Functions

- INTRODUCTION TO FUNCTIONS
- TYPES OF FUNCTIONS
- ELEMENTS OF USER DEFINED FUNCTIONS
- TYPES ON THE BASIS OF ARGUMENTS AND RETURN VALUES
- METHODS OF CALLING A FUNCTION

Introduction to Function

- Block of statements that perform the particular task.
- Enables modular programming.
- Main() is the driver function.
- Has pre defined prototype.
- Same function can be accessed from different places within a program.
- Once a function execution is completed , control return to the place from where the function was called.

Advantages

- Modular Programming
- Length of source program can be reduced
- Easy to locate and isolate faulty function
- Functions can be used by other program's

Types of Functions

- Library (Built In) Functions:
  - They are written in the header files.
  - To use them appropriate header files should be included.

Header Files	Functions Defined
stdio.h	Printf(), scanf(), getchar(), putchar(), gets(), puts(), fopen(), fclose()
conio.h	Clrscr(), getch()
Ctype.h	Toupper(), tolower(), isalpha()
Math.h	Pow(), sqrt(), cos(), log()
Stdlib.h	Rand(), exit()
String.h	Strlen(), strcpy(), strupr()

- User Defined Functions
  - Written by the user at the time of programming.

Elements of User defined functions

- Function Prototype
- Function Call
- Function arguments and parameters
- Function Definitions

Function prototype

- It specify the type of value that is to be return from the function and that is to be passed to the function.
- It is defined in the beginning before the function call is made.
- Syntax:
  - return-type name-of-function(list of arguments);
  - Example
    - Void sum(int, int);

Function Call

- A function can be called by specifying name and list of arguments enclosed in parenthesis and separated by comma.
- If there is no arguments empty parenthesis are place after function name.
- If function return a value, function call is written as assignment statement as:
  - A=sum(x,y);

Function arguments and parameters

- Arguments are also called actual parameters.
- Arguments are written within parenthesis at the time of function call.

- Parameters are also called formal parameters.
- These are written within parenthesis at the time of function definition.

Function Definition

- It is the independent program module.
- It is written to specify the particular task that is to be performed by the function.
- The first line of the function is called function declarator and rest line inside { } is called function body

```c
#include<stdio.h>
#include<conio.h>
void sum(int,int); // function prototype
void main()
{
 int a,b;
 printf("Enter the two integers");
 scanf("%d%d",&a,&b);
 sum(a,b); // function call
 getch();
}
void sum(int a, int b) // function defination
{
 printf("The sum of the numbers you entered is %d",a+b);
}
```

Arguments

Parameters

Return statement

- It is the last statement of the function that return certain values.
- It return certain types of values to the place from where the function was invoked.
- Syntax:
  - return(variable-name or constant);

Categories of function

- Function with no arguments and no return
- Function with arguments but no return
- Function with no arguments and return
- Function with arguments and return

Function with no argument and no return

```
#include<stdio.h>
#include<conio.h>
void sum();
void main()
{
 sum();
 getch();
}
void sum()
{
 int a,b;
 printf("Enter the two integers");
 scanf("%d%d",&a,&b);
 printf("The sum of the numbers you entered is %d",a+b);
}
```

Function with argument and no return

```
#include<stdio.h>
#include<conio.h>
void sum(int,int);
void main()
{
 int a,b;
 printf("Enter the two integers");
 scanf("%d%d",&a,&b);
 sum(a,b);
 getch();
}
void sum(int a, int b)
{
 printf("The sum of the numbers you entered is %d",a+b);
}
```

Function with no argument and return

```
#include<stdio.h>
#include<conio.h>
int sum();
void main()
{
 int x;
 x=sum();
 printf("The sum of the numbers you entered is %d",x);
 getch();
}
int sum()
{
 int a,b;
 printf("Enter the two integers");
 scanf("%d%d",&a,&b);
 return(a+b);
}
```

Function with argument and return

```
#include<stdio.h>
#include<conio.h>
int sum(int,int);
void main()
{
 int a,b,x;
 printf("Enter the two integers");
 scanf("%d%d",&a,&b);
 x=sum(a,b);
 printf("The sum of the numbers you entered is %d",x);
 getch();
}
int sum(int a, int b)
{
 return(a+b);
}
```

Methods of calling function

- Call by value
- Call by reference

Call by value

- Copies the value of actual parameters into formal parameters.
- During execution whatever changes are made in the formal parameters are not reflected back in the actual parameters.

```c
#include<stdio.h>
#include<conio.h>
void swap(int,int);
void main()
{
 int a,b;
 printf("Enter the two integers\n");
 scanf("%d%d",&a,&b);
 printf("a=%d and b=%d before calling function\n",a,b);
 swap(a,b);
 printf("a=%d and b=%d after calling function\n",a,b);
 getch();
}
void swap(int a, int b)
{
 int temp;
 temp=a;
 a=b;
 b=temp;
 printf("a=%d and b=%d in function\n",a,b);
}
```

Call by Reference

- Reference(address) of the original variable is passed.
- Function does not create its own copy, it refers to the original values by reference.
- Functions works with the original data and changes are made in the original data.

```c
#include<stdio.h>
#include<conio.h>
void swap(int*,int*);
void main()
{
 int a,b;
 clrscr();
 printf("Enter the two integers\n");
 scanf("%d%d",&a,&b);
 printf("a=%d and b=%d before calling function\n",a,b);
 swap(&a,&b);
 printf("a=%d and b=%d after calling function\n",a,b);
 getch();
}
void swap(int* a, int* b)
{
 int temp;
 temp=*a;
 *a=*b;
 *b=temp;
}
```