LECTURE - 31

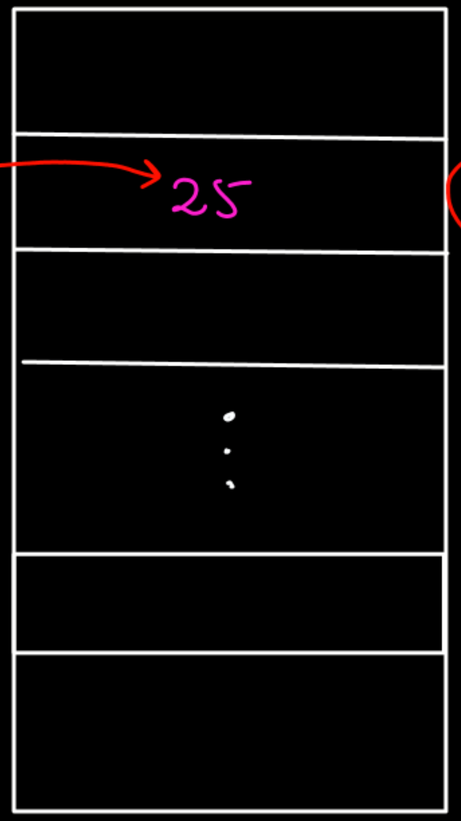
POINTERS IN C (PART 01)

PROGRAMMING IN C

Pointer :

↳ A variable that stores the address of other variable.

```
int x;
x = 25;
printf("%d", x);
```



x ← Variable

int

Value

10110

& ← Address of operator

↑

Scanf( )

```
printf("%d", &x)
```

int format

memory address

```
Scanf("%d", &var)
```

take an int value

Store inside the address of var

```
main( ) {
    int a = 100;
    int address of a = &a;
}
```

normal Variable

4 bytes

normal Variable

Room No.	Size
0000	4
0001	16
0010	4
0011	4
0100	8
0101	8
0110	4
0111	1
1000	1
1001	8
1010	4
1111	4

Every Room has different size.

But the size of the address is same i.e. 4 bits

⇒ The address size and size of Datatype may different

int = 4 bytes

char = 1 Byte

double = 8 Byte

float = 4 Bytes

16 Bytes (64 bit)

0000	0001	0010
double	long Double	double
0011 int	0100 long	0101 long
0110 int	0111 char	1000 char
1001 double	1010 float	1111 int

8 byte

4 Byte

1 Byte

Building

That's why we don't save the address into a normal variable.

↳ To store the address of a variable, we use pointers.

⇒ Address & value at address are different things.

Syntax to declare / define pointer variable:

```
datatype * variableName ; //declaration
datatype *variableName = &anotherVariable ;
```

address of

pointers.c

pointers.c > main()

```c
#include<stdio.h>
int main(){
    int x = 100;
    printf("The value of x is: %d \n", x);
    int *ptr = &x;
    printf("The address of x is %p \n", ptr);
    printf("The size of ptr is %d \n", sizeof(ptr));
    return 0;

}
```

Code

```
The value of x is: 100
The address of x is 000000000061fe24
The size of ptr is 8
PS>
```

Dereferentiation Operator:

- Also Called as 'Value at' operator.
- It access the value present inside the pointer.

```
int m = 250;
int *a = &m;
printf("%d", m);
printf("%d", *a);
```

a = 101

250

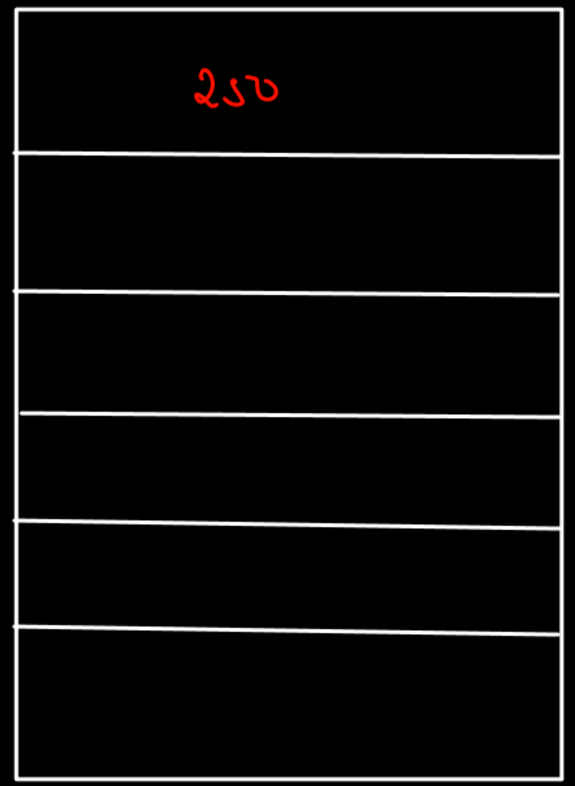
*a = Value at (101)

↳ 250

Value at 'a' → 250

*a = &m

*a = *&m → Value at address of 'm'



```
int main(){
    int x = 10;
    int *ptr = &x;
    printf("%d", x);
    printf("%d", &x);
    printf("%d", *ptr);
    printf("%d", ptr);
    printf("%d", *&ptr);
    printf("%d", *&x);
    return 0;
}
```

printf("%d", x); → 10

printf("%d", &x); → 1001

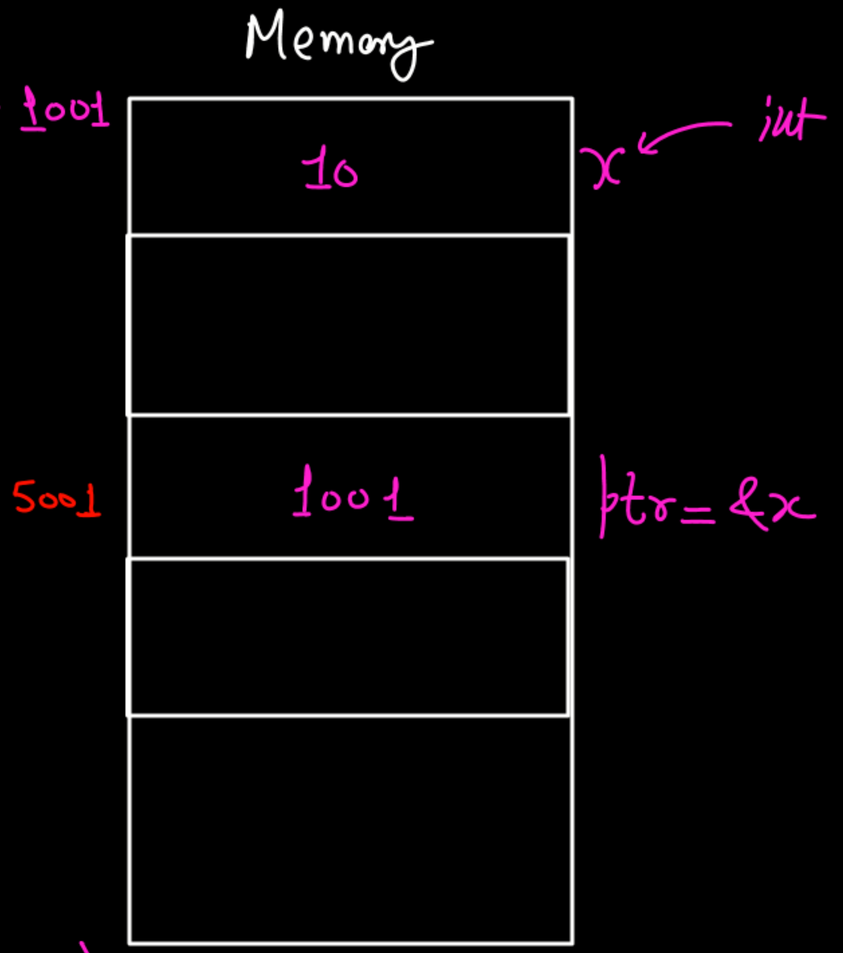
printf("%d", *ptr); → Value at (ptr)
↳ Value at (1001) = 10

printf("%d", ptr); → 1001

printf("%d", *&ptr); → Value at (address of (ptr))
Value at (5001) → 1001

printf("%d", *&x);
↳ Value at (address of (x))
Value at (1001)
= 10

[*& will cancelled out]



pointers.c | pointers2.c | pointer3.c

pointer3.c > main()

```c
#include<stdio.h>
int main()
{
    int a = 50;
    int *b = &a;
    printf("a: %d \n", a);
    printf("b: %d \n", b);
    printf("*&a: %d \n", *&a);
    printf("*&b: %d \n", *&b);
    printf("*b: %d \n", *b);
    printf("b: %p \n", b);
    // printf("*6422060: %d \n", *(000000000061fe2c)); error

    return 0;
}

```

```
a: 50
b: 6422060
*&a: 50
*&b: 6422060
*b: 50
b: 000000000061fe2c
PS>
```