Δομημένος προγραμματισμός

Εισαγωγή στη C

Λίγα λόγια για την C

- Γλώσσα προγραμματισμού υψηλού επιπέδου.
- Σχεδιάστηκε και υλοποιήθηκε από τον Dennis Ritchie στις αρχές της δεκαετίας του 1970 (Bell Labs).
- Η γλώσσα ονομάστηκε C, διότι είναι αποτέλεσμα εργασίας για τη δημιουργία μιας άλλης παλαιότερης γλώσσας με όνομα B (Ken Thompson).
- Ο βασικός σκοπός της δημιουργίας της γλώσσας C ήταν η χρήση της για την ανάπτυξη του λειτουργικού συστήματος UNIX (πρόδρομος του Linux, Android κλπ).
- Αναφέρεται συχνά σαν μια γλώσσα «μεσαίου επιπέδου».
- Είναι αρκετά διαδεδομένη και χρησιμοποιείται για τη δημιουργία λογισμικού τόσο για συστημάτων (system) όσο και για εφαρμογές (applications).

Bell Laboratories

Η ιστορία της C

- 1969-1973: AT&T Bell Labs από Dennis Ritchie ως συνέχεια των BCPL, B (ανάπτυξη του UNIX).
- 1978: "The C Programming Language" K&R: Kernigham & Ritchie. Η πρώτη περιγραφή της C.
- 1983: Σύσταση ANSI Standardization Committee X3J11.
- 1989: Οριστικοποίηση και αποδοχή του προτύπου ANSI/ISO Standard (ANSI C).
- 1990: Το ANSI C γίνεται διαθέσιμο στο κοινό. Από τότε όλοι οι μεταγλωττιστές C υποστηρίζουν το συγκεκριμένο πρότυπο.
- 1990-99: Αναθεώρηση του προτύπου (μικρές αλλαγές, C99).
- Σήμερα: όλοι οι μεταγλωττιστές της C συμμορφώνονται με το πρότυπο ANSI C και πολλοί με το C99.
- Επίσης, το 2011 και 2018 δημοσιεύτηκαν τα πρότυπα C11 και C18

ANSI: American National Standards Institute

Χαρακτηριστικά της C (πλεονεκτήματα)

- Ευέλικτη (λίγοι περιορισμοί) και πολύ γρήγορη.
- Λιτή (λίγες δεσμευμένες λέξεις) και απλό συντακτικό.
- Κώδικας που:
  - μεταφέρεται και εκτελείται σε διαφορετικά συστήματα.
  - διαβάζεται και συντηρείται εύκολα
- Υποστηρίζει τον δομημένο προγραμματισμό και στηρίζεται στην ύπαρξη και κλήση συναρτήσεων.
- Υποστηρίζει προγραμματισμό χαμηλού επιπέδου (ακόμα και την ενσωμάτωση assembly κώδικα).
- Παρέχει έτοιμες συναρτήσεις.
- Υπάρχει μεγάλη εγκατεστημένη βάση προγραμμάτων.
- Αποτελεί το πρώτο βήμα για την εκμάθηση κάποιας γλώσσας αντικειμενοστραφούς προγραμματισμού.

Η διαδικασία μεταγλώττισης κώδικα C

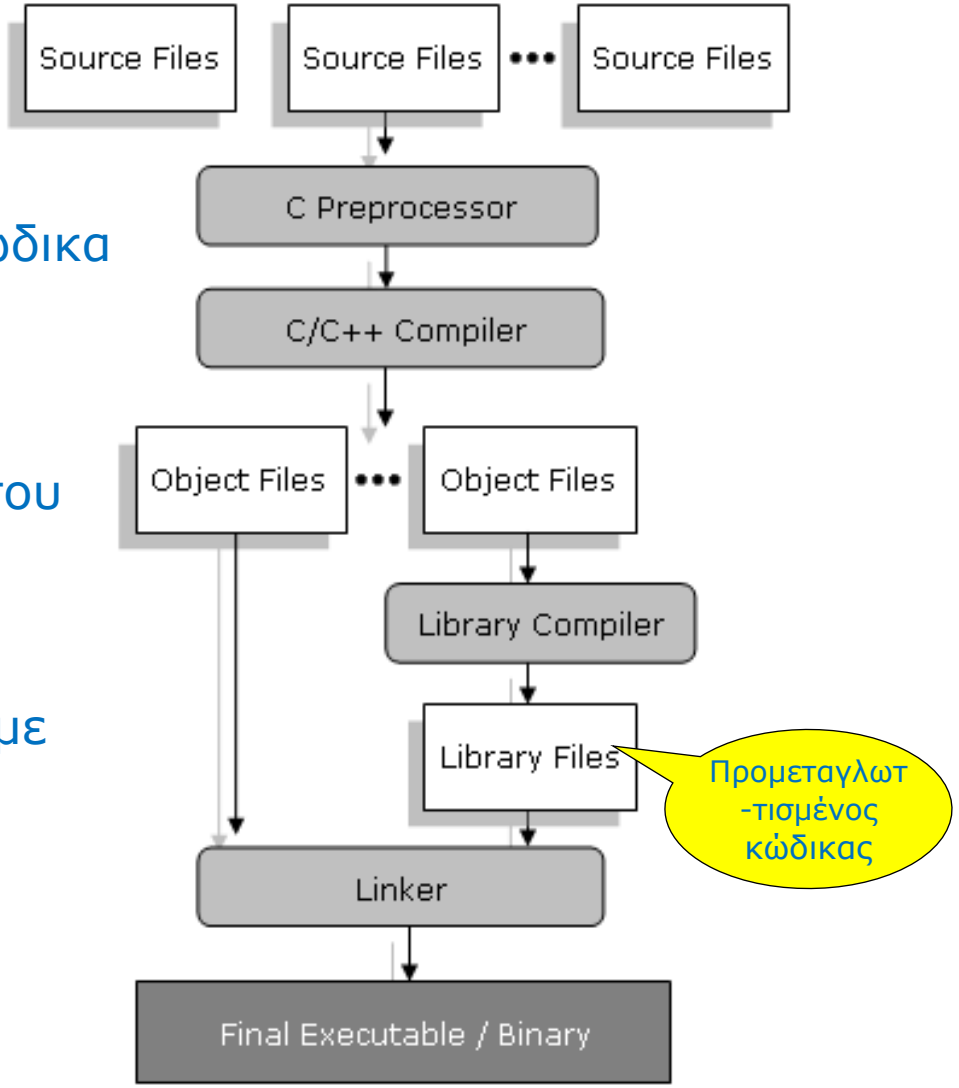
Συγγραφή και αποθήκευση αρχείου κώδικα (πηγαίο πρόγραμμα)

Μεταγλώττιση του πηγαίου κώδικα

Δημιουργία «αντικείμενου» κώδικα (object). Μεταγλωττισμένα τμήματα του κώδικα που δεν περιέχονται στις βιβλιοθήκες.

Σύνδεση του αντικείμενου κώδικα με τις βιβλιοθήκες (περιέχουν προ-μεταγλωττισμένο κώδικα)

Δημιουργία κώδικα μηχανής (εκτελέσιμο αρχείο)



Το "κλασσικό" πρώτο σας πρόγραμμα

```
#include  <stdio.h>
main()
{
    printf(" Hello World!\n");
}
```

#include <stdio.h>

- Οδηγία προ-επεξεργαστή. Πληροφορεί τον compiler της C, ότι θα γίνει χρήση της βιβλιοθήκης προ-μεταγλωττισμένων συναρτήσεων (υπο-προγραμμάτων) ***stdio.h***
- Συγκεκριμένα, η προκαθορισμένη (standard) βιβλιοθήκη ***stdio.h*** αναφέρεται σε συναρτήσεις που έχουν να κάνουν με την είσοδο και έξοδο δεδομένων (**st**andard **i**nput **o**utput).
- Οι έτοιμες βιβλιοθήκες της C όπως η ***stdio.h***, περιέχουν προμεταγλωττισμένες συναρτήσεις για συγκεκριμένο υπολογιστικό σύστημα που ελέγχεται από συγκεκριμένο λειτουργικό σύστημα.

Η οδηγία #include

- Υπάρχει σχεδόν πάντα στην αρχή κάθε προγράμματος και περιλαμβάνει οδηγίες προς τον προ-μεταγλωττιστή.
- Συνήθως, τα αρχεία που περιγράφονται στη οδηγία ***#include*** έχουν κατάληξη ***.h*** και ονομάζονται αρχεία επικεφαλίδας (header files).
- Η οδηγία ***#include*** ξεκινάει πάντα με το σύμβολο **#** και **ΔΕΝ** τελειώνει με ελληνικό ερωτηματικό (**;**)
- Κάποιος προγραμματιστής μπορεί να δημιουργήσει δικά του αρχεία επικεφαλίδας (βιβλιοθήκες συναρτήσεων) και να τα συμπεριλάβει (#include) στο πρόγραμμά του.
- Σύνταξη:

  #include <standard header file>

  #include "programmer's header file"

```
#include  <stdio.h>
main()
{
    printf(" Hello World!\n");
}
```

Η συνάρτηση main()

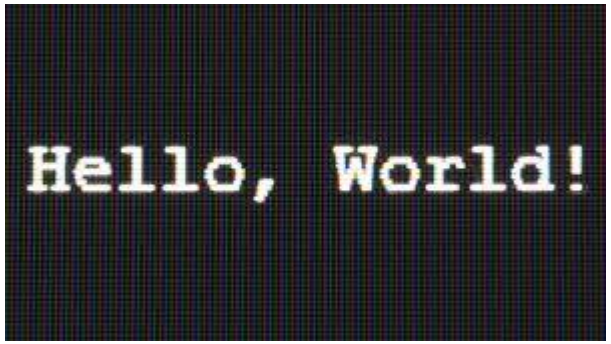
```
#include  <stdio.h>
main()
{
    printf(" Hello World \n");
}
```


```
#include  <stdio.h>
main()
{
}
```

- Ένα πρόγραμμα της C ξεκινά με την κλήση της κύριας συνάρτησης με όνομα main (κύρια συνάρτηση). Υπάρχει ΜΟΝΟ μια συνάρτηση **main**().
- Στην C τα προγράμματα και υποπρογράμματα ονομάζονται συναρτήσεις.
- Το ζευγάρι των παρενθέσεων που ακολουθεί τη λέξη-κλειδί main, υποδεικνύει στον μεταφραστή (compiler) της γλώσσας προγραμματισμού C, ότι πρόκειται για μία συνάρτηση.
- Τα άγκιστρα **{** και **}** δηλώνουν την αρχή και το τέλος του προγράμματος, αλλά και όπως θα δούμε στην συνέχεια την αρχή και το τέλος μιας ενότητας (block) του προγράμματος.
- Οι εντολές της C ενός προγράμματος πρέπει να περιέχονται μέσα σε άγκιστρα.

Η συνάρτηση printf ()

```
#include  <stdio.h>
main()
{
    printf(" Hello, World! \n");
}
```



- Η *συνάρτηση* **printf()** καλείται από την κύρια συνάρτηση main() για να εμφανίσει την ακολουθία των χαρακτήρων που βρίσκονται μέσα στα εισαγωγικά (δηλαδή το Hello Word).
- Βρίσκεται προ-μεταγλωττισμένη στην βιβλιοθήκη ***stdio.h***
- Το **\n** στο τέλος της εντολής printf (χαρακτήρας διαφυγής ή ελέγχου) δημιουργεί μια νέα γραμμή μετά την εμφάνιση του μηνύματος στην οθόνη.
- Οι εντολές τελειώνουν πάντα με το ελληνικό ερωτηματικό (**;**).

Ένα πρόγραμμα με σχόλια

```
/* This is a comment. It is ignored by the compiler */
#include  <stdio.h>
main()
{
      printf(" Hello World \n ");
       printf(" My name is: "); /* A comment is
       allowed to be  continued on another line */
       printf("Ritsa \n"); // from Computeritsa
   // single line comments
}
```

- Τα σχόλια περικλείονται μεταξύ των χαρακτήρων **/*** (αρχή του σχολίου) και ***/** (τέλος σχολίου). Μπορούμε να τα βάλουμε σε οποιοδήποτε σημείο του προγράμματος, αγνοούνται από τον μεταφραστή και είναι δυνατόν να συνεχίζονται σε άλλη γραμμή.
- Σχόλια επίσης μπορούν να εισαχθούν στο πρόγραμμα με τη χρήση των **//**. Η διαφορά είναι ότι αυτού του είδους σχόλια δεν επεκτείνονται σε 2η γραμμή (μιας γραμμής).

Το πρώτο σας πρόγραμμα ...αλλιώς

```
#include  <stdio.h>
main()
{
   printf(" Hello World \n"); }
```

```
#include  <stdio.h>
 int main()  {
      printf(" Hello World \n");
      return 0;
    }
```

```
#include  <stdio.h>
 void main()
   {
      printf(" Hello World \n");
      return 0;
   }
```

```
#include  <stdio.h>
 int main(void)    {
      printf(" Hello World \n");  return 0;
   }
```

- int πριν τη main(): *η main() επιστρέφει έναν ακέραιο (integer) ο οποίος δημιουργείται από την return (άλλη συνάρτηση).*
- return 0: *επιστρέφει τη τιμή 0*
- void πριν την main(): *η main()δεν επιστρέφει τιμή*
- void στις παρενθέσεις της main(): *η main δεν δέχεται ορίσματα*

Το πρώτο σας πρόγραμμα πολύ αλλιώς

```
#include  <stdio.h>
 int main(void)
 {
   char *message[] = {"Hello ", "World"};
   int i;

   for(i = 0; i < 2; ++i)
            printf("%s", message[i]);
   printf("\n");
   getchar();
 }
```

Το βάζουμε όταν δημιουργούμε προγράμματα σε περιβάλλον εργασίας που χρησιμοποιούμε για την ανάπτυξη προγραμμάτων (Dev C). Ελέγχουμε με αυτό τον τρόπο το κλείσιμο της εκτέλεσης του προγράμματος. Υπάρχει και άλλος τρόπος.

Και ένα άλλο λίγο πιο δύσκολο

```
/* program to find factorial of 6 */
#include <stdio.h>
#define VALUE 6
int i, j;
main()
  {
      j=1;
     for (i=1; i<=VALUE; i++)
            j=j*i;
     printf("The factorial of  %d is %d\n", VALUE, j);
}
```

6 είναι η τιμή που δίνουμε στη σταθερά VALUE με την οδηγία #define του προεπεξεργαστή

Χρησιμοποιούνται δύο μεταβλητές (η i και η j)

1 η αρχική τιμή για τη μεταβλητή j

Η for είναι μία από τις εντολές επανάληψης

Η j πολλαπλασιάζεται κατ' επανάληψη με τις τιμές του i

Εμφανίζονται στην οθόνη ο αριθμός 6 και το 6 παραγοντικό

Παρατηρήσεις

- Η C είναι μια γλώσσα προγραμματισμού που κάνει διάκριση μεταξύ πεζών και κεφαλαίων γραμμάτων (case sensitive). Άλλο το Printf() και άλλο το printf().
- Ένα μήνυμα λάθους που εμφανίζει ένας μεταγλωττιστής μπορεί να μην συμβαίνει στη γραμμή που υποδεικνύει, αλλά στις αμέσως προηγούμενες γραμμές.
- Αν ο μεταγλωττιστής εμφανίζει πολλά μηνύματα λάθους, τότε προσπαθήστε να βρείτε και να διορθώσετε μόνο το πρώτο λάθος. Μια νέα μεταγλώττιση μπορεί να εμφανίσει λιγότερα ή και καθόλου λάθη.
- Ο μεταγλωττιστής μπορεί να εμφανίσει μόνο μηνύματα προειδοποίησης (warnings). Καλό είναι να μην τα αγνοείτε.
- Οι μεταγλωττιστές ανιχνεύουν λανθασμένη εφαρμογή των κανόνων της γλώσσας (π.χ. ορθογραφικά λάθη, συντακτικά λάθη, αδήλωτες μεταβλητές, ...) και όχι λάθη λογικής που ενδέχεται να περιέχονται στο πρόγραμμά σας.

Για εξάσκηση. Στο εργαστήριο ή αλλού…

- Εξοικειωθείτε με το περιβάλλον εργασίας του εργαστηρίου ή που έχετε εγκαταστήσει στο υπολογιστή σας.
- Δημιουργήστε, αποθηκεύστε, μεταγλωττίστε και εκτελέστε τα προγράμματα που παρουσιάστηκαν στη διάλεξη της θεωρίας.
- Δημιουργήστε κάποιο λάθος (π.χ. αφαίρεση ενός ερωτηματικού ή / και μια ανορθόγραφη εντολή/συνάρτηση) και μελετήστε τα μηνύματα λάθους του μεταγλωττιστή.
- Γράψτε ένα δικό σας πρόγραμμα το οποίο θα τυπώνει:
  - (α) το όνομα και το επώνυμο σας στη ίδια γραμμή και
  - (β) το τμήμα σας σε άλλη γραμμή
- Αναζητήστε στο διαδίκτυο πηγές και εγχειρίδια εκμάθησης για την γλώσσα προγραμματισμού C.