Object Oriented Programming in C++

Segment-1

Course Code: 2301

Prepared by Sumaiya Deen Muhammad

Lecturer, CSE, IIUC

Contents

- Introduction to OOP
- C++ Console I/O
- Introduction to Classes and Objects
- Basic concept of Object Oriented Programming
- Difference between Structured Programming and Object Oriented Programming
- Difference between C/C++
- Benefits of OOP
- Characteristics of Procedure Oriented Programming
- Characteristics of Object Oriented Programming
- Applications of Object Oriented Programming

Introduction to OOP

- “Object-Oriented Programming" refers to a programming methodology based on objects, instead of just functions and procedures. These objects are organized into classes, which allow individual objects to be group together. Most modern programming languages including Java, C++, and PHP, are object-oriented languages.

Introduction to OOP (cont.)

- OOP is a powerful way to approach the task of programming.
- OOP encourages developers to decompose a problem into its constituent parts.
- Each component becomes a self-contained object that contains its own instructions and data that relate to that object.
- So, complexity is reduced and the programmer can manage larger programs.

Introduction to OOP (cont.)

- All OOP languages, including C++, share three common defining traits:
  - Encapsulation
    - Binds together code and data
  - Polymorphism
    - Allows one interface, multiple methods
  - Inheritance
    - Provides hierarchical classification
    - Permits reuse of common code and data

Introduction to C++

- C++ is an enhanced version of the C language.
- C++ adds support for OOP without sacrificing any of C's power, elegance, or flexibility.
- C++ was invented in 1979 by Bjarne Stroustrup at Bell Laboratories in Murray Hill, New Jersey, USA.
- The features of C++ are highly integrated.
- Both object-oriented and non-object-oriented programs can be developed using C++.

Two Versions of C++

- A traditional-style C++ program -

```
#include <iostream.h>

int main()
{
        /* program code */
        return 0;
}
```

Two Versions of C++ (cont.)

- A modern-style C++ program that uses the new-style headers and a namespace -

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

int main()
{
        /* program code */
        return 0;
}
```

The New C++ Headers

- The new-style headers do not specify filenames.
- They simply specify standard identifiers that might be mapped to files by the compiler, but they need not be.

  <iostream>

  <vector>

  <string>, not related with <string.h>

  <cmath>, C++ version of <math.h>

  <cstring>, C++ version of <string.h>

C++ Console I/O

- **General Format:**

```
cin>>variable;        //for input
cout<<expression;     //for output
```

C++ Console I/O (cont.)

```
#include <iostream>
using namespace std;
int main()
{
        char name[50];
        cout << "Please enter your name: ";
        cin >> name;
        cout << "Your name is: " << name << endl;
}
```

C++ Console I/O (output)

- cout << “Hello World!”;
  - printf(“Hello World!”);
- cout << iCount; //here, iCount is a variable.
  - printf(“%d”, iCount);
- cout << 100.99;
  - printf(“%f”, 100.99);
- cout << “\n”, or cout << ‘\n’, or cout << endl
  - printf(“\n”)
- In general, cout << *expression*;

C++ Console I/O (input)

- cin >> strName; /* char strName[16] */

  In C, scanf("%s", strName);

- cin >> iCount; /* int iCount */

  In C, scanf("%d", &iCount);

- cin >> fValue; /* float fValue */

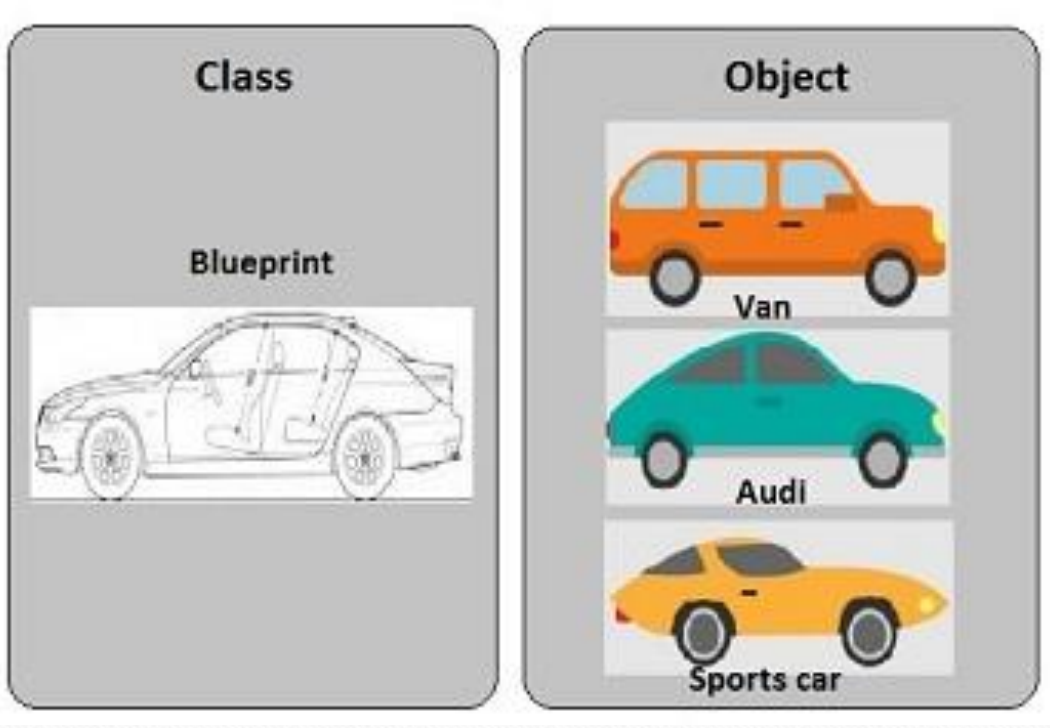
  In C, scanf("%f", &fValue);

- In general, cin >> *variable*;

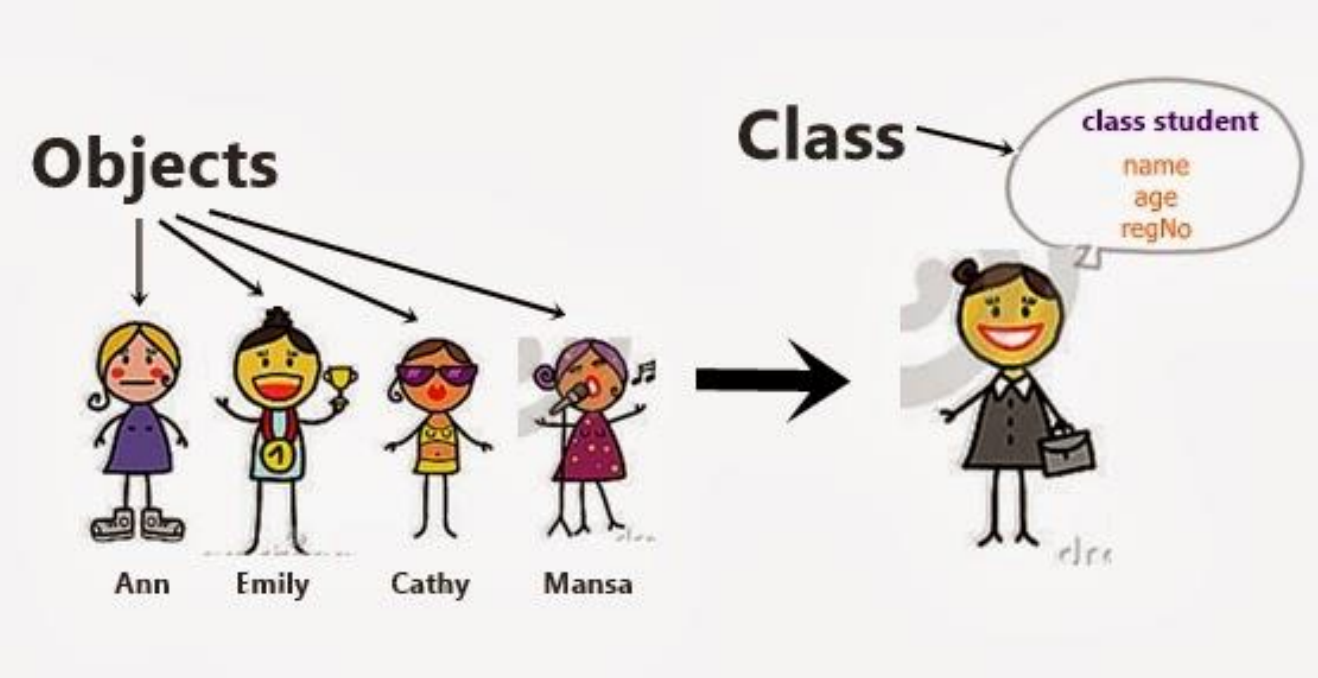
Class, Object

- When we consider a C++ program, it can be defined as a collection of objects that communicate via invoking each other's methods. Let us now briefly look into what a class, object, methods, and instant variables mean.
- **Class** − A class can be defined as a template/blueprint that describes the behaviors/states that object of its type support.
- **Object** − Objects have **states** and **behaviors**. Example: A dog has **states** - color, name, breed as well as **behaviors** - wagging, barking, eating. An object is an instance of a class.
- ✓ **Methods (behavior)** − A method is basically a behavior. A class can contain many methods. It is in methods where the logics are written, data is manipulated and all the actions are executed.
- ✓ **Instance Variables (states)** − Each object has its unique set of instance variables. An object's state is created by the values assigned to these instance variables.

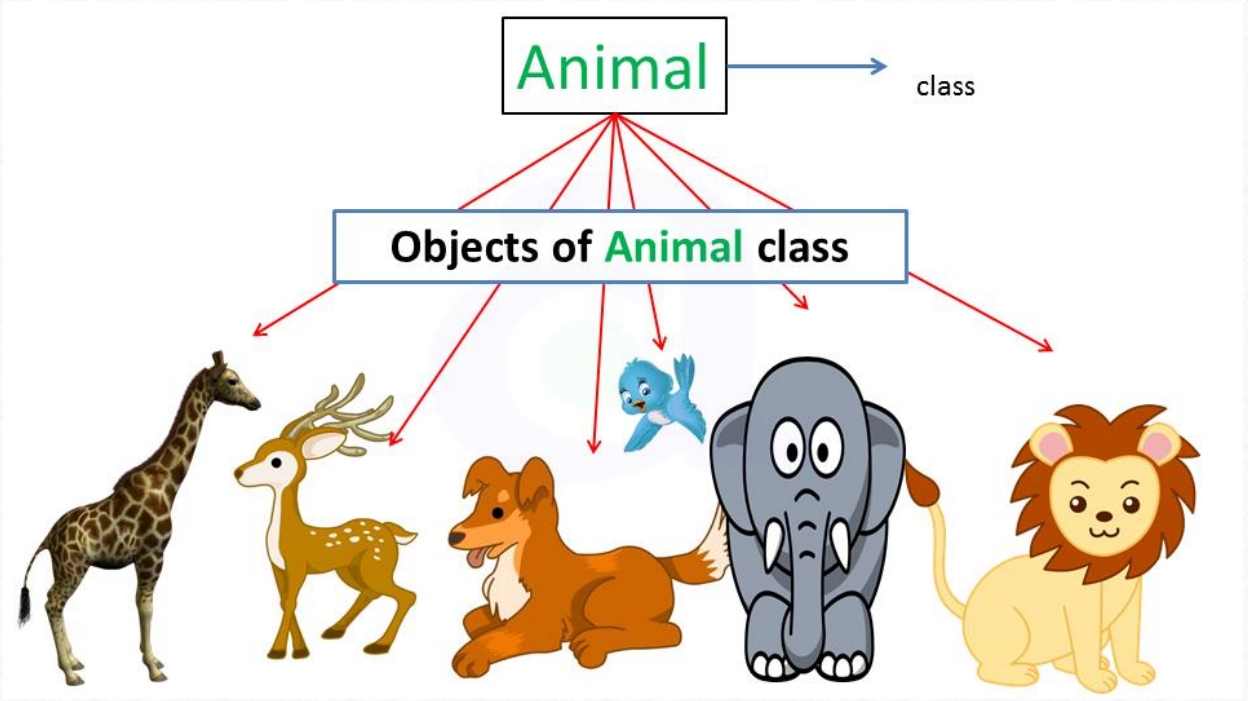
Class, Object Example



Class, Object Example (cont.)



Class, Object Example (cont.)



State, Behavior example

- **Example 1:** Car

1. State(variable):

   - model, color, brandname etc.

2. Behavior (methods):

   -drive(), setGear(), brake(), accelerate() etc.

- **Example 2:** Student

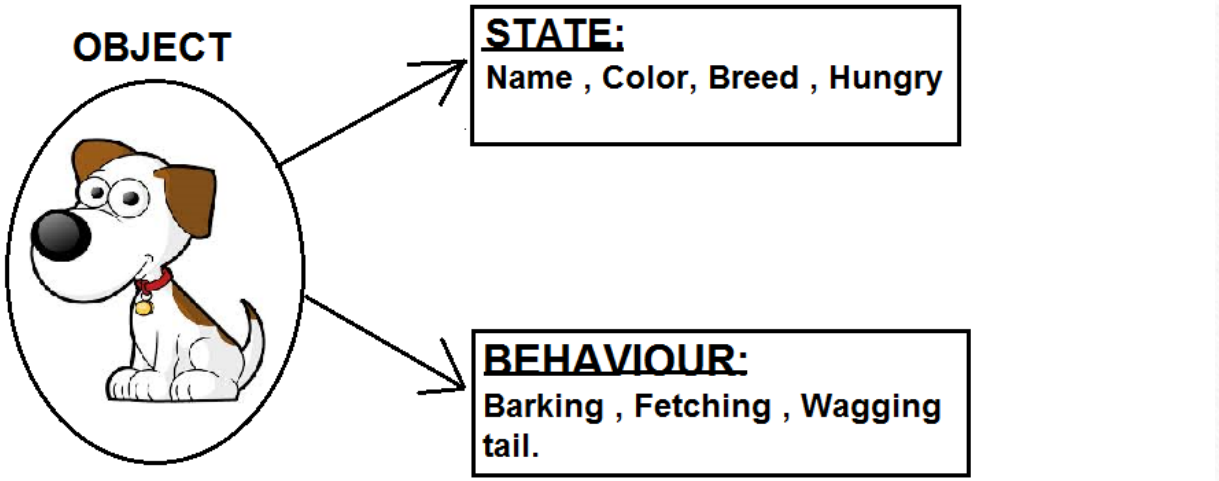
1. State:

- roll, name, dept, age, gender etc.

2. Behavior:

- setName(), getResult(), setDept() etc.

State, Behavior example(cont.)



```
#include <iostream>
 using namespace std;
class Box
{
   public:
   double length;              // Length of a box
   double breadth;            // Breadth of a box
   double height;              // Height of a box
};
int main()
{
Box Box1; // Declare Box1 of type Box
Box Box2; // Declare Box2 of type Box
double volume = 0.0; // Store the volume of a box here
// box 1 specification
Box1.height = 5.0;
Box1.length = 6.0;
Box1.breadth = 7.0;
// box 2 specification
Box2.height = 10.0;
Box2.length = 12.0;
Box2.breadth = 13.0;
// volume of box 1
volume = Box1.height * Box1.length * Box1.breadth;
cout << "Volume of Box1 : " << volume <<endl;    // volume of box 2
volume = Box2.height * Box2.length * Box2.breadth;
cout << "Volume of Box2 : " << volume <<endl;
return 0;
}
```

Structured Programming vs Object Oriented Programming

Structured Programming	Object Oriented Programming
Structured Programming is designed which focuses on **process**/ logical structure and then data required for that process.	Object Oriented Programming is designed which focuses on **data**.
Structured programming follows **top-down approach**.	Object oriented programming follows **bottom-up approach**.
Structured Programming is also known as **Modular Programming** and a subset of **procedural programming language**.	Object Oriented Programming supports **inheritance, encapsulation, abstraction**, **polymorphism**, etc.
In Structured Programming, Programs are divided into small self contained **functions**.	In Object Oriented Programming, Programs are divided into small entities called **objects**.

Structured Programming vs Object Oriented Programming

Structured Programming	Object Oriented Programming
Structured Programming is **less** secure as there is no way of **data hiding**.	Object Oriented Programming is more secure as having data hiding feature.
Structured Programming can solve **moderately** complex programs.	Object Oriented Programming can solve any **complex** programs.
Structured Programming provides **less reusability**, more function dependency.	Object Oriented Programming provides more reusability, less function **dependency**.
Less abstraction and less flexibility.	More abstraction and more **flexibility**.

C vs C++

C	C++
When compared to C++, C is a subset of C++.	C++ is a superset of C. C++ can run most of C code while C cannot run C++ code.
C supports procedural programming paradigm for code development.	C++ supports both procedural and object oriented programming paradigms; therefore C++ is also called a hybrid language.
C does not support object oriented programming; therefore it has no support for polymorphism, encapsulation, and inheritance.	Being an object oriented programming language C++ supports polymorphism, encapsulation, and inheritance.
C does not support function and operator overloading.	C++ supports both function and operator overloading.

C vs C++

C	C++
In C (because it is a procedural programming language), data and functions are separate and free entities.	In C++ (when it is used as object oriented programming language), data and functions are encapsulated together in form of an object. For creating objects class provides a blueprint of structure of the object.
In C, data are free entities and can be manipulated by outside code. This is because C does not support information hiding.	In C++, Encapsulation hides the data to ensure that data structures and operators are used as intended.
C, being a procedural programming, it is a function driven language.	While, C++, being an object oriented programming, it is an object driven language.

C vs C++ (cont.)

C	C++
C has no support for virtual and friend functions.	C++ supports virtual and friend functions.
C does not provide direct support for error handling (also called exception handling)	C++ provides support for exception handling.
Namespace feature is absent in C.	C++ uses NAMESPACE which avoid name collisions.

Benefits of OOP

1. **Improved software-development productivity**: Object-oriented programming is modular, as it provides separation of duties in object-based program development. It is also extensible, as objects can be extended to include new attributes and behaviors. Objects can also be reused within an across applications. Because of these three factors – modularity, extensibility, and reusability – object-oriented programming provides improved software-development productivity over traditional procedure-based programming techniques.
2. **Improved software maintainability:** For the reasons mentioned above, object oriented software is also easier to maintain. Since the design is modular, part of the system can be updated in case of issues without a need to make large-scale changes.

Benefits of OOP (cont.)

3. **Faster development:** Reuse enables faster development. Object-oriented programming languages come with rich libraries of objects, and code developed during projects is also reusable in future projects.
4. **Lower cost of development:** The reuse of software also lowers the cost of development. Typically, more effort is put into the object-oriented analysis and design, which lowers the overall cost of development.
5. **Higher-quality software:** Faster development of software and lower cost of development allows more time and resources to be used in the verification of the software. Although quality is dependent upon the experience of the teams, object oriented programming tends to result in higher-quality software.

Procedure Oriented Programming (POP)

- Procedure Oriented Programming contains step by step procedure to execute. Here, the problems get decomposed into small parts and then to solve each part one or more functions are used. Thus in POP approach, the problem is viewed as a sequence of *things to be done*, such as, input taking, calculating and displaying. The primary focus stays on functions which will be used to accomplish each task.
- Example: C, Fortran, COBOL etc.

Characteristics of Procedure Oriented Programming

1. Puts much importance on Things to be Done.
2. Large problems are divided into smaller programs known as functions.
3. Most of the functions share global data.
4. Data move openly around the system from function to function.
5. Functions transfer data from one form to another.
6. Employs top-down approach in program designing.
7. In the cases of large program, bringing change is difficult and time consuming.
8. Appropriate and effective techniques are unavailable to secure data of a function from others.

Characteristics of OOP

- 1. Encapsulation:

The wrapping up of data and functions into a single unit (called class) is known as *encapsulation*. Data encapsulation is the most striking feature of a class. The data is not accessible to the outside world, and only those functions which are wrapped in the class can access it. These functions provide the interface between the object's data and the program. This insulation of the data from direct access by the program is called *data hiding or information hiding*.

Characteristics of OOP (cont.)

- 2. Abstraction

Abstraction refers to the act of representing essential features without including the background details or explanations. Classes use the concept of abstraction and are defined as a list of abstract *attributes* such as size, weight and cost, and *functions* to operate on these attributes. They encapsulate all the essential properties of the objects that are to be created. The attributes are sometimes called *data members* because they hold information. The functions that operate on these data are sometimes called *methods or member functions*.

Characteristics of OOP (cont.)

3. Inheritance

Inheritance is the process by which objects of one class acquire the properties of objects of another class. It supports the concept of *hierarchical classification*. For example, the bird 'robin' is a part of the class 'flying bird' which is again a part of the class 'bird'. The principle behind this sort of division is that each derived class shares common characteristics with the class from which it is derived

Characteristics of OOP (cont.)

- 4. Polymorphism

Polymorphism is another important OOP concept. Polymorphism, a Greek term, means the ability to take more than one form. An operation may exhibit different behaviours in different instances. The behaviour depends upon the types of data used in the operation. For example, consider the operation of addition. For two numbers, the operation will generate a sum. If the operands are strings, then the operation would produce a third string by concatenation. The process of making an operator to exhibit different behaviours in different instances is known as *operator overloading*.

Application of OOP

- Real-time systems
- Simulation and modeling
- Object-oriented databases
- Hypertext, hypermedia and expertext
- AI and expert systems
- Neural networks and parallel programming
- Decision support and office automation systems
- CIM/CAM/CAD systems

Reference Books

- 1. Object Oriented Programming in C++ by Robert Lafore
- 2. Teach Yourself C++ by Herbert Schildt
- 3. Object Oriented Programming with C++ by E Balagurusamy