Date Class Demo

The Date class is introduced in the text, Chapter 7, page 375-380

What does this class do?

It allows us to write a date and its description into a Data object, hold the information for the Date object, and return a nicely formatted string describing the Date object.

We are gong to write two constructors for it and overload them and use a constructor initializer.

Download the DateClassDemoStarter project from Blackboard in this module. Unzip it and open it and we will get started.

1. Open the Date.h file.
2. Notice how a Date object will be defined. It has integers for month, day, year and a string for description. Then it has an int for dayOfYear. What’s that? That is the number of the day in the year, starting with 1 and ending with 365 or 366, depending on whether it is leap year. This is an additional piece of information about the Date object, as is bLeap, whether it is leap year.
3. It allows us to set the description separately as well as the 4 parameters that define the object. There are 5 separate Get methods for each parameter, except description. For that, we have GetFormattedDate. Hmmm.
4. You see there are no constructors.
5. Add a default constructor:
6. Add an overloaded constructor. It should have 4 parameters, just like the SetDate method.
7. Now open the Date.cpp file.
8. We can look at the SetDate method. It sets the values passed in to the class member variables and then calls DetermineLeapYear(); and CalcDayOfYear(); Now we know why the class has those other class variables and how they are set.
9. Add the implementation for the default constructor. We will have to use the <ctime> class and set the date variables to the computer’s current date.

//Set the Date variables to the computer's date.

time\_t rawtime;

tm OStime;

time(&rawtime);

localtime\_s(&OStime, &rawtime);

month = OStime.tm\_mon + 1;

day = OStime.tm\_mday;

year = OStime.tm\_year + 1900;

description = "Today's Date";

1. This means we have the basic variables, but we can call DetermineLeapYear(); and CalcDayOfYear(); for the others
2. Now let’s write the overloaded constructor,

Date::Date(int m, int d, int y, string desc): month(m), day(d), year(y), description(desc)

And again, call the two methods; DetermineLeapYear(); and CalcDayOfYear();

What happened? We assigned the values to the class variables using the constructor initializer. Nice. That means the variables were constructed with the values given. All in one step. This is a Best Practice.

1. Open the DateDriver file. This is where our project will begin in main. It is where we can create objects and use them:
2. Make one Date object using the default constructor. It contains our default data--which is today's date.

Date myDefdate;

1. Make another Date object and pass in our own data.

Date myOwnDate{ 1,10,1964,"Mom's Birthday" };

1. Now ask each Date object for its formatted string.

string sDefDate = myDefDate.GetFormattedDate();

string sOwnDate = myOwnDate.GetFormattedDate();

1. Display the results:

cout << "\nThe Default Date is:\n";

cout << sDefDate << endl;

cout << "My Own Date is:\n";

cout << sOwnDate << endl;

1. That’s it. Run your program.