Week 10

**1. Fill in the blanks**

a. Function overloading enables C++ to have various functions with the same name that operate on different types or numbers of arguments.

b. The scope resolution operator (::) enables access to a global variable with the same name as a variable in a particular function.

c. A generic function enables a single function to be defined to perform the same task on data of many different types.

**2. Explain what the purpose of default argument is.**

Default Argument allows a function to be called without providing one or more trailing arguments. It is indicated by using the following syntax for a parameter in the parameter-list of a function declaration.

Example

**Code:**

#include<iostream.h>

#include<conio.h>

int Add(int x, int y=20, int z=30)

{

return x + y + z;

}

void main()

{

int rs;

rs = Add(5);

cout<<"\n\tThe sum is : "<<rs;

rs = Add(4,8);

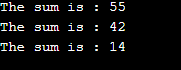
cout<<"\n\tThe sum is : "<<rs;

rs = Add(7,3,4);

cout<<"\n\tThe sum is : "<<rs;

}

**Output:**



**3. Write a program that uses a function template called max to determine the larger of two arguments.**

**Write another function template called min to determine the smaller of two arguments. Test the program using integer, character, and floating-point number arguments.**

**Code:**

#include <iostream>

using namespace std;

// definition of function template minimum; finds the smaller value

template < class T >

T minimum( T value1, T value2 )

{

if ( value1 < value2 )

return value1;

else

return value2;

}

int main()

{

// demonstrate minimum with int values

int int1 = 0; // first int value

int int2 = 0; // second int value

cout << "Input two integer values: ";

cin >> int1 >> int2;

cout << "The smaller integer value is: " << minimum( int1, int2 );

char char1 = ' '; // first char value

char char2 = ' '; // second char value

cout << "\n\nInput two characters: ";

cin >> char1 >> char2;

cout << "The smaller character value is: " << minimum( char1, char2 );

double double1 = 0; // first double value

double double2 = 0; // second double value

cout << "\n\nInput two double values: ";

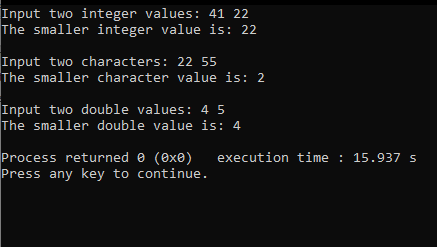
cin >> double1 >> double2;

cout << "The smaller double value is: " << minimum( double1, double2 )

<< endl;

}

**Output:**



**4. Write two overloaded functions named countNegative. The first function receives one integer array and size of the array as parameters and returns the number of negative numbers in the array. The second function receives floating point array and size of the array as parameters and returns the number of negative numbers in the array. Then write a main function to test these functions. The program should be able to check and report the cases when there are no negative numbers in the array. All printing happens in the main function.**

**Code:**

#include <iostream>

using namespace std;

//function to count the negative values in the int array

int countNegative (int arr[], int size)

{

int count=0;

for(int i=0;i<size;i++)

{

if(arr[i]<0)

count++;

}

return count;

}

//overloaded function to count the negative values in the float array

int countNegative (float arr[], int size)

{

int count=0;

for(int i=0;i<size;i++)

{

if(arr[i]<0)

count++;

}

return count;

}

int main()

{

int arr1[]={2,3,-3,-2,8,9,-34};

float arr2[]={23.5,-8.4,21.2,90.4,-23.4,-1,-22};

//value returned by the first function

int neg\_values=countNegative(arr1,7);

//value returnred by the deconds function

int neg\_value=countNegative(arr2,7);

//when there are no neg\_values in the int array

if(neg\_values==0)

cout<<"\nThere are no negative values in the array ";

else

cout<<"\n"<<neg\_values<<" elements in the int array are negative";

//when there are no neg\_values in the float array

if(neg\_value==0)

cout<<"\nThere are no negative values in the array ";

else

cout<<"\n"<<neg\_value<<" elements in the float array are negative";

return 0;

}

**Output:**

