Lecture 7
Object Oriented Programing (OOP) & Procedural Programming

Course: Object Oriented Programming

Outline

- Procedural Programming
- Object Oriented Programing (OOP)
- Intro to Classes & Objects

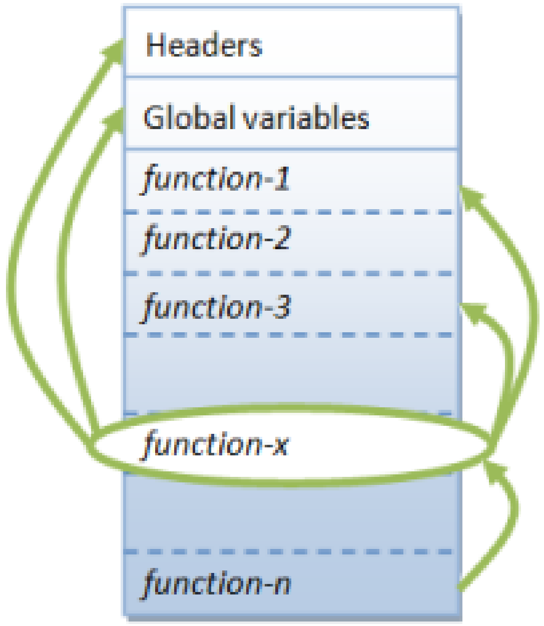
- Member Functions: Access Functions (Accessors and Mutators) Utility Functions

Procedural programming vs Object Oriented Programming

- Procedural programming divides the program into procedures, which are also known as routines or functions, simply containing a series of steps to be carried out.
- It involves writing down a list of instructions to tell the computer what it should

do step-by-step to finish the task at hand.

- Program flow control is achieved through function calls.



A function (in C) is not well-encapsulated

Procedural programming vs Object Oriented Programming

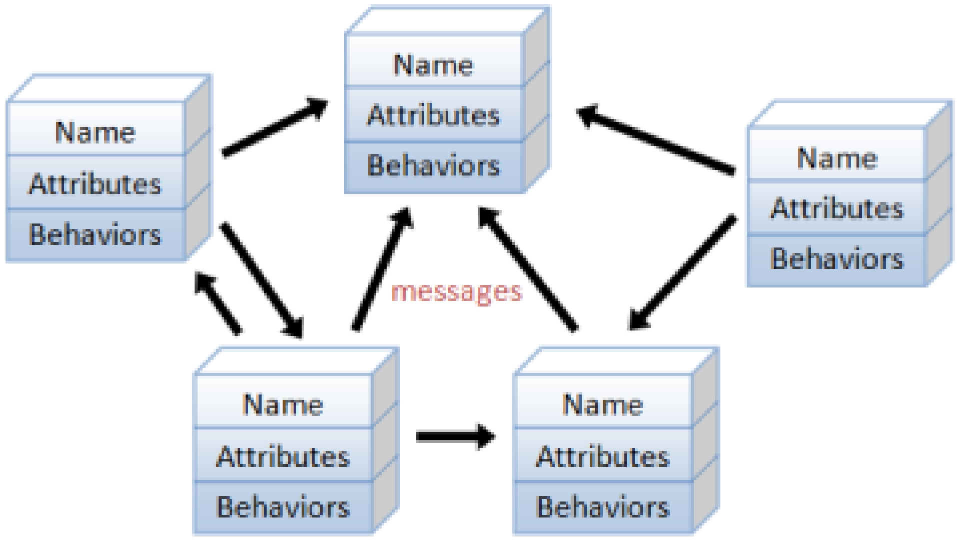
- It is not very easy to add new data structure in the procedural programming because all functions must change.

Limitation^{s}

- The procedural code is often not reusable, need to recreate.
- Difficult to relate with real-world objects.
- The importance is given to the operation rather than the data

Procedural programming vs Object Oriented Programming

- Object-oriented programming (OOP) is a computer programming model that ***organizes software design around objects***, rather than functions.
- Data is hidden and cannot be accessed by external functions. • The main aim of OOP is to *bind together the data and the functions that operate on them* so that no other part of the code can access this data except that function.



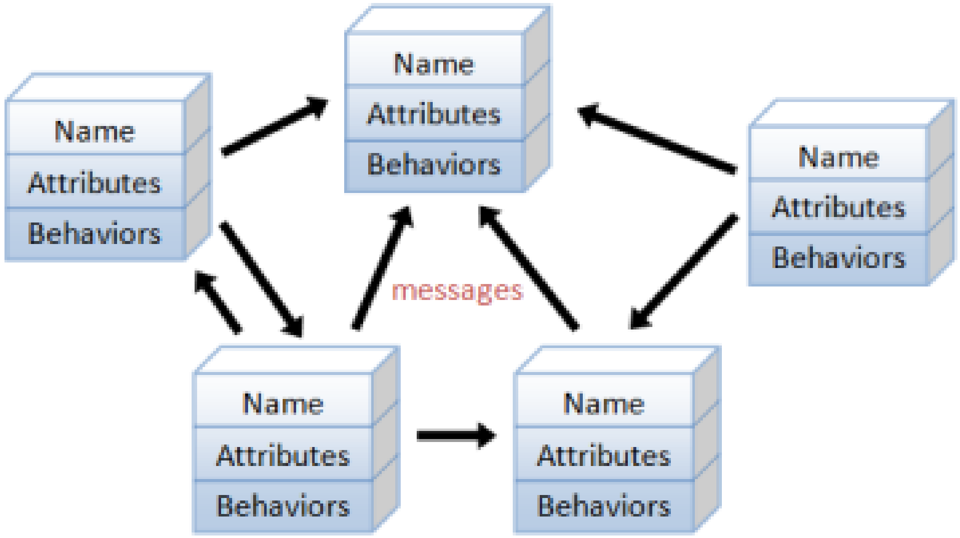
An object-oriented program consists of many well-encapsulated objects and interacting with each other by sending messages

5

Object oriented programming languages

- OOP is a programming paradigm that relies on the concept of classes and objects.
- It is used to structure a software program into simple, reusable pieces of code blueprints (usually called classes)

which are used to create individual instances of objects.



An object-oriented program consists of many well-encapsulated objects and interacting with each other by sending messages

Object oriented programming languages

- A programmer designs a software program by organizing related pieces of information and behaviors together into a template called a class

- The entire software program runs by having multiple objects interact with objects to create the larger program

Building blocks of OOP

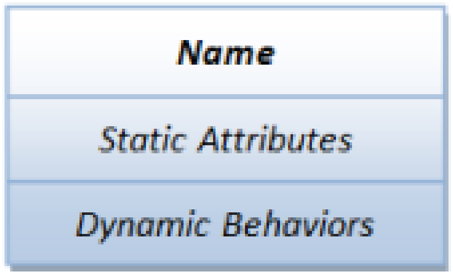
The code building blocks to build an OOP program are:

- Classes (Blue print/prototype/Template)
- Objects (Instance of class)
- Attributes (Data, characteristics, Information, Instance Variables)
- Methods (Behavior / actions, Instance Methods) •

What is Class?

- A class is similar to structures
  - Adds member FUNCTIONS (also known as methods)
  - Not just data member
- A class is integral to object-oriented programming
  - it focuses on objects
  - Object: Contains data and operations
  - In C++, variables of class type are objects

- A class is a 3-compartment box containing the name, variables and the methods.



A class is a 3-compartment box

```
class Car
{
int weight;
String color;
```

drive() {}
brake() {}
}

How to define a class? (Syntax and code example)

- A class is defined similar to structure:

- Example

```
class DayofYear
    { public:
       display();
       int month;
       int day;
};
```

Name of new class

type Member Function

• Notice only member function prototype •
Function’s implementation is elsewhere

What is an object?

• An object is an instance of class. Usually a person, place or thing (a noun).

• objects store data and provides method for accessing and modifying this data.

- Object is considered to be partitioned area of computer memory that stores data and a set of operations that can access the data.

Declaring an Object

- An object is declared same as all variables • Predefined types, structure types

- Example:

```
DayOfYear today, birthday;
```

  - Declares two objects of class type DayOfYear
- Objects include:
  - Data
    - Members month, day
  - Operations (member functions)
    - display()

Accessing class members

- Members of class can be accessed same as that of a Structure
- Suppose today is an object of type DateofYear then data members can be accessed as:

  today.month

  today.date
- And to invoke a member function

  today.display();

```
class DayofYear
   { public:
          display();
          int month;
          int day;
     };
```

class member functions

- Class member functions must be defined or implemented
- They can be defined after main() definition • But needs to

specify class name such as:

```
void DayofYear::display()
                    { }
```

  - :: is scope resolution operator
  - It instructs compiler 'what class' member is from

```
class DayofYear
    { public:
          display();
```

```
        int month;
        int day;
    };
```

Class Member Functions Definition

- Notice display() member function's definition (in next example)
- Refers to member data of class
  - No qualifier
- Function used for all objects of the class
- Will refer to 'that object's' data when

invoked • Example:

today.display();

- Displays today’s object data

birthday.display()

- Display’s birthday’s object data

Complete class example

```
class DayofYear {
      public:
            int month;
            int day;
   display(){
      cout<<" day: "<<day<<" month: "<<month;
```

```
                }
            }

    int main() {
    DayofYear today, birthdate;
    today.month=… ;
    today.day=…;
    birthdate.month = …;
    birthdate.day = …;
    cout<<"today's date is: "<<today.display();
    cout<<"your birthdate is:
    "<<birthdate.display(); }
```

Dot and scope resolution operator

- Used to specify ‘of what thing’ they are members

- Dot operator:
  - Specifies member of particular object

```
void DayofYear::display()
                    { }
```

- :: is scope resolution operator
  - Specifies what class the function definition comes from
  - It instructs compiler ‘what class’ member is from

A class's place

- Class is full-fledged type!
  - Just like data types int, double, etc.
- Can have variables of a class type • We

simply call them 'objects' • Can have

parameters of a class type •

Pass-by-value

  - Pass-by-reference
- Can use class type like any other type!

Encapsulation

- Any data type includes
    - Data (range of data)
    - Operations (that can be performed on data)
- Example:
    - int data type has:
        - Data: +-32,767
        - Operations: +,-,*,/,%,logical,etc.
- Same with classes
    - But WE specify data, and the operations to be

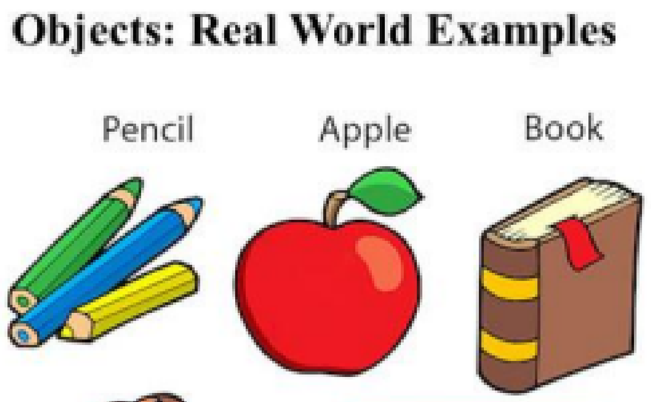
allowed on our data!

More Encapsulation

- Encapsulation
    - Means ‘bringing together as one’
- Declare a class → get an object
- Object is ‘encapsulation’ of
    - Data values
    - Operations on the data (member functions)

Real-world examples of object

Dogs have state (name, color, breed, hungry) and behavior (barking, fetching, wagging tail).



Chair, Bike, Marker, Pen, Table, Car, Book, Apple, Bag etc.

For Example, Pen is an object. Its name is Dollars; color is white, known as its state. It is used to write, so writing is its behavior.

What is an attribute and Function?

Attribute/characteristics of certain object (instance variable / information / property / characteristic / field and state)

Are function/method that manipulate the data, an action performed by an object (a verb)

```
public class MyClass
{
 int x = 5;
 show(){}
}
```