ENGG1410: Introductory Programming for Engineers

Lab 2: “Introduction to C Programming”: Complex Calculations and Decision Making, Debugging a C Program

09/17/2024

Part 1

Problem Statement:

obtain a base 10 integer greater then 0 and less then 15 from the user and output each digit of the number in binary (base 2) line by line

Assumptions and Constraints:

- Input is integer
- Number is an integer that is greater than 0 and less than 16
- You should output four digits

How you solved the Problem:

I) Pseudo Code.

Output -> prompt for num between 0-16
Integer num <- Input

If num < 0 or num > 15 exit program

D4 = num%2
Num /= 2

D3 = num%2
Num /= 2

D2 = num%2
Num /= 2

D1 = num%2
Num /= 2

Output - > “The four digits of that number are as follows: (newline)
Most significant digit: ” + D1 + “(newline) Next digit:” + D2 + “(newline) Next digit:” + D3 + “Least significant digit:” + D4

Error Analysis:

errors encountered during the development and how they were resolved:

- **Program output digits in reverse order, this was fixed by revering the order digits were printed in**
- **Program took in digits beyond bounds, this was fixed by adding if statement that checks if input is in bounds**

Part 2

Problem Statement:

Write a program that obtains the x-y components of two vectors and display the angle between them at the origin in radians and degrees using cosine law

Assumptions and Constraints:

- The coordinates are given in the form x comma y
    - Ex 3,4
- The angle between the vectors is at the origin
- Output angle in radians 3 decimal places and degrees 2 decimal places

How you solved the problem:

I) Pseudo Code

Output - > prompt for coordinate 1 in form (x, y)
Double x1, y1 <- input

Output - > prompt for coordinate 2 in form (x, y)
Double x2, y2 <- input

Double dotproduct = x1*x2 + y1*y2

Double mag1 = sqrt (x1^2 + y1^2)
Double mag2 = sqrt(x2^2+y2^2)

CosOfAngle = dotproduct / (mag1*mag2)

AngleRad = acos(CosOfAngle)
AngleDeg = AngleRad*(180/pi)

Output -> AngleRad (three decimal places) and AngleDeg (two decimal places)

Error analysis:

errors encountered during the development and how they were resolved:

- **Degree of angle was incorrect; this was resolved by dividing by pi**

Part 3

Problem Statement:

Obtain the number of minutes from the user and output the number of hours and remaining minutes rounded to the nearest multiple of 15

Assumptions and Constraints:

- Input is positive
- Output is a positive integer and either 0, 15, 30 or 45

How you solved the problem:

I) **Pseudo Code**

```
Output -> prompt for num of minutes
Minutes <- input

Hours = minutes/60
Minutes = minutes%60

if (minutes >= 0 && minutes <= 7)
    rounded minutes = 0;
else if (minutes >= 8 && minutes <= 22)
    rounded minutes = 15
else if (minutes >= 23 && minutes <= 37)
    rounded minutes = 30
else if (minutes >= 38 && minutes <= 52)
    rounded minutes = 45
else {
    rounded minutes = 0
    hours += 1
}

Output -> hours, rounded minutes
```

Error analysis:

errors encountered during the development and how they were resolved:

- Did not increment hours when remainder was 45 or greater, this was fixed by incrementing hours in said condition

Part 4

Problem Statement:

Obtain an integer between -99999 and 99999 from user then output the some of the integers digits

Assumptions and Constraints:

- Input is an integer between – 99999 and 99999

How you solved the problem:

II) **Pseudo Code**

Output -> prompt for num between – 99999 and 99999
Num <- input

Num = abs(num)

Ttd = num/10000
Num%10000

Td = num/1000
Num%1000

Hd = num/100
Num%100

Tens = num/10
Num%10

Ones = num

Output -> "the sum of the digits is: “ + “+” + ttd + “+” + td + “+” + hd + “+” + tens + “+” + ones + “= ” + (ttd+ td + hd + tens + ones)

Error analysis:

errors encountered during the development and how they were resolved:

- Not outputting the addition of digits, this was resolved by simply outputting the digits and addition symbols

Part 4

Problem Statement:

Debug a C program so it achieves its given purpose, which is to obtain an encrypted 4-digit code and output it decrypted as described in the comments

Assumptions and Constraints:

- Input is a 4-digit positive number

Error analysis:

errors encountered during the development and how they were resolved:

- Remove extra spaces from include statement
- Remove space from scanf format specifiers
- Replace d4, d3, d3, d1 with encComb
- Replace encComb in the four lines with encComb equal to encComb divide by something lines with d4, d3, d3, and d1 respectively
- Remove space in /n and %d in last line

Conclusion and self-assessment:

What you learned in the lab.

In this lab I learned to use the modulo symbol to remove/separate digits, use the math.h libary to use functions like acos, sqrt & abs in C and if, else if & else statements to run code in specific scenarios. I would like to be more formular with modulo so I can write code with similar functions faster