Για να λύσουμε ένα πρόβλημα στη C++ χρειαζόμαστε δυο βασικές έννοιες. Η μια είναι οι οδηγίες – εντολές, ο αλγόριθμος δηλαδή, που πρέπει να ακολουθήσουμε για να λύσουμε το πρόβλημά μας και η άλλη είναι τα δεδομένα του προβλήματος (οι αριθμοί κ.τ.λ.). Το σύνολο των εντολών που δίνουμε, αποτελεί αυτό που λέγεται πρόγραμμα (program), ενώ οι αριθμοί, χαρακτήρες κ.τ.λ. που δίνουμε, αποτελούν τα δεδομένα (data).
Η C++ σα γραπτή γλώσσα που είναι, έχει ένα αλφάβητο. Αυτό αποτελείται από:

1. Τα κεφαλαία γράμματα του αγγλικού αλφάβητου: A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,S,T,U,V,W,X,Y,Z.
2. Τα μικρά γράμματα του αγγλικού αλφάβητου: a,b,c,d,e,f,g,h,I,j,k,l,m,n,o,p,q,r,s,,u,v,w,x,y,z.
3. Τα δέκα ψηφία του δεκαδικού συστήματος: 0,1,2,3,4,5,6,7,8,9.
4. Τα ειδικά σύμβολα: ! # $ % & ‘ ( ) * + - / > < : ; (κενό) . = , \ [ ] ^ { } ? _ “ |

Προκειμένου να γράψουμε τις διάφορες εντολές της C++ χρησιμοποιούμε τα παραπάνω σύμβολα και μόνο αυτά.
Η C++ ,βέβαια, δέχεται και τα γράμματα του ελληνικού αλφαβήτου, κεφαλαία και μικρά, αλλά μόνο στα δεδομένα.

Τα δεδομένα στη C++ χωρίζονται σε 3 κατηγορίες: τους αριθμούς (numbers), τους χαρακτήρες (characters) και τις συμβολοσειρές (strings).

<u>Οι αριθμοί</u>
Οι αριθμοί είναι οι γνωστοί μας αριθμοί του δεκαδικού συστήματος, ακέραιοι ή πραγματικοί, θετικοί ή αρνητικοί. Οι αριθμοί όμως, που δέχεται και επεξεργάζεται ο υπολογισμοί δε μπορεί να έχουν απεριόριστο πλήθος ψηφίων, ούτε το μέγεθός τους να είναι όσο μεγάλο ή όσο μικρό θέλουμε. Έτσι έχουμε ορισμένους περιορισμούς, τόσο στο πλήθος των σημαντικών ψηφίων, που μπορεί να έχει ένας αριθμός, όσο και στο μέγεθος του αριθμού.
Σημαντικά ψηφία: Με τον όρο αυτό εννοούμε όλα τα ψηφία του αριθμού, εκτός από τυχόν μηδενικά που υπάρχουν μπροστά από το πρώτο μη μηδενικό ψηφίο ή μηδενικά που υπάρχουν μετά το τελευταίο μη μηδενικό ψηφίο μετά την υποδιαστολή (αν πρόκειται για πραγματικό αριθμό).
π.χ. ο αριθμός 04.001807600 έχει 8 σημαντικά ψηφία. **Για τους ακεραίους αριθμούς έχουμε 10 το πολύ σημαντικά ψηφία και για τους πραγματικούς 7 το πολύ σημαντικά ψηφία.**
Μέγεθος αριθμού: Ο μεγαλύτερος και ο μικρότερος αριθμός, θετικός ή αρνητικός, που μπορεί να δεχτεί ο υπολογιστής.

Για τους **ακεραίους** τα όρια είναι:
α) -32768 έως και +32767
β) -2147483648 έως και +2147483647 για μεγάλους ακεραίους.
Η δηλωτική εντολή για τους ακεραίους της πρώτης κατηγορίας είναι: int.
Η δηλωτική εντολή για τους ακεραίους της δεύτερης κατηγορίας είναι: long int ή απλά long.
Για τους **πραγματικούς** τα όρια είναι:
$-3.4028235\times10^{+38}$ έως και -1.1754944×10^{-38}, για αρνητικούς και $+1.1754944\times10^{-38}$ έως και $+3.4028235\times10^{+38}$, για θετικούς
Η δηλωτική εντολή για τους πραγματικούς είναι: float.
Αν δώσουμε έναν πραγματικό αριθμό μεγαλύτερο π.χ. του $3.4028235\ 10^{+38}$ ή προκύψει τέτοιος αριθμός μετά από πράξεις, τότε ο υπολογιστής θα μας τυπώσει, την

ένδειξη OVERFLOW (πολύ μεγάλος αριθμός) και θα σταματήσει την εκτέλεση του προγράμματος.
Αντίστοιχα αν ο αριθμός ή το αποτέλεσμα είναι πολύ μικρό, τότε θα μας δώσει την ένδειξη ένδειξη UNDERFLOW (πολύ μικρός αριθμός) οπότε πάλι θα σταματήσει η εκτέλεση του προγράμματος.

Αριθμοί διπλής ακρίβειας

Προκειμένου να έχουμε μεγαλύτερη ακρίβεια και να περιορίσουμε τους περιορισμούς χρησιμοποιούμε τους αριθμούς διπλής ακρίβειας. Με τους αριθμούς αυτούς τα όρια είναι:

$-1.797693134862316\times10^{+308}$ έως και $-2.225073858507201\times10^{-308}$, για αρνητικούς και $2.225073858507201\times10^{-308}$ έως και $1.797693134862316\times10^{+308}$, για θετικούς και έχουμε μέχρι και 16 ψηφία. Κάθε αριθμός διπλής ακρίβειας βέβαια πιάνει διπλάσιο χώρο μέσα στον υπολογιστή, με αποτέλεσμα να χωράνε συνολικά λιγότεροι αριθμοί.

Η δηλωτική εντολή για τους αριθμούς διπλής ακρίβειας είναι: double.

Να σημειωθεί ότι η υποδιαστολή στους πραγματικούς αριθμούς δηλώνεται με τελεία και όχι με κόμμα. Οι εκθετικοί αριθμοί στη θέση της βάσης 10 έχουν το γράμμα e.

Παραδείγματα αποδεκτών αριθμών: +132 -8 126 6.25e-5 0.000008
Παραδείγματα μη αποδεκτών αριθμών: 125,252 (έχει κόμμα), 245I62 (περιέχει το γράμμα I), 234567812,678 (έχει περισσότερα από 7 σημαντικά ψηφία).

<u>Χαρακτήρες (characters)</u>

Ένας οποιοσδήποτε χαρακτήρας από το σύνολο των χαρακτήρων που δέχεται ο υπολογιστής. Κάθε χαρακτήρας πρέπει να γράφεται ανάμεσα σε μονά εισαγωγικά (‘), π.χ. ‘a’, ‘A’, ‘<’, ‘?’ κ.τ.λ.

Η δηλωτική εντολή για τους χαρακτήρες είναι: char.

<u>Συμβολοσειρές (strings)</u>

Η C++ κάνει διάκριση μεταξύ ενός χαρακτήρα και σειράς πολλών χαρακτήρων, την οποία ονομάζουμε συμβολοσειρά. Μια συμβολοσειρά μπορεί να περιέχει κανέναν, έναν η πολλούς χαρακτήρες. Προκειμένου να δηλώσουμε μια συμβολοσειρά, χρησιμοποιούμε τώρα διπλά εισαγωγικά (“), π.χ. “ΙΑΝΟΥΑΡΙΟΣ”, “AB025/XRI-8C” , “3275” κ.τ.λ.

Η δηλωτική εντολή για τις συμβολοσειρές είναι: char όνομα μεταβλητής [αριθμός χαρακτήρων].

<u>Μεταβλητές</u>

Το όνομα μιας μεταβλητής είναι ένας συνδυασμός γραμμάτων (κεφαλαίων ή και πεζών) του αγγλικού αλφάβητου ή ψηφίων (0,1,2,...,9), καθώς και του ειδικού συμβόλου _ (σύμβολο υπογράμμισης) και έχει τους εξής περιορισμούς:

- Πρέπει να αρχίζει με γράμμα ή με το σύμβολο _
- Να μην περιέχει ειδικά σύμβολα (εκτός από το underscore_)
- Να έχει ορισμένο πλήθος (μήκος χαρακτήρων), συνήθως μέχρι 32 χαρακτήρες.
- Δεν επιτρέπεται να περιέχει κανένα γράμμα του ελληνικού αλφάβητου.

Η C++ κάνει διάκριση μεταξύ κεφαλαίων και πεζών γραμμάτων.
Παραδείγματα αποδεκτών μεταβλητών: Aa2 c4ee stx κ.τ.λ.

Παραδείγματα μη αποδεκτών μεταβλητών: x2%3 (το % είναι λάθος σύμβολο)
2xy (αρχίζει με νούμερο) rαx2 (το α είναι ελληνικός χαρακτήρας).

Εκτέλεση αριθμητικών πράξεων

Στη C++ χρησιμοποιούμε τέσσερις βασικές πράξεις για την εκτέλεση των αριθμητικών υπολογισμών. Οι πράξεις αυτές και τα αντίστοιχα σύμβολα είναι:

Πρόσθεση	+
Αφαίρεση	-
Πολλαπλασιασμός	*
Διαίρεση	/

Σειρά εκτέλεσης πράξεων: Σε περίπτωση που η παράσταση δεν έχει παρενθέσεις, τότε η σειρά εκτέλεσης των πράξεων ή ο βαθμός προτεραιότητας, είναι:

Πρώτα οι δυνάμεις (pow(x,y)→ x^y), μετά οι πολλαπλασιασμοί και οι διαιρέσεις και τέλος οι προσθέσεις και οι αφαιρέσεις.
Οι πράξεις με την ίδια προτεραιότητα εκτελούνται από αριστερά προς τα δεξιά.
Αν υπάρχουν παρενθέσεις, θα γίνουν πρώτα οι πράξεις μέσα σε αυτές, με την προτεραιότητα που αναφέραμε, έως ότου δημιουργηθεί παράσταση χωρίς παρενθέσεις. Αν υπάρχουν πολλαπλές παρενθέσεις, τότε οι πράξεις αρχίζουν από τις εσωτερικές παρενθέσεις προς τα έξω.

Παραδείγματα:
1) a+ pow(b,2)-x/y+pow(a,3)
   a+b1-x/y+a1 (b1=pow(b,2), a1=pow(a,3))
   a+b1-d1+a1 (d1=x/y)
   a2-d1+a1 (a2=a+b1)
   a3+a1 (a3=a2-d1)
   a4 (τελικό αποτέλεσμα)

2) a+b*(x+y) – a/(x-y) + pow(b,(-I+K))
   a+b*x1-a/x2+pow(b,j)
   κ.ο.κ.

Τέλος δε μπορούμε να βάλουμε δυο σύμβολα πράξεων το ένα δίπλα στο άλλο. Αν χειάζεται κάτι τέτοιο, τότε θα πρέπει να τα χωρίσουμε με παρενθέσεις, π.χ. x+a/-b είναι λάθος, ενώ x+a/(-b) είναι σωστό.

Σχόλια μέσα στο πρόγραμμα

Τα σχόλια δεν περιέχουν εντολές και αγνοούνται από τον Η/Υ κατά την εκτέλεση. Τα παίρνουμε όμως στην εκτύπωση του προγράμματος και έτσι θυμόμαστε βασικά στοιχεία της δουλειάς μας. Κάθε σχόλιο ορίζεται με τους εξής δύο τρόπους:
/* σχόλιο */ ή
// σχόλιο //

Ας δούμε ένα παράδειγμα για να καταλάβουμε τη δομή ενός προγράμματος στη C++. Το παράδειγμα θα υπολογίζει το εμβαδόν ενός τραπεζίου.

Αρχικά σκεφτόμαστε τον αλγόριθμο του προβλήματος, τα βήματα λύσης δηλαδή.
β1: Υπολογίζουμε το άθροισμα 2 βάσεων (bm θα ορίσουμε το όνομα της μεταβλητής για τη μεγάλη βάση και bs για τη μικρή βάση. x1 θα είναι η μεταβλητή που θα πάρει την τιμή του αθροίσματος των 2 βάσεων).
β2: Πολλαπλασιάζουμε το προηγούμενο άθροισμα με το ύψος (x2 θα είναι η μεταβλητή που θα πάρει την τιμή αυτού του γινομένου, y θα είναι η μεταβλητή για το ύψος).
β3: Διαιρούμε το τελευταίο αποτέλεσμα με το 2 (et θα είναι η μεταβλητή που θα πάρει την τιμή αυτής της διαίρεσης).

```
// Υπολογισμός του εμβαδού τραπεζίου//
# include <stdio.h>
#include <stdlib.h>
int main()
{
   double bm,bs,y,et;
   double x1,x2;
    bm=10.4;
    bs=8.5;
    y=8.0;
    x1=bm+bs;
    x2=x1*y;
    et=x2/2;
    printf("\n area of trapezium=%lf", et);
    system ("pause");
    return 0;
}
```

Η εντολή printf συντάσσεται ως εξής:

printf("φόρμα εκτύπωσης", x1,x2,…,xk)

όπου x1,x2,…,xk είναι μεταβλητές ή σταθερές ή αλγεβρικές παραστάσεις ή συναρτήσεις.
Σημειώνεται ότι στην φόρμα εκτύπωσης γράφουμε το πώς θέλουμε να τυπώσουμε. Επιπλέον δε, ότι έχουμε γράψει στη φόρμα εκτύπωσης, τυπώνεται στην οθόνη μας ως έχει.
π.χ. printf ("\n sxolia=%d ή %f ή %lf ή %c", x1);
όπου το %d δηλώνει ότι θα γίνει εκτύπωση της ακέραιης μεταβλητής x1, το %f δηλώνει ότι θα γίνει εκτύπωση της πραγματικής μεταβλητής χ1, το %lf δηλώνει ότι θα γίνει εκτύπωση της πραγματικής μεταβλητής διπλής ακρίβειας x1 και τέλος το %c δηλώνει ότι θα γίνει εκτύπωση του χαρακτήρα x1.
Τα σχόλια που είναι μέσα στο χώρο " sxolia" θα εκτυπωθούν όλα. Το σύμβολο \n δηλώνει αλλαγή γραμμής.
Σχετικά με την εντολή printf και την αλλαγή γραμμής: Ας πούμε ότι θέλουμε να τυπωθεί
AB
ΓΔ
Τότε γράφουμε printf("\n AB\n ΓΔ");

Αν γράψουμε πολλά συνεχόμενα \n εντός της printf, τότε το πλήθος των κενών γραμμών που θα αφήσει η C++ είναι το πλήθος των \n μείον 1. Για παράδειγμα, η εντολή printf("\n AB \n\n\n ΓΔ") θα τυπώσει

AB
*
*
ΓΔ

Όπου μεταξύ των ΑΒ και ΓΔ υπάρχουν δύο κενές γραμμές (αυτό δηλώνουν οι αστερίσκοι).

Εισαγωγή δεδομένων από το πληκτρολόγιο με την εντολή scanf

Η scanf ορίζεται ως εξής: scanf("%d %f %lf %c", &x1, &x2, &u1, &ch1);
όπου το %d δηλώνει ότι θα γίνει εισαγωγή ακέραιης μεταβλητής και επειδή παραπέμπει στην πρώτη μεταβλητή x1, αυτή θα πρέπει να έχει οριστεί στην αρχή του προγράμματος ως int x1.
Το %f δηλώνει ότι θα γίνει εισαγωγή πραγματικής μεταβλητής και επειδή παραπέμπει στη δεύτερη μεταβλητή x2, αυτή θα πρέπει να έχει οριστεί στην αρχή του προγράμματος ως float x2.
Το %lf δηλώνει ότι θα γίνει εισαγωγή πραγματικής μεταβλητής διπλής ακρίβειας και επειδή παραπέμπει στην τρίτη μεταβλητή u1, αυτή θα πρέπει να έχει οριστεί στην αρχή του προγράμματος ως double u1.
Τέλος το %c δηλώνει ότι θα γίνει εισαγωγή χαρακτήρα και επειδή παραπέμπει στην τελευταία μεταβλητή ch1, αυτή θα πρέπει να έχει οριστεί στην αρχή του προγράμματος ως char ch1.

Να ξαναγράψετε το προηγούμενο πρόγραμμα χρησιμοποιώντας την εντολή εισαγωγής δεδομένων από το πληκτρολόγιο scanf .

```
// Υπολογισμός του εμβαδού τραπεζίου με τη χρήση της scanf//
# include <stdio.h>
#include <stdlib.h>
int main()
{
   double bm,bs,y,et, x1,x2;
   printf("\n Big base=");
   scanf("%lf", &bm);
   printf("\n Small base=");
   scanf("%lf", &bs);
   printf("\n Height=");
   scanf("%lf", &y);
    x1=bm+bs;
    x2=x1*y;
    et=x2/2;
    printf("\n area of trapezium=%lf", et);
    system ("pause");
    return 0;
}
```

Η επαναληπτική εντολή for

Ο βρόχος for, χρησιμοποιείται όταν θέλουμε να έχουμε συγκεκριμένο αριθμό επαναλήψεων της εκτέλεσης μιας ή περισσότερων εντολών. Η γενική μορφή της for θα είναι:

for(<μεταβλητή>=αρχική τιμή; <συνθήκη>; <μεταβολή της τιμής της μεταβλητής>)
{
πεδίο ελέγχου της for
}

π.χ. for (m=0;m<=10;m=m+2)
{
printf("\n m=%d",m);
}

Πολλαπλά for: Μπορούμε να έχουμε for μέσα σε άλλες for, αρκεί οι εξωτερικές να επικαλύπτουν πλήρως τις εσωτερικές, π.χ.

```
for (i=1;i<=n;i++)
  {……;
   ….....;

for (j=0;i<=n;j++)
        {…..;
         …..;
        }
…….;
  }
```

Άσκηση: Να υπολογιστεί και να τυπωθεί το άθροισμα: $\sum_{i=1}^{n} \frac{1}{x^i}$ **, για χ=4**

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int main()
{
        double x,sum;
        int i,n;
        printf("\n metavliti x:");
        scanf("%lf", &x);
  printf("\n metavliti n:");
  scanf("%d", &n);
        sum=0.0;
        for (i=1;i<=n;i++)
        {
                sum=sum+1/pow(x,i);
                printf("\n gia i=%dto sum einai %lf",i,sum);
        }
        system ("pause");
        return 0;
}
```

Οι εντολές if

α) Απλή μορφή

```
if (συνθήκη)
.....;
```

β) Σύνθετη μορφή

```
if (συνθήκη)
{
    .....;
    .....;          Περιοχή ελέγχου της if (ή το σώμα της if)
    .....;
}
```

Αν η περιοχή ελέγχου της if είναι μια μόνο εντολή, τότε οι αγκύλες { και } δε χρειάζονται.

Η εντολή "if – else"

Οι εντολές του τύπου αυτού είναι πολύ χρήσιμες στον προγραμματισμό στη C++, αλλά και σε κάθε γλώσσα προγραμματισμού. Και αυτή η εντολή έχει δυο μορφές, την απλή και τη σύνθετη.

α) απλή μορφή

```
if (συνθήκη)
.....;   (1)
else
.....;   (2)
.....;   (3)
```

β) σύνθετη μορφή

```
if (συνθήκη)
{
    .....;
    .....;   (1)
    .....;

else
    .....;
    .....;   (2)
    .....;
}
.....;       (3)
```

Ο αλγόριθμος εκτέλεσης της if – else

β-1: Ελέγχεται η συνθήκη:

1. Αν είναι αληθής, τότε: εκτελούνται οι εντολές (1) και πηγαίνουμε στο βήμα-2.
2. Αν δεν είναι αληθής, τότε: εκτελούνται οι εντολές (2) και συνεχίζουμε με το βήμα – 2.

β-2: Συνεχίζουμε με την εντολή (3).

Σε ποια if αντιστοιχεί η else

Έχουμε περιπτώσεις όπου η if και else μπορούν να εμφανιστούν σε μια σύνθετη μορφή, όπως η παρακάτω, να έχουμε δηλαδή δυο if και μια else

```
if (συνθήκη-1)  (1)
if (συνθήκη-2)  (2)
.............;
else
.............;
```

Το ερώτημα είναι σε ποια if αντιστοιχεί η else. Το else πηγαίνει με την τελευταία if, η οποία δεν έχει δικό της else. Στο παραπάνω παράδειγμα η else πηγαίνει με την if (2). Αν όμως θέλαμε να πάει με την if (1), τότε θα γράφαμε τις ίδιες εντολές ως εξής:

```
if (συνθήκη-1)
{
        if (συνθήκη-2)
        ……………;
}
else
.............;
```

Οι δυο αγκύλες { και }, κάνουν τη δεύτερη if μη ορατή στο else. Το else λοιπόν θα συνδέεται τώρα με την πρώτη if.

.

Έλεγχος των συνθηκών

Μια συνθήκη μπορεί να είναι πολύπλοκη και να εμπλέκονται σ' αυτήν μαθηματικές πράξεις, τελεστές σύγκρισης, καθώς και λογικοί τελεστές. Είναι λοιπόν απαραίτητο να γνωρίζουμε τον τρόπο με τον οποίο ενεργεί η C++ προκειμένου να αξιολογηθεί μια συνθήκη. Η σειρά που ακολουθείται για την αξιολόγηση μιας συνθήκης είναι η εξής:

Πρώτα οι πράξεις
Μετά οι τελεστές σύγκρισης
Μετά οι Λογικοί τελεστές (αριστερά $\rightarrow$ δεξιά)

Τελεστές σύγκρισης

= =	Ίσον	(a= =b)
<	Μικρότερο	(a<b)
>	Μεγαλύτερο	(a>b)
<=	Μικρότερο ή ίσον	(a<=b)
>=	Μεγαλύτερο ή ίσον	(a>=b)
!=	Διάφορο	(a!=b)

Λογικοί τελεστές

α) Ο λογικός τελεστής 'και' (AND). Το σύμβολό του είναι &&

π.χ.
```
if(x>2 && x<5)
        y=100;
        else
        y=0;
```

β) Ο λογικός τελεστής ‘ή’ (OR). Το σύμβολό του είναι: ||

π.χ.
```
if (a<y || b+c/(d-a)>0)
    printf(……);
```

Η printf θα εκτελεστεί αν μια τουλάχιστον από τις παραπάνω συγκρίσεις αληθεύει.

γ) Ο λογικός τελεστής ‘ΟΧΙ’ (NOT). Συμβολίζεται με το: !

π.χ.
```
if(!(x<5))
    a=b;
```

Μέσα στην ίδια if μπορούμε να κάνουμε πολλούς και πολύπλοκους συνδυασμούς με το AND, το OR και το NOT. Καλό είναι όμως, για την αποφυγή λαθών να χρησιμοποιούμε όσο μπορούμε πιο απλές μορφές.

Σε περίπτωση που υπάρχουν πολλά NOT, AND και OR σε μια λογική if, τότε η σειρά εκτέλεσης είναι:

1α Τα NOT
2α Τα AND
3α Τα OR

```
if (a>b || c<d && r>0)…….;
        ……
        (1)
    …….
    (2)
```

Χρησιμοποιείστε παρενθέσεις αν δεν είστε σίγουροι:

```
if (a>b|| (c<d && r>0))…….;
```

Η εκτέλεση γίνεται από τις εσωτερικές παρενθέσεις προς τα έξω.

<u>Σύνθετες if</u>

Η C++ μας επιτρέπει να χρησιμοποιούμε πλέον σύνθετες if, όπου ένα else να ακολουθείται από νέα if, κ.ο.κ.

```
if (σ1)
.....; (1)
else if (σ2)
.....; (2)
else if (σ3)
.....; (3)
else if (σ4)
.....; (4)
else if (σ5)
.....; (5)
.....; (6)
```

<u>Αλγόριθμος εκτέλεσης της παραπάνω if</u>

- Αν η σ1 αληθεύει, τότε: εκτελούνται οι εντολές (1) και μετά πηγαίνουμε στην (6). Αν η σ1 δεν αληθεύει, πηγαίνουμε στην επόμενη else if.
- Αν η σ2 αληθεύει, τότε: εκτελούνται οι εντολές (2) και μετά πηγαίνουμε στην (6). Αν η σ2 δεν αληθεύει, πηγαίνουμε στην επόμενη else if.

Η παραπάνω διαδικασία επαναλαμβάνεται και για τις επόμενες else if.

Ας δούμε ένα παράδειγμα:

Η μεταβλητή Κ παίρνει τις τιμές 1, 2 και 3, ενώ οι μεταβλητές Α, Β, παίρνουν τιμές που εξαρτώνται από την τιμή του Κ.

αν Κ=1 τότε Α=2.0 και Β=4.0
αν Κ=2 τότε Α=3.0 και Β=5.0
αν Κ=13 τότε Α=-10.0 και Β=-20.0
ενώ αν το Κ είναι διάφορο του 1, 2 ή 3, τότε Α=0.0 και Β=0.0

Γράφουμε λοιπόν:

```
(a) if(k==1)
    {a=2;
     b=4;   (1)
    }
(b) else if(k==2)
    {a=3;
     b=45;   (2)
    }
(c) else if(k==3)
    {a=-10;
     b=-20;   (3)
    }
(d) else
    {a=0;
     b=0;   (4)
    }
(e) ……..;
```

Δηλαδή: Στην (a) εξετάζουμε αν το Κ=1. Αν ναι, τότε εκτελούνται οι εντολές (1) και μετά πάμε στην (e). Αν, όμως, το Κ δεν είναι ίσο με το 1, τότε πάμε στην (b) (χωρίς να εκτελέσουμε τις (1)) και εξετάζουμε αν Κ=2. Αν ναι, τότε εκτελούνται οι (2) και μετά πάμε στην (e). Αν, βέβαια, το Κ δεν είναι ίσο ούτε με το 2, τότε πάμε στην (c) (χωρίς να εκτελέσουμε τις (2)) και εξετάζουμε αν Κ=3. Αν ναι, τότε εκτελούνται οι εντολές (3) και μετά πάμε στην (e). Αν πάλι το Κ δεν είναι ίσο με το 3, δηλαδή δεν είναι ούτε 1, ούτε 2, ούτε 3, τότε θα εκτελεστούν οι εντολές (4) και μετά θα πάμε πάλι στην (e).

Σε όλες τις παραπάνω περιπτώσεις η C++ συνεχίζει με την εντολή (e).

Άσκηση

Να γράψετε ένα πρόγραμμα στη C++ που να υπολογίζει τις ρίζες του τριωνύμου: x^2 -3x + 2, χρησιμοποιώντας τη λογική εντολή if.

Λύση

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int main()
{
	double a, b, c, D, r1, r2, repart, impart;
// a, b, c einai oi syntelestes tou trionimou ax^2+bx+c, D einai h orizousa, r1 kai r2 oi rizes, repart to pragmatiko meros kai impart to fantastiko meros
	printf("\n suntelestis a:");
	scanf("%lf", &a);
	printf("\n suntelestis b:");
	scanf("%lf", &b);
	printf("\n suntelestis c:");
	scanf("%lf", &c);
	D=b*b-4.0*a*c;
if(D>0.0)
{
	r1=(-b+sqrt(D))/(2*a);
	r2=(-b-sqrt(D))/(2*a);
	printf(“\n r1=%lf, r2=%lf”, r1, r2);
}
else if (D==0.0)
{
	r1=r2=-b/(2*a);
	printf(“\n r1=%lf, r2=%lf”, r1, r2);
}
else
{
	repart=-b/(2*a);
	impart=sqrt(-D)/(2*a);
	printf(“\n r1=%lf + i %lf”, repart, impart);
	printf(“\n r2=%lf - i %lf”, repart, impart);
}
	system (“pause”);
	return 0;
}
```

Πίνακες

Εισαγωγή

Όπως σε κάθε γλώσσα προγραμματισμού, έτσι και στην C++ η χρήση πινάκων απαιτεί τη δήλωσή τους στην αρχή του προγράμματος και πριν από οποιαδήποτε χρήση τους. Ένας πίνακας δεν είναι τίποτα άλλο παρά μια συλλογή αριθμών με κοινό όνομα. Οι πίνακες μπορεί να είναι μιας διαστάσεως (διανύσματα), δύο ή περισσότερων διαστάσεων.

Ορισμός πινάκων

Προκειμένου να δηλώσουμε έναν πίνακα, γράφουμε στην αρχή του προγράμματος το όνομά του και το πλήθος των αριθμών (δεδομένων) που μπορεί να έχει. Ακόμη, καθορίζουμε αν οι αριθμοί που θα περιέχει θα είναι ακέραιοι, πραγματικοί κ.τ.λ.
Π.χ.

```
int main()
{
int a[10];
float b[20], c[10][20];
.....;
.....;
}
```

Ο πίνακας a είναι μιας διαστάσεως 10 θέσεων και μπορεί να πάρει μέχρι 10 ακέραιους αριθμούς. Ο b είναι πραγματικός 20 θέσεων, ενώ ο c είναι πραγματικός δύο διαστάσεων και μπορεί να έχει μέχρι $10 \times 20 = 200$ αριθμούς.

Καθορισμός στοιχείων πίνακα

Κάθε στοιχείο πίνακα καθορίζεται από τη θέση που κατέχει μέσα στον πίνακα. Έτσι, για πίνακες μιας διαστάσεως, το πρώτο στοιχείο καθορίζεται από τη θέση 0 (δείκτης μηδέν), το δεύτερο από τη θέση 1 και το τελευταίο από τη θέση n-1.
Π.χ.
a[0] καθορίζει το πρώτο στοιχείο του πίνακα a.
a[9] καθορίζει το $10^{ο}$ στοιχείο του πίνακα a.

Για πίνακες δύο διαστάσεων χρησιμοποιούμε δύο δείκτες. Π.χ.
c[0][0] καθορίζει το στοιχείο του πίνακα c που βρίσκεται στη θέση (1,1) του πίνακα (c(1,1)).
c[9][19] καθορίζει το στοιχείο του πίνακα c που βρίσκεται στη θέση (10,20) του πίνακα (c(10,20)).

Για μονοδιάστατους πίνακες έχουμε:
Κάθε στοιχείο του πίνακα, καθορίζεται από το δείκτη i ο οποίος παίρνει τιμές από 0 έως N-1.
A[i] $0 \le i \le N-1$

Για πίνακες δύο διαστάσεων έχουμε:

Ο πίνακας Α έχει $N \times M$ στοιχεία (αριθμούς). Η θέση κάθε αριθμού μέσα στον πίνακα Α καθορίζεται από δύο δείκτες (i,j).

Ο πρώτος δείκτης i φανερώνει τον αριθμό της γραμμής.

Ο δεύτερος δείκτης j φανερώνει τον αριθμό της στήλης.

A(i,j) $\quad 0 \le i \le N-1 \quad$ και $0 \le j \le M-1$

Αντίστοιχα έχουμε για πίνακες περισσοτέρων διαστάσεων.

<u>Εισαγωγή τιμών σε πίνακα</u>

Μπορούμε να δώσουμε τιμές σε έναν πίνακα με δύο τρόπους:

α) Μέσα στο πρόγραμμα

Να δώσουμε τιμές σε διάφορα στοιχεία του πίνακα ή σε όλο τον πίνακα.

```
int main( )
 {double x1[10], c[5][10];
   int i;
   …….;
    x1[0]=200;
    x1[9]=-152
    c[0][1]=14.73;
    ……..;
    }
```

β) Με ανάγνωση αριθμών από το πληκτρολόγιο

Χρησιμοποιούμε την scanf για να δώσουμε τιμές ατομικά σε διάφορα στοιχεία του πίνακα ή σε συνδυασμό με τη for για μαζική εισαγωγή δεδομένων.

```
int main( )
 {double x1[10], a[10][20];
   int i;
   …….;
   scanf(“%lf %lf”, &a[1][2], &x1[0]);
   …….;
   for (i=0;i<10;i++)
   for (j=0;j<20;j++)
   scanf(“%lf”, &a[i][j]);
   ……..;
   ……..;
   }
```

<u>Χρήση τιμών πίνακα</u>

Κάθε αριθμός που υπάρχει σε έναν πίνακα μπορεί να χρησιμοποιηθεί σε πράξεις, συγκρίσεις κ.τ.λ., αρκεί να καθορίσουμε επακριβώς τη θέση του μέσα στον πίνακα.

```
int main( )
 {
   int a[10], i, j;
   …….;
   for (i=0;i<10;i++)
```

```
    a[i]=i+1;
 ……..;
 b=10;
 for (i=0;i<10;i++)
 { j=i+b;
   a[i]=a[i]/j;
 }
  ……..;
  ……..;
}
```

Στο παραπάνω πρόγραμμα, καθορίσαμε έναν πίνακα a, 10 θέσεων. Μετά του δώσαμε τιμές 1, 2, 3, 4, 5, ..., 10. Τέλος διαιρέσαμε κάθε στοιχείο του με 10,11, 12, ..., 19.

<u>Εκτύπωση πίνακα</u>

Η εκτύπωση επιλεγμένων στοιχείων πίνακα ή μέρους ή όλου του πίνακα μπορεί να γίνει με προσεκτική χρήση των αντίστοιχων δεικτών που καθορίζουν τις αντίστοιχες θέσεις των στοιχείων του πίνακα.
Π.χ.

```
int main()
 {
   double a[10];
   int i, n=10;
   ……;
   for (i=1;i<=n;i++)
   printf("\n %lf", a[i]);
   …….;
  }
```

Στην περίπτωση αυτή, κάθε στοιχείο του πίνακα θα τυπωθεί σε νέα γραμμή. Αυτό, γίνεται για τον απλούστατο λόγο, ότι μέσα στην printf υπάρχει ο χαρακτήρας διαφυγής \n. Αν, βέβαια, θέλουμε να έχουμε μια διαφορετική εκτύπωση, τότε θα πρέπει να σκεφτόμαστε κάθε φορά και τις αντίστοιχες τεχνικές προγραμματισμού. Για την εκτύπωση πινάκων με δύο ή περισσότερες διαστάσεις, ισχύει ότι και στους μονοδιάστατους πίνακες.

<u>Άσκηση</u>

Να γραφτεί πρόγραμμα το οποίο να διαβάζει τις τιμές ενός πίνακα ($n \times m$). Να τυπώνει τον πίνακα καθώς και το άθροισμα των στοιχείων κάθε γραμμής του και κάθε στήλης του.

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
   double  a[100][101], sumrow, sumcolumn;
```

```
    int i, j, nr, nc;
    sumrow=0.0;
    sumcolumn=0.0;
    printf("dwse arithmo grammwm");
    scanf("%d", &nr);
   printf("dwse arithmo sthlwn");
    scanf("%d", &nc);
    printf("\n Dwse ta dedomena tou pinaka kata grammes:");
    for (i=1; i<=nr; i++)
     {
       for (j=1; j<=nc; j++)
         {
          printf("a[%d][%d]=", i, j);
          scanf ("%lf", &a[i][j]);
          }
      }
    printf("\n Ektupwsh arxikou pinaka:");
    for (i=1; i<=nr; i++)
     {
       for (j=1; j<=nc; j++)
         {
          printf("%lf     ", a[i][j]);

          }
      }
    printf("\n Athroismata grammwn:");
    for (i=1; i<=nr; i++)
     {
       for (j=1; j<=nc; j++)
         {
          sumrow=sumrow+a[i][j];
          }
        printf("\n athroisma stoixeiwn grammhs %d = %lf", i, sumrow);
          sumrow=0.0;
      }
    printf("\n Athroismata sthlwn:");
    for (j=1; j<=nc; j++)
     {
       for (i=1; i<=nr; i++)
         {
          sumcolumn=sumcolumn+a[i][j];
          }
        printf("\n athroisma stoixeiwn sthlhs %d = %lf", j, sumcolumn);
         sumcolumn=0.0;
      }
  system ("pause");
  return 0;
}
```