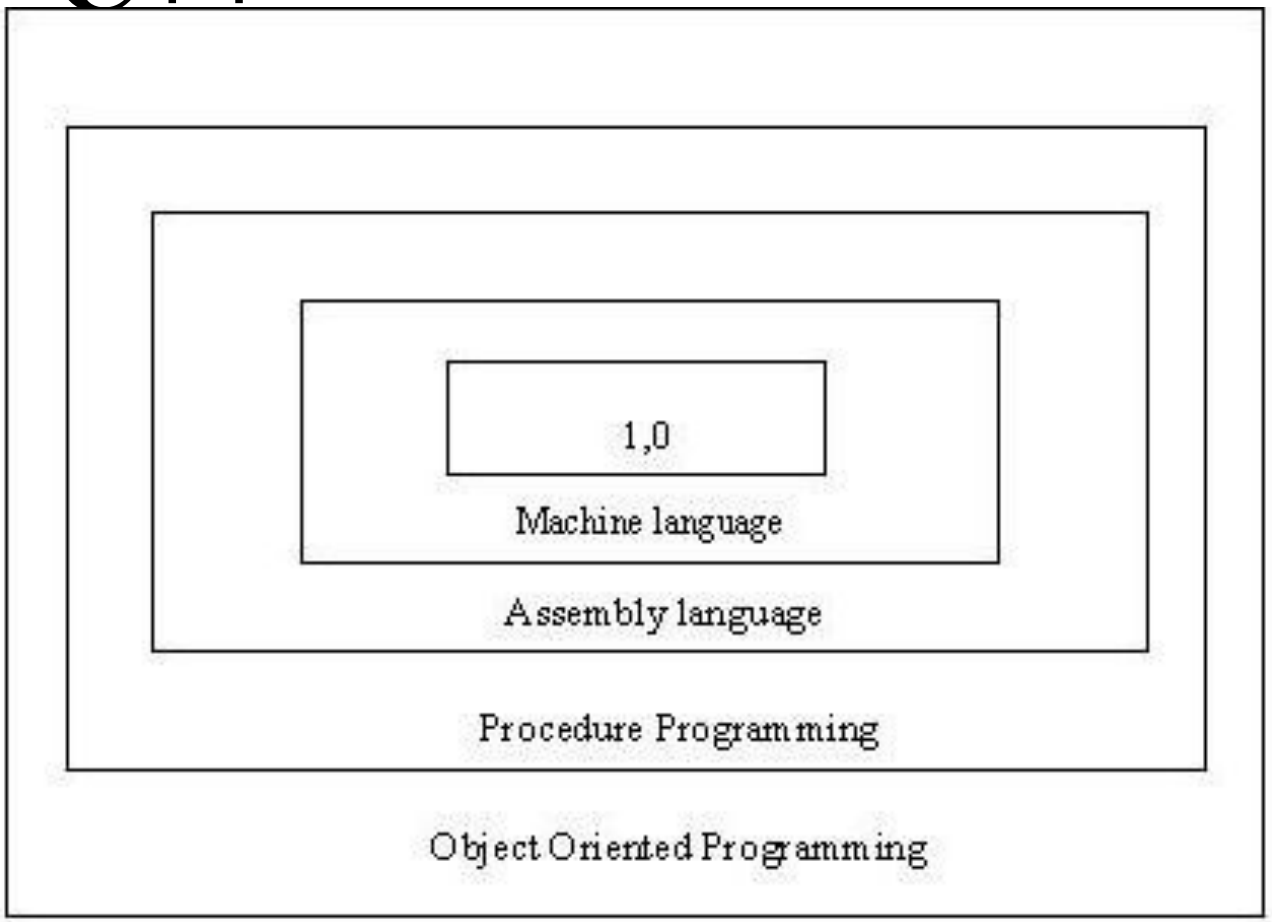
Lecture 1

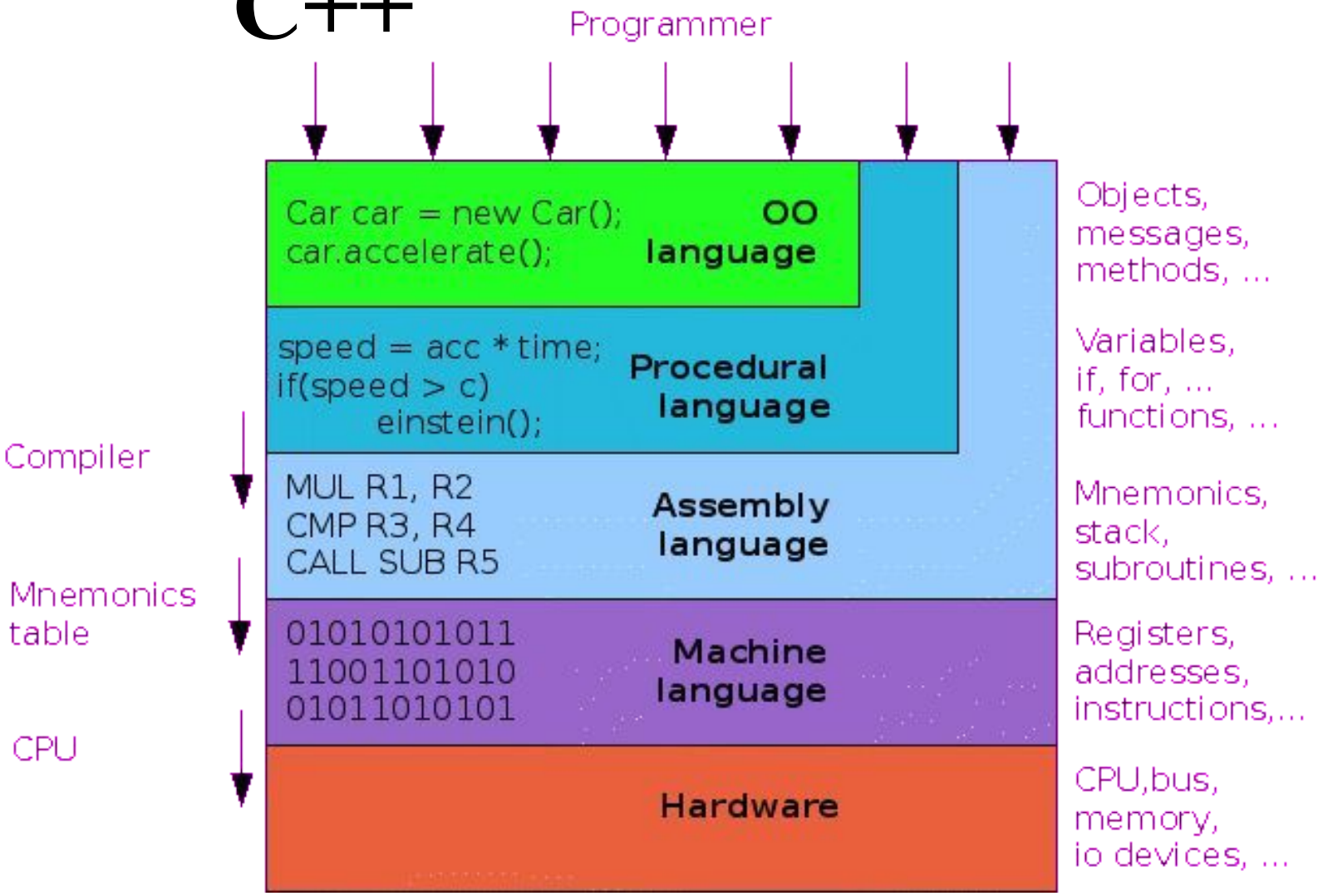
Introduction

Basic Concepts of C++



Layers of Computer Software

Basic Concepts of C++



Basic Concepts of C++

- **Object Oriented Programming** is method of programming where a system is considered as a collection of objects that interact together to accomplish certain tasks. Objects are entities that encapsulate data and procedures that operate on the data.
- In **OOPS** first a concept known as **"**Object Oriented Analysis **(OOA)"** is used to specify the objects in term of real world requirements, their behavior and interactions required. The next concept would be the **"**Object Oriented Design **(OOD)"** that converts these real-time requirements as a hierarchy of objects in terms of software development requirement. Finally OOPS is used to implement the requirements using the C++ programming language.

- The main purpose of object oriented programming is to simplify the design, programming and most importantly debugging a program. So to modify a particular data, it is easy to identify which function to use. To add additional features it is easy to identify where to add functions and its related data.

Basic Concepts of C++

- Object Oriented Programming Language

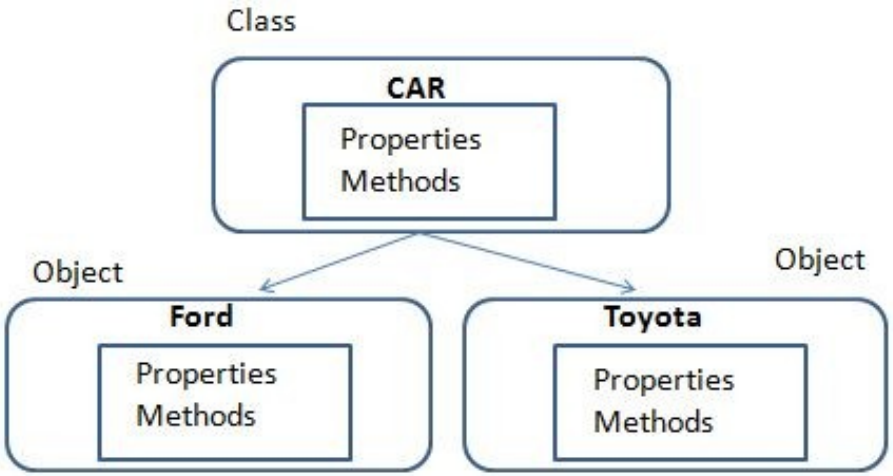
- Basic Concepts:
  - Objects
  - Classes
  - Data Abstraction
  - Encapsulation
  - Inheritance
  - Polymorphism
  - Dynamic Binding
  - Message Passing

Objects

- Basic runtime entity in an object – oriented system.
- Often termed as "*instance*" of a class.
- Example:
  - a person, a place, a bank account *etc.*
- They can be of any type based on its declaration.
- When program is executed, objects interact by sending messages at runtime.
- Example 2:
  - Two objects namely, "customer", "account". Customer object may send a message to the account object requesting for bank balance.

Classes

- Class is a collection of objects of similar type.
- Class is a way to bind the data andits associated functions together.
- Objects are basically variable of the class or an object is an instance of a class.
- Examples:
  - Shape is a class and Rectangle, Square, Triangle are its objects.
  -

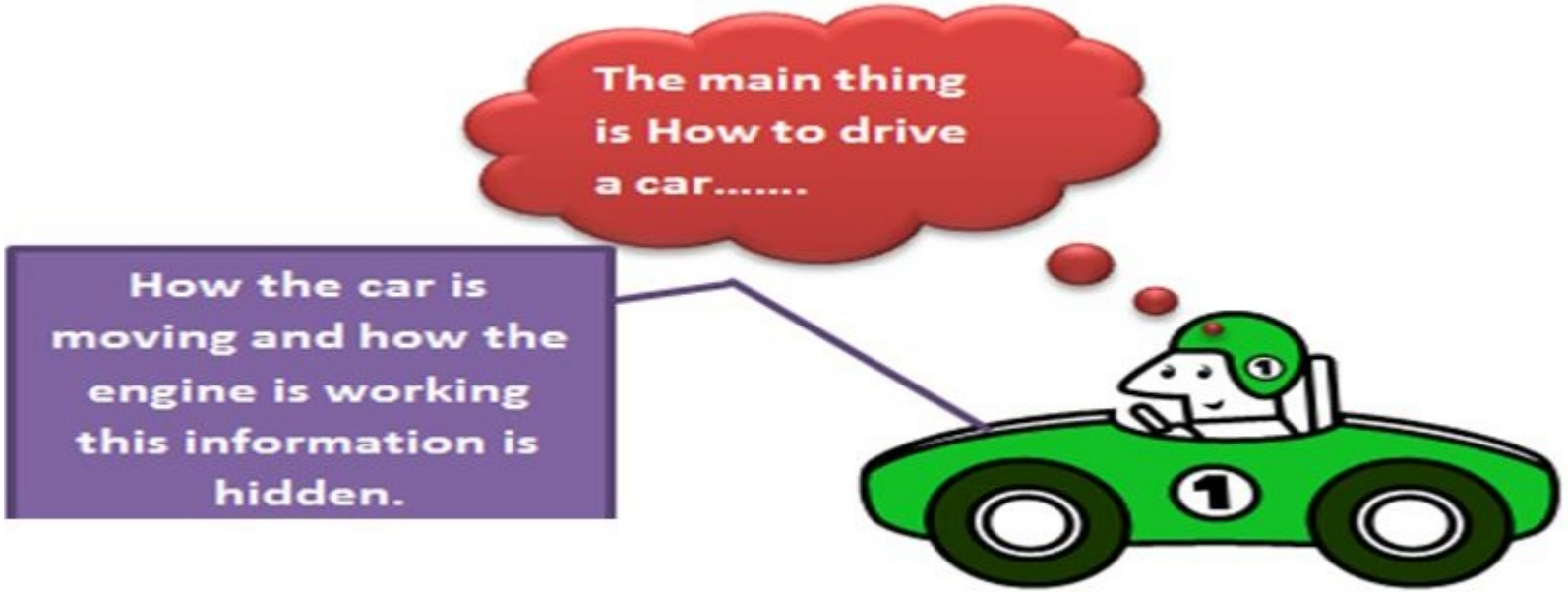


Abstraction

- *Providing only essential information* to the outside world i.e. to represent the needed information in program without presenting the details.
- Data abstraction is a programming/ design technique that relies on the **separation of interface and implementation**.
- In C++, they provide sufficient public methods to the outside world to play with the functionality of the object and to manipulate object data, i.e., state without actually knowing how class has been implemented internally.

- Eg: While using an object (that is an instance of a class) the built in data types and the members in the class are ignored. This is known as **data abstraction**.

Abstraction



Encapsulation

- All C++ programs are composed of the following two fundamental elements:
  - **Program statements (code):** This is the part of a program that performs actions and they are called functions.
  - **Program data:** The data is the information of the program which affected by the program functions.

- Encapsulation is an Object Oriented Programming concept that binds together the data and functions that manipulate the data, and that keeps both safe from outside interference and misuse.

- Data encapsulation led to the important OOP concept of **data hiding**

Encapsulation

- C++ supports the properties of encapsulation and data hiding through the creation of user-defined types, called **classes.**
- Eg: a class can contain
  - **private, protected, public** members.

Inheritance

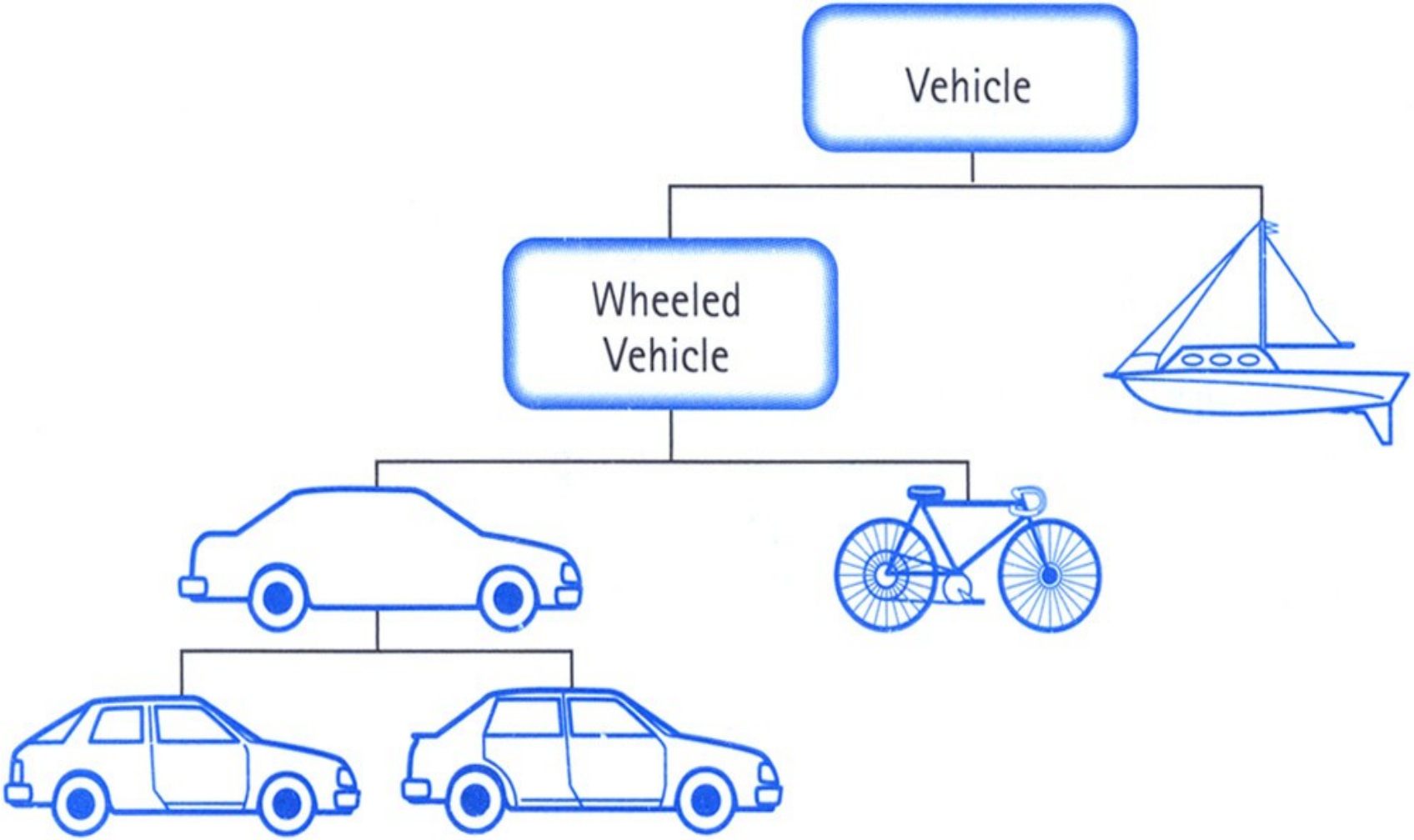
- Inheritance works on the basis of **re-usability**.

- This provides us an opportunity to reuse the code functionality and fast implementation time.

- When creating a class, instead of writing completely new data members and member functions, the programmer can designate that the new class should inherit the members of an existing class. This existing class is called the base class, and the new class is referred to as the derived class.

- The idea of inheritance implements the **is a** *relationship*.

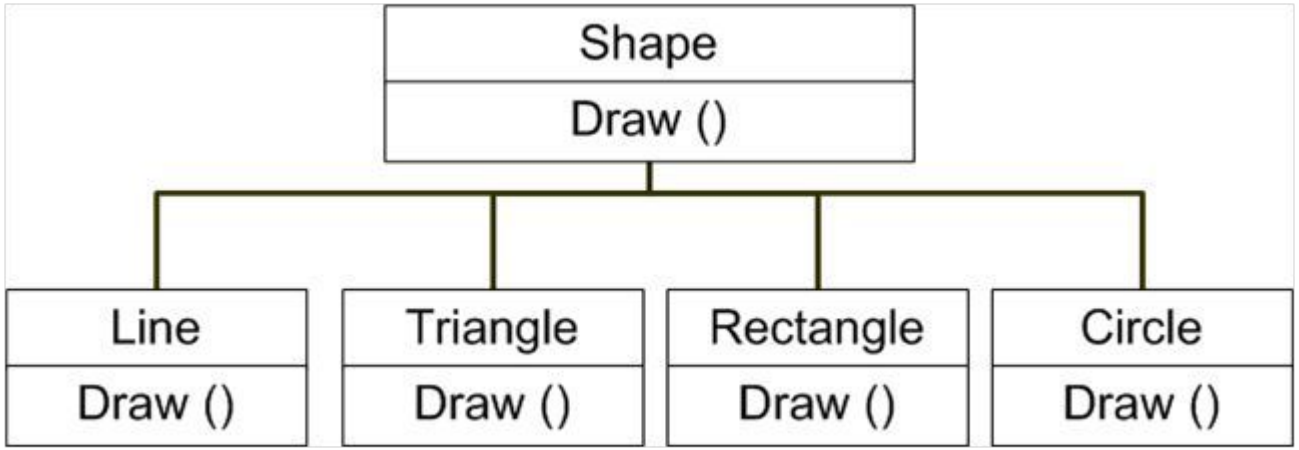
Inheritance



Polymorphism

- *Poly* means **many** and *morphism* means **changing or alterable**.
- The word **polymorphism** means having *many forms*.

Polymorphism



- C++ polymorphism means that a call to a member function will cause a different function to be executed depending on the type of object that invokes the function.

Dynamic Binding

- Binding refers to the linking of a procedure call to the code to be executed in response to the call.

- Dynamic binding (*late binding*) means that the code associated with a given procedure call is **not** known until the **time of the call at run-time**.

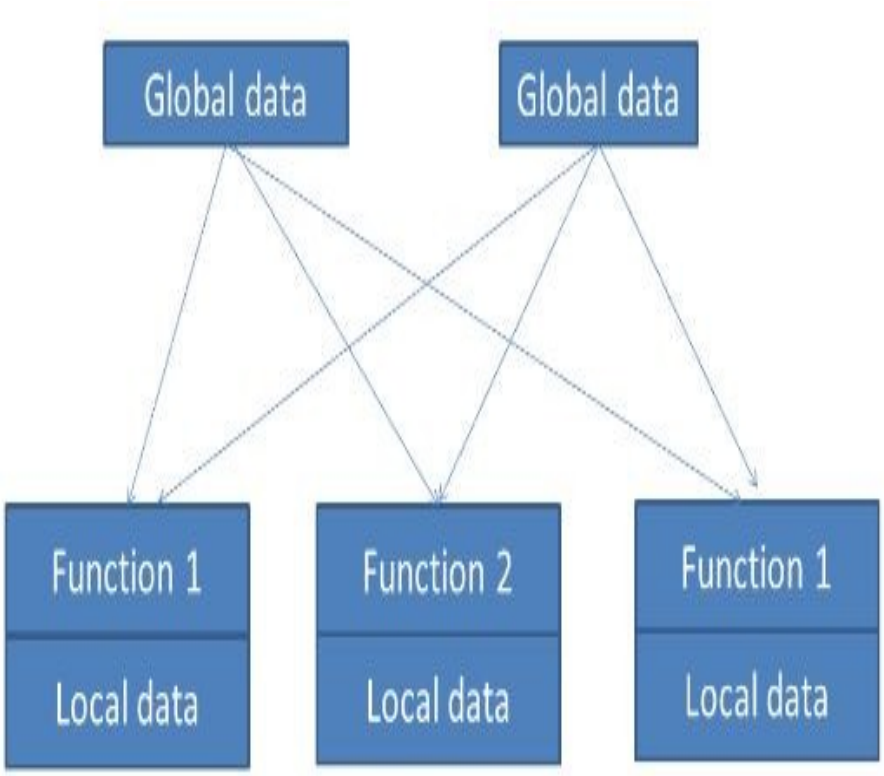
- Associated with *polymorphism* and *inheritance*.

Difference between C and C++

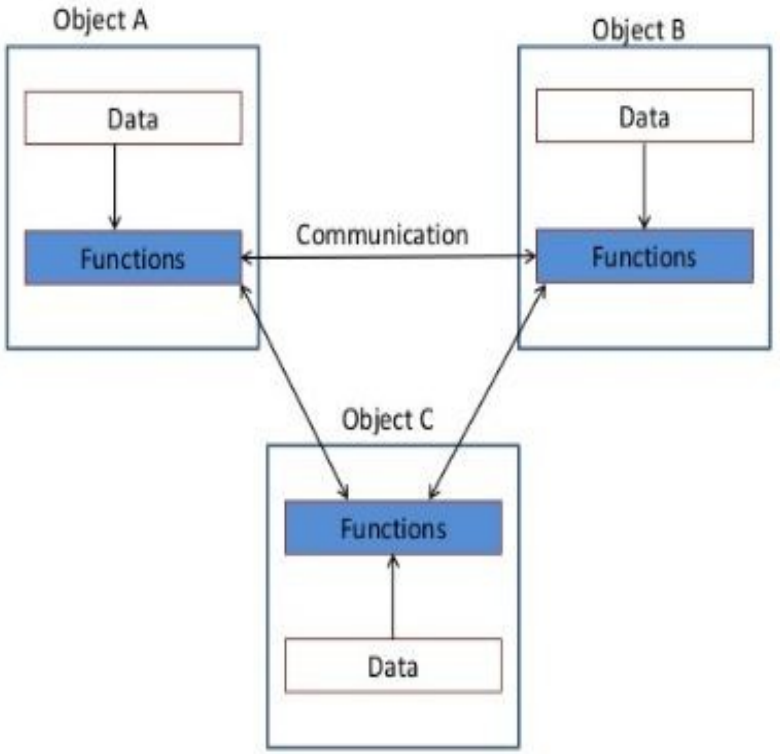
Difference between C and C++

S. No.	Procedural Programming (C)	Object Oriented Programming (C++)
1	Emphasis is on doing things (algorithms).	Emphasis is on data rather than procedure.
2	Large programs are divided into smaller programs in the form of functions.	Programs are divided into objects. Functions that operate together are tied together in the data structure.
3	Most of the functions share global data.	Data is mostly hidden and cannot be accessed by external functions.
4	Data move openly around the system from function to function. Functions transform data from one form to another.	Objects may communicate with each other through functions. New data and functions can be easily added wherever necessary.
5	Employs top-down approach in program design.	Follow bottom-up approach in program design.

Difference between C and C++



Functioning of Procedural Language



Functioning of Object Oriented Programming Language

Difference between C and C++

- A class is an extension of the idea of structure used in C.
- A new way of creating and implementing user-defined data type.

C Structures

- Provide a method for packing together data of different types.
- A convenient tool for handling a group of logically related data items.
- Eg:

```
struct student
{
    char name[20];
    int roll_no;
    float total_marks;
} A;
```

Limitations

- **Cannot treat struct data-type like built in data type.**

  Example:

```
struct Complex
{
    float x;
    float y;
} c1, c2,c3;
```

  Complex nos. c1, c2, c3 can be easily assigned values using dot operator, but we can't directly add or subtract two complex nos. i.e.

  c3 = c1 + c2; // illegal in C.

- **Do not permit *data hiding*. Structure members are public members.**

C++ Classes

- Attempts to bring user defined types as close as possible to the built-in data types.
- Also provides facility to hide the data.
- Inheritance is also supported by C++

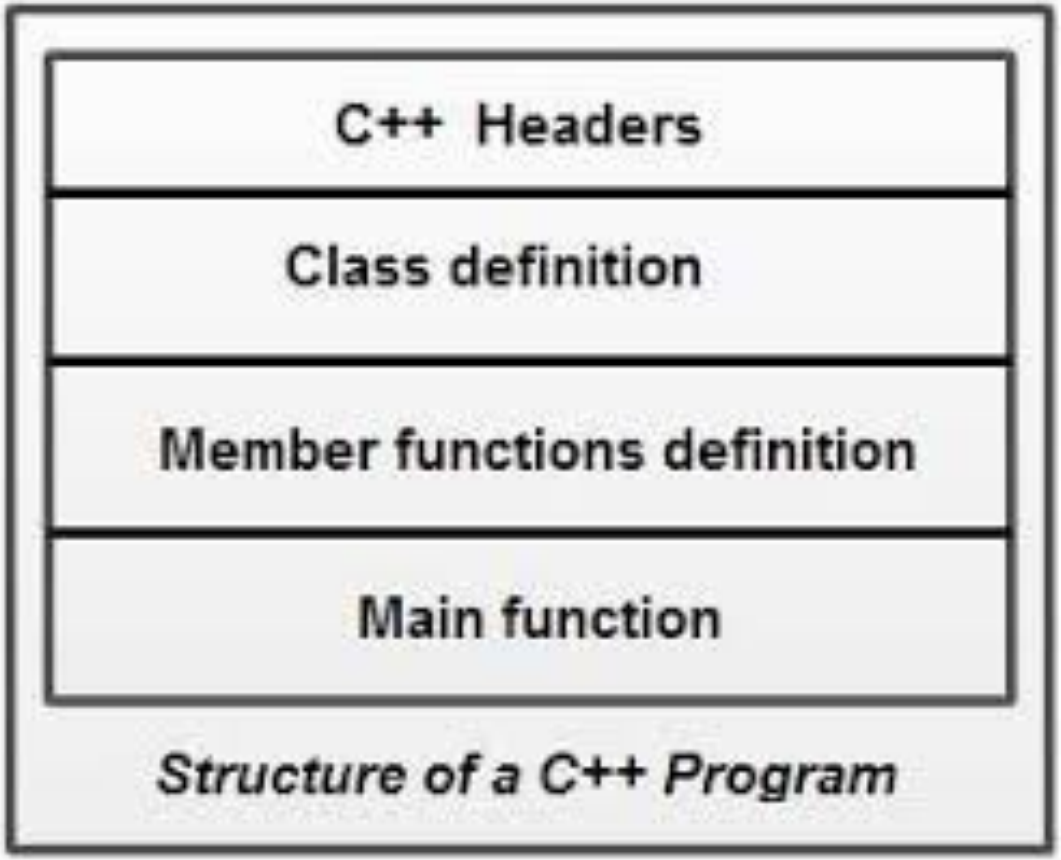
Applications

Applications of OOPs Languages

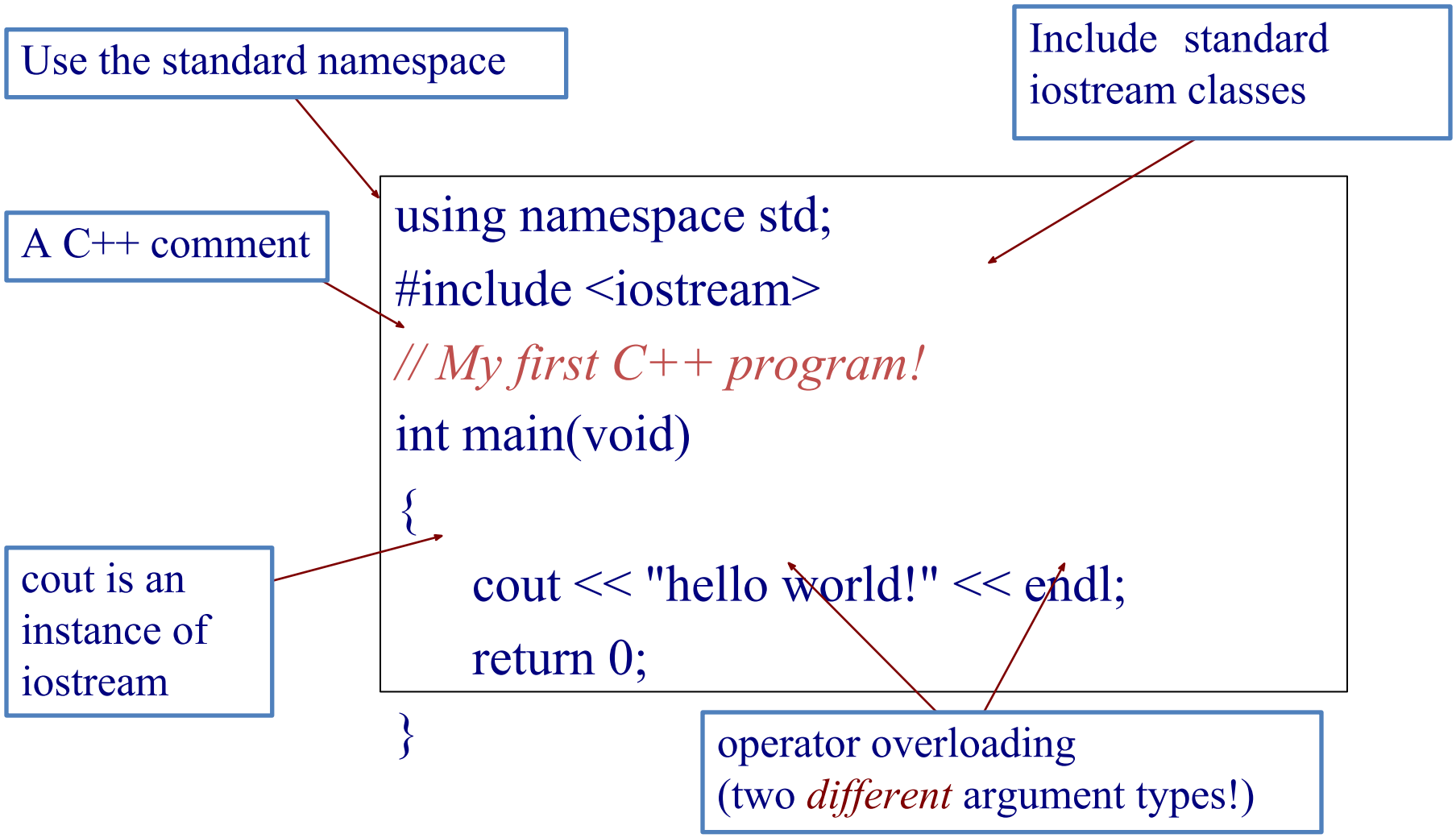
- Real-time systems
- Simulation and Modeling
- Object-oriented Databases
- Hypertext and Hypermedia systems
- AI and expert systems
- Neural Networks and parallel programming
- Decision Support and office automation systems
- CIM/CAM/CAD systems

A simple C++ Program

General Structure of C++ Program



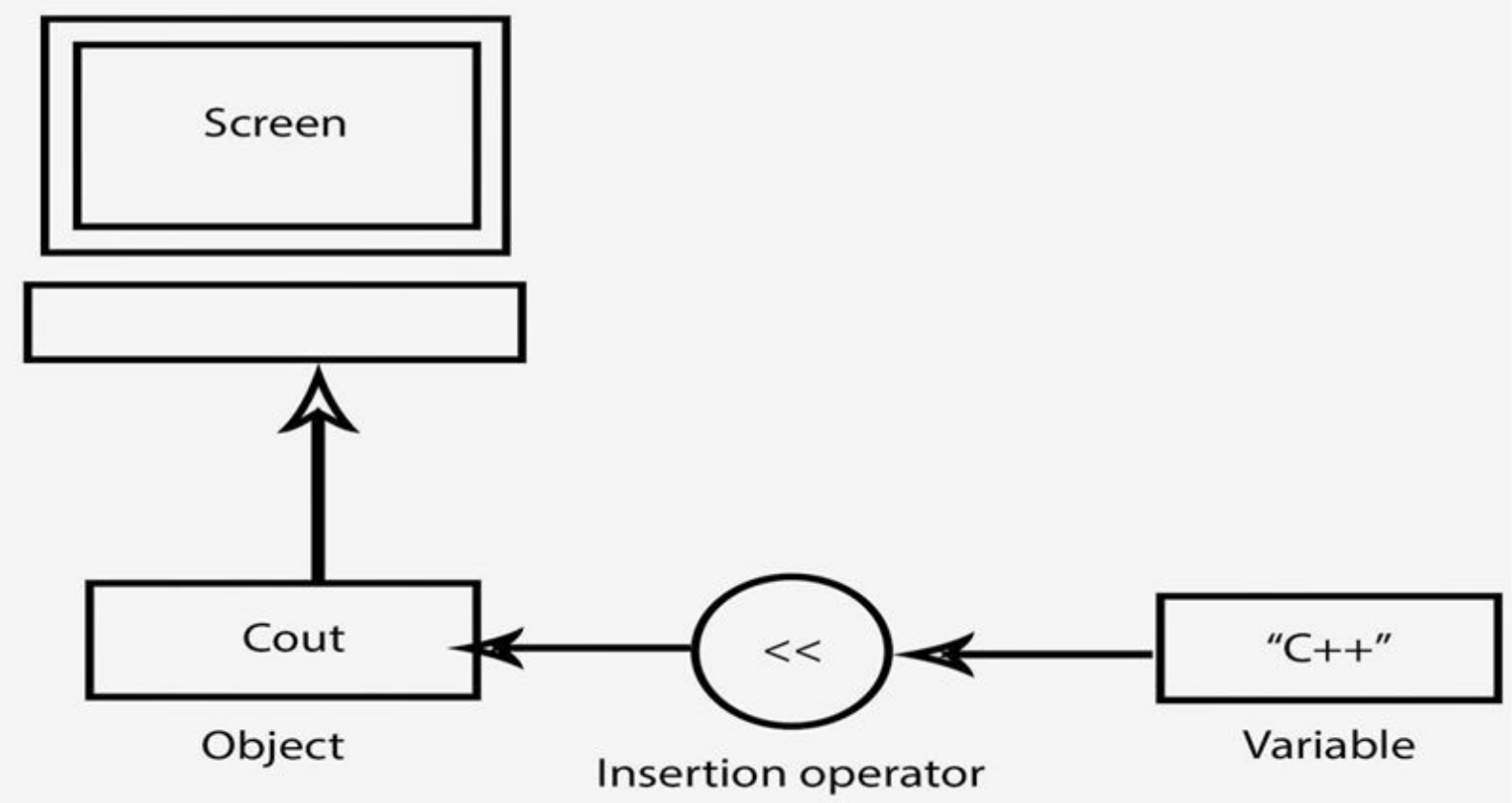
“Hello World” in C++



Console Input / Output

- I/O objects cin, cout
- Defined in the C++ library called <iostream>
- Must have these lines (called pre-processor directives) near start of file:
  - #include <iostream>
    using namespace std;
  - Tells C++ to use appropriate library so we can use the I/O objects cin, cout

Console Output



Output using Insertion Operator

**cout: Standard Output Stream

Console Output

- What can be outputted?
  - Any data can be outputted to display screen
    - Variables
    - Constants
    - Literals
    - Expressions (which can include all of above)
  - cout << numberOfGames << " games played.";

    2 values are outputted:

    "value" of variable numberOfGames,

    literal string " games played."
- Cascading: multiple values in one cout

Separating Lines of Output

- New lines in output
  - Recall: "\n" is escape sequence for the char "newline"
- A second method: object endl
- Examples:

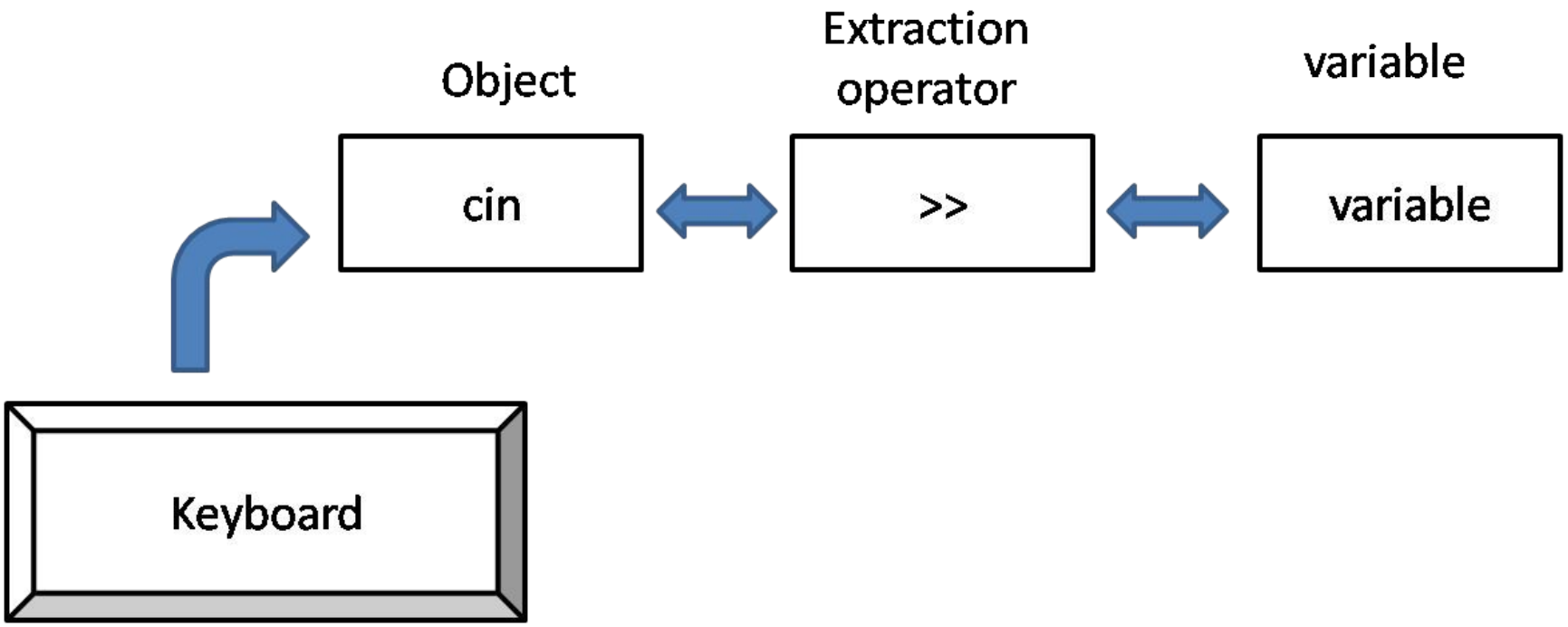
  cout << "Hello World\n";
    - Sends string "Hello World" to display, & escape sequence "\n", skipping to next line

  cout << "Hello World" << endl;
    - Same result as above

Input Using cin

- cin for input, cout for output
- Differences:
  - ">>" (extraction operator) points opposite
    - Think of it as "pointing toward where the data goes"
  - Object name "cin" used instead of "cout"
  - No literals allowed for cin
    - Must input "to a variable"
- cin >> num;
  - Waits on-screen for keyboard entry
  - Value entered at keyboard is "assigned" to num

Console Input



Input using Extraction Operator

**cin: Standard Input Stream

Prompting for Input: cin and cout

- Always "prompt" user for input

  cout << "Enter number of dragons: ";

  cin >> numOfDragons;

  – Note no "\n" in cout. Prompt "waits" on same line for keyboard input as follows:

    Enter number of dragons:

    • Underscore above denotes where keyboard entry is made

Namespaces

- Namespaces defined:
  - Collection of name definitions
- For now: interested in namespace "std"
  - Has all standard library definitions we need
- Examples:

  #include <iostream>
  using namespace std;
    - Includes entire standard library of name definitions
- #include <iostream>using std::cin; using std::cout;
    - Can specify just the objects we want

Thank You