Functions in C++

By – Mohammad Imran

Function

Function is a block of code that has a name and used to perform some specific task. The function contains the set of programming statements enclosed by curly bracket { }. A function can be called multiple times, so it is reusable.

Using the function we can divide a large program into small blocks known as function.

A function is essentially a set of instructions designed to perform a specific task. It accepts input, carries out computations, and generates output.

By – Mohammad Imran

Function

Need

- ✓ **Reduces Code Redundancy**: Functions avoid repetition by encapsulating repeated tasks, promoting efficient maintenance.
- ✓ **Enhances Modularity**: Segments extensive code into manageable units, improving readability and usability.
- ✓ **Provides Abstraction**: Encapsulates complex logic, allowing use of library functions without delving into implementation details.
- ✓ **Streamlines Maintenance**: Centralizes modifications, requiring changes in only one place, simplifying maintenance efforts.

By – Mohammad Imran

Function

Aspects

There are three aspects of a C function.

Function Declaration A function must be declared globally in a c program to tell the compiler about the function name, function parameters, and return type.

Function Call Function can be called from anywhere in the program. The parameter list must not differ in function calling and function declaration. We must pass the same number of functions as it is declared in the function declaration.

Function Definition It contains the actual statements which are to be executed. It is the most important aspect to which the control comes when the function is called. Here, we must notice that only one value can be returned from the function.

By – Mohammad Imran

Function

Types

There are two types of function in C programming

- ✓ Predefined Function (Library Function)
- ✓ User Defined Function

By – Mohammad Imran

Library Functions in C++

Standard Library Functions

Significance of Standard Library Functions

Common Header Files

By – Mohammad Imran

Function

Library Function

Standard Library Functions are basically the inbuilt functions in the C++ compiler that makes things easy for the programmer.

Library functions are built-in functions that are grouped together and placed in a common location called library.

By – Mohammad Imran

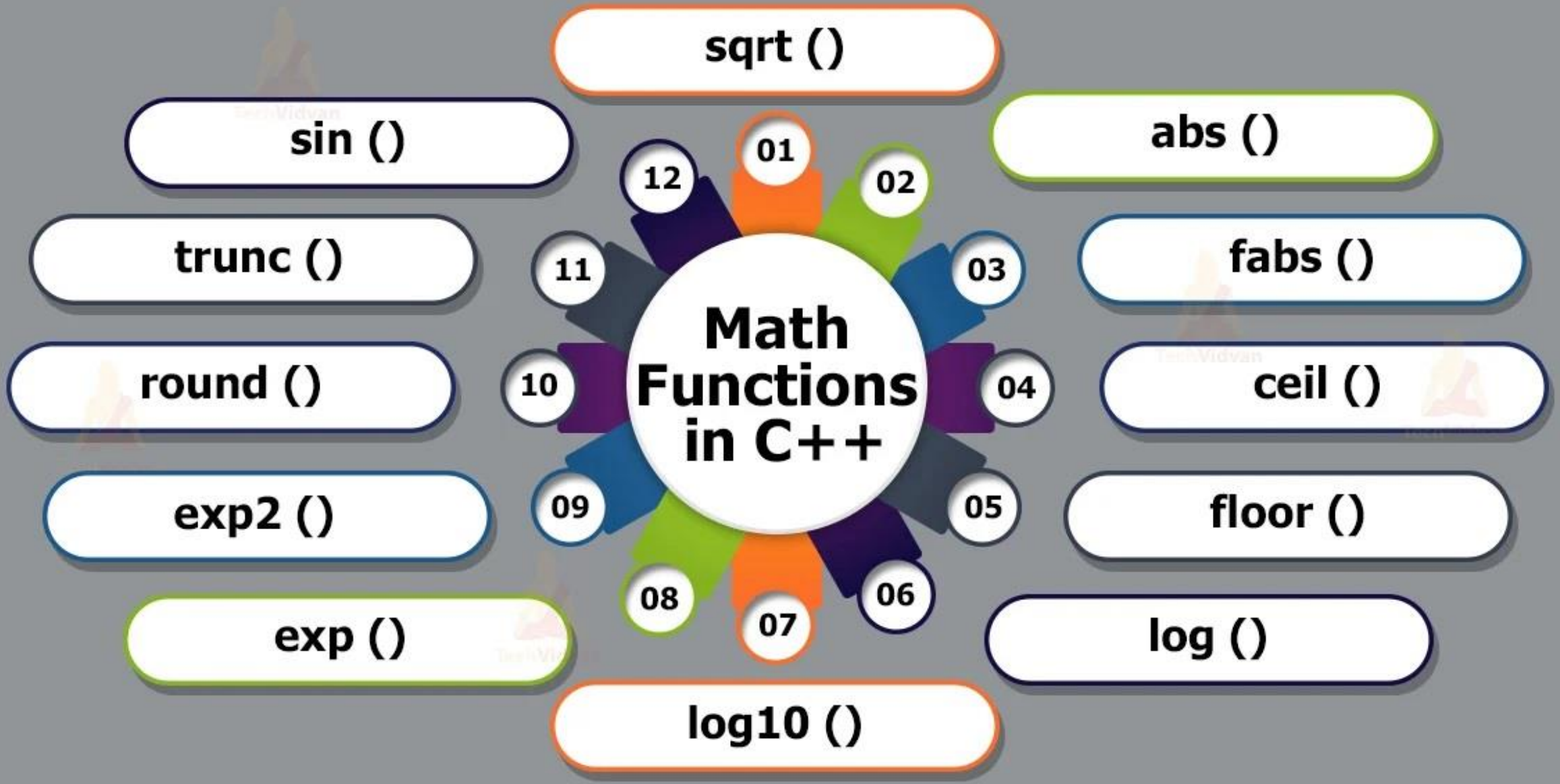
Library Function

Header Files

The library contains the function implementation, and by including the header file for the corresponding library, we can use the required function in our program. The tabular description of some of the standard library header files is presented below.

- ✓ cstdlib
- ✓ ctime
- ✓ iomanip
- ✓ string
- ✓ climits

By – Mohammad Imran



By – Mohammad Imran

Library Function

Math Function

C++ being a superset of C, supports a large number of useful mathematical functions. These functions are available in standard C++ to support various mathematical calculations.

Instead of focusing on implementation, these functions can be directly used to simplify code and programs. In order to use these functions you need to include a header file- **<math.h>** or **<cmath>**.

C++ provides a large set of mathematical functions which are stated below:

By – Mohammad Imran

Library Function

Math Function

- ✓ **abs (number)** – Returns the absolute value of given number.
- ✓ **ceil (number)** – Rounds up the given number. It returns the integer value which is greater than or equal to given number.
- ✓ **floor (number)** – rounds down the given number. It returns the integer value which is less than or equal to given number.
- ✓ **pow (d1,d2)** – Returns the power of given number.

By – Mohammad Imran

Math Function

sqrt()

double sqrt (double)

This function takes a number as an argument and returns its square root value. The number can not be a negative value.

The **<cmath>** header defines two more inbuilt functions for calculating the square root of a number (apart from sqrt()) which has an argument of type **float** and **long double**.

By – Mohammad Imran

Math Function

sqrt()

double sqrt (double)

The sqrt() function returns the square root of a number of type double.

Syntax

```
double sqrt(double arg);
```

By – Mohammad Imran

Math Function | sqrt() | Example - 1

```
#include <cmath>
#include <iomanip>
#include <iostream>
using namespace std;
int main()
{
   int n = 25;
   double val1 = 225.0;
   double val2 = 300.0;
   int sqr = sqrt(n);
   cout << sqr << endl;
   cout << fixed << setprecision(2) << sqrt(val1) << endl;
   cout << fixed << setprecision(4) << sqrt(val2) << endl;
   return (0);
}
```

OUTPUT

```
5
15
17.3204
```

By – Mohammad Imran

Math Function

sqrtf()

float sqrtf (float)

The **sqrtf()** function returns the square root of a number of type double.

Syntax

```
float sqrtf(float arg);
```

By – Mohammad Imran

Math Function

sqrtf()

Example - 2

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

int main()
{
   float val1 = 225.0;
   float val2 = 300.0;
   cout << fixed << setprecision(12) << sqrtf(val1) << endl;
   cout << fixed << setprecision(12) << sqrtf(val2) << endl;
   return (0);
}
```

OUTPUT

```
15.0000
17.3205
```

By – Mohammad Imran

Math Function

sqrtl()

long double sqrtl (long double)

The **sqrtl()** function returns the square root of a number of type double.

Syntax

```
long double sqrtl(long double arg);
```

By – Mohammad Imran

Math Function | sqrtl() | Example - 3

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

int main()
{
   long double var1 = 10000000000000000;
   long double var2 = 99999999999999999;
   cout << fixed << setprecision(12) << sqrtl(var1) << endl;
   cout << fixed << setprecision(12) << sqrtl(var2) << endl;
   return 0;
}
```

OUTPUT

```
100000000.000000000000
999999999.999999999476
```

By – Mohammad Imran

Math Function

Miscellaneous | **Example - 4**

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

int main()
{
   cout << abs(-12) << endl;
   cout << fixed << setprecision(2) << ceil(3.6) << endl;
   cout << fixed << setprecision(2) << ceil(3.3) << endl;
   cout << fixed << setprecision(2) << floor(3.6) << endl;
   cout << fixed << setprecision(2) << floor(3.3) << endl;
   cout << fixed << setprecision(2) << pow(2,4) << endl;
   return 0;
}
```

OUTPUT

```
12
4.00
4.00
3.00
3.00
16.00
```

By – Mohammad Imran

C++ String Functions

By – Mohammad Imran

What is string?

A string is referred to as an array of characters. In C++, a stream/sequence of characters is stored in a char array. C++ includes the **std::string** class that is used to represent strings. It is one of the most fundamental datatypes in C++ and it comes with a huge set of inbuilt functions.

The std::string is a class in C++ since C++98. This class is the standard representation for a text string. It includes some typical string operations like find, replace, concatenate, compare etc. It is present in **<string>** header file.

By – Mohammad Imran

What is string?

The syntax to create a string in C++ is quite simple. We use the ***string*** keyword to create a string in C++.

Syntax:

```
string str_name = "String";
```

Example:

```
string str = "This is string in C++";
```

Strings

Define in C++

Using the ***string*** keyword is the best and the most convenient way of defining a string. We should prefer using the *string* keyword because it is easy to write and understand.

```
string str = "hello";
or
string str("hello");
```

Strings Define in C++

We have also the C-style strings. Here are the different ways to create C-style strings:

```
char str_name[6] = {'h', 'e', 'l', 'l', 'o', '\0'};
char str_name[] = {'h', 'e', 'l', 'l', 'o', '\0'};
```

Alternate way we have

```
char str_name[] = "hello";
or
char str_name[6] = "hello";
```

Strings

Taking Input

To provide our program(s) input from the user, we generally use the **cin** keyword along with the extraction operator (**>>**). By default, the extraction operator considers white space (such as space, tab, or newline) as the terminating character. So, suppose the user enters "**New Delhi**" as the input. In that case, only "**New**" will be considered input, and "Delhi" will be discarded.

Let us take a simple example to understand this:

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

int main()
{
   char str[30];
   cout << "Enter city: ";
   cin >> str;
   cout << "You have entered: " << str;
   return 0;
}
```

OUTPUT

```
Enter city: New Delhi
You have entered: New
```

To counter this limitation of the extraction operator, we can specify the maximum number of characters to read from the user's input using the ***cin.get()*** function.

Syntax:

```
cin.get(variable_name, size)
```

Let us take a simple example to understand this:

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

int main()
{
   char str[30];
   cout << "Enter city: ";
   cin.get(str, 30);
   cout << "You have entered: " << str;
   return 0;
}
```

OUTPUT

```
Enter city: New Delhi
You have entered: New Delhi
```

getline() :

The **getline()** function is used to read a string entered by the user. The **getline()** function extracts characters from the input stream. It adds it to the string object until it reaches the delimiting character. The default delimiter is \n.

Syntax:

```
cin.getline(variable_name, size)
```

Let us take a simple example to understand this:

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

int main()
{
   char str[30];
   cout << "Enter city: ";
   cin.getline(str, 30);
   cout << "You have entered: " << str;
   return 0;
}
```

OUTPUT

```
Enter city: New Delhi
You have entered: New Delhi
```

String Function

length()

The most commonly used capacity functions are:

length()

As the name suggests, the **length()** function returns the length of a string.

String Function | length() | Example - 5

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


int main()
{
   string str = "Calculating length of the string";
   cout << "String length is: " << str.length() << endl;
   return 0;
}
```

OUTPUT

```
String length is: 32
```

By – Mohammad Imran

capacity()

The capacity() function returns the current size of the allocated space for the string. The size can be equal to or greater than the size of the string. We allocate more space than the string size to accommodate new characters in the string efficiently.

String Function

capacity() | **Example - 6**

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

int main()
{
   string str = "C++ Programming";
   cout << "String length : " << str.length() << endl;
   cout << "String capacity : " << str.capacity() << endl;
   return 0;
}
```

OUTPUT

```
String length: 15
String capacity: 15
```

By – Mohammad Imran

resize()

The resize() function is used to either increase or decrease the size of a string.

String Function

resize()

Example - 7

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

int main()
{
   string str = “C++ Programming";
   cout << "Original string : " << str << endl;
   str.resize(10);
   cout << "String after resize : " << str << endl;
   return 0;
}
```

OUTPUT

```
Original string : C++ Programming
String after resize : C++ Progra
```

By – Mohammad Imran

String Function

C-Style

The C-style character string originated within the C language and continues to be supported within C++. This string is actually a one-dimensional array of characters which is terminated by a **null** character **'\0'**. Thus a null-terminated string contains the characters that comprise the string followed by a **null**.

```
char greeting[6] = {'H', 'e', 'l', 'l', 'o', '\0'};
```

C++ supports a wide range of functions that manipulate null-terminated strings

By – Mohammad Imran

String Function

C-Style

C++ supports a wide range of functions that manipulate null-terminated strings. All functions are available in **<cstring>** header file. It is also known as string manipulating functions. Some popular string functions are:

`strlen()`	`strncmp()`
`strcpy()`	`strcat()`
`strncpy()`	`strchr()`
`strcmp()`	`strstr()`

String Function

Strings are an array of characters. We can perform several operation on string to manipulate the strings. Such as finding length of string, comparing two string, displaying formatted output etc. using string function.

All the string functions are available in <**cstring**> header file.

By – Mohammad Imran

String Function

strlen()

String function **strlen()** is used to find the length of given string. It does not count null (‘\0’) character. It returns number of character of the string.

Example: If you want to find the length of **“Computer”** it returns **8**.

By – Mohammad Imran

strlen()

Example - 8

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

int main()

{

   char str[25];

   cout << "Enter string : ";

   cin >> str;

   cout << "String length is : " << strlen(str);

   return 0;

}
```

OUTPUT

```
Enter string : Computer
String length is : 8
```

By – Mohammad Imran

String Function

strrev()

String function **strrev()** is used to reverse a string.

Example: If you want to reverse string it shows **“Hello”** as “**olleH”**.

strrev()

Example - 9

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

int main()

{

   char str[25];

   cout << "Enter string : ";

   cin.get(str, 25);

   cout << “Reverse string : << strrev(str);

   return 0;

}
```

OUTPUT

```
Enter string : CU
Reverse string : UC
```

By – Mohammad Imran

String Function

strcpy()

String function **strcpy()** is used to copy source value to destination variable.

Example: If you want to copy one string to another the syntax is.

Syntax:

```
strcpy (destination, source);
```

By – Mohammad Imran

strcpy()

Example - 10

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

int main()

{
   char str1[25], str2[25];
   cout << "Enter string : ";
   cin >> str1;
   strcpy( str2, str1);
   cout << "Str2 = " << str2;
   return 0;

}
```

OUTPUT

```
Enter string : Hello
Str2 = Hello
```

By – Mohammad Imran

String Function strncpy()

String function **strncpy()** is known as bounded string copy it is used to copy **n** (specified) number of character from source value to destination variable.

Example: If you want to copy one string to another with specified number of character the syntax is.

Syntax:

```
strncpy (destination, source, n);
```

By – Mohammad Imran

strncpy()

Example - 11

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

int main()

{
   char str1[25] = "C Program", str2[25];
   int n = 5;
   strncpy( str2, str1, n);

   cout << "Copied value = " << str2;

   return 0;

}
```

OUTPUT

```
Copied value = C Pro
```

By – Mohammad Imran

String Function strncat()

String function **strcat()** concatenates two strings and result is returned to source string.

Example: If you want to concatenate one string to another strcat is used.

Syntax:

```
strcat (source, new_string);
```

strncat()

Example - 12

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

int main()

{
   char str1[25];
   cout << "Enter string : ";
   gets(str1);
   strcat( str1, "ming");
   cout << "Str1 is : " << str1;
   return 0;

}
```

OUTPUT

```
Enter string : C Program
Str1 is : C Programming
```

By – Mohammad Imran

String Function strcmp()

String function **strcmp()** is used to compare two strings if strings are identical then it returns 0 otherwise it returns non zero values. It is case sensitive it means it checks small and capital letter too.

Example: "Hello" and hello is not same.

Syntax:

```
strcmp (String1, String2);
```

strcmp()

Example - 13

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

int main()
{
   char str1[25] = "Hello";
   char str2[25] = "hello";
   int n = strcmp(str1,str2);
   if(n == 0)
      cout << "Both strings are equal";
   else
      cout << "Strings are not equal";
   return 0;
}
```

OUTPUT

```
Strings are not equal
```

By – Mohammad Imran

String Function stricmp()

String function **stricmp()** function is used to compare two strings if strings are identical then it returns 0 otherwise it returns non zero values. If we use **stricmp** it means it ignores small and capital letter it will check spelling only.

Example: "Hello" and hello is the same.

Syntax: `stricmp (String1, String2);`

stricmp()

Example - 14

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

int main()
{
   char str1[25] = "Hello";
   char str2[25] = "hello";
   int n = stricmp(str1,str2);
   if(n == 0)
      cout << "Both strings are equal";
   else
      cout << "Strings are not equal";
   return 0;
}
```

OUTPUT

```
Both strings are equal
```

By – Mohammad Imran

String Function strncmp()

String function **strncmp()** is used to compare two strings at specified number of character with case sensitivity, if strings are identical then it returns 0 otherwise it returns non zero values.

Syntax:

```
strncmp (String1, String2, n);
```

By – Mohammad Imran

strncmp()

Example - 15

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

int main()
{
   char str1[25] = "Wonderful";
   char str2[25] = "Wonderer";
   int n = strncmp(str1,str2,7);
   if(n == 0)
      cout << "Both strings are equal";
   else
      cout << "Strings are not equal";
   return 0;
}
```

OUTPUT

```
Strings are not equal
```

By – Mohammad Imran

String Function

strlwr()

String function **strlwr()** returns given string in lowercase.

Syntax:

```
strlwr (String1);
```

strlwr()

Example - 16

```
#include <iostream>
#include <cstring>
using namespace std;
int main()
{
   char str1[25] = "Hello";
   cout << "Str1 = " << str1;
   cout << "Lowercase = " << strlwr(str1);
   return 0;
}
```

OUTPUT

```
Str1 = Hello
Lowercase = hello
```

By – Mohammad Imran

String Function

strupr()

String function **strupr()** returns given string in uppercase letter.

Syntax:

```
strupr (String1);
```

strupr()

Example - 17

```
#include <iostream>
#include <cstring>
using namespace std;
int main()
{
   char str1[25] = "Hello";
   cout << "Str1 = " << str1;
   cout << "Uppercase = " << strupr(str1);
   return 0;
}
```

OUTPUT

```
Str1 = Hello
Uppercase = HELLO
```

By – Mohammad Imran

Character Function

By – Mohammad Imran

Character Function

All data is entered into computers as characters, which includes letters, digits and various special symbols.

The character-handling library includes several functions that perform useful tests and manipulations of character data.

Character functions need **<cctype>** header file to be included in the program. Different character functions provided by C++ Language are:

Character Function

isalpha()

isalpha() function in C++ language checks whether given character is alphabet or not.

The **isalpha** function returns false, if it is not a alphabet otherwise it returns true.

Syntax:

```
bool isalpha(char c);
```

By – Mohammad Imran

isalpha()

Example - 18

```
#include <iostream>
#include <cctype>
using namespace std;
int main()
{
   char ch;
   cout << "Enter any character : ";
   cin >> ch;
   if(isalpha(ch))
      cout << "Entered character is alphabet";
   else
      cout << "Entered character is not alphabet";
   return 0;
}
```

OUTPUT

```
Enter any character : A
Enter character is alphabet
```

By – Mohammad Imran

Character Function

isalnum()

isalnum() function in C++ language checks whether given character is alphanumeric or not.

The **isalnum** function returns true if ***c*** is alphanumeric and returns false if *c* is not alphanumeric.

Syntax: **bool isalnum(char c);**

By – Mohammad Imran

isalnum()

Example - 19

```
#include <iostream>
#include <cctype>
using namespace std;
int main()
{
   char ch;
   cout << "Enter any character : ";
   cin >> ch;
   if(isalnum(ch))
      cout << "Character is alphanumeric";
   else
      cout << "Character is not alphanumeric";
   return 0;
}
```

OUTPUT

```
Enter any character : A or 4
Character is alphanumeric
```

By – Mohammad Imran

Character Function

isdigit()

isdigit() function in C++ language checks whether given character is digit or not.

Similar to the last one, it accepts a character and returns false, if that is not a digit otherwise, it returns true.

Syntax:

```
bool isdigit(char c);
```

By – Mohammad Imran

isdigit()

Example - 20

```
#include <iostream>
#include <cctype>
using namespace std;
int main()
{
   char ch;
   cout << "Enter any character : ";
   cin >> ch;
   if(isdigit(ch))
      cout << “Character is digit";
   else
      cout << “Character is not digit";
   return 0;
}
```

OUTPUT

```
Enter any character : 4
Enter character is digit
```

By – Mohammad Imran

Character Function

islower()

islower() function in C++ language checks whether given character is lowercase or not.

The **islower** function in C++ returns false, if the provided character is not in lowercase, otherwise it returns true.

Syntax:

```
bool islower(char c);
```

By – Mohammad Imran

islower()

Example - 21

```
#include <iostream>
#include <cctype>
using namespace std;
int main()
{
   char ch;
   cout << "Enter any character : ";
   cin >> ch;
   if(islower(ch))
      cout << "Lower Case";
   else
      cout << "Not a Lower Case";
   return 0;
}
```

OUTPUT

```
Enter any character : a
Lower case
```

By – Mohammad Imran

Character Function

isupper()

isupper() function in C++ language checks whether given character is uppercase or not.

The **isupper** library function in C++ returns false, if the provided character is not in uppercase, otherwise it returns true.

Syntax:

```
bool isupper(char c);
```

By – Mohammad Imran

isupper()

Example - 22

```
#include <iostream>
#include <cctype>
using namespace std;
int main()
{
   char ch;
   cout << "Enter any character : ";
   cin >> ch;
   if(isupper(ch))
      cout << "Upper Case";
   else
      cout << "Not a Upper Case";
   return 0;
}
```

OUTPUT

```
Enter any character : A
Upper case
```

By – Mohammad Imran

Character Function — tolower()

This is character conversion function

tolower() function in C++ language checks whether given character is alphabetic and convert it to lowercase.

The **tolower** function returns a value equivalent to lowercase of provided character, if no such character exists it returns the same character.

Syntax:

```
char tolower(char c);
```

By – Mohammad Imran

tolower()

Example – 23

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

int main()
{
      char ch;
      cout << "Enter any character : ";
      cin >> ch;
      char c = tolower(ch);
      cout << "Lower Case : " << c;
      return 0;
}
```

OUTPUT

```
Enter any character : A
Lower case : a
```

By – Mohammad Imran

Character Function — toupper()

toupper() function in C++ language checks whether given character is alphabetic and convert it to uppercase.

The **toupper** function is Similar to last one, it returns a value equivalent to uppercase of provided character, if no such character exists it returns the same character.

Syntax:

```
char toupper(char c);
```

toupper()

Example - 24

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

int main()
{
      char ch;
      cout << "Enter any character : ";
      cin >> ch;
      char c = toupper(ch);
      cout << "Upper Case : " << c;
      return 0;
}
```

OUTPUT

```
Enter any character : d
Upper case : D
```

By – Mohammad Imran

C++ User-Defined Function

By – Mohammad Imran

User Defined Function

The function which are created by the programmer is known as user defined function. These methods are modified according to the requirement.

By – Mohammad Imran

User Defined Function

Syntax

The syntax of the user-defined function is as follows,

- ✓ First, we write the return type.
- ✓ Then the function name followed by the argument wrapped inside the parenthesis.
- ✓ After all of this, there exist the definition of function inside the curly braces.

NOTE: Return_Type can be void or can be any one of the data types.

By – Mohammad Imran

User Defined Function

Types

The user-defined function can be classified into four categories according to their parameter and return values.

- ✓ Function with no Argument and no Return Value
- ✓ Function with Argument but no Return Value
- ✓ Function with Return Value but no Argument
- ✓ Function with Argument and Return Value

By – Mohammad Imran

Function with no argument and no return value

The function that neither accepts any argument nor provides a return value comes under this category.

User Defined Function

No Argument No Return Function

Now, as we know, the functions are used to process some input and subsequently provide output, So a question arises "Why are we using such a function which is neither receiving input nor returning some output"? The reason is apart from the parameters and return value, there could be two major scenarios to accept input and provide output.

Ask for the input directly from **stdin** or access the memory location associated with the variable outside the function scope.

Provide output directly to **stdout** or access any memory location associated with the variable outside the scope to store the output.

By – Mohammad Imran

User Defined Function

No Argument No Return Function

Syntax

```
Return_Type Function_Name( )
{
   Statements;
}
```

NOTE: Return_Type can be void or can be any one of the data types. Here in no return function return should be void.

By – Mohammad Imran

No Argument No Return Function

Example - 25

```
#include <iostream>
#include <cmath>
using namespace std;
void sum()
{
   double x, y;
   x = 10;
   y = 20;
   cout << "Sum = " << x+y;
}
```

```
int main ()
{
      sum();
      return 0;
}
```

OUTPUT

```
Sum = 30
```

By – Mohammad Imran

Function with Argument but no Return Value

This is also known as argument function used to take input at runtime and pass those values to the function.

User Defined Function

Argument but No Return Function

Syntax

```
Return_Type Function_Name(Parameter_List)
{
   Statements;
}
```

NOTE: Parameter list is known as argument.

Argument But No Return Function

Example - 26

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

void sum(double x, double y)
{
   cout << "Sum = " << x+y;
}

int main ()
{
    sum(100, 50);
    return 0;
}
```

OUTPUT

```
Sum = 150
```

By – Mohammad Imran

Function with No Argument but Return Value

As we know every function can return a value so we can define a function to return a value to the function. This type of function doesn't accept any argument but returns a value.

User Defined Function

No Argument but Return Function

Syntax

```
Data_Type Function_Name()
{
   Statements;
}
```

NOTE: Here in the return function return type will be any one of primitive data type.

By – Mohammad Imran

No Argument But Return Function

Example - 27

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

double sum()
{
   double x = 30, y = 50;

   return x+y;
}

int main ()
{
   cout << sum();

   return 0;
}
```

OUTPUT

```
Sum = 80
```

By – Mohammad Imran

Function with Argument and Return Value

The function which accepts argument as well as returns a value falls into this category. Basically this is the most used form of functions.

User Defined Function

Argument and Return Value

Syntax

```
Data_Type Function_Name(Parameter_List)
{
   Statements;
}
```

NOTE: Here in the return with argument function return type will be any one of primitive data type and should be argument in the paranthesis.

By – Mohammad Imran

Argument with Return Function

Example - 28

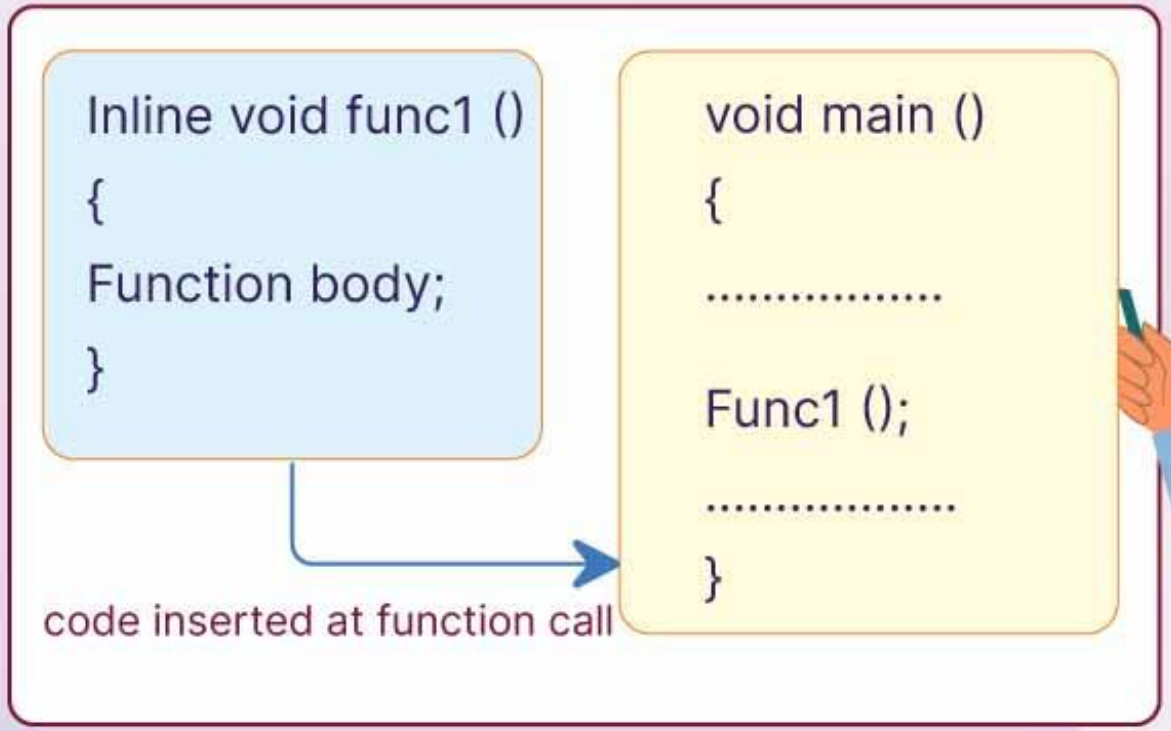
```
#include <iostream>
using namespace std;

double sum(double x, double y)
{
   return x+y;
}

int main ()
{
   double n = sum(100, 50);

   cout << "Sum = " << n;

   return 0;
}
```

OUTPUT

```
Sum = 150
```

By – Mohammad Imran

Understanding Inline Function in C++



By – Mohammad Imran

Inline Function

Introduction

C++ provides inline functions to reduce the function call overhead. An inline function is a function that is expanded in line when it is called. When the inline function is called whole code of the inline function gets inserted or substituted at the point of the inline function call. This substitution is performed by the C++ compiler at compile time. An inline function may increase efficiency if it is small.

By – Mohammad Imran

Inline Function

Introduction

Inline Function are those function whose definitions are small and be substituted at the place where its function call is happened. Function substitution is totally compiler choice.

It saves a huge amount of execution time as well as memory space. For defining an inline function we need to use the " inline " keyword. Every time you need a function in code all we have to do is define an inline function logically and then use it as many times you want in a code, as it will help in boosting the execution speed of the code.

By – Mohammad Imran

Inline Function

Syntax

```
inline return-type function-name(parameters)
{
    // function code
}
```

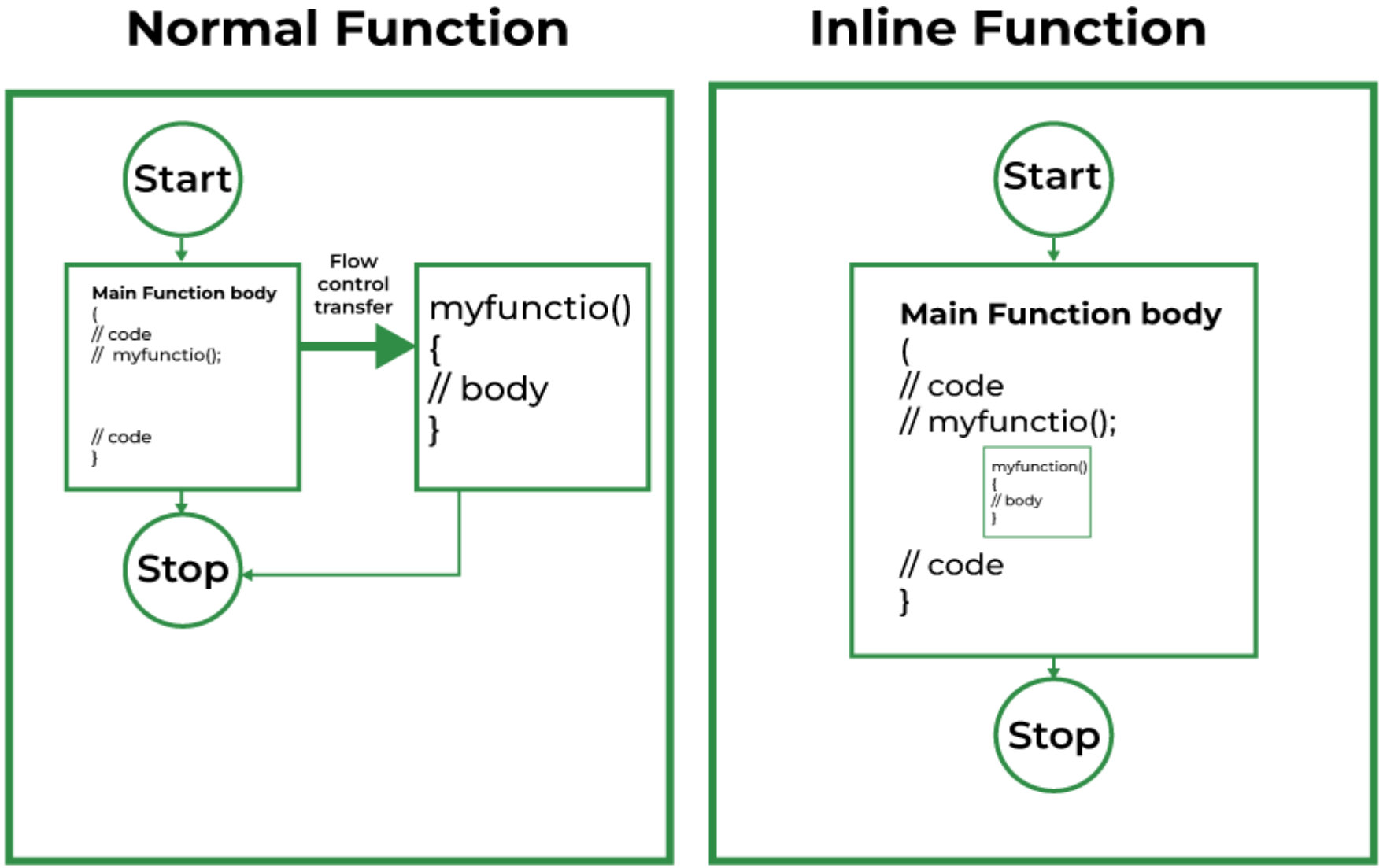
Inline Function

Example

```
inline void cube(int s)
{
    return s*s*s;
}
```

By – Mohammad Imran

Inline Function



By – Mohammad Imran

Inline Function

Remember

Remember, inlining is only a request to the compiler, not a command. The compiler can ignore the request for inlining.

The compiler may not perform inlining in such circumstances as:

- ✓ If a function contains a loop. (*for, while and do-while*)
- ✓ If a function contains static variables.
- ✓ If a function is recursive.
- ✓ If a function return type is other than void, and the return statement doesn’t exist in a function body.
- ✓ If a function contains a switch or goto statement.

Inline Function

Why Used

When the program executes the function call instruction the CPU stores the memory address of the instruction following the function call, copies the arguments of the function on the stack, and finally transfers control to the specified function. The CPU then executes the function code, stores the function return value in a predefined memory location/register, and returns control to the calling function. This can become overhead if the execution time of the function is less than the switching time from the caller function to called function (callee).

By – Mohammad Imran

Inline Function

Why Used

For functions that are large and/or perform complex tasks, the overhead of the function call is usually insignificant compared to the amount of time the function takes to run. However, for small, commonly-used functions, the time needed to make the function call is often a lot more than the time needed to actually execute the function's code. This overhead occurs for small functions because the execution time of a small function is less than the switching time.

By – Mohammad Imran

Inline Function

Advantages

- ✓ Function call overhead doesn’t occur.
- ✓ It also saves the overhead of push/pop variables on the stack when a function is called.
- ✓ It also saves the overhead of a return call from a function.
- ✓ When you inline a function, you may enable the compiler to perform context-specific optimization on the body of the function. Such optimizations are not possible for normal function calls.
- ✓ An inline function may be useful (if it is small) for embedded systems because inline can yield less code than the function called preamble and return.

By – Mohammad Imran

Inline Function

Disadvantages

- ✓ The added variables from the inlined function consume additional registers, After the in-lining function if the variable number which is going to use the register increases then they may create overhead on register variable resource utilization. This means that when the inline function body is substituted at the point of the function call, the total number of variables used by the function also gets inserted. So the number of registers going to be used for the variables will also get increased. So if after function inlining variable numbers increase drastically then it would surely cause overhead on register utilization.

Inline Function

Disadvantages

- ✓ If you use too many inline functions then the size of the binary executable file will be large, because of the duplication of the same code.
- ✓ Too much inlining can also reduce your instruction cache hit rate, thus reducing the speed of instruction fetch from that of cache memory to that of primary memory.
- ✓ Inline functions may not be useful for many embedded systems. Because in embedded systems code size is more important than speed.

Inline Function

Disadvantages

- ✓ The inline function may increase compile time overhead if someone changes the code inside the inline function then all the calling location has to be recompiled because the compiler would be required to replace all the code once again to reflect the changes, otherwise it will continue with old functionality.
- ✓ Inline functions might cause thrashing because inlining might increase the size of the binary executable file. Thrashing in memory causes the performance of the computer to degrade. The following program demonstrates the use of the inline function.

By – Mohammad Imran

Inline Function

Example - 29

```
#include <iostream>
using namespace std;
inline int foo()
{
   return 2;
}

int main()
{
   int ret;
   ret = foo();
   cout << "Output is : " << ret;
   return 0;
}
```

OUTPUT

```
Output is : 2
```

NOTE: Compile time error When use GCC compiler

By – Mohammad Imran

Inline Function

Example - 30

```
#include <iostream>
using namespace std;
class FunInline
{
   public:
   inline void display()
   {
      cout << "Hello";
   }
};
int main()
{
   FunInline fn;
   fn.display();
   return 0;
}
```

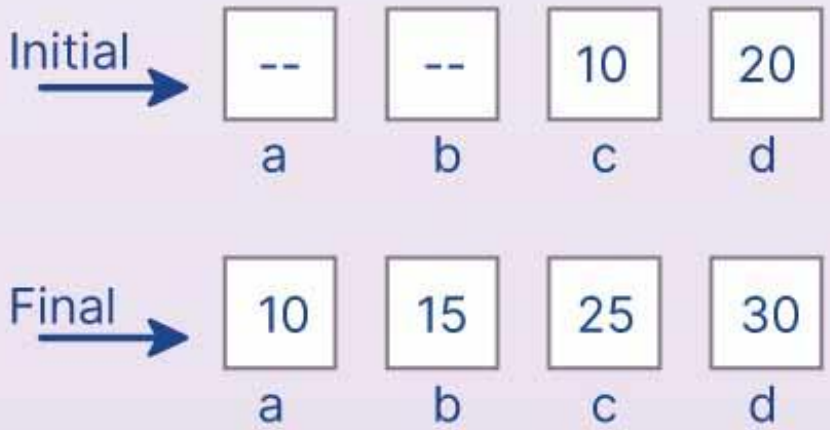
OUTPUT

```
Hello
```

By – Mohammad Imran

Default arguments

In C++



By – Mohammad Imran

Default Argument

Introduction

Arguments in C++ are the values of parameters passed to a function while calling the function. While writing functions in C++ programming that take arguments as function parameters, we can assign default values to those arguments during the function declaration. These default values assigned to the function parameters are known as default arguments in C++. If the function requiring parameters gets called without any arguments, it uses the default arguments, which are assigned automatically by the compiler at the time of compilation. However, the default arguments get ignored when we call the function by passing the required arguments.

By – Mohammad Imran

Default Argument

Definition

A default argument is a value provided in a function declaration that is automatically assigned by the compiler if the calling function doesn’t provide a value for the argument. In case any value is passed, the default value is overridden.

By – Mohammad Imran

Default Value Parameter in Function

When we declare a function in C++, we can specify a default value for parameters. The default value can be assigned using the assignment operator (=).

If a value is not passed during the function call then the default value is used for the corresponding parameter. If the value is specified during the function call then the default value is ignored and the passed value is used.

By – Mohammad Imran

Default Argument

Characteristics

The characteristics of Default Argument in C++ are as follows.

- ✓ The values associated with the default arguments in C++ are not constant. We can use these values only when the function gets called without the appropriate arguments. Otherwise, the declared values get overwritten by the passed values.
- ✓ All the arguments with default values get declared to the right in the list.
- ✓ The default arguments get copied from left to right during a function call.

By – Mohammad Imran

Default Argument

Characteristics

Example

This example shows us the correct way of writing a default argument in the parameters list.

```
void function(int x, int y, int z = 0)
```

Explanation: In the above function, we have three parameters x, y, and z. z has a predefined value of 0, which is the default argument. Here, we have declared z to the right in the list of parameters. Thus, the above function is valid.\

Default Argument

Characteristics

Example

```
void function(int x, int z = 0, int y)
```

Explanation: In the above function, we have three parameters x, y, and z. z has a predefined value of 0, which is the default argument. Here, we have declared z between the parameters x and y, which is not accepted. Thus, the above function is invalid.

Default Value Parameter in Function

Advantages

- ✓ Default arguments are useful when we want to increase the capabilities of an existing function as we can do it just by adding another default argument to the function.
- ✓ It helps in reducing the size of a program.
- ✓ It provides a simple and effective programming approach.
- ✓ Default arguments improve the consistency of a program.

By – Mohammad Imran

Default Value Parameter in Function

Disadvantages

- ✓ It increases the execution time as the compiler needs to replace the omitted arguments by their default values in the function call.

Default Argument

Example - 31

```
#include <iostream>
using namespace std;
int sum(int a, int b=10, int c=20)
{
   int z;
   z = a + b + c;
   return z;
}
int main()
{
   cout << sum(1) << endl;
   cout << sum(1, 2) << endl;
   cout << sum(1, 2, 3) << endl;
   return 0;
}
```

OUTPUT

```
31
23
6
```

By – Mohammad Imran

Default Argument

Example - 32

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

class Calculate

{

   public:

   int sum(int x = 2, int y = 0)

   {

      return x + y;

   }

};
```

```
int main()
{
   Calculate c;
   cout << c.sum() << endl;
   cout << c.sum(1) << endl;
   cout << c.sum(1,2) << endl;

   return 0;
}
```

OUTPUT

```
2
1
3
```

By – Mohammad Imran

Default Argument

Ambiguous | **Example - 33**

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

int sum(int x, int y, int z = 0, int w = 0)

{

    return (x + y + z + w);

}

int sum(int x, int y, float z = 0, float w = 0)

{

    return (x + y + z + w);

}
```

By – Mohammad Imran

Default Argument

Ambiguous | **Example - 33**

```
int main()
{
   cout << sum(10, 15) << endl;
   cout << sum(10, 15, 25) << endl;
   cout << sum(10, 15, 20, 25) << endl;
   return 0;
}
```

OUTPUT

```
sum(10, 15) // Error
sum(10, 15, 20, 25)

Both function are ambiguous
```

By – Mohammad Imran

Coding Question

By – Mohammad Imran

Question - 1

Write a C++ program to find the length of the input string without using pre defined string function.

Click here to see code

By – Mohammad Imran

Question - 2

Write a C++ program to find the square root of given value in long double as given below.

```
long double var1 = 10000000000000000;
long double var2 = 99999999999999999;
```

```
Output:
10000000000.000000000000
999999999.999999999476
```

Required Output

```
10000000000.000000000000
10000000000.000000000000
```

Click here to see code

By – Mohammad Imran

Question - 3

Write a C++ program to take two input first name and last name and display as full name.

Input

```
fname = Peter
Lname = Smith
```

Output

```
Full Name : Peter Smith
```

Click here to see code

By – Mohammad Imran

Question - 4

Write a C++ program to demonstrate static inline function.

Click here to see code

By – Mohammad Imran

Question - 5

Q1. Write a program to generate table from 1 to n numbers.

Q2. Write a program to print all uppercase letter using loop.

Q3. C program to read age of 15 person and count total Baby age, School age and Adult age

a) Baby age – 0 to 5

b) School age – 6 to 17

c) Adult age – 18 and over

Click here to see code

By – Mohammad Imran

QUIZ

By – Mohammad Imran

Quiz - 1

```
#include <iostream>
using namespace std;
int main()
{
   for(int i = 0; 1; i++)
   {
      cout << i;
   }
   return 0;
}
```

OUTPUT

Infinite Loop

By – Mohammad Imran

Quiz - 2

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

int main()
{
   int i=0;
   int max = 5;
   for (; i < max; i++)
   {
      cout << i << endl;
   }
   return 0;
}
```

OUTPUT

```
0
1
2
3
4
```

By – Mohammad Imran

Quiz - 3

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

int main( )

{

   int x = 10, y = 3, z;

   for(z = 0; z < x; )

      z = z++ +y;

   cout << z ;

   return 0;

}
```

OUTPUT

12

By – Mohammad Imran

Quiz - 4

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

int main( )

{

   do

   {

     cout << "Know Program";

   }while();

   return 0;

}
```

OUTPUT

Compile time error

By – Mohammad Imran

Quiz - 5

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

int main( )

{

   do

      cout << “In while loop” << endl;

   while (0);

   cout << "After loop” << endl;

   return 0;

}
```

OUTPUT

```
In while loop
After loop
```

By – Mohammad Imran

Quiz - 6

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

int main( )
{
   int a = 5;
   if (a == 6);
      a = 0;
   if (a == 5)
      a++;
   else
      a += 2;
   cout << a;
   return 0;
}
```

OUTPUT

2

By – Mohammad Imran

Quiz - 7

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

int main( )
{
   int i = 1, j = 1;

   for(--i && j++ ; i < 10; i += 2)

   {

      cout << "loop ";

   }

   return 0;
}
```

OUTPUT

```
Loop loop loop
loop loop
```

By – Mohammad Imran