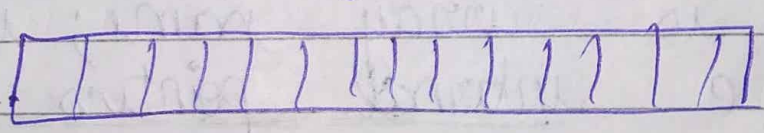
C_48 ⇒ Arrays in C - Part 3

Memory Representation and accessing of array elements

```
int a[5] = {1, 10, 0, -1, 3};
```

×4 ⇒ 20 bytes.



int = 4 bytes

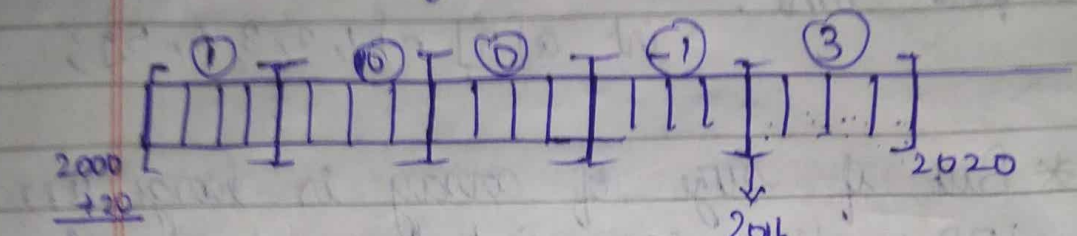
↳ Bytes of memory.

Memory allocation & representation

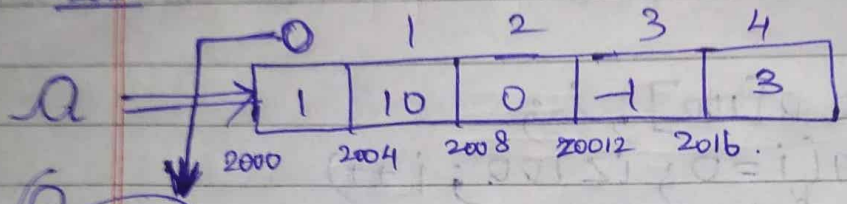
So memory is allocated in a continuous memory.

int a[5] = {1, 10, 0, -1, 3}

Ex: 20 bytes are in continuous order.



2000
+20
2020



Array Index always starts with 0

Accessing of Array :-

* We can access any element of array at constant time.

Formula => Base address + (index × size of int)

a
2000

* 'a' stores address of first element or base address.

* 'a' is array name; which acts as a internal pointer variables.

* Pointers are used to store address of other variables.

To access first element a[0] = 1

To access fourth element a[3] = -1

* So, we can access any random element of array and address is also calculated with base address.

B + (index × size of int)

5th element ⇒ 2000 + (4 × 4)
⇒ 2000 + 16
⇒ 2016

*So we can access any element at constant time (or) order of one time O(1).

Designated Initialization :-

* int arr[10] = {1, 0, 0, 0, 0, 2, 3, 0, 0, 0};

(or) int arr[10] = {[0] = 1, [5] = 2, [6] = 3};

(or) int arr[10] = {[5] = 2, [0] = 1, [6] = 3};

---

* int a[5] = {[0] = 4, [4] = 78}; (Correct)
maximum

Wrong X int a[5] = {[0] = 4, [5] = 78};

5-1 => a[4]

Date
Page

* int a[] = {[2] = 3, [10] = 4, [20] = 5};

↓

Here no length of array; so Compiler will detect the maximum index which is 20; so length is 21 a[21]

* int a[] = {1, 7, 5, [5] = 90, 6, [8] = 4};

≡ Equivalent

int a[] = {1, 7, 5, 0, 0, 90, 6, 0, 4};

* int a[] = {1, 2, 3, [2] = 4, [6] = 45};

≡ Equivalent

a[2]

int a[] = {1, 2, 4, 0, 0, 0, 45};

↓ a[0] ↓ a[1]