Using the base address of LA, the computer calculates the address of any element of LA by the formula

$$\mathbf{LOC\ (LA[K])} = \mathit{Base}\mathbf{(LA)} + \mathbf{w(K - lower\ bound)}$$

Where, w is the number of words per memory cell for the array LA.

DYNAMICALLY ALLOCATED ARRAYS

One Dimensional Array

While writing computer programs, if finds ourselves in a situation where we cannot determine how large an array to use, then a good solution to this problem is to defer this decision to run time and allocate the array when we have a good estimate of the required array size.

Example:

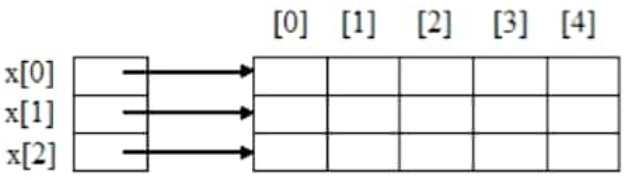
```
int i, n, *list;
printf("Enter the number of numbers to generate:");
scanf("%d", &n);
if(n<1)
        {
                fprintf (stderr, "Improper value of n \n");
                exit(EXIT_FAILURE);
        }
MALLOC (list, n*sizeof(int));
```

The programs fails only when n<1 or insufficient memory to hold the list of numbers that are to be sorted.

Two Dimensional Arrays

C uses array-of-arrays representation to represent a multidimensional array. The two dimensional arrays is represented as a one-dimensional array in which each element is itself a one-dimensional array.

Example: int x[3][5];



Array-of-arrays representation

C find element x[i][j] by first accessing the pointer in x[i].
Where **x[i] = α+ i* sizeof(int)**, which give the address of the zeroth element of row ***i*** of the array.
Then adding **j*sizeof(int)** to this pointer (**x[i]**) **,** the address of the [j]th element of row ***i*** is determined**.**

x[i] = α+ i* sizeof(int)
x[j] = α+ j* sizeof(int)
x[i][j] = x[i]+ i* sizeof(int)

Creation of Two-Dimensional Array Dynamically

```
int **myArray;
myArray  = make2dArray(5,10);
myArray[2][4]=6;

int ** make2dArray(int  rows, int cols)
{ /* create a two dimensional rows X cols array */
        int **x, i;
        MALLOC(x,  rows * sizeof (*x));  /*get memory for row pointers*/
        for (i= 0;i<rows; i++)                  /* get memory for each row */
        MALLOC(x[i],  cols * sizeof(**x));
        return x;
}
```

The second line allocates memory for a 5 by 10 two-dimensional array of integers and the third line assigns the value 6 to the [2][4] element of this array.