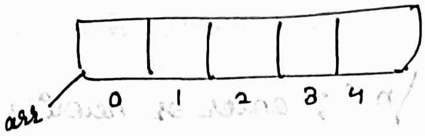
WEEK - 05 :-

CHAR-ARRAYS AND STRINGS # 01

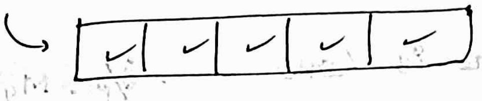
int arr [5]



int → 4 byte

space = 4×5 = 20 byte

→ Char arrays :- char ch [5];



char → 1 byte = 8 bit

space = 5 × 1 = 5 bytes.

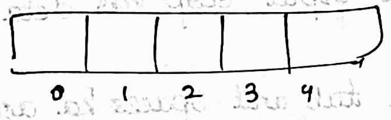
Ch → 1 byte → 8 bit ⇒

Total combination ⇒ 2^8 ⇒ 256 (ASCII table)

char ch [10];

Input :- ~~for (~~ cin >> ~~ch~~ ch ; & output → cout << ch;

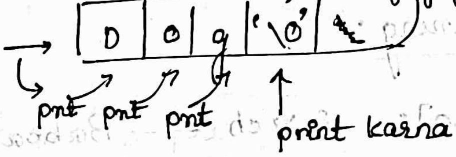
char ch [5];



null character ↓ termination of string. '\0'

cin >> ch ; → i/p: Dog → automatically NC add ho jayega

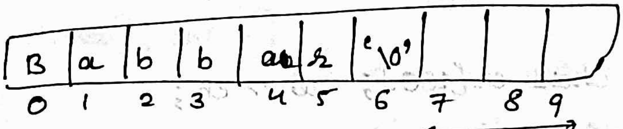
cout << ch; →



print karna ruk jayega.

Null character ('\0') → ASCII value = 0.

Code : → char ch [10] →



→ cin >> ch ⇒ i/p : Babbar

→ cout << ch ⇒ o/p : Babbar ✓

→ Char temp = ch [6]; // null character hai

→ int value = int (temp); → cout << value ; → o/p : 0

↓ Ascii of null char

⇛ ## Delimeter :-

* Jab bhi aap character array ke case ke andar input le rahe honge, toh aapka input kab rukhega? → By Default 'Space'

∴ cin —delimeter→ '\n' ; enter or newline
→ '\t' ; tab
→ '_' ; space

} in all these cases, input stops.

Eg :- My Name is Love → o/p : My.

# cin.getline :- Jab pura line chahiye.

→ cin.getline (ch, 100);

↳ where to store. (ch)
→ maximum input (100)

But, enter se fir bhi input stop kar deta hai.

⇒ cin.getline() → tab and spaces ka asar nahi hoga.

Q.4 Length of String :

char ch [100]; cin>>ch ← Babbar

Sol: if (null_char) → ruk jao;
'\0'

```
char ch[100]; cin>> ch;
int len = findLength (ch, 100);  cout << len ;

int findLength (char ch[], int size)
{
    int length = 0;
    for (int i=0 ; i< size ; i++)
        if (ch [i] == '\0') → break;
        else → length++ ;
}   return length;
```

now, if I had used cin.getline (ch,100);

↳ ★ Space will also be counted.

# function ⟶ strlen (ch) ; ⟶ length of char array

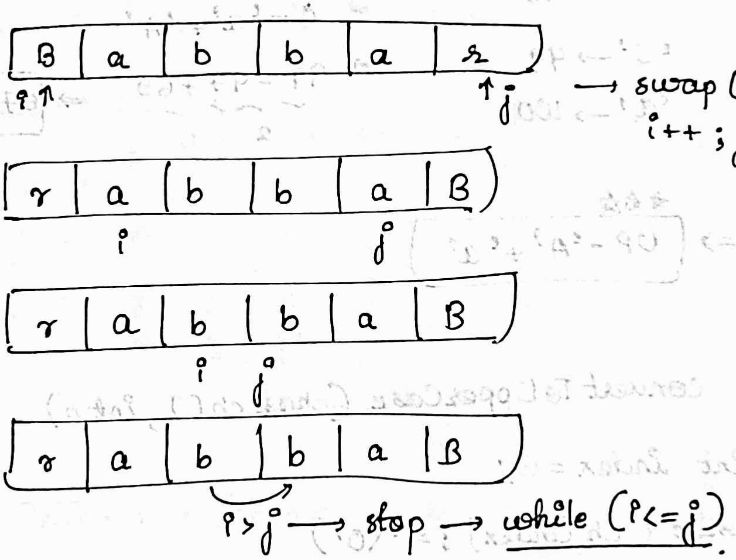
\#include <string.h>

Code :-

```
int ~~len = 0~~, index = 0 ;
while (ch [index] != '\0')
{
    index ++ ;
}
return index ;
```

Q> Reverse a string :-

i/p : Babbar   o/p : rabbaB



→ swap (i,j);
i++ ; j-- ;

i>j ⟶ stop ⟶ while (i<=j)

```
void reverseString( char ch[ ], int n)
{   int i=0 , j=n-1;
    while (i<=j)
    {  swap (a[i], a[j]);
       i++ , --j ;
    }
    return ch;
}
```

```
char ch[100];
cin.getline (ch,100);

int len = findLength (ch,100);
reverseString (ch, len);
cout << ch << endl;
```

Inbuilt function for reverse = ?

Q: Convert to upper case.

i/p :- My name is Love Babbar.

o/p :- MY NAME IS LOVE BABBAR.

'a' → 'A' = ?   ~~let~~ Let, ascii A = 65

↓

'a' = 97

∴ if ($\cancel{97} - \cancel{97} + 65$) = 65 'aa jayega

⇒ lower case − 97 + 65

☆☆☆

{lc → Up} ⇒ $\boxed{l.c - 'a' + 'A'}$ → lowercase to uppercase.

'A' → 65   'a' → 97

'B' → 66   'b' → 98

'C' → 67   'c' → 99

'D' → 68   'd' → 100

now if I want ~~c~~ 'c' to 'C'

⇒ c − 'a' + 'A'

→ $\underbrace{99 - 97}_{2} + 65$ ⇒ $\boxed{67 = C}$

☆☆☆

If $\boxed{UC \to LC}$ ⇒ $\boxed{UP - 'A' + 'a'}$

Code:-
```
void convertToUpperCase (char ch[], int n)
{
        int index = 0;
        while (ch[index] != '\0')
        {
            char currCharacter = ch[index];
          { // check if lowercase, then only convert to upper case.
            if (currCharacter >= 'a' and cur__ <= 'z')
            {
                ~~currCharacter~~ ch[index] = currChar ~~Cha~~ - 'a' + 'A';
            }
            index++;
        }
}
```

Q> Replace @ with space.

i/p :- My @ Name @ is @ Love@Babbar

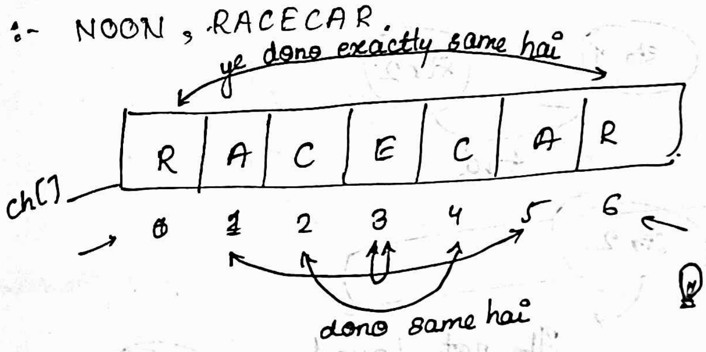
o/p :- My Name is Love Babbar.

```
void replaceCharacter (char ch[], int n);
{   int index =0
    while (ch[index] != '\0')
    {
        if (ch[index] == '@')
        {
            ch[index] = '_';
        }
        index ++;
    }
};
```

☆ Q> Check Palindrome.

⇒ Palindrome :- " Same if reversed "

Eg :- NOON , RACECAR.



Two pointer approach
i++ ; j++ ; if equal.

$(i > j) \rightarrow$ ruk jao.

```
bool CheckPalindrome (char Ch[], int n)
                                  ↳ length of string.
{
    int i=0; j=n-1;
    while (i<=j)
     {
       if (ch[i] == ch[j])
          { i++, j--;
          }
     } else
          { return false;
          }
     }
    return true;
}
```

TC ⇒ $O(n/2) \rightarrow O(n)$

STRINGS → (~~class~~ / ~~object~~) Let's learn as Datatype.

Eg :- string name ;   → datatype (string), ↳ variable (name)

String → dynamic in nature.

⇒ String → collection of characters.

cin >> name ;   cout << name ;   → strings.cpp

printing first character → cout << name[0]<< endl;

Functions ka notes → stringFunctions.cpp

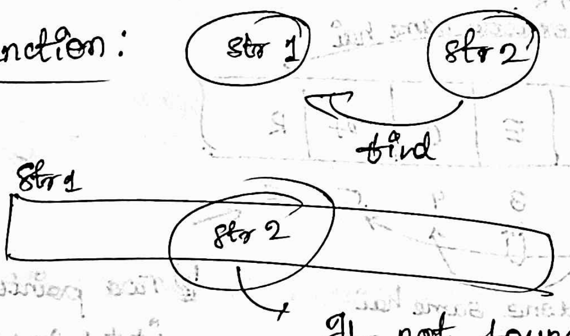
Erase

desc →	T	H	I	S		I	S		A		C	A	R	\0
	0	1	2	3	4	5	6	7	8	9	10	11	12	13

want to remove these 3.

⇒ desc. ~~str~~erase (4 , 3);

Find function :



If not found → npos