Object Oriented Programming in C++

Based on slides from C++ How to Program 4th edition.

Introduction

Object-oriented programming (OOP)

- Encapsulates data (attributes) and functions (behavior) into packages called classes
- Data and functions closely related

Information hiding

- Implementation details are hidden within the classes themselves

Unit of C++ programming: the class

- A class is like a blueprint – reusable
- Objects are instantiated (created) from the class
- For example, a house is an instance of a “blueprint class”

Implementing a Person data type using a Class

Classes

- Model objects that have attributes (data members) and behaviors (member functions)
- Defined using keyword `class`

```
1  class Person {
2  public:
3       Person ();
4       void setName( string name );
5       void setAge ( int age );
6       void printHello();
7  private:
8       string name;
9       int age;
10
11 };
```

`Public:` and `Private:` are member-access specifiers.

`setName`, `setAge`, and `printHello` are member functions. `Person` is the constructor.

name and age are data members.

Implementing a Person data type using a Class

Format

- ✓ Body delineated with braces (`{ and }`)
- ✓ `Class definition terminates with a semicolon`

Member functions and data

- ➢ **Public**

  accessible wherever the program has access to an object of class Time

- ➢ **Private**

  accessible only to member functions of the class

- ➢ **Protected**

  discussed later in the course

Implementing a Person data type using a Class

Constructor

- Special member function that initializes data members of a class object
- Constructors cannot return values
- Same name as the class

Definitions

- Once class defined, can be used as a data type

```
Person p1                          // object of type Person
       ArrayOfPersons[ 5 ]         // array of Person objects
       *pointerToPerson            // pointer to a Person object
```

Note: The class name becomes the new type specifier.

Implementing a Person data type using a Class

Outside a scope

- Use handles
  - An object name, a reference to an object or a pointer to an object

Accessing class members

- Dot `(.)` for objects and arrow `(->)` for pointers
- Example: `person.name` is the person element of t

16.14 Software Reusability

Object-oriented programmers

- Concentrate on implementing useful classes
- Tremendous opportunity to capture and catalog classes
- Accessed by large segments of the programming community
- Class libraries exist for this purpose

Software

- Constructed from existing, well-defined, carefully tested, portable, widely available components
- Speeds development of powerful, high-quality software