

Char Arrays & Strings - 1

datatype \downarrow variable name \downarrow
 \rightarrow char ch[10]; \Rightarrow ch \rightarrow

0	1	2	3	4	5	6	7	8	9

 \rightarrow data will be of character type.
 eg \rightarrow 'a', 'A', '0', '1'
 • char \rightarrow memory space = 1 bytes
 \rightarrow range \rightarrow -128 to 127 (signed)
 unsigned \rightarrow 0 to 255

character array is not a datatype but a data structure.

Taking input \rightarrow

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  $\rightarrow$  deepTi
cout << "You entered " << name << endl;                 o/p  $\rightarrow$  deepTi

// Taking i/p as single character
char ch[100];
char ch[0] = d;
ch[1] = e;
cin >> ch[2];
cout << ch[0] << ch[1] << ch[2];
    
```

i/p \rightarrow d
o/p \rightarrow dee

In memory \rightarrow char name[100];

0	1	2	3	4	5	6	7	8	9	10	...	99
	d	e	e	p	t	i	.	0				

null character \rightarrow 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 " << name[i];
    << name << endl;
}
```

i/p \rightarrow deepTi
o/p \rightarrow

Index: 0 value: d

index: 1 value: e

index: 2 value: e

;

index: 6 value;

0	1	2	3	4	5	6	7	8	9	10	...	99
d	e	e	p	t	i	.	0					

```
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;
```

l/p → Deepthi Joshi

o/p → Deepthi

↓ 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.

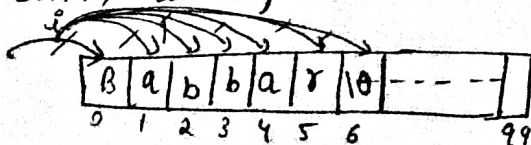
— this means it will maximum 50 size. ✓ size of input.

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;
```



l/p → Babbar

l/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[]){
```

T.C = $O(n)$

n is the length of the string

```
    int length = 0;
```

```
    int i = 0;
```

```
    while (name[i] != '\0'){
```

```
        length++;
```

```
        i++;
```

```
    } return length; → or we can simply return i.
```

```
}
```

```
int main(){
```

```
    char name;
```

```
    cin >> name;
```

```
    cout << getlength(name);
```

```
}
```

- 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 →	a	b	b	a	r	'\0'	-	-

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

name[0] == 'a' → F

↳ length++

$i = 1$

name[1] == 'b' → F

↳ length++

$i = 2$

name[2] == 'b' → F

↳ length++

$i = 3$

name[3] == 'a' → F

↳ length++

$i = 4$

name[4] == 'r' → F

length++

$i = 5$

name[5] == '\0' → F

length++

$i = 6$

name[6] == '\0' → T

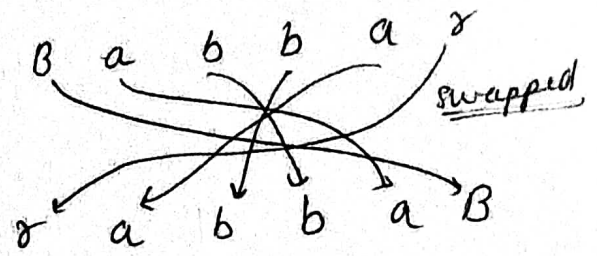
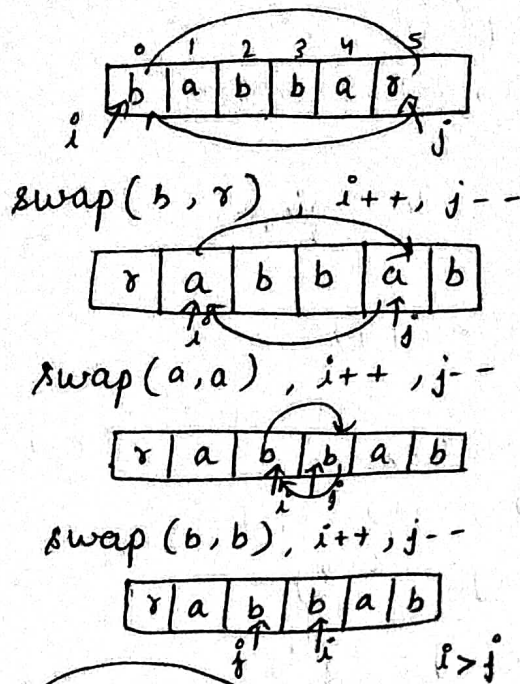
→ stop the loop and return length = 6

Qd- Reverse a String

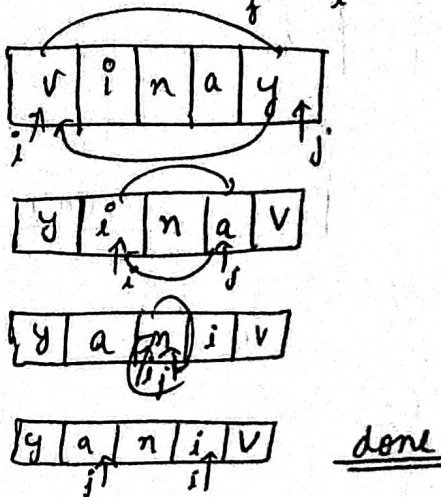
i/p → babbar

o/p → rabbar

even →



odd →



$$T.C = O(n/2)$$

$$= O(n)$$

↑ 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 -

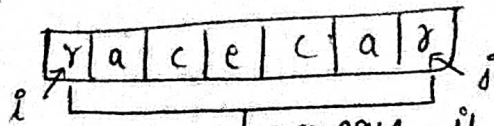
find reverse string.

compare. If both are same return true else false.

T.C = $O(n)$

S.C = $O(n)$

Approach 2 -



↳ compare, if same

↳ $i++$, $j--$,

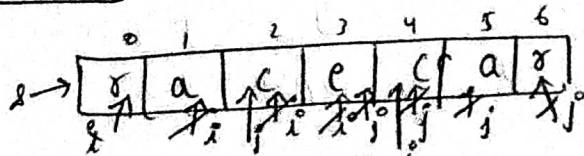
keep doing this until $i > j$.

if not same
return false.

$$T.C = O(n/2) \\ = O(n)$$

$$S.C = O(1)$$

dry run -



$$s[i] == s[j]$$

$$s[0] == s[6] \rightarrow T, i++, j--$$

$$a == a \rightarrow T, i++, j--$$

$$c == c \rightarrow T, i++, j--$$

$$e == e \rightarrow T, i++, j--$$

$i > j$ 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

$$'a' \rightarrow 97$$

$$'A' \rightarrow 65$$

$$'a' - 'a' + 'A'$$

$$97 - 97 + 65 = 65 \rightarrow 'A'$$

$$'c' - 'a' + 'A'$$

$$99 - 97 + 65 = 2 + 65 = 67 \rightarrow 'C'$$

so formula - (lowercase to uppercase)

$$\text{ch}(\text{char}) \text{ in uppercase} = \text{ch} - 'a' + 'A'$$

void convertIntoUppercase(char arr[]){

int n = getlength(arr);

for (int i = 0; i < n; i++){

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

}

}

$$\begin{aligned} T.C &= O(n) \\ S.C &= O(1) \end{aligned}$$

now Uppercase to lowercase - H.W

$$'A' = 65, 'a' = 97$$

$$'A' - 'A' + 'a'$$

$$65 - 65 + 97 = 97 \rightarrow 'a'$$

formula

$$\rightarrow \text{ch} - 'A' + 'a'$$

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')

$$\text{arr[i]} = \text{arr[i]} - 'a' + 'A';$$

① 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;

cout << str;

getline(cin, str);

cout << str;

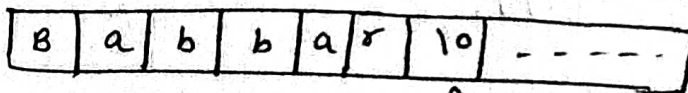
// i/p, Deepthi Joshi

o/p → 'Deepthi'

// i/p → Deepthi Joshi

o/p → Deepthi Joshi

In memory →



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

char array →

0	1	2	3	4	5	6	7
B	a	\0	b	b	a	\0	r

string →

0	1	2	3	4	5	6	7
B	a	\0	b	b	a	\0	r

} print them and see the difference.

Some functions of string →

cout << str.length(); → returns the length of the string. eg, str = "deepthi" 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);
↑ starting index

length of the substring you want.

eg → str = "Deepthi".

cout << str.substr(0, 2);

↳ o/p → De

① compare() → case sensitive func. ($a \neq A$)
 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 →

l	o	v	e	
---	---	---	---	--

b →

l	o	v	e	r
---	---	---	---	---

$a[i] == b[j]$
 i++, j--

else return false;

// implementing our own compare function →

```
bool compareStrings(string a, string b){
  if (a.length() != b.length())
    return false;
```

else ↓
 well this else can be ignored here.

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

// also we don't need to compare with $b[j]$ but we can compare with $b[i]$ too.

```
  return true;
}
```

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 string's first char.
 → > 0 → when greater

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

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

① 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); // dp → 0
```

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23		
H	e	l	l	o		J	e			k	a	i	s	e			h	o			J	a	a	r	e

```
string t = "Jee";
cout << s.find(t); // dp → 6
```

```
string target2 = "Everyone";
```

```
cout << s.find(target2);
```

```
if (s.find(target) == (std::string::npos)) {
```

this std is optional.

```
cout << "Not found";
```

```
}
```

// dp → Not found.

// dp → 1849 — — —
some garbage value is printed.

② replace() func →

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

// Remove "This" and place "Babbar" in its place.

```
cout << str << endl; // o/p → Babbar is my first message
str.replace(" ", 5, "second");
// This will replace first with second.
cout << str; // o/p → This, is my second string Babbar message.
```

* → check out all the syntaxes.

① erase() function →

```
string str = "ABCDEFGHIJKLMN OPQRST";
str.erase(0, 4); // o/p → EFGHIJKLMN OPQRST
cout << str;
str.erase(10, 10); // o/p → ABCDEFGHIJ
cout << str;
```

