Lab 13: Implementation of String Algorithms

*String Operation Methods*

**Strings**

Strings are defined as an array of characters. The difference between a character array and a string is the string is terminated with a special character ‘\0’.

Declaration of strings: Declaring a string is as simple as declaring a one dimensional array. Below is the basic syntax for declaring a string

**String Operations**

**Length**

The number of characters in a string is called its length. The function length(s) returns the length of string s.

**Concatenation**

Concatenation combines the characters of two strings. For example, concatenation of string s1 with string s2 results in a string containing characters of s1 followed by those of s2. Note that string terminator of s1 is replaced by the first character of s2, but string terminator of s2 is retained.

**Substring**

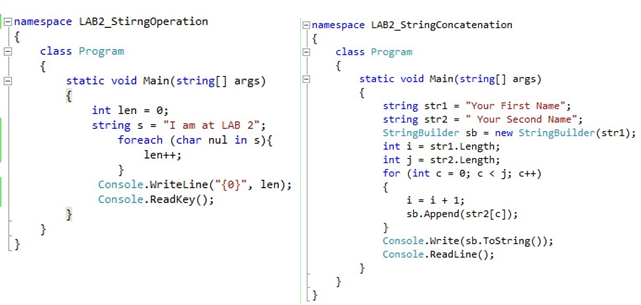
Accessing a substring from a given string requires three pieces of information: a) the name of the string itself, b) the position of the first character of the substring in the given string and c) the length of the substring or the position of the last character of the substring. The function substring (s, ip, len) denotes the substring of a string s beginning in a position ip and having a length len.

**Index**

The function index(T, P) is used to find the position where a string pattern P first appears in a given string text T. This is called pattern matching.

**Example**

**String Length String Concatenation**



**ALGORITHMS:**

**Algorithm A1: length(s)**

STEP 1. [Initialize String] S = “Data Structure” , LEN=0

STEP 2. [Repeat step 3 till the string terminator is encountered].

STEP 3. LEN = LEN +1

STEP 4. Write LEN

STEP 5. Exit

**Algorithm A2: concatenate (s1, s2)**

STEP 1. [Initialize] I = strlen(s1)

STEP 2. [Initialize] J = strlen(s2)

STEP 3. [Initialize] COUNT = 0;

STEP 4. Repeat steps 5 to 7 while COUNT < J

STEP 5. I = I + 1

STEP 6. S1[I] = S2[COUNT]

STEP 7. COUNT= COUNT + 1

STEP 8. Write S1

STEP 9. Exit

**Algorithm A3: substring (s, ip, len)**

STEP 1. [Initialize] Set I = IP, COUNT = 0 and DEST[]

STEP 2. Repeat steps 3, 4 and 5 while COUNT < LEN

STEP 3. DEST[COUNT] = S[I]

STEP 4. COUNT= COUNT+1

STEP 5. I = I + 1

STEP 6. Write DEST[]

STEP 7. Exit

**Algorithm A4: insert (T, P, IP)**

1. [Initialize IP (in position), the position where you insertion starts] IP = some value

2. TEMP1 = substring (T, 0, IP-1) [use strcpy (course\_String, dest\_String )]

3. TEMP2 = substring (T, IP, length(T) – 1) [use strcpy (course\_String, dest\_String )]

4. concatenate (TEMP1, P)

5. concatenate (TEMP1, TEMP2)

6. T = TEMP1

7. Write T

8. Exit

Note :

For Substring : Use function {**substring (string, ip, len)**}

For Concatenate : Use { **concatenate (s1, s2)** }

**Algorithm A5: del (T, ip, Len)**

1.[Initialize] **T** as String **LEN** as length of deleted string

2.TEMP1 = substring (T, 0, IP-1)

3.TEMP2 = substring (T, IP+LEN, Length(T) - 1)

4.concatenate (TEMP1, TEMP2);

5.Write TEMP1;

**Algorithm A6:**

**NAÏVE STRING-MATCHER (T, P)**

STEP 1 [Initialize] n == T.Length

STEP 2. [Initialize] m == P.Length

STEP 3. Repeat for s = 0 to n-m

STEP 4. If P[1…….m] == T[s+1…s+m]

STEP 5. Then Print “Pattern occurs with shift” s

STEP 6. Exit

**Algorithm A6:**

**RABIN -KARP-MATCHER (T, P, d, q)**

STEP 1 [Initialize] n == T.Length

STEP 2. [Initialize] m == P.Length

STEP 3. [Initialize] h == dm-1 mod q

STEP 4 [Initialize] p == 0

STEP 5. [Initialize] q == 0

STEP 6 Repeat for i = 0 to m

STEP 7. p= (dp + P[i])mod q

STEP 8. t= (dt + T[i])mod q

STEP 9. Repeat for s = 0 to n-m

STEP 10. If p=t

STEP 10. If P[1…….m] == T[s+1…s+m]

STEP 11. Then Print “Pattern occurs with shift” s

STEP 12. If P < n-m

STEP 13. t = (d(t- T[s+1]h)) + T[s+m+1] mod q

STEP 14. Exit