UNIT -III INTRODUCTION TO C++ CHAPTER 11

Functions

1.Define Functions. What is functions ? What ate the advantage of functions in C++?

- A large program can be split into small **sub-programs** (blocks) called as **functions**.
- Functions are the **executable segments** in a program.
- Functions are the **building blocks** of C++ programs

Advantage of function

- **Reduce** the size of the program.
- **Reusability** of code

Functions can be classified into two types.

1.Pre - Defined or Build – in or Library functions.

2.User - Defined Functions.

2.Differentiate between Pre - Defined and User - Defined Functions.

Pre - Defined	**User - Defined Functions.**
These are already written, debugged and compiled for various task.	Create new functions to perform specific task by the user.
Definition are stored in Header Files.	The name of task and data required are decided by the user.
Ex. gets(),strlen()	Ex. int add(int r);

3.Define Header file.

- Declaration and definitions for pre-defined functions are grouped and stored in files called Header files.
- It is also called as **Library** files.
- Their file extension is **.h**
- A single header file may contain multiple pre defined functions.

Ex.stdio.h , iostream.h ,conio.h , string.g , iomanip.h

4.Explain some Standard input/output(stdio.h) predefined functions in C++.

The header file **stdio.h** is to be included to **use Standard input/output** functions in a program

getchar()

- It is used to get a single character from keyboard

putchar()

- It is used to to display a single character.

gets()

- It reads a string from standard input and stores it into the variable.
- It treats spaces as part of string

puts()

- It prints the string read by **gets()** function in a newline.

5.Explain few Character functions(ctype.h) in C++.

- The header file **ctype.h** is to be included to use Character functions in a program.

isalnum()

- It is used to check whether a character is **an alphanumeric or not.**
- It returns **1** if a character is an alphanumeric. Otherwise it returns **0**.

Syntax: int isalnum (char c);

isalpha()

- It is used to check whether a character **is an alphabet or not.**
- It returns **1** if a character is an alphabet . Otherwise it returns **0**.

Syntax: int isalpha (char c);

isdigit()

- It is used to check whether a character is **a digit or not.**
- It returns **1** if a character is a digit. Otherwise it returns **0**.

Syntax :int isdigit (char c);

islower()

- It is used to check whether a character **is in lower case or not.**
- It returns **1** if a character is a lower case . Otherwise it returns **0**.

Syntax: int islower (char c);

isupper()

- It is used to check whether a character **is in upper case or not.**
- It returns **1** if a character is a upper case . Otherwise it returns **0.**

Syntax :int isupper (char c);

6.Explain some string functions in C++.(string.h)

strcpy()

- The strcpy() function takes two arguments: **target and source.**
- It copies the **character string** by the **source** to the **memory location** by the **target**.
- The null character (\0) is also copied.

Syntax: strcpy(target,source);

strlen()

- The strlen() returns **length of a character**.
- The length **does not include the null(\0)** character.

Syntax: strlen(source);

strcmp()

- It is used to **compares**(ASCII values are compared) the **two given strings**.
    - if string1 = string2 (equal) it returns 0
    - if string1 **>** string2 it returns 1(Positive)
    - if string1**<** string2 it returns -1 (Negative)

syntax: strcmp(string 1,string2);

strcat()

- It is used to **merge two strings**: target and source

syntax: strcat(target,source);

strupr()

- The strupr() function is used to **convert** the **given string into Uppercase letters.**

strlwr()

- The strlwr() function is used to **convert the given string into Lowercase letters.**

7.Explain some of Mathematical functions (math.h) in C++.

Mathematical functions are defined in **math.h** header file .

cos()

- The cos() function takes **a single argument in radians.**
- It returns the value in the range of [-1, 1].
- The returned value is either in double, float, or long double.

sqrt()

- The sqrt() function returns the **square root** of the given value of the argument.
- It takes a single positive value only
- otherwise , a domain error occurs.

sin()

- The sin() function takes **a single argument in radians.**
- It returns the value in the range of [-1, 1].
- The returned value is either in double, float, or long double.

pow()

- The pow() function returns **the power of exponent**.
- If any value passed to pow() is long double, the return type is promoted to long double.
    - If not, the return type is double.
    - The pow() function takes two arguments:

  1. base 2. exponent

Syntax: pow(base,exponent) ;Ex . pow(3,2);

8.How to generate Random Numbers in C++

- strand() and rand() are used to generate Random Numbers.
- By default the **seed for rand()** is **1**.
- They are defined in **<cstdlib.h>** or **<stdlib.h)**

9.Define Function Definition.

- A function must be defined before it is used anywhere in the program.

Syntax:

```
Return_Data_Type   Function_name(parameter list)
{
  Body of the function
}
```

10.Define function prototype.

- **Functions** should be **declared** before they are used in a program.
- function **prototype** is used to **declare** a function .
- The declaration statement may be **given outside the main()** function.
- It helps the compiler to check the **data requirement** of the function.

syntax

Return type function name(arguments);

11.What are the information the prototype provides to the compiler ?

The prototype provides the following information to the compiler:

1.Number and type of arguments

2.The type of return values

3.Name of the function.

syntax

Return type function name(arguments);

12.Write the information to the compiler from the following prototype. long fact(int a ,double b);

- The return data type is **long**.
- **fact** is the name of the function.
- The function is called with **two** arguments:
- The first argument is of **int** data type.
- The second argument is of **double** data type.

13.What are the uses of void ?

- To indicate the function does not return a value
- To declare a generic pointer.

Ex. void fun(void)

In above example, fun function neither receives value from calling nor return value to the calling statement,

14.Differentiate between actual parameters and formal parameters.

Actual parameters	**Formal parameters**
Parameters associated with **call statement** is called actual parameters.	The parameters associated with **function header** is called formal parameter.
It is within calling statement.	It is within function definition
Only variables are used	The constant, variables or expressions are used.

15.How to access a function?

- A function can be called or invoked from another function by using its **name** and the **required** arguments.
- The compiler refers to the **function prototype** to check whether the function has been called correctly.
- If the argument type **does not match exactly** with the prototype, the compiler will perform **type conversion**, if possible.
  otherwise, the compiler generates **an error message.**

Ex . swap(x,y); `

16.Explain Default arguments with an example.

- The default value is given in the form of **variable initialization.** Ex : **void area (int n1=10, n2=100);**
- The default arguments facilitate the function call statement with **partial or no arguments**.
- The default values can be included in the function prototype **from right to left,**
- Default value **cannot be include** between the argument list.

Ex : void area (int n1=10, n2);//invalid prototype
 void area(int n1, n2 = 10);//valid prototype

Example

```
#include <iostream >
using namespace std;
int  area (int l = 10, int b=20)
{ return (l * b); }
void main ( )
{ int s1 = 4, s2 = 6;
cout <<area (s1);
}
```

```
Output:

80
```

17.Define constant argument. What is const modifier?

- The constant variable can be declared using **const** keyword.
- The constant variable should be **initialized while declaring.**
- The const modifier enables to assign **an initial value to a variable** that **cannot be changed** later inside the body of the function.

Syntax : **const** datatype variable=value;
Example: **const** int a=10;

18.Explain the Methods of calling functions. What are the different ways of passing parameters in C++ functions?

- The call statement communicates with the function through arguments or parameters.
- There are two ways of passing parameters in C++ functions .

1.Call by value 2. Call by reference.

19.Explain call by value in C++ with an example.

- In this method, the formal parameter **creates new variables** and **stores** the value from actual parameter
- This method **copies** the values of actual parameters into the formal parameters
- **Any change** in the formal parameter is **not reflected** back to the actual parameter.

Ex.

```
#include <iostream >
using namespace std;
void swap (int a)
{ a=8;
cout << '\n'<< a;
}
int main ( )
{
int m1 = 10;
cout <<m1 ;
swap (m1);
cout << '\n'<< m1;
}
```

Note:
m1 -> Actual parameter
a -> Formal Parameter

20.Explain call by reference in C++ with an example

- In this method, formal parameters become **alias** to the actual parameters.
- It is working on the **original data**.
- **Any change** made in the formal parameter is **reflected** back in the actual parameter

Ex.

```
#include <iostream >
using namespace std;

void swap (int &a)
{ a=8;
cout << '\n'<<a;
}
int main ( )
{
int m1 = 10;
cout <<m1 ;
swap (m1);
cout << '\n'<< m1;
}
```

```
Output:

10

8

8
```

Note:
m1 -> Actual parameter
a -> Formal Parameter

21.Explain inline function with example

- An inline looks like a normal function in the source file but **inserts** the function's **code directly** into the **calling program**.
- To make a function inline, insert the keyword **inline** in the **function header** (Ex inline void swap (int a))
- inline keyword is just a request to the compiler Sometimes the compiler will **ignore** the request

Advantages of inline functions:

- Inline functions execute **faster** but requires **more memory space.**
- **Reduce** the **complexity** of using STACKS.

Ex.

```
#include <iostream >
using namespace std;
inline void swap (int a)
{ a=8;
cout << '\n'<< a;
}
int main ( )
{
int m1 = 10;
cout <<m1 ;
swap (m1);
cout << '\n'<< m1;
}
```

22.What are the different forms of user defined function ?

1. Function without return value and without parameter

```
#include<iostream.h>
void display()
{
  cout<<"No return value& without parameter ";
}
void  main()
{
  display();
  }
```

In the above program, The name of the function is **display(),** its return data type is **void** and it **does not have any argument.**

2.A Function with return value and without parameter

```
#include <iostream >
using namespace std;
int  display()
{
Int a =10;
  cout<<"With return value& without parameter ";
return a;
}
void  main()
{
 cout<<display();
 }
```

- The name of the function is display(),
- its return type is **int** and it **does not have any argument.**
- The return statement returns a value of a **(10**) to the calling function .

3. A Function without return value and with parameter

```
#include <iostream >
using namespace std;
void  display( int a)
{
cout<<a;
  cout<<"Without return value& with parameter ";
}
int  main()
{
int x=10;
 display(a);
  }
```

The name of the function is display(), its return type is void and it has one parameters **(int a).**

4. A Function with return value and with parameter

```
#include <iostream >
using namespace std;
int  display( int a)
{
Int b=5
Int c=a+b;
  cout<<"With return value& with parameter ";
return c;
}
void  main()
{
int x=10;
cout<< display(a);
  }
```

The name of the function is **display(),** its return type is **int** and it has **one parameter**.
The return statement returns with **c** value to the calling statement.

23.Define return statement in C++.

- The return statement is used to return from a function to the calling function.
- It is a jump statement.
- A return may or may not have a value associated with it.
- A return statement without parameter can be used to terminate the function.

Syntax: **return expression/variable;**

Example : **return(a+b); return(a); return;**

24.Explain the returning value in C++?

- The functions that return no value is declared as **void**.
- Default return data type is **int**.
- if no data type is explicitly mentioned, it is treated as **int**.

Ex. int add (int, int); add (int, int);
In both prototypes, the return value is **int**,
float area(float); float
char name(); char

25.Explain the Returning by reference in C++?

```
#include <iostream >
using namespace std;
int main()
{
int a=150;
int &b=a;
cout<<a<<'\t'<<b;
b++;
cout<<'\n'<<a<<'\t'<<b;
}
```

- The variable **b** is alias to **a**.
- Hence the value of b is altered automatically when the value of a is changed.
- The two variables **a,b shares same memory** or reference.

26.What is Recursive function ?

A function that calls itself is known as recursive function.
Example:

```
int add(int a,int b)
{
……
…..add(a,a+b);

}
void main()
{
add(x,y);
}
```

27.Explain about the different scopes of a variable in C++ with an example.

- Scope refers to the accessibility of a variable.

There are four types of scopes in C++.
They are: 1. Local scope, 2. Function scope, 3. File scope 4.Class scope

1. Local scope

- A local variable is defined **within a local block.**
- A local variable **cannot be accessed** from **outside** the block of its declaration.
- A block of code begins and ends with **curly braces{ }.**
- It is created upon entry into its block and destroyed upon **exit** .

2. Function scope

- The variable declared **within a function**
- The scope of variable is extended to **the function block,** and all **sub-blocks**.
- The life time of a function scope variable, is the life time of the function block.
- Formal parameters is a function scope

3. File scope

- The file scope variable is also called as **global** variable.
- The file scope of variable declared above **main ( ).**
- The life time of a file scope variable is **the life time of a program.**

```
#include <iostream >
using namespace std;
Int b=10;
void swap (int a)
{ int c = a+b;
cout << c;
}
int main ( )
{
int m1 = 10;
cout <<m1 ;
swap (m1);
cout << '\n'<< m1;
}
```

Here,
a - Function scope variable b - File scope variable
c - Local variable

4. Class scope

- A class is a new way of creating and implementing a user defined data type.
- Access specifiers are , Private , protected and public.

```
class name
{
Private:
{ declaration; }
Protected:
{ declaration; }
Public:
{ declaration; }
};
```

28.Explain the use of scope operator with an example?

- **::** is called scope resolution operator
- It is used to refer variables declared at **file level**.
- It is used when the local and file scope variables have the **same name.**

Example:
```
#include <iostream >
using namespace std;
int x=45;
int main() {
int x = 10;
cout << ::x + x; }       Output: 55
```