Νίκος Μιχαλοδημητράκης

Σημειώσεις C

Σημειώσεις Εργαστηρίου Πληροφορικής

Νικόλαος Μιχαλοδημητράκης
1/10/2013

Πίνακας περιεχομένων

1ο Μέρος – Εισαγωγή – Περιβάλλον εργασίας – Βασικές Έννοιες

Τι είναι προγραμματισμός
Οι τεχνικές και μεθοδολογίες παρουσίασης λύσης σε ένα πρόβλημα. Ουσιαστικά περιγράφουμε όλα τα απαραίτητα και αναγκαία βήματα για να αυτοματοποιήσουμε την επίλυση ενός προβλήματος, χρησιμοποιώντας πράξεις και συγκρίσεις με τα δεδομένα μας για να υπολογιστεί η λύση. Το δύσκολο είναι να βρούμε τη λογική για τη λύση ενός προβλήματος παρά να την περιγράψουμε με μια γλώσσα προγραμματισμού.

Τι είναι η γλώσσα προγραμματισμού C
Η C είναι μια γενικής χρήσης ανώτερου επιπέδου γλώσσα προγραμματισμού η οποία αναπτύχτηκε στις αρχές της δεκαετίας 1970-1980 από τον Dennis Richie στα εργαστήρια Bell Labs για να χρησιμοποιηθεί για την ανάπτυξη του λειτουργικού συστήματος UNIX. Απο τότε χρησιμοποιείται ευρύτατα, και ιδιαίτερα για ανάπτυξη προγραμμάτων συστήματος (system software) αλλά και για απλές εφαρμογές. Οι λόγοι της ραγδαίας ανάπτυξης της συγκεκριμένης γλώσσας προγραμματισμού είναι η ταχύτητά της, καθώς και το γεγονός ότι είναι διαθέσιμη στα περισσότερα σημερινά λειτουργικά συστήματα.

Υπάρχουν διάφορες εκδόσεις της γλώσσας C, ανάλογα την εταιρεία και την εξειδικευμένη χρήση της. Το πρότυπο ANSI C είναι αυτό που διδάσκεται στα περισσότερα ακαδημαϊκά ιδρύματα στο κόσμο. Η επέκταση C++ είναι ίσως η πιο γνωστή εμπορική έκδοση της C στην βιομηχανία της πληροφορικής και όχι μόνο.

Περιβάλλον προγραμματισμού
Το περιβάλλον προγραμματισμού που θα χρησιμοποιήσουμε είναι αυτό της Bloodshed Dev C++. Εναλλακτικά υπάρχει και το περιβάλλον της MiracleC ή της Borland Turbo C για όποιον το επιθυμεί. Όλες οι σημειώσεις και τα παραδείγματα αναφέρονται στο περιβάλλον της Bloodshed Dev C++ και έχουν ελεγχθεί πως λειτουργούν κανονικά σε αυτό το περιβάλλον.

Εκκίνηση της Bloodshed Dev C++
Η Bloodshed Dev C++ ξεκινά από την ακόλουθη διαδρομή:

Έναρξη → Προγράμματα → Bloodshed Dev C++ → Dev-C++.

Συγγραφή ενός προγράμματος C
Όλα τα προγράμματα της C, εντολές και σχόλια, τα γράφουμε χρησιμοποιώντας λατινικούς χαρακτήρες. Οι ελληνικοί χαρακτήρες δυστυχώς δεν υποστηρίζονται στον κώδικα της γλώσσας C.

Εκτέλεση ενός προγράμματος C
Εκτελούμε οποιοδήποτε πρόγραμμα C επιλέγοντας την επιλογή **Compile & Run** από τη γραμμή Εργαλείων. Επίσης μπορούμε να χρησιμοποιήσουμε το πλήκτρο: **F9**.

Προβολή αποτελεσμάτων εκτέλεσης προγράμματος
Για να δούμε τα αποτελέσματα της εκτέλεσης ενός προγράμματος που έχουμε γράψει στο περιβάλλον εργασίας της Bloodshed Dev C++ πρέπει να συμπεριλάβουμε τη βιβλιοθήκη **stdlib.h** με την εντολή:

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

καθώς και την γραμμή:

```
system("PAUSE");
```

σαν την τελευταία γραμμή στο πρόγραμμά μας ώστε να διακόψουν την εκτέλεση έως ότου πατήσουμε κάποιο πλήκτρο πριν την έξοδο.

Ένας άλλος τρόπος για να εκτελέσουμε και να δούμε τα αποτελέσματα είναι να «μεταφράσουμε» το πρόγραμμα C σε ένα εκτελέσιμο πρόγραμμα τύπου .EXE και να εκτελέσουμε το συγκεκριμένο .EXE πρόγραμμα μέσα από τη γραμμή εντολών των Windows.

Δομή ενός προγράμματος C

```
#include <stdio.h>
#include <stdlib.h>
/* This is a comment ignored by the compiler */

main()
{
    printf(" Hello World \n ");
    printf(" My name is: "); /* A comment is
    allowed to be continued on another line */
    printf("Mpampis \n");    /* that's my name  */

    system("PAUSE");  /* this is a pause */
}
```

Ένα απλό πρόγραμμα της γλώσσας C αποτελείται από το επάνω μέρος, το μέρος δηλώσεων όπου δηλώνουμε βοηθητικά στοιχεία για το πρόγραμμά μας και το κύριο μέρος του προγράμματος, τη συνάρτηση main().

Η συνάρτηση main()
Η βασική συνάρτηση main() είναι απαραίτητη για να εκτελεστεί το πρόγραμμά μας, διότι ο μεταγλωττιστής αυτόματα εκτελεί αυτή τη συνάρτηση όταν εμείς εκτελέσουμε το πρόγραμμά μας.

Το τμήμα δηλώσεων
Εδώ δηλώνουμε όλα αυτά τα στοιχεία της γλώσσας προγραμματισμού C (και του μεταγλωττιστή της) που είναι απαραίτητα για την σωστή εκτέλεση του προγράμματός μας. Δηλώνουμε βιβλιοθήκες με έτοιμο κώδικα που εκτελεί κάποιες λειτουργίες (που κάποιος άλλος έχει γράψει πριν από εμάς ώστε να μην χρειάζεται κάθε φορά να ξανά-εφευρίσκουμε τον τροχό!). Επίσης δηλώνουμε βοηθητικές μεταβλητές και σταθερές για την επεξεργασία των δεδομένων μας (κάτι αντίστοιχο με τον άγνωστο χ στα μαθηματικά).

Βασικές έννοιες της γλώσσας C

Σχόλια
Τα σχόλια δεν έχουν καμία λειτουργία στη γλώσσα προγραμματισμού αλλά είναι απαραίτητα σε κάποιον ο οποίος θέλει να «διαβάσει» ένα πρόγραμμα (σε οποιαδήποτε γλώσσα προγραμματισμού). Όσο αυξάνει η πολυπλοκότητα και το μέγεθος ενός προγράμματος τόσο πιο δύσκολο έως αδύνατον γίνεται να διαβάσουμε και να κατανοήσουμε τι κάνει ένα πρόγραμμα. Ειδικά εάν ένα πρόγραμμα έχει γραφτεί στο παρελθόν ίσως και από κάποιον άλλο προγραμματιστή.

- **/* οτιδήποτε υπάρχει ανάμεσα στους χαρακτήρες θεωρείται σχόλιο και δεν λαμβάνεται υπόψη κατά τη μεταγλώττιση */**
- Ένα σχόλιο μπορεί να καταλαμβάνει από κάποιους χαρακτήρες έως αρκετές σειρές.

- **Είναι σημαντικό να είμαστε λυτοί αλλά συγκεκριμένοι στα σχόλιά μας,** δεν χρειάζεται να γράφουμε ολόκληρη έκθεση – μόνο αυτό που κάνει το παρακάτω κομμάτι κώδικα.

Τα δεδομένα
Τα δεδομένα (πληροφορίες-data) είναι απαραίτητα στοιχεία ενός προγράμματος, καθώς οι βασικές λειτουργίες ενός προγράμματος είναι η επεξεργασία αυτών των δεδομένων και η εξαγωγή αποτελεσμάτων (δηλαδή άλλα δεδομένα). Απαιτείται λοιπόν η δέσμευση κάποιων χώρων μνήμης για να αποθηκευτούν αυτά τα δεδομένα, κατά την διάρκεια της εκτέλεσης του προγράμματος. Αυτοί οι χώροι μνήμης που τους χρησιμοποιούμε για την φύλαξη δεδομένων όταν εκτελείται ένα πρόγραμμα, ονομάζονται ανάλογα με την χρήση τους σταθερές ή μεταβλητές και κάθε γλώσσα προγραμματισμού μας δίνει την δυνατότητα με διάφορους τρόπους να δηλώσουμε (καθορίσουμε) κάποιους τύπους μεταβλητών/σταθερών, τις οποίες θα χρησιμοποιήσουμε για την αποθήκευση δεδομένων, όταν εκτελείται ένα πρόγραμμα.

Τύποι δεδομένων
Ένας τύπος στην C, αλλά και σε πολλές άλλες δημοφιλείς γλώσσες προγραμματισμού, καθορίζει τον τρόπο χειρισμού κάποιας μεταβλητής. Συγκεκριμένα προσδιορίζει το είδος των τιμών που μπορεί να αποθηκεύσει, και καθορίζει ένα σύνολο λειτουργιών που είναι δυνατόν να εφαρμοσθούν πάνω στις μεταβλητές αυτού του τύπου. Η C έχει κάποιους βασικούς τύπους δεδομένων, οι οποίοι έχουν προκαθοριστεί και μπορούν να χρησιμοποιηθούν οπουδήποτε σε ένα πρόγραμμα, υπό τον όρο να χρησιμοποιούνται σωστά (όλοι οι τύποι δεδομένων δεν υποστηρίζονται απαραίτητα από όλους τους compilers στις διάφορες εκδόσεις της C).

Βασικοί Τύποι Δεδομένων στην C
Η C έχει πέντε βασικούς τύπους δεδομένων:

```
char   (character)
int    (integer)
float  (floating point)
double (double floating point)
void   (no value)
```

Οι πιο γνωστοί τύποι δεδομένων είναι οι int, double, char. Σε αντίθεση με άλλες γλώσσες προγραμματισμού η C δεν έχει τύπο δεδομένων Σωστό/Λάθος (Boolean), υπάρχουν όμως οι τιμές TRUE και FALSE.

Όλοι οι άλλοι τύποι της C βασίζονται σ' αυτούς. Όλοι οι βασικοί τύποι εκτός από τον τύπο void μπορεί ν' αλλάξουν γράφοντας πριν από τον τύπο τον κατάλληλο μετασχηματισμό. Οι μετασχηματισμοί αυτοί είναι οι: signed, unsigned, long, και short. Περισσότερες πληροφορίες μπορείτε να βρείτε στη βιβλιογραφία της θεωρίας.

Οι μεταβλητές
Αντίθετα με τις σταθερές, οι τιμές των μεταβλητών είναι δυνατόν να αλλάξουν κατά την διάρκεια της εκτέλεσης ενός προγράμματος. Οι γλώσσες προγραμματισμού μας δίνουν την δυνατότητα να καθορίσουμε μεταβλητές και στην συνέχεια να τους δώσουμε όποια τιμή επιθυμούμε, που έχει βέβαια σχέση με το πρόγραμμα. Η δήλωση των μεταβλητών γίνεται συνήθως στην αρχή του προγράμματος.

Ονομασίες μεταβλητών
Οι ονομασίες των μεταβλητών πρέπει να αρχίζουν με γράμμα ή με κάτω παύλα. Στην περιγραφή της ονομασίας μπορείτε να χρησιμοποιήσετε και αριθμούς, αλλά δεν επιτρέπονται άλλοι ειδικοί χαρακτήρες. **Τα πεζά και τα κεφαλαία γράμματα παίζουν ρόλο (είναι διαφορετικά!).** Φυσικά

για την ονομασία των μεταβλητών σας δεν θα πρέπει να χρησιμοποιείστε ονόματα που είναι δεσμευμένες λέξεις για την γλώσσα C.

Οι δεσμευμένες λέξεις της standard C είναι:

DATA TYPES	STORAGE CLASSES	STATMENTS
char	auto	break
double	extern	case
enum	register	continue
float	static	default
int		do
long		else
short		for
struct		goto
union		if
unsigned		return
void		switch
sizeof		while
typdef		

Βέβαια ανάλογα με τον compiler που χρησιμοποιείτε, είναι δυνατόν να υπάρχουν και άλλες δεσμευμένες λέξεις της C (cdel, const, far, fortran, huge, near, pascal, signed, volatile κ.α).

Μια πολύ καλή και απλή μέθοδος για να αποφύγουμε ενδεχόμενα λάθη δήλωσης μιας δεσμευμένης λέξης σαν όνομα μιας μεταβλητής είναι να χρησιμοποιήσουμε τα greeklish σαν όνομα μεταβλητής. Έτσι ονόματα όπως: **ginomeno, plithos** κλπ. είναι απολύτως αποδεκτά από την C και συγχρόνως είναι υποδηλώνουν την «έννοια» του αντικειμένου που περιγράφουν.

Δήλωση Μεταβλητών
Τα αναγνωριστικά στη C μπορούν να έχουν όσους χαρακτήρες θέλουμε. Αν το αναγνωριστικό είναι εξωτερικό όνομα (όνομα συνάρτησης η καθολική μεταβλητή) τότε μόνο οι έξι πρώτοι χαρακτήρες είναι σημαντικοί διαφορετικά για εσωτερικά ονόματα οι πρώτοι 31 χαρακτήρες είναι σημαντικοί. Τα κεφαλαία γράμματα στην C είναι διαφορετικά από τα μικρά.

Η δήλωση μιας μεταβλητής έχει την γενική μορφή:

```
<τύπος> <λίστα μεταβλητών>;
```

```
int i=0,j;   Δήλωση με ορισμό τιμής
char q='?';
short int si;  Δήλωση χωρίς ορισμό τιμής
float f,g;
```

Οι σταθερές
Αρκετά προγράμματα απαιτούν ορισμένα δεδομένα τα οποία δεν αλλάζουν (ή δεν πρέπει ν' αλλαχθούν) ποτέ κατά την διάρκεια της εκτέλεσης του. Για παράδειγμα, όταν γράφετε ένα πρόγραμμα, μπορείτε να καθορίσετε ότι η τιμή του π είναι 3.14159 και να χρησιμοποιείτε αυτή την τιμή όταν την χρειάζεστε, ξέροντας ότι είναι διαθέσιμη και σωστή. Οι σταθερές ορίζονται με εντολές του προεπεξεργαστού με την οδηγία **#define** και συνήθως χρησιμοποιούμε κεφαλαία γράμματα για να τις περιγράψουμε:

```
#define PI 3.14159
#define G 9.81
```

(Προσέξτε! αυτή η εντολή δεν τελειώνει με το ερωτηματικό ";". Επίσης υπάρχει κενό μεταξύ του PI και του 3.14159)

Οι βιβλιοθήκες
Μπορούμε να χρησιμοποιήσουμε βιβλιοθήκες με έτοιμο κώδικα (συναρτήσεις) ώστε να μην χρειάζεται κάθε φορά που χρησιμοποιούμε κάποιες επαναλαμβανόμενες διαδικασίες να τις ξαναγράφουμε από την αρχή. Η δήλωση μιας μεταβλητής γίνεται στο αρχικό κομμάτι του προγράμματός μας με τη γενική μορφή: `#include <όνομα βιβλιοθήκης>`
Μπορούμε να δηλώσουμε πολλές βιβλιοθήκες αλλά μία-μία σε κάθε γραμμή π.χ.

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

Σύνταξη εντολών C

Οι εντολές στην γλώσσα προγραμματισμού C συντάσσονται στην αγγλική γλώσσα και τελειώνουν πάντα με το σύμβολο **semi-colon (ελληνικό ερωτηματικό),** εκτός από τη δήλωση σταθερών, βιβλιοθηκών και γενικά εντολών προεπεξεργαστή που δεν τερματίζουν με ;

2ο Μέρος – Εντολές Εισόδου, Εξόδου – Καταχώρηση

Εντολή εξόδου putchar()

Η εντολή εξόδου putchar() μας επιτρέπει να παρουσιάσουμε ένα χαρακτήρα στην οθόνη. Δέχεται σαν όρισμα μια μεταβλητή τύπου χαρακτήρα (char) και παρουσιάζει την τιμή της μεταβλητής στην οθόνη. Εάν θέλουμε να παρουσιάσουμε πολλούς χαρακτήρες τότε πρέπει να χρησιμοποιήσουμε πολλές φορές την εντολή putchar(), μία κλήση της εντολής για κάθε χαρακτήρα.

Παράδειγμα 2-1:
Το παρακάτω παράδειγμα τυπώνει τα αποτελέσματά του, χρησιμοποιώντας την εντολή εξόδου putchar() πολλές φορές, σε μία και μόνο γραμμή.

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

main()
{
   char letter1 = 'A';
   char letter2 = 'B';
```

```
    char letter3 = 'C';
    char letter4 = ' ';

    putchar(letter4);
    putchar(letter1);
    putchar(letter4);
    putchar(letter2);
    putchar(letter4);
    putchar(letter3);
    putchar(letter4);

    system("PAUSE"); /* pause (wait for key to continue) */
}
```

Εντολή εισόδου getchar()

Η εντολή εξόδου getchar() μας επιτρέπει να διαβάσουμε ένα χαρακτήρα από το πληκτρολόγιο. Ο χαρακτήρας αποθηκεύεται σαν αποτέλεσμα της κλήσης της συνάρτησης σαν τιμή σε μια μεταβλητή τύπου char. Εάν θέλουμε να διαβάσουμε πολλούς χαρακτήρες τότε πρέπει να χρησιμοποιήσουμε πολλές φορές την εντολή getchar (), μία κλήση της εντολής για κάθε χαρακτήρα. Επειδή η ροή των χαρακτήρων από το πληκτρολόγιο από την getchar() διαβάζεται «όλη μαζί» σε πολλές κλήσεις της getchar() θα πρέπει να δίνουμε τους χαρακτήρες όλους μαζί και στο τέλος να πατάμε το ENTER.

Παράδειγμα 2-2:
Το παρακάτω παράδειγμα διαβάζει από το πληκτρολόγιο 2 χαρακτήρες χρησιμοποιώντας την εντολή εισόδου getchar() και τυπώνει τα αποτελέσματά του, χρησιμοποιώντας την εντολή εξόδου putchar() πολλές φορές, σε μία και μόνο γραμμή.

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

main()
{
    char letter1;
    char letter2;
    char space = ' ';

    letter1 = getchar();
    letter2 = getchar();

    putchar(space);
    putchar(letter1);
    putchar(space);
    putchar(letter2);
    putchar(space);

    system("PAUSE"); /* pause (wait for key to continue) */
}
```

Εντολή εξόδου puts()

Η εντολή εξόδου puts() μας επιτρέπει να παρουσιάσουμε πολλούς χαρακτήρες στην οθόνη. Δέχεται σαν όρισμα μια σειρά από χαρακτήρες και την παρουσιάζει στην οθόνη. Η εντολή εξόδου puts() πάντοτε αλλάζει την σειρά στην οθόνη μετά την εκτέλεσή της.

Παράδειγμα 2-3:
Το παρακάτω παράδειγμα τυπώνει τα αποτελέσματά του, χρησιμοποιώντας την εντολή εξόδου puts() σε μία και μόνο γραμμή.

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

main()
{
   puts(" HELLO WORLD!");

   system("PAUSE"); /* pause (wait for key to continue) */
}
```

Εντολή εισόδου gets()

Η εντολή εισόδου gets() μας επιτρέπει να διαβάσουμε μια συμβολοσειρά (πολλούς χαρακτήρες) από το πληκτρολόγιο. Δέχεται σαν όρισμα μια συμβολοσειρά στην οποία θα αποθηκευτούν σαν τιμή όλοι οι χαρακτήρες που θα δώσουμε. Η εντολή εξόδου puts() πάντοτε αλλάζει την σειρά στην οθόνη μετά την εκτέλεσή της.

Παράδειγμα 2-4:
Το παρακάτω παράδειγμα τυπώνει τα αποτελέσματά του, χρησιμοποιώντας την εντολή εξόδου puts() σε μία και μόνο γραμμή.

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

main()
{
   char name[20];

   puts(" Dose to onoma sou:");
   gets(name);
   puts(" Geia soy :");
   puts(name);

   system("PAUSE"); /* pause (wait for key to continue) */
}
```

Εντολή εξόδου printf()

Η εντολή (συνάρτηση) εξόδου **printf()** μας επιτρέπει να παρουσιάσουμε αποτελέσματα στον χρήστη με τη διαμόρφωση που θέλουμε εμείς. Η εντολή printf επιγραμματικά κάνει τα ακόλουθα:

- Παρουσιάζει πληροφορίες στην οθόνη
- Παρουσιάζει στην οθόνη κείμενο που έχουμε δηλώσει μέσα σε διπλά εισαγωγικά " "
- Απαιτεί ακολουθία διαφυγής για να παρουσιάσει ορισμένους ειδικούς χαρακτήρες

- Παρουσιάζει μεταβλητές χρησιμοποιώντας τον ειδικό χαρακτήρα μετατροπής
- Επιστρέφει τον αριθμό των χαρακτήρων που τυπώνονται στην οθόνη

Παράδειγμα 2-5:
Το παρακάτω παράδειγμα τυπώνει τα αποτελέσματά του, χρησιμοποιώντας την εντολή εξόδου printf 3 φορές, σε μία και μόνο γραμμή.

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

main()
{
     printf("hello");
     printf("world! ");
     printf("This is my first program!!!");

     system("PAUSE");    /* pause (wait for key to continue) */
}
```

Για να γράψουμε σε παραπάνω από μία γραμμές θα πρέπει να χρησιμοποιήσουμε μια ακολουθία χαρακτήρων που συνήθως λέγεται ακολουθία διαφυγής (escape sequence), και πιο συγκεκριμένα το συνδυασμό ***\n.***

Παράδειγμα 2-6:
Το παρακάτω παράδειγμα τυπώνει τα αποτελέσματά του χρησιμοποιώντας την εντολή εξόδου printf 3 φορές σε πολλές γραμμές κάνοντας χρήση της ακολουθίας διαφυγής \n .

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

main()
{
     printf("hello\n");
     printf("world!\n");
     printf("This is my first program!!!");
     system("PAUSE");    /* pause (wait for key to continue) */
}
```

Οι επιτρεπτές ακολουθίες διαφυγής της standard C είναι οι ακόλουθες:

Ακολουθία	Περιγραφή
\a	Beep (ηχητικό σήμα)
\b	Οπισθοχώρηση κέρσορα (όχι διαγραφή)
\f	Αλλαγή σελίδας (σε εκτυπωτή)
\n	Νέα γραμμή
\r	Επαναφορά κέρσορα στην αρχή της γραμμής
\t	Οριζόντιος στηλογνώμονας (tab)
\v	Κατακόρυφος στηλογνώμονας (κάτω μία γραμμή)
****	Ανάποδη κάθετος
\?	Λατινικό ερωτηματικό
\'	Μονό εισαγωγικό
\"	Διπλό εισαγωγικό
\0	Το κενό (null) byte

Παράδειγμα 2-7:
Το παρακάτω παράδειγμα τυπώνει τα αποτελέσματά του χρησιμοποιώντας την εντολή εξόδου printf 4 φορές σε πολλές γραμμές κάνοντας χρήση διάφορων ακολουθιών διαφυγής για να μορφοποιήσει το αποτέλεσμα.

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

main()
{
     printf(" Hello \n");
     printf(" \t have a nice day \n");
     printf(" I like \'C\' \n");
     printf("\a");        /* make beep! */
     system("PAUSE");     /* pause (wait for key to continue) */
}
```

Εντολή εισόδου scanf()

Η εντολή (συνάρτηση) εισόδου **scanf()** μας επιτρέπει να εισάγουμε δεδομένα από το πληκτρολόγιο σε μεταβλητές του προγράμματός μας. Η εντολή scanf επιγραμματικά συντάσσεται ως εξής:

```
scanf("χαρακτήρες μετατροπής", μεταβλητή);
```

Οι χαρακτήρες μετατροπής προσδιορίζουν το είδος των δεδομένων που θα καταχωρηθούν στην μεταβλητή. Η μεταβλητή στην πραγματικότητα είναι ένας δείκτης, για εμάς αυτό θα σημαίνει πως μπροστά από το όνομα της μεταβλητής θα μπαίνει πάντα ο χαρακτήρας &. Επίσης η εντολή scanf επιστρέφει το πλήθος των εισαγόμενων δεδομένων.

Οι χαρακτήρες μετατροπής είναι ίδιοι με αυτούς που χρησιμοποιούμε με την εντολή printf. Οι χαρακτήρες μετατροπής που χρησιμοποιούμε πιο συχνά είναι οι παρακάτω:

Χαρακτήρες μετατροπής	Περιγραφή
%d	Τύπος int.
%f	Τύπος float, double
%c	Τύπος char. Εκτύπωση ενός χαρακτήρα
%s	Τύπος char. Εκτύπωση πολλών χαρακτήρων

Παράδειγμα 2-8:

Το παρακάτω παράδειγμα υπολογίζει και παρουσιάζει στην οθόνη μας το άθροισμα 2 ακεραίων αριθμών που του δίνουμε εμείς από το πληκτρολόγιο. Χρησιμοποιεί την εντολή scanf ώστε να διαβάσει τα δεδομένα από το πληκτρολόγιο και την εντολή printf ώστε να τυπώσει το αποτέλεσμα.

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

main()
{
  int op1 = 0;
  int op2 = 0;
  int sum = 0;
```

```
  printf("\nEnter first operand: ");
  scanf("%d", &op1);
  printf("Enter second operand: ");
  scanf("%d", &op2);
  sum = op1 + op2;
  printf("The result is %d\n", sum);

  system("PAUSE");  /* pause (wait for key to continue) */
}
```

Παράδειγμα 2-9:

Το παρακάτω παράδειγμα διαβάζει από το πληκτρολόγιο το όνομά μας και μας λέει καλημέρα! Χρησιμοποιεί την εντολή scanf ώστε να διαβάσει το όνομα από το πληκτρολόγιο και την εντολή printf ώστε να τυπώσει το μήνυμα στην οθόνη.

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

main()
{
  char onoma[20];  /* kolpo gia to onoma */

  printf("\Pos se lene?: ");
  scanf("%s", &onoma);
  printf("\n Kalimera %s\n", onoma);

  system("PAUSE");  /* pause (wait for key to continue) */
}
```

Παράδειγμα 2-10:

Το παρακάτω παράδειγμα υπολογίζει και παρουσιάζει στην οθόνη μας το πλήθος και τον αριθμό που του δίνουμε εμείς από το πληκτρολόγιο

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

main()
{
    int i = 0;
    int k;

    printf("input an integer:\n");
    i=scanf("%d", &k);

    printf("total values inputted %d\n", i);
    printf("The input values %d\n", k);

    system("PAUSE");     /* pause (wait for key to continue) */
}
```

Απλές μαθηματικές πράξεις

Οι απλές μαθηματικές πράξεις πραγματοποιούνται πάρα πολύ απλά στην γλώσσα C χρησιμοποιώντας τους παρακάτω τελεστές:

Τελεστής	**Πράξη**
+	Πρόσθεση
-	Αφαίρεση
*	Πολλαπλασιασμός
/	Διαίρεση (πηλίκο)
%	Υπόλοιπο διαίρεσης

Σημείωση: το πηλίκο είναι το ακέραιο πηλίκο εάν η πράξη γίνεται μόνο μεταξύ ακεραίων!

Η ιεράρχηση στις πράξεις είναι η ακόλουθη:

1. Παρενθέσεις
2. Πολλαπλασιασμός/Διαίρεση/Υπόλοιπο
3. Πρόσθεση/Αφαίρεση

Στις περιπτώσεις που οι πράξεις έχουν την ίδια ιεράρχηση, η προτεραιότητα είναι από αριστερά προς τα δεξιά.

Για να παρουσιάσουμε αριθμητικά δεδομένα με την printf() πρέπει να χρησιμοποιήσουμε οπωσδήποτε τους χαρακτήρες μετατροπής, οι οποίοι αρχίζουν πάντα με το σύμβολο %, για τα ορίσματα που αναγράφονται στο τέλος της εντολής και χωρίζονται μεταξύ τους με το κόμμα (,)

Χαρακτήρες μετατροπής	**Περιγραφή**
%d ή %i	Τύπος int.
%o	Τύπος int. οκταδική μορφή
%x	Τύπος int. 16-δική μορφή
%u	Τύπος unsigned int.
%c	Τύπος char. Εκτύπωση ενός χαρακτήρα
%s	Τύπος char. Εκτύπωση πολλών χαρακτήρων
%f	Τύπος float, double
%e	Τύπος float, double Επιστημονική μορφή
%g	Τύπος float, double Επιστημονική ή κοινή εκτύπωση ανάλογα με την ακρίβεια

Παράδειγμα 2-11:

Το παρακάτω παράδειγμα προσθέτει 2 ακέραιους αριθμούς και εκτυπώνει το αποτέλεσμα στην οθόνη χρησιμοποιώντας την εντολή printf

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

main()
{
     int a, b;
     int sum;
```

```
     a = 5;
     b = 11;
     sum = a + b;
     printf("the sum is ");
     printf("the sum is %d\n", sum);

     system("PAUSE");  /* pause (wait for key to continue) */
}
```

Παράδειγμα 2-12:

Το παρακάτω παράδειγμα παρουσιάζει διάφορες μορφοποιήσεις εκτύπωσης στην οθόνη όλων των αριθμών και χαρακτήρων με την εντολή printf

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

main()
{
     char a;
     int b;
     float c;

     a = 'A';
     b = 255;
     c = 3.14159;

     printf("the argument: %c is character. \n", a);
     printf("the number: %d is an integer.\n", b);
     printf("the integer: %d in octal form is: %o.\n", b, b);
     printf("the integer: %d in hexadecimal form is: %x.\n", b,
b);
     printf("the number: %f is a real number.\n", c);
     printf("the number: %f in scientific form is: %e.\n", c);

     system("PAUSE");  /* pause (wait for key to continue) */
}
```

Παράδειγμα 2-13:

Το παρακάτω παράδειγμα παρουσιάζει τον συνολικό αριθμό των χαρακτήρων που τυπώνουμε με την εντολή printf στην οθόνη μας

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

main()
{
    int i = 0;

    i = printf("abcde");
```

```
    printf("\ntotal characters printed %d\n",i);

    system("PAUSE");  /* pause (wait for key to continue) */
}
```

Μπορούμε να διαμορφώσουμε τη μορφή που παρουσιάζονται οι αριθμητικές μεταβλητές και όχι μόνο χρησιμοποιώντας την **τελεία** πριν τον κατάλληλο χαρακτήρα μετατροπής

Παράδειγμα 2-14:

Το παρακάτω παράδειγμα παρουσιάζει αριθμούς και κείμενο στην οθόνη μας χρησιμοποιώντας την κατάλληλη διαμόρφωση για το μέγεθος και τη μορφή των αριθμητικών και μη μεταβλητών

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

main()
{
   int i = 873;
   double f = 123.94536;
   char s[] = "Happy Birthday";

   printf("Using precision for integers\n");
   printf("\t%.4d\n\t%.9d\n\n", i, i);

   printf("Using precision for floating-point numbers\n");
   printf("\t%.3f\n\t%.3e\n\t%.3g\n\n", f, f, f);

   printf("Using precision for strings\n");
   printf("\t%.11s\n", s);
   printf("\t%.8s\n", s);

   system("PAUSE");  /* pause (wait for key to continue) */
}
```

Άσκηση 2.1

Γράψτε ένα πρόγραμμα σε γλώσσα C ώστε να τυπώνει στην οθόνη το παρακάτω σχήμα:

```
  *
 ***
*****
  *
  *
 ===

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

main()
{
  printf("  *  \n");
```

```
  printf(" *** \n");
  printf("*****\n");
  printf("  *  \n");
  printf("  *  \n");
  printf(" === \n");

  printf("Enallaktikh lysh\n");

  printf("  *  \n *** \n*****\n  *  \n  *  \n === \n");

  system("PAUSE");/* pause (wait for key to continue) */
}
```

3ο Μέρος – Παραστάσεις και Τελεστές

Εντολή καταχώρησης

Για να καταχωρήσουμε μία τιμή σε μια μεταβλητή χρησιμοποιούμε τον τελεστή καταχώρησης = . Έτσι η γενική σύνταξη της εντολής καταχώρησης είναι η εξής:

```
μεταβλητή = παράσταση;
```

Προσέξτε πως η παράσταση μπορεί να περιλαμβάνει όχι μόνο μία τιμή ή μια άλλη μεταβλητή αλλά μια ολόκληρη μαθηματική παράσταση, όπως η εξής:

```
int result;
result = 3 * 5 + 44 – 4;
```

στην παραπάνω περίπτωση αξιολογείται η τιμή της αριθμητικής παράστασης και σαν τιμή στη μεταβλητή result καταχωρείται η τιμή 55.

Πολλές φορές χρειάζεται να μεταβάλουμε την τιμή μιας μεταβλητής πολλαπλασιάζοντάς την με κάποιο συντελεστή ώστε να την τριπλασιάσουμε για παράδειγμα. Σε αυτές τις περιπτώσεις αντί να πολλαπλασιάσουμε τη τιμή της μεταβλητής με το συντελεστή 3 και να καταχωρήσουμε το νέο αποτέλεσμα στην τιμή της μεταβλητής μπορούμε να χρησιμοποιήσουμε τη σύνθετη μορφή της εντολής καταχώρησης που είναι η εξής:

```
μεταβλητή τελεστής= παράσταση;
```

Έτσι για να τριπλασιάσουμε τη τιμή μιας μεταβλητής αρκεί να γράψουμε:

`result *= 3;` και όχι `result = result * 3;`

Πολλές φορές εάν θέλουμε να προσαυξήσουμε την τιμή μιας μεταβλητής κατά κάποια ποσότητα αντί να γράφουμε ολόκληρη την πράξη «κολλάμε» την πράξη που θέλουμε να κάνουμε μαζί με το =, όπως για παράδειγμα

στην πράξη `sum += i;` που μεταφράζεται ως εξής: `sum = sum + i;`

Με παρόμοιο τρόπο μπορούμε να γράψουμε και άλλες εκφράσεις, κυρίως για τον υπολογισμό απλών μαθηματικών πράξεων, όπως: **`-=, +=, *=, /=, και %=`**. Σημειώστε πως για τη σύνθετη εντολή **`%=`** θα πρέπει να έχουμε ακέραιους τύπους δεδομένων.

Τελεστές αύξησης/μείωσης κατά 1

Η γλώσσα προγραμματισμού C μας παρέχει διάφορες συντομογραφίες όταν πρόκειται να μεταβάλουμε την τιμή μιας μεταβλητής κατά 1 όπως είναι οι:
`i++` είναι το ίδιο με **`i = i + 1`**, δηλαδή η ποσότητα του i θα αυξηθεί κατά 1
`i--` είναι το ίδιο με **`i = i - 1`**, δηλαδή η ποσότητα του i θα μειωθεί κατά 1
Προσοχή!: άλλο `i++` και άλλο `++i` !!!

Γράφοντας i++ πρώτα χρησιμοποιούμε την τρέχουσα τιμή του i και έπειτα θα γίνει η αύξηση κατά 1. Αντίθετα γράφοντας ++i πρώτα θα γίνει η αύξηση κατά 1 και έπειτα θα είναι διαθέσιμη η νέα τιμή του i. Και στις δύο περιπτώσεις όμως η τελική τιμή του i θα αυξηθεί κατά 1 μετά την εκτέλεση της εντολής.

Παράδειγμα 3-1:

Τι αποτέλεσμα θα βγάλει το παρακάτω πρόγραμμα και γιατί; Προσπαθήσετε να υπολογίσετε τις τιμές των i και j που θα χρησιμοποιηθούν στον πολλαπλασιασμό.

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

main()
{
   int i = 4;
   int j = 4;
   int result;

   result= i++ * ++j;
   printf("%d \n", result);

   system("PAUSE"); /* pause */
}
```

Το παραπάνω πρόγραμμα θα μας εκτυπώσει την τιμή 20 διότι η πράξη που θα γίνει θα είναι η εξής:
result = 4++ * ++4 = 4 * 5 = 20

Σχεσιακοί Τελεστές

Οι σχεσιακοί τελεστές χρησιμοποιούνται στον υπολογισμό συνθηκών αληθείας (Boolean), δηλαδή συνθηκών που **η τιμή τους είναι ή Αληθής (TRUE ή 1) ή Ψευδής (FALSE ή 0).** Οι σχεσιακοί τελεστές συγκρίνουν τις σχετικές τιμές ακέραιων ή δεκαδικών τελεστών. Οι σχεσιακοί τελεστές της C είναι οι ακόλουθοι:

Σχεσιακός Τελεστής	Περιγραφή
>	Μεγαλύτερο από
>=	Μεγαλύτερο ή ίσο από
<	Μικρότερο από
<=	Μικρότερο ή ίσο από

= =	Ίσον με
!=	Διάφορο από

Παραδείγματα Σχεσιακών Τελεστών

Συνθήκη	**Τιμή**
5 < 1	Ψευδής
5 >= 5	Αληθής
3 = = 3	Αληθής
10 <= 20	Αληθής
1 != 1	Ψευδής
1.5 > 1.1	Αληθής

Λογικοί Τελεστές

Οι Λογικοί τελεστές χρησιμοποιούνται στον υπολογισμό παραστάσεων αληθείας (Boolean), δηλαδή παραστάσεων που **η τιμή τους υπολογίζεται σε Αληθής (TRUE) ή Ψευδής (FALSE).** Οι Λογικοί τελεστές συγκρίνουν τις σχετικές τιμές παραστάσεων που μπορεί να περιέχουν σχεσιακούς τελεστές. Οι λογικοί τελεστές της C είναι οι ακόλουθοι:

Σχεσιακός Τελεστής	**Περιγραφή**
&&	Λογικό ΚΑΙ (AND)
||	Λογικό Η (OR)

Το Λογικό ΚΑΙ: && επιστρέφει την τιμή TRUE μόνον εάν και οι δύο εκφράσεις (πριν και μετά το &&) έχουν την τιμή TRUE. Σε οποιαδήποτε άλλη περίπτωση επιστρέφει την τιμή FALSE.

Το Λογικό Η: || επιστρέφει την τιμή TRUE εάν τουλάχιστον μία από τις δύο εκφράσεις (πριν και μετά το &&) έχουν την τιμή TRUE. Επιστρέφει την τιμή FALSE μόνον εάν και οι δύο εκφράσεις έχουν την τιμή FALSE.

Παραδείγματα Σχεσιακών Τελεστών

Συνθήκη	**Τιμή**
(5 < 1) && (5 >= 5)	Ψευδής && Αληθής = Ψευδής
(5 >= 5) || (3 = = 3)	Αληθής || Αληθής = Αληθής
(3 = = 3) && (1.5 > 1.1)	Αληθής && Αληθής = Αληθής
(10 <= 20) || (1 != 1)	Αληθής || Ψευδής = Αληθής

Προτεραιότητα σχέσεων και πράξεων

Η ιεράρχηση στους τελεστές είναι η ακόλουθη:

1. Παρενθέσεις
2. * , / , %
3. + , -
4. > , >= , < , <=
5. = = , !=

Στις περιπτώσεις που οι πράξεις έχουν την ίδια ιεράρχηση, η προτεραιότητα είναι από αριστερά προς τα δεξιά.

4ο Μέρος – Εντολές IF – IF ELSE - SWITCH

Σύνθετες εντολές

Μπορούμε να ομαδοποιήσουμε μια σειρά από εντολές ώστε αυτές να αντιμετωπίζονται ως ένα «μπλοκ» εντολών (δηλαδή σαν μια εντολή). Η ομαδοποίηση αυτή πραγματοποιείται με τη χρήση των αγκίστρων **{ }** πριν και μετά το σύνολο των εντολών που θέλουμε να ομαδοποιήσουμε.

Παράδειγμα ομαδοποίησης εντολών:

```
{
     printf("a = ");
     scanf("%d",&a);
     printf("b = ");
     scanf("%d",&b);
     sum = a + b;
     printf("sum = %d", sum);
}
```

Εάν έχουμε μόνο μία εντολή τότε συνήθως παραλείπουμε τα άγκιστρα και η εντολή γράφεται απλά.

Εντολές ελέγχου ροής

Οι εντολές ελέγχου ροής δίνουν τη δυνατότητα να έχουμε επιλεκτική εκτέλεση εντολών σύμφωνα με την τιμή μιας λογικής συνθήκης. Οι εντολές ελέγχου είναι οι if-else και switch.

Εντολή ελέγχου ροής if

Η εντολή ελέγχου ροής if συντάσσεται ως εξής:

```
if (expr)
  εντολή;
```

Η εντολή θα εκτελεστεί μόνον εάν η λογική συνθήκη expr έχει την τιμή TRUE **αλλιώς ολόκληρη η εντολή θα παραληφθεί**. Η εντολή όμως μπορεί να είναι μια σύνθετη εντολή οπότε η παραπάνω σύνταξη προσαρμόζεται ως εξής:

```
if (expr)
{
    εντολή1;
    εντολή2;
    ....
}
```

Παράδειγμα 4-1:

Το παρακάτω πρόγραμμα εκτελεί την εντολή εξόδου printf μόνο εάν η λογική συνθήκη (i > 0) έχει την τιμή TRUE (Αληθής), διαφορετικά δεν τυπώνει τίποτα.

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

```
#include <stdlib.h>

main()
{
  int i = 5;

  if (i > 0)
    printf(" i > 0. \n");

  system("PAUSE");  /* pause (wait for key to continue) */
}
```

Παράδειγμα 4-2:

Το παρακάτω πρόγραμμα εκτελεί την εντολή εξόδου printf μόνο εάν η λογική συνθήκη (i > 0) έχει την τιμή TRUE (Αληθής), ανάλογα με το τι αριθμό θα δώσουμε από το πληκτρολόγιο με την scanf

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

main()
{
  int i;

  printf(" Enter value of integer i: ");
  scanf("%d",&i);

  if (i > 0)
     printf(" i > 0. \n");

  system("PAUSE");  /* pause (wait for key to continue) */
}
```

Η εντολή ελέγχου if μπορεί να χρησιμοποιηθεί συνδυαστικά για παραπάνω από έναν έλεγχο στη ροή του προγράμματός μας.

Παράδειγμα 4-3:

Το παρακάτω πρόγραμμα διαβάζει έναν ακέραιο αριθμό και ελέγχει πρώτα με την if εάν ο αριθμός είναι μεγαλύτερος του 3 και έπειτα με την δεύτερη if εάν είναι μικρότερος του 6 και μας τυπώνει τα ανάλογα μηνύματα με την εντολή printf.

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

main()
{
  int number = 0;
  printf("\nEnter an integer between 1 and 10: ");
  scanf("%d",&number);

  if (number > 3)
```

```
    printf("You entered %d which is greater than 3\n", number);

  if (number < 6)
    printf("You entered %d which is less than 6\n", number);

  system("PAUSE");  /* pause (wait for key to continue) */
}
```

Στο παραπάνω παράδειγμα οι δύο έλεγχοι, number > 3 και number < 6, θα αξιολογηθούν και οι δύο ανεξάρτητα ο ένας από τον άλλο. Αυτό είναι ενδεικτικό κακής απόδοσης του κώδικά μας, έστω και εάν τα αποτελέσματα είναι τα αναμενόμενα. Μπορούμε να βελτιστοποιήσουν την ταχύτητα του προγράμματός μας εάν τοποθετήσουμε τον δεύτερο έλεγχο μέσα σε μια σύνθετη εντολή που να αντιστοιχεί στον πρώτο έλεγχο. Με αυτό τον τρόπο ο έλεγχο number < 6 θα πραγματοποιείται μόνο στην περίπτωση που ο αριθμός είναι μεγαλύτερος του 3.

Παράδειγμα 4-4:

Το παρακάτω πρόγραμμα διαβάζει έναν ακέραιο αριθμό και ελέγχει πρώτα με την if εάν ο αριθμός είναι μεγαλύτερος του 3 και έπειτα και μόνο στην περίπτωση που ο ακέραιος αριθμός μας είναι μεγαλύτερος του 3 θα ελεγχθεί πάλι με την if εάν είναι μικρότερος του 6 και θα τυπωθούν τα ανάλογα μηνύματα με την εντολή printf.

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

main()
{
  int number = 0;
  printf("\nEnter an integer between 1 and 10: ");
  scanf("%d",&number);

  if (number > 3)
  {
    printf("You entered %d which is greater than 3\n", number);

    if (number < 6)
      printf("You entered %d which is less than 6\n", number);
  }

  system("PAUSE");  /* pause (wait for key to continue) */
}
```

Παρατηρήστε πως τα δύο προγράμματα θα συμπεριφερθούν διαφορετικά εάν ο ακέραιος αριθμός μας είναι αρνητικός (δοκιμάστε με τιμή -1). Στο παράδειγμα ex-4-4.c δεν θα τυπωθεί απολύτως τίποτα.

Παράδειγμα 4-5:

Γράψτε ένα πρόγραμμα σε γλώσσα C ώστε να διαβάζει 2 δεκαδικούς αριθμούς και στη συνέχεια να ελέγξει εάν ο πρώτος αριθμός είναι μεγαλύτερος του 10 και εάν ο δεύτερος είναι διάφορος του

μηδενός. Εάν ο έλεγχος είναι αληθής τότε να υπολογίσει το πηλίκο: $\frac{πρώτος}{δεύτερος}$*. Το αποτέλεσμα να τυπωθεί στην οθόνη εμφανίζοντας 2 δεκαδικά ψηφία*

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

main()
{
   float number1 = 0;
   float number2 = 0;
   float quotient = 0;

   printf("Enter the first number: ");
   scanf("%f",&number1);
   printf("Enter the first number: ");
   scanf("%f",&number2);

   if ((number1 > 10) && (number2 != 0))
   {
      quotient = number1/number2;
      printf(" The quotient of %.2f and %.2f is %.2f\n", number1,
number2, quotient);
   }

  system("PAUSE");  /* pause (wait for key to continue) */
}
```

Παράδειγμα 4-6:

Το παρακάτω πρόγραμμα διαβάζει έναν ακέραιο αριθμό και ελέγχει με την πρώτη εντολή if, πρώτα εάν ο αριθμός είναι μεγαλύτερος του 3 και έπειτα μόνο στην περίπτωση που ο ακέραιος αριθμός μας είναι μεγαλύτερος του 3 θα ελεγχθεί με τη δεύτερη εντολή if εάν είναι μικρότερος του 6 και θα τυπωθούν τα ανάλογα μηνύματα.

```
#include <stdio.h>
#include <stdlib.h>
main()
{
  int number = 0;
  printf("\nEnter an integer between 1 and 10: ");
  scanf("%d",&number);

  if (number > 3)
  {
    printf("You entered %d which is greater than 3\n", number);

    if (number < 6)
      printf("You entered %d which is less than 6\n", number);
  }

  system("PAUSE");  /* pause */
```

```
}
```

Εντολή ελέγχου if else

Η εντολή ελέγχου if else συντάσσεται ως εξής:

```
if (expr)
{
  εντολή1;
}
else
{
  εντολή2;
}
```

Η εντολή1 θα εκτελεστεί μόνον εάν η λογική συνθήκη expr έχει την τιμή TRUE αλλιώς θα εκτελεστεί η εντολή2. Σε κάθε περίπτωση βλέπουμε πως είτε ή μία εντολή είτε η άλλη θα εκτελεστεί. Η εντολή όμως μπορεί να είναι μια σύνθετη εντολή οπότε η παραπάνω σύνταξη προσαρμόζεται ως εξής:

```
if (expr)
{
  εντολή1;
  εντολή2;
  ....
}
else
{
  εντολή3;
  εντολή4;
  ....
}
```

Παράδειγμα 4-7:

Το παρακάτω πρόγραμμα εκτελεί την αντίστοιχη εντολή εξόδου printf ανάλογα με το εάν η λογική συνθήκη (i > 0) έχει την τιμή TRUE (Αληθής) ή FALSE (Ψευδής)

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

main()
{
  int i = -5;

  if (i > 0)
  {
     printf(" o i thetikos. \n");
  }
  else
  {
     printf("o i oxi thetikos. \n");
  }
```

```
  system("PAUSE");  /* pause */
}
```

Ιδιαίτερη προσοχή πρέπει να δώσουμε στις συνθήκες ελέγχου της if και κατά συνέπεια ποιές περιπτώσεις αντιστοιχούν στον κλάδο else της εντολής if. Οι εντολές if – else μπορούν να συνδυαστούν μεταξύ τους με τρόπο ώστε να έχουμε συνεχόμενους ελέγχους. Σε αυτή την περίπτωση αρκεί να θυμόμαστε τον απλό κανόνα: **κάθε else αντιστοιχεί στο αμέσως προηγούμενο if που δεν έχει else.**

Παράδειγμα 4-8:

Το παρακάτω πρόγραμμα ελέγχει την συνθήκη πρώτα i = =0, εάν η συνθήκη είναι αληθής τότε θα εκτυπωθεί το μήνυμα i = = 0. και το πρόγραμμα θα τερματίσει τη λειτουργία του. Εάν η συνθήκη (i = = 0) είναι FALSE (Ψευδής) τότε και μόνον τότε θα γίνει ο έλεγχος (i = = 1) και εάν και αυτός ο έλεγχος είναι FALSE (Ψευδής) τότε και μόνον τότε θα γίνει ο έλεγχος (i = = 2).

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

main()
{
  int i = 2;

  if (i == 0)
  {
     printf(" i == 0. \n");
  } else if (i == 1)
  {
     printf(" i == 1. \n");
  } else if (i == 2)
  {
     printf(" i == 2. \n");
  }

  system("PAUSE");  /* pause */
}
```

Χρησιμοποιήστε πάντοτε κανόνες καλής γραφής (με κενές γραμμές και εσοχές) όταν έχετε συνεχόμενες if – else ώστε το πρόγραμμα να μην είναι δυσανάγνωστο. Ο έλεγχος της λογικής συνθήκης μπορεί να είναι πιο σύνθετος και να περιλαμβάνει και τους σχεσιακούς τελεστές && (Λογικό AND) και || (Λογικό OR).

Παράδειγμα 4-9:

Το παρακάτω πρόγραμμα ζητάει να πληκτρολογήσουμε ένα θετικό δεκαδικό αριθμό από το ένα μέχρι και το 10. Στη συνέχεια ελέγχει μέσω της if εάν έχουμε πληκτρολογήσει τον σωστό αριθμό και μας ενημερώνει ανάλογα. Παρατηρήστε πως αφού σε κάθε περίπτωση θα εκτελεστεί μία και μόνη εντολή δεν χρειάζεται να προσθέσουμε άγκιστρα στην εντολή if else.

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

```
main()
{
  float num = 0;

  printf("Enter the number: ");
  scanf("%f", &num);

  if ( num >= 0 && num <= 10 )
    printf("\nCorrect number\n");
  else
    printf("\nIncorrect number\n");

  system("PAUSE");  /* pause */
}
```

Ο έλεγχος της λογικής συνθήκης μπορεί να είναι πιο σύνθετος και να περιλαμβάνει και τους σχεσιακούς τελεστές && (Λογικό AND) και || (Λογικό OR).

Παράδειγμα 4-10:

Το παρακάτω πρόγραμμα ζητάει να πληκτρολογήσουμε το χαρακτήρα Α. Στη συνέχεια ελέγχει μέσω της εντολής if εάν έχουμε πληκτρολογήσει είτε το κεφαλαίο άλφα είτε το μικρό άλφα και στη συνέχεια μας ενημερώνει ανάλογα. Παρατηρήστε πως αφού σε κάθε περίπτωση θα εκτελεστεί μία και μόνη εντολή δεν χρειάζεται να προσθέσουμε άγκιστρα στην εντολή if - else.

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

main()
{
  char cResponse = '';

  printf("Enter the letter A: ");
  scanf("%c", &cResponse);

  if ( cResponse== 'A' || cResponse == 'a' )
    printf("\nCorrect response\n");
  else
    printf("\nIncorrect response\n");
  system("PAUSE");  /* pause */
}
```

Παράδειγμα 4-11:

Τροποποιήστε το παραπάνω παράδειγμα ώστε να ζητείται ένας αριθμός ανάμεσα στο 1 και το 10, να γίνεται ο έλεγχος ορθότητας και στ συνέχεια να τυπώνεται το ανάλογα μήνυμα

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

main()
```

```
{
  int iResponse = 0;

  printf("Enter a number from 1 to 10: ");
  scanf("%d", &iResponse);

  if ( iResponse < 1 || iResponse > 10 )
    printf("\nNumber not in range\n");
  else
    printf("\nThank you\n");

  system("PAUSE");  /* pause */
}
```

Καμιά φορά η χρήση πολλών και συνεχόμενων if εντολών μπορεί να οδηγήσει σε τελείως άσκοπους ελέγχους και κατά συνέπεια σε ενδεχόμενες καθυστερήσεις εξαιτίας των περιττών ενεργειών.

Παράδειγμα 4-12:

Το παρακάτω παράδειγμα ελέγχει εάν ένας οδηγός είναι άνω των 21 ετών και εάν έχει δίπλωμα οδήγησης πάνω από 3 χρόνια και ανάλογα την περίπτωση υπολογίζει το κατάλληλο ασφάλιστρο.
Οι προσαυξήσεις που υποθέτουμε είναι οι εξής:

- *Ηλικία κάτω των 21 ετών → +10%*
- *Έτη εμπειρίας 1 → +10%,*
- *Έτη εμπειρίας 2 → +5%*
- *Έτη εμπειρίας 3 → +2,5%*

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

main()
{
  char age;
  int license;
  float price = 160;
  float multi = 0.0;

  printf(" Age < 21: (y/n) ");
     scanf("%c",&age);
  printf(" Enter years of license: ");
     scanf("%d",&license);

  if (age == 'y' || age == 'Y')
    multi = multi + 0.1;
  if (license == 1)
    multi = multi + 0.1;
  if (license == 2)
    multi = multi + 0.05;
  if (license == 3)
    multi = multi + 0.025;
```

```
  price = price + (price * multi);
  printf("\n for age < 21 %c with %d years of license, the price
is %.2f \n", age, license, price);

  system("PAUSE");  /* pause */
}
```

Το παραπάνω παράδειγμα αναγκαστικά υπολογίζει το αποτέλεσμα της σύγκρισης **`license == 1, 2 ή 3`** σε κάθε εκτέλεση του προγράμματος με αποτέλεσμα άσκοπες καθυστερήσεις. Η χρήση if – else εντολών θα μπορούσε να λύσει σε κάποιες περιπτώσεις το πρόβλημα αλλά τότε ο κώδικας θα γίνει αρκετά δυσανάγνωστος. Μπορούμε να αποφύγουμε τέτοιου είδους καθυστερήσεις και προβλήματα χρησιμοποιώντας την εντολή επανάληψης switch.

Εντολή ελέγχου ροής Switch

Η εντολή ελέγχου ροής switch συντάσσεται ως εξής:

```
Switch (switch_expr)
{
   case constant_expr1 :  S1;
                          S2;
                          break;
   case constant_expr1 :  S3;
                          S4;
                          break;
   .....
   default             :  S5;
                          S6;
                          break;
}
```

Οι τιμές **`s1, s2`** κλπ. πρέπει να είναι οπωσδήποτε σταθερές τιμές δηλαδή αριθμοί ή χαρακτήρες. Αριθμητικές παραστάσεις ή/και συνδυασμοί με μεταβλητές (π.χ. temp*2) δεν είναι αποδεκτοί σαν μια από τις τιμές της εντολής switch. Ο όρος **`'default'`** είναι προαιρετικός και περιλαμβάνει όλες τις υπόλοιπες δυνατές περιπτώσεις της έκφρασης ελέγχου της switch.

<u>Παράδειγμα 4-13:</u>

Το παρακάτω παράδειγμα ελέγχει για 10 ακέραιους αριθμούς εάν ο κάθε αριθμός είναι άρτιος ή περιττός χρησιμοποιώντας την εντολή switch και τυπώνει το ανάλογο μήνυμα.

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

main()
{
  int i, n = 10;

  for (i = 1; i < n; i = i + 1)
  {
    switch (i%2)
    {
```

```
      case 0 : printf("o arithmos %d einai artios \n",i);
               break;
      case 1 : printf("o arithmos %d einai perittos \n",i);
               break;
    } /* end switch */
  }
  system("PAUSE");  /* pause */
}
```

Η εντολή break είναι απαραίτητη μετά από κάθε τιμή case στην εντολή switch. Εάν αφαιρέσουμε την εντολή break από την περίπτωση case 0 στο παραπάνω παράδειγμα, όταν το εκτελέσουμε θα δούμε πως θα σε κάθε κύκλο της εντολής for θα εκτελούνται και οι εντολές της περίπτωσης case 0 αλλά και της περίπτωσης case 1.

Παράδειγμα 4-14:

Το παρακάτω παράδειγμα λύνεται χρησιμοποιώντας την εντολή switch αντί των εντολών if – else που χρησιμοποιήσαμε στο παράδειγμα 4-12.

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

main()
{
  char age;
  int license;
  float price = 160;
  float multi = 0.0;

  printf(" Age < 21: (y/n) ");
  scanf("%c",&age);
  printf(" Enter years of license: ");
  scanf("%d",&license);

  if (age == 'y' || age == 'Y')
    multi = multi + 0.1;

  switch (license)
  {
    case 1 : multi = multi + 0.1;
             break;
    case 2 : multi = multi + 0.05;
             break;
    case 3 : multi = multi + 0.025;
             break;
  }

  price = price + (price * multi);
  printf("\n for age < 21 %c with %d years of license, the price
is %.2f \n", age, license, price);

  system("PAUSE");  /* pause */
}
```

Παράδειγμα 4-15:

Το παρακάτω παράδειγμα χρησιμοποιεί την εντολή switch για να συγκρίνει έναν αριθμό που γράφεται από το χρήστη με κάποιες προκαθορισμένες τιμές.

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

main()
{
  int choice;

  printf("Enter your choice number (1-10):");
  scanf("%d",&choice);

  if ((choice > 10) || (choice < 1))
    choice = 11;

  switch(choice)
  {
    case 7: printf("\aCongratulations! you win!\n");
            break;

    case 2: printf("You win a free try!\n");
            break;

    case 8: printf("You win the second prize!\n");
            break;

    case 11: printf("\a \a \a error!\n");
             break;

    default: printf("\nSorry, you lose.\n");
             break;
     }

  system("PAUSE");  /* pause */
}
```

5ο Μέρος – Εντολές Επανάληψης: FOR - WHILE

Περιγραφή

Χρησιμοποιούμε την εντολή επανάληψης for όταν γνωρίζουμε εκ των προτέρων πόσες αλλά και ποιες φορές χρειάζεται να επαναληφθεί μια συγκεκριμένη ενέργεια ή διαδικασία.

Σύνταξη for

```
for (εκφρ1; εκφρ2; εκφρ3)
{
```

```
  εντολή1;
  εντολή2;
}
```

Η παράσταση **εκφρ1** θα εκτελεστεί μία και μόνο φορά πριν την εκτέλεση της εντολή for. Η παράσταση **εκφρ2** είναι μια παράσταση αληθείας, εάν δεν υπάρχει λαμβάνεται η τιμή TRUE. Εάν η παράσταση **εκφρ2** λάβει την τιμή FALSE η εντολή επανάληψης for τερματίζεται. Μετά τις εντολές που έχουμε δηλώσει για επανάληψη μέσα στην for θα εκτελεστεί η παράσταση **εκφρ3.**

Βασικό χαρακτηριστικό της εντολής επανάληψης for είναι πως μπορεί να εκτελεστεί πολλές ή και καμία φορές, εάν ο έλεγχος της **εκφρ2** λάβει την τιμή FALSE εξ'αρχής.

Παράδειγμα 5-1:

Το παρακάτω πρόγραμμα εκτελεί την εντολή εξόδου printf για 5 φορές χρησιμοποιώντας την εντολή επανάληψης for. Προσέξτε ότι ο δείκτης i στην for παίρνει τιμές από 1 έως και 5.

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

main()
{
    int i;

    for (i = 1; i <= 5; i++)
    {
       printf("the numbers are %d \n",i);
    }

    system("PAUSE"); /* pause */
}
```

Προσοχή!: άλλο i++ και άλλο ++i !!! όπως έχουμε δει σε προηγούμενα μαθήματα

Συνήθως η εντολή επανάληψης for χρησιμοποιείται μαζί με μεταβλητές-μετρητές οι οποίες χρησιμοποιούνται για τον υπολογισμό διαφόρων ενεργειών αλλά και στις παραστάσεις αληθείας στην σύνταξη της εντολής for. Στην ANSI C που χρησιμοποιούμε εμείς θα πρέπει υποχρεωτικά να έχουμε δηλώσει από πριν αυτές τις μεταβλητές για να γίνουν οι απαραίτητες πράξεις και συγκρίσεις.

Όλες οι παραστάσεις **εκφρ1, εκφρ2, εκφρ3** στην εντολή επανάληψης for μπορούν να παραληφθούν.

Παράδειγμα 5-2:

Το παρακάτω πρόγραμμα εκτελεί την εντολή εξόδου printf έως ότου η μεταβλητή count ξεπεράσει το 10. Προσέξτε πως η εντολή επανάληψης for δεν έχει καθόλου παράσταση στο πρώτο κομμάτι της.

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

main()
```

```
{
  int count = 1;

  for (   ; count <= 10 ; count++)
  {
    printf("\n%d", count);
  }
  printf("\nWe have finished.\n");

  printf("the count is: %d", count);

  system("PAUSE"); /* pause */
}
```

Παρατηρούμε πως στο παραπάνω παράδειγμα η μεταβλητή count υπάρχει και μετά την εντολή επανάληψης for και μάλιστα πως η count διατηρεί την τελική της τιμή.

Προσοχή: Δεν μπορούμε να δηλώσουμε τις μεταβλητές μαζί με τη δήλωση της for.

```
  for (int count = 1 ; count <= 8 ; count++)
    printf("\n*            *");
```

Το παραπάνω κομμάτι κώδικα δεν είναι αποδεκτό στην γλώσσα ANSI C (είναι όμως αποδεκτό σε άλλες γλώσσες προγραμματισμού όπως οι Java, C# κλπ.)

Η εντολή for μπορεί να χρησιμοποιηθεί και με μετρητές-μεταβλητές τύπου char.

Παράδειγμα 5-3:

Το παρακάτω πρόγραμμα μας παρουσιάζει την αντιστοιχία ascii και char για όλους τους χαρακτήρες από Α έως και Ρ. Παρατηρούμε πως τελικά και ο τύπος char είναι ένας μικρός ακέραιος αριθμός.

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

main()
{
  char b;

  for (b = 'A'; b <= 'P'; b++)
  {
    printf("%d - %c \n", b, b);
  }

  system("PAUSE"); /* pause */
}
```

Παράδειγμα 5-4:

Το παρακάτω πρόγραμμα μας ρωτά έναν ακέραιο αριθμό και στη συνέχεια υπολογίζει το άθροισμα για όλους τους ακέραιους αριθμούς από το 1 έως και τον αριθμό που του δώσαμε, χρησιμοποιώντας την εντολή for.

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

main()
{
  int sum = 0;
  int count = 0, i;

  printf("\nEnter the number of integers you want to sum: ");
  scanf("%d", &count);

  for (i = 1; i<= count; i++)
  {
    sum = sum + i;
  }

  printf("\nTotal of the first %d numbers is %d\n", count, sum);

  system("PAUSE"); /* pause */
}
```

Παράδειγμα 5-5:

Το παρακάτω πρόγραμμα εκτελείται χρησιμοποιώντας σύνθετες παραστάσεις με 2 όρους που διαχωρίζονται μεταξύ τους με κόμμα. Η printf θα εκτυπωθεί όσο διάστημα έχουμε `i < 5` και `j > 5`

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

main()
{
    int i, j;

    for (i = 0, j = 10; i < 5 && j > 5; i++, j--)
    {
       printf (" the value of i and j %d %d\n", i, j);
    }

  system("PAUSE"); /* pause */
}
```

Παράδειγμα 5-6:

Το παρακάτω παράδειγμα χρησιμοποιεί την εντολή επανάληψης for ώστε να διαβάσει 10 ακέραιους αριθμούς από το πληκτρολόγιο και μόνο για τους περιττούς (μονούς) υπολογίζει το άθροισμα και το πλήθος τους.

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

```
main()
{
  int num = 0, sum = 0, plithos = 0, i =0;

  for ( i = 1;  i <= 10; i++ )
  {
      printf(" dose ton arithmo: ");
      scanf("%d", &num);

      if ( num % 2 == 1)
      {
         sum = sum + num;
         plithos++;
      }
  }

  printf(" athroisma = %d \n", sum);
  printf(" plithos = %d \n", plithos);

  system("PAUSE");  /* pause */
}
```

Περιγραφή

Χρησιμοποιούμε την εντολή επανάληψης while όταν δεν γνωρίζουμε εκ των προτέρων πόσες ακριβώς φορές χρειάζεται να επαναληφθεί μια συγκεκριμένη ενέργεια (δηλαδή σε αντίθεση με την εντολή επανάληψης for). Αφού δεν γνωρίζουμε τον ακριβή αριθμό των επαναλήψεων χρησιμοποιούμε μια συνθήκη ελέγχου η οποία μας επιτρέπει να ξεκινήσουμε, να συνεχίσουμε ή να διακόψουμε τις επαναλήψεις μας.

Σύνταξη while

```
while (συνθήκη)
{
  εντολή1;
  εντολή2;
  .....
}
```

Η εντολή επανάληψης while μπορεί να εκτελεστεί πολλές (όσο η συνθήκη ελέγχου έχει την τιμή TRUE (Αληθής)) ή και καμία φορά (στην περίπτωση που η συνθήκη ελέγχου λάβει την τιμή FALSE (Ψευδής) στην αρχή).

Παράδειγμα 5-7:

Το παρακάτω πρόγραμμα εκτελεί την εντολή εξόδου printf για 5 φορές χρησιμοποιώντας την εντολή επανάληψης while. Προσέξτε πως η αύξηση του μετρητή i πραγματοποιείται μετά την εντολή printf.

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

```
main()
{
    int i = 1;

    while ( i <= 5 )
    {
        printf(" the value of i is %d\n", i);
        i = i + 1;
    }

    system("PAUSE"); /* pause */
}
```

Παράδειγμα 5-8:

Το παρακάτω πρόγραμμα αθροίζει όλους τους ακέραιους αριθμούς από το 1 έως τον αριθμό που θα του δώσουμε εμείς. Το πρόγραμμα αυτό χρησιμοποιεί την εντολή επανάληψης while και πραγματοποιεί σε κάθε βήμα την αύξηση του αθροίσματος και την αύξηση του μετρητή i σε μία και μόνο γραμμή.

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

main()
{
  int sum = 0, i = 1, count = 0;

  printf("\nEnter the number of integers you want to sum: ");
  scanf(" %d", &count);

  while(i <= count)
  {
    sum += i++;
  }

  printf("Total of the first %d numbers is %d\n", count, sum);

  system("PAUSE"); /* pause */
}
```

Παράδειγμα 5-9:

Το παρακάτω πρόγραμμα χρησιμοποιεί την εντολή while για να παρουσιάσει την ακολουθία fibonacci που είναι μικρότεροι από το 500 χρησιμοποιώντας την εντολή επανάληψης while.

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

main()
{
    int oldnum = 1, curnum = 1, nextnum = 0;
```

```
    printf("%d ", curnum);

    while (curnum < 500)
    {
      printf("%d ", curnum);
      nextnum = curnum + oldnum;

      oldnum = curnum;
      curnum = nextnum;
    }

    system("PAUSE"); /* pause */
}
```

Παράδειγμα 5-10:

Το παρακάτω πρόγραμμα χρησιμοποιεί την εντολή επανάληψης while για να διαβάζει συνεχώς θετικούς ακέραιους αριθμούς από το πληκτρολόγιο και υπολογίζει το πλήθος και το άθροισμά τους. Η διαδικασία αυτή διακόπτεται εάν δοθεί μηδέν ή αρνητικός αριθμός

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

main()
{
  int num = 0, sum = 0, plithos = 0;

  printf("dose ton akeraio: ");
  scanf("%d", &num);

  while ( num > 0 )
  {
     sum = sum + num;
     plithos++;

     printf("dose ton akeraio: ");
     scanf("%d", &num);
  }

  printf("edoses %d arithmous\n", plithos);
  printf("to athroisma einai: %d \n", sum);

  system("PAUSE");  /* pause */
}
```

Οι εντολές επανάληψης for και while μπορούν πολύ εύκολα να αντικαταστήσουν η μία την άλλη ακολουθώντας τον παρακάτω συσχετισμό. Η έκφραση A στην αρχή της εντολής for γράφεται πριν την εντολή while. Η έκφραση B της εντολής for γράφεται ως συνθήκη ελέγχους στην εντολή while. Η έκφραση C της εντολής for γράφεται μέσα στο σώμα της εντολής while ώστε να μεταβληθούν οι τιμές στις υπόλοιπες εκφράσεις για τον επανέλεγχο της while μετά την εκτέλεση του εκάστοτε κύκλου. Σχηματικά μπορούμε να δούμε αυτό τον συσχετισμό μεταξύ των εντολών επανάληψης for και while στον παρακάτω ψευδοκώδικα:

```
for (A;B;C)
{
   printf("%d\t", i);
}
```

Η παραπάνω for μετασχηματίζεται στην παρακάτω while εντολή:

```
A;
while(B)
{
  printf("%d\t", i);
  C;
}
```

Παράδειγμα 5-11:

Το παρακάτω πρόγραμμα παρουσιάζει αριθμούς στην οθόνη έως ότου ικανοποιηθούν οι συνθήκες που έχουμε ορίσει.

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

main()
{
    int i, j;

    for (i = 0, j = 10; i < 5 && j > 5; i++, j--)
    {
       printf (" the value of i and j %d %d\n",i, j);
    }
    system("PAUSE"); /* pause */
}
```

Το παραπάνω πρόγραμμα με την εντολή for μετατρέπεται πολύ εύκολα στο παρακάτω πρόγραμμα που χρησιμοποιεί την εντολή while.

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

main()
{
    int i, j;
    i = 0;
    j = 10;

    while ( i < 5 && j > 5)
    {
       printf (" the value of i and j %d %d\n",i, j);
       i++;
       j--;
    }
    system("PAUSE"); /* pause */
}
```

Παράδειγμα 5-12:

Το παρακάτω πρόγραμμα αθροίζει όλους τους ακέραιους αριθμούς από το 1 έως τον αριθμό που θα του δώσουμε εμείς.

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

main()
{
  long sum = 0;
  int i = 1;
  int count = 0;

  printf("\nEnter the number of integers you want to sum: ");
  scanf(" %d", &count);

  while(i <= count)
  {
    sum += i++;
  }

  printf("Total of the first %d numbers is %ld\n", count, sum);
  system("PAUSE"); /* pause */
}
```

Το παραπάνω πρόγραμμα με την εντολή while μετατρέπεται πολύ εύκολα στο παρακάτω πρόγραμμα που χρησιμοποιεί την εντολή for

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

main()
{
  long sum = 0;
  int i = 1;
  int count = 0;

  printf("\nEnter the number of integers you want to sum: ");
  scanf(" %d", &count);

  for (i = 1; i <= count; i++)
  {
    sum += i;
  }

  printf("Total of the first %d numbers is %ld\n", count, sum);
  system("PAUSE"); /* pause */
}
```

6ο Μέρος – Εντολή Επανάληψης: DO – WHILE, BREAK, CONTINUE, Ένθετες εντολές

Περιγραφή do while

Όπως και στην εντολή while χρησιμοποιούμε την εντολή επανάληψης do-while όταν δεν γνωρίζουμε εκ των προτέρων πόσες ακριβώς φορές χρειάζεται να επαναληφθεί μια συγκεκριμένη ενέργεια. Η διαφορά της εντολής επανάληψης do-while από την εντολή while είναι ότι ενώ στην εντολή while είναι δυνατόν να μην πραγματοποιηθεί καμία επανάληψη (εάν η συνθήκη ελέγχου λάβει την τιμή FALSE (Ψευδής) από την αρχή), η εντολή do-while θα εκτελεστεί τουλάχιστον μία φορά. Αφού δεν γνωρίζουμε τον ακριβή αριθμό των επαναλήψεων χρησιμοποιούμε μια συνθήκη ελέγχου η οποία μας επιτρέπει να συνεχίσουμε ή να διακόψουμε τις επαναλήψεις μας.

Σύνταξη do - while

```
do
{
  εντολή1;
  εντολή2;
  ......
}
while (συνθήκη);
```

Από την σύνταξη της εντολής do-while μπορούμε πολύ εύκολα να παρατηρήσουμε πως η επανάληψη μπορεί να εκτελεστεί πολλές φορές (όσο η συνθήκη ελέγχου έχει την τιμή TRUE (Αληθής) αλλά τουλάχιστον μία φορά (στην περίπτωση που η συνθήκη ελέγχου λάβει την τιμή FALSE (Ψευδής) στην αρχή).

Παράδειγμα 6-1:

Το παρακάτω πρόγραμμα εκτελεί την εντολή εξόδου printf για 5 φορές χρησιμοποιώντας την εντολή επανάληψης do while. Παρατηρήστε πως πρώτα τυπώνουμε τον αριθμό στην οθόνη με την εντολή printf και ύστερα αυξάνουμε την τιμή του i κατά 1.

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

main()
{
    int i = 0;

    do
    {
        printf("the number is %d \n", i);
        i++;
    }
    while (i < 5);

    system("PAUSE"); /* pause */
}
```

*Εάν αλλάξουμε την αρχική τιμή της μεταβλητής i από **i = 0** σε **i = 10** και εκτελέσουμε το παραπάνω πρόγραμμα θα παρατηρήσουμε πως θα εκτελεστεί μία φορά η εντολή printf παρόλο που η συνθήκη ελέγχου στην while έχει την τιμή FALSE (Ψευδής) από την αρχή!*

Παράδειγμα 6-2:

Το παρακάτω πρόγραμμα αντιστρέφει τη σειρά των ψηφίων σε ένα τετραψήφιο αριθμό χρησιμοποιώντας την εντολή do while και εκτελώντας συνεχείς διαιρέσεις με το 10 και λαμβάνοντας το πηλίκο και το υπόλοιπο .

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

main()
{
  int num = 1234, rnum = 0, temp = 0;

  temp = num;

  do
  {
    rnum = 10 * rnum + temp % 10;
    temp = temp / 10;
  }
  while (temp != 0);

  printf ("\nThe number %d reversed is  %d\n", num, rnum );

  system("PAUSE"); /* pause */
}
```

Παράδειγμα 6-3:

Το παρακάτω πρόγραμμα μας παρουσιάζει μια απλή αντίστροφη μέτρηση από έναν αριθμό που θα δώσουμε εμείς από το πληκτρολόγιο έως το 0 με την do while.

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

main()
{
    int num;

    printf("Please enter the number to start\n");
    printf("the countdown: ");

    scanf("%d", &num);

    do
    {
      printf("%d \n", num);
      num--;
    }
```

```
    while (num > 0);

    printf("boom!\n");

    system("PAUSE"); /* pause */
}
```

Παράδειγμα 6-4:

Το παρακάτω πρόγραμμα μας παρουσιάζει μια απλή αντίστροφη μέτρηση από το 10 έως το 0. Παρατηρήστε πως έχουμε μία φωλιασμένη εντολή for (δίχως σώμα) η οποία είναι υπεύθυνη για την καθυστέρηση στην εκτύπωση των αριθμών στην οθόνη.

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

main()
{
    int start = 10;
    long delay;

    do
    {
      printf(" %d\n",start);
      start--;
      for (delay=0; delay < 200000000; delay++); /* delay loop */
    }
    while (start > 0);

    printf("Zero!\nBlast off!\n");
    system("PAUSE"); /* pause */
}
```

Η εντολή break που είδαμε στις εντολές for και while μπορεί να χρησιμοποιηθεί κανονικότητα και στην εντολή επανάληψης do-while με παρόμοιο τρόπο. Όταν ο έλεγχος του προγράμματος φτάσει στην εντολή break; ο οποιοσδήποτε κύκλος επανάληψης διακόπτεται και ο έλεγχος περνά στην επόμενη εντολή **μετά** την εντολή επανάληψης.

Παράδειγμα 6-5:

Το παρακάτω πρόγραμμα χρησιμοποιεί την εντολή break για να τερματίσει την επανάληψη στην εντολή do while, που βρίσκεται σε ένα ατέρμονα κύκλο.

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

main()
{
  int i = 0;

  do
  {
```

```
      i++;
      printf(" the value of i is %d\n",i);
      if (i > 5)
      {
         break;
      }
  }
  while (1); /* always TRUE */

  system("PAUSE"); /* pause */
}
```

Παράδειγμα 6-6:

Το παρακάτω παράδειγμα εκτυπώνει μόνο τους μονούς αριθμούς i κατά φθίνουσα σειρά, κάνοντας χρήση της εντολή continue για να προσπεράσει την εκτύπωση των ζυγών αριθμών i.

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

main()
{
    int i = 10;

    do
    {
       i--;

       if (i % 2 == 0)
       {
          continue;       /* skip even i */
       }

       printf (" the value of i is: %d \n", i);
    }
    while ( i > 0 );

    system("PAUSE"); /* pause */
}
```

Η εντολή **continue;** είναι εντολή άμεσης διακοπής μόνο του συγκεκριμένου κύκλου επανάληψης στην C. Εάν ο έλεγχος εκτέλεσης του προγράμματός μας φτάσει στην **continue;** τότε ο έλεγχος βγαίνει αμέσως από το συγκεκριμένο κύκλο και συνεχίζει με τον επόμενο κύκλο της επανάληψης.

Παράδειγμα 6-7:

Το παρακάτω πρόγραμμα μας δίνει 3 προσπάθειες ώστε να μαντέψουμε έναν προκαθορισμένο αριθμό από το 10 έως το 20

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

```
main()
{
  int chosen = 15, guess = 0, count;

  printf("\nGuess A Number.\n");

  for (count = 3; count > 0; count--)
  {
    printf("\nYou have %d tries left.", count);
    printf("\nEnter a guess: ");
    scanf("%d", &guess);

    if (guess == chosen)
    {
      printf("\nYou guessed it!\n");
      break;
    }

    if ((guess < 1) || (guess > 20))
      printf("between 1 and 20.\n ");
    else
      printf("Sorry. %d is wrong.\n", guess);
  }
  if (count == 0)
    printf("\nYou have had three tries and failed. The number was
%d\n", chosen);

  system("PAUSE"); /* pause */
}
```

Παράδειγμα 6-8:

Το παρακάτω πρόγραμμα εκτελείται έως ότου πατήσουμε το πλήκτρο ~. Προσέξτε πως η εντολή επανάληψης for *δεν έχει καμία παράσταση στην επικεφαλίδα της. Σε αυτή την περίπτωση η* for *δεν θα τερματιστεί από μόνη της αλλά μόνο με παρέμβαση της εντολής διακοπής* break.

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

main()
{
    printf("press ~ to exit");

    for( ; ; )
    {
        char ch=getchar();
        if (ch == '~')
        {
            break;
        }
    }
```

```
    system("PAUSE"); /* pause */
}
```

Η εντολή **break;** είναι εντολή άμεσης διακοπής και εξόδου από τον κύκλο επανάληψης οποιασδήποτε εντολής επανάληψης στην C. Εάν ο έλεγχος εκτέλεσης του προγράμματός μας φτάσει στην **break;** τότε ο έλεγχος βγαίνει αμέσως από την επανάληψη και συνεχίζει με τις επόμενες εντολές εντός της επανάληψης.

Παράδειγμα 6-9:

Το παρακάτω παράδειγμα εκτυπώνει μόνο τα μονά i κατά φθίνουσα σειρά, χρησιμοποιώντας την εντολή for κάνοντας μείωση του δείκτη.

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

main()
{
    int i = 0;

    for (i = 10; i > 0; i--)
    {
       if ( i % 2 == 0)
          continue;       /* skip even i */

       printf (" the value of i is: %d \n", i);
    }

    system("PAUSE"); /* pause */
}
```

Η εντολή **continue;** είναι εντολή άμεσης διακοπής μόνο του συγκεκριμένου κύκλου επανάληψης στην C. Εάν ο έλεγχος εκτέλεσης του προγράμματός μας φτάσει στην **continue;** τότε ο έλεγχος βγαίνει αμέσως από το συγκεκριμένο κύκλο και συνεχίζει με τον επόμενο κύκλο της επανάληψης.

Η εντολή break που είδαμε στην εντολή for μπορεί να χρησιμοποιηθεί κανονικότητα και στην εντολή επανάληψης while με παρόμοιο τρόπο. Όταν ο έλεγχος του προγράμματος φτάσει στην εντολή break; ο οποιοσδήποτε κύκλος επανάληψης διακόπτεται και ο έλεγχος περνά στην επόμενη εντολή **μετά** την εντολή επανάληψης.

Παράδειγμα 6-10:

Το παρακάτω πρόγραμμα χρησιμοποιεί την εντολή break για να τερματίσει την επανάληψη στην εντολή while. Προσέξτε την σύνταξη της while που μας οδηγεί σε έναν ατέρμονα κύκλο.

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

main()
{
  int i = 0;
```

```
  while (1)
  {
      i = i + 1;

      printf(" the value of i is %d\n",i);

      if ( i > 5 )
      {
         break;
      }
  }

  system("PAUSE"); /* pause */
}
```

Παράδειγμα 6-11:

Το παρακάτω παράδειγμα εκτυπώνει μόνο τους μονούς αριθμούς i κατά φθίνουσα σειρά χρησιμοποιώντας την εντολή επανάληψης while.

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

main()
{
    int i = 10;

    while ( i > 0 )
    {
       i--;

       if (i % 2 == 0)
       {
          continue;       /* skip even i */
       }

       printf (" the value of i is: %d \n", i);
    }

    system("PAUSE"); /* pause */
}
```

Παράδειγμα 6-12:

Το παρακάτω πρόγραμμα υπολογίζει το πλήθος και το σύνολο των άρτιων και των περιττών αριθμών που θα του δώσουμε

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

main()
```

```
{
  int artioi = 0, sumart = 0, perittoi = 0, sumper = 0;
  int answer, line = 0;

  printf(" Doste ton arithmo h doste 0 gia telos\n");

  for ( ; ; )
  {
    line++;
    printf("noumero %d = ", line);
    scanf("%d", &answer);

    if (answer == 0)
      break;
    else
    {
      if (answer % 2 == 0)
      {
         artioi++;
         sumart = sumart + answer;
      }
      else
      {
         perittoi++;
         sumper = sumper + answer;
      }
    } /* end if */
  } /* end for */

  printf(" Dosate %d artious kai %d perittoys\n", artioi,
perittoi);
  printf(" Synolo artion: %d, synolo peritton: %d\n", sumart,
sumper);

  system("PAUSE"); /* pause */
}
```

Οι δύο εντολές for και while πολύ συχνά χρησιμοποιούνται μαζί, η μία μέσα στην άλλη, όπως φαίνεται στο παρακάτω παράδειγμα.

Παράδειγμα 6-13:

Το παρακάτω πρόγραμμα αθροίζει όλους τους ακέραιους αριθμούς από το 1 έως το 10 εμφανίζοντας όλα τα ενδιάμεσα στάδια και αθροίσματα. Χρησιμοποιεί την εντολή for για να υπολογίσει το άθροισμα για κάθε αριθμό από το 1 έως το 10 και την εντολή while για το εκάστοτε μερικό άθροισμα (από το 1 έως και τον εκάστοτε αριθμό).

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

main()
{
```

```
  int sum = 1, j, count = 10, i;

  for (i = 1 ; i <= count ; i++)
  {
    sum = 1;
    j = 1;
    printf("\n1");

    while(j < i)
    {
      sum += ++j;
      printf("+%d", j);
    }

    printf(" = %d\n", sum);
  }

  system("PAUSE"); /* pause */
}
```

Παράδειγμα 6-14:

Το παρακάτω πρόγραμμα χρησιμοποιεί τις εντολές for *και* while *για να παρουσιάσει τις δυνάμεις του 2 (που είναι μικρότερες από* 2^{14} *χωρίς να χρησιμοποιεί την μαθηματική βιβλιοθήκη* math.h*).*

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

main()
{
    int ex, result, i;

    for (i=0; i < 14; i++)
    {
      result = 1;
      ex = i;

      while (ex > 0)
      {
        result *= 2;
        ex--;
      }

      printf("2 to the %d power is %d \n", i, result);

    } /* for loop */

    system("PAUSE"); /* pause */
}
```

Παράδειγμα 6-15:

Το παρακάτω πρόγραμμα χρησιμοποιεί την εντολή while δύο φορές, την μία ένθετη στην άλλη, για να για τα υπολογίσει το παραγοντικό ενός ακεραίου αριθμού.

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

main()
{
  int value = 0;
  double result;

  while ( value >= 0)
  {
     printf("Value? (-1 to quit) "); /* -1 to quit */
     scanf("%d", &value);

     if (value >= 0)
     {
       result = 1;
       while (value > 0)
       {
          result *= value;
          value--;
       }

       printf("Factorial = %.0f\n", result);
    }
  }

  system("PAUSE"); /* pause */
}
```

7ο Μέρος – Μονοδιάστατοι Πίνακες

Περιγραφή

Ο μονοδιάστατος πίνακας είναι μια σύνθετη δομή μεταβλητών στις περισσότερες γλώσσες προγραμματισμού. Ένας πίνακας είναι ένα σύνολο από μεταβλητές που έχουν όλες τον ίδιο τύπο δεδομένων και συντάσσονται με έναν ιδιαίτερο τρόπο. Τα στοιχεία ενός πίνακα μπορούν να προσπελαστούν χρησιμοποιώντας το όνομα του πίνακα και έναν αριθμητικό δείκτη ο οποίος αντιστοιχεί στο συγκεκριμένο στοιχείο. Μια ιδιοτροπία των πινάκων είναι πως η αρίθμηση ξεκινά από το 0 και όχι από το 1, δηλαδή το πρώτο στοιχείο του πίνακα έχει δείκτη 0, το δεύτερο έχει δείκτη 1 και το τελευταίο στοιχείο ενός πίνακα θα έχει δείκτη το μέγεθος του πίνακα μείον 1. Προσοχή! Ο δείκτης αναφέρεται στη θέση του στοιχείου στον πίνακα και όχι στην τιμή του περιεχομένου του στοιχείου. Ουσιαστικά ένας πίνακας μπορεί να αντικατασταθεί από πολλές μεταβλητές. Ο λόγος όμως που χρησιμοποιούμε έναν πίνακα και όχι (δεκάδες) απλές μεταβλητές είναι η ευκολία στην προσπέλαση των στοιχείων ενός πίνακα εκμεταλλευόμενοι τις γνωστές μας εντολές επανάληψης.

Σύνταξη

Για να δηλώσουμε έναν πίνακα θα πρέπει υποχρεωτικά να δηλώσουμε εκ των προτέρων το μέγεθος του πίνακα. Η δήλωση γίνεται με παρόμοιο τρόπο όπως κάνουμε με τις απλές μεταβλητές που έχουμε χρησιμοποιήσει έως τώρα, δηλώνοντας τον τύπο δεδομένων και στη συνέχεια το όνομα του πίνακα μαζί με το μέγεθός του μέσα σε αγκύλες.

```
Τύπος_δεδομένων όνομα_πίνακα[μέγεθος πίνακα];
```

Για να προσπελάσουμε ένα στοιχείο ενός πίνακα θα πρέπει να χρησιμοποιήσουμε το όνομα του πίνακα και στην συνέχεια τον αντίστοιχο δείκτη μέσα σε αγκύλες για το συγκεκριμένο στοιχείο του πίνακα.

```
όνομα_πίνακα[δείκτης]
```

Παράδειγμα 7-1:

Το παρακάτω πρόγραμμα χρησιμοποιεί έναν πίνακα 5 θέσεων για να αποθηκεύσει 5 μεταβλητές με τιμές από 1 έως και 5. Προσέξτε πως ταυτίζουμε το δείκτη της θέσης του πίνακα με την τιμή του πίνακα στη θέση αυτή.

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

main()
{
    int a[5]; /* array size is 5 */
    int i;

    for (i = 0; i < 5; i++)
    {
       a[i]=i;
    }

    for (i = 0; i < 5; i++)
    {
        printf("value in array %d\n", a[i]);
    }
    system("PAUSE"); /* pause */
}
```

Μια συνηθισμένη διαδικασία που κάνουμε με τους πίνακες πριν τους χρησιμοποιήσουμε για να κάνουμε πράξεις με τα στοιχεία τους είναι να δώσουμε αρχικές (συνήθως μηδενικές-ουδέτερες τιμές στα στοιχεία).

Παράδειγμα 7-2:

Το παρακάτω πρόγραμμα ορίζει την τιμή μηδέν σε όλα τα στοιχεία ενός πίνακα .

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

main()
{
```

```
   int n[10]; /* array size is 10 */
   int i;

   /* initialize elements of array n to 0 */
   for (i = 0; i < 10; i++)
   {
      n[i] = 0;
   }

   printf("%s%13s\n", "Element", "Value"); /* format labels */

   for (i = 0; i < 10; i++)
   {
      printf("%7d%13d\n", i, n[i]);
   }

   system("PAUSE"); /* pause */
}
```

Συνήθως χρησιμοποιούμε σταθερές για να δηλώσουμε το μέγεθος ενός πίνακα. Με αυτό τον τρόπο δεν χρειάζεται να κάνουμε πολλές αλλαγές στον κώδικά μας εάν χρειαστεί να αλλάξουμε το μέγεθος του πίνακά μας.

Παράδειγμα 7-3:

Το παρακάτω πρόγραμμα χρησιμοποιεί μια σταθερά για να ορίσει το μέγεθος του πίνακα και ορίζει σαν τιμή σε όλα τα στοιχεία ενός πίνακα τα πολλαπλάσια του 2 αρχίζοντας από το 2.

```
#include <stdio.h>
#include <stdlib.h>
#define SIZE 10

main()
{
   int s[SIZE];
   int j;

   for (j = 0; j < SIZE; j++)
   {
      s[j] = 2 + 2 * j;
   }

   printf("%s%13s\n", "Element", "Value"); /* format labels */

   for (j = 0; j < SIZE; j++)
   {
      printf( "%7d%13d\n", j, s[j] );
   }

   system("PAUSE"); /* pause */
}
```

Μπορούμε να χρησιμοποιήσουμε εντολές εισόδου όπως είναι η εντολή scanf για να εισάγουμε τα δεδομένα

Παράδειγμα 7-4:

Το παρακάτω πρόγραμμα μας υπολογίζει το μέσο όρο δέκα αριθμών που θα του δώσουμε εμείς χρησιμοποιώντας έναν πίνακα 10 θέσεων για να καταχωρήσουμε τους αριθμούς. Στη συνέχεια με μία επανάληψη αθροίζουμε τα στοιχεία του πίνακα και υπολογίζουμε το μέσο όρο. Προσέξτε πως κάνουμε τη διαίρεση για να συγκρατήσουμε και το δεκαδικό μέρος.

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

main()
{
  int numbers[10];
  int count = 10; /* total numbers */
  long sum = 0;
  float average = 0;
  int i;

  printf("\nEnter the 10 numbers:\n");

  for(i = 0; i < count; i++)
  {
    printf("%2d> ",i+1);
    scanf("%d", &numbers[i]);
    sum += numbers[i];
  }

  average = (float)sum/count;

  printf("\nAverage of the ten numbers entered is: %f\n",
average);

  system("PAUSE"); /* pause */
}
```

Παράδειγμα 7-5:

Το παρακάτω πρόγραμμα μας αναζητεί το μικρότερο και το μεγαλύτερο από δέκα αριθμούς που θα του δώσουμε εμείς. Οι αριθμοί αποθηκεύονται σε έναν πίνακα και κάνουμε τις αναζητήσεις στα στοιχεία του πίνακα (και όχι μόνο στις τιμές). Στη συνέχεια παρουσιάζει τους αριθμούς αυτούς καθώς και τη σχετική τους θέση στη δεκάδα.

```
#include <stdio.h>
#include <stdlib.h>
#define SIZE 10

main()
{
  int numbers[SIZE];
  int max, min, i;
```

```
  printf("\nEnter the 10 numbers:\n");

  for(i = 0; i < SIZE; i++)
  {
    printf("%2d> ",i+1);
    scanf("%d", &numbers[i]);
  }

  max = 0;
  min = 0;

  for(i = 0; i < SIZE; i++)
  {
    if (numbers[i] < numbers[min])
      min = i;

    if (numbers[i] > numbers[max])
      max = i;
  }

  printf("\nThe min number is: %d at position %d\n", numbers[min],
min + 1);
  printf("\nThe max number is: %d at position %d\n", numbers[max],
max + 1);

  system("PAUSE"); /* pause */
}
```

Παράδειγμα 7-6:

Το παρακάτω πρόγραμμα μας ταξινομεί κατά αύξουσα σειρά (από το μικρότερο προς το μεγαλύτερο) μια σειρά από δέκα αριθμούς ενός πίνακα που θα του δώσουμε εμείς από το πληκτρολόγιο.

```
#include <stdio.h>
#include <stdlib.h>
#define SIZE 10

main()
{
  int numbers[SIZE];
  int i, j, tmp;

  printf("\nEnter the 10 numbers:\n");

  for(i = 0; i < SIZE; i++)
  {
    printf("%2d> ",i+1);
    scanf("%d", &numbers[i]);
  }

  for(i = 0; i < SIZE - 1; i++)
  {
```

```
    for (j = 0; j < SIZE - 1 - i; j++)
    {
      if (numbers[j+1] < numbers[j])
      {
        tmp = numbers[j];
        numbers[j] = numbers[j+1];
        numbers[j+1] = tmp;
      }
    }
  }

  printf("\n the sorted numbers are:\n");

  for(i = 0; i < SIZE; i++)
  {
    printf("%2d> %d\n",i+1, numbers[i]);
  }

  system("PAUSE"); /* pause */
}
```

Άσκηση 7-1:

Γράψτε ένα πρόγραμμα που να διαβάζει τις θερμοκρασίες (σαν απλούς δεκαδικούς αριθμούς) των ημερών ενός μήνα και στη συνέχεια να υπολογίζει το πλήθος και το ποσοστό των ημερών που έχουμε θερμοκρασία ίση ή κάτω του μηδενός. Επίσης να υπολογίζει τη μέση τιμή της θερμοκρασίας για όλες τις ημέρες που έχουμε θερμοκρασία ίση ή κάτω του μηδενός. Χρησιμοποιήστε ένα πίνακα για να αποθηκεύσετε τις θερμοκρασίες και μία σταθερά για το μηδέν. Η παρουσίαση των αριθμών να γίνεται με 3 δεκαδικά ψηφία.

```
#include <stdio.h>
#include <stdlib.h>
#define SIZE 30
#define MHDEN 0

main()
{
  float temp[SIZE];
  int i, count;
  float subtotal, pososto, mesi;

  /* eisodos dedomenon */
  printf("\nDose tis thermokrasies:\n");
  for(i = 0; i < SIZE; i++)
  {
     printf("%2d> ",i+1);
     scanf("%f", &temp[i]);
  }

  /* arxikes times */
  count = 0;
  subtotal = 0;
  pososto = 0;
```

```
  mesi = 0;

  /* eksetasi dedomenon */
  for(i = 0; i < SIZE; i++)
  {
     /* elegxos gia thermokrasia <= 0 */
     if (temp[i] <= MHDEN)
     {
        count++;
        subtotal += temp[i];
     }
  }

  /* ypologismos pososToy kai mesis timis */
  pososto = (float)count/SIZE*100;
  mesi = (float)subtotal/count;

  /* parousiasi stoixeion */
  printf("Exeis %d meres me thermokrasia <= apo %d\n", count,
MHDEN);
  printf("Gia tis hmeres me thermokrasia <= apo %d exoume: \n",
MHDEN);
  printf("To pososto ton hmeron einai %.2f \% \n", pososto);
  printf("H mesi timi ton hmeron einai %.2f \% \n", mesi);

  system("PAUSE"); /* pause */
}
```

Παράδειγμα 7-7:

Το παρακάτω πρόγραμμα καταγράφει τις τιμές της ημερήσιας παραγωγής ενός προϊόντος σε μηνιαία βάση και στη συνέχεια υπολογίζει τη συνολική μηνιαία παραγωγή, το μέσο όρο της ημερήσιας παραγωγής, καθώς επίσης παρουσιάζει τις ημέρες (και την αντίστοιχη) παραγωγή που είναι μικρότερη του 80% του μέσου όρου. Η επιλογή των ημερών αυτών γίνεται με σύγκριση του υπολογισμένου μέσου όρου (σε προηγούμενο βρόχο) με το εκάστοτε στοιχείο του πίνακα.

```
#include <stdio.h>
#include <stdlib.h>
#define MINAS 30

main()
{
  int paragogi[MINAS];
  int synolo = 0;
  float mo = 0.0;
  int i;

  printf("Doste tin imerisia paragogi:\n");

  for (i = 0; i < MINAS; i++)
  {
     printf("imera %d =  ", i+1);
     scanf("%d", &paragogi[i]);
```

```
      synolo += paragogi[i];
   }

   printf("H synoliki paragogi toy mina einai %d\n", synolo);

   mo = (float)synolo/MINAS;
   printf("O mesos oros tis imerisias paragogis einai: %.2f \n",
mo);

   printf("Hmeres me paragogi < 80% tou mesou orou:\n");
   for (i = 0; i < MINAS; i++)
   {
      if (paragogi[i] < 0.8*mo)
           printf("Hmera %d me paragogi: %d \n", i+1, paragogi[i]);
   }

   system("PAUSE"); /* pause */
}
```

Παράδειγμα 7-8:

Το παρακάτω πρόγραμμα καταγράφει τις τιμές της ημερήσιας είσπραξης ενός καταστήματος, δίχως να γνωρίζουμε εκ των προτέρων το πλήθος των ημερών αλλά έως ότου ο χρήστης εισάγει αρνητικό αριθμό. Στη συνέχεια μας παρουσιάζει τη συνολική είσπραξη καθώς και τις ημέρες με τη μεγαλύτερη και τη μικρότερη είσπραξη. Επειδή δεν γνωρίζουμε το μέγεθος του πίνακα, τον δηλώνουμε «πολύ μεγάλο» ώστε να χωρέσουν όλοι οι αριθμοί, δηλαδή μεγέθους 365 που αντιστοιχεί σε ένα έτος.

```
#include <stdio.h>
#include <stdlib.h>
#define ETOS 365

main()
{
   float eispraksi[ETOS];
   float value;
   double synolo = 0.0;
   int orio, min, max, i;

   i = 0;
   orio = 0;

   printf("Doste tin imerisia eispraksi h doste -1 gia telos:\n");
   while (i < ETOS)
   {
      printf("imera %d =  ", i + 1);
      scanf("%f", &value);
      if (value < 0)
           break;
      eispraksi[i] = value;
      synolo += eispraksi[i];
      orio++;
      i++;
   }
```

```
  min = 0;
  max = 0;

  for (i = 0; i < orio; i++)
  {
     if (eispraksi[i] < eispraksi[min])
          min = i;

     if (eispraksi[i] > eispraksi[max])
          max = i;
  }

  printf("\n - - - - - - \n");
  printf("H synoliki eispraksi einai %.2f\n", synolo);
  printf("H megalyteri eispraksi einai %.2f thn imera %d\n",
eispraksi[max], max + 1 );
  printf("H mikroteri eispraksi einai %.2f thn imera %d\n",
eispraksi[min], min + 1 );

  system("PAUSE"); /* pause */
}
```

Παράδειγμα 7-9:

Το παρακάτω πρόγραμμα καταγράφει τιμές σε ένα πίνακα βάση της φόρμουλας $2 * i + i^2$ *και στη συνέχεια ερευνά εάν μια τυχαία δοσμένη τιμή από το χρήστη περιέχεται στις τιμές του πίνακα που γεμίσαμε με στοιχεία ή όχι. Εάν υπάρχει η τιμή αυτή μας τυπώνει και τη θέση του δείκτη στον πίνακα.*

```
#include <stdio.h>
#include <stdlib.h>
#define SIZE 10

main()
{
   int a[SIZE];
   int value;
   int i, flag;

   flag = -1;

   for ( i = 0; i < SIZE; i++ )
   {
      a[i] = 2 * i + i*i ;
   }

   printf( "\nEnter integer search key:\n" );
   scanf( "%d", &value );

   for ( i = 0; i < SIZE; i++ )
   {
      if ( a[i] == value )
        flag = i;
```

```
   }

   if ( flag == -1)
      printf("\n\a value %d not found in array!", value);
   else
     printf("\n value %d found at index %d", value, flag);

   system("PAUSE"); /* pause */
}
```

Παράδειγμα 7-10:

Το παρακάτω πρόγραμμα διαβάζει 10 δεκαδικούς αριθμούς και να τους καταχωρεί σ' έναν πίνακα. Στη συνέχεια να ελέγχει τους όλους τους αριθμούς του πίνακα και όλους τους μη-αρνητικούς να τους καταχωρεί σε ένα άλλο πίνακα. Προσέξτε πως οι δύο πίνακες έχουν ίδια διάσταση (για την περίπτωση που όλοι οι αριθμοί είναι θετικοί

```
#include <stdio.h>
#include <stdlib.h>
#define SIZE 10

main()
{
   float numbers[SIZE];
   float noneg[SIZE];
   int line = 1;
   int i,j;

   /* diabasma ton arithmon */
   printf("Dose toys 10 thetikous arithmous:\n");
   for (i = 0; i < SIZE; i++)
   {
       printf("%2d>", line++);
       scanf("%f", &numbers[i]);
   }

   /* gemisma me 0 ola ta stoixeia toy deyterou pinaka */
   for (i = 0; i < SIZE; i++)
   {
       noneg[i] = 0;
   }

   j = 0;  /* deiktis gia to deytero pinaka */

   for (i = 0; i < SIZE; i++)
   {
       if (numbers[i] >= 0)
       {
          noneg[j] = numbers[i];
          j++;
       }
   }
```

```
   printf("numbers noneg\n"); /* format labels */

   for (i = 0; i < SIZE; i++)
   {
       printf("%.2f \t %.2f\n", numbers[i], noneg[i]);
   }

   system("PAUSE");  /* pause */
}
```

Άσκηση 7-2:

Γράψτε ένα πρόγραμμα που να διαβάζει το υπόλοιπο ενός λογαριασμού καθώς και το ετήσιο σταθερό επιτόκιο και το χρονικό διάστημα για κάθε μία από 5 πιθανές επενδυτικές δραστηριότητες. Στη συνέχεια το πρόγραμμα να επιλέγει την πιο συμφέρουσα από αυτές. Η παρουσίαση των αριθμών να γίνεται με 3 δεκαδικά ψηφία.

```
#include <stdio.h>
#include <stdlib.h>
#define COUNT 5

main()
{
   /* Oi metavlites moy */
   float balance = 0;
   float newbalance = 0;
   float maxbalance = 0;
   int maxi = 0;
   float rate = 0;
   float interest = 0;
   int years = 0;
   float investment[COUNT];
   int i = 0;
   int j = 0;

   printf("Dose to torino ypoloipo toy logarismoy: ");
   scanf("%f", &balance);

   for (i = 0; i < COUNT; i++)
   {
       /* eisodos dedomenon gia ypologismoys */
       printf(" * * * * * * * * * * * * *\n");
       printf(" dose to etisio stathero epitokio tis %d ependysis:
", i+1);
       scanf("%f", &rate);
       printf(" dose to xroniko diastima se eth tis %d ependysis:
", i+1);
       scanf("%d", &years);

       /* prosorino ypoloipo gia ypo0logismoys */
       newbalance = balance;   /* temporary balance for
calculations */
```

```
        for (j = 1; j <= years; j++)
        {
            interest = newbalance * (rate/100);
            newbalance += interest;
        }

        /* kataxorisi tis apodosis tis i ependyshs */
        investment[i] = newbalance;
    }  /* outer for loop */

    /* elegxos gia kalyterh apodosi */
    for (i = 0; i < COUNT; i++)
    {
        if (investment[i] >= maxbalance)
        {
           maxbalance = investment[i];
           maxi = i;
        }
    }

    printf("H kalyterh ependysh einai h %d me apodosi %.3f\n",
maxi+1, maxbalance);

    system("PAUSE");
}
```

8ο Μέρος – Συμβολοσειρές

Τι είναι οι συμβολοσειρές

Μια συμβολοσειρά (string) είναι απλά ένας μονοδιάστατος πίνακας από χαρακτήρες ο οποίος μας χρησιμεύει για να καταχωρούμε και να μεταχειριζόμαστε μια σειρά από χαρακτήρες σαν να είναι μια οντότητα. Μια συμβολοσειρά τερματίζεται πάντοτε με το χαρακτήρα NULL (\0).

Σύνταξη

Μια συμβολοσειρά συντάσσεται όπως ακριβώς ένας (μονοδιάστατος) πίνακας τύπου χαρακτήρα, δηλαδή ως εξής:

```
    char myname[];
```

Μπορούμε να δηλώσουμε συμβολοσειρές με διάφορους τρόπους. Για να δηλώσουμε μια συμβολοσειρά μαζί με την τιμή μπορούμε να χρησιμοποιήσουμε τα διπλά αγγλικά εισαγωγικά για να ορίσουμε την αρχή και το τέλος του περιεχομένου της συμβολοσειράς.

π.χ. `char myname[]="Dan";`

Για να εκτυπώσουμε το περιεχόμενο μιας συμβολοσειράς με την εντολή (συνάρτηση) εξόδου printf() πρέπει να χρησιμοποιήσουμε οπωσδήποτε τους χαρακτήρες μετατροπής %s.

Παράδειγμα 8-1:

Το παρακάτω παράδειγμα δηλώνει μια συμβολοσειρά και στη συνέχεια την εκτυπώνει του σε μία γραμμή.

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

main()
{
  char myword[]="hello world";

  printf(" name is %s \n", myword);

  system("PAUSE"); /* pause */
}
```

Ένας άλλος τρόπος για να δηλώσουμε μια συμβολοσειρά μαζί με την τιμή είναι να γεμίσουμε τον πίνακα με χαρακτήρες

π.χ. `char myname[] = { 'D', 'a', 'n' };`

Παράδειγμα 8-2:

Το παρακάτω παράδειγμα δηλώνει και εκτυπώνει μια συμβολοσειρά.

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

main()
{
   char myname[] = { 'D', 'a', 'n' };

   printf(" name is %s \n", myname);

   system("PAUSE"); /* pause */
}
```

Μπορούμε να μορφοποιήσουμε την εκτύπωση συμβολοσειρών όπως ακριβώς κάναμε με τους αριθμούς.

Παράδειγμα 8-3:

Το παρακάτω παράδειγμα δηλώνει και εκτυπώνει συμβολοσειρές με την κατάλληλη μορφοποίηση.

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

main()
{
   printf("%15s","right\n");
   printf("%-15s","left\n");
```

```
   system("PAUSE"); /* pause */
}
```

Μπορούμε να χρησιμοποιήσουμε την κλασσική σύνταξη της εντολής scanf ώστε να διαβάσουμε μια συμβολοσειρά από το πληκτρολόγιο και να την καταχωρήσουμε σε μια μεταβλητή συμβολοσειράς.

Παράδειγμα 8-4:

Το παρακάτω παράδειγμα ζητά να του δώσουμε μια συμβολοσειρά από το πληκτρολόγιο και στη συνέχεια μας την παρουσιάζει στην οθόνη μας.

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

main()
{
    char yourName[15];

    printf("Enter your name: ");
    scanf("%s",&yourName);
    printf("Hello %s\n", yourName);

    system("PAUSE"); /* pause */
}
```

Προσοχή! Στο παραπάνω παράδειγμα πρέπει η συμβολοσειρά που θα δώσουμε από το πληκτρολόγιο να «χωράει» στη μεταβλητή μας.

Μπορούμε να χρησιμοποιήσουμε την ειδική εντολή εισόδου gets για να διαβάσουμε μια συμβολοσειρά από το πληκτρολόγιο

Παράδειγμα 8-5:

Το παρακάτω παράδειγμα ζητά να του δώσουμε μια συμβολοσειρά από το πληκτρολόγιο και στη συνέχεια μας την παρουσιάζει στην οθόνη μας.

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

main()
{
   char line[100]; /* Line we are looking at */

   printf("Enter a line: ");
   gets(line);

   printf("The line is: %s\n", line);

   system("PAUSE"); /* pause */
}
```

Οι συμβολοσειρές πάντοτε τερματίζονται με τον ειδικό χαρακτήρα NULL (\0). Με αυτό τον τρόπο μπορούμε να διορθώσουμε το παραπάνω παράδειγμα ώστε να μην εκτυπώνονται «σκουπίδια» μετά τη συμβολοσειρά.

Παράδειγμα 8-6:

Το παρακάτω παράδειγμα δηλώνει και εκτυπώνει μια συμβολοσειρά και τερματίζεται με το χαρακτήρα NULL.

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

main()
{
  char myname[4];

  myname[0] = 'D';
  myname[1] = 'a';
  myname[2] = 'n';
  myname[3] = '\0';  /* Null char to remove garbage*/

  printf(" name is %s \n", myname);

  system("PAUSE"); /* pause */
}
```

Εάν ο ειδικός χαρακτήρας NULL (\0) τοποθετηθεί σε λάθος θέση μπορεί να έχουμε απροσδόκητα αποτελέσματα.

Παράδειγμα 8-7:

Το παρακάτω παράδειγμα εκτυπώνει μια συμβολοσειρά και τερματίζεται με το χαρακτήρα NULL πριν το σωστό σημείο .

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

main()
{
  printf("The character \0 is used to terminate a string.");

  system("PAUSE"); /* pause */
}
```

Συναρτήσεις συμβολοσειρών

Για να καταχωρήσουμε μια συμβολοσειρά σε μια ανάλογου τύπου δεδομένων μεταβλητή πρέπει να χρησιμοποιήσουμε την ειδική συνάρτηση strcpy που υπάρχει στη βιβλιοθήκη συμβολοσειρών string.h.

Παράδειγμα 8-8:

*Το παρακάτω παράδειγμα καταχωρεί και στη συνέχεια εκτυπώνει μια συμβολοσειρά στην οθόνη χρησιμοποιώντας τη συνάρτηση **strcpy** που υπάρχει στη βιβλιοθήκη συμβολοσειρών **string.h**.*

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

main()
{
  char name[4];

  clrscr();  /* clear screen */

  strcpy(name, "Sam");
  printf(" name is %s \n", name);

  system("PAUSE"); /* pause */
}
```

Παράδειγμα 8-9:

*Το παρακάτω παράδειγμα καταχωρεί και στη συνέχεια εκτυπώνει μια συμβολοσειρά στην οθόνη χρησιμοποιώντας τη συνάρτηση **strcpy** που υπάρχει στη βιβλιοθήκη συμβολοσειρών **string.h**. Η συμβολοσειρά κόπτεται απότομα χρησιμοποιώντας τον ειδικό χαρακτήρα NULL.*

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
main()
{
   char name[30];

   strcpy(name, "Saaaaaaaaaaaaaam");

   printf("The name is %s\n", name);

   name[4] = '\0';

   printf("The name is %s\n", name);

   system("PAUSE"); /* pause */
}
```

Άλλες χρήσιμες συναρτήσεις για συμβολοσειρές είναι οι εξής:

Συνάρτηση	Περιγραφή
`strcpy(string1, string2)`	Αντιγράφει τη συμβολοσειρά **`string2`** στη συμβολοσειρά **`string1`**
`strcat(string1, string2)`	Προσαρτά τη συμβολοσειρά **`string2`** στο τέλος της συμβολοσειράς **`string1`**
`length = strlen(string)`	Επιστρέφει έναν ακέραιο αριθμό με το μήκος της συμβολοσειράς **`string`** (δηλαδή πραγματικό τον αριθμό των

	χαρακτήρων της)
strcmp(string1, string2)	Επιστρέφει τον αριθμό **0** εάν οι δύο συμβολοσειρές **string1** και **string2** ταυτίζονται, αλλιώς επιστρέφει έναν μη-μηδενικό αριθμό

Επίσης υπάρχει και η συνάρτηση sizeof(name) που μας επιστρέφει έναν ακέραιο αριθμό με το μέγεθος της μεταβλητής της συμβολοσειράς name (δηλαδή τον μέγιστο αριθμό των χαρακτήρων που μπορεί να φιλοξενήσει μια μεταβλητή τύπου δεδομένων συμβολοσειράς)

Μπορούμε να χρησιμοποιήσουμε την ειδική εντολή (συνάρτηση) fgets για να διαβάσουμε μια συμβολοσειρά. Η συνάρτηση fgets συντάσσεται ως εξής:

```
fgets(name, sizeof(name), stdin);
```

όπου:
name είναι μια μεταβλητή τύπου συμβολοσειράς
sizeof(name) είναι το μέγεθος της μεταβλητής της συμβολοσειράς name
stdin είναι η είσοδος της συμβολοσειράς

Παράδειγμα 8-10:

Το παρακάτω παράδειγμα διαβάζει μια συμβολοσειρά και στη συνέχεια μας επιστρέφει το μέγεθός της. Σημείωση: Η συμβολοσειρά περιλαμβάνει και τον ειδικό χαρακτήρα NULL.

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

main()
{
  char line[100]; /* Line we are looking at */
  printf("Enter a line: ");

  fgets(line, sizeof(line), stdin);

  printf("The length of the line is: %d\n", strlen(line));

  system("PAUSE"); /* pause */
}
```

Μπορούμε να μην χρησιμοποιήσουμε τη συνάρτηση length και να υπολογίσουμε μόνοι μας το μέγεθος μιας συμβολοσειράς χρησιμοποιώντας τις γνωστές εντολές επανάληψης που ξέρουμε:

Παράδειγμα 8-11:

Το παρακάτω παράδειγμα διαβάζει μια συμβολοσειρά και στη συνέχεια μας επιστρέφει το μέγεθός της χωρίς να χρησιμοποιήσουμε την ειδική συνάρτηση ***length****.*

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

main()
```

```
{
  char str1[] = "AAA";

  int count = 0;

  while (str1[count] != '\0')    /* Increment count until NULL*/
    count++;

  printf("\nThe length of the string \"%s\" is %d characters.",
str1, count);

  system("PAUSE"); /* pause */
}
```

Μπορούμε να κάνουμε διάφορες ενέργειες με τις συμβολοσειρές όπως να τις ενώσουμε σε μία νέα.

Παράδειγμα 8-12:

Το παρακάτω παράδειγμα διαβάζει δύο συμβολοσειρές από το πληκτρολόγιο και στη συνέχεια μας επιστρέφει μία με το διπλάσιο μέγεθος χρησιμοποιώντας τη συνάρτηση ***strcat****.*

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

main()
{
  char first[100];
  char last[100];
  char full[200];

  printf("Enter first name: ");
  scanf("%s", &first);

  printf("Enter last name: ");
  scanf("%s", &last);

  strcpy(full, first);
  strcat(full, " ");
  strcat(full, last);

  printf("The name is %s\n", full);

  system("PAUSE"); /* pause */
}
```

Μπορούμε να μην χρησιμοποιήσουμε τη συνάρτηση strcat και να ενώσουμε μόνοι μας τις συμβολοσειρές χρησιμοποιώντας τις γνωστές εντολές επανάληψης που ξέρουμε:

Παράδειγμα 8-13:

Το παρακάτω παράδειγμα διαβάζει δύο συμβολοσειρές από το πληκτρολόγιο και στη συνέχεια μας επιστρέφει μία με το διπλάσιο μέγεθος χωρίς να χρησιμοποιεί τη συνάρτηση ***strcat****.*

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

main()
{
  char str[][40] =  { "AAA", "BBBBBBBB" };
  int strLength[] = {0, 0};
  int i;

  for (i = 0 ; i<2 ; i++)
  {
     while (str[i][strLength[i]])
       strLength[i]++;
  }

  if (sizeof str[0] < strLength[0] + strLength[1] + 1)
    printf("\nNo enough space");
  else
  {
    strLength[1] = 0;
    while((str[0][strLength[0]++] = str[1][strLength[1]++]));
    printf("\n%s\n", str[0]);
  }

  system("PAUSE"); /* pause */
}
```

<u>Παράδειγμα 8-14</u>:

Το παρακάτω παράδειγμα διαβάζει δύο συμβολοσειρές από το πληκτρολόγιο και στη συνέχεια μας ενημερώνει εάν οι δύο συμβολοσειρές ταυτίζονται ή όχι.

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

main()
{
  char text1[100];
  char text2[100];

  printf("\nEnter the first string:\n");
  scanf("%s", &text1);

  printf("\nEnter the second string:\n");
  scanf("%s", &text2);

  if ( strcmp(text1, text2) == 0)
     printf(" the two strings are equal\n");
  else
```

```
    printf(" the two strings are NOT equal\n");

  system("PAUSE"); /* pause */
}
```

9ο Μέρος – Πολυδιάστατοι πίνακες

Περιγραφή

Οι πολυδιάστατοι πίνακες είναι μία ακόμη σύνθετη δομή μεταβλητών στις περισσότερες γλώσσες προγραμματισμού. Οι πολυδιάστατοι είναι ένα σύνολο από μεταβλητές που έχουν όλες τον ίδιο τύπο δεδομένων και συντάσσονται με έναν ιδιαίτερο τρόπο, παρόμοιο με τους μονοδιάστατους πίνακες. Τα στοιχεία ενός πολυδιάστατου πίνακα μπορούν να προσπελαστούν χρησιμοποιώντας το όνομα του πίνακα και τους αριθμητικούς δείκτες που αντιστοιχούν στο συγκεκριμένο στοιχείο. Μια ιδιοτροπία των πινάκων είναι πως η αρίθμηση ξεκινά από το 0 και όχι από το 1, δηλαδή το πρώτο στοιχείο του πίνακα έχει δείκτη 0, το δεύτερο έχει δείκτη 1 και το τελευταίο στοιχείο ενός πίνακα θα έχει δείκτη το μέγεθος του πίνακα μείον 1. Προσοχή! Οι δείκτες αναφέρονται στη θέση του στοιχείου στον πίνακα και όχι στην τιμή του περιεχομένου του στοιχείου. Ουσιαστικά ένας πολυδιάστατος πίνακας μπορεί να αντικατασταθεί από πάρα πολλές μεταβλητές ή από πολλούς μονοδιάστατους πίνακες. Ο λόγος όμως που χρησιμοποιούμε έναν πολυδιάστατο πίνακα και όχι (δεκάδες) απλές μεταβλητές είναι η ευκολία στην προσπέλαση των στοιχείων ενός πίνακα εκμεταλλευόμενοι τις γνωστές μας εντολές επανάληψης.

Στην C, θεωρητικά δεν υπάρχει όριο στις διαστάσεις ενός πίνακα, αν και σε ορισμένους μεταγλωττιστές το όριο είναι 256 διαστάσεις. Δημιουργούνται τόσες διαστάσεις όσοι είναι και οι αριθμοί που ακολουθούν στις τετράγωνες αγκύλες. Πίνακες μεγαλύτεροι των 2 διαστάσεων δεν μπορούν να εκτυπωθούν σε δύο διαστάσεις.

Σύνταξη

Για να δηλώσουμε έναν πολυδιάστατο πίνακα θα πρέπει υποχρεωτικά να δηλώσουμε εκ των προτέρων το μέγεθος του πίνακα. Η δήλωση γίνεται με παρόμοιο τρόπο όπως κάνουμε με τις απλές μεταβλητές που έχουμε χρησιμοποιήσει έως τώρα, δηλώνοντας τον τύπο δεδομένων και στη συνέχεια το όνομα του πίνακα μαζί με το μέγεθός του για κάθε διάσταση μέσα σε αγκύλες.

```
Τύπος_δεδομένων όνομα_πίνακα[μέγεθος διάστασης] [μέγεθος
διάστασης] [μέγεθος διάστασης] [μέγεθος διάστασης];
```

Για να προσπελάσουμε ένα στοιχείο ενός πίνακα θα πρέπει να χρησιμοποιήσουμε το όνομα του πίνακα και στην συνέχεια τους αντίστοιχους δείκτες μέσα σε αγκύλες για το συγκεκριμένο στοιχείο του πίνακα.

```
όνομα_πίνακα[δείκτης][δείκτης][δείκτης];
```

Συνήθως χρησιμοποιούμε σταθερές για να δηλώσουμε το μέγεθος ενός πίνακα. Με αυτό τον τρόπο δεν χρειάζεται να κάνουμε πολλές αλλαγές στον κώδικά μας εάν χρειαστεί να αλλάξουμε το μέγεθος του πίνακά μας.

Παράδειγμα 9-1:

Το παρακάτω πρόγραμμα δέχεται τις τιμές ενός πίνακα ακέραιων αριθμών διάστασης 4 x 3 και στη συνέχεια τους παρουσιάζει στην οθόνη

```
#include <stdio.h>
#include <stdlib.h>
#define ROW 4
#define COLUMN 3

main()
{
   int i, j;
   int pinakas[ROW][COLUMN];

   for (i=0; i<ROW; i++)
   {
       for (j=0; j<COLUMN; j++)
       {
           printf("Enter value of Row %d ,Column %d: ",i+1, j+1);
           scanf("%d", &pinakas[i][j]);
       }
   }

   printf("The array is:\n");
   printf("\t\tCOLUMN\n");
   printf("\t 1 \t 2 \t 3 \n");

   for (i=0; i<ROW; i++)
   {
       printf("ROW %d", i+1);
       for (j=0; j<COLUMN; j++)
          printf("\t %d", pinakas[i][j]);

       printf("\n");
   }

   system("PAUSE");
}
```

Παράδειγμα 9-2:

Το παρακάτω πρόγραμμα δέχεται τις τιμές ενός πίνακα δεκαδικών αριθμών διάστασης 5 x 5 και στη συνέχεια υπολογίζει το ελάχιστο στοιχείο στην κύρια και στη δευτερεύουσα διαγώνιο

```
#include <stdio.h>
#include <stdlib.h>
#define NR_ROW 5
#define NR_COL 5

main()
{
    int i, j;
    float mat[NR_ROW][NR_COL], min_maindg, min_secdg;
```

```
    printf("Input matrix elements:\n");
    for (i = 0; i < NR_ROW; i++)
    {
        for (j = 0; j < NR_COL; j++)
        {
            printf("Input element [%d][%d] : ", i, j);
            scanf("%f", &mat[i][j]);
        }
    }

    min_maindg = mat[0][0];  /* min el. is mat(0,0) so loop starts
from 1 */

    for (i = 1; i < NR_ROW; i++)
    {
        if (mat[i][i] < min_maindg)
        {
           min_maindg = mat[i][i];
        }
    }

    min_secdg = mat[0][NR_COL -1];

    for (i = 1; i < NR_ROW; i++)
    {
        if (mat[i][NR_COL-i-1] < min_secdg)
        {
           min_secdg = mat[i][NR_COL-i-1];
        }
    }

    printf("\nSmallest element in main diagonal is: %f",
min_maindg);
    printf("\nSmallest element in second diagonal is: %f",
min_secdg);

    system("PAUSE");
}
```

Παράδειγμα 9-3:

Το παρακάτω πρόγραμμα δέχεται τις ακέραιες τιμές ενός πίνακα ακέραιων αριθμών μέγιστης διάστασης 50 x 50 και στη συνέχεια υπολογίζει τον ανάστροφό του

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_ROW 50
#define MAX_COL 50

main()
{
    int i, j, m, n, temp, dim;
    int mat[MAX_ROW][MAX_COL];
```

```
/* variable dim  is set to smaller value of defined
   maximal number of rows and columns */
if (MAX_ROW < MAX_COL)
   dim = MAX_ROW;
else
   dim = MAX_COL;

do
{
    printf("Input number of rows < %d: ", dim);
    scanf("%d", &m);
    printf("Input number of columns < %d: ", dim);
    scanf("%d", &n);

}
while (m < 1 || m > dim || n < 1 || n > dim);

printf("Input of matrix elements :\n");
for (i = 0; i < m; i++)
{
   for (j = 0; j < n; j++)
   {
      printf("Input element [%d][%d] : ", i, j);
      scanf("%d", &mat[i][j]);
   }
}

printf("\nMatrix before transposing:\n");
for (i = 0; i < m; i++)
{
    for (j = 0; j < n; j++)
    {
        printf("%3d", mat[i][j]);
    }
    printf("\n");
}

for ( i = 0; i < m; ++i )  /* looping must start from i+1 */
{
    for ( j = i + 1; j < n; ++j )
    {
        temp = mat[i][j];
        mat[i][j] = mat[j][i];
        mat[j][i] = temp;
    }
}

printf("\nMatrix after transposing:\n");
for (i = 0; i < n; i++)
{
    for (j = 0; j < m; j++)
    {
```

```
            printf("%3d", mat[i][j]);
        }
        printf("\n");
    }

    system("PAUSE");
}
```

Παράδειγμα 9-4:

Το παρακάτω πρόγραμμα δέχεται τις τιμές ενός πίνακα δεκαδικών αριθμών διάστασης 5 x 5 και στη συνέχεια υπολογίζει το άθροισμα των στοιχείων για κάθε στήλη και το γινόμενο των στοιχείων για κάθε γραμμή. Το πρόγραμμα αποθηκεύει τα αθροίσματα και τα γινόμενα σε μονοδιάστατους πίνακες και στη συνέχεια παρουσιάζει το μικρότερο άθροισμα και το μεγαλύτερο γινόμενο μαζί με τον αριθμό της αντίστοιχης στήλης του αρχικού πίνακα

```
#include <stdio.h>
#include <stdlib.h>
#define NR_ROW 5
#define NR_COL 5

main()
{
   int i, j;
   int min_sum_ind, max_prod_ind;
   float mat[NR_ROW][NR_COL];
   float sum[NR_COL], prod[NR_ROW];

    for (i = 0; i < NR_ROW; i++)
    {
        for (j = 0; j < NR_COL; j++)
        {
            printf("Input element[%d][%d] : ", i, j);
            scanf("%f", &mat[i][j]);
        }
    }

    for (j = 0; j < NR_COL; j++)
    {
        sum[j] = 0;

        for (i = 0; i < NR_ROW; i++)
        {
           sum[j] += mat[i][j];
        }
    }

    for (i = 0; i < NR_ROW; i++)
    {
        prod[i] = 1;

        for (j = 0; j < NR_COL; j++)
        {
            prod[i] *= mat[i][j];
```

```
        }
    }

    min_sum_ind = 0;
    for (j = 1; j < NR_COL; j++)
    {
        if (sum[j] < sum[min_sum_ind])
        {
           min_sum_ind = j;
        }
    }

    max_prod_ind = 0;
    for (i = 1; i < NR_ROW; i++)
    {
        if (prod[i] > prod[max_prod_ind])
        {
            max_prod_ind = i;
        }
    }

    printf("\nSmallest sum: %f, parent index: %d\n",
sum[min_sum_ind], min_sum_ind);
    printf("\nBiggest product: %f, parent index: %d\n",
prod[max_prod_ind], max_prod_ind);

    system("PAUSE");
}
```

10ο Μέρος – Μαθηματικές συναρτήσεις

Μαθηματικές συναρτήσεις

Μπορούμε να χρησιμοποιήσουμε τις πιο γνωστές μαθηματικές συναρτήσεις που βρίσκονται στην βιβλιοθήκη math.h ώστε να μπορούμε να κάνουμε εύκολα και γρήγορα μαθηματικές πράξεις.

Η βιβλιοθήκη **math.h** δηλώνεται στο πρόγραμμά μας ως εξής: **`#include <math.h>`**

Συνάρτηση	Περιγραφή
`sqrt(x)`	Τετραγωνική ρίζα, χ>=0
`pow(x,y)`	x^y
`log(x)`	Λογάριθμος του χ, χ>0
`exp(x)`	e^x
`fabs(x)`	Απόλυτη τιμή του χ
`cos(x)`	Συνημίτονο χ
`sin(x)`	Ημίτονο χ
`tan(x)`	Εφαπτομένη χ

Παράδειγμα 10-1:

Το παρακάτω χρησιμοποιεί τις συναρτήσεις power, sqrt, log, cos και sin που βρίσκονται στην βιβλιοθήκη math.h για να υπολογίσουμε τις αντίστοιχες ποσότητες

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

main()
{
     int a, sqra, sqrta;
     float loga, cosa, sina;

     a = 4;

     sqra = pow(a,2);
     sqrta = sqrt(a);
     loga = log(a);
     cosa = cos(a);
     sina = sin(a);

     printf("the square of: %d is: %d\n", a, sqra);
     printf("the square root of: %d is: %d\n", a, sqrta);
     printf("the logarithm of: %d is: %f.\n", a, loga);
     printf("the cosinus of: %d is: %f.\n", a, cosa);
     printf("the sinus of: %d is: %f.\n", a, sina);

     system("PAUSE"); /* pause */
}
```

Παράδειγμα 10-2:

Το παρακάτω παράδειγμα υπολογίζει την περιφέρεια και το εμβαδόν ενός κυκλικού δίσκου με ακτίνα που του δίνουμε εμείς από το πληκτρολόγιο, χρησιμοποιώντας τη συνάρτηση power.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define PI 3.14159

main()
{
     int aktina;
     double perifereia, embadon;

     printf("Doste tin aktina toy kyklikoy diskoy:");
     scanf("%d", &aktina);

     perifereia = 2 * PI * aktina;
     embadon = PI * pow(aktina, 2);

     printf("O diskos me aktina: %d\n", aktina);
     printf("exei periferia: %.3f\n", perifereia);
     printf("kai embadon: %.3f\n", embadon);
```

```
     system("PAUSE"); /* pause */
}
```

Παράδειγμα 10-3:

Το παρακάτω παράδειγμα υπολογίζει την υποτείνουσα ενός τριγώνου χρησιμοποιώντας το Πυθαγόρειο θεώρημα καθώς και το εμβαδόν του τριγώνου χρησιμοποιώντας τη συνάρτηση sqrt.

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

main()
{
     float sidea, sideb, sidec, embadon;

     printf("Doste tin pleyra A toy orthogonioy trigonou:");
     scanf("%f", &sidea);
     printf("Doste tin pleyra B toy orthogonioy trigonou:");
     scanf("%f", &sideb);

     /* ypologismos pleyras c */
     sidec = sqrt(pow(sidea,2)+pow(sideb,2));

     /* ypologismos embadou */
     embadon = 0.5 * sidea * sideb;

     printf("To trigono me pleyres %.3f kai %.3f\n", sidea,
sideb);
     printf("exei ypotinousa: %.3f\n", sidec);
     printf("kai embadon: %.3f\n", embadon);

     system("PAUSE"); /* pause */
}
```

Παράδειγμα 10-4:

Το παρακάτω παράδειγμα υπολογίζει την μέγιστη θερμοκρασία της εβδομάδας κατά την απόλυτη τιμή χρησιμοποιώντας τη συνάρτηση ***fabs****.*

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define WEEK 7

main()
{
   float temp[WEEK];
   int i, max;

   printf("Doste tis thermokrasies tis ebdomadas:\n");
```

```
   for (i = 0; i < WEEK; i++)
   {
      printf("%d> ", i+1);
      scanf("%f", &temp[i]);
   }

   for (i = 0; i < WEEK; i++)
   {
      if ( fabs(temp[i]) > fabs(temp[max]))
         max = i;
   }

     printf("H megisti kata apolyto thermokrasia tis ebdomadas
einai h %.3f thn %d hmera\n", temp[max], max+1);

     system("PAUSE"); /* pause */
}
```

11ο Μέρος – Συναρτήσεις

Συναρτήσεις

Τι είναι οι συναρτήσεις
Οι συναρτήσεις είναι ένα βασικό δομικό συστατικό σε όλες τις ανώτερες γλώσσες προγραμματισμού. Με τις συναρτήσεις έχουμε τη δυνατότητα να χωρίσουμε εκτεταμένες εργασίες προγραμματισμού σε μικρότερες και παράλληλα μπορούμε να επαναχρησιμοποιήσουμε ή/και να οικοδομήσουμε πάνω σε αυτά που έχουν ετοιμάσει άλλοι προγραμματιστές δίχως να χρειάζεται να αρχίζουμε πάλι από το μηδέν. Γενικά τα προγράμματα της γλώσσας C αποτελούνται από πολλές και μικρές συναρτήσεις παρά από λίγες και μεγάλες. Ο βασικός στόχος της χρήσης συναρτήσεων είναι να δημιουργήσουμε συναρτήσεις για κάποιες τετριμμένες επαναλαμβανόμενες εργασίες ώστε να γίνεται κλήση σε κάποια συνάρτηση παρά «αντιγραφή και επικόλληση» κάποιου κομματιού κώδικα. Με αυτό τον τρόπο πετυχαίνουμε να έχουμε τον κώδικά μας σε ένα και μόνο σημείο πράγμα που διευκολύνει οποιαδήποτε διόρθωση, αλλαγή ή επέκταση του συγκεκριμένου κώδικα αλλά παράλληλα μπορούμε να «μεταφέρουμε» αλγόριθμους και λύσεις σε προβλήματα ανάμεσα στα προγράμματά μας, μιας και ο συγκεκριμένος κώδικας είναι απομονωμένος από το υπόλοιπο πρόγραμμα.

Περιγραφή-Σύνταξη
Μία συνάρτηση μας δίνει τη δυνατότητα να δημιουργήσουμε ένα πρόγραμμα που αποτελείται από βασικά δομικά συστατικά παρά από ένα ολόκληρο «κατεβατό» κώδικα. Οι συναρτήσεις διαχωρίζονται σε δύο κατηγορίες: σε αυτές που επιστρέφουν τιμές και σε αυτές που δεν επιστρέφουν κάποια τιμή. Ένα απλό παράδειγμα συνάρτησης που επιστρέφει μία τιμή είναι η μαθηματική συνάρτηση της τετραγωνικής ρίζας **`sqrt()`** που είδαμε στη βιβλιοθήκη **`math.h`**. Η συνάρτηση **`sqrt()`** επιστρέφει μία τιμή τύπου δεκαδικού αριθμού που αντιστοιχεί στην τετραγωνική ρίζα του αριθμού που θα χρησιμοποιήσουμε σαν όρισμα. Αντίθετα υπάρχουν συναρτήσεις οι οποίες δεν επιστρέφουν κάποια τιμή αλλά κάνουμε μια προκαθορισμένη εργασία, όπως είναι για παράδειγμα η συνάρτηση μορφοποιημένης εξόδου **`printf()`** η οποία διαμορφώνει ανάλογα με τις δικές μας απαιτήσεις την έξοδο στην οθόνη.

Η γενική σύνταξη μιας συνάρτησης στη γλώσσα C είναι η παρακάτω:

```
«Τύπος δεδομένων» Όνομα συνάρτησης (ορίσματα)
{
  Τοπικές μεταβλητές;
  εντολή 1;
  εντολή 2;
  . . . . .
  εντολή ν;
  return τιμή;
}
```

Οι τοπικές μεταβλητές είναι μεταβλητές όπως οι κανονικές μεταβλητές που γνωρίζουμε και χρησιμοποιούμε μέσα στη συνάρτηση **main()**, αλλά με τη διαφορά πως αυτές οι τοπικές μεταβλητές «υπάρχουν» μόνο μέσα στη συνάρτησή μας και δεν είναι «ορατές» οπουδήποτε αλλού. Ο τύπος δεδομένων είναι ο τύπος της τιμής που επιστρέφει η συνάρτησή μας και μπορεί να είναι οποιοσδήποτε από τους γνωστούς μας τύπους (int, float, char κλπ.). Αυτός ο τύπος δεδομένων θα πρέπει να συμφωνεί με την τιμή που θα επιστρέψει η συνάρτηση μέσω της εντολής **return** τιμή; Τα ορίσματα της συνάρτησης είναι ένας συνδυασμός τύπου δεδομένων και ονομάτων μεταβλητών οι οποίες χρησιμεύουν για τη μεταφορά των δεδομένων από την κλήση της συνάρτησης μέσα στη **main()** μέσα στη συνάρτησή μας. Η εντολή **return** είναι η εντολή η οποία επιστρέφει στο κύριο πρόγραμμα οποιοδήποτε αποτέλεσμα θέλουμε να επιστρέφει η συνάρτησή μας.

Παράδειγμα 11-1:

Το παρακάτω πρόγραμμα χρησιμοποιεί μια συνάρτηση για να προσθέσει δύο αριθμούς. Η συνάρτηση δέχεται σαν όρισμα 2 ακέραιους x και y και επιστρέφει έναν ακέραιο αριθμό.

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

int add (int x, int y)
{
    int z;
    z = x + y;
    return z;
}

main ()
{
    int i = 10, j = 20, k = 0;

    k = add(i, j);
    printf ("The value of k is %d\n", k);

    system("PAUSE"); /* pause */
}
```

Παράδειγμα 11-2:

Το παρακάτω πρόγραμμα χρησιμοποιεί μια συνάρτηση για να συγκρίνει ακέραιους 2 αριθμούς και επιστρέφει τη μεγαλύτερη τιμή από τους 2 αριθμούς.

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

```
#include <stdlib.h>

int compare(int x, int y)
{
    if ( x > y)
      return x;
    else
      return y;
}

main ()
{
    int num1, num2, max;

    printf("Dose ton 1o arithmo: ");
    scanf("%d", &num1);
    printf("Dose ton 2o arithmo: ");
    scanf("%d", &num2);

    max = compare(num1,num2);
    printf ("O megalyteros einai o %d\n", max);

    system("PAUSE"); /* pause */
}
```

Παράδειγμα 11-3:

Το παρακάτω πρόγραμμα χρησιμοποιεί μια συνάρτηση για να συγκρίνει 2 αριθμούς και επιστρέφει τη μεγαλύτερη τιμή. Η παραπάνω συνάρτηση καλείται 3 φορές για να επιτύχουμε τη σύγκριση 4 αριθμών.

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

int compare(int x, int y)
{
    if ( x > y)
      return x;
    else
      return y;
}

main ()
{
    int num1, num2, num3, num4, max;

    printf("Dose ton 1o arithmo: ");
    scanf("%d", &num1);
    printf("Dose ton 2o arithmo: ");
    scanf("%d", &num2);
    printf("Dose ton 3o arithmo: ");
    scanf("%d", &num3);
    printf("Dose ton 4o arithmo: ");
    scanf("%d", &num4);
```

```
    max = compare(num1,num2);
    max = compare(max,num3);
    max = compare(max,num4);

    printf ("O megalyteros einai o %d\n", max);

    system("PAUSE"); /* pause */
}
```

<u>Παράδειγμα 11-4</u>:

Το παρακάτω πρόγραμμα χρησιμοποιεί μια συνάρτηση για να συγκρίνει 3 αριθμούς και επιστρέφει ένα χαρακτήρα (A, B ή C) που αντιστοιχεί στο μεγαλύτερο αριθμό.

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

char compare(int x, int y, int z)
{
    if ( x > y)
    {
       if (x > z)
         return 'A';
       else
         return 'C';
    }
    else
    {
       if (y > z)
         return 'B';
       else
         return 'C';
    }
}

main ()
{
    int num1, num2, num3;
    char max;

    printf("Dose ton A arithmo: ");
    scanf("%d", &num1);
    printf("Dose ton B arithmo: ");
    scanf("%d", &num2);
    printf("Dose ton C arithmo: ");
    scanf("%d", &num3);

    max = compare(num1,num2, num3);

    printf ("O megalyteros einai o %c arithmos\n", max);

    system("PAUSE"); /* pause */
```

```
}
```

Παράδειγμα 11-5:

Το παρακάτω πρόγραμμα χρησιμοποιεί μια συνάρτηση για να αυξήσει όλους τους αριθμούς που βρίσκονται σε έναν πίνακα σύμφωνα με ένα ποσοστό που του δίνουμε εμείς.

```
#include <stdio.h>
#include <stdlib.h>
#define SIZE 5

float ayksisi(float pay, float value )
{
    return pay * value;
}

main ()
{
    float misthos[SIZE], raise;
    int i;

    printf("Dose tous misthoys ton ypalhlon: \n");
    for (i = 0; i < SIZE; i++)
    {
        printf("%2d> ", i+1);
        scanf("%f", &misthos[i]);
    }

    printf("Dose tin ayksisi san dekadiko arithmo: \n");
    scanf("%f", &raise);
    raise = 1 + raise/100; /* ypologismos pososToy ayksisis */

    for (i = 0; i < SIZE; i++)
    {
        misthos[i] = ayksisi(misthos[i], raise);
    }

    printf("Oi neoi misthoi einai: \n");
    for (i = 0; i < SIZE; i++)
    {
        printf("%2d> %.2f\n", i+1, misthos[i]);
    }
    system("PAUSE"); /* pause */
}
```

Παράδειγμα 11-6:

Το παρακάτω πρόγραμμα χρησιμοποιεί μια συνάρτηση για να εμφανίσει τετράγωνα όποιας διάστασης του δώσουμε εμείς. Η συνάρτηση δεν επιστρέφει τύπο δεδομένων γι' αυτό και δηλώνεται σαν να επιστρέφει τύπο void.

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

```
#include <stdlib.h>

void tetragono(int size)
{
    int i, j;

    for (i = 1; i <= size; i++)
    {
        for (j = 1; j <= size; j++)
        {
            printf("+");
        }
        printf("\n");
    }

    printf("telos tetragonou \n");
}

main ()
{
    int boxsize, number, i, j;
    char gramma;

    printf("Dose ton arithmo ton tetragonon: ");
    scanf("%d", &number);
    printf("Dose to megethos ton tetragonon: ");
    scanf("%d", &boxsize);

    for (i = 1; i <= number; i++)
    {
        tetragono(boxsize);
    }

    system("PAUSE"); /* pause */
}
```

Παράδειγμα 11-7:

Το παρακάτω πρόγραμμα χρησιμοποιεί μια συνάρτηση για να υπολογίσει το μέσο όρο 10 αριθμών. Προσέξτε ότι η συνάρτηση μπορεί να επιστρέψει μία (αυθαίρετα ορισμένη από εμάς) τιμή ακόμα και όταν εφαρμόσουμε μία διαίρεση με το μηδέν.

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

float mesos(long synolo, int diairetis)
{
      if (diairetis != 0)
         return (float)synolo/diairetis;
      else
         return 0;
}
```

```
main()
{
  int numbers[10];
  long sum = 0;
  float average = 0;
  int i;

  printf("\nEnter the 10 numbers:\n");

  for(i = 0; i < 10; i++)
  {
    printf("%2d> ",i+1);
    scanf("%d", &numbers[i]);
    sum += numbers[i];
  }

  average = mesos(sum, 10);

  printf("\nAverage of the ten numbers entered is: %f\n",
average);

  system("PAUSE"); /* pause */
}
```

Παράδειγμα 11-8:

Το παρακάτω πρόγραμμα χρησιμοποιεί μια συνάρτηση για να υπολογίσει τα έτη που χρειάζονται ώστε να φτάσει το υπόλοιπο ενός λογαριασμού ένα συγκεκριμένο ποσό, δεδομένου ενός σταθερού επιτοκίου που του έχουμε δώσει εμείς.

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

int xronia(float ypoloipo, float epitokio, float stoxos)
{
  float tokos;
  int eth = 0;

  do
  {
    tokos = ypoloipo * (epitokio/100);
    ypoloipo += tokos;
    eth++;
  }
  while (ypoloipo <= stoxos);

  return eth;
}

main()
{
  /* oi metavlites */
  float balance = 0, rate = 0, target = 0, interest = 0;
```

```
  int years = 0;

  /* eisagogi dedomenon */
  printf(" Doste to ypoloipo toy logariasmou: ");
  scanf("%f", &balance);
  printf(" Doste to stathero epitokio: ");
  scanf("%f", &rate);
  printf(" Doste to epithymito epitokio: ");
  scanf("%f", &target);

  /* ypologismon xronon */
  years = xronia(balance, rate, target);

  printf("xreiazontai %d xronia gia na ftasei to ypoloipo sto %.2f
\n", years, target);

  system("PAUSE"); /* pause */
}
```

<u>Παράδειγμα 11-9</u>:

Το παρακάτω πρόγραμμα χρησιμοποιεί μία συνάρτηση που δέχεται σαν όρισμα έναν πίνακα και επιστρέφει την τιμή του δείκτη που αντιστοιχεί στο στοιχείο με τη μεγαλύτερη τιμή.

```
#include <stdio.h>
#include <stdlib.h>
#define SIZE 10

int maximum(int pin[SIZE])
{
    int i, max = 0;

    for (i = 0; i < SIZE; i++)
    {
        if (pin[i] > pin[max])
          max = i;
    }

    return max;
}

main ()
{
    int pinakas[SIZE];
    int i, max;

    printf("Dose toys %d arithmous\n", SIZE);
    for (i = 0; i < SIZE; i++)
    {
        printf("%2d> ", i+1);
        scanf("%d", &pinakas[i]);
    }
```

```
    max = maximum(pinakas);

    printf("o megistos einai o %d arithmos me timi %d\n", max+1,
pinakas[max]);

    system("PAUSE"); /* pause */
}
```

Παράδειγμα 11-10:

*Το παρακάτω πρόγραμμα χρησιμοποιεί μια συνάρτηση για να υπολογίσει το τη δύναμη του βάρους υλοποιώντας τον ορισμό w=m*g, όπου το g δίδεται από μια σταθερά.*

```
#include <stdio.h>
#include <stdlib.h>
#define G 9.81

float force(float maza)
{
   float dyn;

   dyn = maza * G;

   return dyn;
}

main()
{
   float mass = 0, dynami = 0;

   printf(" Dose ti maza se kila: ");
   scanf("%f", &mass);

   dynami = force(mass);

   printf(" H dynami toy barous einai: %.2f \n", dynami);

   system("PAUSE");
}
```

12ο Μέρος – Αρχεία - Δομές

Αρχεία

Γενικά για τα αρχεία στη C

Μπορούμε να χρησιμοποιήσουμε αρχεία στη γλώσσα προγραμματισμού C αντί της πρότυπης εισόδου από το πληκτρολόγιο ή της πρότυπης εξόδου στην οθόνη, δηλαδή μπορούμε να διαβάζουμε δεδομένα για τους υπολογισμούς μας από ένα αρχείο καθώς επίσης και να αποθηκεύουμε τα αποτελέσματα της επεξεργασίας των δεδομένων σε ένα άλλο αρχείο. Μπορούμε να χρησιμοποιήσουμε αρχεία ανάλογα με το τι ενέργειες θέλουμε να κάνουμε αυτό, όπως μόνο για ανάγνωση, για εγγραφή (δημιουργία) ή για ενημέρωση των περιεχομένων των αρχείων.

Προσπέλαση αρχείων

Για προσπελάσουμε ένα αρχείο και τα περιεχόμενά του πρέπει να συνδέσουμε το εξωτερικό όνομα ενός αρχείου με τις εντολές της γλώσσας προγραμματισμού C που διαβάζουν ή γράφουν τα δεδομένα. Έτσι για να μπορεί να γραφτεί ή να διαβαστεί ένα αρχείο, πρέπει να «ανοιχτεί» με τη συνάρτηση βιβλιοθήκης **fopen.** Η συνάρτηση βιβλιοθήκης **fopen** επιστρέφει ένα δείκτη, που λέγεται δείκτης αρχείου, ο οποίος «δείχνει» μια δομή που περιέχει πληροφορίες για το αρχείο και μπορεί να χρησιμοποιηθεί στις επόμενες αναγνώσεις ή καταγραφές τους αρχείου.

Έτσι για να δηλώσουμε και να ανοίξουμε ένα αρχείο πρέπει να χρησιμοποιήσουμε τις παρακάτω δύο εντολές:

```
FILE *fp;
fp = fopen(name, mode);
```

Η πρώτη γραμμή **FILE *fp;** μας δηλώνει πως η fp είναι ένας δείκτης για τύπο FILE, ενώ η δεύτερη εντολή **fp = fopen(name, mode);** μας «ανοίγει» το αρχείο για να το χρησιμοποιήσουμε όπως επιθυμούμε.

Η εντολή **fopen(name, mode)** δέχεται δύο ορίσματα στη σύνταξή της. Το πρώτο όρισμα είναι ένα αλφαριθμητικό που περιέχει το όνομα του αρχείο. Το δεύτερο όρισμα είναι ο τρόπος που δείχνει πως σκοπεύουμε να χρησιμοποιήσουμε το αρχείο, στους επιτρεπτούς τρόπους συγκαταλέγεται η ανάγνωση **"r",** η εγγραφή **"w"** και η προσθήκη **"a".**

Αν ένα αρχείο που δεν υπάρχει, ανοιχτεί για εγγραφή ή για προσθήκη, τότε εφόσον είναι δυνατό δημιουργείται. Το άνοιγμα για γράψιμο ενός αρχείου που ήδη υπάρχει, απορρίπτει τα υπάρχοντα περιεχόμενα, ενώ το άνοιγμα για προσθήκη τα διατηρεί.

Για φορμαρισμένη είσοδο και έξοδο αρχείων μπορούμε να χρησιμοποιήσουμε τις συναρτήσεις **fscanf** και **fprintf**, οι οποίες είναι πανομοιότυπες με τις **scanf** και **printf** με τη διαφορά ότι το πρώτο όρισμα είναι ένας δείκτης αρχείου που καθορίζει ποιο αρχείο θα διαβαστεί ή θα γραφτεί.

Για να διακόψουμε τη σύνδεση ανάμεσα στο δείκτη αρχείου και το εξωτερικό όνομα που έχει καθοριστεί με τον **fopen**, ελευθερώνοντας το δείκτη αρχειου για άλλο αρχείο χρησιμοποιούμε την εντολή **fclose** η οποία είναι η αντίστροφη της **fopen**. Έτσι για να κλείσουμε ένα αρχείο που έχουμε προηγουμένως ανοίξει πρέπει να χρησιμοποιήσουμε την εντολή: **fclose(fp);**

Παράδειγμα 12-1:

*Το παρακάτω παράδειγμα γράφει στο αρχείο test.txt το κείμενο: «**1 asdfasdf 2**». Εάν το αρχείο test.txt δεν υπάρχει τότε θα δημιουργηθεί αυτόματα από το πρόγραμμά μας.*

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

main()
{
  FILE *fp;

  fp = fopen("test.txt","a");
```

```
  fprintf(fp,"%d %s %d\n",1,"asdfasdf",2);

  fclose(fp);

  system("PAUSE"); /* pause */
}
```

Παράδειγμα 12-2:

Το παρακάτω πρόγραμμα καταγράφει τις τιμές της ημερήσιας είσπραξης ενός καταστήματος, δίχως να γνωρίζουμε εκ των προτέρων το πλήθος των ημερών αλλά έως ότου ο χρήστης εισάγει αρνητικό αριθμό. Στη συνέχεια μας γράφει σε ένα αρχείο τη συνολική είσπραξη καθώς και τις ημέρες με τη μεγαλύτερη και τη μικρότερη είσπραξη.

```
#include <stdio.h>
#include <stdlib.h>
#define ETOS 365

main()
{
  float eispraksi[ETOS];
  float value;
  double synolo = 0.0;
  int orio, min, max, i;
  FILE *fp;

  i = 0;
  orio = 0;

  printf("Doste tin imerisia eispraksi h doste -1 gia telos:\n");
  while (i < ETOS)
  {
     printf("imera %d =  ", i + 1);
     scanf("%f", &value);

     if (value < 0)
       break;

     eispraksi[i] = value;
     synolo += eispraksi[i];
     orio++;
     i++;
  }

  min = 0;
  max = 0;

  for (i = 0; i < orio; i++)
  {
     if (eispraksi[i] < eispraksi[min])
       min = i;

     if (eispraksi[i] > eispraksi[max])
```

```
      max = i;
  }

  fp = fopen("out.txt", "a");

  fprintf(fp, "\n - - - - - - \n");
  fprintf(fp, "H synoliki eispraksi einai %.2f\n", synolo);
  fprintf(fp, "H megalyteri eispraksi einai %.2f thn imera %d\n",
eispraksi[max], max + 1 );
  fprintf(fp, "H mikroteri eispraksi einai %.2f thn imera %d\n",
eispraksi[min], min + 1 );

  fclose(fp);

  system("PAUSE"); /* pause */
}
```

Παράδειγμα 12-3:

Το παρακάτω πρόγραμμα διαβάζει από ένα αρχείο τις τιμές της ημερήσιας παραγωγής ενός προϊόντος σε μηνιαία βάση και στη συνέχεια υπολογίζει τη συνολική μηνιαία παραγωγή, το μέσο όρο της ημερήσιας παραγωγής, καθώς επίσης παρουσιάζει τις ημέρες (και την αντίστοιχη) παραγωγή που είναι μικρότερη του 80% του μέσου όρου.

```
#include <stdio.h>
#include <stdlib.h>
#define MINAS 30

main()
{
  int paragogi[MINAS];
  int synolo = 0, i;
  float mo = 0.0;
  FILE *fp;

  printf("Anagnosi arxeioy me tin imerisia paragogi.\n");

  fp = fopen("input.txt", "r");

  for (i = 0; i < MINAS; i++)
  {
     fscanf(fp, "%d", &paragogi[i]);
     synolo += paragogi[i];
  }

  printf("\n - - - - - - \n");
  printf("H synoliki paragogi toy mina einai %d\n", synolo);

  mo = (float)synolo/MINAS;
  printf("O mesos oros tis imerisias paragogis einai: %.2f \n",
mo);

  printf("Hmeres me paragogi < 80% tou mesou orou:\n");
```

```
  for (i = 0; i < MINAS; i++)
  {
     if (paragogi[i] < 0.8*mo)
          printf("Hmera %d me paragogi: %d \n", i+1, paragogi[i]);
  }

  fclose(fp);

  system("PAUSE"); /* pause */
}
```

Δομές

Τι είναι οι δομές
Μπορούμε να συνδυάσουμε τις απλές γνωστές μας μεταβλητές σε σύνθετες δομές, σαν ένα «πακέτο» που λέγεται structure. Με αυτό τον τρόπο μπορούμε να συμβολίσουμε παρόμοιας σημασίας αντικείμενα χρησιμοποιώντας παρόμοιες μεταβλητές σε μιας μορφής «πακέτου» και κατά συνέπεια να έχουμε τη δυνατότητα να περιγράφουμε τα διάφορα «χαρακτηριστικά» μιας οντότητας, κάτι που δεν γινόταν (εύκολα) χρησιμοποιώντας απλές μεταβλητές ή πίνακες.

Σύνταξη
Για να δηλώσουμε μία δομή χρησιμοποιούμε την δεσμευμένη λέξη **struct** πριν το όνομα της δομής, όπως κάναμε με της συνηθισμένες απλές μεταβλητές, και στη συνέχεια μέσα σε ένα μπλοκ δηλώνουμε τον τύπο δεδομένων και το όνομα του κάθε συστατικού της δομής μας. Για παράδειγμα για να δηλώσουμε μία δομή που θα περιγράφει το όνομα, το επώνυμο και το βαθμό ενός μαθητή θα πρέπει πρώτα να δηλώσουμε (εκτός της βασικής συνάρτησης main) το παρακάτω struct:

```
struct student
{
  char *fname;
  char *lname;
  float mark;
}
```

Η δομή μας αποτελείται από 3 χαρακτηριστικά: 2 τύπου char που αποθηκεύουν το όνομα και το επώνυμο ενός μαθητής και 1 τύπου float που αποθηκεύει το δεκαδικό βαθμό του συγκεκριμένου μαθητή. Η παραπάνω δομή δεν δημιουργεί καμία «μεταβλητή» μαθητή αλλά ένα «τύπο» μαθητή, για να χρησιμοποιήσουμε αυτό τον τύπο μαθητή θα πρέπει να δηλώσουμε μια μεταβλητή αυτού του τύπου μέσα στη βασική συνάρτηση main με παρόμοιο τρόπο με αυτόν που δηλώνουμε τις απλές μεταβλητές στα προγράμματά μας.

Έτσι μέσα την συνάρτηση main θα πρέπει να υπάρχει μια δήλωση:

```
struct student mathitis1, mathitis2;
```

Με αυτό τον τρόπο έχουμε δηλώσει 2 μεταβλητές για τους μαθητές (**mathitis1** και **mathitis2**) που σαν τύπο δεδομένων έχουν τη δομή student. Για να αναφερθούμε στα χαρακτηριστικά του structure πρέπει να χρησιμοποιήσουμε το όνομα της μεταβλητής τύπου δομής και στη συνέχεια να χρησιμοποιήσουμε το όνομα του συγκεκριμένου χαρακτηριστικού ενώνοντας τα δύο χαρακτηριστικά με μία τελεία.

Έτσι για να αναφερθούμε στο χαρακτηριστικό fname του mathitis1 πρέπει να γράψουμε την παρακάτω γραμμή κώδικα:

```
mathitis1.fname = "Tom";
```

Με παρόμοιο τρόπο μπορούμε να χρησιμοποιήσουμε τα χαρακτηριστικά όπως ακριβώς χρησιμοποιούσαμε τις μεταβλητές, προσέχοντας πάντα το τύπο δεδομένων του κάθε συστατικού. Για παράδειγμα μπορούμε να έχουμε μέσα στο πρόγραμμά μας τις παρακάτω εντολές:

```
if (mathitis1.mark >= 50)
  printf("mpravo %s perases!\n", mathitis1.fname);
else
  printf("lypamai %s den perases!\n", mathitis1.fname);
```

Παράδειγμα 12-4:

Το παρακάτω πρόγραμμα χρησιμοποιεί ένα structure για να αποθηκεύσει το όνομα, το επώνυμο και το βαθμό ενός μαθητή και στη συνέχεια ελέγχει και τυπώνει ένα μήνυμα ανάλογα με το βαθμό του μαθητή.

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

struct student
{
  char *fname;
  char *lname;
  float mark;
}

main()
{
  struct student mathitis1;

  mathitis1.fname = "nikos";
  mathitis1.lname = "michalodimitrakis";
  mathitis1.mark = 88.9;

  printf("\n - - - - - - \n");

  printf("onoma    mathiti:    %s    %s    \n",    mathitis1.fname,
mathitis1.lname);
  printf("bathmos: %.2f \n", mathitis1.mark);

  if (mathitis1.mark >= 50)
     printf("mpravo %s perases!\n", mathitis1.fname);
  else
     printf("lypamai %s den perases!\n", mathitis1.fname);

  system("PAUSE"); /* pause */
}
```

Παράδειγμα 12-5:

Το παρακάτω πρόγραμμα διαβάζει τα στοιχεία των μαθητών από ένα αρχείο και υπολογίζει τον μέσο όρο των βαθμών. Στη συνέχεια παρουσιάζει τους μαθητές με βαθμό < 50.

```
#include <stdio.h>
#include <stdlib.h>
#define SIZE 8

struct student
{
  int no;
  float mark;
}

main()
{
  int i;
  float sum = 0, average = 0;
  struct student mathites[SIZE];
  FILE *fp;

  clrscr();  /* clear screen */

  printf("Anagnosi arxeioy me stoixeia mathiton\n");

  fp = fopen("studin.txt", "r");

  for (i = 0; i < SIZE; i++)
  {
     fscanf(fp, "%d", &mathites[i].no);
     fscanf(fp, "%f", &mathites[i].mark);
     sum += mathites[i].mark;
  }

  printf("\n - - - - - - \n");

  average = (float)sum/SIZE;
  printf("O mesos oros ton bathmon einai: %.2f \n", average);

  printf("Oi mathites poy exoun bathmo < 50 einai:\n");

  for (i = 0; i < SIZE; i++)
  {
     if (mathites[i].mark < 50)
     {
          printf("- - - - - - - - \n");
          printf("Arithmos Mitrou: %d\n", mathites[i].no);
          printf("Bathmos: %.2f\n", mathites[i].mark);
     }
  }

  fclose(fp);
```

```
  system("PAUSE"); /* pause */
}
```