AMBITION GURU

Object-Oriented programming in C++ (CSIT 202)

II semester, BScCSIT

Compiled by Ankit Bhattarai
Ambition Guru College

Syllabus

AMBITION GURU

Unit	Contents	Hours	Remarks
1.	Introduction to C++ and OOP	3	
2.	Classes and objects	7	
3.	Operator Overloading & Type Conversion	7	
4.	Inheritance	7	
5.	Polymorphism and Virtual Functions	7	
6.	Templates and Exception Handling	7	
7.	I/O Stream	8	

Practical Works

Credit hours : 3

AMBITION GURU

Practical Works

Students must complete a set of practical exercises that cover:

- Creating classes and objects
- Implementing inheritance and polymorphism
- Using file I/O operations
- Building small projects such as a student record system, inventory management, or banking application

AMBITION GURU

Resources

Resources

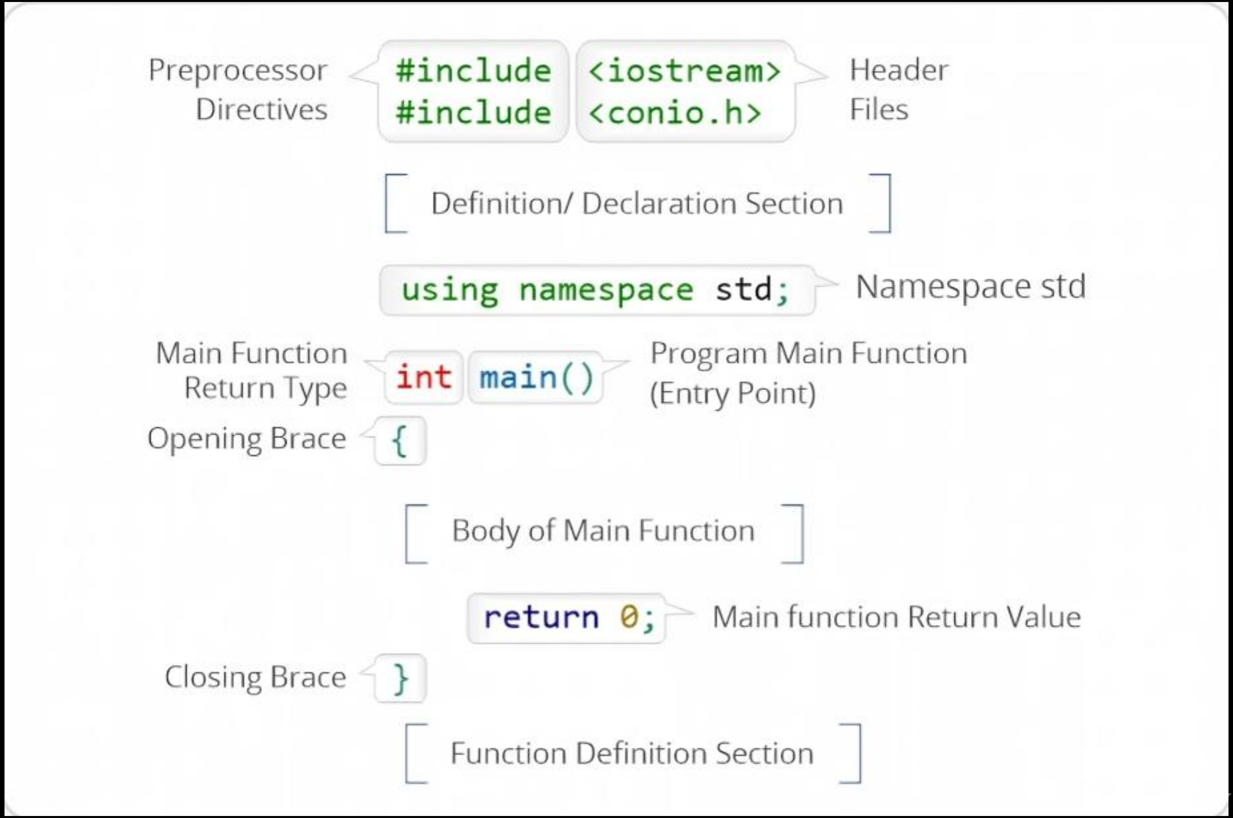
1. Some chapters in pdf form.
2. Question Collection & Assignments (50-80 Questions)

Text Books/Reference Books:

- E. Balagurusamy*, Object-Oriented Programming with C++*, Tata McGraw-Hill publication.
- Herbert Schildt, *C++: The Complete Reference* , Tata McGraw Hill publication.
- Robert Lafore ,*Programming in C++* , SAMS Publication.
- Joyce Farrell*,Object-Oriented Programming in C++* by ,Cengage Learning.

Format for C++ program

AMBITION GURU


Preprocessor Directives
Header Files
Definition/ Declaration Section
Namespace std
Main Function Return Type
Program Main Function (Entry Point)
Opening Brace
Body of Main Function
Main function Return Value
Closing Brace
Function Definition Section

```
#include <iostream>
#include <conio.h>
using namespace std;
int main()
{
return 0;
}
```

Unit 1
Introduction to C++ and OOP
(3 hrs.)

Evolution of C++ from C, Key differences between C and C++, Structure of a C++ program, Basic I/O operations (Cin, Cout), Introduction to OOP concepts: Class, Object, Abstraction, Encapsulation, Inheritance, Polymorphism

AMBITION GURU

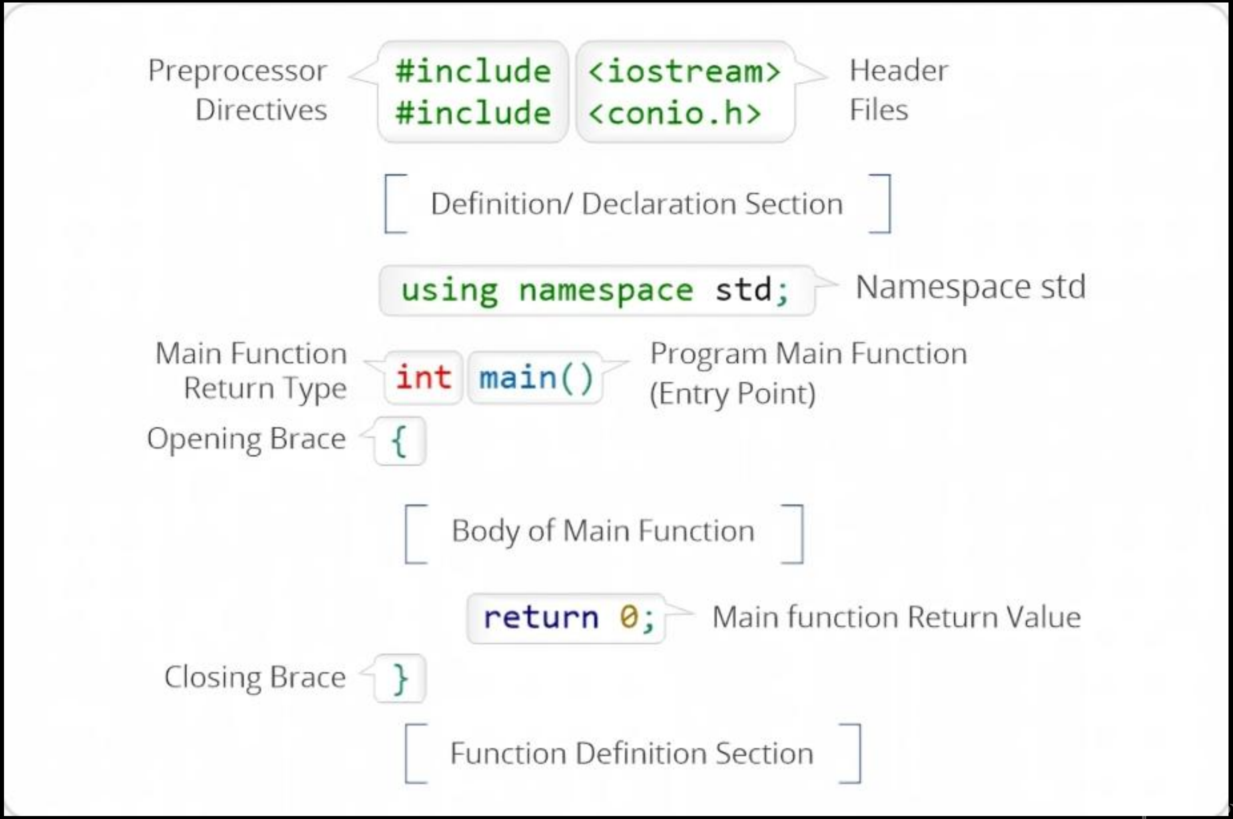
Evolution of C++ from C, Structure of a C++ program

Evolution of C++ from C

- In 1967, Martin Richards developed the BCPL (Basic Combined Programming Language)
- In 1970, Ken Thompson, immersed in the development of UNIX at Bell Labs, created the B language
- In **1973**, Denis Ritchie, had developed the bases of C. The inclusion of types, its handling, as well as the improvement of arrays and pointers, along with later demonstrated capacity of portability without becoming a high-level language, contributed to the expansion of the C language.
- In 1979, Bjarne Stroustrup, from Bell labs, began the development of the C++ language.
- And finally from 1990 on, ANSI committee X3J16 began the development of a specific standard for C++. In the period elapsed until the publication of the standard in 1998, C++ lived a great expansion in its use and today is the preferred language to develop professional applications on all platforms.

Format for C++ program

AMBITION GURU


Preprocessor Directives
Header Files
Definition/ Declaration Section
Namespace std
Main Function Return Type
Program Main Function (Entry Point)
Opening Brace
Body of Main Function
Main function Return Value
Closing Brace
Function Definition Section

```
#include <iostream>
#include <conio.h>

using namespace std;

int main()
{

    return 0;
}
```

Structure of a C++ program

AMBITION GURU

// my first program in C++

```
#include <iostream>
using namespace std;
int main ()
{
cout << "Hello World!";
return 0;
}
```

Output : Hello World!

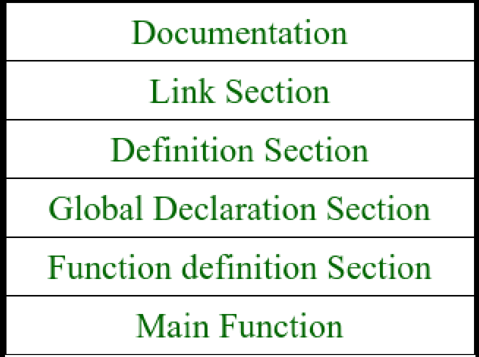
Structure of a C++ program

The structure of the program written in C++ language is as follows:

Documentation Section:
This section comes first and is used to document the logic of the program that the programmer going to code. It can be also used to write for purpose of the program.

Linking Section:
The linking section contains two parts:

1. Header Files:
Standard headers are specified in a program through the preprocessor directive #include.

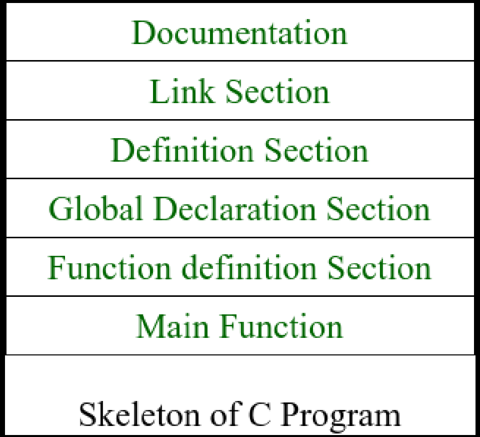


Skeleton of C Program

Structure of a C++ program

2. Namespaces: A namespace permits grouping of various entities like classes, objects, functions, and various C++ tokens, etc. under a single name. Namespaces can be accessed in multiple ways:

```
using namespace std;
using std :: cout;
```



Documentation
Link Section
Definition Section
Global Declaration Section
Function definition Section
Main Function

Skeleton of C Program

Definition Section:

It is used to declare some constants and assign them some value.

AMBITION GURU

Structure of a C++ program

Global Declaration Section:

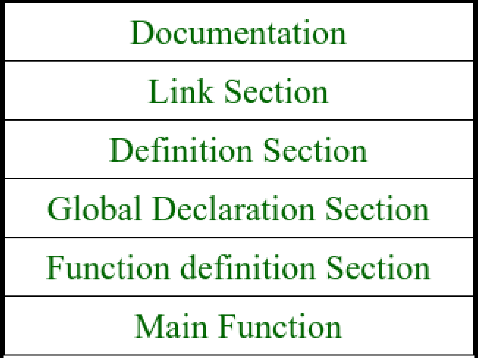
Here, the variables and the class definitions which are going to be used in the program are declared to make them global.

Function Definition Section:

It contains all the functions which our main functions need. Usually, this section contains the User-defined functions.

Main Function:

The main function tells the compiler where to start the execution of the program. The execution of the program starts with the main function.



Skeleton of C Program

Structure of a C++ program

AMBITION GURU

// my first program in C++

```
#include <iostream>
using namespace std;
int main ()
{
cout << "Hello World!";
return 0;
}
```

Output : Hello World!

Structure of a C++ program

AMBITION GURU

```
// my first program in C++
```

- This is a comment line. All lines beginning with two slash signs (//) are considered comments and do not have any effect on the behavior of the program.
- The programmer can use them to include short explanations or observations within the source code itself. In this case, the line is a brief description of what our program is.

Structure of a C++ program

```
#include <iostream>
```

- They are not regular code lines with expressions but indications for the compiler's preprocessor.
- In this case the directive #include <iostream> tells the preprocessor to include the iostream standard file.
- This specific file (iostream) includes the *declarations of the basic standard input-output library in C++,* and it is included because its functionality is going to be used later in the program.

Structure of a C++ program

```
using namespace std;
```

- In C++, most of the features from the standard library (like cout, cin, string, etc.) are stored inside something called a namespace.
- Think of a namespace as a folder that keeps things organized so that names don’t conflict with each other.
- The standard C++ library uses a namespace called std, which stands for standard.

Structure of a C++ program

```
int main ()
```

- This line marks the start of the main function, which is where every C++ program begins execution. No matter where it appears in the code, the program always starts running from here.
  - main is the function name.
  - () means it’s a function (can include parameters).
  - {} contains the code that runs when the program starts.

Every C++ program must have a main() function.

Structure of a C++ program

AMBITION GURU

`cout()`

- cout is used to display output on the screen in C++.
- It stands for "character output".
- It is part of the standard library .
- The << operator is called the insertion operator — it sends the data to the output stream (usually the screen).

NOTE:

`cin()`

- cin is used to take input from the user in C++.
- It stands for "character input".
- It is part of the standard library.
- The >> operator is called the extraction operator — it takes data from the user and stores it in a variable.

AMBITION GURU

Differences between C and C++

Differences between C and C++

	C	C++
History	C was developed by scientist Dennis Ritchie in 1973 at Bell Laboratories.	C++ was developed by Bjarne Stroustup in 1979.
Approach	C follows a top-down approach.	C follows the bottom-up approach.
Programming paradigm	It supports procedural programming.	It supports both procedural and object oriented
Encapsulation	Data and functions are separated in C because it is a procedural programming language.	Data and functions are encapsulated together in form of an object in C++.
Access modifier	It does not support access modifier.	It support access modifier.
Input and output function	In C, scanf() and printf() functions are used to take the input and output respectively.	In C++, cin and cout functions are used to take the input and output respectively.

Differences between C and C++

	C	C++
Allocation and Deallocation of Memory	In C language, we use calloc() and malloc() for dynamic allocation of memory and free() for deallocation of memory.	In C++ language, we use a new operator for the allocation of memory and a delete operator for the deallocation of memory.
Security	C does not have any security features so it can be manipulated by outsider.	C++ is a secure language as it offers security features such as data hiding and encapsulation.
Keywords	C contains 32 keywords.	C++ contains 63 keywords.
Application Development	C language is more suitable for low level implementation such as network driver, text editor, assembler, etc.	C++ language is more suitable for high level implementation such as game development, smartwatches, embedded systems, etc.

AMBITION GURU

Introduction to OOP concepts: Class, Object, Abstraction, Encapsulation, Inheritance, Polymorphism

Object Oriented Programming

Definition: A programming paradigm based on the concept of "objects," which contain data (attributes) and code (methods).

Key Characteristics:

- **Encapsulation:** Bundles data and methods that operate on the data.
- Inheritance: Enables a class to derive properties and behavior from another class.
- Polymorphism: Allows methods to perform different tasks based on the object that invokes them.
- Abstraction: Hides implementation details from the user.

Object Oriented Programming

AMBITION GURU

Advantages:

- Code reusability: Classes can be reused across programs.
- Scalability: Facilitates large-scale application development.
- Easier maintenance: Modular structure makes it easier to manage changes.

Disadvantages:

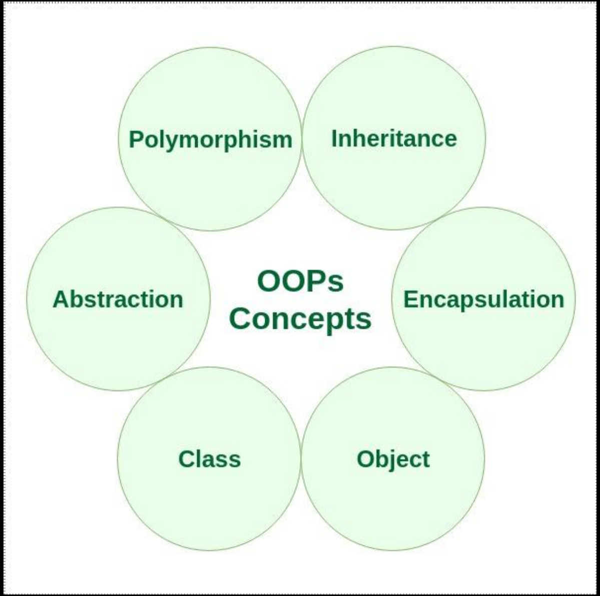
- Complexity: Can be harder to learn and implement for beginners.
- Overhead: May introduce performance and resource consumption issues.

Examples: Languages: Python, Java, C++, C#.

Object Oriented Programming

AMBITION GURU

- 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.



AMBITION GURU

Class & Objects

What is Class ?

It is a user-defined data type, which 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**.

Data members => data variables and member functions => functions

Functions are used to manipulate these variables and together these data members and member functions define the **properties and behavior** of the objects in a Class.

What is Class ?

Example: Class of Cars

There may be many cars with different names and brand but all of them will share some common properties like all of them will have 4 wheels, Speed Limit, Mileage range etc. will act as data members.

Applying brakes, increasing speed may lead to member function of that class.

What is Class ?

A class can be thought of as a template used to create a set of objects.

A class is a static definition; a piece of code written in a programming language.

One or more objects described by the class are ***instantiated*** at runtime.

The objects are called ***instances*** of the class.

Each instance will have its own distinct set of attributes.

Every instance of the same class will have the same set of attributes;

every object has the same attributes but,

each instance will have its own distinct values for those attributes.

Data Members and Member Functions

Data members

The variables which are declared in any class by using any fundamental data types (like int, char, float etc.) or derived data type (like class, structure, pointer etc.) are known as Data Members.

Member Functions

And the functions which are declared in class are known as Member functions.

What do you mean by object ?

- Objects are the runtime entities, existing in the real world. They can be represented as person, a place, a bank account.
- Real-world objects have attributes and behaviors.
  - Examples:

  Dog =>

    - Attributes: breed, color, hungry, tired, etc.
    - Behaviors: eating, sleeping, etc.

  Bank Account =>

  - Attributes: account number, owner, balance
  - Behaviors: withdraw, deposit

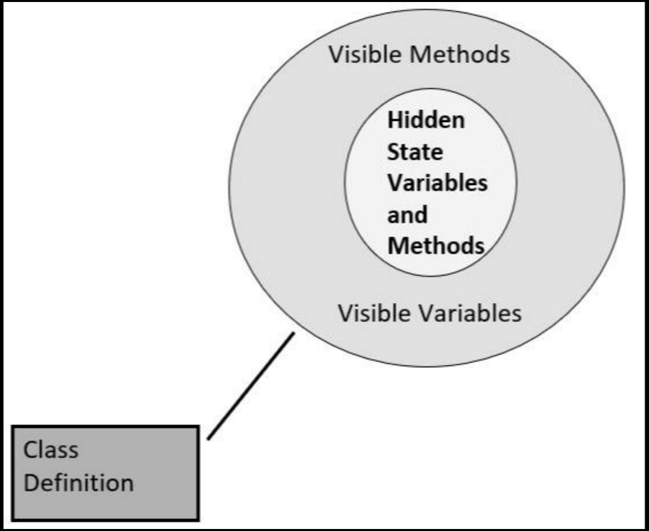
Note: When a class is defined, no memory is allocated but when it is instantiated (i.e. an object is created) memory is allocated. One class can have multiple objects, which can interact with each other.

AMBITION GURU

Encapsulation

Encapsulation

- The wrapping up of data and functions into a single unit is known as encapsulation.
- When classes are defined, programmers can specify that certain methods or state variables remain hidden inside the class.
- These variables and methods are accessible from within the class, but not accessible outside it.
- The insulation of the data from direct access by the program is called data hiding or information hiding.



Encapsulation

- In OOP, hiding member functions means making methods inaccessible from outside the class or limiting their visibility.
- This is done to: Prevent misuse of internal logic, control access to critical functionality and improve modularity and maintainability.

How Member Functions Can Be Hidden

Using Access Modifiers: Most OOP languages (like C++, Java, Python, etc.) support access modifiers:

Modifier	**Description**
private	Accessible only within the same class.
protected	Accessible within the class and its subclasses.
public	Accessible from anywhere.

Encapsulation

AMBITION GURU

Why Hide Member Functions?

- **Encapsulation**: Keep internal details hidden to reduce complexity.
- **Security**: Prevent unauthorized access or changes.
- **Abstraction**: Expose only relevant methods to users.
- **Maintenance**: Easier to update internal logic without affecting external code.

AMBITION GURU

Abstraction

Abstraction

- Abstraction refers to the act of representing essential features without including the background details or explanations.
- Classes use the concept of abstraction and are defined as a list of abstract attributes such as size, weight, and functions to operate on these attributes. They encapsulate all the essential properties of the objects that are to be created.
- 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 accelerator the speed is actually increasing, he does not know about the inner mechanism of the car or the implementation of accelerator, brakes etc. in the car. This is what abstraction is.

AMBITION GURU

Inheritance

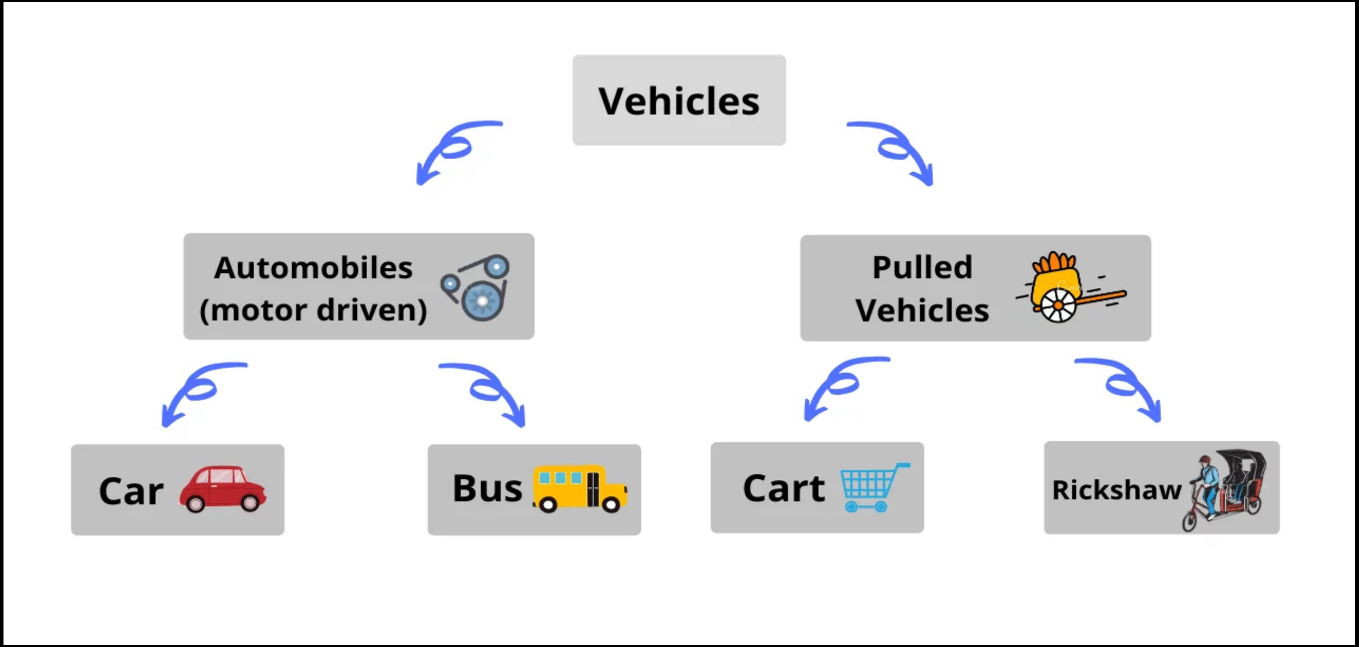
What is inheritance ?

AMBITION GURU

- Inheritance is the process by which objects of one class acquire the properties of objects of another class.
- The capability of a class to derive properties and characteristics from another class is called Inheritance.
- Inheritance is one of the most important features of Object-Oriented Programming.
    - **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.
- **Reusability**: Inheritance supports the concept of "reusability", i.e. when we want to create a new class and there is already a class that includes some of the code that we want, we can derive our new class from the existing class. By doing this, we are reusing the fields and methods of the existing class.

What is inheritance ?

AMBITION GURU



Polymorphism

What is polymorphism ?

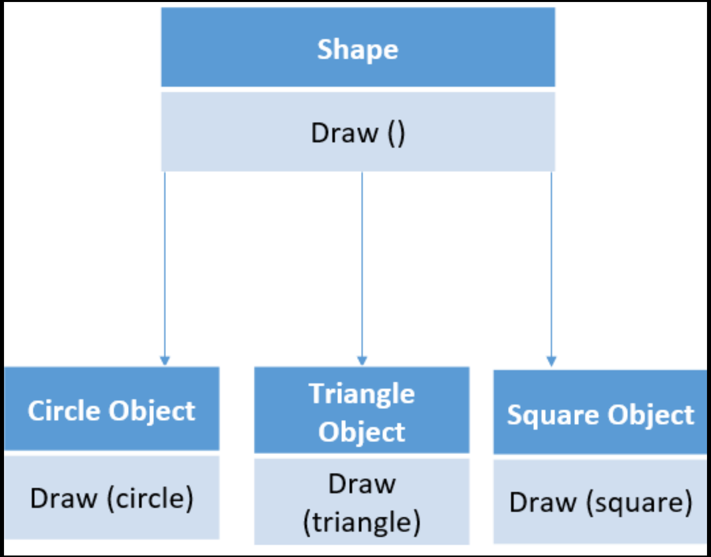
AMBITION GURU

- Polymorphism is one of the essential features of an object- oriented language. It means ability to take more than one form.
- An operation may exhibit different behaviors in different instances. The behavior depends upon the type of data used in the operations.
- **Example:** Addition operation, with integer datatype, it will generate sum, but if operands are strings, it will generate third string by concatenation.
- The process of making an operator to exhibit different behaviors in different instances is known as **operator overloading.**

What is polymorphism ?

AMBITION GURU

- Single function name can be used to handle different number and different types of arguments.
- This is something similar to a particular word having several different meaning depending on the context. So, using *a single function name to perform different types of tasks* is known as function overloading.
- Polymorphism is extensively used in implementing inheritance.



AMBITION GURU

Dynamic Binding

Dynamic Binding

- Dynamic binding refers to linking a procedure call to code that will execute only once.
- It means that the program waits until runtime to decide which version of a function to call.
- The code associated with the procedure is not known until the program is executed, which is also known as late binding.
- Dynamic binding happens when all information needed for a function call cannot be determined at compile-time.
- Dynamic binding can be achieved using the virtual functions.
- **Advantage of using dynamic binding** is that it is flexible since a single function can handle different type of objects at runtime

Message Passing

Message Passing

- An object-oriented program consists of a set of objects that communicate with each other by sending and receiving information same as people pass message to one another.
- Message passing means sending a message from one object to another to request an action.

The process of programming in an object-oriented language involves following steps:

- Creating classes that define objects and their behavior.
- Creating objects from class definition &
- Establishing communication among objects.
- Objects have a life cycle, hence can be created and destroyed Message passing involves specifying the name of the object, name of the function (message) & the information to be sent.

Message Passing

- A message is always given to some object called receiver. The action performed in response to message is not fixed but differ depending upon the class of the receiver .i.e. different object may accept the same message yet perform different action.
- There are three identifiable parts to any message passing expression. These are receiver (object to which message is being sent); the message selector (the text that indicates the particular message being sent) and the argument (used in responding to the message). It is a dynamic process of asking an object to perform action.

  a.getdata(100); where 'a' is declared as an instance of class.

Q. Why object-oriented programming is dominating procedural programming approach? Explain with the features of OOP?

In procedural language, each statement in the language tells the computer to do something, get some i/p, add these numbers, multiply then display the o/p.

A program in a procedural language is a list of instructions. When a program becomes larger, i.e. there may be hundreds of instructions & to overcome this problem. A procedural program is divided into functions and each function has a clearly defined purpose and a clearly defined interface to the other function in the program. But as *program grow even larger and more complex, even the structured programming approach begins to show more signs of problems*.

There are two related problems in procedural language:

1. Functions have unrestricted access to global data.
2. Unrelated function and data.

Q. Why object-oriented programming is dominating procedural programming approach? Explain with the features of OOP?

First problem: When two or more functions need to access the same data, then the data is made global so that it can be accessed by any function.

But, when there are many functions and many global data items, the problem with the procedural paradigm is that this leads to an even larger no. of potential connections between functions and data.

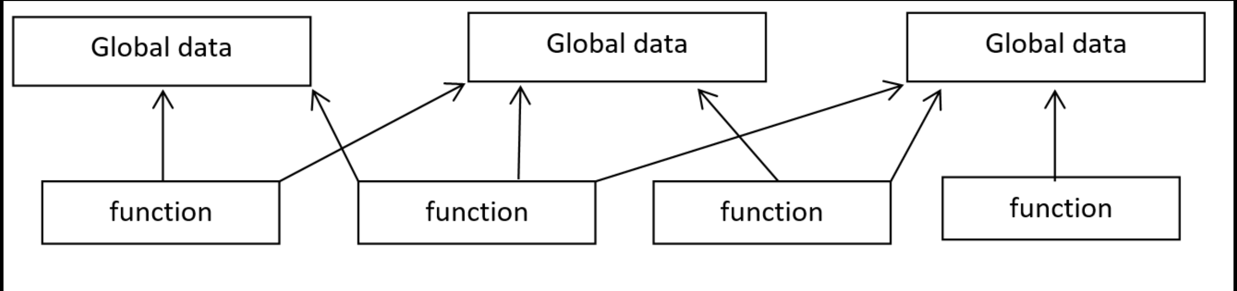


Fig 1: Functions and data in procedural programming.

This approach make a program's structure difficult to conceptualize and difficult to modify.

Q. Why object-oriented programming is dominating procedural programming approach? Explain with the features of OOP?

Second problem: The arrangement of separate data and functions in procedural language does a poor job of modelling thing in the real world. i.e., it does not model the real world very well.

We know that real world objects have both attributes and behaviour. Hence, ***object-oriented approaches model the real world very well***. The fundamental idea behind OOP is to combine data & function which operate on the data as a single unit.

In OOP, an object functions called member functions typically provide the way to access its data. If you need to read a data item in an object then you need to call a member functions of same object. Data cannot be accessed directly i.e., data is hidden, so it is safe from accidental alteration. I.e., Data and its functions are said to be encapsulated in a single entity. So, this simplifies writing, debugging, maintaining & modifying the program.

Q. Why object-oriented programming is dominating procedural programming approach? Explain with the features of OOP?

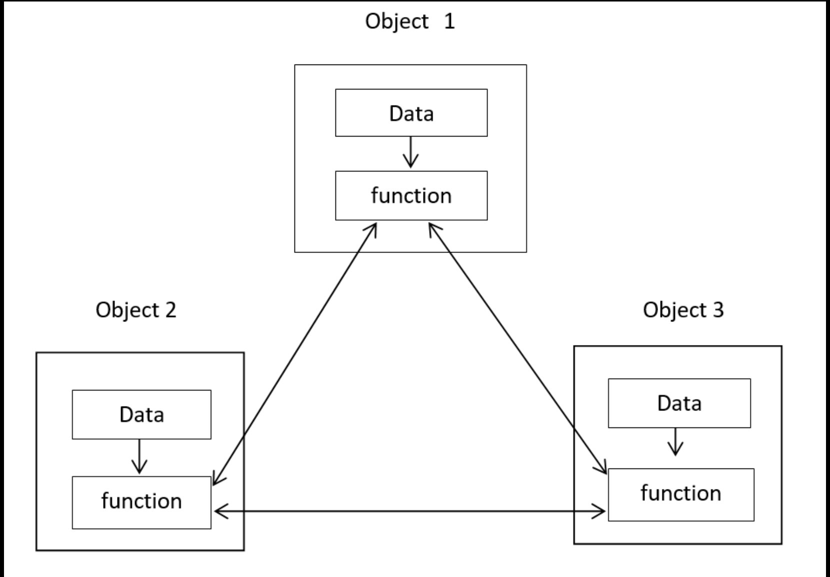


Fig 2: Way of organizing data & function in OOP

Hence, the features like object, class, encapsulation, inheritance, polymorphism; data abstraction in OOP dominates procedural programming approach.

AMBITION GURU

THANK YOU

Any Queries ?