Α.Τ.Ε.Ι. Θεσσαλονίκης
Τμήμα Μηχανικών Πληροφορικής Τ.Ε.

Εργαστηριακές Ασκήσεις Αριθμητικής Ανάλυσης

στη Γλώσσα Προγραμματισμού C

Γουλιάνας Κώστας
Επίκουρος Καθηγητής Α.Τ.Ε.Ι.Θ

Θεσσαλονίκη 2016

Email: gouliana@it.teithe.gr
Ιστοσελίδα : www.it.teithe.gr/~gouliana

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

Βασικό Ρεπερτόριο Εντολών της C

1

Τύποι Δεδομένων
Διάβασμα – Εμφάνιση Δεδομένων
While
Do …While
For
Functions
Απλό if
If…else

- Στο Κεφάλαιο που ακολουθεί παρουσιάζονται οι βασικές εντολές της γλώσσας προγραμματισμού C.

Πρόβλημα

- Να γίνει πρόγραμμα που να διαβάζει 2 ακέραιους αριθμούς και να υπολογίζει και εμφανίζει το **άθροισμά** τους και το **Μέσο Όρο**.

Αλγόριθμος

1) **Διαβάζουμε** 2 ακέραιους αριθμούς `a, b`.
2) **Υπολογίζουμε** το άθροισμά τους `sum` και το Μέσο Όρο `mo`.
3) **Εμφανίζουμε** τις τιμές των `sum, mo`.

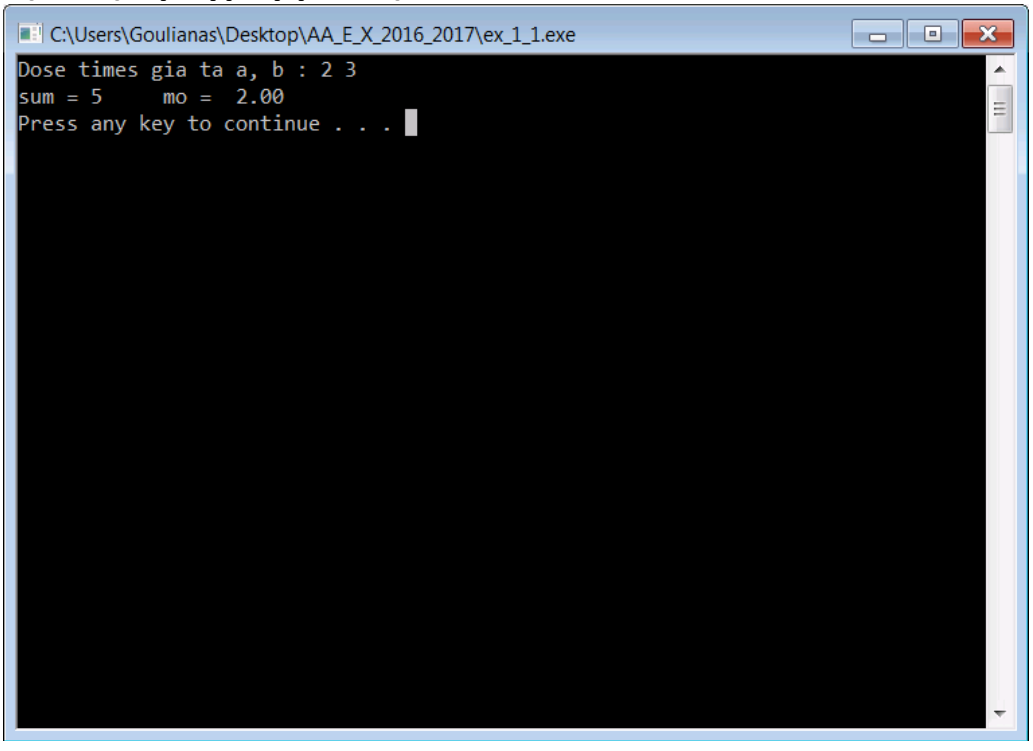
Πρόγραμμα

```
/* Άσκηση 1.1
 Υπολογισμός Αθροίσματος – Μέσου Όρου 2 Αριθμών
 */
#include <stdio.h>
#include <stdlib.h>

main()
{
    int a, b, sum;

    float mo;
    //
    // Διαβάζουμε τους 2 αριθμούς
    //
    printf("Dose times gia ta a, b : ");
    scanf ("%d  %d",&a, &b);
    sum  = a +  b; // Υπολογίζουμε το άθροισμα sum = a + b
    mo   = sum/2; // Υπολογίζουμε το Μέσο Όρο mo = sum/2
    //
    // Εμφανίζουμε τις τιμές των sum, mo
    //
    printf("sum = %d      mo = %5.2f\n", sum, mo);
    system("Pause");
}
```

Έξοδος Προγράμματος



Δομή ενός Προγράμματος C

- Ένα πρόγραμμα σε C αποτελείται απ' τα παρακάτω τμήματα :

	Τμήμα Προγράμματος
/* *Άσκηση 1.1* *Υπολογισμός Αθροίσματος – Μέσου Όρου 2 Αριθμών* */	**Σχόλιο**
#include <stdio.h> #include <stdlib.h>	**Προαιρετικές Εντολές** #include
main()	**Δήλωση Προγράμματος**
{	**Αρχή block Εντολών**
int a, b, sum; float mo;	**Δηλώσεις Μεταβλητών**

//
// *Διαβάζουμε τους 2 αριθμούς*
//
printf("Dose times gia ta a, b : ");
scanf ("%d %d",&a, &b);
sum = a + b; // *Υπολογίζουμε το άθροισμα sum = a + b*
mo = sum/2; // *Υπολογίζουμε το Μέσο Όρο mo = sum/2*
//
// *Εμφανίζουμε τις τιμές των sum, mo*
//
printf("sum = %d mo = %5.2f\n", sum, mo);
system("Pause");
	Εντολές
	Χωρισμένες με ;

}	**Τέλος block Εντολών**

Αλγόριθμος – Εντολές

Εντολές	Ενέργεια
/* Άσκηση 1.1 Υπολογισμός Αθροίσματος – Μέσου Όρου 2 Αριθμών */	**Σχόλια**
#include <stdio.h> #include <stdlib.h>	**Προαιρετικές Εντολές** #include
main()	**Δήλωση Προγράμματος**
int a, b, sum; float mo;	**Δηλώσεις Μεταβλητών**
printf("Dose times gia ta a, b : ");	**Εμφανίζουμε Μήνυμα στην Οθόνη**
scanf ("%d %d",&a, &b);	**Διαβάζουμε τους 2 αριθμούς a, b**
sum = a + b;	**Υπολογίζουμε το άθροισμα sum = a + b**
mo = sum/2;	**Υπολογίζουμε το Μέσο Όρο mo = sum/2**
printf("sum = %d mo = %5.2f\n", sum, mo);	**Εμφανίζουμε τις τιμές των sum, mo**
system("Pause");	**Παγώνουμε την εμφάνιση των αποτελεσμάτων**
}	**Τέλος Προγράμματος**

1.1 Σχόλια

❖ Τα σχόλια γράφονται σε οποιοδήποτε σημείο του προγράμματος και μπορεί να καταλαμβάνουν οσεσδήποτε γραμμές, αρκεί να υπάρχουν οι οριοθέτες **/*** και ***/**. Η σύνταξή τους γενικά είναι : /* <Σχόλια> */. Αν θέλουμε σχόλια στο τέλος μιας εντολής, γράφουμε μετά την εντολή : // <Σχόλια>.

1.2 Ενσωμάτωση Αρχείων

❖ Η εντολή

```
#include <ονομα_αρχείου_c>
```

ενσωματώνει μόνο κατά τη διάρκεια του χρόνου μεταγλώττισης τα περιεχόμενα του αρχείου <ονομα_αρχείου_c> στο σημείο που έχουμε την εντολή `#include`.

❖ Οι συναρτήσεις εισόδου και εμφάνισης δεδομένων απαιτούν την ενσωμάτωση του αρχείου που περιέχει τις αντίστοιχες συναρτήσεις βιβλιοθήκης `scanf`, `printf`, που είναι το `<stdio.h>`, το οποίο γίνεται με την εντολή :

```
#include <stdio.h>
```

❖ Σε οποιοδήποτε σημείο του Προγράμματος μπορούμε να παγώσουμε την οθόνη για να δούμε τα αποτελέσματα με την εντολή `system("Pause");`, η οποία απαιτεί την ενσωμάτωση του αρχείου `<stdlib.h>` που περιέχει την αντίστοιχη συνάρτηση βιβλιοθήκης, το οποίο γίνεται με την εντολή :

```
#include <stdlib.h>
```

1.3 Δήλωση Προγράμματος

❖ Η κύρια δομική μονάδα ενός προγράμματος στη C είναι η συνάρτηση (`function`), η οποία λειτουργεί σαν τις κλασσικές `functions` άλλων γλωσσών ή τις **μεθόδους** της `Java` (επιστρέφοντας 1 τιμή ή καμία) και σαν τα `procedures` ή `subroutines`. Η συνάρτηση που αποτελεί και το κύριο πρόγραμμα είναι η συνάρτηση `main()`. Δε χρειάζεται να επιστρέφει κάποιον τύπο ούτε ορίσματα, οπότε η γενική σύνταξη της εντολής μπορεί να είναι :

```
main()
{
σώμα...
}
```

1.4 Δηλώσεις Μεταβλητών

- Οι δηλώσεις μεταβλητών γίνονται στην αρχή κάθε συνάρτησης ή στην αρχή ενός `block` εντολών, μεταξύ `{` και `}` και γίνονται με τις εντολές :

```
int <κατάλογος_μεταβλητών>;
long int <κατάλογος_μεταβλητών>;
float <κατάλογος_μεταβλητών>;
double <κατάλογος_μεταβλητών>;
long double <κατάλογος_μεταβλητών>;
char <κατάλογος_μεταβλητών>;
```

- Η C υποστηρίζει τους παρακάτω τύπους δεδομένων :

Τύποι Δεδομένων	**Δήλωση**	Είδος Δεδομένων

Χαρακτήρες	`char`
	`signed char`
	`unsigned char`
		Η μεταβλητή αποθηκεύει ένα χαρακτήρα μεγέθους 8 bits

Ακέραιοι	`short int`
	`signed short int`
	`unsigned short int`
	`signed int`
	`unsigned int`
	`long int`
	`signed long int`
	`unsigned long int`
		Η μεταβλητή αποθηκεύει ακεραίους αριθμούς διαφόρων μεγεθών, προσημασμένους ή μη προσημασμένους.

Αριθμοί Κινητής Υποδιαστολής (floating point)	`float`
	`double`
	`long double`
		Η μεταβλητή αποθηκεύει αριθμούς κινητής υποδιαστολής με mantissa και εκθέτη διαφόρων μεγεθών απλής ή διπλής ακρίβειας.

Παρατηρήσεις

- Μια μεταβλητή μπορεί να περιέχει οποιοδήποτε κεφαλαίο ή πεζό χαρακτήρα, ένα αριθμητικό ψηφίο (0 μέχρι 9), και το χαρακτήρα κάτω παύλα (_). Ο πρώτος χαρακτήρας της μεταβλητής δεν μπορεί να είναι αριθμητικό ψηφίο ή κάτω παύλα. Τα ονόματα των μεταβλητών είναι διαφορετικά με κεφαλαία ή μικρά. Π.χ.

```
int a, b, sum;
float mo;
```

- Με τη δήλωση μιας μεταβλητής μπορούμε να της δώσουμε και αρχική τιμή. Π. χ,

```
float pi = 3.14159265358979;
```

- Όταν δίνουμε τιμή σε μια μεταβλητή τύπου **long**, στο τέλος της τιμής προσθέτουμε το χαρακτήρα **L** ή **l** . Π.χ.

```
long int count, sum = 0L;
```

- Όταν δίνουμε τιμή σε μια μεταβλητή κινητής υποδιαστολής (τύπου **float** ή **double**), μπορεί να χρησιμοποιηθεί η εκθετική μορφή, με το **E** κεφαλαίο ή μικρό. Π.χ.

```
float x, y, z = 123.45e-6;

double light_speed = 2.99792577389E8;
```

- Στον επόμενο πίνακα φαίνονται το **Μέγεθος** και η **Κλίμακα Τιμών** των διαφόρων τύπων μεταβλητών.

Τύπος	Μέγεθος	Κλίμακα Τιμών
unsigned char	8 bits	0 έως 255
char	8 bits	-128 έως 127
unsigned int	16 bits	0 έως 65,535
short int	16 bits	-32,768 έως 32,767
int	16 bits	-32,768 έως 32,767
unsigned long	32 bits	0 έως 4,294,967,295
long	32 bits	-2,147,483,648 έως 2,147,483,647
float	32 bits	1.17549435 * (10^-38) έως 3.40282347 * (10^+38)
double	64 bits	2.2250738585072014 * (10^-308) έως 1.7976931348623157 * (10^+308)
long double	80 bits	3.4 * (10^-4932) έως 1.1 * (10^4932)

1.5 Εκτέλεση Αριθμητικών Πράξεων (Αριθμητικοί Τελεστές)

- Οι τελεστές για τις αριθμητικές εκφράσεις της C είναι :

Τελεστής	***Πράξη***
+	Πρόσθεση
-	Αφαίρεση
*	Πολλαπλασιασμός
/	Ακέραια Διαίρεση
%	Υπόλοιπο Διαίρεσης (`mod`)

Παρατηρήσεις

♦ Στις διάφορες πράξεις μεταξύ των μεταβλητών πρέπει να χρησιμοποιούμε τον ίδιο τύπο μεταβλητών, διαφορετικά μπορεί να προκύψουν απρόσμενα αποτελέσματα. Οι κανόνες που ισχύουν είναι :

1. Πράξεις μεταξύ ακεραίων αριθμών παράγουν ακέραιο.

2. Πράξεις μεταξύ πραγματικών αριθμών παράγουν πραγματικό αριθμό.

3. Πράξεις μεταξύ ακεραίων και πραγματικών αριθμών παράγουν πραγματικό αριθμό.

4. Αν αποθηκεύσουμε το αποτέλεσμα μιας πράξης μεταξύ ακεραίων αριθμών σε μια πραγματική μεταβλητή, ο αριθμός γίνεται πραγματικός.

5. Αν αποθηκεύσουμε το αποτέλεσμα μιας πράξης μεταξύ πραγματικών αριθμών σε μια ακέραια μεταβλητή, αποκόπτεται το δεκαδικό μέρος.

6. Μπορούν να αποθηκευθούν ακέραιες τιμές σε μεταβλητές τύπου `char` (οπότε αποθηκεύεται ο `ASCII` κώδικας του αντίστοιχου χαρακτήρα) ή χαρακτήρες σε ακέραιες μεταβλητές. Π.χ. η δήλωση :

   ```
   char zero = 48;
   ```

   έχει σαν αποτέλεσμα να αποθηκευθεί ο χαρακτήρας '0' με ASCII κώδικα = `48` στη μεταβλητή χαρακτήρα zero.

   Η δήλωση :

   ```
   int number = '0';
   ```

έχει σαν αποτέλεσμα να αποθηκευθεί στην ακέραια μεταβλητή number ο αριθμός 48 που είναι ο ASCII κώδικας του χαρακτήρα '0'.

❖ Μπορούμε να μετατρέψουμε τα δεδομένα ενός τύπου σε δεδομένα άλλου τύπου, χωρίς να αλλάξουμε το περιεχόμενο της μεταβλητής στην οποία είναι αποθηκευμένα (διανομή - casting). Π. χ. με την εντολή

```
sum1 = (float)sum;
```

Η τιμή της ακέραιας μεταβλητής sum μετατρέπεται σε πραγματικό αριθμό και αποθηκεύεται στην πραγματική μεταβλητή sum1, χωρίς να αλλάξει το περιεχόμενο της μεταβλητής sum.

1.6 Τελεστής Αντικατάστασης

- Ο τελεστές αντικατάστασης – ανάθεσης τιμής σε μια μεταβλητή είναι το "=". Π.χ.

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

Παρατηρήσεις

- Με τη χρήση των Αριθμητικών Τελεστών μπορούμε να χρησιμοποιήσουμε τους Συντομευμένους τελεστές αντικατάστασης, όπως φαίνονται στον επόμενο πίνακα:

Τελεστής	***Έκφραση***	***Αποτέλεσμα***
+=	x+=y	Πρόσθεση του y στο x και το αποτέλεσμα στο x
-=	x-=y	Αφαίρεση του y απ' το x και το αποτέλεσμα στο x
***=**	x*=y	Πολλαπλασιασμός του y με το x και το αποτέλεσμα στο x
/=	x/=y	Διαίρεση του x με το y και το αποτέλεσμα στο x
%=	x%=y	Διαίρεση του x με το y και το Υπόλοιπο της Διαίρεσης στο x

1.7 Τελεστές Αύξησης και Μείωσης

- Οι τελεστές **++** και **--** αυξάνουν ή μειώνουν κατά 1 την τιμή της έκφρασης στην οποία ενεργούν. Π. χ. η εντολή

```
x++;
```

έχει σαν αποτέλεσμα να αυξηθεί το x κατά 1 και ισοδυναμεί με την εντολή

```
x = x + 1;
```

ενώ η εντολή

```
x--;
```

έχει σαν αποτέλεσμα να μειωθεί το x κατά 1 και ισοδυναμεί με την εντολή

```
x = x - 1;
```

Παρατηρήσεις

- Οι τελεστές **++** και **--** μπορούν να είναι **πριν** ή **μετά** την έκφραση στην οποία ενεργούν.

- Αν οι τελεστές **++** και **--** είναι **πριν** την έκφραση στην οποία ενεργούν, **πρώτα** αυξάνεται ή μειώνεται η τιμή της έκφρασης και **μετά** χρησιμοποιείται η νέα τιμή. Π. χ. στις εντολές

```
a = 0;
b = ++a;
```

πρώτα αυξάνεται το a κατά 1 (οπότε γίνεται 1) και το b παίρνει την ίδια τιμή 1.

- Αν οι τελεστές **++** και **--** είναι **μετά** την έκφραση στην οποία ενεργούν, **πρώτα** χρησιμοποιείται η αρχική τιμή της έκφρασης και **μετά** αυξάνεται ή μειώνεται η τιμή της έκφρασης κατά `1`. Π. χ. στις εντολές

```
a = 0;
b = a++;
```

πρώτα το `b` παίρνει την τιμή του `a = 0`, και **μετά** το `a` αυξάνεται κατά `1` (οπότε γίνεται `1`).

1.8 Λογικοί Τελεστές (Boolean)

Οι λογικοί τελεστές ενεργούν πάνω σε αριθμητικά δεδομένα θεωρώντας κάθε **μη μηδενική** τιμή σαν αλήθεια (**`true`**) και επιστρέφουν την τιμή `1`, ενώ την τιμή `0` σαν ψευδή (**`false`**) και επιστρέφουν την τιμή 0. Οι λογικοί τελεστές είναι :

Τελεστής	*Έκφραση*	*Αποτέλεσμα*
&&	<***έκφραση1***> **&&** <***έκφραση2***>	Λογικό ΚΑΙ στην ***<έκφραση1>*** και ***<έκφραση2>***. Επιστρέφει 1, αν και οι 2 εκφράσεις είναι αληθείς, διαφορετικά 0.
||	<***έκφραση1***> **||** <***έκφραση2***>	Λογικό Ή στην ***<έκφραση1>*** και ***<έκφραση2>***. Επιστρέφει 0, αν και οι 2 εκφράσεις είναι ψευδείς, διαφορετικά 1.

Παρατηρήσεις

Κατά την εκτέλεση των λογικών πράξεων που εκτελούνται από αριστερά προς τα δεξιά, δεν υπολογίζονται οι τιμές όλων των παραστάσεων, αλλά μόνο όσων είναι απαραίτητες. Π.χ. στην έκφραση

```
A && B
```

αν η παράσταση `A` είναι ψευδής, δεν υπολογίζεται η παράσταση `B`, ενώ στην έκφραση

```
A || B
```

αν η παράσταση `A` είναι αληθής, δεν υπολογίζεται η παράσταση `B`.

1.8.1 Προτεραιότητα Τελεστών

Όταν υπάρχουν παρενθέσεις σε μια έκφραση, υπολογίζονται πρώτα αυτές. Η προτεραιότητα των τελεστών κατά αύξουσα τάξη φαίνεται στον επόμενο πίνακα :

Τελεστής	Είδος
`!`	Λογικό ΟΧΙ
`++ --`	Τελεστές αύξησης - μείωσης
`* / %`	Τελεστές πολλαπλασιασμού - διαίρεσης
`+ -`	Τελεστές πρόσθεσης - αφαίρεσης.
`< > <= >=`	Τελεστές ανισότητας
`== !=`	Τελεστές ισότητας
`&&`	Λογικό ΚΑΙ
`||`	Λογικό Ή
`?:`	Conditional.
`= += -=`	Καταχώρηση.

1.9 Είσοδος Δεδομένων - Εμφάνιση Αποτελεσμάτων

Η είσοδος δεδομένων απ' το πληκτρολόγιο και η εμφάνιση των αποτελεσμάτων στην οθόνη μπορεί να γίνει με τις συναρτήσεις – εντολές **`scanf`** (scan formatted) και **`printf`** (print formatted) .

1.9.1 Εμφάνιση Αποτελεσμάτων με Προκαθορισμένη Μορφή

- Η γενική σύνταξη της εντολής `printf` είναι :

  `printf` (“μηνύματα - προσδιοριστές”, <όρισμα-1>, <όρισμα-2>,..., <όρισμα-ν>)’;

❖ Στα διπλά εισαγωγικά περιέχονται **μηνύματα** που θέλουμε να εμφανισθούν, καθώς και **ειδικές οδηγίες** για τη μορφή που θα έχουν τα περιεχόμενα των μεταβλητών τις τιμές των οποίων θα διαβάσουμε ή θα εμφανίσουμε. Οι οδηγίες αυτές ξεκινούν με το χαρακτήρα ‘%’, ο οποίος ακολουθείται από κάποιους **προσδιοριστές**, οι οποίοι μπορεί να είναι :

Προσδιοριστής	**Είδος**
`%d`	Εμφανίζει την τιμή της **ακέραιας** μεταβλητής στο δεκαδικό σύστημα
`%f`	Εμφανίζει την τιμή της μεταβλητής **`float`** στη μορφή ddd.dddddd
`%lf`	Εμφανίζει την τιμή της μεταβλητής **`double`** στη μορφή ddd.dddddd
`%e`	Εμφανίζει την τιμή της μεταβλητής **`float`** ή **`double`** σε **εκθετική** μορφή 1.ddddddE+ddd

Παράδειγμα

Με τις παρακάτω εντολές για την εμφάνιση ενός ακεραίου (μεταβλητή `i`) και του πραγματικού αριθμού `0.1234567` σαν πραγματικό αριθμό απλής και διπλής ακριβείας (μεταβλητές `sp` και `dp`) και σε εκθετική μορφή (μεταβλητή `sp`) :

```
  int i = 5;
  float sp = 0.1234567;
  double dp = 0.1234567;
  printf("i = %d, sp = %f, dp = %lf, fp = %e \n", i, sp, dp,
sp);
```

θα εμφανιστεί :

```
i = 5, sp = 0.123457, dp = 0.123457, fp = 1.234567e-001
```

Παρατηρήσεις

- Τα δεδομένα εκτυπώνονται σε προδιαγεγραμμένη μορφή.
- Οι ακέραιοι αριθμοί καταλαμβάνουν τόσες θέσεις, όσες χρειάζονται.
- Οι πραγματικοί αριθμοί εκτυπώνονται σε δεκαδική ή σε εκθετική μορφή.
- Οι αριθμοί απλής και διπλής ακρίβειας καταλαμβάνουν όσες θέσεις χρειάζονται, με 6 δεκαδικά ψηφία μετά την υποδιαστολή. Αν ο αποθηκευμένος αριθμός έχει περισσότερα από 6 δεκαδικά ψηφία, γίνεται στρογγυλοποίηση σε 6 δεκαδικά ψηφία (π.χ. ο αριθμός 0.1234567 με 7 δεκαδικά ψηφία εμφανίζεται σαν 0.123457). Αν ο αποθηκευμένος αριθμός δεν χρειάζεται και τις 6 θέσεις οι επί πλέον δεξιές θέσεις συμπληρώνονται με μηδενικά.
- Οι αριθμοί απλής και διπλής ακρίβειας όταν εμφανίζονται σε εκθετική μορφή (κωδικός %e) εμφανίζονται στη μορφή 1.dddddde±<εκθέτης>, όπου τα δεκαδικά ψηφία της mantissa καταλαμβάνουν 6 θέσεις και ο εκθέτης παίρνει τιμές μέχρι το 999.

1.9.2 Εμφάνιση Αποτελεσμάτων με τη Μορφή που Επιθυμεί ο Προγραμματιστής

- Αν θέλουμε να εμφανιστούν τα δεδομένα με τη μορφή που θέλουμε, στον κωδικό της εμφάνισης, μεταξύ του συμβόλου % και του προσδιοριστή, μπορούμε να βάλουμε έναν ή δύο αριθμούς, με τους οποίους καθορίζουμε το μέγιστο και τον αριθμό των δεκαδικών ψηφίων που θέλουμε να καταλαμβάνει ο αριθμός. Έτσι, αν χρησιμοποιήσουμε τις εντολές :

```
int i = 5;
float sp = 0.1234567;
double dp = 0.1234567;
printf("i = %5d, sp = %16.12f, dp = %16.12lf\n",i, sp, dp);
```

θα εμφανιστεί :

```
i = ^^^^5, sp = ^^0.123456701636, dp = ^^0.123456700000
```

Παρατηρήσεις

- Τα δεδομένα εκτυπώνονται με τη μορφή που επιθυμεί ο χρήστης.
- Οι ακέραιοι αριθμοί καταλαμβάνουν τόσες θέσεις, όσες και ο αριθμός που υπάρχει μεταξύ % και d. Αν δεν χρειάζονται όλες οι θέσεις, ο αριθμός εμφανίζεται στοιχημένος δεξιά, ενώ οι επί πλέον αριστερές θέσεις συμπληρώνονται με κενά (^^^^).

- Οι αριθμοί απλής και διπλής ακρίβειας καταλαμβάνουν συνολικά με την υποδιαστολή τόσες θέσεις όσο και ο πρώτος αριθμός (`16` στο παράδειγμα), με τόσα δεκαδικά ψηφία μετά την υποδιαστολή όσο και ο δεύτερος αριθμός (`12` στο παράδειγμα). Αν ο αποθηκευμένος αριθμός έχει περισσότερα δεκαδικά ψηφία, γίνεται στρογγυλοποίηση. Αν ο αποθηκευμένος αριθμός δεν χρειάζεται τόσα δεκαδικά ψηφία, οι επί πλέον δεξιές θέσεις συμπληρώνονται με μηδενικά, ενώ οι επί πλέον αριστερές θέσεις συμπληρώνονται με κενά (π.χ. `^^0.123456700000` για τη μεταβλητή διπλής ακρίβειας, ενώ η μεταβλητή απλής ακρίβειας εμφανίζει το `^^0.123456701636`, επειδή ο αριθμός `0.1234567` αποθηκεύεται με σφάλμα).

- Αν θέλουμε να εμφανιστούν τα δεδομένα με αριστερή στοίχιση, στον κωδικό της εμφάνισης, πριν τον ή τους δύο αριθμούς, βάζουμε το '–'. Έτσι, αν χρησιμοποιήσουμε τις εντολές :

```
int i = 5;
float sp = 0.1234567;
double dp = 0.1234567;
printf("i = %-5d, sp = %-16.12f, dp = %-16.12lf\n", i, sp, dp);
```

θα εμφανιστεί :

```
i = 5^^^^, sp = 0.123456701636^^, dp = 0.123456700000^^
```

1.9.3 Χαρακτήρες Ελέγχου

- Στην εντολή printf αν υπάρχει το `/n` πριν κλείσουν τα διπλά εισαγωγικά, η επόμενη εκτύπωση θα γίνει στην επόμενη γραμμή. Αυτός ο χαρακτήρας δεν εκτυπώνεται, αλλά χρησιμοποιείται σαν "οδηγός" για το πού θα γίνει η επόμενη εκτύπωση και είναι σαν χαρακτήρας ελέγχου. Οι χαρακτήρες ελέγχου είναι οι παρακάτω :

Χαρακτήρας	Αποτέλεσμα
κανένας	Εκτύπωση στην ίδια γραμμή.
\n	Εκτύπωση στην επόμενη γραμμή.
\t	Εκτύπωση στην ίδια γραμμή μετά ένα tab.

1.9.4 Είσοδος Δεδομένων με Προκαθορισμένη Μορφή

- Η γενική σύνταξη της εντολής `scanf` είναι :

```
scanf ("προσδιοριστές", &<όρισμα-1>, &<όρισμα-2>,…, &<όρισμα-ν>);
```

❖ Στα διπλά εισαγωγικά περιέχονται **μηνύματα** που θέλουμε να εμφανισθούν, καθώς και **ειδικές οδηγίες** για τη μορφή που θα έχουν τα περιεχόμενα των μεταβλητών τις τιμές των οποίων θα διαβάσουμε ή θα εμφανίσουμε. Οι οδηγίες αυτές ξεκινούν με το χαρακτήρα '%', ο οποίος ακολουθείται από κάποιους **προσδιοριστές**, όπως στην εντολή `printf`.

❖ Πριν από κάθε όρισμα υπάρχει το σύμβολο **&** που σημαίνει ότι περνάμε τη **διεύθυνση** της **μεταβλητής** στην οποία θα αποθηκευθούν τα δεδομένα που διαβάζονται απ' το πληκτρολόγιο.

❖ Στην εκτέλεση του προγράμματος θα πρέπει να δώσουμε τόσες τιμές, όσα και τα ορίσματα στην εντολή `scanf`, χωρισμένες με κενό ή `Enter`.

❖ Αν θέλουμε να διαβαστούν τα δεδομένα με τη μορφή που θέλουμε, στον κωδικό της εισαγωγής δεδομένων, μεταξύ του συμβόλου `%` και του προσδιοριστή, μπορούμε να βάλουμε έναν ή δύο αριθμούς, με τους οποίους καθορίζουμε το μέγιστο και τον αριθμό των δεκαδικών ψηφίων που θέλουμε να καταλαμβάνει ο αριθμός που θα εισάγουμε.

1.10 Τελεστές Σύγκρισης

- Η C διαθέτει τους παρακάτω Τελεστές Σύγκρισης :

Τελεστής	Σύμβολο Πληκτρολογίου
≤	<=
<	<
=	==
≠	|=
≥	>=
>	>

1.11 Η Εντολή `if`

- Με την εντολή `if` ελέγχεται κάποια συνθήκη και αν ισχύει, εκτελείται μια εντολή ή εντολές. Στο παράδειγμα χρησιμοποιείται για τον τερματισμό των επαναλήψεων. Η γενική της σύνταξη είναι :

```
if (<συνθήκη>) <εντολή>;
```

```
if (<συνθήκη>)
{
  <εντολές>;
}
```

1.11.1 `if...else`

- Με τη χρήση του `if…else`, μπορούμε να εκτελέσουμε ένα `block` εντολών, ανάλογα με το αν ισχύει κάποια συνθήκη ή όχι. Η σύνταξή του είναι :

```
if (<συνθήκη>) <εντολή>; ή { <εντολές>; }
else
  <εντολή>; ή { <εντολές>; }
```

1.12 Τερματισμός Προγράμματος (`return`)

- Το πρόγραμμα τερματίζει όταν τελειώσουν οι εντολές με το τελευταίο "`}`", όταν η κύρια συνάρτηση `main()` δηλώνεται χωρίς τύπο ή με τύπο void ή με την εντολή **`return`**. Η εντολή **`return`** μπορεί να εμφανίζεται σε πολλά σημεία του προγράμματος.

1.13 Η εντολή `while`

- Με το while εκτελούνται κάποιες εντολές **για όσο** μια συνθήκη είναι αληθής. Η σύνταξη της εντολής είναι :

```
while ( <συνθήκη> )
    <εντολή>;
```

```
while ( <συνθήκη> )
        {
          εντολές…
        }
```

Παρατηρήσεις

- Ο έλεγχος της συνθήκης γίνεται **πριν** την εκτέλεση της εντολής ή των εντολών.
- Αν η συνθήκη είναι εξ αρχής **ψευδής**, οι εντολές δεν εκτελούνται **καμιά** φορά.
- Οι εντολές εκτελούνται **για όσο** η συνθήκη είναι **αληθής** και **παύουν** να εκτελούνται, όταν η συνθήκη γίνει **ψευδής**.
- Η **μεταβλητή** που ελέγχεται στη συνθήκη θα πρέπει να **έχει** τιμή ***πριν το*** **`while`**, για να εκτελεστούν οι εντολές τουλάχιστον μια φορά και να **αλλάζει** τιμή ***μέσα στο*** **`while`**, για να τερματίσει ο βρόχος.
- Ο έλεγχος της συνθήκης μπορεί να γίνεται **μετά** την εκτέλεση των εντολών του while, οπότε οι εντολές εκτελούνται **τουλάχιστον** μια φορά. Σ' αυτή την περίπτωση, η σύνταξη της εντολής `while` που αποτελεί μια μορφή της εντολής **`repeat…until`** θα είναι :

```
do
  <εντολή>
while ( <συνθήκη> );
```

```
do
  {
    εντολές…
  }
while ( <συνθήκη> );
```

1.14 Ανακύκλωση με την εντολή `for`

- Όταν ξέρουμε τον αριθμό των επαναλήψεων, χρησιμοποιούμε την εντολή `for`, η οποία μπορεί να εμφανιστεί, στη γενική της μορφή, με τον παρακάτω τύπο :

```
for ( < έκφραση1>;< έκφραση2>; < έκφραση3> )
    εντολή;

for ( < έκφραση1>;< έκφραση2>; < έκφραση3> )
    {
     εντολές;
     }
```

Παρατηρήσεις

- Η **< έκφραση1>** υπολογίζεται πριν την πρώτη επανάληψη.
- Μετά από κάθε επανάληψη υπολογίζεται η **< έκφραση3>**.
- Οι εντολές εκτελούνται για όσο η **<έκφραση2>** είναι αληθής (έχει τιμή `1`) και μέχρι να γίνει ψευδής (τιμή `0`).
- Ο έλεγχος για το αν ισχύει η **<έκφραση2>** γίνεται πριν την εκτέλεση των εντολών.
- Μπορούν να χρησιμοποιηθούν οι συντομευμένοι τελεστές. Π.χ.

```
for (x = a; x <= b; x +=h)
```

- Μπορούν να χρησιμοποιηθούν οι τελεστές αύξησης και μείωσης. Π.χ. η εντολή

```
for (x = 1; x <= 10; x ++)
```

ισοδυναμεί με την εντολή

```
for (x = 1; x <= 10; x = x + 1)
```

- Η **<έκφραση1>**, η **<έκφραση2>** και η **<έκφραση3>** μπορούν να παραλειφθούν, αρκεί να υπάρχουν τα ";". Π.χ. η εντολή

```
for (;;)
```

αποτελεί έναν ατέρμονο βρόχο.

1.15 Εσωτερικές Συναρτήσεις της C

- Η `C` δε διαθέτει τον τελεστή ύψωσης σε δύναμη, διαθέτει όμως το δικό της ρεπερτόριο εσωτερικών συναρτήσεων (`intrinsic functions`), για τον υπολογισμό των στοιχειωδών Μαθηματικών και Τριγωνομετρικών Συναρτήσεων, μία εκ των οποίων υπολογίζει την ύψωση σε δύναμη. Οι συναρτήσεις αυτές δηλώνονται στην αρχή του προγράμματος ή του `block` εντολών. Π.χ. η συνάρτηση ύψωσης σε δύναμη θα δηλωθεί σαν

```
float pow(float x, int y);
```

και επιστρέφει την ύψωση του `x` στην δύναμη `y` (x^y), με $x \geq 0$, αν `y` = κλάσμα και $x \neq 0$, αν $y \leq 0$. Στο προηγούμενο παράδειγμα, το **x** έχει δηλωθεί πραγματικό (`float`) ενώ το **y** ακέραιο (`int`).

- Ο Επόμενος πίνακας παρουσιάζει τις πιο συνηθισμένες εσωτερικές συναρτήσεις με το όνομά τους, τη χρήση τους και το αποτέλεσμα που επιστρέφουν :

Συνάρτηση	**Δήλωση**	**Αποτέλεσμα**
acos	**double acos(double *x*);**	Επιστρέφει το το Τόξο Συνημιτόνου του x μεταξύ *0* και π ($-1 \le x \le 1$)
asin	**double asin(double *x*);**	Επιστρέφει το το Τόξο Ημιτόνου του x μεταξύ $-\frac{\pi}{2}$ και $\frac{\pi}{2}$ ($-1 \le x \le 1$)
atan	**double atan(double *x*);**	Επιστρέφει το το Τόξο Εφαπτομένης του x μεταξύ $-\frac{\pi}{2}$ και $\frac{\pi}{2}$
cos	**double cos(double *x*);**	Επιστρέφει το Συνημίτονο του x
cosh	**double cosh(double *x*);**	Επιστρέφει το Υπερβολικό Συνημίτονο του x
sin	**double sin(double *x*);**	Επιστρέφει το Ημίτονο του x
sinh	**double sinh(double *x*);**	Επιστρέφει το Υπερβολικό Ημίτονο του x
tan	**double tan(double *x*);**	Επιστρέφει το Τόξο Εφαπτομένης του x
tanh	**double tanh(double *x*);**	Επιστρέφει την Υπερβολική Εφαπτομένη του x
exp	**double exp(double *x*);**	Επιστρέφει την Εκθετική Συνάρτηση (e^x)
frexp	**double frexp(double *x*, int **exponent*);**	Αναλύει το x σε mantissa και εκθέτη, επιστρέφοντας στη συνάρτηση τη mantissa και στην ακέραια μεταβλητή *exponent* τον εκθέτη ($x = mantissa * 2^{exponent}$)
ldexp	**double ldexp(double *x*, int *exponent*);**	Επιστρέφει το γινόμενο του x * το 2 στην δύναμη *exponent* ($x * 2^{exponent}$)
log	**double log(double *x*);**	Επιστρέφει το Φυσικό Λογάριθμο του x ($\log_e$ x).
log10	**double log10(double *x*);**	Επιστρέφει το Δεκαδικό Λογάριθμο του x ($\log_{10}$ x).
modf	**double modf(double *x*, double **integer*);**	Επιστρέφει το δεκαδικό μέρος ενός δεκαδικού αριθμού, ενώ το ακέραιο μέρος επιστρέφει στη μεταβλητή *integer*
pow	**double pow(double *x*, double *y*);**	Επιστρέφει την ύψωση του x στην δύναμη y (x^y), με $x \ge 0$, αν y = κλάσμα και $x \ne 0$, αν $y \le 0$
sqrt	**double sqrt(double *x*);**	Επιστρέφει την Τετραγωνική Ρίζα.του x ($\sqrt{x}$), με $x \ne 0$
ceil	**double ceil(double *x*);**	Επιστρέφει το μικρότερο ακέραιο ≥ του x
abs	**int abs(int *x*);**	Επιστρέφει την Απόλυτη Τιμή του ακεραίου x ($\lvert x \rvert$)
fabs	**double fabs(double *x*);**	Επιστρέφει την Απόλυτη Τιμή του πραγματικού x
floor	**double floor(double *x*);**	Επιστρέφει το μεγαλύτερο ακέραιο ≤ του x
fmod	**double fmod (double *x*, double *y*);**	Επιστρέφει το υπόλοιπο της διαίρεσης του x με το y. Αν y = 0, επιστρέφει 0 ή μήνυμα λάθους

1.16 Ορισμός macro-Εντολών

- Η συνάρτηση $y = f(x) = 1 + x^2$ μπορεί να δηλωθεί στην αρχή του προγράμματος με τη δήλωση **#define**. Η δήλωσή της θα έχει τη μορφή :

```
#define f(x) (1+x*x)
```

Ο προεπεξεργαστής της C, όπου βρει μια έκφραση που περιέχει το f(x), θα αντικαταστήσει την τιμή του x στην έκφραση. Οι δηλώσεις macro-Εντολών έχουν την παρακάτω γενική μορφή :

#define <Όνομα_function>(Τυπικές_Παράμετροι) (<Αριθμητική_Έκφραση>)

όπου :

<Όνομα_function> = Μεταβλητή (Τύπου integer ή real).

(Τυπικές_Παράμετροι) = Μεταβλητές.

<Αριθμητική_Έκφραση> = Αριθμητική Έκφραση ως προς τις παραμέτρους.

Παρατηρήσεις

- Η Συνάρτηση - Εντολή πρέπει να τοποθετείται στην αρχή του προγράμματος.
- Μπορεί να υπάρχει οποιοσδήποτε αριθμός Τυπικών Παραμέτρων.
- Οι παρενθέσεις στον ορισμό της Συνάρτησης - Εντολής είναι υποχρεωτικές.
- Το όνομα της συνάρτησης πρέπει να ακολουθεί τους κανόνες των μεταβλητών.
- Οι Πραγματικές Παράμετροι μπορεί να είναι σταθερές ή μεταβλητές.
- Η συνάρτηση σε κάθε κλήση της δίνει **μια** τιμή.
- Ο αριθμός, ο τύπος και η σειρά των Πραγματικών Παραμέτρων στη χρήση της συνάρτησης στο κυρίως πρόγραμμα πρέπει να συμφωνεί με τον αριθμό, τον τύπο και τη σειρά των Τυπικών Παραμέτρων στον Ορισμό της συνάρτησης.
- Τα ονόματα των Πραγματικών Παραμέτρων μπορεί να είναι ή να μην είναι ίδια με τα ονόματα των Τυπικών Παραμέτρων.