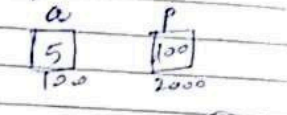
Pointer :-

```
int a = 5;
int *p;
p = &a;
```

int *p = &a;

without using a, changing the value of a

*p = a = 5



Change by address :-

```
void swap(int *, int *);
int main()
{
    int a, b;
    scanf("%d %d", &a, &b);

    swap(&a, &b);

    return 0;
}

void swap(int *p, int *q)
{
    int temp;
    temp = *p;
    *p = *q;
    *q = temp;

    printf("%d %d", *p, *q);
}
```

using main()

```
{
printf("Before calling %d %d", a, b);      5 5
    swap(&a, b);   5 , 3

printf("After calling %d %d", a, b);       5 5
}
void swap(int *p, int q)
{
    int temp;
    temp = *p;
    *p = q;
    q = temp;

    printf("Within function %d %d", *p, q);
}
```

*p = a

★ Glimpse -

→ Value changed by address then change.

* p = a	* a = a
* q = b	* b = b

→ (int *p, int *q) → address

*p = q ⟶ X

address cannot be stored in value variables

Pointers

(a) → normal variable
(*a) → pointer variable

→ *p takes size (8 bytes) of any data type (All online compilers)
Offline may vary b/w

→ But defining data type is imp. because to that type when we p++ or p-- then how much it need to subtract or add (4, 1, 8)

int a = 5;   *j
int p*, *q, **r, **q;
^{1}(p) = (&a)^0; ✓
^{1}(q) = (&p)^0 ✓
^{2}(r) = (& q)^1 ✓

int ***r, *j;
^{2}(**j) = (&j)^1 ✓

(* [ ]) → degree

→ (a)^0 = (b)^0
Same degree value copy.

Pointers (Operation).

```
int a = 5, x;
int *p = &a;
```

① `x = (*p)++;`
`p(x, p, a)`
5, 100, 6

② `x = *(++p);`
`p(x, p, a)`
Garbage, 104, 5

③ `x = *(p++);`
`p(x, a, p)`
5, 5, 104.

④ `x = ++ *p;`
`p(x, p, a)`
6, 100, 6

⑤ `x = *p++;`
`p(x, p, a)`
5, 104, 5

`a = *p;`
`p--;`

*p = a   int *p

int *p;
p = &a;

Array of pointers   *p[5] → p++ → Not allowed ✗
Pointer of an array   (*p)[5]   p++ → Allowed ✓

```
void main()
{
    int a[5];
    int .
    for (int i = 0; i<5; i++)
        scanf("%d", &a[i]);

    for (int i = 0; i<5; i++)
        p[i] = a + i;      [Printing address]
                           [Printing value]
        *p[i] → (*(p+i)).
```