**9CIS 2275 C++ Programming Part II**

# Program 3 Calendar Class

# Write a C++ Class to Generate a Monthly Calendar

**Turn In Requirements:**

1. **5 pts Name your Visual C++ 2019 project LastnameP3, such as NelsonP3.**
2. **5 pts Upload to Blackboard, remove the intermediate files/ folders from your project before uploading it to Blackboard.**

**Program Requirements:**

1. **3 pts Write your name, email address and file name at the top of your source code in a comment.**
2. **5 pts Your main function should have cout statements that write “header” information to the screen. The header info includes your name, course and program information, as well as a 1-2 line description of the program.**
3. **5 pts. Use good C++ programming style and formatting for your program. Use appropriate comments to explain what you are doing.**

This program is a re-write of Program 1 so that the monthly calendar file is built by a C++ class.

You will write a C++ class, CalendarMonth that contains the 6 x7 integer array, monthGrid, as well as arrays for names and lengths for the 12 months and the names for the 7 days of the week. Any additional collections will be vectors. The class does not perform cout or cin functions so the AskForMonth function will have to either be in main or in a set of Functions files. Also, your header is not a function of the CalendarMonth class, so it also should be moved.

**Public methods:**

There will be two constructors for the class, one default and one overloaded to set the user’s month and year. Your class should also contain a set function for the user’s desired month and year. A GetFormattedString method will return a string to display the nicely formatted month. This display should be a neatly laid out month including the name centered across the top of the page, the day names across the top of the columns and the individual date spaced along the rows. The method IsFileOK will return the status of writing the file to main.

**Private Methods:**

CheckLeapYear will set the class variable bLeapYear to true or false.

CreateMonthGrid will take care of defining the calendar for the month as it did in the last program. It will also construct the nicely formatted string which will represent the calendar month. It will be called last from the overloaded constructor and the set function.

WriteMonth will do the work of writing the Month\_Year.txt file containing the calendar. It constructs the filename from the MonthName and the year and “.txt”. For example, the filename will then be “September 1997.txt”. In addition, there will be two additional lines for the month’s fact and the special date. It will be called from CreateMonthGrid. You may add private methods for CreateString and you first day of the month algorithm, if you need to.

We will add some more features to this class. We will add a vector of facts for each month. Each fact should reflect some seasonal or activity-related item pertaining to the month and will be written below the monthly calendar. Also, for each month, identify one day, perhaps a holiday, that should be highlighted using \* \* around the date. The string indicating that special day will also be written below the monthly calendar.

You will have five files: the Driver, which will contain your main function, CalendarMonth.h, containing your method prototypes, CalendarMonth.cpp, containing method bodies, Functions.h and Functions.cpp. Be sure to use include guards around your prototypes in both .h files.

**Exception handling:** We will also add exception handling to this program. First, create a custom exception for an InputOutOfRange exception. I had mine be derived from the out\_of\_range exception from the C++ standard exceptions. So I used #include <stdexcept>.

Typically, when we do this, it is only the Message that changes and indeed, I passed a custom message to the parent constructor. We also need to handle this exception. This will be accomplished in 3 steps:

1. Throw the exception object. Where? When we receive the user input and assign it into our class member variables. So, in the SetCalMoInfo method. And where else? How about the overloaded constructor. Check the inputs and if they are outside the range of month is 1-12 and year is < 1600, then throw the exception. If not, just assign the values into month and year.
2. Need to put a try block somewhere. It needs to be somewhere that can communicate with the user, so in the driver. Put the try just before SetCalMoInfo is called and enclose the rest of the code until the user is asked if they want to do another.
3. Need a catch block directly after the try block. It will catch an InputOutOfRange& object. And it can display an additional message as well as the object.what() message.

**Program Flow:**

In main, first call the header function. The Header is NOT a CalendarMonth function. The class dose not interface with the user. Create two files, Functions.h and Functions.cpp to contain the user interface functions.

That will include the Header function and a function called AskCalendarMonth, where you ask the user to enter month and year. It will use references or pointers to “return” month and year to the calling function , main. You may also write a “do another” function if you wish.

After calling the Header function, main will create any useful variables and Instantiate a CalendarMonth object.

Then open a do-while or while loop. Call AskCalendarMonth to get the month and year. Set the information into the CalendarMonth object, and then cout the formatted String provided by the object.

Call IsFileOk to check if the file was opened and report the filename if the write was successful. If unsuccessful, report that also.

Ask the user if he/she wants to estimate any other operations. If yes, loop back up to AskCalendarMonth. Otherwise, drop out of the loop.

As the program exits, write a good-bye message to the user.

Here is an example of how the output might look:

September 1928

Sun Mon Tue Wed Thu Fri Sat

1

2 3 4 5 6 7 8

9 10 11 12 13 14 15

16 17 18 19 20 21 22

\*23\* 24 25 26 27 28 29

30

\* September 23 is Autumnal equinox \*

September is Better Breakfast Month

Remember, it’s really hard to get the formatting perfect, so just get it “good enough” and focus on learning C++ classes!

Here is the class declaration:

class CalendarMonth

{

private:

int grid[6][7]{};

int monthDays{30};

int month{1};

int year{2020};

int firstDay{1};

string formattedCalendar;

9string filename;

bool bLeapYear{false};

bool bFileOK{true};

string dayNames[7];

string monthName[12];

int monthLength[12] = {};

vector<string> vMonthComment;

vector<string> vSpecialDateComment;

vector<int> vSpecialDate;

//private functions

void CheckLeapYear();

void WriteMonth();

public:

CalendarMonth();

CalendarMonth(int m, int y);

void SetCalMoInfo(int m, int y);

void CreateMonthGrid();

string GetFormattedString(){return formattedCalendar;}

bool IsFileOK(){return bFileOK;}

string GetFilename(){return filename;}

};