Chapter - Function in C Programming, PPT, Semester, Engineering

Introduction

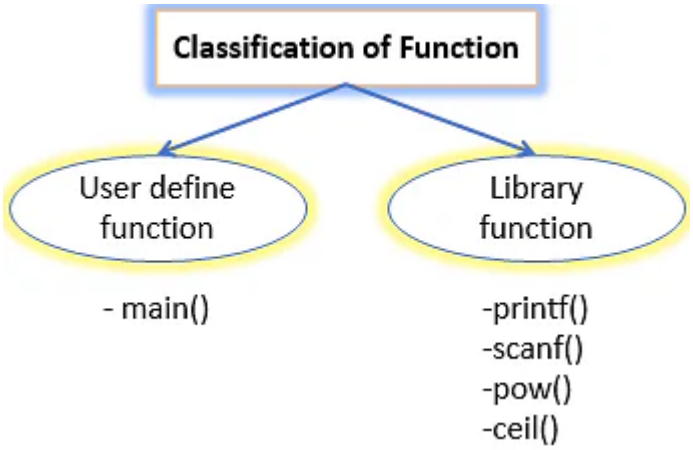
The slides cover the following topics:

1. Functions
2. Advantages of Functions
3. Function Prototypes
4. Function Definitions
5. Function Return Types
6. Types of Functions
7. Pointers

Functions in C Programming ----------Next slide

What is a function

1. A large program in c can be divided to many subprogram
2. The subprogram posses a self contain components and have well define purpose.
3. The subprogram is called as a *function*
4. Basically a job of *function* is to do something
5. C program contain at least one function which is main().



Functions in C Programming ----------Next slide

Advantages of function

1. It is much easier to write a structured program where a large program can be divided into a smaller, simpler task.
2. Allowing the code to be called many times
3. Easier to read and update
4. It is easier to debug a structured program where there error is easy to find and fix

Functions in C Programming ----------Next slide

Example

Functions in C Programming ----------Next slide

How the function works

1. C program doesn't execute the statement in function until the function is called.
2. When function is called the program can send the function information in the form of one or more argument.

3. When the function is used it is referred to as the ***called function***
4. Functions often use data that is passed to them from the ***calling function***
5. Data is passed from the calling function to a called function by specifying the variables in a argument list.
6. **Argument** list cannot be used to send data. Its only copy data/value/variable that pass from the calling function.
7. The called function then performs its operation using the copies.

Functions in C Programming ----------Next slide

Function prototypes

1. Provides the compiler with the description of functions that will be used later in the program
2. Its define the function before it been used/called
3. Function prototypes need to be written at the beginning of the program.
4. The function prototype must have :
   A return type indicating the variable that the function will be return

Syntax for Function Prototype

return-type function_name(arg-type name-1,...,arg-type name-n);

Function Prototype Examples

1. double squared(double number);
2. void print_report(int report_number);
3. int get_menu_choice(void);

Functions in C Programming ----------Next slide

Function Definitions

Functions in C Programming ----------Next slide

Function Definition Examples

```
float conversion (float celsius)
{
    float fahrenheit;
    fahrenheit = celcius33.8
    return fahrenheit;
 }
```

The function name's is conversion

This function accepts arguments celcius of the type float. The function return a float value.

So, when this function is called in the program, it will perform its task which is to convert fahrenheit by multiply celcius with 33.8 and return the result of the summation.

Note that if the function is returning a value, it needs to use the keyword return.

Functions in C Programming ----------Next slide

Function return types

Can be any of C's data type:
char
int
float
long……

Examples:

```
int func1(...)        / Returns a type int.  /
```

```
float func2(...)      / Returns a type float. /
void func3(...)       / Returns nothing.     /
```

Functions in C Programming ----------Next slide

Types of Functions

Function can be divided into 4 categories:

1. A function with no arguments and no return value
2. A function with no arguments and a return value
3. A function with an argument or arguments and returning no value
4. A function with arguments and returning a values

Functions in C Programming ----------Next slide

A function with no arguments and no return value

1. Called function does not have any arguments
2. Not able to get any value from the calling function
3. Not returning any value
4. There is no data transfer between the calling function and called function.

```
#include<stdio.h>
#include<conio.h>
void printline();

void main()
{
  printf("Welcome to function in C");
  printline();
  printf("Function easy to learn.");
  printline();
```

```
  getch();
}

void printline()
{
  int i;
  printf(" ");
  for(i=0;i<30;i++)
  {     printf("-");   }
  printf(" ");
}
```

Functions in C Programming ----------Next slide

A function with no arguments and a return value

1. Does not get any value from the calling function
2. Can give a return value to calling program

```
#include <stdio.h>
#include <conio.h>
int send();

void main()
{
  int z;
  z=send();
  printf(" You entered : %d.",z);
  getch();
}

int send()
{
```

```
  int no1;
  printf("Enter a no: ");
  scanf("%d",&no1);
  return(no1);
}
```

Functions in C Programming ----------Next slide

A function with an argument or arguments and returning no value

1. A function has argument/s
2. A calling function can pass values to function called , but calling function not receive any value
3. Data is transferred from calling function to the called function but no data is transferred from the called function to the calling functionGenerally Output is printed in the Called function
4. A function that does not return any value cannot be used in an expression it can be used only as independent statement.

Functions in C Programming ----------Next slide

```
#include<stdio.h>
#include<conio.h>
void add(int x, int y);

void main()
{
  add(30,15);
  add(63,49);
  add(952,321);
  getch();
```

```
}

void add(int x, int y)
{
  int result;
  result = x+y;
  printf("Sum of %d and %d is %d. ",x,y,result);
}
```

Functions in C Programming ----------Next slide

A function with arguments and returning a values

1. Argument are passed by calling function to the called function
2. Called function return value to the calling function
3. Mostly used in programming because it can two way communication
4. Data returned by the function can be used later in our program for further calculation.

Functions in C Programming ----------Next slide

```
#include <stdio.h>
#include <conio.h>
int add(int x,int y);

void main()

{
  int z;

  z=add(952,321);
  printf("Result %d. ",add(30,55));
```

```
  printf("Result %d. ",z);

  getch();
}
int add(int x,int y)

{
  int result;
  result = x + y;
  return(result);
}
```

Send 2 integer value x and y to add()

Function add the two values and send back the result to the calling function

int is the return type of function

Return statement is a keyword and in bracket we can give values which we want to return.

Functions in C Programming ----------Next slide

Pointers

Variable that declared occupies a memory according to it size

It has address for the location so it can be referred later by CPU for manipulation

The ‘’ and ‘&’ Operator

Int x= 10

We can use the address which also point the same value.

Functions in C Programming ----------Next slide

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


void main()

{
  int i=9;

  printf("Value of i : %d ",i);
  printf("Adress of i %d ", &i);

  getch();
}
```

Functions in C Programming ----------Next slide

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


void main()

{
  int i=9;

  printf("Value of i : %d ",i);
  printf("Address of i %d ", &i);
```

```
  printf("Value at address of i: %d", (&i));

  getch();
}
```

Functions in C Programming ----------Next slide

Passing arguments by value or reference

Functions in C Programming ----------Next slide

Passing arguments by value or reference

```
#include <stdio.h>
#include <conio.h>
void callByValue(int, int);
void callByReference(int , int );


int main()
{
  int x=10, y =20;
  printf("Value of x = %d and y = %d. ",x,y);

  printf(" CAll By Value function call... ");
  callByValue(x,y);
  printf(" Value of x = %d and y = %d. ", x,y);

  printf(" CAll By Reference function call... ");
  callByReference(&x,&y);
```

```
  printf("Value of x = %d and y = %d. ", x,y);

  getch();
  return 0;
}
```

Functions in C Programming ----------Next slide

```
void callByValue(int x, int y)

{
  int temp;
  temp=x;
  x=y;
  y=temp;

  printf(" Value of x = %d and y = %d inside callByValue function",x,y);
}

void callByReference(int x, int y)

{
  int temp;
  temp=x;
  x=y;
  y=temp;

}
```

Functions in C Programming ----------Next slide