

ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΥΠΟΛΟΓΙΣΤΩΝ & ΥΠΟΛΟΓΙΣΤΙΚΗ ΦΥΣΙΚΗ

Μέρος 3ο

ΝΙΚΟΛΑΟΣ ΣΤΕΡΓΙΟΥΛΑΣ



ΤΜΗΜΑ ΦΥΣΙΚΗΣ

ΑΡΙΣΤΟΤΕΛΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΟΝΙΚΗΣ

ΕΚΤΥΠΩΣΗ ΚΕΙΜΕΝΟΥ

Ένα **κείμενο** μπορεί να εκχωρηθεί ως τιμή μιας μεταβλητής **char** με **αόριστο μέγεθος []**. Μπορούμε να καθορίσουμε να εκτυπωθεί μέρος ή ολόκληρο το κείμενο:

```
#include <stdio.h>

int main(void)
{
    char msg_en[] = "This is a text";
    char msg_gr[] = "Αυτό είναι ένα κείμενο";

    printf("%s\n", msg_en);
    printf("%s\n", "This is a text");
    printf("%s\n", msg_gr);
    printf("%s\n", "Αυτό είναι ένα κείμενο");
    printf("%.6s\n", msg_en);
    return 0;
}
```

```
This is a text
This is a text
Αυτό είναι ένα κείμενο
Αυτό είναι ένα κείμενο
This i
```

ΠΡΟΘΕΜΑ

Για την εμφάνιση ενός **short** ακεραίου μπορεί να χρησιμοποιηθεί προαιρετικά το γράμμα **h**, ενώ για την εμφάνιση ενός **long** ακεραίου μπορεί να χρησιμοποιηθεί το γράμμα **l** ή **L**, όπως φαίνεται στο παρακάτω παράδειγμα:

```
#include <stdio.h>

int main(void)
{
    short a = 10;
    long b = 10000;

    printf("%hd %ld\n", a, b);
    return 0;
}
```

```
10 10000
```

ΣΗΜΑΙΕΣ

Οι **σημαίες** χρησιμοποιούνται για περαιτέρω μορφοποίηση των εμφανιζόμενων τιμών, όπως φαίνεται στον παρακάτω πίνακα

Σημαία	Σημασία
-	Η έξοδος στο πεδίο πλάτους στοιχίζεται αριστερά (εξ' ορισμού η έξοδος στοιχίζεται δεξιά).
+	Προσθέτει το πρόσημο + μπροστά από τις θετικές τιμές.
κενός χαρακτήρας	Προσθέτει τον κενό χαρακτήρα μπροστά από τις θετικές τιμές.
#	Προσθέτει το 0 μπροστά από τις οκταδικές τιμές και το 0x ή το 0X μπροστά από δεκαεξαδικές τιμές.
0	Προσθέτει όσα μηδενικά χρειάζονται μπροστά από την εμφανιζόμενη τιμή, ώστε να καλυφθεί το πεδίο πλάτους.

ΠΑΡΑΔΕΙΓΜΑ

```
#include <stdio.h>

int main(void)
{
    int a = 12;

    printf("%4d\n", a);
    printf("%-4d\n", a);
    printf("%04d\n", a);
    printf("%#0x\n", a);
    printf("%#o\n", a);

    return 0;
}
```

```
   12
12
0012
014x
014
```

ΑΚΡΙΒΕΙΑ ΑΡΙΘΜΩΝ

```
#include <stdio.h>

int main(void)
{
    float a = 1.2345678000000000000;
    double b = 1.2345678000000000000;

    printf("Value = %.20f\n", a);
    printf("Value = %.20f\n", b);

    return 0;
}
```

```
Value = 1.23456776142120361328
Value = 1.234567799999999999298
```

← (μόνο τα πρώτα 6 δεκαδικά ψηφία είναι σωστά)

← (τυχαία ψηφία)

← (μόνο τα 15 δεκαδικά ψηφία είναι σωστά αν γίνει στρογγυλοποίηση)

ΠΑΡΑΤΗΡΗΣΕΙΣ

Για να εκτυπώσουμε δύο γραμμές συνεχόμενα ως μία γραμμή χρησιμοποιούμε το \.

```
#include <stdio.h>

int main(void)
{
    printf("Αυτή η εντολή printf μπορεί να γραφεί σε \
δύο γραμμές, αλλά εκτυπώνεται σε μία.\n");

    return 0;
}
```

Αυτή η εντολή printf μπορεί να γραφεί σε δύο γραμμές, αλλά εκτυπώνεται σε μία.

ΠΑΡΑΤΗΡΗΣΕΙΣ

Για να εκτυπώσουμε το σύμβολο % το γράφουμε δύο φορές ως %% .

π.χ. η εντολή `printf("%d%%\n", 100);`

τυπώνει: 100%

ΜΕΤΑΤΡΟΠΗ ΤΥΠΟΥ

Υπάρχουν περιπτώσεις όπου ένας τύπος δεδομένων πρέπει να μετατραπεί προσωρινά σε κάποιον άλλο τύπο δεδομένων.

Π.χ. έστω ότι η μεταβλητή **a** έχει δηλωθεί ως

```
int a;
```

Τότε, σε κάποιο σημείο του προγράμματος (και μόνο εκεί) η έκφραση

```
(float)a
```

μετατρέπει προσωρινά τον τύπο της **a** από **int** σε **float**.

ΠΑΡΑΔΕΙΓΜΑ

```
#include <stdio.h>

int main(void)
{
    int i = 20, j = 30;
    float ratio;

    ratio = i/j;
    printf("%f\n", ratio);

    ratio = (float)i/j;
    printf("%f\n", ratio);

    return 0;
}
```

```
0.000000
0.666667
```

ΠΑΡΑΔΕΙΓΜΑ

Με το παρακάτω πρόγραμμα, εκτυπώνουμε τον κώδικα ASCII ενός χαρακτήρα.

```
#include <stdio.h>

int main(void)
{
    char ch;

    printf("Καταχωρείστε έναν χαρακτήρα:\n");
    scanf("%c",&ch);
    printf("Ο κώδικας ASCII του χαρακτήρα %c είναι %d.\n", ch, (int)ch);

    return 0;
}
```

Καταχωρείστε έναν χαρακτήρα:

F

Ο κώδικας ASCII του χαρακτήρα F είναι 70.

ΠΑΡΑΔΕΙΓΜΑ

Επειδή οι αριθμοί `double` έχουν ακρίβεια **15 ψηφίων** μόνο, είναι πολύ εύκολο να έχουμε λανθασμένα αποτελέσματα όταν κάνουμε πράξεις με μεγάλους αριθμούς.

```
int main(void)
{
    double a, b;

    b = 1.0e15 + 1.0;
    a = b - 1.0e15;
    printf("a = %f\n", a);

    b = 1.0e16 + 1.0;
    a = b - 1.0e16;
    printf("a = %f\n", a);

    return 0;
}
```

```
a = 1.000000
a = 0.000000
```

ΠΑΡΑΔΕΙΓΜΑ

Το ίδιο παράδειγμα με αριθμούς **float** που έχουν ακρίβεια **6 ψηφίων** μόνο.

```
int main(void)
{
    float a, b;

    b = 1.0e6 + 1.0;
    a = b - 1.0e6;
    printf("a = %f\n", a);

    b = 1.0e8 + 1.0;
    a = b - 1.0e8;
    printf("a = %f\n", a);

    return 0;
}
```

```
a = 1.000000
a = 0.000000
```

Η ΣΥΝΑΡΤΗΣΗ `scanf`

Η συνάρτηση `scanf()` χρησιμοποιείται για την είσοδο δεδομένων από το πληκτρολόγιο.

Χρησιμοποιεί παρόμοια σύνταξη με την `printf`.

π.χ.

```
int i;  
scanf("%d", &i);
```

Ο χαρακτήρας `&`, που μπαίνει πριν από το όνομα της μεταβλητής, ονομάζεται **τελεστής διεύθυνσης** και χρησιμοποιείται για να αποθηκευτεί η τιμή που θα εισάγει ο χρήστης στη **διεύθυνση μνήμης** της μεταβλητής `i`.

Η ΣΥΝΑΡΤΗΣΗ `scanf`

Μπορούμε να διαβάσουμε πολλές μεταβλητές:

```
int i;
```

```
float j;
```

```
double k;
```

```
scanf ("%d%f%lf", &i, &j, &k);
```

Εισάγουμε τις τιμές στο πληκτρολόγιο με **κενό** ανάμεσα στους αριθμούς και στο τέλος πατάμε `enter`.

Για την εισαγωγή τιμών σε μεταβλητή `double` χρησιμοποιούμε το **`%lf`**.

ΠΑΡΑΔΕΙΓΜΑ

```
#include <stdio.h>

int main(void)
{
    int i;

    double a;

    printf("Δώσε έναν ακέραιο αριθμό\n");
    scanf("%d", &i);
    printf("Ο ακέραιος αριθμός είναι ο %d\n", i);

    printf("Δώσε έναν πραγματικό αριθμό\n");
    scanf("%lf", &a);
    printf("Ο πραγματικός αριθμός είναι ο %.20e\n", a);

    return 0;
}
```

Δώσε έναν ακέραιο αριθμό

3

Ο ακέραιος αριθμός είναι ο 3

Δώσε έναν πραγματικό αριθμό

3e25

Ο πραγματικός αριθμός είναι ο 3.000000000000000000005704e+25

Η ΣΥΝΑΡΤΗΣΗ `scanf`

Με το `%c` μπορούμε να διαβάσουμε **χαρακτήρες**:

```
char ch;
```

```
scanf ("%c", &ch);
```

Με το `%s` μπορούμε να διαβάσουμε **λέξεις**, π.χ. μέχρι 10 χαρακτήρων:

```
char str[10];
```

```
scanf ("%s", str);
```

Σε αυτή την περίπτωση **δε γράφουμε `&str`**, αλλά **μόνο `str`**, καθώς υπονοείται η πρώτη θέση μνήμης της δήλωσης `str[100]`.

ΠΑΡΑΔΕΙΓΜΑ

```
#include <stdio.h>

int main(void)
{
    char ch, word[10];

    printf("Δώσε έναν χαρακτήρα:\n");
    scanf("%c", &ch);
    printf("Ο χαρακτήρας είναι ο %c\n", ch);

    printf("Δώσε μια λέξη:\n");
    scanf("%s", word);
    printf("Η λέξη είναι η %s\n", word);

    return 0;
}
```

```
Δώσε έναν χαρακτήρα:
d
Ο χαρακτήρας είναι ο d
Δώσε μια λέξη:
monday
Η λέξη είναι η monday
```

ΠΑΡΑΔΕΙΓΜΑ

ΣΗΜΕΙΩΣΗ: Εάν έχουμε διαδοχικά scanf, ενδέχεται κάποιο από αυτά να διαβάσει χαρακτήρες που περίσσεψαν από το προηγούμενο. Για το λόγο αυτό, ανάμεσα από τα scanf αδειάζουμε τη μνήμη του πληκτρολογίου:

```
#include <stdio.h>

int main(void)
{
    char ch, word[10];

    printf("Δώσε μια λέξη:\n");
    scanf("%s", word);
    printf("Η λέξη είναι η %s\n", word);

    while(getchar() != '\n');

    printf("Δώσε έναν χαρακτήρα:\n");
    scanf("%c", &ch);
    printf("Ο χαρακτήρας είναι ο %c\n", ch);

    return 0;
}
```

← (ειδική εντολή για άδειαση μνήμης πληκτρολογίου)

Η ΣΥΝΑΡΤΗΣΗ `scanf`

Μπορούμε να διαβάσουμε και **ολόκληρο κείμενο** π.χ. μέχρι 100 χαρακτήρων:

```
char str[100];
```

```
scanf ("%[^\\n]", str);
```

Με αυτή την μορφή η `scanf` δε σταματά στα κενά ανάμεσα στις λέξεις.

ΠΑΡΑΔΕΙΓΜΑ

```
#include <stdio.h>

int main(void)
{
    char sentence[100];

    printf("Δώσε μια πρόταση:\n");
    scanf("%[^\n]", sentence);
    printf("Η πρόταση είναι η: %s\n", sentence);

    return 0;
}
```

Δώσε μια πρόταση:

This is a sentence.

Η πρόταση είναι η: This is a sentence.

Ο ΤΕΛΕΣΤΗΣ ΕΚΧΩΡΗΣΗΣ =

Με τον τελεστή εκχώρησης = αναθέτουμε μια τιμή σε μια μεταβλητή, π.χ.

`a = 10;`

ή αναθέτουμε σε μια μεταβλητή την τιμή μιας άλλης, π.χ.

`a = b;`

Η ανάθεση γίνεται από τα δεξιά προς τα αριστερά και επιτρέπεται η πολλαπλή εκχώρηση, π.χ.

`a = b = c = 10;`

ΟΙ ΤΕΛΕΣΤΕΣ +, -, *, /, %

Οι τελεστές **+**, **-**, *****, **/** εκτελούν τις συνηθισμένες τέσσερις πράξεις μεταξύ αριθμών.

ΠΡΟΣΟΧΗ: Αν έχουμε ορίσει δύο μεταβλητές ως ακέραιες, τότε η ο τελεστής **/** δίνει το **ακέραιο μέρος της διαίρεσης** μόνο, π.χ. αν

```
int i, j;  
i = 9;  
j = 2;
```

τότε η τιμή του **i/j** είναι 4, ενώ το **υπόλοιπο της διαίρεσης** προκύπτει με τον τελεστή **%** ως **i%j**.

ΠΡΟΣΟΧΗ: Ο τελεστής **%** εφαρμόζεται μόνο μεταξύ ακεραίων αριθμών.

ΟΙ ΤΕΛΕΣΤΕΣ ΑΥΞΗΣΗΣ ++ ΚΑΙ ΜΕΙΩΣΗΣ --

Ο τελεστής αύξησης **++**, αυξάνει κατά 1 την τιμή μιας μεταβλητής.

Αν χρησιμοποιείται **μετά** από το όνομα της μεταβλητής, τότε η τιμή αυξάνεται **αφού** πρώτα χρησιμοποιηθεί η μεταβλητή, π.χ.

```
#include <stdio.h>
int main()
{
    int a,b;

    a = 4;
    b = a++;
    printf("a = %d b = %d\n",a,b);
    return 0;
}
```

Έξοδος: a = 5 b = 4

ΟΙ ΤΕΛΕΣΤΕΣ ΑΥΞΗΣΗΣ ++ ΚΑΙ ΜΕΙΩΣΗΣ --

Αν χρησιμοποιείται πριν από το όνομα της μεταβλητής, τότε πρώτα αυξάνεται η τιμή και μετά χρησιμοποιείται η μεταβλητή, π.χ.

```
#include <stdio.h>
int main()
{
    int a,b;

    a = 4;
    b = ++a;
    printf("a = %d b = %d\n",a,b);
    return 0;
}
```

Έξοδος: a = 5 b = 5

Ο τελεστής μείωσης -- ελαττώνει την τιμή κατά 1 και ισχύουν οι ίδιοι κανόνες όπως για τον τελεστή ++.