Three dimensional (3D) array in C

Software Engineering | C | Interview Problems on Array

A 3D array is a **multi-dimensional array**(array of arrays). A 3D array is a collection of **2D arrays**. It is specified by using three subscripts:Block size, row size and column size.

More dimensions in an array means more data can be stored in that array.

Visualizing 3D array:

If we want to visualize a **2D array**, we can visualize it in this way:
int arr[3][3], it means a 2D array of type integer having 3 rows and 3 columns.It is just a simple matrix

```
int arr[3][3];        //2D array contaning 3 rows and 3 columns

1 2 3
4 5 6
7 8 9
     3x3
```

But, what happen if we add one more dimension here,
i.e, **int arr[3][3][3]**, now it becomes a 3D array.

- **int** shows that the 3D array is an array of **type integer**.
- **arr** is the **name of array**.
- **first dimension** represents the **block size**(total number of 2D arrays).
- **second dimension** represents the **rows of 2D arrays**.
- **third dimension** represents the **columns of 2D arrays**.
  i.e; int arr[3][3][3], so the statement says that we want three such 2D arrays which consists of 3 rows and 3 columns.

```
int arr[3][3][3];        //3D array

block(1)  1 2 3          block(2)  10 11 12          block(3)  19 20 21
          4 5 6                    13 14 15                    22 23 24
          7 8 9                    16 17 18                    25 25 27
               3x3                         3x3
```

Declaring a 3D array:

To declare 3D array:

- Specify data type, array name, block size, row size and column size.
- Each subscript can be written within its own separate pair of brackets.

```
int arr[2][3][3];       //array of type integer
                        //number of blocks of 2D arrays:2 |rows:3 |colum
                        //number of elements:2*3*3=18

block(1)  11 22 33          block(2)  12 13 14
          44 55 66                    21 31 41
          77 88 99                    12 13 14
                  3x3                         3x3
```

Ways to declare 3D array:

1). int arr[2][3][3];

In this type of declaration, we have an array of type integer, block size is 2, row size is 3 and column size is 3.Here we have not stored any values/elements in the array.So the array will hold the garbage values.

```
int arr[2][3][3];    //no elements are stored

block(1)  1221 -543 3421          block(2)  654 5467 -878     //all val
          3342 6543 4221                    456 1567 7890     //garbage
          -564 4566 -345                    567 6561 2433
                      3x3                             3x3
```

2). int arr[2][3][3]={};

In this type of declaration, we have an array of type integer, block size is 2, row size is 3 and column size is 3 and we have put curly braces after assignment operator.So the array will hold 0 in each cells of array.

```
int arr[2][3][3]={};   //0 will be stored

block(1)  0 0 0        block(2)  0 0 0
          0 0 0                  0 0 0
          0 0 0                  0 0 0
               3x3                    3x3
```

3). int arr[3][2][2]={0,1,2,3,4,5,6,7,8,9,3,2}

In this type of declaration, we have an array of type integer, block size is 3, row size is 2, column size is 2 and we have mentioned the values inside the curly braces

during the declaration of array. So all the values will be stored one by one in the arra
cells.

```
int arr[3][2][2]={0,1,2,3,4,5,6,7,8,9,3,2}

block(1)  0 1        block(2)  4 5        block(3)  8 9
          2 3                  6 7                  3 2
             2x2                  2x2                  2x2
```

4). int arr[3][3][3]=
{ {{10,20,30},{40,50,60},{70,80,90}},
{{11,22,33},{44,55,66},{77,88,99}},
{{12,23,34},{45,56,67},{78,89,90}}
};

In this type of declaration, we have an array of type integer, block size is 3, row size is 3, column size is 3 and the values of each blocks are assigned during its declaration.

```
int arr[3][3][3]=
        { {{10,20,30},{40,50,60},{70,80,90}},    //elements of block 1
          {{11,22,33},{44,55,66},{77,88,99}},    //elements of block 2
          {{12,23,34},{45,56,67},{78,89,90}}     //elements of block 3
        };

block(1)  10 20 30        block(2)  11 22 33        block(3)  12 23 34
          40 50 60                  44 55 66                  45 56 67
          70 80 90                  77 88 99                  78 89 90
                 3x3                       3x3                       3
```

Inserting values in 3D array:

In 3D array, if a user want to enter the values then **three for loops** are used.

- First for loop represents the blocks of a 3D array.
- Second for loop represents the number of rows.
- Third for loop represents the number of columns.
  Example:

Following is the implementation in C:

```
#include<stdio.h>
int i,j,k;                          //variables for nested for loops
int main()
```

```
    printf("enter the values in the array: \n");
    for(i=1;i<=2;i++)              //represents block
    {
        for(j=1;j<=3;j++)          //represents rows
        {
            for(k=1;k<=3;k++)      //represents columns
            {
                printf("the value at arr[%d][%d][%d]: ",i,j,k);
                scanf("%d",&arr[i][j][k]);
            }
        }
    }
    printf("printing the values in array: \n");
    for(i=1;i<=2;i++)
    {
        for(j=1;j<=3;j++)
        {
            for(k=1;k<=3;k++)
            {
                printf("%d ",arr[i][j][k]);
                if(k==3)
                {
                    printf("\n");
                }
            }
        }
        printf("\n");
    }
    return 0;
}
```

OUTPUT:

```
enter the values in the array:

the value at arr[1][1][1]: 23
the value at arr[1][1][2]: 34
the value at arr[1][1][3]: 45
the value at arr[1][2][1]: 76
the value at arr[1][2][2]: 78
the value at arr[1][2][3]: 98
the value at arr[1][3][1]: 87
the value at arr[1][3][2]: 67
the value at arr[1][3][3]: 98
the value at arr[2][1][1]: 34
the value at arr[2][1][2]: 23
```

```
the value at arr[2][2][2]: 19
the value at arr[2][2][3]: 84
the value at arr[2][3][1]: 39
the value at arr[2][3][2]: 82
the value at arr[2][3][3]: 44

printing the values in array:
23 34 45
76 78 98
87 67 98

34 23 67
58 19 84
39 82 44
```

In the above program;

- We have declared three variables *i,j,k* for three *for loops*.
- we have declared an array of type integer *int arr[2][3][3];*(blocks:2 rows:3 columns:3)
- First section of nested for loops ask the user to insert the values.
- second section of nested for loops will print the inserted values in the matrix form.

Updating the 3D array:

We can update the elements of 3D array either by specifying the element to be replaced or by specifying the position where replacment has to be done.
For updating the array we require,

- Elements of an array
- Element or position, where it has to be inserted
- The value to be inserted

Example

Following is the implementation in C:

```
#include<stdio.h>
int i,j,k;                          //variables for nested for loop
int num;                            //will hold the value to be replace
int main()
{
```

```
    for(i=1;i<=2;i++)           //represents block
    {
        for(j=1;j<=3;j++)       //represents rows
        {
            for(k=1;k<=3;k++)   //represents columns
            {
                printf("the value at arr[%d][%d][%d]: ",i,j,k);
                scanf("%d",&arr[i][j][k]);
            }
        }
    }
    printf("\nprinting the values in array: \n");
    for(i=1;i<=2;i++)
    {
        for(j=1;j<=3;j++)
        {
            for(k=1;k<=3;k++)
            {
                printf("%d ",arr[i][j][k]);
                if(k==3)
                {
                    printf("\n");
                }
            }
        }
        printf("\n");
    }
    printf("\nenter the block row and column number: \n");
    scanf("%d %d %d ",&i,&j,&k);         //position where we want to
    printf("enter the new number you want to update with: ");
    scanf("%d",&num);                   //element to be replaced
    arr[i][j][k]=num;                   //element assigned to array po
    printf("\narray after updating: \n");
    for(i=1;i<=2;i++)
    {
        for(j=1;j<=3;j++)
        {
            for(k=1;k<=3;k++)
            {
                printf("%d ",arr[i][j][k]);
                if(k==3)
                {
                    printf("\n");
                }
            }
        }
        printf("\n");
```

```
}
```

OUTPUT:

```
enter the values in the array:

the value at arr[1][1][1]: 11
the value at arr[1][1][2]: 22
the value at arr[1][1][3]: 33
the value at arr[1][2][1]: 44
the value at arr[1][2][2]: 55
the value at arr[1][2][3]: 66
the value at arr[1][3][1]: 77
the value at arr[1][3][2]: 88
the value at arr[1][3][3]: 99
the value at arr[2][1][1]: 10
the value at arr[2][1][2]: 20
the value at arr[2][1][3]: 30
the value at arr[2][2][1]: 40
the value at arr[2][2][2]: 50
the value at arr[2][2][3]: 60
the value at arr[2][3][1]: 70
the value at arr[2][3][2]: 80
the value at arr[2][3][3]: 90

printing the values in array:
11 22 33
44 55 66
77 88 99

10 20 30
40 50 60
70 80 90


enter the block row and column number:2 1 1
enter the new number you want to update with: 15

array after updating:
11 22 33
44 55 66
77 88 99

15 20 30
```

In the above program;

- We have declared three variables *i,j,k* for three *for loops* and *num* variable which will hold the value/element to be updated.
- we have declared an array of type integer *int arr[2][3][3];*(blocks:2 rows:3 columns:3)
- First section of nested for loops ask the user to insert the values.
- second section of nested for loops will print the inserted values in the matrix form.
- Position/value will be updated.
- Third section of nested for loop will print the updated 3D array.

Converting 3D array into 2D array

We can convert a 3D array into a 2D array. As we know that a 3D array is a colection of 2D arrays, we just have to follow certain steps to convrt a 3D array into 2D array;

- First declare a 3D array and enter the elements in it.
- After that, declare the 2D arrays(number of 2D arrays should be equal to the total number of blocks of 3D array).
- Copy the elements of 3D array into 2D array.
  Example:

```
#include<stdio.h>
int main()
{
    int i,j,k;              //variables for nested for loop
    int arr[2][3][3];       //declaration of 3D array
    printf("enter the elements: \n");
    for(i=0;i<2;i++)
    {
        for(j=0;j<3;j++)
        {
            for(k=0;k<3;k++)
            {
                printf("element at [%d][%d][%d]: ",i,j,k);
                scanf("%d",&arr[i][j][k]);
                //arr[i][j][k]=++ctr;
            }
```

```
    printf("\nprinting 3D array\n");
    for(i=0;i<2;i++)                        //printing 3d array
    {
        for(j=0;j<3;j++)
        {
            for(k=0;k<3;k++)
            {
                printf("%d ",arr[i][j][k]);
            }
            printf("\n");
        }
        printf("\n");
    }
    int a[3][3];                    //2D array declaration
    int b[3][3];
    printf("\ncopying values in new 2D array: \n");
    for(i=0;i<2;i++)
    {
        for(j=0;j<3;j++)
        {
            for(k=0;k<3;k++)
            {
                if(i==0)
                {
                    a[j][k]=arr[i][j][k];    //copying values in new 2d
                }
                else
                {
                    b[j][k]=arr[i][j][k];
                }
            }
        }
    }
    printf("\nprinting elements in first 2D array: \n");
    for(i=0;i<3;i++)
    {
        for(j=0;j<3;j++)                //printing first 2d array
        {
            printf("%d ",a[i][j]);
        }
        printf("\n");
    }
    printf("\n");
    printf("\nprinting elements in second 2D array: ");
    for(i=0;i<3;i++)
    {
        for(j=0;j<3;j++)                //printing 2nd 2d array
```

```
        }
        printf("\n");
    }
    return 0;
}
```

OUTPUT:

```
enter the elements:
element at [0][0][0]: 10
element at [0][0][1]: 20
element at [0][0][2]: 30
element at [0][1][0]: 40
element at [0][1][1]: 50
element at [0][1][2]: 60
element at [0][2][0]: 70
element at [0][2][1]: 80
element at [0][2][2]: 90
element at [1][0][0]: 11
element at [1][0][1]: 22
element at [1][0][2]: 33
element at [1][1][0]: 44
element at [1][1][1]: 55
element at [1][1][2]: 66
element at [1][2][0]: 77
element at [1][2][1]: 88
element at [1][2][2]: 99

printing 3D array:
10 20 30
40 50 60
70 80 90

11 22 33
44 55 66
77 88 99


copying values in 2D array:
printing elements in first 2D array:
10 20 30
40 50 60
70 80 90

printing elements in second 2D array:
11 22 33
```

Converting 2D array into 3D array

We can also convert a 2D array into a 3D array by following given steps;

- Declare a 2D array and enter the elements in it and can print it.
- Now declare a 3D array and copy the elements of 2D array into 3D array and print them.

Example

```
#include<stdio.h>
int main()
{
    int i,j,k;
    int a[3][3];                    //2D array declaration
    int b[3][3];
    printf("enter the elements in array a: \n");        //entering elem
    for(i=0;i<3;i++)
    {
        for(j=0;j<3;j++)
        {
            printf("element at [%d][%d]: ",i,j);
            scanf("%d",&a[i][j]);
        }
    }
    printf("\nprinting 2D array a\n");                //printing eleme
    for(i=0;i<3;i++)
    {
        for(j=0;j<3;j++)
        {
            printf("%d ",a[i][j]);
        }
        printf("\n");
    }
    printf("\nenter the elements in array b: \n");     //entering eleme
    for(i=0;i<3;i++)
    {
        for(j=0;j<3;j++)
        {
            printf("element at [%d][%d]: ",i,j);
            scanf("%d",&b[i][j]);
        }
```

```
    for(i=0;i<3;i++)
    {
        for(j=0;j<3;j++)
        {
            printf("%d ",b[i][j]);
        }
        printf("\n");
    }
    int arr[3][3][3];                                   //3D array decla
    printf("\ncopying values in 3D array: \n");
    for(i=0;i<2;i++)
    {
        for(j=0;j<3;j++)
        {
            for(k=0;k<3;k++)
            {
                if(i==0)
                {
                    arr[i][j][k]=a[j][k];               //copying elemer
                }
                else
                {
                    arr[i][j][k]=b[j][k];
                }
            }
        }
    }
    printf("\nprinting elements in 3D array: \n");    //printing elemer
    for(i=0;i<2;i++)
    {
        for(j=0;j<3;j++)
        {
            for(k=0;k<3;k++)
            {
            printf("%d ",arr[i][j][k]);
            if(k==2)
            {
            printf("\n");
            }
            }
        }
        printf("\n");
    }
    return 0;
}
```

```
enter the elements in array a:
element at [0][0]: 10
element at [0][1]: 20
element at [0][2]: 30
element at [1][0]: 40
element at [1][1]: 50
element at [1][2]: 60
element at [2][0]: 70
element at [2][1]: 80
element at [2][2]: 90

printing 2D array a
10 20 30
40 50 60
70 80 90

enter the elements in array b:
element at [0][0]: 11
element at [0][1]: 22
element at [0][2]: 33
element at [1][0]: 44
element at [1][1]: 55
element at [1][2]: 66
element at [2][0]: 77
element at [2][1]: 88
element at [2][2]: 99

printing 2D array b
11 22 33
44 55 66
77 88 99

copying values in 3D array:

printing elements in 3D array:
10 20 30
40 50 60
70 80 90

11 22 33
44 55 66
77 88 99
```

Dynamically allocating memory

shrinked).To remove this drawback we use dynamic memory allocation.dynamic array is nothing but it is allocated during runtime with malloc or calloc.

*syntax: int *array=int(int *)malloc(sizeof(int)*element-count);*

Example

```
#include <stdio.h>
#include <malloc.h>              //malloc library
int main(int argc, char* argv[]) //command line arguments
{
  int ***arr;                    //triple pointer
  int block,row,column;          //variables for block, rows and columr
  int i,j,k;                     //nested for loop
  printf("enter the blocks, rows and columns: ");
  scanf("%d %d %d",&block,&row,&column);
  arr=(int ***)malloc(sizeof(int ***)*block);
  for(i=0;i<block;i++)
  {
    arr[i]=(int **)malloc(sizeof(int*)*row);
    for(j=0;j<row;j++)
    {
      arr[i][j]=(int *)malloc(sizeof(int)*column);
    }
  }
  for(i=0;i<block;i++)
  {
    for(j=0;j<row;j++)
    {
      for(k=0;k<column;k++)
      {
        printf("element at [%d][%d][%d] : ",i,j,k);
        scanf("%d",&arr[i][j][k]);
      }
    }
  }
  printf("Printing 3D Array:\n");
  for(i=0;i<block;i++)
  {
    for(j=0;j<row;j++)
    {
      for(k=0;k<column;k++)
      {
        printf("%.2d ",arr[i][j][k]);
      }
      printf("\n");
    }
    printf("\n");
```

```
}
```

OUTPUT

```
enter the blocks, rows and columns: 2 3 3
element at [0][0][0] : 10
element at [0][0][1] : 20
element at [0][0][2] : 30
element at [0][1][0] : 40
element at [0][1][1] : 50
element at [0][1][2] : 60
element at [0][2][0] : 70
element at [0][2][1] : 80
element at [0][2][2] : 90
element at [1][0][0] : 11
element at [1][0][1] : 22
element at [1][0][2] : 33
element at [1][1][0] : 44
element at [1][1][1] : 55
element at [1][1][2] : 66
element at [1][2][0] : 77
element at [1][2][1] : 88
element at [1][2][2] : 99
Printing 3D Array:
10 20 30
40 50 60
70 80 90

11 22 33
44 55 66
77 88 99
```

With this article at OpenGenus, you will have complete idea of handling 3D arrays in C. Enjoy.

Learn more:

- 2D arrays in C by Subhash Bhandari at OpenGenus
- List of C topics at OpenGenus

Subhash Bhandari
Intern at OpenGenus

Read More

Improved & Reviewed by: **OpenGenus Foundation**

SOFTWARE ENGINEERING

Using complex.h header file in C

In this article, we have explored the complex.h header file in C. The header file complex.h defines macros and functions to carry out operations on complex numbers in C.

SOFTWARE ENGINEERING

Two dimensional (2D) array in C

In this article, we have explored 2D arrays in C including declaring 2D array, calculating size, inserting elements and updating 2D array, convert 3D array to 2D array and using malloc to define 2D arrays in C.

SUBHASH BHANDARI