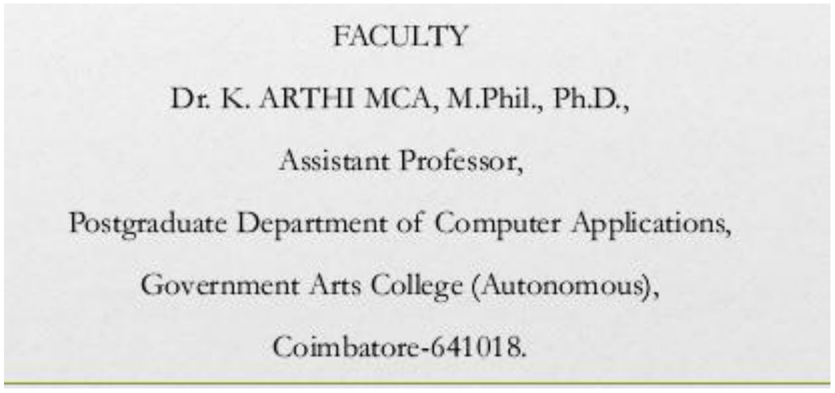
20MCA11C OBJECT ORIENTED PROGRAMMING WITH C++

UNIT I: Principles of Object Oriented Programming



FACULTY

Dr. K. ARTHI MCA, M.Phil., Ph.D.,

Assistant Professor,

Postgraduate Department of Computer Applications,

Government Arts College (Autonomous),

Coimbatore-641018.

UNIT I: Principles of Object Oriented Programming: Software Crisis - Software Evolution - Procedure Oriented Programming - Object Oriented Programming Paradigm - Basic concepts and benefits of OOP - Object Oriented Language - Application of OOP - Structure of C++ - Applications of C++ - Tokens, Expressions and Control Structures - Operators in C++ - Manipulators.

UNIT II: Functions in C++: Function Prototyping - Call by reference - Return by reference - Inline functions - Default, const arguments - Function Overloading - Friend and Virtual Functions. **Classes and Objects:** - Member functions - Nesting of member functions - Private member functions - Memory Allocation for Objects - Static Data Members - Static Member functions - Array of Objects - Objects as function arguments - Friendly functions - Returning objects - const member functions - Pointer to members.

UNIT III: Constructors: Parameterized Constructors - Multiple Constructors in a class - Constructors with default arguments - Dynamic initialization of objects - Copy and Dynamic Constructors - Destructors. **Operator Overloading**: Overloading unary and binary operators - Overloading binary operators using friend functions- Overloading the extraction and the insertion operators.

UNIT IV: Inheritance: Defining derived classes - Single Inheritance - Making a private member inheritable - Multiple inheritance - Hierarchical inheritance - Hybrid inheritance - Virtual base classes - Abstract classes - Constructors in derived classes - Member classes - Nesting of classes.

UNIT V: Streams: String I/O - Character I/O - Object I/O - I/O with multiple objects - File pointers - Disk I/O with member functions. Exception handling - Templates - Redirection - Command line arguments.

TEXT BOOKS:

1.E.Balagurusamy, "Object Oriented Programming With C++", 6th Edition, Galgotia, Publications Pvt. Ltd., 2000.

REFERENCE BOOKS:

1.Herbert Schildt, C++: The Complete Reference, McGraw Hill Inc., 1997.
2.Stanley B. Lippman, Inside the C++ Object Model, Addison Wesley, 1996

Principles of Object-Oriented Programming

software crisis

- How to represent real-life entities of problems in system design?
- How to design systems with open interfaces?
- How to ensure reusability & extensibility of modules?
- How to develop modules that are tolerant to any changes in future?
- How to improve s/w productivity & decrease s/w cost?
- How to improve the quality of s/w?
- How to manage time schedules?
- How to industrialize the s/w development process?

1. 2 Software Evolution

The s/w evolution can be describing it as a tree. The s/w evolution has had distinct phases or *layers* of growth. These layers were built up one by one. With each layer representing an improvement over the previous one. Fig. (1.1) had shown layers of s/w.

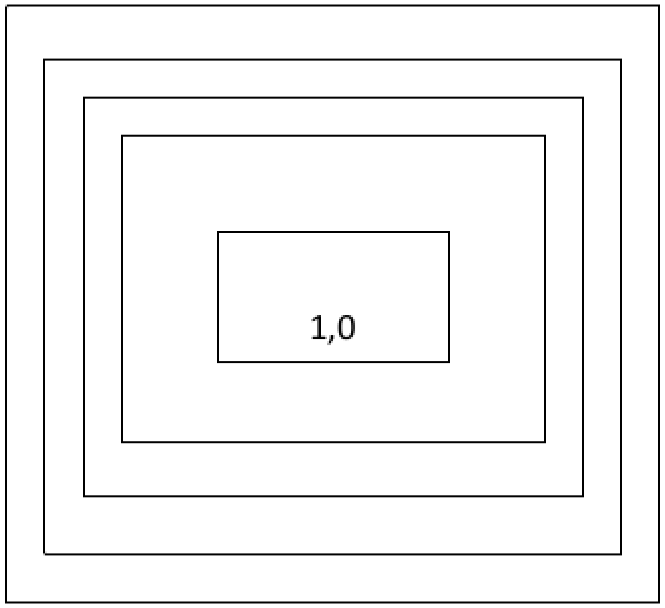


Fig. (1.1) layers of s/w technology

Since the invention of the computer, many programming approaches have been tried. These included techniques such as *modular programming*, *top-down programming*, *bottom-up programming* and *structured programming*. The primary motivation in each case has been the concern to handle the increasing complexity of programs that are reliable & maintainable. These techniques became popular among programmers over the last two decades. With the advent of languages such as C, structured programming become very popular and was the main technique of the 1980s. Structured programming was a powerful tool that enabled programmers to write moderately complex programs fairly easily.

Object-Oriented Programming (OOP): -

Is an approach to programs organization and development that attempts to eliminate some of the pitfalls of conventional programming methods by incorporating the best of structured programming features with several powerful new concepts.

1.3 Procedure–Oriented Programming

Conventional programming using high level languages (COBOL, FORTRAN and C) is commonly known as procedure oriented programming. In the procedure-oriented approach, 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. A typical program structure for procedure programming is show in fig. (1.2).

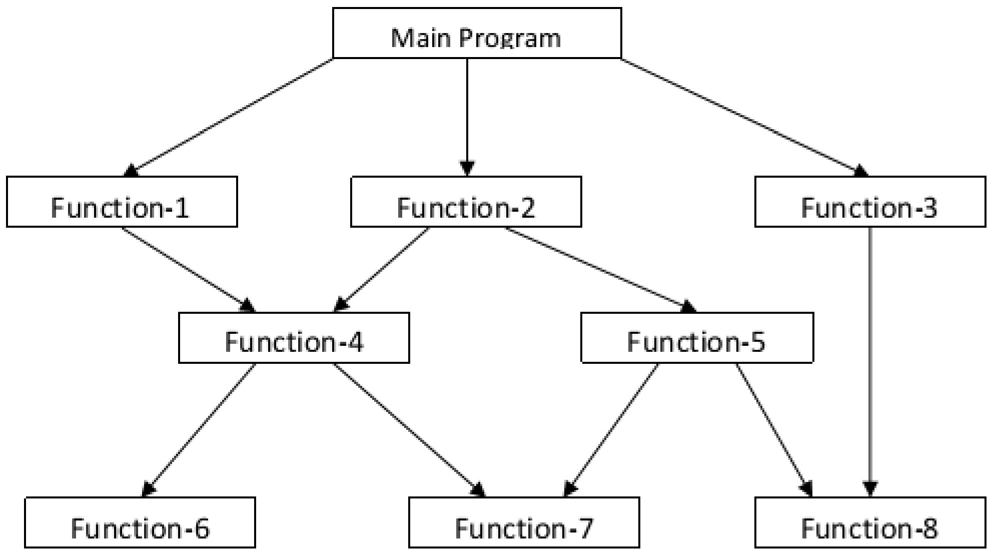
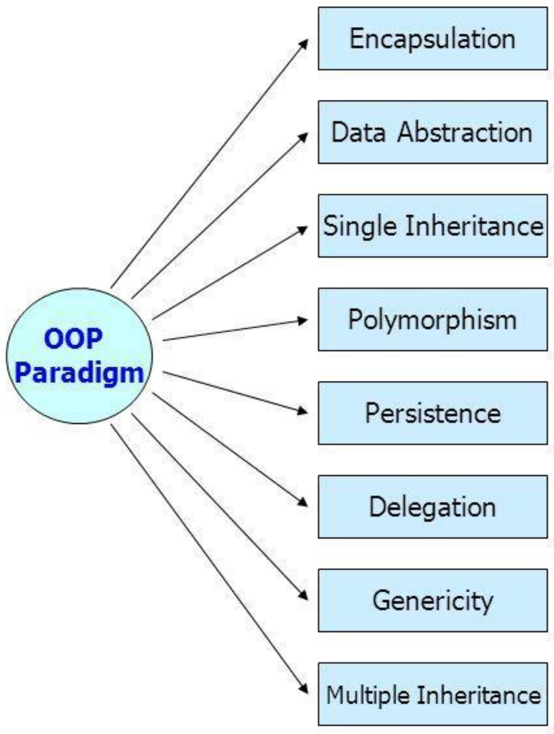


Fig. (1.2) Typical structure of procedure oriented programs

Characteristics exhibited by procedure-oriented programming

1- Emphasis is on doing things (algorithms).
2- Large programs are divided into smaller programs known as "functions".
3- Most of the functions share global data.
4- Data move openly around the system from function to function.
5- Function transforms data from one form to another.
6- Employs *top-down* approach in program design .

Object Oriented Paradigm: Features



Basic concepts and benefits of OOP

Abstraction: The process of picking out i.e. abstracting similar characteristics of procedures and objects.

Class: It means categorizing objects. However, a class defines all the common traits of the numerous objects that fall under it.

Encapsulation: It is defined as wrapping the data under a single, consolidated unit. In Object Oriented Programming, it is defined as binding data with a function that manipulates it.

Inheritance: Inheritance is defined as the ability of one class to derive its characteristics from another class.

Interface: Interface comprises the languages and the codes used by various applications to communicate with each other.

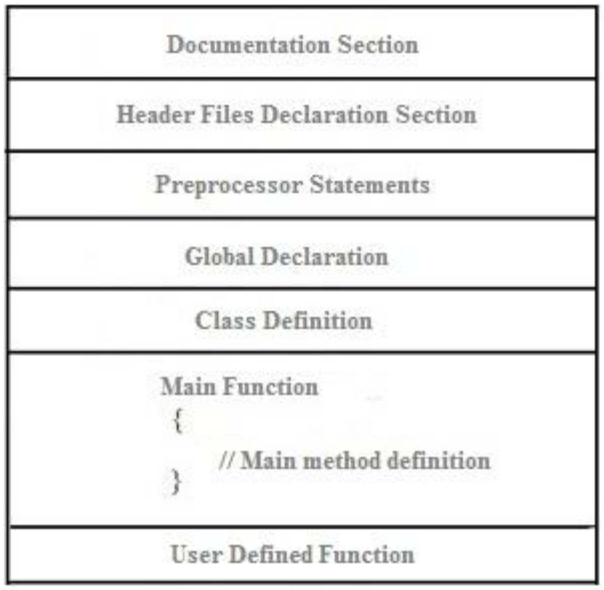
Object: Object is a self-contained entity. It consists of data as well as procedures.

Polymorphism: It refers to a programming language's ability to process objects uniquely according to their data type and class.

1. **Simplicity:** software objects model real world objects, so the complexity is reduced and the program structure is very clear;
2. **Modularity:** each object forms a separate entity whose internal workings are decoupled from other parts of the system;
3. **Modifiability:** it is easy to make minor changes in the data representation or the procedures in an OO program.

4. **Extensibility:** adding new features or responding to changing operating environments can be solved by introducing a few new objects and modifying some existing ones;
5. **Maintainability:** objects can be maintained separately, making locating and fixing problems easier;
6. Re-usability: objects can be reused in different programs

structure of a c++ program



Variables

A variable is the storage location in memory that is stored by its value. A variable is identified or denoted by a variable name. The variable name is a sequence of one or more letters, digits or underscore, for example: character _

Rules for defining variable name:

- A variable name can have one or more letters or digits or underscore for example character.
  .
- White space, punctuation symbols or other characters are not permitted to denote variable name. .
- A variable name must begin with a letter.
  .
- Variable names cannot be keywords or any reserved words of the C++ programming language.
  .

- C++ is a case-sensitive language. Variable names written in capital letters differ from variable names with the same name but written in small letters. For example, the variable name EXFORSYS differs from the variable name exforsys.

A variable is the storage location in memory that is stored by variable value. The amount of memory allocated or occupied by each variable differs as per the data stored. The amount of memory used to store a single character is different from that of storing a single integer. A variable must be declared for the specific data type.

Data Types

Below is a list of the most commonly used *Data Types* in C++ programming language:

- short int
- int
- long int
- float
- double
- long double
- char
- bool

short int : This data type is used to represent short integer.

int: This data type is used to represent integer.

long int: This data type is used to represent long integer.

float: This data type is used to represent floating point number.

double: This data type is used to represent double precision floating point number.

long double: This data type is used to represent double precision floating point number.

char: This data type is used to represent a single character.

bool: This data type is used to represent boolean value. It can take one of two values: True or False.

Using variable names and data type, we shall now learn how to declare variables.

Declaring Variables:

In order for a variable to be used in C++ programming language, the variable must first be declared. The syntax for declaring variable names is

```
data type variable name;
```

The date type can be int or float or any of the data types listed above. A variable name is given based on the rules for defining variable name (refer above rules).

Example:

```
int a;
```

This declares a variable name a of type int.

If there exists more than one variable of the same type, such variables can be represented by separating variable names using comma.

For instance

```
int x,y,z
```

This declares 3 variables x, y and z all of data type int.

The data type using integers (int, short int, long int) are further assigned a value of signed or unsigned. Signed integers signify positive and negative number value. Unsigned integers signify only positive numbers or zero.

For example it is declared as

```
unsigned short int a;
signed int z;
```

By default, unspecified integers signify a signed integer.

For example:

```
int a;
```

is declared a signed integer

It is possible to initialize values to variables:

```
data type variable name = value;
```

Example:

```
int a=0;
int b=5;
```

Constants

Constants have fixed value. Constants, like variables, contain data type. Integer constants are represented as decimal notation, octal notation, and hexadecimal notation. Decimal notation is represented with a number. Octal notation is represented with the number preceded by a zero character. A hexadecimal number is preceded with the characters 0x.

Example

80 represent decimal
0115 represent octal
0x167 represent hexadecimal

By default, the integer constant is represented with a number.

The unsigned integer constant is represented with an appended character u. The long integer constant is represented with character l.

Example:

78 represent int
85u present unsigned int
78l represent long

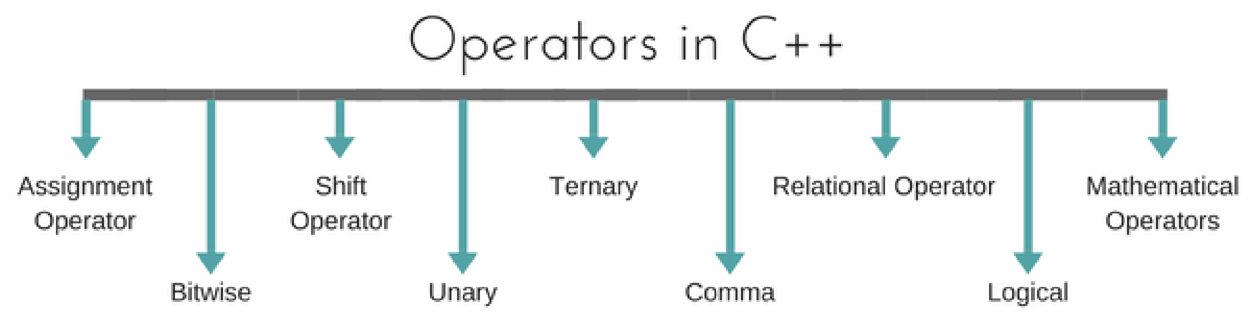
Floating point constants are numbers with decimal point and/or exponent.

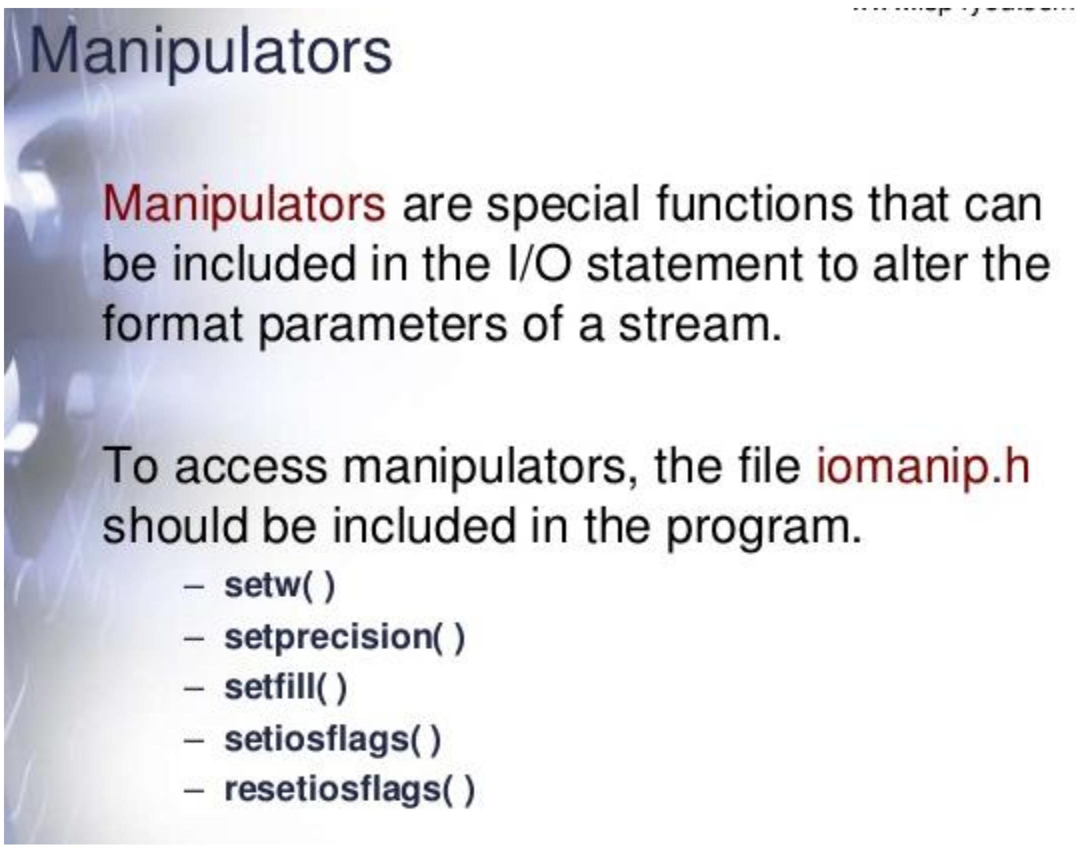
Example

2.1567
4.02e24

These examples are valid floating point constants.

Floating point constants can be represented with f for floating and l for double precision floating point numbers.





Manipulators

Manipulators are special functions that can be included in the I/O statement to alter the format parameters of a stream.

To access manipulators, the file iomanip.h should be included in the program.

- **setw( )**
- **setprecision( )**
- **setfill( )**
- **setiosflags( )**
- **resetiosflags( )**

THANK YOU

This content is taken from the text books and reference books prescribed in the syllabus.