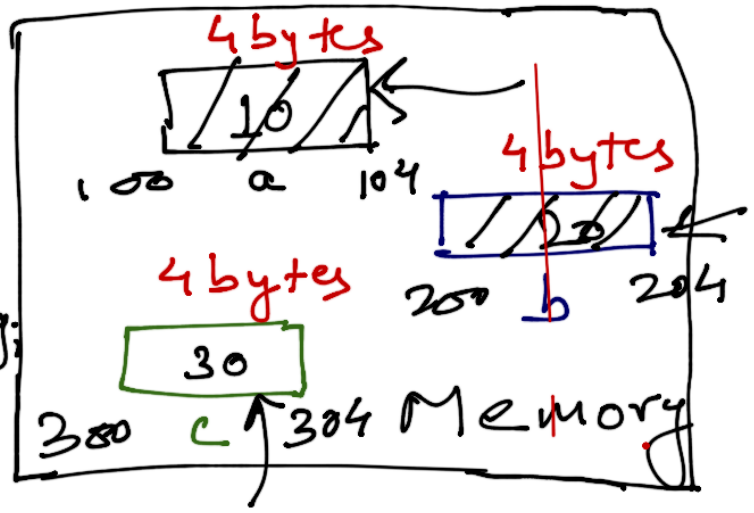
Need of an Array

Example

```
int a;
int b;
int c; float Avg;
Avg = a+b+c / 3
```

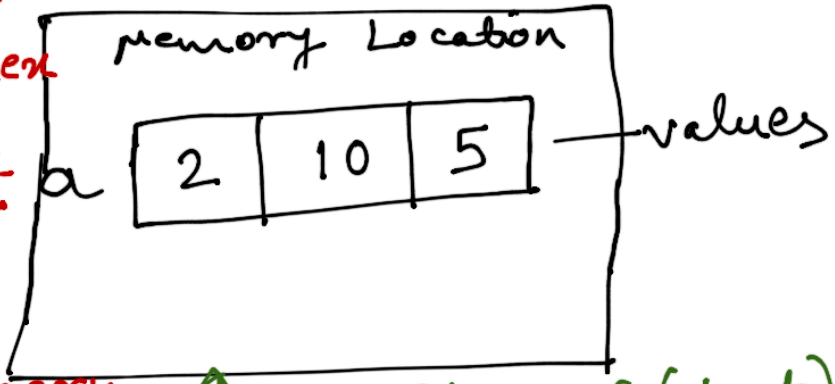


Instead of engaging so many variables, better to go for one identifier & organize the data.

Data type → int a[3]; ← No. of elements

Name of the array

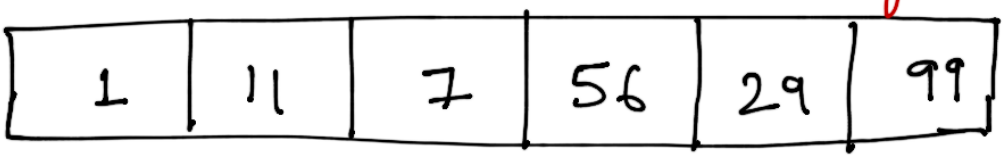
* series of elements
* accessible by an index
* same data type
* unique identifier
* contiguous memory



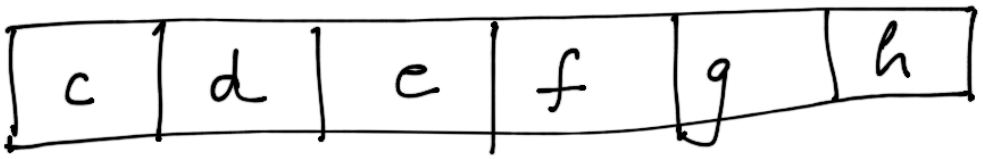
↑ 3 * sizeof(int)

Are these Arrays ??

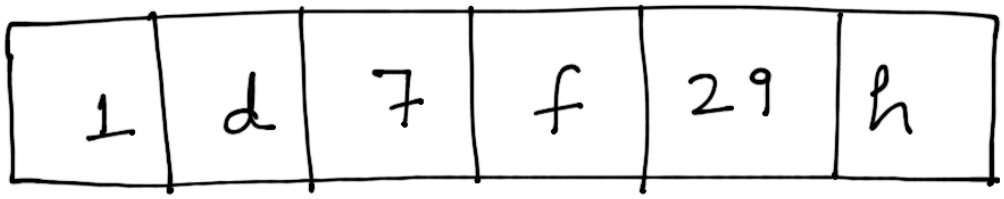
a



b



c



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;
```

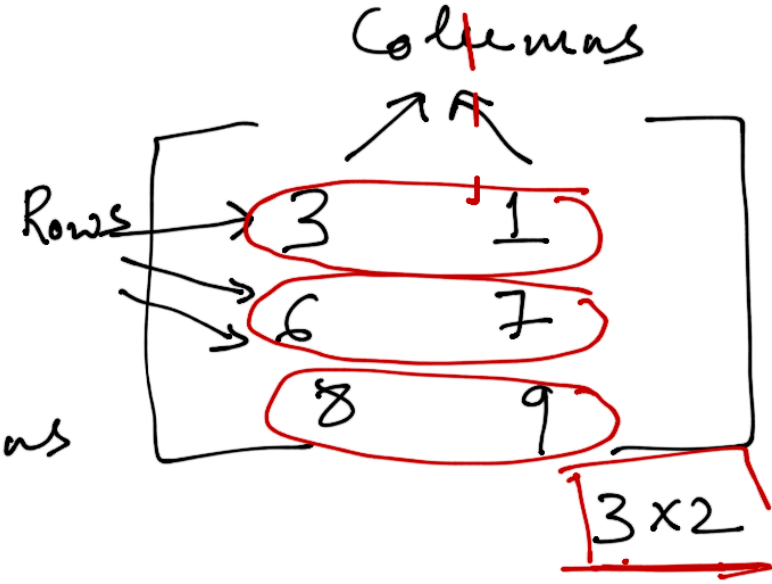
Method 4:

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

2-D Array

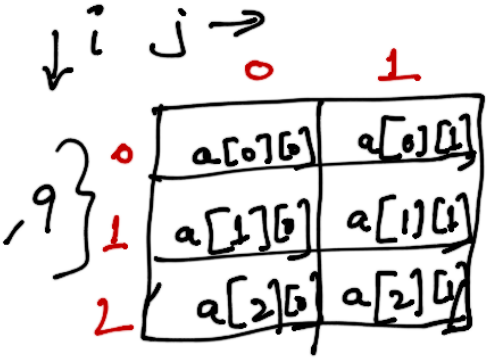
```
int a[3][2];
```

Rows Columns



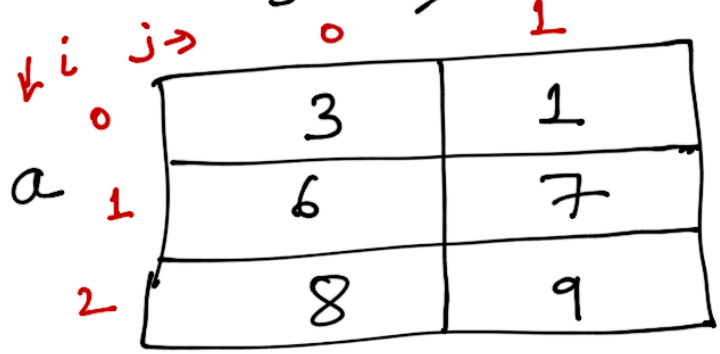
* Array of Arrays

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



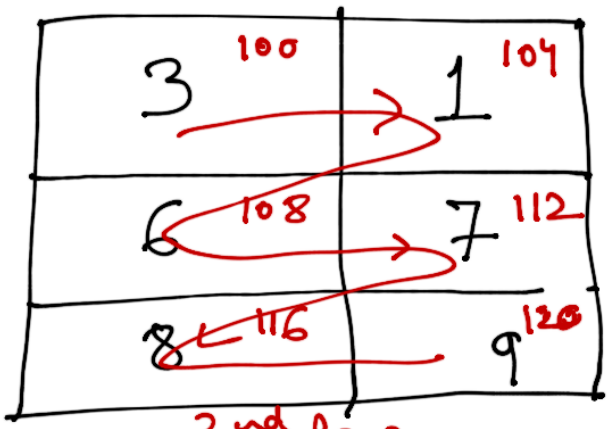
```
int a[3][2] = { {3,1}, {6,7}, {8,9} };
int a[][2] = { };
int a[3][] = X..
```

```
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];
        }
    }
}
```

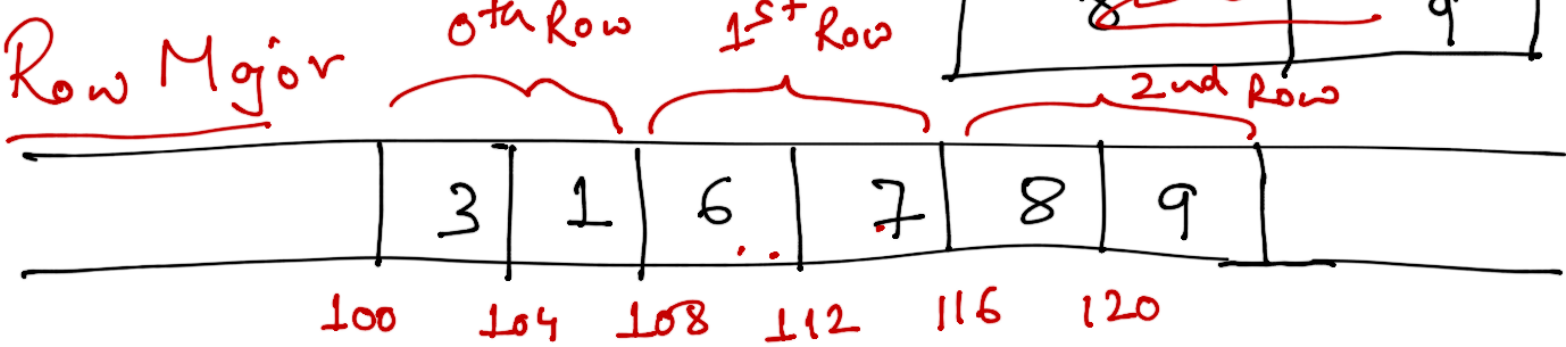


1) Row_Major

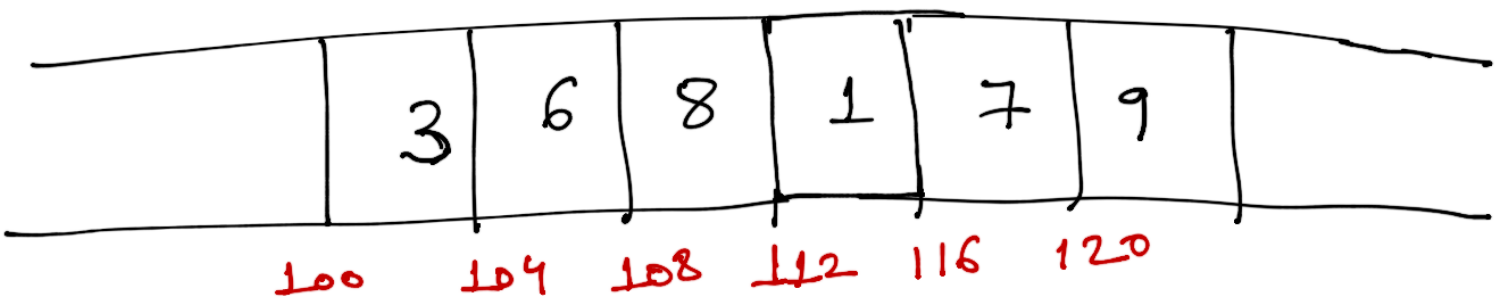
2) Column_Major

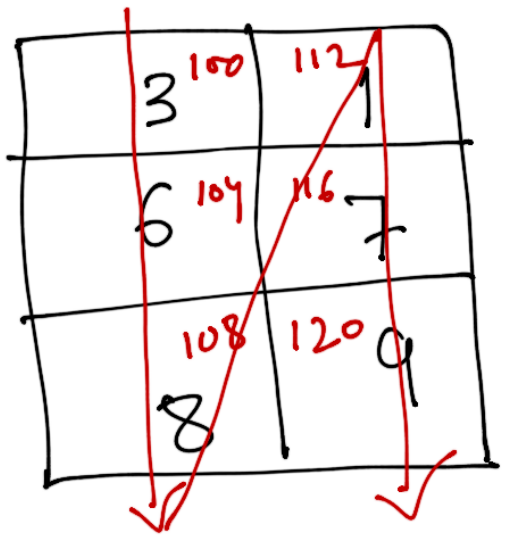


Row Major



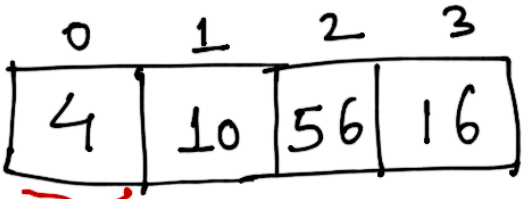
Column Major





Accessing Elements in an Array

for 1-D Array



Base Address$_i$ = Base Address$_0$ + i * Size of data type

= 100 + 2 × 4

for i = 2 = 100 + 8

= 108

for 2-D Array

1) Row-Major

Base Address$_{ij}$ = Base Address$_{00}$ + ((i * n) + j) × size

where, m × n represents size of the array

2) Column - Major

$$\text{Base Address}_{ij} = \text{Base Address}_{00} + ((j * m) + i) * \text{size}$$