CMP 104.3 Object Oriented Programming in C++ (3-1-3)

Objective:

To familiarize with Object Oriented Programming Concept using C++

Course Outline:

1. **Thinking Object Oriented**
2. **Classes and Methods**
3. **Message, Instance and Initialization**
4. **Object Inheritance and Reusability**
5. **Polymorphism**
6. **Template and generic programming**
7. **Object oriented Design**

Books:

- Budd, T., An Introduction to Object Oriented Programming, Second Edition, Addison-Wesley, Pearson Education Asia, ISBN: 81-7808-228-4.
- R. Lafore, Object Oriented Programming in Turbo C++, Galgotia Publications Ltd. India, 1999
- E Balaguruswamy, Object Oriented Programming with C++, Third Edition

1. Thinking Object Oriented

1.1 Object Oriented Programming a New Paradigm:

Procedural-oriented programming (POP) focuses on:

- List of organizing instructions called functions
- Little attention given to data
- Data may access by all functions
- Difficult to identify what data is used by which function
- Does not correspond to the real world problem
- Use top-down program design approach

these leads to the bugs, to remove the flaws encountered in POP an Object Oriented Programming (OOP) was introduced as a new model or example with organizational approach among programming paradigms and still means different things to different people, however deals with decomposition of a problem into a number of interacting agents called objects to solve the problem with following features:

…

- Programs (Problems) are divided into objects
- Emphasis on data rather than procedure
- Data structures are designed such that they characterized the objects
- Related data and functions are tied together, data can be accessed by those functions only
- Correspond to the real world problem
- Follow bottom up approach in programming design

…

So, **Object-oriented programming** (**OOP**) is a programming paradigm that uses "objects" and their interactions to design applications and computer programs. An **object** commonly means a data structure consisting of data fields and procedures (**methods**) that can manipulate those fields. Typically, when calling a method from some object, the object itself should be passed as a parameter to the method.

Object-Oriented programming is much more similar to the way the real world works; it is analogous to the human brain. Each program is made up of many entities called objects. Objects become the fundamental units and have behavior, or a specific purpose, associated with them.

…

Objects cannot directly access another object's data. Instead, a message must be sent requesting the data, just like people must ask one another for information; we cannot see inside each other's heads.

Benefits of Object-Oriented programming include:

- ability to simulate real-world event much more effectively
- code is reusable thus less code may have to be written
- data becomes active
- better able to create GUI (graphical user interface) applications
- programmers are able to reach their goals faster
- Programmers are able to produce faster, more accurate and better-written applications (in the case of a veteran programmer, by a factor of as much as 20 times compared with a procedural program).

…

1.2 A Way of Viewing World Agent

As in handling real world situations, OOP is also based on interacting agents and their communications where the terms agent, message, responsibility and method have major roles.

To illustrate the idea, let us suppose, I wish to send flowers to my grandmother for her birthday. She lives in a city many miles away, so my picking the flowers and carrying them to her door by myself is out of the question. Nevertheless, sending her the flowers in an easy way; for that I merely go down to my local florist (named Flo), tell her kinds and the numbers of flowers I want to sent and my grandmother's address, and I can be assured the flower will be delivered expediently and automatically.

…

1.2.1 Agents, Responsibility, Messages and Methods

The mechanism I used to solve my problem was to find an appropriate **agent** (namely, Flo) and to pass to her **message** containing my request. It is the **responsibility** of Flo to satisfy my request. There is some **method** used by Flo to do this. I do not need to know the particular method she will use to satisfy my request. This information is usually hidden from my inspection.

7

…

So, our first principle of object oriented problem solving is the vehicle by which activities are initiated: Actions is initiated in object oriented programming by the transmission of a message to an agent (an **object**) responsible for the action. The **message** encodes the request for an action and is accompanied by any additional information (arguments) needed to carry out the request. The receiver is the agent to whom the message is sent. If the receiver accepts the message, it accepts the responsibility to carry out the indicated action. In response to a message, the receiver will perform some method to satisfy the request.

1.2.2 Responsibility

Responsibilities permits greater independence between agents, a critical factor in solving complex program. The entire collection of responsibilities is often described by the term protocol.

8

…

1.2.3 Class and Instances

All objects are instances of class and classes are collection of objects of similar type. The method invoked by an object in response to a message is determined by the class of the receiver. All objects of a class use the same method in response to similar message.

1.2.4 Class Hierarchies- Inheritance

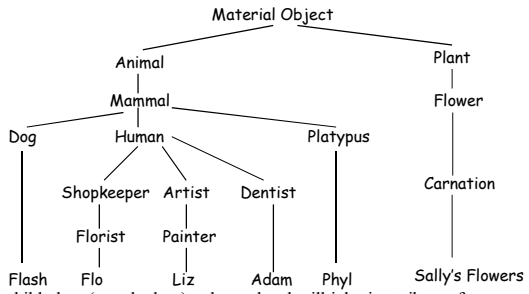
The category of florist is more specialized form of the category of shopkeeper. Any knowledge of shopkeeper is also true of florist and hence of flo. If we analyze how the flo has been organized in terms of a hierarchy of categories as in figure.

Flo is florist, but florist is a specialized form of shopkeeper. Furthermore, shopkeeper is also a human. A human is mammal and mammal is an animal and animal is material object (it has mass and weight).

The principle that knowledge of a more general category is also applicable to a more specific category is called inheritance. The class florist will inherit attributes of the class or category shopkeeper.

9

A class hierarchy of various material objects



A child class (or sub class) at lower level will inherit attributes from a parent class at higher level in the hierarchy. An abstract parent class is a class for which there are no direct instances; it is used only to create subclasses.

10

1.3 Computation as Simulation: …

The traditional model describing the behavior of a computer executing a program is a process-state or pigeon-hole model. In this view, the computer is a data manager, following same pattern of instruction, wandering through memory, pulling values out of various slots (memory address), transforming them in some manner, and pushing the result back into other slots.

In contrast, in the object-oriented framework we never mention memory address, variables, assignment, or any of the conventional programming terms. Instead, we speak of objects, messages, and responsibilities for some action. This view of programming is in many ways similar to a style of **computer simulation (process of representing real situation by computer)** called "discrete event-driven simulation."

11

…

In brief, in a discrete event-driven simulation, the user creates computer models of the various elements of the simulation, describes how they will interact with one another, and sets them moving. This is almost identical to the average object-oriented program, in which the user describes what the various entities in the universe for the program are, and how they will interact with one another, and finally sets them in motion. Thus OOP can be viewed as a Computation as Simulation.

12

...

1.4 Coping with Complexity:

People deal with complex artifacts and situations every day, they nevertheless experience complexity if not yet have created computer program (software). Because of the **non linear behavior of complexity**, the s/w development task is really complex. However, by using a technique called **abstraction** complexity can be managed.

1.4.1 The Nonlinear Behavior of Complexity:

Complexity means difficult to understand. S/w is inherently complex i.e. complexity of software (s/w) development task is essential property not an accidental one, because s/w complexity evolved from different factors such as:

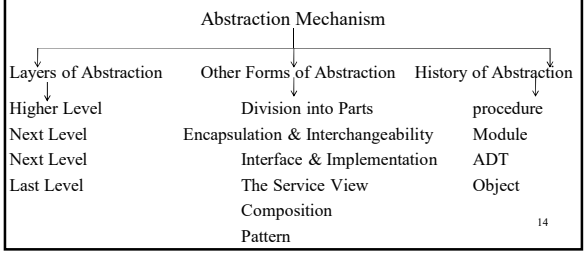
- problem domain
- development process management
- s/w flexibility
- change in digital system technology

13

...

therefore, is in different shape & order. As a result, difficult to measure & characterize the behavior of s/w complexity. Thus, in computer programming, we have the view that it exhibits nonlinear behavior of complexity and is not easy to dealing successfully (coping). However, these constraints can be addressed by abstraction mechanism.

1.4.2 Abstraction Mechanism



14

...

The term **abstraction** refers to the act of representing essential features without including unnecessary detail. Programmers have had to deal with the problem of complexity for a long time. To understand more fully the importance of object-oriented techniques, we should review the variety of mechanism programmers have used to control complexity. Among them abstraction is one of the most basic tools, the ability to encapsulate and isolate design and exaction information. To create, understand and manage complex system, abstraction mechanism can be used in varieties of **layers** and **forms**. However, one should know that the object oriented techniques are not at all revolutionary, but can be seen as natural outcome of a long historical progression from procedures, to modules, to abstract data types, and finally to objects which can be described as **history of abstraction mechanisms**.

15

...

Layers of Abstraction

In OOP, different levels of abstraction can be used to manage complexity of a problem where lower the level search for more detail information as:

Higher level: viewed as community of objects that interact with each other to achieve their common goal

Next level: a group of objects working together combined into an unit which perform similar type of action

Next level: deals with interaction between two objects

Last level: deals with activity of single object

16

...

Other Forms of Abstraction

Idea of abstraction can be further subdivided into different forms to implement in OOP as:

Division into Parts: A problem is divided into parts and each parts are analyzed individually as a single unit

Encapsulation & Interchangeability: Encapsulation is the process of wrapping up related data & method into a single unit which further permits possibility of interchangeability

Interface & Implementation: Interface describes what a system is designed to do where as implementation describes how the task is performed by the system

The Service View: an object is provide a service that is used by other object, the service view and the relationship between the objects give the clear picture about the system

Composition: technique used to create a system from various single parts

Pattern: approach to solve the problem with the help of previous solution

...

History of Abstraction

Procedures:

Procedures/functions were first abstraction mechanism to be widely used in programming languages. Procedures allowed the tasks that were executed repeatedly, or executed with only slight variation, to be collected in one place and reused rather than being duplicated several times. The procedure gave the first possibility of information hiding.

One programmer could write a procedure that was used by many others. The other programmers did not need to know the exact details of the implementation- they needed only the necessary interface. But procedures were not an answer to all problems. There were not effective mechanism for information hiding, and they only partially solved the problem of multiple programmers making use of the same names.

18

...

Modules:

Modules can be viewed simply as an improved technique for creating and managing collections of names and their associated values. A module provides the ability to divide a name space into two parts. The public part is accessible outside the module and the private part is accessible only within the module.

Suppose a programmer announces that he has developed a new type of numeric abstraction, called Complex. He has defined the arithmetic operations for complex numbers- addition, subtraction, multiplication, and so on, and had defined routines to convert number from conventional to complex. There is just one small problem: Only one complex number will be manipulated. Modules provide an effective method of information hiding, but they do not allow us to perform instantiation, which is the ability to make multiple copies of the data areas.

19

...

Abstract Data Types (ADT):

An abstract data types is a programmer defined data types that can be manipulated like the system defined data types. As with system defined types, as abstract data type corresponds to set of legal data values and a number of primitive operation that can be performed on those values. Users can create variables with values that range over the set of legal values and can operate on those values using the defined operations. To build the abstract data types, we must able to:

1. Export a type definition
2. Make available a set of operations that can be used to manipulate instances of the type
3. Protect the data associated with the type so that they can be operated on only by the provided routines. 4. Make multiple instances of the type.

20

...

Objects:

The central concept of object-oriented programming is the object, which is a kind of module containing data and subroutines. An object is a kind of self-sufficient entity that has an internal state (the data it contains) and that can respond to messages (calls to its subroutines). Programming problem is analyzed in terms of objects and the nature of communication between them. They may represent a person, a place, a bank account, a table of data or any item that the program has to handle.

Basic Concepts of OOP

General concepts/characteristics used in OOPs are:

1. Objects and classes
2. Data abstraction and encapsulation
3. Inheritance
4. Polymorphism
5. Dynamic binding
6. Message passing

21

Assignments

1) Why object oriented programming is dominating procedural programming approach? Explain with the features of OOP.
2) What do you mean by abstraction mechanism? Explain with its layer and forms.
3) Write short notes on:
   a)computation as simulation b)nonlinear behavior of complexity
   c)Objects and class

22