

An array stores multiple values in a single variable. Arrays are a kind of data structure that can store a fixed-size sequential collection of elements of the same type (homogeneous) values. 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.

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 number

Arrays are of two types:

- 1) One-dimensional arrays
- 2) Multidimensional arrays

One dimensional array:

Declaration of one-dimensional array:

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

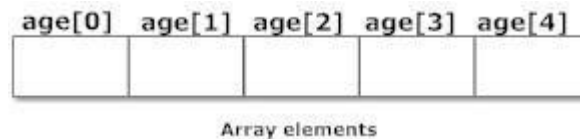
Syntax: `data_type ArrayName [ArraySize];`

Example: `int age [5];`

Here, the name of array is age. The size of array is 5, i.e., there are 5 items (elements) of array age. All elements in an array are of the same type (`int`, in this case).

Array elements

Size of array defines the number of elements in an array. Each element of array can be accessed and used by user according to the need of program. For example: `int age[5];`



Note that, the first element is numbered 0, second element 1 and so on

Initialization of one-dimensional array:

Arrays can be initialized at declaration time in this source code as:

```
int age[5] = {12,14,34,13,19};
```

It is not necessary to define the size of arrays during declare with initialization but data not initialize declare time in array then array size must be define.

```
int age[] = {12,14,34,13,19}; // array declare with initialization
int age[5]; // array declare without initialization
```

Accessing array elements

In C/C++ programming, arrays can be accessed and treated like variables in C/C++.

Example:

```
#include <conio.h>
#include <stdio.h>
void main()
{
    int age[5]={12,14,34,13,19};
    int i;
    for (i=0; i<=4; i++)
    { printf(" Age is %d = %d Years \n ",i ,age[ i ]);
    }
}
```

The following example creates an indexed array If you have a list of students record items (a list of Roll Number, Name and Grade, storing the Rollo , Name and Grade in three array variables to it, and then prints containing the array values.

```
#include <conio.h>
#include <stdio.h>
void main()
{
    int RollNo[3]={21,23,14};
    char Name[3][15] = {"Mansoor", "Muhammad","Ali Imran"};
    char Grade[3]= { 'C', 'A', 'D' };
    printf (" 1st student Roll Number is %d, Name is %s and Grade is %c \n",RollNo[0], Name[0], Grade[0] );
    printf (" 2nd student Roll Number is %d, Name is %s and Grade is %c \n",RollNo[1], Name[1], Grade[1]);
    printf (" 3rd student Roll Number is %d, Name is %s and Grade is %c \n",RollNo[2], Name[2], Grade[2]);
}
```

The index can be assigned manually but you cannot assign string value like this:

```
RollNo[2] = 67;
Grade[2] = 'B';
Name[2] = "Farhan"; // but you cannot assign string value in this way
```

Here you will get program for linear search in C/C++. In linear search Program, we compare targeted element with each element of the array. If the element is found then its print student record is displayed.

```
#include <conio.h>
#include <stdio.h>

void main()
{
    int RollNo[3] = {21,23,14};
    char Name[3][15] = {"Mansoor", "Muhammad", "Ali Imran"};
    char Grade[3] = { 'C', 'A', 'D' };
    int i, Roll, found=0;

    clrscr();
    printf("Enter Roll Number to be find ");
    scanf("%d",&Roll);

    for (i=0 ; i<=2; i++)
    {
        if ( Roll == RollNo [ i ] )
        {
            printf("\n\n Student Roll Number is....: %d\n",RollNo[ i ]);
            printf(" Student Name is .....: %s\n",Name[ i ]);
            printf(" Student Grade is .....: %c\n",Grade[ i ]);
            found =1;
        }
    }
    if (found == 0)
        printf("Record is Not Found ");
}
```

```
DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC
Enter Roll Number to be find 21

Student Roll Number is...: 21
Student Name is .....: Mansoor
Student Grade is .....: C
```

Multi-Dimensional array

In C/C++, we can create an array of an array, known as a multidimensional array. There are two Types Dimensional array double (2D) dimensional and three (3D) dimensional array.

Two (2D) dimensional Array

The two dimensional arrays are such type of arrays which stores an multiple column at each row and column index number instead of single element. It can be created using nested array. These type of arrays can be used to store similar type of elements, but the index is always a number.

Declaration of Two-Dimensional array:

Syntax:

Data_type *arrayName* [*num_of_rows*][*num_of_column*];

OR

Data_type *arrayName* [*y*][*x*];

Where data type can be any valid C/C++ data type and **arrayName** will be a valid C identifier. A two dimensional array can be considered as a table which will have y number of rows and x number of columns. A two-dimensional array **a**, which contains three rows and four columns can be shown as follows. `int a[3][4]`

	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:

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

Example:

```
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
}
```

	Column 0	Column 1	Column 2	Column 3
Row 0	1	2	3	4
Row 1	5	6	7	8
Row 2	9	10	11	12

The nested braces are optional. The following initialization is equivalent to it

```
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.

```
int val1, val2;
```

```
val1 = a[0][0]; // value is 1
val2 = a[1][2]; // value is 7
```

	Column 0	Column 1	Column 2	Column 3
Row 0	1	2	3	4
Row 1	5	6	7	8
Row 2	9	10	11	12

The above 1st statement will take the element from the 0 row and 0 column access value is 1 of the array value put in **val1** variable.

The above 2nd statement will take the element from the 1 row and 2 column access value is 7 of the array value put in **val2** variable

Two Dimensional array Example

```
#include <conio.h>
```

```
#include <stdio.h>
```

```
void main ()
```

```
{
```

```
/* an array with 5 rows and 2 columns*/
```

```
int TwoD_Array[5][2] = { {0,1}, {1,2}, {2,4}, {3,6},{4,8}};
```

```
int row, col;
```

```
/* output each array element's value */
```

```
for ( row = 0; row < 5; row++ )
```

```
{
```

```
    for ( col = 0; col < 2; col++ )
```

```
    {
```

```
        printf("TwoD_Array[%d][%d] = %d\n", row,col, TwoD_Array[row][col] );
```

```
    }
```

```
}
```

```
}
```



```
DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC
TwoD_Array[0][0] = 0
TwoD_Array[0][1] = 1
TwoD_Array[1][0] = 1
TwoD_Array[1][1] = 2
TwoD_Array[2][0] = 2
TwoD_Array[2][1] = 4
TwoD_Array[3][0] = 3
TwoD_Array[3][1] = 6
TwoD_Array[4][0] = 4
TwoD_Array[4][1] = 8
```

Passing an Array to Function

When you pass an array, you actually the address of the array. To pass an entire array to a function, only the name of the array is passed as an argument. However, notice the array name with use of [] in the function definition. This informs the compiler that you are passing a one-dimensional array to the function. Notice that only the array name is uses in the calling argument without a brackets and subscript. You are actually passing the address of the array you do not have to specify he number of elements in the receiving argument just empty brackets [] to indicate an array. Actually array name of main function to be change in the function definition.

Example

```
#include <conio.h>
```

```
#include <stdio.h>
```

```

void Salary_Increase(int Sal[ ])
{
    int row;

    for ( row = 0; row < 5; row++ )
    {
        Sal[row] = Sal[ row ]+2000;
    }
}

void main ()
{
    int Salary[5] = { 10000,15000,20000,25000};
    int row;
    clrscr();
    puts (" Salary display before increasing salary function call \n ");
    for ( row = 0; row < 5; row++ )
    {
        printf(" Employee salary is %d \n ", Salary[ row ]);
    }

    Salary_Increase (Salary);    // Function call
    puts (" Salary display After increasing salary function call \n ");
    for ( row = 0; row < 5; row++ )
    {
        printf(" Employee salary is %d \n ", Salary[ row ]);
    }
}

```

Not: Actually array name **Salary** of main function to be change **Sal** in the calling function definition.

```

DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC
Salary display before increasing salary function call
Employee salary is 10000
Employee salary is 15000
Employee salary is 20000
Employee salary is 25000
Employee salary is 0
Salary display After increasing salary function call
Employee salary is 12000
Employee salary is 17000
Employee salary is 22000
Employee salary is 27000
Employee salary is 2000

```

The following program example to input income and expenses of seven days and calculate sum of total income and total expenses with the help of double dimension array.

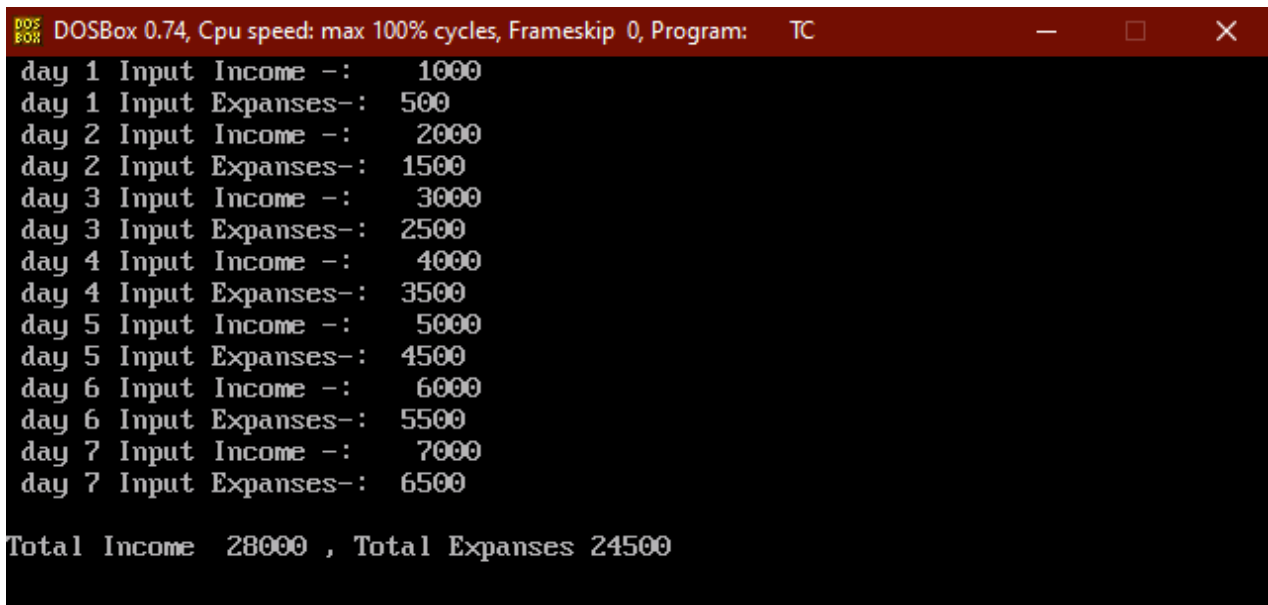
```
#include <conio.h>
#include <stdio.h>

void main ()
{
    int Income_Expances[7][ 2];
    int row,TotIncome,TotExpances;

    TotIncome = TotExpances =0;
    clrscr();

    for ( row = 0; row < 7; row++)
    {
        printf (" day %d Input Income -: ",row+1); scanf("%d",&Income_Expances[ row ][0]);
        printf (" day %d Input Expances-: ",row+1); scanf("%d",&Income_Expances[ row ][1]);
        TotIncome += Income_Expances[ row ][0];
        TotExpances += Income_Expances[ row ][1];
    }

    printf("\nTotal Income %d , Total Expances %d \n ", TotIncome, TotExpances);
}
```



```
DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC
day 1 Input Income -: 1000
day 1 Input Expances-: 500
day 2 Input Income -: 2000
day 2 Input Expances-: 1500
day 3 Input Income -: 3000
day 3 Input Expances-: 2500
day 4 Input Income -: 4000
day 4 Input Expances-: 3500
day 5 Input Income -: 5000
day 5 Input Expances-: 4500
day 6 Input Income -: 6000
day 6 Input Expances-: 5500
day 7 Input Income -: 7000
day 7 Input Expances-: 6500

Total Income 28000 , Total Expances 24500
```


Exercise

Theory Questions

1. What is array?
2. What is the difference between a single dimensional array and a multidimensional array?
3. Describe the two (2D) dimension array.

Practical Questions

1. What would be the output of the following Program?

```
void main(){
    int salary[ ] = {50000,40000,20000,30000,60000};
    for (int i=4; i>=0; i--)
    {
        printf( " Employee %d , and Salary is %d \n ",i, salary[ i ]);
    }
}
```

2. Write a program that takes 10 integers as input and prints their sum by using array.
3. Create a Mark-Sheet for 10 students using arrays which contains the following;

Roll Number

Name

Marks of 3 subjects (Maths, English and Urdu)

and display following form

Student List Of Mark sheet

Roll #	Student Name	Math's	English	Urdu	Obtain Marks	Percentage	Grade
--------	--------------	--------	---------	------	--------------	------------	-------

99	xxxxxxxxxxx	99	99	99	999	99.99	xx
----	-------------	----	----	----	-----	-------	----

99	xxxxxxxxxxx	99	99	99	999	99.99	xx
----	-------------	----	----	----	-----	-------	----

99	xxxxxxxxxxx	99	99	99	999	99.99	xx
----	-------------	----	----	----	-----	-------	----

:

4. Write a program to sort an integer array in ascending order.
5. Write a program that adds up two 2x2 arrays A and B stores the sum in third array C.

Objective MCQ's

- Which declaration of array is correct?
 - `int Number[];`
 - `Number int[];`
 - `int Number[5];`
 - `int Number(5);`
- The first element number start always from ____ index number.
 - 1
 - 0
 - 2
 - NULL
- What is the correct syntax for declaring and initializing an associative array?
 - `char Names[][] = array{"Asif" - "Muhammad" - "Kamran" - "Umer Ahmed"};`
 - `char Names[4][10] = {Asif , Muhammad , Kamran , Umer Ahmed};`
 - `char Names[4][10] = ("Asif" , "Muhammad" , "Kamran" , "Umer Ahmed");`
 - `char Names[4][10] = {"Asif" , "Muhammad" , "Kamran" , "Umer Ahmed"};`
- When you pass an array as an argument to a function, what is actually passed?
 - The values of all elements in array
 - The address of the array
 - The address of the first element in the array
 - The value of first element in the array
- The following declares a two dimensional float array name **No** with 5 rows and 3 column
 - `float No[3,5];`
 - `float No[5,3];`
 - `float No[3][5];`
 - `float No[5][3];`
- An array is a collection of variable's in C/C++
 - Different data types scattered throughout memory.
 - Similar data types scattered throughout memory.
 - Similar data types place continuously in memory.
 - Different data types place continuously in memory.
- Which is last element of array, if we declared integer array `int Salary [10];` ?
 - `Salary[0];`
 - `Salary[1];`
 - `Salary[10];`
 - `Salary[9];`