University of Glasgow | School of Computing Science

CSC1103 Laboratory/Tutorial 3 4 : Data Type, Arithmetic Operations and Control Structures

1. LINE INTERSECTION

1. Problem definition:

Write a program to take in the parameters of the equation of an original line namely m, c and the coordinate of a given point P, x_o & y_o. Print out the equation of this original line and the equation of another two lines that are parallel and perpendicular to this line respectively given that these two lines passed through the point P.

2. Problem Analysis

To find the equation of a line, there are three steps:

a. Find the slope of the line, basically gradient of the line, m
b. Use the slope to find the y-intercept, c
c. Write out the equation in the form of $y = mx + c$

1. Parallel Line

Two lines are parallel if they have the same slope. This mean the gradient of the parallel line m_2 is the same as the gradient of the original line, $m_1 \Longrightarrow m_2 = m_1$

To obtain the y intercept c_2 , use the point P with coordinates x_o& y_o to insert into the equation of a line $y = mx + c$

$$y_0 = m_2 x_o + c_2 \Longrightarrow c_2 = y_0 - m_2 x_o$$

The data requirement for the parallel line are as follow

Input variable

i. The gradient of the original line, m_1 (float m_1)
ii. The y intercept of the original line, c_1 (float c_1)
iii. The x coordinates of the point P, x_o (float x_o)
iv. The y coordinates of the point P, y_o (float y_o)

Output variable

i. The gradient of the parallel line, m_2 (float m_2)
ii. The y intercept of the parallel line, c_2 (float c_2)

2. Perpendicular Line

Two lines are perpendicular if the slope of each other is negative reciprocal of each other. This mean the gradient of the perpendicular line m_3 is negative inverse of the gradient of the original line, $m_1 m_3 = -1 \Longrightarrow m_3 = -{}^{1}/_{m_1}$

To obtain the y intercept c_3 , use the point P with coordinates x_o& y_o to insert into the equation of a line $y = mx + c$

$$y_0 = m_3 x_o + c_3 \Longrightarrow c_3 = y_0 - m_3 x_o$$

The data requirement for the parallel line are as follow

Input variable

i. The gradient of the original line,m_1 (float m_1)
ii. The y intercept of the original line, c_1 (float c_1)
iii. The x coordinates of the point P, x_o (float x_o)
iv. The y coordinates of the point P, y_o (float y_o)

Output variable

i. The gradient of the parallel line, m_3 (float m_3)
ii. The y intercept of the parallel line, c_3 (float c_3)

3. Algorithm

1. Read gradient of the original line , m_1
2. Read y intercept of the original line, c_1
3. Read x coordinates of the point P, x_o
4. Read y coordinates of the point P, y_o
5. Compute $m_2 = m_1$
6. Compute $c_2 = y_0 - m_2 x_o$
7. Compute $m_3 = -{}^{1}/_{m_1}$
8. Compute $c_3 = y_0 - m_3 x_o$
9. Print the equation of the three lines (original, parallel and perpendicular)

Pseudocode

BEGIN
READ m_1, c_1, x_o, y_o
$m_2 \leftarrow m_1$
$c_2 \leftarrow y_0 - m_2 x_o$
$m_3 \leftarrow -1/m_1$
$c_3 \leftarrow y_0 - m_3 x_o$
PRINT "Equation of the original line is =" $y = m_1 x + c_1$
PRINT "Equation of the parallel line is =" $y = m_2 x + c_2$
PRINT "Equation of the perpendicular line is =" $y = m_3 x + c_3$
PRINT "The x and y coordinates of point P =" x_o, y_o
END

2. c value is :inf
d value is :0.000000e+00
e value is :1.100000e-09

as $c = 9e + 68$ which exceeds the limits of float range which is $3.4e + 34$ and $c = \infty$ as $d = 0$ as b exceed the limit of float which is $3.4e + 34$ and $b = \infty$. As such $d = \frac{a}{b} = \frac{a}{\infty} = 0$
as $e = 1.1e - 9$ remains the same since $d = 0$.

3. JULIA CASEAR CIPHER/SHIFT CIPHER

1. Problem definition:

Write a program to take in the plaintext/ciphertext of the message and the secret key and produce the ciphertext/plaintext based on Caser/shift cipher methodology.

2. Problem Analysis

The cryptography process based on shift cipher is

Plaintext	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
value	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Cipher-text	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
value	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	00	01	02

At encryption end:
The ciphertext C $\quad C = (P + k) mod\ 26$

At decryption end
The plaintext P $\quad P = (C - k) mod\ 26$

Assumption:

a) Input character is ASCII and therefore need to convert from and to ASCII for the plaintext and ciphertext after encryption and decryption respectively.

$$A \rightarrow 65, B \rightarrow 66.., Z \rightarrow 90$$

As such convert from ASCII character to non-ASCII plaintext is subtract by 65 while convert from non-ASCII plaintext to ASCII character is add 65.

b) For decryption, before taking mod, need to ensure $(C - k)$ is positive. If negative, add 26.
c) The algorithm work with Uppercase alphabet only (A to Z) and whitespace.

Input variable

i. The ASCII input character message regardless plaintext or ciphertext, $message$ (char $message$)
ii. The length of the message len (int len)

iii. The choice of cryptography, either encryption or decryption $choice$ (int $choice$)
iv. The secret key k of the cryptography , key (int key)

Process variable

i. The counting variable to count number of characters being processed, $count$ (int $count$)
ii. The plaintext P variable that used in the encryption and decryption algorithm, $plaintext$ (char $plaintext$)
iii. The ciphertext C variable that used in the encryption and decryption algorithm, $ciphertext$ (char $ciphertext$)

Output variable

i. The ASCII character message regardless plaintext or ciphertext, $message$ (char $message$)

3. Algorithm

1. Read the length of the message len
2. Read the choice of cryptography, either encryption or decryption $choice$
3. Read secret key k of the cryptography , key
4. Set $count = 1$
5. While $(count \leq len)$ do the following
   5.1 Read the ASCII message character, $message$
   5.2 Switch $(choice)$
      5.2.1 $choice$ is encryption
         5.2.1.1 If $message$ not equal to white space
            5.2.1.1.1 Print plaintext in ASCII format, $message$
            5.2.1.1.2 Convert plaintext in ASCII format to non-ASCII plaintext character
               $message = message - 65$
            5.2.1.1.3 Calculate the ciphertext
               $ciphertext = (message + key) mod 26$
            5.2.1.1.4 Convert ciphertext to ASCII character format, $message$
               $message = ciphertext + 65$
            5.2.1.1.5 Print ciphertext in ASCII format, $message$
            Else
            5.2.1.1.1 Print plaintext in ASCII format, $message$
            5.2.1.1.2 Print ciphertext in ASCII format, $message$
         5.2.1.2 Break from encryption choice

5.2.2 $choice$ is decryption
5.2.2.1 If $message$ not equal to white space
5.2.2.1.1 Print ciphertext in ASCII format, $message$
5.2.2.1.2 Convert ciphertext in ASCII format to non-ASCII ciphertext character
$message = message - 65$
5.2.2.1.3 If $message \leq key - 1$
5.2.2.1.3.1 Calculate the plaintext
$plaintext = ((message - key) + 26) mod 26$
Else
5.2.2.1.3.1 Calculate the plaintext
$plaintext = (message - key) mod 26$
5.2.2.1.4 Convert plaintext to ASCII character format, $message$
$message = plaintext + 65$
5.2.2.1.5 Print plaintext in ASCII format, $message$
Else
5.2.1.2.1 Print ciphertext in ASCII format, $message$
5.2.1.2.2 Print plaintext text in ASCII format, $message$
5.2.2.2 Break from encryption choice
5.3 Increment the count, $count$

Pseudocode

```
BEGIN
    READ len, choice, key
    count ← 1
    WHILE (count ≤ len)
        READ message
        SWITCH (choice)
            CASE 1
                IF message! = whitespace
                    PRINT "Plaintext:" message
                    message ← message − 65
                    ciphertext ← (message + key)mod26
                    message ← ciphertext + 65
                    PRINT "Ciphertext:" message
                ELSE
                    PRINT "Plaintext:" message
                    PRINT "Ciphertext:" message
                ENDIF
                BREAK
```

```
                    CASE 2
                        IF message! = whitespace
                            PRINT "Ciphertext:" message
                            message ← message − 65
                            IF (message) ≤ key − 1
                              plaintext ← ((message − key) + 26)mod26
                            ELSE
                                plaintext = (message − key)mod26
                            ENDIF
                            message ← plaintext + 65
                            PRINT "Plaintext:" message
                        ELSE
                            PRINT "Ciphertext:" message
                            PRINT "Plaintext:" message
                        ENDIF
                        BREAK
            END SWITCH
            count ← count + 1
    END WHILE
END
```