Alex Shoop

due 9/6/13

CS 2303 Assignment 1

**Summary**

I wrote a calendar program in C language. It asks the user for a non-negative number, which represents the year that they want the calendar for. The program then executes and prints out the entire 12-month calendar for the given year.

**How to run it**

Type a non-negative number and press enter.

**How it works**

There are two functions besides the main function, called findDaysInMonth and findStartingDay respectively.

The first function finds the number of days of a given month. It knows to check if February has 28 or 29 days from the global variable inputYear, which is the year that the user inputs.

The second function finds the starting day (ie. January 1) of a given year. The algorithm used is called Zeller’s Algorithm1. It calculates the day of week; this value will tell us what day of the week inputYear starts on (0 = Sunday, 1 = Monday, etc.). K is the year of the century; this is calculated by doing mod(inputYear, 100). J is the actual century, which is calculated by doing . We then calculate R which holds most of the calculations. It’s best to keep in mind that since we only care about January 1, the month m must be 13 and the inputYear must actually be inputYear – 1 in the calculation, since the algorithm accepts January only if it’s a number 13 and if it’s 1 year before. After the calculation of R, we have to modify the modulo calculation because the original algorithm calculates a different day value (0 = Saturday, 1 = Sunday, etc.). In the end, we end up with a number between 0 and 6 inclusively.

The program exits if the user input a non-integer input.

Once the user inputs a number, it gets stored in inputYear. “Calendar for %d” is written on screen. startingDay is then assigned to findStartingDay(inputYear).

Then a large for loop outputs the month name, days of the week, and the day numbers.

Within that large for loop, there’s another for loop that checks if dayOfWeek (ie. the block of space of a certain day of the week) is less than the startingDay (ie. block of space of the starting day of the year). That small for loop prints blanks until dayOfWeek equals startingDay.

There’s another for loop that starts with dateNumber (ie. the day number) as 1 and prints out the numbers each day. When it gets to Saturday, which it knows by an if statement checking if dayOfWeek is less than 6 (ie. Saturday) and the date number does not equal daysInMonth (ie. the number of days in the month), it goes to the next line and sets dayOfWeek to 0 (ie. Sunday).

After the dateNumber for loop finishes, startingDay gets set to dayOfWeek++ modulo 7; this means that startingDay of the next month should be set to the next day, and that it should fall between the 0 to 6 range thanks to the modulo 7.

Once the large for loop ends, we have the whole 12-month calendar of inputYear.

**Loop Invariants**

For monthNumber loop:

1. Statement; holds month number and also holds number of days in month from findDaysInMonth function. In addition to outputting the month name and days, it makes sure that startingDay gets updated for the next month.

2. Statement is true after for loop; the month number gets incremented and goes through findDaysInMonth function again and the startingDay gets incremented as well as goes through modulo 7. The startingDay always get set to the startingDay of the next month.

For dayOfWeek loop:

1. Statement; holds dayOfWeek number, checks if it is less than the value of startingDay, and prints blanks of 5 digit space.

2. Statement is true after for loop; the dayOfWeek number gets incremented and finishes once dayOfWeek becomes equal to startingDay. The dayOfWeek gets set to 0 for each month.

For dateNumber loop:

1. Statement; holds dateNumber and checks if it is less than or equal to the daysInMonth, then outputs the dateNumber.

2. Statement is true after for loop; the dateNumber gets incremented and is always less than or equal to the daysInMonth. The dateNumber is always 1 for each month.

**Encountered issues**

Initially I did not know how to find the startingDay algorithm. I tried to make my own algorithm from the teacher’s suggestion of working off of an already known date, but I couldn’t figure it out. I found Zeller’s Algorithm and it was very helpful.

**Sources**

1 Zeller’s Congruence, <http://en.wikipedia.org/wiki/Zeller's_congruence>

Page 41 of K&R C programming language book

Lukas Hunker (llhunker) friend, who’s also currently in the same class

Loop Invariants powerpoint slide from Professor Lauer