Next Date Calculator

for

SE 420

Prepared by

Doyle D. Bigelow
Diego R. Draguicevich

April 16, 2021

A brief description of the program and how it works.

The chosen language for the project was C# due to the ease of creating a form in Windows with the language. The basic UI has a masked textbox with the format of mm/dd/yyyy. The masked textbox is the manual way to enter dates for a next date check. The alternative is a button labeled "Select BVA Value". When this is clicked on the button is replaced with a combo box that is filled with BVA dates loaded from a text file. When one of these dates is selected the combo box is replaced with the button, and the selected date fills the masked text box. When the user is ready to get the next date they click the big button labeled "Calc Next Date".

Once the user requests the next date the first step is to check if the date input is a valid date. The validate first checks to see if the year is within the given 1900 to 2025 range. Then it checks if the month is between 1 and 12. Then the last part is to check that the day is between 1 and the max number of days in the selected month. A function max days in month takes the month and year and returns the numbers of days in the given month.

The max days in a month function is a switch function where every 31 day month is grouped together and returns 31, and every 30 day month is grouped together and returns 30. The outlier is February which calls a function that calculates the max days in February based on the year. The function checks if the year is divisible by 4 meaning that it could be a leap year. The function then checks to see if the year is divisible by 100, if so it may actually not be a leap year. If it was divisible by 100, the function then checks if the year is divisible by 400, if true means it is a leap year otherwise it is not. At the end of this function it checks whether or not it was a leap year and returns 28 or 29 accordingly.

If the month, day, and year were all within acceptable bounds then the date is returned as valid. The next part will be to get the next day. This is done by incrementing the day by one and checking if it is valid. If it is, that is the next date and will be displayed. Otherwise day will be set to 1 and month will be incremented and checked for validation. If the date is valid once again return that date, otherwise month is set to 1 and year is incremented. The requirements do not require the return date to be less than 2025, just the input, so there is no date validation needed for this step, just to return the date.

A table showing all the test cases in the test suite using equivalence partitioning and boundary value analysis and any other test case. (Diego)

There are a variety of possible outputs based on the inputs:

Year	Month	Day	Expected Result
$y < 1900$	-	-	Invalid
$2025 < y$	-	-	Invalid
$1900 \leq y \leq 2025$	m = 1,3,5,7,8,10,12	$1 \leq d < 31$	m/d + 1/y
$1900 \leq y \leq 2025$	m = 1,3,5,7,8,10	d = 31	m + 1/01/y
$1900 \leq y \leq 2025$	m = 12	d = 31	01/01/y + 1
$1900 \leq y \leq 2025$	m = 4,6,9,11	$1 \leq d < 30$	m/d + 1/y
$1900 \leq y \leq 2025$	m = 4,6,9,11	d = 30	m + 1/01/y
$1900 \leq y \leq 2025$	m = 4,6,9,11	$30 < d$	Invalid
-	-	$d \leq 0$	Invalid
-	-	$31 < d$	Invalid
-	$m \leq 0$	-	Invalid
-	$12 < m$	-	Invalid
$1900 \leq y \leq 2025$	m = 2	$1 \leq d < 28$	02/d + 1/y

-	m = 2	29 < d	Invalid
1900 ≤ y ≤ 2025 y % 4 != 0	m = 2	28 < d	Invalid
1900 ≤ y ≤ 2025 y % 4 != 0	m = 2	d = 28	03/01/y
1900 ≤ x ≤ 2025 y % 4 = 0 y % 100 != 0	m = 2	1 ≤ d < 29	02/d + 1/y
1900 ≤ x ≤ 2025 y % 4 = 0 y % 100 != 0	m = 2	d = 29	03/01/y
1900 ≤ y ≤ 2025 y % 4 = 0 y % 100 = 0 y % 400 = 0	m = 2	1 ≤ d < 29	02/d + 1/y
1900 ≤ x ≤ 2025 y % 4 = 0 y % 100 = 0 y % 400 != 0	m = 2	28 < d	Invalid

The above rules are then covered by the following test cases:

Test Approach	Test Case	Expected Result
BVA + EP	12/31/1899	Invalid
BVA	01/01/1900	01/02/1900
BVA	12/31/2025	01/01/2026
BVA + EP	01/01/2026	Invalid
BVA	01/30/1997	01/31/1997
BVA + EP	01/31/1997	02/01/1997
BVA	03/30/1997	03/31/1997
BVA + EP	03/31/1997	04/01/1997

BVA	05/30/1997	05/31/1997
BVA + EP	05/31/1997	06/01/1997
BVA	07/30/1997	07/31/1997
BVA + EP	07/31/1997	08/01/1997
BVA	08/30/1997	08/31/1997
BVA + EP	08/31/1997	09/01/1997
BVA	10/30/1997	10/31/1997
BVA + EP	10/31/1997	11/01/1997
BVA	12/30/1997	12/31/1997
BVA + EP	12/31/1997	01/01/1998
BVA	04/29/1997	04/30/1997
BVA + EP	04/30/1997	05/01/1997
BVA	06/29/1997	06/30/1997
BVA + EP	06/30/1997	07/01/1997
BVA	09/29/1997	09/30/1997
BVA + EP	09/30/1997	10/01/1997
BVA	11/29/1997	11/30/1997
BVA + EP	11/30/1997	12/01/1997
BVA + EP	04/31/1997	Invalid
BVA + EP	06/31/1997	Invalid
BVA + EP	09/31/1997	Invalid
BVA + EP	11/31/1997	Invalid
BVA + EP	00/01/1997	Invalid
BVA + EP	01/00/1997	Invalid
BVA + EP	01/32/1997	Invalid

BVA + EP	13/01/1997	Invalid
BVA	02/27/1997	02/28/1997
BVA + EP	02/28/1997	03/01/1997
BVA	02/29/1997	Invalid
BVA	02/28/1900	03/01/1900
BVA	02/29/1900	Invalid
BVA	02/28/2000	02/29/2000
BVA	02/29/2000	03/01/2000
BVA + EP	02/30/2000	Invalid
BVA + EP	02/28/2004	02/29/2004
BVA + EP	02/29/2004	03/01/2004

Note that the combination of equivalence partitioning and boundary value analysis is NOT exhaustive, but is likely to catch the vast majority of issues by testing each of the relevant rules for this program.

(Individual) Team members reflection about the project, problems encountered, solutions, and the major task assigned to each team member.

Doyle D. Bigelow

The code itself was not a massive issue as initially I thought it would be. The main issue was compacting and of the code; finding the best method to get the max range in in a month efficiently was. We decided with a fall through switch that returned a number but under February it called another function that then did the modulus comparisons. The other hardest thing was setting up the UI using C#, it wasn't that difficult, it just required some documentation reading. I

was assigned code while Diego did the BVA and EPA analysis. We both helped on the other part but were primarily responsible for our sections.

Diego R. Draguicevich

The number of cases for testing was far more than expected. The actual rules list is rather short, even when expanded out in tabular form is merely 20 cases. However, the final list of test cases is rather expansive, demonstrating the impracticality of manually testing large numbers of cases. I was pleasantly surprised to note the simplicity of actually coming up with the test cases; once the rules were established, it was straightforward to set up each test case.