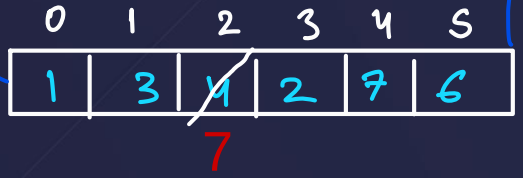
Strings in One Shot

[character arrays]

Quick revision of arrays

data structure → collection (of objects)



{integer array}
{collection of int type objects}

[1-D] array `int arr[7] = {`

`};` → [declaration and initialization]

`arr[2] = 7;` → [accessing to individual elements] + [updation]

`[ char arr[5]; ]` → character type array declaration

- [9-1]
- [it is string]

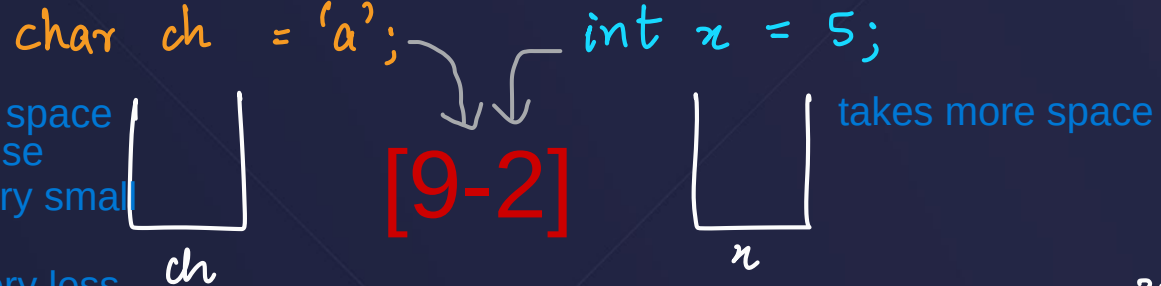
```
int x;
float y;
char ch;
```

→ [data types]

[p.t.o. --> go to page no 5]

Quick revision of characters

Characters and ASCII values, a must understand concept!



[1 byte = 8 bits]

1 byte → 8 bits → 2^8 characters → 256 individual characters can be stored

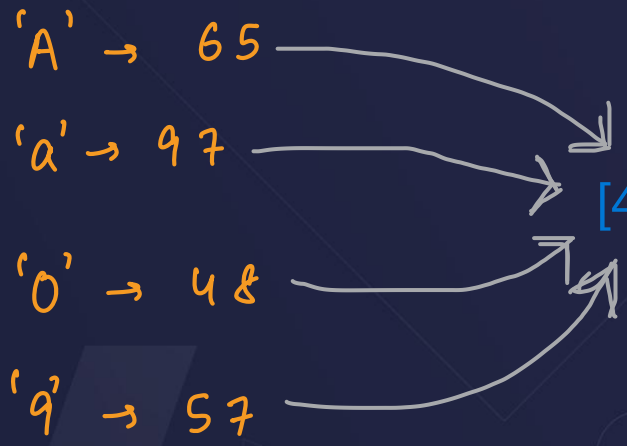
4 bytes → 32 bits → 2^{32} numbers

[-2^{31} to $2^{31}-1$] Range of integers

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

Quick revision of characters

Characters and ASCII values, a must understand concept!



[4 important things to remember]

[9-3]-[9-4]

What are strings?

character arrays

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]);
}
```

[9-5]

Printing character array with loop

[9-5]

Now, let's print character array with null character

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

Null Character → [\0] → 0

[9-6] - [9-10]

Creating string

Is grouping of characters into array enough? X [no]

↓

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

↓

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

same Result

[9-11]

[9-12] - > the power of null character

[9-13] -> computer puts null character automatically

[9-14] -> computer doesn't put null character automatically

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'};
```

[9-14] -> computer doesn't put null character automatically

Initializing a string

Assigning string literal without size

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

[there is a null character present]

What is the size of str? → 14 = 13 + 1

do the bellow page first

Initializing a string

Assigning string literal with size:

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

13 + 1

[9-15] -> Assigning string literal with size

now,go to the upper page

Think...

Is the following code snippet correct?

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

[it's size is 14]

↓

[error]

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;
```

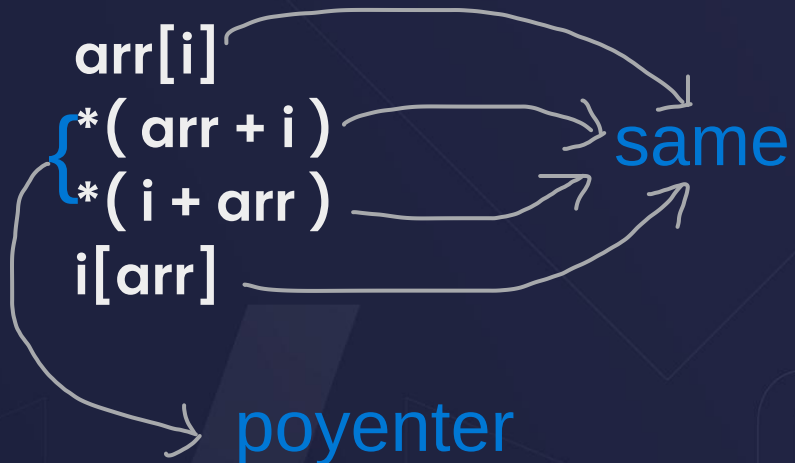
char str[] = “Physics Wallah”;

Predict str after:

(a) str[0] = ‘M’; → Mhysics Wallah

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

Ways of printing particular element



[9-16]

Input and Output of string without loop

gets ( ) , puts ( )

"%c" → character

[9-17]-[9-20]

Can we use string (character arrays) as pointers?

College Wallah \0
↑
ptr

char str[] = “PhysicsWallah”;
char *ptr = str;

```c
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

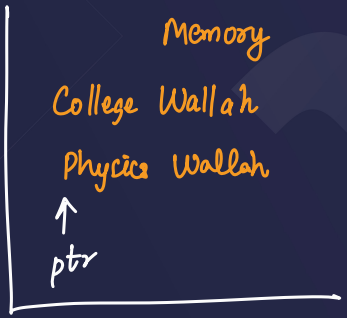
Interesting thing about such initialization

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

In normal initialization, 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);
```

College Wallah

str

Physics Wallah

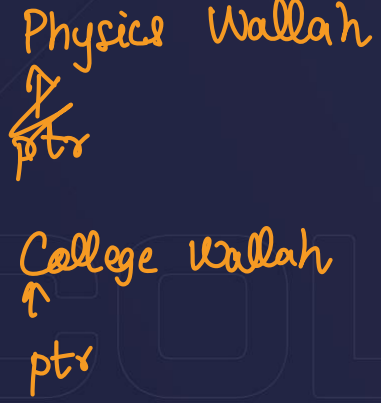
ptr

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

Interesting thing about such initialization

```
char *ptr = “Physics Wallah”;
printf(“%s\n”, ptr);
ptr = “College Wallah”;
printf(“%s”, ptr);
```

Works perfectly!



Why does this happen?

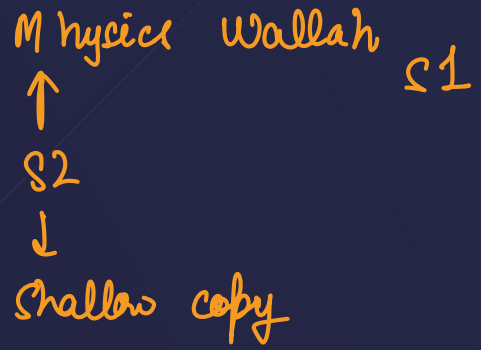
Pointers change the address to which they point after initialising a new character array!

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

Implement: Copy one string to another

```
char s1[] = “Physics Wallah”;
char *s2 = s1;

// Let’s change in s1.
s1[0] = ‘M’;
printf(“%s”, s2);
```



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[] =
```

Making our tasks easy!

s1 = "RaghavGarg";
s2 = "Garg";
strcat(s1,s2);

Useful functions for C already in standard library!

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

Q11

Inserting a character in a string

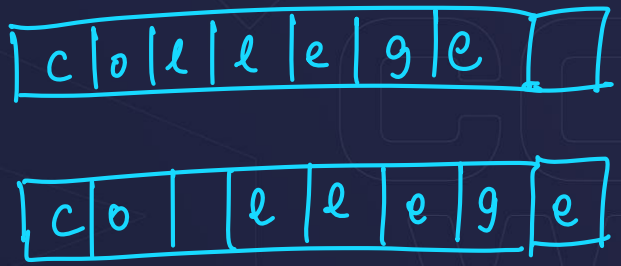
Write a function to insert a new character in a string at a given position.

index

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

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

2nd index → 'l' → Colllege



Reverse a string

take input & print reverse