**Object Oriented Programming Lab 03**

|  |  |
| --- | --- |
| **Course**: Object Oriented Programming (CL1004) | **Semester**: Spring 2025 |
| **Instructor**: Muhammad Monis |  |
| Note:   * Maintain discipline during the lab. * Listen and follow the instructions as they are given. * Just raise hand if you have any problem. * Completing all tasks of each lab is compulsory. * Get your lab checked at the end of the session. |  |

**Classes in C++**

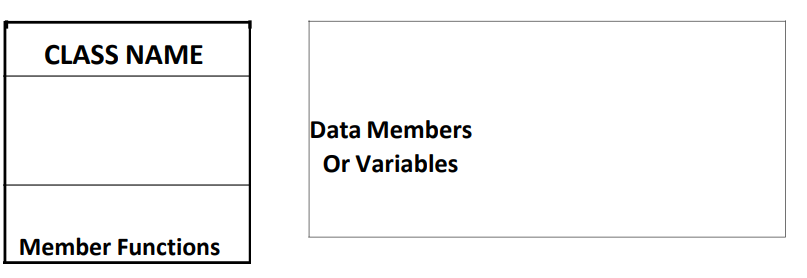
A class is a programmer-defined data type that describes what an object of the class will look like when it is created. It consists of a set of variables and a set of functions. We can think of class as a sketch (prototype) of a house. It contains all the details about the floors, doors, windows etc. based on these descriptions we build the house. House is the object.

As, many houses can be made from the same description, we can create many objects from a class.

Classes are created using the keyword class. A class declaration defines a new type that links code and data. This new type is then used to declare objects of that class. A Class is a user defined data-type which has data members and member functions. Data members are the data variables and member functions are the functions used to manipulate these variables and together these data members and member functions defines the properties and behavior of the objects in a Class.

A class member can be defined as public, private or protected. By default, members would be assumed as private.

In the UML, a class icon can be subdivided into compartments. The top compartment is for the name of the class, the second is for the variables of the class, and the third is for the methods of the class.



**Class Name**

By convention, the name of a user-defined class begins with a capital letter and, for readability, each subsequent word in the class name begins with a capital letter.

**Data Members**

Consider the attributes of some real-world objects:

**RADIO** – station frequency, volume level.

**CAR** – speedometer readings, amount of gas in its tank and what gear it is in.

These attributes form the data in our program. The values that these attributes take (the blue color of the petals, for example) form the state of the object.

**Member Functions**

Consider the operations of some real-world objects:

**RADIO** – setting its station and volume (invoked by the person adjusting the radio’s controls)

**CAR** – accelerating (invoked by the driver), decelerating, turning and shifting gears. These operations form the functions in program. Member functions define the class’s behaviors.

**Objects in C++**

In C++, when we define a variable of a class, we call it instantiating the class. The variable itself is called an instance of the class. A variable of a class type is also called an object. Instantiating a variable allocates memory for the object. Below is the syntax to define an object in C++:

**className objectVariableName**

Following is an example of an object created from a class named as radio:

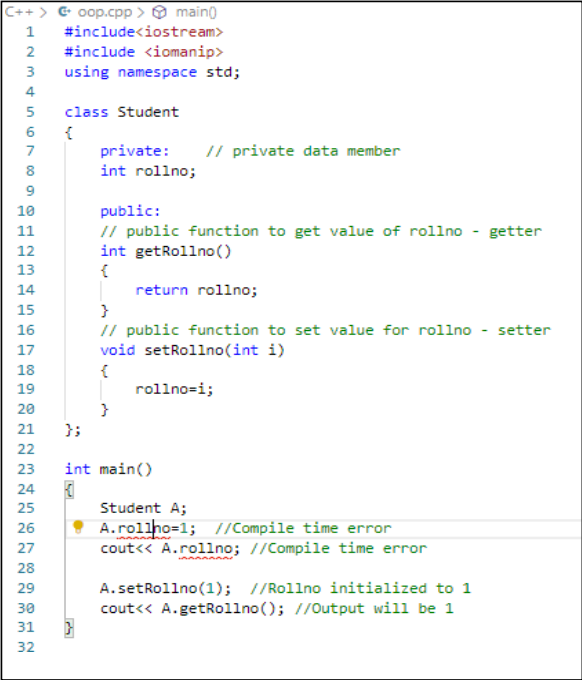
**Radio r;**

**Accessing Public Data Members**

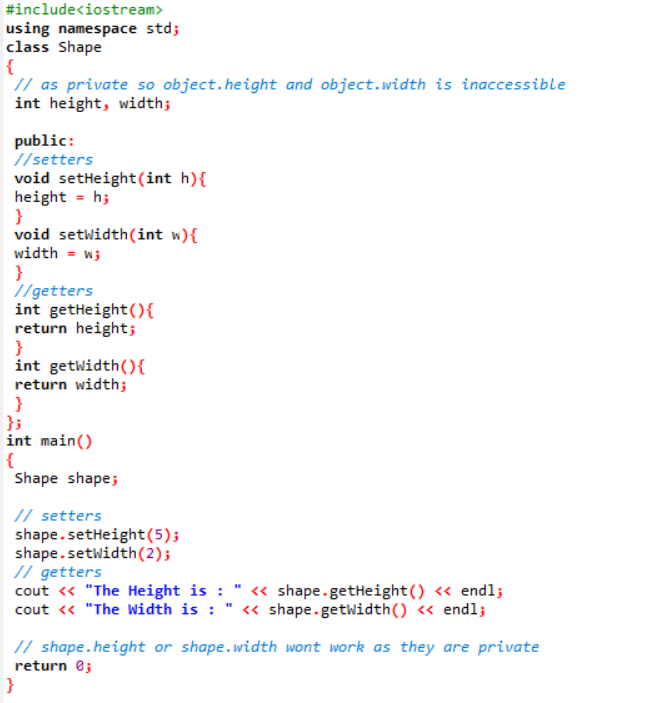
The public data members of objects of a class can be accessed using the direct member access operator (.). However, the private data members are not allowed to be accessed directly by the object. Accessing a data member depends solely on the access control of that data member.

**Accessing Private Data Members**

To access, use and initialize the private data member you need to create getter and setter functions, to get and set the value of the data member. The setter function will set the value passed as argument to the private data member, and the getter function will return the value of the private data member to be used. Both getter and setter function must be defined public.



Consider another example below:



**Getter & Setter , this Keyword in C++ Programming**

The getter function is used to retrieve the variable value and the setter function is used to set the variable value. They this is a keyword that refers to the current instance of the class. They are getters and setters the standard way to provide access to data in Java classes. Setters and Getters allow for an object to contain private variables which can be accessed and changed with restrictions.

A picture containing graphical user interface

Description automatically generated

Graphical user interface, application

Description automatically generated

**Member Functions in Classes**

There are 2 ways to define a member function:

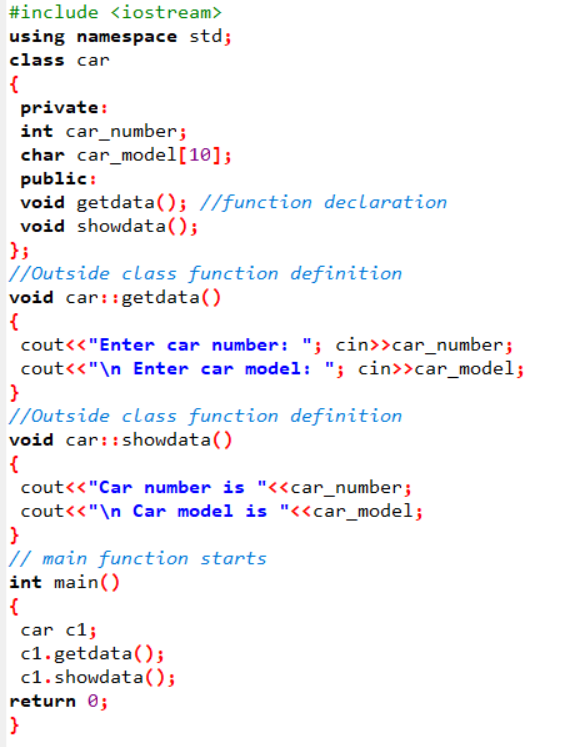
1. Inside class definition.
2. Outside class definition.

**Inside Class Definition**

With an inline function, the compiler tries to expand the code in the body of the function in place of a call to the function. Note that all the member functions defined inside the class definition are by default inline, but you can also make any non-class function inline by using keyword inline with them. Inline functions are actual functions, which are copied everywhere during compilation, like pre-processor macro, so the overhead of function calling is reduced. This type of definition is already shown in the above examples.

**Outside Class Definition**

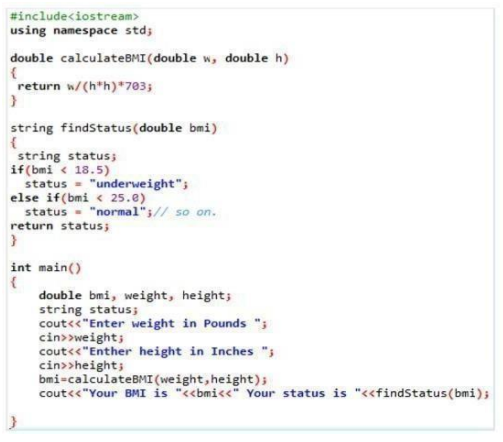
To define a member function outside the class definition we have to use the scope resolution:: operator along with class name and function name.



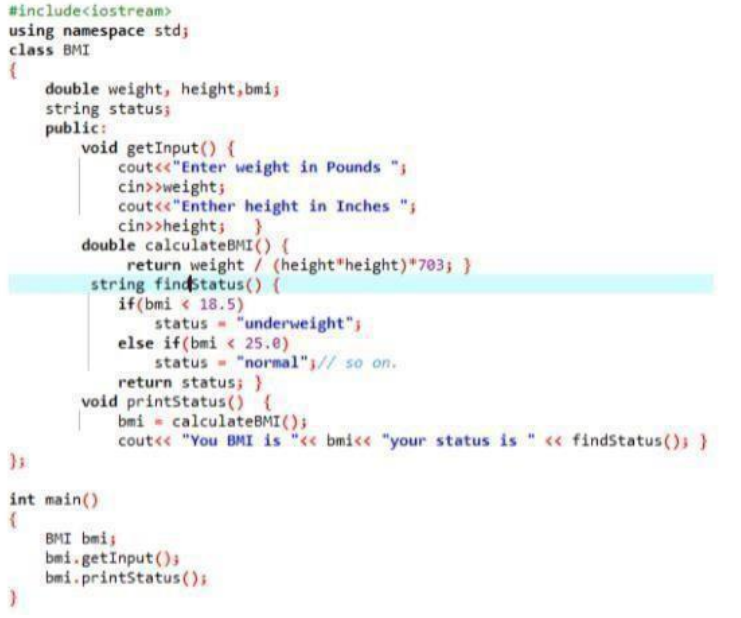
**Struct VS Classes**

By default, all structure fields are public, or available to functions (like the main() function) that are outside the structure. Conversely, all class fields are private. That means they are not available for use outside the class. When you create a class, you can declare some fields to be private and some to be public. For example, in the real world, you might want your name to be public knowledge but your Social Security number, salary, or age to be private.

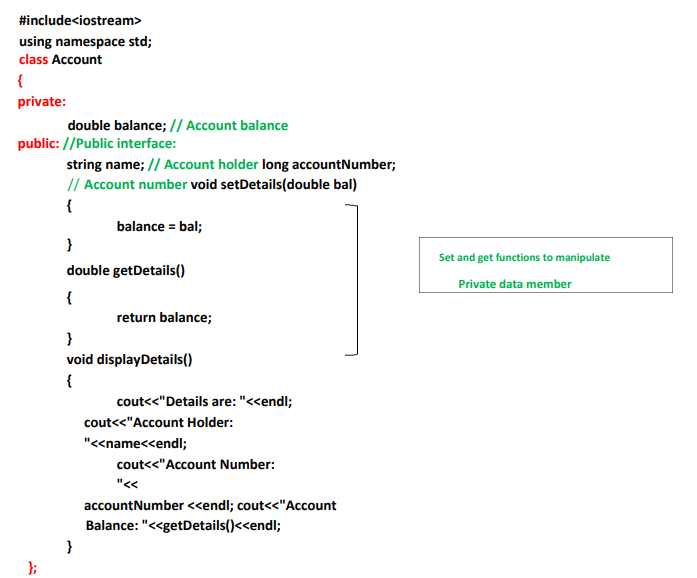
**Transformation from Procedural to Object Oriented Programming**

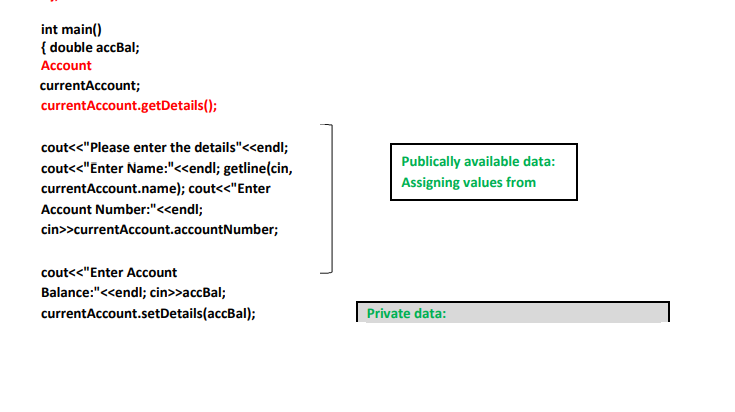


**Procedural Approach**



**Object Oriented Programming Approach**







**Header Files**

Header files are used for declaration. In OOP you should use header files to declare classes and functions. It will make your program look cleaner and more professional.

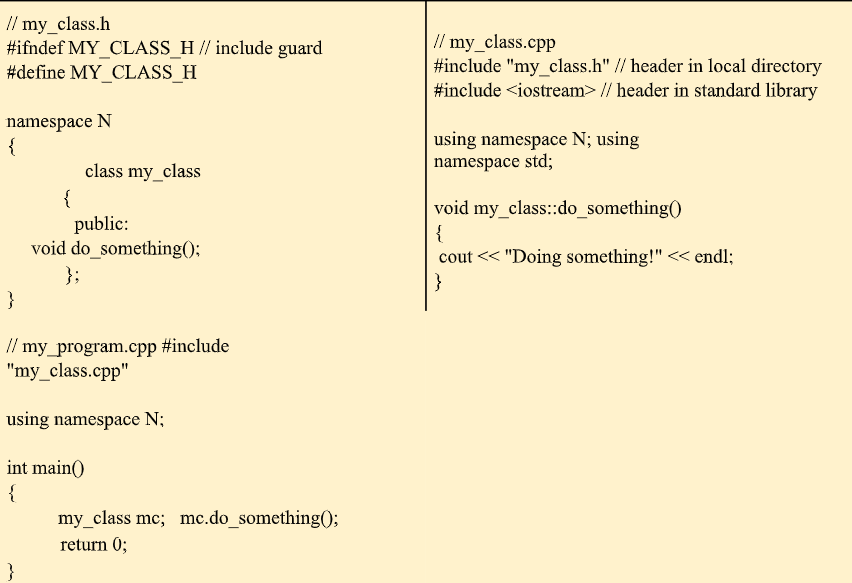
Header file for a class will include:

* Include guards.
* Class definition:
  + Member variables.
  + Function declaration (only prototype).

Implementation File will include:

* Include directive for “header.h”.
* Necessary include directives.
* Function definitions for all the functions of the class.

Consider the below example:



‘

**Exercises: (Note: Inputs must be handled via the main() arguments)**

**Task - 01:**

Create a class User with two public fields: int Age and string Name. In the Main method, create an object of class User and set Name to "Teo" and Age to 24. Then, output to the screen: "My name is {Name} and I'm {Age} years old." using object fields for Name and Age.

**Task - 02:**

Create a class called Date that includes three pieces of information as instance variables—a month (typeint), a day (typeint) and a year (typeint). Provide a method displayDate that displays the month, day and year separated by forward slashes(/). Write a test application named DateTest that demonstrates classDate’s capabilities.

**Task - 03:**

We are prototyping a robot that refills glasses during dinner. Every glass holds 200 milliliters. During dinner, people either drink water or juice, and as soon as there is less than 100 ml left in the glass, the robot refills it back to 200 ml.

Create a class Glass with one public int field LiquidLevel and methods public Drink(int milliliters) that takes the amount of liquid that a person drank and public Refill() that refills the glass to be 200 ml full. Both methods should not return any value. Initially set LiquidLevel to 200. In the Main method create an object of class Glass and read commands from the screen until the user terminates the program (see next). Don't forget to refill the glass when needed!

**Task - 04:**

Create a class called Employee that includes three pieces of information as instance variables—a first name (type String), a last name (type String) and a monthly salary (double). If the monthly salary is not positive, set it to 0.0. Write a test application named EmployeeTest that demonstrates class Employee’s capabilities. Create two Employee objects and display each object’s yearly salary. Then give each Employee a 10% raise and display each Employee’s yearly salary again.

**Task - 05:**

Create a class called Book to represent a book. A Book should include four pieces of information as instance variables‐a book name, an ISBN number, an author name and a publisher. Provide methods (query method) for each instance variable. Inaddition, provide a method named getBookInfo that returns the description of the book as a String (the description should include all the information about the book). You should use this keyword in member methods and constructor. Write a test application named BookTest to create an array of object for 5 elements for class Book to demonstrate the class Book's capabilities.

**Task -06:**

Create a class called Matrix containing a constructor that initializes the number of rows and number of columns of a new Matrix object. The Matrix class has the following information.

1. Number of rows of matrix
2. Number of columns of matrix
3. Elements of matrix in the form of 2D array

The Matrix class has methods for each of the following  
1. Get the number of rows  
2. Get the number of columns  
3. Set the elements of the matrix at the given position (i,j)  
4. Adding two matrices. If the matrices are not addable, “Matrices cannot be added” will be displayed  
5. Multiplying the two matrices