Chapter 9
Introduction To Classes

Mr. Dave Clausen
La Cañada High School

Transition from Structs to Classes

- A struct is similar to a class, in that it is a data structure with members.
  - By default, all the members in a struct are Public.
  - The members in a struct are generally data members, but could be member functions.
- In a class, the members are separated into public members and private members.
  - The members of a class are private by default.
  - The data members are generally private, while functions that use the data are generally public.
  - These member functions are the only functions with access to the private data.

Data Security

- One important problem with the employee_type data structure is security.
  - It is possible for too many functions to manipulate the data.
    - For example, the only functions that should be able to change an employees age should be:
      - init_employee(employee); //Initialize all attributes
      - read_employee(employee); //Input all attributes
      - increment_age(employee); //Update an employees age
    - There is nothing to prevent a user of **employee** to add a statement like this one anywhere in the program.
      - employee.age=employee.age - 5;

Encapsulation

- Encapsulation is the process of hiding and restricting access to the implementation details of a data structure.
  - Data should only be accessible to functions provided with the data structure.
  - These restrictions should be imposed by the compiler.
- An Instance (object, variable) is a computational object that has the attributes and behavior specified by the class.

Attributes and Behaviors

- The first step in developing a new class is to create a list of user requirements. These state the attributes and behaviors that users expect the class to have.
  - Attributes are the properties of an object, like a bank account balance or a bank password.
  - Behaviors are the operations or set of actions that a class of objects support, such as creating a user bank account with name and password, changing a user's password, depositing and / or withdrawing money, and checking the user's balance.

Formal Specifications

- The next step in specifying a class is to create the formal specifications of the class.
  - Formal specifications are the set of preconditions and post conditions for each of the member functions.
  - Classes are usually divided into their declaration sections and their implementation sections.
    - The declaration section is an area used to declare the data members and member functions of a class. This is usually implemented in a header file (.h).
    - The implementation section is an area where the member functions are implemented. This is usually done in a program file (.cpp)

Class Members

- The data and functions that belong to the class are called it's members: data members and member functions.
- Public members can be referenced by any program file that "includes" the class library header file.
- Private members can only be "invoked" indirectly through use of the member functions of the class.

Class Declaration

- The general form of a class declaration is:

```
class <class name>
{
  public:
    <public data declarations>
    <public function declarations>
  private:
    <private data declarations>
    <private function declarations>
}; //Like a struct, a semicolon is necessary to terminate a
   //class declaration, since it also is a C++ statement.
//A struct's members are all public by default.
```

Example Class Declarations

A class to describe birthdays

Bachbday.cpp Bachbday.txt

A class to describe dates

classdat.cpp classdat.txt

A class to describe fractions

Fraction1.cpp Fraction1.txt

A class to describe a general bank account

account.h account.txt

Member Function Categories

- The member functions can be categorized by the types of operations they perform on the data members.
  - Constructors
    - used to create and initialize an instance (object, variable) of the class.
  - Accessors (Observers)
    - used to examine an attribute of an object (variable) without changing it.
  - Modifiers (Transformers)
    - used to change the value of an attribute of an object.

Class Implementations

- Class Implementations will have the form:

```
//Class implementation file: account.cpp
<member function implementation1>
. . .
 <member function implementation>
```

- Class Implementation Headings:

<return type> <class name>:: <function name> (parameters)

  - Scope Resolution Operator
    - The double colons :: are called the scope resolution operator. They specify the class name that the member function belongs to when it is being defined.

Constructors

- When an instance (object, variable) of a class is declared its data is in an unpredictable state.
  - The same is true when a variable is declared.
    - int someInt; could have any undefined value between INT_MIN and INT_MAX.
  - Constructors guarantee that new instances of a class are always initialized.
  - There are three types of constructors
    - Default constructors
    - Parameterized constructors (with parameters)
    - Copy constructors

Constructor Guidelines

- A constructor member function must have the same name as the name of the class itself.
  - A constructor is implicitly invoked whenever a class object is created. (e.g. SomeClass anObject;)
  - Therefore, a constructor is never invoked using dot notation.
  - A constructor cannot return a function value, so no return value is specified when the function is declared. It's purpose is only to initialize a class object's private data.

Default Constructors

- Default constructor
  - A member function that creates and provides reasonable initial values for the data within an object.
  - Usually these values are 0 for integers, 0.0 for type double, and the null string “” or a blank space “ “ for strings and characters.
  - This constructor is implicitly invoked whenever a class object is created.

    bankAccountClass yourAccount, myAccount;

Constructors with Parameters

- This constructor is implicitly invoked when the class object to be declared includes a parameter list.

  e.g. TimeType favoriteTime (3, 0, 0)

  – Parameters are passed to a constructor by placing the actual parameter list immediately after the object name when the name of the class object is being declared.

  – The implementation of this constructor assigns the values of the parameters to the corresponding data members of the account object.

Copy Constructors

- Copy Constructor
  - A member function defined by the programmer and automatically used by the computer to copy the values of objects when they are "passed by value" to functions.
    - Remember that when C++ passes data by value in a parameter,the data value is copied to a temporary memory location for use in the function.
  - If a class doesn't specify its own copy constructor, the computer performs what is called a *shallow copy*. In this case, not all of the data is necessarily copied into temporary memory locations.

Function Overloading

- Function Overloading
  - When two or more functions use the same function name.
    - Using multiple Constructors is an example of function overloading.
    - The functions being overloaded must differ in the number of formal parameters, or use formal parameters of different types, or both.
    - When the function is called, the compiler uses the function definition whose number or type of formal parameters match the actual parameters being sent.

Destructors

- Related to constructors is another special type of member function called a class destructor.
  - A class destructor is implicitly invoked when a class object is destroyed.
    - For example, when program control leaves the block in which the local class object was declared.
  - A destructor uses the same name as the class itself, except that it is preceded by the tilde symbol.
    - For example: ~SomeClass( );
    - It is usually used to deallocate dynamic memory, however, it can be used for anything else for demonstration purposes.

Accessors

- Accessor functions
  - Allow us to observe the attributes of an object without changing them. (“read only” access)
    - In other words, accessors give us access to the values of private member variables without modifying the values.
  - To guarantee that no changes are made, we can declare accessor functions as a const function.
    - A const function doesn’t allow changes to the values or objects.
    - The form for declaring const functions:

<return type> <function name> (<formal parameter list>) **const**;

Modifiers

- Modifier functions set one or more of the attributes of an object to new values.
- In other words, a modifier is a member function that can alter or change private member data values.
- Sometimes called a transformer function.

Polymorphism

- Polymorphism
  - The property of one operator symbol or function identifier having many meanings.
    - The use of the same function name to mean different things.
    - The use of the same symbol to designate the same general operations even though the actual operations performed may vary with the type or structure of the operands.
      - e.g. The symbol / was used to represent division. When used with real numbers the result included the quotient and the decimal portion. With integers only the quotient was calculated.

Object Oriented Programming: OOP

- Benefits:
  - Security of data members
  - Software Maintenance
  - Avoiding Side Effects
  - Reuse of Software
    - Building software out of existing software components
    - e.g. A checking account is similar to Savings account
  - Client / Server
    - Client: object that receives a service of another object
    - Server: object that provides a service to another object

Sample Classes

Identify the categories of the member functions in these sample classes:

Bachbday.cpp	Bachbday.txt
classdat.cpp	classdat.txt
Fraction1.cpp	Fraction1.txt
Fraction2.cpp	Fraction2.txt
account.h	accounth.txt
account.cpp	account.txt

More Class Examples

- Bank account example:

bankdriv.ide bankdriv.cpp bankdriv.txt

rational.h rationalh.txt

rational.cpp rational.txt

ratdriv.ide

Predict The Output

- TestClass Examples:

test1.cpp

test2.cpp

ClassTwo.cpp

DoNothing.cpp