



Unit – 01 : Principles of Object Oriented Programming

- Procedure Oriented Programming (POP) Verses Object Oriented Programming (OOP).
- Basic Concepts of Object-Oriented Programming,
- Object Oriented languages,
- Applications of OOP.
- C verses C++ ,
- Structure of C++ Program,
- Simple C++ Program.
 - Tokens,
 - Keywords,
 - Variables, Constants,
 - Basic Data Types, User Defined Data Types,
 - Operators, Expressions.
- Control Structures: Decision making statements & Loops.
- Scope resolution Operator, Memory Management Operators.
- Arrays, Strings & Structures in C++.
- Type Casting,

Questions to be discussed :

1. Explain the difference between POP and OOP.
2. Explain the various features of OOPs.
3. Differentiate between C and C++ .
4. Describe the structure of C++ program.
5. What is token in C++ ? Explain in brief.
6. Explain data types in C++ with example.
7. Explain various object of operators available in C++ .
8. What is an array? What are the advantage of it?
9. Explain the concept of if-else and switch statements.
10. Discuss about scope resolution operator and memory management operator in c++.
11. Define structure and union.
12. Write short notes on :
 - a. Keyword
 - b. Identifiers
 - c. Enumerated variables
 - d. Type conversions & type casting.

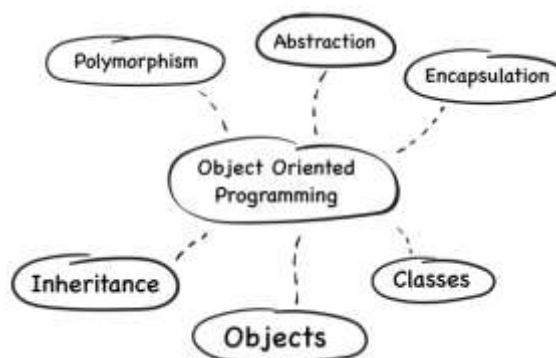
Explain the difference between POP & OOP.

POP	OOP
POP Stands for Procedural Oriented Programming.	OOP Stands for Object Oriented Programming.
Entire program is divided into functions.	Entire program is divided into objects.
It follows Top-down approach.	It follows Bottom-up approach.
No access specifiers are supported.	Access specifiers are supported.
Here, no concept of overloading.	It overloads functions, constructors & operators.
Inheritance is not supported.	Inheritance is supported.
No data hiding is present, so data is insecure	Encapsulation is used to hide in data.
C, VB, FORTRAN, Pascal etc.	C++, JAVA, VB.NET, Python etc.

Basic Concepts of Object-Oriented Programming :

- OOP is a methodology or paradigm to design a program using classes and objects.
- Alan Kay coined the term “object oriented programming” at grad school in 1966 or 1967.
- It simplifies software development and maintenance by providing some concepts:

1. Object
2. Class
3. Inheritance
4. Polymorphism
5. Abstraction
6. Encapsulation



Objects

- Any entity that has state and behavior is known as an object.
- An object is an instance of class.
- For example a chair, pen, table, keyboard, bike, etc.

Example:

Class: Human
Class: Fruit

Object: Man, Woman
Object: Apple, Banana,
Mango, Guava etc.

Class: Mobile phone

Object: iPhone, Samsung,
Moto, Nokia etc.

Class: Food

Object: Pizza, Burger, Samosa

Class

- Collection of objects is called class.
- Class is a user-defined data-type
- It is the basic building block of object oriented programming.

Inheritance

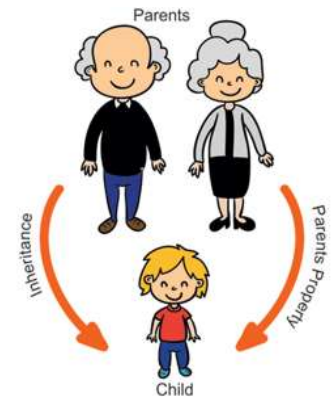
- When one object acquires all the properties and behaviors of a parent object, it is known as inheritance.

Polymorphism

- Existing in multiple forms.
- If one task is performed by different ways, it is known as polymorphism.

Abstraction

- Hiding internal details and showing functionality is known as abstraction.
- For example phone call, we don't know the internal processing.



Encapsulation

- Binding (or wrapping) code and data together into a single unit are known as encapsulation.
- For example capsule, it is wrapped with different medicines.

Advantage and disadvantage of OOPs Concept :

Advantages of OOP:

- Improved software-development productivity.
- Improved software maintainability.
- Faster development.
- Lower cost of development.
- Higher-quality software.

Disadvantages of OOP:-

- Larger program Size.
- Effort.
- Speed.

Application of OOP :

- Client-server system
- Real-Time System design
- Hypertext and Hypermedia
- AI Expert System
- Office automation System
- Stimulation and modeling system
- Object-oriented database

Define object and class in C++.

What is Object ?

- Any entity that has state and behavior is known as an object.
- For example a chair, pen, table, keyboard, bike, etc.
- Object means a real-world entity.
- An Object can be defined as an instance of a class.
- An object contains an address and takes up some space in memory.



Example: A dog is an object because it has states like color, name, breed, etc. as well as behaviors like wagging the tail, barking, eating, etc.

What is Class ?

- Collection of objects is called class.
- Class is a user-defined data-type
- A class is a data-type that has its own members i.e. data members and member functions.
- It is the blueprint for an object in object oriented programming language.
- It is the basic building block of object oriented programming in c++.
- A class can have multiple objects which have properties and behaviour that in common for all of them.



Difference between Object and Class :

Object	Class
Object is an instance of a class.	A class is a user define data type, which holds its own data member and member function.
Object is a real world entity such as pen, laptop, mobile, bed, keyboard etc.	Class is a group of similar objects.
Object is a physical entity.	Class is a logical entity.
Object is created through new keyword	Class is declared using class keyword.
Object is created many times as per requirement.	Class is declared once.
Object allocates memory when it is created.	Class doesn't allocated memory when it is created.



C++ history :

- C++ is a programming language.
- It was developed in 1980 by Bjarne Stroustrup at bell laboratories of AT&T located in U.S.A.
- Bjarne Stroustrup is known as the founder of C++ language.
- C++ is the enhanced name of C.
- The first name of C++ is C with class.
- C++ programming is called a superset of C, it means any valid C program is also a valid C++ program.

Language	Year	Developed By
BCPL	1967	Martin Richard
B	1970	Ken Thompson
Traditional C	1972	Dennis Ritchie
C++	1980	Bjarne Stroustrup

Difference between C and C++ :

C	C++
C follows the procedural style programming.	It supports both procedural and object oriented.
Data is less secured in C.	Data is more secured in C++.
In C, scanf() and printf() are mainly used for input/output.	C++ mainly uses stream cin and cout to perform input and output operations.
File extension in C is .C(dot C).	File extension in C++ is .CPP(dot CPP).
C does not support inheritance.	C++ support inheritance.
C is a subset of C++.	C++ is a superset of C.
Operator overloading is not possible in C.	Operator overloading is possible in C++.

Write the structure of C++ Program

The basic structure of c++ program is given below :

```
#include <iostream.h>
#include<conio.h>
void main()
{
    clrscr();
    cout << "Welcome to C++ Programming.";
    getch();
}
```



#include<iostream.h>

- includes the standard input output library functions.
- It provides cin and cout methods for reading from input and writing to output respectively.

#include <conio.h>

- include the console input output library functions.
- The getch() function is defined in conio.h file.

void main()

- The main() function is the entry point of every program in C++ language.
- The void keyword specifies that it returns no value.

What is Keyword?

- Keywords are those words whose meaning is already defined by Compiler.
- Keywords are predefined reserved words that have special meanings.
- They can't be used as identifiers in your program.
- There are a total of 95 reserved words in C++.
- The following keywords are reserved for Microsoft C++.

Example

int	float	static_cast
auto	for	struct
else	friend	switch
bool	goto	if

Identifiers:

- Identifiers are names of variables or class or function.
- They are user defined names consisting sequence of letters and digits.
- Rules for writing an identifier :
 - Keywords cannot be used as identifiers.
 - First character must be non numeric.
 - Symbols are not allowed except underscore.
 - Both lower case & upper letter are allowed.
 - Space are not allowed.

Examples: x2, area_of_circle, _ans etc.



Explain various object of operators available in C++:

- An operator is a symbol that tells the compiler to perform specific manipulations.
- C++ is rich in built-in operators and provide the following types of operators –

Arithmetic operators	+, -, *, /, %
Assignment operators	=, +=, -=, *=
Relational operators	<, >, <=, >=, !=
Logical operators	&&, , !
Bit wise operators	&, , ^, >>, <<
Conditional (ternary) operators	?
Misc. operators	Sizeof, comma, cast, dot, ampersand, indirection

sizeof Operator:

- This unary operator is used to compute the size of its operand or variable.

Comma Operator(,):

- This binary operator is used to evaluate its first operand and discards the result, it then evaluates the second operand and returns this value (and type).
- It is used to combine various expressions together.

Cast Operator: This unary operator is used to convert one data type into another.

Dot Operator(.): This operator is used to access members of structure variables or class objects in C++.

& Operator: This is a pointer operator and is used to represent the memory address of an operand.

*** Operator:** This is an Indirection Operator

<< Operator: This is the output operator. It prints the output value.

>> Operator: This is the input operator. It gets the input value.

Explain various data types available in C++ using examples.

- In C programming the data types is a keyword.
- It defines the size of data in a program.
- It also defines which type of data can be stored in a memory.
- It is used to create a variable.
- Data types in C++ is divided into three types:

Primitive Data Types:

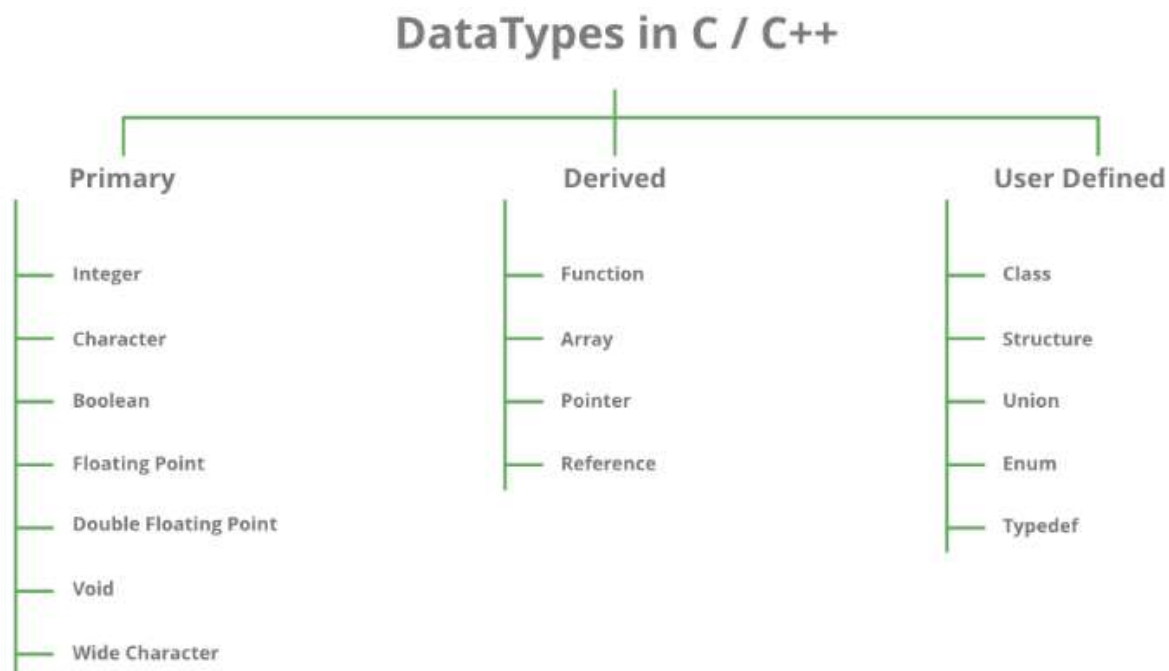
- These data types are predefined data types and can be used directly by the user to declare variables.
- **Example:** int, char , float, bool etc. Primitive data types available in C++ are:

Derived Data Types:

- The data-types that are derived from the primitive data types are referred to as Derived Data Types.
- These can be of four types namely: function, array, pointer, reference etc.

Abstract or User-Defined Data Types:

- These data types are defined by user itself.
- C++ provides the following user-defined datatypes: class, structure, union, enumeration etc.

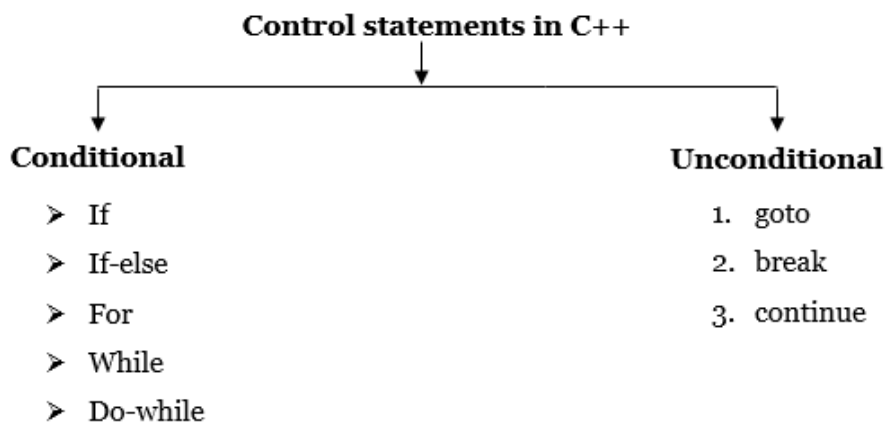


What is token in C++? Explain in brief.

- A token is the smallest element of a C++ program that is meaningful to the compiler.
- The C++ parser recognizes these kinds of tokens:
 - Keywords
 - Identifiers
 - Numeric
 - String and Character Literals
 - User-Defined Literals
 - Operators
 - Punctuators

Control statements in C++

- The statements that help programmers to indicate which sections of code to use at specific times is called control statements.
- Control statements are elements in the source code that control the flow of program execution.
- There are two types of control statements: conditional and unconditional.
 1. Conditional and
 2. Unconditional



C++ if Statement :

Syntax :

```

if(condition)
{
    Statements;
}
```

- The if statement evaluates the condition inside the parentheses ().
- If the condition evaluates to true, the code inside the body of if is executed.
- If the condition evaluates to false, the code inside the body of if is skipped.



If-else statement:

- An **if** statement can be followed by an optional **else** statement, which executes when the boolean expression is false.
- If the boolean expression evaluates to **true**, then the **if block** of code will be executed, otherwise **else block** of code will be executed.

Syntax :

```
if(condition)
{
    statement(s)
}
else
{
    statement(s);
}
```

C++ for loop:

Syntax:

```
for(initialization; condition; steps)
{
    Statements;
}
```

- **initialization** - initializes variables and is executed only once
- **condition** - if **true**, the body of **for** loop is executed
if **false**, the **for** loop is terminated
- **update** - updates the value of initialized variables and again checks the condition

C++ while Loop:

Syntax:

```
while(condition)
{
    Statements;
    Steps;
}
```

- A **while** loop evaluates the **condition**
- If the **condition** evaluates to **true**, the code inside the **while** loop is executed.
- The **condition** is evaluated again.
- This process continues until the **condition** is **false**.
- When the **condition** evaluates to **false**, the loop terminates.



C++ do...while Loop:

Syntax:

```
do
{
    Statements;
    Steps;
} while(condition);
```

- The body of the loop is executed at first. Then the `condition` is evaluated.
- If the `condition` evaluates to `true`, the body of the loop inside the `do` statement is executed again.
- The `condition` is evaluated once again.
- If the `condition` evaluates to `true`, the body of the loop inside the `do` statement is executed again.
- This process continues until the `condition` evaluates to `false`. Then the loop stops.

Unconditional Control Statements

- Unconditional control statements do not need to satisfy any condition.
- They immediately move control from one part of the program to another part.
- Goto, break and continue are unconditional statements in C++.

Scope resolution operator:

- Scope resolution operator is used to access the hidden variable or function of a program.
- The operator is represented as the double colon (`::`) symbol.
- C++, the scope resolution operator is used for the following purposes.
 - To access a global variable when there is a local variable with same name:
 - To define a function outside a class.
 - To access a class's static variables.

Memory Management Operators:

- In C language, we use the **malloc()** or **calloc()** functions to allocate the memory dynamically at run time, and **free()** function is used to deallocate the dynamically allocated memory.
- C++ also supports these functions, but C++ also defines two unary operators:
 1. **new** and
 2. **delete**
- A **new** operator is used to create the object while a **delete** operator is used to delete the object.



Enumerated variables :

- Enumeration is a user defined datatype in C++ language.
- It is used to assign names to the integral constants which makes a program easy to read and maintain.
- The keyword “enum” is used to declare an enumeration.

Syntax:

```
enum enum_name{const1, const2, ..... };
```

Example:

```
enum colors{red, black};
```

Data Conversion :

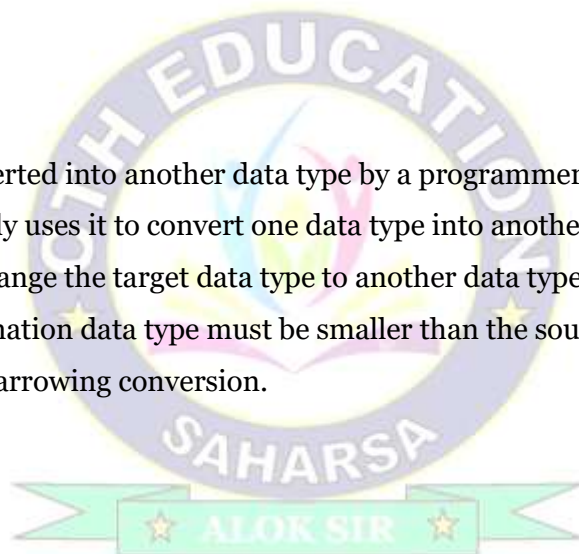
- There are two terms **type casting** and the **type conversion** are used in a program to convert one data type to another data type.

What is a type casting?

- When a data type is converted into another data type by a programmer is known as **type casting**.
- The programmer manually uses it to convert one data type into another.
- It is used if we want to change the target data type to another data type.
- Remember that the destination data type must be smaller than the source data type.
- Hence it is also called a narrowing conversion.

```
float b = 3.0;
```

```
int a = (int) b;
```



What is type conversion?

- If a data type is automatically converted into another data type at compile time is known as type conversion.
- The conversion is performed by the compiler if both data types are compatible with each other.
- Remember that the destination data type should not be smaller than the source type.
- It is also known as **widening** conversion of the data type.

```
int a = 20;
```

```
Float b;
```

```
b = a;
```

Unit – 02: Classes and Objects:

- Class & Object: Introduction,
- Creating Objects, specifying a Class,
- Defining Member Function,
- Memory allocation for Objects.
- Static data members, Static Member Functions,
- Access Specifier Class,
- Friend Function.
- Array of Objects, Object as function Arguments.
- Concepts of constructors, Type of constructors
- Multiple constructors in a class, Constructors with Default Arguments.
- Destructors.

Questions to be discussed:

1. Define object and class in C++ with suitable example.
2. What is friend function? Explain with the help of suitable examples.
3. Define the relation between structure and class.
4. What is access specifiers? Explain its type in brief.
5. What is constructor? List some of the special properties of the constructor function.
6. Explain different types of constructor.
7. What is Destructor? List the some of special properties of constructor function.
8. Discuss the purpose of using constructor and destructor.
9. Write the difference between Constructor and Destructor. ★
10. Purpose of using constructor and destructor in C++.
11. What is an array? What are the advantage of it?
12. Write short notes on:
 - a. Destructor
 - b. Friend function
 - c. Object
 - d. Class

What is Object? How to declare object in C++?

- Any entity that has state and behavior is known as an object.
- For example a chair, pen, table, keyboard, bike, etc.
- Object means a real-world entity.
- An Object can be defined as an instance of a class.
- An object contains an address and takes up some space in memory.



Example: A dog is an object because it has states like color, name, breed, etc. as well as behaviors like wagging the tail, barking, eating, etc.

Declaring Objects:

- When a class is defined, only the specification for the object is defined; no memory or storage is allocated.
- To use the data and access functions defined in the class, you need to create objects.

Syntax:

```
Class_Name Object_Name ;
```

What is Class? How to declare class in C++?

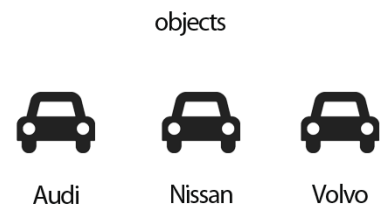
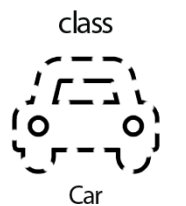
- Collection of objects is called class.
- Class is a user-defined data-type.
- A class is a data-type that has its own members i.e. data members and member functions.
- It is the blueprint for an object in object oriented programming language.
- It is the basic building block of object oriented programming in c++.
- A class can have multiple objects which have properties and behaviour that in common for all of them.

Declaration of Class

- A class is defined in C++ using keyword class followed by the name of class.
- The body of class is defined inside the curly brackets and terminated by a semicolon at the end.

Syntax :

```
class class_Name
{
    private/public/protected :
    Data members;
    Member functions;
};
```



Difference between Object and Class:

Object	Class
Object is an instance of a class.	A class is a user define data type, which holds its own data member and member function.
Object is a real world entity such as pen, laptop, mobile, bed, keyboard etc.	Class is a group of similar objects.
Object is a physical entity.	Class is a logical entity.
Object is created through new keyword	Class is declared using class keyword.
Object is created many times as per requirement.	Class is declared once.
Object allocates memory when it is created.	Class doesn't allocated memory when it is created.

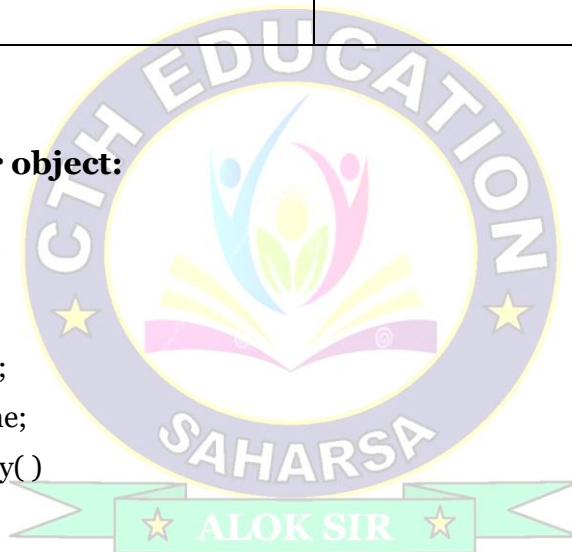
Memory allocation for object:

```

Class student
{
    public:
    int roll_no;
    string Name;
    void display()
    {
        cout<<"Your Name is"<<Name;
    }
};

void main()
{
    Student obj;
    obj.roll_no=20;
    obj.Name="XYZ";
    obj.display();
}

```



Access Specifiers:

- Access Specifiers in a class are used to assign the accessibility to the class members.
- It is used to set some restrictions on the class members so that they can't be directly accessed by the outside functions.
- Access specifiers are also known as access modifiers.
- There are 3 types of access modifiers available in C++:
 1. Public
 2. Private
 3. Protected

Note: If we do not specify any access modifiers for the members inside the class, then by default the access modifier for the members will be **Private**.

Public:

- All the class members declared under the public specifier will be available to everyone.
- The data members and member functions declared as public can be accessed by other classes and functions too.
- The public members of a class can be accessed from anywhere in the program using the direct member access operator (.) with the object of that class.

Private:

- The class members declared as *private* can be accessed only by the member functions inside the class.
- They are not allowed to be accessed directly by any object or function outside the class.
- Only the member functions or the friend functions are allowed to access the private data members of the class.

Protected:

- The protected access modifier is similar to the private access modifier in the sense that it can't be accessed outside of its class unless with the help of a friend class.
- The difference is that the class members declared as Protected can be accessed by any subclass (derived class) of that class as well.

Functions in C++

- A function is a group of statements that together perform a task.
- Every C++ program has at least one function, which is **main()**.
- The function contains the set of programming statements enclosed by curly braces { }.
- The main goal of function to increase the reusability of a program.

Syntax

```
return_type function_name([ arg1_type arg1_name, ... ])
{
    codes ;
}
```

Friend function:

- A friend function is a function which is not the member of a class instead of that it can access private and protected member of class.
- Syntax

```
friend return_type function_name(class_name object)
{
    Body;
}
```

- Friend function is declared in the class with friend keyword.
- Friend function can become friend one or more than one class.

What is constructor?

- A constructor is a member function of a class which initializes objects of a class.
- In C++, Constructor is automatically called when object create.
- It is special member function of the class.
- Constructor has same name as the class itself
- Constructors don't have return type
- If we do not specify a constructor, C++ compiler generates a default constructor for us.

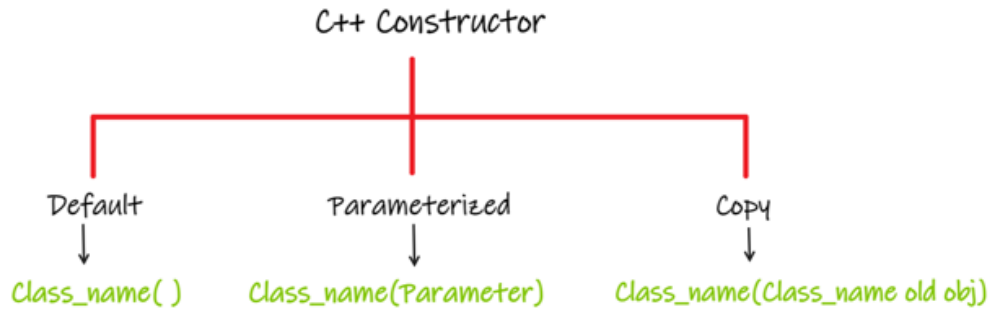
Features/properties of constructors:

Constructor have following special features:

1. A constructor name must be same as that of its class name.
2. Constructors are called automatically when the objects are created.
3. Constructors should be declared in the public section to be available to all the functions.
4. Constructors do not have return type , not even void and therefore they can not return value.

5. Constructors can not be inherited.
6. Constructors can not be static.
7. Constructors can not be virtual.
8. Constructors are member functions so they can be overloaded.

Types of Constructor:



Default Constructors:

- Default constructor is the constructor which doesn't take any argument.
- It has no parameters.
- Even if we do not define any constructor, the compiler will automatically provide a default constructor.

Parameterized Constructors:

- It is possible to pass arguments to constructors.
- Typically, these arguments help initialize an object when it is created.
- To create a parameterized constructor, simply add parameters to it.
- When you define the constructor's body, use the parameters to initialize the object.

Copy Constructor:

- It is a member function which initializes an object using another object of the same class.
- These are special type of Constructors which takes an object as argument, and is used to copy values of data members of one object into other object.
- Whenever we define one or more non-default constructors for a class, a default constructor(without parameters) should also be explicitly defined as the compiler will not provide a default constructor in this case.
- However, it is not necessary but it's considered to be the best practice to always define a default constructor.

What is destructor?

- Destructor is a member function which destructs or deletes an object.
- Destructor is a special class function which destroys the object as soon as the scope of object ends.
- The destructor is called automatically by the compiler when the object goes out of scope.
- 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.

Properties/Features of Destructor:

- Destructor function is automatically called when the objects are destroyed.
- It cannot be declared static or const.
- The destructor does not have arguments.
- It has no return type not even void.
- A destructor should be declared in the public section of the class.

Write the difference between Constructor and Destructor.

Constructor	Destructor
Constructor helps to initialize the object of a class.	Destructor is used to destroy the instances.
Syntax: ClassName(arguments if any) { Constructor's Body }	Syntax: ~ ClassName(no arguments) { };
Constructor can either accept an arguments or not.	While it can't have any arguments.
It is called when an object of a class is created.	It is called while object of the class is deleted.
Constructor can be overloaded.	While it can't be overloaded.
The constructor's name is same as the class name.	Here, it's name is also same as the class name preceded by tiled (~) operator.
In a class, there can be multiple constructors.	While in a class, there is always a single destructor.

Purpose of using constructor and destructor in C++:

The main purpose of using constructor and destructor is given below -

- Constructor and Destructor are the special member functions of the class which are created by the C++ compiler or can be defined by the user.
- Constructors are special class functions which performs initialization of every object.
- The Compiler calls the Constructor whenever an object is created.
- Constructors initialize values to object members after storage is allocated to the object.
- Whereas, Destructor on the other hand is used to destroy the class object.

What is an array?

- Array is a collection of similar type of data.
- It stores data in contiguous manner.
- Array is linear data structure.
- It can store fixed number of values.
- An object of class represents a single record in memory, if we want more than one record of class type, we have to create an array of class or object.
- The array of type class contains the objects of the class as its individual elements.
- Thus, an array of a class type is also known as an array of objects.

The syntax for declaring an array of objects is:

```
class_name array_name [size] ;
```

Advantages of Arrays:

- Array stores data elements of the same data type.
- Maintains multiple variable names using a single name.
- Arrays help to maintain large data under a single variable name.
- This avoid the confusion of using multiple variables.
- Arrays can be used for sorting data elements.
- In an array, accessing an element is very easy by using the index number.
- The search process can be applied to an array easily.
- 2D Array is used to represent matrices.
- For any reason a user wishes to store multiple values of similar type then the Array can be used and utilized efficiently.

Difference between array and structure:

ARRAY	STRUCTURE
Array stores similar type of element.	Structure stores dis-similar types of element.
Array elements are stored in contiguous memory locations.	Structure elements may or may not be stored in a contiguous memory location.
It uses subscripts “[]” for accessing element.	Structure uses dot “.” for accessing element.
Array traversal and searching is easy and fast.	Structure traversal & searching is complex and slow.
No keyword is present to declare an array.	We use the keyword “struct” to define a structure.
It is declared directly.	The structure is a user-defined form of data type.

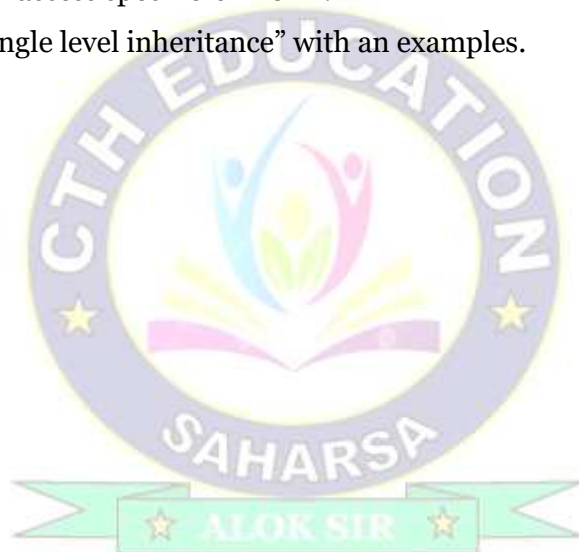


Unit – 03: Extending Classes using Inheritance

- Introduction to Inheritance,
- Types of Inheritance: Single, Multilevel, Multiple, Hierarchical, Hybrid.
- Virtual Base Class, Abstract Class, Constructors in Derived Class.
- Defying a derived class,
- Visibility Modes and Effects.

Questions to be discussed:

1. What is inheritance?
2. Explain different types of inheritance in details.
3. Write the difference between multiple and multi-level inheritance.
4. Describe different types of access specifiers in C++.
5. Explain the concept of “single level inheritance” with an examples.
6. Write short notes on:
 - a. Virtual base class
 - b. Abstract class



What is Inheritance?

- It is a process in which child object acquires all the properties and behaviors of parent object automatically.
- The capability to derive properties and characteristics from another class is called Inheritance.
- In such way, you can reuse, the attributes and behaviors which are defined in other class.

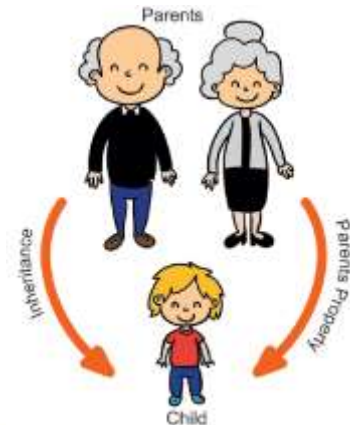
Sub Class: The class that inherits properties from another class is called Sub class or Derived Class.

Super Class: The class whose properties are inherited by sub class is called Base Class or Super class.

Types of Inheritance:

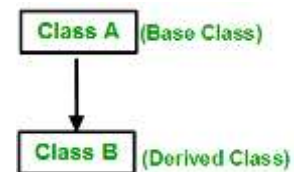
- There are five types of inheritance in C++:

- Single level inheritance
- Multilevel inheritance
- Multiple inheritance
- Hierarchical inheritance
- Hybrid inheritance



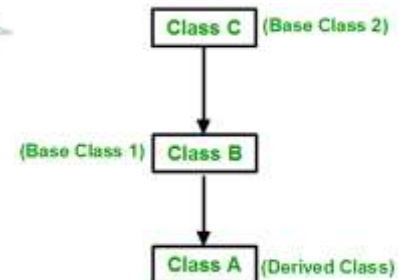
Single Inheritance:

- It is defined as the inheritance in which a derived class is inherited from the only one base class.



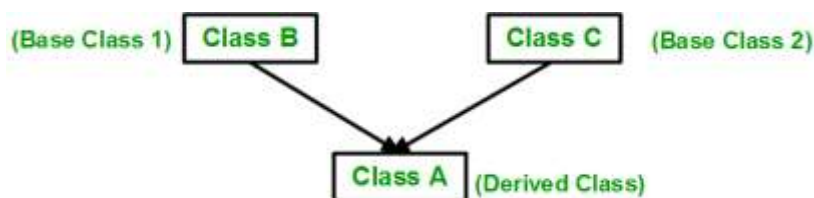
Multi-level Inheritance:

- It is a process of deriving a class from another derived class.



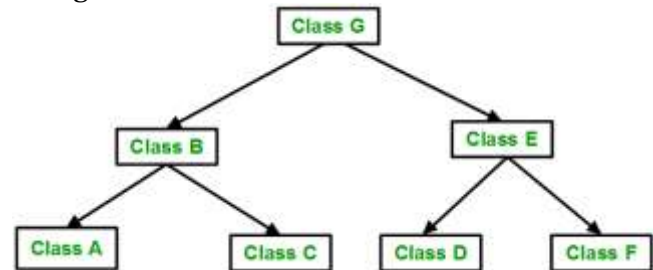
Multiple Inheritance:

- It is the feature of C++ where a class can inherit from more than one classes.
- That means one **sub class** is inherited from more than one **base classes**.



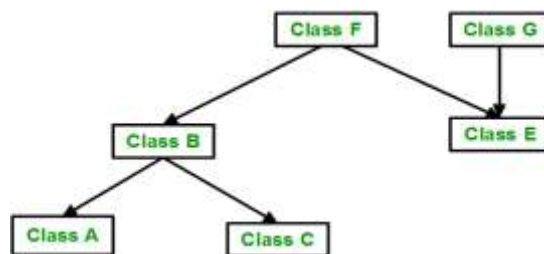
Hierarchical Inheritance:

- In this type of inheritance, more than one sub class is inherited from a single base class.
- i.e. more than one derived class is created from a single base class.

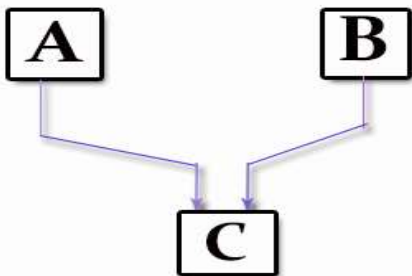
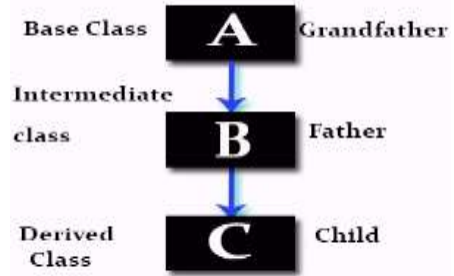


Hybrid (Virtual) Inheritance:

- Hybrid Inheritance is implemented by combining more than one type of inheritance.
- For example: Combining Hierarchical inheritance and Multiple Inheritance.



Write the difference between multiple and multi-level inheritance.

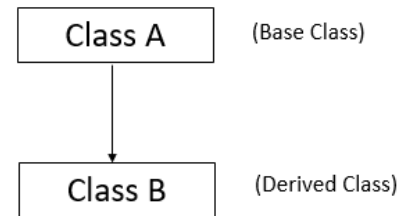
Multiple Inheritance	Multilevel Inheritance
Here, a child class inherits from more than one parent class.	Here, a child class is derived from another derived class.
Multiple Inheritance is not widely used because it makes the system more complex.	Multilevel Inheritance is widely used.
Multiple Inheritance has two class levels namely, base class and derived class.	It has three class levels namely, base class, intermediate class and derived class.
 <p style="text-align: center;">MULTIPLE INHERITANCE</p>	 <p style="text-align: center;">Fig: Multilevel Inheritance</p>

Explain the concept of “single level inheritance” with an examples.

- Inheritance is the process of deriving the properties of one class from another class.
- Single Level Inheritance is the mechanism of deriving a class from only one single base class.
- It is the simplest of all inheritance.

Example:

1. Animal is derived from living things
2. Car is derived from vehicle



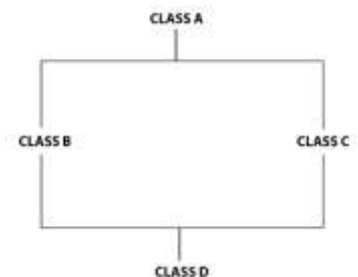
- In the above diagram, Class B is derived from Class A.
- Class A is base class and Class B is a derived class.

Differentiate between base class and derived class:

BASE CLASS	DERIVED CLASS
A class from which properties are inherited.	A class from which is inherited from the base class.
It is also known as parent class or superclass.	It is also known as child class or subclass.
It cannot inherit properties and methods of Derived Class.	It can inherit properties and methods of Base Class.
Syntax: Class base_classname{ ... }.	Syntax: Class derived_classname : access_mode base_class_name { ... }.

What is Virtual Class?

- It is the concept used in multiple inheritances to prevent ambiguity between multiple instances.
- Virtual Class is defined by writing a keyword “virtual” in the derived classes.
- It prevents multiple instances of a class appearing as a parent class in the inheritance.
- Virtual base classes are used in virtual inheritance.



Need for Virtual Base Class in C++:

- To prevent the error and let the compiler work efficiently, we’ve to use a virtual base class when multiple inheritances occur.
- When a class is specified as a virtual base class, it prevents duplication of its data members.
- Only one copy of its data members is shared by all the base classes that use the virtual base class.
- If a virtual base class is not used, all the derived classes will get duplicated data members.
- In this case, the compiler cannot decide which one to execute.



What is an abstract class in C++?

- The purpose of an abstract class is to provide an appropriate base class from which other classes can inherit.
- Abstract classes cannot be used to instantiate objects and serves only as an interface.
- Attempting to instantiate an object of an abstract class causes a compilation error.
- Abstract class must include at least one pure virtual function.
- A virtual function is declared using the pure specifier (= 0) syntax.

Visibility levels in C++/Access Specifiers/Modifiers in C++:

- It used to implement an important aspect of OOP known as Data Hiding.
- It is used to assign the accessibility to the class members.
- It apply some restrictions on the class members not to get directly accessed by the outside functions.
- There are 3 types of access modifiers available in C++:
 1. Public
 2. Private
 3. Protected

Note: If we do not specify any access modifiers for the members inside the class then by default the access modifier for the members will be Private.

Public:

- All the class members declared under the public specifier will be available to everyone.
- The data members and member functions declared as public can be accessed by any other classes and functions.
- The public members of a class can be accessed from anywhere in the program using the direct member access operator dot (.).

Private:

- The class members declared as private can be accessed only by the member functions inside the class.
- They are not allowed to be accessed directly by any object or function outside the class.
- Only the member functions or the friend functions are allowed to access the private data members of a class.

Protected:

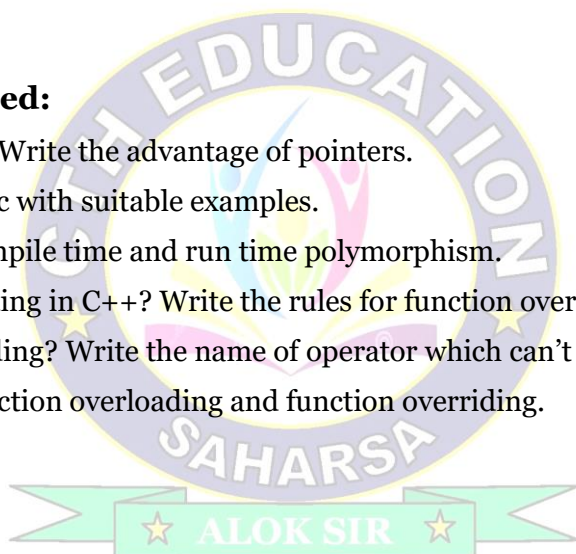
- Protected access modifier is similar to private access modifier.
- In the sense that it can't be accessed outside of it's class unless with the help of friend class.
- The difference is that the class members declared as Protected can be accessed by any subclass(derived class) of that class as well.

Unit – 04: Pointers and Polymorphism in C++

- Concepts of Pointer: Pointer Declaration, Pointer Operator, Address Operator, Pointer Arithmetic.
- Pointer to Objects: Pointer to Object, this pointer, Pointer to derived class.
- Introduction to Polymorphism:
- Run time Polymorphism:
- Function Overloading,
- Operator overloading,
- Overloading of Unary & Binary Operator,
- Rules for Operator Overloading.
- Virtual functions, Rules for virtual function, Pure virtual functions

Questions to be discussed:

1. What is pointers in C++? Write the advantage of pointers.
2. Explain pointer arithmetic with suitable examples.
3. Differentiate between compile time and run time polymorphism.
4. What is function overloading in C++? Write the rules for function overloading.
5. What is operator overloading? Write the name of operator which can't be overload.
6. Differentiate between function overloading and function overriding.
7. Write short notes on:
 - a. This pointer
 - b. Virtual function
 - c. Function overriding



What are Pointers?

- A pointer is a special variable whose value is the address of another variable.
- A pointer variable always stores address of another variable not data.
- It is also known as locator or indicator that points to an address of a value.
- Pointer is a special type of variable whose always stores the address of another variable.

Declaration of pointer:

- Pointer is a special type of variable whose always stores the address of another variable.
- The pointer in C++ language can be declared using * (asterisk symbol).

Syntax:

Data_type *pointer_name ;

Example:

```
int *a;
```

Advantage of pointer:

- 1) Pointer reduces the code and improves the performance.
- 2) It is used to retrieving strings, trees etc. and used with arrays, structures and functions.
- 3) We can return multiple values from function using pointer.
- 4) It makes you able to access any memory location in the computer's memory.

C++ Pointer Operators:

- C++ provides two pointer operators, which are
 - 1) Address of Operator & and
 - 2) Indirection Operator *.

Address of operator (&):

- It is an operator within C++ that returns the memory address of a variable.
- The address operator is a unary operator represented by an ampersand (&).
- It is also known as an address operator.

Indirection operator/Dereference operator (*):

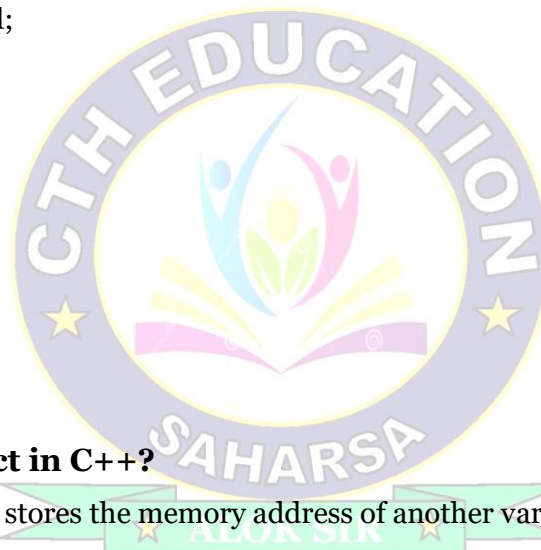
- The indirection operator is a unary operator.
- It returns the value of the variable present at the given address.
- It is completely opposite to the address-of operator.
- It is spelled as a **value pointed at the address**.

Pointer arithmetic:

- As we know that a pointer always stores address of another variable which is a numeric value.
- Therefore, you can perform arithmetic operations on a pointer.
- There are four arithmetic operators that can be used on pointers:
 - 1) Increment operator (++)
 - 2) Decrement operator (--)
 - 3) Addition operator (+) and
 - 4) Subtraction operator (-)

C++ program to elaborate pointer arithmetic:

```
#include<iostream.h>
#include<conio.h>
void main( )
{
    int a=20, b=10, c, d;
    int *p=&a,*q=&b;
    clrscr( );
    (*p)++, (*q)++;
    c=*p + *q;
    d=*p - *q;
    cout<<c<<endl;
    cout<<d<<endl;
    getch( );
}
```



What is Pointer to Object in C++?

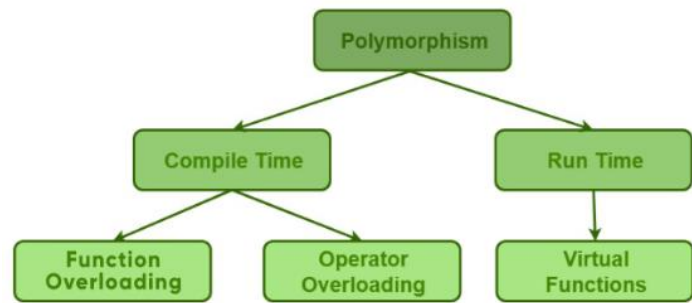
- A pointer is a variable that stores the memory address of another variable (or object).
- A pointer aims to point to a data type which may be int, character, float, etc.
- Pointers to objects aim to make a pointer that can access the object, not the variables.
- Pointer to object in C++ refers to accessing an object.
- Those pointers which are used to accessing object is known as **This Pointer**.

This pointer:

- Every Object in C++ has access to its own address through an important pointer called **This Pointer**.
- The This Pointer is an implicit parameter to all member functions.
- Therefore, inside a member function, this may be used to refer to the invoking object.
- Whenever a member function is called, it is automatically passed an implicit arguments that is This pointer to the invoking object (i.e. The object on which the function is invoked).

Polymorphism:

- The word “polymorphism” means having many forms.
- We can define polymorphism as the ability of a message to be displayed in more than one form.
- It occurs when we have many classes that are related to each other by inheritance.
- A real-life example of polymorphism is a person who at the same time can have different characteristics.
- So the same person exhibits different behavior in different situations, this is called polymorphism.
- There two types of Polymorphism:
 1. **Compile-time Polymorphism.**
 2. **Runtime Polymorphism.**



Difference between compile time and run time polymorphism:

Compile-time polymorphism	Runtime polymorphism
Compile time polymorphism means binding is occurring at compile time	Run time polymorphism where at run time we came to know which method is going to invoke
It can be achieved through static binding	It can be achieved through dynamic binding
Inheritance is not involved	Inheritance is involved
Method overloading is an example of compile time polymorphism	Method overriding is an example of runtime polymorphism
It is achieved by function overloading and operator overloading.	It is achieved by virtual functions and pointers.
It provides fast execution because the method that needs to be executed is known early at the compile time.	It provides slow execution as compare to early binding because the method that needs to be executed is known at the runtime.

C++ Overloading:

- If we create two or more members having the same name but different in number or type of parameter, it is known as overloading.
- There are two types of overloading in C++:
 1. Function overloading
 2. Operator overloading

C++ Function Overloading:

- Function Overloading is the process of assigning two or more function with the same name, but different in parameters is known as function overloading.
- It is only through these differences compiler can differentiate between the functions.
- The **advantage** of Function overloading is that it increases the readability of the program.

Rules of Function Overloading in C++:

- The functions must have the same name.
- The functions must have different types of parameters.
- The functions must have a different set of parameters.
- The functions must have a different sequence of parameters.

Operators Overloading:

- An operator overloading is the process of using an operator instead of a function to do a special task.
- In this concept we assigning a special task to the operator.
- We can overload an existing operator.
- We can't create new operator by using operator overloading.
- We can't change the basic meaning of the operator.
- In operator overloading at least one user defined data we should provide.

Syntax:

Operator that can't be overloaded:

- Scope operator (::)
- Sizeof
- member selector (.)
- member pointer selector (*)
- ternary operator (?:)

Keyword Operator to be overloaded

```

ReturnType classname :: Operator OperatorSymbol (argument list)
{
    \\Function body
}
  
```

What is overriding in C++ ?

- If derived class defines same function in base class, it is known as function overriding.
- What ever methods parent has by default available to the child through inheritance.
- Some times child may not satisfy with parent method implementation.
- Then child is allow to redefine that method based on its requirement, this process is called overriding.

Example:

Class P

```
{
    Public void property()
    {
    }
    Public void marry()
    {
    }
}
```

Class C

```
{
    Public void marry()
    {
    }
}
```

Difference between function overloading and function overriding:

Function Overloading	Function Overriding
Function Overloading provides multiple definitions of the function by changing signature.	Function Overriding is the redefinition of base class function in its derived class with same signature.
An example of compile time polymorphism.	An example of run time polymorphism.
Overloading is used when the same function has to behave differently depending upon parameters passed to them.	Overriding is needed when derived class function has to do some different job than the base class function.
A function has the ability to load multiple times.	A function can be overridden only a single time.
In function overloading, we don't need inheritance.	In function overriding, we need an inheritance concept.

Virtual function:

- A virtual function is a member function which is declared within a base class and is re-defined (overridden) by a derived class.
- When you refer to a derived class object using a pointer or a reference to the base class, you can call a virtual function for that object and execute the derived class's version of the function.
- Virtual functions ensure that the correct function is called for an object.
- They are mainly used to achieve Runtime polymorphism
- Functions are declared with a **virtual** keyword in base class.
- The resolving of function call is done at runtime.

Rules for Virtual Functions:

1. Virtual functions cannot be static.
2. A virtual function can be a friend function of another class.
3. It should be accessed using pointer or reference of base class type to achieve runtime polymorphism.
4. The prototype of virtual functions should be the same in the base as well as derived class.
5. They are always defined in the base class and overridden in a derived class.
6. A class may have virtual destructor but it cannot have a virtual constructor.

Difference between virtual function and pure virtual function:

Virtual function	Pure virtual function
A virtual function is a member function in a base class that can be redefined in a derived class.	A pure virtual function is a member function in a base class whose declaration is provided in a base class and implemented in a derived class.
The classes which are containing virtual functions are not abstract classes.	The classes which are containing pure virtual function are the abstract classes.
In case of a virtual function, definition of a function is provided in the base class.	In case of a pure virtual function, definition of a function is not provided in the base class.
All the derived classes may or may not redefine the virtual function.	All the derived classes must define the pure virtual function.



Unit – 05: File Operations

- C++ stream Classes,
- Classes for File stream operations,
- Opening files, Closing Files, reading from and writing to files.
- Detection of End of file, File Modes.

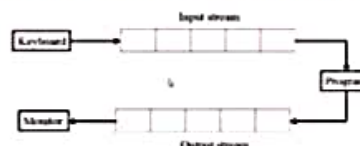
Questions to be discussed:

1. Discuss about different stream classes in C++.
2. Explain file operations in C++.
3. Explain various file modes in C++.



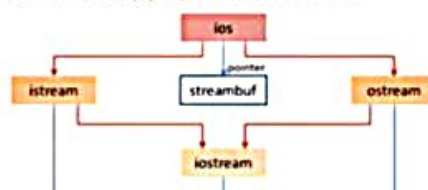
What is stream in C++?

- We give input to the executing program and the execution program gives back the output.
- The sequence of bytes given as input to the executing program and the sequence of bytes that comes as output from the executing program are called stream.
- In other words, a stream is a sequence of bytes taken from a device or sent to a device.
- There are two types of stream:
 1. Input stream(cin)
 2. Output stream(cout)



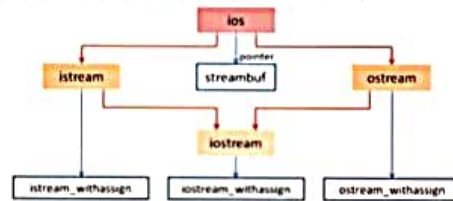
Classes for File stream operations:

- The I/O system of C++ contains a set of classes which define the file handling methods.
- The stream classes are defined in a header file <iostream.h>.
- The stream classes associated with the C++ stream include ios class, istream class, and ostream class.
- These classes are designed to manage the disk files which are declared in stream classes and therefore we must include this file in any program that uses files.



Classes for File stream operations:

- The I/O system of C++ contains a set of classes which define the file handling methods.
- The stream classes are defined in a header file <iostream.h>.
- The stream classes associated with the C++ stream include ios class, istream class, and ostream class.
- These classes are designed to manage the disk files which are declared in stream classes and therefore we must include this file in any program that uses files.



ios:

- ios stands for input output stream.
- This class is the base class for other classes in this class hierarchy.
- ios class is the highest class in the entire hierarchical structure of the C++ stream.
- This class contains the necessary facilities that are used by all the other derived classes for input and output operations.

istream:

- istream stands for input stream.
- This class is derived from the class 'ios'.
- This class handle input stream.
- The extraction operator(>>) is overloaded in this class to handle input streams from files to the program execution.



- This class declares input functions such as get(), getline() and read().

ostream:

- ostream stands for output stream.
- This class is derived from the class 'ios'.
- This class handle output stream.
- The insertion operator(<<) is overloaded in this class to handle output streams to files from the program execution.
- This class declares output functions such as put() and write().

streambuf:

- This class contains a pointer which points to the buffer which is used to manage the input and output streams.

fstreambase:

- This class provides operations common to the file streams.
- Serves as a base for fstream, ifstream and ofstream class.
- This class contains open() and close() function.

ifstream:

- This class provides input operations.
- It contains open() function with default input mode.
- Inherits the functions get(), getline(), read(), seekg() and tellg() functions from the istream.

ofstream:

- This class provides output operations.
- It contains open() function with default output mode.
- Inherits the functions put(), write(), seekp() and tellp() functions from the ostream.

fstream:

- This class provides support for simultaneous input and output operations.
- Inherits all the functions from istream and ostream classes through iostream.

ifstream:

- This class provides input operations.
- It contains open() function with default input mode.
- Inherits the functions get(), getline(), read(), seekg() and tellg() functions from the istream.

ofstream:

- This class provides output operations.
- It contains open() function with default output mode.
- Inherits the functions put(), write(), seekp() and tellp() functions from the ostream.

fstream:

- This class provides support for simultaneous input and output operations.
- Inherits all the functions from istream and ostream classes through iostream.



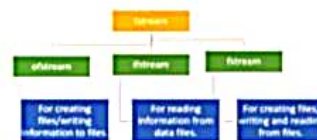
File operations In C++:

C++ language allows us to perform important disk input/output operations such as -

1. Creating a new file on the disk.
2. Reading the file stored on the disk.
3. Writing data to the file stored on the disk.
4. Appending new data to the end of the file stored on the disk.
5. Modifying the content of the file stored on the disk.

To perform any file operations, C++ provides us a few file stream classes, such as:

- **ifstream**, to perform the file input operations.
- **ofstream**, to perform the file output operations.
- **fstream**, to perform any file input and output operations.



Opening files, Closing Files, reading from and writing to files:

How to Open Files:

- Before performing any operation on a file, you must first open it.
- If you need to write to the file, open it using ofstream or ostream objects.
- If you only need to read from the file, open it using the ifstream object.
- The three objects, that is, fstream, ofstream, and ifstream, have the open() function defined in them.

The function takes this syntax:

```
open (file_name, mode);
```

How to Close Files:

- However, as a programmer, you should learn to close open files before the program terminates.
- The fstream, ofstream, and ifstream objects have the close() function for closing files.
- The function takes this syntax:

```
void close();
```

How to Write to Files:

- You can write to file right from your C++ program.
- You use stream insertion operator (<<) for this.
- The text to be written to the file should be enclosed within double-quotes.



How to Read from Files:

- You can read information from files into your C++ program.
- This is possible using stream extraction operator (>>).
- You use the operator in the same way you use it to read user input from the keyboard.
- However, instead of using the cin object, you use the ifstream/istream object.

How to detect an end of a file in C++?

- End of file in C++ can be detected using eof.
- End Of File returns non - zero value if the end of file (EOF) is encountered and a zero otherwise.
- You can either use the ifstream object 'fin' which returns 0 on an end of file or you can use eof() which is a member function of the ios class.
- It returns a non zero value on reaching the end of file.

3. Writing data to the file stored on the disk.
4. Appending new data to the end of the file stored on the disk.
5. Modifying the content of the file stored on the disk.

To perform any file operations, C++ provides us a few file stream classes, such as:

- **ifstream**, to perform the file input operations.
- **ofstream**, to perform the file output operations.
- **fstream**, to perform any file input and output operations.



Opening files, Closing Files, reading from and writing to files:

How to Open Files:

- Before performing any operation on a file, you must first open it.
- If you need to write to the file, open it using **fstream** or **ofstream** objects.
- If you only need to read from the file, open it using the **ifstream** object.
- The three objects, that is, **fstream**, **ofstream**, and **ifstream**, have the **open()** function defined in them.

The function takes this syntax:

```
open (file_name, mode);
```

How to Close Files:

- However, as a programmer, you should learn to close open files before the program terminates.
- The **fstream**, **ofstream**, and **ifstream** objects have the **close()** function for closing files.
- The function takes this syntax:

```
void close();
```

How to Write to Files:

- You can write to file right from your C++ program.
- You use stream insertion operator (<<) for this.
- The text to be written to the file should be enclosed within double-quotes.

How to Read from Files:

- You can read information from files into your C++ program.
- This is possible using stream extraction operator (>>).
- You use the operator in the same way you use it to read user input from the keyboard.
- However, instead of using the **cin** object, you use the **ifstream/fstream** object.

How to detect an end of a file in C++?

- End of file in C++ can be detected using **eof**.
- End Of File returns non - zero value if the end of file (EOF) is encountered and a zero otherwise.
- You can either use the **ifstream** object '**fin**' which returns 0 on an end of file or you can use **eof()** which is a member function of the **ios** class.
- It returns a non zero value on reaching the end of file.

File modes:

- In C++, for every file operation, exists a specific file mode.
- These file modes allow us to create, read, write, append or modify a file.
- The file modes are defined in the class **ios**.

File Modes	Description
ios::in	Searches for the file and opens it in the read mode only(if the file is found).
ios::out	Searches for the file and opens it in the write mode . If the file is found, its content is overwritten. If the file is not found, a new file is created.
ios::app	Searches for the file and opens it in the append mode i.e. this mode allows you to append new data to the end of a file. If the file is not found, a new file is created.
"ios::binary"	Searches for the file and opens the file(if the file is found) in a binary mode to perform binary input/output file operations.
ios::ate	Searches for the file, opens it and positions the pointer at the end of the file. This mode when used with ios::binary , ios::in and ios::out modes, allows you to modify the content of a file.
"ios::trunc"	Searches for the file and opens it to truncate or deletes all of its content(if file is found).
"ios::nocreate"	Searches for the file and if the file is not found, a new file will not be created