CSE 4102
Introduction to Object-Oriented Programming (OOP)

Course Breakdown

- **Object-Oriented concepts:** Objects, classes, abstraction, inheritance, aggregation and polymorphism.
- **Class definition:** attributes, methods – constructor, custom, accessor, mutator, instance and class member methods.
- **Object:** instantiation, method call, message passing.
- **Modelling:** Class, Class relationship diagrams.
- **Binding:** static and dynamic binding. Inheritance: Abstract classes, Interfaces.
- **Polymorphism:** method overriding and overloading.
- **Exception handling and errors, Collections, File streams.**
- Use an object-oriented programming language such

In this session, we will learn about:

- Object-Oriented Programming
- Brief history of Object-Oriented Programming Paradigm
- Classes and Objects
- How to define a class and instantiate an object using C++ programming language

What is Object-Oriented Programming?

A term used to describe a programming approach based on **objects** and **classes**.

It is a programming paradigm based on the concept of **objects**, which can contain **data**, in the form of fields (often known as attributes or properties), and **code**, in the form of procedures (often known as **methods**).

Object-oriented Programming Paradigm

- The Object Oriented programming paradigm(OOPP) allows us to organise software as a collection of **objects** that consist of both **data** and **behaviour**.
- This is in contrast to conventional functional programming practice that only loosely connects data and behaviour.
- The Object Oriented programming paradigm plays an important role in **human computer interface**.
- It has different components that takes **real world objects** and performs actions on them, making live interactions between man and the machine

Object-oriented Programming Paradigm

- Real-world objects share two characteristics:
  - They all have state and behavior.
- In this pictorial example the object ‘Dog’ has both state and behavior.
- The object Dog:
  - Stores its information in attributes e.g breed, color etc
  - Discloses its behavior through methods e.g. barking, wagging tail etc.

State	Name Color Breed Hungry
Behavior	Barking Fetching Wagging Tail

Note: Object-oriented programming takes advantage of our perception of world

Important Components of OOPP

1. OOPP paradigm describes a real-life system where interactions are among real objects.
2. It models applications as a group of related objects that interact with each other.
3. The programming entity is modeled as a class that signifies the collection of related real world objects.
4. Programming starts with the concept of real world objects and classes.
5. Application is divided into numerous packages.
6. A package is a collection of classes.
7. A class is an encapsulated group of similar real world objects.

A Brief History Of Object-oriented Paradigm

- The object-oriented paradigm took its shape from the initial concept of a new programming approach.
- The first object–oriented language was Simula (Simulation of real systems) that was developed in 1960 by researchers at the Norwegian Computing Center.
- In 1970, Alan Kay and his research group at Xerox PARK created a personal computer named Dynabook and the first pure object-oriented programming language (OOPL) - Smalltalk, for programming the Dynabook.

A Brief History Of Object-oriented Paradigm

- In the 1980s, Grady Booch published a paper titled Object Oriented Design that mainly presented a design for the programming language, Ada. In the ensuing editions, he extended his ideas to a complete object–oriented design method.
- In the 1990s, Coad incorporated behavioral ideas to object-oriented methods.
- The other significant innovations were Object Modelling Techniques (OMT) by James Rumbaugh and Object-Oriented Software Engineering (OOSE) by Ivar Jacobson.

Object-oriented Paradigm Programming

Definition

A method of implementation in which programs are organized as cooperative collections of objects, each of which represents an instance of some class, and whose classes are all members of a hierarchy of classes united via inheritance relationships(Grady Booch)

Object-oriented Paradigm Programming

- Object-oriented programming is a programming paradigm based upon **objects** (having both data and methods) that aims to incorporate the advantages of modularity and reusability.
  - Objects, which are usually instances of classes, are used to interact with one another to design applications and computer programs.

Object-oriented Paradigm Programming

Some important features of object–oriented programming are:

- Bottom–up approach in program design
- Programs organized around objects, grouped in classes
- Focus on data with methods to operate upon object's data
- Interaction between objects through functions
- Reusability of design through creation of new classes by adding features to existing classes

•Some examples of object-oriented programming languages are C++, Java, Smalltalk, Delphi, C#, Perl, Python, Ruby, and PHP.

Introduction to C++

- C++ is an object oriented programming language which was developed by Bjarne Stroustrup in 1979 at Bell Laboratories in Murray Hill, New Jersey.
- Stroustrup initially called the new language "C with Classes." but in1983 the name was changed to C++.
- C++ is a superset of C.
- Stroustrup built C++ on the foundation of C, including all of C’s features, attributes, and benefits.

Introduction to C++

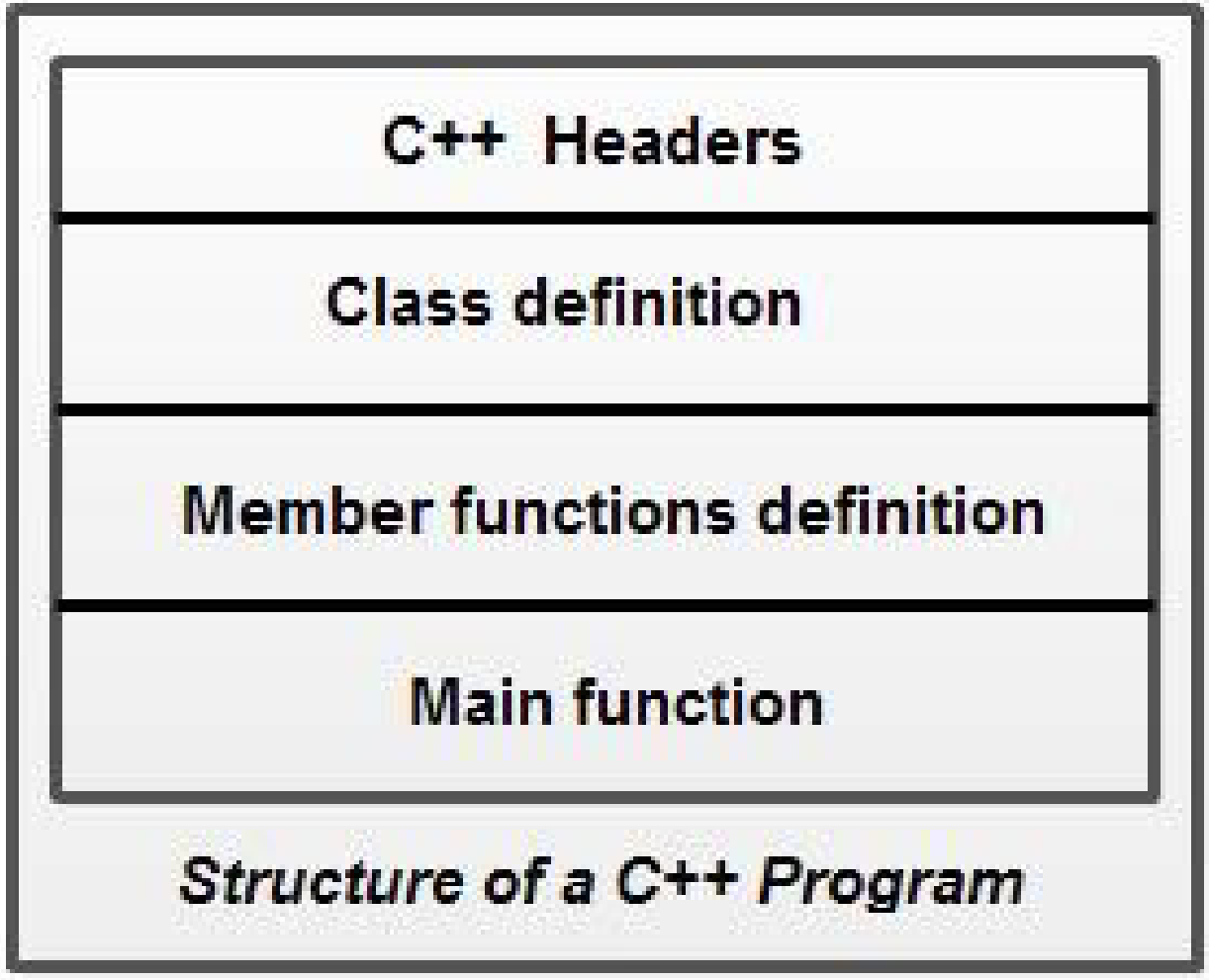
- Most of the features that Stroustrup added to C were designed to support object-oriented programming.
- The features include classes, inheritance, function overloading and operator overloading.
- C++ has many other new features as well, including an improved approach to input/output (I/O) and a new way to write comments.
- C++ is used for developing applications such as editors, databases, personal file systems, networking utilities, and communication programs.

Structure of a C++ Program

The general structure of C++ program with classes is shown as:

1. Documentation Section
2. Preprocessor Directives or Compiler Directives Section
    i. Link Section
    ii. Definition Section
3. Global Declaration Section
4. Class declaration or definition
5. Main C++ program function called main ( )

Structure of a C++ Program



Syntax and Structure of a C++ Program

Example:

```
#include <iostream>
using namespace std;
int main()
{
   cout << "Hello Kenya";
   return 0;
}
```

Structure of a C++ Program

- **Header files** are included at the beginning just like in C program.
  - iostream is a header file which provides us with input & output streams.
  - Header files contained predeclared function libraries, which can be used by users for their ease
  - For many years, C++ applied C-style headers, that is, . h extension in the headers, i.e., **#include<iostream.h>**. However, the standard C++ library introduced new-style headers that include only header name hence most modem compilers do not require any extension, though they support the older .h extension i.e., **#include<iostream>**

Structure of a C++ Program

- **Using namespace std** tells the compiler to use standard namespace.
  - A namespace permits grouping of various entities like classes, objects, functions and various C++ tokens, etc., under a single name.
  - Namespace collects identifiers used for class, object and variables.
  - NameSpace can be used by two ways in a program, either by the use of using statement at the beginning e,g,

```
using namespace std;
cout<<"Hello Kenya";
```

  - OR by using name of namespace as prefix before the identifier with scope resolution (::) operator e.g.

```
std::cout<<"Hello Kenya";
```

Structure of a C++ Program

- **main()** is a startup function that starts the execution of a c++ program.
  - All C++ statements that need to be executed are written within main ( ).
  - The compiler executes all the instructions written within the opening and closing curly braces' {}' that enclose the body of main ( ).
  - Once all the instructions in main () are executed, the control passes out of main ( ), terminating the entire program and returning a value to the operating system.
  - By default, main () in C++ returns an int value to the operating system.
  - Therefore, main () should end with the **return 0** statement. A return value zero indicates success and

Structure of a C++ Program

- **cout <<** is used to print anything on screen, same as printf in C language.
  - Cin>> and cout<< are same as scanf and printf, only difference is that you do not need to mention format specifiers like, %d for int etc, in cout & cin.

Object-Oriented Programming Concepts

Basic concepts of OOP

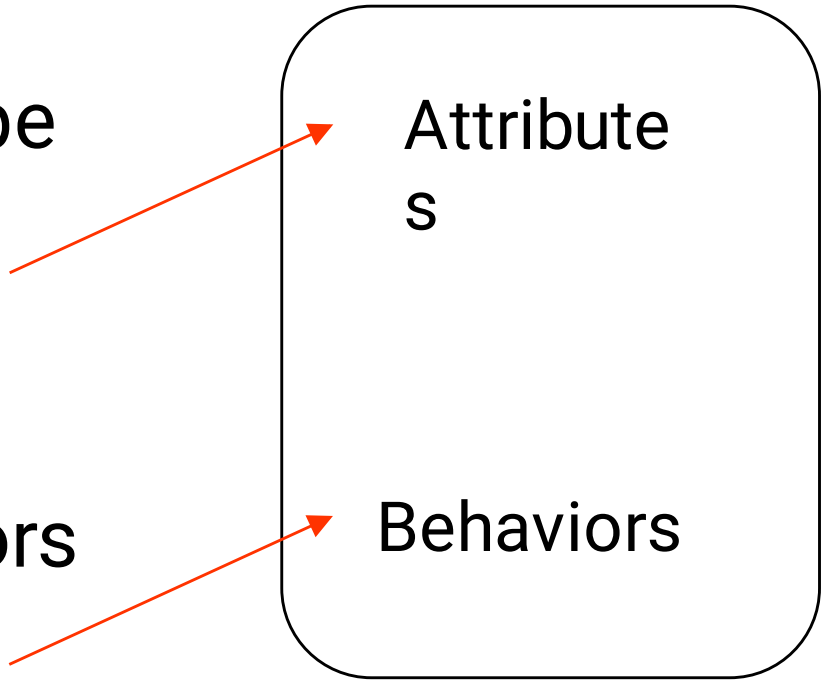
- In this course, we will learn the following basic concepts of OOP:
  - » Objects
  - » Classes
  - » Data Abstraction
  - » Data Encapsulation
  - » Inheritance
  - » Polymorphism
  - » Dynamic Binding
  - » Message Passing

Classes

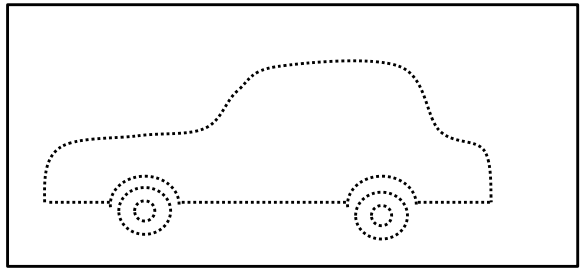
- **A class** is defined as :
  a. a group of similar objects
  b. a user defined data type
  c. a template from which objects are created
     - It can have fields, methods, constructors etc.

Class Object

- A class is a blueprint for objects of a particular type
- Defines the structure (number, types) of the attributes
- Defines available behaviors of its objects (Methods)



Class: Car



Attributes:

- String model
- Color color
- int numPassengers
- double amountOfGas

Behaviors:

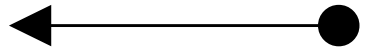
- Add/remove a passenger
- Get the tank filled
- Report when out of gas

Object: a car

Attributes:

- model = "Mustang"
- color = Color.YELLOW
- numPassengers = 0
- amountOfGas = 16.5

Behaviors:



General form of class declaration

```
class class-name {
    access-specifier:
        data and functions
    access-specifier:
        data and functions
    // ...
    access-specifier:
        data and functions
} object-list; //optional
```

Class

- The object-list is optional.
  - If present, it declares objects of the class.
- Access-specifier are either public, private or protected
  - By default, functions and data declared within a class are private to that class and may be accessed only by other members of the class.
  - The public access_specifier allows functions or data to be accessible to other parts of your program.
  - The protected access_specifier is needed only when inheritance is involved.

Class – Example

Example

```
Class myclass { // class declaration
// private members to myclass
   int a;
public: // public members to myclass
   void set_a(int num);
   int get_a( );
};
```

Class – Example

Example of a class named Employee

```
class Employee
 {
    public:
    int id;  //field or data member
    float salary; //field or data member
    String name;//field or data member
 }
```

More on Classes…

- When you define a class, you declare the data that it contains and the code that operates on that data.
- Data is contained in instance variables defined by the class known as data members, and code is contained in functions known as member functions.
- The code and data that constitute a class are called members of the class.

Class – Example

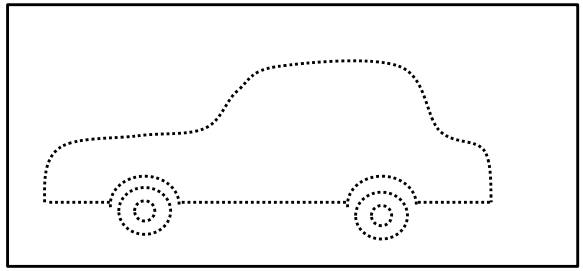
Example of a class named student

```
1.class Student {
2.  public:
3.     int id;//data member (also instance variable)
4.     string name;//data member(also instance variable)
5.     void insert(int i, string n)
6.      {
7.         id = i;
8.         name = n;
12.      }
13.     void display()
14.      {
15.         cout<<id<<"  "<<name<<endl;
16.      }
17.};
```

Objects

- **An object** is :
  a. an instance of a class
     - Object is a runtime entity which is created at runtime
     - All the members of the class can be accessed through object.
  b. a real world entity, for example, chair, car, pen, mobile, laptop etc.

Class: Car



Attributes:

- String model
- Color color
- int numPassengers
- double amountOfGas

Behaviors:

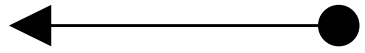
- Add/remove a passenger
- Get the tank filled
- Report when out of gas

Object: a car

Attributes:

- model = "Mustang"
- color = Color.YELLOW
- numPassengers = 0
- amountOfGas = 16.5

Behaviors:



Object – General form

- Once a Class has been declared, we can create objects of that Class by using the class Name like any other built-in type variable as shown:

```
className objectName
```

Example 1 of an object of class student

```
Student s1;  //creating an object of Student
```

Example 2

```
void main( ) {
myclass ob1, ob2; //these are object of type myclass
// ... program code
}
```

Note: A program can create and use more than one object (instance) of the same class.

Accessing Class Members

- The main() cannot contain statements that access class members directly.
- Class members can be accessed only by an object of that class.
- To access class members, use the dot (.) operator.
  - The dot operator links the name of an object with the name of a member.
- The general form of the dot operator is :

  object.member
- **Example**

  ob1.set_a(10);

Accessing Class Members

- The private members of a class cannot be accessed directly using the dot operator, but through the member functions of that class providing data hiding.
- A member function can call another member function directly, without using the dot operator.

Example 1

```
1.  #include <iostream>
2.  using namespace std;
3.  class Student {
4.    public:
5.       int id;//data member (also instance variable)
6.       string name;//
    data member(also instance variable)
7.       void insert(int i, string n)
8.       {
9.          id = i;
10.         name = n;
12.      }
13.      void display()
14.      {
15.         cout<<id<<"  "<<name<<endl;
16.      }
17. };
18. int main(void) {
19.     Student s1; //
    creating an object of Student
20.     Student s2; //
    creating an object of Student
21.     s1.insert(28516651, “John");
22.     s2.insert(28714689, “Joan");
23.     s1.display();
24.     s2.display();
25.     return 0;
26. }
```

Example 2

```
1.  //C++ program to sum of two numbers
    #include<iostream.h>
2.  #include<conio.h>
3.  class A{
4.     int a,b,c;
5.  public:
6.    void sum(){
7.        cout<<"enter two numbers";
8.        cin>>a>>b;
9.        c=a+b;
10.       cout<<"sum="<<c;
11.   }
12. };
13. int main(){
14.   A u;
15.   u.sum();
16.   getch();
17.   return(0);
18. }//end main
```

Scope Resolution operator

- Member functions can be defined within the class definition or separately using scope resolution operator (::).
- Defining a member function within the class definition declares the function **inline**, even if you do not use the inline specifier.
- Defining a member function using scope resolution operator uses following declaration:

```
return-type class-name::func-name(parameter- list) {
```

Example

```
void person :: getdata(void) {
```

Example 3

```
1.  #include<iostream> // include header file
2.  using namespace std;
3.  class person
4.  {
5.      char name[30];
6.      int age;
7.  public:
8.      void getdata(void);
9.      void display(void);
10. };
```

Example 3 cont…

```
11. void person :: getdata(void)
12. {
13.    cout << “Enter name: “;
14.    cin >> name;
15.    cout << “Enter age: “;
16.    cin >> age;
17. }
18. void person : : display(void)
19. {
20.    cout << “\nName: “ << name;
21.    cout << “\nAge: “ << age;
22. }
```

```
22. int main()
23. {
24.    person p;
25.    p.getdata();
26.    p.display();
27.    return 0;
28. } //end of example
```

Class vs. Object

1. A piece of the program's source code
2. Written by a programmer
3. Specifies the structure (the number and types) of its objects' attributes — the same for all of its objects
4. Specifies the possible behaviors of its objects

1. An entity in a running program
2. Created when the program is running (by the main method or a constructor or another method)
3. Holds specific values of attributes; these values can change while the program is running
4. Behaves appropriately when called upon

Assignment

1. Briefly discuss the pros and cons of the following programming techniques:
   a) Unstructured Programming (5 Marks)
   b) Procedural Programming (5 Marks)
   c) Modular & Structural Programming (5 Marks)
   d) Abstract Data Type (5 Marks)
   e) Object-oriented programming (5 Marks)
2. Compare and contrast procedural programming and object-oriented programming (5 Marks).

References

- https://www.tutorialspoint.com/cplusplus/index.htm
- https://www.javatpoint.com/cpp-object-and-class