Tutorial Link https://course.testpad.chitkara.edu.in/tutorials/Two-Dimensional Arrays/59fdea15e63d6b7fd5dec030

TUTORIAL

Two-Dimensional Arrays

Topics

1.5 2-d Array Implementation

C language supports multi-dimensional arrays. To define an two-dimensional array we have to provide the size of both of dimensions in array declarations like: -

```
Type array_name[row_size][column_size];
```

i.e.

```
int age[2][10];
/* it will declare a 2 rows and 10 columns 2-D array making
total of 2x10=20 elements. */
```

Because the RAM is one-dimensional so in memory, two-dimensional arrays are stored in Row major order in C language, i.e. first row is stored first, then second row and so on. For example, if the base address of 2-D array age is 1000 then the memory map will be following: -

Address	Index	Array
1000	Age[0][0]	First row of the array age.
1004	Age[0][1]	
1008	Age[0][2]	
1012	Age[0][3]	
1016	Age[0][4]	
1020	Age[0][5]	
1024	Age[0][6]	
1028	Age[0][7]	
1032	Age[0][8]	
1036	Age[0][9]	
1040	Age[1][0]	Second row of the array age.
1044	Age[1][1]	
1048	Age[1][2]	
1052	Age[1][3]	
1056	Age[1][4]	
1060	Age[1][5]	
1064	Age[1][6]	
1068	Age[1][7]	
1072	Age[1][8]	
1076	Age[1][9]	

A 2-dimensional array can be initialized at compile time and run time in the same manner as one dimensional array.

```
int age[2][4] = {{1, 2, 3, 4}, {5, 6, 7, 8}};   // compile
time initialization.
```

OR

```
int age[2][4] = {1, 2, 3, 4, 5, 6, 7, 8};   // compile time
initialization.
```

In compile time initialization of 2-dimensional array we can omit the first dimension size, but have to provide the second dimension size to be interpreted as row breaks by the compiler i.e. the above array can also be written as: -

```
int age[][4] = {{1, 2, 3, 4}, {5, 6, 7, 8}};   // compile
time initialization.
```

OR

```
int age[][4] = {1, 2, 3, 4, 5, 6, 7, 8};   // compile time
initialization.
```

But can not be written as: -

```
int age[][] = {1, 2, 3, 4, 5, 6, 7, 8};
/* compile time error as there are multiple ways to
interpret this array like 2x4, 1x8, 4x2 all resulting in 8
elements, so later at time of accessing the ambiguity may
arise.*/
```

OR

```
int age[2][] = {{1, 2, 3, 4}, {5, 6, 7, 8}};
// Compile time error even the first dimension is provided
but C requires complete information in this case.
```

Similar to 1-dimensional arrays, multi-dimensional array can be initialized at run time also. The following program will create and use a 2-dimensional array at run time: -

```
#include <stdio.h>

int main()
{
```

```
  int age[2][5], i, j;
  printf("Enter the element of age array of size 2 by 5 : ");
  for(i=0; i<2; i++)
  {
     for(j=0; j < 5; j++)
    {
     scanf("%d", &age[i][j]);
    }
  }
  for(i=0; i < 2; i++)
  {
     for(j=0; j < 5; j++)
    {
        printf("%d\t", age[i][j]);
    }
     printf("\n");
  }
  return 0;
}
```

The above program will take 10 values from user and store them in array and later print them as a 2x5 matrix.

2-d Array Implementation

The following program will copy the elements of a 2-dimensional array in another array and prints it.

```c
#include <stdio.h>
/* Include other headers as needed */
int main()
{
    /* Enter your code here. Read input from STDIN. Print output to STDOUT */
        int array1[2][5]={0,1,2,3,4,5,6,7,8,9};
    int array2[2][5], i, j;
    for(i=0; i<2; i++)
    {
        for(j=0; j < 5; j++)
        {
            array2[i][j] = array1[i][j];
        }
```

```
    }
    for(i=0; i < 2; i++)
    {
        for(j=0; j < 5; j++)
        {
            printf("%d\t", array2[i][j]);
        }
        printf("\n");
    }
  return 0;
}
```

The output of the above program will be

```
0   1   2   3   4
5   6   7   8   9
```

The following program will print the sum of two 2-dimensional arrays: -

```c
#include <stdio.h>
/* Include other headers as needed */
int main()
{
    /* Enter your code here. Read input from STDIN. Print output to STDOUT */
    int array1[2][5]={0,1,2,3,4,5,6,7,8,9};
    int array2[][5]={0,1,2,3,4,5,6,7,8,9};
    int i, j;
    for(i=0; i < 2; i++)
    {
        for(j=0; j < 5; j++)
        {
            printf("%d\t", array1[i][j] + array2[i][j]);
        }
        printf("\n");
    }

```

```
18    return 0;
19 }
```

The output of the above program will be

```
0   2   4   6   8
10   12   14   16   18
```