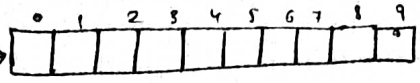
datatype variable name

Char Arrays & Strings - 1

→ char ch[10]; ⇒ ch → 

↳ size

→ data will be of character type.
eg → 'a', 'A', '0', '!'

- char → memory space = 1 byte
  - → range → −128 to 127 (signed)
    unsigned → 0 to 255

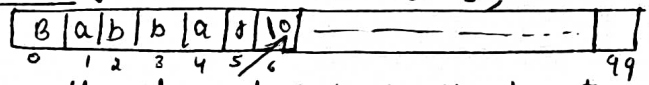
> character array is not a datatype but a data structure.

Taking input →

We can take input as one by one character and as a sequence of character too.

```
char name[100];
cin>>name;          // Input as a sequence.      // input → deepti
cout<<"You entered "<<name<<endl;                o/p → deepti
// Taking i/p as single character
 char ch[100];
  ch[0] = d;                                     i/p → d
      ch[1] = e;                                 o/p → dee
      cin>>ch[2];
      cout<<ch[0]<<ch[1]<<ch[2];
```

In memory → char name[100];



null character → Null character show termination.

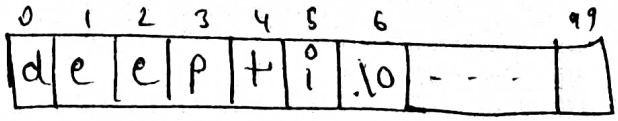
```
cin>> name;                    // Babbar
```

whenever we take input like this a null character is appended at the end of the string (by default).

```
char name[100];
cin>>name;
for(int i=0; i<7; i++){
        cout<<"index:"<<i<<"value "<<ch[i]
            <<name<<endl;
}
```

i/p → deepti
o/p →

index: 0 value : d
index : 1 value : e
index : 2 value : e
:
index : 6 value :



```
int value = (int) name[6];
cout << "value is:" << value << endl;
```

o/p → value is (0).

↳ and 0 is ASCII value of null character.

In rest of the spaces (index from 7 to 99) some garbage value is stored.

Q. Create a char array, store your full name in this and print this.

```
char name[100];
cin >> name;
cout << name << endl;
```

I/p → Deepti Joshi

O/p → Deepti

↳ why only first name?

So the cin keyword reads input only till it does not find a space or tab or enter (new line) character. (whitespaces)

Now what is the solution of this problem?

↳ getline function.

```
char name[100];
cin.getline(name, 50);
```

name of the array. — size. — this means it will maximum 50 size of input.

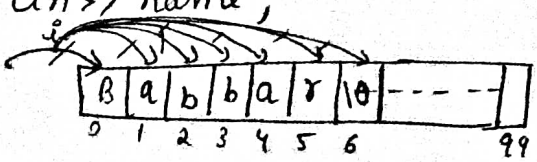
```
getline(cin, name);
```

// find out why this is not working here.

We can set our own delimiter in cin.getline function too. (By default it is enter (new line)).

Q – Length of a string

```
char name[100];
cin >> name;
```



// I/p → Babbar

// O/p → 6

↳ length = 6

→ Traverse the array till we find null character and on each iteration, increase the counter by 1.

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

T.C = O(n)

↑ n is the length of the string

→ or we can simply return i.

- we also have a predefined function to find length of the char array.

  cout << strlen(name);

more functions –

strcmp → ~~two~~ compares two arrays.

strcpy → copy one array to another.

Although we are not going to use these function, we will implement our own codes.

dry run –

	0	1	2	3	4	5	6		
name →	B	a	b	b	a	r	'\0'	—	—

i = 0, length = ~~0 1 2 3 4 5~~ 6

name[0] == '\0' → F
↳ length++

i = 1, name[1] == '\0' → F
↳ length++

i = 2, name[2] == '\0' → F
↳ length++

i = 3, name[3] == '\0' → F
↳ length++

i = 4, name[4] == '\0' → F
length++

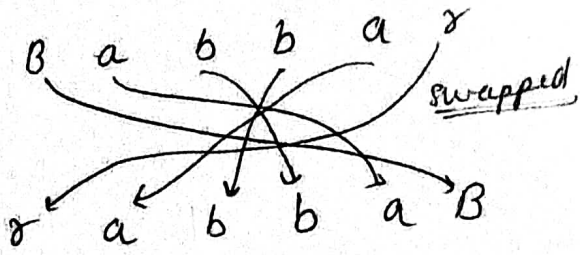
i = 5, name[5] == '\0' → F
length++

i = 6, name[6] == '\0' → T → stop the loop and return length = 6

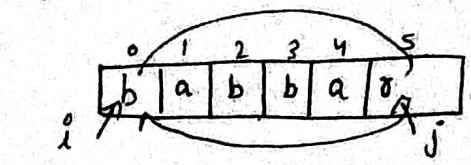
Q4- Reverse a string

i/p → babbar

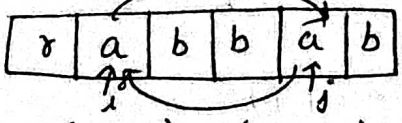
o/p → rabbab



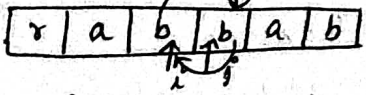
even →



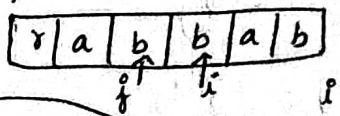
swap(b, r), i++, j--



swap(a, a), i++, j--

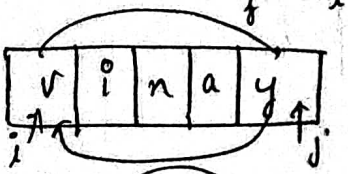


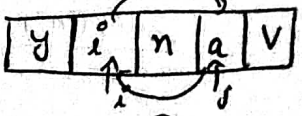
swap(b, b), i++, j--

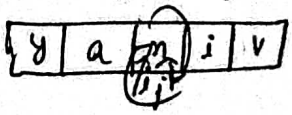


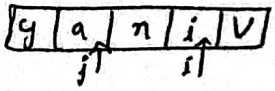
i > j → stop the loop.

odd →









done

T.C = O(n/2)
= O(n)
↑ n is the length of string.
S.C = O(1)

Code-

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

not a inbuilt func, but we implemented this just before this ques.

same logic by for loop.

```
for(; i<=j;){
    swap(name[i], name[j]);
    i++;
    j--;
}
```

Q3- Replace all spaces →

i/p → "My name is Babbar"

spaces

o/p → "My@name@is@Babbar"

```
if (s[i] == ' ')
    s[i] = '@'
```
← simple logic.

```
void replaceSpaces(char sentence[]){ ——— O(n)
    int i = 0;
    int n = strlen(sentence); ——— O(n)
    for (int i = 0; i < n; i++){
        if (sentence[i] == ' ')
            sentence[i] = '@';
    }
}
int main(){
    char sentence[100];
    cin.getline(sentence, 100);
    replaceSpaces(sentence);
    cout << sentence << endl;
}
```

T.C = O(n) + O(n)
T.C = O(n)

Q4- Palindrome :-

noon ←
→

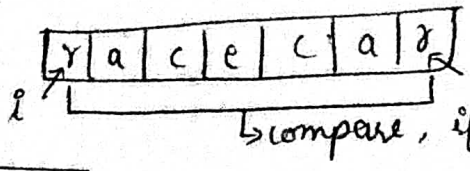
left to right = noon
right to left = noon
> same → so it's a palindrome.

Approach 1-

find reverse string.
Compare. If both are same return true else false.

T.C = O(n)
S.C = O(n)

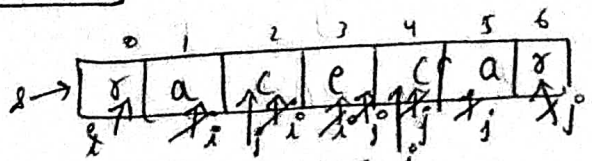
Approach 2 -



↳ compare, if same
↳ i++, j--,
keep doing this until i > j.
if not same
return false.

T.C = O(n/2)
= O(n)
S.C = O(1)

dry run -



s[i] == s[j]

s[0] == s[6] → T, i++, j--

a == a → T, i++, j--

c == c → T, i++, j--

e == e → T, i++, j--

i > j out of loop.
stop return true.

code -

```
bool checkPalindrome ( 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;
}
```

Q → i/p → "babbar" convert into uppercase.
o/p → "BABBAR"

b != B
↳ different ASCII values

'a' → 97

'A' → 65

'a' - 'a' + 'A'

97 - 97 + 65 = 65 → 'A'

'c' - 'a' + 'A'

99 - 97 + 65 = 2+65 = 67 → 'C'

so formula - (lowercase to uppercase)

ch (char) in uppercase = ch - 'a' + 'A'

```
void convertIntoUppercase(char arr[]){
    int n = getLength(arr);
    for(int i = 0; i < n; i++){
        arr[i] = arr[i] - 'a' + 'A';
    }
}
```

T.C = O(n)
S.C = O(1)

now Uppercase to lowercase - HOW

'A' = 65 , 'a' = 97

'A' - 'A' + 'a'

65 - 65 + 97 = 97 → 'a'

formula

→ ch - 'A' + 'a'

now if a character is already in uppercase so we don't need to convert it to uppercase, so we will make a check.

→ `if(arr[i] >= 'a' && arr[i] <= 'z')`
`arr[i] = arr[i] - 'a' + 'A';`

⊙ **Strings →**

↳ sequence of characters.

char array is a data structure of type character values.
And string is a datatype.
↳ dynamic.

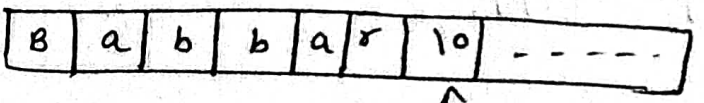
string str;

- To take input →

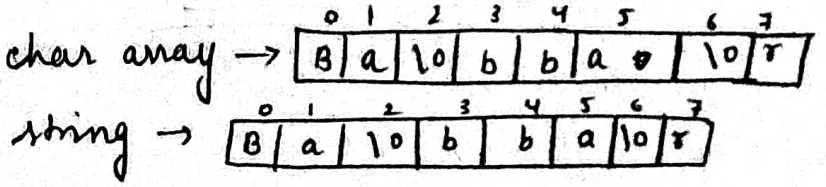
```
cin >> str;          // i/p: Deepti Joshi
cout << str;         o/p → Deepti

getline(cin, str);
cout << str;         // i/p → Deepti Joshi
                     o/p → Deepti Joshi
```

In memory →



↑ NULL character for termination.
But we can't access this.



} print them and see the difference.

Some functions of string →

```
cout << str.length();
```
↳ returns the length of the string. eg, str = "deepti" o/p → 6

```
cout << str.empty();
```
↳ return 0 if the string is empty, otherwise 1.

```
str.push_back('A');
cout << str;
```
↳ append 'A' at the last of the string.

str.pop_back(); removes the last character.

```
cout << str.substr(0, 6) << endl;
```

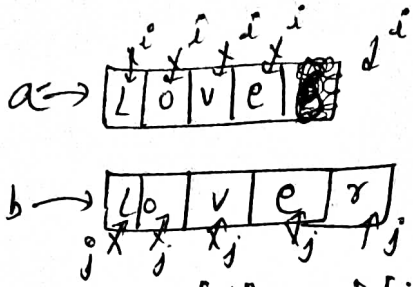
substr() func returns the specified substring of a string. It does not modify the original string.

```
str.substr(0, 6);
```
0 ↳ starting index; 6 ↳ length of the substring you want.

eg → str = "Deepti".
cout << str.substr(0,2);
↳ o/p ⇒ De

① compare( ) → case sensitive func. (a != A)
returns 0 if both strings are same, otherwise 1.

```
string a = "love";
string b = "lover";
if (a.compare(b) == 0)
        cout << "same";
else
        cout << "Not same";
```



a[i] == b[j]
↳ i++, j--

else return false;

// implementing our own compare function →

```
bool compareStrings(string a, string b){
    if(a.length() != b.length())
        return false;
    else{
        int j=0;
        for(int i=0; i<a.length(); i++){
            if(a[i] != b[j]){
                return false;
            }
            j++;
        }
    }
    return true;
}
```

well this else can be ignored here.

// also we dont need to compare with b[j] but we can compare with b[i] too.

compare() function returns
→ 0 → both strings are exactly same.
→ !0 → different strings
→ <0 → when the ASCII value of first string's first character is smaller than second strings' first char.
→ >0 → when " " " " " greater " " " " "

```
string a = "abcd";
string b = "bcda";
cout << a.compare(b);   ⟶ //o/p → -1 (-ve)
cout << b.compare(a);   ⟶ //o/p → 1 (+ve)
```

if first character is equal then we ~~con~~ check on next ~~two~~ character (basically it is decided on first different character of strings).

⊙ **find() func** → finds the first occurrence of a sequence in a string.

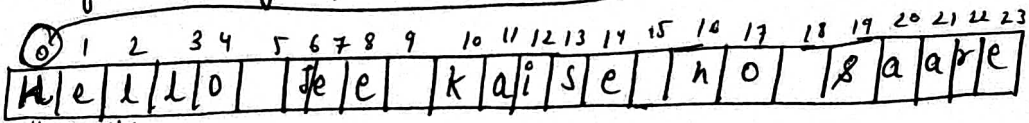
Return value →

It returns the position of the first character of first match.

If no matches were found, the function returns npos.

↳ npos means no position.

```
string s = "Hello Jee kaise ho saare";
string target = "Hello";
cout << s.find(target);        //o/p → 0
```




```
string t = "Je";
cout << s.find(t);             //o/p → 6

string target2 = "Everyone";
cout << s.find(target2);
if(s.find(target) == std::string::npos){
    cout << "Not found";       //o/p → Not found.
}
```

//o/p → 1849 — — some garbage value is printed.

This std is optional.

⊙ **replace func()** →

```
string str = "This is my first message";
string words = "Babbar";
str.replace(0, 4, word);
```

// Remove "This" and place "Babbar" in it's place.

cout << str << endl; //o/p → Babbar is my first message

str.replace (11, 5, "second");

This will replace first with second.

cout << str; //o/p → ~~This~~, Babbar is my second ~~string~~ message.

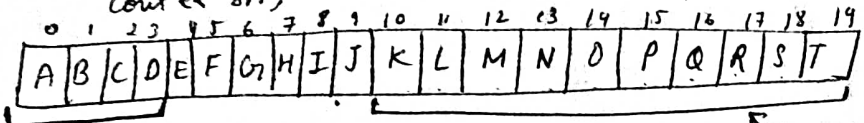
★ → Check out all the syntaxes.

⊙ erase() function →

string str = " ABCDEFGHIJKLMNOPQRST ";

str.erase (0,4); //o/p → EFGHIJKLMNOPQRST
cout << str;

str.erase (10, 10); //o/p → ABCDEFGHIJ
cout << str;



This is deleted (length = 4, starting from 0th index)

str.erase (0,4);

starting index

length of the string you want to delete.

This is deleted. (length = 10, starting from 10th index)