Object Oriented Programming An Introduction

Software Crisis

- Increasing complexity and highly competitive nature of industry arise software crisis.
- Following issues need to be addressed to face this crisis:

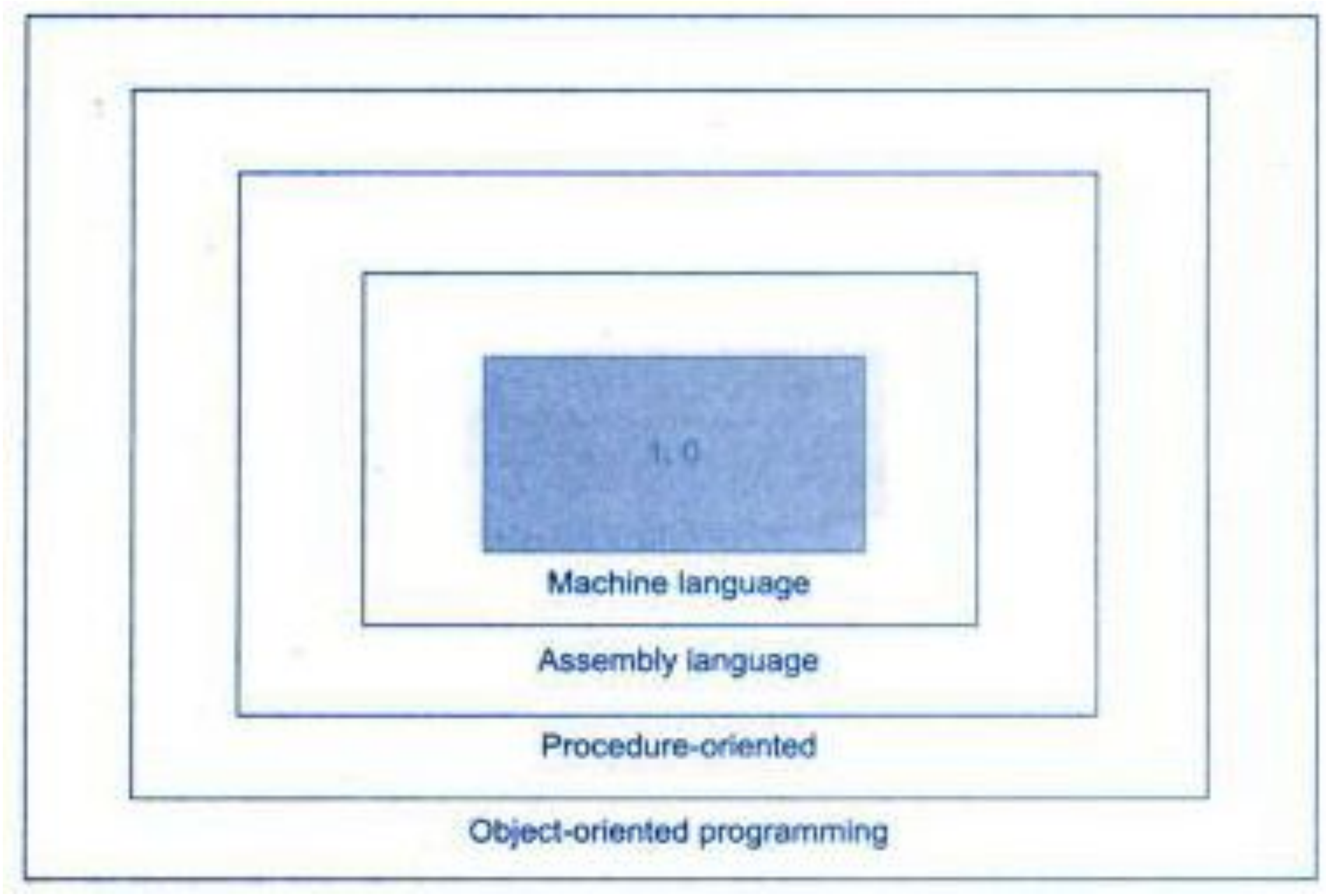
1. How to represent real-life entities of problems in system design
2. How to design systems with open interfaces
3. How to ensure reusability and extensibility of modules
4. How to develop module which can be tolerant to any change in future
5. How to improve software productivity and decrease software cost
6. How to improve quality of software
7. How to manage time schedules

Main Quality Issues

- Correctness
- Maintainability
- Reusability
- Openness and interoperability
- Portability
- Security
- User friendliness

Note:Selection and use of proper software tools would resolve some of these issues.

Software Evolution



Procedure -Oriented Programming

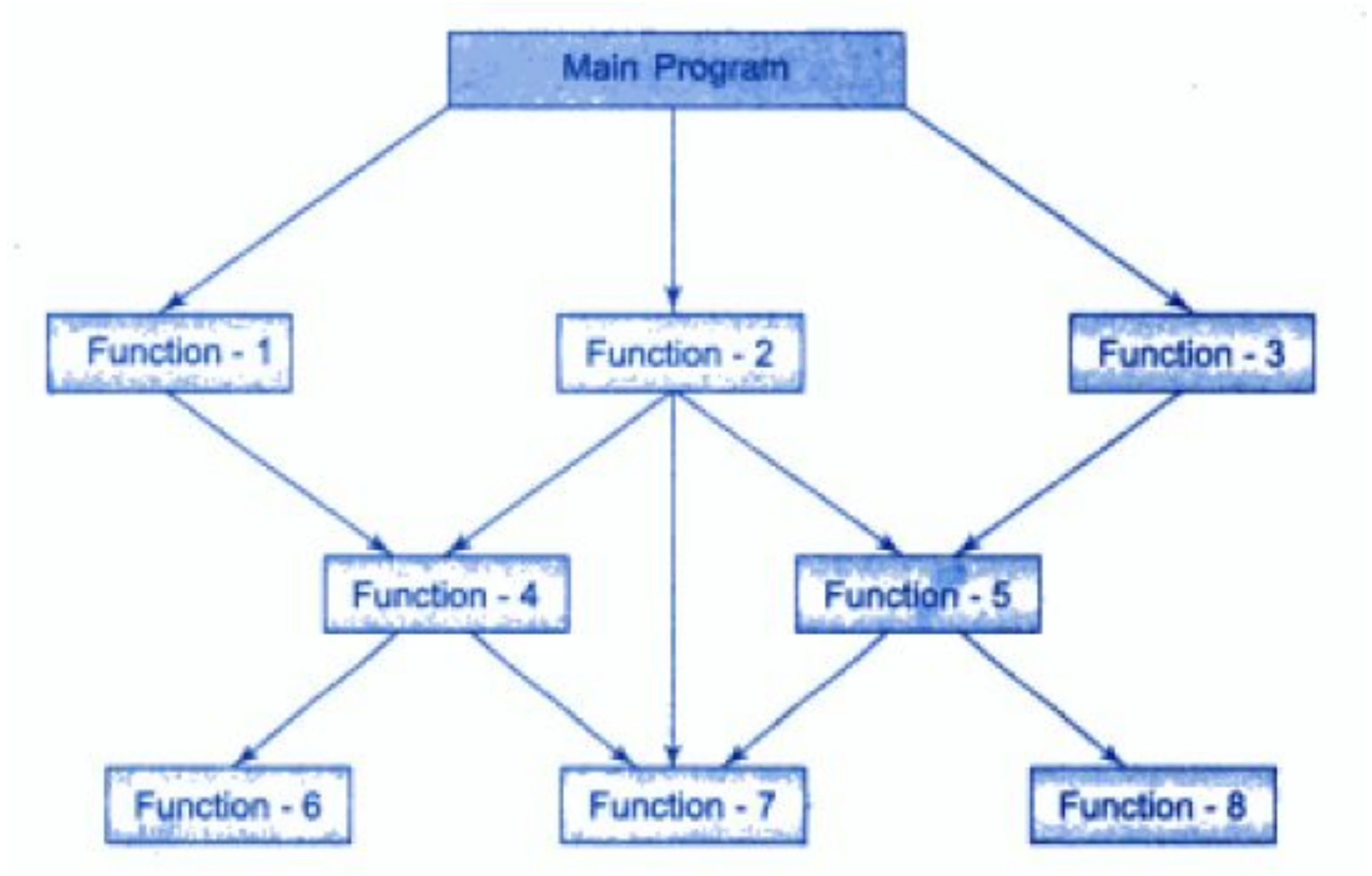
- **Procedure -Oriented Programming (POP) or structured programming :**

1. Each statement in the language tells the computer to do something: Get some input, add these numbers, divide by six, display that output. In simple, a program in a procedural language is a list of instructions.

2. A procedural program is divided into functions, and (ideally, at least) each function has a clearly defined purpose and a clearly defined interface to the other functions in the program.

- POP follows Top down approach
- For very small programs, no other organizing principle (often called a *paradigm) is needed.*
- C, Pascal, FORTRAN are some examples of *POP*

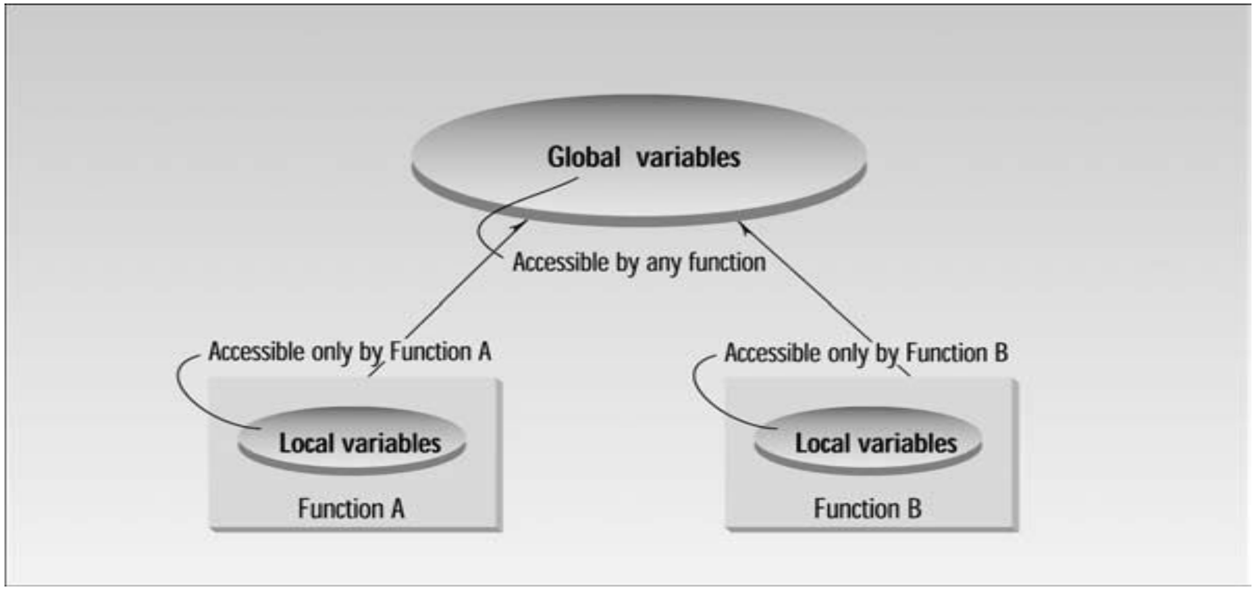
Typical Structure of Procedure-Oriented programs



Problems with structured programming

- No matter how well the structured programming approach is implemented, large programs become excessively complex.
- There are two Main Problems:

1. First, functions have unrestricted access to global data.


Global variables
Accessible by any function
Accessible only by Function A
Accessible only by Function B
Local variables
Function A
Local variables
Function B

Problems with Structured Programming

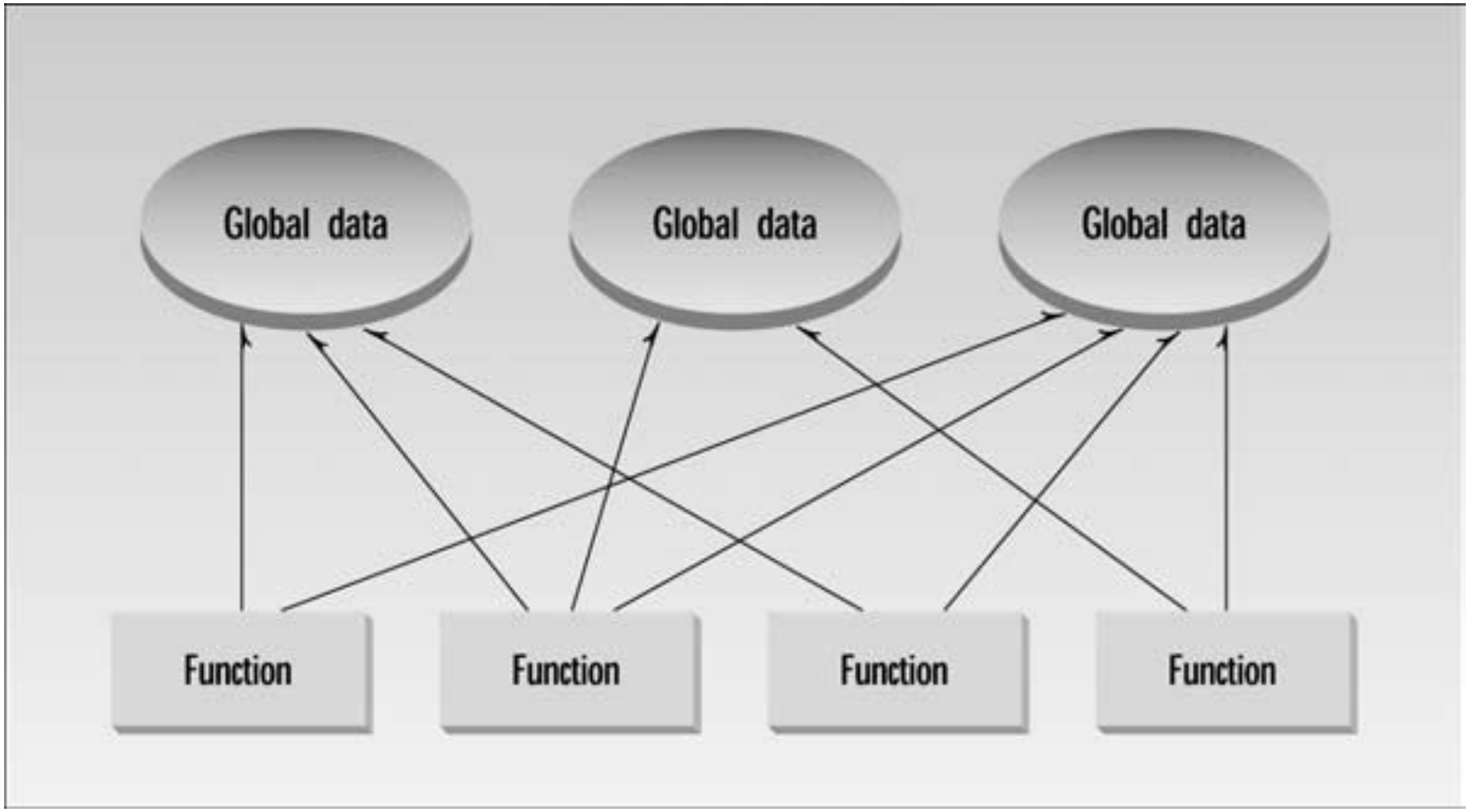


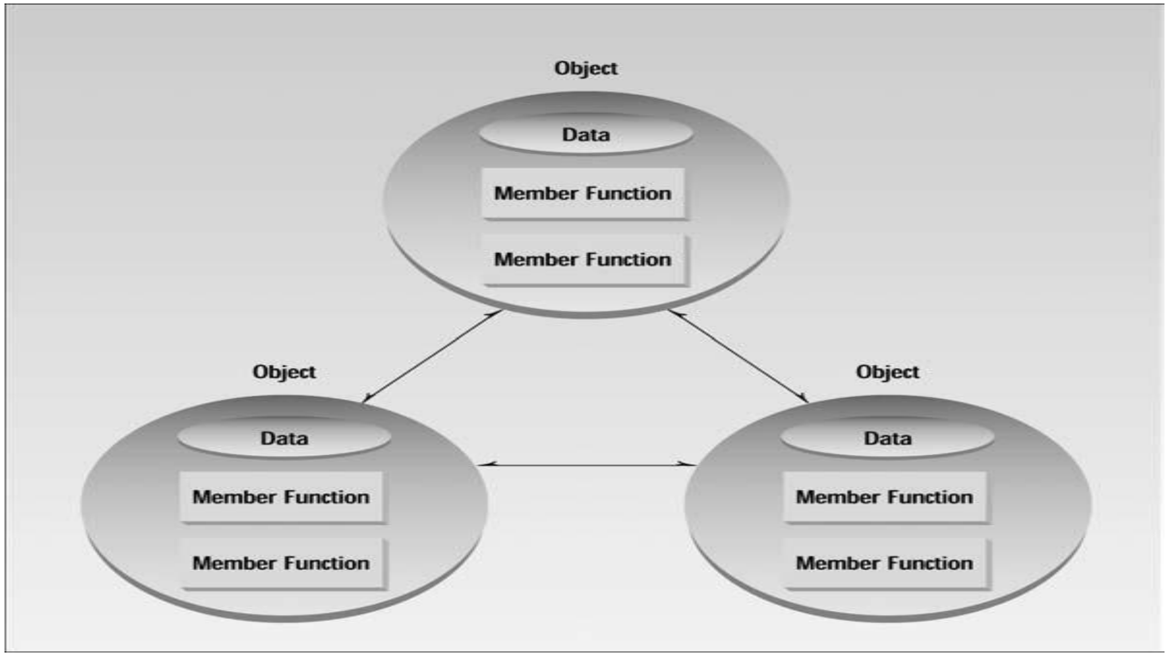
Fig: The procedural paradigm

Problems with Structured Programming

- 2. Not suitable for **Real-World Modeling:**
- The second—and more important—problem with the procedural paradigm is that its arrangement of separate data and functions does a poor job of modeling things in the real world.
- In the physical world we deal with objects such as people and cars. Such objects aren't like data and they aren't like functions.
- Complex real-world objects have both *attributes and behavior*.
- **Attributes:**Examples of attributes (sometimes called *characteristics) are, for people, eye color* and job title; and, for cars, horsepower and number of doors. Attributes in the real world are equivalent to data in a program
- **Behavior:**Behavior is something a real-world object does in response to some stimulus. If you apply the brakes in a car, it will generally stop. Saying something and stopping are examples of behavior.
- Behavior is like a function: you call a function to do something. So neither data nor functions, by themselves, model real-world objects effectively.

Object-oriented programming paradigm

- **Object-oriented programming** (OOP) is a programming paradigm based on the concept of "objects", which may contain data, in the form of fields, often known as attributes; and code, in the form of procedures, often known as methods.



Characteristics of Object-Oriented Languages

1. Class:

- Defines the abstract characteristics of a thing (object), including the thing's characteristics (its attributes, fields or properties) and the thing's behaviors (the "things it can do", or methods, operations or features).
- A class serves as a plan, or blueprint. It specifies what data and what functions will be included in objects of that class.
- Defining the class doesn't create any objects, just as the mere existence of data type int doesn't create any variables.

Characteristics of Object-Oriented Languages(Cont…)

2. **Objects: It** refers to a particular instance of a class where the object can be a combination of variables, functions, and data structures.

- Objects are members of *classes*
- What kinds of things become objects in object-oriented programs?
- **The answer to this is limited only by your imagination.**
- **Four Categories may be.**

I. **Physical objects**
II. **Computer related objects**
III. **Data storage**
IV. **Human entity**

Class	Object
Fruit	apple
Vehicle	car
Animal	cow

Real World Objects Examples

Physical objects	Elements of the computer-user environment	Data-storage constructs	Human entities
Automobiles in a traffic-flow simulation	Windows	Customized arrays	Employees
Electrical components in a circuit-design program	Menus	Stacks	Students
Countries in an	Graphics	Linked lists	Customers

Predict the output?

```
#include <stdlib.h>
#include <stdio.h>
enum {false, true};
int main()
{
   int i = 1;
   do
   {
      printf("%d\n", i);
      i++;
      if (i < 15)
        continue;
   } while (false);

   getchar();
   return 0;
}
```

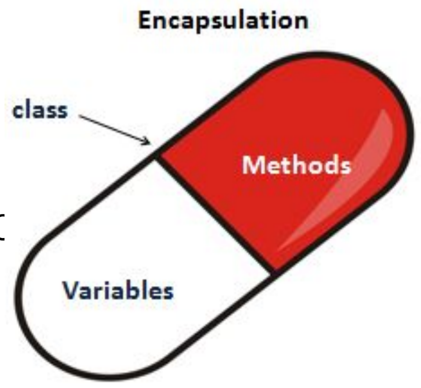
POP vs OOP example

```
int main()
{
int l,b,h;
Calc_Floor_Area(l,b);
Calc_4wall_Area(l,b,h);
}
```

```
class room
{
int l,b,h;
public:
Calc_Floor_Area();
Calc_4wall_Area();
};
```

Encapsulation and data hiding

- **3.Encapsulation** is a process of combining data members and functions in a single unit called class. This is to prevent the access to the data directly, the access to them is provided through the functions of the class.
- **4. Data hiding** is a process of preventing or hiding data from accidental alteration.



- If you want to modify the data in an object, you knc functions interact with it: the member functions in other functions can access the data.

Some Points to remember

- An object's functions, called member functions in C++, typically provide the only way to access its data.
- If you want to read a data item in an object, you call a member function in the object. It will access the data and return the value to you.
- member functions are called methods in some other object-oriented (OO) languages (such as Smalltalk, one of the first OO languages).
- Also, data items are referred to as attributes or instance variables. Calling an object's member function is referred to as sending a message to the object. These terms are not official C++ terminology, but they are used with increasing frequency, especially in object-oriented design.

Cont…

- **5. Data Abstraction:** Data abstraction refers to, providing only essential information to the outside world and hiding their background details, i.e., to represent the required information in program without presenting the details.
- Data abstraction is a programming technique that relies on the separation of interface and implementation.

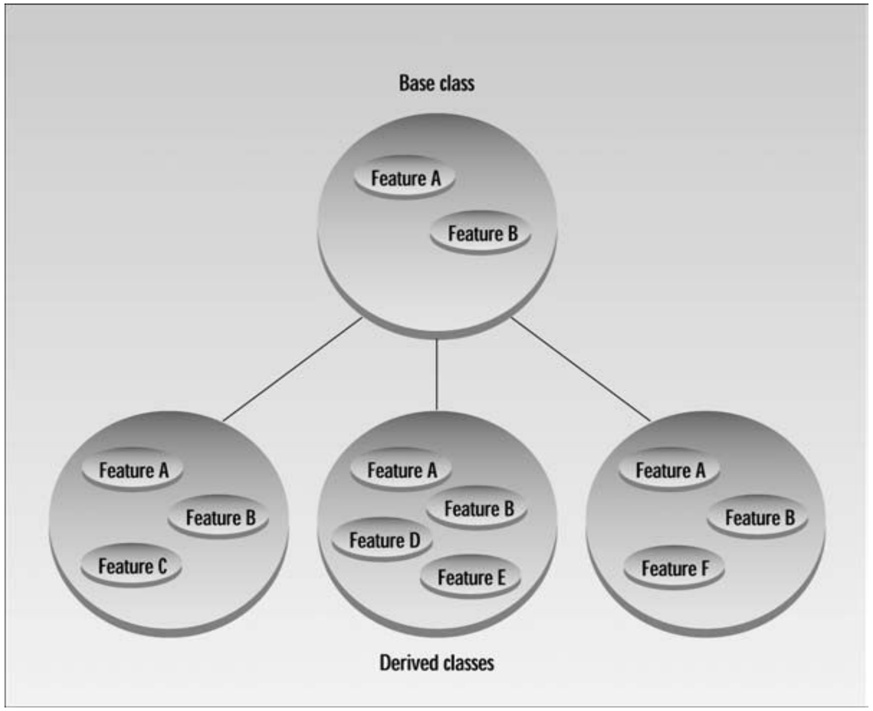
Cont...

- In terms of C++ Programming, C++ classes provides great level of **data abstraction**. 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.

Cont...

6. **Inheritance**

- Inheritance is the process by which objects of one class acquire the properties of objects of another class.



Cont...

- Once a class has been written, created, and debugged, it can be distributed to other programmers for use in their own programs. This is called *reusability.*
- *It is similar to* the way a library of functions in a procedural language can be incorporated into different programs.
- in OOP, the concept of inheritance provides an important extension to the idea of reusability.

Cont...

7. **Polymorphism and Overloading**

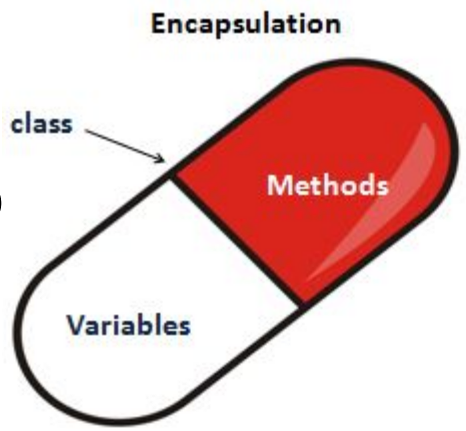
- Poly = many and Morph = form
- So polymorphism is the ability (in programming) to present the same interface for differing underlying forms (data types).
- Using operators or functions in different ways, depending on what they are operating on, is called *polymorphism (one thing with several distinct forms).*
- *When an existing operator, such* as + or =, is given the capability to operate on a new data type, it is said to be *overloaded.*
- The polymorphism is provided in two ways:
  - Function overloading
  - Operator overloading

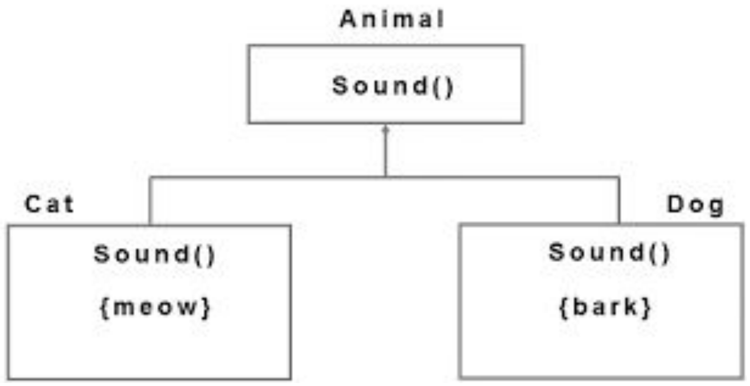
Advantages of Object-Oriented Programming?

1. OOP makes it easy to maintain and modify existing code
2. Objects created for Object-Oriented Programs can easily be reused in other programs. Because it lets you write a set of functions, then expand them in different directions without changing or copying them in any way. (Inheritance)
3. Once an Object is created, knowledge of its implementation is not necessary for its use- Security(abstraction).
4. Objects have the ability to hide certain parts of themselves from programmers(Data hiding). This prevents programmers from tampering with values they shouldn't. For example, a programmer (or another program) cannot set the width of a window to -400.
5. In addition, once a program reaches a certain size, Object Oriented Programs are actually *easier* to program than non-Object Oriented ones.
6. It is suitable for real world problems and real world works
7. It allows you to have many different functions, all with the same name, all doing the same job, but on different data. (Polymorphism)

Summary

- Four Fundamentals of OO
  i) Encapsulation
- ii) Inheritance
- iii) Polymorphism
- iv) Abstraction


Encapsulation
class
Methods
Variables


Animal
Sound()
Cat
Sound()
{meow}
Dog
Sound()
{bark}