C → Procedural / Functional prog.

~~Array~~

i) When we deal with huge same type of data.

Student info., Students marks. <Homogeneous data>

ii) Linear data structure → Array.

Accessing, Searching, Manipulation, --- etc. can be done in linear time → O(n)

iii) 1-D array 2-D array. n-D array.

define

<datatype> name of the array [<size>] ;

Static

int arr[30]; // 30 slots will be created each of size 4 byte.

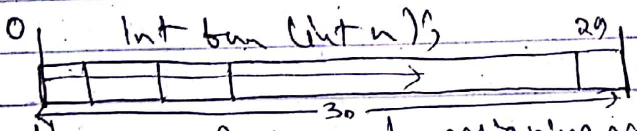
30 * 4 = 120 bytes.

definition → memory allocated

declaration → m" will be "

extern int n

int fun (int n);



i) there are 2 ways of assigning value in an array

```
for (i = 0; i < 30; i++)
{
    scanf ("%d", &arr[i]);
    or  arr[i] = x;
}
```

```
for (i = 0; i < 30; i++)
{
    printf ("%d", arr[i]);
}
```

ii) Hard Coded.

int arr[ ] = { 1, 2, 3 }; (0th, 1st, 2nd)

automatically 3 memory spaces will be created.

int arr[10] = {0};

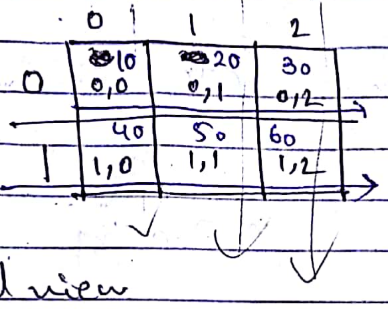
int arr[5] = {1, 2, 3}; | 1 | 2 | 3 | 0 | 0 |

2D-array :-

Row   Column.

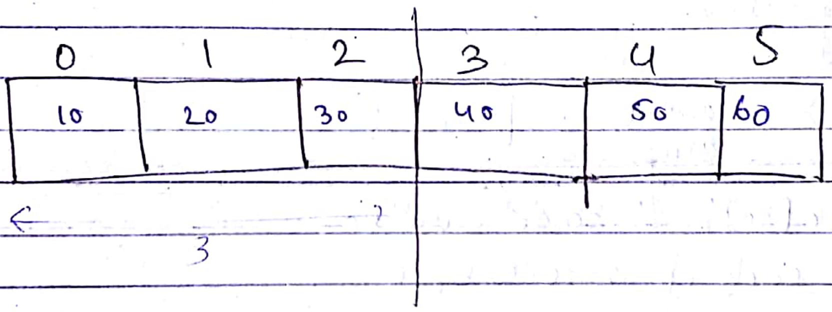
~~Ex~~ <datatype>   <array name>   [Size]   [Size]

int A[2] [3];



} logical view

Physical view



How to pass 1-D array using func.

```
Void  fun (Int arr[ ], Int  Size);//declaratn

main()
{
   int A[3] = {1,2,3};
   fun(A, 3); // calling  call by value as well as
                          call by ref.
   return 0;
}
```

→ ⇕ Copied → 11, 12, 13

Base address of the array/&A[0]

```
Void  fun(int arr[ ], int Size)
{
   for (i=0; i < size; i++ )
   {
      A[i] += 10;
      printf ("%d", A[i]);
   }
}
```

How to pass 2D array   size??
↓
```
void fun (int arr [], [3] , int row, int column);
         (int row, int column,   int arr [] [column]);
main()
{


}
void fun ( -  -  -  -  )
{


}
```

```
                 Column
                   B
for ( i = 0;   i < row;  i++)            Row major order
{                         row B
    for (j = 0;  j < Column;  j++)
    {
        scanf ("%d"  , & A[i] [j]);
        printf ("%d"  , A[i] [j]);
    }                    A [j] [i]
}
                                          Column major

     i = 0        j = 0         A [0] [0]
                    1             [1] [0]
```

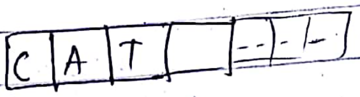
Scanned by CamScanner

Compare → A = = B

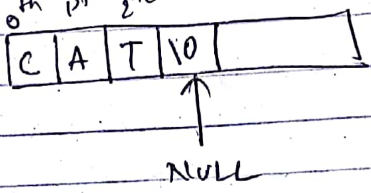
A = B

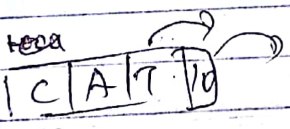
String → Character array ∃ ~~X~~ datatype ~~of~~ called String

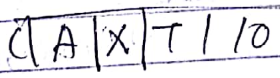
Char A [10]; = { 'C', 'A', 'T' };



Char A [10] = " CAT";







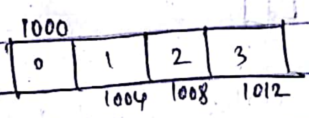
int A [30];

A [12] → ??

A = 1000;

1000 + 12 * 4

1D



1000 + 3 * 4 = 1012

Base 1000

2D

A [100] [200];

Row major

Row major

A [50] [50]; →

(0 – 49)

{(50 * 200) + 50} * 4 + 1000.

((RowIndex * Columnsize) + Column Index) * size of the datatype + Baseaddr.

Column

((Column Index * Row size) + Row Index) *