ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ

- Σε όλες τις γλώσσες προγραμματισμού υπάρχουν μερικές θεμελιώδεις αρχές που πρέπει να γνωρίζετε, πριν μπορέσετε να γράψετε ακόμη και τα πιο στοιχειώδη προγράμματα
  - Σε αυτό το μάθημα θα μιλήσουμε για βασικά θέματα όπως βασική κατασκευή προγράμματος, μεταβλητές και είσοδος-έξοδος
- Θα θεωρήσουμε ότι δεν έχετε κάποια εμπειρία με άλλη γλώσσα προγραμματισμού και υπενθυμίζουμε ότι για να μάθετε να προγραμματίζετε στη C++ δεν χρειάζεται να γνωρίζετε τη C
- Σας συνιστούμε να προμηθευτείτε ένα μεταγλωττιστή (compiler) της C++ , ώστε να κάνετε μόνοι σας εξάσκηση
- Να θυμάστε ότι μόνο μέσα από τη συνεχή ενασχόληση και τη διόρθωση των λαθών σας θα βελτιωθείτε

ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ

- Πρόγραμμα–Λογισμικό (Software)
  - Ένα σύνολο εντολών
  - Προσδιορίζει όλα τα βήματα που απαιτούνται για την επίλυση ενός προβλήματος εκφρασμένα σε μια γλώσσα που μπορεί να ερμηνεύσει το υλικό (hardware)

ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ

- Αλγόριθμος
  - Μια στρατηγική για την επίλυση ενός προβλήματος
  - Πρέπει να πληροί 3 βασικές προϋποθέσεις
    - σαφώς και απερίφραστα ορισμένος
    - αποτελεσματικός
      - Με την έννοια ότι τα βήματα του είναι εκτελέσιμα
    - πεπερασμένος
      - Με την έννοια ότι τερματίζεται μετά από ένα ορισμένο πλήθος βημάτων

ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ

- Η επίλυση προβλημάτων αποτελείται από 2 βήματα
  - Αλγοριθμική σχεδίαση
  - Κωδικοποίηση
- Η κωδικοποίηση γίνεται σε μια γλώσσα προγραμματισμού
  - C και C++ – γλώσσες υψηλού επιπέδου (higher level language)
  - Ανεξάρτητες από συγκεκριμένα χαρακτηριστικά ενός Η/Υ
    - δηλ. Τη γλώσσα μηχανής ενός υπολογιστή

ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ

- Μεταγλωττιστές (compilers)
  - Μεταφράζουν μια γλώσσα υψηλού επιπέδου στη γλώσσα μηχανής του συγκεκριμένου υπολογιστή

- Ένα αρχείο με πρόγραμμα σε μια γλώσσα προγραμματισμού- πηγαίος κώδικας (source code)
  - Μεταφράζεται σε ένα αντικειμενικό αρχείο (object file)
  - Συνδέεται με άλλα αντικειμενικά αρχεία ή βιβλιοθήκες
  - Δημιουργείται το εκτελέσιμο αρχείο (executable file)

- Κάθε γλώσσα προγραμματισμού έχει συντακτικούς κανόνες (syntax rules)
  - Τα λάθη που οφείλονται στην παραβίαση αυτών των κανόνων ονομάζονται συντακτικά λάθη (syntax errors)
  - Τα λάθη που οφείλονται στη λογική του προγράμματος ονομάζονται bugs και η διαδικασία αποσφαλμάτωσής τους debbugging

ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ

- Παράδειγμα: Παρακάτω παρατίθεται ο πηγαίος κώδικας για ένα απλό C πρόγραμμα. Κάντε Cut και paste τον κώδικα σε ένα ASCII αρχείο (π.χ. Με Notepad) και ονομάστε το hello.c

```
#include <stdio.h>

main()
{
    printf ("Hello\n");
}
```

- Η δήλωση #include <stdio.h> δηλώνει στο μεταγλωττιστή να περιλάβει τις δηλώσεις που βρίσκονται στο αρχείο-επικεφαλίδα stdio.h

ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ

❖ Το ίδιο παράδειγμα σε C++ (αρχείο hello.cpp)

```
#include <iostream>

int main()
{
	std::cout << "Hello" << std::endl;
	return 0;
}
```

ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ

- Τα προγράμματα αποτελούνται από εντολές οι οποίες γράφονται σε έναν απλό επεξεργαστή κειμένου, συνήθως στην Αγγλική γλώσσα
- Ο υπολογιστής όμως κατανοεί μόνο τη γλώσσα μηχανής (σειρές από 0 και 1)
- Υπάρχουν δύο τρόποι για να δημιουργηθεί γλώσσα μηχανής από τις εντολές του επεξεργαστή κειμένου
- Ο πρώτος είναι να μεταφραστεί ολόκληρο το πρόγραμμα σε γλώσσα μηχανής
  - Η τελευταία μπορεί να εκτελεστεί σε οποιοδήποτε υπολογιστή και δεν απαιτείται η εγκατάσταση της συγκεκριμένης γλώσσας προγραμματισμού
  - Το σύνολο των εντολών της γλώσσας προγραμματισμού, που αποκωδικοποιεί τις εντολές του επεξεργαστή κειμένου και τις μετατρέπει σε γλώσσα μηχανής λέγεται μεταγλωττιστής (compiler)
  - Όταν αγοράζουμε και εγκαθιστούμε μία γλώσσα προγραμματισμού ουσιαστικά αυτό που αγοράζουμε και εγκαθιστούμε είναι ο μεταγλωττιστής της
  - Η C++ είναι μία γλώσσα που έχει μεταφραστή και δημιουργεί αρχεία εκτελέσιμα τα οποία εκτελούνται και σε υπολογιστικά μηχανήματα που δεν υπάρχει η γλώσσα εγκατεστημένη.

ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ

- Ο δεύτερος τρόπος είναι να λαμβάνεται μία εντολή κάθε φορά, να ερμηνεύεται, να εκτελείται και η διαδικασία να επαναλαμβάνεται με την επόμενη εντολή
- Το σύνολο των εντολών της γλώσσας που ερμηνεύει κάθε εντολή του επεξεργαστή και την εκτελεί, λέγεται διερμηνευτής (interpreter)
- Ο διερμηνευτής λειτουργεί μέσα από το περιβάλλον της γλώσσας προγραμματισμού, για αυτό και η διαδικασία αυτή δεν μπορεί να λειτουργήσει σε υπολογιστές που δεν είναι εγκατεστημένη η γλώσσα
- Μία τέτοια γλώσσα που λειτουργεί με διερμηνευτή είναι η BASIC
- Θα πρέπει ωστόσο να αναφέρουμε ότι στην περίπτωση των μεταφραστών η δημιουργία εκτελέσιμων αρχείων είναι λίγο πιο σύνθετη
  - Οι εντολές που γράψαμε στον επεξεργαστή μεταφράζονται από τον μεταφραστή (compiler) και δημιουργείται ένα αρχείο το οποίο το ονομάζουμε "αντικείμενο πρόγραμμα" και έχει κατάληξη .OBJ (από τη λέξη object=αντικείμενο)
  - Αυτό το αρχείο το λαμβάνει ένας μηχανισμός που ονομάζεται συνδέτης (linker) και το συνδέει με αρχεία τα οποία είναι απαραίτητα στο να δημιουργηθούν τα εκτελέσιμα αρχεία και τα οποία ανήκουν στη γλώσσα προγραμματισμού
  - Έτσι δημιουργείται το εκτελέσιμο αρχείο με κατάληξη .EXE (από τη λέξη executable=εκτελέσιμο) το οποίο μπορεί να εκτελεστεί και σε υπολογιστές στους οποίους δεν υπάρχει η γλώσσα προγραμματισμού

ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ

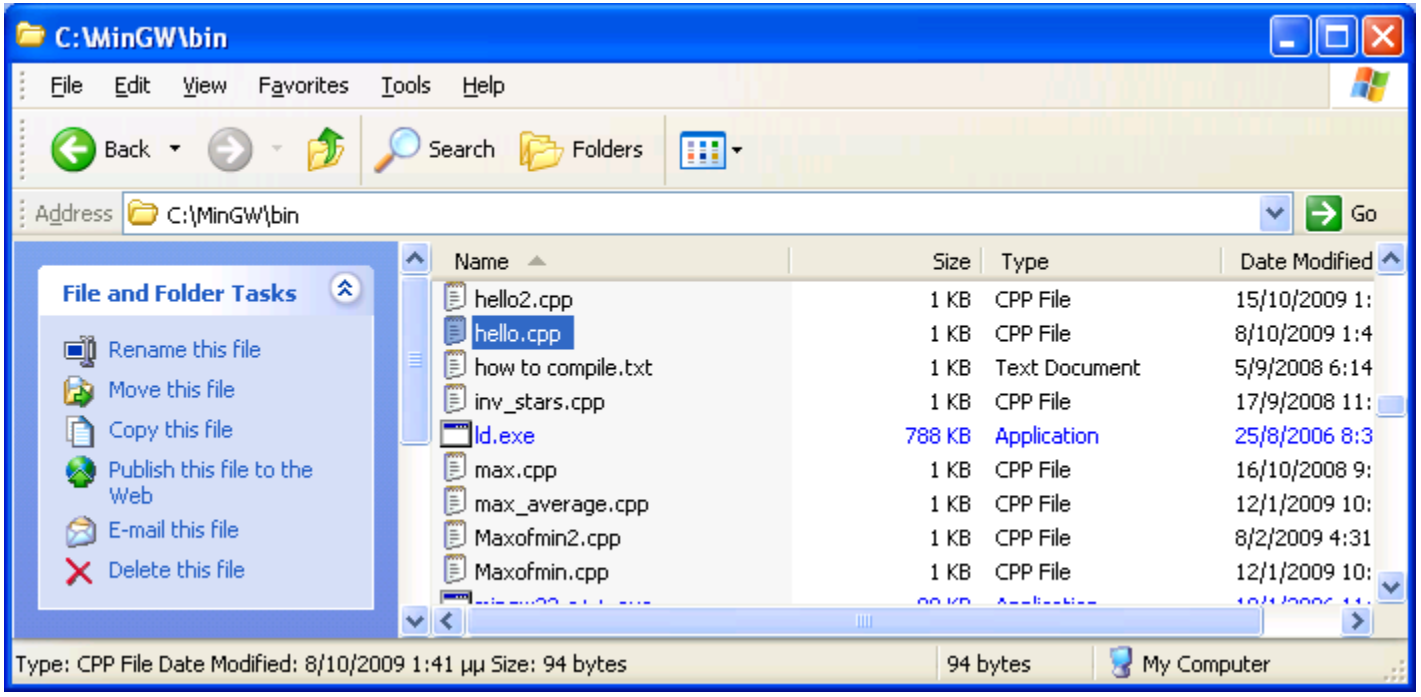
- Μεταγλώττιση του κώδικα όπως περιγράφεται στο εγχειρίδιο του εκάστοτε μεταγλωττιστή
  - Για gnu c compiler:
    - gcc hello.c –o hello (ή gcc –o hello hello.c)
      - Δημιουργεί εκτελέσιμο αρχείο με το όνομα hello.exe
    - gcc hello.c
      - Δημιουργεί εκτελέσιμο αρχείο με το όνομα a.exe
  - Για gnu c++ compiler:
    - g++ hello.cpp –o hello (ή g++ –o hello hello.cpp)
      - Δημιουργεί εκτελέσιμο αρχείο με το όνομα hello.exe
    - g++ hello.cpp
      - Δημιουργεί εκτελέσιμο αρχείο με το όνομα a.exe

ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ

- Μεταγλώττιση σε δύο βήματα
  - Για gnu c++ compiler:
    - g++ -c hello.cpp
      - Δημιουργεί το αντικειμενικό αρχείο με το όνομα hello.o
    - g++ -o hello hello.o (ή g++ hello.o -o hello)
      - Δημιουργεί εκτελέσιμο αρχείο με το όνομα hello.exe

ΠΑΡΑΔΕΙΓΜΑ ΜΕΤΑΓΛΩΤΙΣΣΗΣ ΜΕ ΤΟ MINGW ΣΕ WINDOWS

- Κάνουμε εγκατάσταση του μεταγλωττιστή MINGW
  - https://osdn.net/projects/mingw/releases/
- Δημιουργούμε το αρχείο με τον πηγαίο κώδικά μας
  - Προσοχή: το extension του αρχείου πρέπει να είναι .cpp
  - Π.χ. hello.cpp
- Αποθηκεύουμε το αρχείο στο directory bin του μεταγλωττιστή: c:\mingw\bin



ΠΑΡΑΔΕΙΓΜΑ ΜΕΤΑΓΛΩΤΙΣΣΗΣ ΜΕ ΤΟ MINGW ΣΕ WINDOWS

- Ανοίγουμε ένα παράθυρο command prompt
  - Start -> run -> cmd
- Αλλάζουμε directory σε c:\mingw\bin πληκτρολογώντας
  - cd c:\mingw\bin
- Μεταγλωτίζουμε πληκτρολογώντας
  - g++ hello.cpp
  - Δημιουργεί εκτελέσιμο αρχείο με το όνομα a.exe
- Εκτελούμε το εκτελέσιμο αρχείο πληκτρολογώντας
  - a.exe

ΜΕΤΑΓΛΩΤΙΣΣΗ ΣΕ ΠΕΡΙΒΑΛΛΟΝ MAC OS

- https://github.com/kennethreitz/osx-gcc-installer

- Κάνετε εγκατάσταση του gcc και μεταγλωττίζετε με τις εντολές που προ-αναφέρθησαν

- Εναλλακτικά, χρησιμοποιήστε το περιβάλλον ανάπτυξης λογισμικού (IDE) Xcode: https://developer.apple.com/xcode/

ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ

- Οι συναρτήσεις είναι ένα από τα θεμελιώδη συστατικά της C++
- Το πρόγραμμα αποτελείται σχεδόν όλο από μία συνάρτηση που ονομάζεται main ()
- Το μόνο μέρος του προγράμματος που δεν αποτελεί μέρος της main() είναι η πρώτη γραμμή που αρχίζει με το #include
- Όλες οι συναρτήσεις έχουν ένα όνομα που ακολουθείται από παρενθέσεις
  - Οι παρενθέσεις είναι το διακριτικό χαρακτηριστικό μιας συνάρτησης
  - Χωρίς αυτές ο μεταγλωττιστής θα νόμιζε ότι αναφερόμαστε σε μια μεταβλητή
  - Παρακάτω θα δούμε ότι οι παρενθέσεις δεν είναι πάντοτε κενές
  - Χρησιμοποιούνται για να περιέχουν τα ορίσματα της συνάρτησης, δηλαδή τιμές που μεταβιβάζονται στη συνάρτηση από το πρόγραμμα που τις καλεί

ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ

- Το σώμα μιας συνάρτησης περικλείεται σε άγκιστρα
  - Τα άγκιστρα οριοθετούν ένα τμήμα προτάσεων
    - καθορίζουν δηλαδή την αρχή και το τέλος των εντολών που περικλείονται στην συνάρτηση
  - Κάθε συνάρτηση πρέπει να χρησιμοποιεί αυτό το ζεύγος αγκίστρων

ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ

- Η συνάρτηση main ()
  - Ένα πρόγραμμα στη C++ μπορεί να περιέχει πολλές συναρτήσεις όμως θα πρέπει οπωσδήποτε να περιέχει μία που να ονομάζεται main()
  - Όταν εκτελείται ένα πρόγραμμα στη C++ η πρώτη εντολή που εκτελείται θα είναι στην αρχή μιας συνάρτησης που ονομάζεται main()
  - Εάν δεν έχετε συμπεριλάβει στο πρόγραμμά σας συνάρτηση που να ονομάζεται main() τότε ο συνδέτης (linker) δίνει σήμα ότι βρήκε σφάλμα

ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ

- Επιστροφή τιμών από συναρτήσεις
  - Όταν μία συνάρτηση ολοκληρώσει την εκτέλεσή της μπορεί να επιστρέψει μία μόνο τιμή στο πρόγραμμα που την κάλεσε
  - Αυτή η επιστρεφόμενη τιμή αποτελεί την απάντηση στο πρόβλημα που έλυσε η συνάρτηση και μπορεί να αποθηκευτεί σε μία μεταβλητή
- Είναι ίδια η χρήση των συναρτήσεων όπως μάθαμε στο σχολείο
  - Η συνάρτηση υπολογισμού του ημιτόνου sin() δέχεται ως όρισμα έναν αριθμό και όταν εκτελέσει τους υπολογισμούς της, επιστρέφει μία τιμή η οποία αποθηκεύεται στη μεταβλητή x
- Στη C++ θα πρέπει να είμαστε λίγο πιο σαφείς όταν ορίζουμε τη συνάρτηση: πριν το όνομά της δηλώνουμε τι τύπου θα είναι η επιστρεφόμενη τιμή
- Έτσι εάν επιστρέφεται ακέραια τιμή θα γράψουμε πριν το όνομα τη δεσμευμένη λέξη int, εάν επιστρέφεται πραγματικός αριθμός τη δεσμευμένη λέξη float κ.α.
- Μέσα στο σώμα της συνάρτησης θα πρέπει να υπάρχει μία εντολή με τη δεσμευμένη λέξη return και την τιμή που επιστρέφεται
- Όταν η main() επιστρέφει τιμή τότε αυτή περνά στο λειτουργικό σύστημα.

ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ

- Με τη λέξη void να προηγείται του ονόματος της συνάρτησης υποδηλώνουμε ότι αυτή η συνάρτηση δεν έχει τιμή επιστροφής. Στη C++ σε αντίθεση με άλλες γλώσσες όπου υπάρχει διαχωρισμός μεταξύ των συναρτήσεων (που επιστρέφουν τιμή που μπορεί να αποδοθεί) και των διαδικασιών (που εκτελούν κάποιες εντολές) υπάρχουν μόνο συναρτήσεις

```
#include <iostream>
int main()
{
   std:: cout << "Hello world" ;
   return 0;
}
```

ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ

- Στο πρώτο μας πρόγραμμα υπάρχει μόνο μία εντολή (πρόταση) να εκτελεστεί:

```
std::cout << "Hello world" ;
```

- Αυτή η πρόταση προγράμματος λέει στον υπολογιστή να εμφανίσει στην οθόνη τη φράση που βρίσκεται μέσα στα εισαγωγικά
- Όταν θέλουμε να εμφανιστεί ένα κείμενο (το οποίο ονομάζεται αλφαριθμητική σταθερά) στην οθόνη το περικλείουμε σε διπλά εισαγωγικά
- Το ελληνικό ερωτηματικό σηματοδοτεί το τέλος της πρότασης
  - Αυτός είναι ένας κανόνας που μπορεί πολύ εύκολα να ξεχάσετε
  - Σε μερικές γλώσσες όπως η BASIC το τέλος της πρότασης σηματοδοτείται από το τέλος της γραμμής
    - Αυτό δεν ισχύει όμως στη C++ που μπορείτε να σπάσετε την πρόταση σε όσες γραμμές θέλετε αρκεί στο τέλος να εισάγετε το ελληνικό ερωτηματικό που σηματοδοτεί το τέλος της πρότασης
  - Εάν παραλείψετε το ερωτηματικό ο μεταγλωττιστής θα εμφανίσει ένα μήνυμα σφάλματος

ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ

- Στο πρώτο μας πρόγραμμα υπάρχει μόνο μία εντολή (πρόταση) να εκτελεστεί:

```
std::cout << "Hello world" ;
```

- Αυτή η πρόταση προγράμματος λέει στον υπολογιστή να εμφανίσει στην οθόνη τη φράση που βρίσκεται μέσα στα εισαγωγικά
- Όταν θέλουμε να εμφανιστεί ένα κείμενο (το οποίο ονομάζεται αλφαριθμητική σταθερά) στην οθόνη το περικλείουμε σε διπλά εισαγωγικά
- Το ελληνικό ερωτηματικό σηματοδοτεί το τέλος της πρότασης
  - Αυτός είναι ένας κανόνας που μπορεί πολύ εύκολα να ξεχάσετε
  - Σε μερικές γλώσσες όπως η BASIC το τέλος της πρότασης σηματοδοτείται από το τέλος της γραμμής
    - Αυτό δεν ισχύει όμως στη C++ που μπορείτε να σπάσετε την πρόταση σε όσες γραμμές θέλετε αρκεί στο τέλος να εισάγετε το ελληνικό ερωτηματικό που σηματοδοτεί το τέλος της πρότασης
  - Εάν παραλείψετε το ερωτηματικό ο μεταγλωττιστής θα εμφανίσει ένα μήνυμα σφάλματος

ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ

- Ένα πρόγραμμα για να είναι χρήσιμο θα πρέπει να επικοινωνεί με τον έξω κόσμο
  - Να μπορεί να δέχεται δεδομένα από τον χρήστη και να εμφανίζει τα αποτελέσματα των υπολογισμών του
- Είπαμε παραπάνω ότι η πρόταση Hello world εμφανίζεται στην οθόνη του υπολογιστή
- Αυτό γίνεται με τη βοήθεια του αναγνωριστικού:

```
std::cout
```

- Το cout ουσιαστικά είναι ένα αντικείμενο της κλάσης class ostream που αποτελεί το standard output stream
  - Έχει προκαθοριστεί από τη C++ να είναι το ρεύμα καθιερωμένης εξόδου
  - Εμάς όμως προς το παρόν δεν μας ενδιαφέρουν αυτά, εμείς θα γνωρίζουμε ότι με τη χρήση του cout εμφανίζουμε δεδομένα στη οθόνη
  - Ο τελεστής << ονομάζεται τελεστής εξαγωγής και κατευθύνει τα δεδομένα που βρίσκονται στα δεξιά του προς το αντικείμενο που είναι στα αριστερά του, το cout.
- Για να εισάγουμε δεδομένα στο πρόγραμμά μας χρησιμοποιούμε το αντικείμενο cin
  - Ο τελεστής >> ονομάζεται τελεστής εισαγωγής και κατευθύνει τα δεδομένα που δέχεται το αντικείμενο cin μέσα στο πρόγραμμα. Πχ

```
std::cin >> x ;
```

- Με την παραπάνω εντολή σταματά η ροή εκτέλεσης του προγράμματος αναμένοντας από τον χρήστη να δώσει μία τιμή από το πληκτρολόγιο
  - Δίνοντας την τιμή και πατώντας το enter επιστρέφεται ο έλεγχος πάλι στο πρόγραμμα αποθηκεύεται η τιμή στη μεταβλητή x και συνεχίζεται η εκτέλεσή του από εκεί και κάτω

ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ

- Με τη χρήση ενός cout μπορούμε να εμφανίσουμε πολλά δεδομένα στην οθόνη χρησιμοποιώντας πολλές φορές τον τελεστή εξαγωγής <<

```
std:: cout << "hello world "<< "this is my first program";
```

- Όπως φαίνεται τα δύο αλφαριθμητικά εμφανίζονται στην ίδια γραμμή. Για να αλλάξω γραμμή θα πρέπει να γράψω τη λέξη endl
  - Η λέξη endl είναι ένας χειριστής ο οποίος εισάγει μία αλλαγή γραμμής

```
std:: cout << "hello world "<<std::endl<< "this is my first program" ;
```

- Το std:: δε χρειάζεται σε μερικούς compliers

ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ

- Η πρώτη γραμμή του προγράμματος μπορεί να μοιάζει με πρόταση προγράμματος αλλά δεν είναι
  - Δεν αποτελεί μέρος του σώματος κάποιας συνάρτησης και δεν καταλήγει με το ελληνικό ερωτηματικό
  - Αρχίζει με το σύμβολο # και είναι μια οδηγία προεπεξεργαστή
  - Ο προεπεξεργαστής είναι ένα μέρος του μεταγλωττιστή και χειρίζεται τις οδηγίες αυτές πριν ξεκινήσει η μεταγλώττιση

- Η οδηγία #include λέει στον μεταγλωττιστή να παρεμβάλει ένα άλλο αρχείο (το αρχείο iostream) μέσα στον κώδικά μας
  - Στην πράξη γίνεται αντικατάσταση της οδηγίας #include με τα περιεχόμενα του αρχείου
  - Το iostream χρειάζεται για να αναγνωρίσει η γλώσσα τα αντικείμενα εισόδου και εξόδου cin και cout και να ξέρει πως θα τα χειριστεί για να πραγματοποιηθεί είσοδος και έξοδος δεδομένων από το πρόγραμμα

ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ

- Τα σχόλια είναι ένα σημαντικό τμήμα του προγράμματος γιατί βοηθούν αυτόν που θα διαβάσει τον κώδικα του προγράμματος να καταλάβει τι συμβαίνει
  - Τα σχόλια αρχίζουν με το διπλό σύμβολο της καθέτου // και τελειώνουν με το τέλος της γραμμής
  - Όταν θέλετε τα σχόλια να ξεπεράσουν την μία γραμμή μπορείτε να δηλώσετε την αρχή των σχολίων με το σύμβολο /* και το τέλος τους με το σύμβολο */
- Να χρησιμοποιείται συχνά σχόλια γιατί δεν θα είναι καθόλου περίεργο εάν προσπαθείτε να διαβάσετε ένα πρόγραμμα δικό σας που το γράψατε πριν τρεις μήνες και να μη θυμάστε τι ακριβώς κάνετε μέσα στον κώδικά σας
  - Και κάτι σημαντικό, τα σχόλια δεν θα πρέπει να λένε τι κάνουμε στον κώδικα αλλά κυρίως γιατί το κάνουμε

ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ

- Οι μεταβλητές είναι το πιο θεμελιώδες μέρος οποιαδήποτε γλώσσας
  - Μία μεταβλητή είναι ένα συμβολικό όνομα που μπορεί να δεχτεί διάφορες τιμές
  - Με τη βοήθεια των μεταβλητών εισάγονται δεδομένα στο πρόγραμμα και σε αυτές αποθηκεύονται τα αποτελέσματα των υπολογισμών για να εμφανιστούν στην οθόνη ή να χρησιμοποιηθούν στη συνέχεια του προγράμματος

- Μία μεταβλητή χαρακτηρίζεται από τον τύπο των δεδομένων που περιέχει
  - Οι περισσότερες γλώσσες χρησιμοποιούν τους ίδιους γενικούς τύπους δεδομένων, όπως οι ακέραιοι , οι αριθμοί κινητής υποδιαστολής, οι χαρακτήρες κ.α.
  - Για να δηλώσετε τι τύπου δεδομένα θα περιέχει μία μεταβλητή πριν το όνομά της θα γράψετε την κατάλληλη δεσμευμένη λέξη

ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ

- Ως όνομα μπορείτε να δώσετε όποιο όνομα επιθυμείτε αλλά όχι κάποια δεσμευμένη λέξη από τη C++
  - Να είστε προσεχτικοί με την χρήση των ονομάτων των μεταβλητών
  - Θα πρέπει όπως ακριβώς τη δηλώσετε έτσι ακριβώς να την καλείτε
  - Η C++ είναι ευαίσθητη στη χρήση μικρών ή κεφαλαίων χαρακτήρων σε αντίθεση με την Basic
  - Έτσι εάν μία μεταβλητή τη δηλώσαμε με μικρούς χαρακτήρες δεν μπορούμε να την χρησιμοποιήσουμε στη συνέχεια με κεφαλαία

ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ

- Οι ακέραιες μεταβλητές αποθηκεύουν ακέραιους αριθμούς
- Για να ορίσετε μία ακέραια μεταβλητή θα πρέπει πριν από το όνομά της να γράψετε τη δεσμευμένη λέξη int. Για παράδειγμα:

```
#include <iostream>
int main()
{
	// declaring variables
	int aker1 ;
	int aker2 ;

	// process
	aker1 = 10 ;
	aker2 = aker1+1 ;

	// print out the result
	std::cout << "H aker2 einai "<< aker2 ;
	return 0;
}
```

- Στο πρόγραμμά μας οι προτάσεις:

```
int aker1 ;
int aker2 ;
```

- ορίζουν δύο ακέραιες μεταβλητές τις aker1 και aker2

- Όπως είπαμε, ως όνομα της μεταβλητής μπορείτε να διαλέξετε οποιοδήποτε, αρκεί να μην είναι δεσμευμένη λέξη της C++
- Είναι όμως καλή τεχνική τα ονόματα των μεταβλητών να περιγράφουν τα δεδομένα που είναι αποθηκευμένα σε αυτές
- Κάποιοι προγραμματιστές χρησιμοποιούν μόνο μικρούς χαρακτήρες στα ονόματα των μεταβλητών, άλλοι πάλι χρησιμοποιούν ένα μίγμα μικρών κεφαλαίων φτιάχνοντας σύνθετες λέξεις με την αρχή κάθε λέξης να είναι κεφαλαίος χαρακτήρας ώστε να ξεχωρίζουν οπτικά οι λέξεις (π.χ. DateOfBirth)

ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ

- Εντολές απόδοσης τιμής

- Οι προτάσεις:

  aker1 = 10 ;
  aker2 = aker1+1 ;

- αποδίδουν ή αλλιώς εκχωρούν τιμές στις δύο μεταβλητές aker1, aker2
- Τα σημείο ίσον (=) είναι μία εντολή, η οποία αφού κάνει τις πράξεις στο δεξιό μέλος αποδίδει το αποτέλεσμα στο αριστερό μέλος
- Έτσι στην πρώτη πρόταση αποδίδει την τιμή 10 στη μεταβλητή aker1
  - ενώ στη δεύτερη (ξεκινώντας πάντα από το δεξιό μέλος) παίρνει την τιμή της μεταβλητής aker1 που είναι 10 λόγω της προηγούμενης εντολής της, προσθέτει τη σταθερά 1 και το αποτέλεσμα 11 το αποδίδει στη μεταβλητή aker2

ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ

- Αρχικοποίηση τιμών μεταβλητών

- Όταν ορίζουμε μία μεταβλητή όπως στην πρόταση:

- int aker1 ;

- δεσμεύουμε χώρο στη μνήμη για να συγκρατεί τα δεδομένα της μεταβλητής, όμως στο σημείο αυτό δεν της αποδίδουμε κάποια τιμή
- Μία μεταβλητή η οποία έχει οριστεί αλλά δεν της έχει αποδοθεί κάποια τιμή μπορεί να έχει οποιαδήποτε τιμή, ανάλογα με τα δεδομένα που υπήρχαν στο κομμάτι της μνήμης που δεσμεύτηκε από την μεταβλητή
- Οι τιμές αυτές ονομάζονται συχνά «σκουπίδια».

- Εάν θέλουμε να της αποδώσουμε κάποια τιμή, τη στιγμή της δήλωσής της μπορούμε να το κάνουμε:

- int aker1=10 ;

Παράδειγμα: Πρόσθεση ακεραίων

```
#include <iostream>

int main()
{
    int x, y, total;
    x = 1;
    y = 2;
    total = x + y;
    std::cout << "To athroisma tou " << x << " me ton " << y << " einai " << total <<
    std::endl;
    return 0;
}
```

ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ

- Μεταβλητές χαρακτήρων
- Ένας άλλος τύπος ακέραιας μεταβλητής είναι ο char
- Ο τύπος αυτός αποθηκεύει δεδομένα δεσμεύοντας 1 byte μνήμης (8 bits)
- Οι μεταβλητές τύπου char χρησιμοποιούνται για να αποθηκεύουν χαρακτήρες ASCII όπως οι ‘a’, ‘b’, ‘C’, ‘3’ κ.λ.π.
- Να θυμάστε ότι, όταν αποδίδουμε σταθερές χαρακτήρων σε μία μεταβλητή τύπου char τότε τις περικλείουμε σε μονά εισαγωγικά
- Τα αλφαριθμητικά που αποτελούνται από περισσότερους από έναν χαρακτήρες περικλείονται σε διπλά εισαγωγικά.
- Εάν παραλείψουμε τα εισαγωγικά τότε η γλώσσα θα υποθέσει ότι χρησιμοποιήσαμε το όνομα μιας μεταβλητής.

```
#include <iostream>
int main()
{
    // declaring and initializing variables
    char xar1 = ‘A’ ;
    int x = 10 ;

    // print out
    std::cout << xar1 << std::endl << ‘x’ <<
        std::endl << x << "hello world" ;
    return 0;
}
```

ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ

- Μεταβλητές κινητής υποδιαστολής

- Ο τύπος int που είδαμε μέχρι τώρα αποθηκεύει ακέραιους αριθμούς, χωρίς κλασματικό μέρος
- Τώρα θα εξετάσουμε τις μεταβλητές κινητής υποδιαστολής (floating point) οι οποίες αποθηκεύουν αριθμούς με υποδιαστολή
- Οι αριθμοί αυτοί έχουν ένα ακέραιο μέρος στα αριστερά της υποδιαστολής και ένα κλασματικό μέρος στα δεξιά της. Πχ 3.1415927 -10.5
- Στη C++ υπάρχουν τρία είδη μεταβλητών κινητής υποδιαστολής: τύπου float, τύπου double και τύπου long double.

ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ

Τύπος	Bytes	Περιγραφή	Εύρος
char	1	Χαρακτήρες ή ακέραιοι 8 bits	**signed**: -128 έως127 **unsigned**: 0 έως 255
short	2	Ακέραιοι 16 bits μήκους.	**signed**: -32768 έως 32767 **unsigned**: 0 έως 65535
long	4	Ακέραιοι 32 bits μήκους.	**signed**:-2147483648 έως 2147483647 **unsigned**: 0 έως 4294967295
int		Ακέραιοι. Το μήκος τους εξαρτάται από τον τύπο Word του συστήματος, έτσι στο MSDOS έχουν 16 bits μήκος, ενώ σε συστήματα 32 bit (όπως Windows 9x/2000/NT) έχουν 32 bits long (4 bytes).	Δείτε **short**, **long**
float	4	Αριθμοί κινητής υποδιαστολής.	3.4e + / - 38 (7 digits)
double	8	Διπλής ακριβείας αριθμοί κινητής υποδιαστολής.	1.7e + / - 308 (15 digits)
long double	10	Μεγάλοι διπλής ακριβείας αριθμοί κινητής υποδιαστολής.	1.2e + / - 4932 (19 digits)
bool	1	Λογικές τιμές, μπορεί να πάρει δύο τιμές (true ή false) ΣΗΜΕΙΩΣΗ: είναι ένα τύπος που πρόσφατα προστέθηκε από την ANSI-C++ standard. Δεν τον υποστηρίζουν πολλοί μεταγλωττιστές.	true ή false

ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ

- Ο προσδιορισμός const
- Εάν θέλουμε η τιμή μιας μεταβλητής να μην αλλάξει κατά τη διάρκεια εκτέλεσης του προγράμματος από δικό μας τυχόν λάθος τότε πριν από τη λέξη που εκφράζει τον τύπο δεδομένων της μεταβλητής εισάγουμε τον προσδιορισμό const
  - από τη λέξη constant, σταθερά
- Στο παράδειγμα η μεταβλητή τύπου float με όνομα PI παραμένει σταθερή στη διάρκεια εκτέλεσης του προγράμματος
- Επίσης κάτι άλλο που ίσως προσέξατε είναι η δήλωση της μεταβλητής τύπου float με όνομα embado και η αρχικοποίησή της όχι στην αρχή του προγράμματος αλλά στο τέλος του.
- Αυτή είναι μία διαφορά της C++ σε σχέση με τη C όπου όλες οι μεταβλητές δηλώνονταν στην αρχή
- Αυτό είναι κάτι που μας δίδει ελευθερία αλλάζοντας όμως και την εμβέλεια των μεταβλητών, ένα θέμα για το οποίο θα μιλήσουμε αμέσως μετά

```
#include <iostream>
int main()
{

    //Δήλωση μεταβλητών
    float rad ; //Μεταβλητή τύπου float
    const float PI=3.14; //σταθερά τύπου
        float

    //προτροπή εισαγωγής δεδομένων
    std:: cout<< “Δώσε την ακτίνα του
        κύκλου “;
    std:: cin>> rad ; //εισαγωγή δεδομένων

    //Δήλωση και απόδοση αρχικής τιμής
    float embado = PI * rad * rad ;

    std::cout<< "Εμβαδόν= "<< embado <<
        std::endl;
    return 0;
}
```

Παράδειγμα: Πρόσθεση ακεραίων

```
#include <iostream>

int main()
{
    int x, y, total;
    std::cout << "Dwse to x ";
    std::cin >> x;
    std::cout << "Dwse to y ";
    std::cin >> y;
    total = x + y;
    std::cout << "To athroisma tou " << x << " me ton " << y << " einai " << total <<
    std::endl;
    return 0;
}
```

Παράδειγμα: Ύψωση σε δύναμη

```
#include <iostream>

int main()
{
    int x, y, value, i;
    std::cout << "Dwse thn bash: ";
    std::cin >> x;
    std::cout << "Dwse thn dynamh: ";
    std::cin >> y;
    value = 1;
    for (i = 0; i < y; i++) value = value * x;
    std::cout << x << " eis tin " << y << " ison " << value;
    return 0;
}
```

Παράδειγμα: Παραγοντικό

```
#include <iostream>

int main()
{
    int x, y, i;
    std::cout << "Dwse ton arithmo: ";
    std::cin >> x;
    y = 1;
    for (i = 1; i <= x; i++) y = y * i;
    std::cout << " To paragontiko tou " << x << " einai to " << y;
    return 0;
}
```

Παράδειγμα: Πίνακας προπαίδειας

```
#include <iostream>

int main()
{
    int i, j;
    for (i=1;i<=10;i++)
    {
        for (j=1; j<=10; j++)  std::cout << i*j <<std::endl;
    }
    return 0;
}
```

Παράδειγμα: Πίνακας προπαίδειας (καλύτερο)

```
#include <iostream>

int main()
{
    int i, j;
    for (i=1;i<=10;i++)
    {
        for (j=1; j<=10; j++) std::cout << i << " epi " << j << " kanei " << i*j <<
    std::endl;
    }
    return 0;
}
```

Παράδειγμα: slope

```
^
^^
^^^
^^^^
```

```
#include <iostream>
using namespace std;
int main()
{
    int i,j;
    for (i=1; i<=4;i++)
    {
        for (j=1; j<=i; j++) cout << "^";
        cout << endl;
    }
    return 0;
}
```

Παράδειγμα: X-mass tree

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

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

int main()
{
    int i,j;
    for (i=1; i<=4;i++)
    {
        for (j=1; j<=4-i; j++) cout << " ";
        for (j=1; j<=2*i-1; j++) cout << "*";
        cout << endl;
    }
    return 0;
}
```

Βρόχος for

- Ο βρόχος for χρησιμοποιείται όταν γνωρίζουμε από πριν τον αριθμό των επαναλήψεων. Όταν γνωρίζουμε πριν ακόμη μπούμε στον βρόχο, τον αριθμό των επαναλήψεων του κώδικα.

- Ο βρόχος for συντάσσεται ως εξής:

```
for (i=1; i<10; i++) εντολή;
ή
for (i=1; i<10; i++)
{
    εντολή1;
    εντολή2;
}
```

Βρόχος for

- Μετά τη δεσμευμένη λέξη for ακολουθούν μέσα σε παρένθεση οι οδηγίες εκτέλεσης του βρόχου
- Μία μεταβλητή παίζει το ρόλο του μετρητή των επαναλήψεων
- Η μεταβλητή-μετρητής λαμβάνει αρχική τιμή
  - Αυτή η εντολή εκτελείται μόνο την πρώτη φορά, στην αρχή του βρόχου
- Ακολουθεί μία λογική συνθήκη
  - Όσο η λογική συνθήκη είναι αληθής ο βρόχος θα επαναλαμβάνει τις εντολές που περιέχει
  - Όταν η λογική συνθήκη γίνει ψευδής, ο έλεγχος του προγράμματος μεταπηδά έξω από το βρόχο και συνεχίζει με το υπόλοιπο πρόγραμμα
  - Η εξέταση της λογικής συνθήκης γίνεται πριν ο έλεγχος του προγράμματος εισέλθει στο βρόχο
- Η τελευταία παράμετρος μέσα στις παρενθέσεις είναι η αύξηση του μετρητή
- Όταν θέλουμε ο βρόχος να εκτελεί περισσότερες από μία εντολές τότε αυτές τις εσωκλείουμε μέσα σε άγκιστρα

Παράδειγμα: εύρεση μέγιστου αριθμού

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

int main()
{
    int a, b, max;
    cout << "Dwste ton 1o arithmo " << endl;
    cin >> a;
    cout << "Dwste ton 2o arithmo " << endl;
    cin >> b;
    if (a > b) max = a;
    else max =b;
    cout << "O megaliteros aritmos einai to " << max << endl;
    return 0;
}
```

ΠΡΟΤΑΣΗ if

- Η δεσμευμένη λέξη if ακολουθείται από μία λογική συνθήκη σε παρενθέσεις

  if (Λογική Συνθήκη) εντολή;

- Σε περίπτωση που η λογική συνθήκη είναι αληθής τότε θα εκτελεστεί η εντολή
  - Διαφορετικά ο έλεγχος του προγράμματος θα συνεχιστεί παρακάτω και από την εντολή if δεν θα εκτελεστεί καμία εντολή ως αποτέλεσμά της
- Σε περίπτωση που θέλουμε με την επαλήθευση μιας λογικής συνθήκης να εκτελεστούν παραπάνω από μία εντολές, τότε τις περικλείουμε σε άγκιστρα

```
if (Λογική Συνθήκη)
{
        εντολή1;
        εντολή2;
        ...
}
```

ΠΡΟΤΑΣΗ if

- Εάν θέλουμε όμως να γίνεται κάτι, όταν η λογική συνθήκη είναι ψευδής τότε αυτό θα το περιλάβουμε μετά τη λέξη else
- Έτσι η γενική μορφή της εντολής ελέγχου if είναι:

```
if (Λογική Συνθήκη)
  εντολή1;
else
  εντολή2;
```

ΠΡΟΤΑΣΗ if

- Πολλές φορές χρειάζεται να ελέγξουμε ένα εύρος τιμών για μια μεταβλητή και δεν είμαστε ικανοποιημένοι με τις δύο περιπτώσεις (true ή false) της λογικής συνθήκης
- Έτσι τοποθετούμε προτάσεις if-else μέσα στην αρχική πρόταση
- Το ακόλουθο παράδειγμα ελέγχει την τιμή της μεταβλητής x

```
if (x > 0)
  cout << "x is positive";
else if (x < 0)
  cout << "x is negative";
else
  cout << "x is 0";
```

- Ισοδύναμο με:

```
if (x > 0)
  cout << "x is positive";
else
    if (x < 0)
      cout << "x is negative";
    else
      cout << "x is 0";
```

Παράδειγμα

- Το επόμενο παράδειγμα ελέγχει εάν οι τρεις μεταβλητές έχουν την ίδια τιμή.

```
#include <iostream>
using namespace std;
int main()
{
  //Δήλωση μεταβλητών
  int a, b, c; //Μεταβλητή τύπου int
  //προτροπή εισαγωγής δεδομένων
  cout<< "Δώστε a, b και c "<<endl;
  cin>> a >> b >> c; //εισαγωγή δεδομένων
  if (a==b)
    if (b==c)
      cout<< "τα a, b και c είναι ίδια";
  else
      cout<< "τα a και b είναι διαφορετικά";
  return 0;
}
```

- Παρατηρήστε την λάθος αντιστοίχηση του τελευταίου else
  - Το else αντιστοιχεί στο δεύτερο if !!!
  - Το μήνυμα «τα a και b είναι διαφορετικά» εμφανίζεται όταν το b είναι διαφορετικό του c
  - Όμως για να φτάσει ο έλεγχος στο δεύτερο if, σημαίνει πως η λογική συνθήκη του πρώτου if είναι αληθής (δηλαδή a ίσο με b)
  - Το σωστό είναι:

```
if (a==b)
{
   if (b==c)
      cout<< "τα a, b και c είναι ίδια";
}
else
      cout<< "τα a και b είναι διαφορετικά";
```

Παράδειγμα: εύρεση μέγιστου αριθμού (καλύτερο)

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

int main()
{
    int a, b, max;
    cout << "Dwste ton 1o arithmo " << endl;
    cin >> a;
    cout << "Dwste ton 2o arithmo " << endl;
    cin >> b;
    max = (a > b) ? a : b;
    cout << "O megaliteros aritmos einai to " << max << endl;
    return 0;
}
```

Παράδειγμα: άθροιση αριθμών μέχρι να εισαχθεί το 0

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

int main()
{
	int sum, a;
	cout << "Dwse enan arithmo :" << endl;
	cin >> a;
	sum = 0;
	while (a != 0)
	{
		sum = sum + a;
		cout << "Dwse enan arithmo :" << endl;
		cin >> a;
	}
	cout << "To athroisma olwn twn arithmwn einai :" << sum << endl;
	return 0;
}
```

Βρόχος while

- Ο Βρόγχος while χρησιμοποιείται όταν θέλουμε να επαναλαμβάνονται κάποιες εντολές, χωρίς να ξέρουμε από πριν τον αριθμό των επαναλήψεων
- Ο βρόχος βασίζεται σε μία λογική συνθήκη, ο έλεγχος της οποίας καθορίζει εάν θα επαναλάβουν ακόμη έναν κύκλο ή όχι
- Ο βρόχος while συντάσσεται ως εξής

```
while (i!=0)
    εντολή;
```

ή

```
while (i!=0)
{
    εντολή1;
    εντολή2;
}
```

Βρόχος while

- Για να επαναλάβει τις εντολές του ο βρόχος while ελέγχει τη λογική συνθήκη και εάν αυτή είναι αληθής τότε εισέρχεται στο σώμα του και επαναλαμβάνει τις εντολές που του έχουμε ορίσει
- Όταν εκτελέσει όλες τις εντολές του ο έλεγχος μεταβιβάζεται πάλι στην αρχή και επανεξετάζεται η λογική συνθήκη
- Όσο η λογική συνθήκη παραμένει αληθής ο βρόχος θα επαναλαμβάνεται ξανά και ξανά
- Όταν η λογική συνθήκη γίνει ψευδής τότε ο έλεγχος του προγράμματος δεν θα ξαναμπεί στο σώμα του βρόχου, αλλά θα συνεχίσει με το υπόλοιπο πρόγραμμα μετά από αυτόν
- Θα πρέπει μέσα στο βρόχο να υπάρχει μια τέτοια εντολή, η οποία να αλλάζει την τιμή κάποιας μεταβλητής που συμμετέχει στη λογική συνθήκη, έτσι ώστε να αντιστρέψει τη λογική τιμή της συνθήκης και ο βρόχος κάποτε να τερματιστεί
- Ειδάλλως ο βρόχος δεν θα τερματίζεται ποτέ!
- Εάν η λογική συνθήκη είναι εξ αρχής ψευδής τότε δεν θα εκτελεστεί ούτε μία φορά!

Παράδειγμα: άθροιση αριθμών μέχρι να εισαχθεί το 0

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

int main()
{
	int sum, a;
	sum = 0;
	do
	{
		cout << "Dwse enan arithmo :" << endl;
		cin >> a;
		sum = sum + a;
	} while ( a != 0);
	cout << "To athroisma olwn twn arithmwn einai :" << sum << endl;
	return 0;
}
```

Βρόχος do

- Σε μερικές περιπτώσεις θέλουμε να εξασφαλίσουμε ότι το σώμα του βρόχου θα εκτελεστεί έστω και μία φορά, ανεξάρτητα από την τιμή της λογικής συνθήκης
- Σε αυτή την περίπτωση χρησιμοποιούμε τον βρόχο do, ο οποίος βάζει τον έλεγχο της λογικής συνθήκης στο τέλος του βρόχου
- Η διαφορά του με το βρόγχο while έγκειται στο ότι ο βρόχος while κάνει τον έλεγχο πριν ξεκινήσει το σώμα του, ενώ ο βρόχος do στο τέλος

Παράδειγμα: άθροιση αριθμών μέχρι να εισαχθεί το 0

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

int main()
{
	int sum, a;
	sum = 0;
	while (TRUE)
	{
		cout << "Dwse enan arithmo :" << endl;
		cin >> a;
		if (a == 0) break;
		sum = sum + a;
	}
	cout << "To athroisma olwn twn arithmwn einai :" << sum << endl;
	return 0;
}
```

Εντολή break

- Η εντολή break προκαλεί έξοδο από ένα βρόχο
  - Χρησιμοποιείται για να τερματίζει έναν βρόχο πριν από το φυσικό τέλος του
- Στο επόμενο παράδειγμα ένας βρόχος for ξεκινά να εμφανίσει στην οθόνη τους αριθμούς από το 10 έως το 1
  - Όταν φτάνει όμως στο 3 η εντολή break μέσα στο if προκαλεί την έξοδο από το βρόχο

```
// break example
#include <iostream>
void main ()
{
    int n;
    for (n=10; n>0; n--)
    {
      std::cout << n << endl;
      if (n==3)
      {
        std::cout<<"Αντίστροφη μέτρηση διακόπηκε";
        break;
      }
    }
}
```

Εντολή continue

- Η εντολή continue προκαλεί τη διακοπή εκτέλεσης των εντολών του βρόχου και τη μεταβίβαση του ελέγχου του προγράμματος στην εξέταση της λογικής συνθήκης, ώστε να εκτελεστεί η επόμενη επανάληψη
- Στο επόμενο παράδειγμα ο βρόχος for εμφανίζει τους αριθμούς από το 10 έως το 1
- Όταν φτάνει όμως στο 3, η εντολή continue μέσα στο if αναγκάζει τον έλεγχο να μεταβεί στην αρχή του βρόχου και τη συνέχισή του με την επόμενη επανάληψη
- Έτσι το 3 δεν εμφανίζεται στην οθόνη.

```
// continue example
#include <iostream>
void main ()
{
        for (int n=10; n>0; n--)
        {
            if (n==3) continue;
            std::cout << n << endl;
        }
        cout << "Τέλος";
}
```

Παράδειγμα: υπολογισμός μέσης τιμής βαθμολογίας

```
#include <iostream>
using namespace std;
int main()
{
        int counter, grade;
        float numStudents, total, average;
        total=0;
        counter=1;
        cout << "Enter Number of Students: ";
        cin >> numStudents;
        while(counter<=numStudents)
        {
                cout << "Enter grade: ";
                cin >> grade;
                total=total+grade;
                counter=counter+1;
        }
        average=total/numStudents;
        cout << "Class average is " << average << endl;
        return 0;
}
```

Συναρτήσεις

- Οι συναρτήσεις ομαδοποιούν έναν αριθμό εντολών σχηματίζοντας μία μονάδα και δίνουν στην ομάδα εντολών ένα όνομα
- Κάθε φορά που θα χρειαστεί ο προγραμματιστής να κάνει χρήση αυτής της ομάδας εντολών, χρησιμοποιεί το όνομα αυτής της μονάδας
- Οι συναρτήσεις βοηθούν στην καλύτερη οργάνωση του προγράμματος και στην εξοικονόμηση χρόνου κατά την πληκτρολόγησή του

Συναρτήσεις

- Όπως και με τις μεταβλητές, πριν κάνετε χρήση κάποιας συνάρτησης θα πρέπει να δώσετε οδηγίες στη γλώσσα για τη μορφή της ώστε να ξέρει πως θα τη χρησιμοποιήσει
- Ο ορισμός μιας απλής συνάρτησης είναι ο ακόλουθος:

```
void typestars()
{
for (int i=0; i<20; i++)
   cout << “*”;
cout << endl;
}
```

- Για να ορίσουμε μία συνάρτηση πρώτα δηλώνουμε τον τύπο των δεδομένων που θα επιστρέφει, μετά το όνομά της και τέλος τα ορίσματα που θα δέχεται μέσα σε παρενθέσεις

Συναρτήσεις

- Εάν η συνάρτηση δεν επιστρέφει κάποια τιμή στο κυρίως πρόγραμμα που την καλεί, τότε πριν το όνομά της γράφουμε τη λέξη **void** (κενό)
- Εάν όμως η συνάρτηση επιστρέφει κάποια τιμή στο σημείο του προγράμματος που την καλεί τότε θα πρέπει πριν το όνομά της να δηλώσουμε τον τύπο των δεδομένων που επιστρέφει
  - αν επιστρέφει ακέραια τιμή θα γράφαμε πρώτα int και μετά το όνομά της
  - αν επιστρέφει πραγματικό αριθμό θα γράφαμε πρώτα float και μετά το όνομά της
  - κ.λ.π.
- Στις περιπτώσεις που η συνάρτηση επιστρέφει τιμή είναι η εντολή return μέσα στο σώμα της συνάρτησης
  - Μετά τη δεσμευμένη λέξη return ακολουθεί η παράσταση με την επιστρεφόμενη τιμή
  - Φυσικά ο τύπος δεδομένων της τιμής που επιστρέφεται θα πρέπει να είναι ίδιος με τον τύπο δεδομένων που προηγείται του ονόματος της συνάρτησης

Συναρτήσεις

- Εάν πριν από το όνομα της συνάρτησης δεν υπάρχει η λέξη void αλλά ένας τύπος δεδομένων, τότε μέσα στο σώμα της συνάρτησης θα πρέπει να υπάρχει εντολή **return**
  - Η τιμή αυτή που επιστρέφεται μπορεί να χρησιμοποιηθεί στον υπολογισμό μιας παράστασης ή να αποδοθεί σε μια μεταβλητή
- Μετά το όνομα της συνάρτησης υπάρχουν τα ορίσματά της μέσα σε παρενθέσεις
  - Ακόμα και όταν δεν υπάρχουν ορίσματα είμαστε υποχρεωμένοι να γράφουμε πάντα τις παρενθέσεις
  - Οι παρενθέσεις είναι τα σύμβολα που ξεχωρίζουν τις συναρτήσεις από τις μεταβλητές
- Στο παραπάνω παράδειγμα η συνάρτηση typestars () τυπώνει 20 φορές ένα σύμβολο στην οθόνη. Εάν την αλλάξουμε λίγο έτσι ώστε ο χρήστης να αποφασίζει πόσες φορές θα τυπωθεί το σύμβολο, θα γράφαμε:

```
void typestars(int n)
{
for (int i=0; i<n; i++)//n επαναλήψεις του βρόχου
   cout << “*”;
cout << endl;
}
```

Συναρτήσεις

- Τα ορίσματα τα οποία εσωκλείονται μέσα σε παρενθέσεις είναι μεταβλητές οι οποίες δέχονται τιμές από το σημείο κλήσης της συνάρτησης
  - Μέσα στις παρενθέσεις γίνεται δήλωση των μεταβλητών, για αυτό και η σημειογραφία:

    int n
- Εάν είναι περισσότερα του ενός, τα ορίσματα χωρίζονται μεταξύ τους με κόμμα
- Τα ορίσματα είναι που κάνουν τις συναρτήσεις να εκτελούν διαφορετικά πράγματα, ανάλογα με την είσοδο που δέχονται
- Ειδάλλως χωρίς ορίσματα, οι συναρτήσεις θα επαναλάμβαναν τις ίδιες εντολές κάθε φορά που θα τις καλούσαμε

Κλήση συνάρτησης

- Για να καλέσουμε μία συνάρτηση θα πρέπει προηγουμένως να την έχουμε ορίσει, όπως κάναμε παραπάνω γράφοντας το σώμα της συνάρτησης, ή απλώς να την έχουμε δηλώσει, το οποίο θα εξετάσουμε παρακάτω
- Η κλήση της συνάρτησης γίνεται γράφοντας το όνομά της και τις παρενθέσεις με τα ορίσματα, αν υπάρχουν:

```
#include <iostream>
using namespace std;
void typestars(int n)
{
  for (int i=0; i<n; i++)
  cout << “*”;
  cout << endl;
}

int main()
{
   typestars(15) ; //Κλήση συνάρτησης
   ...
    return 0;
}
```

Κλήση συνάρτησης

- Όταν καλέσουμε μια συνάρτηση τότε ο έλεγχος του προγράμματος κάνει ένα άλμα, αφήνει το σημείο που την καλέσαμε και εισέρχεται μέσα στο σώμα της συνάρτησης
- Αποδίδει στα ορίσματα της συνάρτησης τις τιμές που δώσαμε από το σημείο κλήσης και εκτελεί της εντολές που υπάρχουν στο σώμα της συνάρτησης
- Όταν τελειώσει η συνάρτηση και εκτελεστούν οι εντολές της τότε κάνει ένα δεύτερο άλμα και συνεχίζει από το σημείο που την καλέσαμε και κάτω

Μεταβίβαση ορισμάτων κατά τιμή

- Πολλές φορές όταν καλούμε μια συνάρτηση στη θέση των ορισμάτων μεταβιβάζουμε μεταβλητές αντί σταθερών
  - Τότε οι τιμές των μεταβλητών αποδίδονται με τη σειρά που της έχουμε τοποθετήσει μέσα στις παρενθέσεις, στα ορίσματα της συνάρτησης:

```
#include <iostream>
using namespace std;
void typestars (int n)
{
  for (int i=0; i<n; i++) cout << “*”;
  cout << endl;
}

int main()
{
  int x;
  cout << “Δώσε έναν ακέραιο αριθμό”;
  cin >> x;
  typestars(x) ; //Κλήση συνάρτησης
  ...
  return 0; }
```

Μεταβίβαση ορισμάτων κατά τιμή

- Οι αλλαγές που επιτελούνται μέσα στη συνάρτηση επηρεάζουν τις τοπικές μεταβλητές και οι τυχόν αλλαγές των τιμών τους, δεν έχουν αντίκτυπο στις μεταβλητές με τις οποίες την καλέσαμε.
- Στο παρακάτω παράδειγμα η τιμή της μεταβλητής x της συνάρτησης main() δεν αλλάζει μέσα στη συνάρτηση changeval():

```
#include <iostream>
using namespace std;
void changeval(int i)
{
    i = i+1; //Η τοπική μεταβλητή i αυξάνεται κατά 1
}

int main()
{
        int x = 3;
        changeval(x) ; //Κλήση συνάρτησης
        cout << x ;   // το x εξακολουθεί να είναι 3
       return 0;
}
```

Μεταβίβαση ορισμάτων κατά αναφορά

- Όταν μεταβιβάζουμε ορίσματα κατά αναφορά δεν δημιουργούνται αντίγραφα των μεταβλητών που μεταβιβάζουμε, αλλά δημιουργείται μία αναφορά προς τις αρχικές αυτές μεταβλητές
  - Έτσι οι τυχόν αλλαγές που συντελούνται μέσα στη συνάρτηση στις τοπικές μεταβλητές της, ουσιαστικά είναι αλλαγές των αρχικών μεταβλητών με τις οποίες την καλέσαμε
- Για να μεταβιβάσουμε μεταβλητές κατά αναφορά στον ορισμό της συνάρτησης πριν από το όνομα της μεταβλητής, εισάγουμε το σύμβολο &

```
#include <iostream>
using namespace std;
void changeval(int i, int & j)
{
        i=10;//το10 αποθηκεύεται στο i,το αντίγραφο του x
        j=20;//το 20 αποθηκεύεται στην αναφορά του j,το y
}
int main()
{
        int x = 3;
        int y = 5;
        changeval(x,y) ; //Κλήση συνάρτησης
        cout << x << endl; // το x εξακολουθεί να είναι 3
        cout << y ; // το y είναι πλέον 20
        return 0;
}
```

- Η μεταβλητή i είναι το αντίγραφο της μεταβλητής x και αρχικοποιείται με την τιμή που φέρει η x
  - Με το πέρας της συνάρτησης όταν θα αποδεσμευτεί η μνήμη για τη μεταβλητή i, εάν υπάρξει κάποια αλλαγή σε αυτή, το αποτέλεσμα θα χαθεί
- Η μεταβλητή j είναι ίδια με τη μεταβλητή y και ότι τιμή έχει η μία έχει και η άλλη

Υπερφόρτωση συναρτήσεων

- Η επικεφαλίδα μιας συνάρτησης αποτελείται από το όνομα και τα ορίσματα της συνάρτησης
- Η C++ επιτρέπει τη δημιουργία συναρτήσεων εφόσον έχουν διαφορετική επικεφαλίδα
- Δηλαδή κάποιες συναρτήσεις μπορεί να έχουν το ίδιο όνομα αλλά να διαφέρουν στον αριθμό των ορισμάτων ή στον τύπο δεδομένων αυτών
- Αυτό λέγεται υπερφόρτωση συναρτήσεων

```
void typechar( char ch )
{
        for (int i=0; i<20; i++)
        cout << ch; //Εκτύπωση του χαρακτήρα ch 20 φορές
        cout << endl;
}

void typechar( char ch , int n )
{
        for (int i=0; i<n; i++)
        cout << ch; //Εκτύπωση του χαρακτήρα ch n φορές
        cout << endl;
}
```

Δήλωση συνάρτησης

- Μέχρι τώρα ορίζαμε τις συναρτήσεις, γράφοντας την επικεφαλίδα και το σώμα της συνάρτησης, πριν από την κλήση τους για πρώτη φορά από την main()
- Εάν δοκιμάσουμε να ορίσουμε πρώτα τη main() η οποία καλεί μια συνάρτηση που ορίζεται μετά από αυτή, θα λάβουμε ένα μήνυμα σφάλματος
  - Αυτό γίνεται γιατί όταν θα συναντήσει για πρώτη φορά ο μεταγλωττιστής την κλήση μιας συνάρτησης, θα πρέπει να γνωρίζει τη μορφή αυτής της συνάρτησης ώστε να ξέρει πως θα τη χειριστεί
- Πολλές φορές επιθυμούμε να γράψουμε πρώτα τη κύρια συνάρτηση main() και στη συνέχεια τις συναρτήσεις του προγράμματος
  - Τότε είμαστε υποχρεωμένοι να δηλώσουμε πριν από την main() τις συναρτήσεις
  - Η δήλωση μιας συνάρτησης περιλαμβάνει μόνο την επικεφαλίδα αυτής
  - Δηλαδή τον τύπο δεδομένων που επιστρέφει, το όνομά της και τα ορίσματά της:

```
void typechar( char ch ); //Δήλωση συνάρτησης
```

- Όσον αφορά τα ορίσματα της συνάρτησης σε μια δήλωση, είναι υποχρεωτικό να γράψουμε τον τύπο δεδομένων αυτών, αλλά όχι και τα ονόματά τους:

```
void typechar( char ); //Δήλωση συνάρτησης
```

- Βοηθάει όμως να συμπεριλαμβάνουμε και τα ονόματα των ορισμάτων, όταν είναι τέτοια που μας βοηθούν να τα ξεχωρίσουμε

Δήλωση συνάρτησης

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

 void typechar( char ch ); //Δήλωση συνάρτησης

 int main()
    {
    ...
    typechar('*'); //Κλήση συνάρτησης
    ...
    return 0;
    }

    //Ορισμός συνάρτησης μετά τη main()
    void typechar( char ch )
    {
    for (int i=0; i<20; i++)
    cout << ch; //Εκτύπωση του χαρακτήρα ch 20 φορές
    cout << endl;
    }
```

Ορατότητα και Κύκλος ζωής μεταβλητών

- Ορατότητα
  - Ικανότητα αναγνώρισης της μεταβλητής από block σε block
- Κύκλος ζωής
  - Ικανότητα της μεταβλητής να διατηρεί την τιμή της από κλήση σε κλήση της

Ορατότητα και Κύκλος ζωής μεταβλητών

- Αυτόματες (τοπικές) μεταβλητές
  - Είναι τοπικές σε συνάρτηση ή σε block
  - Αποτελούν τις default μεταβλητές
- Στατικές μεταβλητές
  - Διατηρούν τις τιμές τους μεταξύ διαδοχικών κλήσεων του block ή της συνάρτησης που ανήκουν
  - Δηλώνονται με το static
- Εξωτερικές μεταβλητές
  - Σφαιρικές (global) μεταβλητές
  - Ορίζονται πριν από κάθε συνάρτηση ή ακόμη και πριν από το main
  - Δεν δηλώνονται με κάποιο ιδιαίτερο τρόπο εκτός αν χρησιμοποιούνται σε εσωτερικά blocks ή συναρτήσεις όπου και δηλώνονται ως extern

Παράδειγμα στατικών μεταβλητών

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

void addone();
void minusone();

int main()
{
    addone();
    minusone();
    addone();
    minusone();
    return 0;
}
```

```
void addone()
{
    static int x=1;
    x=x+1;
    cout << x << endl;
}

void minusone()
{
    static int y=1;
    y=y-1;
    cout << y << endl;
}
```

Παράδειγμα εξωτερικών μεταβλητών

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

int x=1000;
void f();

int main()
{
    cout << x << endl;
    f();
    cout << x << endl;
    return 0;
}
```

```
void f()
{
    int x;
    x=1;
    cout << x << endl;
}
```

Παράδειγμα εξωτερικών μεταβλητών

Fmain.cpp

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

extern int x;
void f();

int main()
{
    cout << x << endl;
    f();
    return 0;
}
```

F1.cpp

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

int x=10000;

void f()
{
    int x;
    x=1;
    cout << x << endl;
    {
        int x;
        x=2;
        cout << x << endl; // prints inner x=2
    }
    cout << x << endl; // prints outter x =1
}
```

Παράδειγμα εξωτερικών μεταβλητών

F2.cpp

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

int x=20000;

void f()
{
    int x;
    x=2;
    cout << x << endl; //prints local x
    cout << ::x << endl; // prints global x
}
```

F3.cpp

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

int x=30000;

void f()
{
    int y=x; // assigns global x
    int x=2; //defines local x
    cout << x << endl; //prints new x
    cout << y << endl; // prints y which has
    //the value of global x
}
```

Παράδειγμα εξωτερικών και στατικών μεταβλητών

Smain.cpp

```
#include <iostream>
using namespace std;
extern int x;
int f();
int main()
{
    cout << "x= " << x << endl;
    cout << "1st output of f() " << f() <<
    endl;
    cout << "2nd output of f() " << f() <<
    endl;
    return 0;
}
```

S1.cpp

```
int x = 10000;
int f()
{
    int x=::x;
    return x++;
}
```

S2.cpp

```
int x = 10000;
int f()
{
    static int x=::x;
    return x++;
}
```

Namespaces

- Ένα namespace αποτελεί μία εμβέλεια

```
namespace Example{
  const double PI = 3.14159;
  const double E = 2.71828;
  int myInt = 8;
  void printValues();
}
```

- Μέσα σε namespace γράφουμε δηλώσεις και όχι κώδικα και ορισμούς

- Τα ονόματα ενός namespace γίνονται διαθέσιμα έξω από το namespace με την εντολή using και πλέον δε χρειάζεται να χρησιμοποιηθεί το όνομα του namespace με χρήση του τελεστή επίλυσης εμβέλειας ::

Παράδειγμα namespace

```
#include <iostream>

int myInt = 100000000;

namespace Example{
    const double PI = 3.14159;
    const double E = 2.71828;
    int myInt = 8;
    void printValues();
}
int main()
{
    std::cout << myInt << std:: endl;
     std:: cout << Example::E << std:: endl;
     std:: cout << "The namespace myInt: "
    << Example::myInt << std:: endl ;
    Example::printValues();
    using namespace Example;
    std::cout << "The namespace E from
    main: " << E << std::endl ;
    return 0;
}

void Example::printValues()
{
    std::cout << "Printing from printValues"
    << std::endl;
    std::cout << myInt << std::endl;
    std::cout << E << std::endl;
}
```

Arrays (Μήτρες ή πίνακες)

- Πολλές φορές έχουμε την ανάγκη να ομαδοποιήσουμε δεδομένα σχηματίζοντας μία ενότητα
  - Ο μηχανισμός με τον οποίο γίνεται αυτό είναι η δημιουργία πινάκων
  - Οι πίνακες υπάρχουν σε οποιαδήποτε γλώσσα προγραμματισμού, συμπεριλαμβανομένης της Pascal, BASIC και της C
- Να θυμάστε ότι όποτε χρειαζόμαστε να προσπελάσουμε όλα τα δεδομένα ενός πίνακα (είτε για εισαγωγή, είτε για επεξεργασία, είτε για εξαγωγή αυτών), χρησιμοποιούμε βρόχο επανάληψης for γιατί είναι γνωστός ο αριθμός των δεδομένων

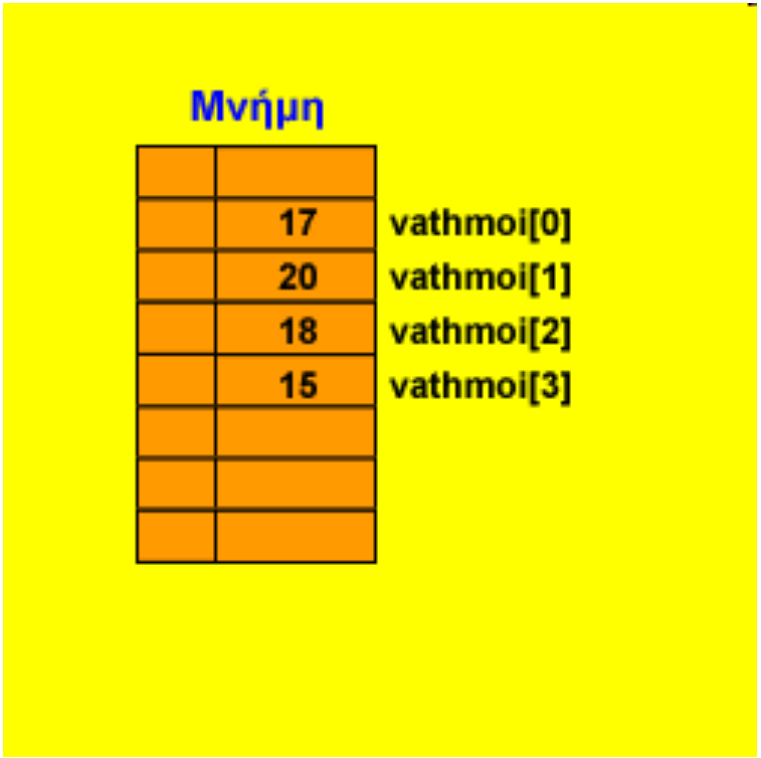
Arrays

- Όπως και με τις μεταβλητές, ένας πίνακας θα πρέπει να οριστεί πριν χρησιμοποιηθεί για αποθήκευση πληροφοριών
  - Για τον ορισμό ενός πίνακα παρέχουμε **3** πληροφορίες
    - Το **όνομα** του πίνακα
    - τον **τύπο δεδομένων** των στοιχείων του και
    - τον **αριθμό των θέσεων** του πίνακα
- Όπως και με τις μεταβλητές πρώτα γράφουμε τον τύπο των δεδομένων που περιέχονται στον πίνακα, μετά το όνομα του πίνακα και στη συνέχεια τον αριθμό των θέσεων του πίνακα κλεισμένο μέσα σε αγκύλες
- Ο αριθμός αυτός των θέσεων του πίνακα λέγεται και μέγεθος του πίνακα καθορίζει πόσες θέσεις δεδομένων του τύπου που ορίσαμε θα δεσμευτούν στη μνήμη

Arrays

- Για να προσπελάσουμε ένα στοιχείο ενός πίνακα γράφουμε το όνομα του πίνακα και μέσα σε αγκύλες τον αριθμό της θέσης
  - Θα πρέπει να δώσετε προσοχή στο εξής: Η C++ αριθμεί τις θέσεις του πίνακα ξεκινώντας από το μηδέν
  - Έτσι για τον πίνακα που ορίσαμε προηγουμένως με τις 4 θέσεις:

```
vathmoi[0] = 17;
vathmoi[1] = 20;
vathmoi[2] = 18;
vathmoi[3] = 15;
```



Arrays

- Έχουμε ήδη αναφέρει ότι κάθε φορά που θα προσπελάσουμε όλα τα στοιχεία ενός πίνακα, είτε για αποθήκευση τιμών, είτε για επεξεργασία, είτε για εμφάνιση των τιμών τους, χρησιμοποιούμε βρόχους for, γιατί ο αριθμός των επαναλήψεων ισούται με τον αριθμό των στοιχείων
- Στο επόμενο παράδειγμα καλούμαστε να γεμίσουμε έναν πίνακα 30 θέσεων με τους βαθμούς σπουδαστών σε ένα μάθημα

Arrays

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

const int theseis=30;

int main ()
    {
     float vathmoi[theseis];
     float sum=0;
     float mesos;
     //αποθήκευση δεδομένων πίνακα
     for (int i=0; i<theseis; i++)
     {
        cout<< "Δώσε το βαθμό του"<< i << "μαθητή" ;
        cin >> vathmoi[i] ;
        cout<< endl;
     }
     //επεξεργασία δεδομένων πίνακα
     for (int i=0; i<theseis; i++)
     {
      sum = sum + vathmoi[i] ;
     }
     mesos = sum / theseis;
     cout << "Ο μέσος όρος είναι: "<<mesos;
     return 0;
    }
```

Arrays

- Όταν επιθυμούμε στο σημείο που δηλώνουμε έναν πίνακα να αποδώσουμε αρχικές τιμές στα στοιχεία του, τότε χρησιμοποιούμε τον τελεστή απόδοσης τιμή (=) και τις αρχικές τιμές μέσα σε άγκιστρα, χωρισμένες με κόμματα:

```
int vathmoi[4]={17, 20, 18, 15}
```

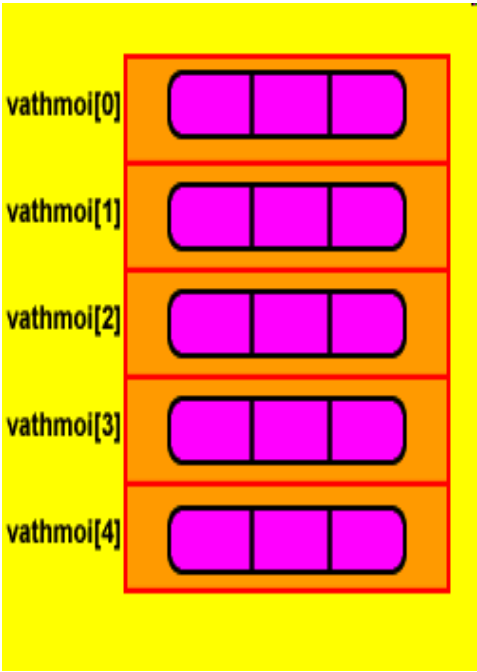
- Στην πραγματικότητα δεν χρειάζεται μέσα στις αγκύλες να δώσουμε το μέγεθος του πίνακα όταν αποδίδουμε αρχικές τιμές, αφού ο μεταγλωττιστής μπορεί να μετρήσει τις τιμές μέσα στις παρενθέσεις
- Εάν όμως δώσουμε το μέγεθος του πίνακα και το πλήθος των αρχικών τιμών μέσα στα άγκιστρα είναι μικρότερος δεν υπάρχει πρόβλημα
  - Στα στοιχεία του πίνακα που απομένουν θα αποδοθεί ως αρχική τιμή το 0
  - Εάν όμως τα στοιχεία του πίνακα είναι λιγότερα από τις αρχικές τιμές, τότε προκαλείται ένα σφάλμα

Arrays

- Εξετάσαμε μέχρι τώρα πίνακες μίας διάστασης που σε κάθε θέση τους αποθηκεύεται ένα στοιχείο
- Οι πίνακες όμως μπορεί να έχουν περισσότερες διαστάσεις
- Στο επόμενο παράδειγμα χρησιμοποιούμε έναν πίνακα δύο διαστάσεων για να αποθηκεύσουμε τους βαθμούς 5 σπουδαστών σε 3 μαθήματα

```
const int spoudastes = 5;
const int mathimata = 3;

//ορισμός δισδιάστατου πίνακα
int vathmoi[spoudastes][mathimata] ;
```



- Στην τελευταία εντολή ορίζεται ο πίνακας με όνομα vathmoi, που θα συγκρατεί ακέραιους
  - Ο πίνακας έχει δύο διαστάσεις για αυτό γράφουμε τις θέσεις που καταλαμβάνει κάθε διάσταση μέσα στις δικές της αγκύλες
- Η ίδια σημειογραφία ακολουθείται και όταν θέλουμε να προσπελάσουμε τα στοιχεία αυτού του δισδιάστατου πίνακα:

```
vathmoi[0][0] = 18;
```

- Οι πίνακες μπορεί να είναι όσον διαστάσεων θέλουμε
  - Ωστόσο μπορούμε να φανταστούμε πως προκύπτει ο δισδιάστατος πίνακας του παραδείγματος, εάν θεωρήσουμε ότι σε κάθε μία από τις 5 θέσεις της πρώτης διάστασης, τοποθετείται ένας μονοδιάστατος πίνακας 3 θέσεων

Arrays

- Μπορούμε πάντα να φανταστούμε τους πολυδιάστατους πίνακες ως πίνακες που περιέχουν πίνακες. Η προσπέλαση όλων των στοιχείων των πολυδιάστατων πινάκων γίνεται με εμφωλευμένους βρόχους for:

```
for (int i=0; i<spoudastes; i++)
{
    cout<<"δώσε τους βαθμούς του"<<i<<"σπουδαστή: ";
    for (int j=0; j<mathimata; j++)
     {
         cout << "μαθημα "<< j << ":" ;
         cin >> vathmoi[i][j];
    }
    cout << endl;
}
```

- Στο παραπάνω παράδειγμα ο εξωτερικός βρόχος for, με μεταβλητή μετρητή τη i, επαναλαμβάνεται 5 φορές όσος είναι ο αριθμός των σπουδαστών
- Ο εσωτερικός βρόχος for, με μεταβλητή μετρητή τη j, επαναλαμβάνεται 3 φορές όσος είναι ο αριθμός των μαθημάτων
- Η αποθήκευση τιμών στον πίνακα, γίνεται με την πρόταση:

```
cin >> vathmoi[i][j];
```

- Για την σωστή ταξινόμηση των τιμών στις κατάλληλες θέσεις χρησιμοποιούνται οι μεταβλητές i (που διατρέχει τους αριθμούς από το 0 έως το 4), και j (που διατρέχει τους αριθμούς από το 0 έως το 2)

Arrays

- Όταν θέλουμε έναν πίνακα να τον μεταβιβάσουμε σε μία συνάρτηση τότε θα πρέπει να ξέρετε ότι ο πίνακας μεταβιβάζεται **πάντα κατά αναφορά**
  - Οι τυχόν αλλαγές που συντελούνται στα στοιχεία του πίνακα μέσα στο σώμα της συνάρτησης, είναι μόνιμες και ισχύουν και με το τέλος της συναρτήσεως.
  - Όταν δηλώνουμε τη συνάρτηση στην επικεφαλίδα της **δεν χρησιμοποιούμε** το σύμβολο &

Arrays

```
#include <iostream>
const int theseis=30;
float average(int pinakas[theseis])
{
   float sum=0;
   for (int i=0; i<theseis; i++)
   {
     sum = sum + pinakas[i];
   }
   return sum/theseis;
}

void gemisepinaka(int pinakas[theseis])
{
 for (int i=0; i<theseis; i++)
 {
   cout << "Δώσε τον "<<i <<"βαθμό";
   cin >> pinakas[i]; //Γέμισμα πίνακα
   cout << endl;
 }
}

int main()
{
 int vathmoi[theseis];
 gemisepinaka(vathmoi);
 cout <<"Ο μέσος όρος: "<< average(vathmoi);
 return 0;}
```

- Παρατηρείστε τον τρόπο με τον οποίο καλούμε συναρτήσεις που δέχονται πίνακα:

  gemisepinaka(vathmoi);
  average(vathmoi);

- Μέσα στις παρενθέσεις των συναρτήσεων γράφουμε μόνο το όνομα του πίνακα

Char Arrays

- Ο τύπος δεδομένων char που έχουμε εξετάσει, αποθηκεύει έναν μόνο χαρακτήρα:

```
char c = 'A' ;
```

- Τι γίνεται όμως όταν θέλουμε να αποθηκεύσουμε μία ομάδα χαρακτήρων, όπως ένα όνομα ή μία φράση;
  - Είναι πολύ χρήσιμο για μία γλώσσα προγραμματισμού να μπορεί να επεξεργαστεί και ομάδες χαρακτήρων εκτός από αριθμούς
  - Μία ομάδα χαρακτήρων ονομάζεται και αλφαριθμητικό υπονοώντας με την ονομασία αυτή συνδυασμό χαρακτήρων αλφαβήτου και αριθμητικών ψηφίων
- Ένα αλφαριθμητικό μπορούμε να το αποθηκεύσουμε σε έναν πίνακα χαρακτήρων, με κάθε χαρακτήρα να καταλαμβάνει τη θέση ενός στοιχείου του πίνακα.

```
char chararray[30];
```

Char Arrays

- Ο πίνακας chararray έχει 30 θέσεις και μπορεί να δεχθεί έως 29 χαρακτήρες
  - Η C++ αναγνωρίζει το τέλος του αλφαριθμητικού όταν εντοπίζει τον πρώτο μηδενικό χαρακτήρα, ο οποίος είναι ο '\0'.
  - Έτσι εάν θέλουμε να αρχικοποιήσουμε τον πίνακα χαρακτήρων αποθηκεύοντας το αλφαριθμητικό "my name" θα γράψουμε:

```
char chararray[30]={'m','y',' ','n','a','m','e'};
```

- ή με τον πιο απλό τρόπο αρχικοποίησης πινάκων που μας παρέχει η γλώσσα ειδικά για τους πίνακες χαρακτήρων:

```
char chararray[30] = "my name" ;
```

- Στις παραπάνω περιπτώσεις οι 7 χαρακτήρες του αλφαριθμητικού αποθηκεύονται στις πρώτες 7 θέσεις του πίνακα και στην 8η θέση εισάγεται ο μηδενικός χαρακτήρας '\0'.
- Το μήκος του αλφαριθμητικού αναγνωρίζεται από την ανάγνωση αυτού του μηδενικού χαρακτήρα
- Η απόδοση μιας ομάδας χαρακτήρων, χρησιμοποιώντας το όνομα του πίνακα μπορεί να γίνει μόνο τη στιγμή της αρχικοποίησης
- Μπορούμε να εκτυπώσουμε απευθείας το αλφαριθμητικό του πίνακα, γράφοντας με το ρεύμα εξόδου cout μόνο το όνομα του πίνακα:

```
cout << chararray ;
```

ΧΕΙΡΙΣΜΟΣ ΑΛΦΑΡΙΘΜΗΤΙΚΩΝ

- Έστω ο πίνακας χαρακτήρων chararray 30 θέσεων, ο οποίος αρχικοποιείται με το αλφαριθμητικό “my name”:

  char chararray[30] = “my name” ;

- Η συνάρτηση strlen() δέχεται ως όρισμα το όνομα ενός πίνακα αλφαριθμητικών και επιστρέφει έναν αριθμό, ο οποίος ισούται με το μήκος του αλφαριθμητικού. Η παρακάτω πρόταση εμφανίζει τον αριθμό 7 στην οθόνη:

  cout << strlen(chararray);

- Η συνάρτηση strcpy() δέχεται ως όρισμα δύο πίνακες αλφαριθμητικών (ή έναν πίνακα κι ένα αλφαριθμητικό) και αντιγράφει στον πρώτο πίνακα, ότι υπάρχει στο δεύτερο πίνακα αλφαριθμητικών.

```
char c[20] = "hello";
char d[20] = "world" ;
strcpy(c,d);
strcpy(d,"John");
cout << c ; //"world"
cout << d ; //"John"
```

ΧΕΙΡΙΣΜΟΣ ΑΛΦΑΡΙΘΜΗΤΙΚΩΝ

- Η συνάρτηση strncpy() δέχεται τρία ορίσματα: δύο πίνακες αλφαριθμητικών και έναν αριθμό. Ο αριθμός φανερώνει τους πρώτους χαρακτήρες του δεύτερου πίνακα που θα αντιγραφούν στον πρώτο:

```
char c[20] = "hello" ;
char d[20] = "world" ;
strncpy(c,d,3);
cout << c ; //"worlo"
```

- Η συνάρτηση strcmp() δέχεται ως όρισμα δύο πίνακες αλφαριθμητικών και τους συγκρίνει. Επιστρέφει τιμή μηδέν όταν οι δύο πίνακες έχουν τα ίδια αλφαριθμητικά, μικρότερη ή μεγαλύτερη από μηδέν ανάλογα με ποιο αλφαριθμητικό είναι μεγαλύτερο ή μικρότερο.

```c
#include <stdio.h>

int main()
{
    char str1[] = "abcd", str2[] = "abCd", str3[] =
     "abcd";
    int result;

    // comparing strings str1 and str2
    result = strcmp(str1, str2);
    printf("strcmp(str1, str2) = %d\n", result);

    // comparing strings str1 and str3
    result = strcmp(str1, str3);
    printf("strcmp(str1, str3) = %d\n", result);

    return 0;
}
```

Output

strcmp(str1, str2) = 32
strcmp(str1, str3) = 0
The first unmatched character between string str1 and str2 is third character. The ASCII value of 'c' is 99 and the ASCII value of 'C' is 67.

Hence, when strings str1 and str2 are compared, the return value is 32.

When strings str1 and str3 are compared, the result is 0 because both strings are identical.

Ο τύπος string

- Η C++ μας δίνει τη δυνατότητα να χρησιμοποιήσουμε μεταβλητές τύπου string αρκεί να συμπεριλάβουμε το αρχείο string στο οποίο είναι υλοποιημένος ο τύπος string:

```
#include <iostream>
#include <string>
int main()
{
  string a = "my name ";
  string b = "is John";
  …
  return 0;
}
```

- Ο τελεστής (+) μας δίνει τη δυνατότητα να ενώσουμε αλφαριθμητικά (string). Έτσι η παρακάτω πρόταση θα εμφανίσει στην οθόνη την ένωση των δύο αλφαριθμητικών:

```
cout << a+b ; //"my name is John"
```

Το ρεύμα stdio

- Μέχρι τώρα έχουμε δει πως μπορούμε να διαβάζουμε / εκτυπώνουμε δεδομένα με χρήση του ρεύματος iostream
- Εκτός από το iostream μπορούμε να χρησιμοποιούμε και το stdio που είναι ο παλαιότερος (κατά c) τρόπος ανάγνωσης / εκτύπωσης δεδομένων
- Με το stdio οι σχετικές εντολές είναι οι
  - scanf
  - printf

Σχεδιασμένη έξοδος στην οθόνη με το ρεύμα STDIO

```
printf(format, arg1, arg2,…, argm);
```

- Το string format αρχίζει και τελειώνει με διπλά εισαγωγικά
- Μέσα στα εισαγωγικά μπορεί να υπάρχει
  - **Κείμενο**
  - **Ακολουθίες διαφυγής (π.χ., \n)**
  - **Προσδιοριστές σχεδιασμού τιμών που αντιστοιχούν στα arg1, arg2, κλπ.**
- **Ένας προσδιοριστής σχεδιασμού τιμής έχει τον παρακάτω γενικό τύπο**

```
%flag width.precision type
```

Σχεδιασμένη έξοδος στην οθόνη με το ρεύμα STDIO

```
%flag width.precision type
```

- **flag:**
  - κανονικά η τιμή τυπώνεται στα δεξιά του πεδίου που προσδιορίζει το width. Αν υπάρχει όμως το πλην (-), τότε η εκτύπωση γίνεται στα αριστερά. Αν υπάρχει το (0), τότε συμπληρώνεται το 0 στα κενά.
- **width:**
  - θετικός ακέραιος που προσδιορίζει το ελάχιστο πλήθος χαρακτήρων που θα τυπωθεί
  - Έχει νόημα κυρίως για δεκαδικούς
- **precision:**
  - θετικός ακέραιος που προσδιορίζει το πλήθος δεκαδικών ψηφίων που θα τυπωθεί σε περίπτωση πραγματικών δεκαδικών

Σχεδιασμένη έξοδος στην οθόνη με το ρεύμα STDIO

- type:
  - Προσδιορίζει τον τύπο της τιμής που θα εκτυπωθεί
    - d: ακέραιες τιμές
    - ld:long
    - i: το ίδιο
    - u: Θετικές ακέραιες τιμές
    - o: ακέραιες στο οκταδικό
    - x: ακέραιες στο δεκαεξαδικό
    - f: δεκαδικές
    - e: δεκαδικές εκθετικής μορφής
      - [-]mm.nnne[+/-t]xx
    - c: χαρακτήρα
    - s: string

Σχεδιασμένη έξοδος στην οθόνη με το ρεύμα STDIO

```
#include <stdio.h>

void main()
{
    long large = 123456789;
    int number = 32000, *point;
    char ch='a', string[10]="TEXT";
    float real = 12345.12345;
    double dreal;
    dreal = 12345.12345;
    point = &number;
    printf(" Times metablitwn me aplo format :\n");
    printf(" %d %c %s %f %e \n", number, ch, string, real, dreal);
    printf(" Eidikes ektipwseis diaforwn timwn :\n");
    printf(" %.9f %.6f %-10d %ld %10.4s \n", real, dreal, number, large, string);
    printf(" Dieuthinsi metablitis number : %p \n", point);
}
```

Σχεδιασμένη έξοδος στην οθόνη με το ρεύμα STDIO

```
#include<stdio.h>

    void main()
    {
            printf(":%s:\n", "Hello, world!");
            printf(":%15s:\n", "Hello, world!");
            printf(":%.10s:\n", "Hello, world!");
            printf(":%-10s:\n", "Hello, world!");
            printf(":%-15s:\n", "Hello, world!");
            printf(":%.15s:\n", "Hello, world!");
            printf(":%15.10s:\n", "Hello, world!");
            printf(":%-15.10s:\n", "Hello, world!");
    }
```

```
:Hello, world!:
:  Hello, world!:
:Hello, wor:
:Hello, world!:
:Hello, world!  :
:Hello, world!:
:     Hello, wor:
:Hello, wor     :
```

- The printf(":%s:\n", "Hello, world!"); statement prints the string (nothing special happens.)
- The printf(":%10s:\n", "Hello, world!"); statement prints the string, but print 15 characters. If the string is smaller the "empty" positions will be filled with "whitespace."
- The printf(":%.10s:\n", "Hello, world!"); statement prints the string, but print only 10 characters of the string.
- The printf(":%-10s:\n", "Hello, world!"); statement prints the string, but prints at least 10 characters. If the string is smaller "whitespace" is added at the end. (See next example.)
- The printf(":%-15s:\n", "Hello, world!"); statement prints the string, but prints at least 15 characters. The string in this case is shorter than the defined 15 character, thus "whitespace" is added at the end (defined by the minus sign.)
- The printf(":%.15s:\n", "Hello, world!"); statement prints the string, but print only 15 characters of the string. In this case the string is shorter than 15, thus the whole string is printed.
- The printf(":%15.10s:\n", "Hello, world!"); statement prints the string, but print 15 characters.
- If the string is smaller the "empty" positions will be filled with "whitespace." But it will only print a maximum of 10 characters, thus only part of new string (old string plus the whitespace positions) is printed.
- The printf(":%-15.10s:\n", "Hello, world!"); statement prints the string, but it does the exact same thing as the previous statement, accept the "whitespace" is added at the end.

Δομές

- Μια δομή είναι μία συλλογή μεταβλητών
- Οι μεταβλητές μιας δομής μπορεί να είναι διαφορετικών τύπων, άλλες μπορεί να είναι τύπου char, άλλες int, άλλες float κ.λ.π.
  - Αυτό έρχεται σε αντίθεση με τα arrays που απαιτείται όλα τα δεδομένα να είναι ιδίου τύπου
- Τα στοιχεία αυτά που αποτελούν τη δομή ονομάζονται μέλη της δομής.

Δομές

- Από τη στιγμή που προσδιορίσαμε μία δομή, μπορούμε να ορίσουμε μεταβλητές αυτού του τύπου. Το όνομα της δομής χρησιμεύει όπως οι δεσμευμένες λέξεις int, char και float. Εμείς ως χρήστες έχουμε ορίσει ένα νέο τύπο δεδομένων που λειτουργεί όπως και οι ενσωματωμένοι της γλώσσας.

```
#include <iostream>
struct proion
{
  int kodikos;
  float timi;
};

int main()
{
  proion sokolata;//Ορισμός μεταβλητής τύπου proion
  int var1; //Ορισμός μεταβλητής τύπου int
  sokolata.kodikos = 15; //Απόδοση τιμών στα μέλη
  sokolata.timi = 1.75; //της δομής

  std::cout << "Κωδικός προϊόντος: "<<sokolata.kodikos;
  std::cout << "Τιμή προϊόντος: "<< sokolata.timi;

  //Ορισμός και απόδοση αρχικών τιμών μαζί
  proion tsigara= {53 , 2.70 };
  return 0;
}
```

ΑΝΤΙΚΕΙΜΕΝΑ ΚΑΙ ΚΛΑΣΕΙΣ

- Έχουμε εξετάσει:
  - τους πίνακες που ομαδοποιούν δεδομένα ίδιου τύπου
  - τις δομές που ομαδοποιούν συνήθως δεδομένα διαφορετικού τύπου
  - τις συναρτήσεις που οργανώνουν ενέργειες, σε ενότητες με ονόματα
- Σε αυτό το μάθημα θα συνδυάσουμε όλα όσα μάθατε μέχρι τώρα, εξετάζοντας τον ακρογωνιαίο λίθο του αντικειμενοστρεφούς προγραμματισμού τις τάξεις και τα αντικείμενα και τα οφέλη που προκύπτουν από τη χρήση τους.

ΑΝΤΙΚΕΙΜΕΝΑ ΚΑΙ ΚΛΑΣΕΙΣ

- Οι κλάσεις περιέχουν δεδομένα και συναρτήσεις που μπορούν να χειριστούν αυτά τα δεδομένα
  - Τις συναρτήσεις αυτές που ανήκουν σε μία κλάση και έχουν το δικαίωμα να αλλάζουν τις τιμές των δεδομένων, τις ονομάζουμε συναρτήσεις-μέλη ή μεθόδους.

ΑΝΤΙΚΕΙΜΕΝΑ ΚΑΙ ΚΛΑΣΕΙΣ

- Η τοποθέτηση των δεδομένων και των συναρτήσεων μαζί, σε μια ενότητα, είναι η κεντρική ιδέα του αντικειμενοστρεφούς προγραμματισμού.

- Όταν καθορίζουμε μία κλάση καθορίζουμε το σχέδιο, από το οποίο θα προκύψουν αντικείμενα
  - Όπως είδαμε στο μάθημα των δομών και των τύπων δεδομένων που ορίζονται από το χρήστη, ένα αντικείμενο έχει την ίδια σχέση με μια κλάση, όπως μια μεταβλητή με έναν τύπο δεδομένων
  - Συνηθίζουμε να λέμε ότι ένα αντικείμενο είναι ένα στιγμιότυπο (instance) μιας κλάσης

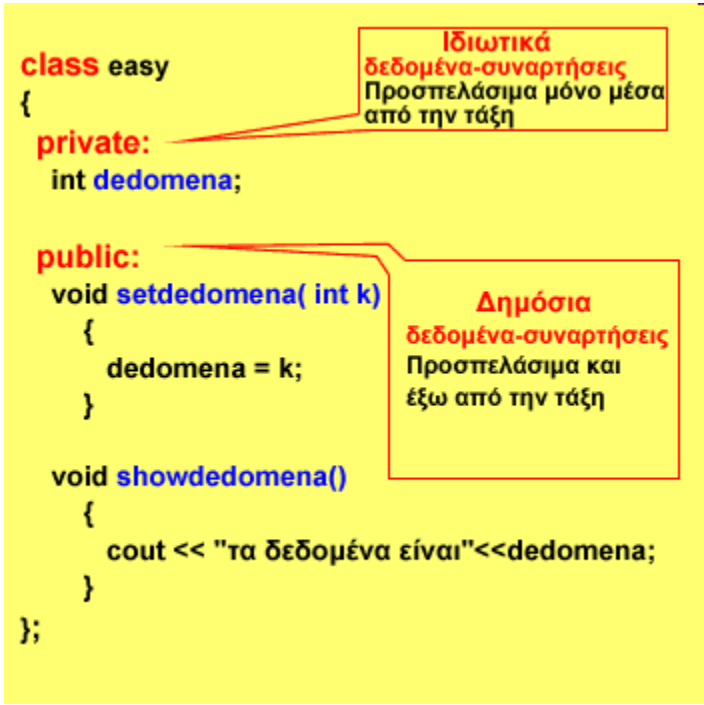
- Ο προσδιορισμός μιας κλάσης ξεκινά με τη δεσμευμένη λέξη class, ακολουθούμενη από το όνομα της
  - Το σώμα της κλάσης οριοθετείται με άγκιστρα (όπως και στη δομή) και τερματίζεται με ελληνικό ερωτηματικό.

```
class όνομα
{
  Δεδομένα
  Συναρτήσεις
};
```

- Να θυμάστε ότι κατασκευές δεδομένων όπως οι δομές και οι τάξεις τελειώνουν με ελληνικό ερωτηματικό

ΑΝΤΙΚΕΙΜΕΝΑ ΚΑΙ ΚΛΑΣΕΙΣ

- Τα δεδομένα ή οι συναρτήσεις που ορίζονται μετά τη λέξη private είναι ιδιωτικά και προσπελαύνονται μόνο μέσα από την κλάση
- Τα δεδομένα ή οι συναρτήσεις που ορίζονται μετά τη λέξη public, είναι δημόσια και είναι προσπελάσιμα και έξω από την κλάση
  - Έτσι όταν θέλουμε να προσπελάσουμε ένα δεδομένο που είναι ιδιωτικό (private), από ένα σημείο του προγράμματός μας έξω από την κλάση, θα καλέσουμε μία δημόσια (public) συνάρτηση της κλάσης που είναι ορισμένη να κάνει ακριβώς αυτό το πράγμα
  - Έτσι δεν υπάρχει ο κίνδυνος να καταστρέψουμε τα δεδομένα επειδή δεν γνωρίζουμε το σωστό τρόπο προσπέλασής τους.




```
class easy
{
 private:
  int dedomena;

 public:
  void setdedomena( int k)
    {
      dedomena = k;
    }

  void showdedomena()
    {
      cout << "τα δεδομένα είναι"<<dedomena;
    }
};
```

ΑΝΤΙΚΕΙΜΕΝΑ ΚΑΙ ΚΛΑΣΕΙΣ

- Οι συναρτήσεις μιας κλάσης λέγονται και συναρτήσεις-μέλη ή απλώς μέθοδοι
- Οι δύο συναρτήσεις-μέλη κάνουν λειτουργίες πολύ συνηθισμένες σε μία, ορίζουν τα δεδομένα που είναι αποθηκευμένα στην κλάση:

```
void setdedomena(int k)
{
dedomena = k; //ο προφυλαγμένος ακέραιος dedomena
}          //δέχεται την τιμή του k
```

- Εμφανίζουν τα δεδομένα που είναι αποθηκευμένα στην κλάση:

```
void showdedomena()
{
cout<< "τα δεδομένα είναι "<< dedomena;
}
```

ΑΝΤΙΚΕΙΜΕΝΑ ΚΑΙ ΚΛΑΣΕΙΣ

- Ο καθορισμός της κλάσης easy του προηγούμενου παραδείγματος, δεν δεσμεύει κάποια ποσότητα μνήμης
- Είναι οδηγίες για το πώς θα κατασκευαστούν αντικείμενα που θα πηγάζουν από αυτή την κλάση
  - Όπως ακριβώς γράφοντας απλώς τη λέξη int, δεν δεσμεύουμε μνήμη γιατί δεν κατασκευάζεται κάποια μεταβλητή μόνο με τη χρήση αυτή της λέξης
- Για να ορίσουμε δύο αντικείμενα obj1 και obj2 που θα πηγάζουν από την κλάση easy μέσα στη main(), θα γράψουμε:

```
int main()
{
    easy obj1 , obj2 ;
        //κατασκευή αντικειμένων obj1 και obj2
       //βάσει της easy
    return 0;
}
```

ΑΝΤΙΚΕΙΜΕΝΑ ΚΑΙ ΚΛΑΣΕΙΣ

- Τώρα έχουν κατασκευαστεί τα αντικείμενα obj1 και obj2 και έχει δεσμευτεί χώρος για αυτά στη μνήμη.

- Κάθε ένα αντικείμενο από αυτά έχει ένα ιδιωτικό δεδομένο τον ακέραιο dedomena και δύο δημόσιες συναρτήσεις-μέλη τις setdedomena() και showdedomena()

- Για να προσπελάσουμε δημόσια δεδομένα ή δημόσιες συναρτήσεις-μέλη ενός αντικειμένου χρησιμοποιούμε πάντα τον τελεστή της τελείας:

```
obj1.setdedomena(20);
obj2.setdedomena(5);
```

- Για να εμφανίσουμε τα προφυλαγμένα δεδομένα των αντικειμένων θα καλέσουμε την κατάλληλη συνάρτηση-μέλος των αντικειμένων αυτών:

```
obj1.showdedomena(); //εμφάνιση 20 στην οθόνη
obj2.showdedomena(); //εμφάνιση 5 στην οθόνη
```

ΑΝΤΙΚΕΙΜΕΝΑ ΚΑΙ ΚΛΑΣΕΙΣ

- Πολλές φορές θέλουμε τη στιγμή που κατασκευάζεται ένα αντικείμενο, να παίρνουν αρχικές τιμές κάποια δεδομένα του
  - Σε αυτή τη περίπτωση όταν προσδιορίζουμε την κλάση ορίζουμε μία μέθοδο εγκατάστασης
  - Αυτή δεν είναι άλλη από μία συνάρτηση-μέλος που έχει ίδιο όνομα με την κλάση, δεν επιστρέφει κάποια τιμή (αφού δουλειά της είναι να απλώς να δίνει αρχικές τιμές) και εκτελείται αυτόματα μόλις κατασκευαστεί το αντικείμενο
  - Οι μέθοδοι εγκατάστασης ονομάζονται και δομητές (constructors).

ΑΝΤΙΚΕΙΜΕΝΑ ΚΑΙ ΚΛΑΣΕΙΣ

```
#include <iostream>
class easy
{
 private:
   int dedomena;
 public:
      //μέθοδος εγκατάστασης
      //έχει ίδιο όνομα με την κλάση και
      //εκτελείται μόλις δημιουργείται ένα
      αντικείμενο
  easy()
  {
    dedomena = 0;
  }

  void setdedomena(int k)
  {
    dedomena = k;
  }
  void showdedomena()
  {
    std::cout<<"τα δεδομένα είναι
      "<<dedomena;
  }
};
```

- Τώρα μόλις δημιουργήσουμε ένα αντικείμενο βάσει της κλάσης που ορίσαμε, καλείται αυτόματα η μέθοδος εγκατάστασης και αποδίδει το μηδέν στον ακέραιο dedomeno

```
int main()
{
     //κατασκευή αντικειμένου obj1 βάσει της
     easy
  easy obj1;
  obj1.showdedomena();
  ...
  return 0;
}
```

Θα εμφανιστεί στην οθόνη:

τα δεδομένα είναι 0

ΑΝΤΙΚΕΙΜΕΝΑ ΚΑΙ ΚΛΑΣΕΙΣ

- Υπάρχει και μία άλλη ειδική συνάρτηση, που λέγεται μέθοδος αποσύνδεσης, η οποία καλείται όταν ένα αντικείμενο καταστρέφεται
  - Η μέθοδος αποσύνδεσης ονομάζεται και αποδομητής (destructor)
- Η μέθοδος αποσύνδεσης έχει το ίδιο όνομα με την κλάση αλλά έχει μπροστά μία περισπωμένη (~)
  - Όπως και οι μέθοδοι εγκατάστασης έτσι και η μέθοδος αποσύνδεσης δεν έχουν τιμή επιστροφής
  - Η πιο συνηθισμένη χρήση των μεθόδων αποσύνδεσης είναι η αποδέσμευση της μνήμης που είχε δεσμευτεί για ένα αντικείμενο

```
class kapoia
{
  private:
    int data;
  public:
    kapoia() //μέθοδος εγκατάστασης
    {
      data = 0;
    }
    ~kapoia() //μέθοδος αποσύνδεσης
    {
    }
};
```

DEFAULT ΤΙΜΕΣ

- Πολλές φορές θέλουμε η μέθοδος εγκατάστασης (δομητής) να έχει κάποιες προκαθορισμένες τιμές για τα ιδιωτικά μέλη αλλά να μπορεί και να τις δέχεται σαν ορίσματα.

DEFAULT TIMEΣ

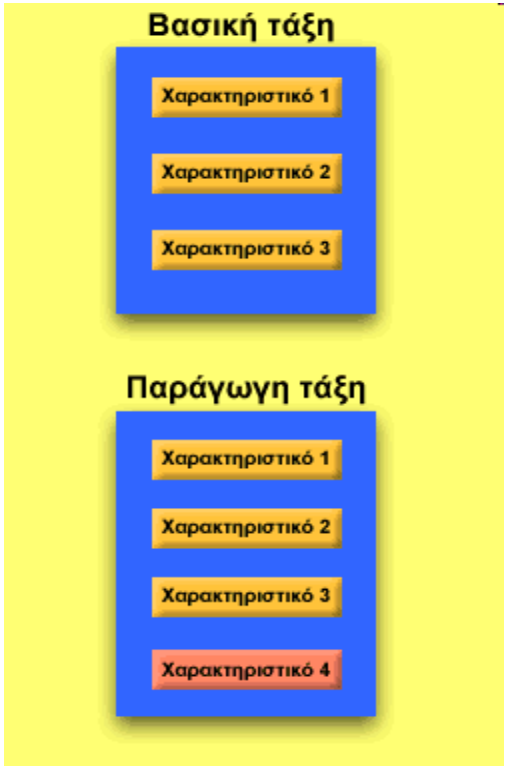
```
#include <iostream.h>
using namespace std;
class Date
{
    private:
        int day; int month; int year;
    public:
        Date(int d = 1, int m = 1, int y = 2004)
        {
            day = d; month = m; year = y;
            cout << "I just created a Date with value: " << day << ' ' << month << ' ' << year << endl;
        }
        ~Date()
        {
            cout << '\n' << "I am destroying a Date with value: "<< day << ' ' << month << ' ' << year
    << endl;
        }

        void print()
        {
            cout << day << ' ' << month << ' ' << year<< endl;
        }
};

int main()
{
    Date d1; Date d2(3); Date d3(3,4); Date d4(3,4,2005);
    return 0;
}
```

ΚΛΗΡΟΝΟΜΙΚΟΤΗΤΑ

- Με τη διαδικασία της κληρονομικότητας, μία νέα κλάση δημιουργείται κληρονομώντας όλα τα χαρακτηριστικά μιας κλάσης και προσθέτοντας νέα δικά της
  - Η νέα κλάση που δημιουργείται ονομάζεται παράγωγη, ενώ η κλάση από την οποία κληρονομούνται τα χαρακτηριστικά ονομάζεται βασική
  - Λέγοντας χαρακτηριστικά εννοούμε είτε δεδομένα είτε συναρτήσεις μέλη



ΚΛΗΡΟΝΟΜΙΚΟΤΗΤΑ

- Για να κληρονομήσει μία παράγωγη κλάση τα μέλη της βασικής, χρησιμοποιούμε την ακόλουθη σύνταξη:

```
class Dog: public Animal
{
   σώμα κλάσης Dog
};
```

- Στο παραπάνω κομμάτι κώδικα υποτίθεται οτι έχουμε κατασκευάσει και ελέγξει μία κλάση που ονομάζεται Animal
- Η κλάση αυτή εκφράζει με ένα γενικό τρόπο τα ζώα έχοντας κάποια δεδομένα (ηλικία, φύλο, βάρος) και κάποιες συναρτήσεις-μέλη (μέθοδοι) οι οποίες χρησιμοποιούνται για να έχουμε πρόσβαση στα δεδομένα
- Εφόσον έχουμε δημιουργήσει και ελέγξει την κλάση Animal, δεν υπάρχει λόγος να την ξαναδημιουργήσουμε όταν θέλουμε να ορίσουμε την πιο ειδική κλάση Dog που εκφράζει τους σκύλους
  - Έτσι της κληρονομούμε όλα τα χαρακτηριστικά της κλάσης Animal και της προσθέτουμε και νέα δικά της (όπως το όνομα)

ΠΑΡΑΔΕΙΓΜΑ

```
#include <iostream>
#include <string>
using namespace std;
class Animal
{
 private:
  float ilikia; bool arren; int baros;
 public:
  Animal() {} //Δομητής χωρίς ορίσματα
  Animal(float i, bool ar, int wei)  {
    ilikia = i; arren = ar; baros = wei;  }
 void dwsestoixeia()
 {
  cout<<"dwse ilikia tou Zwou: ";
  cin>> ilikia;
  int a;
  cout<<"einai arseniko? gia NAI dwse 1: ";
  cin>> a;
  if (a=1) arren=true;
    else arren=false;
  cout<<"dwse Baros tou Zwou: ";
  cin>> baros;
 }
  void showstoixeia();
 //απλή δήλωση
};
```

```
class Dog: public Animal
 {     //η κλάση Dog ΚΛΗΡΟΝΟΜΕΙ την κλάση Animal
 private:
  string onoma;
 public:
  Dog() {} //Δομητής χωρίς ορίσματα
     //Δομητής, καλεί το δομητή της κλάσης Animal
  Dog(float a,bool b,int c,string d):Animal(a,b,c)
  {       onoma = d; //εκτελεί και μία δική του εντολή
  }
 void onomazw(string name)
 {      onoma = name;
  }
  string showonoma()
 {       return onoma;
  }
};

void Animal::showstoixeia() {
//Ορισμός συνάρτησης-μέλους έξω από την κλάση
 cout<<"Ilikia zwou: "<<ilikia<<endl;
 cout<<"fyllo zwou: ";
 if (arren) cout<<"arseniko";
  else cout<<"thiliko";
 cout<<endl;
 cout<<"Baros zwou: "<<baros<<endl;}
```

ΠΑΡΑΔΕΙΓΜΑ

```
int main()
{
	//κατασκευή στη μνήμη αντικειμένου τύπου Dog με δομητή χωρίς ορίσματα
	Dog puppy1;
	cout<<"puppy1"<<endl;

	puppy1.dwsestoixeia();
	//πρόσβαση σε συνάρτηση-μέλος της Βασική κλάσης Animal, γιατί τα αντικείμενα της
	    Dog έχουν πρόσβαση στα public μέλη της Animal λόγω του ορισμού "class Dog:
	    public Animal“
	puppy1.onomazw("ektor");
	cout<<"to onoma tou skylou einai "<<puppy1.showonoma()<<endl;
	//πρόσβαση σε public μέλος της κλάσης

	//κατασκευή στη μνήμη αντικειμένου τύπου Dog με δομητή με ορίσματα
	Dog puppy2(4,false,17,"lassy");
	cout<<"puppy2:"<<endl;
	puppy2.showstoixeia();
	cout<<"to onoma tou skylou einai "<<puppy2.showonoma();
	return 0;
}
```

ΠΡΟΣΒΑΣΗ ΠΡΟΣΤΑΤΕΥΜΕΝΩΝ ΜΕΛΩΝ ΑΠΟ ΠΑΡΑΓΩΓΗ ΚΛΑΣΗ

- Εάν μέσα στο σώμα της κλάσης Dog ήθελα να προσθέσω την παρακάτω μέθοδο η οποία αυξάνει την ηλικία του σκύλου μου κατά ένα έτος:

```
void megalose1xrono()
{
        ilikia = ilikia + 1;
}
```

- τότε θα λάβω μήνυμα σφάλματος, το οποίο με ενημερώνει ότι η μεταβλητή ilikia δεν είναι προσπελάσιμη.
- αυτό συμβαίνει γιατί η μεταβλητή αυτή είναι ιδιωτική (private) της Animal.
- Πρόσβαση σε private μέλη μιας κλάσης έχω μόνο όσο γράφω κώδικα μέσα στο σώμα αυτής.
- Έξω από το σώμα της κλάσης έχω πρόσβαση μόνο στα δημόσια (public) μέλη.

ΠΡΟΣΒΑΣΗ ΠΡΟΣΤΑΤΕΥΜΕΝΩΝ ΜΕΛΩΝ ΑΠΟ ΠΑΡΑΓΩΓΗ ΚΛΑΣΗ

- Τι γίνεται όμως σε αυτή την περίπτωση, που θέλω μέσα στο σώμα μιας παράγωγης κλάσης να προσπελάσω μη δημόσια μέλη της βασικής κλάσης ;
- Εάν πρόκειται μία κλάση να την κληρονομήσω σε μία νέα, τότε τα μέλη στα οποία θέλω να έχω πρόσβαση από το σώμα της παράγωγης κλάσης θα είναι είτε δημόσια (public) είτε προστατευμένα (protected).
- Τα δημόσια μέλη είναι προσπελάσιμα από οποιοδήποτε σημείο έξω από το σώμα της κλάσης. Τα προστατευμένα μέλη είναι σαν ιδιωτικά για οποιοδήποτε σημείο του προγράμματος εκτός από το σώμα της παράγωγης κλάσης . Έτσι εάν τα δεδομένα της κλάσης Animal τα ορίσω ως προστατευμένα:

```
protected:
        float ilikia;
        bool arren;
        int baros;
```

- τότε αυτά θα είναι προσπελάσιμα από το σώμα της παράγωγης κλάσης Dog, αλλά όχι από αντικείμενα της κλάσης Animal.
- Σε κάθε περίπτωση τα αντικείμενα έχουν πρόσβαση μόνο σε public μέλη.