Qn1

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

Qn2

Purpose of default argument:

Support to invoke a function repeatedly with the same argument value for a parameter.

In such cases, the programmer can specify a default argument for those parameters, i.e., a default value to be passed to that parameter.

Qn3.cpp

#include <iostream>

using namespace std;

#include "comparison.h" // include definition of function template maximum

int main()

{

int int1, int2;

char char1, char2;

float float1, float2;

cout << "Input two integer values (with space): ";

cin >> int1 >> int2;

// display the result

cout << "The larger integer value is: "<< maximum( int1, int2)<< endl;

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

cout << "Input two character values (with space): ";

cin >> char1 >> char2;

// display the result

cout << "The larger character value is: "<< maximum( char1, char2)<< endl;

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

cout << "Input two floating-point values (with space): ";

cin >> float1 >> float2;

// display the result

cout << "The larger floating-point value is: "<< maximum( float1, float2)<< endl;

cout<<"The smaller floating-point value is: "<< minimum( float1, float2)<< endl;

}

comparison.h

// or template< typename T

template < class T >

T maximum( T value1, T value2)

{

T maximumValue = value1; // assume value1 is maximum

// determine whether value2 is greater than maximumValue

if ( value2 > maximumValue )

maximumValue = value2;

return maximumValue;

} // end function template maximum

template < class T >

T minimum( T value1, T value2)

{

T minimumValue = value1; // assume value1 is minimum

// determine whether value2 is less than minimumValue

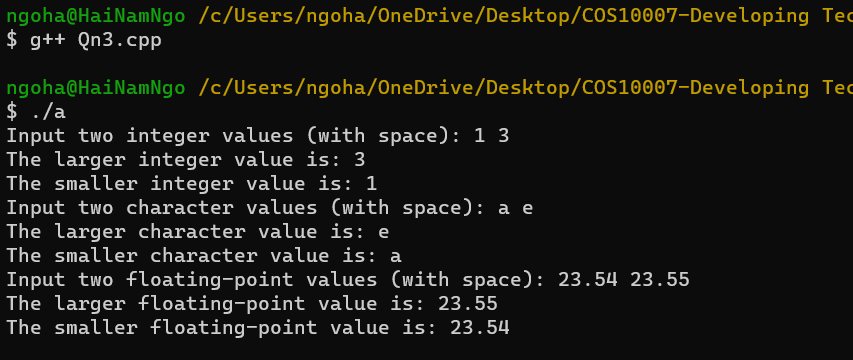
if ( value2 < minimumValue )

minimumValue = value2;

return minimumValue;

}

OUTPUT FOR Qn3.cpp



Qn4.cpp

#include <iostream>

using namespace std;

int countNegative(int array[], int arraySize)

{

int count = 0;

cout << "Insert the elements for the array: ";

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

{

cin >> array[i];

if (array[i] < 0)

{

count += 1;

}

}

cout << "Number of negative elements is: ";

return count;

}

int countNegative(float array2[], int arraySize)

{

int count = 0;

cout << "Insert the elements for the array: ";

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

{

cin >> array2[i];

if (array2[i] < 0)

{

count += 1;

}

}

cout << "Number of negative elements is: ";

return count;

}

int main()

{

int SIZE;

cout << "Enter the number of elements in the array: ";

cin >> SIZE;

int array[SIZE];

float array2[SIZE];

cout << countNegative(array, SIZE) << endl;

cout << countNegative(array2, SIZE) << endl;

return 0;

}

OUTPUT FOR Qn4.cpp

