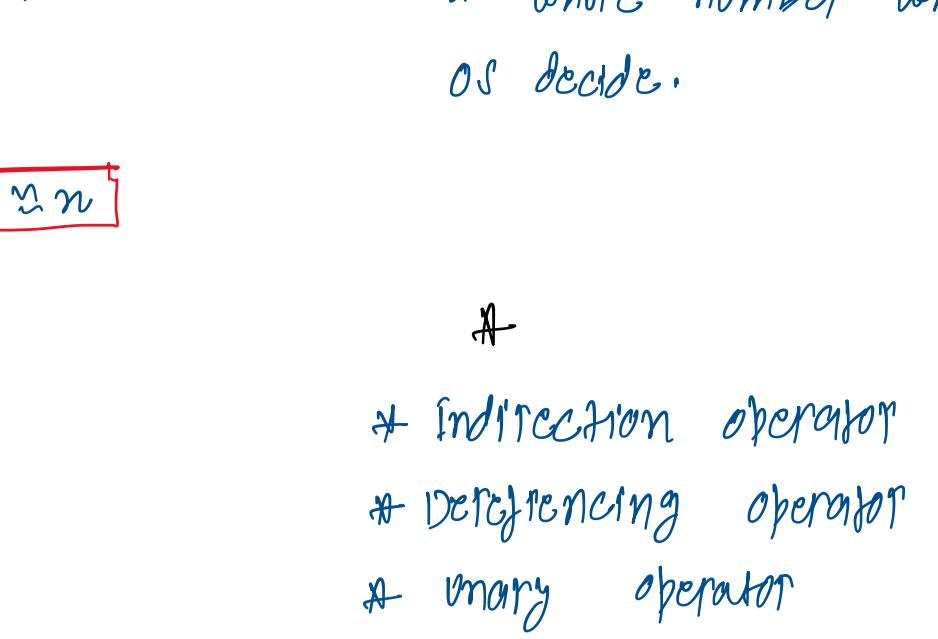


POINTERS

Friday, 5. August 2022 21:58

INTRODUCTION TO MEMORY ADDRESS

```
int n=5;  
char m='A';
```



REFERENCES AND DEREFERENCES OPERATORS

```
int n=5  
printf("%d,%d", &n); → 1000  
printf("%d,%d", +&n); → 5
```

1000 → Address or reference
+ Address would be always
a whole number which
of decide.

*** &n & n**

* Address of operator	* Indirection operator		
* Referencing operator	* Dereferencing operator		
* Unary operator	* Unary operator		
* & variable	* & → address		
int n=5; → error	* &n is not a variable		
&n=7; → &n is just a way to represent	* &n is just a way to represent		
<table border="1"><tr><td>5</td><td>n</td></tr></table>	5	n	address of variable our base is 1000
5	n		
1000	* &n is considered as constant value		

int n=5;

5	n
---	---

 * & n is a pointer variable

int *t;

1000	t
------	---

t = &n;

1000	1000	t
------	------	---

printf("%d,%d,%d", t, &n, t+n);

printf("%d,%d,%d", t, t, t);

1000	1000	1000	t
------	------	------	---

1000	1000	1000	t
------	------	------	---

WHAT IS POINTER?

Pointer is a variable which contains address of another variable.

SIZE OF POINTER

```
int a, *b;  
char b, *q;  
double c, *r;
```

* size of ordinary variable depends on the datatype
* size of pointer variable doesn't depend on its datatype.

BASIC ADDRESS

```
int a, *b;  
char b, *q;  
double c, *r;
```

b = &a *p = a

char *q; q is not going to store character const.
but it's going to store address of char variable.

DATATYPE OF POINTER

```
int a, *b;  
char b, *q;  
double c, *r;
```

b = &a *p = a

a = a p = p

a+1 p+1

p+1 + p+1 → *(&p+1) → *(a+1)

b+1 + b+1 → *(&b+1) → *(a+1)

b+1 + b+1 → *(&b+1