**C++ - OOP**

# What is OOP?

Object-oriented programming (OOP) is a programming paradigm based on the concept of "objects", which may contain data, in the form of fields, often known as attributes; and code, in the form of procedures, often known as methods.

In OOP, computer programs are designed by making them out of objects that interact with one another. There is significant diversity of OOP languages, but the most popular ones are class-based, meaning that objects are instances of classes, which typically also determine their type.

Significant object-oriented languages include Java, C++, C#, Python, PHP, Ruby, Perl, Object Pascal, Objective-C, Dart, Swift, Scala, Common Lisp, and Smalltalk.

***Object-oriented programming is an approach that provides a way of modularizing programs by creating partitioned memory area for both data and functions that can be used as templates for creating copies of such modules on demand.***

Features of object-oriented programming are

1. Emphasis is on data rather than procedure
2. Programs are divided into what are known as objects
3. Data structures are designed such that they characterize the objects
4. Functions that operate on the data of an object are tied together in the data structure
5. Data is hidden and cannot be accessed by external function
6. Objects may communicate with each other through functions
7. New data and functions can be easily added whenever necessary
8. Follows bottom-up approach in program design

OOP concepts include:

1. Objects
2. Classes
3. Data abstraction and encapsulation
4. Inheritance
5. Polymorphism
6. Dynamic binding
7. Message Passing

# Objects

1. Objects are basic run-time entities in an object oriented system
2. Objects are instances of a class these are defined user defined data types
3. Object take up space in memory and have an associated address like a record in Pascal or structure or union in C
4. When a program is executed the objects interact by sending messages to one another
5. Each object contains data and code to manipulate the data
6. Objects can interact without having to know details of each other’s data or code, it is sufficient to know the type of message accepted and type of response returned by the objects

class person {

char name[20];

int id;

public:

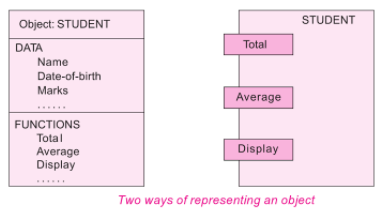
void getdetails(){}

};

int main() {

person p1; //p1 is a object

}



# Classes

1. Class is a blueprint of data and functions or methods
2. Class does not take any space
3. Class is a user defined data type like structures and unions in C
4. By default class variables are private but in case of structure it is public
5. Classes are user defined data type and behave like the built-in types of a programming language

Syntax for class:

class class\_name {

private:

//data members and member functions declarations

public:

//data members and member functions declarations

protected:

//data members and member functions declarations

};

In above example person is a class

# Data abstraction and encapsulation

**Encapsulation**: Wrapping up (combing) of data and functions into a single unit is known as encapsulation.

The data is not accessible to the outside world and only those functions which are wrapping in the class can access it. This insulation of the data from direct access by the program is called data hiding or information hiding.

**Abstraction**: Data abstraction refers to, providing only needed information to the outside world and hiding implementation details.

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.

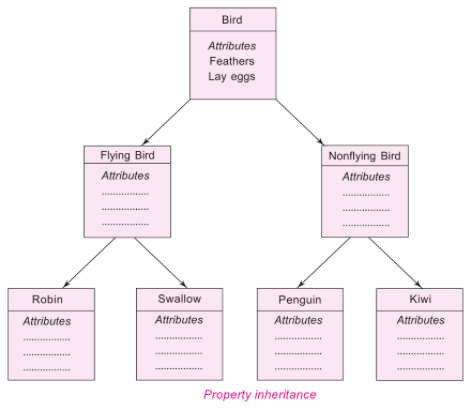
For example, consider a class Complex with public functions as getReal() and getImag(). We may implement the class as an array of size 2 or as two variables. The advantage of abstractions is, we can change implementation at any point, users of Complex class wont’t be affected as out method interface remains same. Had our implementation be public, we would not have been able to change it.

Note:

**Abstract Data Types (ADT)**: Since the classes use the concept of data abstraction, they are known as Abstract Data Types (ADT).

# Inheritance

1. Inheritance is the process by which objects of one class acquire the properties of objects of another class.
2. It supports the concept of hierarchical classification.
3. Inheritance provides reusability.
4. This means that we can add additional features to an existing class without modifying it.



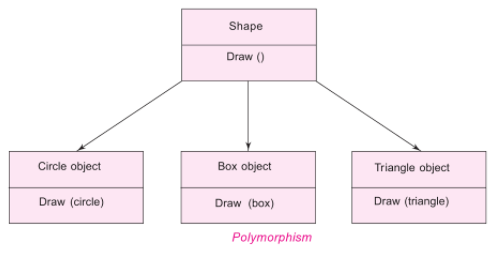
# Polymorphism

Polymorphism means 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.

C++ supports operator overloading and function overloading.

Operator overloading is the process of making an operator to exhibit different behaviours in different instances is known as operator overloading.

Function overloading is using a single function name to perform different types of tasks. Polymorphism is extensively used in implementing inheritance.



# Dynamic binding

1. In dynamic binding, the code to be executed in response to function call is decided at runtime.
2. It is associated with polymorphism and inheritance.
3. C++ has virtual functions to support this.

# Message Passing

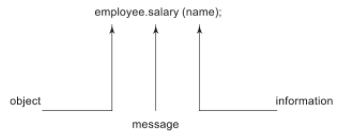
Objects communicate with one another by sending and receiving information to each other.

A message for an object is a request for execution of a procedure and therefore will invoke a function in the receiving object that generates the desired results.

Message passing involves specifying the name of the object, the name of the function and the information to be sent.

Objects have a life cycle. They can be created and destroyed.

Communication with an object is feasible as long as it is alive.



# Benefits of OOP

1. Through inheritance, we can eliminate redundant code extend the use of existing Classes.
2. We can build programs from the standard working modules that communicate with one another, rather than having to start writing the code from scratch. This leads to saving of development time and higher productivity.
3. The principle of data hiding helps the programmer to build secure program that cannot be invaded by code in other parts of a programs.
4. It is possible to have multiple instances of an object to co-exist without any interference.
5. It is possible to map object in the problem domain to those in the program.
6. It is easy to partition the work in a project based on objects.
7. The data-centred design approach enables us to capture more detail of a model can implemental form.
8. Object-oriented system can be easily upgraded from small to large system.
9. Message passing techniques for communication between objects makes to interface descriptions with external systems much simpler.
10. Software complexity can be easily managed.

# References

<https://www.geeksforgeeks.org/object-oriented-programming-in-cpp/>

<https://www.learncpp.com/cpp-tutorial/welcome-to-object-oriented-programming/>

# END OF FILE