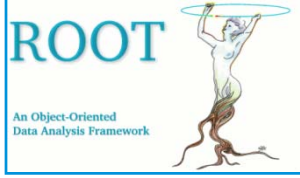
Στοιχειώδης προγραμματισμός σε C++

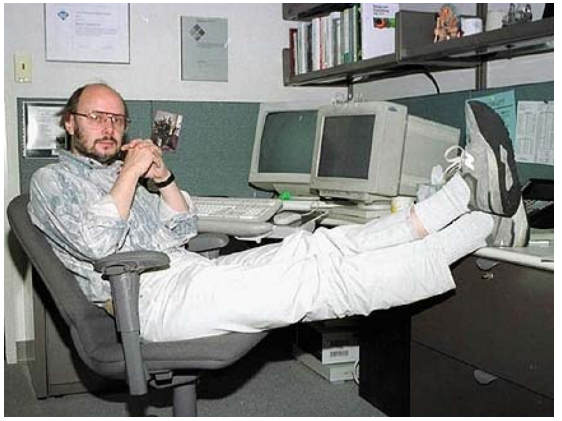
- Σύντομο Ιστορικό.
- Το πρόγραμμα “Hello World”.
- Ο τελεστής εξόδου.
- Μεταβλητές και δηλώσεις τους.
- Αντικείμενα, μεταβλητές, σταθερές.
- Ο τελεστής εισόδου.
- Θεμελιώδεις τύποι.
- Τελεστές.
- Εμβέλεια.

Σύντομο Ιστορικό



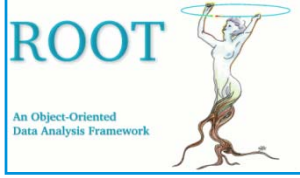
- Η **C++** είναι μια γενικού σκοπού γλώσσα προγραμματισμού Η/Υ.
- Θεωρείται μέσου επιπέδου γλώσσα, καθώς περιλαμβάνει έναν συνδυασμό χαρακτηριστικών από γλώσσες υψηλού και χαμηλού επιπέδου.
- Υποστηρίζει δομημένο, αντικειμενοστραφή και γενικό προγραμματισμό.

- Η γλώσσα αναπτύχθηκε από τον Bjarne Stroustrup το 1979 στα εργαστήρια Bell της AT&T, ως βελτίωση της ήδη υπάρχουσας γλώσσας προγραμματισμού C, και αρχικά ονομάστηκε "C with Classes", δηλαδή C με Κλάσεις.



- Μετονομάστηκε σε C++ το 1983. Οι βελτιώσεις ξεκίνησαν με την προσθήκη κλάσεων, και ακολούθησαν, μεταξύ άλλων, εικονικές συναρτήσεις, υπερφόρτωση τελεστών, πολλαπλή κληρονομικότητα, πρότυπα κ.α.
- Η γλώσσα ορίστηκε παγκοσμίως, το 1998, με το πρότυπο ISO/IEC 14882:1998.
- Η τρέχουσα έκδοση αυτού του προτύπου είναι αυτή του 2003, η ISO/IEC 14882:2003.

Το πρόγραμμα “Hello World”

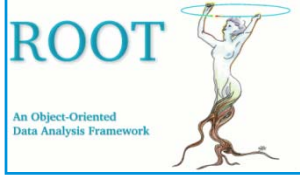



```
#include<iostream>

int main()
{
  std::cout << "Hello World!!!\n";
}
```

```
[panos@pc-247 Cpp]$ c++ hello_world01.cpp
[panos@pc-247 Cpp]$ a.out
Hello World!!!
[panos@pc-247 Cpp]$
```

- #include<iostream> Αποτελεί μια οδηγία στον *προεπεξεργαστή* η οποία λέει στον *μεταγλωττιστή* που θα βρει τον ορισμό του *αντικειμένου* std::cout
- ınt main() Αποτελεί την *κύρια συνάρτηση* του προγράμματος.
- std::cout Το αντικείμενο του *καθιερωμένου ρεύματος (stream) εξόδου* .

Το πρόγραμμα “Hello World”

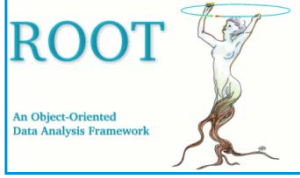



```
#include<iostream>
using namespace std;
int main()
{
  cout << "Hello World!!!\n";
  return 0;
}
```

```
[panos@pc-247 Cpp]$ c++ hello_world01.cpp
[panos@pc-247 Cpp]$ a.out
Hello World!!!
[panos@pc-247 Cpp]$
```

- using namespace std; λέει στον *μεταγλωττιστή* να εφαρμόσει το πρόθεμα std:: για να προσδιορίσει τα ονόματα που χρειάζονται προθέματα.
- Χρησιμοποίηση απ’ ευθείας του cout αντί του std::cout
- Το αντικείμενο cout ορίζεται στο χώρο ονομάτων με το όνομα std στο *αρχείο-κεφαλίδα* <iostream>

Ο τελεστής εξόδου



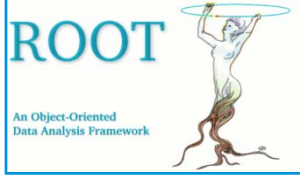
- Το σύμβολο << ονομάζεται *τελεστής εξόδου*.
- Ο τελεστής εξόδου << εκτελεί την ενέργεια της αποστολής της τιμής, της παράστασης που βρίσκεται στα δεξιά του, στο ρεύμα εξόδου που βρίσκεται στα αριστερά του.
- Εάν στο ρεύμα cout τοποθετηθούν διάφορα πράγματα, τοποθετούνται στη σειρά το ένα μετά το άλλο, με τη σειρά που τοποθετήθηκαν στο ρεύμα.

```
#include<iostream>
using namespace std;
int main()
{
  cout << "Hello" << " Worl" << "d!!!" << endl;
   return 0;
}
```

```
[panos@pc-247 Cpp]$ c++ hello_world03.cpp
[panos@pc-247 Cpp]$ a.out
Hello World!!!
[panos@pc-247 Cpp]$
```

- endl το αντικείμενο "τέλος γραμμής" ("end of line"). Έχει το ίδιο αποτέλεσμα με τον χαρακτήρα νέας γραμμής "\n" .

Μεταβλητές και δηλώσεις τους



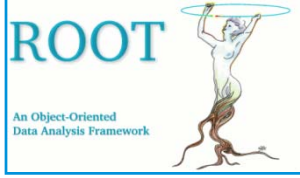
- Η μεταβλητή (variable) είναι ένα σύμβολο το οποίο αντιπροσωπεύει μία θέση αποθήκευσης στη μνήμη του υπολογιστή.
- Η πληροφορία που είναι αποθηκευμένη σε αυτή τη θέση ονομάζεται *τιμή* της μεταβλητής.
- Μια μεταβλητή συνήθως παίρνει τιμές μέσω *ανάθεσης* (assignment):

  μεταβλητή = παράσταση;
- Πρώτα υπολογίζεται η παράσταση και, στη συνέχεια, η τιμή που προκύπτει ανατίθεται στη μεταβλητή.

```
#include<iostream>
using namespace std;
int main()
{
  int m,n;

  m=10;
  cout << "m= " << m << endl;

  n=m*13;
  cout << "n= " << n << endl;

  return 0;
}
```

```
[panos@pc-247 Cpp]$ c++ variable.cpp
[panos@pc-247 Cpp]$ a.out
m= 10
n= 130
[panos@pc-247 Cpp]$
```

Αντικείμενα, μεταβλητές, σταθερές.



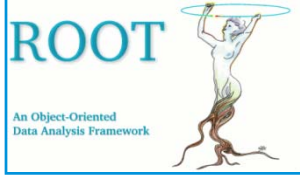
- Ένα αντικείμενο (object) είναι μια συνεχής περιοχή της μνήμης η οποία έχει διεύθυνση, μέγεθος, τιμή και τύπο.
  - Η *διεύθυνση* ενός αντικειμένου είναι διεύθυνση στη μνήμη του πρώτου byte.
  - Το *μέγεθος* είναι ο αριθμός των bytes που καταλαμβάνει το αντικείμενο στη μνήμη.
  - *Τιμή* είναι η σταθερά που καθορίζεται από τα bits που είναι αποθηκευμένα στη θέση της μνήμης.
  - Ο *τύπος* υπαγορεύει τον τρόπο με τον οποίο θα γίνει η ερμηνεία των bits.

- Για παράδειγμα ένα αντικείμενο το οποίο ορίζεται ως:

```
int m = 13;
```

  - Έχει διεύθυνση μνήμης πχ. 0x3fffcda6 (διαφορετική ανά εκτέλεση)
  - Μέγεθος 4 bytes
  - Τιμή 13
  - Τύπο int.

Αντικείμενα, μεταβλητές, σταθερές.

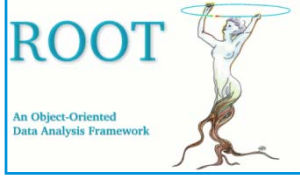


- Η λέξη μεταβλητή χρησιμοποιείται ώστε να υπονοείται πως η τιμή του αντικειμένου μπορεί να αλλάξει.
- Σταθερά (constant) ονομάζεται το αντικείμενο του οποίου η τιμή δεν μπορεί να αλλάξει.
- Παραδείγματα σταθερών:

```
const char BEEP = '\b';
const int MAXINT= 2147483647;
const int N=MAXINT/2;
const float KY_PER_MI = 1.60934;
const double PI=3.14159265358979
```

- Σταθερές ορίζουμε για τιμές όπως το π, που συνήθως χρησιμοποιούνται πολλές φορές σε ένα πρόγραμμα χωρίς να αλλάζουν.
- Για τις σταθερές συνήθως χρησιμοποιούνται κεφαλαίοι χαρακτήρες ώστε να διακρίνονται εύκολα.

Ο τελεστής εισόδου

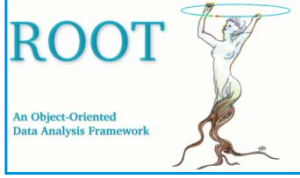


- Στη C++ η είσοδος είναι σχεδόν το ίδιο απλή με την έξοδο. Ο τελεστής εισόδου (input operator) ">>" λειτουργεί παρόμοια με τον τελεστή εξόδου.

```
#include<iostream>
using namespace std;
int main()
{
  int m,n;
  cout << "Dwse dyo akeraious: ";
  cin >> m >> n;
  cout << "m= " << m << ",  n= " << n << endl;


  double x,y,z;
  cout << "Dwse treis dekadikous arithmous: ";
  cin >> x >> y >> z;
  cout << "x= " << x << ",  y= " << y  <<  ",  z= " << z  << endl;


  char c1,c2,c3,c4;
  cout << "Dwse tesseris xaraktires: ";
  cin >> c1 >> c2 >> c3 >> c4;
  cout << "c1= " << c1 << ",  c2= " << c2  <<  ",  c3= " << c3 <<  ",  c4= " <<
}
```

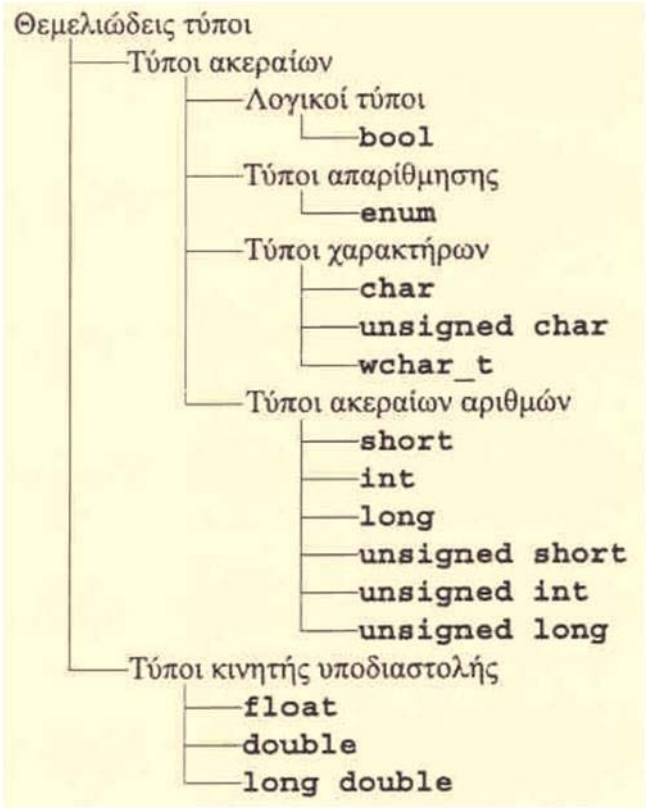
```
[panos@pc-247 Cpp]$ c++ eisodos.cpp
[panos@pc-247 Cpp]$ a.out
Dwse dyo akeraious: 6 89
m= 6,  n= 89
Dwse treis dekadikous arithmous: 5.6 7.89 123.65
x= 5.6,  y= 7.89,  z= 123.65
Dwse tesseris xaraktires: cdef
c1= c,  c2= d,  c3= e,  c4= f
[panos@pc-247 Cpp]$
```

Θεμελιώδεις τύποι



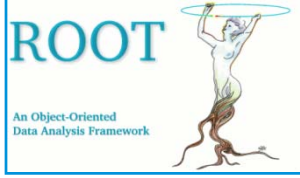
- Η καθιερωμένη C++ έχει 14 διαφορετικούς θεμελιώδεις τύπους : 11 τύπους ακεραίων και 3 τύπους κινητής υποδιαστολής.

- Οι τύποι των ακεραίων περιλαμβάνουν το λογικό (boolean) τύπο **bool**, τους τύπους απαρίθμησης **enum**, τρείς τύπους χαρακτήρων και έξι τύπους ακεραίων αριθμών.
- Οι τρείς τύποι κινητής υποδιαστολής είναι οι **float**, **double** και **long double**



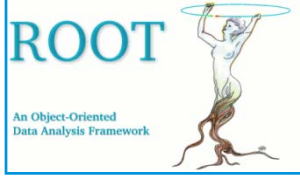
- Οι πιο συχνά χρησιμοποιούμενοι θεμελιώδεις τύποι είναι οι **bool**, **char**, **int** και **double**.

Σύνηθες εύρος τιμών



Τύπος	Μέγεθος σε bits	Τιμές
bool		false(0) ή true(1)
char	8-16	Χαρακτήρες ASCII - Unicode
short	16	-32768 έως 32767
int	32	-2146473648 έως 2147483647
long	64	± 9223372036854775807
float	32	$\pm 1.401298 \times 10^{-45}$ έως $\pm 3.402823 \times 10^{38}$
double	64	$\pm 4.9406565841246 \times 10^{-324}$ έως $\pm 1.7976931348623 \times 10^{308}$

Ο τύπος enum



- Εκτός από τους προκαθορισμένους τύπους όπως int και char, η C++ επιτρέπει να ορίσετε τους δικούς σας ειδικούς τύπους δεδομένων. Ο ποιο ισχυρός τρόπος είναι η χρήση κλάσεων (classes) που θα δούμε αργότερα.
- Ο τύπος *απαρίθμησης* (enumeration type) είναι ένας τύπος ακεραίων ο οποίος ορίζεται από τον χρήστη με την σύνταξη:

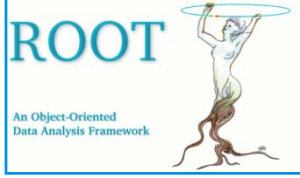
  enum *όνομα_τύπου* {*λίστα_απαριθμητών*}

- Απλό παράδειγμα:

```
enum Season {FALL, WINTER, SPRING, SUMMER};
Season s1,s2;
s1=FALL;
s2=SUMMER;
if(s1==s2)  cout  << “Same season” << endl;
```

Οι τιμές 0,1,2,3 ανατίθενται αυτόματα στα FALL, WINDER, SPRING, SUMMER κατά τον ορισμό του τύπου

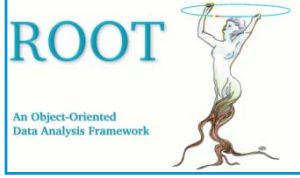
Τελεστές



- Οι **αριθμητικοί τελεστές** είναι οι ακόλουθοι:

Τελεστής	Περιγραφή
+	Πρόσθεση
-	Αφαίρεση
*	Πολλαπλασιασμός
/	Διαίρεση
%	Υπόλοιπο

Τελεστές



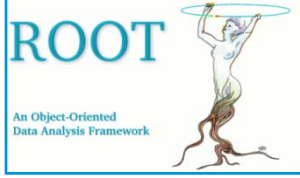
- Οι **συσχετιστικοί τελεστές** είναι οι ακόλουθοι:

Τελεστής	Περιγραφή
>	Μεγαλύτερο
>=	Μεγαλύτερο ή ίσο
<	Μικρότερο
<=	Μικρότερο ή ίσο

- Οι **τελεστές ισότητας** είναι οι ακόλουθοι:

Τελεστής	Περιγραφή
==	Ίσο με
!=	Άνισο με

Τελεστές

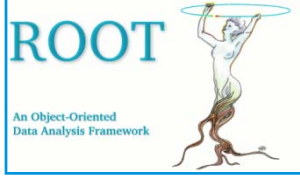


- Οι **λογικοί τελεστές** είναι οι ακόλουθοι:

Τελεστής	Περιγραφή
&&	Λογικός τελεστής AND
||	Λογικός τελεστής OR
!	Λογικός τελεστής NEGATION

- Τους συσχετιστικούς τελεστές, τους τελεστές ισότητας και τους λογικούς τελεστές τους συναντάμε κυρίως στις εντολές **if** , **for** , **while**, **do**.
- Οι παραπάνω τελεστές χρησιμοποιούνται για συγκρίσεις μεταξύ αριθμών, μεταβλητών και παραστάσεων.
- Εάν η σύγκριση είναι **αληθής** τότε το αποτέλεσμα είναι 1 διαφορετικά εάν είναι **ψευδής** τότε το αποτέλεσμα είναι μηδέν.

Τελεστές



- Ο **τελεστής αύξησης** και ο **τελεστής μείωσης** είναι οι ακόλουθοι:

Τελεστής	Περιγραφή
++	Τελεστής αύξησης κατά 1
--	Τελεστής μείωσης κατά 1

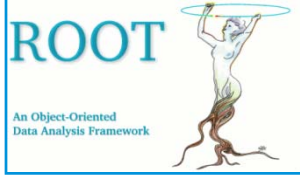
- Οι τελεστές ++ και -- χρησιμοποιούνται όταν θέλουμε να προσθέσουμε ή να αφαιρέσουμε το 1 από μία μεταβλητή. Έτσι

  το ++a; ισοδυναμεί με το a=a+1;

  ενώ το --a; ισοδυναμεί στο a=a-1;

- οι τελεστές ++ και -- μπορούν να χρησιμοποιηθούν είτε ως **προθεματικοί** τελεστές (δηλ. πριν την μεταβλητή, όπως ++a ή--a) είτε ως **επιθεματικοί** (δηλ. μετά την μεταβλητή, όπως a++ ή a--).

Τελεστές



- Στην παράσταση ++a η τιμή του a αυξάνει πριν χρησιμοποιηθεί η τιμή της.
- Στην παράσταση a++ η τιμή του a αυξάνει αφού χρησιμοποιηθεί η τιμή της.
- Παράδειγμα: Έτσι έστω ότι το a ισούται με 5 τότε η

```
a = 5;
```

τότε η παράσταση

```
b = a++;
```

δίνει στο b την τιμή 5 ενώ η παράσταση

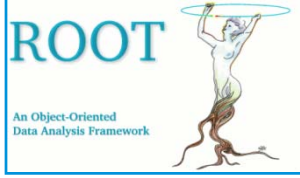
```
b=++a;
```

την τιμή 6. Το a και στις δύο περιπτώσεις γίνεται 6.

- Ο **τελεστής αντιστοίχησης** είναι ο:

Τελεστής	Περιγραφή
=	Τελεστής αντιστοίχησης

Τελεστές

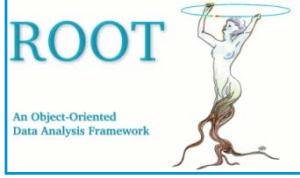


- Οι **τελεστές αντικατάστασης** είναι οι ακόλουθοι

Τελεστής	Περιγραφή
+=	Τελεστής πρόσθεσης και αντιστοίχησης
-=	Τελεστής αφαίρεσης και αντιστοίχησης
*=	Τελεστής πολ/μου και αντιστοίχησης
/=	Τελεστής διαίρεσης και αντιστοίχησης
%=	Τελεστής υπολοίπου και αντιστοίχησης

Το `a += b;` ισοδυναμεί με το `a = a+b;`
Το `a -= b;` ισοδυναμεί με το `a = a-b;`
Το `a *= b;` ισοδυναμεί με το `a = a*b;`
Το `a /= b;` ισοδυναμεί με το `a = a/b;`
Το `a %= b;` ισοδυναμεί με το `a = a %b;`

Τελεστές



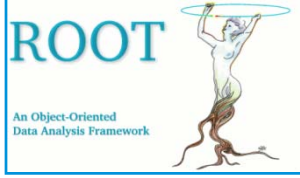
- Οι **τελεστές πράξεων με bits** είναι οι ακόλουθοι :

Τελεστής	Περιγραφή
&	AND για bit
|	OR για bit
^	XOR για bit
~	NOT για bit
>>	Ολίσθηση δεξιά
<<	Ολίσθηση αριστερά

- Οι παραπάνω τελεστές αφορούν πράξεις σε **επίπεδο bits.**
- Οι τελεστές &, |, ^ και ~ αντιστοιχούν στις απλές πράξεις της **άλγεβρας Boole**.
- Οι τελεστές >> και << προκαλούν ολίσθηση στα δεξιά και στα αριστερά αντίστοιχα.
  - Έτσι για παράδειγμα εάν η μεταβλητή a είναι ο δυαδικός αριθμός 01101000 τότε η παράσταση

    `b = a >> 2;` δίνει στη μεταβλητή b την τιμή 00011010.

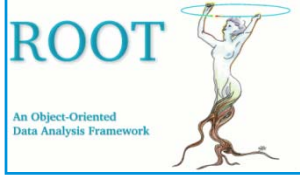
Εμβέλεια



- Η εμβέλεια (scope) ενός αναγνωριστικού είναι το τμήμα του προγράμματος στο οποίο μπορεί να χρησιμοποιηθεί αυτό.
- Για παράδειγμα, οι μεταβλητές δεν μπορούν να χρησιμοποιηθούν πριν δηλωθούν , και έτσι η εμβέλειά τους αρχίζει από τη θέση όπου δηλώνονται.

```
int main()
{ // δείχνει την εμβέλεια των μεταβλητών
   x=11;    //Λάθος: εκτός εμβέλειας της x
   int x;
   {    x=22;  //OK
        y=33;  // Λάθος: εκτός εμβέλειας της y
        int y;
        x=44;  //OK
        y=55;  //OK
   }
   x=66;   //OK
   y=77;   // Λάθος: εκτός εμβέλειας της y
}
```

Εμβέλεια



- Το ακόλουθο είναι ένα παράδειγμα ένθετης και παράλληλης εμβέλειας.

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

int x=11;   //Orismos katholikis metavlitis x

int main()
{
  int x=22;

  { // Eswteriko block
    int x=33;
    cout << "Mesa sto block tis main(): x= " << x << endl;
  }

  cout << "Mesa sti main(): x= " << x << endl;
  cout << "Mesa sti main(): ::x= " << ::x << endl;
}
```

```
[panos@pc-247 Cpp]$ c++ emvelia.cpp
[panos@pc-247 Cpp]$ a.out
Mesa sto block tis main(): x= 33
Mesa sti main(): x= 22
Mesa sti main(): ::x= 11
[panos@pc-247 Cpp]$
```

- Η τελευταία γραμμή χρησιμοποιεί τον τελεστή επίλυσης εμβέλειας (scope resolution operator) "::" για να προσπελάσει την καθολική μεταβλητή x η οποία διαφορετικά, είναι κρυμμένη για την main().