COMPUTER PROGRAMMING

Arrays

Arrays

- An array is simply a collection of variables of the same data type that are referred to by a common name.

- A specific element in an array is accessed by an index.

- All array consist of contiguous memory locations where the lowest address corresponds to the first element whereas the highest address corresponds to the last element.

Arrays

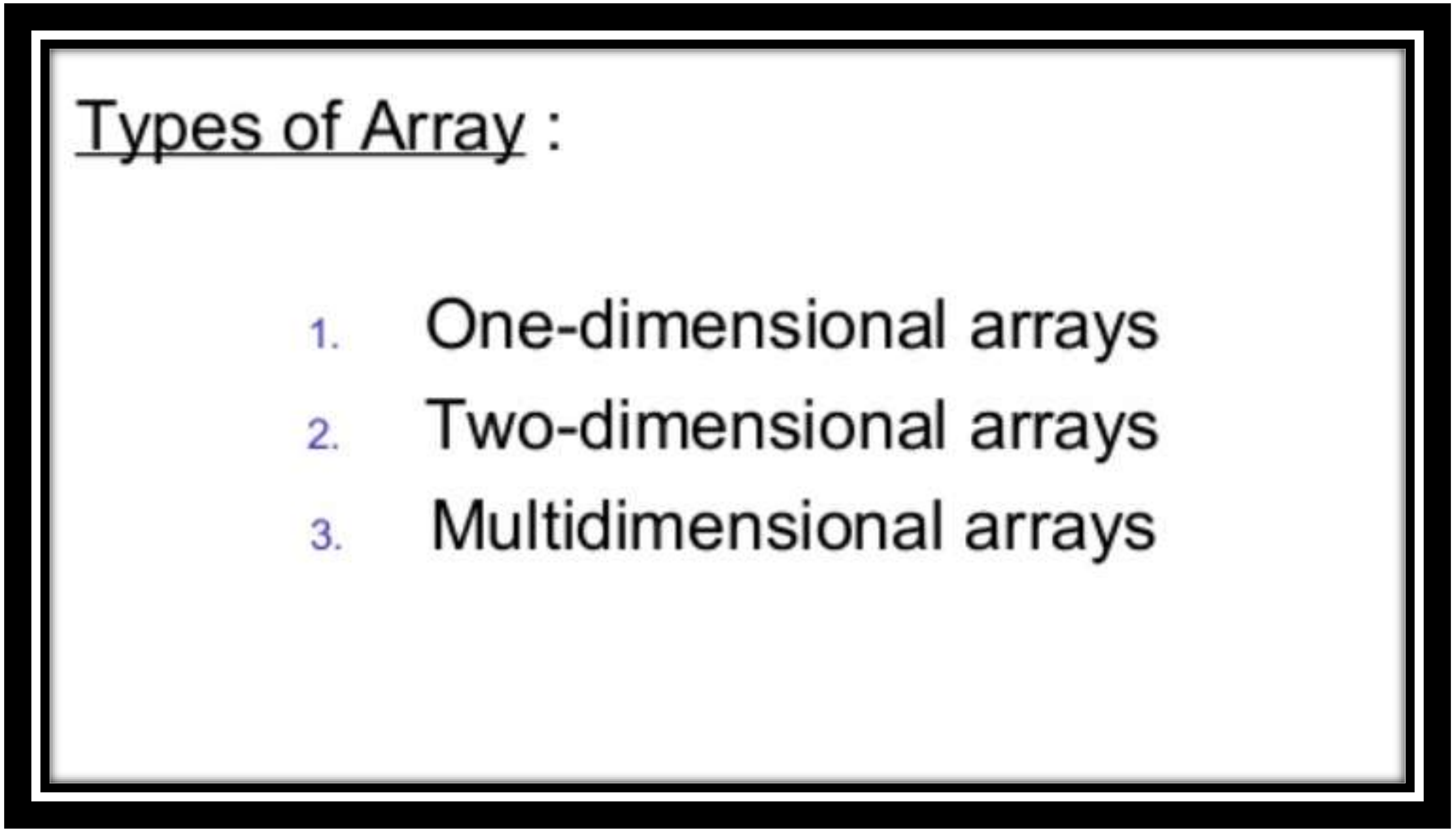
- The first element in the array is numbered 0, so the last element is 1 less than the size of the array.

- An array is also known as a subscripted variable.

- Before using an array its type and dimension must be declared.

- Index always starts from 0 in an array.

Arrays

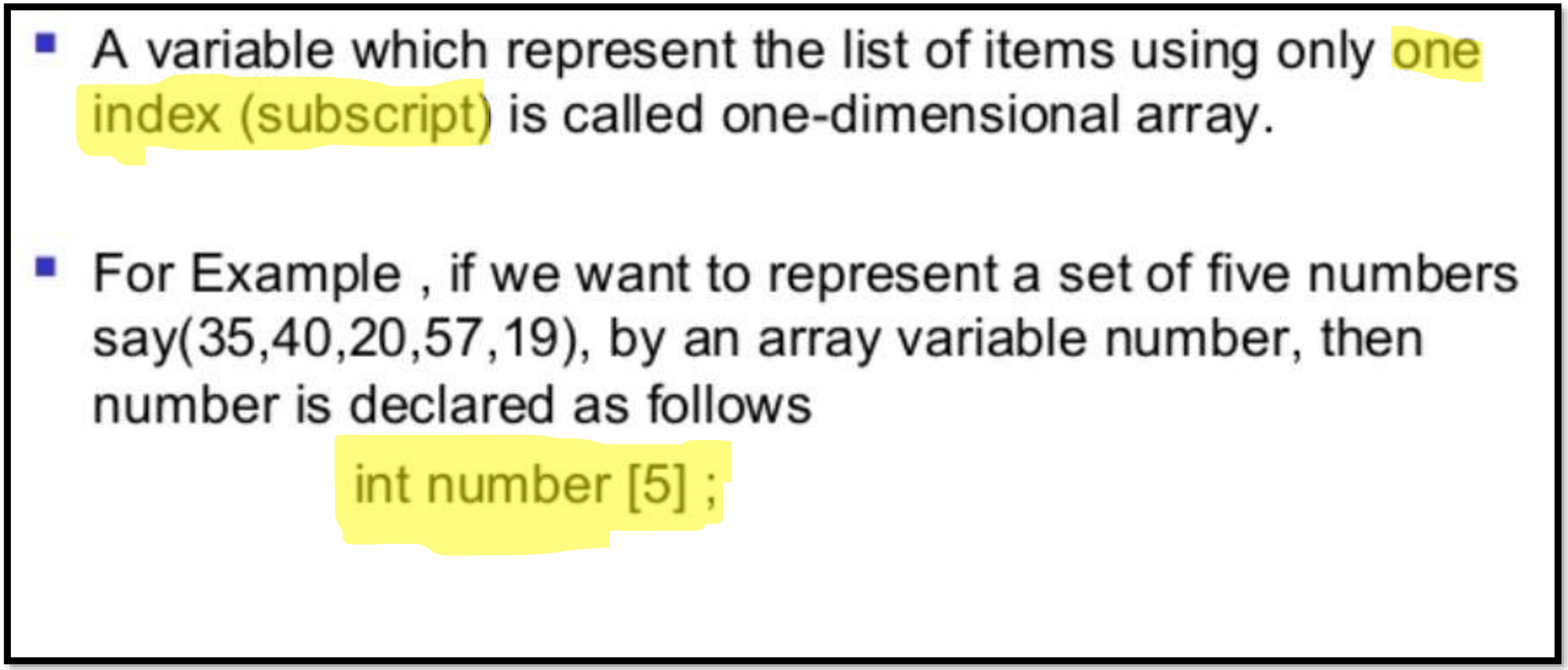


Types of Array :

1. One-dimensional arrays
2. Two-dimensional arrays
3. Multidimensional arrays

One-dimensional Arrays

An array of one dimension is known as a one-dimensional array or 1-D array



- A variable which represent the list of items using only one index (subscript) is called one-dimensional array.
- For Example , if we want to represent a set of five numbers say(35,40,20,57,19), by an array variable number, then number is declared as follows

```
int number [5] ;
```

One-dimensional Arrays

Like other variables an array needs to be declared so that the compiler will know what kind of an array and how large an array we want.

type arr_name[size];

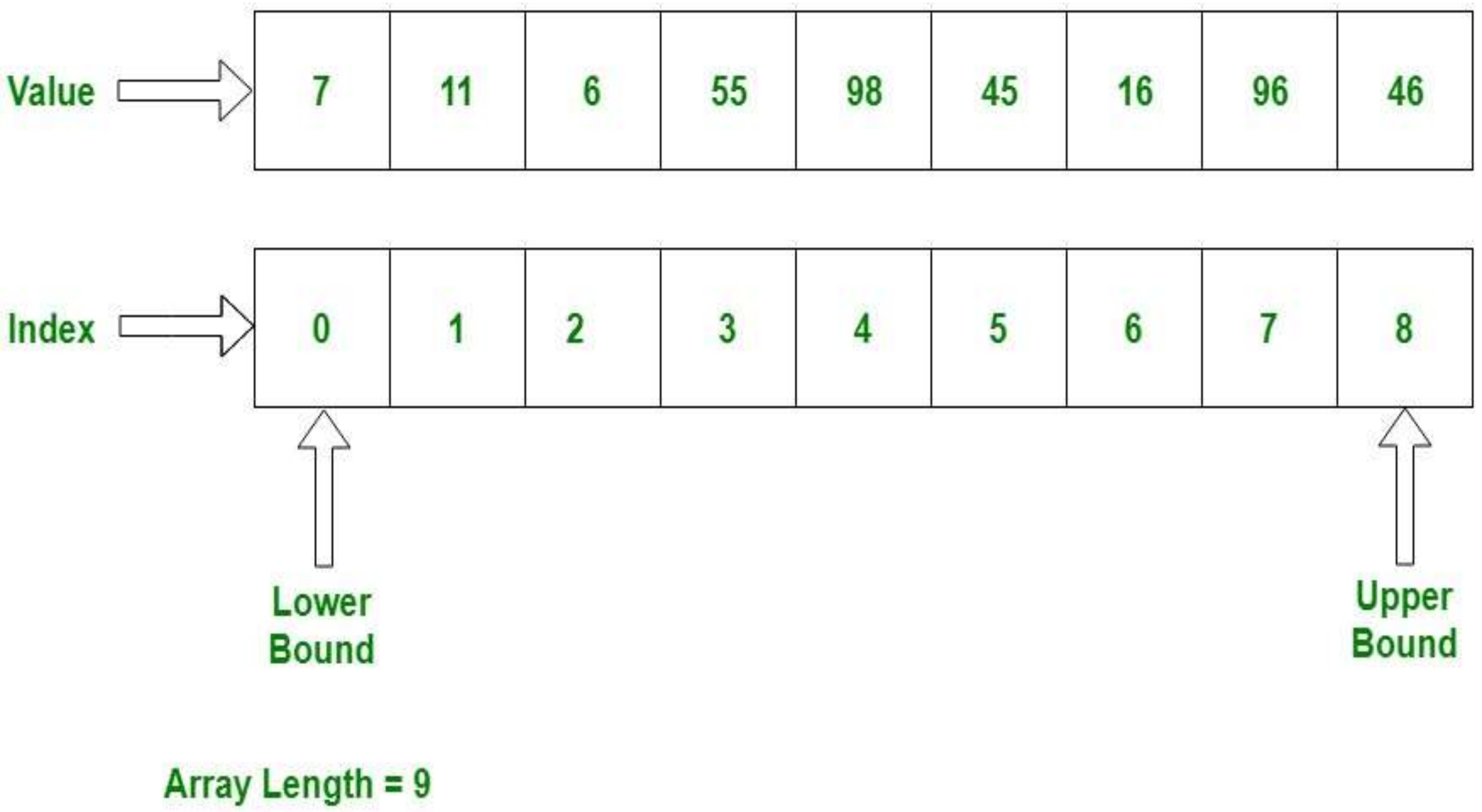
Here type is any valid data type,

arr_name is the name of the array you give

and size is the array size.

In other word, size specify that how many element, array can hold.

One-dimensional Arrays



One-dimensional Arrays

e.g.

```
int marks[30] ;
```

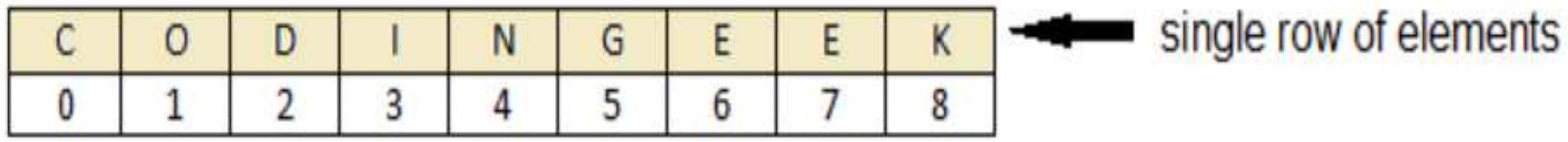
Here, *int specifies the type of the variable,*

The number 30 tells how many elements of the type int will be in our array. This number is often called the "dimension" of the array.

The bracket i.e. [ ] tells the compiler that we are dealing with an array.

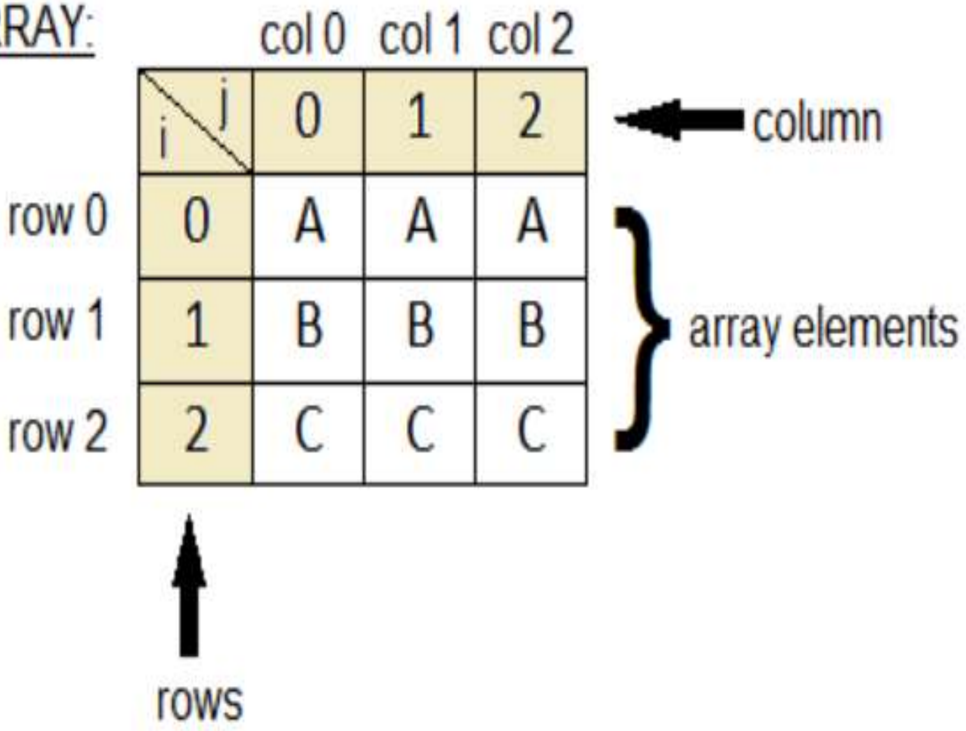
Arrays

1 D ARRAY:



C	O	D	I	N	G	E	E	K
0	1	2	3	4	5	6	7	8

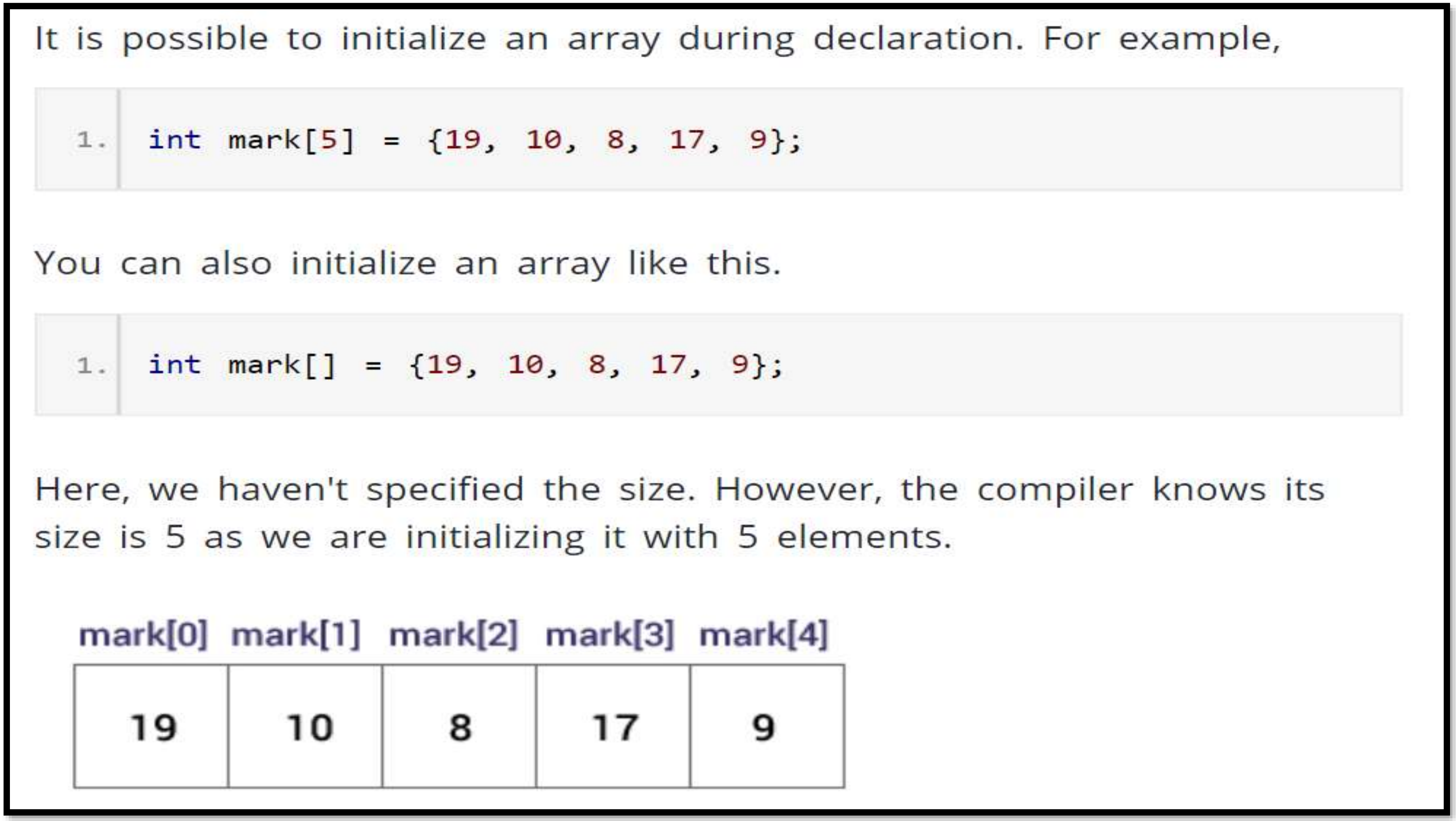
2 D ARRAY:



		col 0	col 1	col 2
	i \ j	0	1	2
row 0	0	A	A	A
row 1	1	B	B	B
row 2	2	C	C	C

One-dimensional Arrays

Initializing array in C



It is possible to initialize an array during declaration. For example,

```c
int mark[5] = {19, 10, 8, 17, 9};
```

You can also initialize an array like this.

```c
int mark[] = {19, 10, 8, 17, 9};
```

Here, we haven't specified the size. However, the compiler knows its size is 5 as we are initializing it with 5 elements.

mark[0]	mark[1]	mark[2]	mark[3]	mark[4]
19	10	8	17	9

One-dimensional Arrays

Initializing Array in C

```
int num[6] = { 2, 4, 12, 5, 45, 5 } ;

int n[ ] = { 2, 4, 12, 5, 45, 5 } ;

float press[ ] = { 12.3, 34.2, -23.4, -11.3 } ;

int b[ 100 ], x[ 27 ];
```

One-dimensional Arrays

Accessing Elements of an Array

- This is done with subscript, the number in the brackets following the array name.
- This number specifies the element's position in the array.
- All the array elements are numbered, starting with 0.
- Thus, arr [2] is not the second element of the array, but the third.

```
int arr[ ] = {1, 2, 3, 4, 5};

val = arr[2];     // val=3
```

One-dimensional Arrays

- Array elements are like normal variables

```
c[ 0 ] =  3;

printf( "%d", c[ 0 ] );
```

  - Perform operations in subscript. If x equals 3

```
c[5-2]==c[3]==c[x]
```

One-dimensional Arrays

1-D Array Input/Output

```
/* Program to take 5 values from the user and store them in an array
& Print the elements stored in the array*/

#include <stdio.h>
int main() {
int values[5];
printf("Enter 5 integers: ");


// taking input and storing it in an array
for(int i = 0; i < 5; ++i)
{
scanf("%d", &values[i]);
}
```

One-dimensional Arrays

1-D Array Input/Output

```
// printing elements of an array

  for(int i = 0; i < 5; ++i)
{
 printf("%d\n", values[i]);
  }
  return 0;
}
```

One-dimensional Arrays

```
/*Compute the sum of the elements of the array */

#include <stdio.h>
int main( void )
{   /* use initializer list to initialize array */
int a[12] = { 1, 3, 5, 4, 7, 2, 99, 16, 45, 67, 89, 45 };
 int i;             /* counter */
int total = 0;      /* sum of array */
   /* sum of contents of array a */
   for ( i = 0; i < SIZE; i++ )
 {
     total += a[ i ];
      } /* end for */
      printf( "Total of array element values is %d\n", total );
   return 0;
}                              Output : Total of array element values is 383
```

One-dimensional Arrays

Passing a single array element to a function

```
#include<stdio.h>

void display(int a);

int main()
{
    int myArray[] = { 2, 3, 4 };
    display(myArray[2]);        //Passing array element myArray[2] only.
    return 0;
}

void display(int a)
{
    printf("%d", a);
}
```

OUTPUT

```
4
```

One-dimensional Arrays

Passing 1-d array to function using call by value method

```
#include <stdio.h>
int main()
{
   char arr[] = {'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i',
'j'};
   for (int x=0; x<10; x++)
   {
       /* I’m passing each element one by one
using subscript*/
       disp (arr[x]);
   }

   return 0;
}
```

```
void disp( char ch)
{
   printf("%c ", ch);
}
```

OUTPUT:

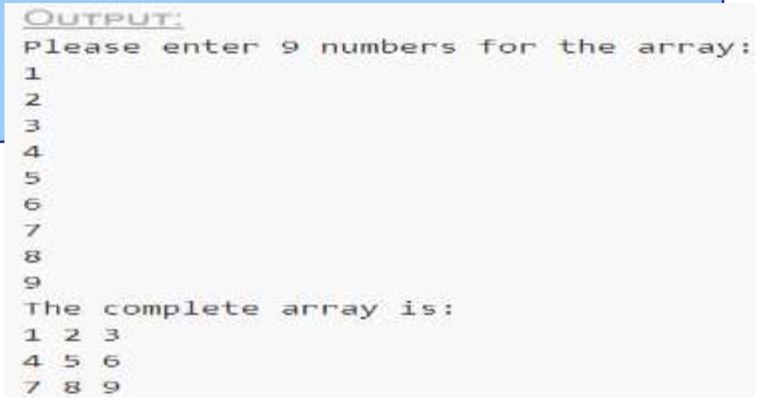
```
a b c d e f g h i j
```

One-dimensional Arrays

Passing a 2D array as a parameter

```
#include<stdio.h>
void displayArray(int arr[3][3]);
int main()
{
int arr[3][3], i, j;
printf("Please enter 9 numbers for
the array: \n");
for (i = 0; i < 3; ++i)
{
 for (j = 0; j < 3; ++j)
  {
    scanf("%d", &arr[i][j]);
   }
 }
// passing the array as argument
    displayArray(arr);
    return 0;
}
```

```
void displayArray(int arr[3][3])
{
int i, j;
printf("The complete array is: \n");
for (i = 0; i < 3; ++i)
 {
 // getting cursor to new line
   printf("\n");
  for (j = 0; j < 3; ++j)
    {
printf("%d\t", arr[i][j]);
       }
    }
}
```



```
OUTPUT:
Please enter 9 numbers for the array:
1
2
3
4
5
6
7
8
9
The complete array is:
1  2  3
4  5  6
7  8  9
```

Practice question:

Find maximum and minimum element in an array

One-dimensional Arrays

```
\\Find maximum and minimum element in an array
#include <stdio.h>
int main()
{
    int arr1[100];
    int i, max, min, n;
printf("Input the number of elements to be stored in the array :");
 scanf("%d",&n);

printf("Input %d elements in the array :\n",n);
       for(i=0;i<n;i++)
        {
              printf("element - %d : ",i);
              scanf("%d",&arr1[i]);
             }
```

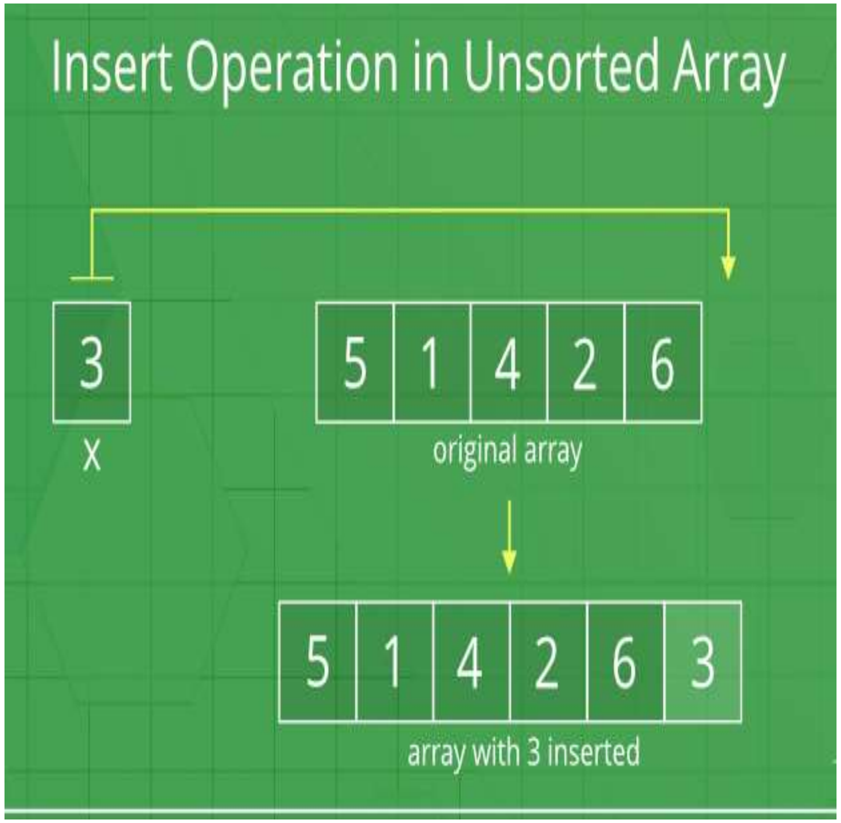
One-dimensional Arrays

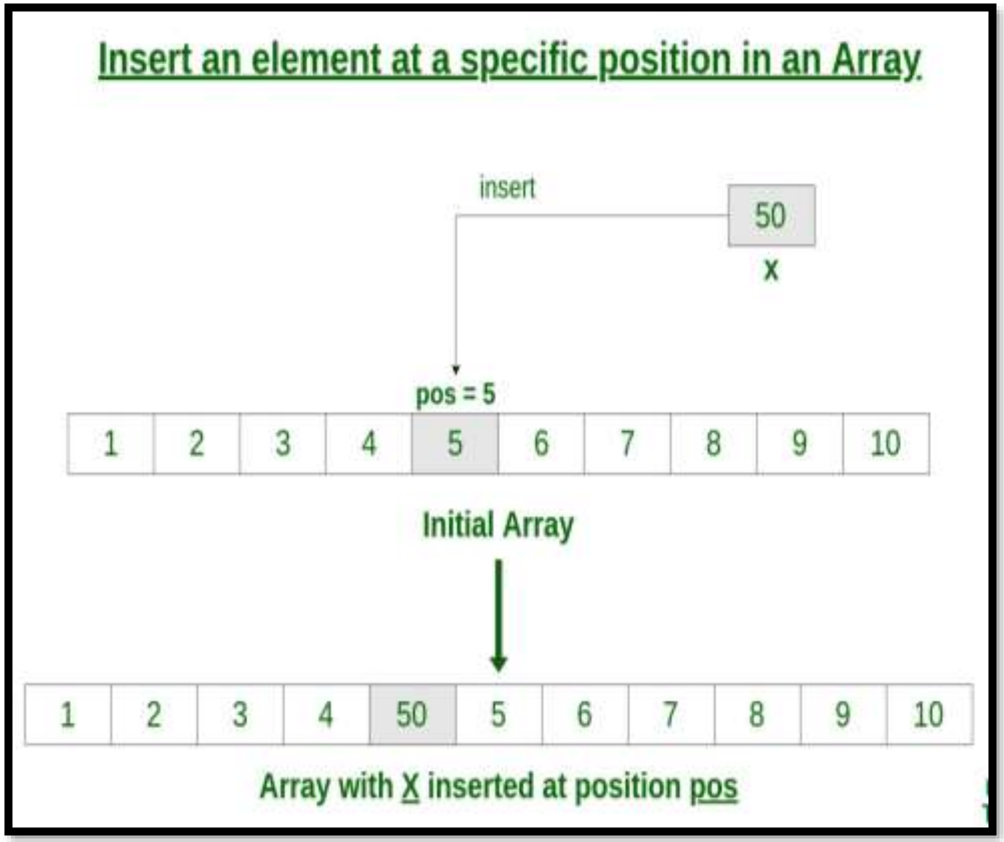
```
\\Find maximum and minimum element in an array
max = arr1[0];
min = arr1[0];
//finding max
for(i=1; i<n; i++)
{
if(arr1[i]>max)
{
max = arr1[i];
}
//finding min
if(arr1[i]<min)
{
min = arr1[i];
}
}
}
```

Practice question:

Insert an element in an Array

Practice question:
Insert an element in an Array




Insert an element at a specific position in an Array
insert
50
X
pos = 5
1 2 3 4 5 6 7 8 9 10
Initial Array
1 2 3 4 50 5 6 7 8 9 10
Array with X inserted at position pos

```
// C Program to Insert an element at a
//specific position in an Array

#include <stdio.h>
 int main()
{
   int arr[100];
   int i, x, pos, n = 10;

   // initial array of size 10
   for (i = 0; i < 10; i++)
      scanf(“%d”, &arr[i]);

   // print the original array
   for (i = 0; i < n; i++)
      printf("%d ", arr[i]);
   printf("\n");

   // element to be inserted
   x = 50;

// position at which element
   // is to be inserted
   pos = 5;

   // increase the size by 1
   n++;

   // shift elements forward
   for (i = n; i >= pos; i--)
      arr[i] = arr[i - 1];

   // insert x at pos
   arr[pos - 1] = x;

   // print the updated array
   for (i = 0; i < n; i++)
      printf("%d ", arr[i]);
   printf("\n");

   return 0;
}
```

Multidimensional Arrays

Multidimensional Arrays

Multiple-Subscripted Arrays

- **Multiple subscripted arrays**
  - Tables with rows and columns (m by n array)
  - Like matrices: specify row, then column

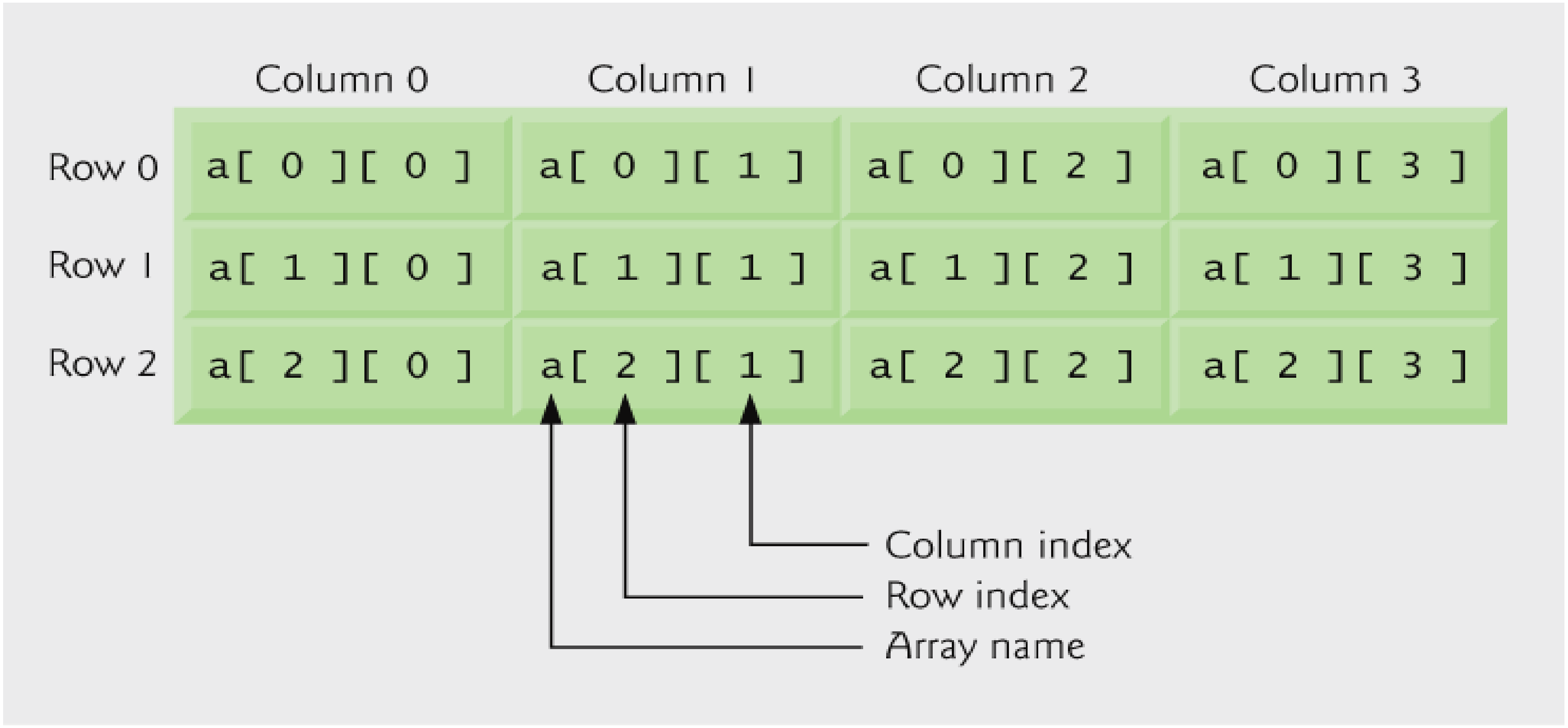
- **Declaration of two-dimensional array in C:**

```
int arr[10][5];
```

- *So the above array have 10 rows and 5 columns.*

Two- Dimensional Arrays

Double-subscripted array (2-D) with three rows and four column.

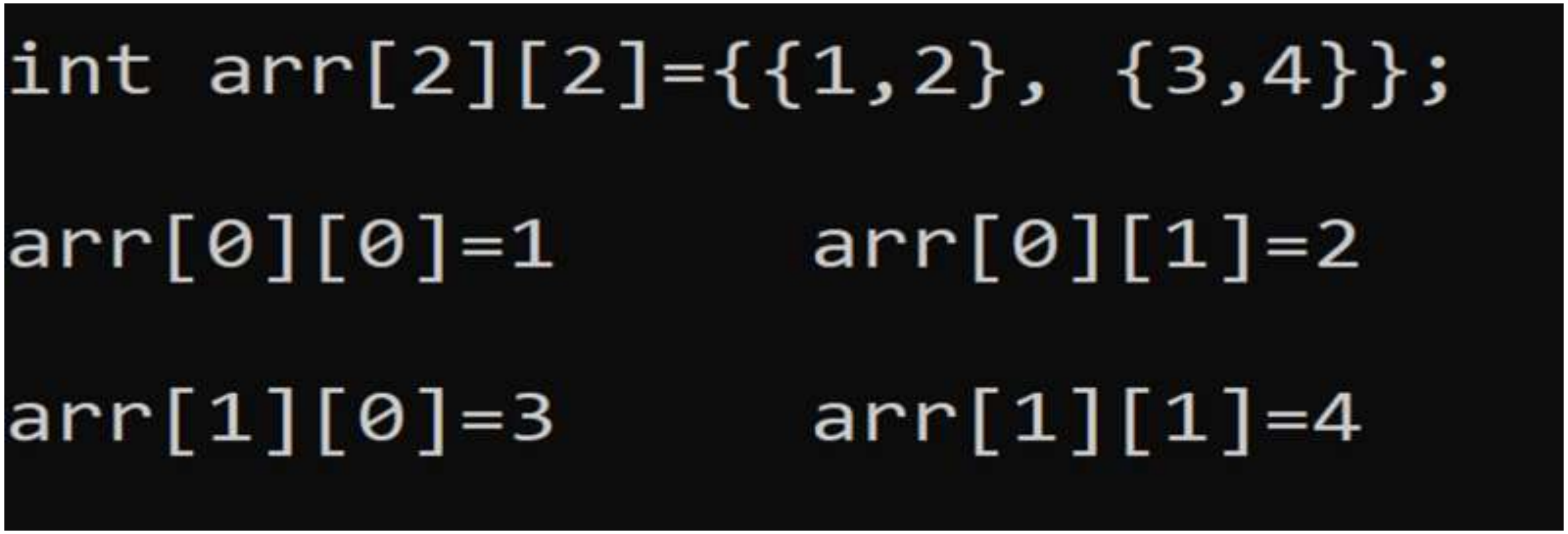


Two- Dimensional Arrays

- **Initialization**

  int arr[2][2] = { { 1, 2 }, { 3, 4 } };

  – Initializers grouped by row in braces



```
int arr[2][2]={{1,2}, {3,4}};

arr[0][0]=1          arr[0][1]=2

arr[1][0]=3          arr[1][1]=4
```

Two- Dimensional Arrays

- **Initialization**
  - If not enough, unspecified elements set to zero

```
int arr[ 2 ][ 2 ] = { { 1 }, { 3, 4 } };
```

- **Referencing elements**
  - Specify row, then column

```
printf( "%d", arr[ 0 ][ 1 ] );
```

Two- Dimensional Arrays

```
/* Example- initializing and displaying elements*/
#include<stdio.h>
void main()
{
        int arr[10][5];
        int i, j;
// e.g. initializing 2-D array elements by 1
        for(i=0; i<10; i++)
        {
                for(j=0; j<5; j++)
                {
                        arr[i][j] = 1;
                }
        }
```

Cont….

Two- Dimensional Arrays…

```
          // displaying 2-D array
for(i=0; i<10; i++)
 {
          for(j=0; j<5; j++)
          {
          printf("arr[%d][%d]=%d\t", i, j, arr[i][j]);
          }
          printf("\n");
          }
}
```

Matrix

Entering and displaying Matrix elements

```
#include<stdio.h>
void main()
{
        int arr[5][3];
        int i, j;
printf("Enter 5*3 Matrix: ");
for(i=0; i<5; i++)
{
for(j=0; j<3; j++)
{
scanf("%d", &arr[i][j]);
}
}
```

Cont….

Cont….

```
//Displaying the Matrix

printf("\nThe Matrix is:\n");
for(i=0; i<5; i++)
{
for(j=0; j<3; j++)
{
printf("%d\t", arr[i][j]);
}
printf("\n");
}   }
```

Matrix

$$A = \begin{pmatrix} 5 & 10 & 20 \\ 8 & 6 & 5 \end{pmatrix} \qquad B = \begin{pmatrix} 3 & 8 & 5 \\ 2 & 9 & 3 \end{pmatrix}$$

Addition of two Matrices

$$A + B = \begin{pmatrix} 5+3 & 10+8 & 20+5 \\ 8+2 & 6+9 & 5+3 \end{pmatrix} = \begin{pmatrix} 8 & 18 & 25 \\ 10 & 15 & 8 \end{pmatrix}$$

Matrix

Multiplication of two Matrices

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

$$\begin{array}{c} \textbf{Matrix 1} \\ * \\ \textbf{Matrix 2} \end{array} \quad \begin{Bmatrix} 1*1+1*2+1*3 & 1*1+1*2+1*3 & 1*1+1*2+1*3 \\ 2*1+2*2+2*3 & 2*1+2*2+2*3 & 2*1+2*2+2*3 \\ 3*1+3*2+3*3 & 3*1+3*2+3*3 & 3*1+3*2+3*3 \end{Bmatrix}$$

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

Matrix

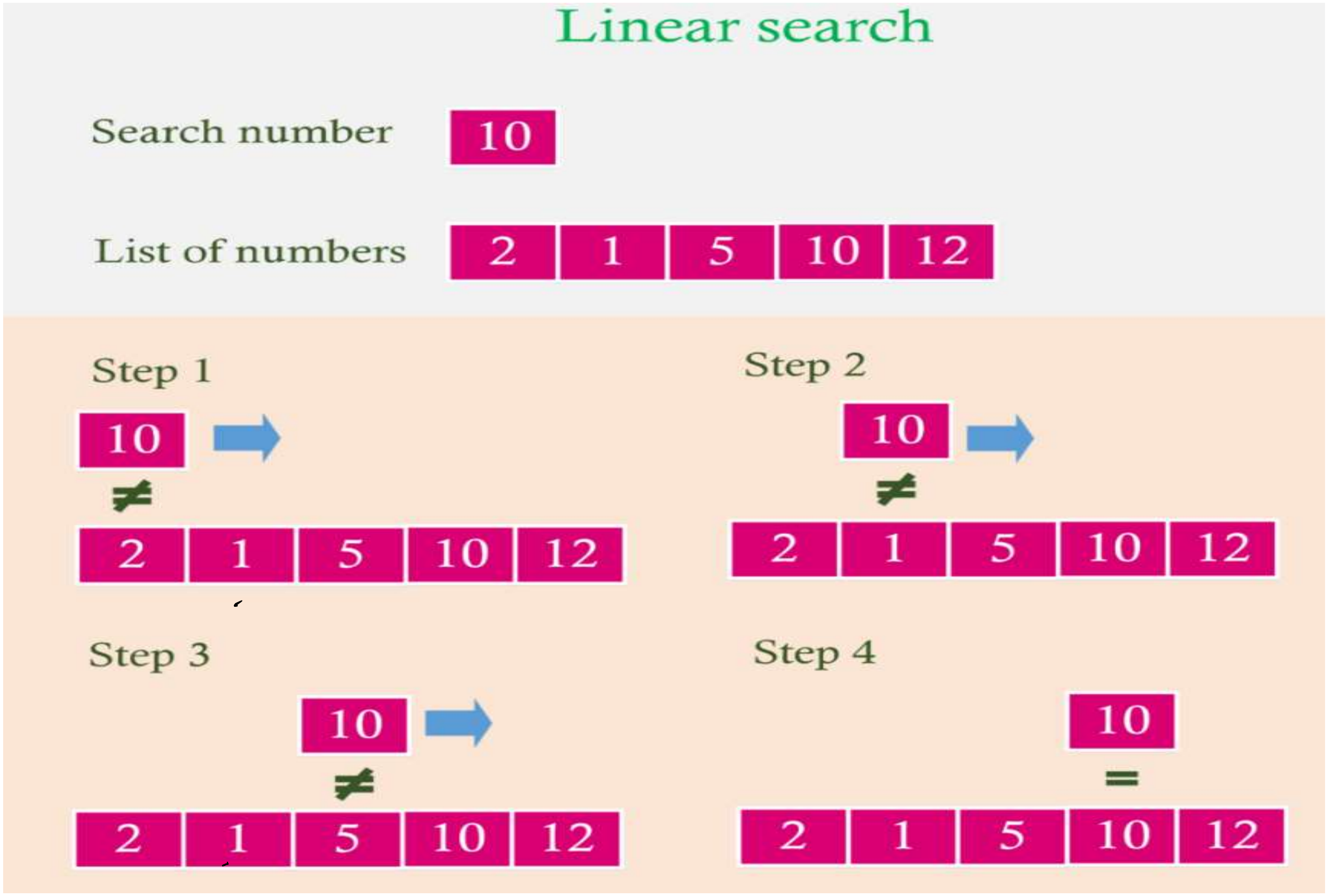
Transpose of a Matrix

ORIGINAL MATRIX
$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

TRANSPOSE MATRIX
$$\begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$$

Linear and Binary Search

Linear Search



Linear Search

Step 1 - Read the search element from the user.

Step 2 - Compare the search element with the first element in the list.

Step 3 - If both are matched, then display "Given element is found!!!" and terminate the function

Step 4 - If both are not matched, then compare search element with the next element in the list.

Step 5 - Repeat steps 3 and 4 until search element is compared with last element in the list.

Step 6 - If last element in the list also doesn't match, then display "Element is not found!!!" and terminate the function.

Linear Search

```
#include <stdio.h>
 int main()
{
  int array[100], search, c, n;
  printf("Enter number of elements in array\n");
  scanf("%d", &n);
  printf("Enter %d integer(s)\n", n);
//Entering array elements
  for (c = 0; c < n; c++)
   scanf("%d", &array[c]);
//Enter number to be searched
 printf("Enter a number to search\n");
  scanf("%d", &search);
```

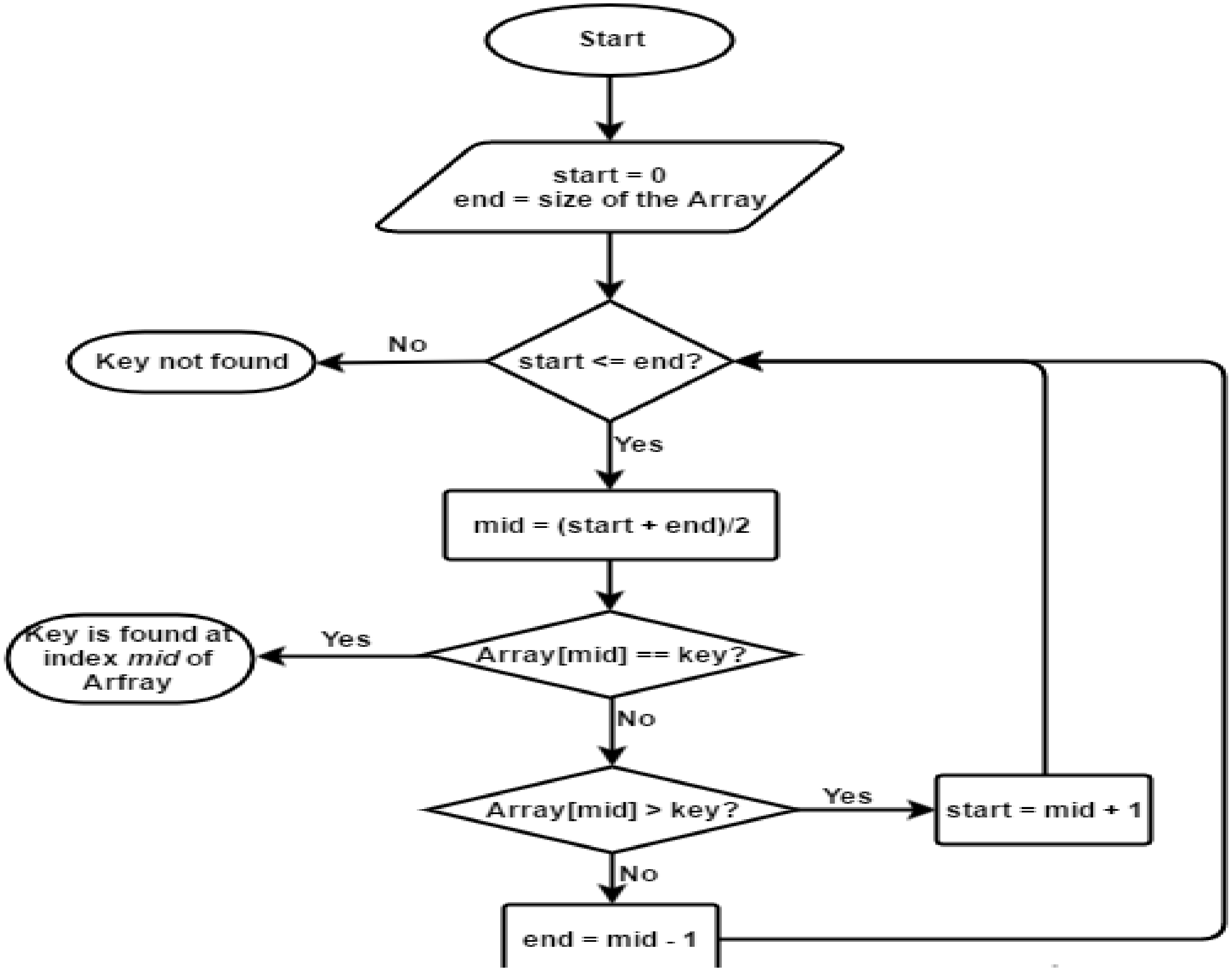
Linear Search

```
//Linear search
for (c = 0; c < n; c++)
  {
    if (array[c] == search)
// If required number is found
{ printf("%d is present at location %d.\n", search, c+1);
break;
    }
  }
// for not found
  if (c == n)
    printf("%d isn't present in the array.\n", search);
  return 0;
}
```

Binary Search

Binary Search

Binary search algorithm: find *key* in a sorted *Array*



Binary Search

```
/* C Program - Binary Search */
#include<stdio.h>
void main()
{
        int n, i, arr[50], search, first, last, middle;
        printf("Enter total number of elements :");
        scanf("%d",&n);
        printf("Enter %d number :", n);
        for (i=0; i<n; i++)
        {
                scanf("%d",&arr[i]);
        }
        printf("Enter a number to find :");
        scanf("%d", &search);
        first = 0;
        last = n-1;
        middle = (first+last)/2;
```

Cont….

```
/* C Program - Binary Search */
          while (first <= last)
          {
                    if(arr[middle] < search)
                    {
                              first = middle + 1;

                    }
                    else if(arr[middle] == search)
                    {
                              printf("%d found at location %d\n", search,
middle+1);
                              break;
                    }
```

Cont….

/* C Program - Binary Search */

Cont….

```
else
                    {
                              last = middle - 1;
                    }
                    middle = (first + last)/2;
          }
          if(first > last)
          {
printf("Not found! %d is not present in the list.",search);
          }
          }
```

Practice questions:

- Deletion of an array element
- Merging two arrays