20180328*指针创建数组*一维数组，二维数组，三维数组

```
1- 用这种就够了，其它的都比较复杂
#include<stdlib.h>
//申请一个3*4*5的整型数组
    int (*a)[4][5] = (int(*)[4][5])malloc(sizeof(int)*3*4*5);
```

1- 一维数组的创建

```
int * arr = new int[10]();  // 创建大小为10 的一维数组

delete [] arr;
```

```
 int *v = new int[n]();

 for (int i = 0; i < n; i++)
        cin >> v[i];
```

2- 二维数组的创建

```
int ** arr;
arr = new *int[10](); // 前面加一个 *, 就是二维指针了。 10 行

for(int j=0 ; j < m; j ++)
    arr[0] = new int [m](); // m 列

delete [] arr;
```

```
long long **dp;
dp = new long long *[n+1]();
for (int k = 0; k <= n; ++k) {
    dp[k] = new long long [w+1]();
}
```

3- 三维数组的创建

创建一个数组`arr[2][3][4]`

这种就比较难创建了。下面的创建的有点问题。

```
int *** arr;
arr = new **int[2];

for( int i=0; i<2;i++ )
    arr[i] = new int(*)[3]();

for(int i=0; i<2; i++)
    for( int j = 0; j < 3;j++ )
        arr[i][j] = new int[4];
```

另外几种动态创建数组的方式。

4- 用数组指针形式申请三维数组

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

int main()
{
    int i,j,k;
    int value = 1;
    //申请一个3*4*5的整型数组
    int (*a)[4][5] = (int(*)[4][5])malloc(sizeof(int)*3*4*5);

    // 遍历三维数组 打印元素地址
    for(i = 0; i < 3; i++)
        for(j = 0; j < 4; j++)
            for(k = 0; k < 5; k++)
                printf("&a[%d][%d][%d] = %p\n",i,j,k,&a[i][j][k]);
```

```
    //输出数组每个元素地址，每个元素的地址是连续的

    // 遍历三维数组赋值
    for (i = 0; i < 3; i++)
        for (j = 0; j < 4; j++)
            for (k = 0; k < 5; k++)
                *(*(*(a + i) + j) + k) = value++;

    // 遍历三维数组 打印元素值
    for (i = 0; i < 3; i++)
        for (j = 0; j < 4; j++)
            for (k = 0; k < 5; k++)
                printf("values[%d][%d][%d] = %d\n", i, j, k, a[i][j][k]);

    // 堆空间回收
    free(a);
    return 0;
}
int main()
{
    int i,j,k;
    int value = 1;
    //申请一个3*4*5的整型数组
    int (*a)[4][5] = (int(*)[4][5])malloc(sizeof(int)*3*4*5);

    // 遍历三维数组 打印元素地址
    for(i = 0; i < 3; i++)
        for(j = 0; j < 4; j++)
            for(k = 0; k < 5; k++)
                printf("&a[%d][%d][%d] = %p\n",i,j,k,&a[i][j][k]);
    //输出数组每个元素地址，每个元素的地址是连续的

    // 遍历三维数组赋值
    for (i = 0; i < 3; i++)
        for (j = 0; j < 4; j++)
            for (k = 0; k < 5; k++)
                *(*(*(a + i) + j) + k) = value++;

    // 遍历三维数组 打印元素值
    for (i = 0; i < 3; i++)
        for (j = 0; j < 4; j++)
            for (k = 0; k < 5; k++)
                printf("values[%d][%d][%d] = %d\n", i, j, k, a[i][j][k]);

    // 堆空间回收
    free(a);
    return 0;
}
```

5- 利用三级指针申请一个三维数组。思路是先申请后分配

这种需要`malloc.h`头文件

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

int*** CreateGrid(int m,int n,int t)
{
    int i = 0;
    int k = 0;
    int*** result = NULL;
    if((m > 0) && (n > 0) && (t > 0))
    {
        int** pp = NULL;
        int* p = NULL;
        result = (int***)malloc(m * sizeof(int**));     // key
        pp = (int**)malloc(m * n * sizeof(int*));       // key
        p = (int*)malloc(m * n * t * sizeof(int));      // key
        if((result != NULL) && (pp != NULL) && (p != NULL))
        {
            for(i = 0;i < m;i++)
            {
                result[i] = pp + i * n; // 三维元素存二维地址
                for (k = 0;k < n;k++)
                {
                    result[i][k] = p + k * t; // // 二维元素存一维地址
                }
                p = p + n*t;
            }
        }
        else
        {
            free(result);
            free(pp);
            free(p);
            result = NULL;
            pp = NULL;
            p = NULL;
        }
    }
    return result;
}
void FreeGrid(int*** p)
{
```

```
    if(*p != NULL)
    {
        if(**p != NULL)
        {
            free(**p);
            **p = NULL;
        }
        free(*p);
        *p = NULL;
    }
    free(p);
    p = NULL;
}

int main(void)
{
    int*** a = CreateGrid(3, 3, 3);
    int i = 0;
    int j = 0;
    int k = 0;
    int value = 1;

// 遍历三维数组赋值
    for (i = 0; i < 3; i++)
        for (j = 0; j < 3; j++)
            for (k = 0; k < 3; k++)
                *(*(*(a + i) + j) + k) = value++;

// 遍历三维数组 打印元素地址
    for (i = 0; i < 3; i++)
        for (j = 0; j < 3; j++)
            for (k = 0; k < 3; k++)
                printf("values[%d][%d][%d] = %p\n", i, j, k, &a[i][j][k]);
// 地址是连续的

// 遍历三维数组 打印元素值
    for (i = 0; i < 3; i++)
        for (j = 0; j < 3; j++)
            for (k = 0; k < 3; k++)
                printf("values[%d][%d][%d] = %d\n", i, j, k, a[i][j][k]);

// 堆空间回收
    if(a != NULL)
        FreeGrid(a);
    return 0;
}
```

6- 三级指针申请一个三维数组

方法三：利用三级指针申请一个三维数组。思路是一边申请一边分配。

看到很多都是使用这种方法，严格来讲这种方法是不准确的。因为可以看到申请出来的数组元素的地址不是连续的。我们都知道数组是一片连续的内存空间。所以这个并不叫一个数组。

```
#include <stdio.h>
#include <stdlib.h>
int*** CreateGrid(int m,int n,int t)
{
    int i = 0;
    int k = 0;
    int*** tt = NULL;
    if((m > 0) && (n > 0) && (t > 0))
    {
        /// 一边申请，一边分配
        tt = (int***)malloc(sizeof(int)*m);
        for(i = 0;i < m;i++)
        {
            tt[i] = (int**)malloc(sizeof(int)*n);
            for (k = 0;k < n;k++)
            {
                tt[i][k] = (int*)malloc(sizeof(int)*t);
            }
        }
    }
    return tt;
}

void FreeGrid(int*** tt,int m,int n,int t)
{
    int i = 0;
    int j = 0;
    if(tt != NULL)
    {
        for(i = 0;i < m;i++)
        {
            for (j = 0;j < n;j++)
            {
                free((tt[i][j]));
            }
            free(tt[i]);
        }
        free(tt);
        tt = NULL;
    }
}
```

```c
int main(void)
{
   int*** a = CreateGrid(3, 3, 3);
   int i = 0;
   int j = 0;
   int k = 0;
   int value = 0;
// 遍历三维数组赋值
    for (i = 0; i < 3; i++)
        for (j = 0; j < 3; j++)
            for (k = 0; k < 3; k++)
                *(*(*(a + i) + j) + k) = value++;
// 遍历三维数组 打印元素地址
    for (i = 0; i < 3; i++)
        for (j = 0; j < 3; j++)
            for (k = 0; k < 3; k++)
                printf("values[%d][%d][%d] = %p\n", i, j, k, &a[i][j][k]);
 // 地址不是连续的

// 遍历三维数组 打印输出
    for (i = 0; i < 3; i++)
        for (j = 0; j < 3; j++)
            for (k = 0; k < 3; k++)
                printf("values[%d][%d][%d] = %d\n", i, j, k, a[i][j][k]);

// 堆空间回收
    if(a != NULL)
        FreeGrid(a,3,3,3);
    return 0;
```