

How to dynamically allocate a 2D array in C?

<http://www.geeksforgeeks.org/dynamically-allocate-2d-array-c>

Following are different ways to create a 2D array on heap (or dynamically allocate a 2D array).

In the following examples, we have considered 'r' as number of rows, 'c' as number of columns and we created a 2D array with r = 3, c = 4 and following values

```
1  2  3  4
5  6  7  8
9 10 11 12
```

1) Using a single pointer:

A simple way is to allocate memory block of size r*c and access elements using simple pointer arithmetic.

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int r = 3, c = 4;
    int *arr = (int *)malloc(r * c * sizeof(int));

    int i, j, count = 0;
    for (i = 0; i < r; i++)
        for (j = 0; j < c; j++)
            *(arr + i*c + j) = ++count;

    for (i = 0; i < r; i++)
        for (j = 0; j < c; j++)
            printf("%d ", *(arr + i*c + j));

    /* Code for further processing and free the
       dynamically allocated memory */

    return 0;
}
```

Output:

```
1 2 3 4 5 6 7 8 9 10 11 12
```

2) Using an array of pointers

We can create an array of pointers of size r. Note that from C99, C language allows variable sized arrays. After creating an array of pointers, we can dynamically allocate memory for every row.

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int r = 3, c = 4, i, j, count;

    int *arr[r];
    for (i=0; i<r; i++)
        arr[i] = (int *)malloc(c * sizeof(int));

    // Note that arr[i][j] is same as (*(arr+i)+j)
    count = 0;
    for (i = 0; i < r; i++)
        for (j = 0; j < c; j++)
            arr[i][j] = ++count; // Or (*(arr+i)+j) = ++count

    for (i = 0; i < r; i++)
        for (j = 0; j < c; j++)
            printf("%d ", arr[i][j]);

    /* Code for further processing and free the
       dynamically allocated memory */

    return 0;
}
```

Output:

```
1 2 3 4 5 6 7 8 9 10 11 12
```

3) Using pointer to a pointer

We can create an array of pointers also dynamically using a double pointer. Once we have an array pointers allocated dynamically, we can dynamically allocate memory and for every row like method 2.

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int r = 3, c = 4, i, j, count;

    int **arr = (int **)malloc(r * sizeof(int *));
    for (i=0; i<r; i++)
        arr[i] = (int *)malloc(c * sizeof(int));

    // Note that arr[i][j] is same as *(*arr+i)+j)
    count = 0;
    for (i = 0; i < r; i++)
        for (j = 0; j < c; j++)
            arr[i][j] = ++count;    // OR *(*arr+i)+j = ++count

    for (i = 0; i < r; i++)
        for (j = 0; j < c; j++)
            printf("%d ", arr[i][j]);

    /* Code for further processing and free the
       dynamically allocated memory */

    return 0;
}
```

Output:

```
1 2 3 4 5 6 7 8 9 10 11 12
```

4) Using double pointer and one malloc call for all rows

```
#include<stdio.h>
#include<stdlib.h>

int main()
{
    int r=3, c=4;
    int **arr;
    int count = 0,i,j;

    arr = (int **)malloc(sizeof(int *) * r);
    arr[0] = (int *)malloc(sizeof(int) * c * r);

    for(i = 0; i < r; i++)
        arr[i] = (*arr + c * i);

    for (i = 0; i < r; i++)
        for (j = 0; j < c; j++)
            arr[i][j] = ++count;    // OR *(*arr+i)+j = ++count

    for (i = 0; i < r; i++)
        for (j = 0; j < c; j++)
            printf("%d ", arr[i][j]);

    return 0;
}
```

Output:

1 2 3 4 5 6 7 8 9 10 11 12