Functions

Rab Nawaz Jadoon

Assistant Professor

COMSATS IIT, Abbottabad

Pakistan

Department of Computer Science

DCS

COMSATS Institute of
Information Technology

What is a function???

- A function is a self-contained block of statements that perform a coherent task of some kind.
  - Every C program can be thought of as a collection of these functions.
    - As we noted earlier, using a function is something like hiring a person to do a specific job for you.
    - Sometimes the interaction with this person is very simple; sometimes it's complex.

Functions

- We will be looking at two things—
  - a function that calls or activates the function
  - and the function itself.

Example

```
main( )
{
      message( ) ;
      printf ( "\nCry, and you stop the monotony!" ) ;
}
message( )
{
      printf ( "\nSmile, and the world smiles with you..." ) ;
}
```

And here's the output...

```
Smile, and the world smiles with you...
Cry, and you stop the monotony!
```

Working of previous example

- Here, main( )itself is a function and through it we are calling the function message( ).
  - The activity of main( ) is temporarily suspended; it falls asleep while the message( ) function wakes up and goes to work.
  - When the message( )function runs out of statements to execute, the control returns to main( ), which comes to life again and begins executing its code at the exact point where it left off.
  - Thus, main( )becomes the 'calling' function, whereas message( )becomes the 'called' function.

Example (calling more than one functions

```
main( )
{
    printf ( "\nI am in main" ) ;
    italy( ) ;
    brazil( ) ;
    argentina( ) ;
}

    italy( )
    {
        printf ( "\nI am in italy" ) ;

    }
    brazil( )
    {
        printf ( "\nI am in brazil" ) ;
    }
    argentina( )
    {
        printf ( "\nI am in argentina" ) ;
    }
```

```
I am in main
I am in italy
I am in brazil
I am in argentina
```

Conclusion

- Any C program contains at least one function.
  - If a program contains only one function, it must be main( ).
  - If a C program contains more than one function, then one (and only one) of these functions must be main( ), because program execution always begins with main( ).
  - There is no limit on the number of functions that might be present in a C program.
  - Each function in a program is called in the sequence specified by the function calls in main( ).
  - After each function has done its thing, control returns to main( ).When main( )runs out of function calls, the program ends.

Example

- One function can call another function it has already called.

```
I am in main
I am in italy
I am in brazil
I am in argentina
I am back in italy
I am finally back in main
```

```
main( )
{
    printf ( "\nI am in main" ) ;
    italy( ) ;
    printf ( "\nI am finally back in main" ) ;
}
italy( )
{
    printf ( "\nI am in italy" ) ;
    brazil( ) ;
    printf ( "\nI am back in italy" ) ;
}
brazil( )
{
    printf ( "\nI am in brazil" ) ;
    argentina( ) ;
}
argentina( )
{
    printf ( "\nI am in argentina" ) ;
}
```

Important Points

- C program is a collection of one or more functions.
- A function gets called when the function name is followed by a semicolon. For example,

```
main( )
{
  argentina( ) ;
}
```

Important Points

- A function is defined when function name is followed by a pair of braces in which one or more statements may be present.
- For example,

```
argentina( )
{
    statement 1 ;
    statement 2 ;
    statement 3 ;
}
```

Important Points

- A function can be called any number of times. For example,

```
main( )
{
    message( ) ;
    message( ) ;
}
message( )
{
    printf ( "\nJewel Thief!!" ) ;
}
```

Important Points

- The order in which the functions are defined in a program and the order in which they get called need not necessarily be same. For example,

```
main( )
{
    message1( ) ;
    message2( ) ;
}
message2( )
{
    printf ( "\nBut the butter was bitter" ) ;
}
message1( )
{
    printf ( "\nMary bought some butter" ) ;
}
```

Important Points

- A function can call itself. Such a process is called 'recursion'.
- A function can be called from other function, but a function cannot be defined in another function.

```
main( )
{
    printf ( "\nI am in main" ) ;
    argentina( )
    {
        printf ( "\nI am in argentina" ) ;
    }
}
```

Types of Functions

- There are two types of functions,
  - Library functions
    - Printf(), scanf(), getche(), exit() etc
  - User defined functions
    - Display(), argentina(), getNumber() etc.

Why use functions???

- Writing functions avoids rewriting the same code over and over.
- Using functions it becomes easier to write programs and keep track of what they are doing.
  - If the operation of a program can be divided into separate activities, and each activity placed in a different function, then each could be written and checked more or less independently.
    - Separating the code into modular functions also makes the program easier to design and understand.

What is the moral of the story?

- Don’t try to cram the entire logic in one function.
  - It is a very bad style of programming.
  - Instead, break a program into small units and write functions for each of these isolated subdivisions.
  - Don’t hesitate to write functions that are called only once.
  - What is important is that these functions perform some logically isolated task

Passing Values between Functions

- This is what actually called communication between calling and called function.
  - The mechanism used to convey information to the function is the ‘argument’.
- You have unknowingly used the arguments in the printf( )and scanf( )functions;
  - The format string and the list of variables used inside the parentheses in these functions are arguments.
  - The arguments are sometimes also called ‘parameters’.

Program

```
//calculating sum using function
#include<stdio.h>
#include<conio.h>
int calsum(int, int, int); //function prototype
main()
{
int a, b, c, sum;
printf("Enter three Integer:  ");
scanf("%d %d %d", &a, &b, &c);
sum=calsum(a,b,c);
printf("the sum of %d, %d and %d is %d", a, b, c, sum);
getche();
}
int calsum(int x, int y, int z)
{
int d=x+y+z;
return(d);
}
```

Important points

- The variables a, b and c are called 'actual arguments',
- whereas the variables x, y and z are called 'formal arguments'.
  - Same name as actual arguments has can be used as formal variables.
- Any number of arguments can be passed to a function being called.
  - However, the type, order and number of the actual and formal arguments must always be same.

Purpose of return statement

- The return statement serve two purposes,
  - On executing the return statement it immediately transfers the control back to the calling program.
  - It returns the value present in the parentheses after return, to the calling program.
    - In the above program the value of sum of three numbers is being returned.

Remarks

- Whenever the control returns from a function some value is definitely returned.
  - If a meaningful value is returned then it should be accepted in the calling program by equating the called function to some variable.
  - For example, sum = calsum (a, b, c) ;

Important point

- If we want that a called function should not return any value, in that case, we must mention so by using the keyword ***void*** as shown below.

```
void display( )
{
 printf ( "\nHeads I win..." ) ;
 printf ( "\nTails you lose" ) ;
}
```

Important point

- A function can return only one value at a time. Thus, the following statements are invalid.
  - return (a, b) ;
  - return (x, 12) ;

Important point

- If the value of a formal argument is changed in the called function, the corresponding change does not take place in the calling function. For example,

```
main( )
{
    int  a = 30 ;
    fun ( a ) ;
    printf ( "\n%d", a ) ;
}

fun ( int  b )
{
    b = 60 ;
    printf ( "\n%d", b ) ;
}
```

```
60
30
```

Quiz 1

- Write a program using function to find out the factorial of a given number???

Solution

```
//Factorial of a number using function
#include<stdio.h>
#include<conio.h>
int factorial(int);  //function prototype
main()
{
int a, fact;
printf(" Enter a number");
scanf("%d", &a);
fact=factorial(a);
printf("The factorial of %d is %d", a, fact);
getche();
}
int factorial(a)
{
   int f=1;
   for(int i=1;i<=a;i++)
   {
   f=f*i;
   }
   return(f);
}
```

Quiz 2

- Write a program that find the area and circumference of a circle using two function,
  - Area()
  - Circumference()
- If the user input the radius of the circle r.


THANKS