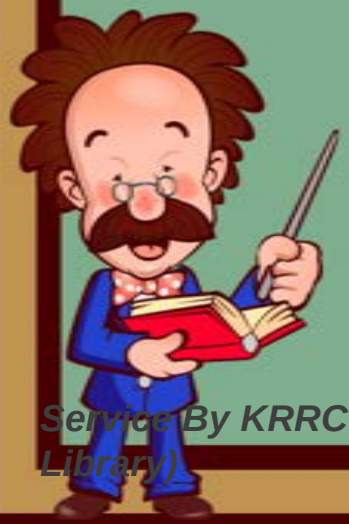
ANJUMAN-I-ISLAM'S
KALSEKAR TECHNICAL CAMPUS
NEW PANVEL
Approved by : All India Council for Technical Education, Council of Architecture, Pharmacy Council of India New Delhi,
Recognised by : Directorate of Technical Education, Govt. of Maharashtra, Affiliated to : University of Mumbai.
SCHOOL OF ENGINEERING & TECHNOLOGY
SCHOOL OF PHARMACY
SCHOOL OF ARCHITECTURE

C - Arrays

Mrs. Apeksha Gopale, Asst. Professor

Department – Computer Engineering



AIKTC SCHOOL OF ENGINEERING

C - Arrays

- Arrays a kind of data structure that can store a fixed-size sequential collection of elements of the same type. An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type.

- Instead of declaring individual variables, such as number0, number1, ..., and number99, you declare one array variable such as numbers and use numbers[0], numbers[1], and ..., numbers[99] to represent individual variables. A specific element in an array is accessed by an index.

    - All arrays consist of contiguous memory locations. The lowest address corresponds to the first element and the highest address to the last element.

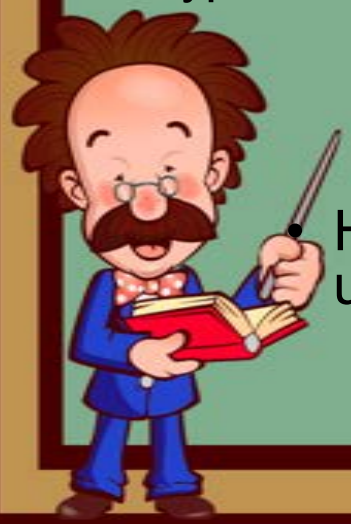
Declaring Arrays

- To declare an array in C, a programmer specifies the type of the elements and the number of elements required by an array as follows −

```
type arrayName [ arraySize ];
```

- This is called a single-dimensional array. The arraySize must be an integer constant greater than zero and type can be any valid C data type. For example, to declare a 10-element array called balance of type double, use this statement −

```
double balance[10];
```



Here balance is a variable array which is sufficient to hold up to 10 double numbers.

Initializing Arrays

- You can initialize an array in C either one by one or using a single statement as follows −

```
double balance[5] = {1000.0, 2.0, 3.4, 7.0, 50.0};
```

- The number of values between braces { } cannot be larger than the number of elements that we declare for the array between square brackets [ ].

- If you omit the size of the array, an array just big enough to hold the initialization is created. Therefore, if you write −

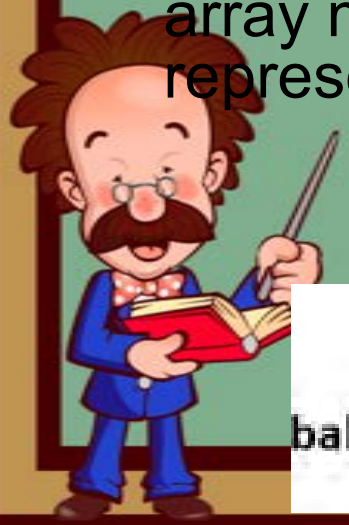
```
double balance[] =
  {1000.0,2.0,3.4,7.0,50.0};
```

Initializing Arrays

- You will create exactly the same array as you did in the previous example. Following is an example to assign a single element of the array −

```
balance[4] = 50.0;
```

- The above statement assigns the 5th element in the array with a value of 50.0. All arrays have 0 as the index of their first element which is also called the base index and the last index of an array will be total size of the array minus 1. Shown below is the pictorial representation of the array we discussed above −



	0	1	2	3	4
balance	1000.0	2.0	3.4	7.0	50.0

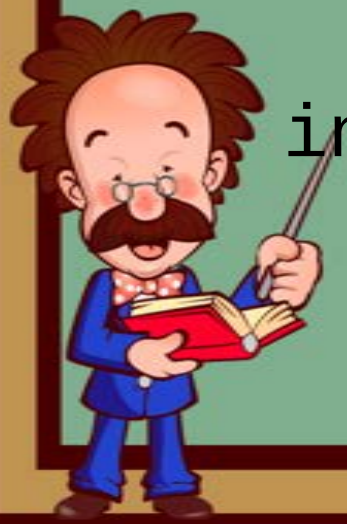
Multi-dimensional Arrays in C

- C programming language allows multidimensional arrays. Here is the general form of a multidimensional array declaration −

```
type name[size1][size2]...[sizeN];
```

- For example, the following declaration creates a three dimensional integer array −

```
int threedim[5][10][4];
```



Two-dimensional Arrays

- The simplest form of multidimensional array is the two-dimensional array. A two-dimensional array is, in essence, a list of one-dimensional arrays. To declare a two-dimensional integer array of size [x][y], you would write something as follows −

```
type arrayName [ x=rows ][ y=cols ];
```

- Where type can be any valid C data type and arrayName will be a valid C identifier. A two-dimensional array can be considered as a table which will have x number of rows and y number of columns. A two-dimensional array a, which contains three rows and four columns can be shown as follows −

	Column 0	Column 1	Column 2	Column 3
Row 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Row 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Row 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]

Initializing Two-Dimensional Arrays

- Multidimensional arrays may be initialized by specifying bracketed values for each row. Following is an array with 3 rows and each row has 4 columns.

```
int a[3][4] = {
   {0, 1, 2, 3} ,   /* initializers for row indexed by 0 */
   {4, 5, 6, 7} ,   /* initializers for row indexed by 1 */
   {8, 9, 10, 11}   /* initializers for row indexed by 2 */
};
```

- The nested braces, which indicate the intended row, are optional. The following initialization is equivalent to the previous example −

```
int a[3][4] = {0,1,2,3,4,5,6,7,8,9,10,11};
```

Accessing Two-Dimensional Array Elements

- An element in a two-dimensional array is accessed by using the subscripts, i.e., row index and column index of the array. For example −

```
int val = a[2][3];
```

- The above statement will take the 4th element from the 3rd row of the array. You can verify it in the above figure.

Passing Arrays as Function Arguments in C

- If you want to pass a single-dimension array as an argument in a function, you would have to declare a formal parameter in one of following three ways and all three declaration methods produce similar results because each tells the compiler that an integer pointer is going to be received. Similarly, you can pass multi-dimensional arrays as formal parameters.
- Way-1

- Formal parameters as a pointer –

```
void myFunction(int *param) {
    .
    .
    .
}
```

- Way-2

- Formal parameters as a sized array –

```
void myFunction(int param[10]) {
    .
    .
    .
}
```

- Way-3

- Formal parameters as an unsized array –

```
void myFunction(int param[]) {
    .
    .
    .
}
```