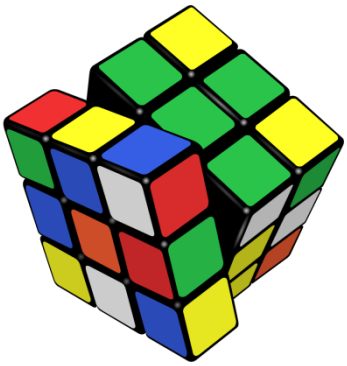
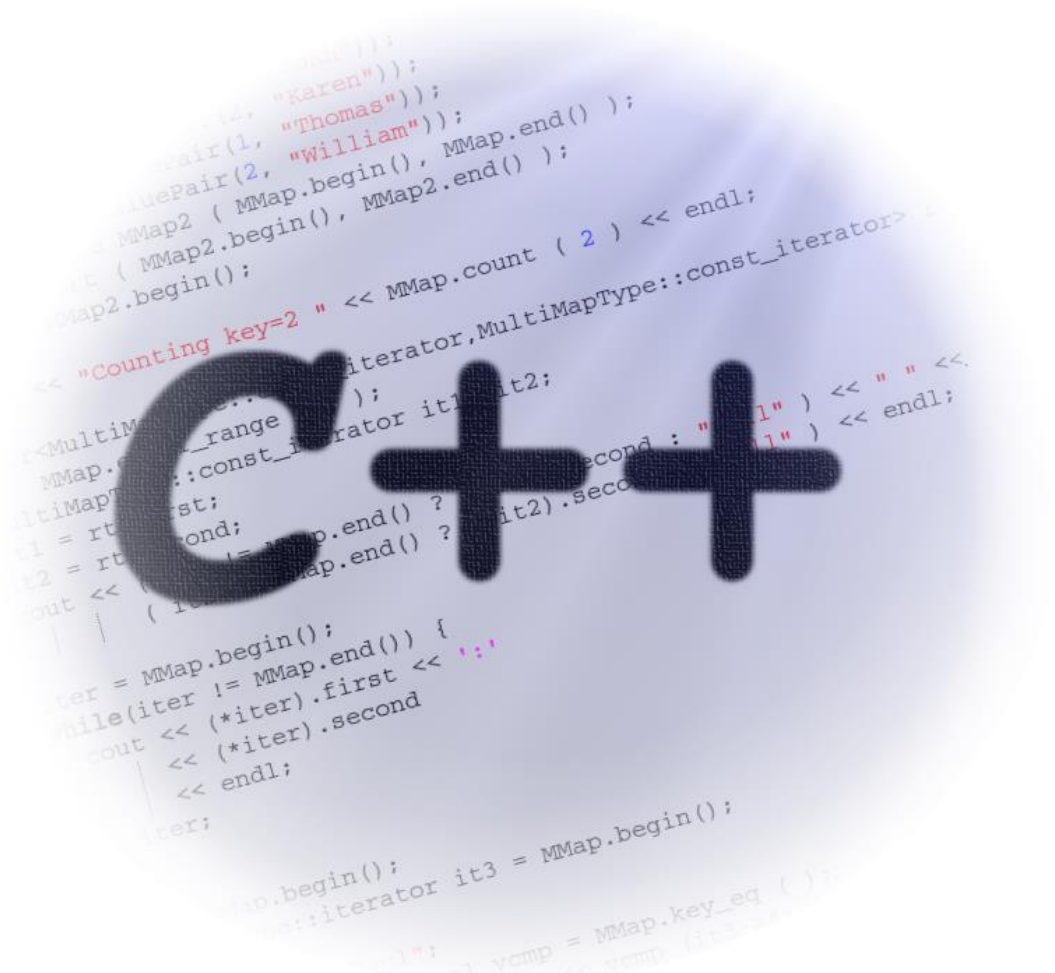
Ζάννειο Πειραματικό Γυμνάσιο

Όμιλοι Αριστείας και Δημιουργικότητας

ΟΜΙΛΟΣ ΑΛΓΟΡΙΘΜΙΚΗΣ



Προγραμματίζοντας σε C++ *(χωρίς το ++)*



Ευριπίδης Βραχνός

http://evripides.mysch.gr/

2018 – 2019

Λίγα λόγια για τις γλώσσες προγραμματισμού

Οι γλώσσες προγραμματισμού χωρίζονται σε τρεις κατηγορίες :

- Γλώσσες Μηχανής
- Γλώσσες χαμηλού επιπέδου
- Γλώσσες υψηλού επιπέδου

Κάθε υπολογιστής μπορεί να δεχθεί μόνο μια συγκεκριμένη γλώσσα μηχανής η οποία είναι σχεδιασμένη με βάση την αρχιτεκτονική του κάθε υπολογιστή. Οι εντολές μιας γλώσσας μηχανής είναι ακολουθίες από 1 και 0 π.χ. 1000001100100001 οι οποίες όπως είναι φανερό δεν είναι εύκολα κατανοητές από τον άνθρωπο.

Έτσι δημιουργήθηκαν οι γλώσσες χαμηλού επιπέδου (συμβολικές γλώσσες) με εντολές που προκύπτουν από συντομογραφίες της αγγλικής γλώσσας όπως για παράδειγμα LOAD A ή ADD A, B . Επειδή αυτές οι γλώσσες δεν ήταν άμεσα κατανοητές από τους υπολογιστές δημιουργήθηκαν οι γνωστοί συμβολομεταφραστές (assemblers) οι οποίοι μετασχηματίζουν τη συμβολική γλώσσα στη γλώσσα μηχανής του κάθε επεξεργαστή.

Όσο αυξανόταν ο όγκος των προγραμμάτων σε συμβολική γλώσσα (Assembly) ήταν φανερή η δυσκολία ελέγχου, επέκτασης και συντήρησής τους. Αυτό είχε ως αποτέλεσμα την ανάπτυξη μιας νέας κατηγορίας γλωσσών, των γλωσσών υψηλού επιπέδου. Οι γλώσσες αυτές δίνουν τη δυνατότητα στον προγραμματιστή, αφού σκεφτεί και μοντελοποιήσει ένα πρόβλημα να το υλοποιήσει άμεσα. Ο έλεγχος και η συντήρηση προγραμμάτων με χιλιάδες γραμμές κώδικα είναι πολύ πιο εύκολος με τις γλώσσες υψηλού επιπέδου. Το πρόγραμμα που μεταφράζει μια γλώσσα υψηλού επιπέδου σε γλώσσα μηχανής λέγεται μεταγλωττιστής (compiler).

Απλά για την … ιστορία

Η γλώσσα προγραμματισμού C είναι μια γενικού σκοπού γλώσσα η οποία αναπτύχθηκε για την υλοποίηση του λειτουργικού συστήματος UNIX. Παρόλο που η C είναι μια γλώσσα υψηλού επιπέδου φτάνει σε ταχύτητα τις συμβολικές γλώσσες (assembly languages) και για αυτόν τον λόγο αναφέρεται σαν μέσου επιπέδου γλώσσα προγραμματισμού. Πολλές από τις ιδέες της C προήλθαν από τη γλώσσα BCPL που αναπτύχθηκε από τον Martin Richards. Η επιρροή αυτή της BCPL πάνω στη C προέρχεται έμμεσα από τη γλώσσα B από τον Ken Thomson το 1970 στα Εργαστήρια Bell για το πρώτο σύστημα UNIX σε έναν DEC PDP-7. Οι B και BCPL είναι γλώσσες δίχως τύπους σε αντίθεση με τη C.

Το 1972 ο Dennis Ritchie στα Εργαστήρια Bell αναπτύσσει τη C το 1978 και στη συνέχεια εκδίδει το βιβλίο The C Programming Language μαζί με τον Brian Kernighan. Αυτό το βιβλίο προκάλεσε μια επανάσταση στον κόσμο της πληροφορικής και ακόμα και σήμερα χρησιμοποιείται ως το βασικό σύγγραμα για το άντιστοιχο μάθημα από τα περισσότερα πανεπιστήμια στον κόσμο

Το 1983 το American National Standards Institute(ANSI) ίδρυσε μια επιτροπή η οποία παρείχε έναν τυπικό ορισμό της γλώσσας C που είναι γνωστός ως το ANSI πρότυπο της γλώσσας.

Η C++ είναι μια επέκταση της C, η οποία αναπτύχθηκε από τον Bjarne Stroustrup στις αρχές της δεκαετίας του 1980 στα εργαστήρια Bell. Η γλώσσα προγραμματισμού C++ προσθέτει έναν αριθμό από δομικά στοιχεία στη C τα οποία επιτρέπτουν την χρήση αντικειμενοστρεφούς προγραμματισμού. Για αυτό και η C++ λέγεται ότι είναι μια αντικειμενοστρεφής γλώσσα σε αντίθεση με τις PASCAL και C που είναι διαδικαστικές γλώσσες.

Σύνταξη, Μεταγλώττιση, Σύνδεση, Εκτέλεση

Ένα πρόγραμμα περνάει από διάφορα στάδια μέχρι να εκτελεστεί. Αυτά είναι τα παρακάτω :

- **Σύνταξη** του προγράμματος. Το πρόγραμμα συντάσσεται σε κάποιον γνωστό editor όπως το notepad++, ο editor της Visual C++, το CodeBlocks και αποθηκεύεται στο δίσκο .
- **Προεπεξεργασία**. Ο προεπεξεργαστής (preprocessor) επεξεργάζεται το πρόγραμμα, ειδικά τις δηλώσεις που ξεκινάνε με #.
- **Μεταγλώττιση**. Ο μεταγλωττιστής (compiler) παράγει τον αντικείμενο κώδικα (object code) και τον αποθηκεύει στο δίσκο.
- **Σύνδεση**. Το πρόγραμμα σύνδεσης (linker) εκτελεί τις απαραίτητες συνδέσεις του αρχείου αντικείμενου κώδικα με τις απαραίτητες βιβλιοθήκες που χρησιμοποιούνται. Έτσι προκύπτει το εκτελέσιμο αρχείο το οποίο και αποθηκεύεται στο δίσκο.
- **Εκτέλεση.** Το εκτελέσιμο αρχείο έρχεται στη μνήμη όπου και εκτελείται.

Ένα άλλο στάδιο κατά τη διαδικασία ανάπτυξης μιας εφαρμογής είναι αυτό της εκσφαλμάτωσης (debugging) σε περίπτωση που το πρόγραμμα έχει λάθη. Τα λάθη αυτά χωρίζονται σε 3 κατηγορίες :

- Λάθη κατά την μεταγλώττιση, τα οποία είναι κατά συντριπτική πλειοψηφία συντακτικά λάθη
- Λάθη κατά τη σύνδεση
- Λάθη κατά την εκτέλεση

Οι δυο τελευταίες κατηγορίες λαθών είναι γνωστές και ως λογικά λάθη και είναι τα πιο δύσκολα να διορθωθούν.

Βασικές Έννοιες

Θα εξετάσουμε την δομή ενός απλού προγράμματος C++ :

```
#include <iostream>      // Το πρώτο μου πρόγραμμα σε C++

using namespace std;

int main() {

    cout << "Houston we have a problem";

    return 0;   // δηλώνει ότι το πρόγραμμα τέλειωσε με επιτυχία
}
```

Η δήλωση

// Το πρώτο μου πρόγραμμα σε C++

είναι ένα σχόλιο στη C++, δηλαδή αγνοείται από τον μεταγλωττιστή και έχει ρόλο επεξηγηματικό. Απευθύνεται σε αυτόν που διαβάζει το πρόγραμμα. Το σύμβολο '//' σημαίνει ότι η υπόλοιπη γραμμή είναι σχόλιο και αγνοείται κατά τη μεταγλώττιση. Αν θέλουμε σχόλια σε παραπάνω από μια συνεχείς γραμμές τότε χρησιμοποιούμε τα '/*', '*/'.

Η δήλωση

#include <iostream>

ενημερώνει τον μεταγλωττιστή (για την ακρίβεια τον προεπεξεργαστή) ότι χρησιμοποιείται η βιβλιοθήκη iostream η οποία περιέχει συναρτήσεις σχετικά με την είσοδο/έξοδο. Αντικαθιστά τη βιβλιοθήκη stdio της C.

Η γραμμή

using namespace std;

δεν θα μας απασχολήσει φέτος , και δεν έχει νόημα να εξηγηθεί. Θεωρήστε ότι και αυτή δηλώνει ότι θα χρησιμοποιηθεί η "συλλογή βιβλιοθηκών" std.

Η γραμμή

int main()

Είναι μέρος της δομής κάθε C++ προγράμματος και περιέχει εντός των '{', '}' το κύριο πρόγραμμα της εφαρμογής. Τελειώνει με την εντολή

return 0;

διότι όπως θα δούμε και παρακάτω επειδή η main είναι συνάρτηση πρέπει να επιστρέφει κάτι και είναι έτσι ορισμένη ώστε να επιστρέφει έναν ακέραιο.
Η εντολή

cout << "Houston we have a problem" << endl;

Εμφανίζει στην οθόνη το σχετικό μήνυμα. Η γενική μορφή αυτής της εντολής είναι η εξής :

cout << Μήνυμα << ……..<< endl;

Το Μήνυμα μπορεί να είναι αριθμός ή κείμενο. Δηλαδή το αντικείμενο cout καταλαβαίνει μόνο του τον τύπο της προς εμφάνιση μεταβλητής,
Τέλος το endl στο τέλος προκαλεί την αλλαγή γραμμής. Όπως ο διακόπτης "\n" της C. Αυτό σημαίνει ότι οι παρακάτω εντολές εμφανίζουν το ίδιο αποτέλεσμα στην οθόνη :

```
cout << "Το πρώτο μου πρόγραμμα σε C++" << endl;
cout << "Το πρώτο μου πρόγραμμα σε C++\n" ;
```

Φανταστείτε την εντολή cout << μήνυμα σαν να δηλώνει ότι το μήνυμα κατευθύνεται στην έξοδο (οθόνη), βλέποντας τον τελεστή << σαν ένα βελάκι ←

Σημειώστε ότι κάθε εντολή στη C++ τελειώνει με ένα ερωτηματικό '**;**'.

Μεταβλητές , Δηλώσεις και Σταθερές

Μια μεταβλητή ορίζεται από τρία πράγματα. Το όνομά της, τον τύπο της και το περιεχόμενο της σε κάποια διεύθυνση στη μνήμη. Έστω το παρακάτω πρόγραμμα :

```
// Πρόσθεση δύο ακεραίων
#include <iostream>
using namespace std;

int main()
{
    int integer1, integer2, sum; // δήλωση τριών ακέραιων μεταβλητών

    cout << "Δώσε τον πρώτο ακέραιο" ;
    cin >> integer1;             // διαβάζει τον πρώτο ακέραιο
    cout << "Δώσε τον δεύτερο ακέραιο" ;
    cin >> integer2;            // διαβάζει τον δεύτερο ακέραιο

    sum = integer1 + integer2;   // Προσθέτει τους δυο αριθμούς και
                                 // αποθηκεύει το αποτέλεσμα στην sum

    cout << "Το άθροισμα είναι = " << sum << endl;
                                // Εμφανίζει το αποτέλεσμα

       return 0;   // δηλώνει ότι το πρόγραμμα τέλειωσε με επιτυχία.
}
```

Στο παραπάνω πρόγραμμα ορίζονται τρεις ακέραιες μεταβλητές. Οι μεταβλητές αυτές αρχικά δεν έχουν κάποια τιμή. Ο χρήστης τους δίνει μια τιμή από το πληκτρολόγιο και στη συνέχεια αποθηκεύεται στη μεταβλητή sum η τιμή του αθροίσματος.

Ο μεταγλωττιστής πρέπει να ξέρει τον τύπο κάθε μεταβλητής για να μπορεί να ελέγξει ποιες πράξεις είναι επιτρεπτές και ποιες όχι και το όνομά της για να μπορεί στη συνέχεια να προσπελάσει το περιεχόμενο στη μνήμη αυτής της μεταβλητής.

Οι βασικοί τύποι που ορίζονται στη C++ είναι ακέραιοι, αριθμοί κινητής υποδιαστολής (πραγματικοί) και χαρακτήρες και δίνονται στον παρακάτω πίνακα:

Τύπος	Μέγεθος (bytes)
int	4
char	1
float	4
double	8

Μερικές φορές θέλουμε να χρησιμοποιήσουμε κάποια συγκεκριμένη τιμή όπως π.χ. το π = 3.14159. Σε αυτή την περίπτωση υπάρχουν οι σταθερές οι οποίες έχουν ακριβώς τις ίδιες ιδιότητες με τις μεταβλητές, δηλαδή τύπο, όνομα και περιεχόμενο αλλά από τη στιγμή που θα αρχικοποιηθούν δεν μπορούν να αλλάξουν τιμή. Έτσι η δήλωση σε αυτή την περίπτωση είναι η εξής :

```
const int pi = 3.14159;
```

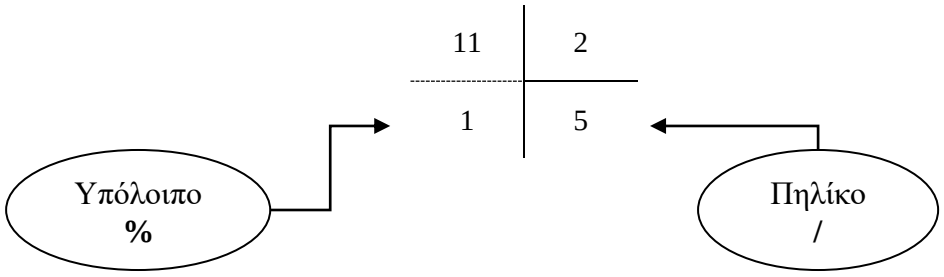
Σημειώστε ότι μια σταθερά πρέπει να αρχικοποιηθεί στη δήλωσή της υποχρεωτικά.

Αριθμητικοί και Λογικοί Τελεστές

Μεταξύ των μεταβλητών που ορίσαμε παραπάνω μπορούμε να εφαρμόσουμε κάποιες πράξεις οι οποίες είναι :

Πράξη	Τελεστής	Αλγεβρική Έκφραση	Έκφραση C++
Πρόσθεση	+	A + B	A + B
Αφαίρεση	—	A – B	A – B
Πολλαπλασιασμός	*	AB	A * B
Διαίρεση	/	A / B	A / B
Υπόλοιπο	%	A % B	A % B

Μέσα σε σύνθετες αριθμητικές παραστάσεις που αποτελούνται από μεταβλητές και τις παραπάνω πράξεις υπάρχει μια προτεραιότητα στους τελεστές των πράξεων αυτών. Έτσι την μεγαλύτερη προτεραιότητα έχουν οι *, /, % και χαμηλότερη οι +,-.



Όλα προκύπτουν από την ταυτότητα της διαίρεσης του Ευκλείδη

$$\Delta = \delta \cdot \pi + \upsilon$$

με $\pi = \Delta$ / δ και $\upsilon = \Delta$ **%** δ , όπου Δ, δ, π, υ ακέραιοι αριθμοί

Λογικοί τελεστές:
Οι λογικοί τελεστές εφαρμόζονται πάνω σε λογικές εκφράσεις (δηλαδή εκφράσεις που το αποτέλεσμά τους είναι **true** ή **false**). Το αποτέλεσμά τους είναι πάλι λογικού τύπου δεδομένων.

Στον παρακάτω πίνακα θεωρούμε ότι τα A και B είναι λογικές εκφράσεις.

A	B	A and B	A or B	not A
False	False	False	False	True
False	True	False	True	True
True	False	False	True	False
True	True	true	True	False

Π.χ. οι παρακάτω εκφράσεις για α = 2 και β = 10 είναι

- **not** (α > 2) **and** (β > 9) **true**
- α < 2 **or** β == 10 **true**
- **not** (α < 2 **and** β == 10) **false**

Σημαντική σημείωση: Επειδή στην C οι παραπάνω τελεστές εκφράζονταν με άλλα σύμβολα (το not με το θαυμαστικό ! το and με το && και το or με το || οι περισσότεροι προγραμματιστές τα χρησιμοποιούν και στην C++). Εσείς μπορείτε να χρησιμοποιήσετε ότι θέλετε.

Η Μεταβλητή

Η μεταβλητή στον προγραμματισμό είναι πολύ διαφορετική έννοια από τη μεταβλητή που ξέρετε στα μαθηματικά. Μια μεταβλητή έχει :

- **Όνομα** με το οποίο αναφερόμαστε σε αυτήν μέσα στο πρόγραμμα π.χ. a, b1, sum
- **Περιεχόμενο** που είναι η τιμή της μεταβλητής
- **Θέση στη μνήμη** όπου αποθηκεύεται το περιεχόμενο

Π.χ. έστω οι ακέραιες μεταβλητές **A**, **B**, έχουμε τις παρακάτω εντολές

A = 2 *η μεταβλητή A έχει την τιμή 2*
B = 10 *η μεταβλητή B έχει την τιμή 10*
A = B *η τιμή της μεταβλητής B αντιγράφεται στην A, το 2 σβήνεται και το 10 παίρνει τη θέση του*
A = A + 25 *η μεταβλητή A γίνεται 10 + 25 = 35.*

Η Εντολή Ανάθεσης τιμής (εκχώρηση) =

Η εντολή ανάθεσης τιμής μπορεί να μεταφραστεί ως εξής: «*Υπολόγισε την τιμή της έκφρασης δεξιά από το σύμβολο ανάθεσης τιμής και βάλε το αποτέλεσμα στη μεταβλητή που βρίσκεται αριστερά από την ανάθεση τιμής*».

Αντιμετάθεση Μεταβλητών	Τι εμφανίζει στην οθόνη
#include <iostream.h> **using namespace** std; **int main() {** **cout** << "Enter two numbers" << endl; **cin** >> A >> B ; C = A A = B B = C **cout** << A << " " << B << endl; **return 0;** **}**	Enter two numbers 10 167 167 10

Εντολές Εισόδου/Εξόδου cin / cout

Η εντολή εξόδου cout

Η εντολή cout εμφανίζει στην οθόνη τις τιμές μιας λίστας εκφράσεων, χωρισμένων από κόμματα. Όταν θέλουμε να εμφανιστεί ένα μήνυμα αυτούσιο χρησιμοποιούμε τα " " , αλλιώς ο διερμηνευτής θα προσπαθήσει να το υπολογίσει. Δείτε το παρακάτω παράδειγμα

Εντολή cout	Τι εμφανίζει στην οθόνη
a ← 2*2*2 **cout** << "a ", a << endl; **cout** << "1 + 2 + 3" << endl; **cout** << 1 + 2 + 3 << endl; **cout** << "4 + α = " ; *// δεν αλλάζει γραμμή* **cout** << 4 + α << endl;	a 8 1 + 2 + 3 6 4 + α = 20

Παρατηρήσεις:

- Όταν θέλουμε να εμφανίσουμε πολλά μηνύματα με μια εντολή cout, τα χωρίζουμε με << .
- Οι εκφράσεις μπορεί να είναι οποιουδήποτε τύπου δεδομένων. Η cout καταλαβαίνει από μόνη της πως θα εμφανίζει κάθε τύπο.
- Γράφουμε cout << endl για να κατέβουμε στην επόμενη γραμμή. Το endl σημαίνει end Line . π.χ. η εντολή cout << endl << endl << endl; δημιουργεί τρεις κενές γραμμές.

Η εντολή εισόδου cin

Η εντολή cin διαβάζει από το πληκτρολόγιο μία λίστα *μεταβλητών*, χωρισμένων από κόμματα. Οι μεταβλητές αυτές μπορεί να είναι οποιουδήποτε τύπου δεδομένων. Στη συνέχεια ο χρήστης πατάει το πλήκτρο <Enter> ώστε να μεταφερθούν τα δεδομένα στις αντίστοιχες μεταβλητές.

Οι παρακάτω ομάδες εντολών είναι ισοδύναμες, δηλαδή κάνουν ακριβώς το ίδιο πράγμα

cin >> a >> b >> c >> d ;	**cin** >> a ; **cin** >> b ; **cin** >> c ; **cin** >> d ;

Προσέξτε, τώρα ότι το >> αλλάζει φορά στο cin!!!
Έχουμε cout << a και cin >> a με το σκεπτικό ότι στην πρώτη περίπτωση η τιμή της μεταβλητής μεταφέρεται από την μνήμη στην έξοδο (οθόνη) ενώ στη δεύτερη η τιμή της a εισάγεται από την είσοδο (πληκτρολόγιο) στη μνήμη.

Εντολή Εξόδου ***cout***	Εντολή Εισόδου ***cin***
Με την εντολή αυτή μπορούμε να εμφανίσουμε στην οθόνη μηνύματα στον χρήστη, ή αποτελέσματα όπως τις τιμές μεταβλητών ή αριθμητικών εκφράσεων	*Με την εντολή αυτή ζητάμε από τον χρήστη να μας δώσει δεδομένα από το πληκτρολόγιο, τα οποία αποθηκεύονται σε μεταβλητές στη μνήμη*
Με την εντολή **cout << "A"** εμφανίζεται στην οθόνη το γράμμα Α ενώ με την εντολή **cout << A** εμφανίζεται στην οθόνη το περιεχόμενο (τιμή) της μεταβλητής Α. (με το **endl** αλλάζουμε γραμμή)	Με την εντολή **cin >> number** η εκτέλεση του προγράμματος παγώνει και περιμένει από τον χρήστη να δώσει έναν αριθμό. Αν ο χρήστης δώσει π.χ. το 28 τότε η μεταβλητή number θα πάρει την τιμή 28

Πρόγραμμα σε C++	**Οθόνη - Έξοδος**	**Εξήγηση**
#include <iostream> using namespace std; int main() {		Χρήση βιβλιοθηκών Ξεκινάει το κύριο πρόγραμμα. Ότι είναι μέσα στη main εκτελείται
int N, S, sqr, code; char letter;		Δήλωση ότι θα χρησιμοποιηθούν οι ακέραιες μεταβλητές N, S, k
cout << "**********" << endl;	**********	Εμφανίζει αστεράκια
cout << "Enter N = " ;	Enter N =	Εμφανίζει μήνυμα στον χρήστη
cin >> N ;	**3**	Ο χρήστης δίνει τον αριθμό 3 και αυτός αποθηκεύεται στην μεταβλητή N.
S = 2*N ;		Η μεταβλητή S γίνεται 2*N = 2*3 = 6
N = N + 1;		Η μεταβλητή N αυξάνεται κατά 1 (N=4)
N++;		Η μεταβλητή N αυξάνεται κατά 1 (N=5)
sqr = N*N;		Η sqr γίνεται sqr = 5*5 = 25
cout << "sqr = " << sqr	sqr = 25	
letter = 'A';		Στη μεταβλητή letter αποθηκεύεται ο χαρακτήρας (γράμμα) Α
code = (int) letter;		Στην code αποθηκεύεται ο κωδικός ASCII του γράμματος Α αφού πρώτα μετατρέψουμε τον χαρακτήρα σε ακέραιο. Αυτό γίνεται με το (int)
cout << letter << ", " << code;	A, 65	Εμφανίζει τα περιεχόμενα των μεταβλητών letter και code

if (N > 0)
cout << "positive" << endl;
else
cout << "negative" << endl;
	positive	Ελέγχει αν ο αριθμός N είναι θετικός ή αρνητικός και εμφανίζει κατάλληλο μήνυμα

}		Τέλος προγράμματος

Γενικές Παρατηρήσεις:

- Όλες οι εντολές τελειώνουν με ελληνικό ερωτηματικό ;
- Οι επιτρεπτές αριθμητικές πράξεις είναι (+)πρόσθεση, (–)αφαίρεση, (*)πολλαπλασιασμός, (/)διαίρεση, (%)υπόλοιπο ευκλείδειας διαίρεσης
- Στο αριστερό μέρος μια εντολής εκχώρησης (=) υπάρχει μόνο μια μεταβλητή

Η ΔΟΜΗ ΕΠΙΛΟΓΗΣ

Η εντολή if

Η εντολή *if* χρησιμοποιείται όταν χρειάζεται διακλάδωση της ροής του προγράμματος, ανάλογα με κάποια συνθήκη.

```
cout << "enter grade :" ;
cin >> grade;
if  ( grade < 10  ) {
        cout <<  " Fail " << endl;
}
    else {
        cout  << "Pass " << endl;
}
```

Όταν έχουμε πολλές περιπτώσεις εκτελούμε διαδοχικούς ελέγχους:

```
    int   grade;
    cout  <<  " Enter grade: ";
    cin >>  grade;
    if  ( grade < 0  ) {
        cout <<  " Error : negative grade " << endl;
    }
    else if  ( grade < 10  ) {
        cout   <<    "Fail " << endl;
    }
  else if  ( grade < 20  ) {
      cout   <<    "Pass " << endl;
  else  {
      cout   <<    "Error: grade is greater than 20 " << endl;
  }
```

Παρατήρηση:

Καλό να βάζετε πάντα τα άγκιστρα για να υποδηλώσετε το τέλος και την αρχή του αντίστοιχου block. Ωστόσο όταν έχουμε μόνο μια εντολή δεν είναι απαραίτητα.

Αλγόριθμος Β1	Αλγόριθμος Β2

if (a > 100)
  cout << "Είναι εκατοντάδες";
else if (a > 10)
  cout "Είναι δεκάδες";
else
  cout "Είναι μονάδες";
	if (a > 100)
	  cout << "Είναι εκατοντάδες";
	if (a > 10)
	  cout << "Είναι δεκάδες";
	if (a > 0)
	  cout << "Είναι μονάδες";

Μπορείτε να σκεφτείτε έναν αριθμό a για τον οποίο οι παραπάνω αλγόριθμοι εμφανίζουν διαφορετικά αποτελέσματα; Γιατί συμβαίνει αυτό;

Τι θα πρέπει να αλλάξουμε στον αλγόριθμο Β2 έτσι ώστε να παρουσιάζει τα ίδια ακριβώς αποτελέσματα με τον Β1 για τα ίδια δεδομένα εισόδου;

Παρακάτω δίνεται ο Αλγόριθμος Β2 με τις κατάλληλες αλλαγές έτσι ώστε να είναι ισοδύναμος με τον Β1.

```
if  ( a > 100 )       cout <<  "Είναι εκατοντάδες" ;
if  ( a > 10 and a <= 100 )     cout <<  "Είναι δεκάδες" ;
if  ( a > 0  and  a <= 10 )       cout <<  "Είναι μονάδες" ;
```

όπως βλέπετε όταν έχουμε μόνο μια εντολή στον κλάδο της if μπορούμε να τα χωρέσουμε όλα σε μια γραμμή.

Επίσης μπορούμε να έχουμε σε μια γραμμή πολλές εντολές αφού κάθε εντολή τελειώνει με **;**

π.χ.

```
k = k + 4; i++; cout << " Enter answer "; cin >> answer; cout << " This is the end";
```

ΔΟΜΕΣ ΕΠΑΝΑΛΗΨΗΣ

Η εντολή for

Η εντολή ***for*** είναι η πιο απλή/εύχρηστη/δημοφιλής εντολή επανάληψης. Χρησιμοποιεί ένα μετρητή για να μετράει πόσες επαναλήψεις γίνονται. Επιλέγουμε να τη χρησιμοποιήσουμε *όταν γνωρίζουμε εκ των προτέρων πόσες επαναλήψεις θα εκτελεστούν*. (Ωστόσο μπορεί να χρησιμοποιηθεί σε κάθε περίπτωση. ☺) Παρακάτω δίνονται τρία παραδείγματα προγραμμάτων που χρησιμοποιούν τη δομή επανάληψης

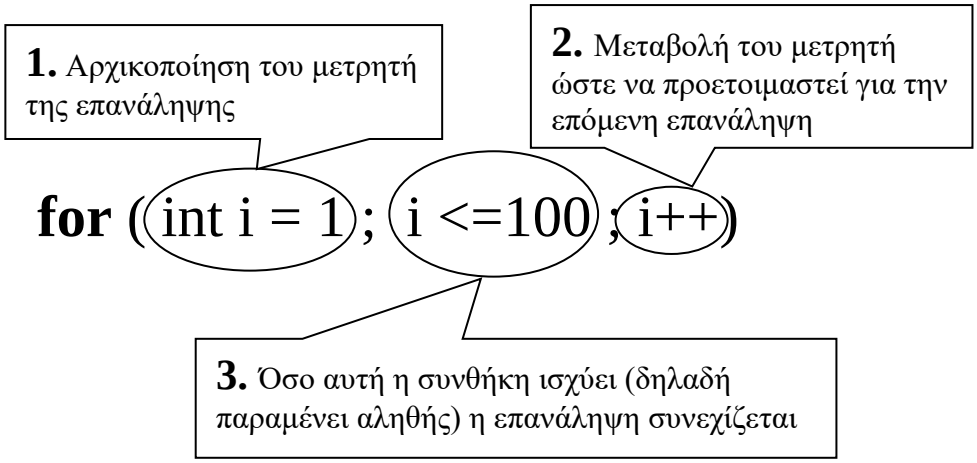
Το πρώτο εμφανίζει έναν αριθμό από αστεράκια '*' στην ίδια γραμμή ενώ το δεύτερο εμφανίζει ένα τετράγωνο από αστεράκια. Το πόσα θα εμφανιστούν καθορίζεται από τον χρήστη.

Τέλος το τρίτο παράδειγμα εμφανίζει τις 4 πρώτες γραμμές της προπαίδειας. Παρατηρήστε τις τιμές που εμφανίζονται σε σχέση με τις τιμές των μετρητών κατά τη διάρκεια της επανάληψης. Τι πρέπει να αλλάξετε στο πρόγραμμα για να εμφανίζει ολόκληρη την προπαίδεια?

Εμφανίζει δυο σειρές σειρά από 'αστεράκια' μια οριζόντια και μια κάθετη	**Τι εμφανίζει στην οθόνη**

#include <iostream>
using namespace std;
int main() {
  int number;
  cout << "Enter number of stars";
  cin >> number;
 	
  for (**int** i = 1; i<= number; i++) {
  cout << "* " ;
  }
  cout << "vertical line" << endl;
  for (**int** i = 1; i<= 7; i++) {
  cout << "*" << endl;
  }
  return 0;
}
	Πόσα αστεράκια να εμφανίσω?
	7
	* * * * * * *
	vertical line
	*
	*
	*
	*
	*
	*
	*

Εμφανίζει την προπαίδεια	Τι εμφανίζει στην οθόνη
#include <iostream> using namespace std; **int** main() **{** **int** i, j; **cout** << "Προπαίδεια" << endl; **for (** i = 1 ; i<=4; i++) { **for (** j = 1 ; j<=10; j++) **cout** << i * j **cout << endl;** **}** **return 0;** **}**	Προπαίδεια 1 2 3 4 5 6 7 8 9 10 2 4 6 8 10 12 14 16 18 20 3 6 9 12 15 18 21 24 27 30 4 8 12 16 20 24 28 32 36 40



Η σειρά με την οποία εκτελούνται τα βήματα της επανάληψης φαίνεται αν χρησιμοποιήσουμε την δομή επανάληψης while.

```
int i = 1;                          Αρχικοποίηση του μετρητή
while ( i <= 10 ) {                 Έλεγχος της συνθήκης για συνέχεια της επανάληψης
    cout << i << endl;              Εκτέλεση εντολών μέσα στην επανάληψη
    i ← i + 1                       Μεταβολή του μετρητή
}
```

Η εντολή while

Η εντολή while είναι η πιο επεξηγηματική εντολή επανάληψης διότι φαίνονται όλα τα βήματα με τη σειρά με την οποία εκτελούνται. Τα παρακάτω τμήματα εντολών εμφανίζουν όλα τα τέλεια τετράγωνα και έχουν ακριβώς το ίδιο αποτέλεσμα ενώ χρησιμοποιούνται διαφορετικές δομές επανάληψης. Ουσιαστικά είναι απολύτως ισοδύναμα.

Με δομή for	Με δομή while

sqr ← 1; step ← 3;
for (**int** i = 1; i<= 100; i++) {
cout << sqr << " " ;
sqr = sqr + step;
step = step + 2;
}
	sqr ← 1; step ← 3;
	int i = 1; *// αρχική τιμή*
	while (i<= 100) { *// έλεγχος*
	cout << sqr << " " ;
	sqr = sqr + step;
	step = step + 2;
	i++; *// μεταβολή μετρητή*
	}

Φύλλο Εργασίας: *Μεταβλητές, Τύποι, Αριθμητικές Πράξεις*

Δραστηριότητα 1

Γράψτε στο CodeBlocks τα παρακάτω προγράμματα και εκτελέστε τα με τη σειρά.
Ποια είναι η λειτουργία του καθενός; Εξηγήστε τι κάνει κάθε εντολή σε κάθε περίπτωση.
Να γράψετε ένα δικό σας πρόγραμμα το οποίο θα διαβάζει έναν ακέραιο τριψήφιο αριθμό και θα εμφανίζει τα ψηφία του ένα σε κάθε γραμμή.

Πρόγραμμα Α

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

int main()
{
  int x;
  cout << "x = ";
  cin >> x;
  int Fx;
  Fx = x*x*x-6*x*x+11*x-6;
  cout << " ************************** " << endl;
  cout << "     F(" << x << ") = " << Fx << endl;
  cout << " ************************** " << endl;

  return 0;
}
```

Πρόγραμμα Β

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

int main()
{
  int x, y;
  cout << "x = ";         cin >> x;
  cout << " y = ";        cin >> y;
  int sum, prod, quot, rem;
  double p;
  sum = x+y;
  prod = x*y;
  quot = x / y;
  rem = x % y;
  p = (double) x / y;
  cout << " ************************** " << endl;
  cout << "   sum = " << sum << endl;
  cout << "   product = " << prod << endl;
  cout << "   quotient = " << quot << endl;
  cout << "   remainder = " << rem << endl;
  cout << "   p = " << p << endl;
  cout << " ************************** " << endl;
  return 0;
}
```

1. Τροποποιήστε το Α έτσι ώστε να εμφανίζει την τιμή της συνάρτησης $f(x) = 5x - 1$
2. Τροποποιήστε το Α έτσι ώστε να εμφανίζει την τιμή της συνάρτησης $f(x) = x^2 + 1$
3. Τροποποιήστε το Β έτσι ώστε να διαβάζει τρεις αριθμούς και να εμφανίζει το γινόμενο και το άθροισμά τους

Πρόγραμμα Γ

```
#include <iostream>
#include <math.h>
using namespace std;

int main()
{
  int x;
  cout << "x = ";
  cin >> x;
  double root = sqrt(x);
  cout << "   root = " << root << endl;

  return 0;
}
```

1. Να τροποποιήσετε το διπλανό πρόγραμμα ώστε να εμφανίζει τον αριθμό που διάβασε εις την 6η δύναμη. Να αναζητήσετε στο Google τη συνάρτηση υπολογισμού δύναμης.
2. Να γράψετε ένα νέο πρόγραμμα το οποίο θα διαβάζει 3 αριθμούς a, b, c και θα υπολογίζει και θα εμφανίζει τους αριθμούς $D = b^2 - 4ac$ και
   $$x_1 = \frac{-b+\sqrt{b^2-4ac}}{2a}, \quad x_2 = \frac{-b-\sqrt{b^2-4ac}}{2a}$$
   Να το εκτελέσετε και να δώσετε τους αριθμούς 1 1 1. Εξηγήστε τα αποτελέσματα.
3. Να αποθηκεύσετε το πρόγραμμα σαν *trionimo.cpp*

Φύλλο Εργασίας: Λογικές μεταβλητές, χαρακτήρες και συνθήκες

Δραστηριότητα 1

Γράψτε στο CodeBlocks το παρακάτω πρόγραμμα.

Πρόγραμμα Α

```
#include <iostream>
#include <math.h>
using namespace std;

int main()
{
  int x, y;
  bool logical;
  cout << "x = ";    cin >> x;
  cout << "y = ";    cin >> y;
  logical = x > y;
  cout << " true value = " << true << endl;
  cout << " false value = " << false << endl;
  cout << "   logical = " << logical << endl;

  return 0;
}
```

1. Να τροποποιήσετε το διπλανό πρόγραμμα ώστε να εμφανίζει 1 όταν οι αριθμοί είναι ίσοι και 0 σε οποιαδήποτε άλλη περίπτωση
2. Να τροποποιήσετε το πρόγραμμα έτσι ώστε να διαβάζει 4 αριθμούς και να εμφανίζει 1 αν είναι σε αύξουσα σειρά αλλιώς 0.

Δραστηριότητα 2

Γράψτε στο CodeBlocks το παρακάτω πρόγραμμα.

Πρόγραμμα Α

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

int main()
{
  int x, y;
  char letter;
  cout << "gramma = ";    cin >> letter;
  if (letter=='a')
     cout << "bravo" << endl;
  else
     cout << "error" << endl;

  int code = letter;
  cout << code << endl;
  cout << 'c' – 'a' << endl;
  cout <<  '0' << endl;
  cout <<  '0' + 0 << endl;

  return 0;
}
```

1. Να τροποποιήσετε το διπλανό πρόγραμμα ώστε να εμφανίζει «μπράβο» αν δοθεί είτε μικρό είτε κεφαλαίο Α.
2. Να εξηγήσετε τα αποτελέσματα των 4 τελευταίων εντολών
3. Να τροποποιήσετε το πρόγραμμα έτσι ώστε αν δοθεί αριθμός να εμφανίζει «Number», αν δοθεί αγγλικό κεφαλαίο γράμμα να εμφανίζει «capital letter» αν δοθεί μικρό να εμφανίζει «letter» αλλιώς να εμφανίζει «undefined».

Φύλλο Εργασίας: *Δομή Επιλογής*

Δραστηριότητα 1

Γράψτε στο CodeBlocks το παρακάτω πρόγραμμα.

Πρόγραμμα Α

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

int main()
{
 int a, b, x;
 if (a==0 and b==0)
    cout << “άπειρες λύσεις” << endl;

 if (a==0 and b!=0)
    cout << “αδύνατο” << endl;

 if (a!=0) {
    x =  –b / a;
    cout << “ x =  ” << x << endl;
 }

 return 0;
}
```

1. Να τροποποιήσετε το διπλανό πρόγραμμα ώστε να εμφανίζει «μπράβο» αν δοθεί είτε μικρό είτε κεφαλαίο Α.
2. Να τροποποιήσετε το πρόγραμμα έτσι ώστε αν δοθεί αριθμός να εμφανίζει «Number», αν δοθεί αγγλικό κεφαλαίο γράμμα να εμφανίζει «capital letter» αν δοθεί μικρό να εμφανίζει «letter» αλλιώς να εμφανίζει «undefined».

Δραστηριότητα 2

Ανοίξτε το πρόγραμμα trionimo.cpp και τροποποιήστε το έτσι ώστε να εμφανίζει τις ρίζες μόνο αν η διακρίνουσα είναι θετική αλλιώς να εμφανίζει το μήνυμα «η εξίσωση είναι αδύνατη». Επίσης αν έχει διπλή ρίζα να εμφανίζει έχει διπλή ρίζα και να εμφανίζει την τιμή της.

Δραστηριότητα 3

Να γράψετε ένα πρόγραμμα που θα διαβάζει έναν ακέραιο αριθμό και θα υπολογίζει και εμφανίζει την απόλυτη τιμή του.

Δραστηριότητα 4

Να γράψετε ένα πρόγραμμα που θα διαβάζει δύο ακέραιους αριθμούς και θα υπολογίζει και θα εμφανίζει τον μεγαλύτερο. Να το επεκτείνετε για τρεις και στη συνέχεια για τέσσερις αριθμούς.

Δραστηριότητα 5*

Να γράψετε ένα πρόγραμμα που θα διαβάζει οκτώ αριθμούς και θα εμφανίζει τη θέση στην οποία θα βρισκόταν ο έκτος αριθμός αν τοποθετούσαμε τους αριθμούς σε αύξουσα σειρά.

Φύλλο Εργασίας: Αναπαράσταση και Επεξεργασία Χαρακτήρων

Δραστηριότητα 1

Γράψτε στο CodeBlocks το παρακάτω πρόγραμμα.

Πρόγραμμα Α

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

int main()
{
  int x, y;
  char letter;
  cout << "letter = ";    cin >> letter;

  int code = letter;
  cout << code << endl;
  cout << 'c' – 'a' << endl;
  cout <<  '0' << endl;
  cout <<  '0' + 0 << endl;
  cout << 'b' + 2 << endl;
  cout << (int) 'd' << endl;
  cout << (int)('b'+2) << endl;
  return 0;
}
```

1. Να εξηγήσετε τα αποτελέσματα που θα εμφανιστούν στην οθόνη κατά την εκτέλεση του προγράμματος
2. Να εμφανίσετε τους κωδικούς ASCII όλων των αγγλικών γραμμάτων κεφαλαίων και μικρών

Δραστηριότητα 2

Γράψτε στο CodeBlocks το παρακάτω πρόγραμμα.

Πρόγραμμα Β

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

int main()
{
  cout << "*** letter ASCII codes *** " << endl;
  cout << "A = " << (int )'A' << endl;
  cout << "A = " << 'A' + 0 << endl;
  cout << (char)('A'+1) << " = " << 'A' + 1 << endl;
  cout << (char)('A'+2) << " = " << 'A' + 2 << endl;
  cout << (char)('A'+3) << " = " << 'A' + 3 << endl;
  cout << "0 = " << (int)'0'  << endl;
  cout << "9 = " << (int)'9'  << endl;

   char letter;
   cout << "letter = ";    cin >> letter;

   int code = letter;
   cout << code << endl;
   cout << letter – 'a' << endl;

   return 0;
}
```

1. Να εξηγήσετε τα αποτελέσματα που θα εμφανιστούν στην οθόνη κατά την εκτέλεση του προγράμματος
2. Να τροποποιήσετε το πρόγραμμα έτσι ώστε να διαβάζει ένα γράμμα **letter** και έναν αριθμό **k** και στη συνέχεια να εμφανίζει το γράμμα που βρίσκεται **k** θέσεις μετά το γράμμα **letter** στο αλφάβητο, είτε αυτό είναι μικρό είτε κεφαλαίο.
3. Να κάνετε έλεγχο αν ο χαρακτήρας που δίνεται είναι αγγλικό γράμμα κεφαλαίο η μικρό και να εμφανίζετε κατάλληλο μήνυμα. Αν δεν είναι γράμμα να εμφανίζει ότι δεν δόθηκε αγγλικό γράμμα.

Φύλλο Εργασίας: Αναπαράσταση και Επεξεργασία Συμβολοσειρών

Δραστηριότητα 3

Γράψτε στο CodeBlocks το παρακάτω πρόγραμμα.

Πρόγραμμα Γ

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

int main()
{
  string word, encrypted;
  int index, key, length;
  cout << " enter a word :  " << endl;   cin >> word;
  cout << "enter the key : " << endl;     cin >> key;
  length = word.length();
  encrypted = word;
  cout << "before encryption : " << encrypted << endl;
  for (index=0; index<length; index++)
      encrypted[index] = word[index]+key;

  cout << "after encryption : " << encrypted << endl;

   return 0;
}
```

1. Να εξηγήσετε τα αποτελέσματα που θα εμφανιστούν στην οθόνη κατά την εκτέλεση του προγράμματος, αφού πρώτα δώσετε τα λέξη **abracadabra** και το κλειδί **2**.
2. Να δώσετε τη λέξη **xyz** και το κλειδί **2**. Τι παρατηρείτε? Πως το εξηγείτε? Τι πρέπει να διορθώσετε στον κώδικα; Αν δεν είστε σίγουροι ξαναδοκιμάστε με τη λέξη SUSYZ και κλειδί **5**.

Δραστηριότητα 4

Γράψτε στο CodeBlocks το παρακάτω πρόγραμμα.

Πρόγραμμα Δ

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

int main()
{
  char cline[100];
  cin.getline(cline, 100);
  string line(cline);

  cout << "************************" << endl;
  for (int i=0; i<line.length(); i++)
       cout << "#" << line[i] << "#" << endl;
  cout << "************************" << endl;

   return 0;
}
```

1. Να εξηγήσετε τα αποτελέσματα που θα εμφανιστούν στην οθόνη κατά την εκτέλεση του προγράμματος, αφού πρώτα δώσετε την πρόταση **Houston we have a problem!**.
2. Να γράψετε ένα πρόγραμμα που να διαβάζει ένα κλειδί και μια πρόταση και στη συνέχεια να την κρυπτογραφεί με τον αλγόριθμο του Καίσαρα και κλειδί μετάθεσης αυτό που δόθηκε.

Φύλλο Εργασίας: Μετρητές

Δραστηριότητα 5

Γράψτε στο CodeBlocks το παρακάτω πρόγραμμα.

Πρόγραμμα Ε

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

int main()
{
   cout << "Give 3 numbers" << endl;
   int positives, negatives, zeros, a;

   cin >> a;
   if (a>0) positives++;
   else if (a<0) negatives++;
   else zeros++;
   cin >> a;
   if (a>0) positives++;
   else if (a<0) negatives++;
   else zeros++;
   cin >> a;
   if (a>0) positives++;
   else if (a<0) negatives++;
   else zeros++;
   cout << "positives = " << positives << endl;
   cout << "negatives = " << negatives << endl;
   cout << "zeros = " << zeros << endl;
  return 0;
}
```

1. Να εκτελέσετε το πρόγραμμα για τους αριθμούς 100, 0, -90. Τι παρατηρείτε? Τι πρόβλημα υπάρχει? Πως θα το διορθώσετε;
2. Να ξαναγράψετε το πρόγραμμα με δομή επανάληψης αφού εντοπίστε το μπλοκ των εντολών που επαναλαμβάνονται, ώστε να μειώσετε τον όγκο του και να αποφύγετε τους πλεονασμούς.
3. Να τροποποιήσετε το προηγούμενο πρόγραμμα ώστε να υπολογίζει τους θετικούς, αρνητικούς και μηδέν αν δώσετε 10 αριθμούς

Δραστηριότητα 6

Γράψτε στο CodeBlocks το παρακάτω πρόγραμμα.

Πρόγραμμα ΣΤ

```
#include <iostream>
using namespace std;
int main()
{
  int count[10] = {0}, a[1000], N;
  cout << "δώσε πλήθος μονοψήφιων αριθμών < 1000 : ";   cin >> N;
  cout << "δώσε " << N << "μονοψήφιους αριθμούς" << endl;

  for (int i=0; i<N; i++)
       cin >> a[i];
  for (int i=0; i<N; i++)
      count[a[i]]++;

  for (int i=0; i<10; i++) {
      cout << i << " : " << count[i] << " φορές" << endl;
  }
  return 0;
}
```

Να εξηγήσετε τα αποτελέσματα που θα εμφανιστούν στην οθόνη κατά την εκτέλεση του προγράμματος, και τις εντολές μία προς μία.

Πίνακες

Δήλωση και Αρχικοποίηση Πινάκων

Ένας πίνακας (array) αποτελείται από συνεχόμενες θέσεις στη μνήμη που έχουν το ίδιο όνομα και τον ίδιο τύπο.
Στο παρακάτω σχήμα φαίνεται ένας πίνακας από ακέραιους αριθμούς με μέγεθος 10 στοιχεία.

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]
2	4	8	16	32	64	128	256	512	1024

Ο παραπάνω πίνακας ορίζεται με την παρακάτω δήλωση

```
int  a[ 10 ];
```

Στον ορισμό ενός πίνακα το πλήθος των στοιχείων είναι υποχρεωτικά ένας σταθερός αριθμός. Αν θέλουμε να χρησιμοποιήσουμε μια μεταβλητή ως στοιχείο καλύτερου προγραμματισμού τότε δηλώνουμε μια σταθερά όπως παρακάτω:

```
const int  numberOfElements = 10;
int a[ numberOfElements ];
```

Για αναφερθούμε σε ένα στοιχείο του πίνακα χρειάζεται να δηλώσουμε το όνομα του πίνακα και τη θέση αυτού του στοιχείου. Για παράδειγμα για να θέσουμε το στοιχείο στη θέση 2, δηλαδή το 3ο στοιχείο του πίνακα, ίσο με την τιμή 23 δίνουμε την εντολή

```
a[ 2 ] = 23;
```

Αν πάλι θέλουμε να αυξήσουμε την τιμή του στοιχείου a[2] κατά 1 δίνουμε την εντολή

```
a[ 2 ]++;
```

Δηλαδή μπορούμε να διαχειριζόμαστε κάθε στοιχείο ενός πίνακα όπως μια μεταβλητή.

Προσοχή!!! Όπως φαίνεται και από το σχήμα η πρώτη θέση ενός πίνακα είναι η θέση 0. Αυτό σημαίνει ότι όταν ορίζουμε έναν πίνακα 10 στοιχείων, αυτά βρίσκονται στις θέσειςαπό 0→ 9.
Άρα το 4ο στοιχείο βρίσκεται στη θέση 3 του πίνακα και το 10ο στη θέση 9.
Γενικά το n-οστό στοιχείο βρίσκεται στη θέση n-1 του πίνακα, δηλαδή είναι το a[n-1] .

Πως μπορούμε όμως να αρχικοποιήσουμε έναν πίνακα την στιγμή που τον δηλώνουμε; Απλά δίνουμε μέσα σε άγγιστρα τις τιμές των στοιχείων του πίνακα με τη σωστή σειρά. Για παράδειγμα για να αρχικοποιήσουμε τον πίνακα a του σχήματος δίνουμε την παρακάτω εντολή:

```
int  a[10] = { 2, 4, 8, 16, 32, 64, 128, 256, 1024 };
```

Προσοχή!!! Αν δώσετε λιγότερα από δέκα στοιχεία τα υπόλοιπα γίνονται αυτόματα 0.
Δηλαδή οι παρακάτω δηλώσεις είναι ισοδύναμες:

```
int  a[10] = { 1, 2, 3, 4 };
int  a[10] = { 1, 2, 3, 4, 0, 0, 0, 0, 0 };
```

Ένα απλό πρόγραμμα

Θα δώσω ένα απλό πρόγραμμα για να δείτε πως δουλεύουμε με τους πίνακες στη C++. Το πρόγραμμα αυτό διαβάζει έναν πίνακα από ακέραιους αριθμούς, αυξάνει κάθε στοιχείο του πίνακα κατά 1 και εμφανίζει το νέο πίνακα.
Το πρόγραμμα δεν θα είναι μονοκόμματο αλλά θα το χωρίσουμε σε τρία τμήματα. Κάθε ξεχωριστή λειτουργία θα επιτελείται από μια συνάρτηση (έτσι θα κάνετε και εσείς στις ασκήσεις σας από εδώ και πέρα)

```
#include <iostream.h>
#include <iomanip.h>     // Βοηθάει στη στοίχιση αυτών που εμφανίζονται στην οθόνη

void readArray(int a[ ], int n);
void incrementArray(int a[ ], int n);
void printArray(int a[ ], int n);

const int size = 10;

main()
{
  int array[size];

  readArray(array, size);

  incrementArray(array, size);

  printArray(array, size);

  return 0;
}

void readArray(int a[ ], int n)
{
  cout << "Δώσε " << n << " ακέραιους αριθμούς : " ;

  for (int i=0; i<n; i++)      //  Για κάθε στοιχείο του πίνακα αύξησέ το κατά 1
    ++a[i];                    // ακριβώς το ίδιο με το να γράφαμε  a[i] = a[i] + 1
}


void incrementArray(int a[ ]);
void printArray(int a[ ]);
```

ΑΡΧΕΙΑ ΣΤΗ C++

Το παρακάτω πρόγραμμα διαβάζει δυο αριθμούς από το αρχείο με όνομα test.in και γράφει το άθροισμά τους στο αρχείο με όνομα test.out.
Προσέξτε ότι στο τέλος μετά το αποτέλεσμα πρέπει να έχετε και ένα τέλος γραμμής

```
#include <iostream>
#include <fstream>
#include <string>

using namespace std;

int main() {
    ifstream in ("test.in");       // άνοιγμα αρχείου για ανάγνωση
    ofstream out ("test.out");     // άνοιγμα αρχείου για εγγραφή
    int a, b;

    in >> a >> b;                  // διαβάζω 2 αριθμούς από αρχείο
    out << a + b  << endl;         // γράφω το αποτέλεσμα σε αρχείο
                                   // δεν ξεχνάω να βάλω τέλος γραμμής

    in.close(); out.close();       // δεν ξεχνάω να κλείσω τα αρχεία
    return 0;
}
```

Εισαγωγή/Ανάγνωση Ν αριθμών από ένα αρχείο εισόδου με όνομα "input.in" (πρώτα διαβάζουμε το πλήθοςτων αριθμών) και καταχώρησή τους σε πίνακα Ν θέσεων	**Εξαγωγή/Γράψιμο Ν αριθμών σε ένα αρχείο εξόδου με όνομα "exodos.out"**	**Υπολογισμός μέσου όρου, αθροίσματος**

ifstream in("input.in");
in >> N;
for (int i=0; i<N; i++) {
  in >> number[i] ;
}
	ofstream out("exodos.out");
	for (int i=0; i<N; i++) {
	  out << number[i] << endl;
	}
		sum=0;
		for (int i=0; i<N; i++)
		  sum = sum + number[i];
		avg = sum / N;

Προσοχή!!! Στην C++/C οι πίνακες ξεκινάνε από το 0 και όχι από το 1. Δηλαδή τα στοιχεία του παραπάνω πίνακα είναι number[0], number[1],….,number[N–1]. Αν θέλουμε να έχουμε τα στοιχεία number[1]….number[N] τότε πρέπει να δημιουργήσουμε έναν πίνακα με N+1 στοιχεία, απλά δεν θα χρησιμοποιούμε το στοιχείο στη θέση number[0] (όχι καλή ιδέα ☹).

Φύλλο Εργασίας: Πίνακες (Arrays) και Αρχεία (Files)

i) Εισαγωγή 6 αριθμών σε πίνακα από πληκτρολόγιο

```
#include <iostream>
using namespace std;
int main() {
     int a[6];
     cin>>a[0];   cin>>a[1];
     cin>>a[2];   cin>>a[3];
     cin>>a[4];   cin>>a[5];
     cout << a[5] << a[4]
        << …… << a[0] << endl;
}
```

ii) Εισαγωγή 6 αριθμών σε πίνακα από το αρχείο numbers.in

```
#include <fstream>
using namespace std;
int main() {
     int a[6];
     ifstream fin("numbers.in");
     fin >> a[0];  fin >> a[1];
     fin >> a[2];  fin >> a[3];
     fin >> a[4];  fin >> a[5];
     fin.close();
}
```

iii) Εξαγωγή 6 αριθμών στο αρχείο "numbers.out"

```
#include <fstream>
using namespace std;
int main() {
     int a[] = {5, 8, 13, 21, 34, 55};
     ofstream fout("numbers.out");
     fout << a[0];  fout << a[1];
     fout << a[2];  fout << a[3];
     fout << a[4];  fout << a[5];
     fout << endl;
     fout.close();
}
```

Τα i), ii), iii) με δομή επανάληψης (είναι απολύτως ισοδύναμα)

```
#include <iostream>
using namespace std;
int main() {
     int a[6];
     for (int i=0; i<6; i++)
          cin >> a[i] ;
     for (int i=5; i>=0; i--)
          cout << a[i] ;
     cout << endl;
}
```

```
#include <fstream>
using namespace std;
int main() {
     int a[6];
     ifstream fin("numbers.in");
      for (int i=0; i<6; i++)
           fin >> a[i] ;
     fin.close();
}
```

```
#include <fstream>
using namespace std;
int main() {
     int a[] = {5, 8, 13, 21, 34, 55};
     ofstream fout("numbers.out");
     for (int i=0; i<6; i++)
          fout << a[i] ;
     fout.close();}
}
```

i) μετράει τα διπλά (2)

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

int main() {
     int a[6];
     int twos = 0;
     if  (a[0] == 2)
          twos = twos + 1;
     if  (a[1] == 2)
          twos = twos + 1;
     if  (a[2] == 2)
          twos = twos + 1;
      ……
     if  (a[5] == 2)
          twos = twos + 1;

     cout << twos << endl;
}
```

Τελεστής αυτοαύξησης

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

int main() {
     int a[6];
     int twos = 0;
     if  (a[0] == 2)  twos++;
     if  (a[1] == 2)  twos++;
     if  (a[2] == 2)  twos++;
     if  (a[3] == 2)  twos++;
     if  (a[4] == 2)  twos++;
     if  (a[5] == 2)  twos++;

     cout << twos << endl;
}
```

Χρήση μεταβλητής δείκτη

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

int main() {
   int twos = 0;
   int i = 0;
   if  (a[i] == 2)  twos ++ ;
   i = i + 1;
   if  (a[i] == 2)  twos ++;
   i = i + 1;
   if  (a[i] == 2)  twos ++ ;
   i = i + 1;
   if  (a[i] == 2)  twos ++ ;
   i = i + 1;
   if  (a[i] == 2)  twos ++ ;
   i = i + 1;
   if  (a[i] == 2)  twos ++;

  cout << twos << endl;
}
```

μετράει τα διπλά (2)

```
int twos = 0 ;
int i = 0 ;

   if  (a[i] == 2)  twos ++ ;
   i = i + 1;
   if  (a[i] == 2)  twos ++;
   i = i + 1;
   if  (a[i] == 2)  twos ++;
   i = i + 1;
   if  (a[i] == 2)  twos ++;
   i = i + 1;
   if  (a[i] == 2)  twos ++;
   i = i + 1;
   if  (a[i] == 2)  twos ++;
   i = i + 1;

cout << twos << endl;
}
```

Επαναληπτικά

```
int twos = 0;
int i = 0;

for (; i<6; ) {
      if  (a[i] == 2)  twos ++ ;
      i← i + 1;
 }
```

τελική μορφή

```
int twos = 0;

for (int i = 0; i<6; i=i+1) {
      if  (a[i] == 2)  twos ++ ;
 }
```

```
int twos = 0;

for (int i = 0; i<6; i++) {
      if  (a[i] == 2)  twos ++ ;
 }
```

```
for (αρχική τιμή ; συνθήκη; μεταβολή μετρητή, βήμα)  {
        Εντολές που εκτελούνται
}
```

συνθήκη: Όσο η συνθήκη αυτή ισχύει (**Αληθής**) η επανάληψη συνεχίζει να εκτελείται. Π.χ. όσο **i<10** οι εντολές μέσα στο for εκτελούνται. Όταν γίνει ψευδής (φτάσει ή ξεπεράσει το 10) η επανάληψη σταματάει και εκτελείται η εντολή που βρίσκεται αμέσως μετά το } του for.

Δεν κρατάει στη μνήμη όλους τους μετρητές

```
c = 0;
for (int i=0; i<N; i++)
     if  (a[i] == 0)   c++;
cout << c;

c = 0;
for (int i=0; i<N; i++)
     if  (a[i] == 1)   c++;
cout << c;

c = 0;
for (int i=0; i<N; i++)
     if  (a[i] == 2)   c++;
cout << c;

……
……

c = 0;
for (int i=0; i<N; i++)
     if  (a[i] == 2)   c++;
cout << c;
```

Τους κρατάει στη μνήμη στον πίνακα C[101]

```
c[0] = 0;
for (int i=0; i<N; i++)
     if  (a[i] == 0)   c[0]++;
cout << c[0];

c[1] = 0;
for (int i=0; i<N; i++)
     if  (a[i] == 1)   c[1]++;
cout << c[1];

c[2] = 0;
for (int i=0; i<N; i++)
     if  (a[i] == 2)   c[2]++;
cout << c[2];

……
……

c[100] = 0;
for (int i=0; i<N; i++)
     if  (a[i] == 2)   c[100]++;
cout << c[100];
```

Επαναληπτικά

```
for (int num=0; num<N; num++) {
    c[num] = 0;
    for (int i=0; i<N; i++)
        if  (a[i] == num)
            c[num]++;
    cout << c[num];
}
```

ή

```
// μηδενισμός των μετρητών
for (int num=0; num<N; num++)
    c[num] = 0;

/* μέτρημα πόσες φορές εμφανίζεται  κάθε
αριθμός από 1 έως N */
for (int num=0; num<N; num++)
    for (int i=0; i<N; i++)
        if  (a[i] == num)
            c[num]++;

// εμφάνιση αποτελεσμάτων
for (int num=0; num<N; num++)
    cout << c[num];
```

Ταξινόμηση Πίνακα με τη sort της βιβλιοθήκης STL

Στο παρακάτω πρόγραμμα ορίζουμε και αρχικοποιούμε έναν πίνακα με πλήθος 10 στοιχείων. Επίσης ορίζουμε και μια συνάρτηση print η οποία εμφανίζει τα στοιχεία ενός πίνακα σε μια γραμμή. Στη συνέχεια χρησιμοποιούμε τη συνάρτηση sort η οποία ορίζεται στη βιβλιοθήκη *algorithm* για να ταξινομήσουμε τα στοιχεία του πίνακα σε αύξουσα σειρά, δηλαδή από το μικρότερο στο μεγαλύτερο.

```
#include <iostream>
#include <algorithm>

using namespace std;

/*
   Για μην χρειαστεί να επαναλαμβάνω τον κώδικα με την for κάθε φορά που
θέλω να εκτυπώσω τα στοιχεία ενός πίνακα στην οθόνη, ορίζω τη συνάρτηση
print και όποτε καλώ αυτήν θα εκτελείται ο συγκεκριμένος κώδικας.
Η συνάρτηση πρέπει να ξέρει ποιον πίνακα θα εμφανίσει και πόσα στοιχεία
έχει αυτός. Για αυτό περνάω σαν παράμετρο τον πίνακα και το πλήθος των
στοιχείων.
*/
void print(int a[], int len) {
    for (int i=0; i<len; i++)
        cout << a[i] << " ";
    cout << endl;
}

const int N = 10;    // ορίζω μια σταθερά N ως το μέγεθος του πίνακα

int main() {
    int a[N] = {1,2,5,13,21,34,55, 8, 3, 1};     // δημιουργία πίνακα

    print(a,N);                  // τον εμφανίζω πριν

    sort(a, a+N);                // καλώ την sort για ταξινόμηση του a

    print(a, N);                 // τον εμφανίζω μετά την ταξινόμηση

    return 0;
}
```

Τι κάνουμε όμως αν θέλουμε φθίνουσα και όχι αύξουσα ταξινόμηση. Σε αυτή την περίπτωση θα ορίσουμε εμείς τον τρόπο σύγκρισης και θα τον περάσουμε παράμετρο στην sort. Έτσι ορίζουμε την παρακάτω συνάρτηση compare (πριν τη main)

```
bool compare(int a, int b) {   // επιστρέφει true αν το a είναι
                               // μεγαλύτερο του b αλλιώς false
     return a > b;
}
```

και τώρα ο τρόπος κλήσης της sort είναι ο παρακάτω:

```
    sort(a, a+N, compare);      // καλώ την sort για ταξινόμηση του a
```

το υπόλοιπο πρόγραμμα παραμένει το ίδιο. Έτσι μπορούμε να ταξινομήσουμε οποιοδήποτε τύπο στοιχείων ενός πίνακα, αφού μπορούμε εμείς να ορίσουμε τον τρόπο σύγκρισης.

Χρήση της Στοίβας της βιβλιοθήκης STL

Το παρακάτω πρόγραμμα διαβάζει από τον χρήστη αριθμούς και σταματάει την ανάγνωση όταν ο χρήστης δώσει την τιμή 0. Δεν ξέρουμε δηλαδή εκ των προτέρων πόσους αριθμούς θα δώσει ο χρήστης. Οι αριθμοί αποθηκεύονται σε μια δυναμική δομή δεδομένων γνωστή ως στοίβα. Το χαρακτηριστικό της στοίβας είναι ότι ο τελευταίος αριθμός που μπήκε είναι ο πρώτος που θα βγει. Η εισαγωγή και η διαγραφή (γνωστές και ως ώθηση και απώθηση) γίνονται από το πάνω άκρο της στοίβας. Στη συνέχεια εμφανίζονται οι αριθμοί σε αντίστροφη σειρά από αυτή με την οποία δόθηκαν, δηλαδή ο τελευταίος αριθμός εμφανίζεται πρώτος και ο πρώτος τελευταίος.

Η στοίβα είναι ίσως η σημαντικότερη δομή δεδομένων της επιστήμης της πληροφορικής. Χρησιμοποιείται για την επίλυση προβλημάτων, όπως για παράδειγμα την εύρεση μονοπατιού σε λαβύρινθο.

Άσκηση για σκέψη: Πως θα μπορούσε να εκμεταλλευτεί τη στοίβα ο κοντορεβυθούλης για να βρει τον δρόμο της επιστροφής;

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

int main()
{
    cout << "*** Enter integer numbers    ***" << endl;
    cout << "*** Enter 0 for end of input ***" << endl;

    int numbersSize = 0;
    int num;
    stack<int> numberStack;   // δημιουργία στοίβας για ακέραιους

    cin >> num;
    while (num > 0) {
        numberStack.push( num );  // θέτω το num στην κορυφή της στοίβας

        cin >> num;
        numbersSize++;        // μετράω πόσους αριθμούς δίνει ο χρήστης
    }
    cout << numbersSize << " numbers were given." << endl;

    while (!numberStack.empty()) {     // όσο η στοίβα δεν είναι άδεια

        num = numberStack.top();    // θέση στη num το πάνω στοιχείο

        numberStack.pop();          // αφαίρεσε αυτό το στοιχείο από τη
                                    // στοίβα
        cout << num << "  ";
    }
    cout << endl;
    return 0;
}
```

Για να δημιουργήσουμε μια στοίβα που να περιέχει δεδομένα με τύπο τύπος_δ γράφουμε:
`stack< τύπος_δ > stoiva;`
Από εκεί και πέρα οι βασικές λειτουργίες της στοίβας για να προσπελάσουμε το στοιχείο στην κορυφή, να το αφαιρέσουμε και να προσθέσουμε ένα νέο στο τέλος είναι αντίστοιχα οι :
`top( ), pop( ), push (neo_stoixeio)`
τις οποίες καλούμε με τον συμβολισμό τελεία όπως `stoiva.pop()` .

Ασκήσεις

Άσκηση 1
Μια πιτσαρία δίνει στις 3 πίτσες τη μια δώρο. Να γράψετε αλγόριθμο ο οποίος να διαβάζει την τιμή της πίτσας και πόσες πίτσες θα παραγγείλουμε και θα υπολογίζει το συνολικό κόστος. Π.χ. για 10 πίτσες και τιμή 5 ευρώ η μια το συνολικό κόστος είναι 35 ευρώ.

Άσκηση 2
Να γράψετε αλγόριθμο ο οποίος θα διαβάζει ένα ποσό σε ευρώ και θα το μετατρέπει σε δραχμές. Στη συνέχεια θα το εκτυπώνει. Δίνεται ότι 1€ = 340 δρχ.

Άσκηση 3
Να γράψετε αλγόριθμο ο οποίος θα διαβάζει έναν τετραψήφιο ακέραιο αριθμό και θα εμφανίζει τα ψηφία του.

Άσκηση 4
Να γράψετε αλγόριθμο ο οποίος θα διαβάζει έναν τετραψήφιο ακέραιο αριθμό και θα εμφανίζει τον ανάστροφό του. π.χ. ο ανάστροφος του 4876 είναι ο 6784. Οι αριθμοί αυτοί είναι γνωστοί και ως κατοπτρικοί.

Άσκηση 5
Να γράψετε αλγόριθμο ο οποίος να διαβάζει 5 ακέραιους αριθμούς και στη συνέχεια να υπολογίζει και να εμφανίζει πόσοι από αυτούς είναι άρτιοι

Άσκηση 6
Θα γράψετε έναν αλγόριθμο που θα διαβάζει τρεις αριθμούς a, b, c και θα εμφανίζει στην οθόνη

- Το μεγαλύτερο από τους τρεις
- Το μικρότερο
- Το μέσο όρο τους
- Το άθροισμα των τετραγώνων τους
- Τη λύση της εξίσωσης $ax + b = 0$
- Τις λύσεις της εξίσωσης $ax^2 + bx + c = 0$.

Άσκηση 7
Θα γράψετε έναν αλγόριθμο που θα διαβάζει έναν αριθμό και θα ελέγχει αν αυτός είναι άρτιος ή περιττός.

Άσκηση 8
Θα γράψετε έναν αλγόριθμο που θα διαβάζει έναν αριθμό που παριστάνει ένα έτος (π.χ. 2002) και θα ελέγχει αν αυτό είναι δίσεκτο. *Δίνεται ότι ένα έτος είναι δίσεκτο αν είναι αιώνας και διαιρείται με το 400 ή αν δεν είναι αιώνας και διαιρείται με το 4*.

Άσκηση 9
Θα γράψετε έναν αλγόριθμο που θα διαβάζει 100 ακέραιους αριθμούς και θα υπολογίζει το γινόμενό τους, το άθροισμά τους και τον μέσο όρο τους.

Άσκηση 10
Θα γράψετε έναν αλγόριθμο που θα διαβάζει δυο ακέραιους αριθμούς και θα υπολογίζει το γινόμενό τους χωρίς να χρησιμοποιεί την πράξη του πολλαπλασιασμού.

Άσκηση 11

Ο Δείκτης Μάζας του ανθρώπινου Σώματος (ΔΜΣ) υπολογίζεται από το βάρος (Β) σε χλγ. και το ύψος (Υ) σε μέτρα με τον τύπο $\Delta\text{ΜΣ} = \frac{\text{Β}}{\text{Υ}^2}$.

Ο ανωτέρω τύπος ισχύει για άτομα άνω των 18 ετών. Το άτομο ανάλογα με την τιμή του ΔΜΣ χαρακτηρίζεται σύμφωνα με τον παρακάτω πίνακα:

ΔΜΣ < 18,5	"αδύνατο άτομο"
18,5 ≤ ΔΜΣ < 25	"κανονικό άτομο"
25 ≤ ΔΜΣ < 30	"βαρύ άτομο"
30 ≤ ΔΜΣ	"υπέρβαρο άτομο"

Να γράψετε αλγόριθμο ο οποίος:

α. να διαβάζει την ηλικία, το βάρος και το ύψος του ατόμου

β. εάν η ηλικία είναι μεγαλύτερη των 18 ετών, τότε

1. να υπολογίζει το ΔΜΣ

2. να ελέγχει την τιμή του ΔΜΣ από τον ανωτέρω πίνακα και να εμφανίζει τον αντίστοιχο χαρακτηρισμό

γ. εάν η ηλικία είναι μικρότερη ή ίση των 18 ετών, τότε να εμφανίζει το μήνυμα "δεν ισχύει ο δείκτης ΔΜΣ".

Άσκηση 12

Το κλασικό παιχνίδι «Πέτρα-Ψαλίδι-Χαρτί» παίζεται με δύο παίκτες. Σε κάθε γύρο του παιχνιδιού, ο κάθε παίκτης επιλέγει ένα από τα ΠΕΤΡΑ, ΨΑΛΙΔΙ, ΧΑΡΤΙ, και παρουσιάζει την επιλογή του ταυτόχρονα με τον αντίπαλό του. Η ΠΕΤΡΑ κερδίζει το ΨΑΛΙΔΙ, το ΨΑΛΙΔΙ το ΧΑΡΤΙ και το ΧΑΡΤΙ την ΠΕΤΡΑ. Σε περίπτωση που οι δύο παίκτες έχουν την ίδια επιλογή, ο γύρος λήγει ισόπαλος. Να αναπτύξετε αλγόριθμο ο οποίος διαβάζει τα ονόματα των δύο παικτών και υλοποιεί το παραπάνω παιχνίδι **για έναν μόνο γύρο** ως εξής:

Α)Διαβάζει την επιλογή κάθε παίκτη, η οποία μπορεί να είναι μία από τις εξής: ΠΕΤΡΑ, ΨΑΛΙΔΙ, ΧΑΡΤΙ

Β)Συγκρίνει τις επιλογές των παικτών και διαπιστώνει το νικητή του γύρου ή την ισοπαλία.

Γ). Εμφανίζει το όνομα του νικητή ή, αν δεν υπάρχει νικητής, το μήνυμα «ΤΟ ΠΑΙΧΝΙΔΙ ΕΛΗΞΕ ΙΣΟΠΑΛΟ».

Δ*) Σχεδιάστε το παιχνίδι για περισσότερους γύρους. Στο τέλος κάθε γύρου το πρόγραμμα θα ρωτάει τον χρήστη "Continue? (yes/no)". Αν ο χρήστης δώσει no το παιχνίδι θα τερματίζει.

Άσκηση 13

Θα γράψετε έναν αλγόριθμο που θα διαβάζει έναν περιττό αριθμό *n* και θα εμφανίζει στην οθόνη ένα τρίγωνο του οποίου η βάση θα έχει n χαρακτήρες "*". π.χ. για *n = 3* και *n = 5* εμφανίζονται στην οθόνη τα αντίστοιχα τρίγωνα:

```
  *        *
* * *    * * *
       * * * * *
```

Άσκηση 14

Δίνονται οι ακέραιοι αριθμοί **α, β.**

Θα γράψετε έναν αλγόριθμο που θα υπολογίζει

i. Το πηλίκο της διαίρεσης **α / β**

ii. Το υπόλοιπο της διαίρεσης **α / β**

χωρίς να χρησιμοποιεί τις πράξεις του πολλαπλασιασμού και της διαίρεσης, *,/

Άσκηση 15
Θα γράψετε έναν αλγόριθμο που θα διαβάζει τον αριθμό *n* και θα εμφανίζει στην οθόνη όλους τους διαιρέτες του αριθμού.

Άσκηση 16
Θα γράψετε έναν αλγόριθμο που θα διαβάζει τον αριθμό *n* και θα ελέγχει αν είναι πρώτος. (Πρώτοι λέγονται οι αριθμοί που διαιρούνται μόνο με το 1 και τον εαυτό τους.)

Άσκηση 17
Θα γράψετε έναν αλγόριθμο που θα διαβάζει τον αριθμό *n* και θα ελέγχει αν είναι τέλειος.
(Τέλειος είναι ένας αριθμός αν είναι ίσος με το άθροισμα των διαιρετών του)

Άσκηση 18*
Δύο αριθμοί λέμε ότι είναι φιλικοί (amicable) αν το άθροισμα των διαιρετών του ενός είναι ίσο με τον άλλον. Να διαβάζει δύο αριθμούς και να ελέγχει αν είναι φιλικοί.

Άσκηση 19*
Θα γράψετε έναν αλγόριθμο που θα διαβάζει έναν αριθμό και θα εμφανίζει στην οθόνη όλα τα ψηφία του ξεχωριστά.

Άσκηση 20
Θα γράψετε έναν αλγόριθμο που θα διαβάζει έναν τετραψήφιο αριθμό *n* και θα **υπολογίζει** τον κατοπτρικό του. Π.χ. 12234 ➔ 43221

Άσκηση 21***
Θα γράψετε έναν αλγόριθμο που θα διαβάζει τον αριθμό *n* και θα υπολογίζει τον κατοπτρικό του.
Π.χ. 11223455 ➔ 55432211

Άσκηση 22***
Θα γράψετε έναν αλγόριθμο που θα διαβάζει έναν αριθμό *n* και θα ελέγχει αν είναι παλινδρομικός. Παλινδρομικός είναι ο αριθμός που είναι ίσος με τον κατοπτρικό του. π.χ. 1221.

Άσκηση 23*
Θα γράψετε έναν αλγόριθμο ο οποίος θα εμφανίζει όλους τους τετραψήφιους αριθμούς που είναι ίσοι με το άθροισμα των ψηφίων τους υψωμένων στην τέταρτη δύναμη. Οι αριθμοί αυτοί λέγονται αριθμοί *Armstrong*.
π.χ. $1634 = 1^4 + 6^4 + 3^4 + 4^4$

Άσκηση 24*
Θα γράψετε έναν αλγόριθμο ο οποίος θα εμφανίζει όλους τους τετραψήφιους αριθμούς που είναι ίσοι με το άθροισμα των ψηφίων τους υψωμένων στην τέταρτη δύναμη.
π.χ. $1634 = 1^4 + 6^4 + 3^4 + 4^4$

Άσκηση 25*
Οι αριθμοί Armstrong είναι οι αριθμοί που είναι ίσοι με το άθροισμα των ψηφίων τους υψωμένων στη δύναμη του πλήθους τους. Για παράδειγμα $1634 = 1^4 + 6^4 + 3^4 + 4^4$, $371 = 3^3+7^3+1^3$, $1^1=1$. Οι αριθμοί αυτοί είναι γνωστοί και ως νάρκισσοι αριθμοί.
Θα γράψετε έναν αλγόριθμο ο οποίος θα εμφανίζει όλους τους αριθμούς *Armstrong*.

Άσκηση 26*
Μια κατηγορία πολύ σπάνιων αριθμών είναι αυτοί που είναι ίσοι με το γινόμενο του αθροίσματος και του γινομένου των ψηφίων τους. π.χ. ένας από αυτούς είναι ο
144 = (1+4+4) x (1x4x4). Μπορείτε να βρείτε τους υπόλοιπους;

Άσκηση 27***
Οι αριθμοί Dudeney είναι οι θετικοί αριθμοί οι οποίοι είναι τέλειοι κύβοι και το άθροισμα των ψηφίων τους είναι ίσο με την κυβική τους ρίζα. Π.χ. 512 = 8^3 και 5+2+1=8. Έχει αποδειχτεί ότι υπάρχουν μόνο 6 τέτοιοι αριθμοί. Μπορείτε να τους βρείτε;

Άσκηση 28***
Οι αριθμοί Μυνχάουζεν είναι οι αριθμοί που είναι ίσοι με το άθροισμα των ψηφίων τους υψωμένων στη δύναμη του ίδιου του ψηφίου. Για παράδειγμα 3435 = $3^3 + 4^4 + 3^3 + 5^5$. Πόσοι άλλοι αριθμοί Μυνχάουζεν υπάρχουν στο σύμπαν;

Άσκηση 29
Η ακολουθία αριθμών στην οποία ο κάθε αριθμός είναι ίσος με το άθροισμα των δύο προηγούμενων είναι γνωστή ώς **ακολουθία Fibonacci**: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181, ... Να γράψετε αλγόριθμο που θα διαβάζει έναν αριθμό n και θα εμφανίζει τους n πρώτους αριθμούς Fibonacci.

Άσκηση 30**
Θα γράψετε έναν αλγόριθμο που θα υπολογίζει όλες τις ακέραιες ρίζες μιας πολυωνυμικής συνάρτησης στο διάστημα [α,β]. Θα διαβάζει από το πληκτρολόγιο **α)** το βαθμό του πολυωνύμου **β)** τους αριθμούς α, β **γ)** τους συντελεστές του πολυωνύμου
**Μπορείτε με χρήση του θεωρήματος Bolzano να κάνετε τον αλγόριθμό σας να συγκλίνει αρκετά πιο γρήγορα στη λύση?

Άσκηση 31**
Να σχεδιάσετε αλγόριθμο ο οποίος να διαβάζει δύο ακέραιους αριθμούς και να εμφανίζει τον Μέγιστο Κοινό Διαιρέτη τους. Να σχεδιάσετε έναν αποδοτικό αλγόριθμο. Ο αλγόριθμος του Ευκλείδη δεν είναι δεκτός. Στη συνέχεια να υπολογίσετε και το ελάχιστο κοινό πολλαπλάσιο.

Άσκηση 32***
Ένα από τα άλυτα προβλήματα της θεωρίας αριθμών που έχουν εφαρμογή στην κρυπτογραφία είναι η ανάλυση ενός ακέραιου αριθμού σε πρώτους παράγοντες.
Να σχεδιάσετε αλγόριθμο ο οποίος να διαβάζει έναν ακέραιο αριθμό και να τον αναλύει σε γινόμενο πρώτων παραγόντων με όσο πιο αποδοτικό τρόπο μπορείτε. Για παράδειγμα η ανάλυση του αριθμού 300 είναι $\mathbf{300 = 3 \cdot 2 \cdot 2 \cdot 5 \cdot 5}$

Θέμα 33**
Ο Ινδός μαθηματικός και αστρονόμος Madhava υπολόγισε την τιμή του π σύμφωνα με τη σχέση:

$$\pi = \sqrt{12}\left(1 - \frac{1}{3 \cdot 3} + \frac{1}{5 \cdot 3^2} - \frac{1}{7 \cdot 3^3} + \frac{1}{9 \cdot 3^4} - \frac{1}{11 \cdot 3^5} + \ldots\right)$$

Να σχεδιαστεί αλγόριθμος που θα διαβάζει σαν είσοδο ένα θετικό αριθμό κ. Ο υπολογισμός θα σταματά όταν η απόλυτη τιμή κάποιου όρου γίνει μικρότερη από κ.
Ο αλγόριθμος θα εκτυπώνει την τιμή του π που υπολόγισε, καθώς και το πλήθος των όρων του αθροίσματος που χρειάστηκαν μέχρι να ικανοποιηθεί η συνθήκη διακοπής.

Χαλασμένο Χρονόμετρο

Ο Μίλτος το δελφίνι είναι ένα πολύ ταξιδιάρικο δελφίνι και λατρεύει να διανύει μεγάλες αποστάσεις και να χρονομετρεί τον εαυτό του. Το χρονόμετρο όμως που χρησιμοποιεί έχει δυστυχώς χαλάσει και δείχνει τη μισή ώρα από αυτή που έπρεπε να δείξει κανονικά. Δηλαδή αν ο Μίλτος κάνει μια απόσταση σε 38 δευτερόλεπτα το χρονόμετρο θα δείξει ότι πέρασαν μόνο 19 δευτερόλεπτα.

Ο Μίλτος δεν ξέρει τι να κάνει και στράφηκε σε εσένα να τον βοηθήσεις. Μπορείς να γράψεις ένα πρόγραμμα που να δέχεται σαν είσοδο τις λανθασμένες ενδείξεις του χρονομέτρου και να υπολογίζει ποιες είναι οι σωστές;

Δεδομένα Εισόδου (time.in)
Το πρόγραμμα θα δέχεται σαν είσοδο τις λανθασμένες ενδείξεις του χρονομέτρου στη μορφή ΩΩ:ΛΛ:ΔΔ, όπου ΩΩ, ΛΛ, ΔΔ είναι ένας διψήφιος ακέραιος με 00<=ΩΩ<=99, 00<=ΛΛ<=59, 00<=ΔΔ<=59. Δεν θα υπάρχει είσοδος στην οποία θα έχουν περάσει περισσότερες από 50 ώρες.

Δεδομένα Εξόδου (time.out)
Το πρόγραμμα θα τυπώνει ποια είναι η ένδειξη που θα έπρεπε να δείξει το χρονόμετρο στην ίδια μορφή με την ένδειξη της εισόδου.

Παράδειγμα εισόδου (αρχείο "time.in")	**Παράδειγμα εξόδου (αρχείο "time.out")**
`12:44:03`	`25:28:06`

Εισιτήρια για το ενυδρείο

Το σχολείο σου ετοιμάζει μια εκδρομή για τους μαθητές στο ενυδρείο και εσύ ανέλαβες να πάρεις τηλέφωνο και να ρωτήσεις για την έκπτωση που θα έχετε μιας και θα πάνε τόσα πολλά άτομα. Ο διευθυντής του σχολείου σε έχει πληροφορήσει ότι Ν άτομα έχουν δηλώσει συμμετοχή. Ο υπεύθυνος του ενυδρείου σου είπε ότι αν πάνε τουλάχιστον Α άτομα θα έχετε 10% έκπτωση, αν πάνε τουλάχιστον Β 20%, τουλάχιστον Γ 30% και τουλάχιστον Δ 50%. Η τιμή του κανονικού εισιτηρίου είναι 10 ευρώ.

Μπορείς να βρεις ποιο είναι το ελάχιστο συνολικό ποσό που πρέπει να πληρώσει το σχολείο έτσι ώστε να μπορέσουν να πάνε όλοι οι Ν μαθητές στο ενυδρείο. Προσοχή, σε ορισμένες περιπτώσεις μπορεί να συμφέρει να αγοραστούν περισσότερα εισιτήρια ώστε να έχετε καλύτερο πακέτο προσφοράς.

Δεδομένα Εισόδου (tickets.in)
Η πρώτη γραμμή θα περιέχει έναν ακέραιο Ν (Ν <= 10000). Η δεύτερη γραμμή θα περιέχει τους 4 ακεραίους Α, Β, Γ και Δ.
(Α,Β,Γ,Δ <= 10000)

Δεδομένα Εξόδου (tickets.out)
Ένας ακέραιος, η ελάχιστη συνολική τιμή.

Παράδειγμα εισόδου (αρχείο "tickets.in")	**Παράδειγμα εξόδου (αρχείο "tickets.out")**
30 10 20 50 100	240
Παράδειγμα εισόδου 2 (αρχείο "tickets.in")	**Παράδειγμα εξόδου 2 (αρχείο "tickets.out")**
95 10 20 50 100	500

Λήψη Αρχείων

Κατεβάζεις κάποια αρχεία από το διαδίκτυο και θες να ξέρεις σε πόσο χρόνο θα έχουν κατέβει πλήρως όλα.

Για κάθε αρχείο που κατεβαίνει γνωρίζεις την τρέχουσα ταχύτητα (σε KB/s) και τον χρόνο που απομένει σε κάθε αρχείο βασισμένο σε αυτή την ταχύτητα (σε δευτερόλεπτα). Το άθροισμα των ταχυτήτων όλων των αρχείων είναι ίσο με το bandwidth, το οποίο παραμένει σταθερό και χρησιμοποιείται πλήρως όσο κατεβαίνουν τα αρχεία. Αυτό σημαίνει πως όταν η λήψη ενός αρχείου ολοκληρωθεί, το διαθέσιμο bandwidth θα κατανεμηθεί στα υπόλοιπα αρχεία με αποτέλεσμα να αρχίσουν να κατεβαίνουν πιο γρήγορα. Ο τρόπος με τον οποίο θα κατανεμηθεί δεν επηρεάζει την τελική απάντηση.

Για παράδειγμα, έστω ότι κατεβαίνουν 2 αρχεία:
1) Ταχύτητα = 3 KB/s, Απομένουν 57 δευτερόλεπτα
2) Ταχύτητα = 2 KB/s, Απομένουν 22 δευτερόλεπτα

Μετά από 22 δευτερόλεπτα, το δεύτερο αρχείο θα έχει ολοκληρωθεί. Το πρώτο θα χρειάζεται όμως 35 δευτερόλεπτα ακόμα, αλλά αυτός ο χρόνος έχει υπολογιστεί με βάση την αρχική του ταχύτητα. Λόγω όμως του διαθέσιμου bandwidth, η ταχύτητα του θα αυξηθεί από 3 σε 5 KB/s. Ο νέος χρόνος που απομένει είναι: 35*3/5 = 21 δευτερόλεπτα. Οπότε ο χρόνος που χρειάζεται για να έχουν κατέβει πλήρως και τα 2 αρχεία είναι 22+21=43 δευτερόλεπτα.

Δεδομένα Εισόδου (downloads.in)

Η πρώτη γραμμή της εισόδου περιέχει τον αριθμό N (1<=N<=100) που είναι ο αριθμός των αρχείων που κατεβαίνουν. Ακολουθούν N γραμμές οι οποίες περιέχουν 2 ακεραίους T_ν και X_ν, την ταχύτητα και τον χρόνο αντίστοιχα του ν-οστού αρχείου

Δεδομένα Εξόδου (downloads.out)

Το πρόγραμμα πρέπει να τυπώνει σε μια γραμμή έναν αριθμό, το χρόνο (σε δευτερόλεπτα) που χρειάζεται για να κατέβουν όλα τα αρχεία. Ο αριθμός πρέπει να είναι στρογγυλοποιημένος στον πλησιέστερο ακέραιο.

αρχείο "downloads.in"	**αρχείο "downloads.out"**
2 3 57 2 22	43

Εικασία του Goldbach

Ένα από τα μεγαλύτερα άλυτα προβλήματα της αρχαιότητας αποτελεί η εικασία του Goldbach, η οποία δηλώνει ότι κάθε άρτιος αριθμός μεγαλύτερος του 2 μπορεί να γραφεί ως άθροισμα δύο πρώτων αριθμών. Ως πρώτος αριθμός ορίζεται κάθε φυσικός αριθμός μεγαλύτερος του 1, ο οποίος έχει ως διαιρέτες μόνο το 1 και τον εαυτό του. Έτσι οι πρώτοι δέκα πρώτοι αριθμοί είναι: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29.

Καλείστε να κατασκευάσετε πρόγραμμα προκειμένου να επαληθεύσετε την εικασία του Goldbach.

Δεδομένα Εισόδου (goldbach.in)

Το αρχείο εισόδου αποτελείται από έναν άρτιο ακέραιο αριθμό N (2 < N <= 1000000)

Δεδομένα Εξόδου (goldbach.out)

Το αρχείο εξόδου θα αποτελείται από μία μόνο γραμμή με δύο πρώτους αριθμούς P1 P2 χωρισμένους από ένα κενό. Θα πρέπει να ισχύει P1 + P2 = N. Σε περίπτωση που υπάρχουν πολλές λύσεις τυπώστε αυτή με το μικρότερο P1.

αρχείο "goldbach.in"	**αρχείο "goldbach.out"**
142 22	3 11

Πως λένε το δελφίνι;

Το όνομα κάθε δελφινιού είναι πολύ σημαντικό καθώς υποδηλώνει πολλά στοιχεία του χαρακτήρα του. Κάποιες φορές όμως υπάρχουν πολλά ονόματα εξίσου καλά για ένα δελφίνι. Για να αποφύγουν τα διλήμματα και να απλοποιήσουν τη διαδικασία επιλογής σκέφτηκαν την παρακάτω μέθοδο.

Όλα τα επιθυμητά ονόματα συγκεντρώνονται σε μια λίστα και από αυτά επιλέγεται το όνομα που έχει έστω και μία ισοδύναμη μορφή που είναι λεξικογραφικά μικρότερη από όλα τα άλλα ονόματα και τις ισοδύναμες μορφές τους.

Δυο λέξεις είναι ισοδύναμες αν μπορούμε από την πρώτη να πάμε στη δεύτερη βγάζοντας το τελευταίο γράμμα κάθε φορά και τοποθετώντας το στην αρχή. Για παράδειγμα οι λέξεις MILTOS, SMILTO, OSMILT, TOSMIL, LTOSMI, ILTOSM είναι ισοδύναμες.

Έστω ότι έχουν επιλεχθεί τα παρακάτω ονόματα : MILTOS ALKIS THALIS

Σύμφωνα με την παραπάνω μέθοδο το επικρατέστερο όνομα είναι το THALIS. Η ισοδύναμη μορφή του ALISTH είναι μικρότερη λεξικογραφικά από κάθε άλλο όνομα και κάθε ισοδύναμη μορφή του. Δηλαδή:
ALISTH < MILTOS, SMILTO, OSMILT, TOSMIL, LTOSMI, ILTOSM
ALISTH < ALKIS, SALKI, ISALK, KISAL, LKISA

Δεδομένα Εισόδου (names.in)

Η πρώτη γραμμή της εισόδου περιέχει τον αριθμό N (1<=N<=1000) που είναι ο αριθμός των επιθυμητών ονομάτων. Ακολουθούν N λέξεις, μία σε κάθε γραμμή, τα επιθυμητά ονόματα. Κάθε όνομα αποτελείται από 20 το πολύ κεφαλαίους λατινικούς χαρακτήρες.

Δεδομένα Εξόδου (names.out)

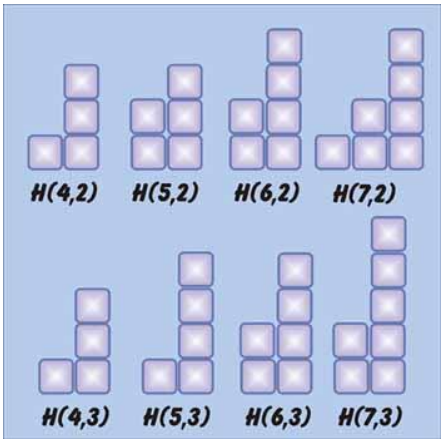
Το πρόγραμμα πρέπει να τυπώνει σε μια γραμμή το όνομα που επιλέγεται από την παραπάνω μέθοδο.

αρχείο "names.in"	**αρχείο "names.out"**
3 MILTOS ALKIS THALIS	THALIS

Στοιβάζοντας κύβους

Σου έχουν δοθεί N κύβοι 1x1x1 τους οποίους θες να τους στοιβάξεις με τέτοιο τρόπο έτσι ώστε να δημιουργήσεις μια σκάλα. Στόχος σου είναι να χρησιμοποιήσεις αυτή τη σκάλα για να φτάσεις ένα συγκεκριμένο ύψος. Πρέπει να φτιάξεις τη σκάλα με τέτοιο τρόπο ώστε να μη χρειαστεί πηδώντας από τη μία σκάλα στην άλλη να ανέβεις ύψος μεγαλύτερο από K κύβους.

Έστω H(N,K) το μέγιστο ύψος που μπορείς να ανέβεις χρησιμοποιώντας N κύβους αν μπορείς να ανεβαίνεις το πολύ K κύβους τη φορά. Η παρακάτω εικόνα δείχνει τη λύση για μικρές τιμές του N και του K.



Καλείσαι να φτιάξεις ένα πρόγραμμα που θα υπολογίζει το H(N,K).

Υπόδειξη: Το πρόβλημα του να βρεις ποια είναι τα ελάχιστα κυβάκια που χρειάζεσαι για να ανέβεις ένα συγκεκριμένο ύψος, δηλαδή να βρείς το ελάχιστο N συναρτήσει του H και του K, είναι αρκετά πιο εύκολο από το παραπάνω. Πώς μπορείς να χρησιμοποιήσεις τη λύση στο πρόβλημα αυτό για να λύσεις το ζητούμενο;

Δεδομένα Εισόδου (cubes.in)	**Δεδομένα Εξόδου (cubes.out)**
Σε μια γραμμή θα υπάρχουν δύο ακέραιοι N (1<=N<=100000) και K (1<=K<=1000).	Ένας ακέραιος, το μέγιστο ύψος που μπορείς να φτάσεις.
Παράδειγμα εισόδου (αρχείο "cubes.in")	**Παράδειγμα εξόδου (αρχείο "cubes.out")**
7 3	5

Ασανσέρ για δελφίνια

Ο βυθός της θάλασσας πάντα προκαλούσε μια ιδιαίτερη αίσθηση στα δελφίνια. Εκεί μπορούν να βρουν πολλά διαφορετικά είδη φυτών και ψαριών. Τους αρέσει λοιπόν να πηγαίνουν όσο πιο βαθιά μπορούν για να ανακαλύψουν νέες περιοχές. Για το λόγο αυτό έφτιαξαν ένα ασανσέρ που μπορεί να τους πάει σε περιοχές που θα ήταν αδύνατο να πάνε κανονικά.

Όλα τα δελφίνια θέλουν να χρησιμοποιήσουν το καινούριο ασανσέρ. Το ασανσέρ αυτό όμως χωράει το πολύ 2 δελφίνια τη φορά και είναι πολύ επικίνδυνο να φορτώνεται με μεγάλο βάρος.

Αν ξέρεις πόσα δελφίνια θέλουν να χρησιμοποιήσουν το ασανσέρ και ποιο είναι το βάρος του κάθε δελφινιού μπορείς να βρεις πως πρέπει να γίνουν οι διαδρομές έτσι ώστε να είναι οι λιγότερες δυνατές και το μέγιστο βάρος που θα μπει στο ασανσέρ σε κάποια διαδρομή να είναι όσο μικρότερο γίνεται.

Δεδομένα Εισόδου (elevator.in)

Στην πρώτη γραμμή θα δίνεται ένας αριθμός N (1<=N<=100000) το πλήθος των δελφινιών. Οι επόμενες N γραμμές περιέχουν έναν ακέραιο Wi (1<=Wi<=100000), το βάρος του κάθε δελφινιού.

Δεδομένα Εξόδου (elevator.out)

Πρέπει να δίνονται δύο αριθμοί χωρισμένοι με ένα κενό, το πλήθος των διαδρομών που πρέπει να γίνουν και το μέγιστο συνολικό βάρος που θα μπει στο ασανσέρ σε κάποια διαδρομή.

αρχείο "elevator.in"	**αρχείο "elevator.out"**
6 1 4 3 50 5 4	3 51
3 12 5 7	2 12