CSE 232

# Programming with C++

## Lecture 8

### Object Oriented Programming

*Prepared by*

**Md. Mijanur Rahman, Prof. Dr.**
Dept. of Computer Science and Engineering
**Jatiya Kabi Kazi Nazrul Islam University, Bangladesh**
Email: mijan@jkkniu.edu.bd
Web: www.mijanrahman.com

# Contents

Object Oriented Programming

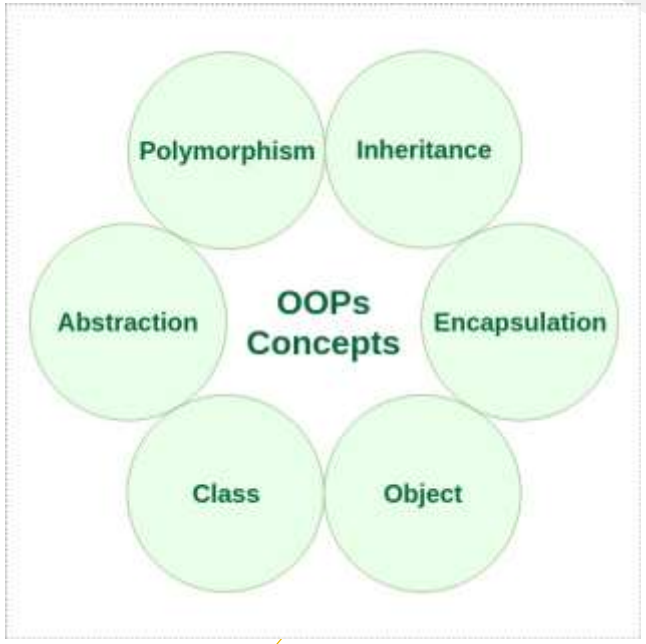
- Class
- Object
- Access Modifier
- Member Function
- Constructor
- Destructor

# Object Oriented Programming

- Object-oriented programming – As the name suggests uses objects in programming. Object-oriented programming aims to implement real-world entities like inheritance, hiding, polymorphism, etc. in programming.
- The main aim of OOP is to bind together the data and the functions that operate on them so that no other part of the code can access this data except that function.

# Object Oriented Programming

- There are some basic concepts that act as the building blocks of OOPs i.e.
  - Class
  - Object
  - Encapsulation
  - Abstraction
  - Polymorphism
  - Inheritance
  - Dynamic Binding
  - Message Passing



# C++ Classes and Objects

- In C++, classes and objects are the basic building block that leads to Object-Oriented programming in C++.
- It is a user-defined data type, which holds its own data members and member functions, which can be accessed and used by creating an instance of that class. A class is like a blueprint for an object.
- An Object is an identifiable entity with some characteristics and behavior. An Object is an instance of a Class. When a class is defined, no memory is allocated but when it is instantiated (i.e. an object is created) memory is allocated.

# C++ Classes

- A Class is a user-defined data type that has data members and member functions.
- Data members are the data variables and member functions are the functions used to manipulate these variables together these data members and member functions define the properties and behavior of the objects in a Class.
- For Example: Consider the Class of Cars. There may be many cars with different names and brands but all of them will share some common properties like all of them will have 4 wheels, Speed Limit, Mileage range, etc. So here, the Car is the class, and wheels, speed limits, and mileage are their properties.

# C++ Classes

- **Defining Class in C++**
- A class is defined in C++ using the keyword class followed by the name of the class. The following is the syntax:

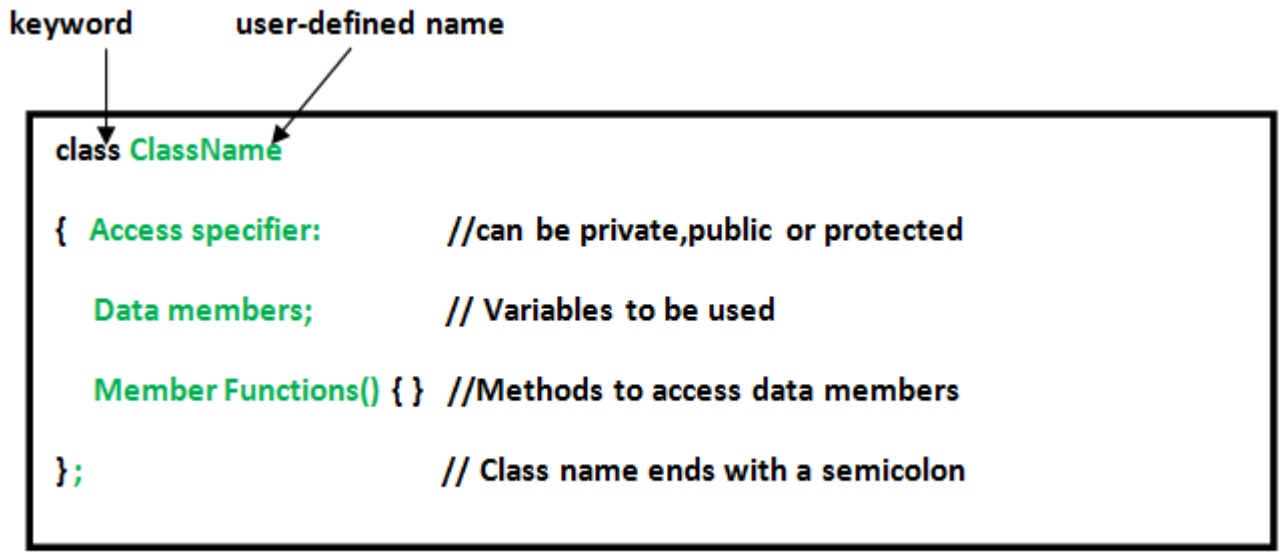
```
class ClassName {
    access_specifier:
    // Body of the class
};
```

- Here, the access specifier defines the level of access to the class's data members.

# C++ Classes

- Example

```
class ThisClass {
    public:
    int var;     // data member
    void print() {          // member method
        cout << "Hello";
     }
};
```



# C++ Objects

- When a class is defined, only the specification for the object is defined; no memory or storage is allocated. To use the data and access functions defined in the class, you need to create objects.
- **Syntax to Create an Object**
- We can create an object of the given class in the same way we declare the variables of any other inbuilt data type.

```
ClassName ObjectName;
```

- **Example**

```
MyClass obj;
```

- In the above statement, the object of MyClass with name obj is created.

# C++ Objects

- **Accessing Data Members and Member Functions**
- The data members and member functions of the class can be accessed using the dot('.') operator with the object. For example, if the name of the object is obj and you want to access the member function with the name printName() then you will have to write:

```
obj.printName()
```

# Example of Class and Object in C++

- This program shows how to define a simple class and how to create an object of it.

```
Hi, my name is Rahman and I am 30 years old.
```

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

// Define a class named 'Person'
class Person {
public:
    // Data members
    string name;
    int age;

    // Member function to introduce the person
    void introduce()
    {
        cout << "Hi, my name is " << name << " and I am "
             << age << " years old." << endl;
    }
};

int main()
{
    // Create an object of the Person class
    Person person1;
    // accessing data members
    person1.name = "Rahman";
    person1.age = 30;
    // Call the introduce member method
    person1.introduce();
    return 0;
}
```

# Access Modifiers

- In C++ classes, we can control the access to the members of the class using Access Specifiers. Also known as access modifier, they are the keywords that are specified in the class and all the members of the class under that access specifier will have particular access level.
- In C++, there are 3 access specifiers that are as follows:
  - Public: Members declared as public can be accessed from outside the class.
  - Private: Members declared as private can only be accessed within the class itself.
  - Protected: Members declared as protected can be accessed within the class and by derived classes.

# Access Modifiers

- Example of Access Specifiers:

```cpp
#include <bits/stdc++.h>
using namespace std;

class NameDisplay {
private:
    string name;
    // Access specifier
public:
    // Member Functions()
    void setName(string n) { name = n; }

    void printname() { cout << "My name is: " << name; }
};
int main()
{
    // Declare an object of class geeks
    NameDisplay obj1;
    // accessing data member
    // cannot do it like: obj1.name = "Rahman";
    obj1.setName("Rahman");
    // accessing member function
    obj1.printname();
    return 0;
}
```

# Member Function

- Member Function in C++ Classes: There are 2 ways to define a member function:
  - Inside class definition
  - Outside class definition
- In previous example, we have defined the member function inside the class, but we can also define the member function outside the class.
- To define a member function outside the class definition,
  - We have to first declare the function prototype in the class definition.
  - Then we have to use the scope resolution:: operator along with the class name and function name.

# Member Function

- Example of defining Member Function in C++ Classes:

```cpp
#include <bits/stdc++.h>
using namespace std;
class Sample {
public:
    string name;
    int id;
    // printname is not defined inside class definition
    void printname();
    // printid is defined inside class definition
    void printid() { cout << "id is: " << id; }
};
void Sample::printname()
{
    cout << "Name is: " << name;
}
int main()
{
    Sample obj1;
    obj1.name = "Rahman";
    obj1.id = 2001;
    // call printname()
    obj1.printname();
    cout << endl;
    // call printid()
    obj1.printid();
    return 0;
}
```

```
Name is: Rahman
id is: 2001
```

# Constructors and Destructors

- Constructors are special class members which are called by the compiler every time an object of that class is instantiated. Constructors have the same name as the class and may be defined inside or outside the class definition.

- Destructor is another special member function that is called by the compiler when the scope of the object ends. It deallocates all the memory previously used by the object of the class so that there will be no memory leaks. The destructor also have the same name as the class but with tilde(~) as prefix.

# Constructors in C++

- **Syntax of Constructors in C++**
- The prototype of the constructor looks like this:

```
<class-name> (){
...
...
}
```

# Constructors in C++

- Types of Constructor Definitions in C++
- 1. Defining the Constructor Within the Class

```
<class-name> (list-of-parameters) {
    // constructor definition
}
```

- 2. Defining the Constructor Outside the Class

```
<class-name>: :<class-name>(list-of-parameters) {
    // constructor definition
}
```

# Constructors in C++

- Example of defining constructor within a class:

```
Enter the RollNo:101
Enter the Name:Rahman
Enter the Fee:1000

101 Rahman  1000
```

```
#include <iostream>
using namespace std;
class student {
    int rno;
    char name[50];
    double fee;

public:
    student()
    {
        cout << "Enter the RollNo:";
        cin >> rno;
        cout << "Enter the Name:";
        cin >> name;
        cout << "Enter the Fee:";
        cin >> fee;
    }

    void display()
    {
        cout << endl << rno << "\t" << name << "\t" << fee;
    }
};

int main()
{
    student s;
    s.display();
    return 0;
}
```

# Constructors in C++

- Example of defining constructor outside a class:

```
Enter the RollNo:102
Enter the Name:Hossain
Enter the Fee:2000

102 Hossain 2000
```

```cpp
#include <iostream>
using namespace std;
class student {
    int rno;
    char name[50];
    double fee;

public:
    student();
    void display();
};

student::student()
{
    cout << "Enter the RollNo:";
    cin >> rno;
    cout << "Enter the Name:";
    cin >> name;
    cout << "Enter the Fee:";
    cin >> fee;
}

void student::display()
{
    cout << endl << rno << "\t" << name << "\t" << fee;
}

int main()
{
    student s;
    s.display();
    return 0;
}
```

# Destructors in C++

- Destructor is an instance member function that is invoked automatically whenever an object is going to be destroyed. Meaning, a destructor is the last function that is going to be called before an object is destroyed.
- The syntax for defining the destructor within the class:

```
~ <class-name>() {
    // some instructions
}
```

- Just like any other member function of the class, we can define the destructor outside the class too:

```
<class-name> {
public:
    ~<class-name>();
}
<class-name> :: ~<class-name>() {
    // some instructions
}
```

# Destructors in C++

- Example of defining destructor in C++

```
No. of Object created: 1
No. of Object created: 2
No. of Object created: 3
No. of Object created: 4
No. of Object destroyed: 4
No. of Object destroyed: 3
No. of Object destroyed: 2
No. of Object destroyed: 1
```

```cpp
#include <iostream>
using namespace std;
static int Count = 0;

class Test {
public:
    Test()
    {
        Count++;
        cout << "No. of Object created: " << Count << endl;
    }

    // User-Defined Destructor
    ~Test()
    {
        cout << "No. of Object destroyed: " << Count << endl;
        Count--;
    }
};

int main()
{
    Test t, t1, t2, t3;

    return 0;
}
```

# Lecture 8

## Object Oriented Programming

# THE END