Ch-5(L-1) : Char Array I

Char Array

- Array is a data structure into which you can store data
- Char is a datatype which explain which type of data can be stored in data structure
- Input of Char array can be taken by either single element(`cin >> ch[i]`) or directly whole string(`cin >> ch`)

▼ Whenever you take input of string through cin, by default at the last null character comes ("\0") which shows termination of string

I	S	H	A			

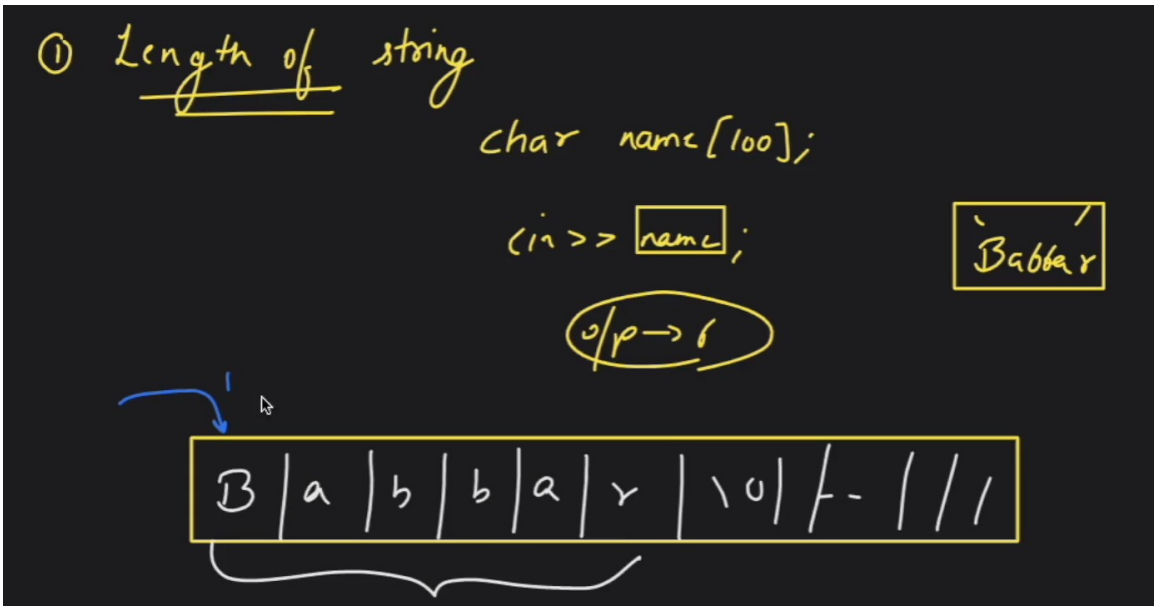
💡 Null character is a terminator for string

- Cin only reads till space, enter, tab i.e if you have enter your full name with space/enter/tab it will only read the first name
- So for reading spaces we can use `cin.getline(arr,max input,delimeter)`

Classwork Questions

▼ Length of array

▼ Question



▼ Logic

- linear search of character krna h or jesehi humko null character milga hum ruk jayenge
- We can also use `strlen(arr)` function to find the length

> 💡 We can use various functions for character arrays like
> strlen → length,
> strcmp → comparison ,
> strcpy → copy

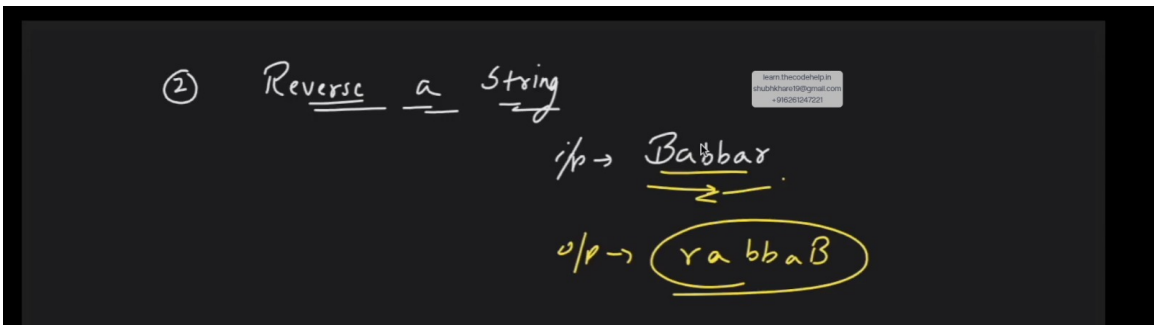
▼ Code

> 💡 Time Complexity : O(n) and space complexity O(1)

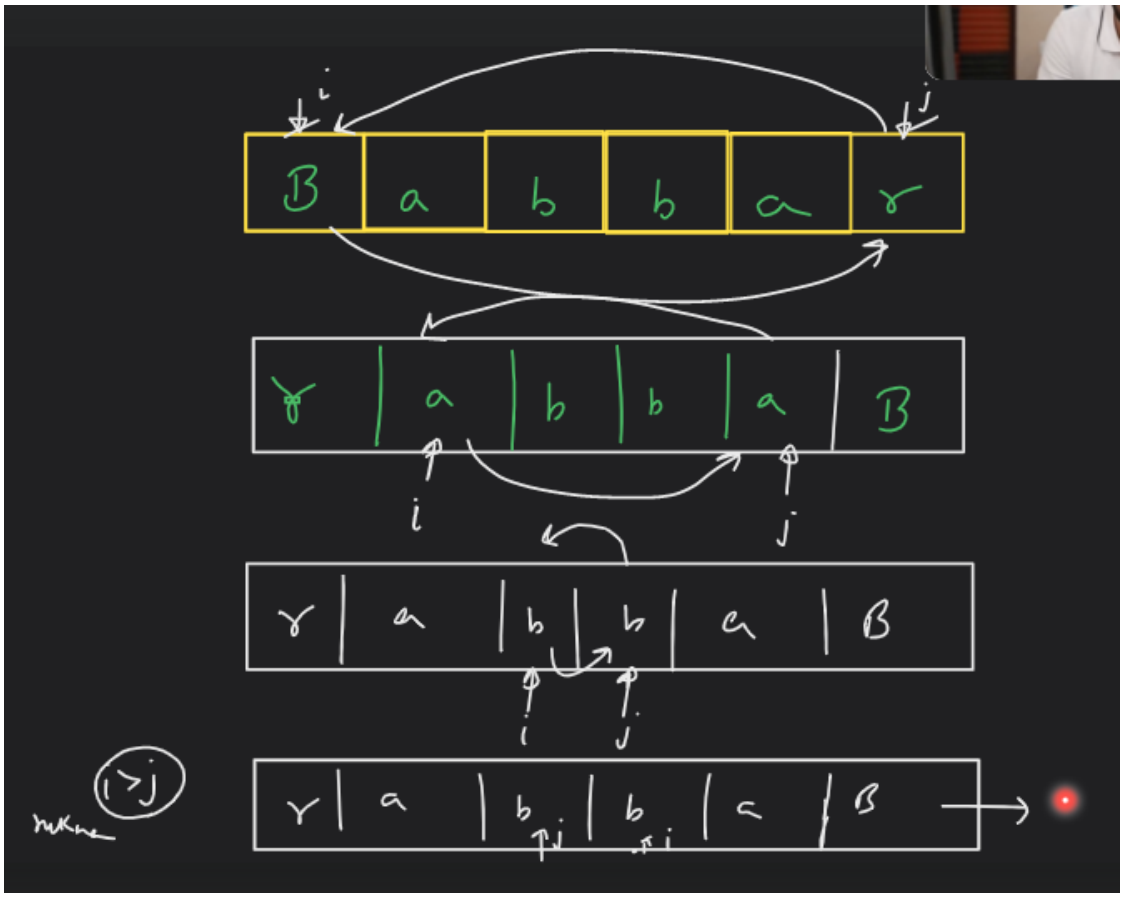
```
#include<iostream>
using namespace std;
#include<string.h>

int getLength(char name[]){
    int length = 0;
    int i=0;
    while(name[i] != '\0'){
        length++;
        i++;
    }
    return length;
}
int main (){
    char name[100];
```

```
    cin>>name;

    cout<<"length is :" << getLength(name)<<endl;


    // Predefin function to get length of the string
    cout << "length is ->" << strlen(name)<<endl;


}
```

- Reverse a string
  - Question



  - Logic



1. We can solve this using 2 pointer approach and by swapping the two pointers
2. We will swap all the elements
3. we need to stop when i > j

▼ Code

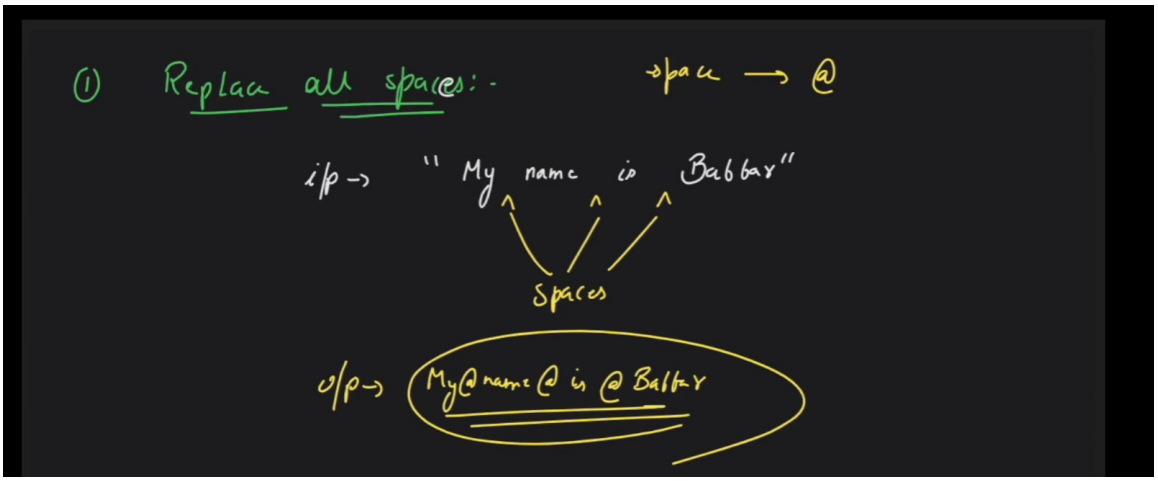
💡 Time Complexity : O(n) and space complexity : O(1)

```
#include<iostream>
#include<string.h>
using namespace std;

void reveserstring(char name[]){
    int i=0;
    int n=strlen(name);
    int j=n-1;
    while(i<=j){
        swap(name[i],name[j]);
        i++;
        j--;
    }
```

```
}
int main(){
    char name[100];

    cin>>name;

    cout << "intilally : " << name <<endl;

    reveserstring(name);

    cout<<"finally : " << name <<endl;
}
```

▼ Replace all the spaces

▼ Question



▼ Logic

- pure string ko traverse krrege or jaha space likhega waha space ko @ se replace kr denge

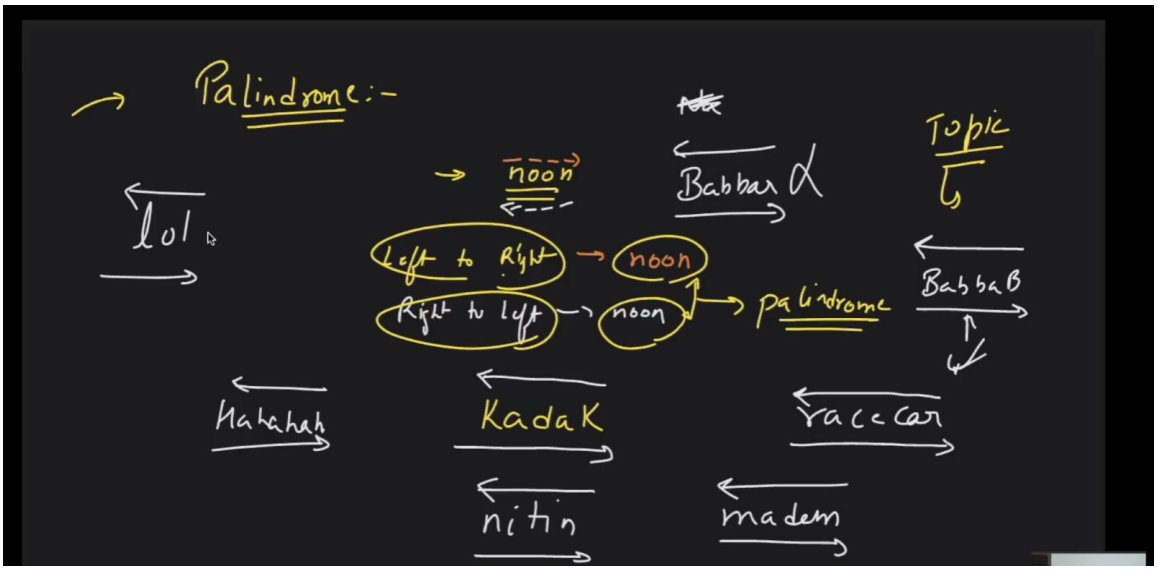
▼ code

💡 Time Complexity : O(n) and space complexity : O(1)

```
#include<iostream>
using namespace std;
#include<string.h>

void replaceSpaces(char sent[]){
    int i =0;
    int n=strlen(sent);
    for(int i=0;i<n;i++){
        if(sent[i] == ' '){
            sent[i] = '@';
```

```
        }
    }
}
int main(){
    char sent[100];

    cin.getline(sent,50);


    replaceSpaces(sent);

    cout<<"finally : " << sent <<endl;
}
```

- ▼ Palindrome problem
  - ▼ Question



  - ▼ logic
    1. Method -1 : phele normal string ko reverse krro and then dono ko compare krlo but it has TC : O(n) and SC : O(n)
    2. Method 2 : by 2 pointer approach, First letter ko last letter se compare krna h
  - ▼ code

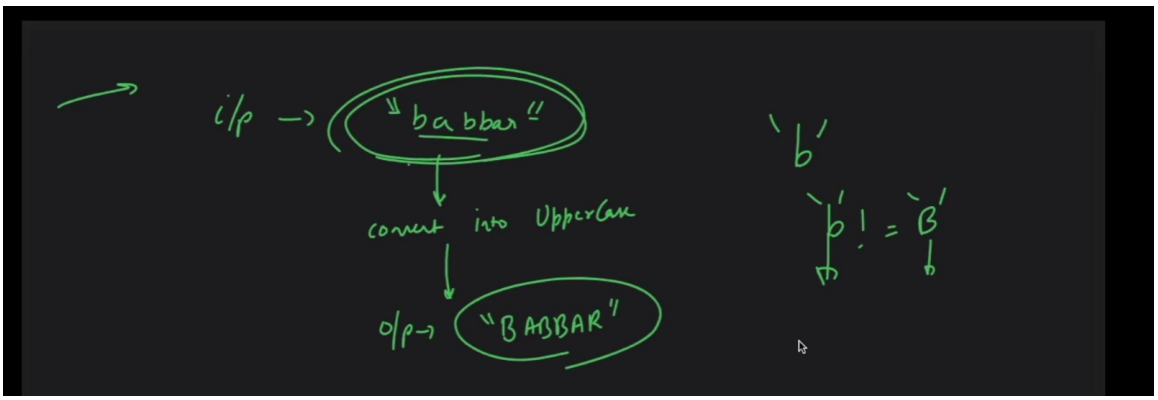
💡 Time Complexity : O(n) and space complexity : O(1)

```
bool Palindrome(char word[]){
int i=0;
```

```
int n = strlen(word)
int j=n-1;

while(i<=j)
{
  if(word[i] != word[j]){
    return false;
  }
  else{
    i++;
    j--;
  }
}
return true;
}
```

▼ Lower case to upper case letters

▼ Question



▼ Logic

1. Convert lower case into upper case → $LC - a + A$.
   Example : c-a+A = C (99-97+65 = 67).
2. Convert upper case into lower case → $UC - A + a$.
   Example : C-A+a = c (67-65+97 = 99).

▼ Code

💡 **Time Complexity : O(n) and space complexity : O(1)**

```
#include<iostream>
using namespace std;
#include<string.h>

void lowertIntoUpper(char arr[]){
    int n=strlen(arr);
    for(int i =0;i<n;i++){
```

```
        arr[i] = arr[i] -'a'+'A';
    }
}

void upperIntoLower(char arr[]){
    int i=0;
    int n=strlen(arr);
    for(int i =0;i<n;i++){
        arr[i] = arr[i] -'A'+'a';
    }
}
int main(){
    char arr1[100] = "ishan";
    char arr2[50] = "SHUBHANSHU";
    // converting lower case to upper case
    lowertIntoUpper(arr1);
    cout << arr1 <<endl;

    // converting upper case to lower case
    upperIntoLower(arr2);
    cout<<arr2<<endl;

}
```

String

- String is a datatype where as character array is a data structure
- Dynamic character array like vector
- creating string → `string str;`
- Input of string
  - `cin >> str;` → This cannot take spaced string as input
  - `getline(cin , str);` → This can take spaced string as input
- Output of string → `cout << str;`

▼ Some Common functions of String

▼ `Compare` → Compare strings

- If `Compare = 0` → both the strings are exactly same
- if `Compare != 0` → both are different
  - `Compare < 0` → First string's first character ASCII value is less than second string's first character and if equal then check second
  - `Compare > 0` → First string's first character ASCII value is greater than second string's first character and if equal then check second

```
// compare
    string a = "love";
    string b = "lover";
```

```
    if(a.compare(b) == 0){
        cout << "both are same" << endl;
    }
    else{
        cout << "both not same are same" << endl;
    }
```

▼ `substr` → Generate substring

- syntax → `str.substr(index,length)`

▼ Example

```
// substr
cout << str.substr(0,3)<<endl;
```

▼ `find` → Find content in string

- It return the index of the value which you are searching

```
// find
    string sent = "hello jee kaise ho";
    if(sent.find("every") == string :: npos){
        cout << "not found";
    }
```

▼ `replace` → Replace portion of string

- first argument denotes konse index se start krna h
- second argument denotes kaha tak replace krna h
- third argument denotes kya replace krna h

```
// replace
    string str1 = "This is my first name";
    string word = "Ishan";

    str1.replace(0,4,word);
    cout << str1 << endl;
```

▼ `erase` → Erase characters from string

- first argument denotes konse index se start krna h deletion
- second argument denotes kitne character delete krna chahte ho

```
// erase
    string str2 = "ABCDEFGH";
```

```
    str2.erase(0,4);
    cout << str2;
```

- `pop_back` → Delete last character

```
 // pop_back
    str.pop_back();
    cout << str<< endl;
```

- `push_back` → Append character to string

```
// push_back
    str.push_back('A');
    cout << str<< endl;
```

- `empty` → Test if string is empty

```
// empty
    cout << "empty : " << str.empty() << endl;
```

- `length` → Return length of string

```
// Length
cout << "length : " << str.length() << endl;
```

- `s.c_str ()` → Returns the character of the string

Homework

- Difference between char array and string

In C++, both character arrays and strings are used to store sequences of characters, but they have some key differences:

1. Syntax: Character arrays are declared using square brackets, while strings are declared using double quotes or using the string class. For example:

```
c++
char arr[] = {'h', 'e', 'l', 'l', 'o'};
char str[] = "hello"; std::string s = "hello";
```

2. Size: The size of a character array must be specified at the time of declaration and cannot be changed later, while the size of a string can be changed dynamically as needed.

3. Null termination: C-style character arrays need to be null terminated, which means that a null character (`\0`) needs to be added at the end of the array to indicate the end of the string. String objects automatically add a null character at the end, so there's no need to do this manually.

4. String manipulation: Strings have several built-in member functions that can be used to manipulate them, such as `append()`, `substr()`, `find()`, etc. Character arrays do not have these built-in functions, so string manipulation must be done manually using functions like `strcpy()`, `strcat()`, `strlen()`, etc.

5. Compatibility: Character arrays are compatible with C-style strings and can be used with functions that take C-style strings as arguments, while strings can be used with C++ string manipulation functions.

In general, strings are easier to use and more flexible, while character arrays are faster and more memory-efficient. However, the choice between the two depends on the specific requirements of the program and the preferences of the programmer.