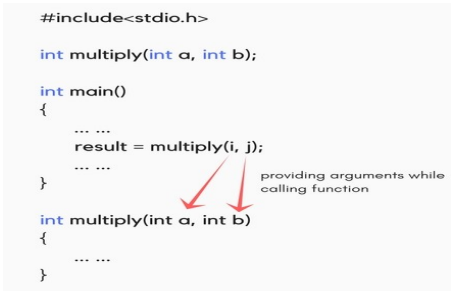
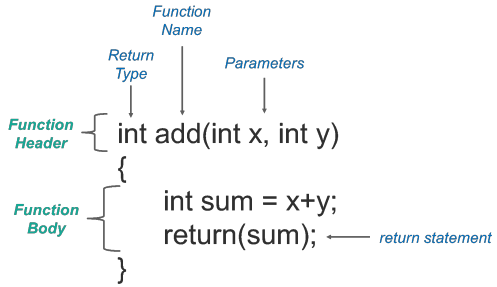
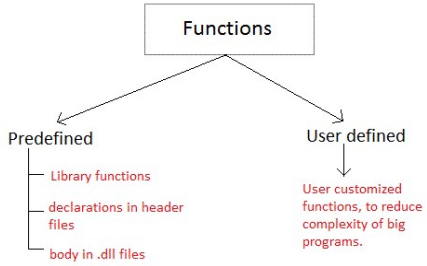
AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

CHAPTER - 5
FUNCTIONS







Subject : PPS **Code : 3110003**	**Prepared By:** **Asst. Prof. Rupali Patel** **(CSE Department, ACET)**	AMIRAJ COLLEGE OF ENGINEERING & TECHNOLOGY

Functions

A function is a group of statements that together perform a task. Every C program has at least one function, which is **main().**
A function **declaration** tells the compiler about a function's name, return type, and parameters. A function **definition** provides the actual body of the function.

Defining a Function
The general form of a function definition in C programming language is as follows −

```
return_type function_name( parameter list )
{
   body of the function
}
```

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

Functions (cont..)

Return Type − A function may return a value. The **return_type** is the data type of the value the function returns. Some functions perform the desired operations without returning a value. In this case, the return_type is the keyword **void**.

Function Name − This is the actual name of the function.

Parameters − A parameter is like a placeholder. When a function is invoked, you pass a value to the parameter. This value is referred to as actual parameter or argument. The parameter list refers to the type, order, and number of the parameters of a function. Parameters are optional; that is, a function may contain no parameters.

Function Body − The function body contains a collection of statements that define what the function does.

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

Function Declarations

A function **declaration** tells the compiler about a function name and how to call the function. The actual body of the function can be defined separately.

A function declaration has the following parts −
return_type function_name(parameter list);

For the above defined function max(), the function declaration is as follows −
int max(int num1, int num2);

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

Calling a Function

When a program calls a function, the program control is transferred to the called function.

A called function performs a defined task and when its return statement is executed or when its function-ending closing brace is reached, it returns the program control back to the main program.

To call a function, you simply need to pass the required parameters along with the function name, and if the function returns a value, then you can store the returned value.

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

Parameter Passing In C

In C, there are two types of parameters and they are as follows

- **Actual Parameters**
- **Formal Parameters**

The **actual parameters** are the parameters that are speficified in calling function.

The **formal parameters** are the parameters that are declared at called function. When a function gets executed, the copy of actual parameter values are copied into formal parameters.

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

Call by Value

In **call by value** parameter passing method, the copy of actual parameter values are copied to formal parameters and these formal parameters are used in called function.

The changes made on the formal parameters does not effect the values of actual parameters.

That means, after the execution control comes back to the calling function, the actual parameter values remains same. For example consider the following program

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

Call by Value Example

```
#include<stdio.h>
#include<conio.h>
void swap(int,int) ; // function
declaration
void main()
{
   int num1, num2 ;
   num1 = 10 ;
   num2 = 20 ;
   printf("\nBefore swap: num1 =
%d, num2 = %d", num1, num2) ;
      swap(num1, num2) ;
      printf("\nAfter swap: num1 =
%d\nnum2 = %d", num1, num2);
   getch() ;
}

void swap(int a, int b)  // called
function
{
   int temp ;
   temp = a ;    temp=10;
   a = b ;       a=20;
   b = temp ;      b=10;
 }
```

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

Call By Reference

In **Call by Reference** parameter passing method, the memory location address of the actual parameters is copied to formal parameters.

This address is used to access the memory locations of the actual parameters in called function.

In this method of parameter passing, the formal parameters must be **pointer** variables.That means in call by reference parameter passing method, the address of the actual parameters is passed to the called function and is recieved by the formal parameters (pointers).

Whenever we use these formal parameters in called function, they directly access the memory locations of actual parameters.

the changes made on the formal parameters effects the values of actual parameters.

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

Call By Reference Example

```
#include<stdio.h>
#include<conio.h>
void swap(int *,int *) ; //
        function declaration
 void main()
{
   int num1, num2 ;

   num1 = 10 ;
   num2 = 20 ;
   printf("\nBefore swap: num1 =
%d, num2 = %d", num1, num2) ;
   swap(&num1, &num2) ; //
        calling function
   printf("\nAfter swap: num1 =
%d, num2 = %d", num1, num2);
   getch() ;
}
void swap(int *a, int *b)  //
        called function
{
   int temp ;
   temp = *a ;
   *a = *b ;
   *b = temp ;
}
```

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

C Macros

A *macro* is a fragment of code which has been given a name.

Whenever the name is used, it is replaced by the contents of the macro.

There are two kinds of macros. They differ mostly in what they look like when they are used.

A macro is a segment of code which is replaced by the value of macro. Macro is defined by #define directive.

There are two types of macros:

1. Object-like Macros
2. Function-like Macros

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

C Macros (cont..)

Object-like Macros

The object-like macro is an identifier that is replaced by value. It is widely used to represent numeric constants.

For example:

#define PI 3.14

Here, PI is the macro name which will be replaced by the value 3.14.

Function-like Macros

The function-like macro looks like function call.

For example:

#define circleArea(r) (3.1415*(r)*(r))

Every time the program encounters circleArea(argument), it is replaced by (3.1415*(argument)*(argument))

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

Pre-processor

The **C Preprocessor** is not a part of the compiler, but is a separate step in the compilation process.

In simple terms, a C Preprocessor is just a text substitution tool and it instructs the compiler to do required pre-processing before the actual compilation.

All preprocessor commands begin with a hash symbol (#).

It must be the first nonblank character, and for readability, a preprocessor directive should begin in the first column.

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

Pre-processor (cont..)

Analyze the following examples to understand various directives.
#define MAX_ARRAY_LENGTH 20

This directive tells the CPP to replace instances of MAX_ARRAY_LENGTH with 20. Use *#define* for constants to increase readability.

#include <stdio.h>
#include "myheader.h"

These directives tell the CPP to get stdio.h from **System Libraries** and add the text to the current source file. The next line tells CPP to get **myheader.h** from the local directory and add the content to the current source file.

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY