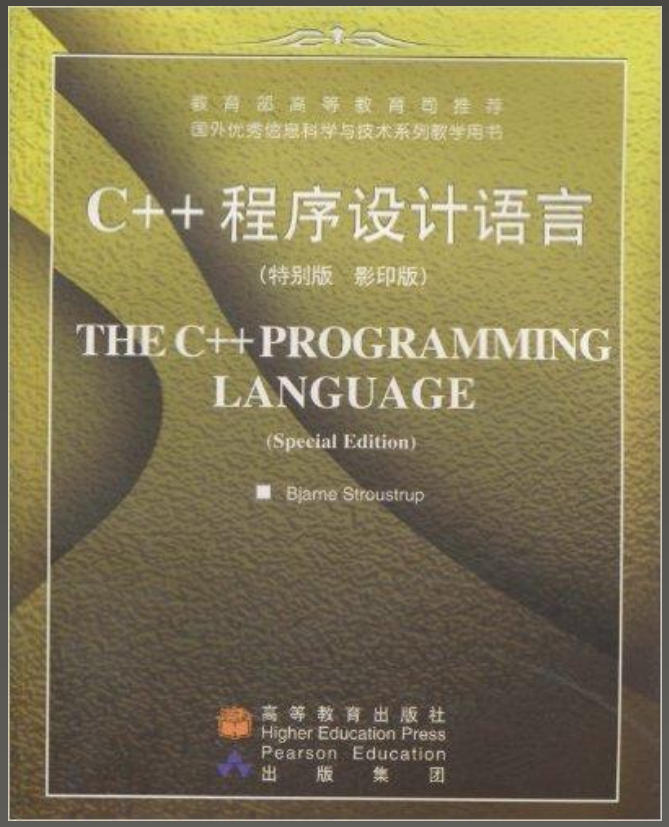
OBJECT-ORIENTED PROGRAMMING

Jin Rong
February 2016

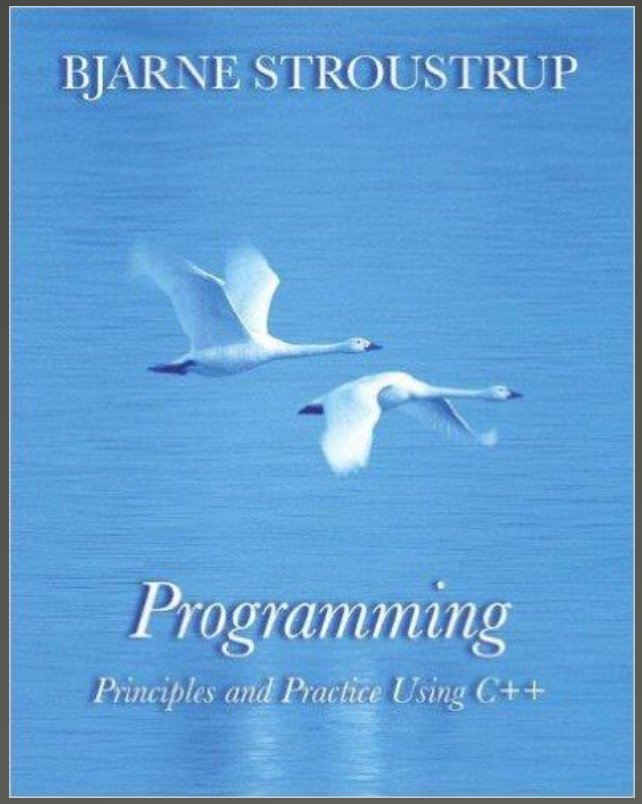
Textbook

- The C++ Programming Language (Special Edition)
  Bjarne Stroustrup



References

- Programming : Principles and Practice Using C++
  Bjarne Stroustrup



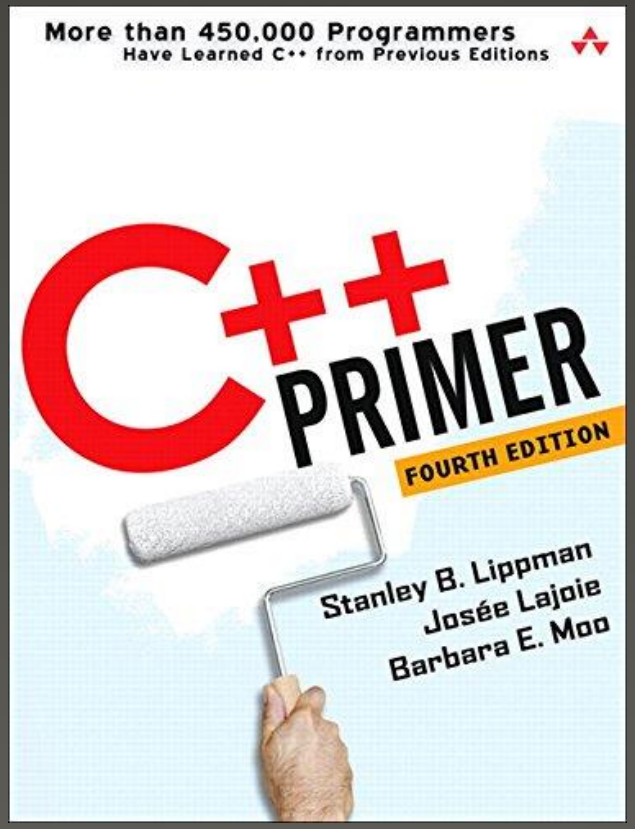
References

- C++ PRIMER

  Stanley B. Lippman

  Josée Lajoie

  Barbara E. Moo

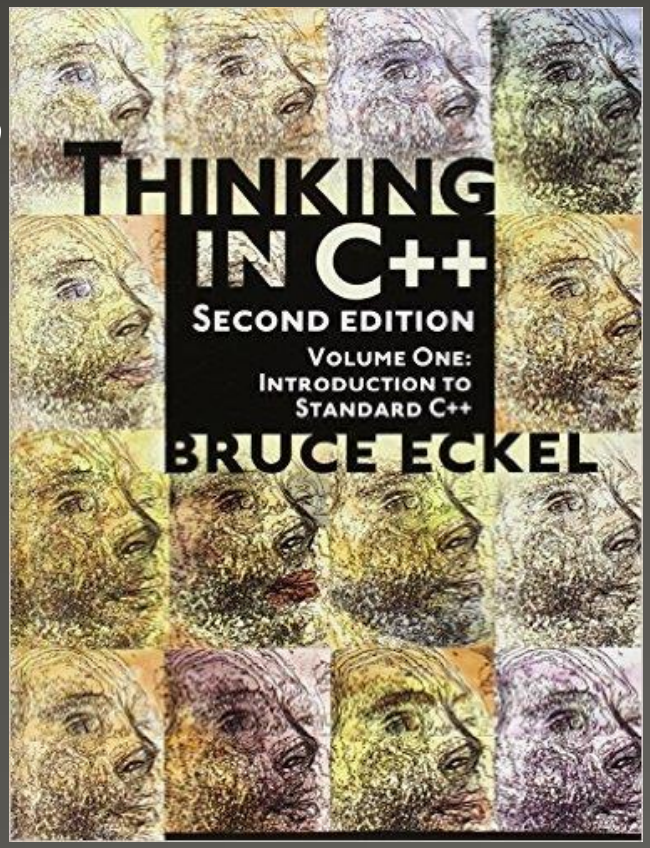


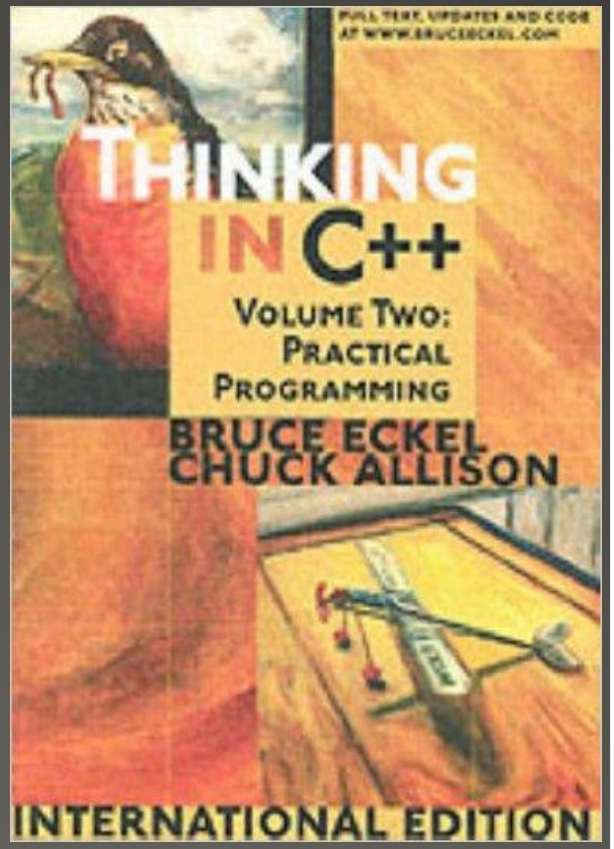
References

- THINKING IN C++

  Bruce Eckel

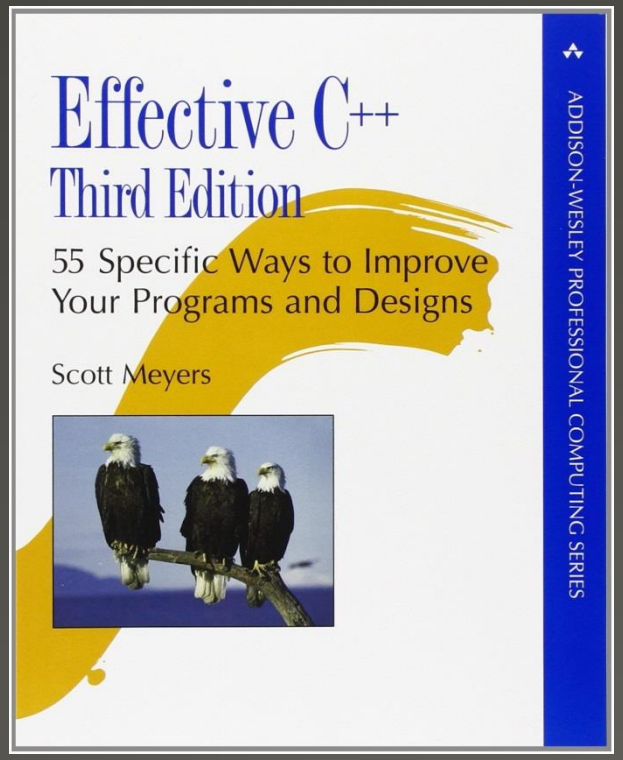
  Chuck Allison

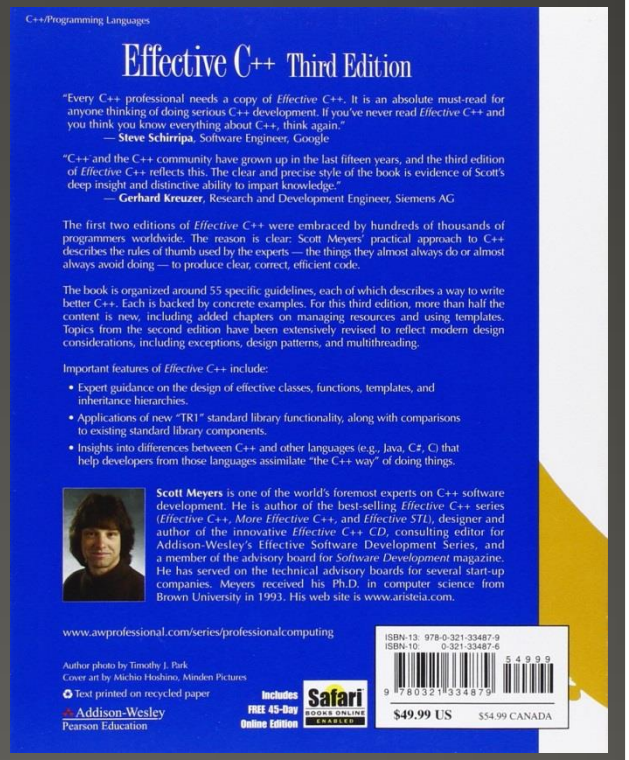




References

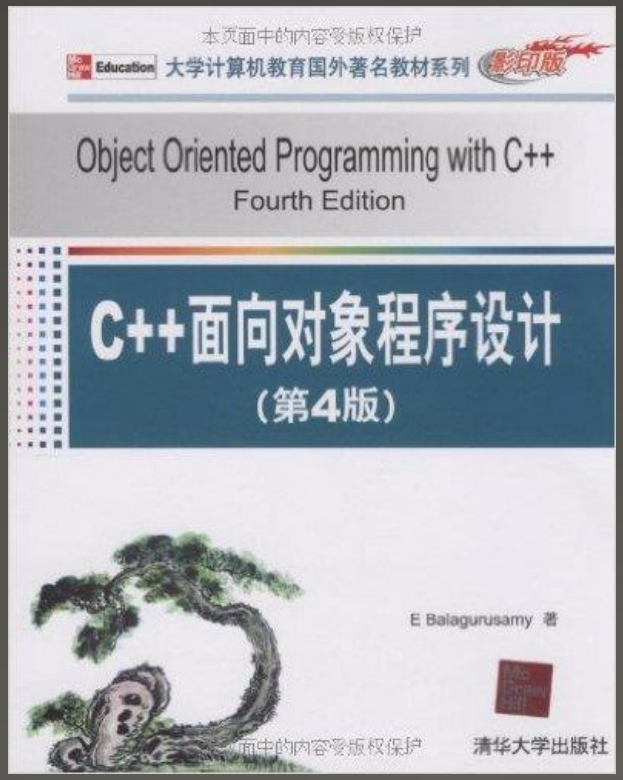
- Effective C++ (Third Edition)
  Scott Meyers





References

- Object Oriented Programming with C++ (Fourth Edition)
  E Balagurusamy



About the course

- Total Periods: 48
  - Teaching Periods : 39
    - Week : 1,2,3,4,7,8,11,12,13,15,16
    - 2-S210
  - Practicing Periods : 9
    - Week : 5,9,14
    - Computerroom

Final Grade Assignment

- Each student will receive a numeric score for the course, based on a weighted average of the following
  - Attendance : 10%
  - Assignments : 30%
  - Final Exam : 60%
- Couse Design

Policies

- Working Alone on Assignments
  - You will work on all assignments by yourself, with the possible exception of the assignment, where you might be allowed to work with a partner.
- Handing in Late Assignments
  - The penalty for late assignments is 10% per day.
- Mobile devices and other distractions
  - Please turn the sound off so that you do not disrupt other students' learning.

Lesson 1

Introduction to Object-Oriented Programming

Evolution of Programming Languages

Evolution of Programming Languages

- Machine languages
  - Machine dependent
  - Example
    - A section of early machine-language program that adds overtime pay to base pay and stores the result in gross pay

```
+1300042774
+1400593419
+1200274027
```

Evolution of Programming Languages

- Assembly languages
  - Using Englishlike abbreviations to represent elementary operations.
  - Translators
    - Convert assembly-language programs to machine languages at computer speeds.
  - Example
    - A section of an assembly-language also adds overtime pay to base pay and stores the result in gross pay

```
load    basepay
add     overpay
store   grosspay
```

Evolution of Programming Languages

- High-level languages
  - Single statement can accomplish substantial tasks.
  - Example
    - A payroll program written in a high-level language

```
grossPay = basePay + overTimePay
```

Evolution of Programming Languages

- Procedure-oriented Programming
  - Structure Programming
    - Data and algorithm are separate

```
int hour;
int minute;
int second;
Void SetTime()
{… }
Void DisplayTime()
{…}
```

Procedure-Oriented Programming(POP)

- The problem is viewed as a sequence of things to be done such as reading, calculating and printing.
- A number of functions are written to accomplish these tasks these tasks.
- The primary focus is on functions.

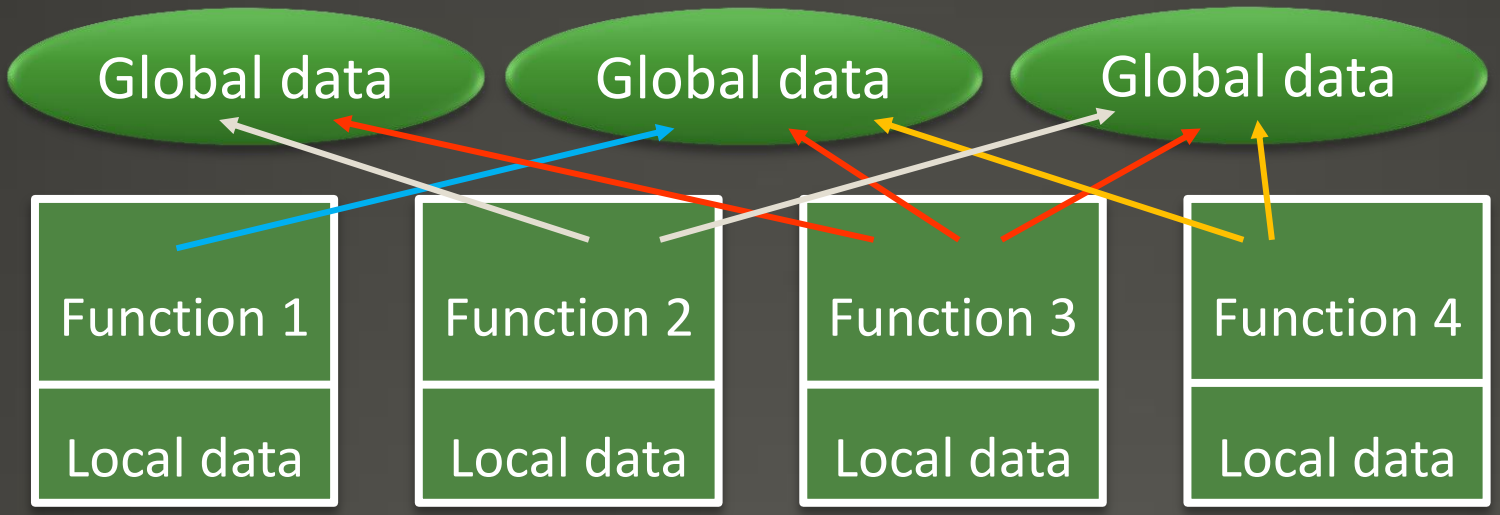
Procedure-Oriented Programming(POP)

- While we concentrate on the development of functions, very little attention is given to the data that are being used by various functions.
- What happens to the data?
- How are they affected by the functions that work on them?

Procedure-Oriented Programming(POP)

- In a multi-function program, many important data items are placed as **global** so that they may be accessed by all the functions.
- Each function may have its own **local data**.

Procedure-Oriented Programming(POP)



Relationship of data and functions in procedural programming

Procedure-Oriented Programming(POP)

- Global data are more vulnerable to an inadvertent change by a function.
- In a large program it is very difficult to identify what data is used by which function.
- In case we need to revise an external data structure, we also need to revise all functions that access the data.
- This provides an opportunity for bugs to creep in.

Procedure-Oriented Programming(POP)

- Another drawback with the procedural approach is
  - It does not model real world problems very well.
  - This is because functions are action-oriented and do not really correspond to the element of the problem.

Procedure-Oriented Programming(POP)

- Some characteristics exhibited by procedure-oriented programming are
  - Emphasis is on doing things (algorithms).
  - Large programs are divided into smaller programs known as functions.
  - Most of the functions share global data.
  - Data move openly around the system from function to function.
  - Functions transform data from on form to another.
  - Employs top-down approach in program design.

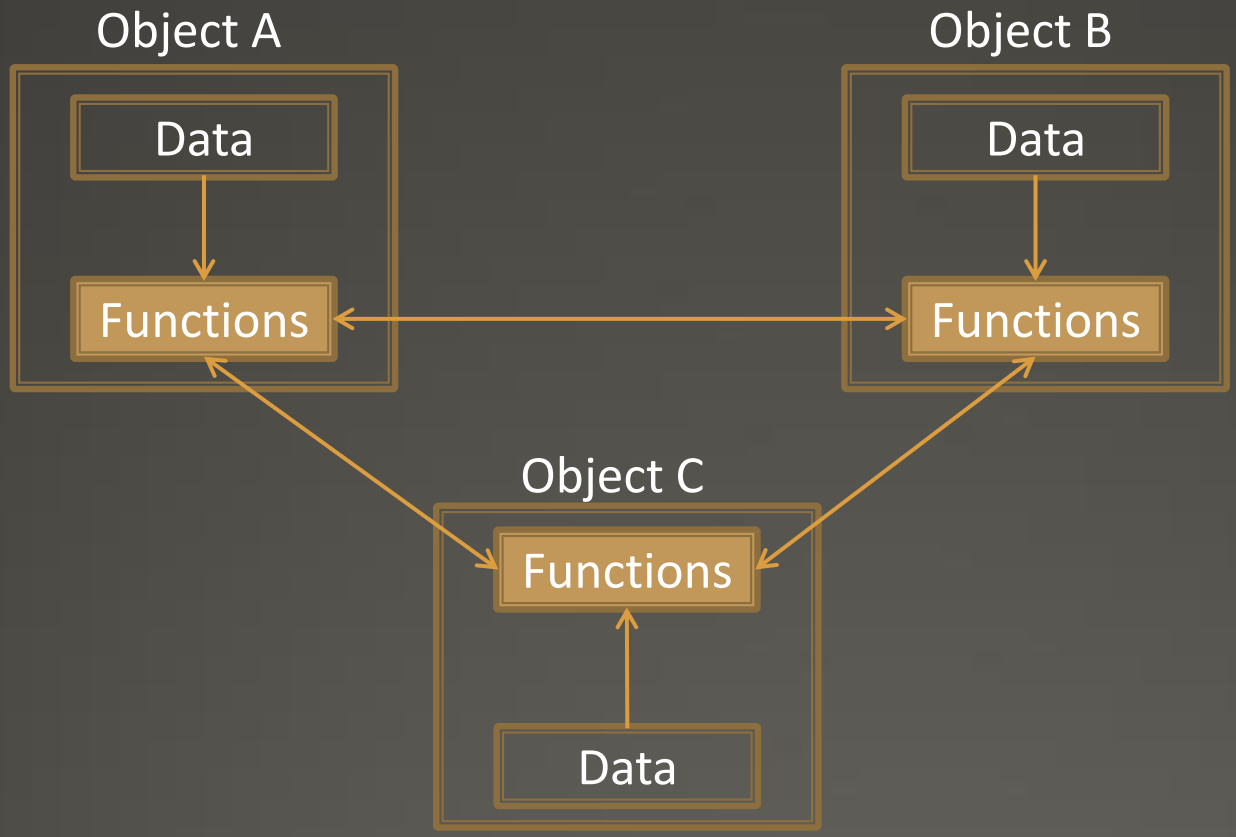
Object-Oriented Programming Paradigm

- The major motivating factor in the invention of object-oriented approach is to remove some of the flaw encountered in the procedural approach.
- OOP treats data as a critical element in the program development and does not allow it to flow freely around the system.
- OOP ties data more closely to the functions that operate on it, and protects it from accidental modification from outside functions.

Object-Oriented Programming Paradigm

- OOP allows decomposition of a problem into a number of entities called ***objects*** and then builds data and functions around these objects.

Object-Oriented Programming Paradigm



Organization of data and functions in OOP

Object-Oriented Programming Paradigm

- The data of an object can be accessed only by the functions associated with that object.
- However, functions of one object can access the functions of other objects.

Object-Oriented Programming Paradigm

- Some of the striking features of object-oriented programming are
  - Emphasis is on data rather than procedure.
  - Programs are divided into what are known as objects.
  - Data structures are designed such that they characterize the objects.
  - Functions that operate on the data of an object are tied together in the data structure.
  - Data is hidden and cannot be accessed by external functions.
  - Objects may communicate with each other through functions.

Object-Oriented Programming Paradigm

- Some of the striking features of object-oriented programming are
  - New data and functions can be easily added whenever necessary.
  - Follows bottom-up approach in program design.

Object-Oriented Programming Paradigm

- Definition of object-oriented programming
  - An approach that provides a way of modularizing programs by creating partitioned memory area for both data and functions that can be used as templates for creating copies of such modules on demand.
  - Thus, an object is considered to be a partitioned area of computer memory that stores data and set of operations that can access that data.
  - Since the memory partitions are independent, the objects can be used in a variety of different programs without modifications.

Object-Oriented Programming Paradigm

```
Class Clock
{
  private:
     int hour;
     int minute;
     int second;
  public:
     void SetTime();
     void DisplayTime();
};
```

Basic Concepts of OOP

- Objects
- Classes
- Data abstraction and encapsulation
- Inheritance
- Polymorphism
- Dynamic binding
- Message passing

Basic Concepts of OOP

- Objects
  - Basic run-time entities in an object-oriented system.
  - They may represent a person, a place, a bank account, a table of data or any item that the program has to handle.
  - They may also represent user-defined data such as vectors, time and lists.
  - Programming problem is analyzed in terms of objects and the nature of communication between them.
  - Program objects should be chosen such that they match closely with the real-world objects.
  - Objects take up space in the memory and have an associated address like a record in Pascal, or a structure in C.

Basic Concepts of OOP

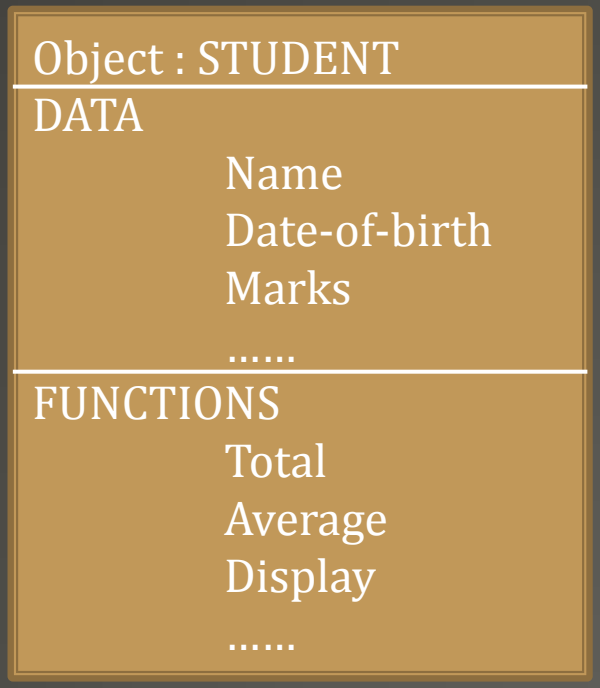
- Objects
  - When a program is executed, the objects interact by sending **messages** to one another.
  - For example
    if “customer” and “account” are two objects in a program, then the customer object may send a message to the account object requesting for the bank balance.

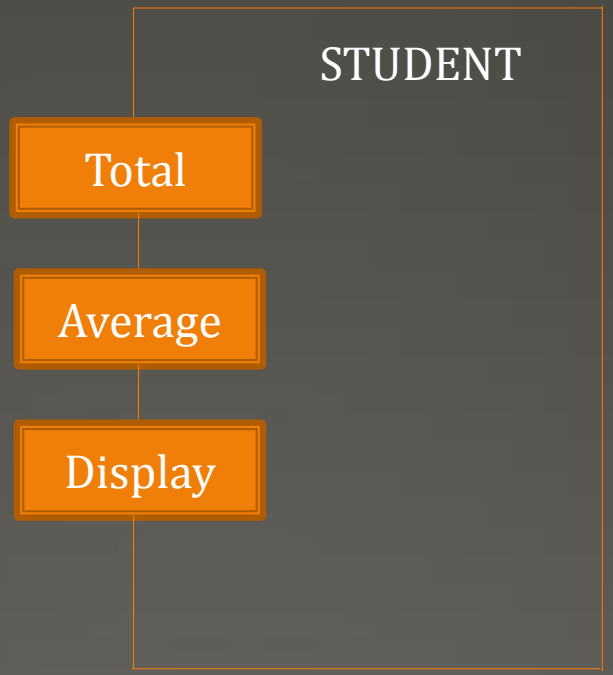
Basic Concepts of OOP

- Objects
  - Each object contains **data**, and **code** to manipulate the data.
  - Objects can interact without having to know details of each other's data or code.
  - It if sufficient to know the type of **message** accepted, and the type of response returned by the objects.

Basic Concepts of OOP

- Objects
  - Two notations that are popularly used in OOP analysis and design





Basic Concepts of OOP

- Classes
  - In fact, objects are variables of the type ***class***.
  - Once a class has been defined, we can create any number of objects belonging to that class.
  - Each object is associated with the data of type class with which they are created.
  - A class is thus a collection of objects of similar type.
  - For example
    mango, apple and orange are members of the class fruit.

Basic Concepts of OOP

- Classes
  - Classes are user-defined data types and behave like the built-in types of a programming language.
  - The syntax used to create an object is no different than the syntax used to create an integer object in C.
  - If fruit has been defined as a class, then the statement

```
fruit mango;
```

  - Will create an object **mango** belonging to the class fruit.

Basic Concepts of OOP

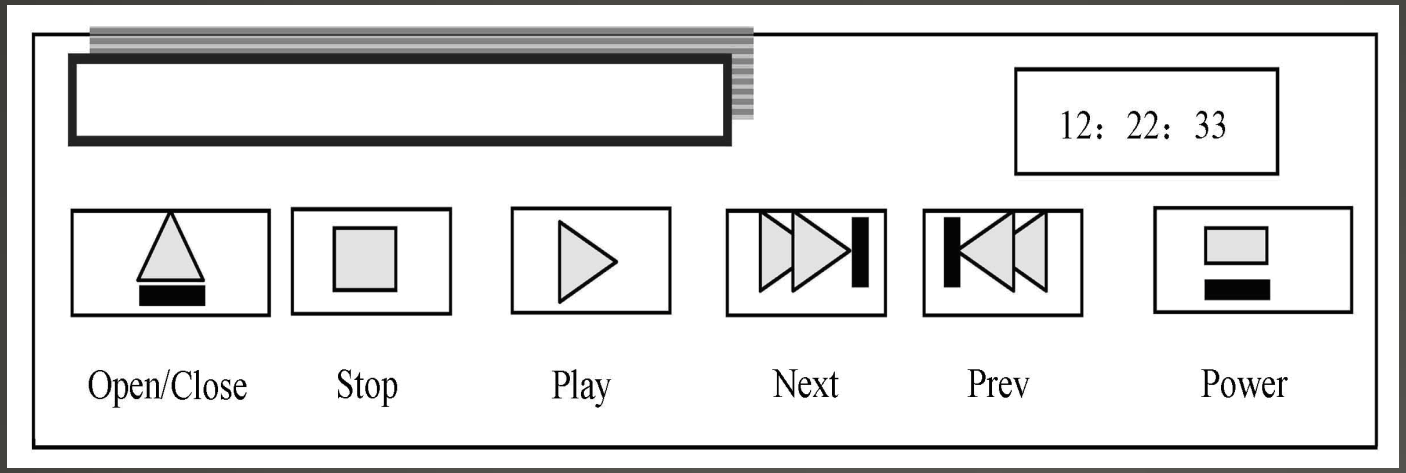
- Data Abstraction and Encapsulation
  - **Encapsulation** : The wrapping up of data and functions into a single unit (called class) is known as encapsulation.
  - Data encapsulation is the most striking feature of a class.
  - The data is not accessible to the outside world, and only those functions which are wrapped in the class can access it.
  - These functions provide the interface between the object's data and the program.
  - This insulation of the data from direct access by the program is called ***data hiding*** or ***information hiding***.

Basic Concepts of OOP

- Data Abstraction and Encapsulation
  - **Abstraction** refer 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 cost, and ***functions*** to operate on these attributes.
  - The ***attributes*** are sometimes called ***data members*** because they hold information.
  - The ***functions*** are operate on these data are sometimes called ***methods*** or ***member functions***.

Basic Concepts of OOP

- Data Abstraction and Encapsulation

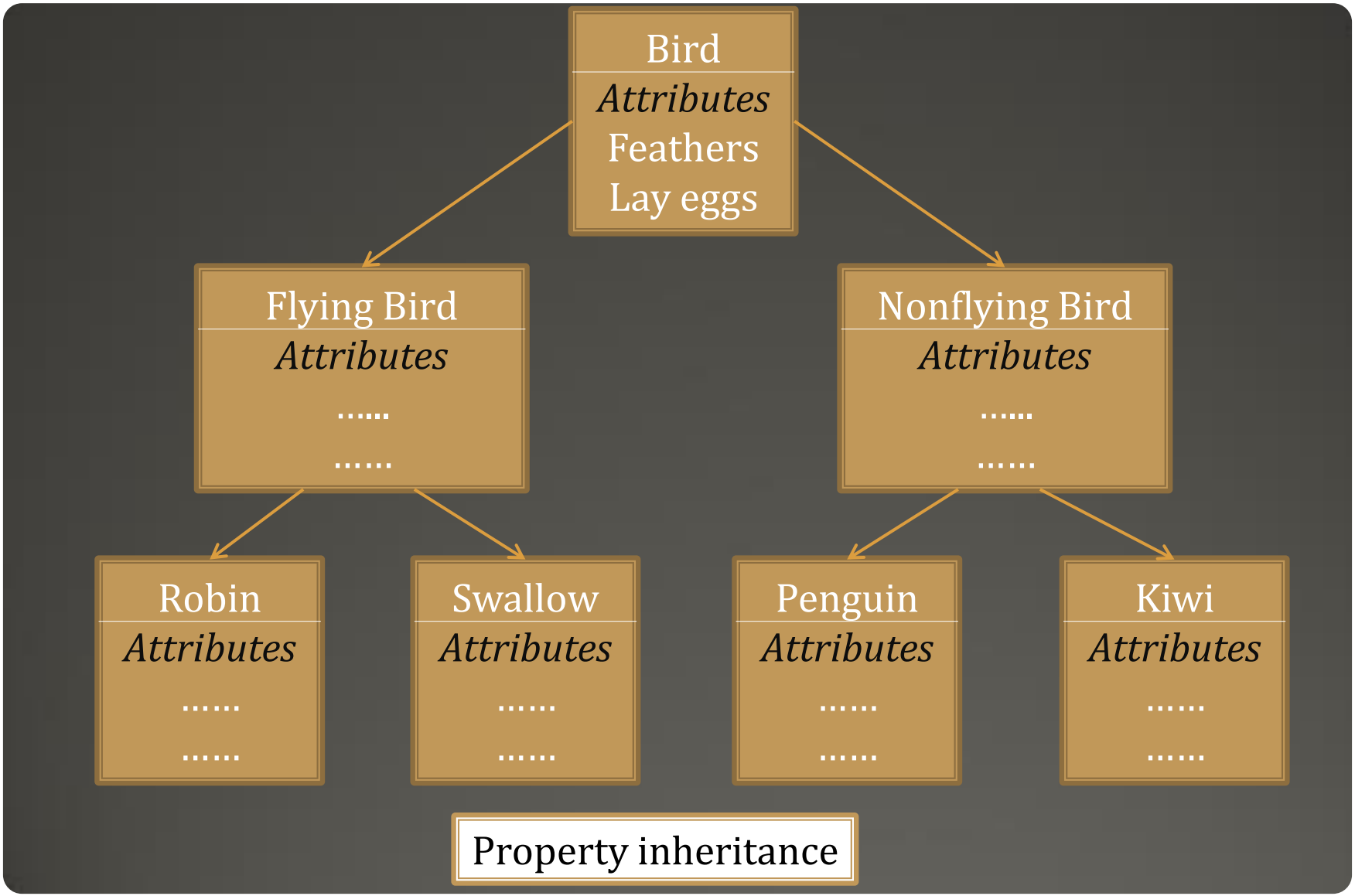


Basic Concepts of OOP

- Data Abstraction and Encapsulation
  - Since the classes use the concept of data abstraction, they are known as ***Abstract Data Types*** (**ADT**).

Basic Concepts of OOP

- Inheritance
  - Inheritance is the process by which objects of one class acquire the properties of objects of another class.
  - It support the concept of ***hierarchical classification***.
  - For example
    the bird 'robin' is a part of the class 'flying bird' which is again a part of the class 'bird'.
  - The principle behind this sort of division is that each derived class shares common characteristics with the class from which it is derived.


Bird
Attributes
Feathers
Lay eggs
Flying Bird
Attributes
.....
.....
Nonflying Bird
Attributes
.....
.....
Robin
Attributes
.....
.....
Swallow
Attributes
.....
.....
Penguin
Attributes
.....
.....
Kiwi
Attributes
.....
.....
Property inheritance

Basic Concepts of OOP

- Inheritance
  - In OOP, the concept of inheritance provides the idea of ***reusability***.
  - This means that we can add additional features to an existing class without modifying it.
  - This is possible by deriving a new class from the existing one.
  - The new class will have the combined features of both the classes.

Basic Concepts of OOP

- Inheritance
  - The appeal and power of the inheritance mechanism is that it allows the programmer to ***reuse*** a class that is almost, but not exactly, when he wants, and to tailor the class in such a way that it does not introduce any undesirable side-effects into the rest of the classes.

Basic Concepts of OOP

- Inheritance
  - Note that each sub-class defines only those features that are unique to it.
  - Without the use of classification, each class would have to explicitly include all of its features.

Basic Concepts of OOP

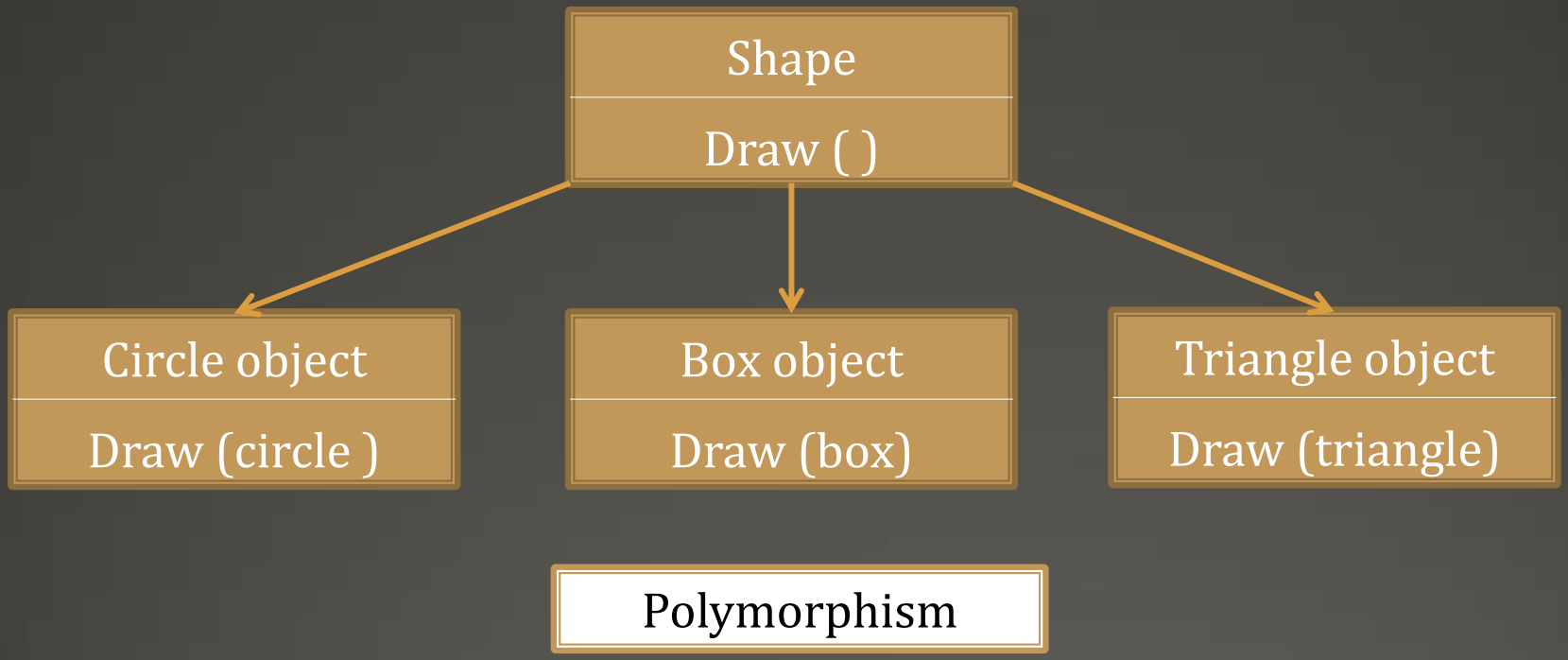
- Polymorphism
  - A Greek term, means the ability to take more than one form.
  - An operation may exhibit different behaviours in different instances.
  - The behaviour depends upon the types of data used in the operation.

Basic Concepts of OOP

- Polymorphism
  - For example, consider the operation of addition. For two numbers, the operation will generate a sum. If the operands are strings, then the operation would produce a third string by concatenation.
  - The process of making an operator to exhibit different between different behaviours in different instances is known as ***operator overloading***.

Basic Concepts of OOP

- Polymorphism
  - A 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 meanings depending on the context.
  - Using a single function name to perform different types of tasks in known as ***function overloading***.


Shape
Draw ( )
Circle object
Draw (circle)
Box object
Draw (box)
Triangle object
Draw (triangle)
Polymorphism

Basic Concepts of OOP

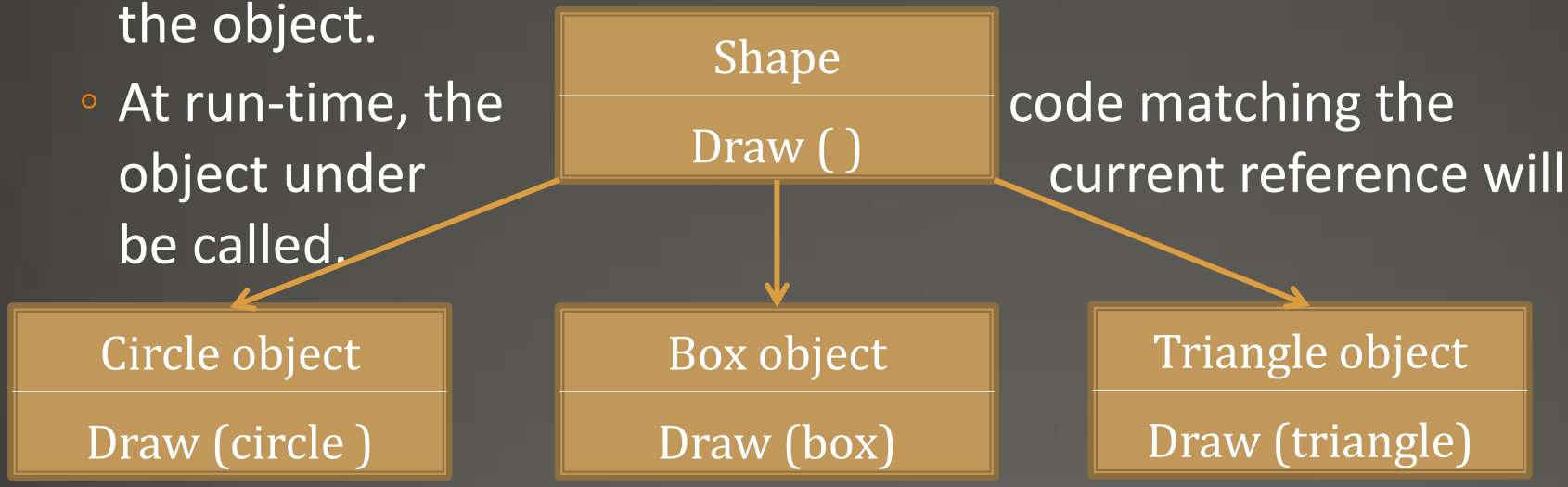
- Polymorphism
  - Polymorphism plays an important role in allowing objects having different internal structures to share the same external interface.
  - This means that a general class of operations may be accessed in the same manner even though specific actions associated with each operation may differ.
  - Polymorphism is extensively used in implementing inheritance.

Basic Concepts of OOP

- Dynamic Binding
  - ***Dynamic binding*** (also known as ***late binding***) means that the code associated with a given procedure call is not known until the time of the call at run-time.
  - It is associated with polymorphism and inheritance.
  - A function call associated with a polymorphic reference depends on the dynamic type of that reference.

Basic Concepts of OOP

- Dynamic Binding
  - Consider the procedure “draw”.
  - By inheritance, every object will have this procedure.
  - Its algorithm is, however, unique to each object and so the draw procedure will be redefined in each class that defines the object.
  - At run-time, the object under code matching the current reference will be called.



Basic Concepts of OOP

- Message Passing
  - An object-oriented program consists of a set of objects that communicate with each other.
  - The process of programming in an object-oriented language, therefore, involves the following basic steps:
    1. Creating classes that define objects and their behaviour,
    2. Creating objects from class definitions, and
    3. Establishing communication among objects.

Basic Concepts of OOP

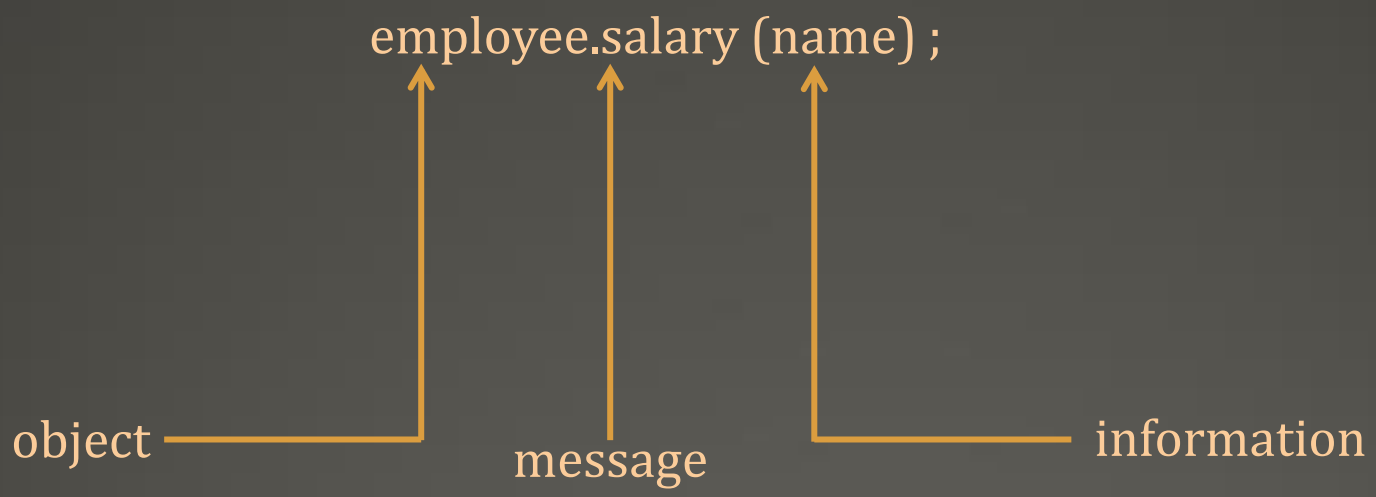
- Message Passing
  - Objects communicate with one another by sending and receiving information much the same way as people pass messages to one another.
  - The concept of ***message passing*** makes it easier to talk about building systems that directly model or simulate their real-world counterparts.

Basic Concepts of OOP

- Message Passing
  - A message for an object is a request for execution of a procedure, and therefore will invoke a function (procedure) in the receiving object that generates the desired result.
  - ***Message passing*** involves specifying the name of the object, the name of the function (message) and the information to be sent.

Basic Concepts of OOP

- Message Passing
  - Example



Basic Concepts of OOP

- Message Passing
  - Objects have a life cycle.
  - They can be created and destroyed.
  - Communication with an object is feasible as long as it is alive.

Benefits of OOP

- Through ***inheritance***, we can eliminate redundant code and extend the use of existing classes;
- We can build programs from the standard working modules that communicate with one another, rather than having to start writing the code from scratch.
  - This leads to saving of development time and higher productivity.
- The principle of ***data hiding*** helps the programmer to build secure programs that cannot be invaded by code in other parts of the program.

Benefits of OOP

- It is possible to have multiple instances of an object to co-exist without any interference.
- It is possible to map objects in the problem domain to those in the program.
- It is easy to partition the work in a project based on objects.
- The data-centered design approach enables us to capture more details of a model in implementable form.

Benefits of OOP

- Object-oriented systems can be easily upgraded from small to large systems.
- ***Message passing*** techniques for communication between objects makes the interface descriptions with external systems much simpler.
- Software complexity can be easily managed.

Object-Oriented Languages

- Depending upon the features they support, they can be classified into the following two categories:
  - Object-based programming languages
  - Object-oriented programming languages

Object-Oriented Languages

- Object-based programming languages
  - **Object-based programming** is the style of programming that primarily supports encapsulation and object identity.
  - Major features that are required for object-based programming are :
    - Data encapsulation
    - Data hiding and access mechanisms
    - Automatic initialization and clear-up of objects
    - Operator overloading

Object-Oriented Languages

- Object-based programming languages
  - Languages that support programming with objects are said to be object-based programming language.
  - They do not support inheritance and dynamic binding.
  - Ada is a typical object-based programming language.

Object-Oriented Languages

- Object-oriented programming languages
  - **Object-oriented programming** incorporates all of object-based programming features along with two additional features, namely, inheritance and dynamic binding.
  - Object-oriented programming can therefor be characterized by the following statement :

    Object-based features + inheritance + dynamic binding

  - Languages that support these features include C++, Smalltalk, Object Pascal and Java.

Applications of OOP

- The most popular application of object-oriented programming, up to now, has been in the area of user interface design such as windows.
  - Hundreds of windowing systems have been developed, using the OOP techniques.

Applications of OOP

- The promising areas for application of OOP include :
  - Real-time systems
  - Simulation and modeling
  - Object-oriented databases
  - Hypertext, hypermedia and expertext
  - AI and expert systems
  - Neural networks and parallel programming
  - Decision support and office automation system
  - CIM/CAM/CAD systems

Review Questions

1. What is procedure-oriented programming? What are its main characteristics?
2. Discuss an approach to the development of procedure-oriented programs.
3. Describe how data are shared by functions in a procedure-oriented program.
4. What is object-oriented programming? How is it different from the procedure-oriented programming?
5. How are data and functions organized in an object-oriented program?

Review Questions

6. What are the unique advantages of an object-oriented programming paradigm?
7. Distinguish between the following terms :
   a) Objects and classes
   b) Data abstraction and data encapsulation
   c) Inheritance and polymorphism
   d) Dynamic binding and message passing
8. What kinds of things can become objects in OOP?
9. Describe inheritance as applied to OOP?
10. What do you mean by dynamic binding? How is it useful in OOP?

Review Questions

11. How does object-oriented approach differ from object-based approach?
12. List a few areas of application of OOP technology.

Beginning with C++

- What is C++?
- Applications of C++
- A Simple C++ Program
- More C++ Statements
- An Example with Class
- Structure of C++ Program
- Creating the Source File
- Compiling and Linking

What is C++?

- C++ is an object-oriented programming language.
- It was developed by Bjarne Stroustrup at AT&T Bell Laboratories in Murray Hill, New Jersey, USA, in the early 1980's.
- C++ is an extension of C with a major addition of the class construct feature of Simula67.
- Since the class was a major addition to the original C language, Stroustrup initially called the new language 'C with classes'.

What is C++?

- The idea of C++ comes from the C increment operator ++, thereby suggesting that C++ is an augmented (incremented) version of C.
- C++ is a superset of C.
- Most of what we already know about C applies to C++ also.
- The most important facilities that C++ adds on to C are classes, inheritance, function overloading, and operator overloading.

What is C++?

- The object-oriented features in C++ allow programmers to build large programs with clarity, extensibility and ease of maintenance, incorporating the spirit and efficiency of C.
- The addition of new features has transformed C from a language that currently facilitates top-down, structured design, to one that provides bottom-up, object-oriented design.

Applications of C++

- Since C++ allows us to create hierarchy-related objects, we can build special object-oriented libraries which can be used later by many programmers.
- While C++ is able to map the real-world problem properly, the C part of C++ gives the language the ability to get close to the machine-level details.
- C++ programs are easily maintainable and expandable. When a new feature needs to be implemented, it is very easy to add to the existing structure of an object.

A Simple C++ Program

- Prints a string on the screen.(Example 2-1)

```
/ 2-1.cpp
#include<iostream>                          //include header file
using namespace std;
int main()
{
        cout<<"C++ is better than C.\n";   //C++ Statement
        return 0;
}                                          //End of example
```

A Simple C++ Program

- Program Features
  - Like C, the C++ program is a collection of functions.
  - The example 2-1 contains only one function, main().
  - Every C++ program must have a main().
  - C++ is a free-form language.
  - With a few exceptions, the compiler ignores carriage returns and white spaces.
  - Like C, the C++ statements terminate with semicolons.

A Simple C++ Program

- Comments
  - C++ introduces a new comment symbol // (double slash).
  - Comments start with a double slash symbol and terminate at the end of the line.
  - A comment may start anywhere in the line, and whatever follows till the end of the line is ignored.
  - Note that there is no closing symbol.

A Simple C++ Program

- Comments
  - The double slash comment is basically a single line comment.
  - Multiline comments can be written as follows :

```
//This is an example of
//C++ program to illustrate
//Some of its features
```

  - The C comment symbols /*,*/ are still valid and are more suitable for multiline comments.

```
/*      This is an example of
        C++ program to illustrate
        Some of its features
*/
```

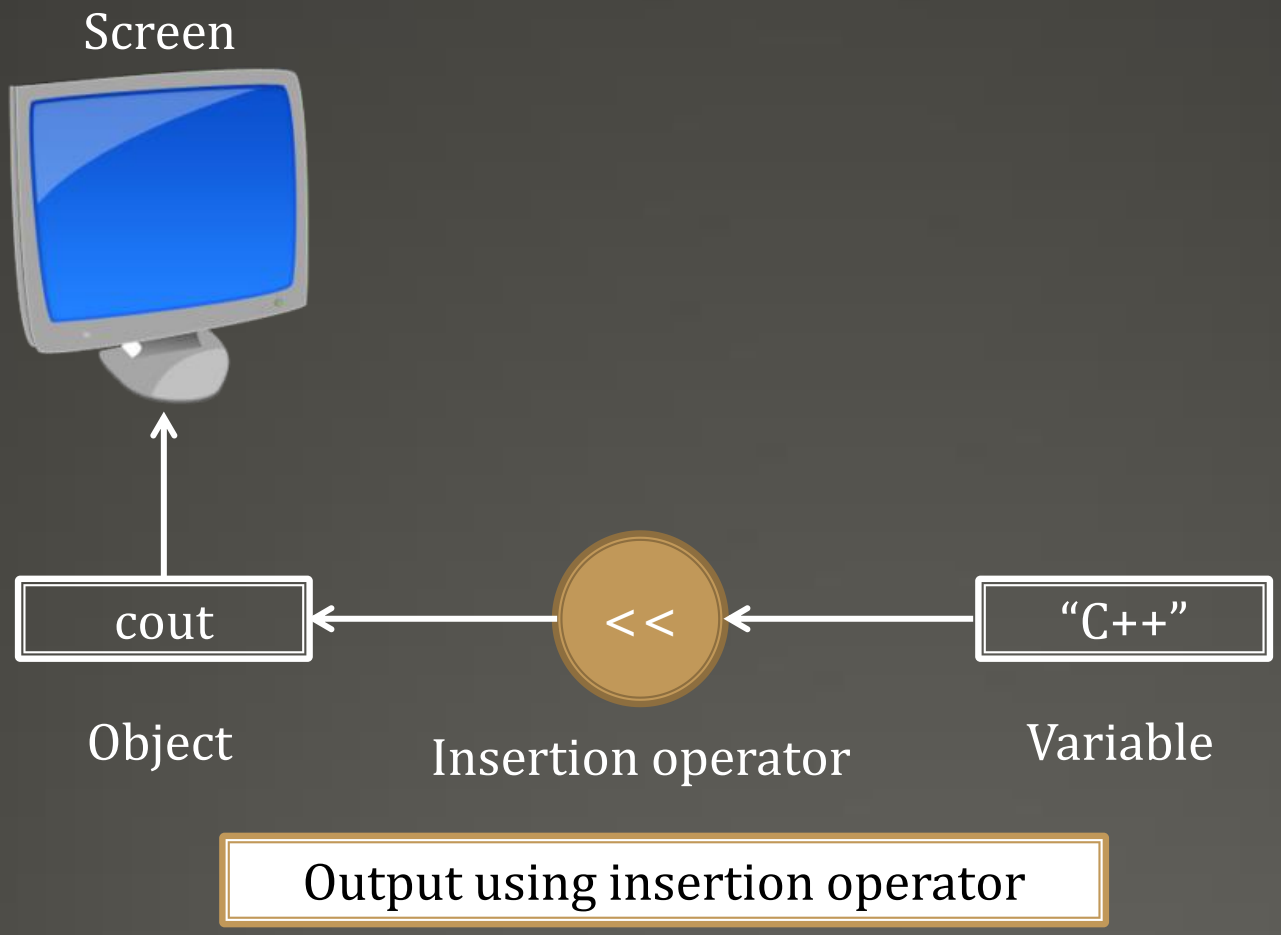
A Simple C++ Program

- Comments
  - Remember that we can not insert a // style comment within the text of a program line.

A Simple C++ Program

- Output Operator
  - The statement causes the string in quotation marks to be displayed on the screen.
  - This statement introduces two new C++ features, cout and <<.
  - The identifier cout (pronounced as “C out”) is a predefined object that represents the standard output stream in C++.
  - Here, the standard output stream represents the screen.
  - The operator << is called the **insertion** or **put to** operator.
  - It inserts (or sends) the contents of the variable on its right to the object on its left.

A Simple C++ Program



Output using insertion operator

A Simple C++ Program

- Output Operator
  - If string represents a string variable, then the following statement will display its contents :

```
cout << string;
```

  - You may recall that the operator << is the bit-wise left-shift operator and it can still be used for this purpose.
  - This is an example of how one operator can be used for different purposes, depending on the context.
  - This concept is known as operator overloading, an important aspect of polymorphism.

A Simple C++ Program

- Output Operator
  - It is important to note that we can still use printf() for displaying an output.
  - C++ accepts this notation.
  - However, we will use cout << to maintain the spirit of C++.

A Simple C++ Program

- The iostream File
  - The following #include directive

```
#include <iostream>
```

  - Causes the preprocessor to add the contents of the iostream file to the program.
  - It contains declarations for the identifier cout and the operator <<.
  - Some old versions of C++ use a header file called iostream.h.
  - The header file iostream should be included at the beginning of all programs that use input/output statements.

A Simple C++ Program

- Namespace
  - Namespace is a new concept introduced by the ANSI C++ standards committee.
  - This defines a scope for the identifiers that are used in a program.
  - For using the identifiers defined in the namespace scope, we must include the using directive, like

```
using namespace std;
```

  - std is the namespace where ANSI C++ standard class libraries are defined.
  - This will bring all the identifiers defined in std to the current global scope.

A Simple C++ Program

- Return Type of main()
  - In C++, main() returns an integer type value to the operating system.
  - Therefore, every main() in C++ should end with a return(0) statement; otherwise a warning or an error might occur.
  - Since main() returns an integer type value, return type for main() is explicitly specified as int.

A Simple C++ Program

- Return Type of main()
  - The following main without type and return will run with a warning :

```
main ( )
{
        ……
        ……
}
```

More C++ Statements

- A slightly more complex C++ program.(Example 2-2)
- Assume that we would like to read two numbers from the keyboard and display their average on the screen.

More C++ Statements

- Variables
  - All variables must be declared before they are used in the program.

More C++ Statements

- Input Operator
  - The statement is an input statement.

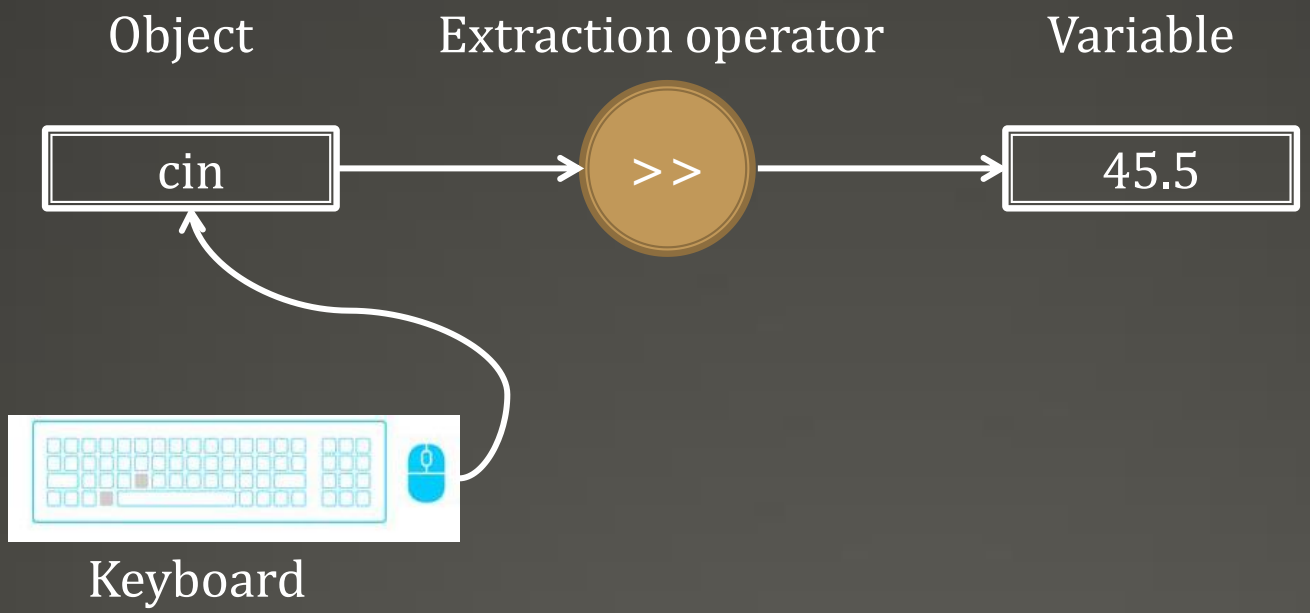
```
cin >> number1;
```

  - It causes the program to wait for the user to type in a number.
  - The number keyed in is placed in the variable number1.
  - The identifier cin (pronounced ‘C in’) is a predefined object in C++ that corresponds to the standard input stream.
  - Here, this stream represents the keyboard.

More C++ Statements

- Input Operator
  - The operator >> is known as extraction or get from operator.
  - It extracts (or takes) the value from the keyboard and assigns it to the variable on its right.
  - This corresponds to the familiar scanf() operation.
  - Like <<, the operator >> can also be overloaded.

More C++ Statements



Input using extraction operator

An Example with Class

- One of the major features of C++ is class.
- Class provides a method of binding together data and functions which operate on them.
- Like structures in C, classes are user-defined data types.

An Example with Class

- Example 2-3
  - The program defines person as a new data of type class.
  - The class person includes two basic data type items and two functions to operate on that data.
  - These functions are called member functions.
  - The main program uses person to declare variables of its type.
  - Class variables are known as objects.
  - Here, p is an object of type person.

Structure of C++ Program

- A typical C++ program would contain four sections.
- These sections may be placed in separate code files and then compiled independently or jointly.

Include files
Class declaration
Member function definitions
Main function program

Creating the Source File

- Like C programs, C++ programs can be created using any text editor.
- Some systems provide an integrated environment for developing and editing programs.

Compiling and Linking

- Visual C++
  - It is a Microsoft application development system for C++ that runs under Windows.

Programming Exercises

- Write a program to display the following output using a single cout statement.

```
Maths       = 90
Physics     = 77
Chemistry   = 69
```