UNIT – 2
STRINGS

OUTLINE

- String and their representation
- Reading and Writing Strings
- String Operations
  - String Length Operation
  - String Copy Operation
  - Concatenation Operation
  - Substring Operation
  - String Comparison
  - String (text) Insertion
  - String (text) Deletion
  - String Appending
  - Reversing a String
  - Converting character of string into upper case and lower case

String and their representation

What is string?

- A string is an array of characters.
- It is also define as number of characters written in double quotation mark.

  Example: “Hi hello”
- Each string is terminated by the NULL character, which indicates end of string.
- NULL character is denoted by the escape sequence ’\0’.
- The NULL character is automatically appended to the end of the characters in a string constant when they are stored.

String and their representation

Character set of string

1. Alphabets:
   - Lower case: a to z
   - Upper case: A to Z
2. Numeric: 0 to 9
3. Special character: +, -, *, /, (), {}, [], ””, Null etc.

String and their representation

Common operation performed on string

- Reading and writing strings
- Count length of string
- Concatenation of string
- Copying one string to another
- Comparing string for equality
- Extracting some portion of a string (Sub-string)
- Text editing
- Pattern matching

Reading and Writing Strings

- The string is declared as follows,

  char name[15];

- Where name is a string which can store maximum 14 characters and one byte is needed to store the (‘\0’) NULL character at end of string.
- String can be initializes as below,

  char name[4] = “xyz”;

- The same can also be declared by

  char name[4] = {‘x’,’y’,’z’};

- In above declaration, size is not compulsory and if not given compiler automatically computes and put it.

Reading and Writing Strings

Reading a string

- There are two functions used for reading a string: scanf() and gets().
- The major difference between them is scanf() is useful to read only words while gets() can be used to read string with space and tab also.

  scanf("%s", name);
- When we enter following input from keyboard

  Hi hello
- It accepts only "Hi" to string variable name.
- The second word "hello" remains into buffer and subsequent scanf() may be read.
- The gets() function used as,

  gets(name);

Reading and Writing Strings

Writing a string

- For writing a string use printf() statement with %s format specifier for control string.
- For example, **printf(“%s”,name);**
- This prints the string stored in the string variable name.

String Operations

1. **String Length Operation:**

- The number of characters in a string is called its length.

  Example: The string “Computer” has length 8.

Algorithm: LEN (STR)

Step 1: [Initialization]
I ← 0

Step 2: [Read string]
Read (STR)

Step 3: repeat While (STR[I] <>NULL)
I ← I+1

Step 4: [Display length]
Write (‘Length of string: I’)

Step 5: [Finished]
Exit.

String Operations

2. String Copy Operation:

- Suppose we have two string s1 and s2, then if we want to copy string s2 into s1 we required copy function.

  S1 = Blank or NULL string

  S2 = “Computer”

- XSTRCOPY(s1, s2) will copy string s2 into s1.

Algorithm: XSTRCOPY (STR 1, STR 2)

Step 1: [Initialization]

STR1 ← NULL

i ← 0

Step 2: [Read the string STR2]

Read (STR 2)

String Operations

Step 3: [copy operation performed]

repeat While (i<> LEN (STR 2))

(a) STR1 [i] ← STR2 [i]

(b) i ← i+1

Step 4: [Print the string after copy operation performed]

write (‘STR1’)

Step 5: [Finished]

Exit.

String Operations

3. Concatenation Operation:

- Suppose we have two string s1 and s2 then concatenation of s1 and s2 are combining two strings together in one string.
- We required a concatenation function such as XSTRCAT(s1, s2).

Example: if s1 = “hi” and s2 = “hello”

Then concatenation of s1 and s2 = “hihello”.

Algorithm: XSTRCAT(STR1, STR2, STR3)

Step 1: [Initialization]

STR3 ← NULL

i ← 0

j ← 0

k ← 0

String Operations

Step 2: [Copy STR1 string into STR3]

repeat while(i<> LEN (STR1))

(a) STR3 [k] ← STR1 [i]

(b) i ← i+1

(c) k ← k+1

Step 3: [Append STR2 string into STR3]

repeat while (j <> LEN(STR2))

(a) STR3 [k] ← STR2 [j]

(b) j ← j+1

(c) k ← k+1

Step 4: [Print the string after concatenation Operation performed]

write (“STR3”)

Step 5 : [Finished]

Exit

String Operations

4. Substring Operation:

- In a given string, if we want to find a specific string or character, we required a substring function.
- For finding a substring from a string we must specify a starting character position and number of character of a string.

  Example: String are **s1** = “Computer”,

  cursor = starting position of sub string

  num = number of character of sub string

- A sub string function as XSUBSTR(s1, cursor, num).
- Suppose we have cursor = 4 and num = 3.

  XSUBSTR(s1, 4, 3) = “put”.

String Operations

Algorithm: XSUBSTR (STR, cursor, num)

Step 1: [initialization]

i ← 0

subject ← NULL

Step 2: repeat while (num <> 0)

(i) subject [i] ← STR [cursor]

(ii) cursor ← cursor +1

(iii) i ← i+1

(iv) num ← num-1

Step 3: [Print original string and sub string]

(i)write (“subject”)

Step 4: [Finished]

Exit.

String Operations

5. String Comparison:

- In string comparison operation, if two strings s1 and s2 are given.
- If we compare s1 and s2 character by character.
- If both are same then print “Both string are equal” and if both are different then print “Both string are not equal”.
- Example: If **s1 = “Hello”** and **s2 = “Hello”**

  then XSTRCMP(s1, s2) = true.

  If **s1 = “Hi”** and **s2 = “Hello”**

  then XSTRCMP(s1, s2) = false.

String Operations

Algorithm: XSTRCMP(STR1, STR2)

Step 1: [Initialization]
i ← 0
Flag ← 0

Step 2: [Read two string]
Read (STR1)
Read (STR2)

Step 3: [Finding the length of two string]
i1 ← LEN (STR1)
i2 ← LEN (STR2)

Step 4: [Check the length of both string]
if(i1 <> i2)
then
Write(“Both string are different”)
Exit

String Operations

Step 5: [Compare two string character by character]

Repeat while (i <= i1)

if(STR1 [i]<> STR2[i])

then

Write (“Both string are different”)

Flag ← 1

Exit

i ← i+1

Step 6: if(Flag = 0)

then

Write(“ Both string are same”)

Step 7: [Finished]

Exit

String Operations

6. String (text) Insertion:

- If we want to insert a sub string (word) at some location (position). We required insert function.

  INSERTION(string, position, sub string)

- Example: Suppose we have a string, **string = "My name XYZ"** and we have to insert a sub string **"is"** at position number 8.

  Using insertion function, INSERTION (string, 8, "is")

  Output: My name is xyz.

String Operations

Algorithm: Insertion (text, position, string)

Step 1 : [Initialization]

i ← 0

j ← 0

Temp <- null

Step 2 : [To reach at position for insert]

repeat While(i <> position)

(i) temp[i] ← text[i]

(ii) i ← i+1

Step 3 : [Insert a string at position]

repeat While(j <> LEN(string))

(a) temp[i] ← string[j]

(b) i ← i+1

(c) j ← j+1

String Operations

Step 4 : [Insert rest of character after Inserting String]

repeat While (text[position] <> NULL)

(i) temp[i] ← text[position]

(ii) i ← i +1

(iii) position ← position+1

Step 5: [Print the string after insertion]

Write (“temp”)

Step 6 : [finished]

Exit.

String Operations

7. String (text) Deletion:

- If we want to delete a sub string from a string at specific position with number of characters. We required a deletion function DELETE(string, position, length).
- Example: Suppose we have a string,

string = “Computer science and engg.”

position = 13 (Starting position of word)

length = 3 (Number of characters to delete)

DELETE(string, position, length)

Output: Computer scie and engg.

String Operations

Algorithm : Deletion (text, position, length)

Step 1: [Initialization]

i ← 0

j ← length

k ← 0

temp <- Null

Step 2: [To reach at position for delete a string]

repeat While (i<> position)

(i) temp [k] ← text [i]

(ii) i ← i + 1

(iii) k ← k+1

Step 3: [To find new position after deletion a string]

i ← (position + length)

String Operations

Step 4: [Insert rest of character]

repeat While (text[i] <> NULL)

(i)temp[k] ← text[i]

(ii)k ← k + 1

(iii) i ← i + 1

Step 5: [Print the string after deletion]

Write (“temp”)

Step 6: [finished]

Exit

String Operations

8. String Appending:

- Appending a string in existing string means we add new string at end of existing string.
- For appending we assume following variable,

text = Original string

string = word or new string which is append in original string

APPEND (text, string)

Algorithm: APPEND (text, string)

Step 1: [Initialization]

i ← 0

j ← 0

String Operations

Step 2: [to reach at end of text]

Repeat While (i <> LEN(text))

(i) i ← i + 1

Step 3: [appending a string]

Repeat While(j <> LEN(string))

(i) text[I] ← string[j]

(ii) i ← i + 1

(iii) j ← j + 1

Step 4: [Print the string after appending]

Write (“text”)

Step 5: [finished]

Exit.

String Operations

9. Reversing a String:

- Reversing a string means existing string print from last character to first character.
- For reversing we assume following variable,

  string1 = Original string

  string2 = Used for storing reverse string

  REVERSE (string1, string2)

Algorithm: REVERSE(string1,string2)

Step 1:[initialization]

i ← 0

j ← 0

String Operations

Step 2:[to reach at end of original string]

Repeat While(i<>LEN(string 1))

i ← i+1

Step 3: [store reverse string from original string]

Repeat While(i >= 0)

string2[j] ← string1[i]

j ← j+1

i ← i-1

Step 4: [print the reverse]

Write(“string2”)

Step 5: [finished]

Exit.

String Operations

Converting character of string into upper case and lower case:

- Converting a string from upper case to lower case by adding 32 in existing string and lower case to upper case by subtracting 32 from existing string.
- For upper case and lower case we assume following variable,

  string1: Original string

  string2: to store upper case string

  UPPER (string1, string2)

  LOWER (string1, string2)

String Operations

Algorithm: UPPER_CASE(string1,string2)

Step 1: [initialization]

i ← 0

Step 2: [to convert lower to upper case]

Repeat While(i<>LEN(string1))

if(string1[i]>='a' AND string1[i]<='z') then

string2[i] ← string1[i]-32

else

string2[i] ← string1[i]

i ← i+1

Step 3: [print the upper case string]

Write("string2")

Step 4: [finished]

Exit

String Operations

Algorithm: LOWER_CASE(string 1,string 2)

Step 1:[initialization]

i ← 0

Step 2:[to convert upper to lower case]

Repeat While(i<>LEN(string1)

if(string1[i]>='A' AND string1[i]<='Z') then

string2[i] ← string1[i]+32

else

string2[i] ← string1[i]

i ← i+1

Step 3:[print the lower case string]

Write("string2")

Step 4:[finished]

Exit.

Thank You