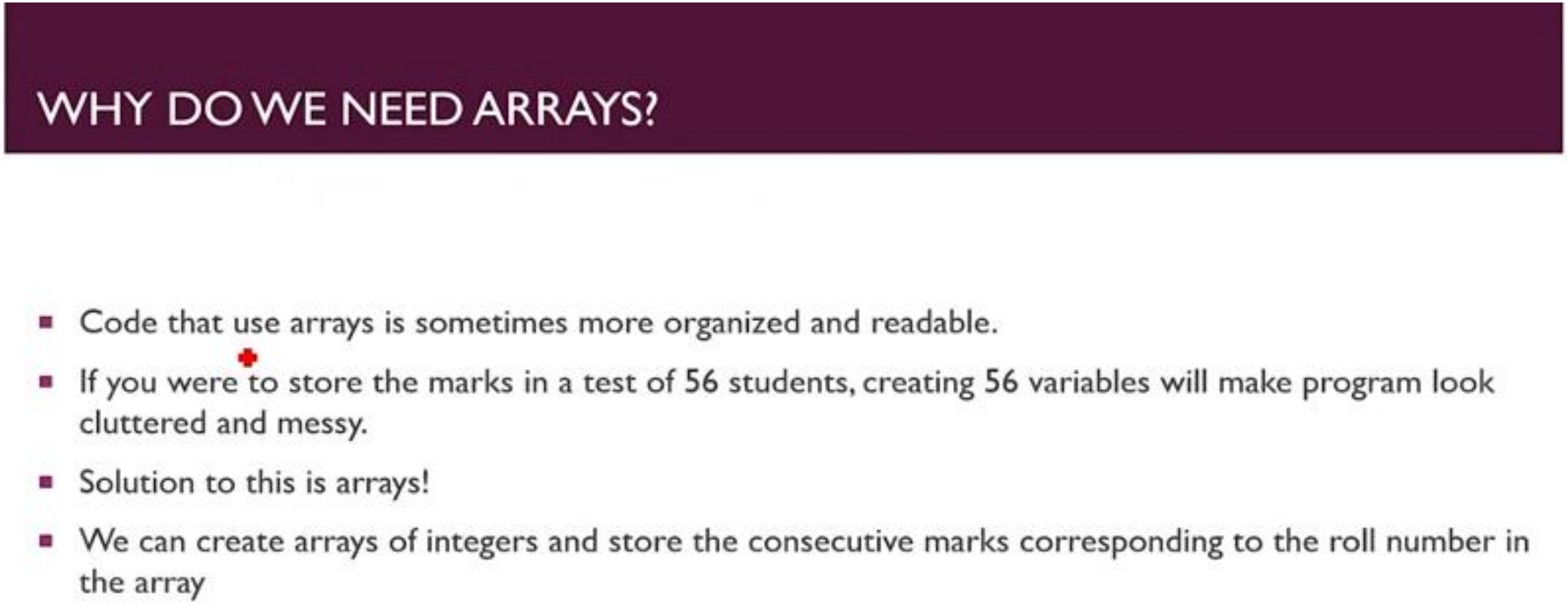
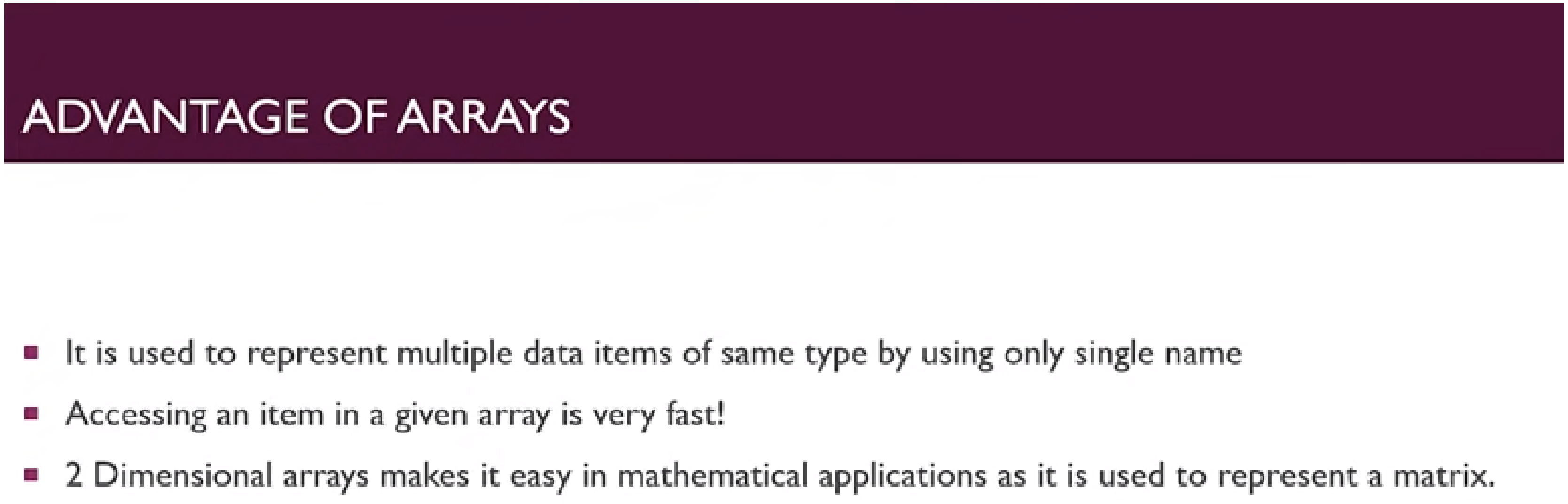
Array



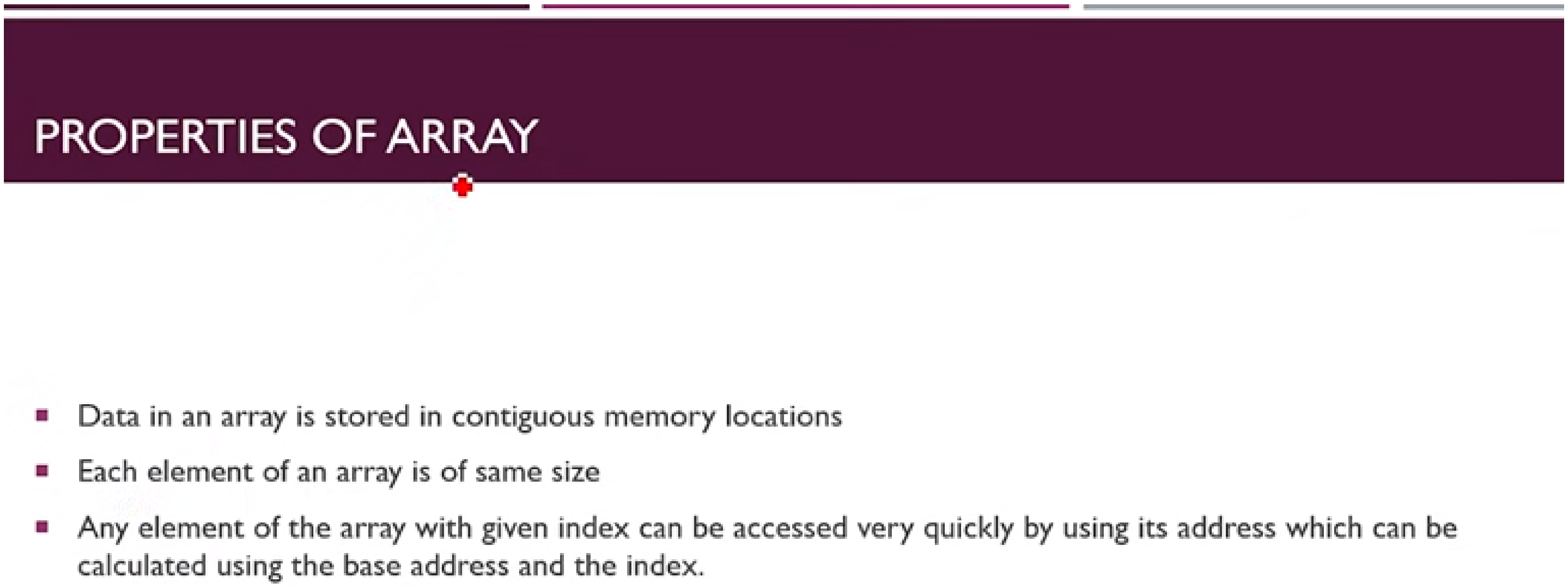
WHY DO WE NEED ARRAYS?

- Code that use arrays is sometimes more organized and readable.
- If you were to store the marks in a test of 56 students, creating 56 variables will make program look cluttered and messy.
- Solution to this is arrays!
- We can create arrays of integers and store the consecutive marks corresponding to the roll number in the array



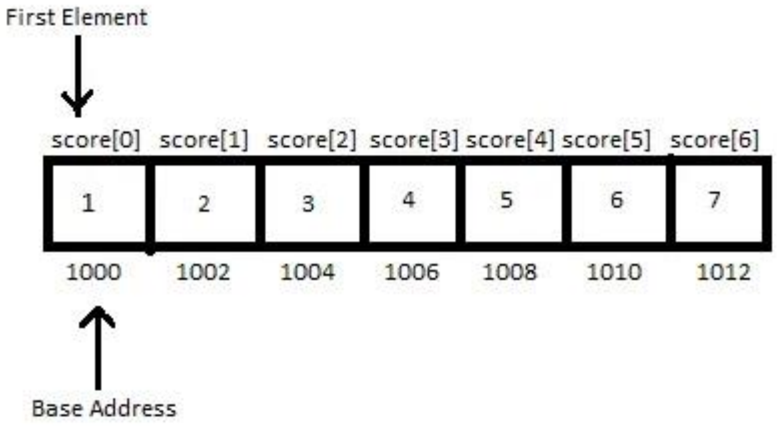
ADVANTAGE OF ARRAYS

- It is used to represent multiple data items of same type by using only single name
- Accessing an item in a given array is very fast!
- 2 Dimensional arrays makes it easy in mathematical applications as it is used to represent a matrix.



PROPERTIES OF ARRAY

- Data in an array is stored in contiguous memory locations
- Each element of an array is of same size
- Any element of the array with given index can be accessed very quickly by using its address which can be calculated using the base address and the index.


First Element
score[0]
score[1]
score[2]
score[3]
score[4]
score[5]
score[6]
1
2
3
4
5
6
7
1000
1002
1004
1006
1008
1010
1012
Base Address

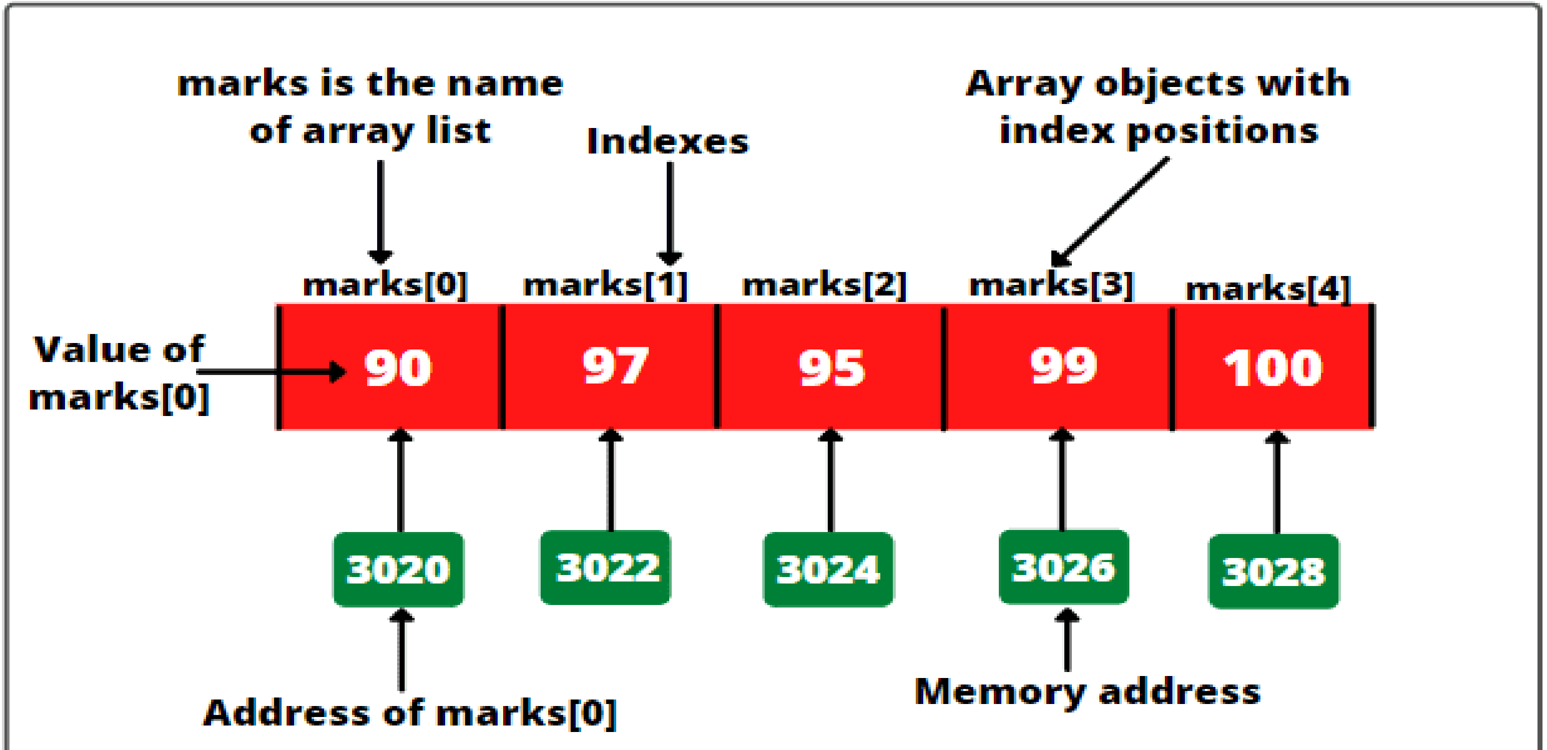
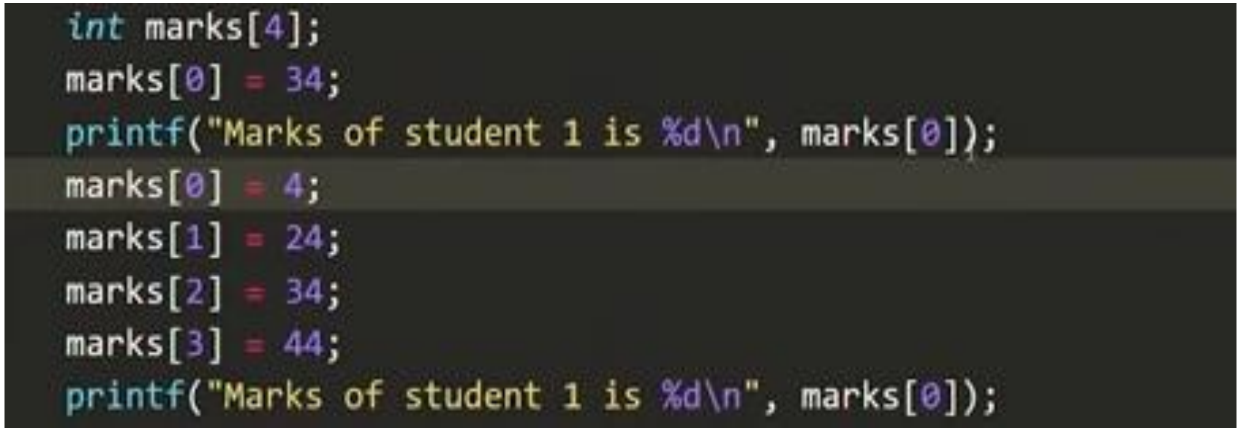
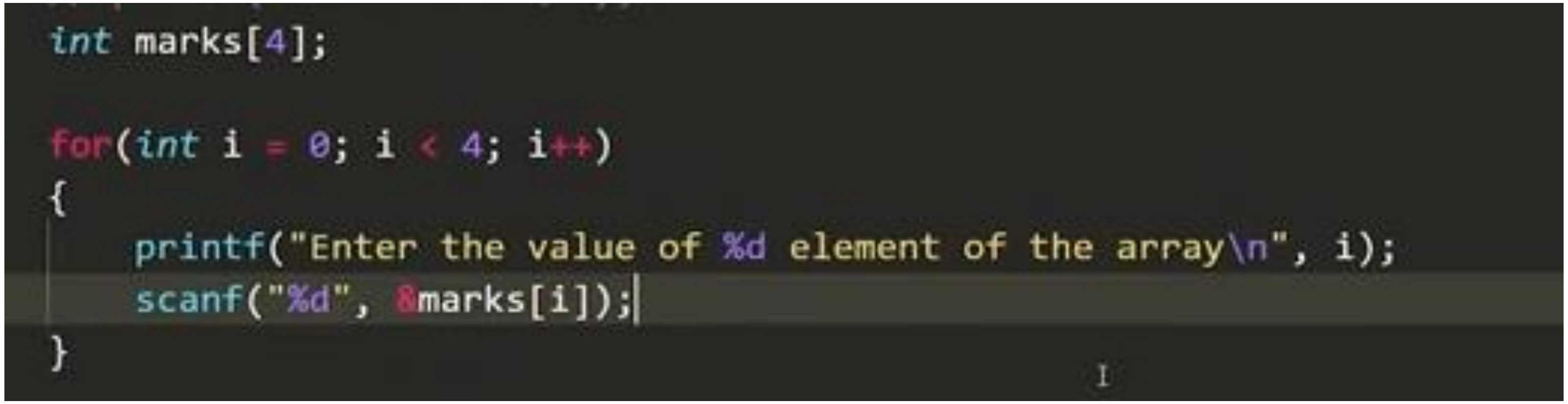


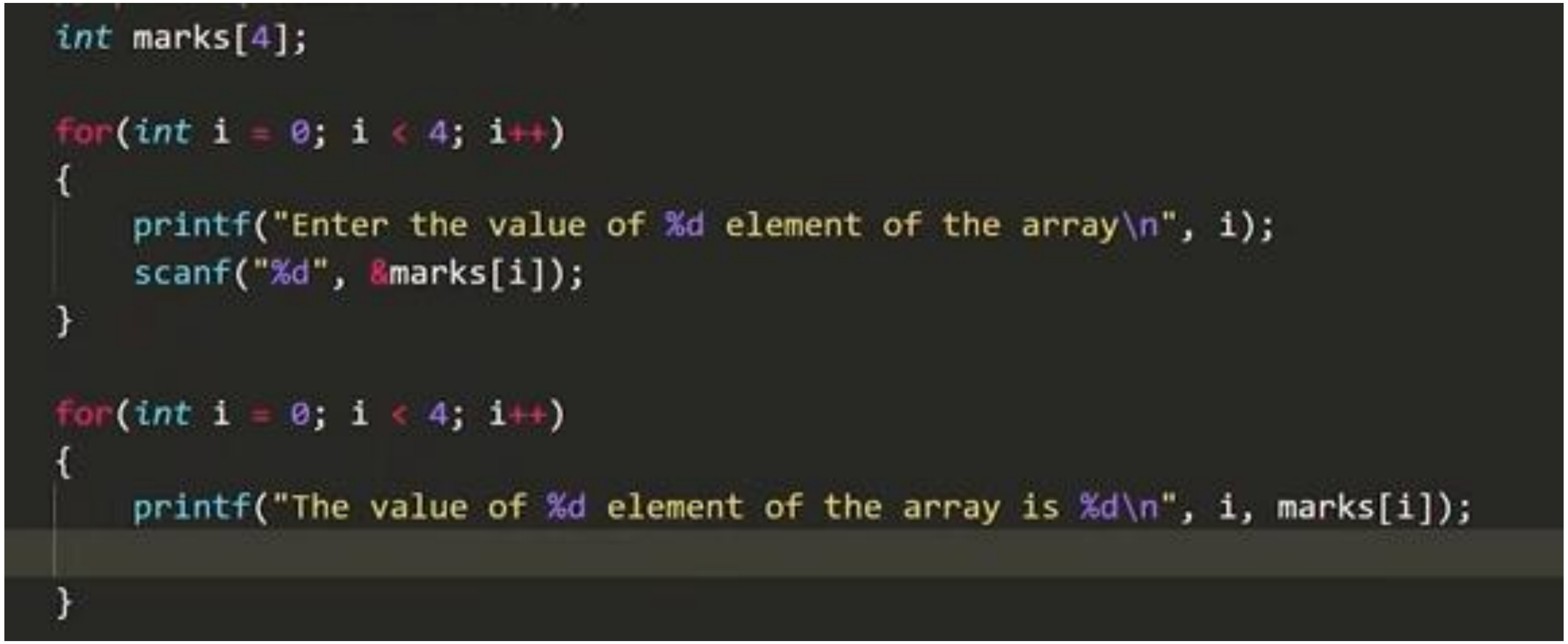
Fig: Memory address for array marks



```
int marks[4];
marks[0] = 34;
printf("Marks of student 1 is %d\n", marks[0]);
marks[0] = 4;
marks[1] = 24;
marks[2] = 34;
marks[3] = 44;
printf("Marks of student 1 is %d\n", marks[0]);
```



```c
int marks[4];

for(int i = 0; i < 4; i++)
{
    printf("Enter the value of %d element of the array\n", i);
    scanf("%d", &marks[i]);
}
```

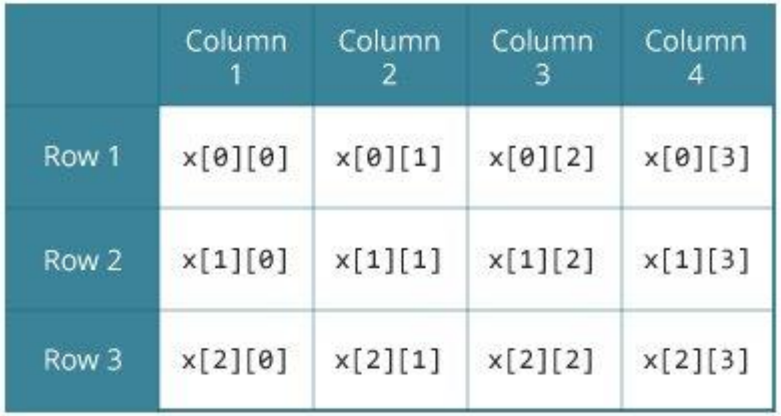


```c
int marks[4];

for(int i = 0; i < 4; i++)
{
    printf("Enter the value of %d element of the array\n", i);
    scanf("%d", &marks[i]);
}

for(int i = 0; i < 4; i++)
{
    printf("The value of %d element of the array is %d\n", i, marks[i]);

}
```

2-D Array

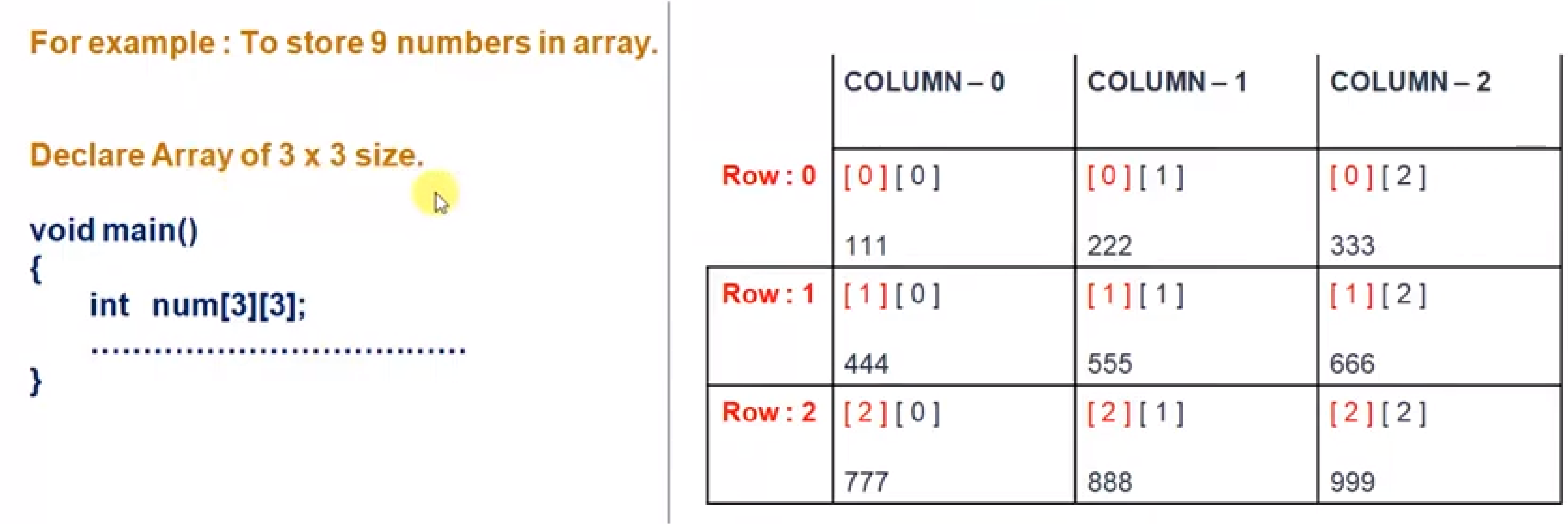
- Tow dimensional array is an array of one dimensional array.
- It means, it stores the data in combination of row and column. So whenever we declare 2D array, we have to specify two indices (row, column).
- Generally 2D array is declared matrix .
- General declaration of 2D array :

- **data_type array_name [row_size] [column_size];**
- For example : int x [3] [3];



	Column 1	Column 2	Column 3	Column 4
Row 1	x[0][0]	x[0][1]	x[0][2]	x[0][3]
Row 2	x[1][0]	x[1][1]	x[1][2]	x[1][3]
Row 3	x[2][0]	x[2][1]	x[2][2]	x[2][3]

2-D Array Declaration



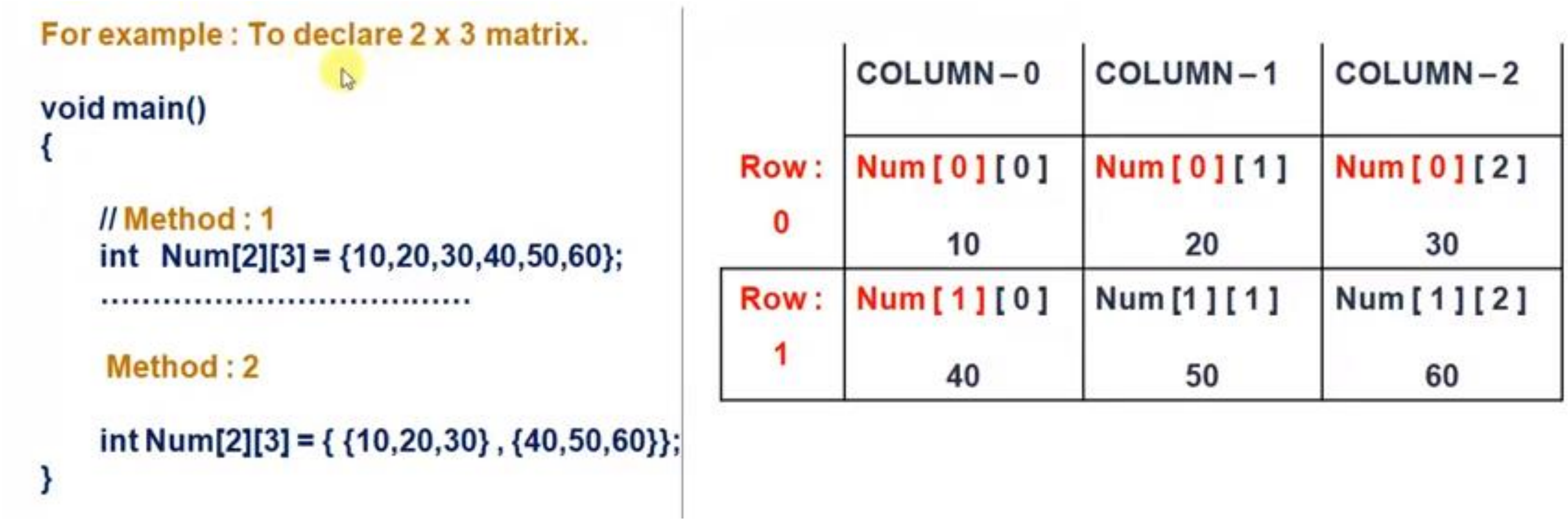
For example : To store 9 numbers in array.

Declare Array of 3 x 3 size.

```
void main()
{
      int   num[3][3];
      ......................................
}
```

	COLUMN – 0	COLUMN – 1	COLUMN – 2
Row : 0	[0] [0] 111	[0] [1] 222	[0] [2] 333
Row : 1	[1] [0] 444	[1] [1] 555	[1] [2] 666
Row : 2	[2] [0] 777	[2] [1] 888	[2] [2] 999

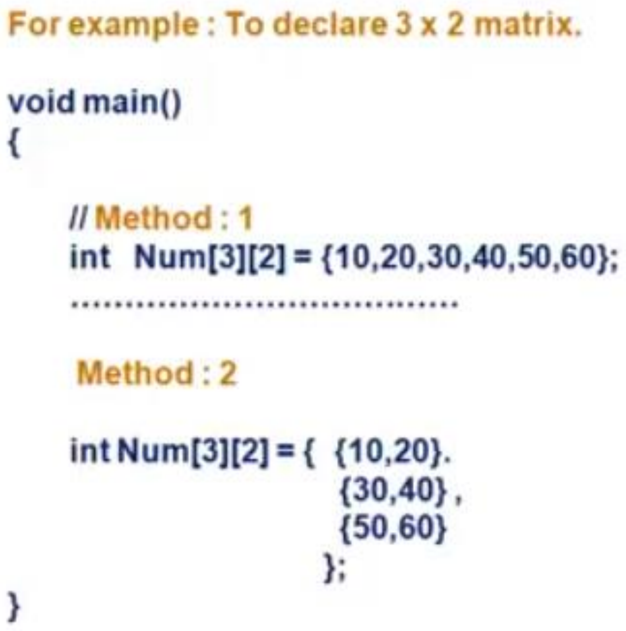
2-D Array Initialization



For example : To declare 2 x 3 matrix.

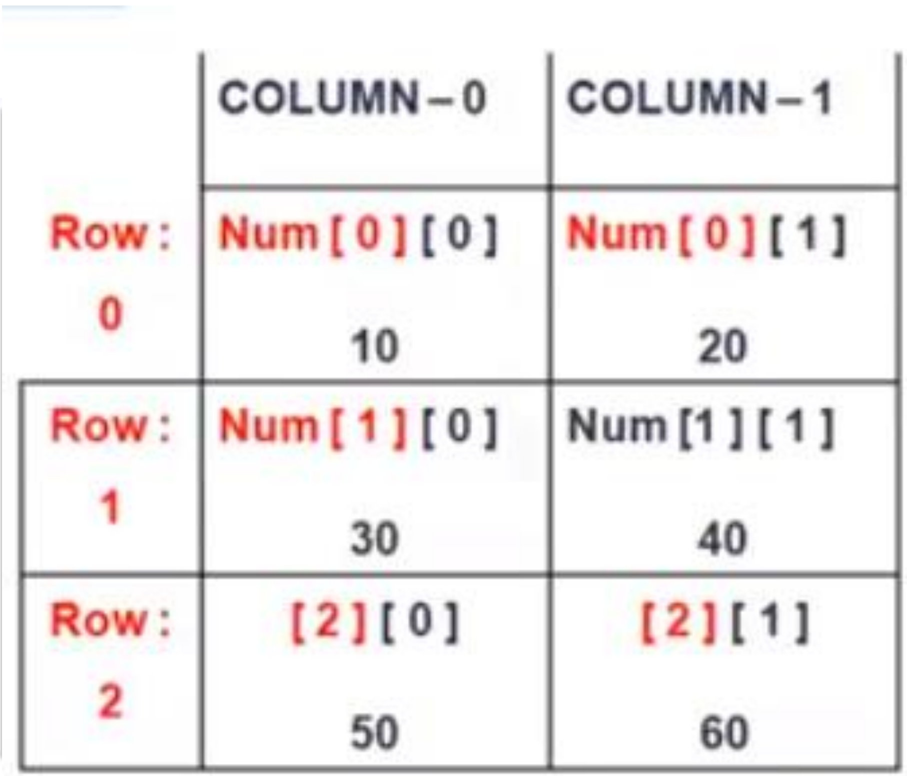
```
void main()
{

      // Method : 1
      int   Num[2][3] = {10,20,30,40,50,60};
      ………………………………

       Method : 2

      int Num[2][3] = { {10,20,30} , {40,50,60}};
}
```

	COLUMN – 0	COLUMN – 1	COLUMN – 2
Row : 0	Num [0] [0] 10	Num [0] [1] 20	Num [0] [2] 30
Row : 1	Num [1] [0] 40	Num [1] [1] 50	Num [1] [2] 60



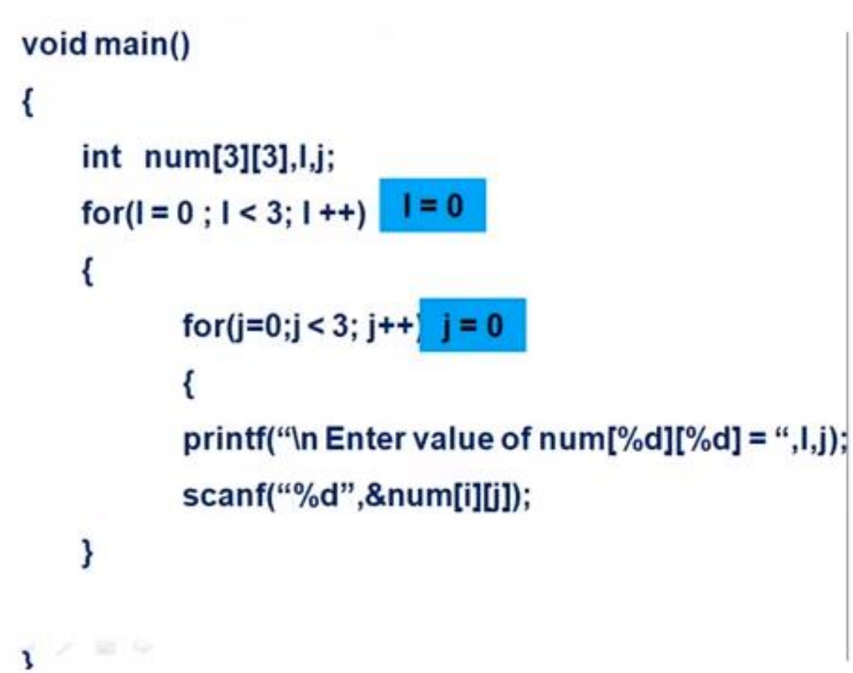
For example : To declare 3 x 2 matrix.

```
void main()
{

      // Method : 1
      int  Num[3][2] = {10,20,30,40,50,60};
      ........................................

       Method : 2

      int Num[3][2] = {  {10,20}.
                         {30,40} ,
                         {50,60}
                       };
}
```



	COLUMN – 0	COLUMN – 1
Row : 0	Num [0] [0] 10	Num [0] [1] 20
Row : 1	Num [1] [0] 30	Num [1] [1] 40
Row : 2	[2] [0] 50	[2] [1] 60

2-D Array User Input



```
void main()
{
    int  num[3][3],I,j;
    for(I = 0 ; I < 3; I ++)  I = 0
    {
        for(j=0;j < 3; j++  j = 0
        {
        printf("\n Enter value of num[%d][%d] = ",I,j);
        scanf("%d",&num[i][j]);
    }

}
```

	COLUMN 0	**COLUMN 1**	**COLUMN 2**
ROW 0	NUM [0] [0] 10	NUM [0] [1] 20	NUM [0] [2] 30
ROW 1	NUM [1] [0] 40	NUM [1] [1] 50	NUM [1] [2] 60
ROW 2	NUM [2] [0] 70	NUM [2] [1] 80	NUM [2] [2] 90

$$\textbf{Matrix 1} \begin{Bmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 3 & 3 & 3 \end{Bmatrix} \qquad \textbf{Matrix 2} \begin{Bmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 3 & 3 & 3 \end{Bmatrix}$$

$$\begin{array}{c}\textbf{Matrix 1}\\ * \\ \textbf{Matrix 2}\end{array} \begin{Bmatrix} 1*1+1*2+1*3 & 1*1+1*2+1*3 & 1*1+1*2+1*3 \\ 2*1+2*2+2*3 & 2*1+2*2+2*3 & 2*1+2*2+2*3 \\ 3*1+3*2+3*3 & 3*1+3*2+3*3 & 3*1+3*2+3*3 \end{Bmatrix} \qquad \begin{array}{c}\textbf{Matrix 1}\\ * \\ \textbf{Matrix 2}\end{array} \begin{Bmatrix} 6 & 6 & 6 \\ 12 & 12 & 12 \\ 18 & 18 & 18 \end{Bmatrix}$$