**Chapter-1: Introduction about basic programming and C++**

**Different types of Programming language.**

1. **Low level programming Language**
2. **Machine Language:**

Machine language is a type of low-level programming language. Machine language is easier to read for computer system because it is normally displayed in binary form. It does not require a translator to convert the programs because computers directly understand the machine language programs.

1. **Assembly Language**

Assembly language (ASM) is also a type of low-level programming language that is designed for some specific processors (8085, 8086, 8255 processor) which represents the set of instructions to complete a specific task.

1. **High-level programming language**

High-level programming language (HLL) is designed for **developing user-friendly software programs and websites.** This programming language requires a compiler or interpreter to translate the program into machine language (execute the program).

The main advantage of a high-level language is that it is **easy to read, write, and maintain.**

Examples of high level programming language are C, Pascal, C++, JAVA, Python PHP etc.

1. **Procedural Programming Language**

Procedural programming language can be defined as a programming model which is based on structured programming that means it consists of a series of computational steps to execute the instructions which follow top-down approch. Examples of some procedural programming language are C, Pascal etc.

1. **Objected oriented Programming Language**

Objected oriented Programming Language can be defined as a programming model which is based upon the concept of objects and class. Objects define the basic run time entities and class is a collection of objects. Examples of some objected oriented programming language are C++, JAVA, Python etc.

**How different object in OOPs communicate to each other**

**Class-B**

**Class-A**

**Data member**

**Data member**

**Member Function**

**Member Function**

**Data member**

**Class-C**

**Member Function**

**What is compiler?**

A compiler is computer software which translates high level programming language into machine code or low-level language i.e program to binary (1’s and 0’s) which is understand by a system. It analyses the entire program and converts it into machine code. The working of a compiler can be categorized into the following phases:

* **Lexical analysis**: Splitting of source code generate tokens t identify whether it is a keyword, a string, or some other variable.
* **Syntax Analysis**: It responsible to check the errors in the program syntax.
* **Semantic Analysis**: It responsible to check the wrong variable assigned, using an undeclared variable, using keywords as variable, etc.
* **Intermediate Code Generation**: The process of compilation generates two or more intermediate code forms.
* **Optimization**: The compilation process looks for multiple ways through which the task can be enhanced.
* **Code generation**: The compiler converts the intermediate optimized code into a machine code after which the source program is converted to an object program.

## What is Interpreter?

An interpreter works more or less similar to a compiler. The only difference between their working is that the interpreter does not generate any intermediate code forms, reads the program line to line checking for errors, and runs the program simultaneously.

* It is responsible for translating the program line by line.
* Interpreters are smaller in size.
* Allows modification of program while executing.
* Relatively lesser time is consumed for analysis as it runs line by line.
* Execution of the program is relatively slow as analysis takes place every time the program is run.

# What is a Scripting Language?

Scripting language is a computer language that does not need the compilation step and is rather interpreted one by one at runtime. Scripting languages contain different libraries, which help to create new applications in web browsers. So they are different from normal programming languages. It directly executes the files without needing compilation. There are two types of scripting languages. Python, PHP, and Perl are examples of server-side scripting languages, while JavaScript is the greatest example of a client-side scripting language.

## Server-side Scripting

It runs on the back end web server. It helps provide a response to every request that comes in from the user/client which is not visible to the client side of the application. Example: PHP, ASP.net, Python, Ruby on Rails.

## Client-side Scripting

It helps work with the front end which is visible to user and run on client browser. Example: HTML, CSS, JavaScript.

NOTE: JavaScript JavaScript provides facility to validate the form on the client-side so that data processing will be faster than server-side validation. We are going to validate the email, date, mobile numbers, name and password etc. The name can’t be empty and password can’t be less than 6 characters long.

**What is C++?**

C++ programming gives a clear idea about Object Oriented Programming. It is an extension of C, it is also known as C with classes. C++ is the most widely used programming languages in application and system programming.

**Difference between C and C++**

|  |  |
| --- | --- |
| **C** | **C++** |
| C was developed by Dennis Ritchie in 1972 1973 at AT&T Bell Labs. | C++ was developed by Bjarne Stroustrup in 1979 |
| C follows the top-down approach | C++ follows the Bottom-up approach |
| For the development of code, C supports [procedural programming](https://www.geeksforgeeks.org/introduction-of-programming-paradigms/). | C++ is known as hybrid language because C++ supports both [procedural](https://www.geeksforgeeks.org/introduction-of-programming-paradigms/) and [object oriented programming paradigms](https://www.geeksforgeeks.org/introduction-of-programming-paradigms/).. |
| C does no support polymorphism, encapsulation, and inheritance | C++ supports [polymorphism](https://www.geeksforgeeks.org/polymorphism-in-c/), [encapsulation](https://www.geeksforgeeks.org/encapsulation-in-c/), and [inheritance](https://www.geeksforgeeks.org/inheritance-in-c/) |
| C does not support information hiding. | C++ support information hiding using encapsulation. |
| Function and operator overloading is not supported in C. | Function and operator overloading is supported by C++ |
| Namespace features are not present inside the C. | Namespace features are present inside the C++. |
| Standard IO header is [stdio.h](https://www.geeksforgeeks.org/whats-difference-between-and/) | Standard IO header is [iostream.h](https://www.geeksforgeeks.org/basic-input-output-c/) |
| Virtual and friend functions are not supported by C | [Virtual](https://www.geeksforgeeks.org/virtual-function-cpp/) and [friend functions](https://www.geeksforgeeks.org/friend-class-function-cpp/) are supported by C++. |
| C provides [malloc()](https://www.geeksforgeeks.org/dynamic-memory-allocation-in-c-using-malloc-calloc-free-and-realloc/) and [calloc()](https://www.geeksforgeeks.org/dynamic-memory-allocation-in-c-using-malloc-calloc-free-and-realloc/) functions for [dynamic memory allocation](https://www.geeksforgeeks.org/dynamic-memory-allocation-in-c-using-malloc-calloc-free-and-realloc/), and [free()](https://www.geeksforgeeks.org/dynamic-memory-allocation-in-c-using-malloc-calloc-free-and-realloc/) for memory de-allocation. | C++ provides [new operator](https://www.geeksforgeeks.org/new-and-delete-operators-in-cpp-for-dynamic-memory/) for memory allocation and [delete operator](https://www.geeksforgeeks.org/new-and-delete-operators-in-cpp-for-dynamic-memory/) for memory de-allocation. |
| [scanf()](https://www.geeksforgeeks.org/scanf-and-fscanf-in-c-simple-yet-poweful/) and printf() functions are used for input/output in C | [cin and cout](https://www.geeksforgeeks.org/basic-input-output-c/) are used for [input/output in C++](https://www.geeksforgeeks.org/basic-input-output-c/) |
| File extension is “.c” | File extension is “.cpp” |
| There are 32 keywords in the C | There are 97 keywords in the C++ |

## Why Do You Need Object-Oriented Programming?

In object-oriented programming, it is easy to maintain the code with the help of classes and objects. Using inheritance, there is code reusability, i.e., you don’t have to write the same code again and again, which increases the simplicity of the program. Concepts like encapsulation and abstraction provide data hiding as well.

## Advantages of C++ over other Languages

### 1. Object-Oriented

C++ is an [object-oriented programming language](https://www.simplilearn.com/tutorials/cpp-tutorial/oops-concepts-in-cpp) which means that the main focus is on objects and manipulations around these objects.

### 2. Speed

The compilation and execution time of a[C++ program](https://www.simplilearn.com/tutorials/cpp-tutorial/cpp-basics) is much faster than most general-purpose[programming languages.](https://www.simplilearn.com/best-programming-languages-start-learning-today-article)

### 3. Rich Library Support

The C++ Standard Template Library (STL) has many functions available to help write code quickly.

### 4. Pointer Support

C++ also supports pointers which are often not available in other objected oriented programming languages.

### 5. Closer to Hardware

C++ is closer to hardware than most general-purpose programming languages. This makes it very useful in those areas where hardware and software are closely coupled together.

**Application of C++ programming language.**

1. **Developing an operating System.**

2. **Game Development.**

3. **GUI based application**.

4. **Web Browsers:** Mozilla Firefox is completely developed from C++. Google applications like Chrome and Google File System are partly written in C++.

**C++ Program to add 2 number without using class and object concept**

#include <iostream>

using namespace std;

int main()

{

int num1, num2, sum;

cout << "Enter two numbers: ";

cin >> num1 >> num2;

sum = num1 + num2;

cout << "The sum of " << num1 << " and " << num2 << " is: " << sum << endl;

return 0;

}

* Namespace provide the space where we can define or declare identifier i.e. variable,  method, classes.

**Advantages of Namespace**

* Example, you might be writing some code that has a function called xyz() and there is another library available which is also having same function xyz(). Now the compiler has no way of knowing which version of xyz() function you are referring to within your code.
* A namespace is designed to overcome this difficulty and is used as additional information to differentiate similar functions, classes, variables etc.

**Namespace also define some other things as follows:**

#include <iostream>

<here we not write down namespace std>

namespace name1

{

int age=25;

string name=”manab”;

}

namespace name2

{

int age=26;

}

int main()

{

cout<<name2::age<<endl;

cout<<name1:: name<<endl;

}

//here output must produce 3 errors for 3 things (endl, cout and string) because these 3 things are not identified as these 3 things defined within namespace std.

**Now how to resolve the above problem:**

#include <iostream>

namespace name1

{

int age=25;

std::string name=”manab”;

}

namespace name2

{

int age=26;

}

int main()

{

std::cout<<name2::age<<std::endl;

std::cout<<name1:: name<<std::endl;

}

OR

#include <iostream>

using std::cout;

using namespace std;

using std::endl;

using std::string;

namespace name1

{

int age=25;

string name=”manab”;

}

namespace name2

{

int age=26;

}

int main()

{

cout<<name2::age<<endl;

cout<<name1:: name<<endl;

}

**Why we use return 0 in a program:**

In C++ the return value of the main function is considered the "Exit Status" of the application. Also use of return in main function define the program execute successfully. If we not write down return 0 C++ by default take return 0. Also some cases return 0 means it return false and return 1 means it return true value (following Example).

#include <iostream>

using namespace std;

int checkAge(int age)

{

if (age >= 18)

return 1;

else

return 0;

}

void check(int age)

{

if (checkAge(age))

cout << "You are an adult\n";

else

cout << "You are not an adult\n";

}

int main()

{

int age = 15;

check(age);

return 0;

}

**C++ Program General Syntax based on class and object**

<Declare header file>

**using namespace std;**

class <class name>

{

**Private:** // Access specifies

<Data type> <List of variable>

**public:** // Access specifies

void fun1()

{

Body of the function

}

void fun2()

{

Body of the function

}

};

int main()

{

<class name> <object name>;

<object name>.fun1();

<object name>.fun2();

return 0;

}

**C++ Program to add 2 number using class and object concept (Using namespace std)**

#include <iostream>

using namespace std;

class Addition

{

private:

int num1, num2, sum;

public:

void input()

{

cout << "Enter two numbers: ";

cin >> num1 >> num2;

}

void calculate()

{

sum = num1 + num2;

}

void output()

{

cout << "The sum of " << num1 << " and " << num2 << " is " << sum << endl;

}

};

int main()

{

Addition add;

add.input();

add.calculate();

add.output();

return 0;

}

**//Without using namespace std before start the program**

#include <iostream>

class Addition

{

private:

int num1, num2, sum;

public:

void input()

{

std::cout << "Enter two numbers: ";

std::cin >> num1 >> num2;

}

void calculate()

{

sum = num1 + num2;

}

void output()

{

std::cout << "The sum of " << num1 << " and " << num2 << " is " << sum << std::endl;

}

};

int main()

{

Addition add;

add.input();

add.calculate();

add.output();

return 0;

}

**//factorial of a given number using object and class**

#include<iostream>

using namespace std;

class factorial

{

public:

long int fact=1;

int num;

int input();

void fact\_function();

void display();

};

int factorial::input()

{

cout<<"Please enter a number: ";

cin>>num;

}

void factorial::fact\_function()

{

for(int i=1;i<=num;i++)

{

fact=fact\*i;

}

}

void factorial::display()

{

cout<<"Factorial of entered number is: "<<fact;

}

int main()

{

factorial object;

object.input();

object.fact\_function();

object.display();

}

**What is Object?**

It is a basic unit of Object-Oriented Programming and represents the real life entities.  A typical Java program creates many objects to interact by invoking methods.

**Object** is an instance of a class or Objects are nothing but a basic run time entity which defines the states and behaviors of the entity. These real-world objects share two characteristics. For example any person consists of two characteristic one is state (that means height, chest ,color etc.) and another one is behavior (that means dancer, singer etc.).

**An object consists of following things:**

1. **State**: It is represented by attributes of an object. It also reflects the properties of an object.
2. **Behavior**: It is represented by methods of an object. It also reflects the response of an object with other objects.
3. **Identity**: It gives a unique name to an object and enables one object to interact with other objects.

Behavior

* dancer
* singer

State

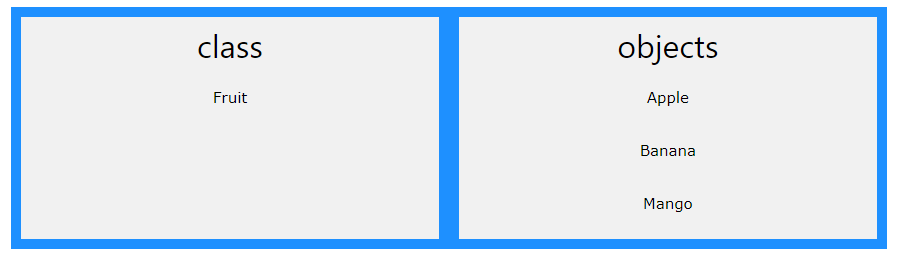
* height
* chest
* age

Identity

Name of the person

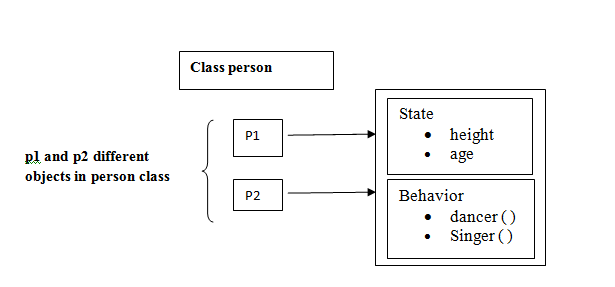
**What is class?**

**Class** is a basic concept of Object-Oriented Programming which defines the real-life entities. Class in OOPs determines how an object will behave. A class is a user defined blueprint or prototype from which objects are created.  It represents the set of properties that are common to all objects of one type.



In general, class declarations can include these components, in order:

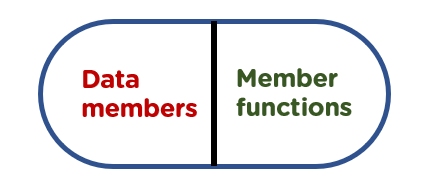
1. **Modifiers**: A class can be public, private or has protected access.
2. **Class keyword:**class keyword is used to create a class.
3. **Class name:** The name should begin with an initial letter (capitalized by convention).
4. **Super class (if any):** The name of the class’s parent (superclass), if any, preceded by the keyword extends. A class can only extend (subclass) one parent.
5. **Body:** The class body surrounded by braces, { }.



### Figure-1: multiple object created for a particular class

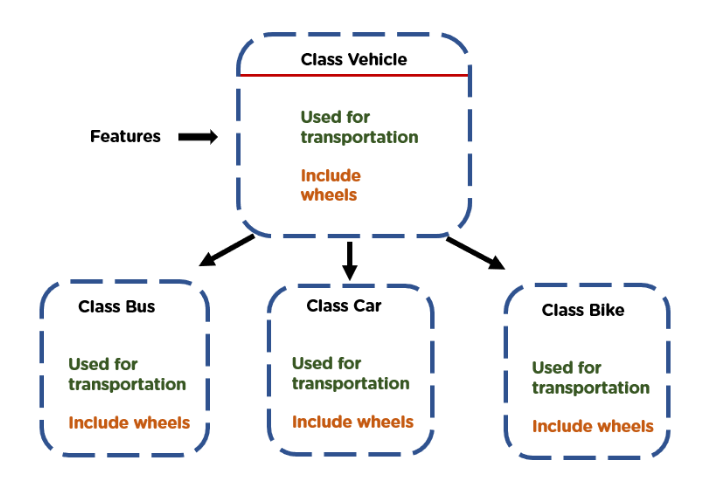
### Encapsulation

The wrapping up of data and functions together in a single unit is known as encapsulation. It can be achieved by making the data members' scope private and the member function’s scope public to access these data members. Encapsulation makes the data non-accessible to the outside world.



### Inheritance

[Inheritance](https://www.simplilearn.com/tutorials/cpp-tutorial/inheritance-in-cpp) is the process in which two classes have an is-a relationship among each other and objects of one class acquire properties and features of the other class. The class which inherits the features is known as the child class, and the class whose features it inherited is called the parent class. For example, Class Vehicle is the parent class, and Class Bus, Car, and Bike are child classes.



### Polymorphism

[Polymorphism](https://www.simplilearn.com/tutorials/cpp-tutorial/polymorphism-in-cpp) means many forms. It is the ability to take more than one form. It is a feature that provides a function or an operator with more than one definition. It can be implemented using function overloading, operator overload, [function overriding](https://www.simplilearn.com/tutorials/cpp-tutorial/function-overriding-in-cpp), virtual function.

1. **C++ program for linear search (without class and object)**

#include<iostream>

using namespace std;

int main()

{

int arr[10], i, num, index,n;

cout<<"Enter number less than 10: ";

cin>>n;

for(i=0; i<n; i++)

cin>>arr[i];

cout<<"\nEnter a Number to Search: ";

cin>>num;

for(i=0; i<n; i++)

{

if(arr[i]==num)

{

index = i+1;

break;

}

}

cout<<"\nFound at Index No."<<index;

cout<<endl;

return 0;

}

1. **C++ program for linear search (with class and object concept)**

#include<iostream>

using namespace std;

class linearsearch

{

public:

int arr[10], i, num, index,n;

void input()

{

cout<<"Enter Numbers less than 10: ";

cin>>n;

for(i=0; i<n; i++)

cin>>arr[i];

}

void search()

{

cout<<"\nEnter a Number to Search: ";

cin>>num;

for(i=0; i<n; i++)

{

if(arr[i]==num)

{

index = i;

break;

}

}

}

void display()

{

cout<<"\nFound at Index No."<<index+1;

cout<<endl;

}

};

int main()

{

linearsearch ob1;

ob1.input();

ob1.search();

ob1.display();

return 0;

}

1. **C++ Program for binary search (Without class object concept)**

#include <iostream>

using namespace std;

int main ()

{

int arr[100], st, mid, end, i, num, tgt;

cout << " Define the size of the array: " << endl;

cin >> num;

cout << " Enter the values in sorted array ascending order : " << endl;

for (i = 0; i < num; i++)

{

cout << " arr [" << i << "] = ";

cin >> arr[i];

}

st = 0;

end = num - 1;

cout << " Define a value to be searched from sorted array: " << endl;

cin >> tgt;

while ( st <= end)

{

mid = ( st + end ) / 2;

if (arr[mid] == tgt)

{

cout << " Element is found at index " << (mid + 1);

exit(0); // use for exit the program

}

else if ( tgt > arr[mid])

{

st = mid + 1;

}

else if ( tgt < arr[mid])

{

end = mid - 1;

}

}

cout << " Number is not found. " << endl;

return 0;

}