CSC 101 Lab Week 3
Number of a Day in the Year

ISSUED: Monday, 9 April 2011
DUE: Monday, 16 April 2011, by the end of lab
POINTS POSSIBLE: 1
WEIGHT: 1% of total class grade

Overview

The problem for this lab is figuring out what day of the year it is. This is an extended version of Problem 7 in Chapter 4 of the book.

Write a program that determines the day number (1 to 366) in a year for a date that is provided as input data. As an example, January 1, 1994, is day 1. December 31, 1993, is day 365. December 31, 1996, is day 366 since 1996 is a leap year. A year is a leap year if it is divisible by four, except that any year divisible by 100 is a leap year only if it divisible by 400. Your program should accept the month, day, and year as integers. The program will output the day number to the user.

Sample Runs

The following sample runs show precisely what your program input and output should look like:

Sample Run 1:

```
Enter a month number: 2
Enter a day of that month: 28
Enter the year: 1999
That is day 59 of that year!
```

Sample Run 2:

```
Enter a month number: 12
Enter a day of that month: 31
Enter the year: 1995
That is day 365 of that year!
```

Sample Run 3:

```
Enter a month number: 12
Enter a day of that month: 31
Enter the year: 1996
That is day 366 of that year!
```

Sample Run 4:

```
Enter a month number: 1
Enter a day of that month: 1
Enter the year: 2000
That is day 1 of that year!
```

Sample Run 5:

```
Enter a month number: 19
Enter a day of that month: 23
Enter the year: 2000
Error there are only 12 months in any year
```

Sample Run 6:

```
Enter a month number: 10
Enter a day of that month: 32
Enter the year: 1000
Error: no month has more than 31 days
```

Sample Run 7:

```
Enter a month number: 4
Enter a day of that month: 31
Enter the year: 2009
Error: that month only has 30 days
```

Sample Run 8:

```
Enter a month number: 2
Enter a day of that month: 30
Enter the year: 2000
Error: it’s a leap year, so that month has 29 days
```

Exercise 1: Implement the Program without Error Checking

Implement the program described in the overview, assuming that the user provides correct values for all of the inputs. This version of the program will perform sample runs 1 through 4 correctly, however it will not output the error messages in sample runs 5 through 8.

In your implementation of this exercise, use only `if-else` statements in the logic of your program. Exercise 3 below will have you use a `switch` statement.

Save this version of the program in a file name `daynum.c`.

Exercise 2: Add Error Checking

Refine the program of exercise 1 to perform the following error checking:

- The month number is between 1 and 12.
- The day number is a minimum of 1.
- Depending on the month, the day number is a maximum of 28 to 31. Use the normal "30 days has September ..." rule to determine the maximum number of days in a particular month.
- Account for the leap year when checking the maximum day in February.

As with Exercise 1, use only `if-else` statements, without a switch. Save this version of the program in file named `daynum_2.c`.

Exercise 3: Use a Switch Statement

Refine the program of exercise 2 by using at least one `switch` statement, in place of some of the `if-else` statements. It doesn’t make sense to replace all of the `if-else` logic with switches. However, there is at least place case where it makes pretty good sense. It is up to year to figure out where it is sensible to use a `switch`.

Save this version of the program in a file name `daynum_3.c`.

Exercise 4: Think about Additional Error Checking

Try the following inputs on your solution to Exercise 3:

```
Enter a month number: February
Enter a day of that month: twenty-eight
Enter the year: nineteen ninety-nine
```

What does your program do? Why does it behave the way it does? What would it take to have your program output an error message when the user enters alphabetic strings instead of numbers?

You only need to think about these questions for this lab, and give a reasonable answer when you give your live demo. You do not need to implement the checking for numeric-only inputs.

Program Testing and Live Check of Your Program in Lab

The sample runs above provide some useful test cases to check that your program runs correctly. You should test it on additional cases to convince yourself that it it is operating correctly, according to the specifications above.

When you think it works, I will come around and look at your results, and have you answer the questions in Exercise 4. I will also ask you to run your program on some test cases in addition to those in the samples above.

In upcoming lectures, we will discuss the theory and practice of thorough program testing. In particular, we will begin to answer the question "How may test cases is enough?".

Submitting Your Program

After your program has been checked in person during lab, submit it on unix1 using the command:

```
    handin gfisher 101_lab3 daynum.c daynum_2.c daynum_3.c
```

Note that you are handing in three separate files, for exercise 1, 2, and 3.