Introduction to Object-Oriented Programming

Structural programing and object-oriented programming

- Structural (procedural) programming
  - Programming using well defined control structures
    - Conditionals, loops, sequence, expression and assignments
    - Data (variables, arrays, structures) are separated from their operations
    - It provides an abstraction of the hardware.
    - You know this from COP3014
- Object-oriented programming
  - Built on top of structural (procedural) programming
  - Programming based on the concept of object.
    - Objects bundle data with their operations.
    - Enables information hiding, which allow us to organize the program in a more manageable way.

Object-Oriented basics

- A fundamental concept in an object-oriented language is the encapsulation of data and procedures (functions) together into units called **objects**.
  - An object consists of:
    - **Name** – a way of referring to an object inside a program (eg. A Fraction object might be called F1).
    - **Member Data** – data contained within an object (eg. Fraction has an integer numerator and denominator).
    - **Member Functions** – routines that act upon the data within an object (eg. The fraction object might have a function that returns the decimal representation of the fraction stored).
    - **Interface** – defines the ways a programmer may directly access the member data/functions of an object (more on this next lecture).

Classes

- A **class** is another fundamental concept in an object-oriented language that provides a blueprint for a new type ('classification') of object.
  - A class outlines the data, functions and the interface objects of that class will receive.
  - A class also defines how objects of that class behave by providing code that implements the functions associated with the class.
  - A programmer can create one or more objects from a class
    - Similar to building multiple houses from one set of blueprints.

How to define and use a class in a program

- DDU – Declare, Define, Use
  - Declare a class
    - Choose what objects of this class will store (member variables), and how objects will behave (member functions).
  - Define member functions
    - Provide an implementation for the member functions in the class.
  - Use class to create objects
    - You can declare an new object instance of your class just like declaring any other variable (eg. int x).

Example Class Declaration

```
class Circle
{
public: /* interface, we will cover later */
    void SetRadius(double r); /* sets member variable radius to r */
    double AreaOf(); /* returns area of circle as a double */
    double radius; /* radius of circle stored as double */
}; /* don't forget ';' */
```

Define Member Functions

- There are two ways to provide the member function definitions for a class:
  - Inside the class declaration using {} (we will not use)
  - After the class declaration (this is the method we choose)
- Refer to a member function: **className::memberFuntionName**
  - This identifier refers to the member function **memberFunctionName** of class **className** (e.g. Circle::SetRadius)
  - The double colon :: is called the scope resolution operator
- After the class declaration, member functions are defined just like any other function

Example member function definition

```
//Declaration:
class Circle
{
public:
    void SetRadius(double r); /*sets member variable radius to r */
    double AreaOf(); /* returns area of circle as a double */
private:
    double radius; /* radius of circle */
};

/* Definition (Implementation) */
void Circle::SetRadius(double r)
{
    radius = r; /* radius refers to this object’s member variable */
}

double Circle::AreaOf()
{
    return (3.14*radius*radius);
}
```

Object Use

- After a class has been declared and defined, an object of that class can be declared (also known as creation or instantiation) and used, a class is just like another type (int, char, etc).
- A programmer can declare an object with the following format:

ClassName ObjectName;

- This statement creates an object based on the blueprint of class ‘**ClassName**’ and the object can be referred to by the identifier (variable name) ‘**ObjectName**’
- The ‘ . ’ (dot) operator can be used to access an object’s public members
- The format for referring to an object’s member is:

ObjectName.MemberFunction() OR

ObjectName.MemberVariable

Putting it All Together

- See sample1.cpp
- To recap, this program:
  - declares the class Circle and outlines its members and interface
  - defines the implementation for the member functions of the Circle class
  - declares two objects of the class Circle, referred to as C1 and C2
  - uses the interfaces of C1 and C2 to store the radius of two circles and later to calculate the area of those circles

Summary

- An object is a unit that encapsulates data and functions. It has four elements: a name, data members, function members, and an interface.
- A class specifies the (user-defined) form of objects.
- The use of an object in a C++ program follows the declare, define, and use sequence.
- What does scope resolution operator (::) do?
- What does the dot operator (.) do?