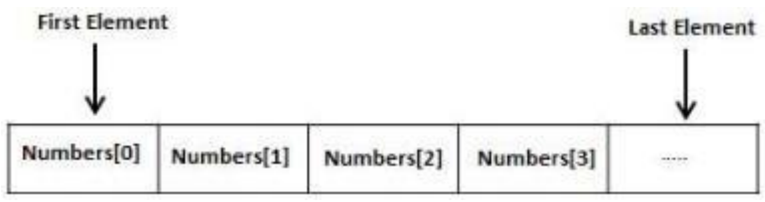
Module 4

CO 4 – Understand and Develop Programs Using One Dimensional and Two Dimensional Arrays

Defining, initializing and accessing of one dimensional arrays – Programs using one dimensional array.
Defining, initializing and accessing of two dimensional arrays – Programs using two dimensional arrays.

Array

- An array is a group (or collection) of same data types stored at contiguous memory locations.
- For example an int array holds the elements of int types while a float array holds the elements of float types.
- It can store primitive types of data like int, char, float, double etc.
- The lowest address corresponds to the first element and the highest address to the last element.
- Array is a derived data type.
- Eg: Set of salaries of a group of employees in an organization.
- Arrays are of three types:
    - One-dimensional arrays
    - Two-dimensional arrays
    - Multidimensional arrays



- A specific element in an array is accessed by an index or subscript of the desired element within a square bracket ([]) after the array name.
- Array subscript must be of integer type.
- Array indices start at zero and stop one less than the array size.

One dimensional array

- A one-dimensional array is like a list.

Array Declaration

- One dimensional array can be declared as:

  Data_type array_name[array_size];
- The array_size must be an integer constant greater than zero and type can be any valid C data type
- The size and type of an array cannot be changed once it is declared.

Eg: int mark[50];

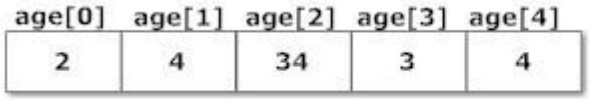
- Here, **int** specifies the type of the array the word **marks** specifies the name of the array variable.
- The **[30]** tells maximum number of elements of the type **int** can be stored in our array. (maximum size of array).

Array Initialization

- An array can be initialized while declaring it like variables.
- Array initialization in C can be either one by one or using a single statement.
- As with traditional methods, all uninitialized values are set to zero.
- If the size of the array is not given, then the largest initialized position determines the size of the array.
- Place the initialization data in curly {} braces following the equals sign.
- An array may be partially initialized, by providing fewer data items than the size of the array. The remaining array elements will be automatically initialized to zero.
- If an array is to be completely initialized, the dimension of the array is not required. The compiler will automatically size the array to fit the initialized data.
- Examples:
  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 age[] = {2, 4, 34, 3, 4};
- In this case, the compiler determines the size of array by calculating the number of elements of an array.

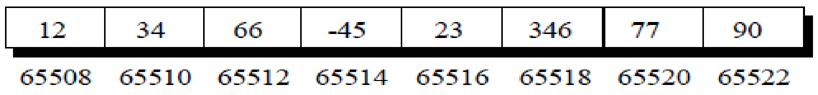


Initialization of one-dimensional array

Array Elements in Memory

Consider the following array initialization:
int arr[8] = {12, 34, 66, -45, 23, 346, 77, 90}

This arrangement of array elements in memory as:



Whatever be the initial values, all the array elements would always be present in contiguous memory locations.

Accessing Elements of an Array

- The arrays can be accessed and treated like variables in C.
- Once an array is declared, the individual elements in the array can be accessed by indexing the array name.
- 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 and the last element is 1 less than the size of the array.
- For example : double salary = balance[9];

  The above statement will take the 10th element from the array and assign the value to salary variable.

```
scanf("%d",&age[2]); -  /* statement to insert value in the third element of array age[]. */
printf("%d",age[0]); - /* statement to print first element of an array. */
```

The following example shows how to use all the three above mentioned concepts - declaration, assignment, and accessing arrays −

```
#include <stdio.h>
 int main () {
   int n[ 10 ];    /* n is an array of 10 integers */
   int i,j;

   /* initialize elements of array n to 0 */
   for ( i = 0; i < 10; i++ ) {
      n[ i ] = i + 10; /* set element at location i to i + 10 */
   }

   /* output each array element's value */
   for (j = 0; j < 10; j++ ) {
      printf("Element[%d] = %d\n", j, n[j] );
   }
    return 0;
}
```

Entering Data into an Array

Here is the section of code that places data into an array (for eg, marks of 5 students)

```
printf ( "\nEnter marks " ) ;
for ( i = 0 ; i <=5 ; i++ )
{
        scanf ( "%d", &marks[i] ) ;
}
```

Reading Data from an Array

Suppose we want to find the average of marks of all students

```
for ( i = 0 ; i <= 5 ; i++ )
{
        sum = sum + marks[i] ;
}
avg = sum / 5 ;
printf ( "\nAverage marks = %d", avg ) ;
```

Example Programs

1. Program to find the average of n numbers using arrays

```
#include <stdio.h>
int main() {

  int num[10], i, n, sum = 0, average;

  printf("Enter number of elements: ");
  scanf("%d", &n);

  for(i=0; i < n; ++i) {
        printf("Enter number%d: ",i+1);
        scanf("%d", &num[i]);

        // adding integers entered by the user to the sum variable
         sum += num[i];
  }

  average = sum / n;
printf("Average = %d", average);

return 0;
}
```

2. Write a C program to read n number of values in an array and display it in reverse order.

```
#include <stdio.h>
void main()
{
  int i,n,a[100];
   printf("\n\nRead n number of values in an array and display it in reverse order:\n");

  printf("Input the number of elements to store in the array :");
                          scanf("%d",&n);
printf("Input %d number of elements in the array :\n",n);
for(i=0;i<n;i++)
        scanf("%d",&a[i]);

printf("\nThe values store into the array are : \n");
for(i=0;i<n;i++)
  {
          printf("\t% d",a[i]);
         }

printf("\n\nThe values store into the array in reverse are :\n");
for(i=n-1;i>=0;i--)
   {
          printf("\t%d",a[i]);
         }
}
```

3. Write a program t o sort numbers in descending order using bubble sort.

```
#include <stdio.h>
int main()
{
        int array[100], n, c, d, temp;
        printf("Enter number of elements:");
        scanf("%d", &n);
        printf("Enter %d integers\n" , n);
        for(c=0; c<n; c++)
                scanf("%d", &array[c]);
        for(c=0; c < n-1; c++)
        {
                for(d=c +1; d <n; d++)
                {
                        if (array[c]<array[d])
                        {
                                temp = array[c];
                                array[c] = array[d];
                                array[d] = temp;
                        }
                }
        }
        printf("Sorted array i n descending order i s:\n");
        for(c=0; c<n; c++)
                printf("%d\n", array[c]);
        return 0;

}
```

Exercise

1. Program to find the average of N even numbers in an array.
2. Program to sort N numbers in ascending order.
3. Program to search an element in an array.
4. Program to find the highest and lowest element in an array.
5. Write a C program to insert an element at a particular position in an integer array.

Two dimensional array

- Also called matrix.
- It can be defined as an array of arrays.
- It is organized as matrices which can be represented as the collection of rows and columns.
- A two dimensional array can be considered as a collection of a number of one dimensional arrays

Declaration of two dimensional Array

The syntax for declaring a Two-dimensional array

data_type array_name[rows][columns];

Eg: int A[10][5];

Here, the two-dimensional array is named as A which has 10 rows and 5 columns.

Initializing two dimensional arrays

Two ways to initialize a two dimensional array during declaration.

Method 1

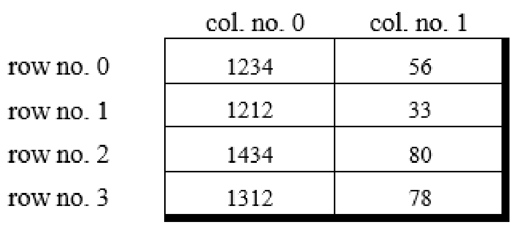
To initialize a two dimensional array A of size x * y, without using any nested brace.

Syntax : int A[x][y] = {element 1, element 2, ... element xy}

Example :

int stud[4][2]= { 1234,56, 1212,33, 1434,80, 1312,78 };

Conceptually, the array arrangement can be shown as follows:



	col. no. 0	col. no. 1
row no. 0	1234	56
row no. 1	1212	33
row no. 2	1434	80
row no. 3	1312	78

Here, 1234 is stored in stud[0][0], 56 is stored in stud[0][1] and so on.

- The above arrangement highlights the fact that a two- dimensional array is nothing but a collection of a number of one- dimensional arrays placed one below the other.

- The arrangement of array elements in a two-dimensional array of stud, which contains roll nos. In one column and the marks in the other.
- The array arrangement shown above is only conceptually true. This is because memory doesn't contain rows and columns.
- In memory whether it is a one-dimensional or a two-dimensional array the array elements are stored in one continuous chain.
- The arrangement of array elements of a two-dimensional array in memory is shown below:

s[0][0]	s[0][1]	s[1][0]	s[1][1]	s[2][0]	s[2][1]	s[3][0]	s[3][1]
1234	56	1212	33	1434	80	1312	78
65508	65510	65512	65514	65516	65518	65520	65522

Method 2

A two-dimensional array in C can also be initialized using nested braces, which makes the visualization of each row and column a bit easier.

Syntax is : int Arr[x][y] = {{ele 1, ele 2, .. ele y} , {......} , {..., ele xy-1, ele xy}};

Eg:

```
        int stud[4][2]={
        { 1234, 56 },
        { 1212, 33 },
        { 1434, 80 },
        { 1312, 78 },
    }
```

Each nested brace denotes a single row, with the elements from left to right being the order of elements in the columns in 2D array.

Thus, the Number of nested braces = the Number of rows.

Accessing Two Dimensional Array Elements

- To access the Two Dimensional Array elements using indexes.
- Using the index, we can access or alter/change each element present in the array separately.
- Index value starts at 0 and ends at n-1, where n is the size of a row or column.

For example, if an array of Student[8][5] will stores 8 row elements and 5 column elements. To access or alter 1st value use Student[0][0], to access or alter 2nd row 3rd column value then use Student[1][2] and to access the 8th row 5th column then use Student[7][4].

Example : Program to create a square matrix of order m x n

```
main()
{
    int mat[10][10], m,n,i,j;
    printf(“enter the order of matrix m and n”);
    scanf(“%d%d”,&m,&n);
    printf(“enter the elements of matrix”);
    for(i=0;i<m;i++)
            for(j=0;j<n;j++)
                    scanf(“%d”,&mat[i][j]);
//display the matrix
printf(“the matrix you created is”);
    for(i=0; i<m; i++)
    {
            for(j=0; j<n; j++)
            {
                    printf(“%d ”, mat[i][j]);
            }
            printf(“\n”);
    }
}
```