

CSE 459.22 Programming in C++

- Name: Ren-Shiou Liu
- Time: R 8:30am-9:18am
- Room: BO 0124
- E-mail: rslu@cse.ohio-state.edu
- Office: DL 285
- Office hours: T 2:30pm-4:30 (tentative)
- Webpage: <http://www.cse.ohio-state.edu/rslu/459.22>



Overview

- Sign the roster list
- Syllabus and Course Policies
- Introduction to C++
- About Lab 1
- Fill Questionnaire



What is programming?

Programming is taking

A *problem*

Find the area of a rectangle

A set of *data*

length

width

A set of *functions*

area = length * width

Then,

Applying functions to data to solve the problem

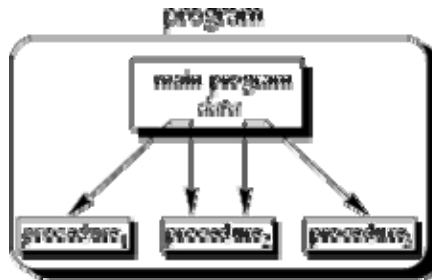


Programming Concept Evolution

- Unstructured
- Procedural
- Object-Oriented



Procedural Programming



- The main program coordinates calls to procedures and hands over appropriate data as parameters.

Copyright 2005, The Ohio State University

5



Procedural Concept (II)

- Procedural Languages
 - C, Pascal, Basic, Fortran
 - Facilities to
 - Pass arguments to functions
 - Return values from functions
- For the rectangle problem, we develop a function

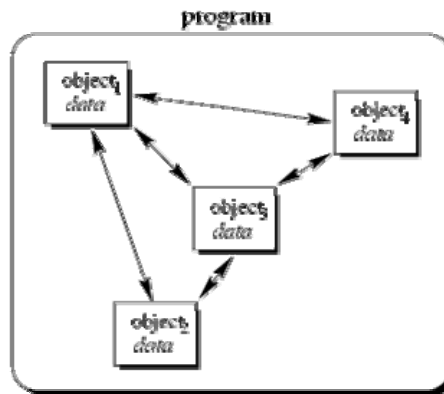
```
int compute_area (int l, int w){  
    return ( l * w );  
}
```

Copyright 2005, The Ohio State University

6



Object-Oriented Concept



- Objects of the program interact by sending messages to each other

Copyright 2005, The Ohio State University

7



Objects

An object is an encapsulation of both functions and data

- **Objects are an Abstraction**
 - represent real world entities
 - Classes are data types that define shared common properties or attributes
 - Objects are instances of a class
- **Objects have State**
 - have a value at a particular time
- **Objects have Operations**
 - associated set of operations called methods that describe how to carry out operations
- **Objects have Messages**
 - request an object to carry out one of its operations by sending it a message
 - messages are the means by which we exchange data between objects

Copyright 2005, The Ohio State University

8



OO Perspective

Let's look at the Rectangle through object oriented eyes:

- Define a new type Rectangle (a class)
 - Data
 - width, length
 - Function
 - area()
- Create an instance of the class (an object)
- Request the object to return its area

In C++, rather than writing a procedure, we define a class that encapsulates the knowledge necessary to answer the question - here, what is the area of the rectangle.



Example Code

```
class Rectangle
{
private:
    int width, length;
public:
    Rectangle(int w, int l)
    {
        width = w;
        length = l;
    }
}
```

```
int area()
{
    return width*length;
}
```

```
main()
{
    Rectangle rect(3,5);
    cout<<rect.area()<<endl;
}
```



Encapsulation

```
class Circle
{
    private:
        int radius
    public:
        Circle(int radius);

        // The area of a circle
        int area();
};
```

```
class Triangle
{
    private:
        int a, b, c;
    public:
        Triangle (int a, int b, int c);

        // The area of a triangle
        int area();
};
```



Abstract Types

```
class Shape
{
    public:
        Shape();

        // Calculate the area for
        // this shape
        virtual int area() = 0;
};
```

- Decouple the interface from the representation and give up genuine local variables
- **virtual** means “may be redefined later in a class derived from this one”
- “= 0” means pure virtual



Inheritance and Polymorphism

```
class Circle : public Shape
{
private:
    int radius
public:
    Circle (int radius);
    int area();
};
```

```
class Triangle : public Shape
{
private:
    int a, b, c;
public:
    Triangle (int a, int b,
int c);
    int area();
};
```

```
int sum_area(Shape s1, Shape s2) {
    return s1.area() + s2.area(); // Example of polymorphism
}
```

Copyright 2005, The Ohio State University

13



Characteristics of OOPL

- **Encapsulation:** Combining data structure with actions
 - Data structure: represents the properties, the state, or characteristics of objects
 - Actions: permissible behaviors that are controlled through the member functions

Data hiding: Process of making certain data inaccessible

- **Inheritance:** Ability to derive new objects from old ones
 - permits objects of a more specific class to inherit the properties (data) and behaviors (functions) of a more general/base class
 - ability to define a hierarchical relationship between objects
- **Polymorphism:** Ability for different objects to interpret functions differently

Copyright 2005, The Ohio State University

14



Basic C++

- Inherit all ANSI C directives
- Inherit all C functions
- You don't have to write OOP programming in C++



Basic C++ Extension from C

- comments

```
/* You can still use the old comment style, */  
/* but you must be // very careful about mixing them */  
// It's best to use this style for 1 line or partial lines  
/* And use this style when your comment  
consists of multiple lines */
```

- cin and cout (and #include <iostream.h>)

```
cout << "hey";  
char name[10];  
cin >> name;  
cout << "Hey " << name << ", nice name." << endl;  
cout << endl; // print a blank line
```

- declaring variables almost anywhere

```
// declare a variable when you need it  
for (int k = 1; k < 5; k++){  
    cout << k;  
}
```



Basic C++ Extension from C (II)

- `const`
 - In C, `#define` statement
 - Preprocessor - No type checking.
 - `#define n 5`
 - In C++, the `const` specifier
 - Compiler - Type checking is applied
 - `const int n = 5; // declare and initialize`
- New data type
 - Reference data type “&”.

```
int ix; /* ix is "real" variable */
int & rx = ix; /* rx is "alias" for ix. Must initialize*/
ix = 1; /* also rx == 1 */
rx = 2; /* also ix == 2 */
```

Copyright 2005, The Ohio State University

17



C++ - Advance Extension

- C++ allows function overloading
 - In C++, functions can use the same names, within the same scope, if each can be distinguished by its name *and* signature
 - The signature specifies the number, type, and order of the parameters expressed as a comma separated list of argument types

Copyright 2005, The Ohio State University

18



C++

- Is a better C
- Expressive
- Supports Data Abstraction
- Supports OOP
- Supports Generic Programming
 - Containers
 - Stack of char, int, double etc
 - Generic Algorithms
 - sort(), copy(), search() any container Stack/Vector/List



Take Home Message

- There are many different kinds of programming paradigms, OOP is one among them.
- In OOP, programmers see the execution of the program as a collection of dialoging objects.
- The main characteristics of OOPL include encapsulation, inheritance, and polymorphism.
- Don't panic if you don't understand the materials for the first class. Things will become clear in time.

