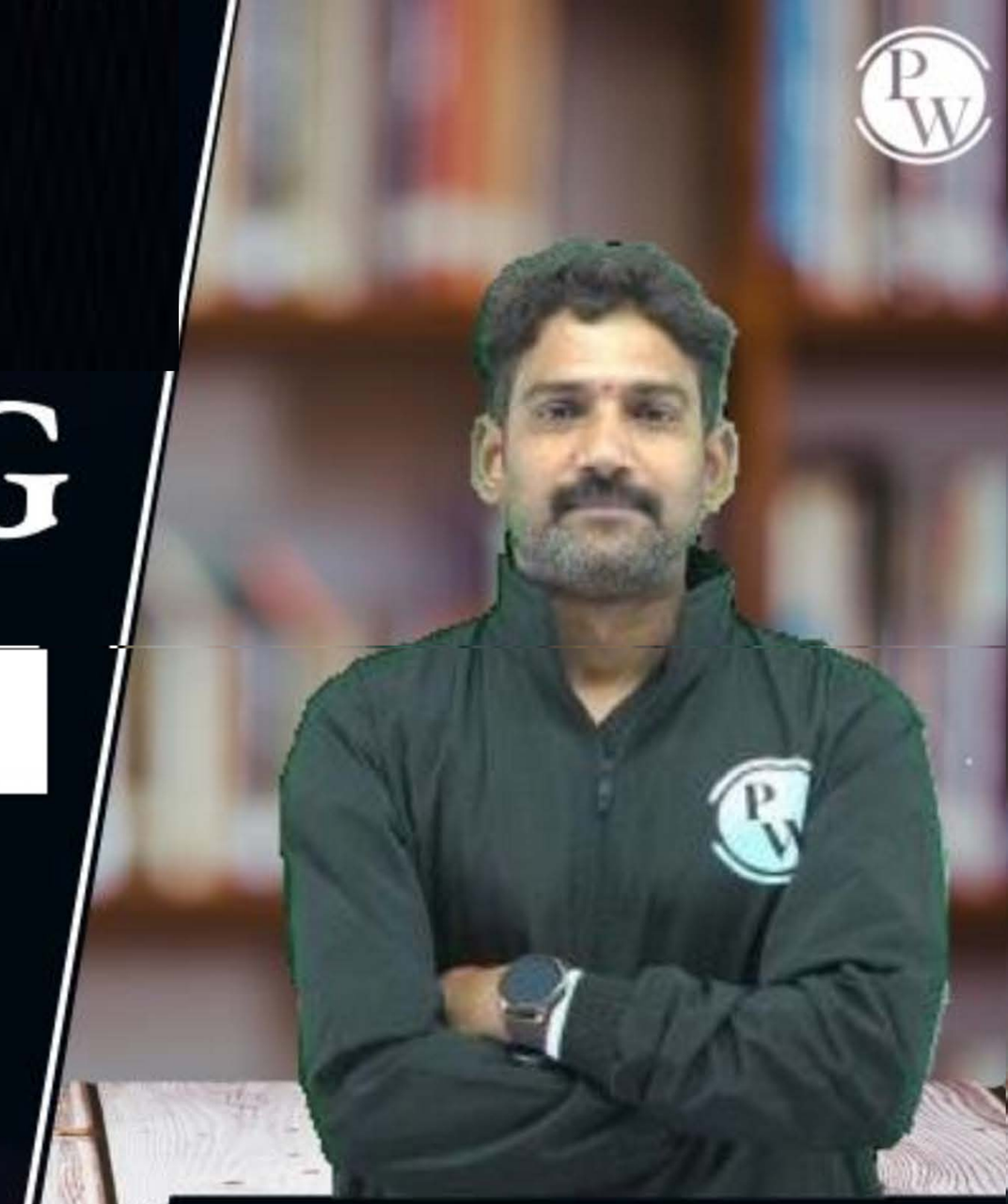
CS & IT ENGINEERING

C Programming

Pointers & Arrays

Lecture No.- 03



By- Satya sir

Recap of Previous Lecture

1-D array

- Declaration
- Initialization
- Base address, index
- How to input & output 1-D array

Topics to be Covered

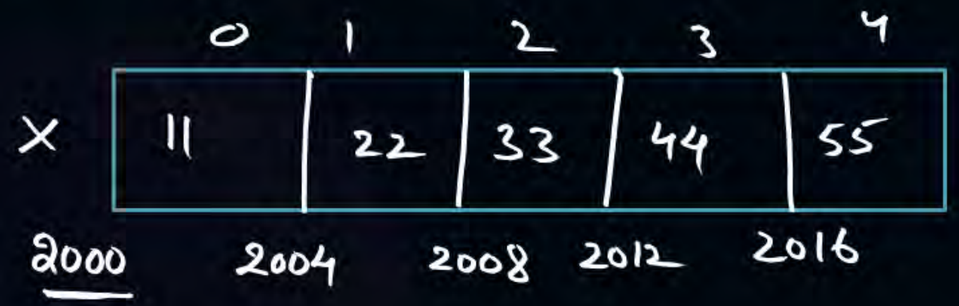
- 1-D array Programs
- 1-D array with Pointers
- String handling in 'C'

Topic : String Handling in 'C'

In arrays, default addressing from zero, but it is Not Mandatory.

Ex: `int x[5] = {11,22,33,44,55};`

Let 1 int = 4 Bytes



$\&x[3] \Rightarrow 2012 \Rightarrow 2000 + 3*4$

$\&x[1] = 2004 \Rightarrow 2000 + 1*4$

$\&x[0] = 2000 \Rightarrow 2000 + 0*4$

`int x[2016];` Let Base add = 2000

$\&x[1047] = ?$

$$\&x[i] = B + i * n$$

Indexing from Zero.

B = Base address

i = index

n = size of each Element (or) scale factor

Topic : String Handling in 'C'

Let indexing from -2

```
int x[5] = {11, 22, 33, 44, 55};
```

	-2	-1	0	1	2
X	11	22	33	44	55
	2000	2004	2008	2012	2016

$\&x[i] = B + i * n$ ⇒ only Indexing from zero

$\&x[0] = 2000 + 0 * 4 = 2000$

$\&x[-2] = 2000 + -2 * 4 = 2000 - 8 = 1992$ } wrong

General formula

Indexing from 'k'

$\&x[i] = B + (i - k) * n$

Indexing from -2

$\&x[-2] = 2000 + [-2 - (-2)] * 4$

$= 2000 + (-2 + 2) * 4$

$= 2000 + 0 * 4 = 2000$ ✓

$\&x[1] = 2000 + [1 - (-2)] * 4$

$= 2000 + (1 + 2) * 4$

$= 2000 + 12$

$= 2012$ ✓

Topic : String Handling in 'C'

```
int x[5] = {10,20,42,64,73};
```

Let Base address = 2000, 1 int = 4 Bytes

```
Printf(" %d", x);
```

a) 10, 20, 42, 64, 73

b) 10

c) 2000

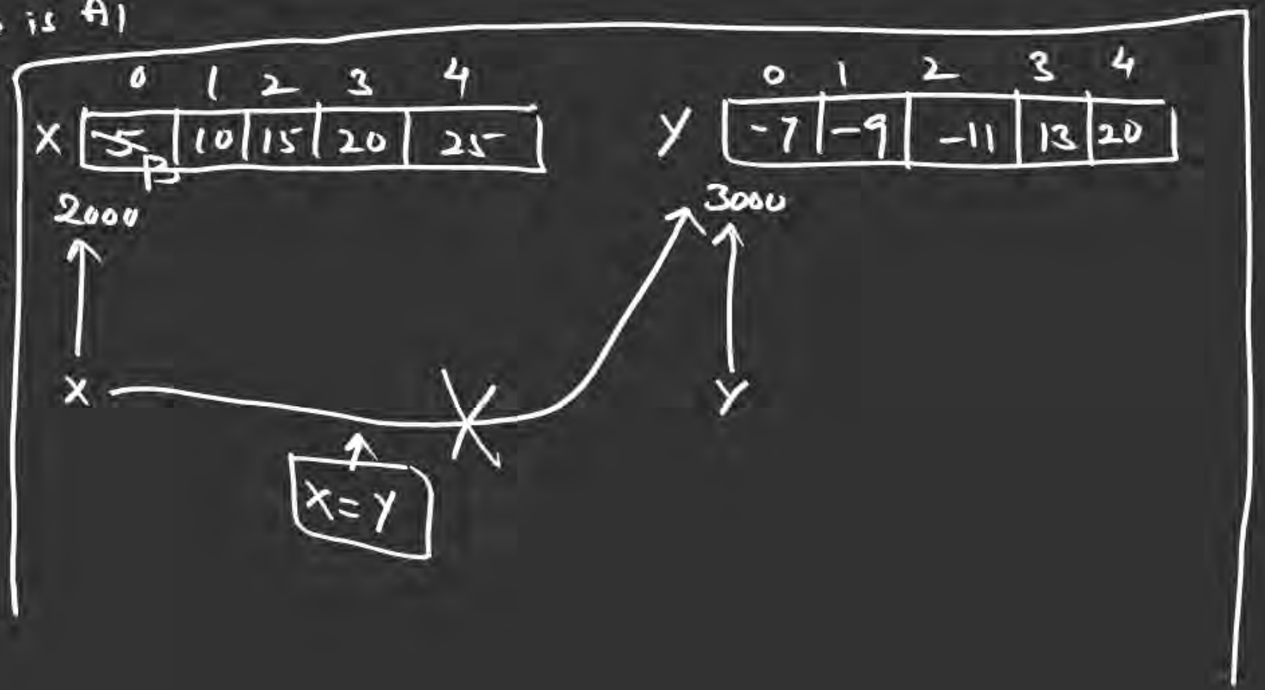
d) None

```
int i=3;
Printf("%d", i);   //3
```

NOTE: Every array is a Constant Pointer

Pointer Vs Array ⇒ Pointer is Variable, Array is Constant.


```
int i=5, j=7, k=9;
int *p; // Variable in Nature
int x[5]={ 5,10,15,20,25}, y[5]={-7,-9,-11,13,20};
   p=&i;  // P Pointing to i P value is A1
   p=&j;  // P Pointing to j, A10
   p=&k;  // P Pointing to k, A20
// x always Points to Same address.
   x=y; // Error
   x[0]=y[3]; // Valid
```



2) Array must be initialised while declaration, but Pointer can be initialized at any time.

```
int x[5];
x[ ]={10,20,30,40,50}; // Invalid

int i=5,*p;
  p=&i;     (OR)    int i=5, *p=&i;
```

(3) Size of array depends on Number of Elements; But, Pointer size is always integer size.

int *P, *Q;

int x[5] = {11, 10, 5, 9, 7};

q = x; P = x+4;

x[3] (or) P[-1]

	0	1	2	3	4
X	11	10	5	9	7

2000 2004 2008 2012 2016

x (or) x+0

&x (or)

&x[0] (or)

P-4

q = 2000

&x[1] (or)

x+1 (or)

P-3

q+1

&x[2] (or)

x+2 (or)

P-2

q+2

&x[3] (or)

x+3 (or)

P-1

q+3

&x[4] (or)

x+4 (or)

P

q+4

X => Base address (B)

int x[5], *P;

P = x;

&x[i] == X + i

== B + i * n (OR)

== X + i * n

&x[i] == &P[i]

== x+i == P+i

x[i] == i[x] == P[i] == i[P]

== *(x+i) == *(P+i)

* = value at

String : Group of characters.

→ String can be created in 2 ways:

1) As character array Ex: char s[10];

2) As character Pointer Ex: char *s;

- As array means, memory allocated at Compile time and is fixed (Static)
- As Pointer means, memory allocated at run time and is Variable (Dynamic)
- All Strings End with a special character '\0' called NULL character.

Topic : String Handling in 'C'

Pre defined String functions

To be contd ...

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.

2 mins Summary

- Address of an Element in 1-D array
- Pointer vs array
- Strings

THANK - YOU