ARRAYS

Introduction to Arrays

- An Array is a collection of similar data elements
- These data elements have the same data type
- The elements of the array are stored in consecutive memory locations and are referenced by an index

Definition

- An array is a data structure that is used to store data of the same type. The position of an element is specified with an integer value known as index or subscript.

Example

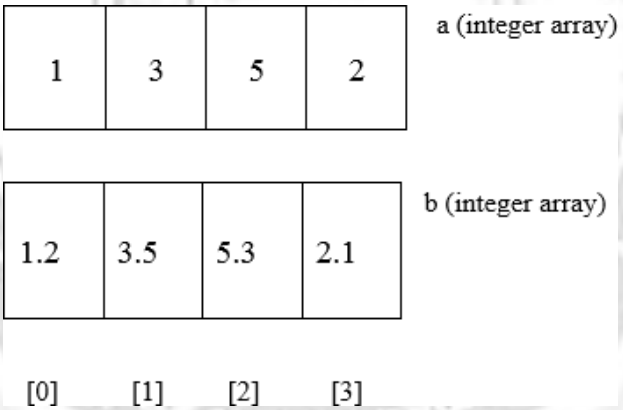


Fig. 1.16 Array Structure

Characteristics

- All the elements of an array share a common name called as array name
- The individual elements of an array are referred based on their position
- The array index in c starts with 0

Advantages of C array

- **Code Optimization** : Less code to access the data
- **Easy to traverse data** : By using the for loop, we can retrieve the elements of can array easily
- **Easy to sort data**: To sort the elements of array, we need a few lines of code only
- **Random Access** : We can access any element randomly using the array

Disadvantages of array

- **Fixed Size**: Whatever size, we define at the time of declaration of array, we can't exceed the limit. So it doesn't grow the size dynamically like Linked List

Classifications

- In general arrays are classified as:
- One-Dimensional Array
- Two-Dimensional Array
- Multi-Dimensional Array

Declaration of an Array

Array has to be declared before using it in C program. Declaring array means specifying three things.

Data_type	Data Type of Each Element of the array
Array_name	Valid variable name
Size	Dimensions of the Array

Arrays are declared using the following syntax:

```
type name[size]
```

Here the type can be either int, float, double, char or any other valid data type. The number within the brackets indicates the size of the array, i.e., the maximum number of elements that can be stored in the array.

Example: i) int marks[10]

ii) int a[5]={10,20,5,56,100}

The declaration of an array tells the compiler that, the data type, name of the array, size of the array and for each element it occupies memory space. Like for int data type occupies 2 bytes for each element and for float occupies 4 bytes for each element etc. The size of the array operates the number of elements that can be stored in an array.

Initialization of arrays

Elements of the array can also be initialized at the time of declaration as in the case of every other variable. When an array is initialized, we need to provide a value for every element in the array. Arrays are initialized using the following syntax:

```
type array_name [size] = { list of values};
```

The values are written with curly brackets and every value is separated by a comma.

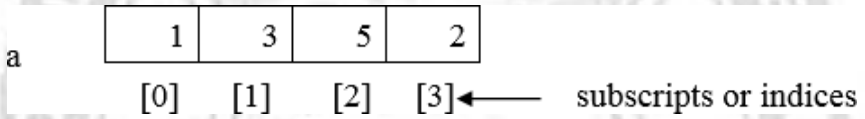
It is a compiler error to specify more number of values than the number of elements in the array.

Example: int marks [5] = {90, 92, 78, 82, 58};

ONE DIMENSIONAL ARRAY

- It is also known as single-dimensional arrays or linear array or vectors
- It consists of fixed number of elements of same type
- Elements can be accessed by using a single subscript

Example



Declaration of Single Dimensional Array

- An array must be declared before being used. Declaring an array means specifying three things.
  1. Data type
  2. Name
  3. Size

Syntax

```
datatype  arrayname [array size];
```

Example

```
int a[4];        // a is an array of 4 integers
char b[6];       //b is an array of 6 characters
```

Initialization of single dimensional array

- Elements of an array can also be initialized. After declaration, the array elements must be initialized otherwise they hold garbage value. An array can be initialized at compile time or at run time.
- Elements of an array can be initialized by using an initialization list. An initialization list is a comma separated list of initializers enclosed within braces.

Example

1. int a[3]={1,3,4};
2. int i[5] ={1, 2, 3, 4, 5};
3. float a[5]={1.1, 2.3, 5.5, 6.7, 7.0};
4. int b[ ]={1,1,2,2};

- In the fourth example the size has been omitted (it can be) and have been declared as an array with 4 elements having 1, 1, 2 and 2 as initial values.
- Character arrays that hold strings allow a shortcut initialization of the form:

char array_name[size]="string"

For example,

char mess[ ]={'w','e','l','c','o','m','e'};

- If the number of initializers in the list is less than array size, the leading array locations gets initialized with the given values. The rest of the array locations gets initialized to

0 - for int array

0.0 - for float array

\0 - for character array

Example

int a[2]={1};

a

1	0

char b[5]={'A'.'r','r'};

b

'A'	'r'	'r'	'\0'	'\0'

Example Programs

Program 1.32

/*Program to find the maximum number in an array * /

```
#include<stdio.h>
void main( )
{
    int a[5], i, max;
```

```
        printf("Enter 5 numbers one by one \n");
        for(i=0;i<5;i++)
        {
                scanf("%d", & a[i]);
        }
        max=a[0];
        for(i=1;i<5;i++)
        {
                if (max<a[i])
                        max =a[i];
        }
                printf("\n The maximum number in the array is %d",max);
        getch( ) ;
}
```

Output:

Enter 5 numbers one by one

5 7 3 6 4

The maximum number in the array is 7

Program 1.33

/*Program for reversing an array*/

```
#include<stdio.h>
void main( )
{
        int a[10], i;
        int n;
        printf("Enter the maximum number of elements\n");
        scanf("%d", &n);
        for(i=0; i<n; i++)
        {
```

```
            scanf("%d",&a[i]);
        }
        printf("Array in the reverse order\n");
        for(i=n–1; i>=0; i--)
        {
        printf("%d\t", a[i]);
        }
        getch( );
    }
```

Output

Enter the maximum number of elements

5 11 12 13 14 15

Array in the reverse order

15 14 13 12 11

Program 1.34

/* Program to calculate sum of array content */

```
    # include<stdio.h>
    void main( )
    {
        int a[20], n, i, sum = 0;
        print f("\n Enter the size of the array:");
        scanf("%d", &n)
        printf ("\n Enter the %d numbers one by one:");
        for (i=0; i<n; i++)
        {
            scanf("%d", &a[i]);
            sum = sum + a[i];
        }
```

```
        printf ("The sum of array content = %d", sum);
}       getch( );
```

Output

Enter the size of the array: 5

Enter the 5 number one by one:

10 20 30 40 50

The sum oif the array content = 150

MULTI-DIMENSIONAL ARRAY

- A multi-dimensional array is an array that has more than one dimension. It is an array of arrays; an array that has multiple levels. The simplest multi-dimensional array is the 2D array, or two-dimensional array and 3D or three-dimensional array.

Two Dimensional Array

- A two dimensional array is an array of one dimensional arrays and can be visualized as a plane that has rows and columns.
- The elements can be accessed by using two subscripts, row subscript (row number), column subscript (column number).
- It is also known as matrix.
- A single dimensional array can store a list of values, whereas two dimensional array can store a table of values.

Example

a[3][5]

1	2	3	6	7
9	10	5	0	4
3	1	2	1	6

Declaration

datatype arrayname [row size][column size]

Example: int a [2][3]; //a is an integer array of 2 rows and 3 columns

Number of elements=2*3=6

Initialization

1. By using an initialization list, 2D array can be initialized.

e.g. int a[2][3] = {1,4,6,2}

a

1	4	6
2	0	0

2. The initializers in the list can be braced row wise.

e.g int a[2][3] = {{1,4,6} , {2}};

Program 1.35

/ * Example for two dimensional array handling * /

```
#include <stdio.h>
void main( )
{
    int a[10][10],i,j,sum,d,n1,n,rowsum,colsum,diasum;
    printf("Enter order[row][col] of the matrix\n");
    scanf("%d %d",&n,&n1);
    printf("Enter %d elements\n",n1*n);
    for(i=0;i<n;i++)
    for(j=0;j<n1;j++)
    scanf("%d",&a[i][j]);
    / * Program module to sum all elements * /
    sum=0;
    for(i=0;i<n;i++)
    for(j=0;j<n1;j++)
    sum+=a[i][j];
    printf("Sum of all elements%d\n",sum);
    / * Program to module to sum row wise */
    for(i=0;i<n;i++)
```

```
{
rowsum=0;
for(j=0;j<n1;j++)
        {
        rowsum+=a[i][j];
        printf("row no = %d sum = %d\n",i,rowsum);

        }
}
/* Program module to sum colwise */
for(i=0;i<n;i++)
        {
        colsum=0;
        for(j=0;j<n1;j++)
        colsum+=a[j][i];
        printf("col no=%d sum=%d\n ",i,colsum);
        }
/ * Program module to sum principle diagonal * /


}
```

Output

```
d
i
a
s
u
m
=
0
;

f
o
```

```
r(i=0;i<n;i++)
for(j=0;j<n1;j++)
if(i==j) diasum+=a[i][j];
printf("Principle diagonal sum %d\n",diasum);
/ * Program module to sum off diagonal */
diasum=0;
for(i=0;i<n;i++)
    {
    j= -n1;
    diasum +=a[i][j];
    }
printf("Off diagonal sum%d\n",diasum);
```

Enter order [row][col] of the matrix

3 3

Enter 9 elements

1 2 3 4 5 6 7 8 9

Sum of all elements 45

row no = 0 sum = 6

row no = 1 sum = 15

row no = 2 sum = 24

col no = 0 sum = 12

col no = 1 sum = 15

col no = 2 sum = 18

Principle diagonal sum 15

Off diagonal sum 15

Three-Dimensional Arrays

Initialization of a 3d array

Initialize a three-dimensional array in a similar way to a two-dimensional array.

Example

```
int test[2][3][4] = {
        {{3, 4, 2, 3}, {0, -3, 9, 11}, {23, 12, 23, 2}},
        {{13, 4, 56, 3}, {5, 9, 3, 5}, {3, 1, 4, 9}}};
```

Program 1.36

Write a C Program to store and print 12 values entered by the user

```
#include <stdio.h>
int main()
{
 int test[2][3][2];
 printf("Enter 12 values: \n");
 for (int i = 0; i < 2; ++i)
 {
  for (int j = 0; j < 3; ++j)
  {
   for (int k = 0; k < 2; ++k)
   {
    scanf("%d", &test[i][j][k]);
   }
  }
 }
 // Printing values with the proper index.
 printf("\nDisplaying values:\n");
 for (int i = 0; i < 2; ++i)
 {
  for (int j = 0; j < 3; ++j)
  {
   for (int k = 0; k < 2; ++k)
   {
    printf("test[%d][%d][%d] = %d\n", i, j, k, test[i][j][k]);
```

```
      }
    }
  }
  return 0;
}
```

Output

```
Enter 12 values:
1
2
3
4
5
6
7
8
9
10
11
12
Displaying Values:
test[0][0][0] = 1
test[0][0][1] = 2
test[0][1][0] = 3
test[0][1][1] = 4
test[0][2][0] = 5
test[0][2][1] = 6
test[1][0][0] = 7
test[1][0][1] = 8
test[1][1][0] = 9
```

test[1][1][1] = 10

test[1][2][0] = 11

test[1][2][1] = 12