Assg 05: Binomial Coefficient.

CSci 515 Spring 2015

2015-02-11

Dates:

Due: Tuesday February 24, by Midnight

Objectives

- More practice writing C functions
- Learn about implementing algorithms using iteration and recursion.
- Practice writing a recursive function.
- Become more familiar with implementing mathematical series on a computer.

Description

In this assignment we will write a recursive function to calculate what is known as the binomial coefficient. The binomial coefficient is a very useful quantity, it allows us to count the number of ways of selecting i items out of a set of n elements. For example, if we have 3 items `A, B, C`, there are 3 ways to choose 1 element from the items: choose A, or choose B or choose C. There are also 3 ways to choose 2 elements from the items: AB, AC, BC. There is only 1 way to choose 3 elements form a set of 3 items: ABC. When we choose 2 elements from a set of 3 items, we normally speak of this as 3 select 2, and mathematically we write this as a binomial coefficient

$$\binom{3}{2} = 3$$

Where the result of the binomial coefficient is to count up the number of combinations we will have for n items when we select i elements. As another example, just to make this clear, if we have a set of 4 items, and we choose 2 elements, we get: AB, AC, AD, BC, BD, CD = 6:

$$\binom{4}{2} = 6$$

Mathematically we can compute directly the number of combinations for n choose i using factorials:

$$\binom{n}{i} = \frac{n!}{i!(n-i)!}$$

Where ! represents the factorial of a number, as we discussed in class.

However, another way of computing the number of combinations is by defining a recursive relationship:

$$\binom{n}{i} = \binom{n-1}{i-1} + \binom{n-1}{i}$$

You can think of this as a recursive function that takes two parameters n and i, and computes the answer recursively by adding together two smaller subproblems. For this recursive definition of the binomial coefficient, the base cases are:

$$\binom{n}{0} = \binom{n}{n} = 1$$

We have already seen why n items choose n elements will always be 1. The other base case is used by definition, and simply means that there is only 1 way of choosing no items from a set (e.g. you don't choose).

In this assignment you will write 3 functions. You need to first of all write a function to compute the number of combinations of n items choosing i elements directly using our first formula (involving factorials). You should also (re)create the factorial function (that we developed in class) that you will call from this function to compute factorials of numbers. You will also implement a recursive version of calculating the number of calculations, using the recursive definition of the binomial coefficient and the base cases given above.

Perform the following tasks:

1. Write a function called `factorial()`. This function should take a single integer as its parameter, and it will compute the factorial $n!$ of the value and return it as its result.

2. Write a function called `countCombinationsDirectly()`. This function will compute the number of combinations that result from choosing i elements from a set of n items. The function should take two integer values as its parameters n and i, which will be integers $>= 0$. The function should directly compute the number of combinations, using the equation involving factorials of n and i. The function will return a single integer as its result.

3. Write a second function called `countCombinationsRecursive()`. This function will also count the number of combinations of choosing i elements from a set of n items. However, you need to implement this calculation as a recursive function, using the recursive definition and two base cases given above. Your function will take the same two integer parameters n and i with values $>= 0$, and will return an integer result, the count of the number of combinations from n choose i items.

4. In your `main()` function, prompt the user to enter a value for n the number of items in a set to choose from, and i, the number of items to select from that set. You will calculate the number of combinations using both of your methods, and display the result.

Your program output should look something close to the following when I run your program. I have run the program multiple times, so that you can also see some examples of correct output for some different values of n and i:

```
$ assg05
Enter n (an integer >= 0), the number of items in a set to choose from: 4
Enter i (an integer >= 0), the number of elements to select from the set: 2
The number of combinations (determined directly)    of 4 choose 2 is: 6
The number of combinations (determined recursively) of 4 choose 2 is: 6

$ assg05
Enter n (an integer >= 0), the number of items in a set to choose from: 4
Enter i (an integer >= 0), the number of elements to select from the set: 0
The number of combinations (determined directly)    of 4 choose 0 is: 1
The number of combinations (determined recursively) of 4 choose 0 is: 1
```

```
$ assg05
Enter n (an integer >= 0), the number of items in a set to choose from: 4
Enter i (an integer >= 0), the number of elements to select from the set: 4
The number of combinations (determined directly)    of 4 choose 4 is: 1
The number of combinations (determined recursively) of 4 choose 4 is: 1

$ assg05
Enter n (an integer >= 0), the number of items in a set to choose from: 9
Enter i (an integer >= 0), the number of elements to select from the set: 4
The number of combinations (determined directly)    of 9 choose 4 is: 126
The number of combinations (determined recursively) of 9 choose 4 is: 126

$ assg05
Enter n (an integer >= 0), the number of items in a set to choose from: 10
Enter i (an integer >= 0), the number of elements to select from the set: 6
The number of combinations (determined directly)    of 10 choose 6 is: 210
```

NOTE: Now that our programs have more functions than just the `main()` function, the use of the function headers becomes meaningful and required. Make sure that all of your functions (`main`, `nthFibonacciIterative`, `nthFibonacciRecursive`) have function headers preceding them that document the purpose of the functions, and the input parameters and return value of the function.

Assignment Submission

An eCollege dropbox has been created for this assignment. You should upload your version of the assignment to the eCollege dropbox named `Assg 05 Fibonacci Sequence` created for this submission. Work submitted by the due date will be considered for evaluation.

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. 25+ pts. Your program must have the 2 required named functions, that accept the required input parameters and return the required values (if any).

3. 25+ pts. Your iterative implementation must use loops/iteration to implement its calculation. The function must of course correctly compute the n^{th} term of the series.

4. 40+ pts. Your recursive implementation must perform its calculation using recursion. You must have the correct base cases defined. Your function must of course correctly compute the n^{th} term of the series. trials, and count up the successful trials from all of the trials performed, and return the correct probability ratio. Your ratio must be correct.

5. 10+ pts. You must prompt the user for n in main, and correctly display the returned results form your function as shown.

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 assignment 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 assignment will result in at least 1/3 of the points (33) for the assignment being removed for each guideline that is not followed (up to 3 before getting a 0 for the assignment). 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.