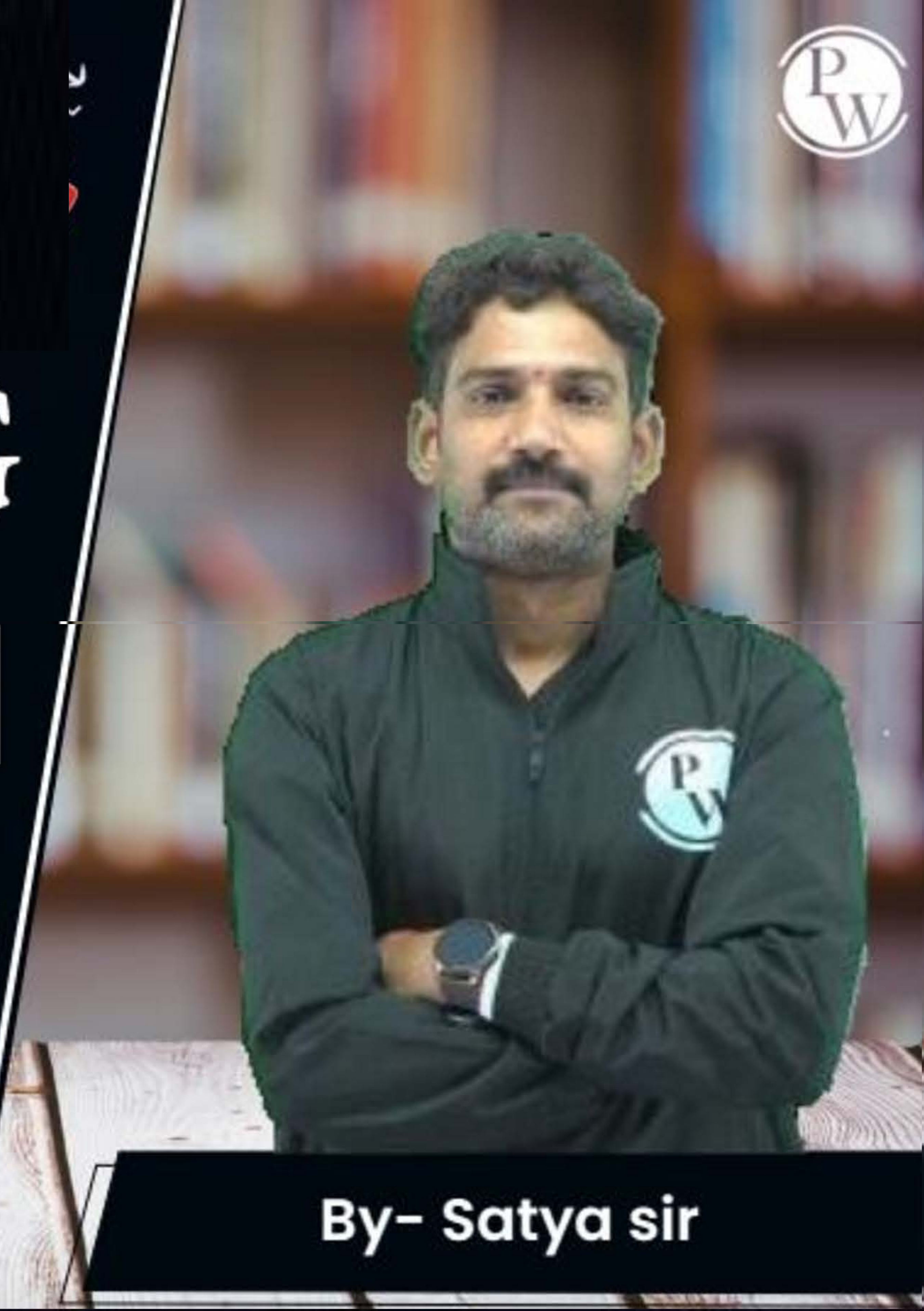
CS & IT ENGINEERING

C Programming

Pointers & Arrays

Lecture No.- 04



By- Satya sir

Recap of Previous Lecture

- Find address of 1-D array Element
- Strings : group of characters
  - → As character array
  - → As character Pointer
- Array vs Pointer

Topics to be Covered

- String handling functions in 'C'
- 2-D arrays
    - Initializatio, Declaration
    - accen elements
    - Address of an Element

Topic : String Handling in 'C'

No.	Function	Description
1)	strlen(string_name)	returns the length of string name.
2)	strcpy(destination, source)	copies the contents of source string to destination string.
3)	strcat(first_string, second_string)	concats or joins first string with second string. The result of the string is stored in first string.
4)	strcmp(first_string, second_string)	compares the first string with second string. If both strings are same, it returns 0.
5)	strrev(string)	returns reverse string.
6)	strlwr(string)	returns string characters in lowercase.
7)	strupr(string)	returns string characters in uppercase.
8)	strstr(str1, str2)	It returns a pointer to the first occurrence of the given substring str2 within the given string str1

Topic : String Handling in ‘C’

No.	Function	Description
9)	strncmp()	It compares two strings only to n characters.
10)	strncat()	It concatenates n characters of one string to another string.
11)	strncpy()	It copies the first n characters of one string into another.
12)	strchr()	It finds out the first occurrence of a given character in a string.
13)	strrchr()	It finds out the last occurrence of a given character in a string.
14)	strnstr()	It finds out the first occurrence of a given string in a string where the search is limited to n characters.
15)	strcasecmp()	It compares two strings without sensitivity to the case.
16)	strncasecmp()	It compares n characters of one string to another without sensitivity to the case.

Topic : String Handling in 'C'

Examples

1) Char x[ ] = "C PROGRAMMING";

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
X	C		P	R	O	G	R	A	M	M	I	N	G	\0

Sizeof(x) ⇒ 14

NOTE: Escape sequences are Not Printed

2) Char ch[10] = "GATEWALLAH";

Printf("%s", ch);

G	A	T	E	W	A	L	L	A	~~H~~ \0

⟵ 10 Bytes ⟶

a) GATEWALLAH
b) GATEWALLA ✓
c) GATE
d) GATEWALLA\0

Topic : String Handling in 'C'

Examples

3) char x[ ] = "GATE EXAM";

0	1	2	3	4	5	6	7	8	9
G	A	T	E		E	x	~~A~~ \0	M	\0

$\underbrace{\qquad\qquad\qquad\qquad}_{\text{10 characters}}$

Printf(" %d", sizeof(x)); //10

Printf(" %d", strlen(x)); // 9 (Excluding NULL)

x[7] = '\0';

Printf(" %d, %d" $\underset{10}{\underline{\text{sizeof(x)}}}$, $\underset{7}{\underline{\text{strlen(x)}}}$);

a) 10, 9

b) 9, 9

c) 9, 7

[✓ d) 10, 7]

- Sizeof ( ) returns memory allocated in Bytes
- Strlen( ) returns index of first NULL character

Topic : String Handling in 'C'

4) char X[10] = "GATE";

0	1	2	3	4	5	6	7	8	9
G	A	T	E	\0	\0	\0	\0	\0	\0

```
Printf("%c", X[6]); // No output
```

5) char *S1 = "GATE", *S2 = "MATE", *S3 = "DATE";

```
Printf("%d %d %d", sizeof(S1), sizeof(S2), sizeof(S3));
```

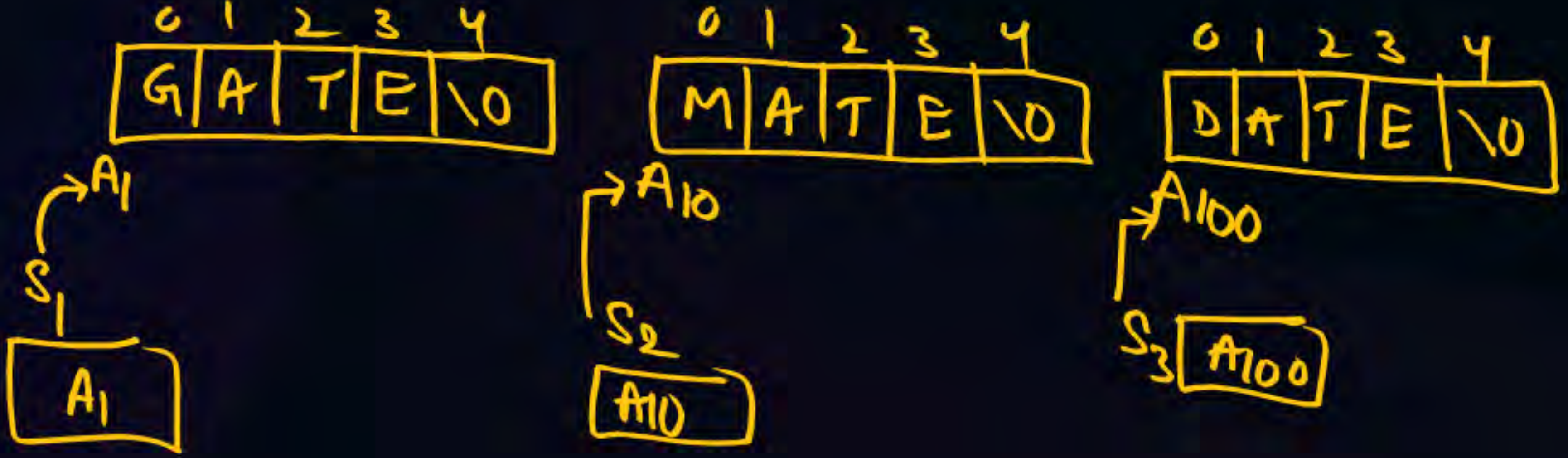
Let $\underset{4\ \text{Bytes}}{\underline{32\text{-bit}}}$ Processor

a) 4, 4, 4 ✓

b) 5, 5, 5

c) 2, 2, 2

d) 4, 6, 4



Topic : 2-D Arrays

Let 32-bit Processor

6)
```
Char *S1 = "ABCD";
Char S2[] = "ABCD";
```
$\text{Printf("\%d, \%d, \%d", } \underbrace{\text{sizeof(S1)}}_{\text{4Bytes}}, \underbrace{\text{sizeof(S2)}}_{\text{5 Bytes}}, \text{sizeof}\underbrace{\text{("ABCD")}}_{\text{5Bytes}}\text{);}$

a) 4, 5, 4

b) 4, 5, 5 ✓

c) 4, 4, 4

d) 4, 4, 5

GATEMATEGATE

```
Char *S1 = "~~GATE~~",   *S2 = "~~DATE~~",   *S3 = "~~MATE~~";
                            GATE                MATEGATE
strcpy(S2,S1);      S2 <— S1

Strcat (S3,S2);
Strcat (S1, S3);
                                 12            4             8
Printf (" %d, %d, %d", strlen(S1), strlen(S2), strlen(S3));

Printf ("\nS1= %s \n  S2= %s\n  S3= %s", S1, S2, S3);
```

o/p: 12, 4, 8

GATEMATEGATE
GATE
MATEGATE

```
Char *S1 = "GATE";
Char *S2 = "DATE";
int i;
i = strcmp(S1, S2);
Printf("%d", i);   // 3
```

$S_1 - S2 \Rightarrow \left(\sum \text{ASCII values of Each Character in S1} \right) - \left(\sum \text{ASCII values of Each Char in S2} \right)$

$$S1 = \overset{'G'}{71} + \overset{'A'}{65} + \overset{'T'}{84} + \overset{'E'}{69}$$

$$S2 = \overset{'D'}{68} + \overset{'A'}{65} + \overset{'T'}{84} + \overset{'E'}{69}$$

$$\boxed{S1 - S2 = 3}$$

Means ⇒ $S1 - S2$

Strcmp(S1, S2) ⇒ Compares given 2 strings and return

- Positive value (>0) when $S1 > S2$
- Zero ($=0$) $S1 == S2$
- Negative value (<0) when $S1 < S2$

Topic : 2-D Arrays

2-D arrays : Also known as Matrix.

- Only For representation, row, column are Used.
- But, in Memory, they get stored in linear manner.

- Declaration

Syntax: datatype arrayname[No.of rows] [No.of cols];

Ex: int arr [3] [4];

Topic : 2-D Arrays

Initialization of 2-D array Elements

datatype arrayname [rows] [cols] = { values }; (OR) datatype array [rows] [cols];

⋮

array [row index] [col index] = value;

Ex: int A [2][2] = { 11, 12, 14, 19 };

(OR)

int A[2] [2];

⋮

A[0][0] = 11;
A[0] [1] = 12;
A[1] [0] = 14;
A[1] [1] = 19;

NOTE: Assignment of values always will be row-wise

Topic : 2-D Arrays

```
int x[3][4] = {10, 20, 15, 25, 18, 28, 30, 40, 60, 70, 1, 5};
```

(OR)

```
int x[3][4] = { {10, 20, 15, 25}, {18, 28, 30, 40}, {60, 70, 1, 5} };
```

x	col 0	col 1	col 2	col 3
Row 0	10	20	15	25
Row 1	18	28	30	40
Row 2	60	70	1	5

Topic : 2-D Arrays

```
int x[3][4] = {5,10,15,20,25,30,35}; ⇒
```

x	0	1	2	3
0	5	10	15	20
1	25	30	35	0
2	0	0	0	0

```
int x[3][4] = {{5,10,15},{20,25},{30,35}};
```

x	0	1	2	3
0	5	10	15	0
1	20	25	0	0
2	30	35	0	0

2 mins Summary

- if All values are Initialized, Then either row dimension (or) col dimension may be Omitted.

```
int x[2][ ] = {5,10,15,20,25,30};
(OR)
int x[ ][3] = {5,10,15,20,25,30};
(OR)
int x[2][3] = {5,10,15,20,25,30};

int x[ ][ ] = {5,10,15,20,25,30}; //Error
```

options: 1x6 array
6x1 array
2x3 array
3x2 array

} Compiler Cannot Decide what dimensions to Consider

- String handling in 'C'
- 2-D arrays
  - Declaration
  - Initialization

To be Contd ...

THANK - YOU