Course Name: PROGRAMMING IN C++
Course Code: 21CS382

Module-1:
Introduction to Object Oriented Programming: Computer programming background- C++ overview-First C++ Program -Basic C++ syntax, Object Oriented Programming: What is an object, Classes, methods and messages, abstraction and encapsulation, inheritance, abstract classes, polymorphism.
Textbook 1: Chapter 1(1.1 to 1.8)

A Review of structures:

Structure: is the key concept of Procedure Oriented Programming (POP)

is Derived data types and Collection of one or more related variables of different data types, grouped under a single name

the keyword struct is used to define a

Structure

```
syntax:     struct  tag_name
            {
              type   member:
            };
```

```
Ex:    struct  student
         {
       int  rollno;
       float  avg ;

       char  grade ;
            };
```

Declaration of Structure variable
struct student s ;

Accessing members of Structure
s.rollno;
s. avg ;
s. grade

Disadvantages of Structure:

members of Structure are public by default.

structs are usually used for smaller amounts of data.

Structure does not support the inheritance.

Overview of C++

- ✓ C++ extension was first invented by "Bjarne Stroustrup" in 1979.
- ✓ He initially called the new language " C with Classes".
- ✓ However in 1983 the name was changed to C++.
- ✓ c++ is an extension of the C language, in that most C programs are also c++programs.
- ✓ C++, as an opposed to C, supports "Object-Oriented Programming".

Object Oriented Programming System (OOPS)

- In OOPS we try to model real-world objects.
- Most real world objects have internal parts (Data Members) and interfaces (Member Functions) that enables us to operate them.

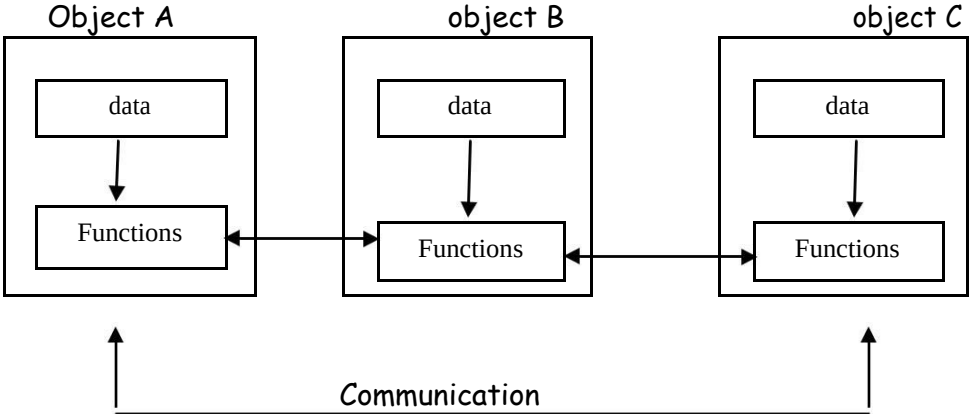
Object:

- ✓ Everything in the world is an object.
- ✓ An object is a collection of variables that hold the data and functions that operate on the data.
- ✓ The variables that hold data are called Data Members.
- ✓ The functions that operate on the data are called Member Functions.

The two parts of an object:

- ✓ Object = Data + Methods (Functions)
- ✓ In object oriented programming the focus is on creating the objects to accomplish a task and not creating the procedures (Functions).
- ✓ In OOPs the data is tied more closely to the functions and does not allow the data to flow freely around the entire program making the data more secure.
- ✓ Data is hidden and cannot be easily accessed by external functions.
- ✓ Compliers implementing OOP does not allow unauthorized functions to access the data thus enabling data security.

- ✓ Only the associated functions can operate on the data and there is no change of bugs creeping into program.
- ✓ The main advantage of OOP is its capability to model real world problems.
- ✓ It follows Bottom Up approach in program design.



- ✓ Identifying objects and assigning responsibilities to these objects.
- ✓ Objects communicate to other objects by sending messages.
- ✓ Messages are received by the methods (functions) of an object.

Basic concepts (features) of Object-Oriented Programming

1. Objects
2. Classes
3. Data abstraction
4. Data encapsulation
5. Inheritance
6. Polymorphism
7. Binding
8. Message passing

- **Objects and Classes:**
  - Classes are user defined data types on which objects are created.
  - Objects with similar properties and methods are grouped together to form class.
  - So class is a collection of objects.
  - Object is an instance of a class.
- **Data abstraction**
  - Abstraction refers to the act of representing essential features without including the background details or explanation.
  - **Ex:** Let's take one real life example of a TV, which you can turn on and off, change the channel, adjust the volume, and add external components such as speakers, VCRs, and DVD players, BUT you do not know its internal details, that is, you do not know how it receives signals over the air or through a cable, how it translates them, and finally displays them on the screen.
  - **Ex**: #include <iostream>

```
int main( )
{
  cout << "Hello C++" <<endl;
  return 0;
}
```

  - Here, you don't need to understand how cout displays the text on the user's screen. You need to only know the public interface and the underlying implementation of cout is free to change.
- **Data encapsulation**
  - Information hiding
  - Wrapping (combining) of data and functions into a single unit (class) is known as data encapsulation.
  - Data is not accessible to the outside world, only those functions which are wrapped in the class can access it.

- **Inheritance**
  - Acquiring qualities.
  - Process of deriving a new class from an existing class.
  - Existing class is known as base, parent or super class.
  - The new class that is formed is called derived class, child or sub class.
  - Derived class has all the features of the base class plus it has some extra features also.
  - Writing reusable code.
  - Objects can inherit characteristics from other objects.
- **Polymorphism**
  - The dictionary meaning of polymorphism is "having multiple forms".
  - Ability to take more than one form.
  - A single name can have multiple meanings depending on its context.
  - It includes function overloading, operator overloading.
- **Binding**
  - Binding means connecting the function call to the function code to be executed in response to the call.
  - Static binding means that the code associated with the function call is linked at compile time. Also known as early binding or compile time polymorphism.
  - Dynamic binding means that the code associated with the function call is linked at runtime. Also known as late binding or runtime polymorphism.
- **Message passing**

  Objects communicate with one another by sending and receiving information.

The process of programming in an OOP involves the following basic steps:

1. Creating classes that define objects and behavior.
2. Creating objects from class definitions.
3. Establishing communications among objects.

Advantages of OOPS

- ➔ Data security
- ➔ Reusability of existing code
- ➔ Creating new data types
- ➔ Abstraction
- ➔ Less development time
- ➔ Reduce complexity
- ➔ Better productivity

Benefits of OOP

- ➔ Reusability
- ➔ Saving of development time and higher productivity
- ➔ Data hiding
- ➔ Multiple objects feature
- ➔ Easy to partition the work in a project based on objects.
- ➔ Upgrade from small to large systems
- ➔ Message passing technique for interface.
- ➔ Software complexity can be easily managed.

Applications of OOP

- ➔ Real time systems
- ➔ Simulation and modeling
- ➔ Object oriented databases

- ➔ Hypertext, hypermedia
- ➔ AI (Artificial Intelligence)
- ➔ Neural networks and parallel programming
- ➔ Decision support and office automation systems
- ➔ CIM/CAD/CAED system

Difference between POP(Procedure Oriented Programming) and OOP(Object Oriented Programming)

Sl.No	POP	OOP
1.	Emphasis is on procedures (functions)	Emphasis is on data
2.	Programming task is divided into a collection of data structures and functions.	Programming task is divided into objects (consisting of data variables and associated member functions)
3.	Procedures are being separated from data being manipulated	Procedures are not separated from data, instead, procedures and data are combined together.
4.	A piece of code uses the data to perform the specific task	The data uses the piece of code to perform the specific task
5.	Data is moved freely from one function to another function using parameters.	Data is hidden and can be accessed only by member functions not by external function.
6.	Data is not secure	Data is secure
7.	Top-Down approach is used in the program design	Bottom-Up approach is used in program design
8.	Debugging is the difficult as the code size increases	Debugging is easier even if the code size is more

Comparison of C with C++

Sl.No	C	C++
1.	It is procedure oriented language	It is object-oriented language
2.	Emphasis is on writing the functions which performs some specific tasks.	Emphasis is on data which uses functions to achieve the task.
3.	The data and functions are separate	The data and functions are combined
4.	Does not support polymorphism, inheritance etc.	Supports polymorphism, inheritance etc.
5.	They run faster	They run slower when compared to equivalent C program
6.	Type checking is not so strong	Type checking is very strong
7.	Millions of lines of code management is very difficult	Millions of lines of code can be managed very easily
8.	Function definition and declarations are not allowed within structure definitions	Function definitions and declarations are allowed within structure definitions.

Console Output/input in C++

Cin: used for keyboard input.

Cout: used for screen output.

Since Cin and Cout are C++ objects, they are somewhat "Intelligent".

- ✓ They do not require the usual format strings and conversion specifications.
- ✓ They do automatically know what data types are involved.
- ✓ They do not need the address operator and ,
- ✓ They do require the use of the stream extraction (>>) and insertion (<<) operators.

Extraction operator (>>):

- ✓ To get input from the keyboard we use the extraction operator and the object Cin.

- ✓ Syntax: Cin>> variable;
- ✓ No need for "&" infront of the variable.
- ✓ The compiler figures out the type of the variable and reads in the appropriate type.
    - Example:

```
 #include<iostream.h>
Void main( )
{
        int x;
        float y;
        cin>> x;
        cin>>y;
}
```

Insertion operator (<<):

- ➢ To send output to the screen we use the insertion operator on the object Cout.
- ➢ Syntax: Cout<<variable;
- ➢ Compiler figures out the type of the object and prints it out appropriately.

Example:

```
#include<iostream.h>
void main( )
{
        cout<<5;
        cout<<4.1;
        cout<< "string";
        cout<< '\n';
}
```

Programs

Example using Cin and Cout

```
#include<iostream.h>
void main( )
{
        int a,b;
        float k;
        char name[30];
        cout<< "Enter your name \n";
        cin>>name;
```

```
        cout<< "Enter two Integers and a Float \n";
        cin>>a>>b>>k;
        cout<< "Thank You," <<name<<",you entered\n";
        cout<<a<<","<<b<<",and"<<k<<'/n';
}
```

Output:

```
        Enter your name
        Mahesh
        Enter two integers and a Float
        10
        20
        30.5
                Thank you Mahesh, you entered
                10, 20 and 30.5
```

C++ program to find out the square of a number

```
#include<iostream>
int main( )
{
        int i;
        cout<< "this is output\n";
        cout<< "Enter a number";
        cin>>i;
        cout<<i<< "Square is" << i*i<<"\n";
        return 0;
}
```

Output:

```
This is output
        Enter a number  5
         5 square is 25
```