Overview of Today’s Class

- Review class declarations
- Class member function implementations, emphasizing the Date class
- Small example calendar program
- Summarize material from the text

See Tuesday’s notes for reading material about today’s class.

Review from Tuesday

- Information hiding and abstract data types
- Public vs. private in C++ class declaration
- Access is through the member functions.
- Example of Date class

Class Concepts — The Basics

We will illustrate the concepts through the Date example.

Class scope notation

- Date:: indicates that what follows is within the scope of the class.
- You will have to get accustomed to the notation of class member functions
- Within class scope, the member functions and member variables are accessible without the name of the object.
Constructors

These are special functions that initialize the values of the member variables.

• The syntax of the call to the constructor can be confusing. It mixes variable definitions and function calls.

• “Default constructors” have no arguments.

• Multiple constructors are allowed, just like multiple functions with the same name are allowed.

• The compiler determines which one to call based on the argument list. This must match (in type) the parameter list, just like any other function call.

Member Functions

These are like ordinary functions except

• they can access and modify the object’s member variables,

• they can call the member functions without using an object name, and

• their syntax is slightly different because they are defined within class scope.

isEqual member function

• Note that a second Date object is passed.

• Observe how the information about the second date is accessed through the member functions.

• These could also be accessed directly through the member variables, even though these member variables are private! We will see how in class.
Accessing and changing member variables

- In general, when the member variables are private, the only means of accessing them and changing them from outside the class is through member functions.

- If member variables are made public, they can be accessed directly. We will see how in class.

Header files (.h) and Implementation files (.cpp)

The code is in three files:

- Date.h contains the class declaration.
- Date.cpp contains the member function definitions. Date.h is included.
- main.cpp contains the code outside the class. Date.h again is included.
- The files Date.cpp and main.cpp are compiled separately and linked to form the executable program.
- Different organizations of the code are possible.
  - For example, we could have combined Date.h and Date.cpp into a single file and included them in the main program.
  - In this case, we would not have to compile two separate files.

Constant member functions

Member functions that do not change the member variables should be declared const

- For example,

  bool Date::isEqual(const Date &date2) const

- This must appear consistently in both the member function declaration in the class declaration and the member function definition.
• const objects, usually passed as parameters, can ONLY use const member functions
  – Remember, you should generally not pass objects by value. Pass them by const reference if you don’t want them to change.

• While you are learning, you will tend to make many syntax errors in determining which member functions should or should not be const.

We will examine which \texttt{Date} member functions to make const during lecture.

\textbf{Calendar Example}

We will look at an extensive example in class. Important notes include

• Use of the \texttt{string} class from the C++ standard library.
• Objects being passed by reference
• Arrays as member variables; arrays of objects
• The simple “driver” main program
• Constructors and initialization within constructors
• The \texttt{TODolist} member function

\textbf{Material from the Carrano and Prichard text}

• ADTs and the example of an abstract list. Think about the operations you might want.
• Use of UML in specifying classes (the notation is explained on pp. 22-23)
• Implementation and C++ classes, including a review of many of the concepts discussed in lecture.
• We will come back to list ADTs after we have a better understanding of C++ and classes.
Summary of Today’s Lecture

- Class scope
- The role and definition of constructors
- Member functions
- Access to member variables
- .h and .cpp files
- const
- arrays of objects