COP3330 Programming Assignment 2

Date and day of week calculator for 2013

Objectives

* Implement a simple class with public and private members and multiple constructors.
* Gain a better understanding of the building and using of classes and objects.
* Practice problem solving using OOP.

Overview

You will implement a date and day of week calculator for the 2013 calendar year. The calculator repeatedly reads in three numbers (in a line) from the standard input that are interpreted as *month dayofmonth daysafter***,** calculates the dates in the year and days of week for the dates, and outputs the information. For example, input “1 1 31” is interpreted as the following: the first 1 means the 1st month in the year, January; the second 1 means the 1st day in January; and the 31 means 31 days after the date January 1, 2013 (we assume the year is 2013 to simplify the program), which is February 1, 2013. The program also calculates the days of week for each of the dates. More specifically, for input “1 1 31”, the calculator should produce the following output:

“31 days after Tuesday, January 1, 2013 is Friday, February 1, 2013. “

The first input number must be from 1 to 12 representing the 12 months in 2013, the second input number must be a day in the month (e.g. for 1-31 for January, 1-28 for February, and so forth). The third number is larger than or equal to 0. The program should report an error (and exit) if the input is incorrect. If a day is not in 2013, the program should output that. Following are a sample input file (redirect to be the standard input) and the corresponding output.

Input file:

1 1 20

1 1 31

2 1 0

1 1 32

4 5 0

2 1 28

1 1 59

6 10 100

7 20 300

12 20 2

Output:

20 days after Tuesday, January 1, 2013 is Monday, January 21, 2013.

31 days after Tuesday, January 1, 2013 is Friday, February 1, 2013.

0 days after Friday, February 1, 2013 is Friday, February 1, 2013.

32 days after Tuesday, January 1, 2013 is Saturday, February 2, 2013.

0 days after Friday, April 5, 2013 is Friday, April 5, 2013.

28 days after Friday, February 1, 2013 is Friday, March 1, 2013.

59 days after Tuesday, January 1, 2013 is Friday, March 1, 2013.

100 days after Monday, June 10, 2013 is Wednesday, September 18, 2013.

300 days after Saturday, July 20, 2013 is a date not in 2013.

2 days after Friday, December 20, 2013 is Sunday, December 22, 2013.

Details

1. This project has two components. The first component is to implement a *date2013* class specified in the following. The second component is a driver program that uses the date2013 class to realize the calculator.
2. The date2013 class should have two private data members of the *int* type, **d** and **m** with **m** encoding the month (1 – January, 2-February, …) and **d** being the day in the month. For example to represent April 5, **d = 5**, **m = 4**. To represent a date not in 2013, m=-1, d=-1.
3. The date2013 class should have the following public functions:

date2013();

date2013(int dd);

date2013(int dd, int mm);

void setdate(int dd, int mm);

void print();

void plusday(int days);

1. The default constructor date2013() should set the default date of the object to be January 1, 2013.
2. The parameter for the constructor with 1 parameter date2013(int dd) ranges from 1 to 365 (the code should report error and exit if the parameter is not in the range) and is day of year for the date in 2013. The constructor converts day of year to month and day of month. For example the 70th day of the year is March 11: the constructor should make m =3 and d = 11 when date2013(70) is invoked.
3. For the two parameter constructor date2013(int dd, int mm), you just need to make sure that month (mm) and day (dd) are legitimate and set d = dd, and m = dd. If either the day or the month is not legitimate, the code should report error and exit.
4. Setdate is similar to the two parameter constructor.
5. The print() function calculates the day of week for the date2013 object and output to the standard output the date in the form of ‘dayofweek, month day, 2013’. For example, when the object have d=1, m=1, the print should output “Tuesday, January 1, 2013”. For a date not in 2013 (m=-1 and d =-1), this routine should output “a date not in 2013”.
6. The *plusday(int days)* function modifies the m and d for the new date, which is *days* after the current date.
7. You should declare the class in a file called proj2.h; and the implement of the class (class functions) in a file called proj2.cpp. You should have a statement ‘#include “proj2.h”’ in proj2.cpp. You should test your implementation using the provided date2013\_test.cpp. To compile the code: ‘g++ proj2.cpp date2013\_test.cpp’.
8. Once your date2013 class is tested correctly, you should write a driver program called proj2\_driver.cpp that reads in lines of three numbers as specified earlier and generates the corresponding output as specified using the date2013 class. The driver is very small (about 20 lines of code) performing the basic IO.

Submission

The due time for this assignment is May 29 (Wendesday), 2013. 11:59pm.

Name your program as proj2.h, proj2.cpp, proj2\_driver.cpp. Tar all of your files for this assignment (which should include at least four files proj1.cpp and bug\_fixing\_log.txt), name the tar file yourlastname\_firstinitial\_proj2.tar and submit the tar file in blackboard. The bug fixing log file must follow the template given.

Grading policy

The program must work on linprog. O point for programs with any g++ compiler error on linprog. You will need to track compiling errors that **you fixed by yourselves** in a log for fixed compiler bugs that needs to have at least two entries for 5 points/each entry (10 points max) that are different from those in the bug fixing log in project 1.

* Program with no compiler error (20 points)
* Log for fixed compiler bugs (10 points)
* Implementation of the date2013 class, tested by a program similar to date2013\_test.cpp (55 points)
  + Default constructor and print (program able to print the default date, first output line for date2013\_test.cpp) (20 points)
  + Constructor with one parameter, second/third output line (10 points). The 10 points include error reporting points.
  + Constructor with 2 parameters, fourth and fifth output line (10 points). The 10 points include error reporting points.
  + setdate() function, sixth output line (5 points)
  + plusday() function, sixth to eighth output lines (10 points)
* Correct proj2\_driver.cpp (15 points)

Hints

1. Start the project as soon as possible.
2. You should first create proj2.h and have an empty implementation in proj2.cpp for each member function with the body of the function containing one line

*cout << “XXXX function has not been implemented.\n” ;*

1. You should then compile the proj2.cpp and date2013\_test.cpp and run the code. From here, you can implement and test one function after another until the whole class is implemented.
2. You should then implement the default construct date2013(), and write a simple print() function that simply prints out the value of **d** and **m**.
3. After that you can implement the member functions in the following order date2013(int dd, int mm), setdate(int dd, int mm), date2013(int dd), print(), playday(int days). Remember to compile and run the program at least once everytime you add one function.
4. You should write the driver only after your date2013 class is completely correct.
5. To compute the date for *X* days after the current date, one can convert the current date into the day of year by adding up all days in the previous months and the day in the month, add *X* to the day of year to obtain the new day of the year, and then convert it back to the month and day of month representation (using the reverse computation process to convert current date to day of year). For example, the day of year for April 5 is daysinmonth[Jan] + daysinmonth[Feb] + daysinmonth[March] + 5.
6. To compute day of the week for a given date, one can have the following encoding: 0 – Sunday, 1-Monday, 2-Tuesday, 3-Wednesday, 4-Thursday, 5-Friday, 6-Saturday. Using this encoding, one can first convert the date into day of year, let it be X. Since January 1, 2013 is a Tuesday, the encoded day of week = (2 + X -1) % 7.
7. For reference, a sample proj2.cpp has 125 lines of code, proj2\_driver.cpp has 23 lines.