Computer Programming

Contents

LL 02 = Learning Level 02 – Comprehension, LL 04 = Learning Level 04 – Analysis

Contents

- Passing Arguments to a Function By Reference *(LL 04)*
- Program Examples *(LL 04)*

LL 02 = Learning Level 02 – Comprehension, LL 04 = Learning Level 04 – Analysis

Function

- Most computer programs that solve real-world problems are much larger than the programs we have done in the class.
- Experience has shown that the best way to develop and maintain a large program is to construct it from small, simple pieces, or components.
- This technique is called **divide and conquer**.
- A larger program is created by developing smaller components individually and them assembling them together as whole.

Function

- A function groups a number of program statements into a unit and gives it a name.
- A function is a group of statements that together perform a task.
- Every C++ program has at least one function, which is **main( )**.

Types of Functions

- There are two types of functions:

Pre-Defined Functions

- Also called as **built-in** functions.
- The functions which are already built inside the C++ standard library, are called as pre-defined functions.
- You do not need to create them just call them whenever you want to use them.

- Fe w of the examples of pre-defined functions are:

Pre-Defined Functions

Function	Header File	Purpose	Parameter(s) Type	Result
abs(x)	<cmath>	Returns the absolute value of its argument: abs(-7) = 7	int (double)	int (double)
ceil(x)	<cmath>	Returns the smallest whole number that is not less than x: ceil(56.34) = 57.0	double	double
cos(x)	<cmath>	Returns the cosine of angle: x: cos(0.0) = 1.0	double (radians)	double
exp(x)	<cmath>	Returns e^x, where e = 2.718: exp(1.0) = 2.71828	double	double
fabs(x)	<cmath>	Returns the absolute value of its argument: fabs(-5.67) = 5.67	double	double

Pre-Defined Functions

Function	Header File	Purpose	Parameter(s) Type	Result
`floor(x)`	`<cmath>`	Returns the largest whole number that is not greater than `x`: `floor(45.67) = 45.00`	`double`	`double`
`islower(x)`	`<cctype>`	Returns `true` if `x` is a lowercase letter; otherwise, it returns `false`; `islower('h')` is `true`	`int`	`int`
`isupper(x)`	`<cctype>`	Returns `true` if `x` is a uppercase letter; otherwise, it returns `false`; `isupper('K')` is `true`	`int`	`int`
`pow(x, y)`	`<cmath>`	Returns x^y; if `x` is negative, `y` must be a whole number: `pow(0.16, 0.5) = 0.4`	`double`	`double`
`sqrt(x)`	`<cmath>`	Returns the nonnegative square root of `x`; `x` must be nonnegative: `sqrt(4.0) = 2.0`	`double`	`double`
`tolower(x)`	`<cctype>`	Returns the lowercase value of `x` if `x` is uppercase; otherwise, it returns `x`	`int`	`int`
`toupper(x)`	`<cctype>`	Returns the uppercase value of `x` if `x` is lowercase; otherwise, it returns `x`	`int`	`int`

User-Defined Functions

- Also called as **Programmer-Defined** functions.
- The functions which are not the part of C++ standard library.
- The programmer defines the functionality of the functions by themselves.
- Because C++ does not provide every function that you will ever need, you must learn to write your own functions.
- Fo r example there is no any function in C++ that finds the maximum number out of an array or swap two variables with each other.
- Fo r these tasks you need to create your own (User-Defined) functions.

User-Defined Functions

- A function is like a box (with group of statements inside) which receives some of the inputs, apply processing on them and gives you the output.
- A function can have zero or more inputs.
- A function can have zero or one output.
- The group of statements will define what operation, the function will perform on the inputs.

User-Defined Functions

- User-defined functions in C++ are classified into two categories:

1. Value-Returning Functions

Functions that have a return type. These functions return a value of a specific data type using the return statement.

2. Void Functions

Functions that do not have a return type. These functions do not use a return statement to return a value.

User-Defined Functions

- In first case the function takes some inputs, performs calculations and returns one value as a result. Lets say you want to create a function that receives the radius of the circle and returns you the area of the circle, in this case the function returns one value i.e. the area, hence it is a value returning function.
- In second case the function just does its job but do not return a value. Say, we need to create a function that receives two strings and displays both the strings after concatenating. In this case the function does not perform any calculation and will not return any value.

Creating User-Defined Functions in C++

- In order to create a user-defined function in C++ you need to provide:

- **Function Declaration**
- **Function Definition**
- **Function Calling**

Creating User-Defined Functions in C++

- Lets create two functions:

Function 1:

Create a function that receives the radius of the circle and returns you the area of the circle.

Functions 2:

Create a function that receives two strings and displays both the strings after concatenating.

Function Declaration

- The function declaration just tells the compiler how the function looks like. It includes the **function name**, the **parameter list** and the **return type**.

- The function declaration just tells the compiler that, we are going to create one of the function in this program and it looks like this.

- Function declaration is also called as **Function Prototype**.

Function Declaration

Function-1 Declaration:

Function Name

Parameter List

Return Type

```
float area( float ) ;
```

Function Declaration

Function-2 Declaration:

Function Name

Parameter List

Return Type

```
void display( string , string ) ;
```

Function Definition

- The function declaration provides the details of the functions. Here we writ all the statements that make up a function.
- Function definition tells the compiler, what the function will do.
- It includes, the **function name**, **return type**, **parameter list** and the **body of the function**.

Function Definition

Function-1 Definition:

Parameter List

Function Name

Return Type

```
float area( float radius )
{
    float const PI = 3.14;
    float a = PI * radius * radius;

    return a;
}
```

Function Body

Function Definition

Function-2 Definition:

Function Name

Parameter List

Return Type

Function Body

```
void display( string firstname, string lastname )
{
    string str = firstname + " " + lastname;
    cout<<str;
}
```

Function Calling

- Once the function is created, it can be used inside the program by calling it.
- It includes, the **function name** and **argument list**.
- Arguments are different then parameters.
- Parameters are the variables that we use while function definition.
- Arguments are the values/variables that we use while calling the function.

Function Calling

Function-1 Calling:

Argument List

Function Name

```
area(5.6) ;
```

Function Calling

Function-2 Calling:

Argument List

Function Name

```
display( "Ali" , "Asghar");
```

Program Examples

Functions in C++

Program Example 01

Problem Statement:

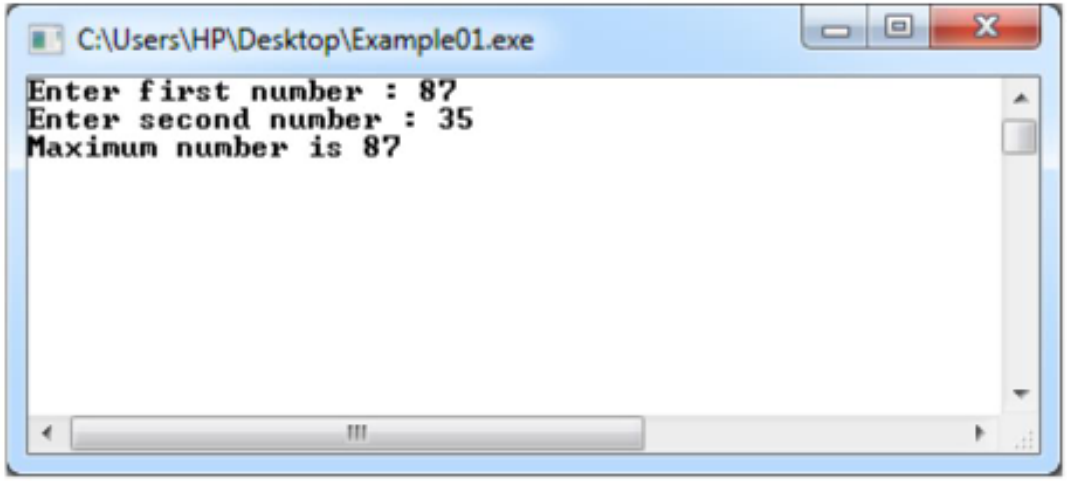
Create a function that receives two integer numbers and returns the maximum number out of them.

Program Example 01

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

using namespace std;

int max(int, int);

int main()
{
    int n1, n2;
    cout<<"Enter first number : ";
    cin>>n1;
    cout<<"Enter second number : ";
    cin>>n2;

    cout<<"Maximum number is "<<max(n1,n2);

    getch();
    return 0;
}
```

```
int max(int num1, int num2)
{
    if(num1 >= num2)
        return num1;
    else
        return num2;
}
```

Program Example 01



Program Example 02

Problem Statement:

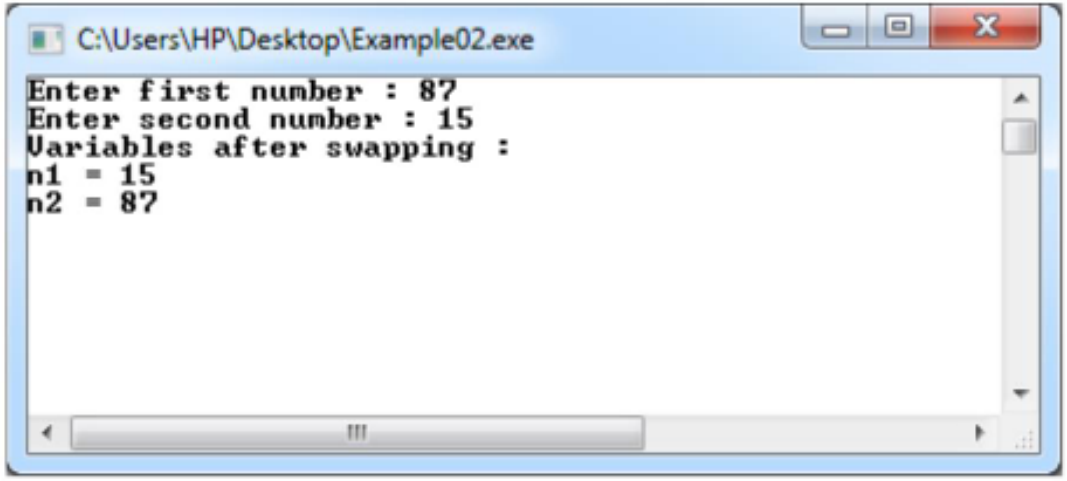
Create a function that receives two integer numbers and swaps them.

Program Example 02

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

using namespace std;

void swap(int&, int&);

int main()
{
    int n1, n2;
    cout<<"Enter first number : ";
    cin>>n1;
    cout<<"Enter second number : ";
    cin>>n2;

    swap(n1,n2);

    cout<<"Variables after swapping : "<<endl;
    cout<<"n1 = "<<n1<<endl<<"n2 = "<<n2;

    getch();
    return 0;
}
```

```
void swap(int& num1, int& num2)
{
    int temp = num1;
    num1 = num2;
    num2 = temp;
}
```

Program Example 02



Program Example 03

Problem Statement:

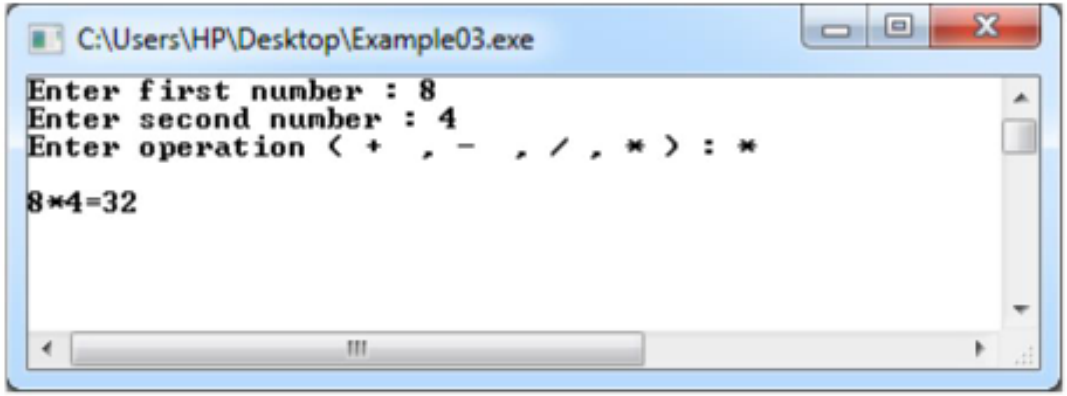
Create a function that receives two floating point number and an operator (+ , -, / , *). The function returns the result of the operation.

Program Example 03

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

using namespace std;

float calculate(float, float, char);

int main()
{
    int n1, n2;
    char op;

    cout<<"Enter first number : ";
    cin>>n1;
    cout<<"Enter second number : ";
    cin>>n2;
    cout<<"Enter operation ( +  , -  , / , * ) : ";
    op = getche();

    cout<<endl<<endl<<n1<<op<<n2<<"="<<calculate(n1,n2,op);

    getch();
    return 0;
}
```

```
float calculate(float num1, float num2, char op)
{
    float ans;

    switch(op)
    {
        case '+': ans = num1 + num2; break;
        case '-': ans = num1 - num2; break;
        case '*': ans = num1 * num2; break;
        case '/': ans = num1 / num2; break;
    }

    return ans;
}
```

Program Example 03



Program Example 05

Problem Statement:

Create two functions, first receives integer array and returns maximum item. The second receives integer array and returns minimum item.

Program Example 05

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

using namespace std;

int max(int[]);
int min(int[]);

const int SIZE = 10;

int main()
{
    int array1[] = {8,4,7,5,6,9,7,1,5,2};
    int array2[] = {-85,-71,25,0,57,15,47,85,-90};

    cout<<max(array1)<<endl;
    cout<<min(array1)<<endl;
    cout<<max(array2)<<endl;
    cout<<min(array2)<<endl;

    getch();
    return 0;
}
```

```
int max(int arr[])
{
    int maxItem = arr[0];

    for(int i=1; i<SIZE; i++)
    {
        if(arr[i] > maxItem)
            maxItem = arr[i];
    }

    return maxItem;
}

int min(int arr[])
{
    int minItem = arr[0];

    for(int i=1; i<SIZE; i++)
    {
        if(arr[i] < minItem)
            minItem = arr[i];
    }

    return minItem;
}
```

Program Example 05