

CC211L

Object Oriented Programming

Laboratory 01

Introduction to C++ and OOP

Version: 1.0.0

Release Date: 25012024

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
 - PreLab Activity
 - Printing a Line of Text
 - `#include` preprocessor directive
 - Reading an integer from user
 - Reading a string from user
 - Task 01: First C++ Program
 - Task 02: Middle Number
 - Task-03 : Student Class

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 | Syed Jaffar | bitf21m032@pucit.edu.pk |
| | Aqdas Shabbir | bitf21m022@pucit.edu.pk |
| | Ibad Nazir | bitf21m024@pucit.edu.pk |
| | Muhammad Umer | bitf21m010@pucit.edu.pk |
| | Muhammad Usman Munir | bitf19m020@pucit.edu.pk |

Background and Overview:

Shift from C to Cpp:

Cpp is extension of C language. The initial major difference between the two languages was introduction of classes in Cpp. That is why Cpp was initially called as C with Classes. Here are the following points of relation between the two languages:

1. C++ as an Extension of C:

C++ was developed as an extension of the C programming language. The early versions of C++ (C with Classes) were essentially C with additional features.

2. Backward Compatibility:

C++ is designed to be backward compatible with C. This means that most of the valid C programs can be compiled and executed in a C++ compiler without any or with minimal modifications.

3. Common Syntax:

C++ shares a significant portion of its syntax and basic structure with C. Many of the C keywords and constructs are valid in C++.

4. C Features in C++:

All features of C, such as pointers, arrays, structures, and functions, are available in C++. C++ allows Cstyle programming, making it familiar to developers already proficient in C.

5. Header Files:

C++ supports the use of Cstyle header files (e.g., `.h` files) to declare functions and constants. This allows existing C code to be easily integrated into a C++ program.

6. Common Compilation Process:

The compilation process for both C and C++ involves a preprocessor, compiler, assembler, and linker. C++ compilers can typically process both C and C++ code, allowing for mixedlanguage programming.

7. Compatibility Libraries:

C++ includes the Standard Template Library (STL), which provides generic classes and functions. However, it also includes the C Standard Library, allowing C++ programs to use C functions seamlessly.

8. Manual Memory Management:

Both C and C++ allow manual memory management using functions like `'malloc'` and `'free'`. This makes it easy to share memory management concepts between the two languages.

9. Function Prototypes:

The way functions are declared in C++ is similar to C, allowing for compatibility. C++ also supports function overloading, but C functions can be called from C++ without overloading.

10. Object Oriented Features:

C++ introduces object oriented programming (OOP) features like classes, objects, and inheritance, which are not present in the original C language. However, C++ developers can still choose to write procedural code without using these features.

Differences:

C and C++ are related programming languages, but they have distinct differences. Here are some key contrasts between C and C++:

1. Programming Paradigm:

C: C is a procedural programming language. It follows a structured programming paradigm and is focused on procedures or routines.

C++: C++ is a multi paradigm language that supports both procedural and objectoriented programming (OOP) paradigms. It extends C by adding classes and objects for better organization and encapsulation of code.

2. Class and Object:

C: Does not have the concept of classes and objects.

C++: Introduces the concept of classes and objects, allowing for the implementation of object oriented principles like encapsulation, inheritance, and polymorphism.

3. Memory Management:

C: Relies on manual memory management using functions like `'malloc'` and `'free'`.

C++: Supports both manual memory management and automatic memory management through features like constructors, destructors, and the `'new'` and `'delete'` operators. Additionally, C++ has features like smart pointers for more robust memory management.

4. Function Overloading:

C: Does not support function overloading. In C, function names must be unique.

C++: Supports function overloading, allowing multiple functions with the same name but different parameter lists. This contributes to a more expressive and flexible code structure.

5. Operator Overloading:

C: Does not support operator overloading.

C++: Allows overloading of operators for user defined types. This feature enhances the readability and usability of code.

6. Standard Template Library (STL):

C: Does not have a standard template library.

C++: Provides a powerful Standard Template Library (STL) that includes generic classes and functions, such as containers, algorithms, and iterators, making it easier to write efficient and generic code.

7. Header Files:

C: Uses header files (e.g., `.h` files) for function prototypes and declarations.

C++: Extends the use of header files but often uses `.hpp` for headers that contain C++ code. Additionally, it uses `.h` files for compatibility with C.

8. Namespace:

C: Does not have the concept of namespaces.

C++: Introduces namespaces, allowing for better organization and avoiding naming conflicts.

9. Compatibility:

C: Code written in C is often compatible with C++.

C++: While C++ is designed to be backward compatible with C, there are some cases where C code might need modifications to compile and run as C++ code.

Conclusion:

While C++ incorporates many features and concepts not found in C, the relationship between the two languages allows for a smooth transition from C to C++. Developers can leverage their knowledge of C when learning 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 userdefined 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.

Data Members:

In object-oriented programming, a data member is a variable that is a part of a class and holds data relevant to the class. Data members represent the attributes or properties of objects created from the class. They contribute to the state of an object and encapsulate the data within the class.

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:

PreLab Activities:

Printing a Line of Text:

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

A screenshot of a code editor showing a C++ program. The code is as follows:

```
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:

A screenshot of the Microsoft Visual Studio Debug Console. The console window is titled "Microsoft Visual Studio Debug Console" and shows the output "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++.

A screenshot of a code editor showing a C++ program. The code is as follows:

```
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 Visual Studio logo and the text 'Microsoft Visual Studio Debug Console'. The console area is black with white text. It shows two lines of output: 'Enter an integer: 10' and 'The entered number is 10.'.

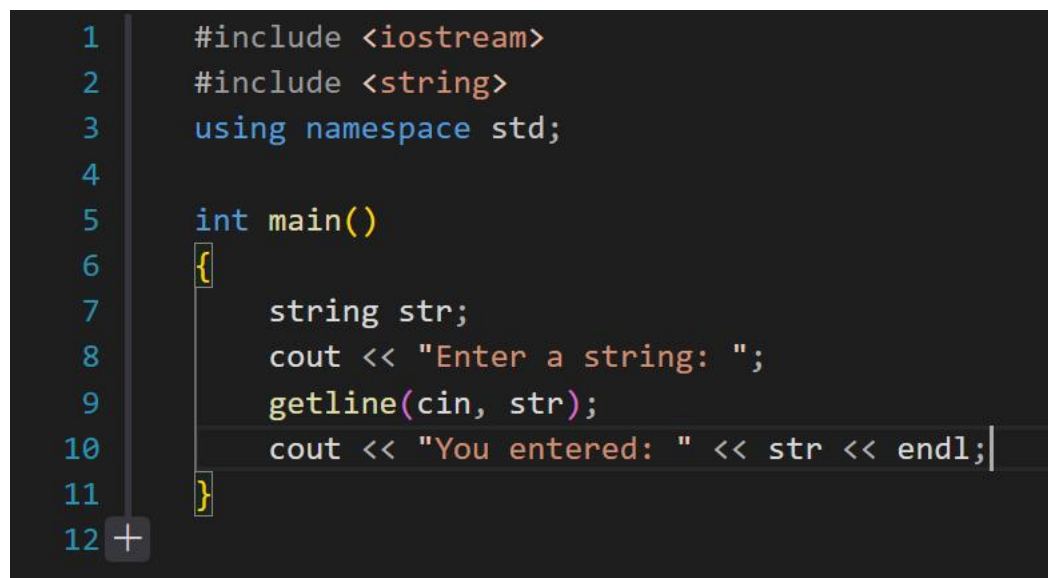
```
Microsoft Visual Studio Debug Console
Enter an integer: 10
The entered number is 10.
```

Fig 04. (Reading an Integer)

Working with Strings in cpp:

In C we work with strings as character arrays. But in Cpp, we are provided with a new header file `<string>`. So we have a new Data type **string**. Strings in cpp have their own built in functions, which can be used to access, use and manipulate data in a string.

Example:

A screenshot of a code editor showing C++ code. The code is as follows:

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  int main()
6  {
7      string str;
8      cout << "Enter a string: ";
9      getline(cin, str);
10     cout << "You entered: " << str << endl;
11 }
12 +
```

In the above example, `getline` function is used. `Getline` function is used to get a text line input from user. When we use strings with `cin`, if we include space in input, only the part before the space is included. But if we are using `getline`, the part after the space and the space is also included in string until user presses Enter key.

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

Write a C++ program which input of string from user. After that take a character and then print the index of second occurrence of that character. If that character does not exist or the second occurrence does not exist then print 1.

Input Case:

| Input | Output |
|---------|--------|
| Pucit p | 1 |
| Apple p | 2 |

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 |

Task 03: Student Class:**[Estimated 20 minutes / 30 marks]**

Write a Cpp program in which:

1- Create following data members:

- Roll number (string)
- Name (string)
- Section (string)
- Cgpa (float)

All the data members should not be directly accessible outside the class.

2- Create following Data members:

- Default constructor
- Setters for all data members
- Getters for all members
- Display all Data

In-Lab Activities:

Strings in C++:

Strings as Variables:

To create a variable of type string, simply:

“string yourName;” Assignment, is the same:

“yourName = “Aqdas”;;”

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

“string ~ yourName = “Jaffar”;;”

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: Umer
Your name is Umer
```

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: Umer Farooq
Your name is Umer
```

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: Ibad Nazir
Your name is Ibad Nazir
```

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 }
```

String find() and substr() functions:

find() function of string is used to find a first occurrence of a character in string. It returns the index number of that first occurrence of the found character. **Substr()** function of string is used to get a sub string from a given string.

```
#include <iostream>
#include <string>
using namespace std;

int main()
{
    string str;
    cout << "Enter Your first and last name: ";
    getline(cin, str);
    int spaceIndex = str.find(" ");
    string firstName = str.substr(0, spaceIndex);
    string lastName = str.substr(spaceIndex + 1, str.length() - spaceIndex);
    cout << "Your first name is: " << firstName << endl;
    cout << "Your last name is: " << lastName << endl;
    return 0;
}
```

In the above code, we initially get first and last name from user. Then we find index of space character in that string. Then we use substr function to get sub string in variable firstName. It will contain the part of string before space. Then we use substr function again to get lastName from the original string.

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.height = 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:

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 minimize, maximize, and close buttons. The console area is black with white text. It displays two lines of output: "Volume of Box1: 1000" and "Volume of Box2: 125". A vertical scrollbar is visible on the right side of the console area.

```
Microsoft Visual Studio Debug Console
Volume of Box1: 1000
Volume of Box2: 125
```

Fig 18. (Class)

Task 01: Count the non repeating**[30 minutes / 30 marks]**

Write a C++ program that:

- Input a string from user.
- Create a function **findCount**.

In this function:

- Find the number of characters in first not repeating group in string.
- You need to calculate number of characters before a repeating character is encountered.
- You wont count the character that is being repeated.
- Return the count from the function.

| Input | Output |
|------------------|----------|
| abcdefghijklgskc | Count: 6 |
| Pucit | Count:5 |
| Ppakistan | Count:0 |

Task 02: Union of Arrays**[30 minutes / 30 marks]**

Create a C++ program that

- Create an array Array1 in main function and takes size of array from user.
- Create another array Array2 and takes size of this array from user as well.
- Create a function “**Input**” to take input from user in an array. You will pass array and size of array to this function
- Create another function “**Union**” that will take an two arrays and their sizes as parameter and return an array that will have union of both arrays.
- Union of two arrays is collection of all elements of both arrays.
- You can assume that each array will have unique elements with respect to itself.

| Input | Output |
|----------------------------|------------------------|
| [1,2,3,4,5] , [6,7,8,9,10] | [1,2,3,4,5,6,7,8,9,10] |
| [1,2,3,4,5], [1,2,3,4,5] | [1,2,3,4,5] |
| [1,2,3,4], [1,2,3,4,5] | [1,2,3,4,5] |

Task 03: Circle class**[20 minutes / 30 marks]**

Create a C++ Program that:

- Creates a structure Circle with data members
 - Radius (float)
 - Pi (float)
- Create Default constructor
- Create setters for both data members
- Create getters for both Data members
- Create a function “isACircle” which will validate if the given values of radius and pi are valid to form a circle.
- This function will return boolean value based on the validation.
- Create an object of above Class
- Ask user to Input of radius and pi.
- If the circle is valid then display the Area of that Circle.
- Formula for Area of Circle is $2 \times \pi \times \text{radius}$.

| Input | Output |
|-------------------------|---------------------|
| Radius: 2 Pi: 3.14 | Area: 12.56 |
| Length:4 Width: 2.44 | Wrong value for pi. |

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

Find and correct the errors from the following code:

```
#include <iostream>

using namespace std;

class myClass
{
    int id
    string name
    myClass()
    {
        id = "one";
        name = Ali;
    }
    display()
    {
        cout << "ID: " << id << endl;
        cout << "Name: " << name << endl;
    }
};

int main()
{
    myClass obj;
    display();
}
```

Task-5: Person Class

create a Employee class:

Attributes:

- CNIC
- Name
- Father's name
- Address

- Phone

All data members will be hidden from class user.

Methods:

- Default constructor
- Setters
- Getters
- Display all data

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
 - maxSpeed
- 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.
- Make sure the current Speed does not go below 0 or above max speed.

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