Assg 04: Finding Prime Numbers

CSci 515 Spring 2015

2015-01-23

Dates:

Due: Tuesday February 17, by Midnight

Objectives

- Write functions in C that take 1 or more parameters and return a result.
- Learn and practice breaking up a larger problem into smaller sub-parts.
- More practice using control structures for looping and conditional execution in C.
- Practice with mathematical operations and concepts, like the modulus operator.

Description

As you should know, an integer is said to be **prime** if it is divisible by only 1 and itself. For example, 2, 3, 5 and 7 are prime, but 4, 6, 8 and 9 are not.

Your overall task is to write a function, called `findPrimes()` that will find all of the prime numbers in a range from M to N. To solve this task, you will break down the big problem into two sub-problems:

1. Write a function called `isPrimeNumber()` that determines whether some particular number is a prime number or not.
2. Write a function, called `findPrimes()` that will use the first function to search for primes in a range of values.

If you recall we gave an example of a brute force method for determining if a number is a prime or not previously. A simple method to determine if a number X is a prime is to test all possible divisors from 2 to X/2. If any number is found that can divide the number X evenly, then the number can not be a prime. If no such divisor is found, then you have proven the number is a prime. Recall that you can use the modulus operator '%' in C to test if 2 numbers are evenly divisible (with no remainder). Your `isPrimeNumber()` function should take a single integer value as its input parameter. The function will return a `bool` Boolean value as its result. Your function should return `true` if the number passed into it in its parameter is a prime, and it should return `false` otherwise.

The second function you should write will use your first function to search for prime numbers in a range of numbers. Your second function should be called `findPrimes()`. This function will take 2 integer parameters as input, the `beginRange` and `endRange` of the range that the function should search within (inclusive) for prime numbers. The function will not return any result, so it will be a `void` function. The function will communicate its results by displaying prime numbers that it finds to standard output.

Your program should initially prompt the user for the minimum and maximum values that should be used for the range that is to be searched for primes. This prompt and input of values should be done in your `main()` function. The output from running your program with the described functions should look exactly like this:

```
This program finds all of the prime numbers within a range from M to N.
At what value should we begin searching: 2
At what value should we end searching: 100
2 is a prime number
3 is a prime number
5 is a prime number
7 is a prime number
11 is a prime number
13 is a prime number
17 is a prime number
19 is a prime number
23 is a prime number
29 is a prime number
31 is a prime number
37 is a prime number
41 is a prime number
```

```
43 is a prime number
47 is a prime number
53 is a prime number
59 is a prime number
61 is a prime number
67 is a prime number
71 is a prime number
73 is a prime number
79 is a prime number
83 is a prime number
89 is a prime number
97 is a prime number
```

NOTE: Now that our programs have more functions than just the `main()` function the use of the function headers becomes meaningful. Make sure that all of your functions (`main`, `findPrimes`, `isPrimeNumber`) have function headers before them that document the purpose of the functions, and the input values and return value of the function.

Lab Submission

An eCollege dropbox has been created for this lab. You should upload your version of the lab by the end of lab time to the eCollege dropbox named `Lab 04 Finding Primes`. Work submitted by the end of lab will be considered, but after the lab ends you may no longer submit work, so make sure you submit your best effort by the lab end time in order to receive credit.

Requirements and Grading Rubrics

Program Execution, Output and Functional Requirements

1. Your program must compile, run and produce some sort of output to be graded. 0 if not satisfied.
2. 40+ pts. Your program must have the 2 required named functions, that accept the required input parameters and return the required values (if any).
3. 30+ pts. Your algorithm for determining if a number is a prime in the `isPrimeNumber()` function must work correctly.

4. 30+ pts. Likewise the `findPrimes()` function must work, and produce the output as shown for the assignment.

Program Style

Your programs must conform to the style and formatting guidelines given for this course. The following is a list of the guidelines that are required for the lab to be submitted this week.

1. The file header for the file with your name and program information and the function header for your main function must be present, and filled out correctly.
2. A function header must be present for all functions you define. You must document the purpose, input parameters and return values of all functions.
3. You must indent your code correctly and have no embedded tabs in your source code. (Don't forget about the Visual Studio Format Selection command).
4. You must not have any statements that are hacks in order to keep your terminal from closing when your program exits.
5. You must have a single space before and after each binary operator.
6. You must have a single blank line after the end of your declaration of variables at the top of a function, before the first code statement.
7. You must have a single blank space after , and ; operators used as a separator in lists of variables, parameters or other control structures.
8. You must have opening `{` and closing `}` for control statement blocks on their own line, indented correctly for the level of the control statement block.

Failure to conform to any of these formatting and programming practice guidelines for this lab will result in at least 1/3 of the points (33) for the assignment being removed. Failure to follow other class/textbook programming guidelines may result in a loss of points, especially for those programming practices given in our Deitel textbook that have been in our required reading so far.