UNBC Laboratory Assignment

Computer Science 101—Winter 2007

Classes, and Member Functions

Due Date:

This assignment is due 16 February 2004 *at the beginning of lecture.*

Information on Gregorian dates:

We tend to take dates for granted, and assume, for instance, that everyone in the world uses the same system of dates. This, of course, isn't true. Both the epoch (that is day one of year one), and whether the calendar is solar- or lunar- based vary throughout the world. Even in Western Europe common dates are only a phenomonen of the last 250 years or so. Before that, the slowness of the Protestant countries to adopt the calendar reforms of Pope Gregory meant that the date depended on which country you were in.

For the purposes of this assignment, we shall assume the use of the Gregorian calendar, and that the year is 1800 or later.

Days in a month

In the Gregorian calendar there are thirty-one days in January, March, May, July, August, October, and December. There are thirty days in April, June, September, and November. February has twenty-eight days except in leap years, when it has twenty-nine.

Leap years

Leap years occur every four years on years divisible by four, but exclude years ending in '00', except for those divisible by four hundred. Thus, 2004, and 2008 are leap years, but 2100 is not. By the four hundred rule, 2000 was a leap year.

Coding exercise:

⇒ Build a `Date` class similar to that found in Chapter 6 of *Deitel and Deitel.*

- Perform error checking on the initializer values for `year`, `month`, and `day`.
- Modify the print functions to use ISO formatted dates (2007-02-05), rather than American "02/05/07" format.
- Provide member functions to get and set the year, month and day.
- Provide member functions `void addDay()` and `void addDays(int)` to increment the date by one or more days. The `Date` object should remain in a consistent state.

- [Bonus] Write a member function `int daysTo(Date const&) const` that computes the number of days between two dates.
- Provide member functions `Date tomorrow() const` and `Date yesterday() const` that return the day after or the day before the object on which they are called.

Notes

For this version of the `Date` class use, store the year, month, and day in three separate `int` variables. Feel free to add `private:` member functions that make it easier to implement your class.

Testing

Write a test program or test programs that carefully test the complete functionality of your `Date` class. Be sure to test that adding a day that takes you from one month to another, or from one year to another work correctly.

Try to write your test programs so that their output is short and easy for the reader to find and understand.

Re-implementation

⇒ Copy your code and tests to a new directory, and re-implement the `Date` class so that the year is stored in an `unsigned short` variable, and the month and day are stored in `unsigned char` variables.

- Write test programs that show the size of the old and new `Date` objects.
- Carefully write down what changes you need to make in your code in order to make the member functions work with your new representation. Provide these comments as a text file with the rest of your assignment.
- Show that your old test programs run correctly with the re-implemented `Date` class.