0301201
INTRODUCTION TO OBJECT ORIENTED PROGRAMMING

Introduction to Procedural Programming

- Procedural programming uses a list of instructions to tell the computer what to do step-by-step.
- It has standard approach used in traditional languages like Fortran, COBOL and C.
- Procedural programming creates step by step program that guides the application through a sequence of instructions.
- Each instruction is executed in order one by one.
- If you want a computer to do something, you should provide step-by-step instructions on how to do it. It is, therefore, no surprise that most of the early programming languages are all procedural.
- In procedural programming variables and functions are stored in seperate memory locations while in OOP variables and functions are stored in same memory location.

Limitation or Advantages/Disadvantages of Procedural Programming

Advantages

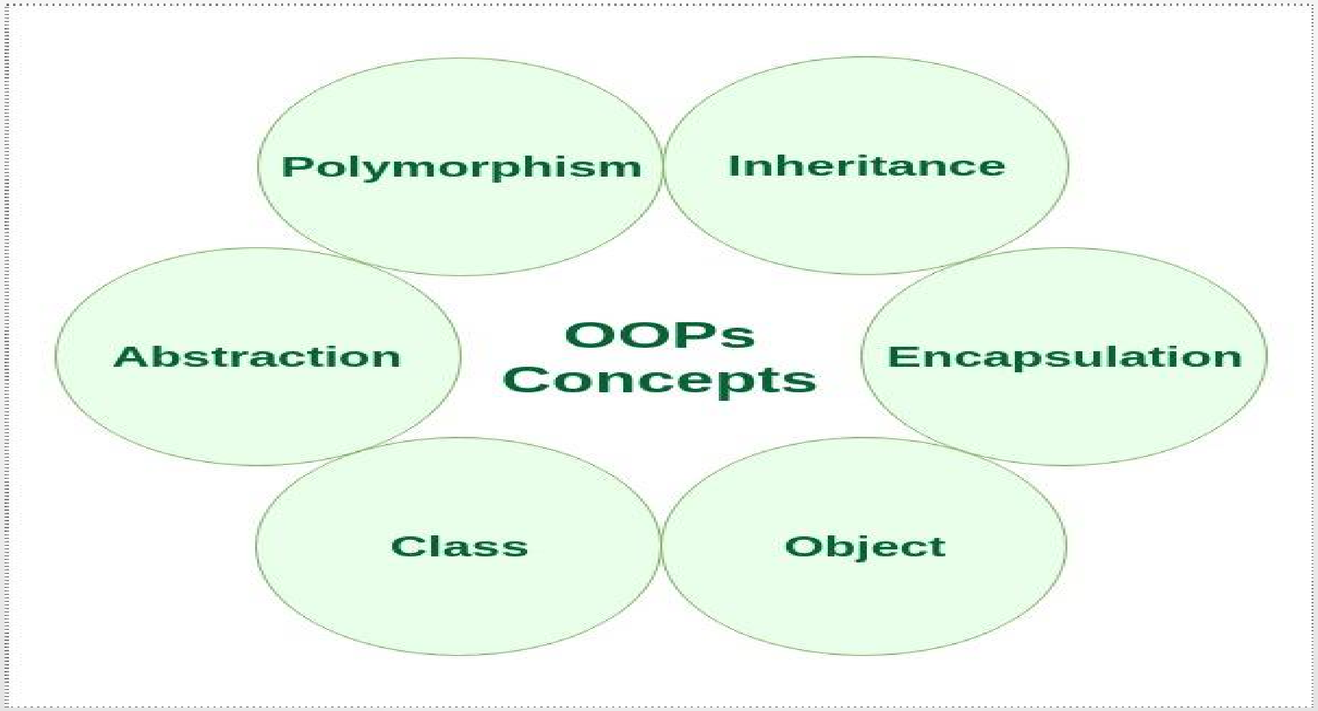
- Relative simplicity, and ease of implementation of compilers and interpreters.
- The ability to re-use the same code at different places in the program without copying it.
- An easier way to keep track of program flow.
- Needs less memory.

Disadvantages

- Data is exposed to whole program, so no security for data.
- Difficult to relate with real world objects.
- Importance is given to the operation on data rather than the data.

Introduction to OOP OR What is OOP ?

- Object Oriented programming is a programming style that is associated with the concept of **Class**, **Objects** and various other concepts revolving around these two, like **Inheritance, Polymorphism, Abstraction, Encapsulation** etc.



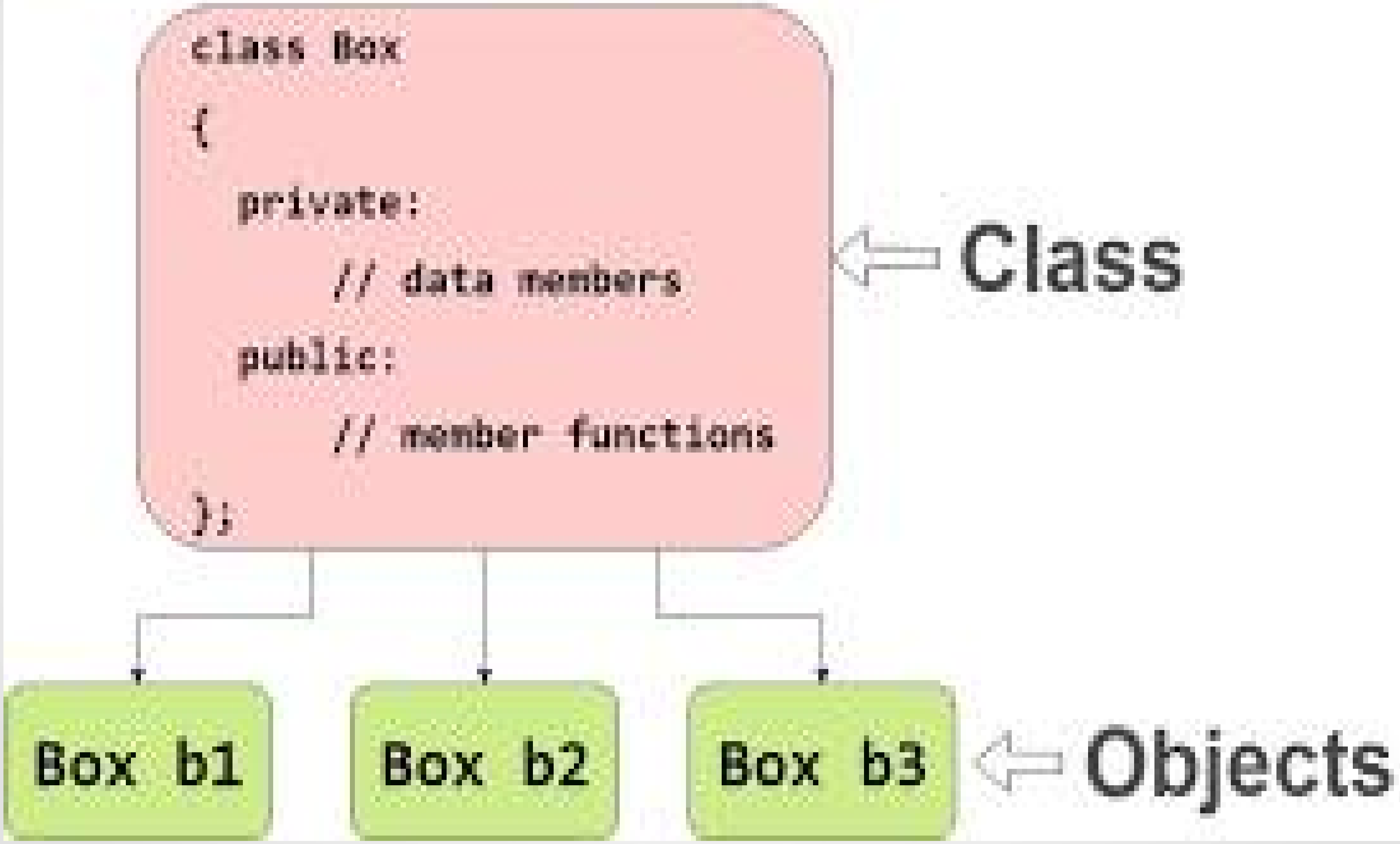
What is Class?

- The building block of C++ that leads to Object-Oriented programming is a 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.

What is Class in C++

- A class is a way to bind data and its associated function together.
- It allows the data and function to be hidden, if necessary.
- When defining class we are creating new abstract data type that can be treated as other built-in data type

Classes & Objects



```
class Box
{
  private:
      // data members
  public:
      // member functions
};
```

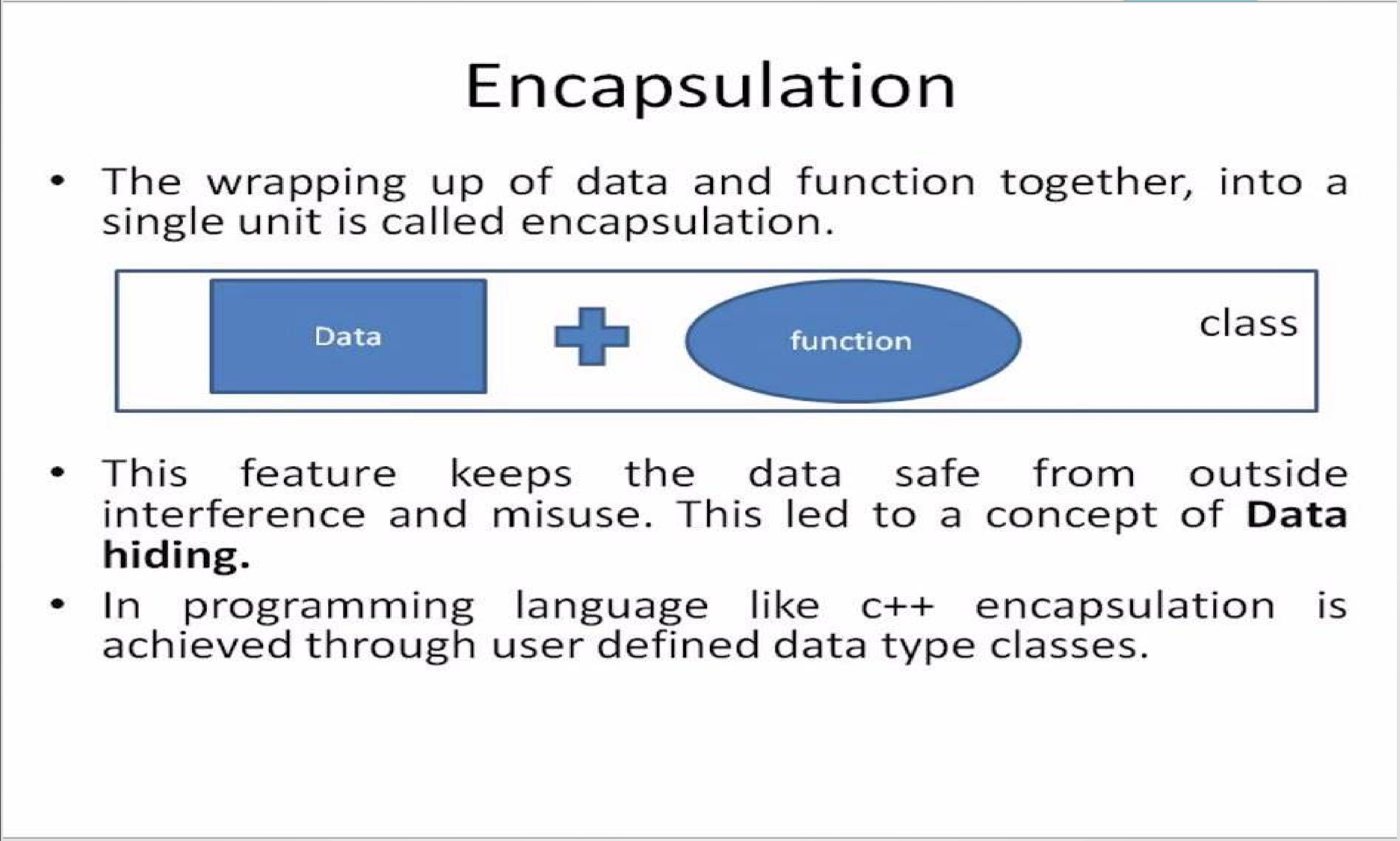
For Example: Consider the Class of Cars. There may be many cars with different names and brand but all of them will share some common properties like all of them will have 4 wheels, Speed Limit, Mileage range etc. So here, Car is the class and wheels, speed limits, mileage are their properties.

- A Class is a user-defined data-type which has data members and member functions.
- Data members are the data variables and member functions are the functions used to manipulate these variables and together these data members and member functions define the properties and behaviour of the objects in a Class.
- So in class Car, the data member will be speed limit, mileage etc and member functions can apply brakes, increase speed etc.

What is Object ?

- An Object is an identifiable entity with some characteristics and behaviour.
- 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.

Encapsulation



Encapsulation

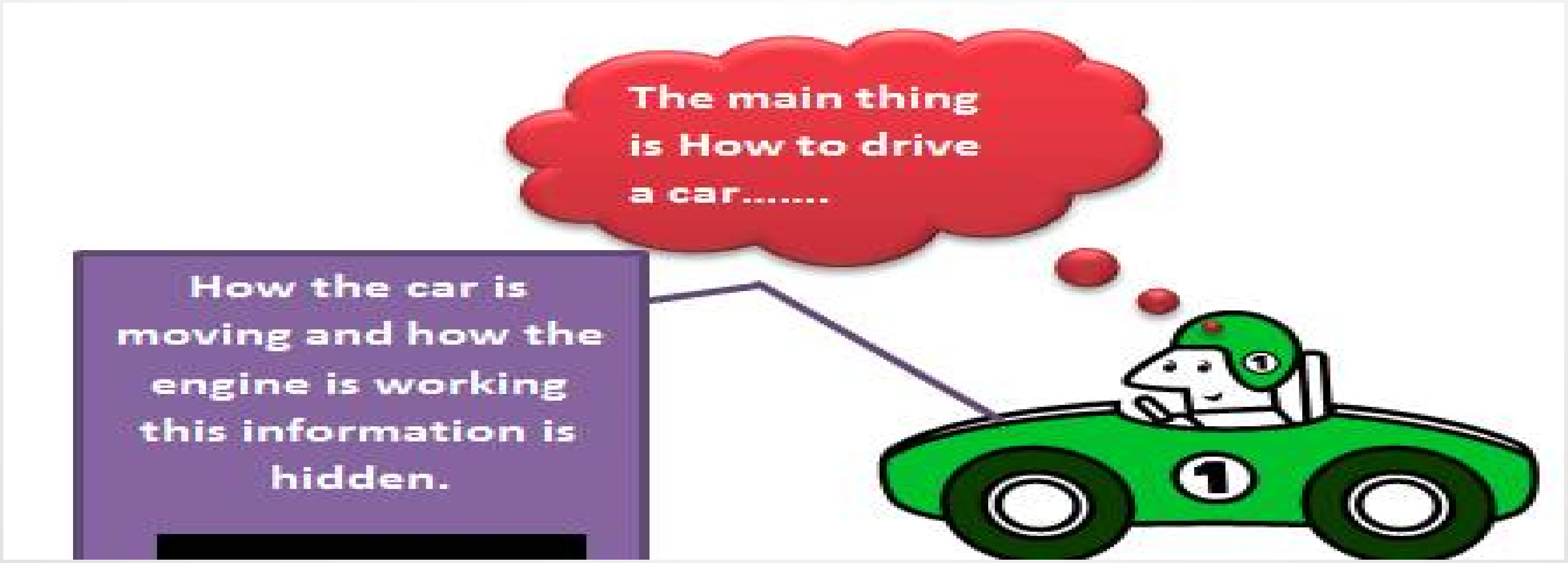
- The wrapping up of data and function together, into a single unit is called encapsulation.

Data ✚ function class

- This feature keeps the data safe from outside interference and misuse. This led to a concept of **Data hiding.**
- In programming language like c++ encapsulation is achieved through user defined data type classes.

Abstraction

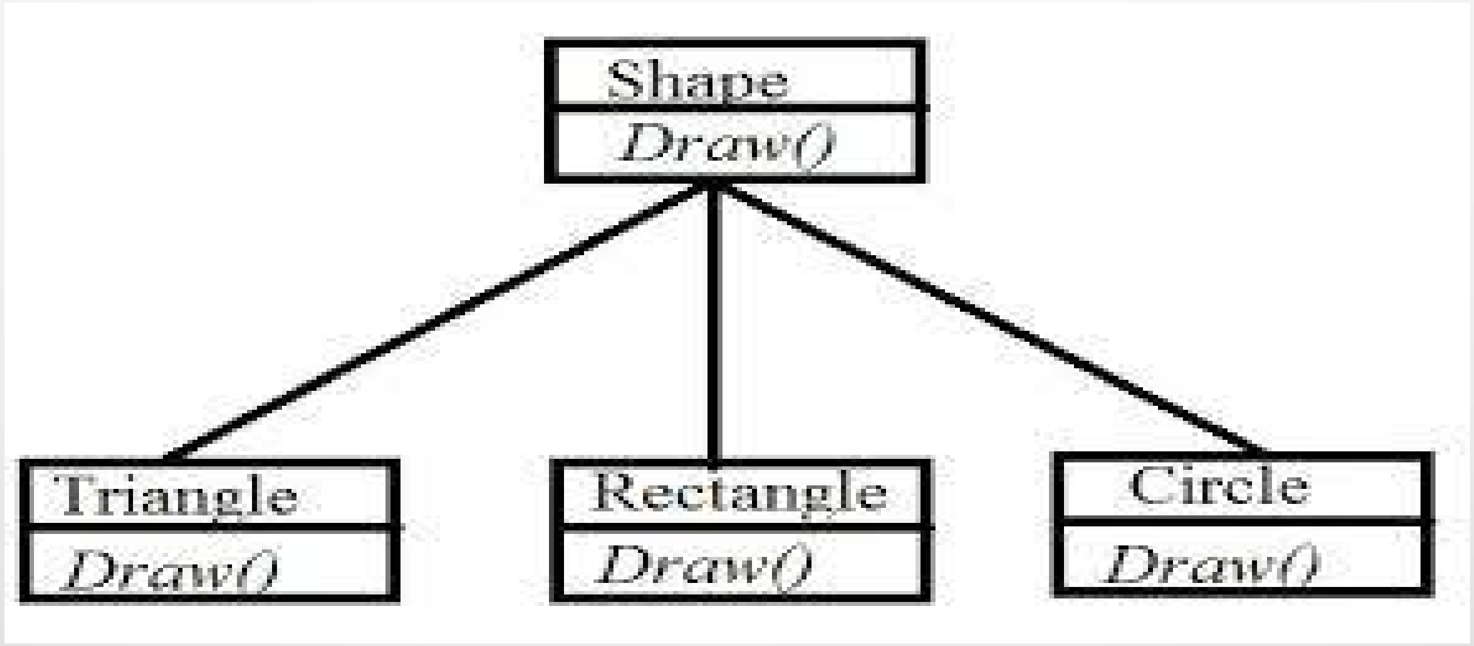
Data abstraction refers to, providing only essential information to the outside world and hiding their background details, i.e., to represent the needed information in program without presenting the details. Hiding internal details and showing functionality is known as abstraction. For example: phone call, we don't know the internal processing.


The main thing is How to drive a car.......
How the car is moving and how the engine is working this information is hidden.

Abstraction	Encapsulation
1. Abstraction solves the problem in the design level.	1. Encapsulation solves the problem in the implementation level.
2. Abstraction is used for hiding the unwanted data and giving relevant data.	2. Encapsulation means hiding the code and data into a single unit to protect the data from outside world.
3. Abstraction lets you focus on what the object does instead of how it does it	3. Encapsulation means hiding the internal details or mechanics of how an object does something.
4. **Abstraction**- Outer layout, used in terms of design. For Example:- Outer Look of a Mobile Phone, like it has a display screen and keypad buttons to dial a number.	4. **Encapsulation**- Inner layout, used in terms of implementation. For Example:- Inner Implementation detail of a Mobile Phone, how keypad button and Display Screen are connect with each other using circuits.

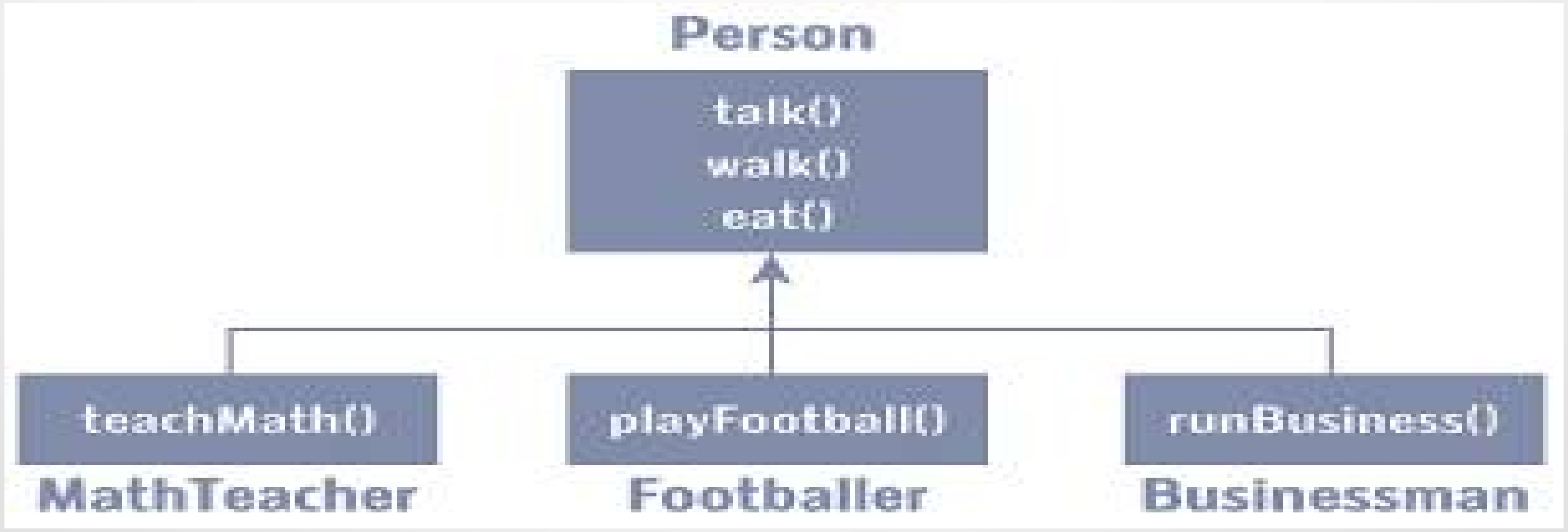
Polymorphism

When one task is performed by different ways i.e. known as polymorphism. In oops, we use method overloading and method overriding to achieve polymorphism. Example to speak something e.g. cat speaks meaw, dog barks woof etc.



Inheritance

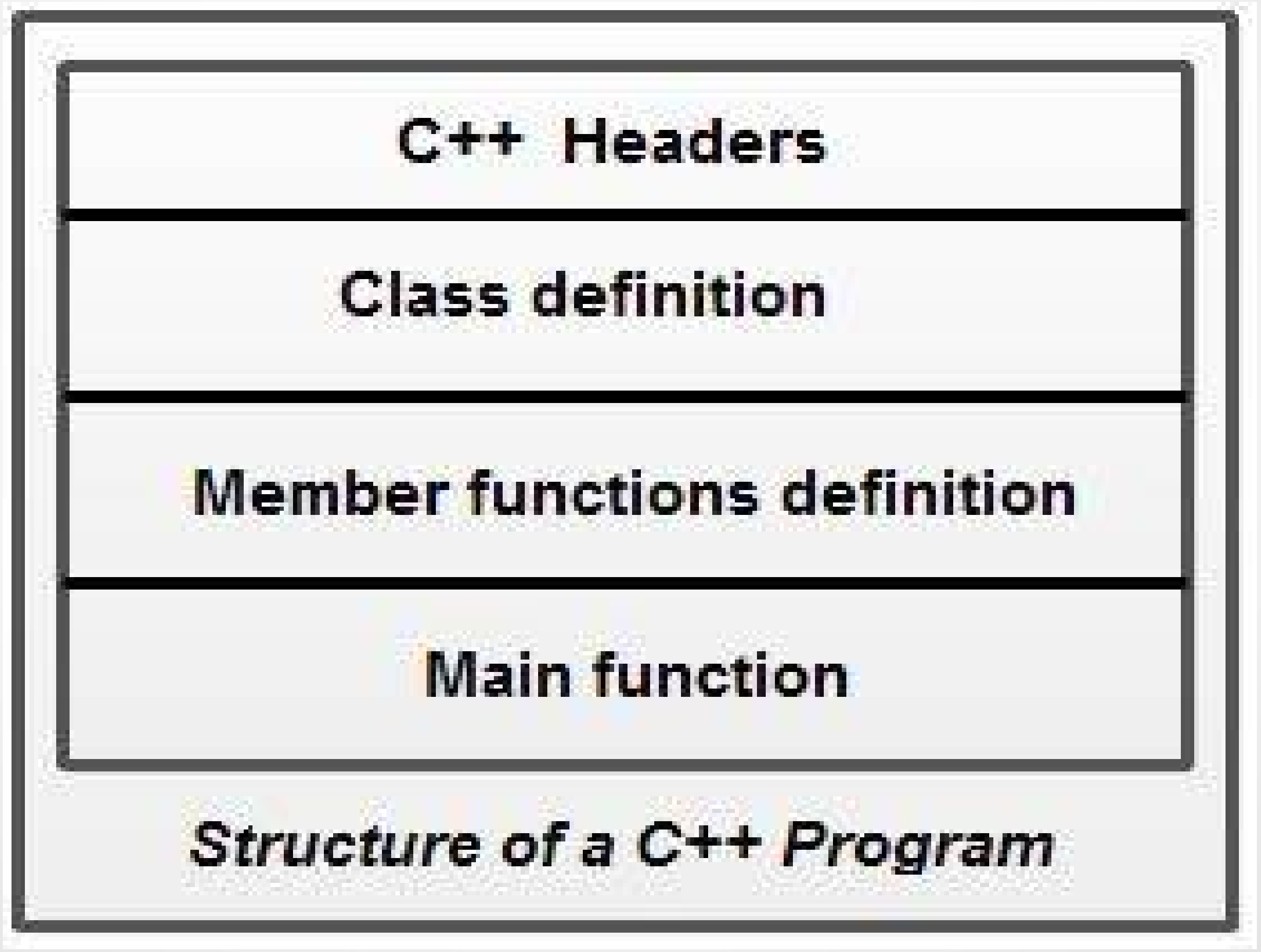
- When one object acquires all the properties and behaviours of parent object i.e. known as inheritance.
- It provides code reusability.
- Inheritance is the process of forming a new class from an existing class that is from the existing class called as base class, new class is formed called as derived class.



Introduction to C++

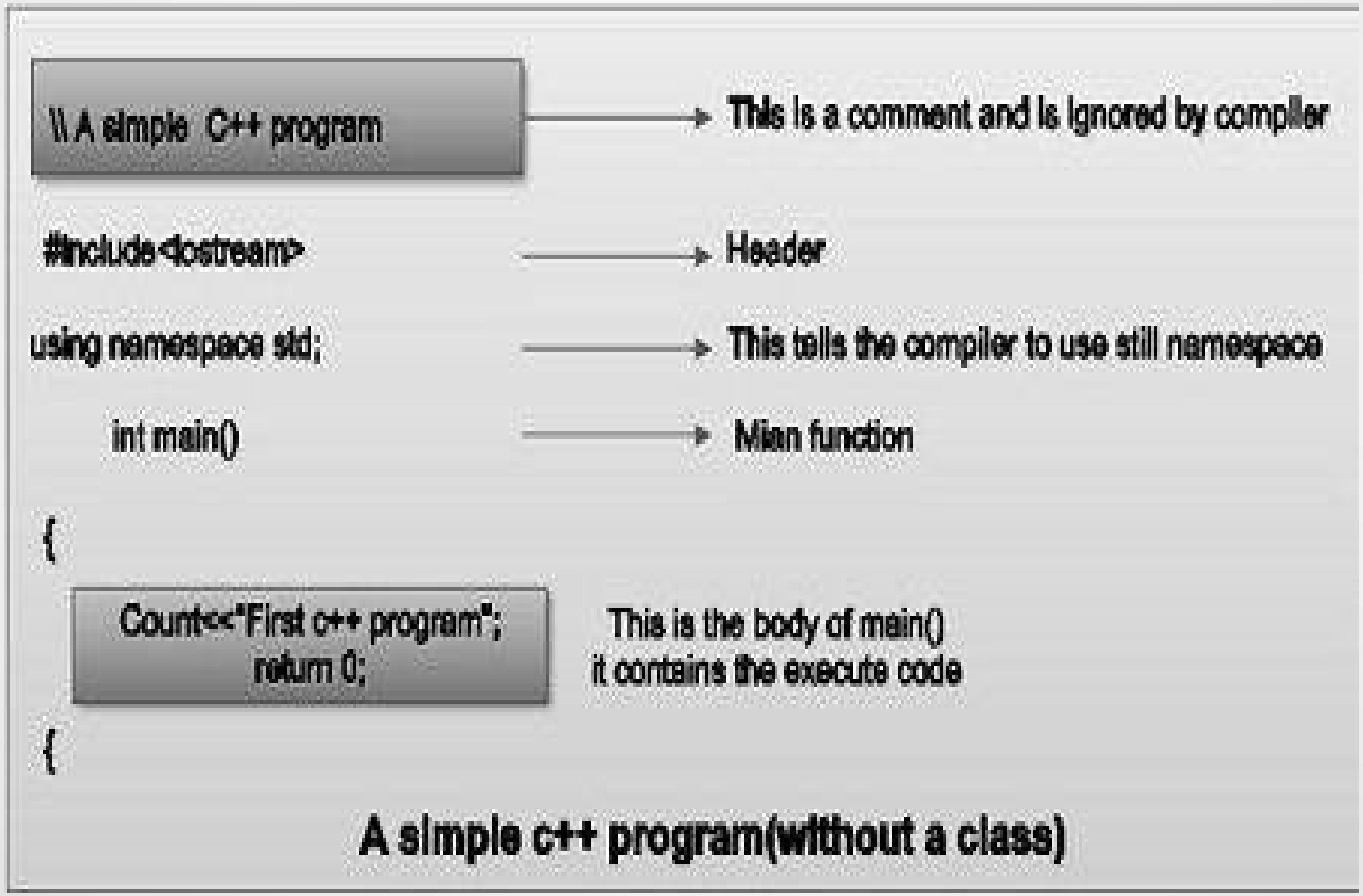
- C++ is a multi-paradigm programming language that supports object-oriented programming.
- It is created by **Bjarne Stroustrup** in 1983 at Bell Labs.
- C++ is an extension(superset) of C programming.

Structure of c++ Program



Structure of a C++ Program

Basic structure of c++



\\ A simple C++ program → This is a comment and is ignored by compiler

#include<iostream> → Header

using namespace std; → This tells the compiler to use still namespace

int main() → Mian function

{

Count<<"First c++ program";
return 0;

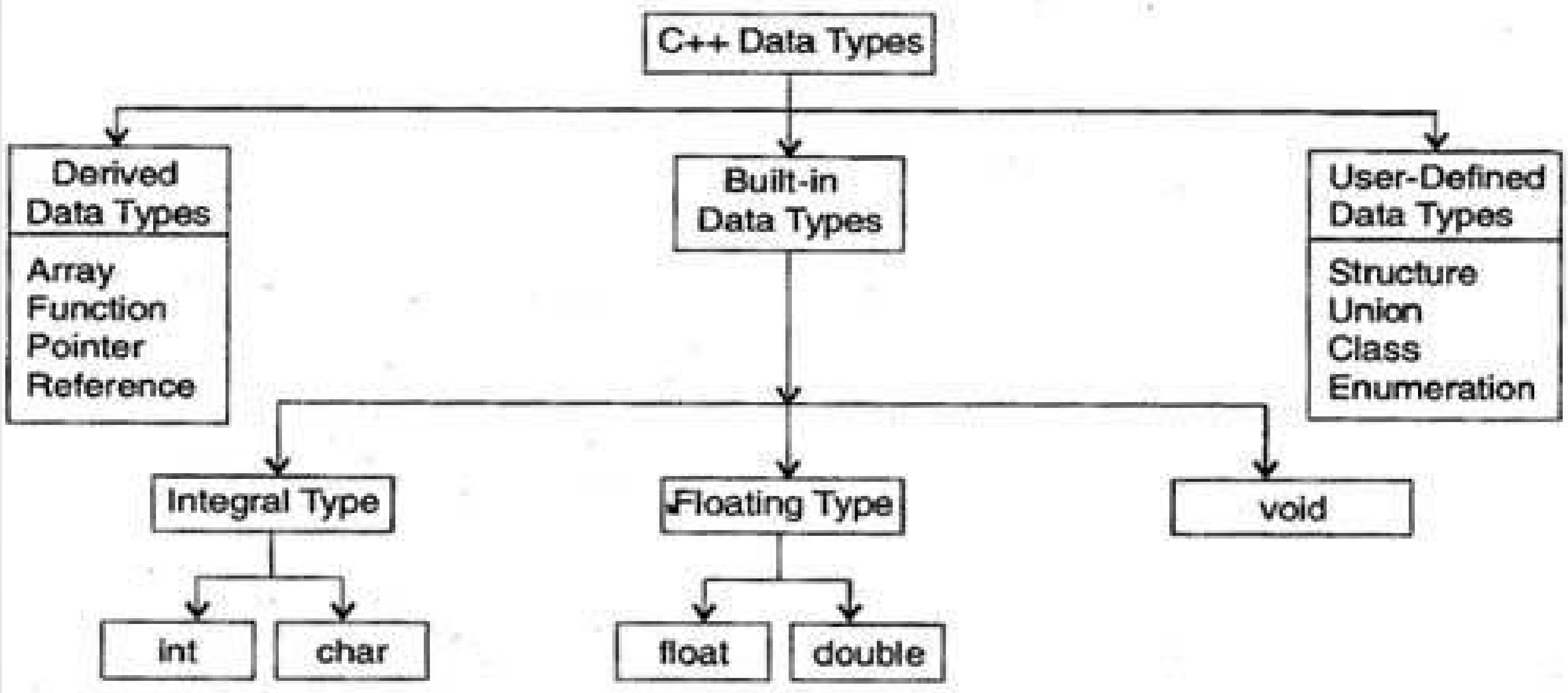
This is the body of main()
it contains the execute code

{

A simple c++ program(without a class)

Data Types

- A data type determines the type and the operations that can be performed on the data.
- C++ provides various data types and each data type is represented differently within the computer’s memory.
- The various data types provided by C++ are built-in data types, derived data types and user-defined data types



Various Data Types in C++

char

- All single characters used in programs belong to character type. char keyword is used to denote character type.
- Size of char type variable is **1 byte.**
- The range of char type: -128 to 127.
- When character is assigned to variable char type it stores ASCII value of the character.
- **Example: char c; //declaration. c='a'; //value assign**

unsigned char

- It is variation of char type. Size of a variable is **1 byte.**
- The range of unsinged char type: 0 to 255

int

- All numeric data without fraction is stored in integer type.
- Size of the variable of type int is different computer to compter. if it is 16-bit machine then it occupies **4 bytes.**
- Range of int type: -32768 to 32767
- int type provides **3 modifiers**: unsigned int, short int, long int.

unsigned int

- Size of a variable of unsigend int is also 2 bytes. It can only stores positive values.
- Range is: 0 to 655535
- Example: unsigned int a;

short int

- The size of the variable of short int is 2 bytes. It is same as int type.
- Example: short int a;

long int

- he size of a variable of long int type is 4 bytes.
- Range is: -2147483648 to 2147483647
- Example: long int l;
- The combination of insigned short int and unsigned long int is also possible. But range will be in positive.
- Example: unsigned i; short s; long l; unsigned short us; unsigned long ul;

float

- float is used to declare variable of real type i.e. fractional numerical values.
- float type occupies 4 bytes.
- Range: 3.4e-38 to 3.4e+38
- It stores floating point numbers with 6 digits of precision. Floating poimnt numbers stored in float variable are called **single-precision numbers.**
- **Example: float f;**
- Modifiers of float type type include **long float** and **long double.**

double

- A variable of type double requires 8 bytes of memory.
- Range: 1.7e -308 to 1.74e +308.
- It stores floating point numbers with 14 digits of precision.
- Example: double a;

long double

A variable of long double type requires 12 bytes of memory space.

- Range:3.4e -4932 to 1.1e+ 4932.
- It stores floating point numbers with 14 digits of precision.
- It has more accuracy.
- Example: long double d;

Bool

•The bool data type has one of two possible values: true or false.
• Booleans are used in conditional statements and loops.
• For example,

```
bool cond = false;
```

void

•the void keyword indicates an absence of data. It means "nothing" or "no value".
•void cannot be used to declare simple variables.

Tokens

A token is the smallest element of a program that is meaningful to the compiler. Tokens can be classified as follows:

- Keywords
- Identifiers
- Constants
- Strings
- Special Symbols
- Operators

Keywords

- A keyword is a reserved word. You cannot use it as a variable name, constant name etc.
- These are the words used for special purposes or predefined tasks.

Identifiers

- Identifiers are used as the general terminology for the naming of variables, functions and arrays.
- Identifier names must differ in spelling and case from any keywords.
- You cannot use keywords as identifiers
- The first character of identifier should be an alphabet or underscore. The rest can be letters or digits or underscore.
- Lower case and upper case letters are distinct.
- Maximum length of an identifier can be of 31 characters. However the lengthvaries from one version of compiler to another version.
  - Some valid examples:- sum, a1, a2, _1, _a, average, a_b, x123y...
  - Some invalid examples:- 1a, a-b, float

Constants

- Constants are also like normal variables. But, the only difference is, their values can not be modified by the program once they are defined. Constants refer to fixed values.
- Constants may belong to any of the data type.
- Syntax:

  const data_type variable_name; (or) const data_type *variable_name;
- Types of Constants:
  - **Integer constants** – Example: 0, 1, 1218, 12482
  - **Real or Floating-point constants** – Example: 0.0, 1203.03, 30486.184
  - **Octal & Hexadecimal constants** – Example: octal: (013)8 = (11)10, Hexadecimal: (013)16 = (19)10
  - **Character constants** - Example: ‘a’, ‘A’, ‘z’
  - **String constants** - Example: “GLS University”

Variables

A variable is a name given to a memory location. It is the basic unit of storage in a program.

- The value stored in a variable can be changed during program execution.
- A variable is only a name given to a memory location, all the operations done on the variable effects that memory location.
- In C++, all the variables must be declared before use.

Namespace in C++

- A namespace is designed to overcome this difficulty and is used as additional information to differentiate similar functions, classes, variables etc. with the same name available in different libraries.
- Using namespace, you can define the context in which names are defined. In essence, a namespace defines a scope.

Namespaces in C++

- A **namespace** is designed to overcome this difficulty and is used as additional information to differentiate similar functions, classes, variables etc. with the same name available in different libraries. Using namespace, you can define the context in which names are defined. In essence, a namespace defines a scope.

- **Namespace** is a container for identifiers. It puts the names of its members in a distinct space so that they don't conflict with the names in other namespaces or global namespace.

Defining a Namespace

- The member can be accessed in the program as:

```
namespace_name::member_name;
```

- **For Example:**

```
namespace sample {
   int a;
   int sum(int x, int y) {
      return (x+y);
   }
}
```

- Now, the variable *a* and function *sum()* can be accessed in the program as:

```
sample::a = 5;
int x = sample::sum(5,9);
```

Namespace in C++

- A namespace is a named block which contains variables, functions, classes.
- This encloses all these members within a named scope, which helps in avoiding name collisions
- The "using" directive avoids us to type out the complete named scope, when accessing a member of the namespace.

```
namespace MyNamespace {
        int my_data;
}
MyNamespace::my_data = 100;
```

:: (scope resolution)

The :: (scope resolution) operator is used to get hidden names due to variable scopes so that you can still use them. The scope resolution operator can be used as both unary and binary. You can use the unary scope operator if a namespace scope or global scope name is hidden by a particular declaration of an equivalent name during a block or class. For example, if you have a global variable of name my_var and a local variable of name my_var, to access global my_var, you'll need to use the scope resolution operator.

example

```
#include <iostream>
using namespace std;
int my_var = 0;
int main(void) {
   int my_var = 0;
   ::my_var = 1;  // set global my_var to 1
   my_var = 2;    // set local my_var to 2
   cout << ::my_var << ", " << my_var;
   return 0;
}
```

Comments

One important part of good documentation is Comments.

- In computer programming, a comment is a programmer-readable explanation or annotation in the source code of a computer program
- Comments are statements that are not executed by the compiler and interpreter.
- In C/C++ there are two types of comments :
  - Single line comment - start with two forward slashes (//)
  - Multi-line comment - start with /* and ends with */.

Input/Output Statements

C++ I/O operation is using the stream concept. **Stream** is the sequence of bytes or flow of data. It makes the performance fast.

- If bytes flow from main memory to device like printer, display screen, or a network connection, etc, this is called as **output operation.**
- If bytes flow from device like printer, display screen, or a network connection, etc to main memory, this is called as **input operation.**

- Header File used for I/O statement:

  <iostream> - It is used to define the cout, cin and cerr objects, which correspond to standard output stream, standard input stream and standard error stream, respectively.

Cin statement

- cin is a predefined variable that reads data from the keyboard with the extraction operator (>>).
- The cin is the input statement.
- The cin is a predefined object of istream class.
- The symbol >> is called extraction operator.
- The syntax of the statement is: **cin>> variable;**
- **Example: int b; cin>>b;**
  - Here cin accepts value of the variable b.
- If there is more than one variable then valuescan be accepted with single cin statement.
- **Example: cin>>a>>b;**

  here variable a and b accepts value of the variable.

Cout statement

- The cout object, together with the << operator, is used to output values/print text
- The cout is a predefined object of ostream class.
- The cout statement is used for displaying the outputs.
- Syntax of the statement is: **cout<<variable;**
- The symbol << is called insertion operator.
- Example: int b = 20; cout<<b;
  - Displays the value of a variable b.

Example

```
  #include <iostream>
 int main( )
{
   int age;
    cout << "Enter your age: ";
    cin >> age;
    cout << "Your age is: " << age << endl;
 }
```