Κεφάλαιο 3.1,3.3-3.4:

Συναρτήσεις Ι

(Διάλεξη 11)

Διδάσκων: Δημήτρης Ζεϊναλιπούρ

Μη-Δομημένος Προγραμματισμός

- Το πρόγραμμα στα αριστερά δεν είναι Αρθρωτό (δεν έχει σωστή δομή).
- Όλη η λειτουργικότητα ορίζεται μέσα στην main.
- Το αποτέλεσμα είναι:
  - Δύσκολη Κατανόηση
  - Δύσκολη Διόρθωση Λαθών
  - Δύσκολη Συντήρηση
  - Δύσκολη Επέκταση

```
// ENA PROGRAMMA MH-ARTHRWTO
#include <stdio.h>

// ORISMOS SYNARTHSHS MAIN
int main() {

        int a = 20;
        int x;      int b = 5;
        int total = 1;
        int i; int f = 1;

        //Paragontiko
        for (i=1; i<=b; i++) {
                    f *= i;
        }

        //Power
        for(i = 0; i < b; i++) {
                    total = total * a;
        }

        printf("q1(%d)=%d \n", b, f);
        printf("q2(%d,%d)=%d \n", a, b, total);
}
```

Στην συνέχεια θα δούμε πως θα γράφαμε το ίδιο πρόγραμμα δομημένα

Δομημένος Προγραμματισμός

- Πραγματικά συστήματα είναι πολύ μεγάλα.
- Εκατομμύρια γραμμές κώδικα σε συστήματα
  - Windows 3.1 (1992) (3,000,000)
  - Windows XP (2002) (40,000,000)
  - Windows Vista (2006) (50,000,000)

- Αρχές **δομημένου προγραμματισμού**:

  - Η **ροή** ελέγχου στο πρόγραμμα πρέπει να είναι όσο το δυνατόν πιο **απλή**

  - Η συγγραφή προγράμματος ακολουθεί το μοντέλο «Από **πάνω-προς τα κάτω**» (top-down)

Σχεδιασμός Top-down

- **Αναγωγή προβλήματος σε απλούστερα προβλήματα**
- Τελικά καταλήγουμε
  - Σε πολλά μικρά προβλήματα
  - Καθένα μπορεί να επιλυθεί με ευκολία

    π.χ. θέλουμε ένα πρόγραμμα που υπολογίζει εμβαδά και περιφέρεια ενός κύκλου. Διασπάμε το πρόβλημα σε υπό-προβλήματα: 1) Εύρεση Εμβαδού, 2) Εύρεση Περιφέρειας.
- Στην γλώσσα C, όπως και στις περισσότερες γλώσσες προγραμματισμού, υπάρχει η έννοια της **συνάρτησης**.
- Η συνάρτηση χρησιμοποιείται για να κωδικοποιεί τα απλά προβλήματα.

Συναρτήσεις Βιβλιοθήκης

- Έχουμε ήδη δει κάποιες βασικές συναρτήσεις (printf, scanf)
- Αυτές είναι αποθηκευμένες σε βιβλιοθήκες
  - Ονομάζονται συναρτήσεις βιβλιοθήκης
  - Δεν ορίζονται από τον χρήστη
  - Ενσωμάτωση με το **include**

Συναρτήσεις (Functions)

- Γιατί μας ενδιαφέρουν
  - **Αφαιρετικότητα (abstraction):** διαχωρισμός ανάμεσα στο τι και το πώς
  - **Αρθρωτός σχεδιασμός** (modular design)
  - **Επαναχρησιμοποίηση** (reuse)

Συναρτήσεις Βιβλιοθήκης

```
#include <stdio.h>


int main ( )
{
        printf (“Hello World!\n”);
        return 0;
}
```

printf είναι το όνομα συνάρτησης στην βιβλιοθήκη stdio

Αυτή η εντολή είναι κλήση συνάρτησης

Αυτό είναι ένα string που περνάμε ως όρισμα στην συνάρτηση printf

Συναρτήσεις

Παρατήρηση:

- Κάθε συνάρτηση παίρνει 0 ή περισσότερες τιμές εισόδου (arguments)
- Κάθε συνάρτηση επιστρέφει ΠΑΝΤΑ ΕΝΑ αποτέλεσμα (return).

```
#include <stdio.h>

// Είσοδος (τίποτα)Αποτέλεσμα(0)
int main ( )
{
        // Είσοδος (“Hello World!\n”)
        //Αποτέλεσμα(0)
        printf (“Hello World!\n”);
        return 0;
 }
```

Αν κάποιος ορίσει την ακόλουθη πρόταση :

printf (“%d”, printf(“Hello World!\n”));

θα εκτυπωθεί στην οθόνη: 0

Αυτό δείχνει ότι ακόμα και το printf επιστρέφει κάτι επιπλέον της εκτύπωσης που κάνει

Συνάρτηση **main**

```
int main(void)
{
        printf("Hello!\n");
        return 0;
}
```

```
int main()
{
        printf("Hello!\n");
        return 0;
}
```

```
main()
{
        printf("Hello!\n");
}
```

```
main(void)
{
        printf("Hello!\n");
}
```

Όλες οι περιπτώσεις είναι σωστές – Όμως η πάνω αριστερά είναι η πιο ξεκάθαρη (διότι δηλώνουμε ρητά το input και το output)

Συναρτήσεις Χρήστη (User Functions)

Ο προγραμματιστής μπορεί να γράψει και τις δικές του συναρτήσεις (user defined functions)

Τιμές Εισόδου

Αποτέλεσμα επιστροφής

```
#include <stdio.h>

int sum(int, int);

int sum(int a, int b)
{
      return a + b;
}

int main()
{
   int a = 5, b=6;

   printf("%d + %d = %d", a, b, sum(a,b));

   return 0;
}
```

- Αυτό θα είναι το βασικό αντικείμενο μελέτης αυτής της ενότητας.
- Θα μελετήσουμε στην συνέχεια καλύτερα τις συναρτήσεις χρήστη.

Δομή Προγράμματος με χρήση συναρτήσεων Χρήστη

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

A

```
void PrintMessage ();
```

Πρότυπο (Δήλωση) Συνάρτησης

(Δηλώνουν τι επιπλέον συναρτήσεις θα χρησιμοποιήσουμε χωρίς να ορίζουμε πως)

B

```
main  ( )
{
    PrintMessage ( );
    (Ζητάμε να εκτελεστεί η συνάρτηση)
}
```

Κλήση Συνάρτησης

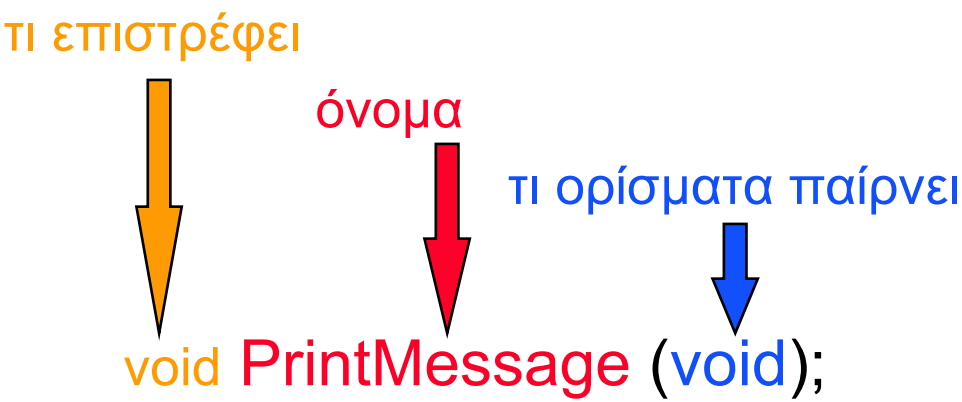
C

```
void PrintMessage (void)
{    (Ορίζουμε ποιες εντολές θέλουμε να εκτελεστούν με την κλήση)
    printf ("A message for you:\n\n");
    printf ("Have a Nice Day!\n");
}
```

Ορισμός Συνάρτησης

Α) Το πρότυπο συνάρτησης

- Πληροφορεί τον compiler ότι υπάρχει μία συνάρτηση που θα οριστεί αργότερα.
- Σημασία: επιτρέπει χρήση μιας συνάρτησης πριν ακόμα ορισθεί



- Καλός προγραμματισμός: λίστα με πρότυπα όλων των συναρτήσεων σε ένα πρόγραμμα

Β) Κλήση συνάρτησης

- Περνάει ο έλεγχος προγράμματος στη συνάρτηση
- Πρέπει να έχει το ίδιο όνομα, αριθμό και τύπο παραμέτρων

```
void PrintMessage (void)
{
        printf("Hello");
}

main ( )
{
        PrintMessage ( );
}
```

ίδιο όνομα

δεν υπάρχουν ορίσματα

Εκτυπώνει Hello στην οθόνη

C) Ορισμός συνάρτησης

- Στον ορισμό μιας συνάρτησης ορίζουμε ποιες ακριβώς εντολές θέλουμε να εκτελούνται

```
void PrintMessage (void)
{
        printf (“A message for you:\n\n”);
        printf (“Have a Nice Day\n”);
}
```

- Μόλις τελειώσει η συνάρτηση PrintMessage, ο έλεγχος επιστρέφει σε αυτόν που την κάλεσε, εδώ στη main ( )

C) Ορισμός συνάρτησης: Σύνταξη

Σύνταξη
<Τύπος τιμής εξόδου> <Όνομα Συνάρτησης> (<Λίστα Παραμέτρων>)

- <Τύπος τιμής εξόδου>:
  int, char, float, double, void, *σύνθετος*…

- <Όνομα Συνάρτησης>:
  συντακτικά ορθό όνομα

- <Λίστα Παραμέτρων>:
  άδεια (void) ή
  τύπος μεταβλητής, τύπος μεταβλητής, ….

Δομημένος Προγραμματισμός
Παράδειγμα 1

```
#include <stdio.h>
void PrintMessage (int);

main ( )
{
     int num;
     printf ("Enter an integer: ");
     scanf ("%d", &num);
     PrintMessage (num);
}

void PrintMessage (int counter)
{
     int i;
     for (i = 0; i < counter; i++)  {
               printf ("Have a nice day\n\n");
     }
}
```

Το πρόγραμμα εκτυπώνει :

Have a nice day

counter φορές

Δομημένος Προγραμματισμός
Παράδειγμα 1

```
/*        Τα σχόλια κάνουν τον κώδικα επεξηγηματικό
 * PrintMessage
 * Input: Some integer @counter
 * Functionality: Prints out “Have a nice day” @counter times
 * Output: void
 */
void PrintMessage (int counter)
{
    int i;
    for (i = 0; i < counter; i++)
    {
        printf (“Have a nice day\n”);
    }
}
```

Δομημένος Προγραμματισμός
Παράδειγμα 2

Ας δούμε ξανά το παράδειγμα του μη-δομημένου προγράμματος της διαφάνειας 11.2. Αυτή την φορά θα το δομήσουμε με συναρτήσεις

```
#include <stdio.h>

// PROTYPA SYNARTHSEWN
int mypower(int,int);
int myfactorial(int);

int main() {

    int a = 2;
    int b = 5;

    // KLHSH SYNARTHSEWN
    printf("pow(%d,%d)=%d \n", a, b, mypower(a,b));
    printf("fact(%d)=%d \n", b, myfactorial(b));
}
```

συνεχίζεται

Εκτυπώνει

pow(2,5)=32

fact(5)=120

Δομημένος Προγραμματισμός
Παράδειγμα 2

```
// Mypower υψωνει το val στην δυναμη pow
int mypower(int val, int pow) {
   int total = 1;    int i;
   for(i = 0; i < pow; i++) {
        total = total * val;
   }
   return total;
}
// Myfactorial επιστρεφει το παραγοντικο του val
int myfactorial(int val) {
   int factorial = 1; int i;
   if (val == 0)       factorial = 1;
   else
        for (i=1; i<=val; i++) {
                factorial *= i;
        }
   return factorial;
}
```