CPP

Introduction to Object Oriented Programming

OOP is a design philosophy. It stands for Object Oriented Programming.

C++ started with extension of C by Bjarne Stroustrup at Bell Lab

Evolution of programming languages

In initial days of computers the instructions were passed to the machines as 'machine codes'

Strings of 0s and 1s and very complex to read/write by human

readable for humans and easy to manage larger tasks

FORTRAN was one of the very early languages that people

accepted

Then came the 'structured' programming languages

C and Pascal were the flag bearers

What is Object Oriented Programming actually

In the most general sense, a program can be organized in one of two

ways:

Around its code (what is happening)

Around its data (who is being affected)

A program written in a structured language such as C is defined by

its functions, any of which may operate on any type of data used by

the program

In an object-oriented language, you define the data and the routines

that are permitted to act on that data

A data type defines precisely what sort of operations can be applied on it

In the most general sense, a program can be organized in one of two ways:

Around its code (what is happening)

Around its data (who is being affected)

A program written in a structured language such as C is defined by its functions, any of which may operate on any type of data used by the program

In an object-oriented language, you define the data and the routines that are permitted to act on that data

A data type defines precisely what sort of operations can be applied on it

What is an Object?

Objects can be either physical or logical

- Physical objects: pen, paper, laptops, table etc.
- Logical objects: Employee ID, bank account no, social media ids etc.

Attributes and operations

Each object can have some attributes and operations

- Object: Person
- Attributes: Name, age, weight
- Operations: Eat, sleep, work, walk
- Object: Motorcycle
- Attribute: Company, model, price
- Operations: Drive, stop, repair
- Object: Bank account

 Attributes: Account number, balance, name

 Operations: Fund transfer, credit money, debit money

Class and Objects

Classes are the blueprints for objects

 Objects are an instance of the class

Class: Animal kingdom, Objects: Dog, cat, human

The blueprint: living organism, eats, walk, sleeps, has eyes/ears/legs

Instance: 2/4 legs, living conditions, eats differently, can walk/run slower or faster

Class: Vehicles, Objects: Motorcycles, cars, trucks

The blueprint: Uses fuel, moves around

Instances: 2/3/4/8 wheels, electric/diesel/petrol, capacity, speed

Procedural Vs. Object Oriented Programming

Principles of OOP

There are mainly four OOP Principles

 Abstraction

 Encapsulation

 Inheritance

 Polymorphism

Abstraction

It refers to the act of representing essential features without including the background details or explanations.

 It provides you a generalized view of your classes or object by providing relevant information.

 It is the process of hiding the working style of an object, and showing the information of an object in understandable manner.

Encapsulation

The wrapping up of data and functions into a single unit

 Data hiding: insulation of the data from direct access by the program

 It is the process of enclosing one or more details from outside world through access right (public, private, protected)

Abstraction Vs Encapsulation

 Abstraction says what details to be made visible & Encapsulation provides the level of access right to that visible details.

 Ex: When we switch on the Bluetooth the device is able to connect another mobile but not able to access the other mobile features like dialling a number, accessing inbox etc.

 The abstraction is given the details of the device name, frequency etc., the encapsulation makes sure we cannot access anything that was not allowed

Inheritance

 In a real world, a child inherits parent's properties

 Similarly a class can derive properties and characteristics from other class

 Super class: think of it as the parent class

 Sub class: think of it as the child class

note: there are no actual parent-child relations like in process threads

 Inheritance has 'is-a' relationship between the classes

 A car is a vehicle

 An software developer is an employee of the organization

Polymorphism

 It means one 'entity' having different 'roles'

 Example1: a table can be used as work desk, a dining table, a musical instrument etc.

 Example2: A women can be a wife, a daughter, a mother, an employee, a boss etc. it depends on the environment and context

 Programming version: the symbol '+' can be used to 'add' two integer values, it can be used to 'concatenate' two strings or 'add' two objects or practically do anything depending on the implementation

 Two types of polymorphism:

 Compile time: operator/function overloading

 Run time: virtual functions

CALCULATOR PROGRAM

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

```
class Calculator{
public:
int input1;
int input2;
void setInput(int a, int b){
input1 = a;
input2 = b;
}
int add(){
return input1+input2;
}

};
int main(){
Calculator obj1;
obj1.setInput(10,2);
cout << "The inputs: " << obj1.input1 << " " << obj1.input2 <<endl;
cout << "The sum of inputs is: " << obj1.add();
return 0;
```

Access specifiers

All the variables and member function in a class will have one of following three properties: Public, Private or Protected.

Private Members

• Private members of the class can be accessed within the class and from member functions of the class.

• They cannot be accessed outside the class or from other programs, not even from inherited class.

• If you try to access private data from outside of the class, compiler throws error.

• This feature in OOP is known as Data hiding / Encapsulation.

• If any other access modifier is not specified then member default acts as Private member.

Public Members

• The public keyword makes data and functions public.

• Public members of the class are accessible by any program from anywhere.

• Class members that allow manipulating or accessing the class data are made public.

Passing Objects as Function Arguments

 We can pass an object in the function argument just like any other data type

 We need to define the operations inside the function carefully

 In Calculator program we can modify the add() to take two objects as arguments

 We need to make some changes accordingly

Returning Objects from a Function

 We can return an object variable from any function as output, similar to any other

type of variable

 In Calculator program that we just modified, add() takes two objects as input and returns an integer

 Instead of return an integer variable we can return an object variable
 In this example we will take two objects, each object has two integer variables
 We will add input1 of both objects and input2 of both objects and create another object with these values
 We will return this newly created object from the add()

 We need to make some changes accordingly

Member function calling other member functions

 We can call one member function from other member function

 This is also called as nested function calls

 In your Calculator program, you can call subtraction() from the multiplication() function

 Make these changes and see if you get any error

Memory allocation of objects

The member functions are created and placed in the memory space only once at the time they are defined as part of a class specification.

No separate space is allocated for member functions when the objects

are created.

Only space for member variable is allocated separately for each object because, the member variables will hold different data values for different objects.

Friend Function

• In C++ a Friend Function that is a "friend" of a given class is allowed access to private and protected data in that class.

• It is declared using friend keyword

Friend function can be declared either in public or private part of the class.

• It is not a member of the class so it cannot be called using the object.

• Usually, it has the objects as arguments.

Use of friend function

It is possible to grant a nonmember function access to the private members of a class by using a friend function.

It can be used to overload binary operators.

Static Data members

A static data member is useful, when all objects of the same class must share a common information.

Static Data Members

Data members of the class which are shared by all objects are known as static data members.

Only one copy of a static variable is maintained by the class and it is common for all

objects.

Static members are declared inside the class and defined outside the class.

It is initialized to zero when the first object of its class is created.

you cannot initialize a static member variable inside the class declaration.

It is visible only within the class but its lifetime is the entire program.

Static members are generally used to maintain values common to the entire class.

Static Member Functions

Static member functions can access only static members of the class.

Static member functions can be invoked using class name, not object.

There cannot be static and non-static version of the same function.

They cannot be virtual.

They cannot be declared as constant or volatile.

A static member function does not have this pointer.

What is constructor ?

A constructor is a block of code which is,

similar to member function

has same name as class name

called automatically when object of class created

A constructor is used to initialize the objects of class as soon as the object is created.

Constructor should be declared in public section
because private constructor cannot be invoked outside
the class so they are useless.
Constructors do not have return types and they cannot
return values, not even void.

Types of Constructors

Default constructor
Parameterized constructor
Copy constructor

Default Constructor

Default constructor is the one which invokes by default when object of the class is
created.
It is generally used to initialize the default value of the data members.
It is also called no argument constructor.

Parameterized Constructor

Constructors that can take arguments are called parameterized
constructors.
Sometimes it is necessary to initialize the various data elements of
different objects with different values when they are created.
We can achieve this objective by passing arguments to the
constructor function when the objects are created.

Copy Constructor

A copy constructor is used to declare and initialize an object from another object using an object as argument.

A copy constructor is used to initialize an object
from another object
using an object as argument.

A Parameterized constructor which accepts a reference to its own class as a parameter is called copy constructor.

Destructor

Destructor is used to destroy the objects
that have been created by a constructor.
The syntax for destructor is same as that
for the constructor,
the class name is used for the name of
destructor,
with a tilde (~) sign as prefix to it.

Inline Functions

Every time a function is called it takes a lot of extra time to
execute series of instructions

Default Arguments

while invoking a function If the argument/s are not passed then, the default values are used.

We must add default arguments from right to left.

We cannot provide a default value to a particular argument in the middle of an argument list.

Function overloading

• Function overloading does not depend on return type.

C++ provides function overloading which allows to use multiple functions sharing the same name .

Function overloading is also known as Function Polymorphism in OOP.

It is the practice of declaring the same function with different signatures.

Inheritance

Inheritance is the process, by which class can acquire(reuse) the properties and methods of another class.

The mechanism of deriving a new class from an old class is called inheritance.

The new class is called derived class and old class is called base class.

The derived class may have all the features of the base class and the programmer can add new features to the derived class.

Types of Inheritance

1. Single Inheritance
2. Multilevel Inheritance
3. Multiple Inheritance

4. Hierarchical Inheritance

5. Hybrid Inheritance

Function Overriding / Method Overriding

• If base class and derived class have member functions with same name and arguments then method is said to be overridden and it is called function overriding or method overriding in C++.

Virtual base class

Virtual base class is used to prevent the duplication/ambiguity.

In hybrid inheritance child class has two direct parents which themselves have a common base class.

So, the child class inherits the grandparent via two separate paths. it is also called as indirect parent class.

All the public and protected member of grandparent are inherited twice into child.

We can stop this duplication by making base class virtual.

Pointers and objects

• Just like pointers to normal variables and functions, we can have pointers to class members (variables and methods).

This pointer

Within member function, the members can be accessed directly, without any object or class qualification.

But implicitly members are being accessed

using this pointer

Pointer to derived class

We can use pointers not only to the base objects but also to the objects of derived classes.

Virtual Function

A virtual function is a member function that is declared within a base class and redefined by a derived class.

To create a virtual function, precede the function's declaration in the base class with the keyword virtual.

Pure Virtual Function

• A pure virtual function is virtual function that has no definition within the base class.

Abstract Class

A class that contains at least one pure virtual function is called abstract class.

You can not create objects of an abstract class, you can create pointers and references to an abstract class.

C++ Exceptions:

When executing C++ code, different errors can occur: coding errors made by the programmer, errors due to wrong input, or other unforeseeable things.

When an error occurs, C++ will normally stop and generate an error message. The technical term for this is: C++ will throw an exception (throw an error).

Types of file access

• Sequential access: With this type of file access one must read the data in order, much like with a tape whether the data is really stored on tape or not

• Random access: This type of file access lets you jump to any location in the file, then to any other, etc. all in a reasonable amount of time.

Templates in c++

It allows functions or class to work on more than one data type at once, without writing different codes for different data types.
Templates are often used in larger programs for the purpose of code reusability and flexibility of program

Class template

Similar to the function templates, we can also create class templates

A class template can represent various similar
classes operating on different data types

Graphs

A graph is a non-linear data structure. A graph can be defined as a
collection of Nodes which are also called “vertices” and “edges” that
connect two or more vertices

A graph can also be seen as a cyclic tree where vertices do not have
a parent-child relationship but maintain a complex relationship
among them.

Graphs: types

Direction
Directed
Un-directed

Graphs: types

Weight
Weighted
Un-weighted

What is design patterns

• Software design patterns are abstractions that help structure system
Designs

cmake

• The make utility and Makefiles provide a build system that can be used to manage the compilation and re-compilation of programs that are written in any programming language

• CMake is a cross-platform Makefile generator! Simply put, CMake automatically generates the Makefiles for your project.