Προγραμματισμός Υπολογιστών

Τα στοιχεία ενός προγράμματος.

Εκτύπωση μηνυμάτων και ανάγνωση τιμών

Νικόλαος Ζ. Ζάχαρης
Καθηγητής

Πανεπιστήμιο Δυτικής Αττικής
Τμήμα Μηχανικών Πληροφορικής και Υπολογιστών

Το *Αλφάβητο και το Λεξιλόγιο της γλώσσας C*

Ένα C πρόγραμμα μπορεί να αποτελείται από τους παρακάτω χαρακτήρες:

Μικρά	a, b, c, …, z
Κεφαλαία	A, B, C, …, Z
Ψηφία	0, 1, 2, …, 9
Ειδικούς	+ = _ - ( ) * & % $ # ! | < > . , ; : / ? { } ~ \ [ ] ^

καθώς και οι χαρακτήρες κενό και νέα γραμμή που είναι γνωστά ως white space.

Όλοι οι ανωτέρω χαρακτήρες ομαδοποιούνται από τον compiler σε λέξεις, οι οποίες διακρίνονται σε:

- Σχόλια (comments)
- Λέξεις κλειδιά (reserved words)
- Μεταβλητές
  - Τύποι Δεδομένων
  - Αναγνωριστικά Ονόματα (identifiers)
- Σταθερές (constants)
- Συμβολοσειρές (strings)
- Τελεστές (operators) και Διαχωριστές (separators)

1. Σχόλια – Σημειώσεις (comments)

Όλες οι γλώσσες προγραμματισμού δίνουν την δυνατότητα στον προγραμματιστή να συμπεριλάβει σχόλια στο πηγαίο κώδικα. Τα σχόλια χρησιμοποιούνται για την κατανόηση και την συντήρηση του προγράμματος. Τα σχόλια αγνοούνται από το μεταγλωττιστή και δεν περιλαμβάνονται στο εκτελέσιμο πρόγραμμα. Ένα σχόλιο στην γλώσσα C++ είναι οτιδήποτε περιλαμβάνεται ανάμεσα στους χαρακτήρες /* και */ ή οτιδήποτε μετά τους χαρακτήρες // μέχρι το τέλος της γραμμής.

```
// Ένα σχόλιο σε μία γραμμή

/**** Ένα άλλο σχόλιο ***********/

/*

   Ένα ακόμα σχόλιο

*/

/**************************

 *   Ένα ακόμα σχόλιο   *

**************************/
```

Είναι καλή και χρήσιμη συνήθεια να γράφεται σχόλια μαζί με το πηγαίο κώδικα για να εξηγείται την δομή ή την λογική του προγράμματος

2. Λέξεις Κλειδιά (reserved words)

Όλες οι γλώσσες προγραμματισμού ορίζουν ένα σύνολο από λέξεις οι οποίες ονομάζονται δεσμευμένες λέξεις ή λέξεις κλειδιά και έχουν ειδική σημασία. Η σημασία αυτών των λέξεων δεν μπορεί να τροποποιηθεί από τον προγραμματιστή. Μερικές από αυτές τις λέξεις φαίνονται παρακάτω:

auto	break	case	char
const	continue	default	do
int	long	register	return
short	signed	sizeof	static
struct	switch	typedef	union
unsigned	void	volatile	while
double	else	enum	extern
float	for	goto	if

3. *Μεταβλητές*

Οι μεταβλητές είναι «κουτιά τιμών» που ορίζονται από το προγραμματιστή για να αποθηκεύσει πληροφορίες σχετικές με το πρόγραμμα. Για να δημιουργήσουμε μια μεταβλητή θα πρέπει πρώτα να δηλωθεί ως εξής :

Τύπος_Δεδομένων Όνομα_Μεταβλητής;

Η δήλωση (δημιουργία) μιας μεταβλητής δεσμεύει χώρο στη μνήμη του υπολογιστή και μπορούμε να δημιουργήσουμε όσα «κουτιά τιμών» θέλουμε στα πλαίσια ενός προγράμματος αρκεί να υπάρχει διαθέσιμη μνήμη. Τα δύο στοιχεία που χαρακτηρίζουν μια μεταβλητή είναι :

ο **Τύπος_Δεδομένων** δηλαδή το σχήμα καθώς και το πόσο μεγάλο θα είναι το «κουτί» τιμών. Με τη λέξη σχήμα εννοούμε τι πρόκειται να βάλουμε μέσα στο κουτί τιμών, για παράδειγμα έναν ακέραιο αριθμό ή ένα πραγματικό αριθμό ή ένα χαρακτήρα, και με το μέγεθος εννοούμε το εύρος των τιμών που θα αποθηκεύσουμε. (περισσότερα θα δούμε στους τύπους δεδομένων)

και το **Όνομα_Μεταβλητής** δηλαδή πως θα αναφερόμαστε στο συγκεκριμένο «κουτί τιμών».

3.1 Τύποι δεδομένων

Ένας *τύπος δεδομένων* είναι ένα σύνολο τιμών και ένα σύνολο λειτουργιών (πράξεων) που μπορούν να εφαρμοστούν σε αυτές τις τιμές.

Απλοί τύποι – οι τιμές δεν μπορούν να διασπασθούν σε απλούστερα στοιχεία.

Σύνθετοι τύποι – απαρτίζονται από απλούς τύπους και συνήθως τους δημιουργεί ο προγραμματιστής. Θα τους δούμε σε επόμενες διαλέξεις.

Απλοί Τύποι	***Λέξη Κλειδί***	***Bytes***	***Τιμές***
Χαρακτήρας	**char**	1	-128 έως 127
Ακέραιος	**int**	4	-2147483648 έως 2147483647
Μεγάλος Ακέραιος	**long**	4	-2147483648 έως 2147483647

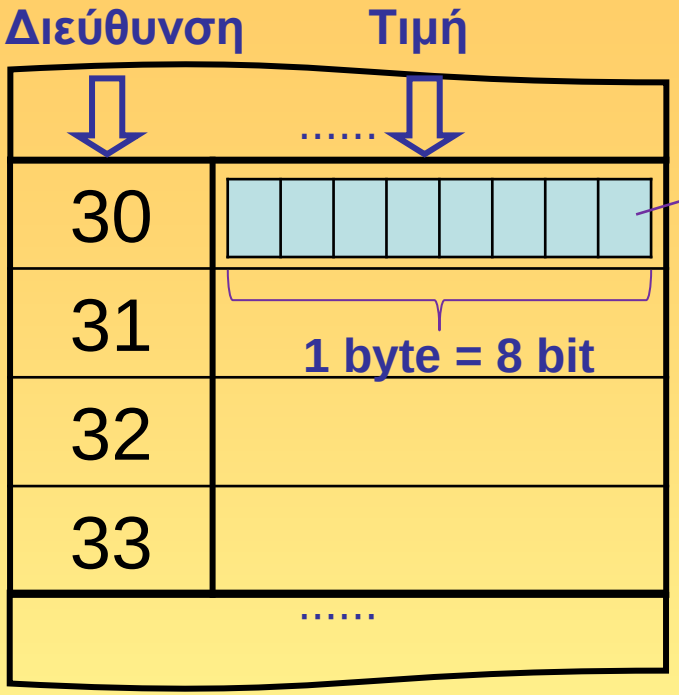
Όλοι οι παραπάνω τύποι μπορούν να συνταχθούν και με την δεσμευμένη λέξη **unsigned** που σημαίνει χωρίς πρόσημο και σε αυτή τη περίπτωση χρησιμοποιείται το εύρος των αρνητικών αριθμών για να αυξήσει το εύρος των θετικών αριθμών.

Έτσι λοιπόν με την έκφραση **unsigned char** περιγράφουμε το σύνολο των ακεραίων αριθμών από **0 έως 255** ενώ με την έκφραση **unsigned int** περιγράφουμε το σύνολο των ακεραίων αριθμών από **0 έως 4,294,967,295**. Υπάρχει και ο short τύπος όπου ο **short int** καταλαμβάνει 2 bytes και με εύρος τιμών από -32,768 έως 32,767.

Τύπος Δεδομένων	Μνήμη (bytes)	Εύρος Τιμών	Χαρακτηριστικό Μορφοποίησης
signed char	1	-128 έως 127	%c
unsigned char	1	0 έως 255	%c
short int	2	-32,768 έως 32,767	%hd
unsigned short int	2	0 έως 65,535	%hu
unsigned int	4	0 έως 4,294,967,295	%u
int	4	-2,147,483,648 έως 2,147,483,647	%d
long int	4	-2,147,483,648 έως 2,147,483,647	%ld
unsigned long int	4	0 έως 4,294,967,295	%lu
long long int	8	-9223372036854775808 έως 9223372036854775808	%lld
unsigned long long int	8	0 έως 18,446,744,073,709,551,615	%llu
float	4	$1.2*10^{-38}$ έως $3.4*10^{38}$	%f
double	8	$2.2*10^{-308}$ έως $1.8*10^{308}$	%lf
long double	12	$3.4 * 10^{-4932}$ έως $1.1 * 10^{+4932}$	%Lf

Σε συστήματα 32 bit ο int έχει το ίδιο εύρος με το long ενώ σε 64 bit ο long έχει εύρος long long int.

Τύποι δεδομένων (συνέχεια)



1 bit που μπορεί να πάρει σαν τιμή 0 ή 1.

Η μνήμη του υπολογιστή αποτελείται από συνεχόμενες θέσεις που ονομάζονται bytes. Όλες οι θέσεις της μνήμης είναι αριθμημένες σαν τις διευθύνσεις ενός δρόμου όπου τα σπίτια είναι οι χώροι αποθήκευσης των τιμών των μεταβλητών.

Το λειτουργικό σύστημα του υπολογιστή είναι υπεύθυνο για την διαχείριση της μνήμης για αυτό το λόγο χρησιμοποιεί ένα πίνακα στον οποίο καταγράφει ποιές διευθύνσεις μνήμης που χρησιμοποιούνται και από ποια προγράμματα καθώς και το μέγεθος που καταλαμβάνουν στη μνήμη.

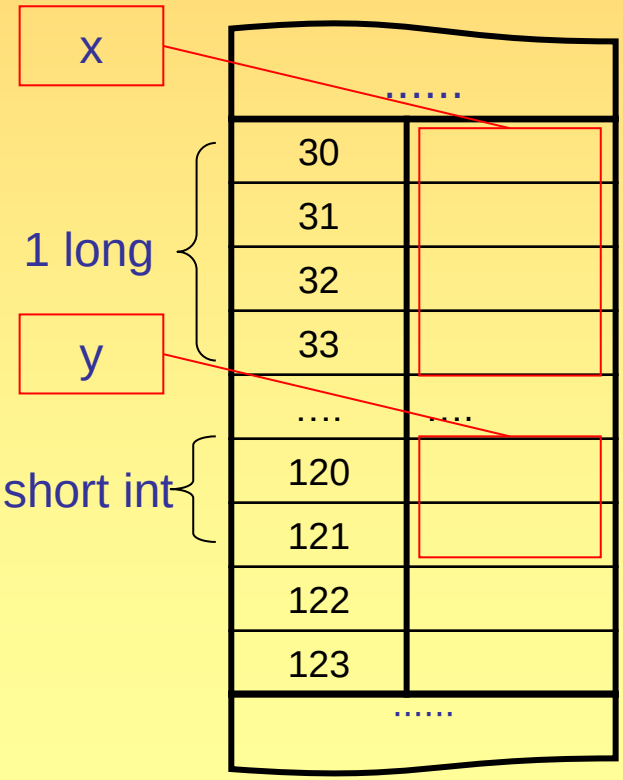
Η δήλωση σε ένα πρόγραμμα :

long x;

short int y;

θα δεσμεύσει 4 συνεχόμενα bytes για τη μεταβλητή x και δυο συνεχόμενα bytes για τη μεταβλητή y.

ΜΝΗΜΗ ΥΠΟΛΟΓΙΣΤΗ



ΛΕΙΤΟΥΡΓΙΚΟ ΣΥΣΤΗΜΑ

Όνομα μεταβλητής	Διεύθυνση	Μέγεθος
x	30	4
y	120	2

Όταν δηλώνουμε μια μεταβλητή δεσμεύονται τόσα συνεχόμενα bytes όσα περιγράφονται από το τύπο δεδομένων της μεταβλητής. Στη συγκεκριμένη χρονική στιγμή το λειτουργικό σύστημα βρήκε 4 bytes ελεύθερα στη διεύθυνση 30 και 2 bytes ελεύθερα στη διεύθυνση 120.

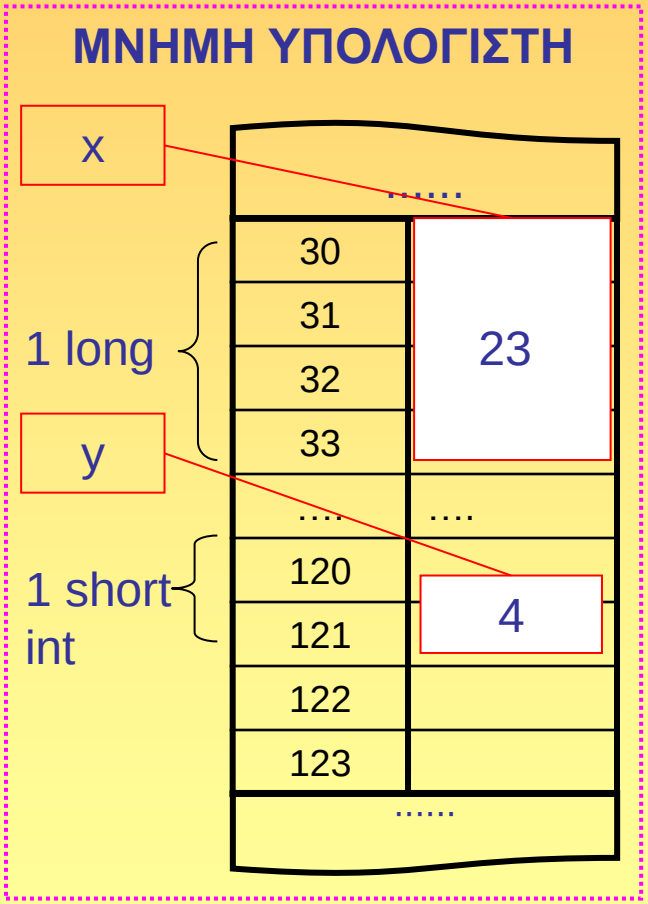
Νικόλαος Ζ. Ζάχαρης

Οι επόμενες εντολές στο ίδιο πρόγραμμα :

x = 23;

y = 4;

θα αποθηκεύσουν τις αντίστοιχες τιμές στα κουτιά τιμών των μεταβλητών.



Αν στη μεταβλητή x πρόκειται να αποθηκεύσουμε την ηλικία ενός ανθρώπου ή οποία σαν τιμή μπορεί να αποθηκευτεί και σε ένα short int – η οποία καταλαμβάνει 2 bytes - τότε χρησιμοποιούμε περισσότερη μνήμη από όσο χρειάζεται το πρόγραμμα και αυξάνουμε το χρόνο επεξεργασίας γιατί η διαχείριση κατά την αποθήκευση και την ανάγνωση της τιμής από την x γίνεται και στα 4 bytes.

Δεν μπορεί ο προγραμματιστής να δεσμεύσει μνήμη σε μια συγκριμένη διεύθυνση γιατί αυτό είναι εργασία του λειτουργικού συστήματος, να βρει επαρκεί χώρο για τη κάθε μεταβλητή.

Νικόλαος Ζ. Ζάχαρης

3.2 Αναγνωριστικά Ονομάτα (identifiers)

Τα Αναγνωριστικά Ονομάτα ή απλά ονόματα χρησιμοποιούνται στις γλώσσες προγραμματισμού για να δηλώσουμε τα ονόματα μεταβλητών και συναρτήσεων.

Κανόνες Σύνταξης στη C

Ένα όνομα στη C μπορεί ν' αποτελείται από γράμματα, ψηφία και τον ειδικό χαρακτήρα "_" που λέγεται υπογράμμιση (underscore).

Ο πρώτος χαρακτήρας πρέπει να είναι γράμμα ή υπογράμμιση.

Δεσμευμένες λέξεις η ονομασίες συναρτήσεων που ορίζονται στις βιβλιοθήκες δεν μπορούν να χρησιμοποιηθούν ως ονομασίες.

Κεφαλαία και μικρά γράμματα θεωρούνται διαφορετικά (Case Sensitive). Δηλαδή οι ονομασίες foros Foros FOROS είναι διαφορετικές μεταξύ τους

Παραδείγματα μη επιτρεπτών ονομασιών:

18Foros for one point my-foros 99999

Προαιρετικός Κανόνες Σύνταξης Ονομάτων

Καλό θα είναι οι ονομασίες να είναι αυτοεπεξηγηματικές.

Foros = Poso * PosostoFPA / 100 ή K = L * M / 100

Παράδειγμα δημιουργίας μεταβλητών

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

int main(int argc, char *argv[])
{
    int sum;                    // Δήλωση ενός ακεραίου αριθμού
    char c;                     // Δήλωση ενός χαρακτήρα
    int x, y, z;                // Δήλωση τριών μεταβλητών σε 1 γραμμή
    double k;                   // Δήλωση ενός πραγματικού αριθμού
    unsigned long foros;        // Δήλωση ενός μεγάλου ακεραίου αριθμού
                                // χωρίς πρόσημο (δηλαδή μόνο θετικοί αριθμοί)
    return  0;
}
```

Ανάθεση τιμής σε μεταβλητή

Η ανάθεση τιμής σε μια μεταβλητή μπορεί να γίνει :

α) κατά την δημιουργία της

```
int x = 7, y = 15;          // Η μεταβλητή x παίρνει τη τιμή 7 και η y το 15
```

β) κατά την εκτέλεση ενός προγράμματος

```
int x, y;
// άλλες εντολές
x = 7;                      // Η μεταβλητή x παίρνει τη τιμή 7
// άλλες εντολές
y = x + 2;                  // Η μεταβλητή y παίρνει τη τιμή 9
```

γ) μέσω της εντολής scanf η οποία διαβάζει μια τιμή από το πληκτρολόγιο του ΗΥ (δες παρακάτω) και την αναθέτει σε μια μεταβλητή

Εκτύπωση μηνυμάτων, τιμών και αποτελεσμάτων στη οθόνη του ΗΥ

Η εντολή printf μας επιτρέπει να εμφανίσουμε είτε μηνύματα είτε τις τιμές των μεταβλητών είτε αποτελέσματα πράξεων στην οθόνη του Ηλεκτρονικού Υπολογιστή. Η σύνταξη της εντολής για την εκτύπωση ενός από τα παραπάνω δεδομένα είναι :

// Μόνο μήνυμα

printf("**Μήνυμα**");

// Μόνο δεδομένα

printf("**Αναγνωριστικά Δεδομένων**", Δεδομένα);

// Συνδυασμός από μηνύματα - δεδομένα

printf("**Μήνυμα Αναγνωριστικά Δεδομένων** ", Δεδομένα);

Εκτύπωση μηνυμάτων, τιμών και αποτελεσμάτων στη οθόνη του ΗΥ (συνέχεια)

Τα δεδομένα που μπορούμε να εκτυπώσουμε στην οθόνη του ΗΥ μπορεί να είναι :

Α) Μηνύματα

```
printf("%s", "Type a number : ");   // πληκτρολόγησε ένα αριθμό
printf("Type a number : ");         // εναλλακτικός τρόπος γραφής
                                    // Το %s είναι χαρακτηριστικό μορφοποίησης
                                    // αλφαριθμητικού (string)
```

Β) Τιμές μεταβλητών

```
int x = 7;
printf("%d", x);                   // θα εκτυπωθεί το 7
```

Γ) Αποτελέσματα πράξεων

```
int y = 7;
printf("%d", y+2);                 // θα εκτυπωθεί το 9
printf("%d", 2+3+7);               // θα εκτυπωθεί το 12
```

Ανάγνωση τιμών από το πληκτρολόγιο

Για την ανάγνωση τιμών από το πληκτρολόγιο θα πρέπει να χρησιμοποιήσουμε την εντολή scanf η οποία συντάσσεται όπως παρακάτω

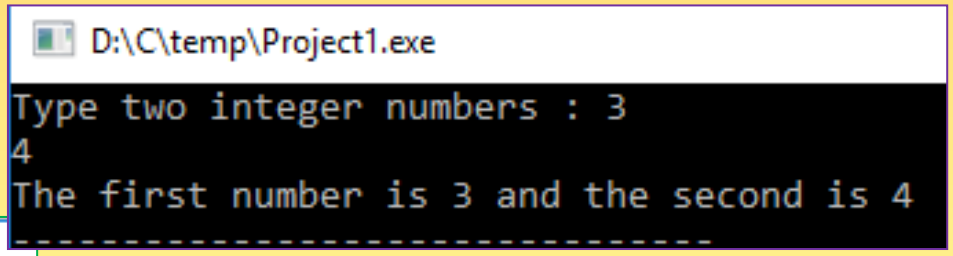
scanf("χαρακτηριστικά ή/και κείμενο μορφοποίησης", Διευθύνσεις μεταβλητών);

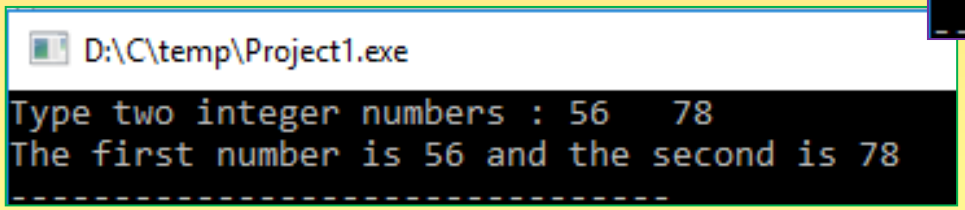
Α) Χαρακτηριστικά μορφοποίησης

```
int main(int argc, char *argv[]) {
        int x, y;
        printf("Type one integer number : ");
        scanf("%d", &x);
        printf("The number is : %d", x);
        return 0;
}
```

Α) Χαρακτηριστικά μορφοποίησης (συνέχεια)

```
int main(int argc, char *argv[]) {
    int x, y;
    printf("Type two integer numbers : ");
    scanf("%d%d", &x, &y);
    printf("The first number is %d and the second is %d", x, y);
    return 0;
}
```

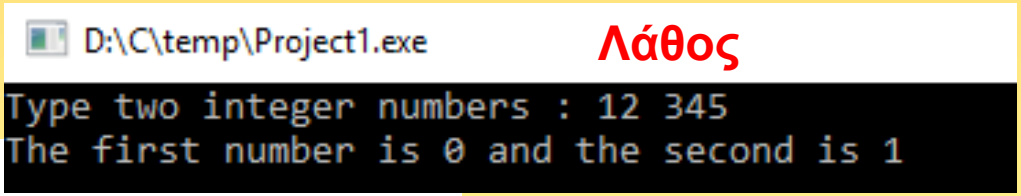




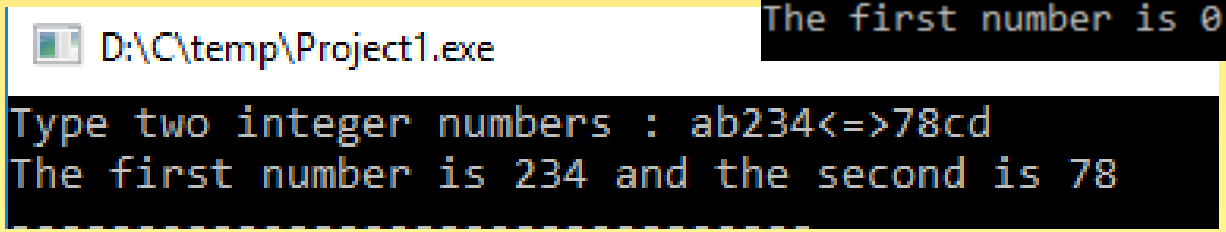
Για την ανάθεση των τιμών θα πρέπει είτε να πληκτρολογήσουμε την πρώτη τιμή, να πατήσουμε enter και να πληκτρολογήσουμε τη δεύτερη τιμή και ξανά enter είτε αφού πληκτρολογήσουμε την πρώτη τιμή να αφήσουμε ένα ή περισσότερα κενά και να πληκτρολογήσουμε τη δεύτερη τιμή και στο τέλος να πατήσουμε enter.

Β) Χαρακτηριστικά και κείμενο μορφοποίησης

```
int main(int argc, char *argv[]) {
        int x, y;
        printf("Type two integer numbers : ");
        scanf("ab%d<=>%dcd", &x, &y);
        printf("The first number is %d and the second is %d", x, y);
        return 0;
}
```



Λάθος



Για την ανάθεση των τιμών θα πρέπει να πληκτρολογήσουμε πρώτα τους χαρακτήρες ab μετά την πρώτη τιμή εν συνεχεία να πληκτρολογήσουμε <=> μετά τη δεύτερη τιμή και τέλος cd και enter.

Αρχικοποίηση μεταβλητών

Η C μετά την δημιουργία μιας μεταβλητής ΔΕΝ τοποθετεί στην μεταβλητή μια αρχική τιμή. Για παράδειγμα στο παρακάτω παράδειγμα :

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


int main(int argc, char *argv[])
{
    int sum;
   printf("H timi toy sum einai %d",sum);
   return0;
}
```

δεν πρέπει να θεωρούμε ότι θα εκτυπωθεί το 0 ή κάποια άλλη τιμή.

Αρχικοποίηση μεταβλητών (συνέχεια)

Η διαχείριση της μνήμης του υπολογιστή γίνεται από το λειτουργικό σύστημα, το οποίο ανταποκρίνεται στις αιτήσεις των προγραμμάτων για παραχώρηση μνήμης η οποία θα χρησιμοποιηθεί για την αποθήκευση δεδομένων. Το λειτουργικό σύστημα έχει τον έλεγχο και παραχωρεί θέσεις μνήμης, οι οποίες δεν χρησιμοποιούνται από άλλη εφαρμογή. Όταν τερματιστεί μια εφαρμογή πάλι το λειτουργικό σύστημα «μαρκάρει» τις θέσεις μνήμης που χρησιμοποιούσε η εφαρμογή σαν διαθέσιμες για χρήση ΑΛΛΑ ΔΕΝ ΚΑΘΑΡΙΖΕΙ ΤΑ ΠΕΡΙΕΧΟΜΕΝΑ ΤΟΥΣ.

Για παράδειγμα αν θεωρήσουμε ότι τώρα κάνουμε εκκίνηση τον υπολογιστή και «τρέχουμε» το πρόγραμμα Microsoft Word το οποίο χρησιμοποιεί μια θέση μνήμης στην οποία αποθηκεύει την τιμή 100. Αν κλείσουμε το Word τότε το λειτουργικό σύστημα θα «μαρκάρει» τη συγκεκριμένη θέση καθώς και όλες που χρησιμοποιούσε το Word σαν ελεύθερες προς χρήση.

Αν εν συνεχεία εκτελέσουμε την δική μας εφαρμογή και ζητήσουμε μνήμη για τη μεταβλητή sum και έστω ότι το λειτουργικό σύστημα μας δίνει τη θέση μνήμης που χρησιμοποιούσε πριν το Word. Τότε κατά την εκτύπωση της τιμής θα εκτυπωθεί το 100, η τιμή που είχε αφήσει το Word.

Είναι υπεύθυνος ο προγραμματιστής για την αρχικοποίηση των μεταβλητών.

Νικόλαος Ζ. Ζάχαρης

Δηλώσεις μεταβλητών

Όλες οι μεταβλητές στη C θα πρέπει να δηλωθούν προτού χρησιμοποιηθούν, και ο λόγος είναι ότι ο μεταγλωττιστής πρέπει να γνωρίζει τον τύπο τους ώστε να κρατήσει τον ανάλογο χώρο στην μνήμη του υπολογιστή.

Όλες οι μεταβλητές μπορούν να δηλωθούν σε οποιοδήποτε σημείο μέσα σε ένα τμήμα εντολών όπως για παράδειγμα στην main στο παραπάνω πρόγραμμα.

```
int main(int argc, char *argv[])
{
  int a = 3, b = 5, sum;
  sum = a + b;
  int x = 10;
  sum = sum + x;
 printf("To apotelesma einai %d",sum);
 return 0;
}
```

Συνήθως οι μεταβλητές δηλώνονται αμέσως μετά την έναρξη του τμήματος εντολών

Δεν υπάρχει πρόβλημα να δηλωθούν και αργότερα αρκεί να προηγείται η δήλωση από την χρήση.

Σταθερές

Ένα αναγνωριστικό που δηλώνεται με το πρόθεμα **const** ονομάζεται σταθερά και η τιμή της δεν μπορεί να μεταβληθεί κατά τη διάρκεια του προγράμματος. Τις σταθερές τις χρησιμοποιούμε μόνο για ανάγνωση και συμβολίζουν πράγματα που δεν μεταβάλλονται. Η δήλωση των σταθερών μπορεί να γίνει με δύο τρόπους :

Α Τρόπος

const Τύπος Όνομα_Μεταβλητής = Τιμή;

Π.χ.

const double pi = 3.14159;

Β Τρόπος

#define name value

Π.χ.

#define pi 3.14159

Σταθερές (συνέχεια)

Σαν σταθερές μπορούν να οριστούν όλοι οι τύποι δεδομένων καθώς και οι παρακάτω ειδικοί χαρακτήρες αλλά μπορούν να χρησιμοποιηθούν και μέσα στο κείμενο μορφοποίησης της εντολής printf καθώς και σε αλφαριθμητικό.

Ειδικοί χαρακτήρες	
\n	αλλαγή γραμμής
\'	μονό εισαγωγικό
****	χαρακτήρας **** (backslash)
\t	αλλαγή στήλης (tab)
\"	διπλό εισαγωγικό
\0	χαρακτήρας με ASCII = 0 (null)
\037	χαρακτήρας με ASCII = 37 (οκταδικό)
\0x1f	χαρακτήρας με ASCII = 1f (δεκαεξαδικό)

Δες το παράδειγμα στη συνέχεια.

Παράδειγμα χρήσης των ειδικών χαρακτήρων

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

#define ENTER '\n'
#define DQT   '\"'

int main(int argc, char *argv[]) {
   printf("He said %cHello%c and %cwalked away.", DQT, DQT, ENTER);
   return 0;
}
```

Ορισμός σταθερών

```
He said "Hello" and
walked away.
```

```
printf("He said \"Hello\" and \nwalked away.");
```

χρήση στη printf

```
printf("%s", "He said \"Hello\" and \nwalked away.");
```

χρήση σε αλφαριθμητικό

Τελεστές και διαχωριστές

Υπάρχουν πολλοί χαρακτήρες που έχουν ειδική σημασία στη γλώσσα C. Για παράδειγμα είναι οι χαρακτήρες :

+, -, *, /

που αντιστοιχούν στις συνήθεις μαθηματικές πράξεις και λέγονται τελεστές και μάλιστα αριθμητικοί τελεστές. Οι ίδιοι χαρακτήρες όπως και οι ειδικοί χαρακτήρες, το κενό (Spacebar), η νέα γραμμή (Enter), το ελληνικό ερωτηματικό χρησιμοποιούνται και σαν διαχωριστές. Για παράδειγμα, κατά την ανάγνωση από το μεταγλωττιστή της παρακάτω γραμμής :

x = a + b;y=3*x;

το κενό τερματίζει την ανάγνωση του ονόματος της μεταβλητής, το x, το ίδιο στο a και στο b. Το ελληνικό ερωτηματικό τερματίζει τη πρώτη δήλωση και μετά αρχίζει η ανάγνωση του y, και τερματίζει με το = και εν συνεχεία το * τερματίζει την ανάγνωση του 3 και το ; τερματίζει την ανάγνωση του x και τη δεύτερη δήλωση. Στη δεύτερη δήλωση δεν υπάρχουν κενά αλλά γίνεται κατανοητή η δήλωση από το μεταγλωττιστή.

Μια ακολουθία χαρακτήρων που περικλείεται μέσα στους χαρακτήρες **" "** ορίζει μια συμβολοσειρά.

"Ενα σφάλμα έχει συμβεί κατά την εκτέλεση"

" c = a + b**"**

Νικόλαος Ζ. Ζάχαρης

Αριθμητικοί Τελεστές

Τελεστής	*Σημασία*
=	Ανάθεση τιμής
+	Πρόσθεση
-	Αφαίρεση, Πρόσημο
*	πολλαπλασιασμός
/	διαίρεση
%	υπόλοιπο
++	αύξηση της τιμής της μεταβλητής κατά 1
--	ελάττωση της τιμής της μεταβλητής κατά 1

Πράξεις με αριθμητικούς τελεστές.

int x = 3;	Ανάθεση τιμής x = 3
x = x + 4;	Πρόσθεση x = 7
x = x - 1;	Αφαίρεση x = 6
x = x % 2;	Υπόλοιπο x = 0
int y = ++x;	Αύξησε το x κατά 1 δηλαδή να γίνει x = 1 και μετά απόδωσε το x στο y. Το y έχει τιμή 1.
y = x++;	Απόδωσε το x στο y και μετά αύξησε το x κατά 1. Το y έχει τιμή 1 και το x έχει τιμή 2.
y = --x;	Μείωσε το x κατά 1 και μετά απόδωσε το x στο y. Το x έχει τιμή 1 και το y έχει τιμή 1.
y = x--;	Απόδωσε το x στο y και μετά μείωσε το x κατά 1. Το y έχει τιμή 1 και το x έχει τιμή 0.

Νικόλαος Ζ. Ζάχαρης

Παράδειγμα χρήσης αριθμητικών τελεστών.

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

int main(int argc, char *argv[]) {
 int x = 3;
    printf("x = %d\n", x);
    x = x + 4;
    printf("x = %d\n", x);
    x = x - 1;
    printf("x = %d\n", x);
    x = x % 2;
    printf("x = %d\n", x);
    int y = ++x;
    printf("x = %d \t y = %d\n", x , y);
    y = x++;
    printf("x = %d \t y = %d\n", x , y);
    y = --x;
    printf("x = %d \t y = %d\n", x , y);
    y = x--;
    printf("x = %d \t y = %d\n", x , y);
    return 0;
}
```

```
x = 3
x = 7
x = 6
x = 0
x = 1    y = 1
x = 2    y = 1
x = 1    y = 1
x = 0    y = 1
```

Νικόλαος Ζ. Ζάχαρης

Δηλώσεις μεταβλητών

Όλες οι μεταβλητές στη C μπορούν να δηλωθούν και εκτός του τμήματος εντολών της main όπως παρακάτω :

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


int a = 3, b = 5, sum;


int main(int argc, char *argv[])
{
  sum = a + b;
  int x = 10;
  sum = sum + x;
  printf("The result is %d",sum);
  return 0;
}
```

Οι μεταβλητές που δηλώνονται εκτός οποιουδήποτε τμήματος εντολών ονομάζονται **καθολικές** (**global**).

Οι μεταβλητές που δηλώνονται οποιουδήποτε σε ένα τμήμα εντολών ονομάζονται **τοπικές** (**local**).

Προτεραιότητα πράξεων (1-2)

Όλες οι γλώσσες προγραμματισμού ορίζουν μια προτεραιότητα εκτέλεσης των αριθμητικών πράξεων δηλαδή η παρακάτω έκφραση :

```
int x;
x = 2 + 3 * 5;
```

αποδίδει στη μεταβλητή x την τιμή 17 γιατί η γλώσσα C ορίζει ότι ο πολλαπλασιασμός έχει μεγαλύτερη προτεραιότητα από τη πρόσθεση με αποτέλεσμα να υπολογιστεί πρώτα το γινόμενο 3 * 5 και στο αποτέλεσμα να προστεθεί το 2.

Ο κατασκευαστής της κάθε γλώσσας προγραμματισμού ορίζει στο εγχειρίδιο χρήσης την προτεραιότητα των αριθμητικών πράξεων όμως οι παρενθέσεις έχουν πάντα την μεγαλύτερη προτεραιότητα. Για αυτό το λόγο μπορούμε να χρησιμοποιήσουμε παρενθέσεις για να δηλώσουμε με ακρίβεια την σειρά με την οποία θέλουμε να εκτιμηθεί η κάθε συνθήκη. Αν στην ανωτέρω έκφραση θέλουμε να γίνει πρώτα η πρόσθεση τότε θα πρέπει να γράψουμε :

```
x = (2 + 3) * 5;
```

Η χρήση των παρενθέσεων δεν αυξάνει το χρόνο εκτέλεσης των πράξεων.

Προτεραιότητα πράξεων (2-2)

Category	Operator	Associativity
Postfix	() [] -> . ++ - -	Left to right
Unary	+ - ! ~ ++ - - (type)* & sizeof	Right to left
Multiplicative	* / %	Left to right
Additive	+ -	Left to right
Shift	<< >>	Left to right
Relational	< <= > >=	Left to right
Equality	== !=	Left to right
Bitwise AND	&	Left to right
Bitwise XOR	^	Left to right
Bitwise OR	|	Left to right
Logical AND	&&	Left to right
Logical OR	||	Left to right
Conditional	?:	Right to left
Assignment	= += -= *= /= %=>>= <<= &= ^= |=	Right to left
Comma	,	Left to right

Πρόσημο

Πράξη

Πηγή : https://www.tutorialspoint.com/cprogramming/c_operators_precedence.htm

Άσκηση

Μία εταιρεία έχει υπαλλήλους που εργάζονται ορισμένες ημέρες του μήνα και με διαφορετική ημερήσια αποζημίωση. Να αναπτύξετε μία εφαρμογή η οποία θα ζητά το πλήθος των ημερών εργασίας καθώς και την ημερήσια αποζημίωση και θα υπολογίζει και θα εμφανίζει τη μεικτή αμοιβή του εργαζόμενου. Επίσης, η εφαρμογή θα εμφανίζει το καθαρό ποσό που πρέπει να εισπράξει ο εργαζόμενος, και αυτό θα υπολογίζεται από τη μικτή αμοιβή αν αφαιρέσουμε το 15% για ασφάλεια και το 20% για την εφορία. Για παράδειγμα αν ένας υπάλληλος εργάστηκε 10 ημέρες με ημερήσια αποζημίωση 30 ευρώ τότε το πρόγραμμα θα πρέπει να εμφανίσει;

Μεικτή αμοιβή : 300 (10 * 30)

Ασφάλεια : 45 (0.15 * 300)

Εφορία : 60 (0.20 * 300)

Καθαρό ποσό : 195 (300 – 45 – 60)

```
Type days : 10
Type daily salary  : 30
Gross pay  : 300.000000
Insurance  : 45.000000
Tax  : 60.000000
Net pay  : 195.000000
```

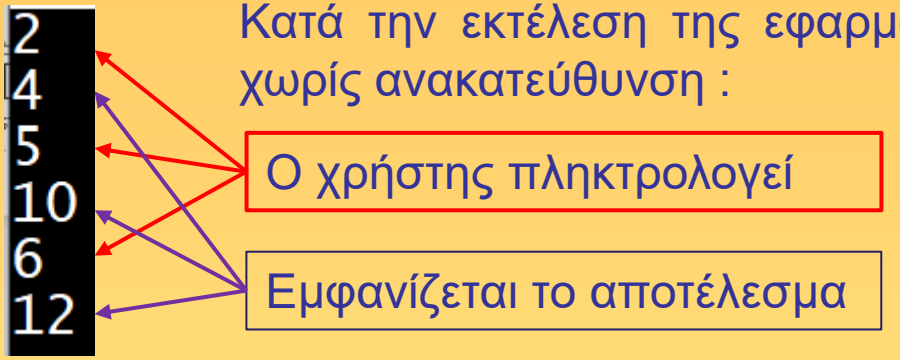
Ο κώδικας της άσκησης

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

int main(int argc, char *argv[]) {
    int days;
    float dailypay, grosspay, insurance, tax, money;

    printf("Type days : ");
    scanf("%d", &days);
    printf("Type daily salary  : ");
    scanf("%f", &dailypay);
    grosspay = dailypay * days;
    printf("Gross pay  : %f\n", grosspay);

    insurance = grosspay * 0.15;
    printf("Insurance  : %f\n", insurance);

    tax = grosspay * 0.20;
    printf("Tax  : %f\n", tax);

    money = grosspay - insurance - tax;
    printf("Net pay  : %f\n", money);

    return  0;
}
```

Νικόλαος Ζ. Ζάχαρης

Ανακατεύθυνση εισόδου εξόδου (1-2)

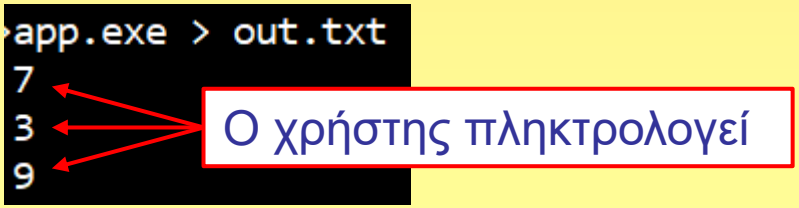
```c
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[]) {
    int i = 0, no;
    scanf("%d", &no);
    printf("%d\n", 2 * no);

    scanf("%d", &no);
    printf("%d\n", 2 * no);

    scanf("%d", &no);
    printf("%d\n", 2 * no);

    return 0;
}
```



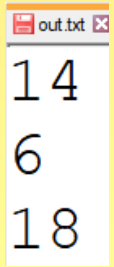
Κατά την εκτέλεση της εφαρμογής χωρίς ανακατεύθυνση :

Η προκαθορισμένη είσοδος για το εκτελέσιμο πρόγραμμα είναι το πληκτρολόγιο και η έξοδος είναι η οθόνη. Χωρίς αλλαγή στο πρόγραμμα μπορείτε να κάνετε ανακατεύθυνση εισόδου και εξόδου σε αρχεία.

α) Ανακατεύθυνση εξόδου



Οι τιμές εξόδου δεν εμφανίζονται στην οθόνη αλλά αποθηκεύονται στο αρχείο out.txt, το οποίο δημιουργείται τώρα.



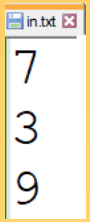
Ο τελεστής ανακατεύθυνσης > δημιουργεί νέο αρχείο εξόδου ενώ ο τελεστής >> κάνει προσάρτηση(append) στο αρχείο εξόδου.

Νικόλαος Ζ. Ζάχαρης

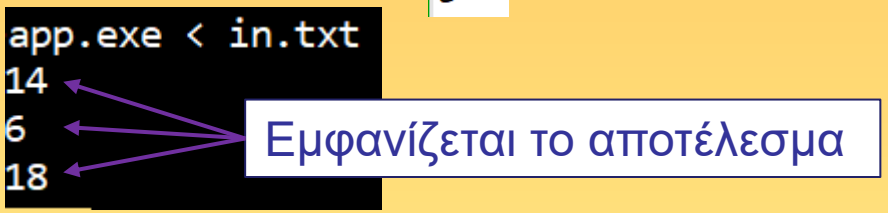
Ανακατεύθυνση εισόδου εξόδου (2-2)

β) Ανακατεύθυνση εισόδου

Δημιουργούμε το αρχείο in.txt το οποίο περιέχει τις τιμές.



Κατά την εκτέλεση της εφαρμογής ο χρήστης δεν πληκτρολογεί τις τιμές αλλά χρησιμοποιούνται αυτές από το αρχείο in.txt



Ο τελεστής ανακατεύθυνσης < χρησιμοποιείται για την ανάγνωση τιμών από αρχείο.

γ) Ανακατεύθυνση εισόδου - εξόδου

Μπορούμε να κάνουμε ταυτόχρονα ανακατεύθυνση εισόδου και εξόδου σε αρχεία.

```
app.exe < in.txt > out.txt
```

Κατά την εκτέλεση της εφαρμογής ο χρήστης δεν πληκτρολογεί τις τιμές αλλά χρησιμοποιούνται αυτές από το αρχείο in.txt και επίσης δεν εμφανίζονται αποτελέσματα στην οθόνη αλλά αποθηκεύονται στο αρχείο out.txt

Αν χρησιμοποιηθεί ο τελεστής >> θα γίνει προσάρτηση στο αρχείο.