University of Glasgow | School of Computing Science

CSC1103 Tutorial 8/9 : Pointers and Strings

1. Statistics Algorithm

Problem definition:

Design the pseudocode to take a group of N sample of data and calculate the sample statistics of the data namely the mean, variance and the min and max using function, array and pointer variables. The user will also be prompted to input the size of the sample of data.

Problem Analysis

The statistics function mainly do three statistics calculation using pointer variables.

1. Sample mean

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

   where x_i is the i^{th} observation data and N is the number of observations also referred to sample size and $\bar{x}$ is the sample mean.

2. Sample variance

$$s_x^2 = \frac{1}{N-1}\left(\sum_{i=1}^{N} x_i^2 - N\bar{x}^2\right)$$

Input variable

1. The list of float data type elements, x $(float\ x[\,]\ )$
2. The size of float data type elements, $N(int\ N\ )$

Process variable

1. The summation of all data $x_1 \cdots x_{N,}$ $sumx\ (float\ sumx)$
2. The summation of all square data $x_1^2 \cdots x_N^2$ $,sumxx\ (float\ sumxx)$

Output variable

1. The mean, variance and coefficient of variation, $mean, var, min, max$ (float $mean, var, min, max$)

1. Algorithm

The program is divided into following functions

a) **main ()** :

i. to read the size N and the list of float data type number and store in the data array $x[]$.

ii. call the function **stats ()** through pointer variable to perform statistics calculation namely mean, variance and min and max

iii. print out these statistics result

b) **stats** ():

i. compute the $sumx$ and then obtain the mean, $mean$

ii. compute the $sumxx$ and then obtain the variance,var

iii. call **shellsort**() to sort out the min and max

Algorithm for **main ()**

```
1. Read $N$
2. For $i = 0\ to\ N - 1$ do the following
      2.1 Read $x\ [i]$
3. Call stats ($x\ , N, \&min, \&max, \&mean, \&var$)
4. Print the $mean, var, min, max$
```

Algorithm for **stats** ($x\ , N, \&min, \&max, \&mean, \&var$)

```
1. Set $sumx = 0$
2. Set $sumxx = 0$
3. Call shellsort(x,N)
4. Set the min result at the address indicated by pointer $p\_min$= $x[0]$
5. Set the max result at the address indicated by pointer $p\_max$= $x[N - 1]$
6. For $i = 1\ to\ N - 1$ do the following
    3.1 Set $sumx = sumx + x[i]$
    3.2 Set $sumxx = sumxx + x[i] * x[i]$
7. Calculate $mean$ from $sumx/N$ and store the result at the address indicated
   by pointer $p\_mean$
8. Calculate $var$ from
   $(sumxx - N * (result\ at\ the\ address\ indicated\ by\ pointer\ p\_mean)^2)/(N - 1)$
   and store the result at the address indicated by the pointer $p\_var$
```

Algorithm for **shellsort (**$x\ ,N$)

```
1. Set $jump = N/2$
2. While ($jump \geq 1$) do the following
    2.1 Set $last = N - jump$
    2.2 Set $is\_sorted = FALSE$
    2.3 While ($is\_sorted == FALSE$)
        2.3.1 $is\_sorted = TRUE$
        2.3.2 For ($i = 0\ to\ last - 1$) do the following
            2.3.2.1 if ($x[i] \geq x[i + jump]$)
                2.3.2.1.1 $temp = \ x[i]$
                2.3.2.1.2 $x[i] = x[i + jump]$
                2.3.2.1.3 $x[i + jump] = temp$
                2.3.2.1.4 $is\_sorted = FALSE$
    2.4 $jump = floor(jump/2)$
```

Pseudocode

```
BEGIN
    READ $N$
    FOR $i = 0\ to\ N - 1$ do
        READ $x[i]$
    END FOR
    $stats(x, N, \&min, \&max, \&mean, \&var)$
    PRINT “sample size”, $N$
    PRINT “Mean value”, $mean$
    PRINT “Min”, $min$
    PRINT “Max”, $max$
    PRINT “Var”, $var$
END
```

```
FUNCTION $stats(x, N, p\_min, p\_max, p\_mean, p\_var)$
    $p\_mean$      $refTofloat \rightarrow \&mean$
    $p\_var$       $refTofloat \rightarrow \&var$
    $p\_min$       $refTofloat \rightarrow \&min$
    $p\_max$       $refTofloat \rightarrow \&max$


    $sumx \leftarrow 0$
    $sumxx \leftarrow 0$
    shellsort(x,N)
    $* p\_min \leftarrow x[0]$
    $* p\_max \leftarrow x[N-1]$
    FOR $i = 1\ to\ N$ do
        $sumx \leftarrow sumx + x[i]$
        $sumxx \leftarrow sumxx + x[i] * x[i]$
    END FOR
    $* p\_mean \leftarrow sumx/N$
    $* p\_var \leftarrow (sumxx - N * (* p\_mean ** p\_mean))/(N-1)$
ENDFUNCTION

FUNCTION $shellsort(x, N)$
    $jump \leftarrow N/2$
    WHILE ($jump \geq 1$)
        $last \leftarrow N - jump$
        $is\_sorted \leftarrow FALSE$
        WHILE ($is\_sorted == FALSE$)
            $is\_sorted \leftarrow TRUE$
            FOR $i = 0\ to\ last - 1$ do
                IF ($x[i] \geq= x[i + jump]$)
                    $temp \leftarrow x[i]$
                    $x[i] \leftarrow x[i + jump]$
                    $x[i + jump] \leftarrow temp$
                    $is\_sorted \leftarrow FALSE$
                END IF
            END FOR
        END WHILE
        $jump \leftarrow floor(jump/2)$
    END WHILE
ENDFUNCTION
```

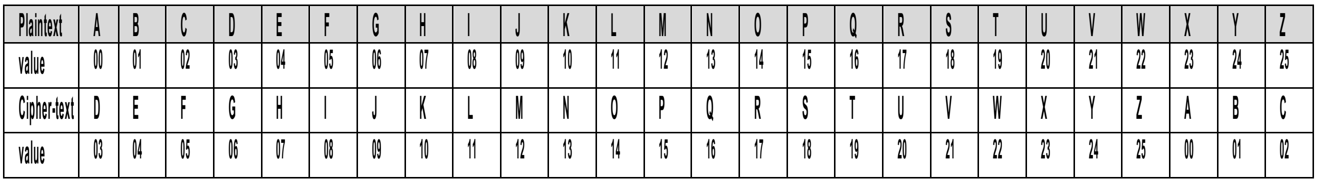
2. JULIA CASEAR CIPHER/SHIFT CIPHER USING STRING POINTER

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 using string pointer.

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 string message regardless plaintext or ciphertext, $message$ (char $message$ [])

ii. The choice of cryptography, either encryption or decryption $choice$ (int $choice$)

iii. The secret key k of the cryptography , key (int key)

Process variable

i. The ASCII input duplicate character string message regardless plaintext or ciphertext, $c\_message$ (char $c\_messsage$ [])

Output variable

i. The ASCII character message regardless plaintext or ciphertext, $c\_message$ (char $c\_message$ [])

Algorithm

The program is divided into following functions

a) **main ()** :

i. to get the ASCII string message size $message$, the choice of cryptography, $choice$ and the secret key, key

ii. call the function **strcpy ()** through string pointer to duplicate the original input string message, $message$ in another string message $c\_message$

iii. call the function **crypto ()** through string pointer to perform cryptography on string message $c\_message$

iv. print out both the original message (plaintext or ciphertext), $message$ and the duplicate message (ciphertext or plaintext), $c\_message$

b) **crypto ():**

i. convert the duplicate message, $c\_message$, to ciphertext or plaintext depending on the choice of cryptography, $choice$ and the secret key, key

Algorithm for **main ()**

1. Get the ASCII string message character, $message$
2. Read the choice of cryptography, either encryption or decryption $choice$
3. Read secret key k of the cryptography , key
4. Copy the string $message$ to another string c_$message$ by calling C library function **strcpy(**$c\_message, message$**)**
5. Call **crypto(**$key, choice, c\_message$**)**
6. If $(choice = encryption)$
    - 6.1 string print plaintext in ASCII format, $message$

```
        6.2     string print ciphertext in ASCII format, $c\_message$
    Else
        6.1     string print ciphertext in ASCII format, $message$
        6.2     string print plaintext in ASCII format, $c\_message$
```

Algorithm for **crypto** $(key, choice, c\_message)$

```
1. While ($value$ stored at the address pointed by pointer $p\_message \neq '\backslash 0'$ ) do the following
    1.1     Switch ($choice$ )
            1.1.1.  $choice$ is encryption
                    1.1.1.1     If $value$ stored at the address pointed by pointer $p\_message \neq$ white space
                                1.1.1.1.1   subtract the $value$ stored at the address pointed by pointer $p\_message$ by 65
                                1.1.1.1.2   add the $value$ stored at the address pointed by pointer $p\_message$ by $key$ and mod by 26
                                1.1.1.1.3   add the $value$ stored at the address pointed by pointer $p\_message$ by 65
                                1.1.1.1.4   break the encryption choice

            1.1.2   $choice$ is decryption
                    1.1.1.2     If $value$ stored at the address pointed by pointer $p\_message \neq$ white space
                                1.1.1.2.1   subtract the $value$ stored at the address pointed by pointer $p\_message$ by 65
                                1.1.1.2.2   If ( $value$ stored at the address pointed by pointer $p\_message \leq key - 1$ )
                                            1.1.1.2.2.1   subtract the $value$ stored at the address pointed by pointer $p\_message$ by $key$ and add 26 and mod 26
                                Else
```

1.1.1.2.2.1 subtract the $value$ stored at the address pointed by pointer $p\_message$ by key and mod 26

1.1.1.2.3 add the $value$ stored at the address pointed by pointer $p\_message$ by 65

1.1.1.2.4 break the decryption choice

1.2 Increment the address pointed by pointer $p\_message$ by 1

Pseudocode

```
BEGIN
    GETS $Message$
    READ $choice$
    READ $key$
    $strcpy(c\_message, message)$
    $crypto(key, choice, c\_message)$
    IF $(choice = encryption)$
        STRING PRINT “plaintext”, $message$
        STRING PRINT “ciphertext”, $c\_message$
    ELSE
        STRING PRINT “ciphertext”, $message$
        STRING PRINT “plaintext”, $c\_message$
    END IF
END

FUNCTION $crypto(key, choice, p\_message)$
    $p\_message$    $refTochar \rightarrow \&c\_message$

    WHILE $(* p\_message \neq '\backslash 0')$
        SWITCH $(choice)$
            CASE 1
                IF $(* p\_message \neq' ')$
                    $* p\_message =* p\_message - 65$
                    $* p\_message = (* p\_message + key) mod26$
                    $* p\_message =* p\_message + 65$
                END IF
                BREAK
                CASE 2
                    IF ($* p\_message \neq ' \ '$)
                        $* p\_message = * p\_message - 65$
                        IF ($* p\_message \leq key - 1$)
                            $* p\_message = ((* p\_message - key) + 26\ ) mod26$
                        ELSE
                            $* p\_message = (* p\_message - key) mod26$
                        END IF
                    END IF
                    BREAK
            END  SWITCH
            $p\_message = p\_message + 1$
    END WHILE
ENDFUNCTION
```

3. String Pointer

Problem definition:

Design the strcmp() function pseudocode to read in two strings and compare the two strings using pointer. If no difference, return the value 0 to the main program. Use the standard library strcmp () function to compare with own strcmp () function

Problem Analysis

Based on ASCII codes table, the strings have been sorted alphabetically since the ASCII code have been chosen so that

A <B<C<…..<Z….<a<b…z

As such, just need to subtract the two ASCII code value according. If the two letters are the same, the value difference will be zero.

Assumption: length of first string is longer than second string

Input variable

1. The first string to be compared, $first\_str$ $(char\ first\_str[\ ])$
2. The second string to be compared, $second\_str$ $(char\ second\_str[\ ])$

Output variable

1. The standard library comparison result, $result$ $(int\ result)$
2. The comparison result using own strcmp function, $result1$ $(int\ result1)$

Algorithm

The program is divided into following functions

a) **main ()** :

i. to read the two strings for comparison, $first\_str$ and $second\_str$
ii. call the library function **strcmp ()** through pointer variable
iii. call the own function **strcmp1 ()** through pointer variable
iv. print out the comparison result from the library function **strcmp ()**, $result$
v. print out the comparison result from the own function **strcmp1 ()**, $result1$

b) **strcmp1** ():

i. subtract each corresponding character ASCII code of the two strings one by one till terminating null character. Store the accumulating difference in $result$
ii. return the comparison result, $result$

Algorithm for **main ()**

```
1. Read $first\_str$ and $second\_str$
2. $result$ =$\boldsymbol{strcmp}(first\_str, second\_str)$
3. $result1$ =$\boldsymbol{strcmp1}(first\_str, second\_str)$
4. Print the $result, result1$
```

Algorithm for **strcmp1** ($first\_str, second\_str$)

```
1. Set $result = 0$
2. While $(value\ in\ address\ indicated\ by\ first\_str\ \neq\ '/0\ ')$ do the following
    2.1 If $(value\ in\ address\ indicated\ by\ first\_str\ \neq\ v\ alue\ in\ address\ indicated\ by\ second\_str)$
        2.1.1 Set $result = result + value\ in\ address\ indicated\ by\ first\_str\ -\ value\ in\ address\ indicated\ by\ second\_str$
    2.2 Increment $address\ indicated\ by\ first\_str$
    2.3 Increment $address\ indicated\ by\ second\_str$
3. Return $result$
```

Psuedocode

```
BEGIN
    READ $first\_str$ and $second\_str$
    $result \leftarrow strcmp(first\_str, second\_str)$
    $result1 \leftarrow strcmp1(first\_str, second\_str)$
    PRINT “result”, $result$
    PRINT “result1”, $result1$
END

FUNCTION $strcmp1(first\_str, second\_str)$
     $first\_str$        $refTochar \rightarrow \&first\_str$
     $second\_str$       $refTochar \rightarrow \&second\_str$

     WHILE $*(first\_str)$
          IF $(*(first\_str) != *(second\_str))$
               $result \leftarrow result + (*first\_str -* second\_str)$
          END IF
          $first\_str = first\_str + 1$
          $second\_str = second\_str + 1$
     END WHILE
     return $result1$
ENDFUNCTION
```