

25/02/2023

Character Arrays

`int arr[10];` By this statement, in memory 10 blocks of size = 4 bytes each will be allocated. Integers are stored here.

`char ch[10];` By this statement, in memory 10 blocks of size = 1 byte each will be allocated. Each block will contain a character.

Signed char range \rightarrow -128 to 127

Unsigned char range \rightarrow 0 to 255

`char arr[10];`

↑

data type `arr` is a data structure & not data type.

✓ `int arr[10];`

If we want to take input for 3rd element, then we do `cin >> arr[2];`

✓ `char ch[10];`

In this we can take input as single character or as sequence

`cin >> ch[2];` // Single character input

`cin >> ch;` // As a sequence

Also we take input via for loop also just like

integer arrays.

```
char name[100];
```

```
cin >> name;
```

Whenever we take input like this, by default there will be a null character at the end & it represents the termination.

name → Babbar

→ null character

B	a	b	b	a	r	l	o		-----		
---	---	---	---	---	---	---	---	--	-------	--	--

100 blocks of size = 1 byte

Integer value of null character i.e the ASCII code of null character is 0.

Note →

```
char name[100];
```

```
char name[1000];
```

```
char name[10];
```

Space complexity = $O(1)$

```
char name[100];
```

```
cin >> name; → Bhavya Bhalla
```

```
cout << name; → Bhavya
```

cin does not read spaces or tabs & hence only Bhavya is printed. But how can we print Bhavya Bhalla i.e how can we detect spaces.

Reading spaces

Rather than using cin we use getline

function.

`cin.getline (arr, 100);`
↗ name of array

↘ maximum length of input

We can also set our own delimiter. By default the delimiter is space.

Problem Solving

Q1 Find the length of string.

i/p → Bhavya

o/p → 6

We just have to simply traverse linearly until we get the null character & as we pass each character, len is incremented.

Code

```
int getLength (char name []) {
```

```
    int len = 0;
```

```
    int i = 0;
```

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

```
        len++;
```

```
        i++;
```

```
    }
```

```
    return len;
```

```
}
```

Inbuilt functions for string

1) `strlen (name)` ; → To find length of the string.

2) strcmp // For comparison

But we don't have to dependent on the inbuilt functions.

Q2 Reverse the string

i/p \rightarrow abc
o/p \rightarrow cba

Algorithm

B a b b a r
0 1 2 3 4 5

1) $S = 0$
 $e = 5$

swap (name [S], name [e]) ;
 $S++$, $e--$

r a b b a B
0 1 2 3 4 5

2) $S = 1$
 $e = 4$

swap (name [S], name [e]) ;
 $S++$, $e--$

r a b b a B
0 1 2 3 4 5

3) $S = 2$
 $e = 3$

swap (name [S], name [e]) ;
 $S++$, $e--$

r a b b a B
0 1 2 3 4 5

now $s > e$, hence exit the loop as we already have reversed the string.

Code

void

```
reverseString (char name []) {
```

```
    int s = 0;
```

```
    int e = getLength(name) - 1;
```

```
    while (s <= e) {
```

```
        swap (name[s], name[e]);
```

```
        s++;
```

```
        e--;
```

```
    }
```

```
}
```

Time complexity : $O(n)$

Space complexity : $O(1)$

Note → Array is passed as reference to function whereas a copy of vector is created.

Q3 Replace all spaces in the string

i/p → My name is Babbar

o/p → My @name@ is @Babbar

Just we have to use a single if condition & we are at space then make it @.

Code


```
void replaceSpaces (char name[], int len)
```

```
    for (int i=0; i<len; i++) {
        if (name[i] == ' ') {
            name[i] = '@';
        }
    }
}
```

Time complexity : $O(n)$

Space complexity : $O(1)$

Q4 Palindrome of string
Palindrome means same if we read from left to right or right to left.

i/p \rightarrow racecar

o/p \rightarrow True

Algorithm - 1 \rightarrow Takes extra space of $O(n)$.

i/p \rightarrow racecar

Make new string & store reverse string in it.

Compare reverse string & i/p string & if equal then palindrome, else not a palindrome.

Algorithm - 2

Using 2 pointer approach

r a c e c a r

1) $s=0$
 $e=6$

$arr[s] == arr[e] \rightarrow True, s++ \& e--$

2) $s=1, e=5$

$arr[s] == arr[e] \rightarrow True, s++ \& e--$

3) $s=2, e=4$

$arr[s] == arr[e] \rightarrow True, s++ \& e--$

4) $s=3, e=3$

$arr[s] == arr[e] \rightarrow True, s++ \& e--$

$s=4, e=2$ $s > e$? exit the loop & hence all were true & hence this is the palindrome.

Code

```
bool checkPalindrome(char name[]) {
    int s = 0;
    int e = strlen(name) - 1;
    while (s <= e) {
        if (name[s] != name[e]) {
            return false;
        }
        else {
            s++;
            e--;
        }
    }
    return true;
}
```


Q5 Convert string to uppercase

i/p → bhavya
o/p → BHAVYA

To convert any lowercase character to the uppercase character, we need to subtract 'a' & add 'A' to the character

$ch = ch - 'a' + 'A';$

Code

```
void convertStringToUpper (char name[])
{
    int n = strlen(name);
    for (int i = 0; i < n; i++) {
        if (name[i] >= 'a' && name[i] <= 'z') {
            name[i] = name[i] - 'a' + 'A';
        }
    }
}
```

Strings

Sequence of characters. String is a data type whereas character array is a data structure.

There is very minute difference b/w strings & character array. Null character is also present in strings.

We will be using strings only in the future. Rarely we will be using character arrays.

```
string str;
```

```
cin >> str; // Taking input in string
```

```
getline (cin, str); // To read spaces also.
```

Note → We can not access null character in case of strings, however we don't have any access issue in case of character arrays.

Inbuilt functions of strings

str → Bhavya

- 1) str.length() ; → 6 . Tells length of string
- 2) str.empty() ; → 0 . 0 as string is not empty
1 if string is empty
- 3) str.push-back('s') ; → Bhavyas
- 4) str.pop-back() ; → Bhavya . Removes last character
- 5) str.substr(0, 6) ; → Bhavya

↑ ↳ length of substring
 0th index start

substr function is very important & we might forget this.

- ↳ Case sensitive
- 6) s1.compare(s2) → 0 Equal strings

1 s1 > s2
 -1 s2 > s1

However we use compare function to know whether the strings are equal or not.

Implement compare function

```
bool compareString (string a, string b) {
```

```
    if (a.length() != b.length())
        return false;
```

```
    else {
```

```
        for (int i = 0; i < a.length(); i++) {
```

```
            if (a[i] != b[i]) {
```

```
                return false;
```

```
            }
```

```
        }
```

```
    }
```

```
        return true;
```

```
}
```

⌞ This also we might forget

7) find function

```
string sentence = "hello all";
```

```
string target = "hello";
```

```
sentence.find(target); → 0
```

```
sentence.find("all"); → 6
```

```
sentence.find("hi"); → Some random  
value or  
npos.
```

std::string::npos → To access npos.

↳ no position

8) replace

```
string str = "This is my first message";  
string word = "Babbar";
```

→ how many characters to remove
`str.replace(0, 4, word);`
 ^{start} → what to put there?

O/p → Babbar is my first message

9) erase

```
String str = "ABCDEFGH IJKLMN OPQRST";  
str.erase(0, 4); → delete 4 characters  
    start from 0
```

EFGH IJKLMN OPQRST → O/p of above code