INDEXING OFA MULTI DIMENSIONAL ARRAY

SUBMITTED BY:SHNEHILL ADHIKARY

DEPARTMENT:CSE

ROLL NO:11900121038

REGISTRATION NO:21119010011034

SUBJECT:DATA STRUCTURE AND ALGORITHMS

SUBJECT CODE:PCC-CS301

TEACHER:MRS.SUCHARITA DAS

YEAR:2ND

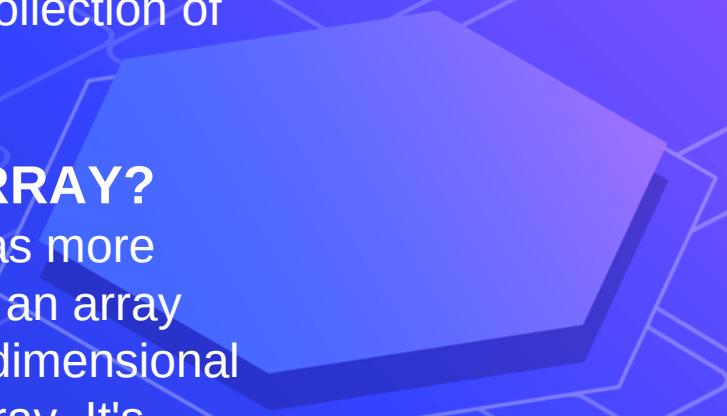
SEM:3RD

WHAT IS AN ARRAY?

Array is a kind of data structure that can store a fixed-size sequential collection of elements of the same type. 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 of the same type.

WHAT IS A 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. It's technically an array of arrays, as you will see in the code.



TWO DIMENSIONAL ARRAYS

The simplest form of multidimensional array is the two-dimensional array. A two-dimensional array is, in essence, a list of one-dimensional arrays. To declare a two-dimensional integer array of size [x][y], you would write something as follows −

Int a[ ][ ]=[x][y];

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

	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

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

```
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 */
};
```

The nested braces, which indicate the intended row, are optional. The following initialization is equivalent to the previous example −

```
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. For example:-

int val = a[2][3];

The above statement will take the 4th element from the 3rd row of the array. You can verify it in the above figure. Let us check the following program where we have used a nested loop to handle a two-dimensional array……

```
#include <stdio.h>

int main ()
{

   /* an array with 5 rows and 2 columns*/
   int a[5][2] = { {0,0}, {1,2}, {2,4}, {3,6},{4,8}};
   int i, j;

   /* output each array element's value */
   for ( i = 0; i < 5; i++ ) {

      for ( j = 0; j < 2; j++ ) {
         printf("a[%d][%d] = %d\n", i,j, a[i][j] );
      }
   }

   return 0;
}
```

When the above code is compiled and executed, it produces the following result −

```
a[0][0]: 0
a[0][1]: 0
a[1][0]: 1
a[1][1]: 2
a[2][0]: 2
a[2][1]: 4
a[3][0]: 3
a[3][1]: 6
a[4][0]: 4
a[4][1]: 8
```

As explained above, you can have arrays with any number of dimensions, although it is likely that most of the arrays you create will be of one or two dimensions.

SIMPLE MATRIX ADDITION PROGRAM USING INDEXING IN 2D ARRAY

OUTPUT

```
C:\TURBOC3\BIN>TC
Enter the number of rows and columns of matrix
2
2
Enter the elements of first matrix
1 5
2 4
Enter the elements of second matrix
1 7
4 5
Sum of entered matrices:-
2       12
6       9
Enter the number of rows and columns of matrix
```

```
#include < stdio.h >
int main()
{
int m, n, c, d, first[10][10], second[10][10],
sum[10][10];
printf("Enter the number of rows and columns of
matrix\n");
scanf("%d%d", & m, & n);
printf("Enter the elements of first matrix\n");
for (c = 0; c < m; c++)
for (d = 0; d < n; d++) scanf("%d", & first[c][d]);
printf("Enter the elements of second matrix\n");
for (c = 0; c < m; c++)
for (d = 0; d < n; d++) scanf("%d", & second[c][d]);
printf("Sum of entered matrices:-\n");
for (c = 0; c < m; c++)
{
for (d = 0; d < n; d++)
{
sum[c][d] = first[c][d] + second[c][d];
printf("%d\t", sum[c][d]);
}
printf("\n");
}
return 0;
}
```

CONCLUSION

In this presentation I was able to study about indexing in multi-dimensional arrays .It helped me to study in depth about the arrays and also how to operate them. I would like to thank our DSA teacher Miss Sucharita Das for giving me an opportunity to do this presentation. These words mark the end of my ppt.

Thankyou