## Outline

- Pointer Recap
- Operations on a pointer
- Pointers to Pointers
- Array of pointers
  - ↓ vs Multidimensional arrays.
- Structures.
- Typedef.

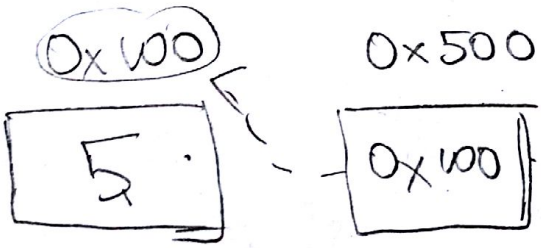
---

```
int a = 5.
int *pa = &a;
```

→ address of

* - Dereference operator.

Value_at_address operator.




```
printf("%p", &a) → 0x100
printf("%p", pa) → 0x100
printf("%d", *pa) → 5.
```

* pa

↓

value at (0x100)

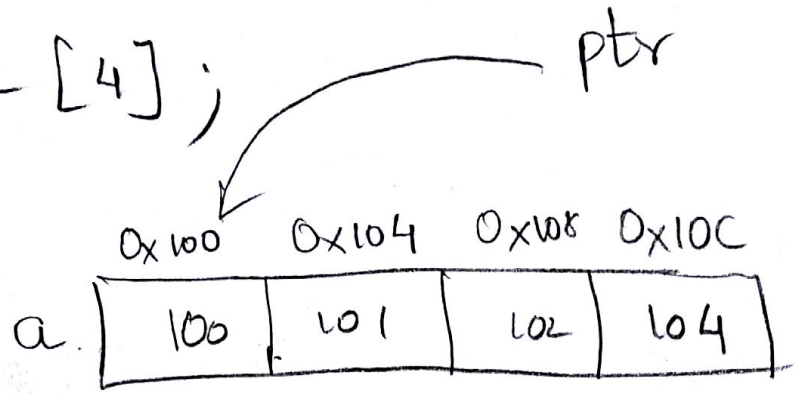
→ 5

```
*pa = 10 // change the value at 0x100 to 10.
```

Scanned by CamScanner

Pointers & Arrays

int a[4];




a[0] a[1] a[2] a[3].

int *ptr = a;

(or)

int *ptr = &a[0];

Name of the array is a synonym of the location of the first element of the array.

printf("%d", *ptr);

=> 100.

*ptr++;

printf("%d", *ptr);

=> 101.

*(ptr+1)

ptr + (1 * sizeof(*ptr))

0x100 + (1 * 4)

0x104.

ptr = ptr + 2

ptr = ptr + 3

Scanned by CamScanner

## Operations on a pointer

```
xptr = &x;
xptr = yptr;
xptr++
xptr--
xptr + 5
xptr = NULL   } same thing.
xptr = 0
```

---

## Not allowed (Not advised)

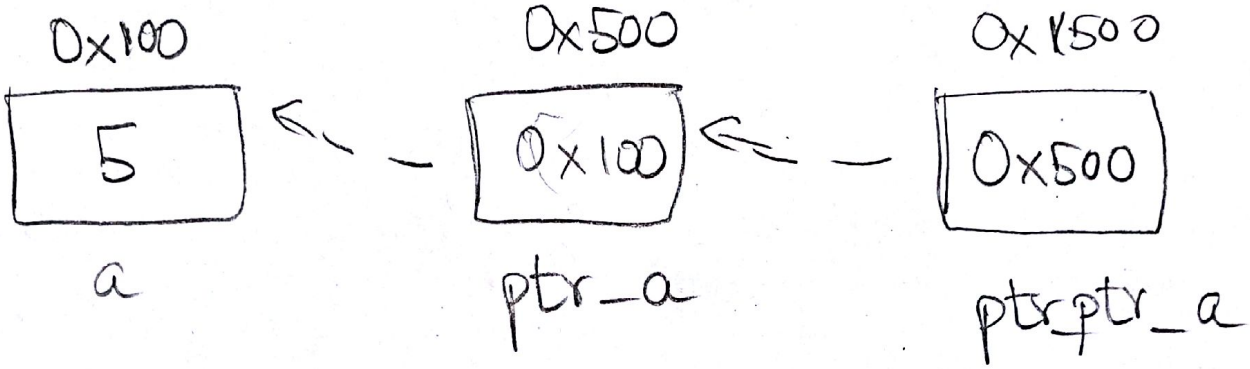
```
xptr = 0x5001.
xptr = yptr + zptr
xptr = yptr * 3
```

---

Scanned by CamScanner

# Pointer to a Pointer

We can make a pointer variable store the address of another pointer variable.




int **ptr_ptr_a = &ptr_a;

("%d", a) → 5

("%d", *ptr_a) → 5.

value at (0x100)

("%d", **ptr_ptr_a)

* (value at 0x500)

(value at 0x100)

5.

Scanned by CamScanner

# Array of pointers

char *names[10];  names[0] = "Jim";

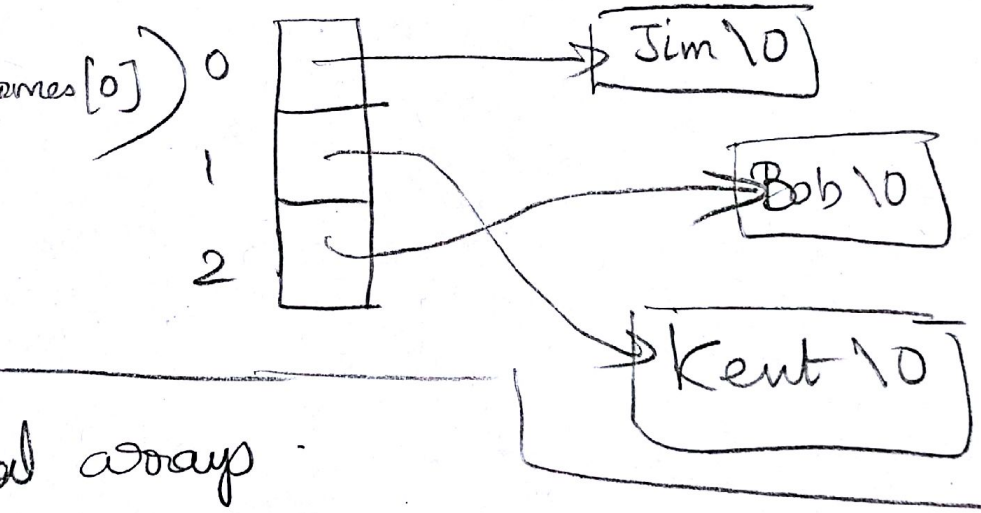
↓

names is an array of 10 elements, each element of which is a pointer to a char.

char *names[] = {"Jim", "Kent", "Bob"};

printf("%s", names[0])

⇒ Jim.




## Multidimensional arrays

int matrix[2][4] = {1, 2, 3, 4 .. 8};

matrix[0][0] = 1;
matrix[0][1] = 2;

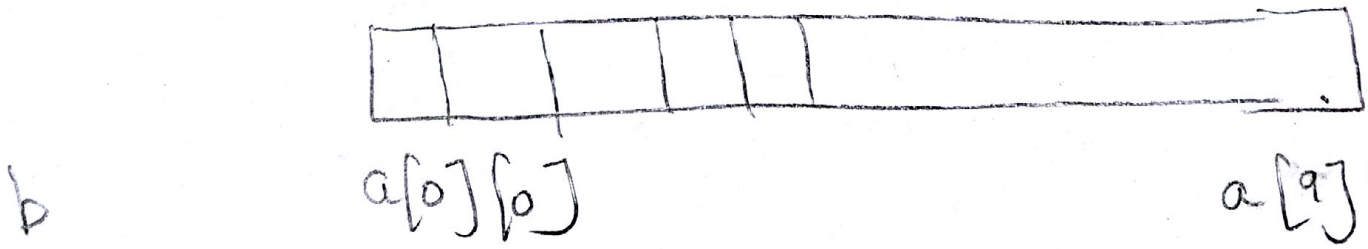
Scanned by CamScanner

# Array of Pointers vs Multi-dimentional arrays.

```
int a[10][20];
int *b[10];
```

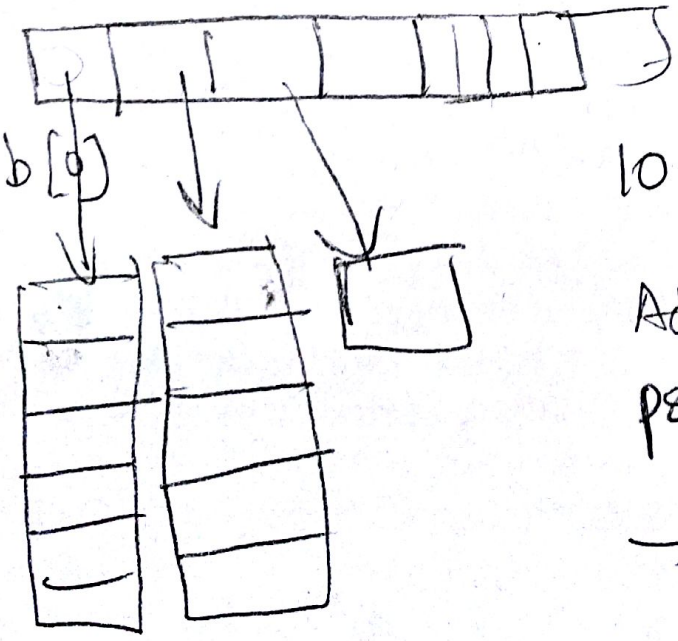
a[5][6] and b[5][6] are syntactically the same thing. (refer to some single integer)

But a → true 2D array




a[row][col]

a + (20 × row + col).




Advantage of using pointer arrays

→ rows can be different lengths

Scanned by CamScanner

# Command line arguments.

① `int main( )` // One way to start declaration.

② `int main (int argc , char * argv[] )`

argc: number of white space strings on the command line

argv: an array of pointers each of which points to an argument

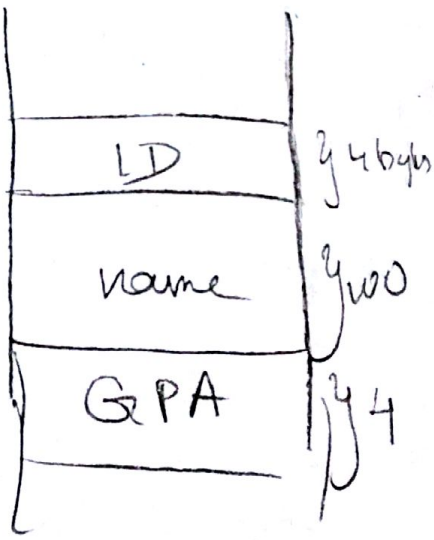
./sizep  argc = 1

---

## Struct

→ a derived type that groups together variables of different types.

```
struct student {
    int ID;
    char name[100];
    float GPA;
};
```




Scanned by CamScanner

Struct student dale ;

```
dale.ID = 1234
dale.name = "Dale Carnegie";
dale.GPA = 3.95;
```

---

typedef → define a new type.

```
typedef unsigned long long
        int BIG_NUMBER;
BIG_NUMBER x = 1000 ... ;
```

---

```
typedef struct student stdnt;
stdnt dale; // Just use 'stdnt' to
            define a variable
            instead of
            'struct student'
```

---

Scanned by CamScanner