Strings

2.1 String representation: Reading and Writing Strings

String: It is sequence of collection of characters.

- In terms of c language string is an array of characters.
- The string terminated with NULL or "\0" is known as null terminated string.
- For example: To store "HELLO" in array the array must be declared as chara[5].
- String "HELLO" is stored as shown in fig
- For example:



• String Character set:

• Lower case: a to z

• Upper case: A to Z

• Number: 0 to 9

• Special Characters: + - * % / () [] { } \$ # &, . ? @ Etc.

Declaration of String

Char stringName[Size];Char str[10];

- Initialization of String
 - ✓ Char str[10]="Hello";
 - \checkmark Char str[10]={,,H","e","l","l","o"};
 - ✓ Char str[10];

```
Scanf("%s",str);
```

• getchar()

It is used to get a single character from the terminal.

Example: char str; str=getchar();

gets()

The function read line of, containing whitespace until the new line character.

Example: char str[10];gets(str);

printf("%s",str);

Explain putchar() and puts().

• putchar()

It is used to put a single character on the terminal.

Example:

putchar(str);

• puts()

The function print line of, containing whitespace until the new linecharacter.

Example: char str[10]="Hello";puts(str);

***** 2.2 String operations

- 1. String Length: This function finds the length of the string.
- 2. Uppercase: Returns string characters in uppercase.
- 3. Lowercase: Returns string characters in lowercase.
- 4. String Concate: This function concate two strings and store it in to the another string.
- 5. String Append: It is used to append a given string str1 to another specified string
- 6. Reverse string: This operation is used to reverse the given string.
- 7. String Copy: This function copy one string in to another string.
- 8. String Compare: This function compare two strings.
- 9. Insertion: Is used to insert characters in string at specified
- 10. Substring: This function finds one string into another string.
- 11. Deletion: Is used to delete characters in string at specified position.

1. Write an algorithm to find length of string.

• This algorithm counts the length of the given string.

Algorithm:

STR_LEN(str)

str : given string.i : index pointer to str

Step: 1 [Initialization]

i**←**0

Step: 2 [Read String]

Read(str)

Step: 3[Process until end of the string]

Repeat while (str[i]! = NULL)

 $i\leftarrow i+1$

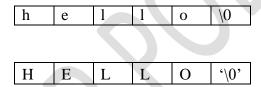
Step: 4 [Print length]

Write("Length of string: i")

Step: 5 [Finished]

Exit.

2. Write an algorithm to convert characters of string into uppercase.



- We have two string, str1 and str2.
- STR1 is in lowercase. To convert STR1 in to uppercase, this algorithm is used.

Algorithm:

STR_UPPER (str1,str2)

str1: given string1.str2: converted string2.i: index pointer to str1j: index pointer to str2

Step: 1 [Initialization]

i**←**0

Step: 2 [Read String]

Read (str1)

Step: 3 [Convert lowercase to Uppercase]

Repeat while (str1[i]!='\0')

$$if(str1[i] >= 'a' \text{ and } str1[i] <= 'z')$$
 $str2[j] \leftarrow str1[i] -32$
else
 $str2[j] \leftarrow str1[i]$
 $i \leftarrow i + 1$
 $j \leftarrow j + 1$

Step: 4 [Print the Uppercase String]

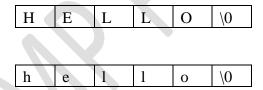
$$str2[j] \leftarrow '\0'$$

Write (str2)

Step: 5 [Finished]

Exit.

 ${\bf 3.}\ Write\ an\ algorithm\ to\ convert\ characters\ of\ string\ into\ Lowercase.$



- We have two string, str1 and str2.
- STR1 is in uppercase. To convert STR1 in to lowercase, this algorithm is used.

Algorithm:

STR_LOWER (str1, str2)

str1: given string1.str2: converted string2.i: index pointer to str1j: index pointer to str2

```
Step: 1 [Initialization]
                  i←0
                  j←0
Step: 2 [Read String]
                  Read (str1)
Step: 3 [Convert Uppercase to Lowercase]
         Repeat while str1[i]! = '\0')
        If (str1[i] \ge 'A' \text{ and } str1[i] \le 'Z')
                  str2[j] \leftarrow str1[i] +32
        else
                 str2[i] \leftarrow str1[i]
        i\leftarrow i+1
        j← j+1
 Step: 4 [Print the Lowercase String]
          str2 [j] ←'\0'
          Write (str2)
```

Step: 5 [Finished]

Exit.

4. Write an algorithm for string concatenation.

- We have two string, str1 and str2.
- Concatenation operation, combine two string str1 and str2 in one string.
- Example: str1="Hello" and str2="World" then, str3=str1+ str2 means str3="Hello World"

Н	e		1	1	()	' \0'				
W	0	r	,	1	d		' \0'				
	Н	e	1	1	О	W	О	r	1	d	' \0'

Algorithm: STR_CONCATE(str1,str2,str3)

str1: given string1.

str2 : given string2.

str3: Concatenated String3.

i : index pointer to str1j : index pointer to str2

k : index pointer to str3

Step: 1 [Initialization]

i**←**0

i**←**0

k**←**0

str3 ←Null

Step: 2 [Read String]

Read(str1)

Read(str2)

Step: 3 [Copy String1 into String3]

Repeat while (str1[i] != '\0')

 $Str3[k] \leftarrow str1[i]$

i**←**i + 1

 $k \leftarrow k + 1$

Step: 4 [Copy String2 into String3]

Repeat while $(str2[j] != '\0')$

 $str3[k] \leftarrow str2[j]$

j**←**j + 1

 $k \leftarrow k + 1$

Step: 5 [Print the string after Concatenation operation performed]

 $str3[k] \leftarrow '\0'$

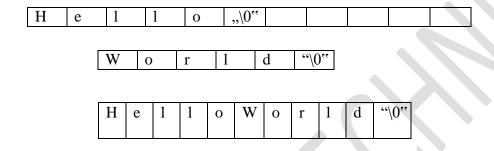
write(str3)

Step: 6 [Finished]

Exit.

5. Write an algorithm for string Append.

- It add new string at end of existing string.
- We have two string, str1 and str2.
- Append operations, combine two string str1 and str2 and store in str1.
- Example: str1="Hello" and str2="World" then, str1=str1+ str2 means str1="Hello World"



Algorithm:

STR APPEND(str1,str2)

str1: given string1.str2: given string2.i: index pointer to str1j: index pointer to str2

Step: 1 [Initialization]

i**←**0

i**←**0

Step: 2 [Read String]

Read(str1)

Read(str2)

Step: 3 [Reach at end of string1]

Repeat while $(str1[i] != '\0')$

 $i\leftarrow i+1$

Step: 4 [Append String]

Repeat while $(str2[j] != '\0')$

 $str1[i] \leftarrow str2[j]$

$$i\leftarrow i+1$$

$$j \leftarrow j + 1$$

Step: 5 [Print the string after Append operation performed]

$$str1[i] \leftarrow '\0'$$

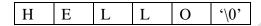
write(str1)

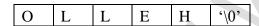
Step: 6 [Finished]

Exit.

6. Write an algorithm for string Reverse.

- We have two string, str1 and str2.
- If we have to reverse string str1 and store into str2 then we required reverse function.





Algorithm: STR_REVERSE (str1, str2)

str1: given string1.

str2: reverse string2.

i: index pointer to str1

j: index pointer to str2

Step: 1 [Initialization]

i**←**0

i**←**0

str2←Null

Step: 2 [Read String]

Read(str1)

Step: 3[To reach at end of original string]

Repeat while (str1[i] != '\0')

 $i \leftarrow i + 1$

Step: 4 [Store Reverse string from Original String]

Repeat while (i>=0)

$$str2[j] \leftarrow str1[i]$$

 $j \leftarrow j + 1$
 $i \leftarrow i - 1$

Step: 5 [Print the Reverse String]

 $str2[j] \leftarrow NULL$

Write (str2)

Step: 6 [Finished]

Exit.

7. Write an algorithm String Copy:

We have two string, str1 and str2.

If we have to copy string str2 into str1 then we required copy function.

Н	Е	L	L	0	'\0'		
	1	1					

Algorithm:

STR_COPY(str1,str2)

str2 : given string2. str1 : new string1.

i : index pointer to str1j : index pointer to str2

Step: 1 [Initialization]

i**←**0

j**←**0

str1 ←Null

Step: 2 [Read String]

Read(str2)

Step: 3[Copy Operation Performed]

Repeat while (str2[j] != '\0')

$$str1[i] \leftarrow str2[j]$$

 $i \leftarrow i + 1$
 $j \leftarrow j + 1$

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

$$str1[i] \leftarrow '\0'$$

write(str1)

Step: 5 [Finished]
Exit.

8. Write an algorithm for String Comparison.

- We have two string, str1 and str2.
- Compare str1 and str2 character by character.
- If both are same then give result "Equal".
- If both are different then give result"Not equal".
- Example: str1="computer" and str2="computer", then both strings are equal.
- If str1="computer" and str2="Comp" then strings are not equal.

Algorithm: STR_COMPARE(str1,str2)

str1: given string1.
str2: given string2.
i: index pointer to str1
j: index pointer to str2
L1: length of string1
L2: length of string2

Step: 1 [Initialization]

i**←**0

j**←**0

Step: 2 [Read two Strings]

Read (str1)

Read (str2)

Step: 3[Find Length of two strings]

```
L1 \leftarrow strlen(str1)
                         L2 \leftarrow strlen(str2)
Step: 4 [Check the length of both strings]
                 If (L1! = L2)
                        Write ("Both strings are different")
                        Exit
Step: 5 [Compare two string character by character]
         Repeat while (str1 [i] != '\0')
               if(str1[i]!=str2[j])
                        Write ("Both Strings are different")
                        Exit
               else
                        i← i+1
                       j \leftarrow j+1
Step: 6 [Return Equal string]
                 Write ("Both Strings are Equal")
Step: 7 [Finished]
```

9. Write an algorithm for string Insertion.

Exit.

```
STR_Insertion (str1, str2, str3)

str1: given string1.

str2: string2 to insert.

str3: new string

i: index pointer to str1

j: index pointer to str2

k: index pointer to str3

Step: 1 [Initialization]

i←0

j←0

k←0

Step: 2 [Read String]

Read (str1)

Read (str2)

Read (position)
```

Step: 3[To reach at position for insert]

Repeat while (i! = position -1)

$$str3[k] \leftarrow str1[i]$$

 $i \leftarrow i + 1$

Repeat while $(str2 [j]! = '\0')$

$$str3 [k] \leftarrow str2[j]$$

$$j \leftarrow j + 1$$

$$k \leftarrow k + 1$$

Repeat while (str1[i] != "\0")

$$str3[k] \leftarrow str1[i]$$

$$i \leftarrow i + 1$$

$$k \leftarrow k + 1$$

Step: 4 [Print the string]

Str3 [k] \leftarrow "\0" write (str3)

Step: 5 [Finished]

Exit.

10. Write an algorithm for Substring.

Н	Е	L	L	O	'\0'
L	L	0	' \0'		

- We have two string, str1 and str2.
- Read string 1 from 2nd position and print total 3 characters.

str1: given string1.

str2: new string

num: Position for substring

total: number of character to read

• Algorithm:

STR_SUBSTRING (str1, str2)

Step: 1 [Initialization]

```
i←0
          j←0
             str2←NULL
Step: 2 [Read String1, print position and total no of characters]
             Read (str1)
            Read (num)
            Read (Total)
Step: 3 [Process for Substring]
            num = num-1 While (Total> 0)
                    Str2[j] \leftarrow str1[num]
                    num \leftarrow num + 1
                    j \leftarrow j + 1
                    total ←total - 1
Step: 4 [Print Substring]
            Str2 [j] \leftarrow '\0'
            Write (str2)
Step: 5 [Finished]
            Exit.
```

11. Write an algorithm for string Deletion.

STR_Deletion (str1, str2) Step: 1 [Initialization] i**←**0 j**←**0 str2←NULL Step: 2 [Read String] Read (str1)

Read (Total)

Read (position)

```
Step: 3[To reach at position for deletion]
```

Repeat while (i! = position -1)

Str2 [j]
$$\leftarrow$$
 str1 [i]
 $i \leftarrow$ i + 1
 $j \leftarrow$ j + 1

Step: 4 [Reset new value of i after delete character] $i\leftarrow$ + total

Step: 5

While str1[i]! = NULL

$$Str2 [j] \leftarrow str1 [i]$$

$$i \leftarrow i + 1$$

$$j \leftarrow j + 1$$

Step: 6 [Finished]

Exit.