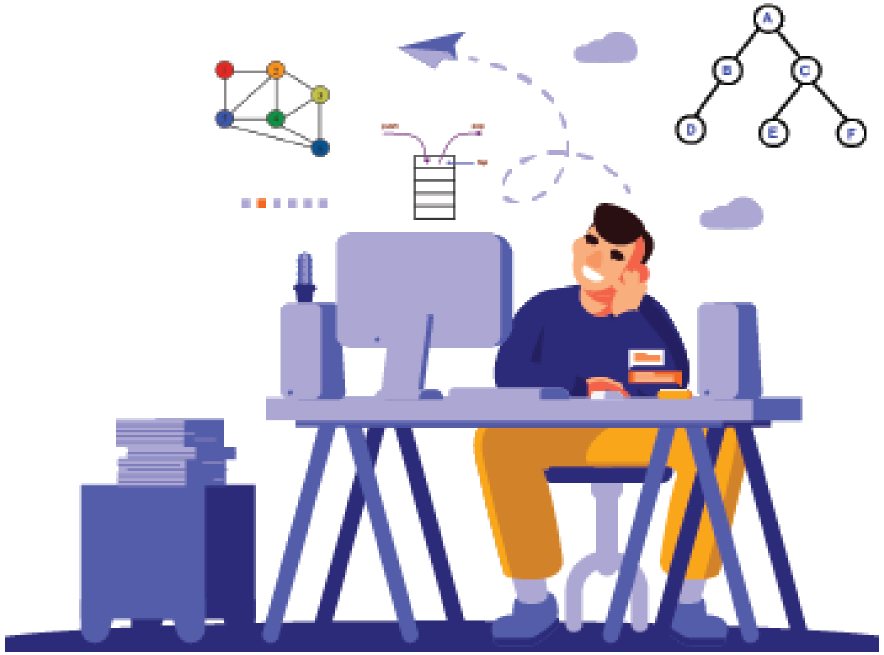
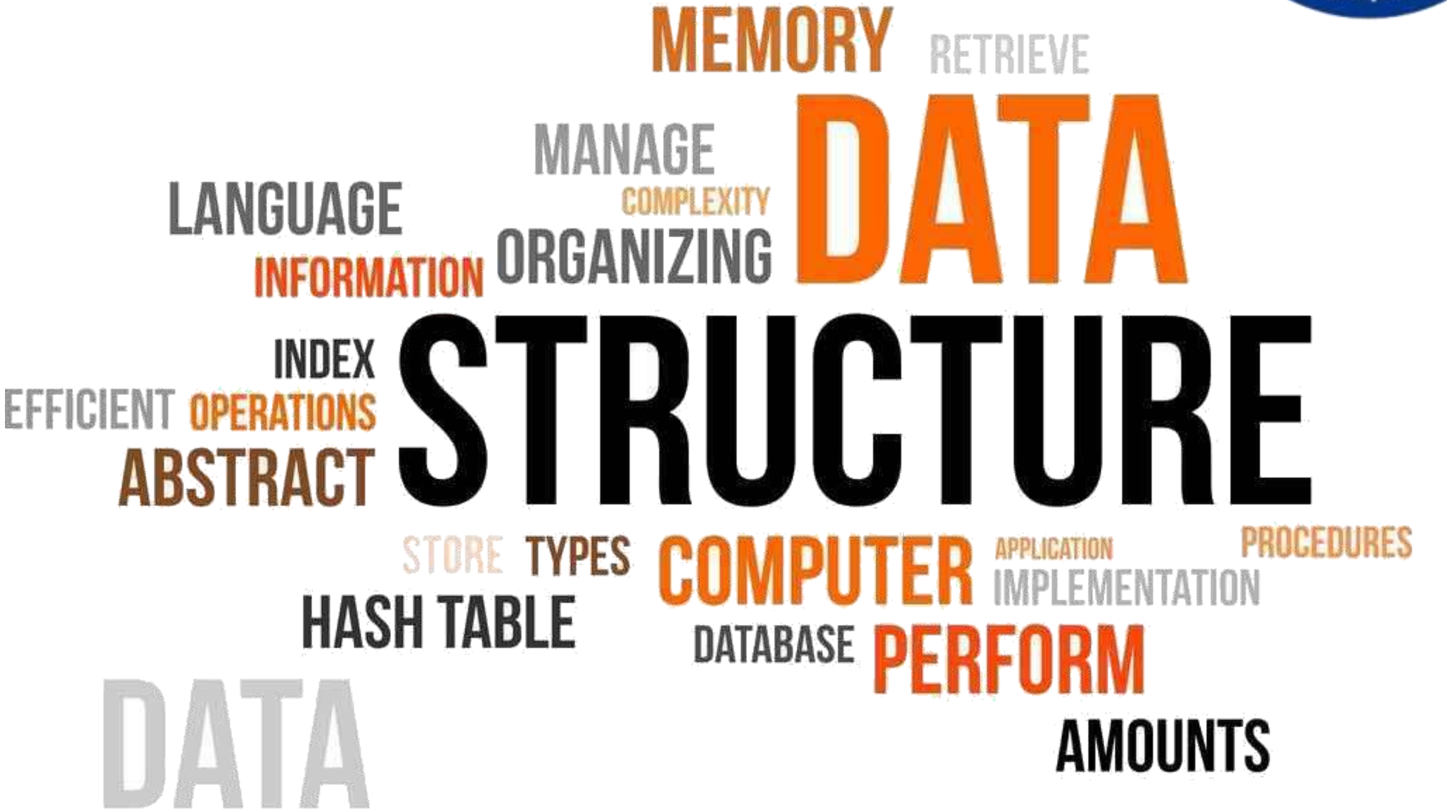


Data Structures

Course code: IT623

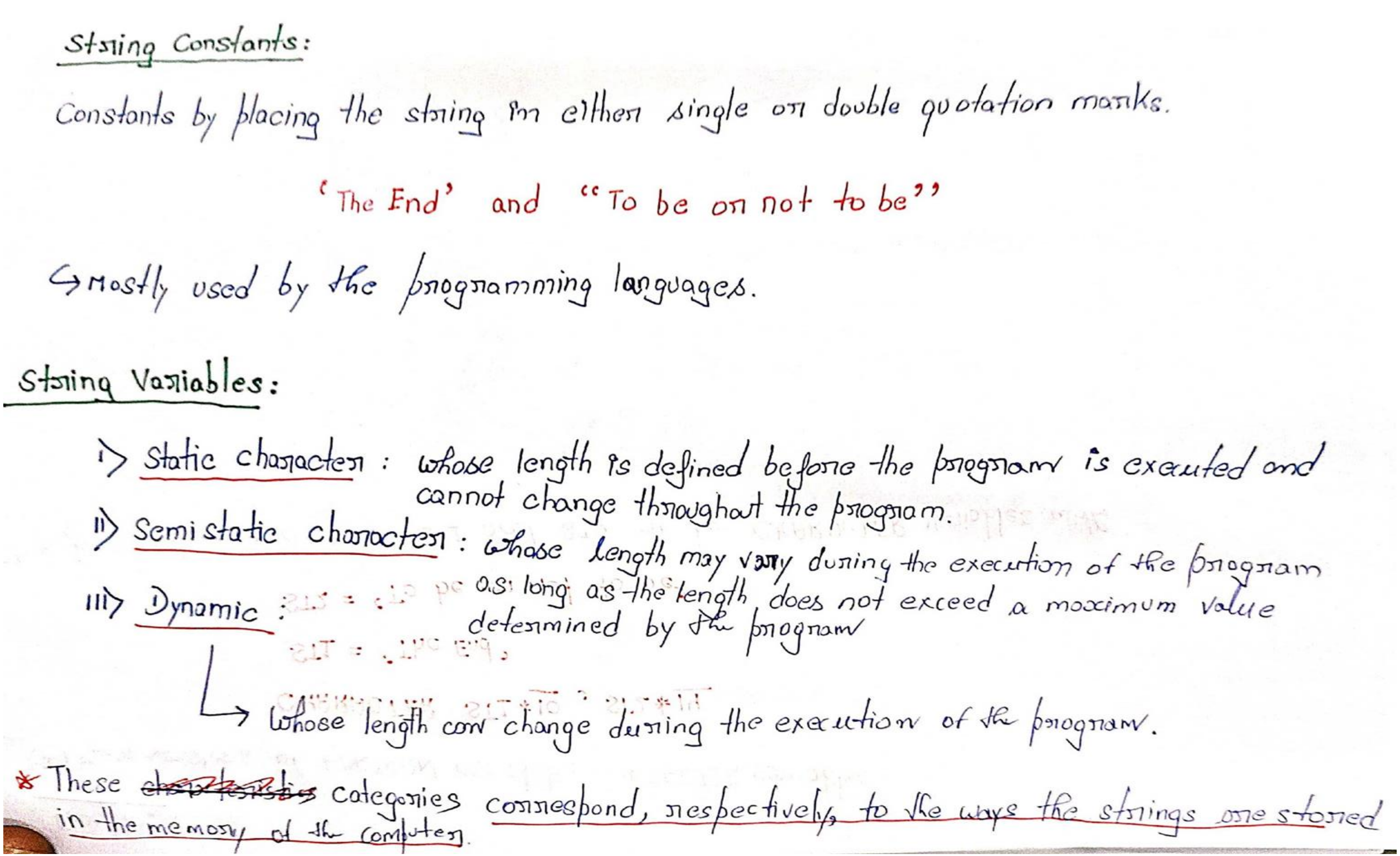


Dr. Rahul Mishra
Assistant Professor
DA-IICT, Gandhinagar

Disclaimer: Images are sourced from the Internet and may be copyright protected.

Lectures 10

String Processing…



String Constants:

Constants by placing the string in either single or double quotation marks.

'The End' and "To be or not to be"

↳ mostly used by the programming languages.

String Variables:

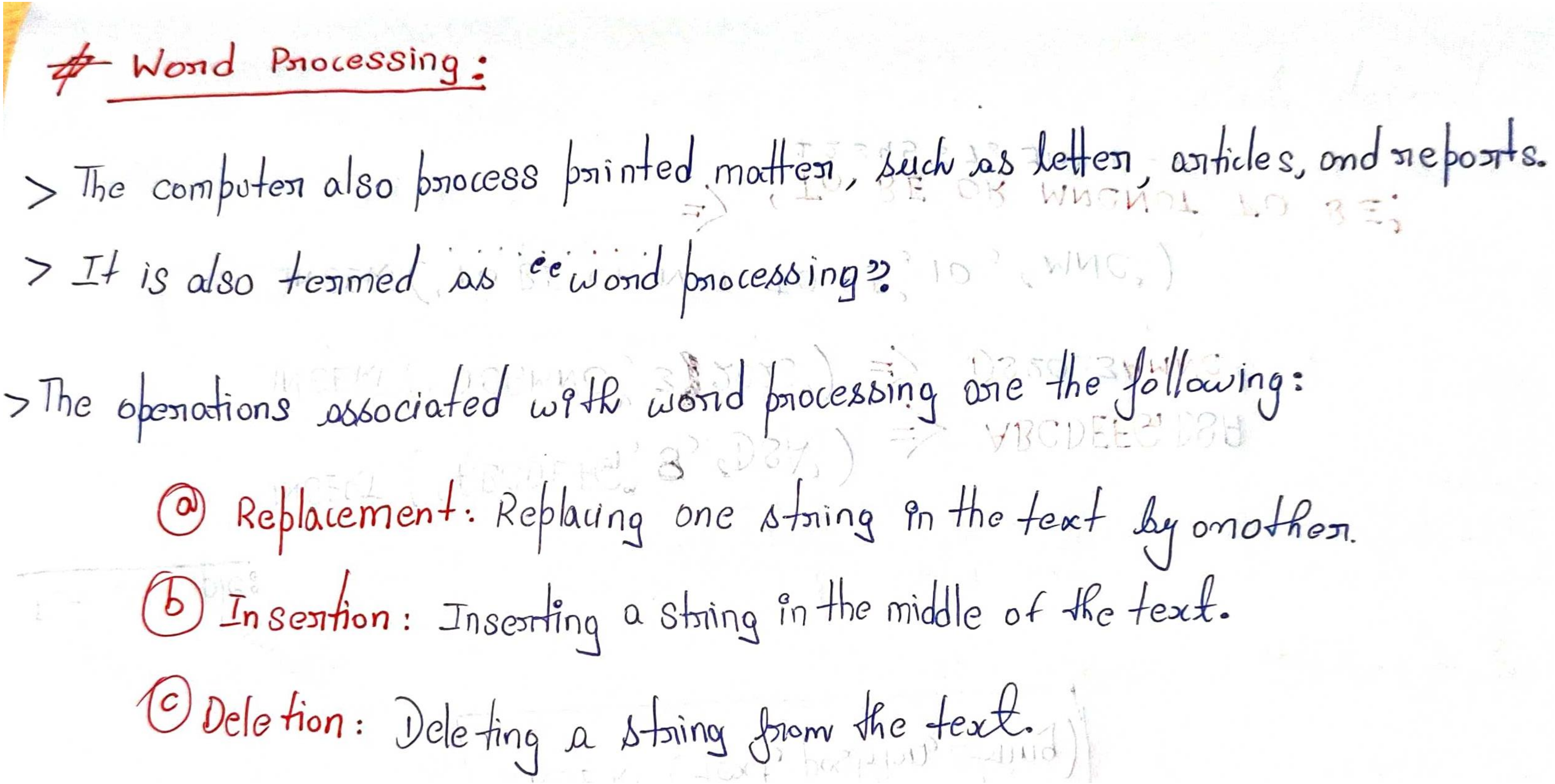
i) Static character : whose length is defined before the program is executed and cannot change throughout the program.

ii) Semi static character : whose length may vary during the execution of the program as long as the length does not exceed a maximum value determined by the program

iii) Dynamic :

↳ whose length can change during the execution of the program.

* These ~~characteristics~~ categories correspond, respectively, to the ways the strings are stored in the memory of the computer.



Word Processing:

> The computer also process printed matter, such as letter, articles, and reports.

> It is also termed as "word processing".

> The operations associated with word processing are the following:

ⓐ Replacement: Replacing one string in the text by another.

ⓑ Insertion: Inserting a string in the middle of the text.

ⓒ Deletion: Deleting a string from the text.

INSERTION:

Let us consider a text T and we have to insert a string S, given S begins in position K.

The insertion operation is denoted as:

INSERT (text, position, string)

For example:

INSERT ('ABCDEFG', 8, 'DSA') => ABCDEFGDSA

INSERT ('DSAMNC', 3, '2023') => DS2023AMNC

INSERT ('TO BE OR NOT TO BE', 10, 'MNC')

=> 'TO BE OR MNCNOT TO BE'

1 2 3 4 5 6 7 8 9 10 11 12 - - - -

* we can also implement INSERT( ) using the previously defined SUBSTRING(T, 1, K).

string

index of first term

final index

Thus,

$$* \boxed{\text{INSERT}(T, K, S) = \text{SUBSTRING}(T, 1, K-1) \,//\, S \,//\, \text{SUBSTRING}(T, K, \text{LENGTH}(T) - K + 1)}$$

string

position of insertion

substring to be inserted

The initial substring of T before the position K, which has length K−1, is concatenated with the string S, and the result is concatenated with the remaining part of T, which begins in position K and has length LENGTH(T) − (K−1) = LENGTH(T) − K+1.

DELETION:

Suppose in a given text T we wont to delete the substring which begins in position K and has length L.

DELETE (text, position, length)

Example:

DELETE ('ABC<u>DE</u>FG', 4, 2) = 'ABCFG'

DELETE ('ABCDEFG', 2, 4) = 'AFG'

Nothing will be deleted if position $\boxed{K=0}$. Thus;

DELETE ('ABCDEFG', 0, 2) = 'ABCDEFG'.

The DELETE function can be implemented using the string operations given in the preceding as:

$$\text{DELETE}\,(T, K, L) = \text{SUBSTRING}\,(T, 1, K-1) \,//\, \text{SUBSTRING}\,(T, K+L, \text{LENGTH}(T) - K - L + 1)$$

* Here, the initial substring of T before position K is concatenated with the terminal substring of T begining in position K+L. The length of the initial substring is K-1, and the length of the terminal substring is:

$$\text{LENGTH}(T) - (K+L-1) = \text{LENGTH}(T) - K - L + 1$$

* We also assume that DELETE (T, K, L) = T when K=0.

* Suppose text T and pattern P are given and we want to delete from T the first occurence of the pattern P.

It is performed using DELETE function:

DELETE (T, INDEX(T,P), LENGTH(P))

→ Here, we first compute INDEX (T, P), the position where P first occurs in T and then we compute LENGTH(P) the number of characters in P

Example :-

T = 'ABCDEFG' and P = 'CD'

INDEX (T, P) = 3 and LENGTH (P) = 2

DELETE ('ABCDEFG', 3, 2) = 'ABEFG'.

Other example:

T = 'ABCDEFG'

P = 'DC'

Perform deletion as previous ?

5 minutes

DELETE (T, INDEX(T, P), LENGTH (P))

Note:

* This pattern deletion can be performed multiple times to remove all the similar occurring patter supported by some iterative logic or control structure.