ENGG1410: Introductory Programming for Engineers

Lab #4 | “Introduction to C Programming”: Looping in C and Debugging a Program

Eric Cao, Jeremiah George

October 1, 2024

Contents

[Question 1 – Series Approximation of Pi 3](#_Toc179742035)

[Objective 3](#_Toc179742036)

[Constraints 3](#_Toc179742037)

[Pseudocode 3](#_Toc179742038)

[Question 2 – Greatest Common Divisor 4](#_Toc179742039)

[Objective 4](#_Toc179742040)

[Constraints 4](#_Toc179742041)

[Pseudocode 4](#_Toc179742042)

[Question 3 – Finding Prime Numbers in an Interval 5](#_Toc179742043)

[Objective 5](#_Toc179742044)

[Constraints 5](#_Toc179742045)

[Pseudocode 5](#_Toc179742046)

[Question 4 – Debugging Sum Expression 6](#_Toc179742047)

[Objective 6](#_Toc179742048)

[Constraints 6](#_Toc179742049)

[Errors and Modifications 6](#_Toc179742050)

# Question 1 – Series Approximation of Pi

## Objective

Write a program that prompts the user for the number of terms to include in the following series that approximates pi:

## Constraints

* The input must be a non-negative integer, or else the user will be prompted again for a correct value.
* The number of terms and the calculated approximation of pi is output to the terminal.

## Pseudocode

*integer variables:* termCount

*double variables:* piApproximation = 0

**do** {  
 termCount <- **receive input**

**if** (termCount < 0) **output** -> invalid input, try again

} **repeat if** (termCount < 0)

**for** (*integer:* index = 0, **while** index <= termCount) {

piApproximation **=** piApproximation

+ (1 **if** termCount is divisible by 2, **else** -1) \* 1/(index \* 2 + 1)

index = index + 1;

}

pi = pi \* 4

**output** -> termCount, piApproximation

# Question 2 – Greatest Common Divisor

## Objective

Write a program that asks the user for two numbers and outputs their greatest common divisor.

## Constraints

* The two numbers received as input must be non-negative integers or else the user is prompted again for valid inputs.
* The GCD is calculated using the Euclidean algorithm (i.e. the difference between the two values has the same GCD as both values) and then output to the terminal.

## Pseudocode

*integer variables:* numberA, numberB

**do** {

numberA, numberB <- **received input**

} **repeat if** (numberA < 0 **or** numberB < 0)

*integer variables: a* = numberA*, b* = numberB

**if** (a < b) a <-**swap** **values**-> b

**while** (a != b) {

**if** (a > b) a = a – b

**else** b = b - a

}

**output** -> “common denominators: ” + numberA, numberB + “ GCD: ” + a

# Question 3 – Finding Prime Numbers in an Interval

## Objective

Write a C program that accepts a lower and upper bound to find all the prime numbers included within the interval.

## Constraints

* The lower and upper bound are non-negative integers, and the first is greater than the second or else the user is prompted again to correct it.
* The prime numbers (if found, or else a statement to say none were found) are output to terminal.

## Pseudocode

*integer variables:* upperBound, lowerBound

**do** {

lowerBound, upperBound <- **received input**

**if** (lowerBound > upperBound) **output** -> ask to re-enter values properly

} **repeat if** (upperBound < lowerBound)

**output** -> “the interval between” + lowerBound, upperBound

+ “includes these prime numbers:”

**while** (lowerBound <= upperBound) {

*boolean variable:* prime = **false**

if (lowerBound < 2) **skip to** *last*

*factors:* **for** (*integer:* index = 0, **while** index <= (lowerBound) {

**if** (remainder(lowerBound / index) == 0) {

prime = **true**

**break out of** factors

}

**if** (prime == **true**) **output** -> lowerBound

}

*last:* lowerBound = lowerBound + 1

}

# Question 4 – Debugging Sum Expression

## Objective

Debug a C program that is intended to prompt the user for a sequence of numbers and arithmetic signs that calculate to a positive sum. If the sum falls below zero, the program notifies the user and terminates.

## Constraints

* The input received contains characters alternating between digits of 0 to 9 and positive/negative signs (e.g. 1 + 3 + 5 + 9 – 0).
* Upon receiving “=” as input, the program outputs the sum to the terminal and terminates.
* When the real-time calculation of the sum falls below 0, the program displays “Sum fell below zero.” and terminates.

## Errors and Modifications

* Line 10: the **do-while** loop does not include a way for the user to end the program to display their calculated sum. To fix this, an **if** statement is inserted that allows the user to input the character “=”to terminate the program.
* Line 27: the **output** statement assumes that the program only terminates when the sum falls below zero. After introducing the user-mandated termination on *line 10*, an **if** statement checks if the sum is zero or not zero; if the sum is non-zero then it is displayed to the terminal.