Project Overview

- Read a maximum number
- Loop until maximum number entered is < 2
  - Open a user-specified output file.
    - Prompt for the file name, echo print the filename
    - Open the output file
    - Use a while loop to handle file open failures. Continue to loop until an output file is successfully opened
  - Find prime numbers up to the maximum number entered using two methods
    - Print out headings to for each method to the output file
    - Print out the prime numbers for each method to the output file
  - Compare the efficiency of the two methods
  - Close the output file and read another maximum number

Assumptions and Constraints

- ***Arrays and user defined functions are not allowed!!***
- ***Output shall match that of the sample solution***

- Program is to give user multiple attempts at opening an output file
- Use a max number less than 100,000 (field width for prime numbers is 6 – so printing numbers greater than 100,00 results in no spaces between numbers)
- Assume that only integers are entered for the maximum number
- Program continues to loop on the maximum number entered

Algorithm - Part I

Prompt the user for a maximum number

Loop as long as the number is >= 2

Open output file specified by user

If unable to open the file, enter a loop that outputs an error message, resets the file stream and obtains another file to open

Output header for method 1

Use Method 1 to find all primes from 2 to number entered

Output trailer for method 1

Algorithm - Part II

Output header for method 2

Use Method 2 to find all primes from 2 to number entered

Output trailer for method 2

Output number of prime numbers found, number of iterations required for each method and the ratio of the iterations (see the sample solution output file)

Close the output file ← important step

Prompt for and read another number // end of loop

Finished when number < 2

Algorithm - Part III – Method 1

Method 1 for finding prime numbers – brute force

A for loop starting at 2 going to number entered

Increment iteration counter

Initialize inner loop as necessary (flag indicating number is prime)

A for loop starting at 2 going to outer loop control variable value – 1

Increment iteration counter

if inner loop control variable divides into outer loop control variable, then

exit inner loop (set loop control variable to outer loop control variable)

loop control variable value is not a prime number

End of inner loop

If outer loop control variable is prime, then write it to the output file

End of outer loop

Algorithm - Part IV

Method 2 is modified from method 1 as described in the project description

When printing the prime numbers, use the following to terminate the line of numbers (modification to variable names is necessary)

If (numPrimes1 % MAX_PER_LINE == 0)
      outFile << endl;

(note numPrimes1 is the count of the number of prime numbers found so far, and MAX_PER_LINE is a constant equal to 10)

A similar line is used when the outer for loop of the method is exited – this time the current line is terminated if the prime number count is not a multiple of 10:

If (numPrimes1 % MAX_PER_LINE != 0)
      outFile << endl;

General Program Instructions

1. Declare all variables at the beginning of your functions (in this program *main* is the only function) – This means do not place variable declarations throughout your code.

2. Be sure to end *main* with a *return 0;* statement.

3. Close all open file streams before exiting from main.

4. Read the project description – especially the hints section

5. **Run the sample solutio**n. Look at the output file generated

6. Can use redirected input with the input files provided:
./Project_06 < P8_in1.txt

Output/Heading Information

- **All output is left justified**
- Lines of dashes ('-') and asterisks ('*') are 60 characters long
- Prime numbers are printed in a field width of 6 left justified
- Look at output file generated by sample solution:
  - Shows headings and separation lines
  - Shows output of prime numbers
  - Shows information output after method 2