Programming and Problem Solving through C Language
O Level / A Level

Chapter - 6 : Functions

Approach of Problem Solving

There are three general approaches to writing a program:

1. **Top down** - In the top down approach one starts with the top-level routine and move down to the low level routine.
2. **Bottom up** - The bottom-up approach works in the opposite direction on begins with the specific routines, build them into progressively more complex structures, and end at the top level routine.
3. **Ad hoc** - The ad hoc approach specifies no predetermined method.
4. C as a structured language lends itself to the top down approach. The top down method can produce clean readable code that can easily be maintained.

Top-down approach

- A top-down approach also helps one to clarify the overall structure and operation of the program before one code the low-level functions.
- The top down method starts with a general description and works towards specifics.
- A good way to design a program is to define exactly what the program is going to do at the top level.
- Each entry in the list should contain only one functional unit.
- A functional unit can be thought of as a black box that performs a single task.

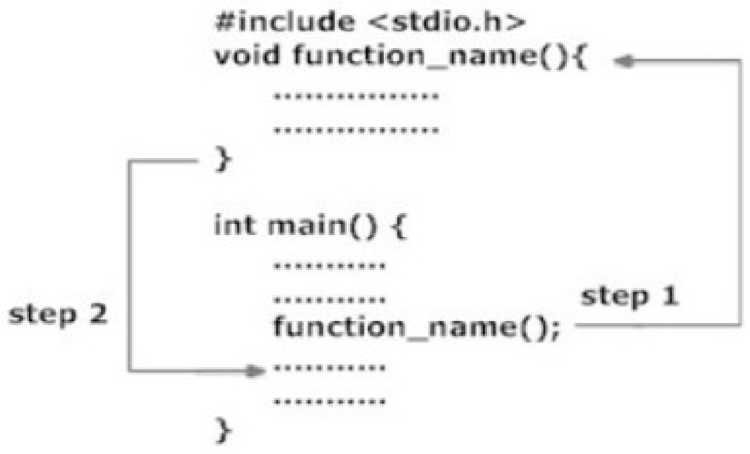
Modular programming

- Modular programming is a style that adds structure and readability to the program code.
- It may not make much difference on small projects, but as one starts to work on something bigger it can make the code much easier to read and maintain.
- Structuring the code is a simple task of splitting the program into manageable part so that each part is self-contained.
- By creating these self-contained modules, one can focus on programming each part.

Functions

- A function is a named, independent section of C code that performs a specific task and optionally returns a value to the calling program.
- A function is named. Each function has a unique name.
- By using the name in another part of the program, one can execute the statements contained in the function. This is known as **calling the function.**

- A function can be called from within any other function.
- A function is independent.
- A function can perform its task without interference from or interfering with other parts of the program.



```
#include <stdio.h>
void function_name(){
    ...............
    ...............
}

int main() {
    ...........
    ...........
    function_name();
    ...........
    ...........
}
```

Standard Library of C functions

- A collection of reusable functions (code for building data structures, code for performing math functions and scientific calculations, etc.), which will save C programmers time especially when working on large programming projects.
- The C Library is part of the ANSI (American National Standard Institute) for the C Language.
- The C programs can call on a large number of functions from the standard C library.
- These functions perform essential services such as input and output.
- They also provide efficient implementations of frequently used operations.
- Many macros and type definitions accompany these functions and help them to make better use of the standard C functions.
- Function prototype and data definitions of these functions are written in their respective header file.
- For example: If you want to use printf() function, the header file < stdio.h > should be included.
- In C programming you can find the square root by just using sqrt() function which is defined under header file "math.h".

Some list of the standard C Libraries.

• stdio.h	- Supports File Input/Output Operations.
• stdlib.h	- Supports Miscellaneous declarations.
• math.h	- Supports Definitions used for mathematical calculations.
• string.h	- Supports string functions.
• time.h	- Supports system time functions.
• ctype.h	- functions to handle characters (especially test characters).

C Library math.h functions

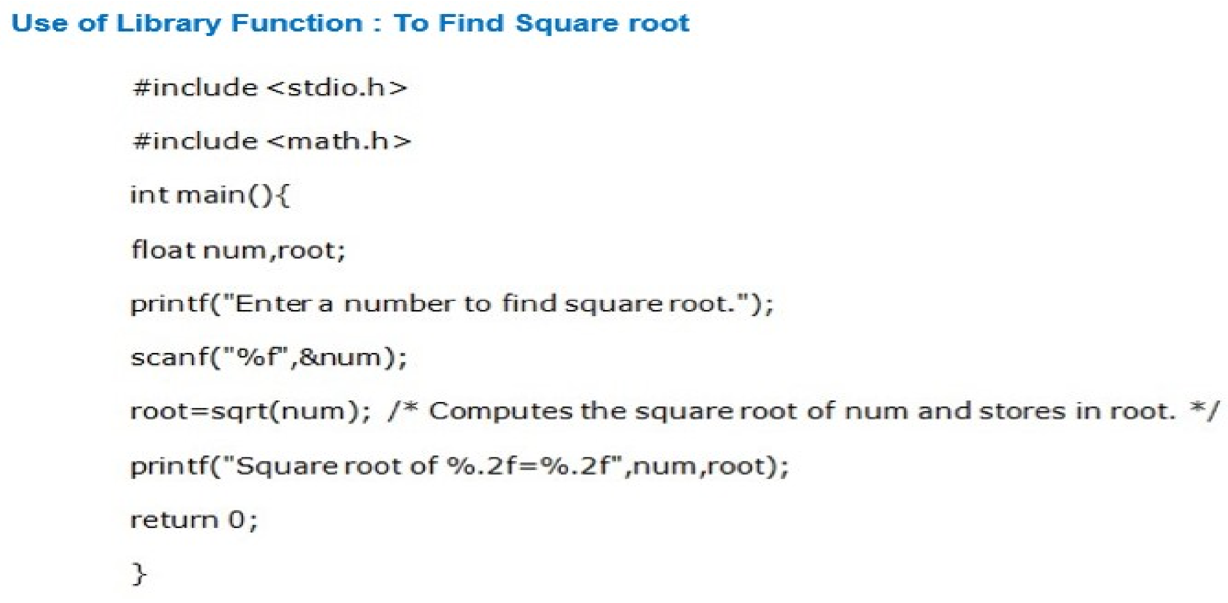
1. **double ceil(double x)**: It returns the smallest integer value greater than or equal to x.
2. **double floor(double x):** It returns the largest integer value less than or equal to x.
3. **double fabs(double x)**: It returns the absolute value of x.
4. **double log(double x)**: It returns the natural logarithm (base-e logarithm) of x.
5. **double log10(double x)**: It returns the common logarithm (base-10 logarithm) of x.
6. **double sqrt(double x)**: It returns the square root of x.
7. **double pow(double x, double y)**: It returns x raised to the power of y i.e. xy.

Example Programs of standard C Libraries

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

void main()
{
        int i = -10, e = 2, d = 10; /* Variables Defining and Assign values */
        float rad = 1.43;
        double d1 = 3.0, d2 = 4.0;

        printf("%d\n", abs(i));
        printf("%f\n", sin(rad));
        printf("%f\n", cos(rad));
        printf("%f\n", exp(e));
        printf("%d\n", log(d));
        printf("%f\n", pow(d1, d2));
}
```



Use of Library Function : To Find Square root

```
#include <stdio.h>
#include <math.h>
int main(){
float num,root;
printf("Enter a number to find square root.");
scanf("%f",&num);
root=sqrt(num); /* Computes the square root of num and stores in root. */
printf("Square root of %.2f=%.2f",num,root);
return 0;
}
```

C Library ctype.h functions

Sr.No.	Function	Description
1	int isalnum(int c)	This function checks whether the passed character is alphanumeric.
2	int isalpha(int c)	This function checks whether the passed character is alphabetic.
3	int iscntrl(int c)	This function checks whether the passed character is control character.
4	int isdigit(int c)	This function checks whether the passed character is decimal digit.
5	int isgraph(int c)	This function checks whether the passed character has graphical representation using locale.
6	int islower(int c)	This function checks whether the passed character is lowercase letter.
7	int isprint(int c)	This function checks whether the passed character is printable.
8	int ispunct(int c)	This function checks whether the passed character is a punctuation character.
9	int isspace(int c)	This function checks whether the passed character is white-space.
10	int isupper(int c)	This function checks whether the passed character is an uppercase letter.
11	int isxdigit(int c)	This function checks whether the passed character is a hexadecimal digit.

Example : The following program identifies the number of alphabets, digits:

```
#include <stdio.h>

// Header file containing character functions
#include <ctype.h>

void main()
{
   // String Initialization
   char a[]   = "Hi 1234,  Welcome to NIELIT Gorakhpur";
   int count_alpha = 0, count_digit = 0;

   for (int i = 0; a[i] != '\0'; i++) {
      // To check the character is alphabet
      if (isalpha(a[i]))
         count_alpha++;

      // To check the character is a digit
      if (isdigit(a[i]))
         count_digit++;
   }
   printf("The number of alphabets are %d\n",    count_alpha);
   printf("The number of digits are %d",    count_digit);
}
```