

Strings in One Shot

COLLEGE
WALLAH

Quick revision of arrays

↓
data structure → collection

0	1	2	3	4	5
1	3	4	2	7	6

```
int arr[7] = { }
```

```
arr[2] = 7
```

```
char arr[5];
```

```
int n;
```

```
float y;
```

```
char ch;
```

COLLEGE
WALLAH

Quick revision of characters

Characters and ASCII values, a must understand concept!

`char ch = 'a';`



ch

1 byte

↓

8 bits

↓

2^8 characters \rightarrow 256

`int n = 5;`



n

4 bytes \rightarrow 32 bits \rightarrow 2^{32} numbers

-2^{31} to $2^{31} - 1$

0 1 2 3 4 5 6 7 8 9 a b c d e f

Quick revision of characters

Characters and ASCII values, a must understand concept!

'A' → 65

'a' → 97

'0' → 48

'9' → 57

COLLEGE
WALLAH

What are strings?



character arrays

COLLEGE
WALLAH

Creating character arrays

Hello World

↓

```
char arr[11] = { 'H', 'e', 'l', 'l', 'o', ' ', 'W', 'o', 'r', 'l', 'd' };
```

```
for(int i=0; i<11; i++){
    printf("%c", arr[i]);
}
```

COLLEGE
WALLAH

Printing character array with loop

COLLEGE
WALLAH

Now, let's print character array with null character

'a' 'b' 'c' 'A' '\$' 'C'

Null Character \rightarrow $\backslash 0 \rightarrow 0$

Creating string

Is grouping of characters into array enough?



```
char arr[] = {'H', 'e', 'l', 'l', 'o'};
```



```
char arr[] = "Hello";
```

COLLEGE
WALLAH

Initializing a string

Just how we make arrays! (Already done)

```
char str[] = {'P', 'h', 'y', 's', 'i', 'c', 's', ' ', 'W', 'a', 'l', 'l', 'a', 'h', '\0'};
```

COLLEGE
WALLAH

Initializing a string

Assigning string literal without size

```
char str[] = "PhysicsWallah";
```

What is the size of str? → 14

COLLEGE
WALLAH

Initializing a string

Assigning string literal with size:

```
char str[50] = "PhysicsWallah";
```

↓
13 + 1

COLLEGE
WALLAH

Think...

Is the following code snippet correct?

```
char str[13] = "PhysicsWallah";
```



error

COLLEGE
WALLAH

Accessing individual characters

```
char str[] = "Physics Wallah";
```

Predict the output?

(a) `printf("%c", str[5]);` *c*

(b) `printf("%d", str[9]);` *97*

Modifying individual characters

```
int arr[5] = { 1, 2, 3, 4, 5};
```

```
arr[0] = 10;
```

```
char str[] = "Physics Wallah";
```

Predict str after:

(a) `str[0] = 'M';` → Mhysics Wallah

(b) `str[1] = 97;` // a → Paysics Wallah

Ways of printing particular element

```
arr[i]  
*(arr + i)  
*(i + arr)  
i[arr]
```

COLLEGE
WALLAH

* Input and Output of string without loop

gets() , puts()

"%c" → character

COLLEGE
WALLAH

Can we use string (character arrays) as pointers?

```
char str[] = "PhysicsWallah";
char *ptr = str;
```

College Wallah \0
↑
ptr

```
char str[] = "College Wallah";
char* ptr = str; // ptr now points to str[0]
int i = 0;
while(*ptr!='\0'){
    printf("%c",*ptr);
    ptr++;
    i++;
}
return 0;
```

Output

Co

We get another way of initialising strings

```
char *ptr = "Physics Wallah";
```

character's pointer can also be used to store address of a string.

Note: Such direct initialisation using pointers results in a read-only memory allocation of character arrays and hence, causes **undefined behaviour** when we try to change the characters.

```
ptr[0] = 'm';      Error!
```

↓
individual

Interesting thing about such initialization

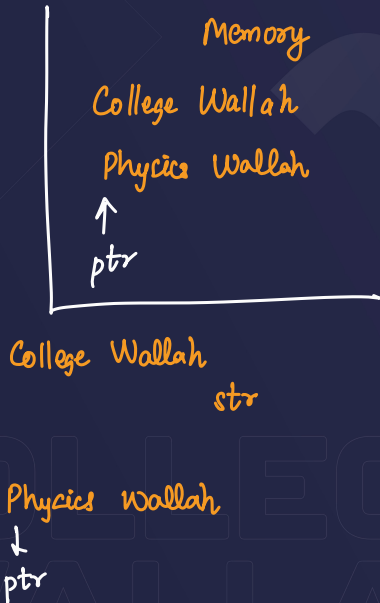
```
char str[] = "Physics Wallah";  
printf("%s", str);  
str = "College Wallah"; Error!
```

In normal initialisation, we can modify individual characters but not the ENTIRE string.

Pointer initialization, we can modify entire string but not the individual characters.

```
char* ptr = "College Wallah";  
ptr = "Physics Wallah";
```

```
char str[] = "College Wallah";  
char* ptr = str;  
ptr = "Physics Wallah";  
printf("%s", str);
```



```
char str[] = "College Wallah";
// char* ptr = str;
// ptr = "Physics Wallah";
char* p = str;
*p = 'P';
printf("%s",str);
```

P
 college Wallah
 ↑
 P
 str

COLLEGE
WALLAH

Interesting thing about such initialization

```
char *ptr = "Physics Wallah";  
printf("%s \n", ptr);  
ptr = "College Wallah"; Works perfectly!  
printf("%s", ptr);
```

Physics Wallah
↑
ptr

College Wallah
↑
ptr

Why does this happen?

Pointers change the address to which they point after initialising a new character array!

```
char *ptr = "Physics Wallah";  
printf("Address 1: %p \n", ptr);  
ptr = "College Wallah";  
printf("Address 2: %p", ptr);
```

COLLEGE
WALLAH

Implement: Copy one string to another

```
char s1[] = "Physics Wallah";
char *s2 = s1;
```

```
// Let's change in s1.
s1[0] = 'M';
printf("%s", s2);
```

Physics Wallah s1
 ↑
 s2
 ↓
 shallow copy

Implement: Copy one string to another

```
char s1[] = "Physics Wallah";  
char *s2 = s1;
```

```
// Let's change in s1.  
s1[0] = 'M';  
printf("%s", s2);
```

Not a deep copy: Here, s2 points to the same character array and hence, change in s1 is also reflect in s2.

Copy one string to another creating a deep copy

```
char str[] = "Hello";
```

```
char str2[] =
```

COLLEGE
WALLAH

Making our tasks easy!

s1 = "Raghav Garg";

s2 = "Garg";

Useful functions for C already in standard library!

strcat(s1,s2);

strlen(char *str)	Returns the length of string
strcpy(char *s1, char *s2)	Copies the contents of string s2 to string s1
strcat(char *s1, char *s2)	Concat s1 string with s2 and stores the result in s1
strcmp(char *s1, char *s2)	Compares the two strings
strncpy(char *s2, char *s1, int len)	Copy substring of size len starting from s1 character pointer into s2.

```
char s1[7] = "Raghav";
```

```
char s2[5] = "Garg";
```

```
strcat(s1, s2);
```

Raghav Garg

COLLEGE
WALLAH

Q41

Inserting a character in a string

Write a function to insert a new character in a string at a given position.

index

`char str[] = "College";`



C	o	l	l	e	g	e
0	1	2	3	4	5	6

2nd index \rightarrow 'l' \rightarrow College

c	o	l	l	e	g	e	
---	---	---	---	---	---	---	--

c	o		l	l	e	g	e
---	---	--	---	---	---	---	---

* Reverse a string

take input & print reverse

COLLEGE
WALLAH