλεί, ως εξής: αν ο πελάτης αγοράσει προϊόντα συνολικής αξίας πάνω από €600, τότε θα έχει 30% έκπτωση πάνω στη συνολική αξία των προϊόντων, διαφορετικά θα έχει έκπτωση μόνο 15%.

(α) Να σχεδιάσετε λογικό διάγραμμα που να δέχεται τη συνολική αξία των προϊόντων που θα αγοράσει ο πελάτης και να υπολογίζει και να παρουσιάζει την έκπτωση που θα πάρει καθώς και την τελική τιμή (συνολική αξία των προϊόντων – έκπτωση) που θα πληρώσει.

(Βαθμοί 3)

(β) Να γράψετε στη γλώσσα προγραμματισμού C++ την εντολή που ελέγχει αν η συνολική αξία των προϊόντων είναι πάνω από €600.

(Βαθμός 1)

(γ) Να γράψετε στη γλώσσα προγραμματισμού C++ την εντολή που παρουσιάζει την τελική τιμή που θα πληρώσει ο πελάτης.

(Βαθμός 1)

Άσκηση 2

Δίνονται οι δυαδικοί αριθμοί Α=10101010 και Β=01000111.

(α) Να μετατρέψετε την τιμή του αριθμού Α στην αντίστοιχη τιμή στο δεκαδικό σύστημα.

(Βαθμός 1)

(β) Να γράψετε το συμπλήρωμα ως προς 2 του αριθμού Α και του αριθμού Β.

(Βαθμοί 2)

(γ) Να υπολογίσετε και να γράψετε στο δυαδικό σύστημα το αποτέλεσμα της πρόσθεσης Α+Β.

(Βαθμοί 2)

Άσκηση 3

(α) Να γράψετε το αποτέλεσμα της πιο κάτω εντολής, η οποία είναι γραμμένη στη γλώσσα προγραμματισμού C++, σημειώνοντας τα βήματα που ακολουθήσατε για να φτάσετε στο συγκεκριμένο αποτέλεσμα:

```
cout<<round(-5.3)+3*pow(2,3)+abs(-12)-trunc(6.2+(5%2))+sqrt(16);
```

(Βαθμοί 2)

(β) Οι μεταβλητές a, b και c έχουν τις ακόλουθες τιμές: a=2, b=1 και c=-3.

Να γράψετε τις τιμές που θα έχουν οι λογικές μεταβλητές x και y (boolean), όταν εκτελεστούν οι πιο κάτω εντολές στη γλώσσα προγραμματισμού C++ :

```
ii)   x = (a==(2*a-1)) || ((abs(c)+2)!=a/b) && (a>=(a+pow(c,2)));
iii)  y = ((a+b)<=2*c) && !((a-b)>=4);
```

(Βαθμοί 2)

(γ) Να σχεδιάσετε το λογικό κύκλωμα που αντιστοιχεί στην πιο κάτω λογική συνάρτηση, χρησιμοποιώντας μόνο πύλες AND και OR δύο εισόδων.

```
F (a, b, c) = (a && b) || (a && c)
```

(Βαθμός 1)

Άσκηση 4

(α) Έστω A, B και C οι γωνιές ενός τριγώνου. Το τρίγωνο ονομάζεται ορθογώνιο αν μία από τις γωνιές είναι ίση με 90 μοίρες και το άθροισμα των γωνιών είναι 180 μοίρες. Να γράψετε τη λογική έκφραση η οποία ελέγχει το πιο πάνω.

(Βαθμός 1)

(β) Δίνεται το πιο κάτω τμήμα προγράμματος στη γλώσσα προγραμματισμού C++. Χωρίς να αλλοιωθεί η λογική του, να γράψετε στο τετράδιό σας αντίστοιχο τμήμα προγράμματος χρησιμοποιώντας τη δομή επανάληψης **while**.

```
for (int i=10; i>=0; i=-2)
     cout << i << endl;
```

(Βαθμοί 2)

(γ) Το πιο κάτω πρόγραμμα στη γλώσσα προγραμματισμού C++ δέχεται **10 ακέραιους αριθμούς** και υπολογίζει και παρουσιάζει το πλήθος των **περιττών** αριθμών. Στο πρόγραμμα υπάρχουν λογικά ή/και συντακτικά λάθη. Να γράψετε στο τετράδιο απαντήσεών σας **τέσσερα (4)** από αυτά, αναφέροντας τον αριθμό της γραμμής στην οποία εμφανίζεται το κάθε λάθος μαζί με τη διορθωμένη εντολή. Στο πρόγραμμα να μη γίνει καμία προσθήκη ή αφαίρεση εντολής.

```
/*1*/   #include<iostream>
/*2*/   #define N=10
/*3*/   using namespace std;
/*4*/   int main(){
/*5*/      int x, p;
/*6*/      for (i=0; i<N; i++){
/*7*/         cout << "Δώσε αριθμό " << i+1 << endl;
/*8*/         cin >> x;
/*9*/         if (x/2!=0)
/*10*/          p++;
/*11*/      }
/*12*/      cout << p;
/*13*/  return 0;
/*14*/  }
```

(Βαθμοί 2)

Άσκηση 5

Δίνεται το πιο κάτω πρόγραμμα στη γλώσσα προγραμματισμού C++, το οποίο υπολογίζει το άθροισμα των ακέραιων αριθμών που υπάρχουν μέσα στο αρχείο **askisi5IN.txt** και καταχωρεί το αποτέλεσμα στο αρχείο **askisi5OUT.txt**. Από το πρόγραμμα απουσιάζουν κάποιες εντολές.

```
#include<fstream>
using namespace std;
int main(){
     A
     int N, sum = 0;

     while( B ){

          fin >> N;
          Γ
     }
     fout << sum;
return 0;
}
```

(α) Να γράψετε στο τετράδιο απαντήσεών σας τις εντολές που πρέπει να τοποθετηθούν στη **θέση Α**, ώστε να υλοποιηθούν οι κατάλληλες ροές ανάγνωσης από και εγγραφής προς τα αρχεία.

(Βαθμοί 2)

(β) Να γράψετε στο τετράδιο απαντήσεών σας την έκφραση/συνθήκη που πρέπει να τοποθετηθεί στη **θέση Β,** ώστε να διαβάζονται δεδομένα μέχρι να εντοπιστεί το τέλος του αρχείου **askisi5IN.txt**.

(Βαθμοί 2)

(γ) Να γράψετε στο τετράδιο απαντήσεών σας την εντολή που πρέπει να τοποθετηθεί στη **θέση Γ** ώστε να υπολογίζεται το άθροισμα όλων των ακεραίων αριθμών που διαβάζονται από το αρχείο **askisi5IN.txt**.

(Βαθμός 1)

Άσκηση 6

Δίνεται πιο κάτω η κύρια συνάρτηση (main) ενός προγράμματος στη γλώσσα προγραμματισμού C++ :

```
int main(){
     int A, B, D;
     float C;
     cin >> A >> B;
     cout << triple(A) << endl;
     print(A, B);
     C = calculate(A, B, D);
     cout << C << endl;
     cout << D << endl;
return 0;
}
```

(α) Να γράψετε το πρότυπο της συνάρτησης **triple**, η οποία υπολογίζει και επιστρέφει το τριπλάσιο του ακέραιου αριθμού **Α**.

(Βαθμός 1)

(β) Να γράψετε το πρότυπο της συνάρτησης **print**, η οποία παρουσιάζει στην οθόνη όλους τους ακέραιους αριθμούς από τον αριθμό Α μέχρι και τον αριθμό Β, συμπεριλαμβανομένων, χωρισμένους με κενό.

(Βαθμός 1)

(γ) Να γράψετε τη συνάρτηση **calculate**, η οποία υπολογίζει και επιστρέφει στην κύρια συνάρτηση (main) τον μέσο όρο των αριθμών Α και Β και το άθροισμα των ψηφίων των μονάδων των αριθμών Α και Β.

(Βαθμοί 3)

ΤΕΛΟΣ Α' ΜΕΡΟΥΣ

ΜΕΡΟΣ Β'

Άσκηση 7

Δίνεται η πιο κάτω λογική συνάρτηση:

$$F(A, B, C) = AB'C + A'BC + ABC$$

(α) Να δημιουργήσετε τον πίνακα αληθείας της συνάρτησης.

(Βαθμοί 3)

(β) Να σχεδιάσετε τον χάρτη Karnaugh που αντιστοιχεί στη συνάρτηση.

(Βαθμοί 3)

(γ) Να απλοποιήσετε τη συνάρτηση με τη χρήση χάρτη Karnaugh και να σχεδιάσετε το λογικό κύκλωμα που θα προκύψει μετά την απλοποίηση.

(Βαθμοί 4)

Άσκηση 8

Να γράψετε πρόγραμμα στη γλώσσα προγραμματισμού C++ το οποίο:

(α) Να διαβάζει **10** ακέραιους αριθμούς και να τους αποθηκεύει στον **πίνακα Α**.

(Βαθμοί 3)

(β) Να βρίσκει και να εμφανίζει τον **μεγαλύτερο** και τον **μικρότερο** αριθμό του **πίνακα Α**.

(Βαθμοί 3)

(γ) Να δέχεται ένα ακέραιο αριθμό S (0<S<=10) και να εκτελεί S κυκλικές μετακινήσεις. Σε μία κυκλική μετακίνηση ο αριθμός που βρίσκεται στη θέση 0 θα μετακινηθεί στη θέση 1, ο αριθμός που βρίσκεται στη θέση 1 θα μετακινηθεί στη θέση 2 κλπ. Ο αριθμός που βρίσκεται στην τελευταία θέση του πίνακα θα μετακινηθεί στην θέση 0 (1η θέση) του πίνακα. Αφού εκτελεστούν οι κυκλικές μετακινήσεις να τυπωθεί ο πίνακας στην οθόνη.

(Βαθμοί 4)

Παράδειγμα Εισόδου (πληκτρολόγιο)	**Παράδειγμα Εξόδου (οθόνη)**
9 2 3 4 1 -7 1 1 6 5 3	Δώσε 10 ακέραιους αριθμούς Μεγαλύτερος=9 Μικρότερος=-7 Δώσε αριθμό κυκλικών μετακινήσεων Πίνακας Α 1 6 5 9 2 3 4 1 -7 1

Άσκηση 9

Να γράψετε πρόγραμμα στη γλώσσα προγραμματισμού C++, το οποίο να δέχεται δύο συμβολοσειρές **stA**, **stB** και έναν ακέραιο αριθμό N. Να θεωρήσετε ότι, οι συμβολοσειρές περιέχουν μόνο κεφαλαία γράμματα του λατινικού αλφαβήτου. Στη συνέχεια το πρόγραμμα πρέπει:

(α) Να εμφανίζει στην οθόνη τη συμβολοσειρά **stA** μέχρι τον N-οστό χαρακτήρα. Αν η συμβολοσειρά έχει μέγεθος μικρότερο ή ίσο με το N, τότε να την εμφανίζει ολόκληρη.

(Βαθμοί 3)

(β) Να εμφανίζει στην οθόνη το πλήθος των εμφανίσεων του πρώτου χαρακτήρα της συμβολοσειράς **stA** μέσα στη συμβολοσειρά **stB**.

(Βαθμοί 3)

(γ) Να εμφανίζει στην οθόνη τον χαρακτήρα με τις περισσότερες εμφανίσεις μέσα στη συμβολοσειρά **stB**. Σε περίπτωση που υπάρχουν περισσότεροι από ένας χαρακτήρες με μέγιστο πλήθος εμφανίσεων, να εμφανίζεται αυτός που προηγείται αλφαβητικά.

(Βαθμοί 4)

Παράδειγμα Εισόδου 1 (πληκτρολόγιο)	**Παράδειγμα Εξόδου 1 (οθόνη)**
KALIMERA ANNA 5	Δώσε συμβολοσειρές Α, Β και τον αριθμό Ν Εμφάνιση συμβολοσειράς Α μέχρι τον Ν-στο χαρακτήρα KALIM Πλήθος εμφανίσεων του 1ου χαρακτήρα της Α μέσα στη Β:0 Χαρακτήρας με τις περισσότερες εμφανίσεις μέσα στη Β:A
Παράδειγμα Εισόδου 2 (πληκτρολόγιο)	**Παράδειγμα Εξόδου 2 (οθόνη)**
VIOLIN VELVET 10	Δώσε συμβολοσειρές Α, Β και τον αριθμό Ν Εμφάνιση ολόκληρης της συμβολοσειράς Α VIOLIN Πλήθος εμφανίσεων του 1ου χαρακτήρα της Α μέσα στη Β:2 Χαρακτήρας με τις περισσότερες εμφανίσεις μέσα στη Β:E

Άσκηση 10

Να γράψετε πρόγραμμα στη γλώσσα προγραμματισμού C++, το οποίο:

(α) Να δέχεται 36 ακέραιους αριθμούς και να τους καταχωρεί σε τετραγωνικό πίνακα, διαστάσεων 6x6 με όνομα **arr2,** καθώς και 6 ακέραιους αριθμούς και να τους καταχωρεί σε έναν μονοδιάστατο πίνακα με όνομα **arr1**.

(Βαθμοί 3)

(β) Να δηλώνει και να χρησιμοποιεί τη συνάρτηση **Max2,** η οποία να δέχεται ως παράμετρο τον δισδιάστατο πίνακα **arr2** και να επιστρέφει στην κύρια συνάρτηση (main) την τιμή του μεγαλύτερου αριθμού που υπάρχει στον πίνακα. Το αποτέλεσμα να εμφανίζεται στην οθόνη από την κύρια συνάρτηση.

(Βαθμοί 3)

(γ) Να εμφανίζει στην οθόνη την τιμή **1** στην περίπτωση που τα στοιχεία του πίνακα **arr1** είναι τα ίδια με τα στοιχεία οποιασδήποτε στήλης του πίνακα **arr2**, διαφορετικά να εμφανίζει την τιμή **0**.

(Βαθμοί 4)

Παράδειγμα Εισόδου 1 (πληκτρολόγιο)	**Παράδειγμα Εξόδου 1 (οθόνη)**
10 21 3 44 10 5 2 4 11 33 1 390 12 4 16 53 1 30 33 4 11 37 1 90 32 14 51 38 1 3 21 54 1 5 1 39 1 2 3 4 51 65	Δώσε τα στοιχεία του δισδιάστατου πίνακα Δώσε τα στοιχεία του μονοδιάστατου πίνακα Μέγιστος αριθμός δισδιάστατου πίνακα:390 0
Παράδειγμα Εισόδου 2 (πληκτρολόγιο)	**Παράδειγμα Εξόδου 2 (οθόνη)**
10 21 3 44 10 5 -2 4 11 33 1 90 12 -4 16 53 99 30 33 4 11 37 1 90 32 14 51 38 1 3 21 54 1 5 1 39 21 4 -4 4 14 54	Δώσε τα στοιχεία του δισδιάστατου πίνακα Δώσε τα στοιχεία του μονοδιάστατου πίνακα Μέγιστός αριθμός δισδιάστατου πίνακα:99 1

ΤΕΛΟΣ Β' ΜΕΡΟΥΣ

ΜΕΡΟΣ Γ'

Άσκηση 11

Η Μαρία είναι υπεύθυνη μιας αλυσίδας καταστημάτων με **1000** καταστήματα σε όλο τον κόσμο. Στο τέλος κάθε έτους το κάθε κατάστημα στέλνει στη Μαρία τα καθαρά του κέρδη για κάθε μήνα.

Να γράψετε πρόγραμμα στη γλώσσα προγραμματισμού C++, το οποίο:

(α) Να καταχωρεί στον μονοδιάστατο πίνακα **shops** το όνομα του κάθε καταστήματος και στον παράλληλο δισδιάστατο πίνακα **profits** (1000 x 12) τα κέρδη κάθε καταστήματος για κάθε έναν από τους 12 μήνες του έτους.

(Βαθμοί 3)

(β) Να χρησιμοποιεί τη συνάρτηση **TotalProfits**, η οποία θα λαμβάνει ως παραμέτρους από την κύρια συνάρτηση (main) τους πίνακες **profits** και **totals**, για να υπολογίζει και να καταχωρεί στον πίνακα **totals** τα συνολικά ετήσια κέρδη για κάθε ένα από τα καταστήματα.

(Βαθμοί 3)

(γ) Να ταξινομεί τους πίνακες **shops** και **totals** σε αλφαβητική σειρά με βάση τα ονόματα των καταστημάτων του πίνακα shops. Η ταξινόμηση να γίνει με τη χρήση του αλγόριθμου ταξινόμησης φυσαλίδας (**bubble sort**). Ακολούθως, να εμφανίζει τους δύο ταξινομημένους πίνακες, με τα ετήσια κέρδη να εμφανίζονται με ακρίβεια 2 δεκαδικών ψηφίων.

(Βαθμοί 4)

(δ) Να δέχεται το όνομα ενός καταστήματος και να εντοπίζει και να εμφανίζει στην οθόνη τα συνολικά κέρδη του καταστήματος. Σε περίπτωση που δεν βρεθεί το κατάστημα, να εμφανίζεται ο αριθμός **-1**. Η αναζήτηση να γίνεται με τη χρήση του αλγορίθμου δυαδικής αναζήτησης **(binary search)**.

(Βαθμοί 5)

Το πρόγραμμα πρέπει να εμφανίζει στην οθόνη τα κατάλληλα μηνύματα για την εισαγωγή των δεδομένων και την εξαγωγή των αποτελεσμάτων.

Παράδειγμα Εισόδου (για 5 καταστήματα και 6 μήνες) (πληκτρολόγιο)

```
London 14050.20 33040.55 16060.25 10050.00 44150.20 53421.40
Paris 35555.40 43242.55 90022.55 144050.10 45650.20 43211.15
Berlin 4567.25 7543.75 3060.00 6730.10 7553.00 5362.20
Tokyo 23050.60 33377.85 46362.50 48120.10 45251.40 30005.00
Cairo 34120.40 43040.00 50190.25 43253.00 41150.75 54210.00
Paris
```

Παράδειγμα Εξόδου (οθόνη) (για 5 καταστήματα και 6 μήνες)

```
Δώσε όνομα καταστήματος και έσοδα μήνα
Καταστήματα - Κέρδη
Berlin 34816.30
Cairo 265964.41
London 170772.59
Paris 401731.94
Tokyo 226167.47
Δώσε όνομα καταστήματος προς αναζήτηση
Κέρδη:401731.94
```

Άσκηση 12

Ο Γιώργος είναι ο ταμίας του τμήματός του. Στην προσπάθειά του για καλύτερο έλεγχο των οικονομικών του τμήματος, σας ζητά να τον βοηθήσετε δημιουργώντας ένα πρόγραμμα στη γλώσσα προγραμματισμού C++, το οποίο:

(α) Να δημιουργεί μία εγγραφή με όνομα **student** η οποία να περιλαμβάνει τα πιο κάτω μέλη:

- Αριθμός Μητρώου (ακέραιος αριθμός)
- Εισφορά στο ταμείο (πραγματικός αριθμός)

Στη συνέχεια, να διαβάζει τα στοιχεία (τον αριθμό μητρώου και την αντίστοιχη αρχική εισφορά στο ταμείο) των 12 μαθητών της τάξης από το αρχείο **askisi12IN.txt** και να τα αποθηκεύει σε έναν πίνακα εγγραφών τύπου **student**, με όνομα **studentsC1**.

(Βαθμοί 3)

(β) Να διαβάζει από το αρχείο **eisfores.txt** τον αριθμό μητρώου και την αντίστοιχη επιπρόσθετη εισφορά του κάθε μαθητή. Υπάρχει περίπτωση κάποιοι μαθητές να μην έχουν συνεισφέρει επιπρόσθετη εισφορά στο ταμείο. Για κάθε μαθητή που έχει ήδη συνεισφέρει, η επιπρόσθετη εισφορά θα προστίθεται στην αρχική εισφορά. Να θεωρήσετε ότι στο αρχείο **eisfores.txt** δεν υπάρχουν καταχωρήσεις για άλλους μαθητές εκτός των 12 μαθητών της τάξης που ήδη καταχωρήθηκαν από το αρχείο **askisi12IN.txt**

(Βαθμοί 4)

(γ) Να χρησιμοποιεί τη συνάρτηση **total**, η οποία να δέχεται ως παραμέτρους τον πίνακα εγγραφών **studentsC1** και το μέγεθός του και να επιστρέφει τη συνολική εισφορά της τάξης. Το αποτέλεσμα να εμφανίζεται στην οθόνη με ακρίβεια 2 δεκαδικών ψηφίων.

(Βαθμοί 3)

(δ) Να ταξινομεί τα στοιχεία του πίνακα εγγραφών **studentsC1** σε αύξουσα σειρά ως προς την εισφορά, χρησιμοποιώντας τον αλγόριθμο εισαγωγής (**insertion sort**) και να καταχωρεί τον πίνακα στο αρχείο **askisi12OUT.txt**. Οι εισφορές να καταχωρούνται στο πιο πάνω αρχείο με **ακρίβεια 2 δεκαδικών** ψηφίων.

(Βαθμοί 5)

Το πρόγραμμα πρέπει να εμφανίζει στην οθόνη τα κατάλληλα μηνύματα για την εισαγωγή των δεδομένων και την εξαγωγή των αποτελεσμάτων.

Παράδειγμα Εισόδου **(από αρχείο askisi12IN.txt)**	**Παράδειγμα Εξόδου**
1001 10.50 1010 12.00 1005 10.20 1007 8.20 1011 11.50 1210 12.20 1023 5.30 1027 13.30 1301 14.50 1019 12.70 1205 4.20 1034 14.20 **(από αρχείο eisfores.txt)** 1205 2.00 1023 2.50 1007 1.20	Συνολική εισφορά:134.50 **(στο αρχείο askisi12OUT.txt)** 1205 6.20 1023 7.80 1007 9.40 1005 10.20 1001 10.50 1011 11.50 1010 12.00 1210 12.20 1019 12.70 1027 13.30 1034 14.20 1301 14.50

ΤΕΛΟΣ ΕΞΕΤΑΣΗΣ

ΤΥΠΟΛΟΓΙΟ ΣΥΝΑΡΤΗΣΕΩΝ ΣΤΗ ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ C++

ΣΥΝΑΡΤΗΣΕΙΣ ΤΗΣ ΒΙΒΛΙΟΘΗΚΗΣ <cmath>		
Συνάρτηση	**Χρήση**	**Παράμετροι**
sqrt(x)	Επιστρέφει την **τετραγωνική ρίζα** του αριθμού x. Η επιστρεφόμενη τιμή είναι πραγματικός αριθμός.	Ένας θετικός αριθμός (ακέραιος ή πραγματικός)
abs(x)	Επιστρέφει την **απόλυτη τιμή** του αριθμού x. Η επιστρεφόμενη τιμή εξαρτάται από τον τύπο του αριθμού x.	Ένας αριθμός (ακέραιος ή πραγματικός)
pow(x,y)	Επιστρέφει το **αποτέλεσμα** της **δύναμης x^y**. Η επιστρεφόμενη τιμή είναι πραγματικός αριθμός.	Δύο πραγματικοί αριθμοί
trunc(x)	Επιστρέφει το **ακέραιο μέρος** του αριθμού x σε πραγματική μορφή, **αγνοώντας το δεκαδικό μέρος του**.	Ένας πραγματικός αριθμός
round(x)	Επιστρέφει το **ακέραιο μέρος** του αριθμού x σε πραγματική μορφή, **στρογγυλοποιημένο** στην **πλησιέστερη τιμή.**	Ένας πραγματικός αριθμός
ΣΥΝΑΡΤΗΣΕΙΣ ΤΗΣ ΒΙΒΛΙΟΘΗΚΗΣ <string>		
size()	**Επιστρέφει** το **μέγεθος** μιας συμβολοσειράς. Η επιστρεφόμενη τιμή είναι **ακέραιος αριθμός** που συμβολίζει από πόσα bytes αποτελείται μια συμβολοσειρά.	Καμία παράμετρος
clear()	**Διαγράφει** το **περιεχόμενο** μιας συμβολοσειράς. Δεν επιστρέφει τίποτα.	Καμία παράμετρος
empty()	**Ελέγχει** αν μια συμβολοσειρά είναι **άδεια**. Η επιστρεφόμενη τιμή είναι τύπου **Boolean**.	Καμία παράμετρος
getline(x,y)	**Αποθηκεύει ολόκληρη** μια συμβολοσειρά που μπορεί να εισαχθεί από το πληκτρολόγιο ή από αρχείο (x) στο αντικείμενο y.	Η 1η παράμετρος (x) αφορά στη μέθοδο εισαγωγής της συμβολοσειράς (π.χ. από το πληκτρολόγιο ή από αρχείο) και η 2η παράμετρος (y) αφορά στο αντικείμενο στο οποίο θα αποθηκευτεί η συμβολοσειρά, η οποία έχει διαβαστεί αρχικά.
ΣΤΑΘΕΡΕΣ ΤΗΣ ΒΙΒΛΙΟΘΗΚΗΣ <climits>		
INT_MAX	**Μέγιστο αριθμητικό** όριο μεταβλητής ή σταθεράς τύπου **integer.** Η ακριβής αριθμητική τιμή της είναι **32767 (**στα **2 bytes)** ή **2147483647 (**στα **4 bytes)**	
INT_MIN	**Ελάχιστο αριθμητικό** όριο μεταβλητής ή σταθεράς τύπου **integer.** Η ακριβής αριθμητική τιμή της είναι **-32767 (**στα **2 bytes)** ή **-2147483647 (**στα **4 bytes)**	

ΣΗΜΕΙΩΣΕΙΣ

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

ΣΗΜΕΙΩΣΕΙΣ

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

ΣΗΜΕΙΩΣΕΙΣ

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

ΣΗΜΕΙΩΣΕΙΣ

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

ΣΗΜΕΙΩΣΕΙΣ

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

ΣΗΜΕΙΩΣΕΙΣ

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....