FACULTY NAME:J.VIJAY SATHIARAJ
DESIGNAION:GUEST LECTURER
DEPARTMENT:GEOGRAPHY
CLASS:M.TECH
SEMESTER:II
SUBJECT:.NET PROGRAMMING
SUBJECT CODE:22EC03

C# Language Basics: Introduction to C# - variables, data types, Array, operators and expressions - Control Statements - Loops - object oriented programming

INTRODUCTION OF C#

C# is pronounced as "C-Sharp". It is an object-oriented programming language provided by Microsoft that runs on .Net Framework.

By the help of C# programming language, we can develop different types of secured and robust applications:

- Window applications
- Web applications
- Distributed applications
- Web service applications
- Database applications etc.

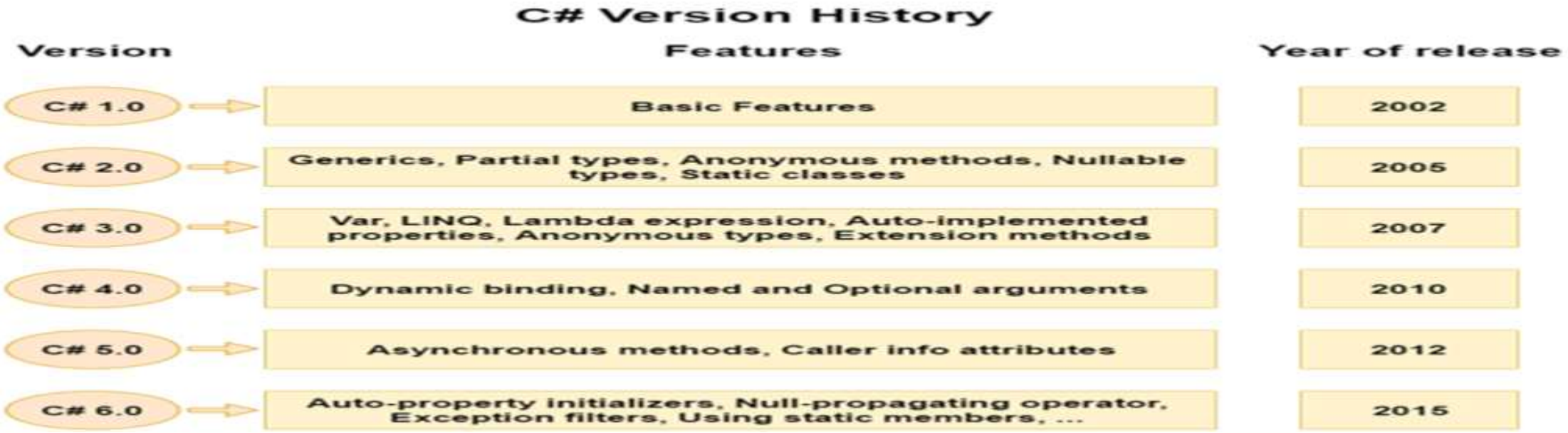
C# is approved as a standard by ECMA and ISO. C# is designed for CLI (Common Language Infrastructure). CLI is a specification that describes executable code and runtime environment.

C# programming language is influenced by C++, Java, Eiffel, Modula-3, Pascal etc. languages.

Anders Hejlsberg is known as the founder of C# language.

It is based on C++ and Java, but it has many additional extensions used to perform component oriented programming approach.

C# has evolved much since their first release in the year 2002. It was introduced with .NET Framework 1.0 and the current version of C# is 5.0.



C# Features

C# is object oriented programming language. It provides a lot of features that are given below.

1. Simple
2. Modern programming language
3. Object oriented
4. Type safe
5. Interoperability
6. Scalable and Updateable
7. Component oriented
8. Structured programming language
9. Rich Library
10. fast speed

1) **Simple**

C# is a simple language in the sense that it provides structured approach (to break the problem into parts), rich set of library functions, data types etc.

2) **Modern Programming Language**

C# programming is based upon the current trend and it is very powerful and simple for building scalable, interoperable and robust applications.

3) **Object Oriented**

C# is object oriented programming language. OOPs makes development and maintenance easier where as in Procedure-oriented programming language it is not easy to manage if code grows as project size grow.

4)**Type Safe**

C# type safe code can only access the memory location that it has permission to execute. Therefore it improves a security of the program.

5) **Interoperability**

Interoperability process enables the C# programs to do almost anything that a native C++ application can do.

6) **Scalable and Updateable**

C# is automatic scalable and updateable programming language. For updating our application we delete the old files and update them with new ones.

7) **Component Oriented**

C# is component oriented programming language. It is the predominant software development methodology used to develop more robust and highly scalable applications.

8)Structured Programming Language

C# is a structured programming language in the sense that we can break the program into parts using functions. So, it is easy to understand and modify.

9) **Rich Library**

C# provides a lot of inbuilt functions that makes the development fast.

10) **Fast Speed**:The compilation and execution time of C# language is fast.

C# Variable

A variable is a name of memory location. It is used to store data. Its value can be changed and it can be reused many times.

It is a way to represent memory location through symbol so that it can be easily identified.

variable Type	Example
Nullable types	Nullable data types
Boolean types	True or false value, as assigned
Integral types	int, char, byte, short, long
Floating point types	float and double

1. type variable_list;

The example of declaring variable is given below:

1. int i, j;
2. double d;

1. float f;
2. char ch;

Here, i, j, d, f, ch are variables and int, double, float, char are data types.

We can also provide values while declaring the variables as given below:

1. int i=2,j=4; //declaring 2 variable of integer type
2. float f=40.2;
3. char ch='B';

Rules for defining variables

1. A variable can have alphabets, digits and underscore.
2. A variable name can start with alphabet and underscore only. It can't start with digit.
3. No white space is allowed within variable name.
4. A variable name must not be any reserved word or keyword e.g. char, float etc.
5. int _x;

1. int x;
2. int k20;

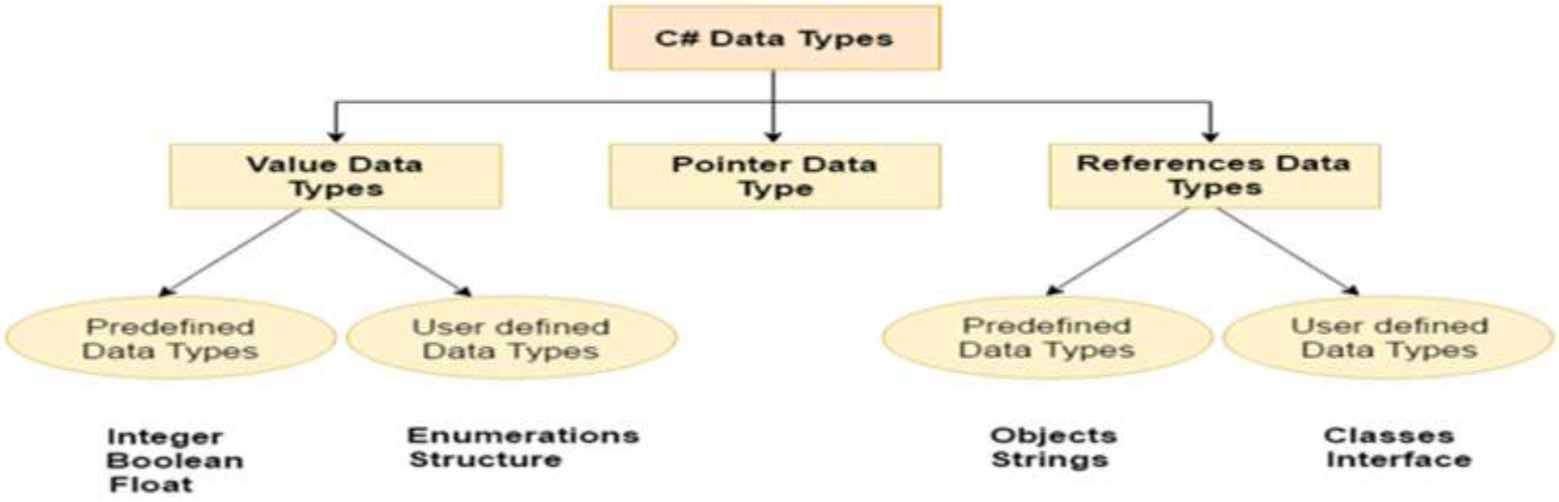
Invalid variable names:

1. int 4;
2. int x y;

int double;

C# Data Types

A data type specifies the type of data that a variable can store such as integer, floating, character etc.



There are 3 types of data types in C# language

Types	Data Types
Value Data Type	**short, int, char, float, double etc**
Reference Data Type	**String, Class, Object and Interface**
Pointer Data Type	**Pointers**

Value Data Type

The value data types are integer-based and floating-point based. C# language supports both signed and unsigned literals.

There are 2 types of value data type in C# language.

1)Predefined Data Types - such as Integer, Boolean, Float, etc.

2) User defined Data Types - such as Structure, Enumerations, etc.

The memory size of data types may change according to 32 or 64 bit operating system.

Data Types	Memory Size
Char	1 byte
signed char	1 byte
unsigned char	1 byte
Short	2 byte
signed short	2 byte
unsigned short	2 byte
Int	4 byte
signed int	4 byte
unsigned int	4 byte
Long	8 byte
signed long	8 byte
unsigned long	8 byte

Float	4 byte
Double	8 byte
Decimal	16 byte

Reference Data Type

The reference data types do not contain the actual data stored in a variable, but they contain a reference to the variables.

If the data is changed by one of the variables, the other variable automatically reflects this change in value.

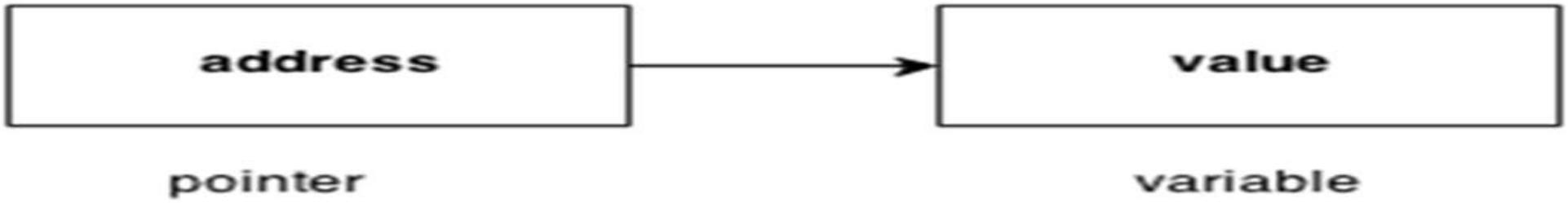
There are 2 types of reference data type in C# language.

1) Predefined Types - such as Objects, String.

User defined Types - such as Classes, Interface.

pointer Data Type

The pointer in C# language is a variable, it is also known as locator or indicator that points to an address of a value.



Symbols used in pointer

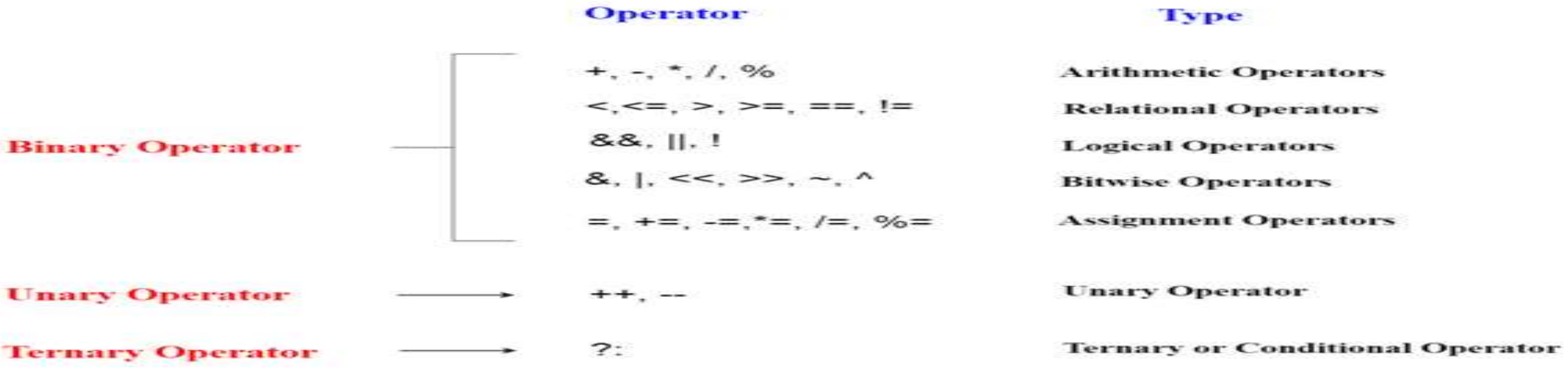
Symbosymbol	Namenname	Description
&amersand sign)	Addresaddress operator	Determine the address of a variable.
* (ast&* astrike sign	Indirection operator	Access the value of an address.

C# operators

An operator is simply a symbol that is used to perform operations. There can be many types of operations like arithmetic, logical, bitwise etc.

There are following types of operators to perform different types of operations in C# language.

- Arithmetic Operators
- Relational Operators
- Logical Operators
- Bitwise Operators
- Assignment Operators
- Unary Operators
- Ternary Operators
- Misc Operators



Precedence of Operators in C#

The precedence of operator specifies that which operator will be evaluated first and next. The associativity specifies the operators direction to be evaluated, it may be left to right or right to left.

1. int data= 10+ 5*5

The "data" variable will contain 35 because * (multiplicative operator) is evaluated before + (additive operator).

the precedence and associativity of C# operators is given below:

Category (By Precedence)	Operator(s)
Unary	+ - ! ~ ++ -- (type)* & sizeof
Additive	+ -
Multiplicative	% / *
Relational	< > <= >=
Shift	<< >>
Equality	== !=
Logical AND	&
Logical OR	|
Logical XOR	^
Conditional OR	||
Conditional AND	&&
Null Coalescing	??

C# Arrays

Arrays are used to store multiple values in a single variable, instead of declaring separate variables for each value.

To declare an array, define the variable type with square brackets:

```
string[] cars;
```

We have now declared a variable that holds an array of strings.

To insert values to it, we can use an array literal - place the values in a comma-separated list, inside curly braces:

```
string[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
```

To create an array of integers, you could write:

```
int[] myNum = {10, 20, 30, 40};
```

Access the Elements of an Array

You access an array element by referring to the index number.

This statement accesses the value of the first element in cars:

Example

string[] cars = {"Volvo", "BMW", "Ford", "Mazda"}; Console.WriteLine(cars[0]); // Outputs Volvo

Change an Array Element

To change the value of a specific element, refer to the index number:

Example

cars[0] = "Opel";

Example

string[] cars = {"Volvo", "BMW", "Ford", "Mazda"};

cars[0] = "Opel";

Console.WriteLine(cars[0]); // Now outputs Opel instead of Volvo

Control Statements

C# - Loops

Looping in a programming language is a way to execute a statement or a set of statements multiple times depending on the result of the condition to be evaluated to execute statements. The result condition should be true to execute statements within loops.

Loops are mainly divided into two categories:

1.Entry Controlled Loops

2.Exit Controlled Loops

Entry Controlled Loops

The loops in which condition to be tested is present in beginning of loop body are known as **Entry Controlled Loops**. **while loop** and **for loop** are entry controlled loops. Entry Controlled Loops are one of the most commonly used loops in Java. Let us learn them.

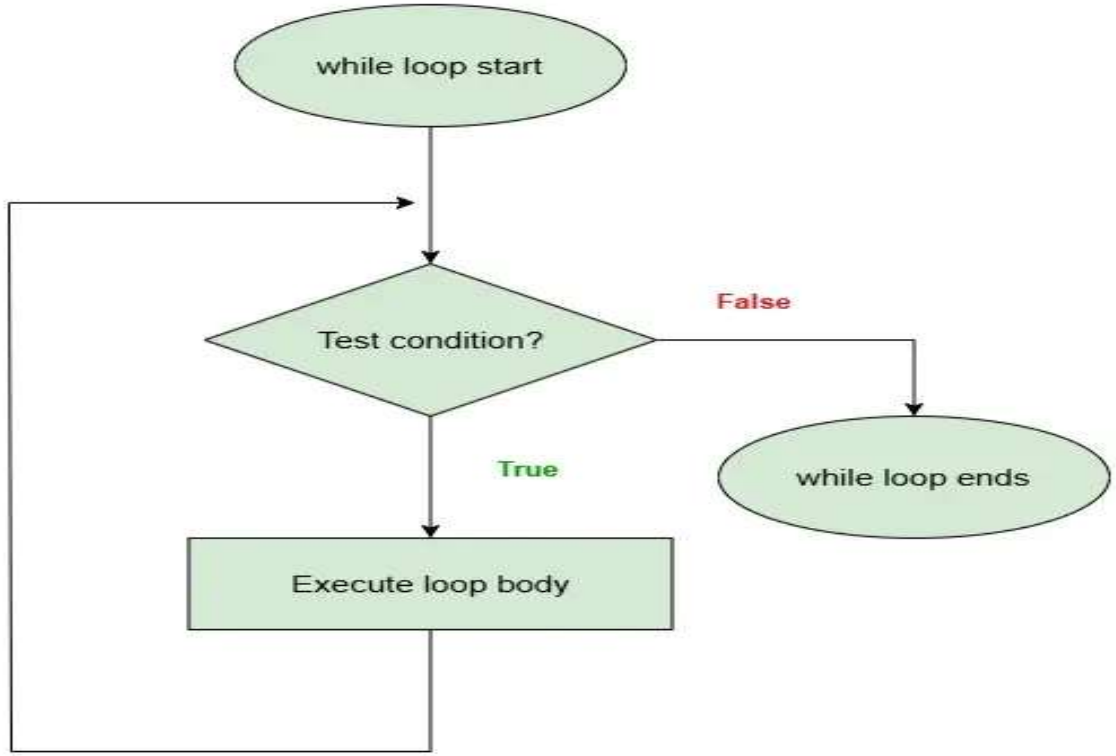
while loop

The test condition is given in the beginning of the loop and all statements are executed till the given Boolean condition satisfies when the condition becomes false, the control will be out from the while loop.

Syntax:

```
while (boolean condition)
{
loop statements…
}
```

Flowchart:



```
// Exit when x becomes greater than 4
        while (x <= 4)
        {
            Console.WriteLine("GeeksforGeeks");
             // Increment the value of x for
            // next iteration
            x++;
// C# program to illustrate while loop
using System;
class Geeks
{
    public static void Main()
    {
        int x = 1;
           }
    }
}
```

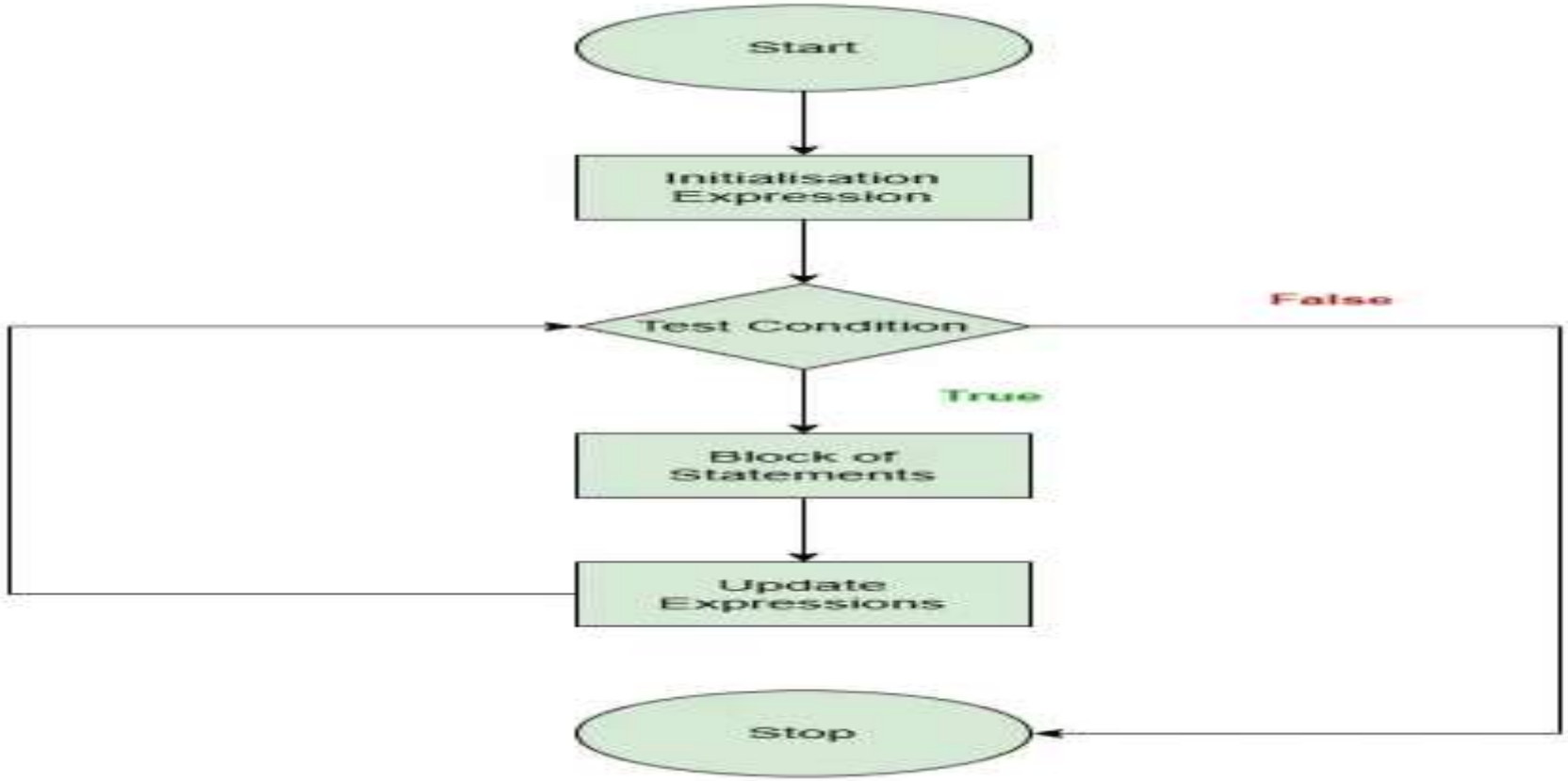
Output

GeeksforGeeks
GeeksforGeeks
GeeksforGeeks
GeeksforGeeks

2. for loop

for loop has similar functionality as while loop but with different syntax. for loops are preferred when the number of times loop statements are to be executed is known beforehand. The loop variable initialization, condition to be tested, and increment/decrement of the loop variable is done in one line in for loop thereby providing a shorter, easy to debug structure of looping.

```
for (loop variable initialization ; testing condition; increment / decrement)
{
// statements to be executed
}
```



•**Initialization of loop variable:** Th expression / variable controlling the loop is initialized here. It is the starting point of for loop. An already declared variable can be used or a variable can be declared, local to loop only.

•**Testing Condition:** The testing condition to execute statements of loop. It is used for testing the exit condition for a loop. It must return a boolean value **true or false**. When the condition became false the control will be out from the loop and for loop ends.

Exit Controlled Loops

The loops in which the testing condition is present at the end of loop body are termed as Exit Controlled Loops.

Note: In Exit Controlled Loops, loop body will be evaluated for at-least one time as the testing condition is present at the end of loop body.

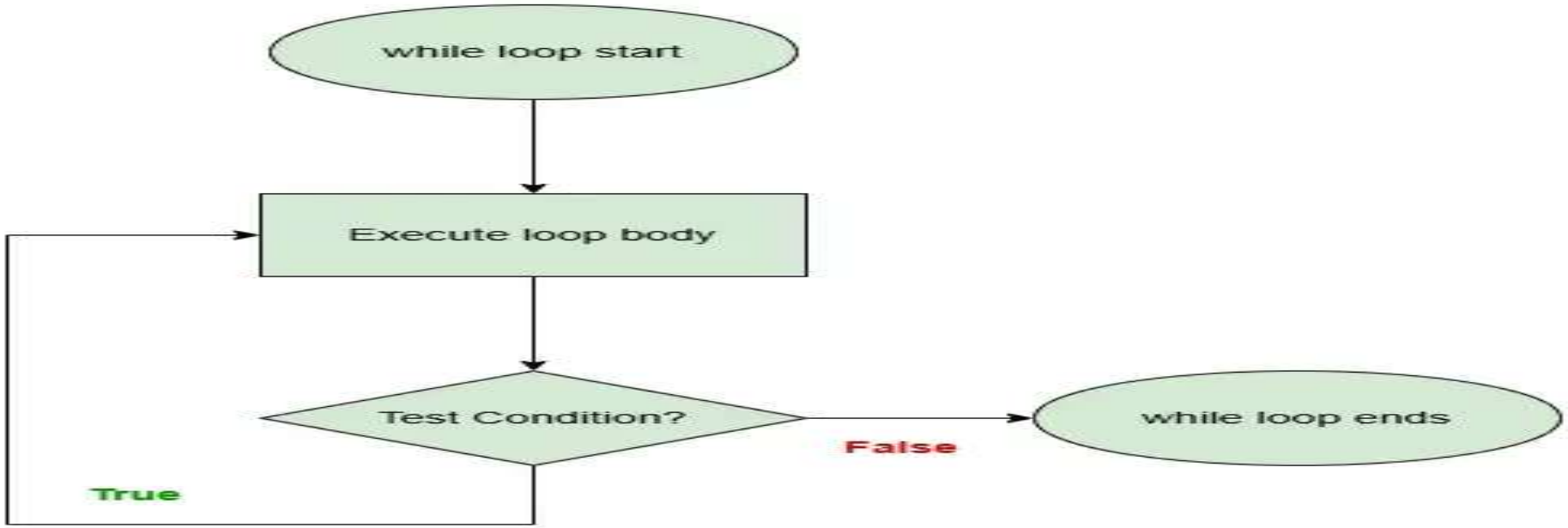
1. **do-while loop**

do while loop is similar to while loop with the only difference that it checks the condition after executing the statements, i.e it will execute the loop body one time for sure because it checks the condition after executing the statements.

Syntax :

```
do
{
    statements..
} while (condition);
```

Flowchart:




```
// Using do-while loop
using System;

class Geeks
{
    public static void Main()
    {
        int x = 21;
```

```
do
        {
            // The line will be printed even
            // if the condition is false
            Console.WriteLine("GeeksforGeeks");
            x++;
        }
        while (x < 20);
    }
}
```

Output

GeeksforGeeks

2. Infinite Loops

The loops in which the test condition does not evaluate false ever tend to execute statements forever until an external force is used to end it and thus they are known as infinite loops.

```
// Demonstration of infinite loop
using System;

class Geeks
{
    public static void Main()
    {
        // The statement will be printed
        // infinite times
        for(;;)
            Console.WriteLine("This is printed infinite times");
    }
}
```

Output:

```
This is printed infinite times
This is printed infinite times
This is printed infinite times
This is printed infinite times
This is printed infinite times
This is printed infinite times
This is printed infinite times
.....
```

3. Nested Loops

When loops are present inside the other loops, it is known as nested loops.

Example:

```
// Using nested loops
using System;

class Geeks
{
    public static void Main()
    {
        // loop within loop printing GeeksforGeeks
        for (int i = 2; i < 3; i++)
            for (int j = 1; j < i; j++)
                Console.WriteLine("GeeksforGeeks");
    }
}
```

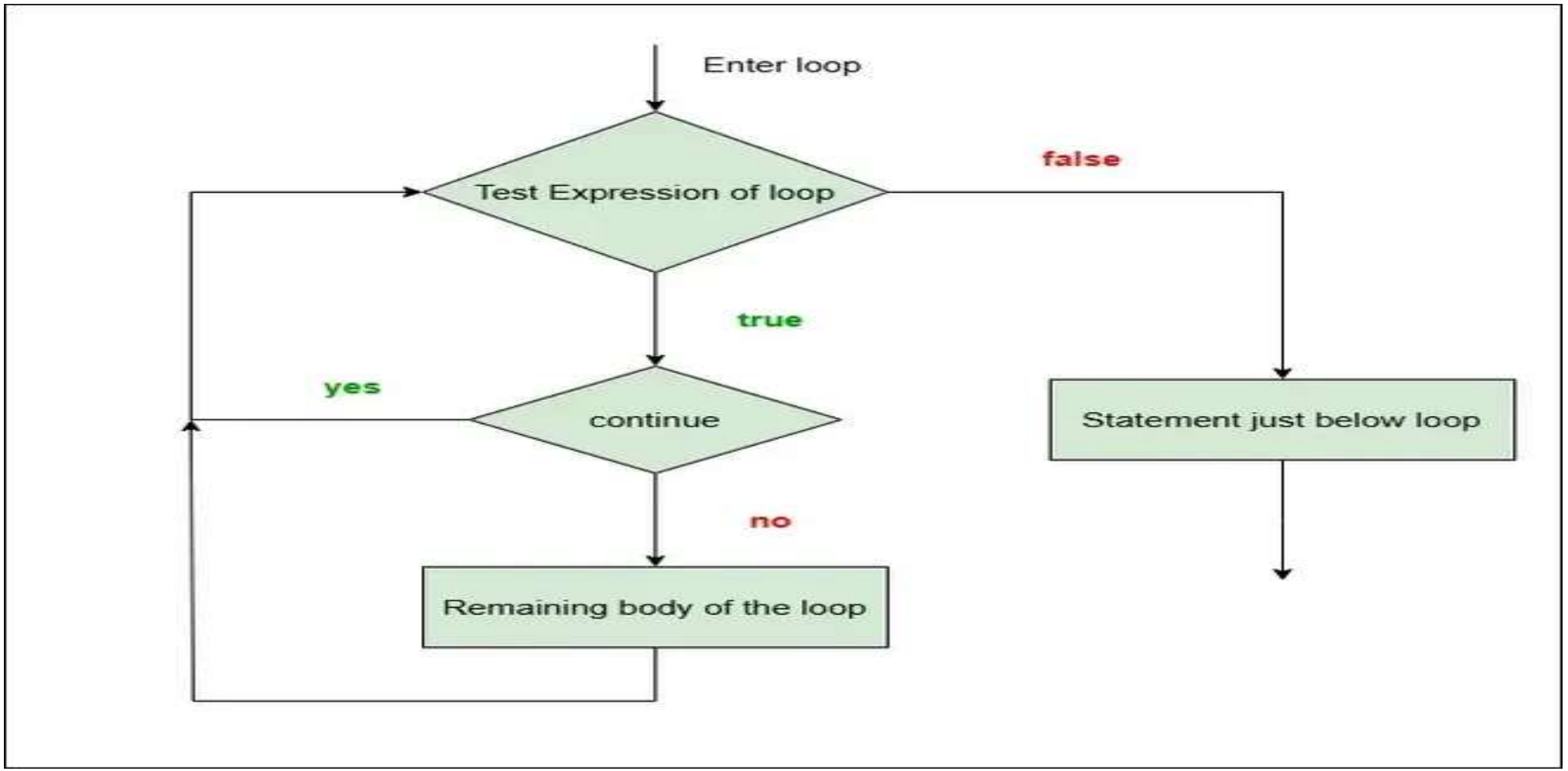
Output

```
GeeksforGeeks
```

4. Continue Statement

Continue statement is used to skip over the execution part of loop on a certain condition and move the flow to next updation part.

Flowchart:



```
// Using continue statement
using System;

class Geeks
{
    public static void Main()
    {
        // GeeksforGeeks is printed only 2 times
        // because of continue statement
        for (int i = 1; i < 3; i++)
        {
            if (i == 2)
                continue;

            Console.WriteLine("GeeksforGeeks");
        }
    }
}
```

Output

GeeksforGeeks

Introduction of Object Oriented Programming

As the name suggests, Object-Oriented Programming or OOPs refers to languages that use objects in programming. Object-oriented programming aims to implement real-world entities like inheritance, hiding, polymorphism, etc in programming. The main aim of OOP is to bind together the data and the functions that operate on them so that no other part of the code can access this data except that function.

OOPs Concepts:

- Class
- Objects
- Data Abstraction
- Encapsulation
- Inheritance

•Polymorphism

•Dynamic Binding

•Message Passing

1. Class:

A class is a user-defined data type. It consists of data members and member functions, which can be accessed and used by creating an instance of that class. It represents the set of properties or methods that are common to all objects of one type. A class is like a blueprint for an object.
For Example: Consider the Class of Cars. There may be many cars with different names and brands but all of them will share some common properties like all of them will have 4 wheels, Speed Limit, Mileage range, etc. So here, Car is the class, and wheels, speed limits, mileage are their properties.

2. Object:

It is a basic unit of Object-Oriented Programming and represents the real-life entities. An Object is an instance of a Class. When a class is defined, no memory is allocated but when it is instantiated (i.e. an object is created) memory is allocated. An object has an identity, state, and behavior. Each object contains data and code to manipulate the data. Objects can interact without having to know details of each other's data or code, it is sufficient to know the type of message

3. **Data Abstraction:**

Data abstraction is one of the most essential and important features of object-oriented programming. Data abstraction refers to providing only essential information about the data to the outside world, hiding the background details or implementation. Consider a real-life example of a man driving a car. The man only knows that pressing the accelerators will increase the speed of the car or applying brakes will stop the car, but he does not know about how on pressing the accelerator the speed is increasing, he does not know about the inner mechanism of the car or the implementation of the accelerator, brakes, etc in the car. This is what abstraction is.

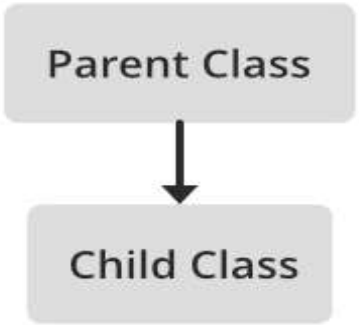
4. **Encapsulation:**

Encapsulation is defined as the wrapping up of data under a single unit. It is the mechanism that binds together code and the data it manipulates. In Encapsulation, the variables or data of a class are hidden from any other class and can be accessed only through any member function of their class in which they are declared. As in encapsulation, the data in a class is hidden from other classes, so it is also known as data-hiding.

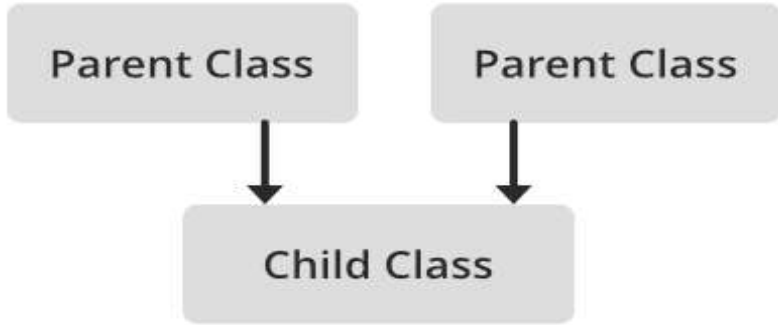
Consider a real-life example of encapsulation, in a company, there are different sections like the accounts section, finance section, sales section, etc. The finance section handles all the financial transactions and keeps records of all the data related to finance. Similarly, the sales section handles all the sales-related activities and keeps records of all the sales. Now there may arise a situation when for some reason an official from the finance section needs all the data about sales in a particular month. In this case, he is not allowed to directly access the data of the sales section. He will first have to contact some other officer in the sales section and then request him to give the particular data. This is what encapsulation is. Here the data of the sales section and the employees that can manipulate them are wrapped under a single name “sales section”.

5. **Inheritance:**

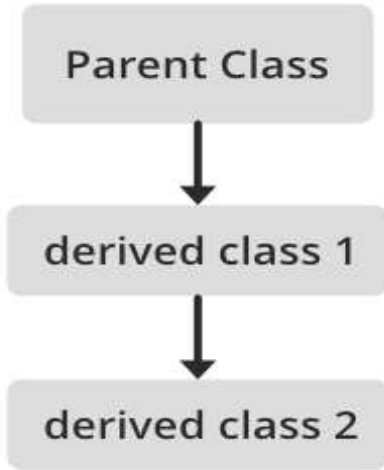
Inheritance is an important pillar of OOP(Object-Oriented Programming). The capability of a class to derive properties and characteristics from another class is called Inheritance. When we write a class, we inherit properties from other classes. So when we create a class, we do not need to write all the properties and functions again and again, as these can be inherited from another class that possesses it. Inheritance allows the user to reuse the code whenever possible and reduce its redundancy.



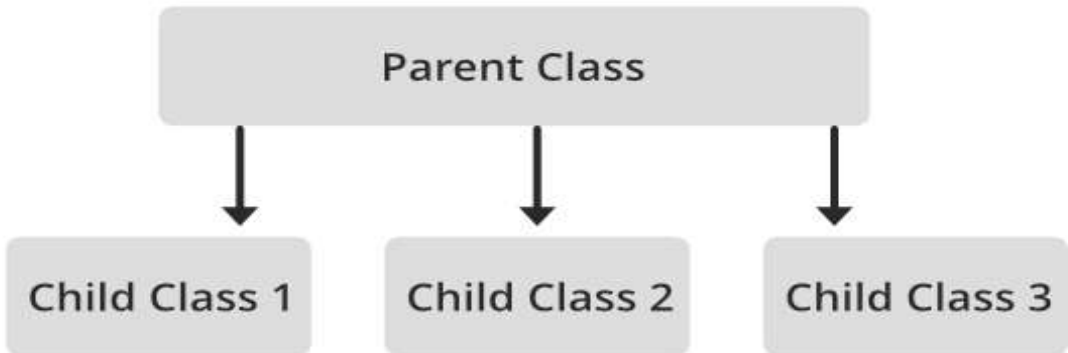
Single inheritance



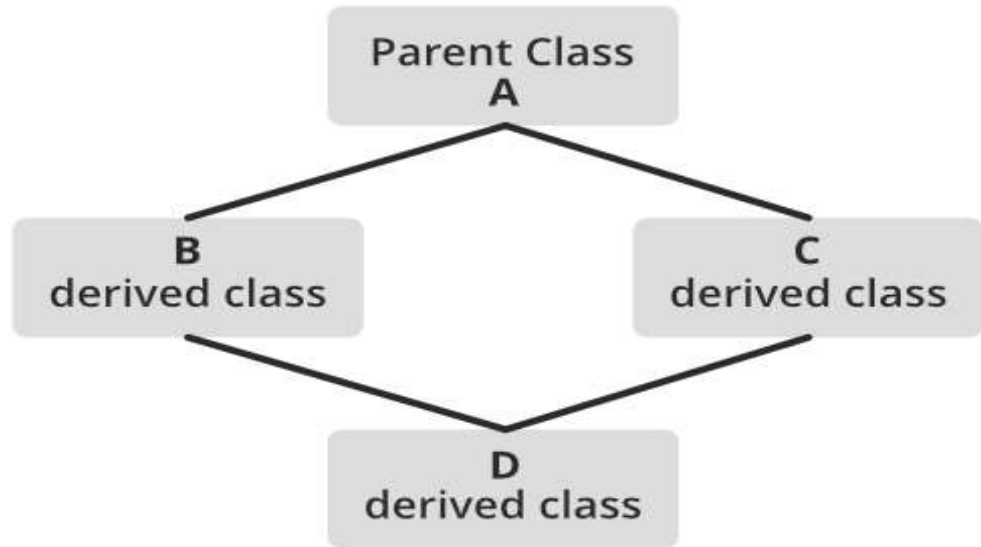
Multiple inheritance



Multilevel inheritance



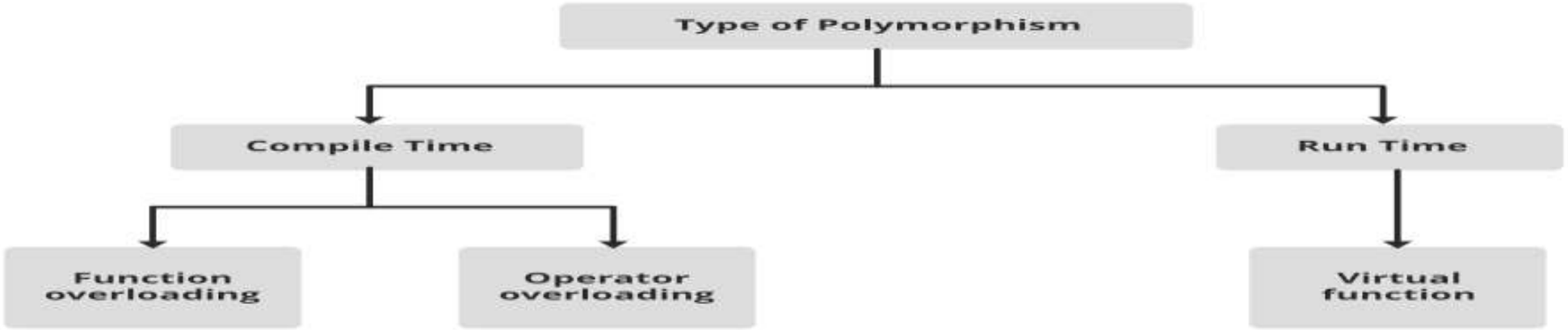
Hierarchical inheritance



Hybrid Class

6. Polymorphism:

The word polymorphism means having many forms. In simple words, we can define polymorphism as the ability of a message to be displayed in more than one form. For example, A person at the same time can have different characteristics. Like a man at the same time is a father, a husband, an employee. So the same person posses different behavior in different situations. This is called polymorphism.



7. Dynamic Binding:

In dynamic binding, the code to be executed in response to the function call is decided at runtime. Dynamic binding means that the code associated with a given procedure call is not known until the time of the call at run time. Dynamic Method Binding One of the main advantages of inheritance is that some derived class D has all the members of its base class B. Once D is not hiding any of the public members of B, then an object of D can represent B in any context where a

8. **Message Passing:**

It is a form of communication used in object-oriented programming as well as parallel programming. 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.

Why do we need object-oriented programming

To make the development and maintenance of projects more effortless.

To provide the feature of data hiding that is good for security concerns.

We can solve real-world problems if we are using object-oriented programming.

It ensures code reusability.

It lets us write generic code: which will work with a range of data, so we don’t have to write basic stuff over and over again.