//COS10007 – Developing Technical Software//  
//StudentID – 103234103 | Son Nguyen//  
//Lab 10 – 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 (;;) (symbol) 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  
  
2/ Explain the purpose of default argument is  
Default arguments is used in case user have not entered any argument. In the case of Lab 9 Question 4 where we attempt to clarify a number. We can set the argument as a specific number but if we didn’t set for another number – it would print out this default argument unless specified.  
  
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.  
  
#include <iostream>

template <typename T>

T max(T a, T b) {

return (a > b) ? a : b;

}

template <typename T>

T min(T a, T b) {

return (a < b) ? a : b;

}

int main() {

int a = 10 , b = 6 ;

std::cout << "max(" << a << ", " << b << ") = " << max(a, b) << std::endl;

std::cout << "min(" << a << ", " << b << ") = " << min(a, b) << std::endl;

char c1 = 'a', c2 = 'b';

std::cout << "max(" << c1 << ", " << c2 << ") = " << max(c1, c2) << std::endl;

std::cout << "min(" << c1 << ", " << c2 << ") = " << min(c1, c2) << std::endl;

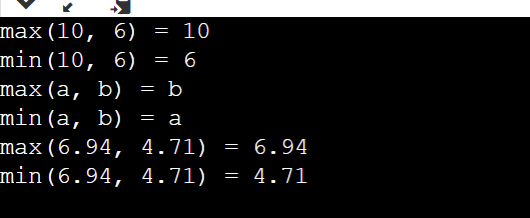
float f1 = 6.94, f2 = 4.71;

std::cout << "max(" << f1 << ", " << f2 << ") = " << max(f1, f2) << std::endl;

std::cout << "min(" << f1 << ", " << f2 << ") = " << min(f1, f2) << std::endl;

return 0;

}

Output Code   
  
  
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  
  
#include <iostream>

// Overload 1: countNegative for integer arrays

int countNegative(int arr[], int size) {

int count = 0;

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

if (arr[i] < 0) { // If the current element is negative, increment the count

count++;

}

}

return count;

}

// Overload 2: countNegative for floating point arrays

int countNegative(float arr[], int size) {

int count = 0;

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

if (arr[i] < 0) { // If the current element is negative, increment the count

count++;

}

}

return count;

}

int main() {

// Test the countNegative function with integer array

int arr1[] = {1, -2, 3, -4, 5};

int size1 = sizeof(arr1) / sizeof(arr1[0]);

int negCount1 = countNegative(arr1, size1);

if (negCount1 > 0) { // If there are negative numbers, print how many there are

std::cout << "There are " << negCount1 << " negative numbers in the integer array." << std::endl;

} else { // Otherwise, print that there are none

std::cout << "There are no negative numbers in the integer array." << std::endl;

}

// Test the countNegative function with floating point array

float arr2[] = {1.5, -2.3, 3.7, 4.2, -5.1};

int size2 = sizeof(arr2) / sizeof(arr2[0]);

int negCount2 = countNegative(arr2, size2);

if (negCount2 > 0) { // If there are negative numbers, print how many there are

std::cout << "There are " << negCount2 << " negative numbers in the floating point array." << std::endl;

} else { // Otherwise, print that there are none

std::cout << "There are no negative numbers in the floating point array." << std::endl;

}

// Test the countNegative function with floating point array that has no negative numbers

float arr3[] = {1.5, 2.3, 3.7, 4.2, 5.1};

int size3 = sizeof(arr3) / sizeof(arr3[0]);

int negCount3 = countNegative(arr3, size3);

if (negCount3 > 0) { // If there are negative numbers, print how many there are

std::cout << "There are " << negCount3 << " negative numbers in the floating point array." << std::endl;

} else { // Otherwise, print that there are none

std::cout << "There are no negative numbers in the floating point array." << std::endl;

}

return 0;

}

Output   
