Εισαγωγικά της C

Δομημένος προγραμματισμός

Ο **δομημένος προγραμματισμός** είναι ένα πρότυπο (paradigm) προγραμματισμού που βασίζεται στη **λογική διάσπαση** του προγράμματος σε **σαφώς καθορισμένα, μικρά και επαναχρησιμοποιήσιμα τμήματα**. Η ιδέα είναι να γράφουμε **καθαρό, κατανοητό και εύκολα συντηρήσιμο** κώδικα.

Κύριες αρχές του δομημένου προγραμματισμού

1. **Ακολουθία (Sequence)**
   Οι εντολές εκτελούνται με τη σειρά που γράφονται.
2. **Επιλογή (Selection)**
   Χρήση εντολών ελέγχου ροής, όπως if, if...else, switch, ώστε το πρόγραμμα να παίρνει αποφάσεις.
3. **Επανάληψη (Iteration / Looping)**
   Επαναλαμβάνουμε εντολές με for, while, do...while.
4. **Διαίρεση σε υποπρογράμματα (Modularity)**
   Ο κώδικας χωρίζεται σε **συναρτήσεις (functions)** που εκτελούν συγκεκριμένες εργασίες.

Διαδικασία Μεταγλώττισης (Compilation Process)

Η C είναι **μεταγλωττιζόμενη γλώσσα** (compiled language). Αυτό σημαίνει ότι ο κώδικας που γράφεις (source code) δεν εκτελείται άμεσα, αλλά πρέπει πρώτα να **μετατραπεί σε εκτελέσιμη μορφή**.

Τα βασικά στάδια είναι:

Όταν γράφουμε ένα πρόγραμμα στη **C**, η διαδρομή είναι η εξής:

1. **Προεπεξεργαστής (Preprocessor)**
   ➤ Επεξεργάζεται οδηγίες όπως `#include`, `#define`.
2. **Compiler (Μεταγλωττιστής)**
   ➤ Μεταφράζει τον κώδικα C σε **assembly code**.
   (π.χ. `program.c` → `program.s`)
3. **Assembler**
   ➤ Μεταφράζει το αρχείο **assembly** σε **αντικειμενικό αρχείο** (object file).
   (π.χ. `program.s` → `program.o` ή `program.obj`)
4. **Linker**
   ➤ Ενώνει όλα τα object files και βιβλιοθήκες σε ένα **εκτελέσιμο πρόγραμμα**.
   (π.χ. `program.o` → `program.exe`)

Τι είναι ο Compiler

Ο **compiler (μεταγλωττιστής)** είναι ένα πρόγραμμα που:

- **Μεταφράζει** τον κώδικα της C (π.χ. program.c)
- Σε **μηχανικό κώδικα** που καταλαβαίνει ο υπολογιστής.

Τι είναι το IDE

Το **IDE (Integrated Development Environment)** είναι ένα **περιβάλλον ανάπτυξης** που περιλαμβάνει:

- **Κειμενογράφο** (για να γράφεις τον κώδικα),
- **Compiler/Linker** (για μεταγλώττιση),
- **Debugger** (για εντοπισμό λαθών),
- **Εργαλεία εκτέλεσης και ανάλυσης**.

Τι παράγει ο προγραμματιστής

Ο προγραμματιστής γράφει **πηγαίο κώδικα (source code)** — δηλαδή αρχεία `.c` (και `.h` για κεφαλίδες).

Μετά τη μεταγλώττιση και σύνδεση, παράγεται ένα **εκτελέσιμο πρόγραμμα** που μπορεί να τρέξει στον υπολογιστή.

Παράδειγμα 1: Hello World

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

Ανάλυση γραμμή-γραμμή

1. #include <stdio.h>. **Είναι** οδηγία προεπεξεργαστή **(ξεκινά με #). Λέει στον μεταγλωττιστή να** εισάγει **το αρχείο κεφαλίδας (header file) `stdio.h`, το οποίο περιέχει δηλώσεις συναρτήσεων όπως η `printf`.`stdio` σημαίνει *Standard Input/Output* — δηλαδή για εισαγωγή/εξαγωγή δεδομένων.**
2. **`int main(void)` Δηλώνει τη** κύρια συνάρτηση **του προγράμματος, από όπου ξεκινά η εκτέλεση.**
3. **Κάθε πρόγραμμα C πρέπει να έχει** μία **συνάρτηση `main()`.Το `int` σημαίνει ότι η συνάρτηση**

επιστρέφει έναν ακέραιο αριθμό **(integer).Το** `(void)` **δηλώνει ότι** δεν δέχεται παραμέτρους**.**

4. **Οι** αγκύλες **ορίζουν το** σώμα (body) **της συνάρτησης, δηλαδή τις εντολές που θα εκτελεστούν.**
5. `printf("Hello, world!\n");` **Η συνάρτηση** `printf()` **εκτυπώνει κείμενο στην οθόνη. Το** `"Hello, world!\n"` **είναι** αλφαριθμητικό (string) **που θα εμφανιστεί στην κονσόλα. Το** `\n` **σημαίνει** νέα γραμμή **(newline) — ο cursor πάει στην επόμενη γραμμή μετά την εκτύπωση.**
6. **Το** `return 0;` **Επιστρέφει την τιμή** `0` **στο λειτουργικό σύστημα, δηλώνοντας ότι το πρόγραμμα** τερματίστηκε επιτυχώς**. Αν επέστρεφε διαφορετικό αριθμό, συνήθως αυτό σημαίνει** σφάλμα **ή** ειδική κατάσταση**.**

Τι είναι Μεταβλητή

Μια **μεταβλητή** (variable) είναι ένας **χώρος στη μνήμη του υπολογιστή** που έχει:

- **Όνομα** (identifier), π.χ. `age`
- **Τύπο δεδομένων** (data type), π.χ. `int`
- **Τιμή**, π.χ. `25`

Στη C, **πρέπει πάντα να δηλώσεις** τον τύπο της μεταβλητής πριν τη χρησιμοποιήσεις.

Παράδειγμα

```
int age = 25; float height = 1.75; char grade = 'A';
```

Εδώ:

- `int` → δηλώνει ακέραιο αριθμό
- `float` → δηλώνει δεκαδικό αριθμό
- `char` → δηλώνει χαρακτήρα

Κανόνες Ονοματοδοσίας Μεταβλητών (Identifiers)

Τα **ονόματα μεταβλητών** (identifiers) στη C:

1. Μπορούν να περιέχουν **γράμματα**, **ψηφία**, και **underscore** (_).
2. **Δεν** μπορούν να αρχίζουν με αριθμό.
3. Είναι **case sensitive** (Age ≠ age).
4. **Δεν** μπορούν να είναι δεσμευμένες λέξεις (π.χ. `int`, `for`, `return`).

Σωστά: `myAge`, `total_sum`, `_counter`

Λάθος: `2value`, `int`, `my-age`

Κύριοι Τύποι Δεδομένων

Τύπος	**Περιγραφή**	**Παράδειγμα δήλωσης**	**Εύρος τιμών (ενδεικτικά)**	**Μέγεθος (bytes)**
`int`	Ακέραιος αριθμός	`int x = 10;`	περίπου −32,768 έως 32,767 (ή −2B έως 2B σε 32bit)	2 ή 4
`float`	Δεκαδικός αριθμός (μονής ακρίβειας)	`float y = 3.14;`	~ $\pm 3.4 \times 10^{38}$	4
`double`	Δεκαδικός αριθμός (διπλής ακρίβειας)	`double z = 3.141592;`	~ $\pm 1.7 \times 10^{308}$	8
`char`	Ένας χαρακτήρας (γράμμα, σύμβολο, ψηφίο)	`char grade = 'A';`	1 byte	1
`void`	Χωρίς τύπο (π.χ. για συναρτήσεις που δεν επιστρέφουν τιμή)	`void func()`	—	—

Προσδιοριστές (Type Modifiers)

Οι **προσδιοριστές τύπου (modifiers)** αλλάζουν το **εύρος** και το **μέγεθος** του βασικού τύπου.

Προσδιοριστής	Τι κάνει	Παράδειγμα	Περιγραφή
short	μικρότερος ακέραιος	short int s;	συνήθως 2 bytes
long	μεγαλύτερος ακέραιος	long int l;	συνήθως 4 ή 8 bytes
unsigned	μόνο θετικές τιμές	unsigned int u;	διπλασιάζει το θετικό εύρος
signed	θετικές και αρνητικές (προεπιλογή)	signed char c;	μπορεί να είναι αρνητικό ή θετικό

Παράδειγμα:

```
unsigned int population = 120000;
short int age = 18;
 long double pi = 3.1415926535;
```

Παράδειγμα προγράμματος

```
#include <stdio.h>
 int main(void)
{
int age = 20;
float height = 1.82;
char grade = 'A';
unsigned int score = 95;
 printf("Age: %d\n", age);
printf("Height: %.2f\n", height);
 printf("Grade: %c\n", grade);
 printf("Score: %u\n", score);
```

```
return 0;
}
```

- **Σημειώσεις για τα format specifiers της** printf**:**

Τύπος	**Ειδικός Χαρακτήρας**	**Παράδειγμα**
int	%d ή %i	printf("%d", x);
float / double	%f και %lf για double	printf("%.2f", y);
char	%c	printf("%c", ch);
unsigned int	%u	printf("%u", u);
string (π.χ. char name[] = "Nikos";)	%s	printf("%s", name);

Οι **εντολές εισόδου και εξόδου** (I/O = *Input / Output*) είναι βασικό κομμάτι του προγραμματισμού στη C, γιατί επιτρέπουν την **επικοινωνία του προγράμματος με τον χρήστη**.

1. Εντολές Εξόδου (Output)

Η βασική εντολή εξόδου στη C είναι η printf() (print formatted).
Χρησιμοποιείται για να **εμφανίζει δεδομένα** στην οθόνη.

Μορφή:

```
printf("Κείμενο ή μορφοποιημένη έξοδος", μεταβλητές);
```

Παράδειγμα 1:

```
printf("Hello, world!\n");
```

Εμφανίζει:

```
Hello, world!
```

Παράδειγμα 2 (με μεταβλητές):

```
int age = 20;
```

```
float height = 1.80;
printf("Η ηλικία μου είναι %d και το ύψος μου %.2f μέτρα.\n", age, height);
```

Εμφανίζει:

```
Η ηλικία μου είναι 20 και το ύψος μου 1.80 μέτρα.
```

Format Specifiers (Μορφοποιητές)

Τύπος Δεδομένων	Format Specifier	Παράδειγμα
int	%d ή %i	printf("%d", x);
float	%f	printf("%f", y);
double	%lf	printf("%lf", z);
char	%c	printf("%c", ch);
string (char[])	%s	printf("%s", name);
unsigned int	%u	printf("%u", num);

Μπορούμε να ελέγξουμε **πόσα δεκαδικά** θα εμφανιστούν:
%.2f → 2 δεκαδικά ψηφία.

2. Εντολές Εισόδου (Input)

Η βασική εντολή εισόδου είναι η scanf() (*scan formatted*).
Χρησιμοποιείται για να **διαβάζει τιμές που εισάγει ο χρήστης** από το πληκτρολόγιο.

Μορφή:

```
scanf("μορφή", &μεταβλητή);
```

Το σύμβολο & **(ampersand)** είναι απαραίτητο, γιατί περνάμε τη **διεύθυνση μνήμης** της μεταβλητής — δηλαδή πού θα αποθηκευτεί η τιμή.

Παράδειγμα 1:

```
int age;
printf("Δώσε την ηλικία σου: ");
 scanf("%d", &age);
printf("Η ηλικία σου είναι %d.\n", age);
```

Παράδειγμα εκτέλεσης:

```
Δώσε την ηλικία σου: 25
Η ηλικία σου είναι 25.
```

Παράδειγμα 2 (πολλαπλές τιμές):

```
int a, b;
 printf("Δώσε δύο αριθμούς: ");
 scanf("%d %d", &a, &b);
 printf("Το άθροισμά τους είναι %d.\n", a + b);
```

Εκτέλεση:

```
Δώσε δύο αριθμούς: 5 7
Το άθροισμά τους είναι 12.
```

Παράδειγμα 3 (δεκαδικοί αριθμοί και χαρακτήρες):

```
float height;
 char grade;
 printf("Δώσε ύψος: ");
 scanf("%f", &height);
printf("Δώσε βαθμό: ");
scanf(" %c", &grade); // το κενό πριν το %c αγνοεί τα newline
 printf("Υψος: %.2f, Βαθμός: %c\n", height, grade);
```

Σύνοψη: **printf()** και **scanf()** μαζί

Ενέργεια	**Συνάρτηση**	**Παράδειγμα**
Εμφάνιση στην οθόνη	printf()	printf("Age: %d", age);
Ανάγνωση από χρήστη	scanf()	scanf("%d", &age);

Συνηθισμένα Λάθη

1. Ξεχνάμε το & στο scanf() → **δε διαβάζει σωστά**.
2. Σωστό: scanf("%d", &x);

3. Ξεχνάμε το κενό πριν το %c όταν διαβάζουμε χαρακτήρα μετά από αριθμό.
   Σωστό: scanf(" %c", &ch);
4. Δίνουμε λάθος format specifier.
   Αν float, χρησιμοποίησε %f — όχι %d.