Εισαγωγή στη C++

```
#include <iostream>
using namespace std;
int main()
{
  cout << “hello world”;
  return 0;
}
```

#include <iostream>

- Η πρόταση αυτή αναγκάζει τον μεταγλωττιστή της C++ να συμπεριλάβει το αρχείο iostream κατά τη διαδικασία της μεταγλώττισης
- Αυτό το αρχείο περιέχει τις δηλώσεις των αντικειμένων και των συναρτήσεων εισόδου / εξόδου της καθιερωμένης βιβλιοθήκης της C++

using namespace std;

- Η πρόταση αυτή ενημερώνει τον μεταγλωττιστή της C++ ότι τα ονόματα που θα χρησιμοποιούνται (αντικείμενα, συναρτήσεις κλπ) από εκείνο το σημείο και μετά θα ανήκουν στον χώρο ονομάτων **std**.
- Η χρήση χώρων ονομάτων δίνει στη C++ τη δυνατότητα να χρησιμοποιεί οντότητες (μεταβλητές, συναρτήσεις, αντικείμενα κλπ) που έχουν το ίδιο όνομα αλλά εκτελούν διαφορετική λειτουργία ανάλογα με το όνομα του χώρου στον οποίο ανήκουν

main()

- Η συνάρτηση main() είναι υποχρεωτική σε κάθε πρόγραμμα
- Είναι η πρώτη που εκτελείται
- Με τη πρόταση return 0 η main() επιστρέφει τον κωδικό εξόδου (το 0) στο λειτουργικό σύστημα ώστε να του γνωστοποιήσει τον κανονικό τερματισμό του προγράμματος
- Υπό αυτή την έννοια η main() δηλώνεται ως int αφότου επιστρέφει την τιμή 0

Σχόλια προγράμματος

- Τα σχόλια μπορούν να εμπεριέχονται στο ζεύγος χαρακτήρων **/*** ...σχόλια... ***/**
- Τα τελευταία πρότυπα της γλώσσας εισήγαγαν μια νέα μορφή σχολίων, τα σχόλια γραμμής, τα οποία ξεκινούν με **//**

```
// ==============================
#include <iostream>
using namespace std;
int main()
{
        /* η χρήση του αντικειμένου cout */
        cout << "Hello world";      // Θα εμφανίσει τη φράση "Hello world"
        cout << "ΤΕΛΟΣ";            // Θα εμφανίσει τη λέξη "ΤΕΛΟΣ"
        return 0;
}
```

Σχόλια προγράμματος

- Ο μεταγλωττιστής αγνοεί οτιδήποτε υπάρχει εντός του ζεύγους **/* */** και οτιδήποτε υπάρχει μετά από διπλή κάθετο **//** μέχρι το τέλος της γραμμής

Δηλώσεις μεταβλητών

- int, char, float, double, bool
- **int a, b;**
- Τα ονόματα των μεταβλητών αλλά και των άλλων οντοτήτων της C++ καλούνται αναγνωριστικά
- Ένα αναγνωριστικό μπορεί να αποτελείται από λατινικούς χαρακτήρες, αριθμούς και τον χαρακτήρα _
- Υπάρχει διάκριση μεταξύ πεζών και κεφαλαίων

Ανάθεση τιμής σε μεταβλητή

- c=12.5;
- ch='A';
- Το = καλείται **τελεστής ανάθεσης**

Αρχικές τιμές μεταβλητών

- **int** a=4, b;
- **float** c=4.6, m;
- **char** ch='A';
- **bool** ans=true;

Μεταβλητές μόνο για ανάγνωση

- Στη C++ μπορούμε να δηλώνουμε μεταβλητές και να τους δίνουμε μια αρχική τιμή η οποία στη συνέχεια δεν μπορεί να αλλάξει, προσθέτοντας μπροστά από τον τύπο της το προσδιοριστικό **const**
- **const** int a=8;
- Μια μεταβλητή μόνο για ανάγνωση πρέπει απαραίτητα να αρχικοποιείται και δεν μπορούμε αργότερα να της αναθέσουμε άλλη τιμή

Τελεστής ανάθεσης

- a=6; //καταχωρίζει το 6 στη μεταβλητή a
- b=a=6; // καταχωρίζει το 6 στη μεταβλητή a και στη μεταβλητή b
- Στη C++ μια μεταβλητή προτού της ανατεθεί κάποια τιμή έχει απροσδιόριστο περιεχόμενο

Συναρτήσεις

- Για να καλέσουμε μια συνάρτηση χρησιμοποιούμε μόνο το όνομα της
- Η συνάρτηση μπορεί να επιστρέφει κάποια τιμή στον κώδικα που την κάλεσε
- Μια συνάρτηση στη C++ μπορεί να είναι συνάρτηση βιβλιοθήκης και να έχει προκαθορισμένη λειτουργία ή να ορίζεται μέσα στο ίδιο πρόγραμμα ως ξεχωριστό τμήμα
- Μια συνάρτηση μπορεί να αποτελεί μέθοδο μιας κλάσης και να εφαρμόζεται σε αντικείμενα αυτής της κλάσης

Παραστάσεις με μέλη διαφορετικού τύπου

- Η ιεραρχική σειρά των βασικών τύπων δεδομένων της C++ από τον χαμηλότερο προς τον υψηλότερο είναι:
- bool, char, int, float, double

```
int a=10, b=5;
double c;
c=(a+b)/2;
```

Η μεταβλητή c θα πάρει την τιμή 7. Για να πάρει την τιμή 7.5 πρέπει η παράσταση να γραφεί c=(a+b)/2.0 ώστε το δεξιό μέρος της ισότητας να επιστρέφει τιμή double

Λογικές παραστάσεις

- Μια λογική παράσταση απαρτίζεται από μια ή περισσότερες λογικές εκφράσεις που συνδέονται μεταξύ τους με τους λογικούς τελεστές
- Στη C++ κάθε λογική παράσταση έχει ως αποτέλεσμα 1 (**true**) και 0 (**false**)

Λογικές παραστάσεις

Συγκριτικοί τελεστές		Λογικοί τελεστές	
==	Ίσο	&&	Λογικό AND
!=	Διάφορο	||	Λογικό OR
>	Μεγαλύτερο	!	Λογικό NOT
>=	Μεγαλύτερο ή ίσο		
<	Μικρότερο		
<=	Μικρότερο ή ίσο		

Αλφαριθμητικές παραστάσεις

- Οι αλφαριθμητικές σταθερές είναι σύνολα χαρακτήρων μέσα σε διπλά εισαγωγικά

"Αυτό είναι μια αλφαριθμητική σταθερά"

- Δεν πρέπει να δημιουργείται σύγχυση με τις σταθερές χαρακτήρων οι οποίες κλείνονται μέσα σε μονά εισαγωγικά και χειρίζονται από τη C++ ως αριθμοί πχ **'C++'**

Είδη προτάσεων

- **Δηλωτικές προτάσεις** (declaration statements): δήλωση μεταβλητών, συναρτήσεων
- **Εκτελέσιμες προτάσεις**: εντολές, παραστάσεις ή κλήσεις συναρτήσεων
- **Προτάσεις ορισμού**: ορισμός συνάρτησης ή έναν νέτο τύπο δεδομένων

Μετατροπή τύπου κατά την ανάθεση τιμή σε μεταβλητή

```
int a, b;
float k=7.5, l=2;
a=12.465;
b=k/l;
k=2;
```

Στη παράσταση a=12.465, στη μεταβλητή a θα καταχωρηθεί το 12. Στη μεταβλητή b θα καταχωρηθεί ο αριθμός 3. Στη μεταβλητή k θα καταχωρηθεί ο αριθμός 2.0

Μετατροπή τύπου κατά την ανάθεση τιμή σε μεταβλητή

- Στη περίπτωση που η παράσταση δεξιά του τελεστή ανάθεσης είναι διαφορετικού τύπου από τη μεταβλητή αριστερά του τελεστή η C++ μετατρέπει τη τιμή της παράστασης στον τύπο της μεταβλητής. Στη περίπτωση που η τιμή της παράστασης έχει τύπο υψηλότερης ιεραρχίας από τον τύπο της μεταβλητής τότε υπάρχει περίπτωση απώλειας πληροφοριών

Σύνθετη πρόταση

```
 a=4;
{
   m=8;
   cout << "αυτή είναι μια σύνθετη πρόταση"
}
```

//Μέσα σε μια σύνθετη πρόταση μπορούν να δηλωθούν μεταβλητές οι οποίες έχουν ισχύ μόνο μέσα στην σύνθετη πρόταση όπου δηλώθηκαν

Προ – μεταγλωττιστής της C++

- Μέσα στον πηγαίο κώδικα του προγράμματος μας είναι δυνατόν να συμπεριλάβουμε οδηγίες προς τον μεταγλωττιστή της C++
- Οι οδηγίες που συναντάμε συνήθως σε ένα πρόγραμμα της C++ είναι οι οδηγίες **#include** και **#define**

Οδηγία #include

- Αναγκάζει τον μεταγλωττιστή της C++ να συμπεριλάβει κατά τη διαδικασία της μεταγλώττισης και κάποιο άλλο πηγαίο αρχείο στο οποίο συνήθως δηλώνονται οντότητες (αντικείμενα, συναρτήσεις κλπ) βιβλιοθήκης

Οδηγία #define

- Ορίζει ένα αναγνωριστικό και ένα σύνολο χαρακτήρων που θα αντικαταστήσει αυτό το αναγνωριστικό κατά τη διαδικασία της μεταγλώττισης του πηγαίου κώδικα
- Συνήθως χρησιμοποιούμε την οδηγία αυτή για να ορίσουμε κάποιες σταθερές στο πρόγραμμα μας

#define PI 3.141593;

Η C++ και η μνήμη

- Οι μεταβλητές καταλαμβάνουν χώρο στη μνήμη RAM
- Κάθε θέση μνήμης RAM έχει μέγεθος 1 byte
- Κάθε μεταβλητή καταλαμβάνει πλήθος θέσεων μνήμης ανάλογα από τον τύπο της, το τύπο του συστήματος Η/Υ και από τον μεταγλωττιστή
- Στους 32bit επεξεργαστές οι bool και char μεταβλητές καταλαμβάνουν 1 byte, οι int και float 4 byte και οι double 8 byte

Η C++ και η μνήμη

- Διεύθυνση μιας μεταβλητής είναι η διεύθυνση του πρώτου byte των θέσεων (ή της θέσης μνήμης) που δεσμεύει η μεταβλητή
- Μέγεθος μιας μεταβλητής είναι ο αριθμός των byte που δεσμεύει η μεταβλητή

Ο τελεστής &

- Ο τελεστής & επιστρέφει έναν αριθμό ο οποίος προσδιορίζει τη διεύθυνση μιας μεταβλητής
- Η διεύθυνση μιας θέσης μνήμης στα 32bit συστήματα είναι πάντα ένας αριθμός 4 byte ο οποίος συνήθως αναπαρίσταται σε δεκαεξαδική μορφή

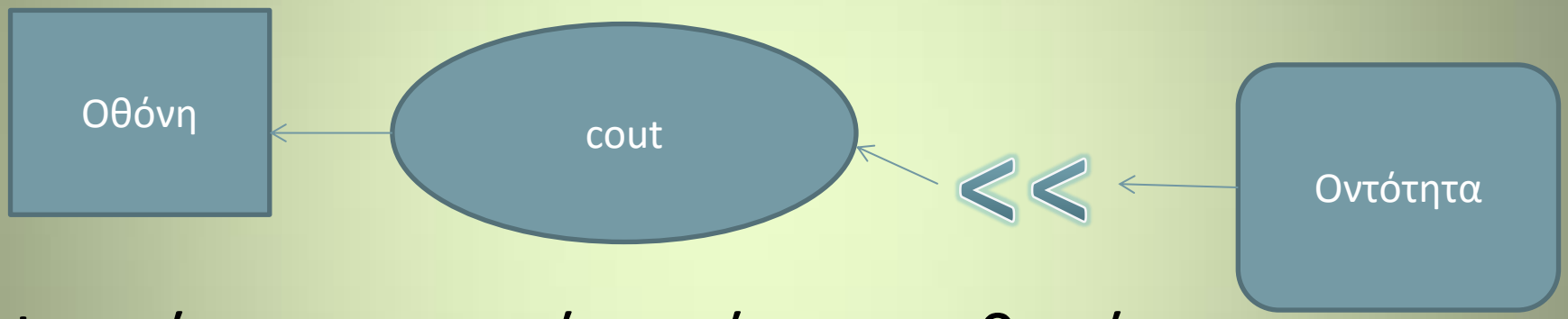
Ο τελεστής sizeof

- Επιστρέφει το πλήθος των byte που δεσμεύει μια μεταβλητή ή ένας τύπος δεδομένων

int a;

sizeof a → 4

Αντικείμενο cout

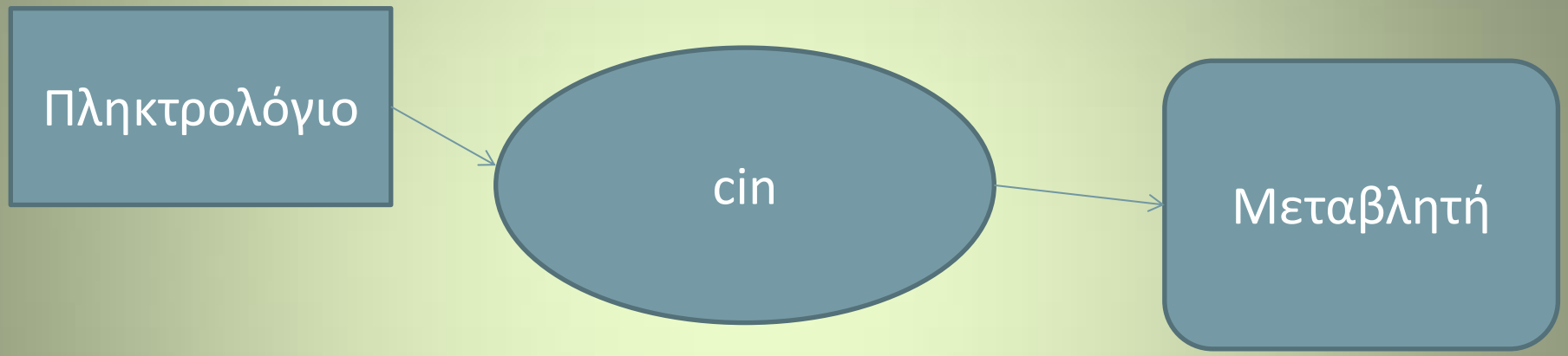


Η οντότητα μπορεί να είναι σταθερά, μεταβλητή, παράσταση, συνάρτηση, χειριστής

```
#include <iostream>
using namespace std;
int main()
{
        int a;
        char c;
        float b;
        a=15;
        b=a/2.0;
        c='A';
        cout <<"a+2="<<a+2<<" b="<<b<<endl<<"c="<<c<<endl<<"Τέλος\n";
        return 0;
}
```

a+2=17 b=7.5
c=A
Τέλος

Αντικείμενο cin



Μετά από κάθε χρήση του τελεστή εξαγωγής >> πρέπει να υπάρχει το όνομα μιας μεταβλητής η οποία δέχεται δεδομένα

```
#include <iostream>
using namespace std;
int main()
{
        int a,b;
        float c;
        cout << "δώσε 2 αριθμούς:"; //εμφανίζει το μήνυμα
        cin >> a >> b;                  //περιμένει τη πληκτρολόγηση δεδομένων
        c=(a+b)/2.0;                            //υπολογισμός μέσου όρου
        cout    << "ο μέσος όρος των "     //εμφάνιση αποτελεσμάτων
                        << a
                        << " και "
                        << b
                        << " είναι "
                        << c
                        << endl;
        return 0;
}
```

Οι αλλαγές γραμμής αγνοούνται από τον μεταγλωττιστή

Η συνάρτηση exit()

- Έχει ως αποτέλεσμα τον άμεσο τερματισμό του προγράμματος
- Ο κωδικός εξόδου 0 δηλώνει τον κανονικό τερματισμό του προγράμματος πχ **exit(0)**

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
        int a;
        char c;
        cout << "Αυτή η πρόταση θα εκτελεστεί";
        exit(0);
        cout << "Αυτή δε θα εκτελεστεί ποτέ";
        return 0;
}
```

Η συνάρτηση exit() δηλώνεται στο αρχείο κεφαλίδας **cstdlib**

Εντολή if

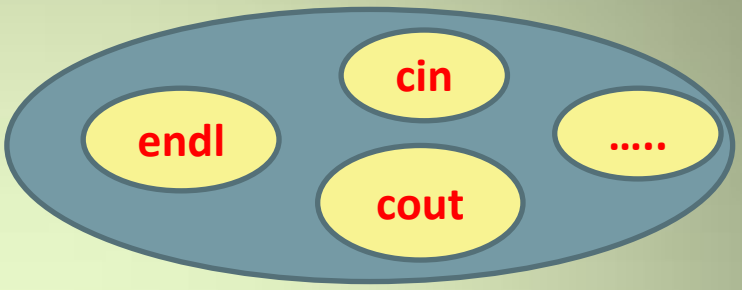
```
if (x>5)
    cout << "x>5\n";
else
    cout <<"x<=5\n";

if(συνθήκη)
  {
     Σύνθετη πρόταση Α
   }
 else
   {
      Σύνθετη πρόταση Β
   }
```

Χώροι ονομάτων στη C++

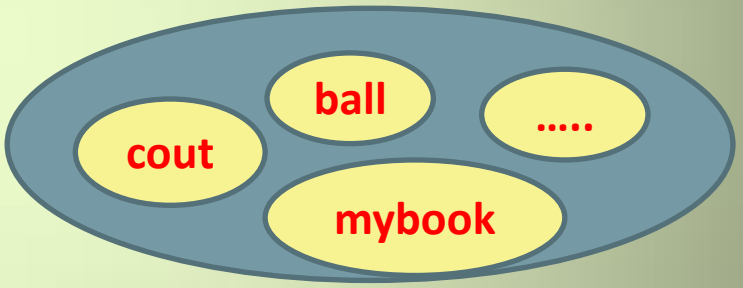
- Οι χώροι ονομάτων (αναφοράς) μας επιτρέπουν να δίνουμε σε ίδιες έννοιες διαφορετικό νόημα ανάλογα με το περιβάλλον στο οποίο τις χρησιμοποιούμε
- Κάθε βιβλιοθήκη της C++ διαθέτει ένα όνομα αναφοράς (χώρο ονομάτων). Η καθιερωμένη βιβλιοθήκη της C++ (standard library) διαθέτει τον χώρο ονομάτων **std**. Άλλες βιβλιοθήκες διαθέτουν χώρους ονομάτων που προσδιορίζονται από τους κατασκευαστές τους.

Χώροι ονομάτων στη C++

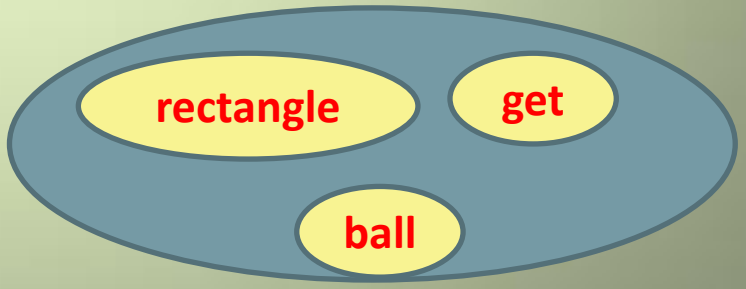
Χώρος ονομάτων **std**



Χώρος ονομάτων **kitsos**



Χώρος ονομάτων **mitsos**



Χώροι ονομάτων στη C++

- Στο προηγούμενο σχήμα απεικονίζονται 3 βιβλιοθήκες.
- Οι βιβλιοθήκες είναι πιθανό να περιέχουν οντότητες με το ίδιο όνομα αλλά να έχουν εντελώς διαφορετική λειτουργία
- Για να αναφερθούμε σε κάποια από αυτές τις οντότητες θα πρέπει να προσδιορίσουμε τον χώρο ονομάτων στον οποίο ανήκει με τη χρήση του **τελεστή επίλυσης εμβέλειας ::**
- **std::cout** //αναφέρεται στην οντότητα cout του χώρου ονομάτων std
- Η πρόταση **using namespace std;** Στην κορυφή ενός προγράμματος ορίζει ως χώρο αναφοράς τον χώρο ονομάτων std, επομένως όταν αναφερόμαστε σε οντότητες της καθιερωμένης βιβλιοθήκης της C++ δεν χρειάζεται να χρησιμοποιούμε το πρόθεμα **std::**
- Στο παρακάτω παράδειγμα η πρόταση έχει παραληφθεί, συνεπώς πριν από κάθε οντότητα της βιβλιοθήκης είμαστε υποχρεωμένοι να προσθέτουμε το πρόθεμα χώρου στον οποίο ανήκει, διαφορετικά ο μεταγλωττιστής δεν θα την αναγνωρίζει

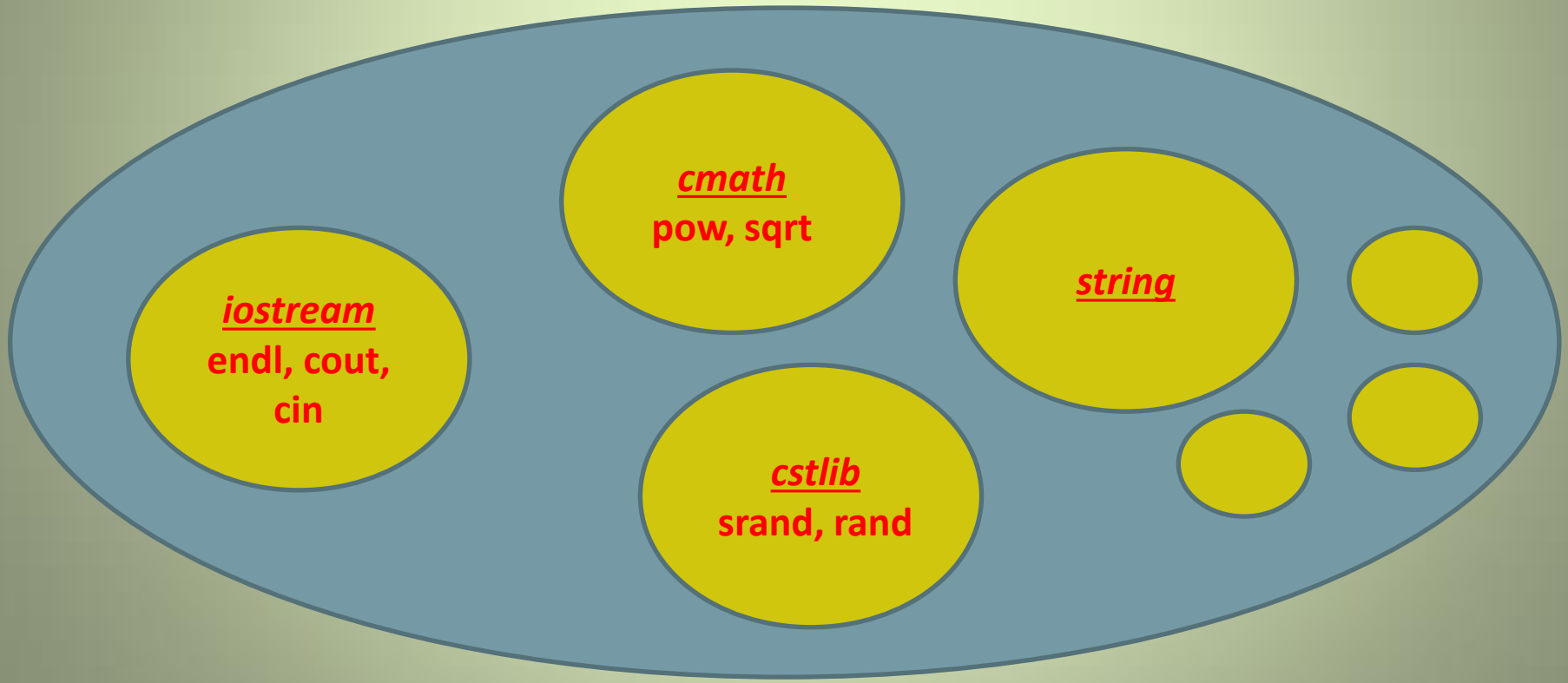
```
#include <iostream>

int main()
{
        float a,b,mo;
        std::cout << "Δώσε δύο αριθμούς :";
        std::cin >> a >> b;
        mo=(a+b)/2;
        std::cout << "Ο μέσος όρος είναι "
                        << mo << std::endl;
        return 0;
}
```

C++ standard library

- Τεράστια συλλογή έτοιμων οντοτήτων
- Περιέχει αντικείμενα, κλάσεις, συναρτήσεις, χειριστές κλπ
- Χωρίζεται σε ενότητες κάθε μια από τις οποίες περιέχει οντότητες που σχετίζονται με κάποιο θέμα πχ μαθηματικές, χειρισμό συμβολοσειρών.
- Οι οντότητες κάθε ενότητας δηλώνονται μέσα σε αντίστοιχα αρχεία κώδικα τα οποία λέγονται **αρχεία κεφαλίδας**

Αρχεία κεφαλίδας της καθιερωμένης βιβλιοθήκης της C++

Χώρος ονομάτων **std**



Παραδείγματα

- Το παρακάτω πρόγραμμα ζητάει τρεις βαθμολογίες ενός μαθητή και υπολογίζει τον μέσο όρο τους. Αν ο μέσος όρος είναι μεγαλύτερος ή ίσος με το 10 εμφανίζει τη φράση «Πέρασες με βαθμό ###» διαφορετικά εμφανίζει τη λέξη «Κόπηκες»

```
#include <iostream>
using namespace std;
int main()
{
	int b1,b2,b3;
	float mo;
	cout << "Δώσε τρεις βαθμούς:";
	cin >> b1 >> b2 >> b3;
	mo=(b1+b2+b3)/3.0;
	if(mo>=10)
		cout << "Πέρασες με βαθμό "<< mo << endl;
	else
		cout << "Κόπηκες" << endl;
	cout << "============================";
	return 0;
}
```

- Το επόμενο πρόγραμμα ζητάει έναν αριθμό και μας δείχνει αν είναι ζυγός ή μονός

```
#include <iostream>
using namespace std;
int main()
{
	int a,p;
	cin >> a;
	p=a/2;
	if(a==p*2)
		cout << "Ζυγός" << endl;
	else
		cout << "Μονός" << endl;
	return 0;
}
```

- Το παρακάτω πρόγραμμα ζητάει να πληκτρολογηθούν 2 ακέραιοι αριθμοί και εμφανίζει τον μέσο όρο τους και το άθροισμα τους

```
#include <iostream>
using namespace std;
int main()
{
        int a,b,s;
        float mo;
        cin >> a >> b;
        mo=(a+b)/2.0;
        s=a+b;
        cout <<"MO="<<mo<<endl;
        cout <<"Αθροισμα="<<s<<endl;
        return 0;
}
```

Οι βασικοί τύποι δεδομένων

- int (ακέραιοι) 4 byte, Το 31^{ο} bit συμβολίζει το πρόσημο
- long int: 4 byte, ±2.147.483647
- short int: 2 byte, ±32.768
- unsigned int (το πρώτο bit του αριθμού δεν χρησιμοποιείται για το πρόσημο αλλά συμμετέχει στην αναπαράσταση του αριθμού, μόνο θετικοί αριθμοί

Τελεστές ++ --

- Όταν ο τελεστής βρίσκεται πριν από τη μεταβλητή (προθεματικός) τότε πρώτα γίνεται η πράξη και μετά επιστρέφεται η νέα τιμή της μεταβλητής.
- Όταν ο τελεστής βρίσκεται μετά από τη μεταβλητή (επιθεματικός) τότε πρώτα επιστρέφει την τιμή της μεταβλητής (την παλιά τιμή) και μετά αλλάζει το περιεχόμενο της μεταβλητής

Τελεστές σύντμησης

Σύντμηση	Αντίστοιχη παράσταση
x+=5	x=x+5
x-+5	x=x-5
x*=5	x=x*5
x/=5	x=x/5
x%=5	x=x%5

Τελεστές bitwise

- Η C++ όπως και η C μας επιτρέπει τον χειρισμό πληροφοριών σε επίπεδο δυαδικών αριθμών
- 6 τελεστές bitwise οι οποίοι επιδρούν σε μεμονωμένα bit ενός ακεραίου
- Οι δυαδικοί τελεστές εκτελούν αριθμητικές και όχι λογικές πράξεις
- Το αποτέλεσμα μιας δυαδικής πράξης μπορεί να είναι οποιοσδήποτε αριθμός ενώ μιας λογικής μπορεί να είναι το 0 ή το 1
- Ιδιαίτερα χρήσιμοι σε προγράμματα που ελέγχουν μικροελεγκτές στη ρομποτική

Τελεστές bitwise

Τελεστής	Ερμηνεία
&(bitwise AND)	Εκτελεί την πράξη END
|(bitwise OR)	Εκτελεί την πράξη OR
^(bitwise XOR)	Εκτελεί την πράξη XOR
~(συμπλήρωμα)	Αντιστρέφει τα bit
<< (ολίσθηση αριστερά)	Μετακινεί τα bit προς τα αριστερά με αποτέλεσμα το αριστερότερο να χάνεται και το δεξιότερο να γίνεται 0
>> (ολίσθηση δεξιά)	Μετακινεί τα bit προς τα αριστερά με αποτέλεσμα το δεξιότερο να χάνεται και το αριστερότερο να γίνεται 0

```
#include <iostream>
using namespace std;
int main()
{
        int a,b,c,d;
        a=166;
        b=176;
        c=a && b;
        d=a & b;
        cout << "c=" << c << endl << "d=" << d << endl;
        return 0;
}
```

c=1
d=160

```
#include <iostream>
using namespace std;
int main()
{
        int a,b,c,d;
        a=8;
        b=4;
        c=a || b;
        d=a | b;
        cout << "c=" << c << endl << "d=" << d << endl;
        return 0;
}
```

c=1
d=12

Προτεραιότητα τελεστών

Τελεστής	Σημασία
!, ~, ++, --	Λογικό NOT, συμπλήρωμα, αύξηση, μείωση
&	Απόδοση διεύθυνσης
sizeof	Απόδοση μεγέθους
*,/,%	Πολλαπλασιασμός, διαίρεση, υπόλοιπο
+,-	Πρόσθεση, αφαίρεση
<<, >>	Ολίσθηση
<, <=, >, >=	Μικρότερο, μεγαλύτερο
==, !=	Ίσο, διάφορο
&	Bitwise AND
^	Bitwise XOR
|	Bitwise OR
&&	Λογικό AND
||	Λογικό OR
?:	OR συνθήκης
=	Ανάθεση

Τύποι float και double

- Στη C++ όποιος αριθμός περιέχει τον χαρακτήρα της υποδιαστολής (.) θεωρείται double. Για να θεωρηθεί float πρέπει να οριστεί ως εξής 123.4F
- Float: 4byte
- Double: 8byte
- Ο αριθμητικός τελεστής % δεν εφαρμόζεται σε δεδομένα κινητής υποδιαστολής

Μορφοποιημένη έξοδος με το cout

- Χειριστής setw(n) : χρησιμοποιείται για τον καθορισμό των θέσεων που θα καταλάβει η αμέσως επόμενη έξοδος του αντικειμένου cout. Ο αριθμός των θέσεων καθορίζεται από την παράμετρο n του χειριστή

cout << setw(10)<< a << endl;

Θα έχει ως αποτέλεσμα να εμφανιστεί το περιεχόμενο της μεταβλητής a σε έναν χώρο 10 θέσεων

- Οι χειριστές που χρησιμοποιούν παραμέτρους δηλώνονται στο αρχείο κεφαλίδας iomanip, συνεπώς επιβάλλεται η οδηγία **#include <iomanip>**

Μορφοποιημένη έξοδος με το cout

- Στη περίπτωση που έχουμε δεκαδικά ψηφία γίνεται η χρήση των χειριστών **setprecision(n)** ο οποίος καθορίζει το πλήθος των δεκαδικών ψηφίων και **fixed** ο οποίος καθορίζει τον τρόπο εμφάνισης ενός δεκαδικού αριθμού έτσι ώστε να εμφανίζεται στην κανονική μορφή του και όχι σε εκθετική ή άλλη μορφή
- Ο χειριστής setw(n) εφαρμόζεται μόνο στην επόμενη έξοδο, ενώ ο setprecision(n) και fixed σε όλες τις εξόδους που ακολουθούν

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
        float a,b;
        a=21.234;
        b=5467.1;
        cout << setw(10) << setprecision(3) << fixed << a << endl;
        cout << setw(10) << setprecision(3) << fixed << b << endl;
   return 0;
}
```

21.234
5467.100

Οι συναρτήσεις pow() και sqrt()

- Ύψωση σε δύναμη και τετραγωνική ρίζα
- **#include <cmath>**
- Επιστρέφουν τιμές τύπου double
- Το επόμενο πρόγραμμα υπολογίζει το 4^5 και την τετραγωνική ρίζα του 200

```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
        double c, t;
        c=pow(4,5);
        cout << "4 εις την 5 = " << c << endl;
        t=sqrt(200);
        cout << "Η τετραγωνική ρίζα του 200 είναι" <<t
        << endl;
        return 0;
}
```

- Το επόμενο πρόγραμμα ζητάει δυο δεκαδικούς διπλής ακρίβειας και μετά εμφανίζει τους αριθμούς και το άθροισμα σε 10 θέσεις με 4 δεκαδικά ψηφία

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
        double a,b,c;
        cin >> a >> b;
        c=a+b;
        cout << setw(10) << setprecision(4) << fixed << a << endl;
        cout << setw(10) << b << endl;
        cout << "==========" << endl;
        cout << setw(10) << c << endl;
        return 0;
}
```

```
15.2143
345.1234
15.2143
345.1234
==========
360.3377

--------------------------------
Process exited after 19.26 seconds with return value 0
Press any key to continue . . .
```

Ο τύπος δεδομένων char

- Σταθερές τύπου char: ΄u΄, ΄3΄, ΄*΄
- Οι χαρακτήρες είναι αριθμοί
- Ο αριθμός αυτό είναι ο κωδικός **ASCII** του χαρακτήρα και προκύπτει από τον πίνακα μετατροπής (κωδικοποίηση 256 χαρακτήρων)
- Η C++ χειρίζεται τους χαρακτήρες ως αριθμούς και επομένως μπορούν να μετέχουν σε αριθμητικές παραστάσεις
- cout << ΄Α΄+1 // Θα εμφανίσει το 66 διότι ο αριθμός που αντιστοιχεί στο Α είναι το 65

Χαρακτήρες διαφυγής

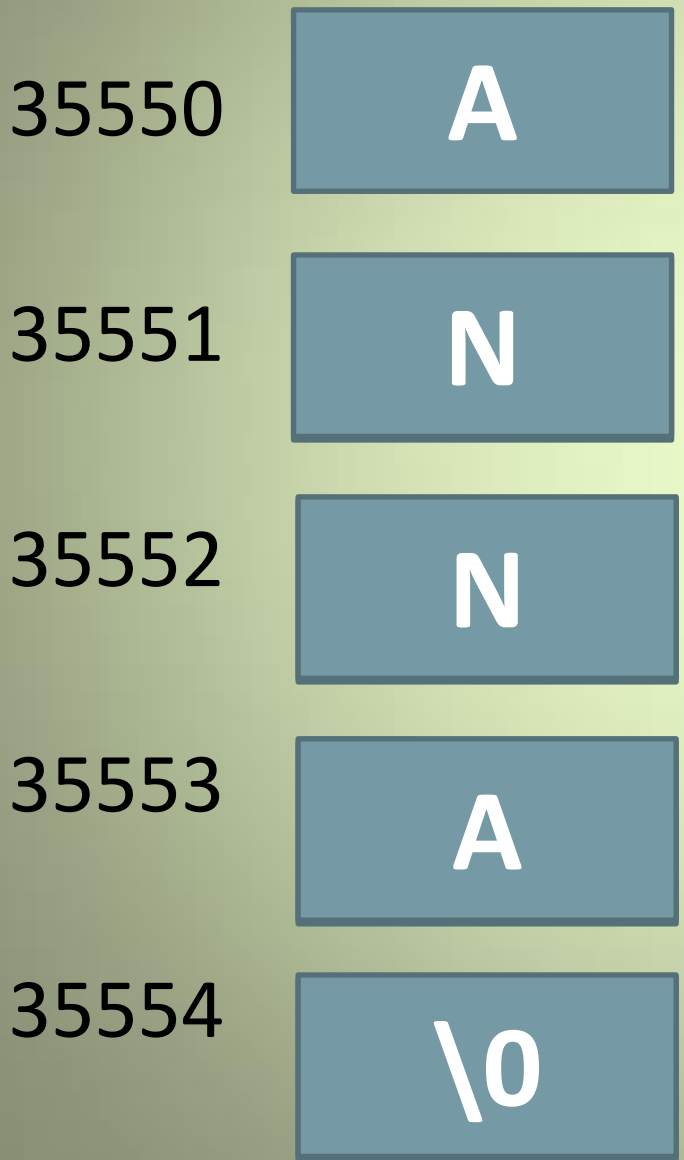
- Είναι χαρακτήρες ελέγχου των λειτουργιών της οθόνης και δεν εμφανίζονται στην οθόνη

Χαρακτήρας Διαφυγής	Ενέργεια
\n	New line (νέα γραμμή)
\b	Backspace (μια θέση πίσω)
\f	Form feed (νέα σελίδα)
\r	Carriage return (αρχή γραμμής)
\t	Tab (στηλοθέτης)
\\	Backslash (ο χαρακτήρας ανάποδης καθέτου)
\’	Απλά εισαγωγικά
\"	Διπλά εισαγωγικά
\0	Χαρακτήρας τερματισμού 00000000

Συμβολοσειρές (character strings)

- Η τιμή μιας συμβολοσειράς είναι η αρχική διεύθυνση της μνήμης όπου είναι αποθηκευμένη η συμβολοσειρά και όχι οι ίδιοι οι χαρακτήρες της
- Η C++ χειρίζεται μια συμβολοσειρά ως διεύθυνση

Συμβολοσειρές (character strings)



Συμβολοσειρές (character strings)

- Στις συναρτήσεις της C++ που χειρίζονται συμβολοσειρές μεταβιβάζεται πάντα ως όρισμα η αρχική διεύθυνση της συμβολοσειράς
- Η συνάρτηση εντοπίζει το τέλος της συμβολοσειράς όταν ανιχνεύσει τον χαρακτήρα τερματισμού \0

Η κλάση string

- Βρίσκεται στην καθιερωμένη βιβλιοθήκη της C++
- Χρησιμοποιείται για την αποθήκευση συμβολοσειρών όπως στη C γίνεται χρήση πινάκων
- Όταν δηλώνουμε μεταβλητές τύπου string δημιουργούμε αντικείμενα της κλάσης string
- Η δήλωση ενός αντικειμένου string γίνεται με μια δηλωτική πρόταση πχ **string lex;**
- Το αντικείμενο lex είναι ένας αποθηκευτικός χώρος στον οποίο μπορούν να καταχωρηθούν συμβολοσειρές με τον τελεστή ανάθεσης

Πράξεις και συγκρίσεις μεταξύ αντικειμένων κλάσης string

- Μπορούμε να χρησιμοποιήσουμε τον τελεστή +

```
string lex1, lex2, lex3
lex1="panepistimio";
lex2="kapodistriako";
lex3=lex2+ "  " +lex1;
```

- Επίσης με τα αντικείμενα της κλάσης string μπορούμε να χρησιμοποιήσουμε τελεστές σύγκρισης
- Το επόμενο πρόγραμμα ζητάει από τον χρήστη να πληκτρολογήσει τρεις διαφορετικές λέξεις και συνθέτει μια φράση που αποτελείται από αυτές τις λέξεις. Ακολούθως συγκρίνει τις δυο πρώτες λέξεις μεταξύ τους ώστε να διαπιστώσει ποια είναι αλφαβητικά μεγαλύτερη

```
#include <iostream>
using namespace std;
int main()
{
	string lex1,lex2,lex3,frasi;
	cout << "Δώσε τρεις λέξεις μιας φράσης χωρισμένες με <enter>:";
	cin >> lex1 >> lex2 >> lex3;
	frasi=lex1+" "+lex2+" "+lex3;
	cout << "Η φράση είναι:" << frasi << endl;
	if (lex1>lex2)
		cout << "Η πρώτη λέξη που έδωσες είναι αλφαβητικά μεγαλύτερη
από τη δεύτερη"<< endl;
	else if (lex2>lex1)
		cout << "Η δεύτερη λέξη που έδωσες είναι αλφαβητικά
μεγαλύτερη από τη πρώτη"<< endl;
	else
		cout << "Πρώτη και δεύτερη λέξη είναι ίδιες" << endl;
	return 0;
}
```

Παρόλο που μπορούμε να συγκρίνουμε αντικείμενα string μεταξύ τους, δεν μπορούμε να συγκρίνουμε συμβολοσειρές (η C++ τις χειρίζεται ως θέσεις μνήμης)

Αντικείμενα string και χαρακτήρες

- Στις παραστάσεις μπορούν να αναμιγνύονται χαρακτήρες και αντικείμενα της κλάσης string χωρίς κανένα πρόβλημα
- Σε ένα αντικείμενο της κλάσης string μπορεί να καταχωριστεί ένας μόνο χαρακτήρας
- Το πρόγραμμα που ακολουθεί ζητάει από τον χρήστη να πληκτρολογήσει μια λέξη και 2 χαρακτήρες και ακολούθως προσθέτει τον έναν στην αρχή και τον άλλον στο τέλος της λέξης

```
#include <iostream>
using namespace std;
int main()
{
	string lex1,startstop;
	char ch1,ch2;
	startstop='-';
	cout << "Δώσε μια λέξη:";
	cin >> lex1;
	cout << "Δώσε δύο χαρακτήρες χωρισμένους με κενό:";
	cin >> ch1 >> ch2;
	lex1=startstop+ch1+lex1+ch2+startstop;
	cout << lex1 << endl;
	return 0;
}
```

Αντικείμενα string και χαρακτήρες

- Η δυνατότητα χρήσης του τελεστή ανάθεσης = για την καταχώρηση μιας συμβολοσειράς σε ένα αντικείμενο string, των τελεστών σύγκρισης για να συγκρίνουμε αντικείμενα αυτής της κλάσης, καθώς και του τελεστή + για να τα προσθέτουμε τόσο μεταξύ τους όσο και με συμβολοσειρές και χαρακτήρες, οφείλεται στην υπερφόρτωση αυτών των τελεστών, ένα από τα χαρακτηριστικά του **πολυμορφισμού**. Η **υπερφόρτωση τελεστών** είναι μια διαδικασία με την οποία εκπαιδεύουμε τη γλώσσα για το πώς θα εφαρμόζει συγκεκριμένους τελεστές στα αντικείμενα μιας κλάσης.

Ο τύπος δεδομένων bool

- Λογικά δεδομένα με τιμές true ή false
- Καταλαμβάνουν 1 byte
- Το παρακάτω πρόγραμμα ζητάει 3 αριθμούς και στη περίπτωση όπου δοθούν με αύξουσα σειρά καταχωρίζει σε μια λογική μεταβλητή την τιμή true, αλλιώς την τιμή false

```
#include <iostream>
using namespace std;
int main()
{
	float a,b,c;
	bool sosti_seira;
	cin >> a >> b >> c;
	if (a<=b && b<=c)
		sosti_seira=true;
	else
		sosti_seira=false;
	if (sosti_seira)
		cout << "Οι αριθμοί δόθηκαν με τη σειρά";
	else
		cout << "Λάθος σειρά";
	return 0;
}
```

Απόλυτος προσδιορισμός τύπου αριθμητικών σταθερών

Τύπος δεδομένων	Επίθημα τύπου	Παραδείγματα
Int		12
Long int	L	12L
Unsigned int	U	12U
Unsigned long int	UL	12UL
Float	F	12.34F
Double		12.34
Long double	L	12.34L

Αυτόματη μετατροπή τύπου

- Όταν χρησιμοποιούμε τον τελεστή ανάθεσης με δεδομένα διαφορετικού τύπου η C++ μετατρέπει το αποτέλεσμα της παράστασης που βρίσκεται δεξιά από το = στον τύπο της μεταβλητής που βρίσκεται αριστερά
- Όταν η μετατροπή γίνεται από υψηλότερη ιεραρχία δεδομένων σε χαμηλότερη ενδέχεται να υπάρξει απώλεια πληροφοριών

Αυτόματος τύπος δεδομένων

- Το πρότυπο C++11 εισήγαγε τον αυτόματο τύπο δεδομένων auto ο οποίος λέει στον compiler να καθορίσει εκείνος τον τύπο της μεταβλητής ανάλογα με την τιμή που της ανατίθεται ως αρχική τιμή
- Πχ **auto k=3;** // ο k θεωρείται μεταβλητή τύπου int

Εντολές συνθήκης

- if, if – else, if – else –if
- switch – case
- Ανάλογο συντακτικό με τη C
- Η C++ θεωρεί αληθή οποιαδήποτε παράσταση επιστρέφει τιμή διάφορη του 0

Η εντολή switch - case

- Ελέγχει αν η τιμή μιας ακεραίας παράστασης ισούται με συγκεκριμένες σταθερές (και μόνο σταθερές) και ανάλογα εκτελεί συγκεκριμένες προτάσεις

Λύση εξίσωσης 2ου βαθμού

```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
	float a,b,c,r1,r2,d;
	cout << "Δωσε τους συντελεστές a, b και c:";
	cin >> a >> b >> c;
	d=pow(b,2)-4*a*c;
	if(d>0)
	{
		r1=(-b + sqrt(d))/(2*a);
		r2=(-b - sqrt(d))/(2*a);
		cout << "Οι ρίζες είναι " << r1 << " και " << r2;
	}
	else if(d==0)
	{
		r1=-b/(2*a);
		cout << "Διπλή ρίζα " << r1;
	}
	else
		cout << "Δεν υπάρχουν πραγματικές ρίζες";
	return 0;
}
```

- Το επόμενο πρόγραμμα ζητάει 2 αριθμούς και έναν χαρακτήρα και εκτελεί την ανάλογη πράξη

```
#include <iostream>
using namespace std;
int main()
{
	int a,b;
	char ch;
	cout << "Δώσε δύο αριθμούς και ένα χαρακτήρα:";
	cin >> a >> b >> ch;
	switch(ch)
	{
		case '+':
			cout << "Το άθροισμα είναι " << a+b << endl;
			break;
		case '-':
			cout << "Η διαφορά είναι " << a-b << endl;
			break;
		case '*':
			cout << "Το γινόμενο είναι " << a*b << endl;
			break;
		default:
			cout << "Λάθος πράξη\n";
	}
	return 0;
}
```

Δομές επανάληψης

- **while:** δεν είναι καθορισμένο το πλήθος των επαναλήψεων
- **do – while**: χρησιμοποιείται όταν η επανάληψη του βρόχου εξαρτάται από τιμές που εισάγονται από τον χρήση σε προτάσεις οι οποίες βρίσκονται μέσα στον βρόχο. Στις περιπτώσεις αυτές θέλουμε οι προτάσεις του βρόχου να εκτελεστούν τουλάχιστον μια φορά ώστε ο χρήστης να δώσει τις πρώτες τιμές
- **for:** συγκεκριμένος αριθμός επαναλήψεων

while

```
#include <iostream>
using namespace std;
int main()
{
        int a,b;
        a=5;
        while (a>0)
        {
                cout << "a=" << a << endl;
                --a;
        }
        cout << "Τέλος"<<endl;
        return 0;
}
```

do - while

```
#include <iostream>
using namespace std;
int main()
{
        int poso,sum=0,plithos=0;
        do
        {
                cout << “Δώσε ποσό: ";
                cin >> poso;
                sum=sum+poso;
                plithos++;
        } while (sum<1000);
        cout << "Συνολικό ποσό"<<sum<<endl;
        cout << "Άτομα που μετείχαν στον έρανο: "<<plithos<<endl;
        return 0;
}
```

Εντολές break και continue

- Η εντολή **break** τερματίζει μια πρόταση case στην εντολή switch και επιβάλλει τον άμεσο τερματισμό ενός βρόχου while, do – while, for
- Η εντολή **break** διακόπτει την εκτέλεση μιας επαναληπτικής διαδικασίας και μεταφέρει τον έλεγχο του προγράμματος αμέσως μετά από την επαναληπτική δομή
- Η εντολή **continue** δεν τερματίζει τον βρόχο αλλά επιβάλλει την εκτέλεση της επανάληψης παραλείποντας τις ενδιάμεσες εντολές από την continue μέχρι το τέλος του βρόχου

Καταμέτρηση και άθροιση σε επαναλαμβανόμενες διαδικασίες

- Όταν θέλουμε να μετρήσουμε πόσες φορές προέκυψε ένα συμβάν χρησιμοποιούμε μια μεταβλητή (**cnt – μετρητής**) την οποία την αρχικοποιούμε με τιμή 0 και κάθε φορά που προκύπτει το συμβάν την αυξάνουμε κατά ένα **cnt++**
- Όταν θέλουμε να αθροίσουμε τις διαφορετικές τιμές που παίρνει μια μεταβλητή a χρησιμοποιούμε μια θέση μνήμης (**αθροιστής sum**) και κάθε φορά που η μεταβλητή παίρνει μια νέα τιμή αυξάνουμε τον αθροιστή κατά το περιεχόμενο της μεταβλητής (**sum = sum + a**)
- Το παρακάτω πρόγραμμα διαβάζει έναν προς έναν 12 αριθμούς που δίνονται από το πληκτρολόγιο και εμφανίζει το άθροισμα τους. Για τη καταμέτρηση των αριθμών χρησιμοποιείται η μεταβλητή **cnt** και για το άθροισμα τους η μεταβλητή **sum**

```
#include <iostream>
using namespace std;
int main()
{
	int a, cnt, sum;
	cnt=sum=0;
	while (cnt<12)
	{
		cout <<"Δώσε αριθμό:";
		cin >> a;
		cnt++;
		sum=sum+a;
	}
	cout << "Άθροισμα=" << sum << endl;
	return 0;
}
```

Συναρτήσεις

- Οι συναρτήσεις εκτελούνται μόνο αν κληθούν από μια άλλη συνάρτηση. Η μοναδική συνάρτηση που εκτελείται αυτόματα είναι η **main()**
- Δηλώνονται (ή και ορίζονται) πριν από τη main() (προκαταβολική δήλωση – forward declaration)
- Στη C++ κάθε συνάρτηση ανήκει σε κάποιο τύπο δεδομένων (int, float κλπ). Μια συνάρτηση μπορεί να έχει καμία, μία ή περισσότερες παραμέτρους
- Στη περίπτωση που η συνάρτηση δεν επιστρέφει τιμή δηλώνεται ως τύπου void
- Οι παράμετροι της συνάρτησης ορίζονται μαζί με τον τύπο τους μέσα στις παρενθέσεις που ακολουθούν το όνομα της συνάρτησης
- Οι παράμετροι της συνάρτησης αποτελούν και αυτές τοπικές μεταβλητές της συνάρτησης

Συναρτήσεις χωρίς παραμέτρους

- Στο παρακάτω πρόγραμμα η main() καλεί τη συνάρτηση display_numbers() δυο φορές. Κάθε φορά η συνάρτηση εμφανίζει τους ίδιους αριθμούς. Από τη στιγμή που δεν έχει παραμέτρους δεν υπάρχει τρόπος να της μεταβιβάσουμε πληροφορίες και να επηρεάσουμε την λειτουργίας της

```
#include <iostream>
using namespace std;
void display_numbers()
{
        int i;
        for (i=5;i<=10;i++) cout << i <<endl;
}
int main()
{
        cout<<"Πρώτη κλήση της συνάρτησης"<<endl;
        display_numbers();
        cout<<"Δεύτερη κλήση της συνάρτησης"<<endl;
        display_numbers();
        return 0;
}
```

Συναρτήσεις με παραμέτρους

- Στο παρακάτω πρόγραμμα η συνάρτηση display_numbers() καλείται 2 φορές με διαφορετικά ορίσματα
- Στη περίπτωση που δεν υπάρχει εντολή **return** στο σώμα της συνάρτησης, η συνάρτηση επιστρέφει τον έλεγχο στον κώδικα που την κάλεσε μόλις εκτελεστεί η τελευταία της πρόταση, χωρίς όμως να επιστρέψει κάποια τιμή
- Μια συνάρτηση επιστρέφει στον κώδικα που την κάλεσε μόλις εκτελεστεί η τελευταία της πρόταση ή μόλις συναντήσει την εντολή **return**

```
#include <iostream>
using namespace std;
void display_numbers(int apo, int eos)
{
        int i;
        for (i=apo;i<=eos;i++) cout << i <<endl;
}
int main()
{
        display_numbers(15,20);
        display_numbers(122,140);
        return 0;
}
```

Συναρτήσεις που επιστρέφουν τιμή

- Το επόμενο πρόγραμμα καλεί τη συνάρτηση **mo()** 2 φορές
- Την πρώτη φορά δεν χρησιμοποιεί την τιμή που επιστρέφεται, ενώ τη δεύτερη την καταχωρίζει στη μεταβλητή **f**

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

float mo(int x, int y, int z)
{
        float m;
        m = (x+y+z)/3.0;
        return(m);
}

int main()
{
        int a=10,b=3,c=13;
        float f;
        mo(a,b,c);
        f=mo(a,b,c);
        cout << "Ο μέσος όρος είναι:" << f <<endl;
        return 0;
}
```

Συναρτήσεις που δεν επιστρέφουν τιμή

- Οποιαδήποτε συνάρτηση που δεν επιστρέφει τιμή πρέπει να δηλώνεται ως τύπου **void**
- Έτσι ο μεταγλωττιστής γνωρίζει ότι η συνάρτηση δεν επιστρέφει τιμή και προστατεύει τον προγραμματιστή από ενδεχόμενα λάθη
- Αν θέλουμε η συνάρτηση να επιστρέψει στον κώδικα που την κάλεσε πριν φτάσει στο τέλος της χρησιμοποιούμε την εντολή return χωρίς παράσταση

Ορίσματα και μεταβίβαση παραμέτρων

- Οι τιμές με τις οποίες καλείται μια συνάρτηση λέγονται **ορίσματα** κλήσης
- Αυτός ο τρόπος μεταβίβασης ορισμάτων λέγεται κλήση ή **μεταβίβαση κατ αξία** (by value) όπου οι τιμές των ορισμάτων αντιγράφονται στις αντίστοιχες τυπικές παραμέτρους της συνάρτησης
- Οι μεταβλητές των παραμέτρων που ορίζονται σε μια συνάρτηση λέγονται τυπικές παράμετροι (**formal parameters**)
- Όταν καλείται μια συνάρτηση πρέπει το πλήθος και ο τύπος των ορισμάτων να είναι αντίστοιχο με τις τυπικές παραμέτρους της όπως αυτές έχουν δηλωθεί στον ορισμό της συνάρτησης
- Η C++ υποστηρίζει και την κλήση κατ αναφορά (**by reference**)

Συνάρτηση χωρίς παραμέτρους με ρητή δήλωση

- Όταν μια συνάρτηση δεν έχει παραμέτρους τότε για να αποφευχθεί ο λάθος χειρισμός της δηλώνουμε εντός των παρενθέσεων τη λέξη **void**
- πχ **display_numbers(void)**

Υπερφόρτωση συναρτήσεων

- Η C++ επιτρέπει τον ορισμό διαφορετικών συναρτήσεων με το ίδιο όνομα (**τεχνική της υπερφόρτωσης – function overloading**)
- Η υπερφόρτωση μιας συνάρτησης επιτυγχάνεται με τον ορισμό διαφορετικών εκδόσεων της συνάρτησης με το ίδιο όνομα. Κάθε έκδοση πρέπει να έχει διαφορετικό πλήθος ή τύπο παραμέτρων. Όταν καλείται μια υπερφορτωμένη συνάρτηση, εκτελείται η έκδοση της συνάρτησης της οποίας οι παράμετροι ταιριάζουν με τα ορίσματα με τα οποία την καλούμε
- Στο παρακάτω παράδειγμα η συνάρτηση **test()** υπερφορτώνεται τρεις φορές:

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

void test(int x, int y);
void test(int x);
void test(double x);

int main()
{
		test(23);
		test(4,10);
		test(34.56);
		return 0;
}

//Πρώτη έκδοση της test()
void test(int x, int y)
{
		cout << "Πρώτη έκδοση της test()" << endl;
		cout << "Με δύο παραμέτρους int: " << x << "," << y << endl;
}
//Δεύτερη έκδοση της test()
void test(int x)
{
		cout << "Δεύτερη έκδοση της test()" << endl;
		cout << "Με μια παράμετρο int:" << x << endl;
}
//Τρίτη έκδοση της test()
void test(double x)
{
		cout << "Τρίτη έκδοση της test()" << endl;
		cout << "Με μια παράμετρο double:" << x << endl;
}
```

Υπερφόρτωση συναρτήσεων

- Σε μια υπερφορτωμένη συνάρτηση, κάθε έκδοση της μπορεί να εκτελεί εντελώς διαφορετικές λειτουργίες από τις υπόλοιπες εκδόσεις. Παρόλα αυτά, θα πρέπει να γίνεται προσπάθεια ώστε οι διαφορετικές εκδόσεις μιας υπερφορτωμένης συνάρτησης να επιτελούν παρόμοιες λειτουργίες
- Οι διαφορετικές εκδόσεις μιας υπερφορτωμένης συνάρτησης μπορεί να είναι διαφορετικού τύπου. Στο επόμενο πρόγραμμα η συνάρτηση **calc()** διαθέτει πέντε εκδόσεις και οι πέντε διαφορετικού τύπου

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

void calc();
char calc(int x);
int calc(int x, int y);
float  calc(double x, double y);
double calc(int x, int y, int z);

int main()
{
        cout << calc(5,6) << endl;
        cout << calc(65) << endl;
        cout << calc(5,6,12)
             << endl;
        cout << calc(10.5,6.26) <<
endl;
        calc();
        return 0;
}
```

```
void calc()
{
        int i;
        for (i=1;i<=2;i++)
    cout << "Υπερφόρτωση συναρτήσεων"
<< endl;
}

char calc(int x)
{
        int ch;
        return x+1;
}

int  calc(int x, int y)
{
        return x+y;
}

float  calc(double x, double y)
{
        return x/y;
}

double calc(int x, int y, int z)
{
        return (x+y+z)/3.0;
}
```

Συναρτήσεις με προκαθορισμένες τιμές παραμέτρων

- Η C++ μας δίνει την δυνατότητα στις παραμέτρους μια συνάρτησης να έχουν προκαθορισμένες τιμές
- Όταν κάποια από τις παραμέτρους μιας συνάρτησης έχει προκαθορισμένη τιμή, και η συνάρτηση κληθεί χωρίς το αντίστοιχο όρισμα για αυτή την παράμετρο, ως τιμή της παραμέτρου χρησιμοποιείται η προκαθορισμένη τιμή της
- Στο παρακάτω πρόγραμμα η συνάρτηση **mult()** καλείται 4 φορές με διαφορετικό πλήθος ορισμάτων κάθε φορά (μέχρι 4, αν κληθεί με 5 ορίσματα θα είναι λάθος)

```
#include <iostream>
using namespace std;
double mult(double a=0, double b=1, double c=1, double d=1)
{
        double ginomeno;
        ginomeno = a*b*c*d;;
        return ginomeno;
}
int main()
{
        cout << mult() <<endl;
        cout << mult(10) <<endl;
        cout << mult(10,2,3) <<endl;
        cout << mult(10,2,3,2) <<endl;
   return 0;
}
```

```
0
10
60
120
```

Συναρτήσεις με προκαθορισμένες τιμές παραμέτρων

- Στη περίπτωση που η συνάρτηση οριστεί μετά από τη main() τότε οι προκαθορισμένες τιμές των παραμέτρων της δεν δίνονται στον ορισμό της συνάρτησης αλλά στην προκαταβολική δήλωση της
- Από τη στιγμή που μια παράμετρος δηλωθεί με προκαθορισμένη τιμή, όλες οι παράμετροι που ακολουθούν πρέπει να έχουν προκαθορισμένες τιμές

```
#include <iostream>
using namespace std;
double mult(double a=0, double b=1, double c=1, double d=1);

int main()
{
        cout << mult() <<endl;
        cout << mult(10) <<endl;
        cout << mult(10,2,3) <<endl;
        cout << mult(10,2,3,2) <<endl;
   return 0;
}
double mult(double a, double b, double c, double d)
{
        double ginomeno;
        ginomeno = a*b*c*d;;
        return ginomeno;
}
```

Υπερφόρτωση συναρτήσεων και ασάφεια

- Ασάφεια (ambiguity) είναι η κατάσταση κατά την οποία ο μεταγλωττιστής δεν μπορεί να επιλέξει τη σωστή έκδοση της υπερφορτωμένης συνάρτησης που πρέπει να καλέσει (εμφάνιση μηνύματος σφάλματος)
- Ο μεταγλωττιστής της C++ ακολουθεί μια σειρά βημάτων για να μπορέσει να επιλέξεις τη σωστή έκδοση της συνάρτησης που πρέπει να κληθεί:

Υπερφόρτωση συναρτήσεων και ασάφεια

1. Εντοπίζει όλες τις συναρτήσεις που έχουν το ίδιο όνομα με αυτή που καλείται
2. Αποκλείει τις συναρτήσεις που δεν έχουν το σωστό πλήθος παραμέτρων σύμφωνα με τα ορίσματα της κλήσης
3. Αν το πλήθος των παραμέτρων δεν ταιριάζει για καμιά συνάρτηση διακόπτεται η μεταγλώττιση με μήνυμα λάθους
4. Αν υπάρχουν περισσότερες από μια συμφωνίες επιλέγεται η καλύτερη
5. Αν δεν υπάρχει δυνατότητα να επιλεγεί η καλύτερη συμφωνία ο μεταγλωττιστής παράγει ένα σφάλμα ασάφειας διακόπτοντας την μεταγλώττιση και εμφανίζοντας μήνυμα λάθους

Υπερφόρτωση συναρτήσεων και ασάφεια

- Στη περίπτωση όπου υπάρχουν αρκετές υποψήφιες εκδόσεις με τον ίδιο αριθμό παραμέτρων, για να εντοπιστεί η καλύτερη συμφωνία ακολουθούνται τα παρακάτω βήματα:

1. Γίνεται έλεγχος για ακριβή συμφωνία τύπων
2. Ο μεταγλωττιστής πραγματοποιεί προαγωγές τύπων στα ορίσματα ώστε να επιτύχει συμφωνία: οι τύποι char, unsigned char και short int προάγονται σε int. Ο τύπος float προάγεται σε double. Ο τύπος bool προάγεται σε int
3. Ο μεταγλωττιστής πραγματοποιεί μετατροπές τύπων για να υπάρξει συμφωνία παραμέτρων και ορισμάτων: οι ακέραιοι μετατρέπονται σε τύπο κινητής υποδιαστολής, μετατροπές μεταξύ τύπων κινητής υποδιαστολής, μετατροπές μεταξύ ακέραιων τύπων και κινητής υποδιαστολής σε bool

- Στο παρακάτω πρόγραμμα η συνάρτηση fn() υπερφορτώνεται 4 φορές :

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

void fn(int x);
void fn(long int x);
void fn(double x);
void fn(long double x);

int main()
{
        int i=12;
        char c='A';
        double d=12.23;
        float f=5.5;
        long double ld=1.345;
        fn(c);
        fn(f);
        fn(ld);
        fn((long int)i);
        return 0;
}
```

```
//1η έκδοση της fn()
void fn(int x)
{
  cout << "ver1 - int" << endl;
}

//2η έκδοση της fn()
void fn(long int x)
{
  cout << "ver2 - long int" << endl;
}
//3η έκδοση της fn()
void fn(double x)
{
  cout << "ver3 - double" << endl;
}

//4η έκδοση της fn()
void fn(long double x)
{
  cout << "ver4 - long double" << endl;
}
```

- Στη 1η κλήση ο τύπος char προάγεται σε int
- Στη κλήση της 3ης έκδοσης ο τύπος float προάγεται σε double

ver1 - int
ver3 - double
ver4 – long double
ver2 – long int

- Στη κλήση της 4ης έκδοσης υπάρχει ακριβής συμφωνία παραμέτρου ορίσματος
- Στη κλήση της 2ης έκδοσης το όρισμα μετατρέπεται σε long int με τη μέθοδο της ρητής μετατροπής τύπου