Κεφάλαιο ΙΙ:

Εισαγωγή στη γλώσσα προγραμματισμού C.

2.1 Η γλώσσα C.

Η C είναι μια γλώσσα προγραμματισμού υψηλού επιπέδου η οποία αναπτύχθηκε για πρώτη φορά στις αρχές της δεκαετίας του ’70 από τον Dennis Ritchie στα AT&T Bell Labs. Η σημερινή μορφή της γλώσσας ακολουθεί το πρότυπο ANSI (American National Standards Institute) γιαυτό και φέρει την ονομασία “ANSI C”. Ως γλώσσα υψηλού επιπέδου παρουσιάζει τα παρακάτω πλεονεκτήματα:

- **Αναγνωσιμότητα** : Τα προγράμματα διαβάζονται εύκολα.
- **Συντήρηση** : Τα προγράμματα είναι εύκολο να συντηρηθούν.
- **Μεταφερσιμότητα** : Τα προγράμματα εύκολα μεταφέρονται σε διαφορετικά λειτουργικά συστήματα και μηχανές.

Κάθε γλώσσα υψηλού επιπέδου χρειάζεται έναν **μεταγλωττιστή** (**compiler**) ο οποίος αναλαμβάνει να μεταφράσει τις γραμμές ενός προγράμματος σε **γλώσσα μηχανής**. Είναι η γλώσσα μηχανής αυτή η οποία γίνεται αντιληπτή από έναν Η/Υ. Με αυτόν τον τρόπο ένα πρόγραμμα αναπτύσσεται πρώτα σε γλώσσα προγραμματισμού C στη συνέχεια μεταγλωττίζεται σε γλώσσα μηχανής και εκτελείται από τον Η/Υ. Την χρήση του μεταγλωττιστή θα την δούμε σε επόμενη παράγραφο

2.2 Το πρώτο απλό πρόγραμμα C.

Στο περιβάλλον επιφάνειας εργασίας μας ανοίγουμε μία κονσόλα και δίνουμε την εντολή:

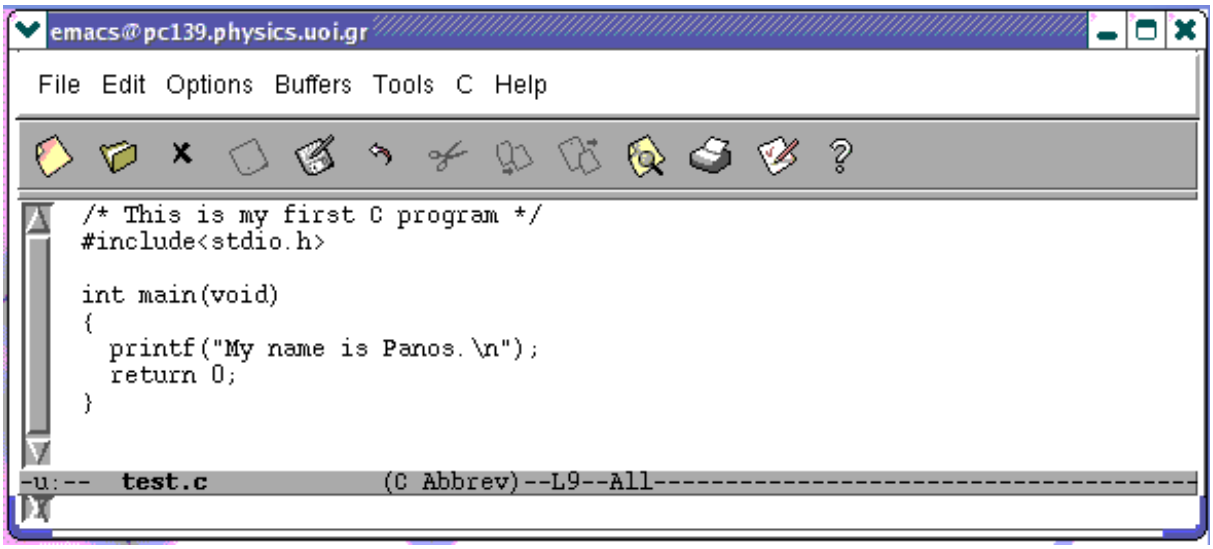
emacs example1.c &

Με αυτόν τον τρόπο ανοίγουμε ένα παράθυρο στον Emacs κάτω από το όνομα example1.c. Στη συνέχεια δακτυλογραφούμε το ακόλουθο πρόγραμμα (σχήμα 2.1):

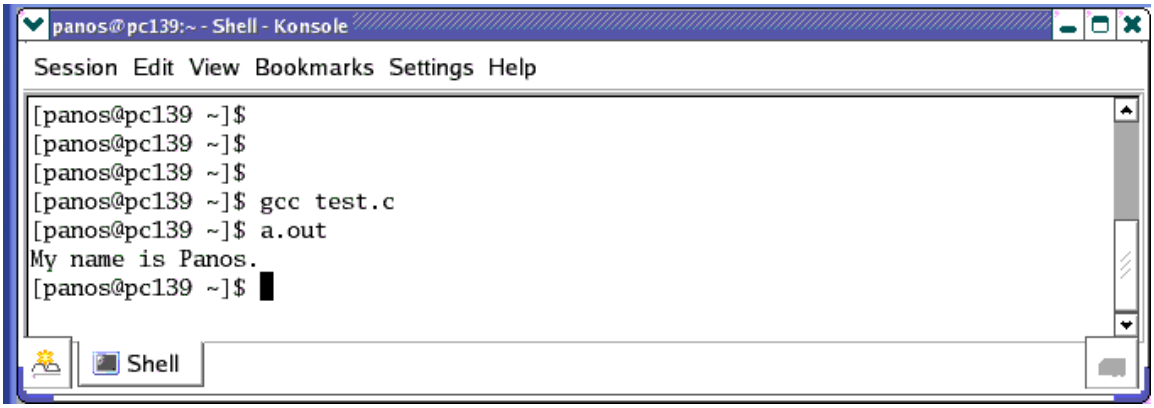
Όταν τελειώσουμε την δακτυλογράφηση αποθηκεύουμε το αρχείο μας εκτελώντας την εντολή File-Save (Current Buffer) (σχήμα 1.9). Στη συνέχεια για να εκτελέσουμε το πρόγραμμά μας πρέπει να το **μεταγλωττίσουμε** (**compilation**). Αυτό γίνεται εκτελώντας την ακόλουθη εντολή:

gcc example1.c

Η παραπάνω εντολή κάνει την μεταγλώττιση και δημιουργεί το εκτελέσιμο αρχείο **a.out**. Το αρχείο αυτό περιέχει την μεταγλώττιση, του απλού προγράμματός μας, σε γλώσσα μηχανής έτοιμη να εκτελεστεί. Η εκτέλεση του αρχείου γίνεται απλά γράφοντας το όνομά του και πατώντας το Enter. Το αποτέλεσμα της εκτέλεσης εικονίζεται στο παρακάτω σχήμα 2.2:



Σχήμα 2.1: Ένα απλό πρόγραμμα σε C.



Σχήμα 2.2: Η εκτέλεση του απλού προγράμματος.

Στις επόμενες παραγράφους αναλύουμε τις απλές εντολές οι οποίες εμφανίζονται στο παραπάνω πρόγραμμα και τον μεταγλωττιστή.

2.3 Ανάλυση του απλού προγράμματος C.

Αναλύοντας μία-μία τις γραμμές του προγράμματος του σχήματος 2.1 παρατηρούμε τα ακόλουθα :

- **Σχόλια**. Η πρώτη γραμμή περιέχει ένα σχόλιο:
  /* This is my first C program */

Παρατηρήστε ότι αυτή η γραμμή ξεκινά με έναν συνδυασμό καθέτου και αστερίσκου, **/*** και τελειώνει με ***/**. Στη C, το /* ονομάζεται **σημάδι αρχής σχολίου** και το */ είναι το **σημάδι τέλους σχολίου**. Ο C μεταγλωττιστής αγνοεί οτιδήποτε μεταξύ του αρχικού και του τελικού σημαδιού.

- **Η εντολή #include και τα αρχεία επικεφαλίδων**. Η δεύτερη γραμμή του προγράμματος είναι η ακόλουθη:

  #include<stdio.h>

  Η γραμμή αυτή ξεκινά με ένα #, το οποίο ακολουθείται από την εντολή include. Το include είναι μια εντολή του **προεπεξεργαστή** (ενός προγράμματος το οποίο κάνει κάποια προετοιμασία πριν μεταγλωττιστεί ο κώδικας) η οποία ψάχνει στην διαδρομή **/usr/include** να βρει το αρχείο stdio.h. Επί πλέον η εντολή include ζητά από τον προεπεξεργαστή να τοποθετήσει το αρχείο stdio.h στη θέση της εντολής μέσα στο πρόγραμμα.

  Τα αρχεία τα οποία περιλαμβάνονται με την εντολή #include, όπως το stdio.h, ονομάζονται **αρχεία επικεφαλίδων** και περιέχουν μια πληθώρα συναρτήσεων έτοιμων προς χρήση για τον προγραμματιστή. Αποτελούν δηλαδή τις **βιβλιοθήκες** συναρτήσεων οι οποίες είναι διαθέσιμες στον προγραμματιστή. Εκτός από το stdio.h υπάρχουν και άλλα αρχεία επικεφαλίδων, όπως τα math.h, stdlib.h, string.h κτλ. Κάθε φορά ο προγραμματιστής πρέπει να περιλαμβάνει τα απαραίτητα αρχεία επικεφαλίδων στην αρχή του κώδικά του.

- Η **συνάρτηση main()** αποτελεί μια πολύ ειδική συνάρτηση της C. Κάθε πρόγραμμα πρέπει να περιέχει μία και μόνο μία συνάρτηση main() η οποία εκτελείται πάντα πρώτη ακόμη και εάν είναι στο κάτω μέρος του προγράμματος. Στο συγκεκριμένο παράδειγμα η συνάρτηση main έχει δηλωθεί ως ακέραιου τύπου (**int**) ενώ δεν περιέχει κάποιο όρισμα (**void**). Περισσότερα για τις συναρτήσεις παρουσιάζονται σε επόμενη παράγραφο.

- Η **συνάρτηση printf()** εμφανίζει και τυπώνει μηνύματα στην οθόνη μας. Ο χαρακτήρας \n ο οποίος εμφανίζεται στο τέλος του μηνύματος λέει στον υπολογιστή να μετακινήσει τον δρομέα στην αρχή της επόμενης γραμμής. Η συνάρτηση printf() ορίζεται στο αρχείο επικεφαλίδας stdio.h. Αυτός είναι και ο λόγος που περιλαμβάνουμε το συγκεκριμένο αρχείο επικεφαλίδας με την εντολή #include στην κορυφή του προγράμματος.

- Η **εντολή return**. Όλες οι συναρτήσεις της C μπορούν να επιστρέφουν τιμές. Στο συγκεκριμένο παράδειγμα η συνάρτηση main η οποία έχει δηλωθεί ως ακέραια συνάρτηση επιστέφει την τιμή μηδέν και με αυτόν τον τρόπο το πρόγραμμα τερματίζεται κανονικά.

2.4 Ο μεταγλωττιστής (compiler).

Στο Linux που χρησιμοποιούμε είναι διαθέσιμοι δύο μεταγλωττιστές για τη γλώσσα C, ο **cc** και ο **gcc**. Ο **cc** αποτελεί τον πιο απλό μεταγλωττιστή που συνοδεύει το σύστημα ενώ ο **gcc** αποτελεί τον GNU μεταγλωττιστή του Ιδρύματος Ελευθέρου Λογισμικού (Free Software Foundation). Ο μεταγλωττιστής gcc παρέχει σαφώς μεγαλύτερες δυνατότητες και αποτελεί τον καθιερωμένο μεταγλωττιστή για το Linux. Η εντολή

gcc example1.c

όπως εξηγήσαμε δημιουργεί το εκτελέσιμο αρχείο a.out. Εάν θέλουμε να δώσουμε στο εκτελέσιμο αρχείο ένα άλλο όνομα , πχ. το όνομα ex1 εκτελούμε την ακόλουθη εντολή:

gcc –o ex1 example1.c

Με αυτόν τον τρόπο το εκτελέσιμο αρχείο μας τώρα έχει το όνομα ex1 και όχι το a.out.

Ας εξηγήσουμε τώρα με περισσότερες λεπτομέρειες την διαδικασία μεταγλώττισης ενός προγράμματος. Ξεκινάμε με ένα πρόγραμμα το οποίο είναι γραμμένο σε γλώσσα C, και το οποίο ονομάζεται **πηγαίος κώδικας**. Το όνομα αυτού του πηγαίου κώδικα τελειώνει με την επέκταση **.c**.

Στη συνέχεια εκτελείται η εντολή gcc, όπως αναφέραμε παραπάνω, και η οποία περιλαμβάνει τις εξής τρεις διεργασίες:

- Ο **προεπεξεργαστής** είναι ένα πρόγραμμα το οποίο κάνει κάποια προετοιμασία πριν μεταγλωττιστεί ο πηγαίος κώδικας. Ποιο συγκεκριμένα ψάχνει στην διαδρομή **/usr/include** βρίσκει τα συγκεκριμένα αρχεία επικεφαλίδας και τα συμπεριλαμβάνει στον κώδικα.

- Ο **μεταγλωττιστής** είναι ένα πρόγραμμα το οποίο λαμβάνει τον κώδικα από τον προεπεξεργαστή και δημιουργεί ένα αρχείο το οποίο ονομάζεται **αντικειμενικό αρχείο (object file)**. Τα αντικειμενικά αρχεία έχουν την επέκταση **.o**. Τα αρχεία αυτά δεν εκτελούνται επειδή υπάρχει κάποιος κώδικας ο οποίος λείπει. Θα πρέπει να γίνει το επόμενο βήμα: η σύνδεση.

- Η **σύνδεση** γίνεται καλώντας ένα ειδικό πρόγραμμα το οποίο ονομάζεται **linker** (πρόγραμμα σύνδεσης) και που περιέχεται μέσα στον μεταγλωττιστή. Η σύνδεση χρησιμοποιείται για να συνδεθούν το αντικειμενικό αρχείο, η τυπική ANSI C βιβλιοθήκη και άλλες βιβλιοθήκες που έχει δημιουργήσει ο χρήστης, για να δημιουργηθεί το εκτελέσιμο αρχείο – ο δυαδικός κώδικας. Σε αυτή τη φάση, συνδυάζεται ο δυαδικός κώδικας των συναρτήσεων που καλούνται στον πηγαίο κώδικα με το αντικειμενικό αρχείο. Το αποτέλεσμα αποθηκεύεται σε ένα νέο αρχείο – το **εκτελέσιμο αρχείο (executable file)**.

Εδώ σημειώνουμε πως το αντικειμενικό αρχείο και το εκτελέσιμο αρχείο εξαρτώνται και τα δύο από την αρχιτεκτονική του υπολογιστή. Δεν συμβαίνει το ίδιο για τον πηγαίο κώδικα ο οποίος εφ’ όσον έχει γραφεί σε ANSI C είναι ανεξάρτητος από τον υπολογιστή και μπορεί να μεταφέρεται όπως είναι.

Εάν ο χρήστης δεν επιθυμεί να δημιουργήσει εκτελέσιμο αρχείο αλλά να σταματήσει στην δημιουργία του αντικειμενικού αρχείου αυτό γίνεται με την εντολή:

gcc –c filename.c : Οπότε και δημιουργείται το αρχείο filename.o.

Ας υποθέσουμε πως το αρχείο filename.c περιέχει χρήσιμες συναρτήσεις για το πρόγραμμα το οποίο βρίσκεται στο αρχείο main.c. Τότε κατά την μεταγλώττιση του main.c πρέπει να συνδεθεί και το αρχείο filename.o. Αυτό γίνεται ως εξής:

gcc –o pgm main.c filemane.o : οπότε το αρχείο pgm αποτελεί το εκτελέσιμο αρχείο μας.

2.5 Τα γενικά σε ένα πρόγραμμα C.

Στην παράγραφο αυτή δίνουμε τους ορισμούς σε βασικά στοιχεία της γλώσσας όπως μεταβλητές, σταθερές, αριθμητικοί τελεστές, παραστάσεις και εντολές.

- Οι **μεταβλητές** είναι οντότητες οι οποίες μπορούν να παίρνουν διαφορετικές τιμές. Επίσης δύνανται να αλλάζουν τιμές στη ροή του προγράμματος. Για παράδειγμα, σκεφτείτε τις παρακάτω τέσσερις γραμμές:

  a = 1.35;
  b = 32;
  a = 7.5;
  b = 10;

  Σε αυτές ορίζονται δύο μεταβλητές οι a και b οι οποίες αρχικοποιούνται στις τιμές 1.35 και 32 αντίστοιχα και στη συνέχεια αλλάζουν τιμές. Ως ονόματα μεταβλητών μπορούμε να χρησιμοποιήσουμε μονολεκτικές λέξεις χρησιμοποιώντας πεζούς, κεφαλαίους χαρακτήρες και αριθμούς. Απαγορεύονται τα ονόματα τα οποία είναι δεσμευμένα από την γλώσσα όπως main, return,for κτλ. Επίσης απαγορεύεται η χρήση χαρακτήρων όπως (, ; ’ “ ? κτλ).

- Οι **σταθερές** είναι οντότητες οι οποίες αναφέρονται σε σταθερές τιμές, οι τιμές των οποίων δεν μπορούν να μεταβάλλονται από κανένα πρόγραμμα. Η χρήση των σταθερών είναι φυσιολογική και διαισθητική. Παραδείγματα σταθερών είναι οι αριθμοί 100 και 3.1415 όπως και οι χαρακτήρες ‘k’, ‘u’ κτλ.

- Στη C έχουν οριστεί οι παρακάτω **αριθμητικοί τελεστές** οι οποίοι μας επιτρέπουν να κάνουμε πράξεις:

Τελεστής	**Σημασία**
+	Πρόσθεση
-	Αφαίρεση
*	Πολλαπλασιασμός
/	Διαίρεση
%	Υπόλοιπο διαίρεσης

  Ο τελευταίος τελεστής αφορά το υπόλοιπο της διαίρεσης δύο ακεραίων πχ. 7%5 ισούται με 2.

- Μία **παράσταση** είναι ένας συνδυασμός σταθερών, μεταβλητών, τελεστών ή και συναρτήσεων. Οι παραστάσεις χρησιμοποιούνται για να δηλώσουν **υπολογισμούς**. Για παράδειγμα, σκεφτείτε τις παρακάτω γραμμές:

```
a = 10;
b = 5;
c = (a+5)*b;
```

  Ο όρος (a+5)*b αποτελεί μια παράσταση. Η μεταβλητή c λαμβάνει την τιμή 75.

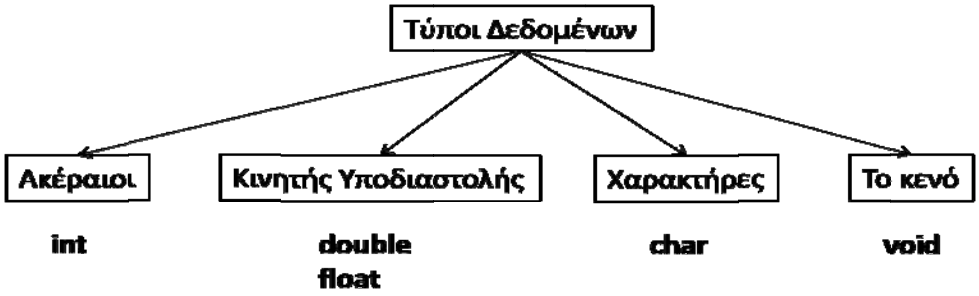
- Μία **εντολή** είναι μια οδηγία η οποία τελειώνει πάντοτε με ένα **ερωτηματικό**. Στο προηγούμενο παράδειγμα η γραμμή

```
c = (a+5)*b;
```

  αποτελεί μια εντολή. Η τιμή 75 η οποία υπολογίζεται από την παράσταση (a+5)*b προσδίδεται στην μεταβλητή c.

2.6 Οι τύποι δεδομένων.

Στο σχήμα 2.3 εικονίζονται οι τέσσερις βασικές κατηγορίες δεδομένων που συναντούμε στη C καθώς και οι **πέντε βασικοί τύποι δεδομένων**. Οι πέντε αυτοί τύποι χαρακτηρίζουν τις μεταβλητές τις οποίες ορίζουμε στα προγράμματά μας.



Σχήμα 2.3: Οι τέσσερις βασικές κατηγορίες δεδομένων που συναντούμε στη C καθώς και οι πέντε βασικοί τύποι δεδομένων.

Σε έναν υπολογιστή διαχωρίζουμε τους αριθμούς σε δύο κατηγορίες στους **ακεραίους** και σε αυτούς που δύνανται να έχουν δεκαδικά ψηφία. Οι τελευταίοι ονομάζονται αριθμοί **Κινητής Υποδιαστολής**. Ο λόγος αυτού του διαχωρισμού έγκειται στο ότι ο υπολογιστής κωδικοποιεί διαφορετικά του ακεραίους αριθμούς από αυτούς που διαθέτουν δεκαδικά ψηφία. Με αυτό τον τρόπο προκύπτουν δύο κατηγορίες αριθμητικών δεδομένων.

- **Ακέραια Δεδομένα:** Περιγράφονται με το πρόθεμα **int**. Ο τύπος int αναφέρεται σε μεταβλητές ή σταθερές οι οποίες μπορούν να διαχειριστούν ακεραίους αριθμούς. Στους περισσότερους υπολογιστές ένας ακέραιος int έχει μέγεθος 32 bit (4 byte). Αυτό σημαίνει πως το εύρος των ακεραίων αριθμών είναι από 2147483647 (δηλαδή 2^{31}-1) έως –2147483648.
  Παράδειγμα δήλωσης και αρχικοποίησης δύο ακεραίων μεταβλητών:

```
int a,b;
a=5;
b=7;
```

- **Δεδομένα Κινητής Υποδιαστολής:** Περιγράφονται από τα προθέματα **double** και **float**. Οι τύποι double και float αναφέρονται σε μεταβλητές ή σταθερές οι οποίες μπορούν να διαχειριστούν αριθμούς κινητής υποδιαστολής (αριθμούς με δεκαδικά ψηφία). Ο τύπος float έχει μέγεθος 32 bit (4 byte) ενώ ο τύπος double 64 bit (8 byte). Ο τύπος float έχει τη δυνατότητα να συγρατήσει αριθμούς με εύρος από $\pm 1.4 \times 10^{-45}$ έως $\pm 3.4 \times 10^{-38}$ και έως 7 σημαντικά ψηφία συνολικά. Ο τύπος double έχει τη δυνατότητα να συγκρατήσει εύρος αριθμών από $\pm 4.9 \times 10^{-324}$ έως $\pm 1.8 \times 10^{-308}$ και έως 15 σημαντικά ψηφία συνολικά. Εδώ πρέπει να τονίσουμε πως σε αντίθεση με το ανθρώπινο μυαλό ο υπολογιστής δεν έχει άπειρη ακρίβεια στην αποθήκευση και διαχείριση των αριθμητικών δεδομένων. Είναι μία μηχανή *πεπερασμένης ακρίβειας*. Παρ' όλα αυτά συνήθως στους υπολογισμούς μας δεν χρειαζόμαστε ακρίβεια μεγαλύτερη από αυτή που παρέχει ο τύπος double, τον οποίο και θα πρέπει να χρησιμοποιούμε κατά κόρον. Ο τύπος double αναφέρεται συνήθως και ως διπλής ακρίβειας για ευνόητους λόγους.
  Παράδειγμα δήλωσης και αρχικοποίησης δύο μεταβλητών κινητής υποδιαστολής:

```
double x;
float y;
x=3.14159;
y=7.;
```

  <u>Προσοχή!</u> Παρ' ότι η μεταβλητή y αρχικοποιείται σε έναν "ακέραιο" αριθμό τον 7, καλό είναι ΠΑΝΤΟΤΕ να βάζουμε και μια τελίτσα μετά το 7 ώστε να δηλώνουμε κατευθείαν πως είναι κινητής υποδιαστολής.

- **Δεδομένα Χαρακτήρων:** Περιγράφονται με το πρόθεμα **char**. Ο τύπος char αντιπροσωπεύει ένα χαρακτήρα από τον σύνολο των χαρακτήρων οι οποίοι χρησιμοποιούνται στον υπολογιστή μας. Ένας υπολογιστής τύπου PC περιέχει ένα σύνολο από 256 χαρακτήρες. Στο σύνολο των χαρακτήρων για παράδειγμα περιέχονται όλα τα γράμματα της αλφαβήτου (A έως Z και a έως z), όλα τα ψηφία (0 έως 9), όλοι οι χαρακτήρες τους οποίους βλέπουμε στο πληκτρολόγιό μας. Στον υπολογιστή μας ο κάθε χαρακτήρας καταλαμβάνει μέγεθος ίσο με **8 bit** (1 byte). Σε κάθε λοιπόν χαρακτήρα αντιστοιχείται ένας αριθμός των 8 bit, δηλαδή από το 0 (2^0-1) έως το 255 (2^8-1) ο οποίος ονομάζεται **ASCII** (American Standard Code for Information Interchange) κωδικός. Αυτό που πρέπει να γίνει κατανοητό είναι πως όταν πληκτρολογούμε για παράδειγμα τον χαρακτήρα A, αυτός για τον υπολογιστή μας αντιστοιχεί στην αριθμητική τιμή 65 (ή σε δεκαεξαδική μορφή 0x41), ο χαρακτήρας a αντιστοιχεί στην αριθμητική τιμή 97 (ή σε δεκαεξαδική μορφή 0x61). Στον Πίνακα 2.1 εικονίζεται το σετ των κωδικών ASCII για τους βασικούς 128 χαρακτήρες του υπολογιστή.

- **Το κενό:** Η δεσμευμένη λέξη **void** είναι ένας προσδιοριστής τύπου και χρησιμοποιείται κυρίως για την δήλωση συναρτήσεων οι οποίες δεν επιστρέφουν τιμές. Χρησιμοποιείται επίσης για να δηλώσει την έλλειψη ορισμάτων σε συναρτήσεις (πχ στη συνάρτηση main(void)).

Εδώ πρέπει να σημειώσουμε πως εκτός των παραπάνω βασικών τύπων υπάρχουν και οι ακόλουθοι **τροποποιητές τύπων δεδομένων (qualifiers)** οι οποίοι εφαρμόζονται σε αυτούς. Έτσι έχουμε:

- **Οι τροποποιητές short και long** αναφέρονται στους τύπους δεδομένων int. Έτσι μπορούμε να έχουμε:

  short int

  long int

  Ο short int συνήθως έχει μήκος 16 bit (2 byte) ενώ ο long int 32 bit (4 byte). Επίσης ο προσδιοριστής long εφαρμόζεται και στον τύπο double. Έτσι ο τύπος

  long double

  έχει συνήθως μήκος 80 bit (10 byte) και χρησιμοποιείται οταν θέλουμε την μέγιστη δυνατή ακρίβεια στους υπολογισμούς μας.

- **Οι τροποποιητές unsigned και signed** αναφέρονται στους τύπους δεδομένων int. Οι unsigned είναι πάντοτε θετικοί αριθμοί ή μηδέν ενώ οι signed μπορεί να είναι θετικοί μηδέν ή αρνητικοί αριθμοί. Για παράδειγμα μπορούμε να έχουμε τις ακόλουθες μεταβλητές:

  unsigned int a;

  signed int b;

  Επίσης μπορούμε να έχουμε συνδυασμούς όπως:

  unsigned short int e;

  signed long int f;

Τέλος στον πίνακα 2.2 συνοψίζονται οι βασικοί τύποι δεδομένων και το εύρος τιμών τους.

Πίνακας 2.1 Οι κωδικοί ASCII για τους βασικούς 128 χαρακτήρες του υπολογιστή.

Non-Printing Characters					Printing Characters								
Name	Control	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
null	ctrl-@	0	00	NUL	32	20	Space	64	40	@	96	60	`
start header	ctrl-A	1	01	SOH	33	21	!	65	41	A	97	61	a
start of text	ctrl-B	2	02	STX	34	22	"	66	42	B	98	62	b
end of text	ctrl-C	3	03	ETX	35	23	#	67	43	C	99	63	c
end of xmit	ctrl-D	4	04	EOT	36	24	$	68	44	D	100	64	d
enquiry	ctrl-E	5	05	ENQ	37	25	%	69	45	E	101	65	e
acknowledge	ctrl-F	6	06	ACK	38	26	&	70	46	F	102	66	f
bell	ctrl-G	7	07	BEL	39	27	'	71	47	G	103	67	g

backspace	ctrl-H	8	08	**BS**	40	28	**(**	72	48	**H**	104	68	**h**
horizontal tab	ctrl-I	9	09	**HT**	41	29	**)**	73	49	**I**	105	69	**i**
line feed	ctrl-J	10	0A	**LF**	42	2A	*	74	4A	**J**	106	6A	**j**
vertical tab	ctrl-K	11	0B	**VT**	43	2B	+	75	4B	**K**	107	6B	**k**
form feed	ctrl-L	12	0C	**FF**	44	2C	,	76	4C	**L**	108	6C	**l**
carriage feed	ctrl-M	13	0D	**CR**	45	2D	-	77	4D	**M**	109	6D	**m**
shift out	ctrl-N	14	0E	**SO**	46	2E	.	78	4E	**N**	110	6E	**n**
shift in	ctrl-O	15	0F	**SI**	47	2F	/	79	4F	**O**	111	6F	**o**
data line escape	ctrl-P	16	10	**DLE**	48	30	**0**	80	50	**P**	112	70	**p**
dev.cntrl.1	ctrl-Q	17	11	**DC1**	49	31	**1**	81	51	**Q**	113	71	**q**
dev.cntrl.2	ctrl-R	18	12	**DC2**	50	32	**2**	82	52	**R**	114	72	**r**
dev.cntrl.3	ctrl-S	19	13	**DC3**	51	33	**3**	83	53	**S**	115	73	**s**
dev.cntrl.4	ctrl-T	20	14	**DC4**	52	34	**4**	84	54	**T**	116	74	**t**
neg acknowledge	ctrl-U	21	15	**NAK**	53	35	**5**	85	55	**U**	117	75	**u**
sync.idel	ctrl-V	22	16	**SYN**	54	36	**6**	86	56	**V**	118	76	**v**
end of xmit block	ctrl-W	23	17	**ETB**	55	37	**7**	87	57	**W**	119	77	**w**
cancel	ctrl-X	24	18	**CAN**	56	38	**8**	88	58	**X**	120	78	**x**
end of medium	ctrl-Y	25	19	**EM**	57	39	**9**	89	59	**Y**	121	79	**y**
substitute	ctrl-Z	26	1A	**SUB**	58	3A	**:**	90	5A	**Z**	122	7A	**z**
escape	ctrl-[	27	1B	**ESC**	59	3B	**;**	91	5B	**[**	123	7B	**{**
file separator	ctrl-\	28	1C	**FS**	60	3C	<	92	5C	\	124	7C	|
group separator	ctrl-]	29	1D	**GS**	61	3D	=	93	5D	**]**	125	7D	**}**
record separator	ctrl-^	30	1E	**RS**	62	3E	>	94	5E	^	126	7E	~
unit separator	ctrl-_	31	1F	**US**	63	3F	**?**	95	5F	_	127	7F	**DEL**

Πίνακας 2.2 . Οι βασικοί τύποι δεδομένων και το εύρος τιμών τους.

Τύπος Δεδομένων	Εύρος σε bits	Εύρος Τιμών
char	8	
int	32	-2147483648 έως 2147483647
short int	16	-32768 έως 32767
long int	32	Όπως και ο int
float	32	$\pm 1.4 \times 10^{-45}$ έως $\pm 3.4 \times 10^{38}$
double	64	$\pm 4.9 \times 10^{-324}$ έως $\pm 1.8 \times 10^{308}$

2.7 Περισσότερα για τις μεταβλητές.

Οι μεταβλητές έχουν τη δυνατότητα να αποθηκεύουν και να διαχειρίζονται δεδομένα, κατά συνέπεια είναι οι πιο βασικές οντότητες σε ένα πρόγραμμα. Για τον λόγο αυτό πρέπει να τονίσουμε εδώ τους βασικούς κανόνες που τις διέπουν.

Η **δήλωση** όλων των μεταβλητών που χρειαζόμαστε γίνεται πάντοτε πάνω-πάνω και στην αρχή κάθε συνάρτησης. Για παράδειγμα όταν αναπτύσσουμε την συνάρτηση main το πρώτο πράγμα που κάνουμε μετά την γραμμή

```
int main(void){
```

είναι η δήλωση όλων των μεταβλητών που χρειάζεται η συνάρτηση.

Η **εμβέλεια** κάθε μεταβλητής ξεκινά από την γραμμή της δήλωσής της μέχρι το τέλος της συνάρτησης μέσα στην οποία έχει δηλωθεί. Εάν ένα πρόγραμμα αποτελείται από δύο ή περισσότερες συναρτήσεις τότε κάθε συνάρτηση έχει τις δικές της μεταβλητές οι οποίες ισχύουν μόνο και μόνο μέσα στην συγκεκριμένη συνάρτηση.

Στην C υπάρχει η δυνατότητα να ορίζουμε μεταβλητές και έξω από τις συναρτήσεις αμέσως μετά τα αρχικά #include. Σε αυτή την περίπτωση οι συγκεκριμένες μεταβλητές έχουν εμβέλεια μέσα σε κάθε συνάρτηση που περιέχει το πρόγραμμα. Αυτές ειδικά οι μεταβλητές ονομάζονται *Γενικές Μεταβλητές* (global variables). Συνιστάται σε αρχάριους στον προγραμματισμό να ΜΗΝ χρησιμοποιούν τέτοιου είδους γενικές μεταβλητές.

Αμέσως μετά την δήλωση των μεταβλητών σε μία συνάρτηση πρέπει να ακολουθεί η **αρχικοποίησή** τους. Εδώ χρειάζεται προσοχή διότι όταν ορίζουμε μια μεταβλητή αυτή δεν αρχικοποιείται αυτόματα (πχ. στην τιμή 0) αλλά περιέχει μια τυχαία τιμή (η οποία βρίσκεται εκείνη την στιγμή στην μνήμη του υπολογιστή που καταλαμβάνει η συγκεκριμένη μεταβλητή που ορίσαμε). Ο προγραμματιστής οφείλει να αρχικοποιεί κάθε μεταβλητή που έχει ορίσει ΠΡΙΝ την χρησιμοποιήσει. Αφού ο προγραμματιστής δηλώσει και αρχικοποιήσει τις μεταβλητές στη συνέχεια μπορεί να αναπτύξει το πρόγραμμά του κάνοντας πράξεις, εισάγοντας λογική σε αυτό κτλ.

Σχετικά με τον τύπο των μεταβλητών που πρέπει να χρησιμοποιείτε για αριθμητικά δεδομένα στα προγράμματά σας, ως αρχάριοι, καλό θα είναι να ακολουθήσετε τους παρακάτω απλούς κανόνες.

- Μην χρησιμοποιείτε άσκοπα μεταβλητές τύπου int. Χρησιμοποιείστε κατά κόρον μεταβλητές τύπου double ή float. Ακόμη και εάν τα αριθμητικά δεδομένα σας σε προβλήματα είναι ακέραιοι αριθμοί χρησιμοποιείστε μεταβλητές τύπου double.
- Οι ακέραιοι έχουν περιορισμένη ακρίβεια και χρησιμοποιούνται μόνο σε ορισμένες περιπτώσεις όπως στην εντολή for και στους πίνακες όπως θα δούμε παρακάτω.
- Πράξεις με ακεραίους και ειδικότερα οι διαιρέσεις είναι πολύ πιθανό να σας οδηγήσουν σε λανθασμένα αποτελέσματα. Για παράδειγμα στον παρακάτω κώδικα:

```
int a,b;
double x;
a=5;
```

```
b=10;
x=a/b;
```

Το αποτέλεσμα θα είναι x=0 και όχι x=0.5. Αυτό συμβαίνει γιατί η πράξη a/b γίνεται ως ακέραια και το αποτέλεσμα χάνει τα δεκαδικά ψηφία.

- Όταν σε μια μεταβλητή double ή float αντιστοιχούμε ακεραίους αριθμούς (αριθμητικές σταθερές) καλό είναι να βάζουμε στο τέλος τη τελεία (.). Για παράδειγμα ο κώδικας Β είναι προτιμότερος του Α:

Κώδικας Α	Κώδικας Β
double x, y;	double x, y;
x=4;	x=4.;
y=9;	y=9.;
z=x*y;	z=x*y;

- Την τελεία πρέπει αν την βάζουμε σε κάθε αριθμητική σταθερά που εμπλέκεται σε παραστάσεις, δηλώνοντας με αυτόν τον τρόπο πως πρόκειται για αριθμητική σταθερά κινητής υποδιαστολής. Για παράδειγμα ο κώδικας:

```
double x, y,z;
x=2.;
y=10.;
z=(1/2)*x*y;
```

Δίνει αποτέλεσμα z=0 διότι ο όρος (1/2) αποτελεί διαίρεση ακεραίων και κατά συνέπεια το αποτέλεσμα θα είναι επίσης ακέραιος δηλαδή 0. Ο σωστός κώδικας είναι:

```
double x, y,z;
x=2.;
y=10.;
z=(1./2.)*x*y;
```

Ο οποίος δίνει αποτέλεσμα z=10.

2.8 Μετατροπές του τύπου δεδομένων (casting).

Θα πρέπει να είμαστε πολύ προσεκτικοί όταν μετατρέπουμε έναν τύπο δεδομένων σε κάποιον άλλον. Αυτό εν γένει δεν είναι πάντοτε δυνατό. Για παράδειγμα δεν μπορούμε πάντοτε να μετατρέπουμε έναν double ή float σε int γιατί απλά ένας ακέραιος δεν είναι σε θέση να διαχειριστεί πολύ μεγάλους αριθμούς (άνω των 2 δις). Ομοίως δεν μπορούμε πάντοτε να μετατρέπουμε έναν double σε float. Ακολουθούν βασικά παραδείγματα μετατροπής:

- Στο παρακάτω παράδειγμα ο a θα πάρει την τιμή 2.

```
int a;
double x;
x=2.7;
a=(int)x;
```

- Στο παρακάτω παράδειγμα ο a θα προσλάβει μια εντελώς λάθος τιμή.

```
int a;
double x;
x=1.234e24;
a=(int)x;
```

- Στα παρακάτω παραδείγματα ο x θα προσλάβει την τιμή 25.

```
int a;                  int a;
double x;               float x;
a=25;                   a=25;
x=(double)a;            x=(float)a;
```

Όπως παρατηρούμε στα παραπάνω παραδείγματα η μετατροπή ενός τύπου σε άλλον γίνεται με το όνομα του νέου τύπου σε παρένθεση πριν την μεταβλητή. Αυτό ονομάζεται μετατροπή τύπου (ή casting).

Γενικά η C επιτρέπει μέσα στην ίδια έκφραση την ανάμειξη διαφορετικών τύπων δεδομένων. Αυτό όμως είναι ένα πράγμα που ο αρχάριος προγραμματιστής ΠΡΕΠΕΙ να αποφεύγει. Δεν υπάρχει κάποιος σοβαρός λόγος στα προγράμματά σας να αναμειγνύετε διαφορετικούς αριθμητικούς τύπους δεδομένων. Ξεκινήστε δηλώνοντας όλες τις μεταβλητές σας να είναι του αυτού τύπου (πχ όλες double). Αν παρ΄ όλα αυτά χρειάζεστε μετατροπές ακολουθήστε τους κανόνες των παραπάνω παραδειγμάτων.

Πρέπει να γνωρίζουμε επίσης πως η C όταν αναμειγνύουμε διαφορετικούς τύπους δεδομένων στην ίδια έκφραση κάνει "αυτόματα" τις μετατροπές ακολουθώντας αυστηρούς κανόνες. Οι μετατροπές μπορούν να γίνουν όπως στο παρακάτω διάγραμμα από δεξιά προς τα αριστερά:

long double ← double ← float ← int ← short int

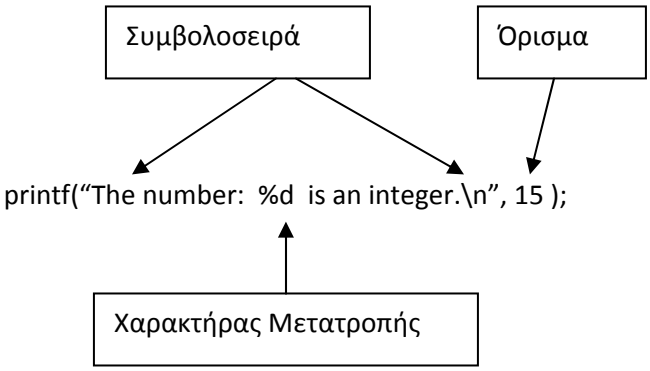
2.9 Φορμαρισμένη έξοδος - η συνάρτηση printf().

Η συνάρτηση **printf()** χρησιμοποιείται για να τυπώνει μηνύματα στην οθόνη του υπολογιστή μας, και εμπεριέχεται στο αρχείο επικεφαλίδας **stdio.h**.

Στο σχήμα 2.3 βλέπουμε ένα απλό παράδειγμα σύνταξης της συνάρτησης printf(). Το συγκεκριμένο παράδειγμα τυπώνει στην οθόνη μας την ακόλουθη συμβολοσειρά:

The number 15 is an integer.

Γενικά η συμβολοσειρά περιέχει χαρακτήρες, καθώς και χαρακτήρες μετατροπής για τα ορίσματα τα οποία αναγράφονται στο τέλος της εντολής. Παρατηρείστε πως οι χαρακτήρες μετατροπής ξεκινούν με το σύμβολο **%**. Οι βασικοί χαρακτήρες μετατροπής εικονίζονται στον πίνακα 2.3

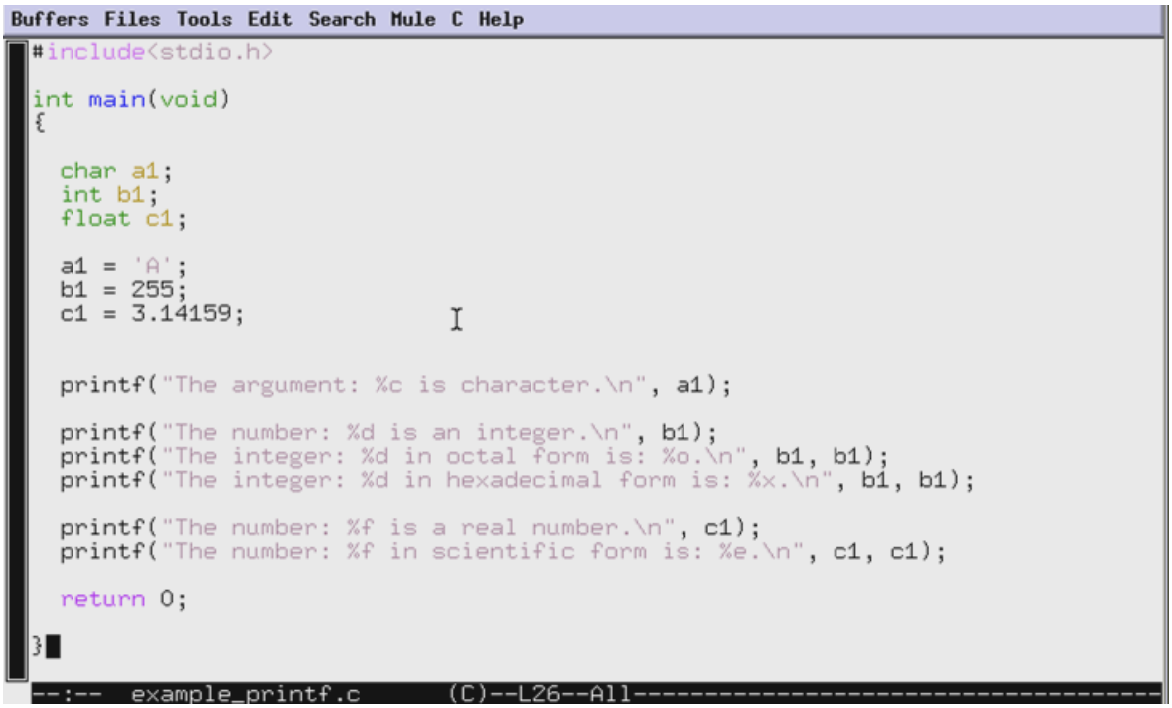


Σχήμα 2.4 Παράδειγμα σύνταξης της συνάρτησης printf().

Πίνακας 2.3 Οι βασικοί χαρακτήρες μετατροπής της συνάρτησης printf().

Χαρακτήρας μετατροπής	Τύπος ορίσματος – Μορφή εκτύπωσης
d,i	Τύπος int.
o	Τύπος int. Εκτύπωση σε Οκταδική μορφή.
x,X	Τύπος int. Εκτύπωση σε Δεκαεξαδική μορφή.
u	Τύπος unsigned int.
c	Τύπος char. Εκτύπωση ενός χαρακτήρα.
s	Τύπος char. Εκτύπωση πίνακα χαρακτήρων.
f	Τύπος float, double.
e,E	Τύπος float, double. Επιστημονική εκτύπωση.
g,G	Τύπος float, double. Επιστημονική ή κοινή εκτύπωση ανάλογα με την ακρίβεια.

Προσοχή ο χαρακτήρας '\n' ο οποίος μπαίνει στο τέλος της συμβολοσειράς είναι ο χαρακτήρας της *νέας γραμμής* και μετακινεί τον κάρσορα του υπολογιστή στην επόμενη γραμμή. Παράληψη του χαρακτήρα '\n' θα "κολλήσει" την επόμενη εκτύπωση στην προηγούμενη. Σημειώνουμε επίσης πως εάν θέλουμε να εκτυπώσουμε το σύμβολο % αυτό γίνεται ως **%%**. Από την παραπάνω λίστα παραλείψαμε τον χαρακτήρα μετατροπής p ο οποίος αναφέρεται σε δείκτες και θα τον συναντήσουμε αργότερα. Στο παρακάτω σχήμα 2.5 εικονίζεται ένα απλό πρόγραμμα στο οποίο ορίζονται, αρχικοποιούνται και εκτυπώνονται τρεις απλές μεταβλητές. Όπως φαίνεται σε αυτό χρησιμοποιούνται οι περισσότεροι από τους παραπάνω χαρακτήρες μετατροπής του πίνακα 2.3.



Σχήμα 2.5 Πρόγραμμα παράδειγμα για την χρήση μερικών από τους χαρακτήρες μετατροπής.

Η εκτέλεση του παραπάνω προγράμματος εμφανίζει στην οθόνη τα ακόλουθα:

The argument: A is a character.
The number: 255 is an integer.
The integer: 255 in octal form is: 377.
The integer: 255 in hexadecimal form is: ff.
The number: 3.141590 is a real number.
The number: 3.141590 in scientific form is: 3.141590e+00

Η συνάρτηση printf() επιτρέπει στον χρήστη τον καθορισμό του **ελάχιστου πλάτους πεδίου** κατά την εκτύπωση ενός ορίσματος. Αυτό γίνεται με την παρεμβολή ενός αριθμού μεταξύ του συμβόλου % και του χαρακτήρα μετατροπής. Για παράδειγμα το **%12f** καθορίζει ότι το αποτέλεσμα ενός πραγματικού αριθμού θα τυπωθεί με τουλάχιστον 12 χαρακτήρες. Ο αριθμός στοιχίζεται στα δεξιά. Εάν θέλουμε αριστερή στοίχιση του αριθμού τότε χρησιμοποιούμε το **%-12f**.

Επίσης η συνάρτηση printf() επιτρέπει τον καθορισμό του αριθμού δεκαδικών ψηφίων. Για παράδειγμα το **%.3f** καθορίζει ότι το αποτέλεσμα ενός πραγματικού αριθμού θα τυπωθεί με τρία δεκαδικά ψηφία, ενώ το **%12.3f** καθορίζει ότι το αποτέλεσμα ενός πραγματικού αριθμού θα τυπωθεί με τουλάχιστον 12 χαρακτήρες εκ των οποίων τρία ψηφία θα είναι δεκαδικά. Με αυτόν τον τρόπο καθορίζουμε την ακρίβεια κατά την εκτύπωση σε ένα πραγματικό αριθμό. Δείτε το επόμενο παράδειγμα και παρατηρείστε με προσοχή τις εκτυπώσεις:

```
double x;
int a;
x=3.14159;
a=35;                                   Εκτυπώσεις
printf("x=%f\n",x);          =>         x=3.14159
printf("x=%.3f\n",x);        =>         x=3.141
printf("x=%10.3f\n",x);      =>         x=      3.141
printf("x=%d\n",a);          =>         a=35
printf("a=%10d\n",a);        =>         a=        35
```

2.10 Οι βασικές μαθηματικές συναρτήσεις στη C.

Σε αυτή την παράγραφο θα αναφερθούμε στις βασικές μαθηματικές συναρτήσεις που περιλαμβάνει η C. Οι συναρτήσεις αυτές χρησιμοποιούν το αρχείο επικεφαλίδας *math.h* άρα στην κορυφή κάθε προγράμματος ο προγραμματιστής οφείλει να αναγράφει το **#include<math.h>**.

Στους ακόλουθους πίνακες αναγράφονται οι βασικότερες μαθηματικές συναρτήσεις. Σημειώνουμε πως όλες οι συναρτήσεις επιστρέφουν τύπο δεδομένων double. Στους παρακάτω πίνακες οι μεταβλητές x,y είναι τύπου double ενώ η μεταβλητή n είναι τύπου int. Στις τριγωνομετρικές συναρτήσεις οι γωνίες εκφράζονται σε ακτίνια.

Για περισσότερες μαθηματικές συναρτήσεις πρέπει να ανατρέξετε στο βιβλίο σας.

- **Τετραγωνική ρίζα, εκθετικές-λογαριθμικές και άλλες βασικές συναρτήσεις**.

Συνάρτηση	**Περιγραφή**
sqrt(x)	Τετραγωνική ρίζα του x, με x>=0.
exp(x)	Εκθετική συνάρτηση e^x
log(x)	Φυσικός λογάριθμος ln(x), με x>0
log10(x)	δεκαδικός λογάριθμος log10(x), με x>0
pow(x,y)	ύψωση σε δύναμη x^y
fabs(x)	Απόλυτη τιμή |x|
ldexp(x,n)	$x*2^n$

- **Τριγωνομετρικές συναρτήσεις. Προσοχή! τα ορίσματα είναι όλα σε rad.**

Συνάρτηση	Περιγραφή
cos(x)	συνημίτονο του x
sin(x)	ημίτονο του x
tan(x)	εφαπτομένη του x
acos(x)	τόξο συνημιτόνου του x στο διάστημα [0, π], x [-1,1]
asin(x)	τόξο ημιτόνου του x στο διάστημα [-π/2, π/2], x [-1,1]
atan(x)	τόξο εφαπτομένης του x στο διάστημα [-π/2, π/2]
atan2(y,x)	τόξο εφαπτομένης του y/x στο διάστημα [-π, π]
cosh(x)	υπερβολικό συνημίτονο του x
sinh(x)	υπερβολικό ημίτονο του x
tanh(x)	υπερβολική εφαπτομένη του x

Παράδειγμα: Να αναπτύξετε ένα πρόγραμμα το οποίο να υπολογίζει και να εκτυπώνει το συνημίτονο, το ημίτονο και την εφαπτομένη της γωνίας των 60°.

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

int main(void)
{
        double x;                       /*Ορισμός μεταβητής x */
        x=60.;                          /*Αρχικοποίηση μεταβλητής x */
        x=x*3.14159/180.;               /* Μετατροπή από μοίρες σε rad */

        printf("cos(60)=%f\n",cos(x));          /*Εκτύπωση συνημιτόνου */
        printf("sin(60)=%f\n",sin(x));          /*Εκτύπωση ημιτόνου */
        printf("tan(60)=%f\n",tan(x));          /*Εκτύπωση εφαπτομένης */

        return 0;               /* Τερματισμός συνάρτησης main() και προγράμματος */
}
```

Ας υποθέσουμε ότι το παραπάνω πρόγραμμα αποθηκεύεται στο αρχείο example.c. Η μεταγλώττιση του αρχείου αυτού γίνεται ως εξής:

gcc example.c -lm

Το εκτελέσιμο αρχείο θα ονομαστεί a.out όπως έχουμε ήδη αναφέρει. **Προσοχή!!!** Κατά την μεταγλώττιση πρέπει να χρησιμοποιήσουμε τον όρο **–lm** (link mathematics) για να υποδείξουμε στον μεταγλωττιστή ότι χρησιμοποιούμε μαθηματικές συναρτήσεις. Εάν εκτελέσουμε το παραπάνω πρόγραμμα θα εμφανιστούν στη οθόνη μας τα εξής:

cos(60)=0.500001
sin(60)=0.866025
tan(60)=1.732047

2.11 Οι βασικοί τελεστές στη C και η ντιρεκτίβα #define.

Εκτός από τους αριθμητικούς τελεστές, τους οποίους έχουμε ήδη αναφέρει, στη C συναντάμε και τους ακόλουθους βασικούς τελεστές:

- Οι **συσχετιστικοί τελεστές** είναι οι ακόλουθοι:

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

- Οι **τελεστές ισότητας** είναι οι ακόλουθοι:

Τελεστής	Περιγραφή
==	Ίσο με
!=	Άνισο με

- Οι **λογικοί τελεστές** είναι οι ακόλουθοι:

Τελεστής	Περιγραφή
&&	Λογικός τελεστής AND (και)
||	Λογικός τελεστής OR (ή)
!	Λογικός τελεστής NEGATION (όχι)

Τους συσχετιστικούς τελεστές, τους τελεστές ισότητας και τους λογικούς τελεστές τους συναντάμε κυρίως στις εντολές **if**, **for**, **while**, **do-while**. Οι παραπάνω τελεστές χρησιμοποιούνται για συγκρίσεις μεταξύ αριθμών, μεταβλητών και παραστάσεων. Εάν η σύγκριση είναι **αληθής** τότε το αποτέλεσμα είναι 1 διαφορετικά εάν είναι **ψευδής** τότε το αποτέλεσμα είναι μηδέν. Ας δώσουμε ένα παράδειγμα. Έστω:

```
int a, b, c, d;
a = 15;
b = 7;
c = 25;
d = 15;

τότε η παράσταση    a>10    επιστρέφει τον αριθμό 1
η παράσταση    a>b      επιστρέφει τον αριθμό 1
η παράσταση    a>c      επιστρέφει τον αριθμό 0
η παράσταση    a>=b     επιστρέφει τον αριθμό 1
η παράσταση    a<b      επιστρέφει τον αριθμό 0
η παράσταση    a<c      επιστρέφει τον αριθμό 1
η παράσταση    a<=b     επιστρέφει τον αριθμό 0
η παράσταση    a==b     επιστρέφει τον αριθμό 0
η παράσταση    a==d     επιστρέφει τον αριθμό 1
η παράσταση    a!=b     επιστρέφει τον αριθμό 1
η παράσταση    a!=d     επιστρέφει τον αριθμό 0
επίσης
η παράσταση    (a>b)&&(a<c)     επιστρέφει τον αριθμό 1
η παράσταση    (a>b)&&(a>c)     επιστρέφει τον αριθμό 0
η παράσταση    (a>b)||(a<c)     επιστρέφει τον αριθμό 1
η παράσταση    (a>b)||(a>c)     επιστρέφει τον αριθμό 1
η παράσταση    (a>c)||(d>c)     επιστρέφει τον αριθμό 0
η παράσταση    !(a>b)    επιστρέφει τον αριθμό 0
η παράσταση    !(a==b) επιστρέφει τον αριθμό 1
```

- Ο **τελεστής αύξησης** και ο **τελεστής μείωσης** εικονίζεται παρακάτω:

Τελεστής	Περιγραφή
++	Τελεστής αύξησης κατά 1
--	Τελεστής μείωσης κατά 1

Οι τελεστές ++ και -- χρησιμοποιούνται όταν θέλουμε να προσθέσουμε ή να αφαιρέσουμε το 1 από μία μεταβλητή. Έτσι

το ++a; ισοδυναμεί με το a=a+1;
ενώ το --a; ισοδυναμεί στο a=a-1;

Εδώ πρέπει να σημειώσουμε πως οι τελεστές ++ και – μπορούν να χρησιμοποιηθούν είτε ως **προθεματικοί** τελεστές (δηλ. πριν την μεταβλητή, όπως ++a ή --a) είτε ως **μεταθεματικοί** (δηλ. μετά την μεταβλητή, όπως a++ ή a--). Και στις δύο περιπτώσεις η τιμή του a αυξάνει ή ελαττώνεται κατά 1. Για παράδειγμα στην παράσταση ++a η τιμή του a αυξάνει πριν χρησιμοποιηθεί η τιμή της, ενώ στην παράσταση a++ η τιμή του a αυξάνει αφού χρησιμοποιηθεί η τιμή της. Έτσι έστω ότι το a ισούται με 5 τότε η παράσταση

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

δίνει στο b την τιμή 5 ενώ η παράσταση

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

την τιμή 6. Το a και στις δύο περιπτώσεις γίνεται 6.

- Ο **τελεστής αντιστοίχησης** είναι ο:

Τελεστής	Περιγραφή
=	Τελεστής αντιστοίχησης

Ο τελεστής = αντιστοιχεί την δεξιά τιμή μιας παράστασης στην αριστερή. Έτσι η παράσταση a=b; αντιστοιχεί την τιμή του b στο a.

- οι **τελεστές αντικατάστασης** είναι οι ακόλουθοι

Τελεστής	Περιγραφή
+=	Τελεστής πρόσθεσης και αντιστοίχησης
-=	Τελεστής αφαίρεσης και αντιστοίχησης

*=	Τελεστής πολ/μου και αντιστοίχησης
/=	Τελεστής διαίρεσης και αντιστοίχησης
%=	Τελεστής υπολοίπου και αντιστοίχησης

Τα παρακάτω παραδείγματα δίνουν την έννοια των παραπάνω τελεστών:

Το a += b; ισοδυναμεί με το a = a+b;
Το a -= b; ισοδυναμεί με το a = a-b;
Το a *= b; ισοδυναμεί με το a = a*b;
Το a /= b; ισοδυναμεί με το a = a/b;
Το a %= b; ισοδυναμεί με το a = a %b;

- Οι **τελεστές πράξεων με bits** είναι οι ακόλουθοι :

Τελεστής	**Περιγραφή**
&	AND για bit
|	OR για bit
^	XOR για bit
~	NOT για bit
>>	Ολίσθηση αριστερά
<<	Ολίσθηση δεξιά

Οι παραπάνω τελεστές αφορούν πράξεις σε επίπεδο bits και για να τους καταλάβουμε πρέπει να διαχειριζόμαστε τους διάφορους αριθμούς σε δυαδική μορφή. Οι τελεστές &, |, ^ και ~ αντιστοιχούν στις απλές πράξεις της **άλγεβρας Boole**. Οι τελεστές >> και << προκαλούν ολίσθηση στα δεξιά και στα αριστερά αντίστοιχα. Έτσι για παράδειγμα εάν η μεταβλητή a είναι ο δυαδικός αριθμός 01101000 τότε η παράσταση

b = a >> 2;

δίνει στη μεταβλητή b την τιμή 00011010.

Εδώ θα κάνουμε μια παρένθεση και θα αναφερθούμε σε σταθερές οι οποίες ονομάζονται **Συμβολικές Σταθερές**. Σε πολλά προγράμματα επιθυμούμε να χρησιμοποιούμε πολλές φορές κάποιες τιμές οι οποίες συνήθως είναι φυσικές σταθερές όπως ο αριθμός π=3.14159, η επιτάχυνση της

βαρύτητας g=9.81m/sec^2 κτλ. Οι τιμές αυτές μπορούν να εισαχθούν σε κάποιες σταθερές που ονομάζονται συμβολικές σταθερές. Ο συμβολικές σταθερές μπορούν να οριστούν μία φορά και όποτε χρειάζεται μπορούν να χρησιμοποιηθούν σε οποιοδήποτε σημείο του προγράμματος. Οι συμβολικές σταθερές είναι οντότητες που μπορούν να δημιουργηθούν με την χρήση της ντιρεκτίβας **#define**. Η γενική μορφή της ντιρεκτίβας #define είναι:

#define όνομα_μακροεντολής ακολουθία_χαρακτήρων

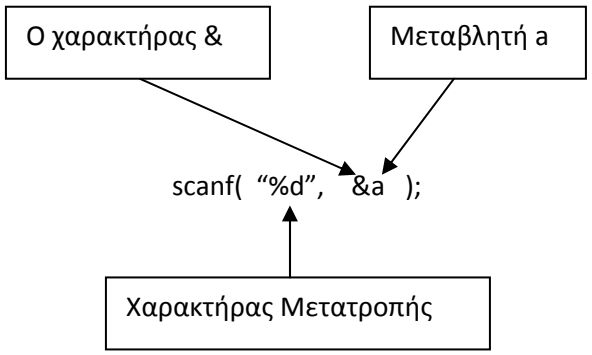
και προκαλεί την αντικατάσταση μιας ακολουθίας χαρακτήρων με μία άλλη. Ακολουθεί ένα απλό πρόγραμμα το οποίο υπολογίζει την περίμετρο και το εμβαδό ενός κύκλου ακτίνας 10cm.

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

#define pi 3.14159              /* Δημιουργία συμβολικής σταθεράς pi=3.14159*/

int main(void)
{
        double R, S, E;                 /* Δήλωση μεταβλητών */
        R=10.;                          /* Αρχικοποίηση ακτίνας */
        S=2.*pi*R;                      /* Υπολογισμός περιμέτρου - χρήση του pi*/
        E=pi*pow(R,2.);                 /* Υπολογισμός εμβαδού - χρήση του pi */
        printf("S=%f cm  E=%f cm^2 \n",S,E);        /*Εκτύπωση αποτελεσμάτων */
        return 0;
}
```

2.12 Φορμαρισμένη είσοδος - η συνάρτηση scanf().

Η συνάρτηση **scanf()** χρησιμοποιείται για να διαβάζουμε διάφορους τύπους δεδομένων ή συμβολοσειρές από την τυπική είσοδο (πληκτρολόγιο). Η εκτέλεση της συνάρτησης scanf() σταματά την ροή του προγράμματος έως ότου ο χρήστης πληκτρολογήσει τα απαραίτητα δεδομένα. Στη συνέχεια το πρόγραμμα συνεχίζει λαμβάνοντας υπ΄όψη και τα δεδομένα που έχουν πληκτρολογηθεί. Η συνάρτηση scanf() όπως και η printf() εμπεριέχεται στο αρχείο επικεφαλίδας **stdio.h**. Στο σχήμα 2.6 βλέπουμε ένα απλό παράδειγμα σύνταξης της συνάρτησης scanf().

Στο παράδειγμα του σχήματος 2.6 εισάγουμε έναν ακέραιο αριθμό στην μεταβλητή a. Οι χαρακτήρες μετατροπής που χρησιμοποιούμε στην συνάρτηση scanf() εικονίζονται στον πίνακα 2.4. Εδώ πρέπει να τονίσουμε την χρήση του **Τελεστή Διεύθυνσης &**, ο οποίος είναι απαραίτητος και αναγράφεται πριν το όνομα της μεταβλητής. Ο τελεστής & υπολογίζει την διεύθυνση στην μνήμη του υπολογιστή που καταλαμβάνει μια μεταβλητή. Περισσότερα για τον τελεστή & θα μάθουμε όταν αναφερθούμε στους δείκτες σε επόμενο κεφάλαιο.



Σχήμα 2.6 Παράδειγμα σύνταξης της συνάρτησης scanf().

Πίνακας 2.4 Οι βασικοί χαρακτήρες μετατροπής της συνάρτησης scanf().

Χαρακτήρας μετατροπής	Τύπος μεταβλητής
d	int.
c	char.
s	Πίνακας χαρακτήρων.
lf	double.
f	float.
c	char.

Στο παρακάτω σχήμα 2.7 εικονίζεται ένα απλό παράδειγμα για την χρήση της συνάρτησης scanf(). Στο παράδειγμα ζητούμε να αναπτύξουμε ένα πρόγραμμα στο οποίο να εισάγουμε από το πληκτρολόγιο δύο αριθμούς int και έναν float, να τους αποθηκεύσουμε σε κατάλληλες μεταβλητές και στη συνέχεια να τους εκτυπώσουμε.

```
Buffers Files Tools Edit Search Mule C Help
#include<stdio.h>

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

  printf("Please insert two integers:\n");
  scanf("%d %d",&a, &b);

  printf("Please insert one real number:\n");
  scanf("%f",&c);

  printf("The integers you have typed are: %d %d\n",a,b);
  printf("The real number you have typed is: %f\n",c);

  return 0;

}
-1:--  example_scanf.c      (C)--L19--All----------------------------------
Wrote /home/student1/kef2/example_scanf.c
```

Σχήμα 2.7 Απλό παράδειγμα για την χρήση της συνάρτησης scanf().

Η εκτέλεση του παραπάνω προγράμματος είναι η ακόλουθη (με έντονα γράμματα αναγράφονται οι αριθμοί τους οποίους πληκτρολογούμε):

Please insert two integers:
12345 98765
Please insert one real number:
987.654321
The integers you have typed are: 12345 98765
The real number you have typed is: 987.654321

2.13 Οι συναρτήσεις εισόδου-εξόδου χαρακτήρων getchar(), putchar().

Ειδικά για τους χαρακτήρες, η C είναι εφοδιασμένη με κατάλληλες συναρτήσεις εισόδου-εξόδου. Η συνάρτηση **getchar()** μπορεί να χρησιμοποιηθεί για εισαγωγή χαρακτήρων από το πληκτρολόγιο, ενώ η συνάρτηση **putchar()** μπορεί να χρησιμοποιηθεί για εκτύπωση χαρακτήρων στην οθόνη του υπολογιστή. Ακολουθούν παραδείγματα σύνταξης των δύο νέων συναρτήσεων και ο αντίστοιχος ισοδύναμος κώδικας με τις συναρτήσεις scanf() και printf().

Κώδικας με την getchar()	Ισοδύναμος κώδικας με την scanf()
char c;	char c;
c=getchar();	scanf(“%c”,c);

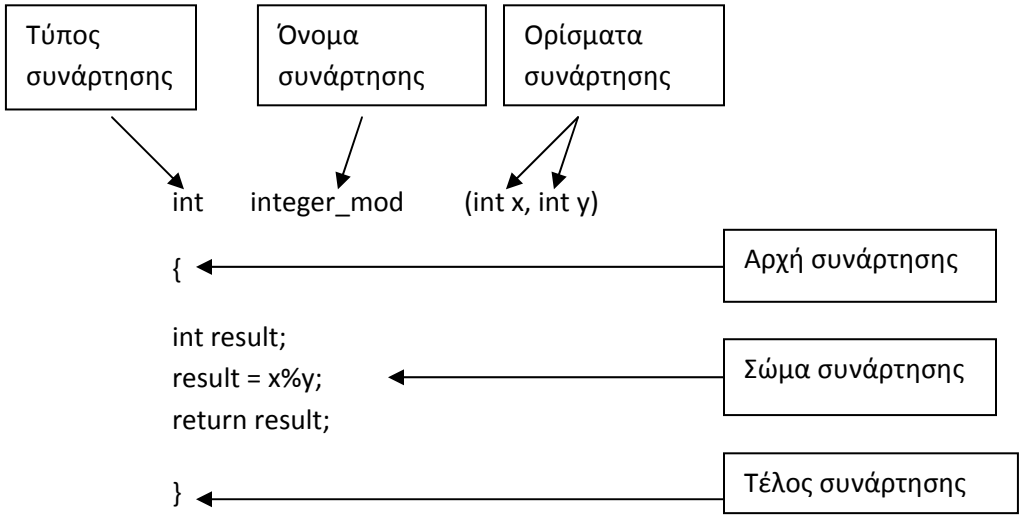
Κώδικας με την putchar()	Ισοδύναμος κώδικας με την printf()
char c;	char c;
c=’k’;	c=’k’;
putchar(c);	printf(“%c”,c);

Σημειώνουμε εδώ πως στην putchar() μπορεί να χρησιμοποιηθεί ως όρισμα ο κωδικοί ASCII (πίνακας 2.1) για κάθε χαρακτήρα πχ. η κλήση putchar(65) θα τυπώσει τον χαρακτήρα ‘A’.

2.14 Οι συναρτήσεις στη C.

Οι συναρτήσεις στη γλώσσα προγραμματισμού C αποτελούν τα βασικά δομικά τμήματα από τα οποία αποτελούνται τα διάφορα προγράμματα. Ένα πρόγραμμα μπορεί να περιέχει πολλές συναρτήσεις. Μία από αυτές είναι η συνάρτηση main(). Η συνάρτηση main() είναι μία ειδική συνάρτηση η οποία δεν μπορεί να απουσιάζει από κανένα πρόγραμμα. Όταν ο υπολογιστής εκτελεί ένα πρόγραμμα ψάχνει βρίσκει την συνάρτηση main() και αρχίζει να εκτελεί σειριακά μία-μία τις γραμμές της. Το πρόγραμμα σταματά αυτόματα όταν εκτελεστεί και η τελευταία γραμμή της main().

Εάν υπάρχουν και άλλες συναρτήσεις σε ένα πρόγραμμα, τότε αυτές για να χρησιμοποιηθούν πρέπει να κληθούν μέσα από την main(). Ας πάρουμε ως παράδειγμα την παρακάτω συνάρτηση η οποία υπολογίζει το υπόλοιπο δύο ακεραίων αριθμών:



Σχήμα 2.7 Παράδειγμα συνάρτησης η οποία υπολογίζει το υπόλοιπο δύο ακεραίων αριθμών.

Κάθε συνάρτηση όπως φαίνεται και στο παραπάνω σχήμα αποτελείται από τα εξής μέρη:

- Ο **τύπος της συνάρτησης** δείχνει τι είδους τιμή επιστρέφει η συνάρτηση μετά την εκτέλεσή της. Εάν η τιμή που επιστρέφει είναι int τότε και η συνάρτηση οφείλει να είναι τύπου int. Εάν η τιμή που επιστρέφει είναι double τότε και η συνάρτηση οφείλει να είναι τύπου double κτλ. Εάν μια συνάρτηση δεν επιστρέφει κάποιο αποτέλεσμα τότε οφείλει να είναι τύπου void.

- Καλό είναι το **όνομα της συνάρτησης** να δίνεται με τέτοιο τρόπο ώστε να δείχνει τι μπορεί να κάνει η συνάρτηση. Στο όνομα της συνάρτησης μπορούν να χρησιμοποιηθούν όλοι οι χαρακτήρες (A έως Z και a έως z), όλα τα ψηφία (0 έως 9) και ο χαρακτήρας υπογράμμισης (_). Σημειώνουμε πως το όνομα της συνάρτησης δεν μπορεί να ξεκινά με ψηφίο ενώ απαγορεύεται η χρήση των αριθμητικών συμβόλων (+, -, *, /), τα εισαγωγικά (“), η απλή απόστροφος (‘), η τελεία (.) και μερικοί άλλοι χαρακτήρες όπως τα #, ?, &, @ κτλ.

- Τα **ορίσματα της συνάρτησης** αποτελούν τις μεταβλητές οι οποίες εισάγονται και χρησιμοποιούνται από αυτή. Με αυτόν τον τρόπο η συνάρτηση μπορεί να χρησιμοποιηθεί όσες φορές θέλουμε κάθε φορά με διαφορετικές τιμές στα ορίσματα. Οι μεταβλητές που μπαίνουν ως ορίσματα έχουν εμβέλεια μόνο μέσα στην συγκεκριμένη συνάρτηση και δεν χρειάζεται να ξαναορισθούν μέσα σε αυτή. Εάν μια συνάρτηση δεν έχει ορίσματα τότε χρησιμοποιείται το void στη θέση των ορισμάτων. Τέτοιο παράδειγμα αποτελεί το *int main(void)*.

- Μία συνάρτηση **αρχίζει** πάντοτε με αριστερή αγκύλη ({) και τελειώνει με **δεξιά** αγκύλη (}).

- Το **σώμα μιας συνάρτησης** περιέχει πάνω-πάνω όλες τις δηλώσεις των μεταβλητών οι οποίες έχουν εμβέλεια μόνο μέσα στην συγκεκριμένη συνάρτηση. Στη συνέχεια μπορεί να περιέχει τις παραστάσεις, τους υπολογισμούς και τις εντολές οι οποίες απαιτούνται. Τέλος εφ όσον επιστρέφει κάποιο αποτέλεσμα αυτό γίνεται με την εντολή return.

Στις παρακάτω γραμμές εικονίζεται η ανάλυση της συνάρτησης του σχήματος 2.7.

```
int   integer_mod (int x, int y)     /* Ορισμός συνάρτησης τύπου int, με όνομα: integer_mod
                                        και με ορίσματα x και y */
{
        int result;                  /* Δήλωση μεταβλητής με το όνομα result */
        result = x%y;                /* Η μεταβλητή result λαμβάνει την τιμή της πράξης x%y */
        return result;               /* Επιστροφή του αποτελέσματος μέσω της μεταβλητής result*/
}
```

Σημειώνουμε εδώ πως μία συνάρτηση καλείται από κάποια άλλη απλά με το όνομά της. Ας δούμε τώρα ένα πλήρες παράδειγμα το οποίο χρησιμοποιεί την παραπάνω συνάρτηση. Ζητούμε να αναπτύξουμε ένα πρόγραμμα στο οποίο να εισάγονται από το πληκτρολόγιο δύο ακέραιοι αριθμοί, να αποθηκεύονται σε δύο κατάλληλες μεταβλητές (τις a και b) στη συνέχεια να καλείται η συνάρτηση και το αποτέλεσμα να επιστρέφεται σε μία άλλη μεταβλητή (την c), η οποία τελικά να εκτυπώνεται. Το πρόγραμμα έχει ως εξής:

```
#include<stdio.h>

int   integer_mod (int x, int y);            /* Δήλωση πρωτότυπο της συνάρτησης */

int main(void)
{
        int a,b,c;                                  /* Δήλωση μεταβλητών a,b,c  */
        printf("Eisagete ta a kai b:");
        scanf("%d %d",&a,&b);                       /* Εισαγωγή δεδομένων */

        c=integer_mod(a,b);                         /* Κλήση συνάρτησης */
        printf("Ypoloipo diairesis %d\n",c);
        return 0;
}


int   integer_mod (int x, int y)
{
        int result;
        result = x%y;
        return result;
}
```

Προσοχή! εάν θέλουμε να χρησιμοποιήσουμε μια συνάρτηση πρέπει να την δηλώσουμε πάνω-πάνω στο πρόγραμμά μας αμέσως μετά τα #include, όπως φαίνεται στο παραπάνω παράδειγμα. Αυτό ονομάζεται δήλωση *πρωτότυπου της συνάρτησης*.

Παράδειγμα: Να γραφεί μια συνάρτηση η οποία να υπολογίζει την τιμή της μαθηματικής παράστασης

$$f(x, y, z) = x^3 + y^2 + z$$

Στη συνέχεια να αναπτύξετε ένα πρόγραμμα στο οποίο να εισάγονται από το πληκτρολόγιο δεδομένα, τα οποία να αποθηκεύονται σε τρεις κατάλληλες μεταβλητές (τις x,y και z). Στη συνέχεια να καλείται η συνάρτηση και το αποτέλεσμα να επιστρέφεται σε μία άλλη μεταβλητή (πχ την a), η οποία τελικά να εκτυπώνεται. Το πλήρες πρόγραμμα με τις κατάλληλες επεξηγήσεις έχει ως εξής:

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

double   fx(double x, double y, double z);          /* Δήλωση πρωτότυπο της συνάρτησης */

int main(void)
{
        double x,y,z,a;                  /* Δήλωση μεταβλητών a,y,z,a  */

        printf("Dose to x: ");
        scanf("%lf",&x);                 /* Εισαγωγή του x */
        printf("Dose to y: ");
        scanf("%lf",&y);                 /* Εισαγωγή του y */
        printf("Dose to z: ");
        scanf("%lf",&z);                 /* Εισαγωγή του z */

        a=fx(x,y,z);                     /* Κλίση συνάρτησης */
        printf("a=%f\n",a);
        return 0;
}


double   fx(double x, double y, double z);
{
        double e;                         /* Δήλωση μεταβλητής e  */
        e=pow(x,3.)+pow(y,2.)+z ;         /* Υπολογισμός παράστασης */
        return e;                         /* Επιστροφή του αποτελέσματος */
}
```