**Matrix Operations**

1. **Write a Program to perform Addition of two matrices.**

#include <stdio.h>

void main()

{

int a[100][100], b[100][100], c[100][100];

int i,j,row1,col1,row2,col2;

printf("Enter the number of rows and columns 1st matrix");

scanf("%d%d",&row1,&col1);

printf("Enter the number of rows and columns 2nd matrix");

scanf("%d%d",&row2,&col2);

if((row1==row2)&&(col1==col2))

{

printf("\nEnter elements of 1st matrix:\n");

for (i = 0; i < row1; i++)

{

for (j = 0; j < col1; j++)

{

scanf("%d", &a[i][j]);

}

}

printf("Enter elements of 2nd matrix:\n");

for (i = 0; i < row2; i++)

{

for (j = 0; j < col2; j++)

{

scanf("%d", &b[i][j]);

}

}

// adding two matrices

for (i = 0; i < row1; i++)

{

for (j = 0; j < col1; j++)

{

c[i][j] = a[i][j] + b[i][j];

}

}

// printing the result

printf("\nSum of two matrices: \n");

for (i = 0; i < row1; i++)

{

for (j = 0; j < col1; j++)

{

printf("%d ", c[i][j]);

}

printf("\n");

}

}

else

{

printf(“Matrix addition is not possible”);

}

}

OUTPUT

Enter the number of rows and columns 1st matrix3

2

Enter the number of rows and columns 2nd matrix3

2

Enter elements of 1st matrix:

1 2

3 4

5 6

Enter elements of 2nd matrix:

7 8

9 10

11 12

Sum of two matrices:

8 10

12 14

16 18

1. **Write a Program to perform Multiplication of two matrices.**

#include <stdio.h>

int main()

{

int m, n, p, q, i, j, k;

int first[10][10], second[10][10], c[10][10];

printf("Enter number of rows and columns of first matrix\n");

scanf("%d%d", &m, &n);

printf("Enter elements of first matrix\n");

for (i= 0; i < m; i++)

{

for (j= 0; j < n; j++)

{

printf("Enter element a%d%d: ", i, j);

scanf("%d", &first[i][j]);

}

}

printf("Enter number of rows and columns of second matrix\n");

scanf("%d%d", &p, &q);

if (n != p)

printf("The multiplication isn't possible.\n");

else

{

printf("Enter elements of second matrix\n");

for (i = 0; i < p; i++)

{

for (j= 0; j< q; j++)

{

printf("Enter element a%d%d: ", i, j);

scanf("%d", &second[i][j]);

}

}

//Matrix Multiplication

for (i = 0; i < m; i++)

{

for (j= 0; j < q; j++)

{

c[i][j]=0;

for (k = 0; k < p; k++)

{

c[i][j]= c[i][j]+ first[i][k]\*second[k][j];

}

}

}

printf("Product of the matrices:\n");

for (i = 0; i < m; i++)

{

for (j = 0; j < q;j++)

{

printf("%d ", c[i][j]);

}

printf("\n");

}

}

}

OUTPUT:

Enter number of rows and columns of first matrix

2

3

Enter elements of first matrix

Enter element a00: 1

Enter element a01: 2

Enter element a02: 3

Enter element a10: 4

Enter element a11: 5

Enter element a12: 7 6

Enter number of rows and columns of second matrix

3

2

Enter elements of second matrix

Enter element a00: 1

Enter element a01: 2

Enter element a10: 3

Enter element a11: 4

Enter element a20: 5

Enter element a21: 6

Product of the matrices:

22 28

49 64

Multidimensional Array

C allows for arrays of two or more dimensions.

A two-dimensional (2D) array is an array of arrays.

A three-dimensional (3D) array is an array of arrays of arrays.

In C programming an array can have two, three, or even ten or more dimensions. The maximum dimensions in a C program can have,depends on which compiler is being used.

More dimensions in an array means more data be held, but also means greater difficulty in managing and understanding arrays.

## How to Declare a Multidimensional Array in C

A multidimensional array is declared using the following syntax:

**Datatype array\_name[d1][d2][d3][d4]………[dn];**

Where each **d** is a dimension, and **dn** is the size of final dimension.

Examples:

1. **int table[5][5][20];**

* **int** designates the array type integer.
* **table** is the name of our 3D array.
* Our array can hold 500 integer-type elements. This number is reached by multiplying the value of each dimension. In this case: **5x5x20=500**.

1. **float arr[5][6][5][6][5];**

* Array **arr** is a five-dimensional array.
* It can hold 4500 floating-point elements (**5x6x5x6x5=4500**).

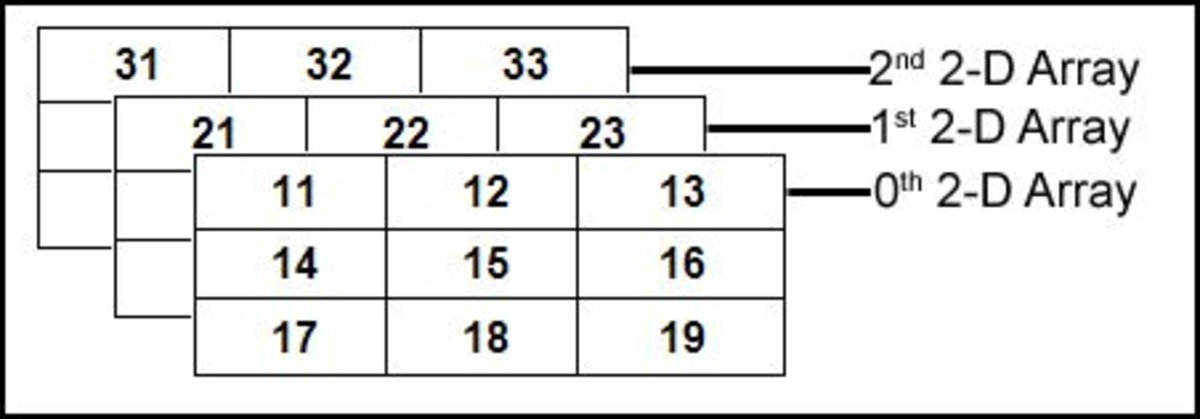
## 3D Array

A 3D array is essentially an array of arrays of arrays:

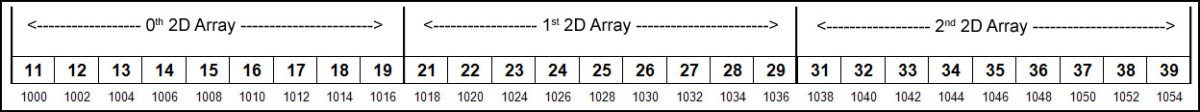
it's an array or collection of 2D arrays, and a 2D array is an array of 1D array.

As you practice working with multidimensional arrays, you start to grasp the logic.

The diagram below may help you understand:



3D Array Conceptual View



3D array memory map.

## Declaration:

**Syntax:**

**datatype array\_name[SIZE1][SIZE2][SIZE3];**

Example:

int a[3][3][3];

## Initializing a 3D Array in C

Like any other variable or array, a 3D array can be initialized at the time of **compilation.** By default, in C, an uninitialized 3D array contains “garbage” values, not valid for the intended use.

Let’s see a complete example on how to initialize a 3D array:

**Compile time Initialization:**

int arr[3][3][3]=

{

{ {11, 12, 13},{14, 15, 16},{17, 18, 19} },

{ {21, 22, 23},{24, 25, 26},{27, 28, 29} },

{ {31, 32, 33},{34, 35, 36},{37, 38, 39} }

};

Example 2:

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}}};

**Run time Initialization:**

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]);

}

}

}

**Accessing 3D Array elements:**

for (int i = 0; i < 2; ++i)

{

for (int j = 0; j < 3; ++j)

{

for (int k = 0; k < 2; ++k)

{

printf("%d”, test[i][j][k]);

}

}

}

1.Write a C Program to declare, Initialize and accessing 3D Array elements.

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

#include <stdio.h>

void 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 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]);

}

}

}

}

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