

# Need of an Array

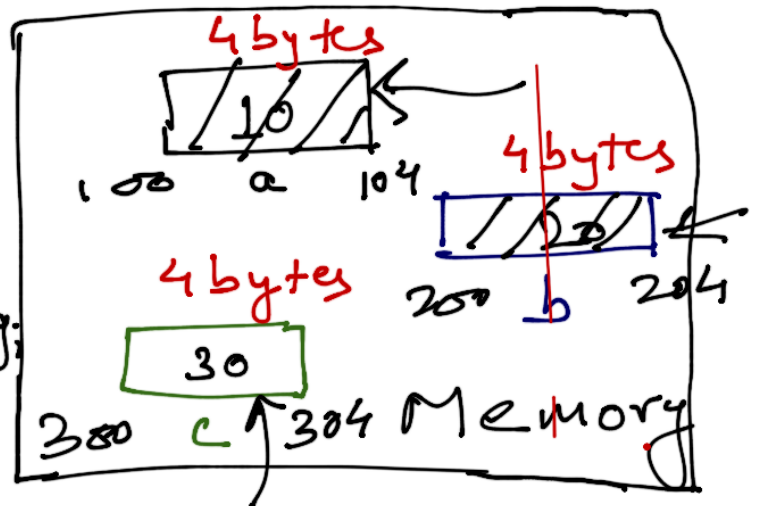
## Example

int a;

int b;

int c; float Avg;

$$\text{Avg} = \frac{a+b+c}{3}$$



Instead of engaging so many variables, better to go for one identifier & organize the data.

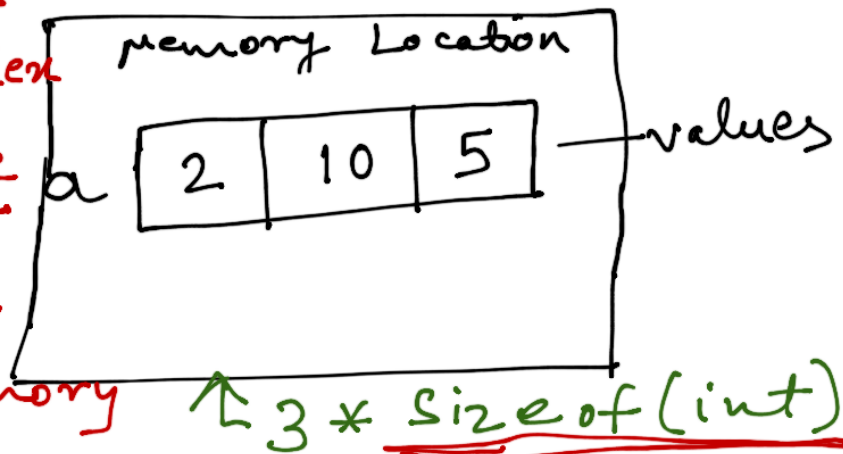
Data type  $\rightarrow$  int    Name of the array  $\rightarrow$  a    No. of elements  $\rightarrow$  3  
 $\text{int } a[3];$

- \* Series of elements
- \* accessible by an index

\* Same data type

\* unique identifier

\* contiguous memory



Are these Arrays ??

a

1	11	7	56	29	99
---	----	---	----	----	----

---

b

c	d	e	f	g	h
---	---	---	---	---	---

---

c

1	d	7	f	29	h
---	---	---	---	----	---

---

# Length of an array

→ It can be specified by any

integer constant expression.

Correct / Incorrect

`int a[5];`

\_\_\_\_\_

`int a[5+5];`

\_\_\_\_\_

`int a[5*3];`

\_\_\_\_\_

`int a[-5];`

\_\_\_\_\_

`int a, int b[a=22/3]`

\_\_\_\_\_

→ Specify length of array using macro.

```
# define N 10  
int a[N];
```

## Initialization of Arrays

Method 1

```
int a[5] = {1, 2, 5, 30, 32};
```

Method 2

```
int a[] = {1, 2, 5, 30, 32};
```

Method 3

```
int a[5];
```

```
a[0] = 1;
```

```
a[1] = 2;
```

```
a[2] = 5;
```

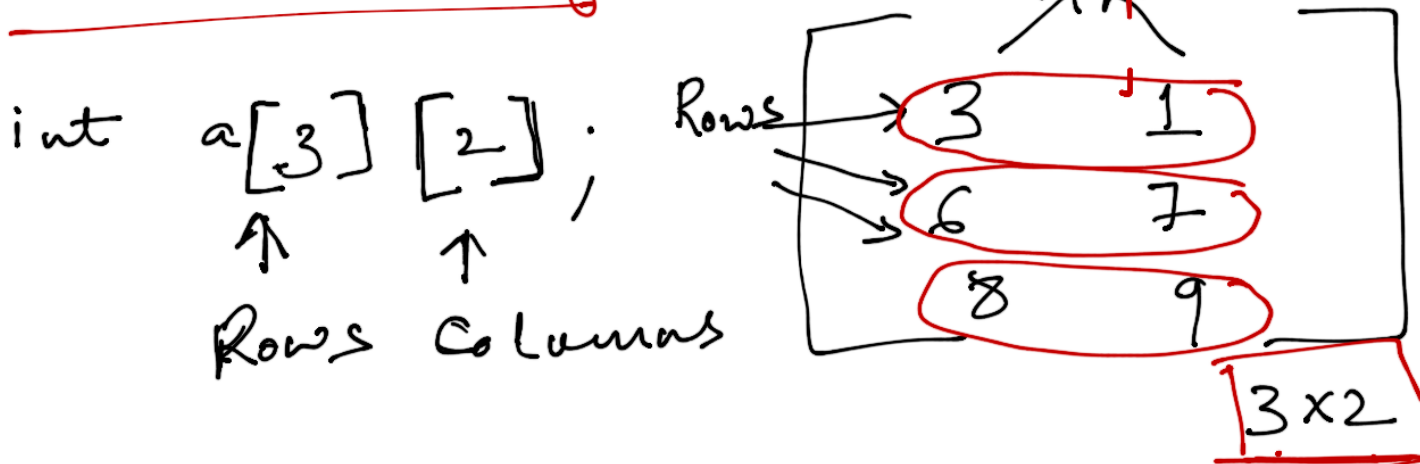
```
a[3] = 30;
```

```
a[4] = 32;
```

## Method 4:

```
int a[5];  
for (i = 0; i < 5; i++)  
{  
    cout << "Enter the value";  
    cin >> a[i];  
}
```

## 2-D Array



⇒ Array of Arrays

int a[3][2] = {3, 1, 6, 7, 8, 9}

↓ i j →

	0	1
0	a[0][0]	a[0][1]
1	a[1][0]	a[1][1]
2	a[2][0]	a[2][1]

```
int a[3][2] = { {3,1}, {6,7}, {8,9} };  
int a[][2] = {  
int a[3][] = X..
```

```
int a[3][2], i, j;  
cout << "Enter the elements";  
for (i=0; i < 3; i++)  
{  
    for (j=0; j < 2; j++)  
    {  
        cout << "Enter a element";  
        cin >> a[i][j];  
    }  
}
```

Diagram illustrating the memory layout of the 2D array `a` (3 rows, 2 columns). The indices `i` and `j` are shown in red above the table.

	0	1
0	3	1
1	6	7
2	8	9

1) Row - Major

2) Column - Major

Row Major

0th Row

1st Row

2nd Row

3 <sup>100</sup>	1 <sup>104</sup>
6 <sup>108</sup>	7 <sup>112</sup>
8 <sup>116</sup>	9 <sup>120</sup>

	3	1	6	7	8	9	
	100	104	108	112	116	120	

Column Major

	3	6	8	1	7	9	
	100	104	108	112	116	120	

	3 <sup>100</sup>	1 <sup>104</sup>
	6 <sup>108</sup>	7 <sup>112</sup>
	8 <sup>116</sup>	9 <sup>120</sup>

# Accessing Elements in an Array

## for 1-D Array

0	1	2	3
4	10	56	16

$$\text{Base Address}_i = \text{Base Address}_0 + i \times \text{Size of data type}$$

$$= 100 + 2 \times 4$$

for

$$\underline{i=2} = 100 + 8$$

$$= 108$$

## for 2-D Array

1) Row-Major

$$\text{Base Address}_{ij} = \text{Base Address}_0 + ((i \times n) + j) \times \text{Size}$$

where,  $m \times n$  represents size of the array



2) Column - Major

$$\text{Base Address}_{ij} = \text{Base Address}_0 + ((j \times m) + i) \times \text{Size}$$