How to dynamically allocate a 2D array in 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

```
#include<stdio.h>
#include<stdlib.h>
int main()
```

```
{
    int r=3, c=4, len=0;
    int *ptr, **arr;
    int count = 0,i,j;
     len = sizeof(int *) * r + sizeof(int) * c * r;
    arr = (int **)malloc(len);
      // ptr is now pointing to the first element in of 2D array
    ptr = (int *)(arr + r);
      // for loop to point rows pointer to appropriate location in 2D array
    for(i = 0; i < r; i++)
        arr[i] = (ptr + 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
```

Matrix multiplication (dynamic memory allocation) where that the user can enter any valid order for matrix multiplication (i.e. column1=row2).

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


main(){
int **mat1, **mat2,**res,i,j,r1,c1,r2,c2;

printf("\nEnter the Order of the First matrix...\n");
scanf("%d %d",&r1,&c1);
printf("\nEnter the Order of the Second matrix...\n");
scanf("%d %d",&r2,&c2);

if(c1!=r2){
   printf("Invalid Order of matrix");
   exit(EXIT_SUCCESS);
}

mat1= (int**) malloc(r1*sizeof(int*));

for(i=0;i<c1;i++)
   mat1[i]=(int*)malloc(c1*sizeof(int));

mat2= (int**) malloc(r2*sizeof(int*));

for(i=0;i<c2;i++)
   mat2[i]=(int*)malloc(c2*sizeof(int));

res=(int**)calloc(r1,sizeof(int*));

for(i=0;i<c2;i++)
   res[i]=(int*)calloc(c2,sizeof(int));

//Input Matrix1
   for(i=0;i<r1;i++)
      for(j=0;j<c1;j++)
         scanf("%d",&mat1[i][j]);
//Input Matrix2
   for(i=0;i<r2;i++)
      for(j=0;j<c2;j++)
         scanf("%d",&mat2[i][j]);

//Printing Input Matrix 1 and 2

printf("\n Entered Matrix 1: \n");
for(i=0;i<r1;i++){
   for(j=0;j<c1;j++)
      printf("%d ",mat1[i][j]);
   printf("\n");
}
```

```
printf("\n Entered Matrix 2: \n");
for(i=0;i<r2;i++){
    for(j=0;j<c2;j++)
        printf("%d ",mat2[i][j]);
    printf("\n");
}

//Computation


//Multiplication

    for(i=0;i<r1;i++){
        for(j=0;j<c2;j++){
                res[i][j]=0;
                for(k=0;k<c1;k++)
                    res[i][j]+= mat1[i][k]*mat2[k][j];

        }
        printf("\n");
    }


   printf("\nThe Multiplication of two matrix is\n");
   for(i=0;i<r1;i++){
       printf("\n");
       for(j=0;j<c2;j++)
           printf("%d\t",res[i][j]);
   }
    printf("\n");

/*  Addition
for(i=0;i<r1;i++)
        for(j=0;j<c2;j++)
                res[i][j]=mat1[i][j]+mat2[i][j];


printf("\nThe Addition of two matrix is\n");
   for(i=0;i<r1;i++){
       printf("\n");
       for(j=0;j<c2;j++)
           printf("%d\t",res[i][j]);
    }
*/


return 0;}
```