

Arrays

What is Array?



- An array is a fixed-size sequential collection of elements of same data types that share a common name.
- It is simply a group of data types. An array is a derived data type.
- An array is used to represent a list

Example of Arrays:



- For Example :

1. List of employees in an organization.
2. Test scores of a class of students.
3. List of customers and their telephone numbers.
4. List of students in the college.

- For Example, to represent 100 students in college , can be written as

student [100]

- Here student is a array name and [100] is called index or subscript.

Types of Arrays



- Types of Array :

1. One-dimensional arrays
2. Two-dimensional arrays
3. Multidimensional arrays

One-dimensional Arrays.

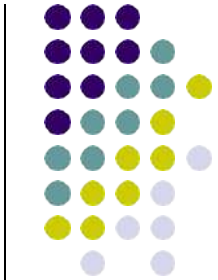


- A variable which represent the list of items using only one index (subscript) is called one-dimensional array.

- For Example , if we want to represent a set of five numbers say(35,40,20,57,19), by an array variable number, then number is declared as follows

```
int number [5] ;
```

One-dimensional Arrays.



- and the computer store these numbers as shown below :

  number [0]

  number [1]

  number [2]

  number [3]

  number [4]

- The values can be assigned to the array as follows : number [0] = 35;

  number [1] = 40;

  number [2] = 20;

  number [3] = 57;

  number [4] = 19;

DECLARATION OF ONE-DIMENSIONAL ARRAYS :

- The general form of array declaration is :

  type *array-name[size];*

- Here the type specifies the data type of elements contained in the array, such as int, float, or char.
- And the size indicates the maximum numbers of elements that can be stored inside the array.
- The size should be either a numeric constant or a symbolic constant.

Example

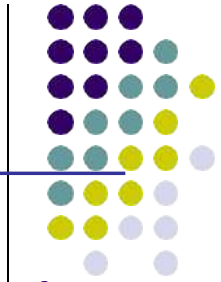


- For example :

  int group [10] ;

- here int is type, group is a variable name , 10 is a size of array and the subscripts (index) is start from 0 to 9.

INITIALIZATION OF ONE-DIMENSIONAL ARRAYS :



- An array can be stored in following stages :

1. At compile time
2. At run time

- Compile time initialization :

- In compile time initialization, the array is initialized when they are declared.

- The general form of initialization of array is : type array-name[size] = { list of values }; The list of values are separated by commas.

Compile time initialization



Example :

```
int number[3] = {4,5,9};
```

- Here array number of size 3 and will assign 4 to first element(number[0]), 5 is assign with second element(number[1]) and 9 is assign with third element(number[2]).

- If the number of values in the list is less than the number of elements, then only that many elements will be intialized. The remaining elements will be set to zero automatically.

Compile time initialization



Example :

```
int number[ ] = {1,2,3,4};
```

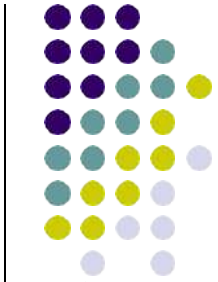
- The character array can be initialized as follows :

```
char name[ ] = {'j','o','h','n','\0'};
```

- The character array can also be initialized as follows :

```
char name[ ] = "john";
```

Run time initialization :



- In run time initialization, the array is explicitly initialize at run time.
- This concept generally used for initializing large
- arrays. Example:

```
for(i=0; i < 100; i++)
{
        if( i < 50)
        sum[i] = 0.0;
        else
        sum[i] = 1.0;
}
```

- Here first 50 elements of the array sum are initialized to 0 time.



TWO DIMENSIONAL ARRAY

CONTENT



- *Introduction to two dimensional array*
- *Declaration*
- *Initialization*
- *Input and output of a 2d array*
- *Storage allocation*

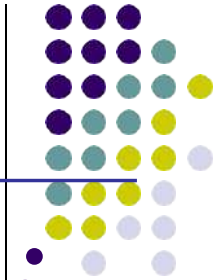
Two-dimensional Arrays



- A variable which represent the list of items using two index (subscript) is called two-dimensional array.
- In Two dimensional arrays, the data is stored in rows and columns format.
- For example:

```
int table[2][3];
```

DECLARATION OF TWO-DIMENSIONAL ARRAYS :



- The general form of two dimensional array declaration is :

  type *array-name[row_size][column_size];*

- Here the type specifies the data type of elements contained in the array, such as int, float, or char.
- The size should be either a numeric constant or a symbolic constant.

Two-Dimensional Arrays



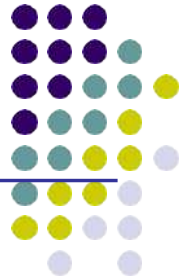
- **Two dimensional arrays are declared similarly to one dimensional arrays:**

```
char t[20][10];
```

(and referred to as a matrix, or as a "table")

The first subscript gives the row number, and the second subscript specifies column number.

INITIALIZATION OF TWO-DIMENSIONAL ARRAYS :



- The general form of initializing two-dimensional array is :

  type array-name[row_size][column_size] = {list of values};

- Example :

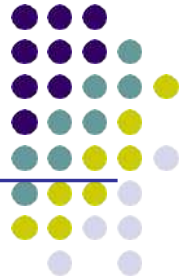
  int table[2][3] = {0,0,0,1,1,1};

- Here the elements of first row initializes to zero and the elements of second row initializes to one.
- This above statement can be written as :

  int table[2][3] = {{0,0,0}, {1,1,1}};

- In two-dimensional array the row_size can be omitted.

INITIALIZATION OF TWO-DIMENSIONAL ARRAYS :



- Example :

  int table[ ][3] = {{0,0,0}, {1,1,1}};

- If the values are missing in an initializer, they are automatically set to zero.

- Example :

  int table[2][3] = {1,1,2};

- Here first row initialize to 1,1 and 2, and second row initialize to 0,0 and 0 automatically.

Declaration and Initialization



We can also initialize a two-dimensional array in a declaration statement; E.g.,

```
int m[2][3] = {{1, 2, 3}, {4, 5, 6}};
```

which specifies the elements in each row of the array (the interior braces around each row of values could be omitted).The matrix for this example is

$$m = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

Declaration and Initialization



Specification of the number of rows could be omitted. But other subscript specifications can not be omitted. So, we could also write the initialization as

```
int m[ ][3] = {{1, 2, 3}, {4, 5, 6}};
```

Memory Storage for a Two Dimensional Array

```
int a[2][3]= {1,2,3,4,5,6};
```

specifies 6 integer locations.

Storage for array elements are in contiguous locations in memory in row major order (unlike column major order in Fortran), referenced by subscripts(or index) values starting at 0 for both rows and columns.

RAM

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

Memory Layout of Two-dimensional array :



- In Two dimensional arrays, the data is stored in rows and columns format.
- For example:

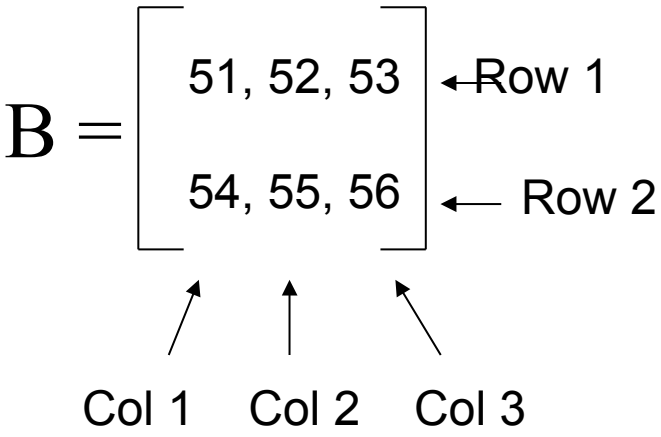
```
int table[2][3] = {1,2,3,4,5,6};
```

- The memory layout of above example :

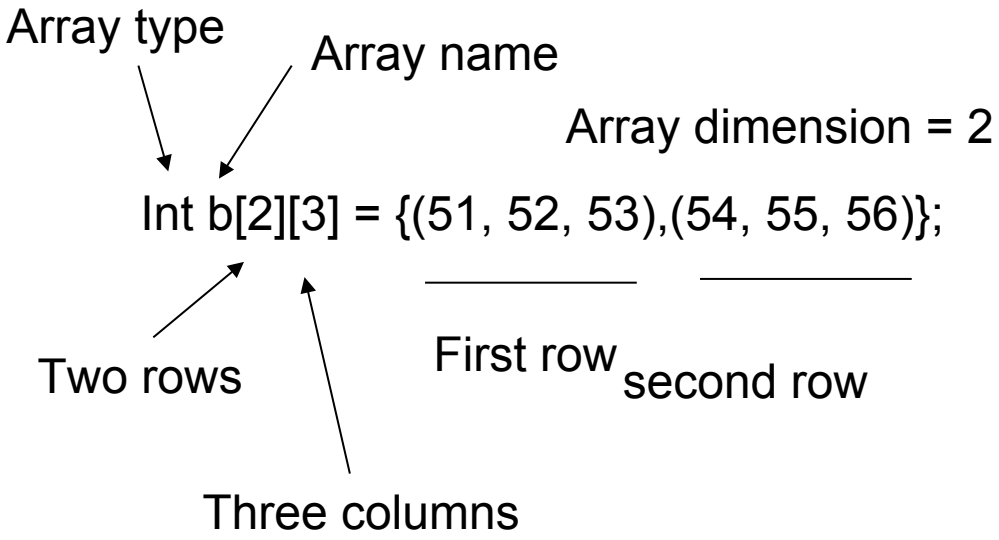
table[0][0] = 1;
table[0][1] = 2
table[0][2] = 3;
table[1][0] = 4;
table[1][1] = 5;
table[1][2] = 6;

Two - Dimensional Arrays

- What is a Two-dimensional array?



Algebraic notation



C notation

Indexes in 2D arrays



- Assume that the two dimensional array called `val` is declared and looks like the following:

`val`	Col 0	Col 1	Col 2	Col 3
Row 0	8	16	9	52
Row 1	3	15	27	6
Row 2	14	25	2	10

- To access the cell containing 6, we reference `val[1][3]`, that is, row 1, column 3.

DECLARATION



- How to declare a multidimensional array?

  int b[2][3];

  the name of the array to be b

  the type of the array elements to be int

  the dimension to be 2 (two pairs of brackets

  []) the number of elements or size to be 2*3 =

  6

Declaration Statement



datatype var_name[row_size][col_size]

for example

int a[3][3];

int a[3][3]

The elements of the fourth row of array can be accesed as follow

```
for(j=0;j<3;j++)
    ...a[3][j]....
```

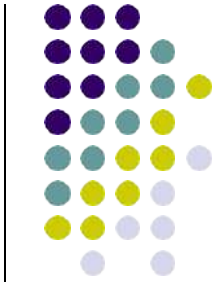
The elements of the third column of the array can be accesed as follow:

```
for(i=0;i<3;i++)
    ...a[i][3]....
```

All of the elements of the array can be accesed by rows as follow:

```
for(i=0;i<3;i++)
{
        for(j=o;j<3;j++)
        {
                .....a[i][j]....
        }
}
```

INITIALIZATION



- How to initialize a Two-Dimensional array?
  - Initialized directly in the declaration statement
    - int b[2][3] = {51, 52, 53, 54, 55, 56};
    - b[0][0] = 51 b[0][1] = 52 b[0][2] = 53
  - Use braces to separate rows in 2-D arrays.
    - int c[4][3] = {{1, 2, 3},
      {4, 5, 6},
      {7, 8, 9},
      {10, 11, 12}};
      - int c[ ][3] = {{1, 2, 3},
        {4, 5, 6},
        {7, 8, 9},
        {10, 11, 12}};

Implicitly declares the number of rows to be 4.

Input of Two-Dimensional Arrays



- Data may be input into two-dimensional arrays using nested *for* loops interactively or with data files.
- A nested for loop is used to input elemts in a two dimensional array.
- In this way by increasing the index value of the array the elements can be entered in a 2d array.

Output of Two-Dimensional Arrays



- The output of two-dimensional arrays should be in the form of rows and columns for readability. Nested *for* loops are used to print the rows and columns in row and column order.

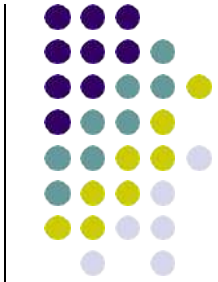
- By increasing the index value of the array the elements stored at that index value are printed on the output screen.

A program to input elements in a two dimensional array and print it.



```
#include<stdio.h>
#include<conio.h>
void main()
{
   int a[3][3];
   int i,j;
   clrscr();
    printf("enter the elements in the
    array:");
```

```
 for(i=0 ; i<3 ; i++)
    {
          for(j=0 ; j<3 ; j++)
          {
               scanf("%d",&a[i][j]);
          }
     }
for(i=0 ; i<3 ; i++)
    {
          for(j=0 ; j<3 ; j++)
          {
               printf("%d",a[i][j]);
          } printf("\n");
     }
 getch();
}
```



OUTPUT :-

Enter elements in array:

1

2

3

4

5

6

7

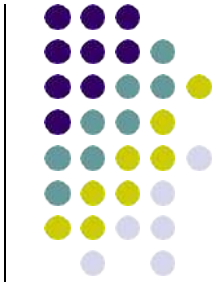
8

9

123

456

789



Storage Allocation



In storage allocation of array contagious memory is allocated to all the array elements.

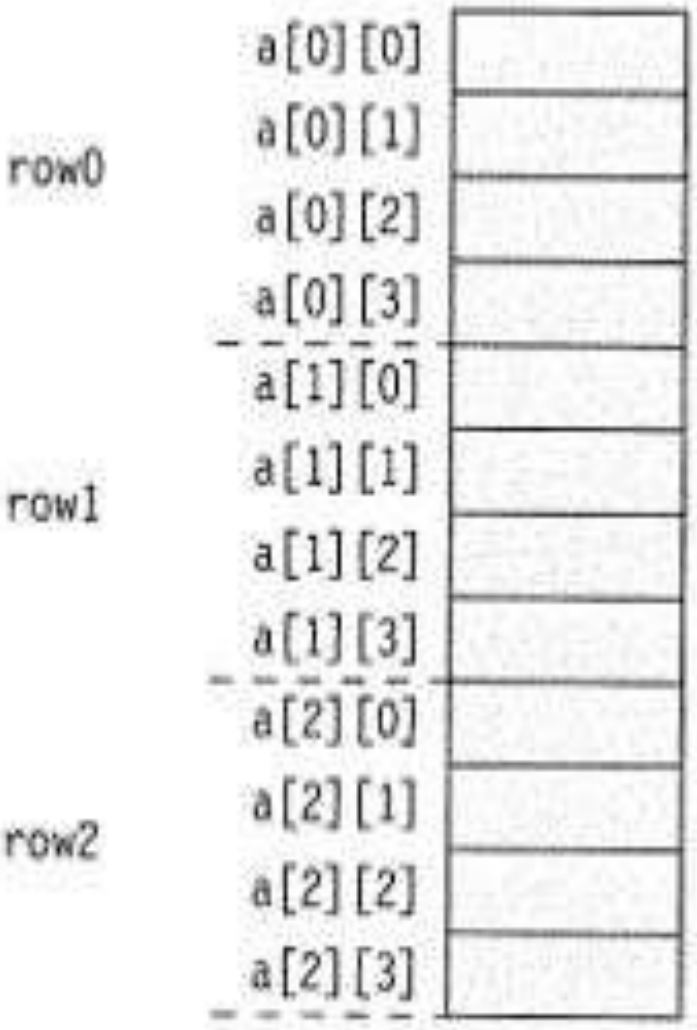
The storage arrangement shown in this example uses the array subscript, also called the array indices.

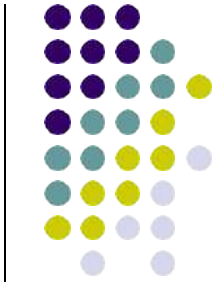
Array declaration: `int a[3][4];`

Array elements:

a[0][0]	a[0][1]	a[0][2]	a[0][3]
a[1][0]	a[1][1]	a[1][2]	a[1][3]
a[2][0]	a[2][1]	a[2][2]	a[2][3]

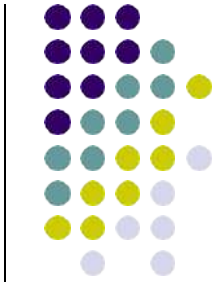
Storage allocation:





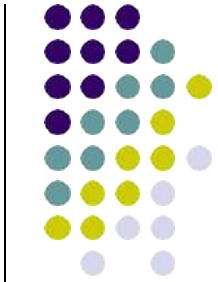
EXAMPLES BASED ON TWO-DIMENSIONAL ARRAY

A program to add two matrix entered by the user and print it.



```
#include<stdio.h>
#include<conio.h>
void main()
{
    int a[3][3],b[3][3],c[3][3];
    int i,j;
    clrscr();
    printf("enter the elements in both the array:");
```

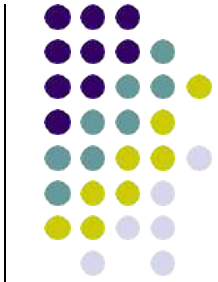
```
for(i=0 ; i<3 ; i++)
    {
        for(j=0 ; j<3 ; j++)
        {
            scanf("%d",&a[i][j]);
        }
     }
for(i=0 ; i<3 ; i++)
    {
        for(j=0 ; j<3 ; j++)
        {
            scanf("%d",&b[i][j]);
        }
    }
```



```
for(i=0 ; i<3 ; i++)
    {
        for(j=0 ; j<3 ; j++)
        {
            c[i][j]=a[i][j]+b[i][j]
            ;
            printf("%d",c[i][j]);
        }
        printf("\n");
     }
 getch();
}
```



OUTPUT:-



Enter elements in array:-

1
2
3
4
5
6
7
8
9
1
2

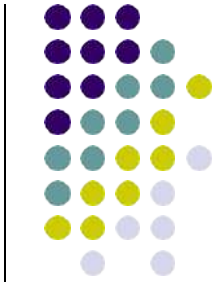
3
4
5
6
7
8
9
2 4 6
81012
141618

A program to input a matrix and print its transpose.



```
#include<stdio.h>
#include<conio.h>
void main()
{
    int a[3][3],b[3][3];
    int i,j;
    clrscr();
    printf("enter the elements in the array");
```

```
 for(i=0 ; i<3 ; i++)
    {
          for(j=0 ; j<3 ; j++)
          {
               scanf("%d",&a[i][j]);
          }
     }
for(j=0 ; i<3 ; i++)
    {
          for(i=0 ; j<3 ; j++)
          {
               printf("%2d",&b[j][i]);
          }
    }
getch();
}
```



OUTPUT:-



Enter elements in array: 1

2

3

4

5

6

7

8

9

147

258

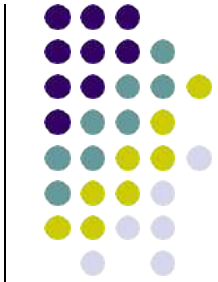
369

A program to multiply two matrix entered by the user and print it.

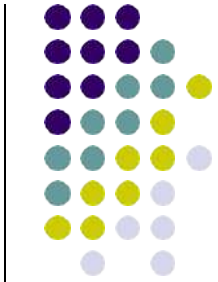


```
#include<stdio.h>
#include<conio.h>
void main()
{
    int a[3][3],b[3][3];
    int i,j;
    clrscr();
    printf("enter the elements in the array");
```

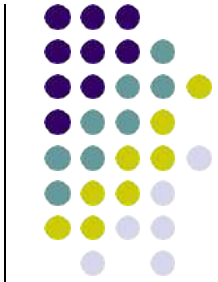
```
for(i=0 ; i<3 ; i++)
    {
        for(j=0 ; j<3 ; j++)
        {
            scanf("%d",&a[i][j]);
        }
     }
for(i=0 ; i<3 ; i++)
    {
        for(j=0 ; j<3 ; j++)
        {
            scanf("%d",&b[i][j]);
        }
    }
```



```
for(i=0 ; i<3 ; i++)
    {
          for(j=0 ; j<3 ; j++)
           c[i][j]=0;
          {
                for(k=0 ; k<2 ; k++)
               {
                     c[i][j]=c[i][j]+a[i][k]*b[k][j
                     ]  printf("%3d",c[i][j]);
                }
          }
          printf("\n");
      }
 getch();
}
```



OUTPUT:-



Enter elements in array:-

1
2
3
4
5
6
7
8
9
1
2

3
4
5
6
7
8
9
30 66 102
36 81 121
42 96 150

Multi-Dimensional Arrays



- Multidimensional arrays are derived from the basic or built-in data types of the C language.

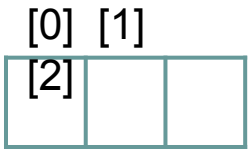
- Two-dimensional arrays are understood as rows and columns with applications including two-dimensional tables, parallel vectors, and two-dimensional matrices.

- Mostly Two-dimensional array are used in Multi-dimensional array.

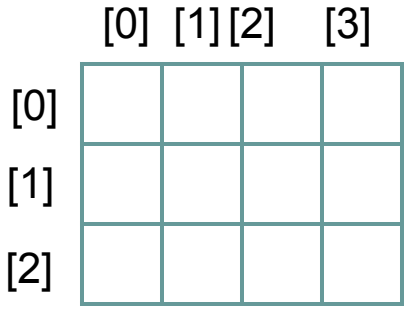
Arrays of Greater Dimension



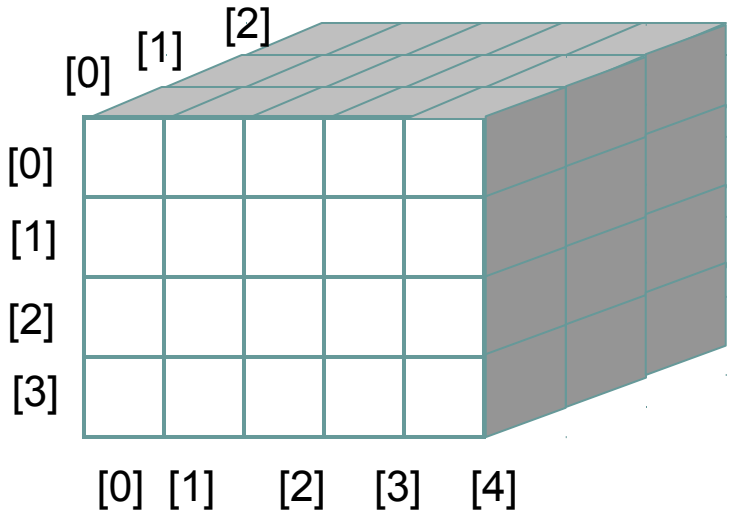
One-Dimensinal array are linear containers.



Multi-dimensional Arrays



Two-Dimensional



Three-dimensional

multi-dimensional Arrays



- A variable which represent the list of items using more than two index (subscript) is called multi-dimensional array.
- The general form of multi dimensional array is :

  type *array-name[s1][s2][s3]…….[sn];*

multi-dimensional Arrays



- Where S is the size. Some examples are :

```
int     survey[3][5][6];

float
table[5][4][5][3];
```

- Here survey is a three-dimensional array And table is a four-dimensional array.

Applications Of arrays



- Using pointers for accessing arrays.
- Passing arrays as function parameters.
- Arrays as members of structures.
- Using structure type data as array elements.
- Arrays as dynamic data structures.
- Manipulating character arrays and strings.



THANK YOU