**University of Management and Technology, Lahore Campus**

**Lab- 15 Manual**

Lab Instructor: Riaz Ahmad

Department of Computer Science

Email: [riazahmad@umt.edu.pk](mailto:riazahmad@umt.edu.pk)

**Strings****, Function Overloading and Templates**

* 1. **Objective:**

Learn how to declare, initialize and use one-dimensional character arrays (string).

Learn What is Function Overloading.

Learn What are Templates and how they work.

* 1. **Scope:**

The student should know the following:

* + 1. What is string?
    2. How to Read and Print Strings.
    3. Some important built-in string functions. How to use them?
    4. What is Function Overloading.
    5. What are Templates and how they work.
  1. **Useful Concepts:**

What is String?

A string is any sequence of characters *enclosed* in double quotes. There is no separate data type for strings as integer, float or double. The string data type can be considered as a char array. The difference is that a character variable can hold only *one character* but a string can have *more than one character* in a character array. For example, a string called name with 9 characters can be declared as:

char name [9] = “I like C” ;

Check for the double quotes, the char array size 9. Here the name can be considered as one string or as an array of characters. That is if you refer name it is “I like C” and if you refer name[0] it is ‘I’, name[1] is ‘ ‘ (a *blank*), name[2] is ‘l’, name[3] is ‘i’, and so on. The last character name[8] is ‘\0’ which indicates a NULL character which is not displayed but is stored as last character of the string to indicate its end.

**Note**: Strings are stored as arrays of characters and have a special ‘\0’ termination character called NULL appended (attached) to them to signify the end of the string.

Note that if the number of characters including the ‘\0’ is more than the size of the array the results will be unpredictable. However, if the size of the array is more than the number of characters the extra spaces after the last ‘\0’ character are kept blank and not referred because the string ends at ‘\0’. So, always make sure that the size of the array is sufficient for the string. For example, the above declaration would be wrong if we write

char name [8] = “I like C” ;

The size can be ignored also. In that case the size is considered as that of the number of characters specified in the declaration.

char name [] = “I like C” ;

**String Input/Output:**

The easiest way to input strings is by using the C++ library function getline(). The getline()function reads a string of characters entered at the keyboard until you strike the enter key (carriage return). The carriage return does not become part of the string; instead a null terminator ‘\0’ is placed at the end.

For example, in the following program fragment

char str[80] ; cin.getline(str,80);

and if the user enters

Muslim Abbas

and presses the enter key the string variable str has the characters “Muslim Abbas” with ‘\0’

appended at the end but is not displayed.

To display the string, we can use

cout<<str;

**Built-in String Functions:**

Large collections of string processing functions are provided in C++ through cstring file. So, include the cstring file in the programs to use these functions. To use the string functions make sure that the size of the array is sufficient so that the strings are terminated with the ‘\0’ character or the functions will not work properly.

**strcat ( string1, string2 );**

The strcat function concatenates or joins the string1 and string2. A copy of string2 is put at the end of the string1. Make sure that string1 size is long enough to hold the resulting string (string1 + string2).

**Example:** char string1[81] = “abc”, string2 [] = “def” ; strcat ( string1, string2) ;

cout<<string1; // outputs “abcdef” which is stored in string1

**strcpy (string1, string2 );**

The **strcpy** function copies **string2** into **string1**. Again make sure that **string1** size is long enough to hold the resulting string.

**Example:** char string1 [81], string2 [] = “memory”; strcpy (string1, string2) ;

cout<< string1; //outputs “memory” copied into string1

**strcmp ( string1, string2 );**

The strcmp function compares the string1 to string2 and returns an integer value to show the status of the comparison. A value of **0** indicates that the two strings are identical. A value of **less than 0** shows that string1 is lexicographically (according to alphabetic ordering) less than string2. A value of **greater than 0** shows that string1 is lexicographically (according to alphabetic ordering) greater than string2.

**strlen(string1);**

The strlen function returns an integer equal to the **length** of the stored string including blanks, not including the termination character.

**strchr(string, ch );**

The strchr function searches string for the first occurrence of ch. This function only tells whether the string contains ch or not and it will not tell the position of the ch in the string if found.

* 1. **Examples:**

**Example 15.1:** Write a program that prints a string.

#include <iostream>

using namespace std;

int main()

{

char array[20] = "Mir Zakriya Javed";

cout<<array<<endl;

return 0;

}

The output of the program is:

Graphical user interface, text

Description automatically generated

**Example 15.2:** Write a program that input a string. (Using getline)

#include <iostream>

using namespace std;

int main()

{

char array[20];

cout<<"Enter a string: ";

cin.getline(array, 20);

cout<<"You entered the string: "<<array<<endl;

return 0;

}

The output of the program is:

Text

Description automatically generated

**Example 15.3:** Write a program that input a string. (Using get method)

#include <iostream>

using namespace std;

int main()

{

char str[10];

cout << "Enter a string: ";

cin.get(str, 10);

cout << "You entered: " << str << endl;

return 0;

}

**Example 15.4:** Write a Program that Inputs a string using cin only.

#include <iostream>

using namespace std;

int main()

{

char array[20];

cout<<"Enter a string: ";

cin>>array;

cout<<"You entered the string: "<<array<<endl;

return 0;

}

Text

Description automatically generatedThe output of the program is:

Note that the cin can only input single word strings, to receive strings containing spaces use the

getline function.

**Example 15.5:** Operations on Strings

#include <iostream>

#include <string>

using namespace std;

int main ()

{

string str1 = "Muslim";

string str2 = "Abbas";

string str3;

int len ;

// copy str1 into str3

str3 = str1;

cout << "str3 : " << str3 << endl;

// concatenates str1 and str2

str3 = str1 + str2;

cout << "str1 + str2 : " << str3 << endl;

// total length of str3 after concatenation

len = str3.size();

cout << "str3.size() : " << len << endl;

return 0;

}

The output of the program is:

Graphical user interface, text

Description automatically generated

**Example 15.6:** Demonstation of Library functions.

#include <iostream>

#include <cstring>

using namespace std;

int main ()

{

char str1[10] = "Muslim";

char str2[10] = "Abbas";

char str3[10];

int len ;

// copy str1 into str3

strcpy( str3, str1);

cout << "strcpy( str3, str1) : " << str3 << endl;

// concatenates str1 and str2

strcat( str1, str2);

cout << "strcat( str1, str2): " << str1 << endl;

// total lenghth of str1 after concatenation

len = strlen(str1);

cout << "strlen(str1) : " << len << endl;

return 0;

}

The output of the program is:

Text

Description automatically generated

**Example 15.:** Write a program to compare two strings using *strcmp*. Also make use of *gets()* to read the string.

#include <iostream> #include <cstring>

using namespace std; int main () {

char a[100], b[100];

cout<<"Enter the first string: "; gets(a);

cout<<"Enter the second string: "; gets(b);

if (strcmp(a,b) == 0)

cout<<"Entered strings are equal.\n"; else

cout<<"Entered strings are not equal.\n";

return 0;

}

The output of the program is:

Text

Description automatically generated

**Example 15.8:** Write a program that prints string in reverse order.

#include <iostream> #include <cstring>

using namespace std; int main () {

char arr[100];

cout<<"Enter a string to reverse\n";

gets(arr);

strrev(arr);

cout<<"Reverse of entered string is: "<<arr<<endl;

return 0;

}

The output of the program is:

Text

Description automatically generated

* 1. **Exercises for lab**

**Exercise 13.1** Write a program that reads a string and print the number of vowels letters.

**Exercise 13.2** Write a Program number that reads a string and print upper case and number of lower case letters.

**Exercise 13.3** Write a program that reads a string and print the string with the first letter capitalized and the remaining in lower case.

* 1. **Home Work** 
     1. Write a program that reads a string and print the reverse of that string:

e.g. User Enter string: This is a cat

Output: - tac a si siht

* + 1. Write a program that reads a string and print the string’s words in reverse order:

e.g. User Enter string: This is a cat

Output: - cat a is This

Hints: - Use two arrays of same size. One is used to take value from user and other is used to store reverse string.

**Functions and Templates**

**Objective(s):**

##### To Understand about:

**1.** Function Overloading in C++

## Overloading Functions

Sometimes you will create two or more functions that perform the same operation, but use a different set of parameters or parameters of different data types. For instance, in below example there is a square function that uses a double parameter. But, suppose you also wanted a square function that works exclusively with integers, accepting an int as its argument. Both functions would do the same thing: return the square of their argument. The only difference is the data type involved in the operation. If you were to use both these functions in the same program, you could assign a unique name to each function. For example, the function that squares an int might be named squareInt, and the one that squares a double might be named squareDouble. C++, however, allows you to *overload* function names. That means you may assign the same name to multiple functions, as long as their parameter lists are different. Program 6-27 uses two overloaded square functions.

## Create a Function

## void myFunction() {

## // code to be executed

## }

## Call a Function

## // Create a function

## void myFunction() {

## cout << "I just got executed!";

## }

## int main() {

## myFunction(); // call the function

## return 0;

## } // Outputs "I just got executed!"

## Function Declaration and Definition

## void myFunction() { // declaration // the body of the function (definition)

## }

## Parameters and Arguments

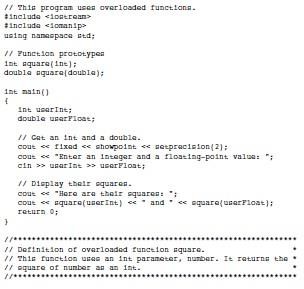
## void functionName(parameter1, parameter2, parameter3) {

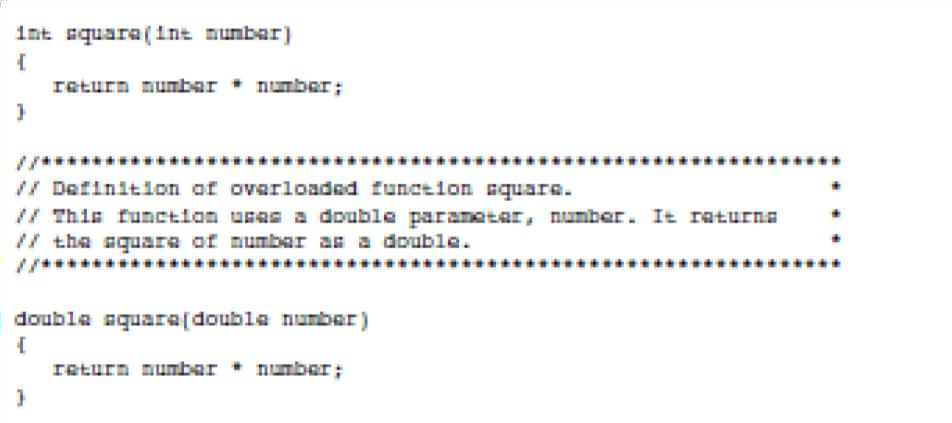
## // code to be executed

## }

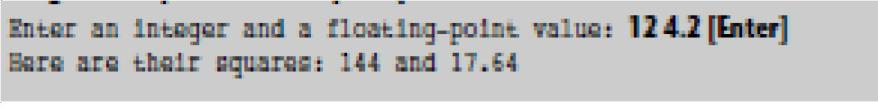
##### Example:

##### // **Function declaration** void myFunction(); // The main method int main() {   myFunction();  // **call** the function   return 0; } // **Function definition** void myFunction() {   cout << "I just got executed!"; }



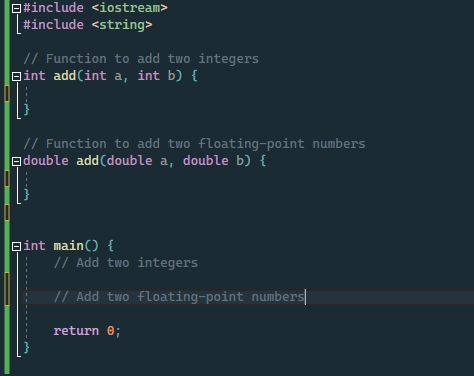


**Program Output:**



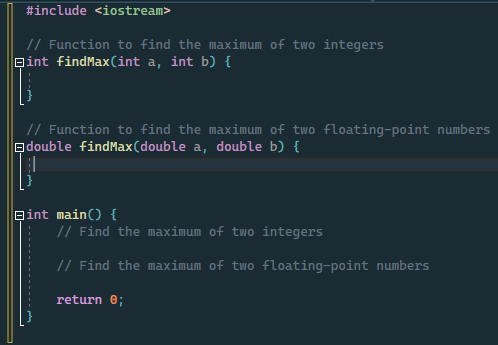
**Task 1: Function Overloading for Addition**

Write a C++ program that demonstrates function overloading for addition. Create multiple functions named `add` with different parameter lists to add two integers, two floating-point numbers, and two strings. Call these functions from `main` and display the results.



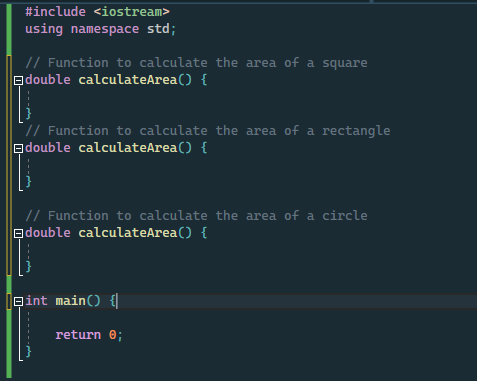
**Task 2: Function Overloading for Finding Maximum**

Write a C++ program that demonstrates function overloading for finding the maximum of different data types. Create a function named `findMax` that takes two parameters of any data type and returns the maximum value. Overload this function for integers, floating-point numbers, and characters. Call these overloaded functions from `main` to find and display the maximum values.



**Task 3: Function Overloading for Area Calculation**

Create a C++ program that demonstrates function overloading for calculating the area of various geometric shapes. Write functions named `calculateArea` to calculate the area of a square, a rectangle, and a circle. Overload the function for each shape, and call these functions from `main` to calculate and display the areas.



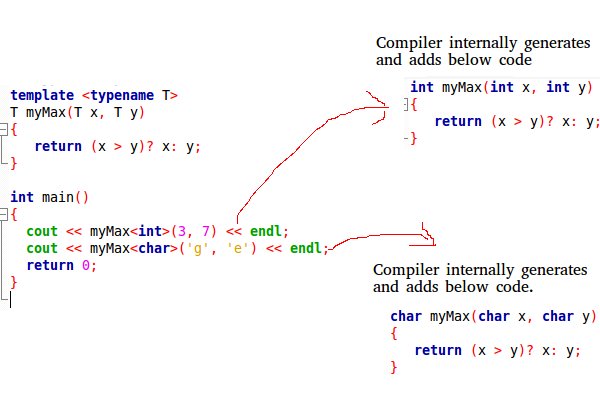
**Templates:**

A **template** is a simple yet very powerful tool in C++. The simple idea is to pass the data type as a parameter so that we don’t need to write the same code for different data types. For example, a software company may need to sort() for different data types. Rather than writing and maintaining multiple codes, we can write one sort() and pass the datatype as a parameter.

C++ adds two new keywords to support templates: ***‘template’*** and ***‘type name’***. The second keyword can always be replaced by the keyword **‘class’**.

## **How Do Templates Work?**

Templates are expanded at compiler time. This is like macros. The difference is, that the compiler does type-checking before template expansion. The idea is simple, source code contains only function/class, but compiled code may contain multiple copies of the same function/class.



|  |
| --- |
| #include <iostream>  **using** **namespace** std;    // One function works for all data types.  This would work  // even for user defined types if operator '>' is overloaded  **template** <**typename** T> T myMax(T x, T y)  {  **return** (x > y) ? x : y;  }    **int** main()  {      // Call myMax for int      cout << myMax<**int**>(3, 7) << endl;      // call myMax for double      cout << myMax<**double**>(3.0, 7.0) << endl;      // call myMax for char      cout << myMax<**char**>('g', 'e') << endl;    **return** 0;  } |