ΥΠΟΛΟΓΙΣΤΕΣ ΙΙ

ΤΥΠΟΙ ΔΕΔΟΜΕΝΩΝ
ΠΡΑΞΕΙΣ
ΜΕΤΑΒΛΗΤΕΣ

Τύποι δεδομένων

Η γλώσσα προγραμματισμού C++ υποστηρίζει τους παρακάτω τύπους δεδομένων:

1) Ακέραιοι αριθμοί (int).
2) Πραγματικοί αριθμοί διπλής ακρίβειας (double).
3) Πραγματικοί αριθμοί απλής ακρίβειας (float).
4) Χαρακτήρες (char).

Ακέραιοι αριθμοί (int)

- Πρόκειται για αριθμούς χωρίς δεκαδικά ψηφία.
- Για την αναπαράσταση των ακέραιων αριθμών χρησιμοποιούνται 4 bytes μνήμης.
- Λαμβάνουν τιμές στην περιοχή -2147483648 έως 2147483647.

Πράξεις μεταξύ ακεραίων αριθμών

Μεταξύ δύο ακεραίων αριθμών μπορούν να γίνουν οι παρακάτω πράξεις χρησιμοποιώντας τον αντίστοιχο τελεστή:

`+`	Πρόσθεση
`-`	Αφαίρεση
`*`	Πολλαπλασιασμός
`/`	Διαίρεση
`%`	Υπόλοιπο διαίρεσης

Το αποτέλεσμα μιας πράξης μεταξύ ακεραίων αριθμών είναι **πάντα** ακέραιος αριθμός.

Προσοχή: Δεν υπάρχει τελεστής για ύψωση σε δύναμη.

Ακέραια διαίρεση

Η διαίρεση δύο ακεραίων αριθμών δίνει ως αποτέλεσμα επίσης ένα ακέραιο αριθμό.

Κατά τη διαίρεση δύο ακεραίων αριθμών τυχόν δεκαδικά ψηφία που προκύπτουν κατά τη διαίρεση **αποκόπτονται**.

Παραδείγματα:

`9/2`	Αποτέλεσμα: `4` (όχι `4.5`)
`1/3`	Αποτέλεσμα: `0` (όχι `0.33333…`)
`2/3`	Αποτέλεσμα: `0` (όχι `0.66666…`)
`-6/4`	Αποτέλεσμα: `-1` (όχι `–1.5`)

Η λειτουργία αυτή ονομάζεται **ακέραια διαίρεση**.

Παράδειγμα #1

`3+5`	Αποτέλεσμα: `8`
`4-8`	Αποτέλεσμα: `-4`
`6*2`	Αποτέλεσμα: `12`
`8/3`	Αποτέλεσμα: `2`
`7%3`	Αποτέλεσμα: `1`
`12%5`	Αποτέλεσμα: `2`

Σύνθετες αριθμητικές παραστάσεις

Για να υπολογιστούν πιο σύνθετες αριθμητικές παραστάσεις όπως πχ.

2+4*3/2-7

6+(4/2-8)%2

έχει ανατεθεί σε κάθε τελεστή **προτεραιότητα** και **προσεταιριστικότητα**.

Τελεστής	Προτεραιότητα	Προσεταιριστικότητα
`* / %`	Υψηλή	Από αριστερά προς δεξιά
`+ -`	Χαμηλή	Από αριστερά προς δεξιά

Κανόνες υπολογισμού σύνθετων παραστάσεων

Σε μια σύνθετη αριθμητική παράσταση:

1. Πρώτα γίνονται οι πράξεις με τη μεγαλύτερη προτεραιότητα.
2. Μεταξύ πράξεων με την ίδια προτεραιότητα η σειρά των πράξεων καθορίζεται από την προσεταιριστικότητα.
3. Εάν υπάρχουν παρενθέσεις, τότε πρώτα γίνονται οι πράξεις εντός του πιο εσωτερικού ζεύγους παρενθέσεων.

Πραγματικοί αριθμοί διπλής ακρίβειας (double)

- Πρόκειται για αριθμούς με δεκαδικά ψηφία
- Απαιτούνται 8 bytes για την αποθήκευσή τους.
- Έχουν περίπου 15 σημαντικά ψηφία.
- Λαμβάνουν τιμές στην περιοχή -10^{308}...-10^{-308} 10^{-308}...10^{308}
- Στη C++ οποιαδήποτε αριθμητική σταθερά με δεκαδικά ψηφία θεωρείται πραγματικός αριθμός διπλής ακρίβειας.

Πραγματικοί αριθμοί: Επιστημονική αναπαράσταση

Αριθμός	Επιστ. αναπαράσταση	Σημαίνει
`32.76`	`3.276e1`	3.276×10^{1}
`-98541.34`	`-9.854134e4`	-9.854134×10^{4}
`0.000035`	`3.5e-5`	3.5×10^{-5}

Πραγματικοί αριθμοί απλής ακρίβειας (float)

- Απαιτούνται 4 bytes για την αποθήκευσή τους.
- Έχουν περίπου 7 σημαντικά ψηφία.
- Λαμβάνουν τιμές στην περιοχή -10^{38}...-10^{-38} 10^{-38}...10^{38}
- Για να ξεχωρίσουν από τους πραγματικούς διπλής ακρίβειας γράφονται με το επίθεμα f

Παράδειγμα #2

3.14	Διπλής ακρίβειας (double)
-1.2e4	Διπλής ακρίβειας (double)
67.51f	Απλής ακρίβειας (float)
2.2e-3f	Απλής ακρίβειας (float)

Πράξεις μεταξύ πραγματικών αριθμών

Μεταξύ πραγματικών αριθμών γίνονται οι πράξεις:

`+  -  *  /`

Η διαίρεση γίνεται κατά το συνήθη τρόπο, δηλαδή αν προκύψουν δεκαδικά ψηφία, αυτά παραμένουν στον αριθμό.

Σημείωση: Κάποιοι μεταφραστές επιτρέπουν και το υπόλοιπο διαίρεσης % ως πράξη, χωρίς αυτό όμως να περιλαμβάνεται στο πρότυπο της γλώσσας.

Μικτή αριθμητική

Όταν σε μια πράξη συμμετέχουν αριθμοί από δύο διαφορετικούς τύπους δεδομένων, τότε:

τα δεδομένα **προάγονται** αυτόματα στον ανώτερο από τους δύο τύπους

και κατά συνέπεια το αποτέλεσμα θα είναι του ανώτερου τύπου.

Οι τύποι δεδομένων ιεραρχούνται ως εξής:

double	Ανώτερος
float	
int	Κατώτερος

Συναρτήσεις

Η γλώσσα προγραμματισμού C++ περιέχει μια σειρά από ενσωματωμένες μαθηματικές συναρτήσεις.

Για να χρησιμοποιήσουμε οποιαδήποτε μαθηματική συνάρτηση πρέπει να προσθέσουμε στην αρχή του προγράμματος:

```
#include <cmath>
```


Συναρτήσεις

Για να υπολογίσουμε		Γράφουμε
$\sqrt{x}$	Τετραγωνική ρίζα	`sqrt(x)`
e^x	Εκθετικό	`exp(x)`
$\|x\|$	Απόλυτη τιμή	`abs(x)`
lnx	Νεπέρειος λογάριθμος	`log(x)`
logx	Λογάριθμος με βάση 10	`log10(x)`
sinx	Ημίτονο	`sin(x)`
cosx	Συνημίτονο	`cos(x)`
tanx	Εφαπτομένη	`tan(x)`
$\sin^{-1}x$	Τόξο ημιτόνου	`asin(x)`
$\cos^{-1}x$	Τόξο συνημιτόνου	`acos(x)`
$\tan^{-1}x$	Τόξο εφαπτομένης	`atan(x)`

Συναρτήσεις

Όλες οι συναρτήσεις δέχονται ως όρισμα αριθμούς float ή double και επιστρέφουν αποτέλεσμα του αντίστοιχου τύπου. Πχ.

```
sin(1.0)        Σωστό
sin(1)          Λάθος
```

Οι τριγωνομετρικές συναρτήσεις δέχονται το όρισμα σε **ακτίνια**, όχι μοίρες.

Οι αντίστροφες τριγωνομετρικές συναρτήσεις επιστρέφουν αποτέλεσμα σε **ακτίνια**, όχι μοίρες.

Οι αριθμοί π, e

Για να υπολογίσουμε τον αριθμό π γράφουμε:

```
acos(-1.0)
```

Για να υπολογίσουμε τον αριθμό e (βάση των φυσικών λογαρίθμων) γράφουμε:

```
exp(1.0)
```


Ύψωση σε δύναμη

Η ύψωση σε δύναμη a^b γίνεται με τη συνάρτηση pow(a,b)

Η βάση (a) και ο εκθέτης (b) μπορεί να είναι:

Τύπος βάσης	Τύπος εκθέτη	Τύπος αποτελέσματος
float	float	float
double	double	double
float	int	float
double	int	double

Προσοχή: Η βάση δεν μπορεί να είναι ακέραιος αριθμός.
Πχ. pow(4,2) είναι ΛΑΘΟΣ

Παράδειγμα #3

Όγκος σφαίρας με ακτίνα r: $\frac{4}{3}\pi r^3$

```
4*acos(-1.0)*pow(r,3)/3
```

Απόσταση μεταξύ των σημείων (x_1,y_1) και (x_2,y_2):

$$\sqrt{(x_1-x_2)^2+(y_1-y_2)^2}$$

```
sqrt( pow(x1-x2,2)+pow(y1-y2,2) )
```

Χαρακτήρες (char)

- Καταλαμβάνουν 1 byte μνήμης.
- Υπάρχουν 256 διαφορετικοί χαρακτήρες.
- Ορισμένοι χαρακτήρες δεν εμφανίζονται στην οθόνη (πχ. esc, backspace)
- Κάθε χαρακτήρας αντιστοιχεί σε ένα ακέραιο αριθμό.
- Οι χαρακτήρες a..z, A…Z και 0..9 βρίσκονται στη σειρά, δηλαδή αντιστοιχούν σε διαδοχικούς ακέραιους αριθμούς.
- Γράφονται μέσα σε απλές αποστρόφους. Πχ. 'a' '8' 'E'
- Ορισμένοι ειδικοί χαρακτήρες γράφονται χρησιμοποιώντας το σύμβολο \ και ονομάζονται χαρακτήρες διαφυγής. Πχ. '\n'

Χαρακτήρες διαφυγής

Χαρακτήρας	Τι σημαίνει
\n	Αλλαγή γραμμής (new line)
\b	Μία θέση πίσω (backspace)
\f	Αλλαγή σελίδας (form feed)
\r	Αρχή γραμμής (carriage return)
\t	Στηλοθέτης (tab)
\\	Ο χαρακτήρας \
\'	Απλή απόστροφος
\"	Διπλή απόστροφος
\0	Χαρακτήρας που αντιστοιχεί στον ακέραιο 0

Παράδειγμα #4

Τι θα εμφανιστεί στην οθόνη από τις παρακάτω εντολές;

```
cout << "HELLO \"WORLD\"\n";
```

```
HELLO "WORLD"
```

```
cout << "HELLO \bWORLD\n";
```

```
HELLOWORLD
```

```
cout << "HELLO\b\b\b\b\bWORLD\n";
```

```
WORLD
```

```
cout << "HELLO\rWORLD\n";
```

```
WORLD
```


Μεταβλητές

Οι μεταβλητές είναι συμβολικά ονόματα που δίνουμε σε θέσεις μνήμης όπου αποθηκεύονται αριθμοί (ή άλλοι τύποι δεδομένων).

Κάθε μεταβλητή έχει **όνομα** και **τύπο**, που δηλώνονται υποχρεωτικά στην αρχή του προγράμματος.

Κανόνες ονομάτων

- Επιτρέπονται μόνο λατινικοί χαρακτήρες, αριθμοί και το σύμβολο _
- Ο πρώτος χαρακτήρας πρέπει να είναι γράμμα.
- Κεφαλαία και μικρά θεωρούνται διαφορετικά.
- Δεν υπάρχει μέγιστο μήκος στο όνομα, όμως πρακτικά κάθε μεταφραστής επιβάλλει ένα μέγιστο μήκος (πχ 32 χαρακτήρες).

Οι ίδιοι κανόνες ισχύουν και για άλλα ονόματα, πχ. ονόματα υποπρογραμμάτων.

Δηλώσεις μεταβλητών

Οι μεταβλητές δηλώνονται στην αρχή κάθε προγράμματος:

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

int main ( )
{
      int k, m;
      float a;
      double x, y, z;

      … εντολές …
}
```


Ανάθεση τιμών

Για να δώσουμε τιμή σε μια μεταβλητή χρησιμοποιούμε τον **τελεστή** ανάθεσης τιμής =

Μεταβλητή = *Αριθμητική παράσταση*

Αριστερά του = είναι πάντα μια **μεταβλητή**
Δεξιά του = είναι πάντα μια **αριθμητική παράσταση**

Παραδείγματα:

```
r = 5.
d = b*b-4*a*c
area = acos(-1.0)*r*r
x1 = (-b+sqrt(d))/(2*a)
```


Ανάθεση τιμών στη δήλωση

Μπορούμε να δώσουμε αρχική τιμή σε μια μεταβλητή μέσω της δήλωσής της. Πχ.

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

int main ( )
{
      int k=0;
      double x=1.45;

      … εντολές …
}
```

Στα δεξιά του = πρέπει να είναι μια σταθερή τιμή.

Είσοδος - έξοδος μεταβλητών

Οι τιμές των μεταβλητών μπορούν να εμφανιστούν στην οθόνη με την εντολή cout. Πχ.

```
cout << x ;
```

Μηνύματα και μεταβλητές μπορεί να εμφανιστούν από την ίδια εντολή cout:

```
cout << "Το αποτέλεσμα είναι: " << x;
```


Είσοδος - έξοδος μεταβλητών

Εάν θέλουμε να αλλάξουμε γραμμή θα πρέπει να χρησιμοποιήσουμε το χαρακτήρα διαφυγής \n

```
cout << x << '\n';
```

Ο χαρακτήρας `'\n'` έχει το ειδικό όνομα endl που μπορούμε να χρησιμοποιήσουμε:

```
cout << x << endl;
```

Εάν θέλουμε να εμφανίσουμε δύο μεταβλητές στην ίδια γραμμή θα πρέπει να αφήσουμε κενό ανάμεσα τους:

```
cout << x << "  " << y << endl;
```


Είσοδος - έξοδος μεταβλητών

Για να εισάγουμε μεταβλητές από το πληκτρολόγιο χρησιμοποιούμε την εντολή cin:

```
cin >> x;
```

ή για περισσότερες μεταβλητές:

```
cin >> x >> y >> z;
```

Η εντολή cin χρησιμοποιείται συνήθως μαζί με μια εντολή cout:

```
cout << "Εισάγετε τις συντεταγμένες ";
cin >> x >> y >> z;
```


Παράδειγμα #5

Γράψτε πλήρες πρόγραμμα που θα υπολογίζει τις παραστάσεις:

$$a = \frac{1+\sqrt{\pi}}{1-\sqrt{\pi}}$$

$$b = \frac{e^2 - 1}{\sqrt[3]{e}}$$

$$c = \frac{\sin 80^0}{\cos 20^0}$$

Παράδειγμα #5

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

int main ( )
{
     double pi, r80, r20, a, b, c;

     pi = acos(-1.0);              // Ο αριθμός π
     a = (1.0+sqrt(pi)) / (1.0-sqrt(pi));

     b = (exp(2.0)-1.0) / exp(1.0/3.0);

     r80 = 80*pi/180;              // Μετατροπή 80⁰ σε rad
     r20 = 20*pi/180;              // Μετατροπή 20⁰ σε rad
     c = sin(r80)/cos(r20);

     cout << a << endl;
     cout << b << endl;
     cout << c << endl;
}
```


Τελεστές αύξησης και μείωσης

Για την ειδική περίπτωση της πράξης μεταξύ ακεραίων:

```
k = k+1;
```

η C++ έχει ένα ειδικό τελεστή. Μπορούμε να γράψουμε:

```
++k;
```
(προθεματική μορφή)

ή

```
k++;
```
(επιθεματική μορφή)

Τελεστές αύξησης και μείωσης

Στην προθεματική μορφή:
Πρώτα αυξάνεται η τιμή της μεταβλητής και κατόπιν η τιμή χρησιμοποιείται.

Στην επιθεματική μορφή:
Πρώτα χρησιμοποιείται η τιμή της μεταβλητής και κατόπιν η τιμή της μεταβλητής αυξάνεται.

Παράδειγμα:

```
k = 5;
a = ++k;
```

Η k γίνεται 6
Η a γίνεται 6

```
k = 5;
a = k++;
```

Η k γίνεται 6
Η a γίνεται 5

Τελεστές αύξησης και μείωσης

- Αντίστοιχα με τον τελεστή αύξησης ++ υπάρχει ο τελεστής μείωσης -- σε προθεματική και επιθεματική μορφή.
- Οι τελεστές ++ και -- χρησιμοποιούνται μόνο σε ακέραιους αριθμούς.
- Εντός μιας αριθμητικής παράστασης οι τελεστές ++ και -- έχουν υψηλότερη προτεραιότητα από τους τελεστές `+ - * / %`
- Η προσεταιριστικότητα των τελεστών ++ και -- είναι από δεξιά προς τα αριστερά.

Παράδειγμα #6

Ποια είναι η τιμή των μεταβλητών n, j μετά από τις εντολές:

```
n = 10;
j = ++n % 4;
```

Η n γίνεται 11
Η j γίνεται 3

Παράδειγμα #7

Ποια είναι η τιμή των μεταβλητών m, k μετά από τις εντολές:

```
m = 4;
k = 2+ m++;
```

Η m γίνεται 5
Η k γίνεται 6

Άλλοι τελεστές ανάθεσης τιμής

Η ανάθεση τιμής: `k = k+m`

μπορεί να γραφεί στη C++ ως: `k += m`

Όμοια υπάρχουν και οι τελεστές:

Τελεστής	Σημαίνει
`k -= m`	`k = k-m`
`k *= m`	`k = k*m`
`k /= m`	`k = k/m`
`k %= m`	`k = k%m` (μόνο για ακεραίους)

Προτεραιότητες

Τελεστής	Προτεραιότητα	Προσεταιριστικότητα
`++ --`	Υψηλή	Από δεξιά προς αριστερά
`* / %`		Από αριστερά προς δεξιά
`+ -`		Από αριστερά προς δεξιά
`= += -= *= /= %=`	Χαμηλή	Από δεξιά προς αριστερά

Παράδειγμα #8

Γράψτε πρόγραμμα το οποίο θα υπολογίζει την απόσταση δύο σημείων (x_1,y_1) και (x_2,y_2) στο επίπεδο.

Απόσταση μεταξύ των σημείων (x_1,y_1) και (x_2,y_2):

$$\sqrt{(x_1-x_2)^2+(y_1-y_2)^2}$$

Παράδειγμα #9

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

int main ( )
{
      double x1, y1, x2, y2;
      double d;

      cout << "Εισάγετε το πρώτο σημείο ";
      cin >> x1 >> y1;
      cout << "Εισάγετε το δεύτερο σημείο ";
      cin >> x2 >> y2;

      d = sqrt( pow(x2-x1,2)+pow(y2-y1,2) );
      cout << "Η απόσταση είναι " << d << endl;
}
```


Παράδειγμα #9

Γράψτε ένα τμήμα προγράμματος που θα αντιμεταθέτει τις τιμές δύο μεταβλητών.

Παράδειγμα:
Αν αρχικά
a=1 και b=4
μετά την εκτέλεση των εντολών θα πρέπει να είναι
a=4 και b=1

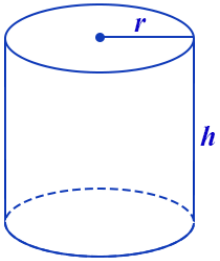
Παράδειγμα #9

Χρησιμοποιούμε μια βοηθητική μεταβλητή t

```
t = a;
a = b;
b = t;
```

Παράδειγμα #10

Κατασκευάστε πρόγραμμα το οποίο θα υπολογίζει τη συνολική επιφάνεια και τον όγκο κυλίνδρου όταν δίνεται η ακτίνα βάσης r και το ύψος h.



Υπενθύμιση:

Επιφάνεια: $S = 2\pi rh + 2\pi r^2$

Όγκος: $V = \pi r^2 h$

Παράδειγμα #10

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

int main ( )
{
      double r, h;
      double pi;
      double e, v;

      cout << "Εισάγετε ακτίνα και ύψος ";
      cin >> r >> h;

      pi = acos(-1.0);
      e = 2*pi*r*h + 2*pi*r*r;
      v = pi*r*r*h;

      cout << "Το εμβαδόν είναι " << e << endl;
      cout << "Ο όγκος είναι " << v << endl;
}
```


Παράδειγμα #11

Σε ένα κατάστημα οι συναλλαγές στο ταμείο γίνονται μόνο με ακέραια πολλαπλάσια του ευρώ (δεν υπάρχουν λεπτά). Αν για μια πληρωμή ξέρουμε πόσα ρέστα θα πάρουμε, βρείτε πόσα χαρτονομίσματα των 20, 10 και 5 ευρώ καθώς και πόσα κέρματα των 2 και 1 ευρώ πρέπει να δοθούν ως ρέστα.

Παράδειγμα:
Αν τα ρέστα είναι 27 €, τότε πρέπει να δοθούν:
1 χαρτονόμισμα των 20 €
1 χαρτονόμισμα των 5 €
1 κέρμα των 2€

Παράδειγμα #11

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

int main ( )
{
      int r;                  // Τα ρέστα
      int n20, n10, n5;       // Πλήθος χαρτονομισμάτων
      int n2, n1;             // Πλήθος κερμάτων

      cout << "Πόσα ρέστα ? ";
      cin >> r;
```

Συνεχίζεται...

Παράδειγμα #11

...Συνέχεια

```
n20 = r/20;
r = r%20;

n10 = r/10;
r = r%10;

n5 = r/5;
r = r%5;

n2 = r/2;
r = r%2;

n1 = r;
```

Συνεχίζεται...

Παράδειγμα #11

...Συνέχεια

```
        cout << "Χαρτονομίσματα 20: " << n20 << endl;
        cout << "Χαρτονομίσματα 10: " << n10 << endl;
        cout << "Χαρτονομίσματα  5: " << n5  << endl;
        cout << "       Κέρματα  2: " << n2  << endl;
        cout << "       Κέρματα  1: " << n1  << endl;
}
```


Παράδειγμα #12

Κατασκευάστε πρόγραμμα το οποίο θα μετατρέπει ένα δεδομένο αριθμό δευτερολέπτων σε ώρες, λεπτά και δευτερόλεπτα.

Παράδειγμα #12

```
#include <iostream>
using namespace std;
int main ( )
{
        int sec, min, hour, ypolsec, ypolmin;

        cout << "Πόσα δευτερόλεπτα ? ";
        cin >> sec;

        min = sec / 60;
        ypolsec = sec % 60;

        hour = min / 60 ;
        ypolmin = min % 60 ;

        cout << sec     << " δευτερόλεπτα είναι: " << endl;
        cout << hour    << " ώρες " << endl;
        cout << ypolmin << " λεπτά " << endl;
        cout << ypolsec << " δευτερόλεπτα " << endl;
}
```