Why Do We Need Object-Oriented Programming?

Procedural Languages

- **C, Pascal, FORTRAN, and similar languages are *procedural languages.***
- **A program in a procedural language is a list of instructions. and the computer carries them out.**
- **Few programmers can understand a program of more than a few hundred statements unless it is broken down into smaller units**

Division into Functions

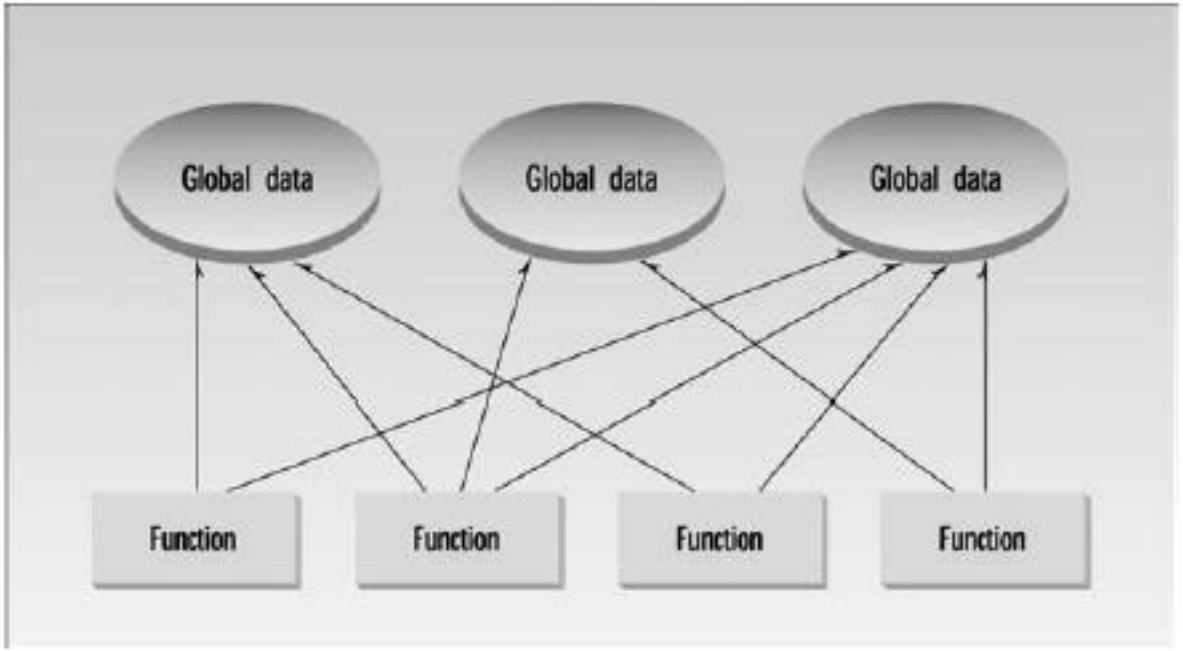
- **the *function* was adopted as a way to make programs more comprehensible to their human creators. In other languages the same concept may be referred**
- **to as a subroutine, a subprogram, or a procedure.) A procedural program is divided into functions, and (ideally, at least) each function has a clearly defined purpose**

Problems with Structured Programming

- As programs grow ever larger and more complex, even the structured programming approach begins to show signs of strain.
- What are the reasons for these problems? There are two related problems.
  - First, functions have unrestricted access to global data.
  - Second, unrelated functions and data,

Unrestricted Access

- In a large program, there are many **functions** and many **global data items**. The problem with the procedural paradigm is that this leads to an even larger number of potential **connections** between functions and data,

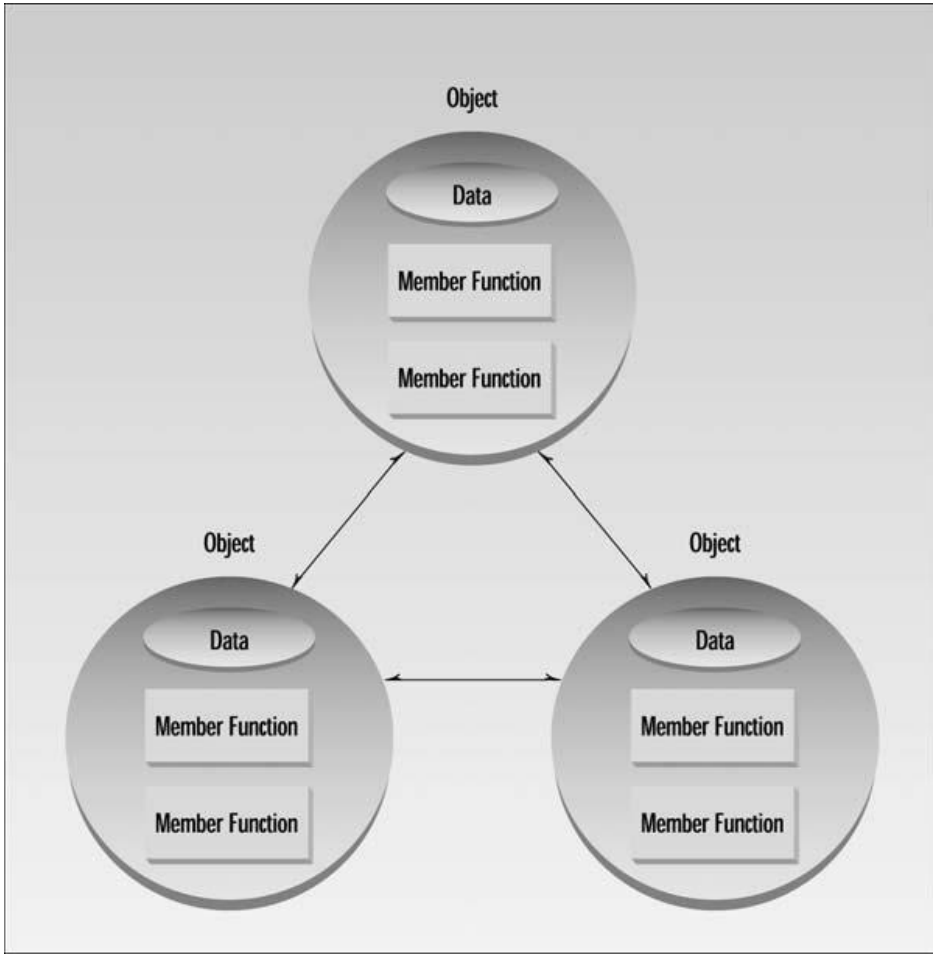

Global data
Global data
Global data
Function
Function
Function
Function

- it makes the program difficult to modify. A change made in a global data item may necessitate rewriting all the functions that access that item.
- someone may decide that the product codes for the inventory items should be changed from 5 digits to 12 digits. This may necessitate a change from a short to a long data type.
- Now all the functions that operate on the data must be modified

- **When data items are modified in a large program it may not be easy to tell which functions access the data, and even when you figure this out, modifications to the functions may cause them to work incorrectly with other global data items.**

The Object-Oriented Approach

- The fundamental idea behind object-oriented languages is to **combine into a single unit both** *data* and the *functions that operate on that data*.
- Such a unit is called an ***object***. An object's **functions**, called ***member functions*** in C++, typically provide the only way to **access its data**.



A C++ program typically consists of a number of objects, which communicate with each other by calling one another's member functions

- **Data** and its **functions** are said to be ***encapsulated*** into a single entity.

- Data **encapsulation** and data **hiding** are key terms in the description of **object-oriented languages.**

Classes

- In **OOP** we say that **objects** are **members of *classes***
- **class** serves as a **plan**, or **blueprint**. It specifies **what** **data** and what **functions** will be included in **objects** of that **class**.
- A **class** is thus a **description** of a number of **similar objects**.