In C programming, one of the frequent problems is to handle similar types of data. For example, the user want to store marks of 100 students. We can do this by creating 100 variables individually. But this is tedious and impractical and error-prone. Arrays are intended for such uses.

Array

Array is a collection of *homogenous* data stored under one name. The values in an array are called as 'elements' of an array. These elements are accessed by numbers called as 'subscripts' or 'indices'. The index of the first element of an array is 0.

Types of Arrays

Arrays are of two types:

1. One dimensional arrays
2. Multi-dimensional arrays

One dimensional arrays

The array which is used to represent and store data in a linear form is called as 'single' or one dimensional array.

Declaration

Arrays must be declared before they can be used in the program. Standard array declaration is as follows:

```
data_type array_name[size];
```

Here `size` is the number of elements that you plan to store in the array.

int halley [5];

Here, the name of array is halley. The size of array is 5; there are 5 items (elements) of array halley. The first element is halley[0] and the last is halley[4].

You must follow these rules when you declare an array in C:

- The data type can be any valid data type such as **int**, **float**, char, struct or union.

halley[0]	halley[1]	halley[2]	halley[3]	halley[4]

Table 1: Uninitialized array

- The name of an array must follow naming rules of variables.
- The size of the array must be a constant positive integer.

Initialization

Arrays can be initialized at declaration time.

int halley [5] = {2061, 1986, 1910, 1835, 1759};

It is not necessary to define the size of arrays during initialization, *if all the elements you need are being initialized*

int halley [ ] = {2061, 1986, 1910, 1835, 1759};

In this case, the compiler determines the size of array by calculating the number of elements of an array.

halley[0]	halley[1]	halley[2]	halley[3]	halley[4]
2221	2242	2234	2223	2254

Table 2: Initialized array

Accessing array elements

Arrays can be accessed and treated like variables in C. For example,

```
1      scanf(``%d'', &halley[2]);
2      /* ------------------------------------------ *
3       * statement to insert value in the third   *
4       * element of array halley.                  *
5       * ------------------------------------------ */
6
7      scanf(``%d'', &halley[i]);
8      /* ------------------------------------------ *
9       * statement to insert value in (i+1)th     *
```



```
10        * element of array halley.                    *
11        * ------------------------------------------- */
12
13      printf(``%d'', halley[0]);
14      /* ------------------------------------------- *
15       * statement to print first element of        *
16       * the array halley.                             *
17       * -------------------------------------------*/
```

Example

Program to find the sum of marks of 'n' students using arrays

```
1  #include <stdio.h>
2  int main() {
3      int marks[10];
4      int n;
5      int sum = 0;
6      printf("Enter number of students : ");
7      scanf("%d", &n);
8      for (int i = 0; i < n; ++i) {
9          printf("Enter marks of student %d: ", i+1);
10         scanf("%d", &marks[i]);
11         sum += marks[i];
12     }
13     printf("Sum = %d\n", sum);
14     return 0;
15 }
```

Note that if the number of students is more than 10, or if a number more than 10 is entered by mistake, the behaviour of this program is not at all good. This is one of the problem areas of C. While it is solveable, it requires lot of extra, painstaking work and is error-prone.

Multi-Dimensional Arrays

The array which is used to represent and store data in a tabular form is called as 'two dimensional array.' Such type of array specially used to represent data in a matrix form.


```
$
$ c99 Program-10-1.c
$ ./a.out
Enter number of students : 6
Enter marks of student 1: 56
Enter marks of student 2: 67
Enter marks of student 3: 78
Enter marks of student 4: 89
Enter marks of student 5: 90
Enter marks of student 6: 65
Sum = 445
$
```

Figure 1: Sum of marks

In C, we essentially create arrays of arrays to provide multi-dimensional arrays.

`data_type array_name[rows][columns];`

Example

float a [3][6];

Here, a is a multi-dimensional array; specifically a two dimensional array. This has 3 rows and 6 columns.

	col 1	col 2	col 3	col 4	col 5	col 6
row 1	a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]	a[0][5]
row 2	a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]	a[1][5]
row 3	a[2][0]	a[2][1]	a[2][2]	a[2][3]	a[2][4]	a[2][5]

Table 3: Multi-dimensional Array::Layout

Example

Program to find out sum of diagonal elements of the given matrix.

```
1 #include <stdio.h>
2 int main() {
3     int matrix[3][3];
4     int sum_diag = 0;
```



```
5
6       printf("Enter 9 integer elements: \n");
7       for (int i = 0; i < 3; i++) {
8           for (int j = 0; j < 3; j++) {
9               scanf("%d", &matrix[i][j]);
10          }
11      }
12
13      printf("given matrix is: \n");
14      for (int i = 0; i < 3; i++) {
15          for (int j = 0; j < 3; j++) {
16              printf("%3d\t", matrix[i][j]);
17          }
18          printf("\n");
19      }
20
21      for (int i = 0; i < 3; i++) {
22          for (int j = 0; j < 3; j++) {
23              if (i == j)
24                  sum_diag += matrix[i][j];
25          }
26      }
27      printf("Diagonal sum of given matrix is: %d\n" ,sum_diag);
28      return 0;
29  }
```

Of course the computation loop can also be written more simply as:

```
for (int n = 0; n < 3; n++)
    sum_diag += matrix[n][n];
```

Things to note

1. Arrays are very useful *only* when you are working with sequences of the *same* kind of data.
2. Array elements are stored in contiguous memory locations.


```
$
$ c99 Program-10-2.c
$ ./a.out
Enter 9 integer elements:
1 3 5 7 9 8 6 4 2
given matrix is:
  1       3       5
  7       9       8
  6       4       2
Diagonal sum of given matrix is: 12
$
```

Figure 2: Sum of diagonal

3. C does not have bounds checking mechanism for array sizes.
4. We cannot change size of array at the run time.
5. To delete or insert an element we need to traverse throughout the array, and copy the affected elements.

