# Array in C

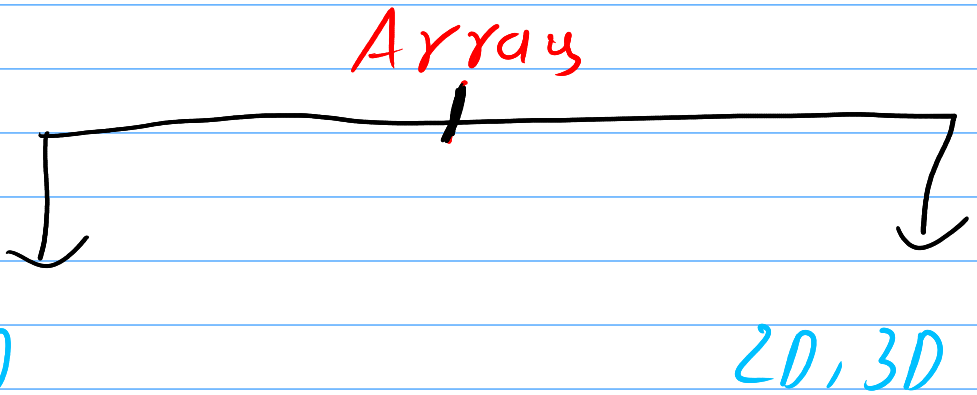
Array :- Collection of element of the same type that reffred by common name

→ Occupie continuous memory

→ derived datatype

Drawback :- fixed size sequantial collection of element

## TYPES OF Array



# 1D Array

→ The array has only one index (subscript)

→ can be viewed as a linear sequence

→ ++ & -- in single direction

## syntax of 1-D Array

### Declaration of Array

Type arrayname [size];

Type : Data type

size : max number of element that can be stored in array

```
int num[3];
```

# 1D Array map

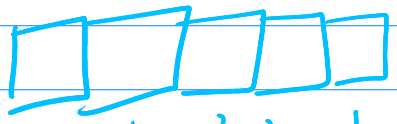
size of array = 5

```
Arr [5];
```

Array Declaration

Memory allocated

Arr




← Array index

## Array initialization

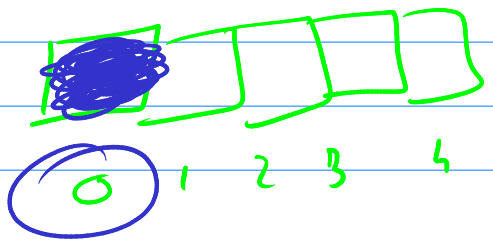
```
Arr [5] = [2, 4, 8, 16, 32];
```


## Access Array Element

Array variable

Index to be accessed by element

```
Arr [0];
```



# Types of array

## Initialization

In compile - time, the array in initialized when they are declared

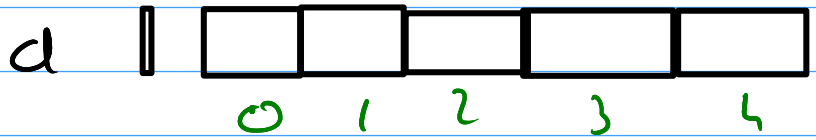
```
int num[ ] : [3, 4, 5] ; int array declar
char num[ ] : ['a', 'b', 'c'] ;

char num : "abc" ;
```

## Run time initialization

Use for large size array

```
for( i:0; i<5 ; i++)
  {
     sum : 1+ sum[i];
  }
```

# Summerized - 1D Array




size of the array = 5

Index = 0, 1, 2, 3, 4

first Index = 0 upper bound

last Index = 4 lower bound

# 2D - Array

Visualize it in a tabular form

Ex rows & columns

## Declaration Array

Types arrayname [size1][size2] → same

Type : Data type → same

size1 : rows

size 2 : columns

size 1 & size 2 :

max number of rows & column

Ex Int num[3][3];

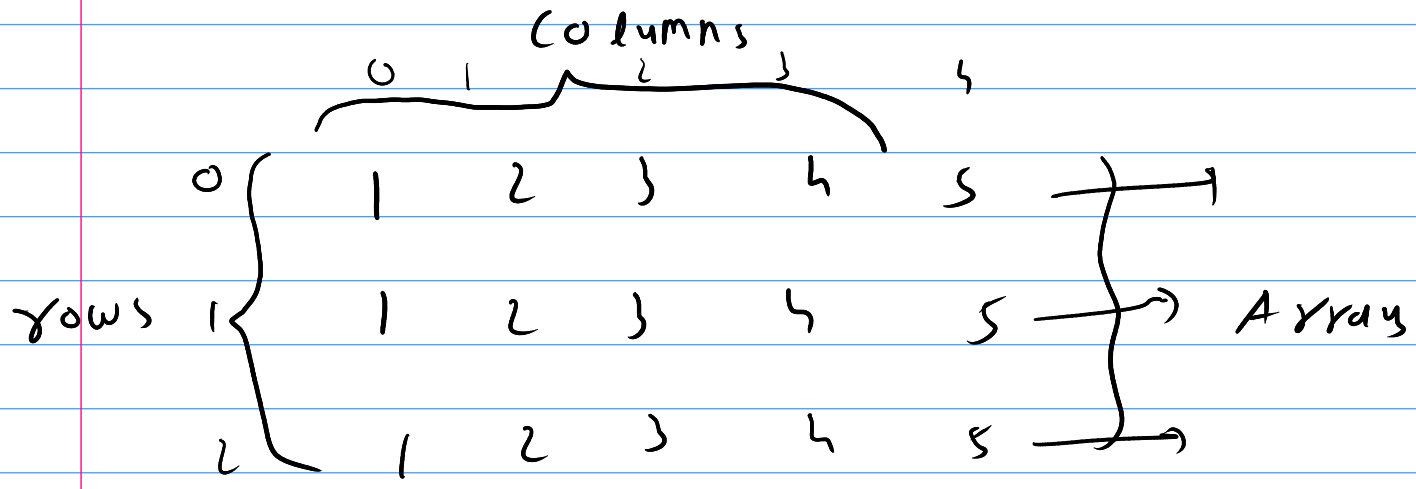
## initialization of Array

Ex Int num[3][3]

= { 1, 2, 3, 4, 5, 6 };

# Memory map

## 2D Array



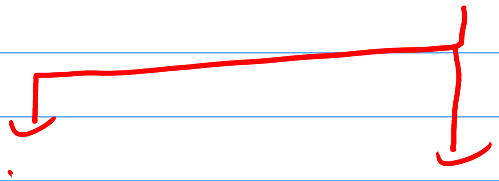
# Advantages

→ searching is very easy

→ Multiple value store in single variable

→ Used to represent complex data like Matrix

→ they have very low overhead



easily declare

initialize

# Disadvantage

→ fixed size

++ , -- not done in this array