* What is c++ , why c++
* What is procedure language and functional language?
* Why C++ is a mid-level programming language

<https://www.naukri.com/code360/guided-paths/basics-of-c/content/118623/offering/1377972?leftPanelTabValue=PROBLEM&customSource=studio_nav>

what is header files? Why use?

**iostream** stands for Input/Output stream, meaning this header file is necessary to take input through the user or print output to the screen. This header file contains the definitions for the functions:

* **cin**: used to take input
* **cout**: used to print output
* **namespace** defines which input/output form is to be used

<https://www.naukri.com/code360/guided-paths/basics-of-c/content/118623/offering/1472636?leftPanelTabValue=NOTE&customSource=studio_nav>

**main()**: This is the portion of any C++ code inside which all the commands are written and executed.

* This is the line at which the program will begin executing. This statement is similar to the start block of flowcharts.

**Class**

A class is a logical entity used to define a new data type. A class is a user-defined data type that describes what a particular kind of object will look like. Thus, a class is a template or blueprint for an object. A class contains variables, methods, and constructors.

A class holds its own data members and member functions, which can be accessed and used by creating an instance of that class. A class is like a blueprint for an object.//

// When a class is defined, no memory is allocated but when it is instantiated (i.e. an object is created) memory is allocated.

The car class would allow the programmer to store similar information unique to each car (different models, maybe different colors, etc.) and associate the appropriate information with each car.

**Object**

An object is an instance of a Class. It is an identifiable entity with some characteristics and behavior. Objects are the basic units of object-oriented programming.  It may be any real-world object like a person, chair, table, pen, animal, car, etc.

A simple example of an object would be a car. Logically, you would expect a car to have a model number

or name. This would be considered the property of the car. You could also expect a car to be able to do something, such as starting or moving. This would be considered a method of the car.

Syntax to create an object dynamically in C++:

class\_name \* objectName = new class\_name();

The class’s default constructor is called, and it dynamically allocates memory for one object of the class. The address of the memory allocated is assigned to the pointer, i.e., objectName.

**Features of OOPs**: Abstraction, Polymorphism, Inheritance, Encapsulation.

### ****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 that will work with a range of data, so we don't have to write basic stuff repeatedly.

Example link :

<https://www.naukri.com/code360/guided-paths/oops-in-c/content/238489/offering/3210644?leftPanelTabValue=NOTE&customSource=studio_nav>

**Constructor:-**

 Constructors are special class functions that perform the initialization of every object. In C++, the constructor is automatically called when an object is created. It is a special method of the class because it does not have any return type. It has the same name as the class itself.

A constructor initializes the class data members with **garbage value** if we don’t put any value to it explicitly.

The constructor must be placed in the public section of the class because we want the class to be instantiated anywhere. For every object in its lifetime constructor is called only once at the time of creation.

### Types of Constructors: 3

* Default constructor
* Parameterized Constructor
* Copy Constructor

**Note:** If we have not defined a constructor in our class, the C++ compiler will automatically create a default constructor with an empty code and no parameters, which will initialize data members with garbage values. Default constructor is available only till the point we don’t create our own constructor.

Link :

<https://www.naukri.com/code360/guided-paths/oops-in-c/content/266821/offering/3651488?leftPanelTabValue=PROBLEM&customSource=studio_nav>

### ****Constructor Overloading****

Constructor overloading can be defined as the concept of having more than one constructor with different parameters so that every constructor can perform a different task.

The corresponding constructor is called depending on the number and type of arguments passed.

What is a copy constructor?

These are a particular type of constructor that takes an object as an argument and copies values of one object’s data members into another object. In this constructor, we pass the class object into another object of the same class.

### Destructor:-

A destructor is a special member function that works just opposite to a constructor; unlike constructors that are used for initializing an object, destructors destroy (or delete) the object. The purpose of the destructor is to free the resources that the object may have acquired during its lifetime.

~class\_name() { }

Link: <https://www.naukri.com/code360/guided-paths/oops-in-c/content/266821/offering/3651490?leftPanelTabValue=NOTE&customSource=studio_nav>

**Access Modifier**

It sets some restrictions on the class members from accessing the outside functions directly.

There are three types of access modifiers available in C++:

* Public
* Private
* Protected: The access level of a protected modifier is within the class and outside the class through the child class (or subclass). If you do not make the child class, it cannot be accessed outside the class.

By default all members in C++ are private.

## Interview Questions

## <https://www.naukri.com/code360/guided-paths/oops-in-c/content/265025/offering/3625237?leftPanelTabValue=NOTE&customSource=studio_nav>

* Constructor: <https://www.naukri.com/code360/guided-paths/oops-in-c/content/266821/offering/3651492?leftPanelTabValue=NOTE&customSource=studio_nav>
* <https://www.naukri.com/code360/guided-paths/oops-in-c/content/274158/offering/3757643?leftPanelTabValue=NOTE&customSource=studio_nav>

### this Pointer

this pointer holds the address of the current object.

There can be three main usages of this keyword in C++.

* It can be used to refer to a current class instance variable (object).
* It can be used to pass the current object as a parameter to another method.
* It can be used to declare indexers.

When an object calls some member function, **it implicitly passes** itself as an argument. This lets the compiler know which member should be used for the purposes. This also allows to reduce the ambiguity among the variable and data member names.

The ‘this’ pointer is available only within the non-static member functions of a class. If the member function is static, it will be common to all the objects, and hence a single object can’t refer to those functions independently.

## Shallow and Deep Copy

<https://www.naukri.com/code360/guided-paths/oops-in-c/content/274158/offering/3757642?leftPanelTabValue=NOTE&customSource=studio_nav>

**Pillars of OOPs:**

Using access specifiers, we can achieve encapsulation. Using this, we can, in turn, implement data abstraction. It’s not necessary that we only use private access.

**Encapsulation:**

By Encapsulation, all the necessary data and methods are bind together and all the unnecessary details are hidden to the normal user. So Encapsulation is the process of binding data members and methods of a program together to do a specific job, without revealing unnecessary details.  
  
Encapsulation can also be defined in two different ways:  
  
1) **Data hiding:** Encapsulation is the process of hiding unwanted information, such as restricting access to any member of an object.  
  
2) **Data binding:** Encapsulation is the process of binding the data members and the methods together as a whole, as a class.

\*\*All object-oriented programming concepts are inspired by real-life examples

And encapsulation is also one such concept, have you ever seen a capsule used for medication it is made up of two components first is the medication itself and the second is the outer shell which is used to protect medication from various factors like cold and hot environment so important medication is wrapped into some container call capsule which is used to protect it from outside this concept call encapsulation,

Similarly in object-oriented programming variables and functions are wrapped into classes and objects. you can consider a class as an outer container that protects variables and functions from any outside unauthorized access so wrapping variables and functions into a single unit like a class its called encapsulation in Oops

Encapsulation is used for restricting access to the variables and functions and it is also used for data hiding

Real-life example: When you send an email to someone, you just click send, and you get the success message; what happens when you click send, how data is transmitted over the network to the recipient is hidden from you (because it is irrelevant to you).

We can create a fully encapsulated class by making all the data members private.

**Abstraction:**

With a real-life example consider the example of YouTube, YouTube shows only relevant data to you like the title of the video, likes, views, and comments, and hides confidential irrelevant information like the ID of the video, the country of the original video

And which algorithm does YouTube have to use for this video recommended to you

Abstraction is a mechanism that allows us to hide irrelevant data or confidential information from the user so the user will have access only to essential details

We can achieve abstraction in many ways:

Abstract class

Access modifiers

Interfaces etc.

**What is the difference between Encapsulation and Abstraction?**

Answer: because both of them talk about hiding data

Abstraction means showing only the required things and hiding the background details.

\_Encapsulation means wrapping data and methods into a single unit so that data remains hidden and can only be accessed by the properties.

Abstraction is a broader concept of hiding. It was done by various ways.

But Encapsulation is like a proper implementation where you wrap the method and the data together to do encapsulation data must be part of the encapsulation

Encapsulation is one of the ways of achieving abstraction

Polymorphism: many forms :

Two types: Compile time polymorphism ---function overloading

---operator overloading

Runtime polymorphism -----function overriding (virtual function)

Polymorphism is a concept that allows you to perform a single action in different ways

**Real-life example**:  A person at the same time can have different characteristics. Like a man at the same time is a father, a husband, and an employee. So the same person possesses different behaviors in different situations. This is called polymorphism.

a)**Function overloading:**

"Function overloading is a feature that allows you to have multiple functions with the same name but different parameter lists. The compiler differentiates them based on the number and type of parameters."

### b)Operator Overloading "Operator overloading allows you to redefine the behavior of operators for user-defined types.

### Practice Answer:

"Function overloading is when multiple functions have the same name but different parameters. For example, in C++, you can have several add functions: one that takes two integers, another that takes three integers, and another that takes two doubles. The compiler determines which function to call based on the arguments.

Operator overloading lets you redefine the behavior of operators for custom types. For instance, in C++, you can overload the + operator for a Complex class to add complex numbers by adding their real and imaginary parts. This makes the class easier to use and more intuitive, as it allows you to use standard operators with custom objects."

Note:

Within the class, you can access private members of the same class object

Runtime polymorphism is resolved during runtime. It is achieved using inheritance and virtual functions.

* **Runtime polymorphism:**

Runtime polymorphism is also known as dynamic polymorphism. Method overriding is a way to implement runtime polymorphism.

**Method overriding:**

Method overriding is a feature that allows you to redefine the parent class method in the child class based on its requirement. In other words, whatever methods the parent class has by default are available in the child class. But, sometimes, a child class may not be satisfied with parent class method implementation. The child class is allowed to redefine that method based on its requirement. This process is called method overriding.

**Rules for method overriding:**

* The parent class method and the method of the child class must have the same name.
* The parent class method and the method of the child class must have the same parameters.
* It is possible through inheritance only.

<https://www.naukri.com/code360/guided-paths/oops-in-c/content/274159/offering/3757648>

Virtual Function:

Have a virtual key word before the function name, it has to be overwritten in the child class

When we created a pointer object and call a virtual function statically so its work in run time , its wich object precent run time in pointer object , uska function call hoga

#include <iostream>

using namespace std;

class Base {

public:

virtual void show() {

cout << "Base class" << endl;

}

};

class Derived : public Base {

public:

void show() override { // Override the virtual function

cout << "Derived class" << endl;

}

};

int main() {

Base\* b;

Derived d;

b = &d;

// Calls Derived class show() method at runtime due to dynamic binding

b->show(); // Outputs: Derived class

return 0;

}

**Object-specific vtable Pointer:** Each object of a class with virtual functions has a pointer to the vtable.

**Dynamic Binding:** When a virtual function is called on an object through a base class pointer or reference, the program uses the vtable to determine the correct function to call at runtime, enabling dynamic binding.

"In object-oriented programming, a virtual function is a function defined in a base class that can be overridden by derived classes. It's declared using the virtual keyword. The key feature of a virtual function is that it supports runtime polymorphism, which allows the program to decide at runtime which function to call based on the actual type of the object.

This is achieved using a mechanism called the virtual table (vtable). When a class has virtual functions, the compiler creates a vtable for that class. Each object of the class has a pointer to the vtable. When a virtual function is called, the program uses the vtable to find the correct function implementation to call at runtime. This allows derived classes to provide their specific implementations of the virtual function, enabling dynamic behavior and flexibility in the code.

<https://www.youtube.com/watch?v=p2h8rGnkD0o>

Inheritance:

**What is early binding and late binding?**

### Python

While Python doesn't support function overloading in the traditional sense, it does support operator overloading through special methods, also known as magic methods.