Pointers (P-3)

⊛ Void Pointer

$$\begin{tikzcd} & \text{generic} \\ \text{Void Pointer} \arrow[ur] \arrow[r] \arrow[dr] & \text{unknown} \\ & \text{General Purpose} \end{tikzcd}$$

```
void * ptr;
```

Casting ⟶ Must

Ex:-

```
int x = 5;
void * ptr;
⟹ ptr = &x;
printf("%i", *ptr);   X X
⟹ printf("%i", *(int *)ptr);
```

⊛ Pointer to func :-

⟹ declaring a pointer to func :-

Syntax

return-type (* ptr) Argu

Ex

① `void (*ptr) (void);`

② `void (*ptr) (int)`

③ `int (*ptr) (int, int);`

---

ptr = func; $\rightarrow$ $\boxed{\text{Name}}$

Calling $\Longrightarrow$ ① `x = ptr(✓, ✓);`   `// x=func(✓,✓);`

② `x = (*ptr)(✓, ✓);`

---

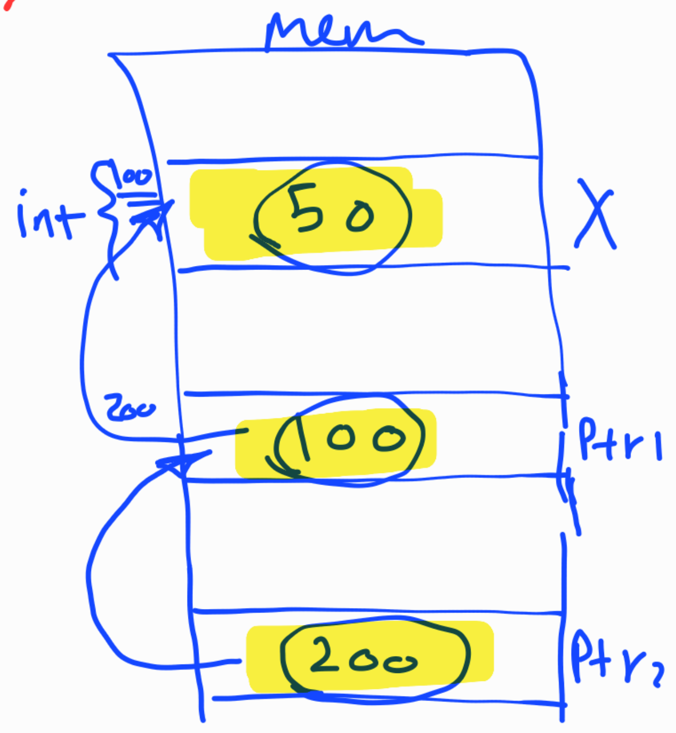
* Pointer to Pointer

```
int x = 50;

int *ptr1 = &x;

int **ptr2 = &ptr1;
```

Access $\longrightarrow$ `**`

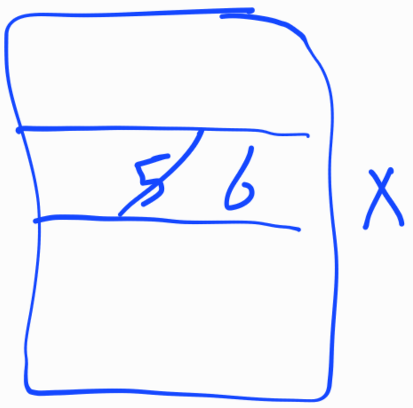


`printf("%i", **ptr2);` $\Longrightarrow$ $\boxed{50}$

```
const int x = 5;

int *ptr = &x;

*ptr = 6;
```

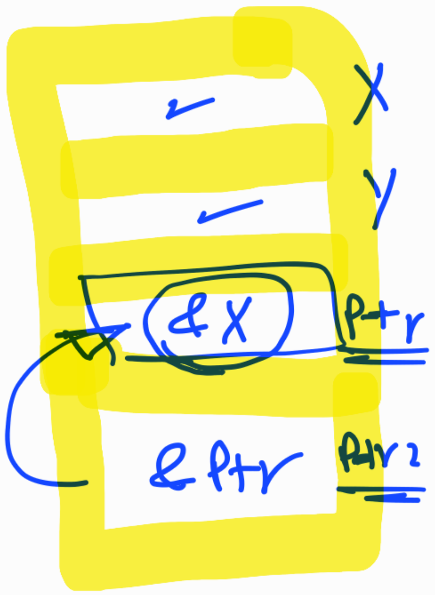


```
int * const ptr = &x;

     ptr = &y        X X

int ** ptr2 = &ptr;

   *ptr2 = &y;
```



EX =

swap

* How to read a complex expression.

Two methods

① Numeration

② SOAC → Spiral outwards anti clockwise

I Numeration → Pencead

P.nce	trans
① ( ) , [ ]	( ) → func ___ takes ___ return ___ ↳ oper [ ] → array of
② * , ident	* ptr ⟹ Pointer to ______
③ data type	______

char (*ptr) [5] ;

* Ptr is a pointer to an array of 5 elements of char datatype.

$\boxed{2}$ SOAC

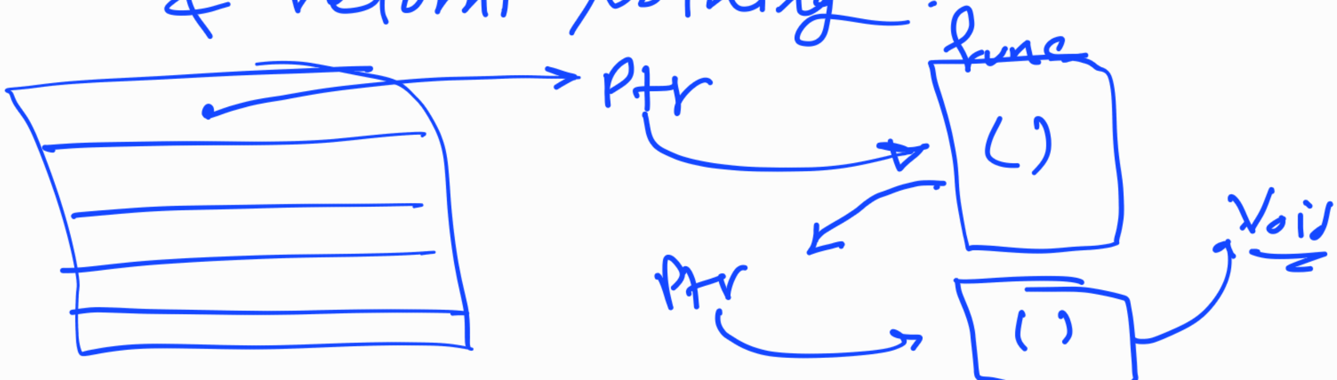
```
char (*ptr) [5]
```

Ptr is a Pointer to an array of 5 elements of char data type.

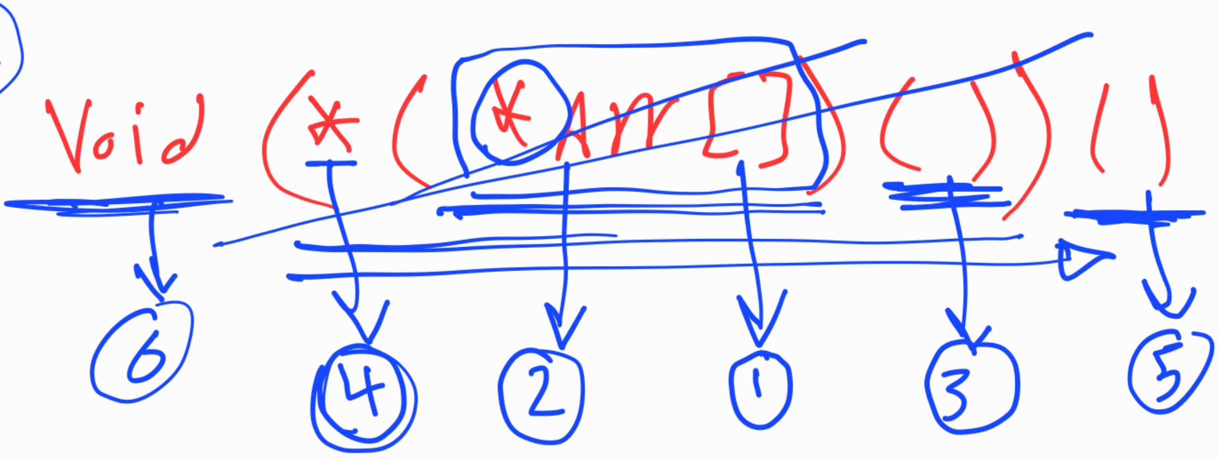
$\boxed{1}$

```
void (*(*Arr[-])(-))(-)
```

Arr is an array of Pointers to func— takes void & returns Pointer to func takes Nothing & returns Nothing.



②



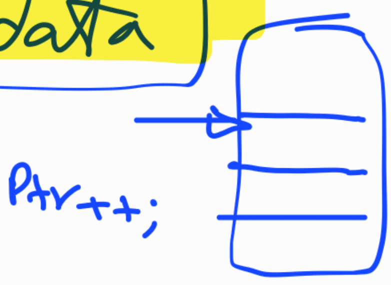
Arr is an Array of Pointer to func take —— & return Pointer to func tn— & ret— Noth.

---

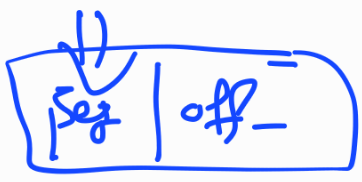
hint → Pointers in ES

→ Stream of data

4B by 4B



+ Another types of Pointers:-



→ (Near & far & huge)

Near → 2 bytes

far → 4 bytes

huge → 4 bytes