

CC-211L

Object Oriented Programming

Laboratory 01

Introduction to C++ and OOP

Version: 1.0.0

Release Date: 03-01-2023

Department of Information Technology

University of the Punjab

Lahore, Pakistan

Contents:

- Learning Objectives
- Required Resources
- General Instructions
- Background and Overview
 - Strings C++
 - Class
 - Object
 - Member Function
- Activities
 - Pre-Lab Activity
 - Printing a Line of Text
 - #include preprocessor directive
 - Reading an integer from user
 - Task 01: First C++ Program
 - Task 02: Middle Number
 - In-Lab Activity
 - Strings in C++
 - Strings as Variables
 - Input/Output with Strings
 - String Operators
 - String Processing
 - length() and size() function
 - at(index)
 - Object Orientation in C++
 - Class
 - Declaring Objects of Class
 - Accessing Data Members
 - Task 01: Iterate Strings
 - Task 02: Palindrome
 - Task 03: Geometry
 - Task 04: Find and Correct errors
 - Post-Lab Activity
 - Class Methods
 - Task 01: Cars
- Submissions
- Evaluations Metric
- References and Additional Material
- Lab Time and Activity Simulation Log

Learning Objectives:

- New Concepts of C++
- Introduction to OOP

Resources Required:

- Desktop Computer or Laptop
- Microsoft ® Visual Studio 2022

General Instructions:

- In this Lab, you are **NOT** allowed to discuss your solution with your colleagues, even not allowed to ask how is s/he doing, this may result in negative marking. You can **ONLY** discuss with your Teaching Assistants (TAs) or Lab Instructor.
- Your TAs will be available in the Lab for your help. Alternatively, you can send your queries via email to one of the followings.

Teachers:		
Course Instructor	Prof. Dr. Syed Waqar ul Qounain	swjaffry@pucit.edu.pk
Teacher Assistants	Saad Rahman	bsef19m021@pucit.edu.pk
	Zain Ali Shan	bcsf19a022@pucit.edu.pk

Background and Overview:

Strings C++:

A string in C++ is a type of object representing a collection (or sequence) of different characters. Strings in C++ are a part of the standard string class (`std::string`). The string class stores the characters of a string as a collection of bytes in contiguous memory locations.

Class:

The notion of class was invented by an Englishman to keep the general population happy. It derives from the theory that people who knew their place and function in society would be much more secure and comfortable in life than those who did not. The famous Dane, Bjarne Stroustrup, who invented C++, undoubtedly acquired a deep knowledge of class concepts while at Cambridge University in England, and appropriated the idea very successfully for use in his new language.

A class is a data type that you define. It can contain data elements, which can either be variables of the basic types in C++ or other user-defined types. The data elements of a class may be single data elements, arrays, pointers, arrays of pointers of almost any kind or objects of other classes, so you have a lot of flexibility in what you can include in your data type. A class can also contain functions which operate on objects of the class by accessing the data elements that they include.

Object:

When you define a class, you define a blueprint for a data type. This doesn't actually define any data, but it does define what the class name means, that is, what an object of the class will consist of and what operations can be performed on such an object. It's much the same as if you wrote a description of the basic type double. This wouldn't be an actual variable of type double, but a definition of how it's made up and how it operates.

Member Function:

The data and functions within a class are called members of the class. Funnily enough, the members of a class that are data items are called data members and the members that are functions are called function members or member functions.

Activities:

Pre-Lab Activities:

Printing a Line of Text:

Following program is used to print a line in C++.



```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     cout << "Welcome to C++";
6 }
```

Fig 01. (Printing a Line of Text)

The output of the above program is:



```
Microsoft Visual Studio Debug Console
Welcome to C++
```

Fig 02. (Printing a Line of Text)

#include Preprocessor directive:

#include <iostream> is a preprocessing directive, which is a message to the C++ preprocessor. Lines that begin with # are processed by the preprocessor before the program is compiled. This line notifies the preprocessor to include in the program the contents of the input/output stream header. This header is a file containing information the compiler uses when compiling any program that outputs data to the screen or inputs data from the keyboard using C++'s stream input/output.

The std namespace:

The std namespace is required when we use names that we've brought into the program by the preprocessing directive #include. The notation specifies that we are using a name, in this case cout, that belongs to namespace std. The names cin (the standard input stream) and cerr (the standard error stream) also belong to namespace std.

The Stream Insertion Operator:

In the context of an output statement, the << operator is referred to as the stream insertion operator. When this program executes, the value to the operator's right, the right operand, is inserted in the output stream.

Reading an integer from user:


Following program is used to print a line in C++.



```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     int number;
6     cout << "Enter an integer: ";
7     cin >> number;
8
9     cout << "The entered number is " << number << ".";
10 }
```

Fig 03. (Reading an Integer)

The output of the above program is:

A screenshot of the Microsoft Visual Studio Debug Console window. The window has a title bar with the text "Microsoft Visual Studio Debug Console" and standard window controls (minimize, maximize, close). The console area is black with white text. It shows two lines of output: "Enter an integer: 10" and "The entered number is 10." There is a small upward-pointing arrow on the right side of the console area, indicating it can be scrolled.

```
Microsoft Visual Studio Debug Console
Enter an integer: 10
The entered number is 10.
```

Fig 04. (Reading an Integer)

Task 01: First C++ Program**[Estimated 10 minutes / 10 marks]**

Write a C++ program which:

- Displays “**This is my first C++ program**” on the Console

Task 02: Middle Number**[Estimated 10 minutes / 10 marks]**

Write a C++ program which:

- Declares three integers
- Read the integers from the user
- Displays the integer which is neither the smallest nor the largest among the three on the Console

Note: if statement is same as used in C language.

Input	Output
2 3 4	Middle number is: 3
2 5 -1	Middle number is: 2

In-Lab Activities:**Strings in C++:****Strings as Variables:**

To create a variable of type string, simply:

“string yourName;” Assignment, is the same:

“yourName = “Saad”;;”

Or like before, we could always combine the two with an initialization:

“string ~ yourName = “Saad”;;”

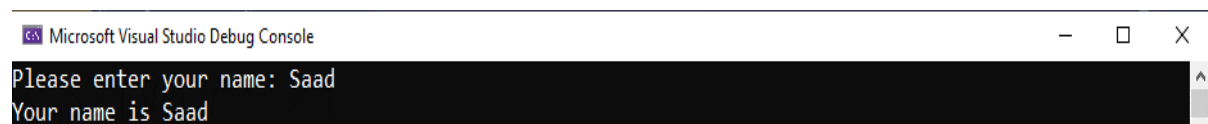
Input/Output with Strings:

I/O with string is shown in the code below:

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      string yourName;
5      cout << "Please enter your name: ";
6      cin >> yourName; //get the name
7      cout << "Your name is " << yourName;
8  }
```

Fig 05. (Strings I/O)

The output of the above program is:



```
Microsoft Visual Studio Debug Console
Please enter your name: Saad
Your name is Saad
```

Fig 06. (Strings I/O)

The code above only input characters before first space character because the space characters are termination for cin as shown in figure below:



```
Microsoft Visual Studio Debug Console
Please enter your name: Saad Rahman
Your name is Saad
```

Fig 07. (Strings I/O)

To read a complete line “getline()” function is used with a preprocessor directive “#include <string>”.

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4  int main() {
5      string yourName;
6      cout << "Please enter your name: ";
7      getline(cin, yourName); //get the name
8      cout << "Your name is " << yourName;
9  }
```


Fig 08. (Strings I/O)

The output of the above program is:



```
Microsoft Visual Studio Debug Console
Please enter your name: Saad Rahman
Your name is Saad Rahman
```

Fig 09. (Strings I/O)

String Operators:

Assignment:

Assignment (=): As used before, assign a string to a variable of type string.

```
string Name = "Deitel";
```

```
string anotherName = Name;
```

Both now hold **"Deitel"**.

Concatenation:

"+" sign is used to put the string at the end of the other.



```
1 #include <iostream>
2 #include <string>
3 using namespace std;
4 int main() {
5     string firstName = "Harvey";
6     string lastName = "Deitel";
7     string fullName = firstName + " " + lastName;
8     cout << firstName << endl;
9     cout << lastName << endl;
10    cout << fullName;
11 }
```

Fig 10. (String Concatenation)

The output of the above code is:



```
Microsoft Visual Studio Debug Console
Harvey
Deitel
Harvey Deitel
```

Fig 11. (String Concatenation)

String Processing:

The string library offers many useful string processing methods.

length() and size():

This method returns the integer length of the string. The length() and size() are the same.

```

1  #include <iostream>
2  #include <string>
3  using namespace std;
4  int main() {
5      string firstName = "Harvey";
6      cout << "Length Function: " << firstName.length() << endl;
7      cout << "Size Function: " << firstName.size() << endl;
8  }

```

Fig 12. (length() and size() function)

The output of the above program is:



```

Microsoft Visual Studio Debug Console
Length Function: 6
Size Function: 6

```

Fig 13. (length() and size() function)

at(index):

This method returns the character at the specified index. Indices start from 0.

```

1  #include <iostream>
2  #include <string>
3  using namespace std;
4  int main() {
5      string firstName = "Harvey";
6      cout << "First Element: " << firstName.at(0) << endl;
7      cout << "Last Element: " << firstName.at(firstName.size()-1) << endl;
8  }

```

Fig 14. (String at())

The output of the above program is:



```

Microsoft Visual Studio Debug Console
First Element: H
Last Element: y

```

Fig 15. (String at())

Following is the alternate of at():

```

1  #include <iostream>
2  #include <string>
3  using namespace std;
4  int main() {
5      string firstName = "Harvey";
6      cout << "First Element: " << firstName[0] << endl;
7      cout << "Last Element: " << firstName[firstName.size()-1] << endl;
8  }

```

Fig 16. (String at() alternate)

Object Orientation in C++:

Class:

A class is a data type that you define. It can contain data elements, which can either be variables of the basic types in C++ or other user-defined types. The data elements of a class may be single data elements, arrays, pointers, arrays of pointers of almost any kind or objects of other classes, so you have a lot of flexibility in what you can include in your data type. A class can also contain functions which operate on objects of the class by accessing the data elements that they include. So, a class combines both the definition of the elementary data that makes up an object and the means of manipulating the data belonging to individual instances of the class.

Let's create a class of boxes, we will define Box datatype using the keyword class as follows:

```
4 class Box {  
5     public:  
6         double length;  
7         double breadth;  
8         double height;  
9     };
```

Fig 17. (Class)

The name that we've given to our class appears following the keyword and the three data members are defined between the curly braces. The data members are defined for the class using the declaration statements that we already know and the whole class definition is terminated with a semicolon. The names of all the members of a class are local to a class. You can therefore use the same names elsewhere in a program without causing any problems.

Declaring Objects of Class:

We declare objects of class with following statements:

Box box1;

Box box2;

Both of the objects Box1 and Box2 will, of course, have their own data members. The object name Box1 embodies the whole object, including its three data members. They are not initialized to anything, however - the data members of each object will simply contain garbage values.

Accessing Data Members:

The data members of objects of a class can be referred to using the direct member access operator ".". So, to set the value of the data member length of the object box2 to, say, 10.0, we could write this assignment statement:

box2.length = 18.0; // Setting the value of a data member.

Following code demonstrates the creation of objects and accessing the data members:

```
1  #include <iostream>
2  using namespace std;
3  class Box {
4  public:
5
6      double length;
7      double breadth;
8      double height;
9  };
10 int main()
11 {
12     Box box1;
13     Box box2; //Class Objects
14     //Data Members of box1
15     box1.length = 10;
16     box1.height = 10;
17     box1.breadth = 10;
18     //Data Members of box2
19     box2.length = 5;
20     box2.breadth = 5;
21     box2.height = 5;
22
23     int vol1 = box1.length * box1.height * box1.breadth;
24     int vol2 = box2.length * box2.height * box2.breadth;
25
26     cout << "Volume of Box1: " << vol1 << endl;
27     cout << "Volume of Box2: " << vol2 << endl;
28 }
```

Fig 17. (Class)

The output of the above program is:



Microsoft Visual Studio Debug Console

```
Volume of Box1: 1000
Volume of Box2: 125
```

Fig 18. (Class)

Task 01: Iterate String**[30 minutes / 30 marks]**

Write a C++ program that:

- Inputs a string from user
- Count the number of words and spaces in the strings
- Display the count on Console

Input	Output
This is a string.	Words: 4 Spaces: 3
String	Words: 1 Spaces: 0

Task 02: Palindrome**[30 minutes / 30 marks]**

Create a C++ program that

- Inputs a string from user
- Check if string is Palindrome or not
- Display the result on the Console

Input	Output
101	Palindrome
100	Not a Palindrome
1	Palindrome

Task 03: Geometry**[20 minutes / 30 marks]**

Create a C++ Program that:

- Creates a Class Geometry with data members
 - Length
 - Width
- Create an object of above Class
- Ask user to Input the Length and Width
- Check if length and width are equal
- Display “Square” on the Console if they are equal otherwise display “Rectangle”

Input	Output
Length: 10 Width: 10	Square
Length:10 Width: 5	Rectangle

Task 04: Find and Correct the error**[20 minutes / 20 marks]**

Find and correct the errors from the following code:

```
#include <iostream>
class Student {
public:
    string name;
    string rollNo;
    string subjects[5];
}
main()
{
    Student std1, std2;

    Student.name = "Name";

    std2.name = std1.name;
    std2.rollNo = std1.rollNo;
    std2.subjects = std1.subjects;

    cout << std2 << endl;
}
```

Fig 19. (In-Lab Task 04)

Post-Lab Activities:**Class Methods:**

Methods are functions that belongs to a Class. They can be defined inside and outside of a Class.

Following code shows a method defined inside a Class:

```
1  #include <iostream>
2  using namespace std;
3  class Box {
4  public:
5
6      double length;
7      double breadth;
8      double height;
9      int Volume() {
10         return length * breadth * height;
11     }
12 };
13 int main()
14 {
15     Box box1;
16
17     box1.length = 10;
18     box1.height = 10;
19     box1.breadth = 10;
20
21     int vol1 = box1.Volume();
22     cout<<"Volume of Box1: "<<vol1<<endl;
23 }
```

Fig 20. (Class Methods)

Following code shows a method defined outside a Class:

```
1  #include <iostream>
2  using namespace std;
3  class Box {
4  public:
5
6      double length;
7      double breadth;
8      double height;
9      int Volume();
10 };
11
12 int Box::Volume() {
13     return length * breadth * height;
14 }
15
16 int main()
17 {
18     Box box1;
19
20     box1.length = 10;
21     box1.height = 10;
22     box1.breadth = 10;
23
24     int vol1 = box1.Volume();
25     cout<<"Volume of Box1: "<<vol1<<endl;
26 }
```

Fig 21. (Class Methods)

The scope resolution operator is used to show the relationship of the function with the particular Class.

The output for both of the above codes is same:



Fig 22. (Class Methods)

Task 01: Cars**[Estimated 60 minutes / 50 marks]****Part (a):**

Create a C++ program that:

- Makes a car Class with data members:
 - wheels
 - doors
 - currentSpeed
- And with Class Methods
 - speed (Inside Class)
 - break (Outside Class)
- Make two objects in the name of ferrari and mustang.
- Every time when you call speed function currentSpeed is increased by 5 while break function decrease it by 5.

Submissions:

- For In-Lab Activity:
 - Save the files on your PC.
 - TA's will evaluate the tasks offline.
- For Pre-Lab & Post-Lab Activity:
 - Submit the .c file on Google Classroom and name it to your roll no.

Evaluations Metric:

- All the lab tasks will be evaluated offline by TA's
- **Division of Pre-Lab marks:** [20 marks]
 - Task 01: First C++ Program [10 marks]
 - Task 02: Middle Number [10 marks]
- **Division of In-Lab marks:** [110 marks]
 - Task 01: Iterate String [30 marks]
 - Task 02: Palindrome [30 marks]
 - Task 03: Geometry [30 marks]
 - Task 04: Find and Correct errors [20 marks]
- **Division of Post-Lab marks:** [50 marks]
 - Task 01: Cars [50 marks]

References and Additional Material:

- C++ Strings
https://www.w3schools.com/cpp/cpp_strings.asp
- C++ Input/Output
https://www.w3schools.com/cpp/cpp_user_input.asp
- Classes and Objects
https://www.w3schools.com/cpp/cpp_classes.asp

Lab Time Activity Simulation Log:

- Slot – 01 – 00:00 – 00:15: Class Settlement
- Slot – 02 – 00:15 – 00:30: In-Lab Task
- Slot – 03 – 00:30 – 00:45: In-Lab Task
- Slot – 04 – 00:45 – 01:00: In-Lab Task
- Slot – 05 – 01:00 – 01:15: In-Lab Task
- Slot – 06 – 01:15 – 01:30: In-Lab Task
- Slot – 07 – 01:30 – 01:45: In-Lab Task
- Slot – 08 – 01:45 – 02:00: In-Lab Task
- Slot – 09 – 02:00 – 02:15: In-Lab Task
- Slot – 10 – 02:15 – 02:30: In-Lab Task
- Slot – 11 – 02:30 – 02:45: Evaluation of Lab Tasks
- Slot – 12 – 02:45 – 03:00: Discussion on Post-Lab Task