64 Multidimensional Arrays in C++ (2D arrays)

今天讲的是多维数组，请务必了解31 Arrays in C++和16 POINTERS in C++，因为处理任何类型的数组时，指针都是非常重要的（因为数组就是内存块，处理内存的简单方法就是使用指针）

1. 多维数组与常规数组的不同之处

从二维数组开始作为一个例子，实际上它只是**数组的数组**（三维数组就是数组的数组的数组.....），就是数组的集合。
我们考虑处理数组的一种策略就是使用指针，我们有一个指针，指向数组在内存中的开头位置。可以想象一下有一个指针的数组，最终你会得到一个内存块，里面包含的是连续的指针，每个指针都指向内存中的某个数组，所以得到的是指向数组的指针的集合，也就是数组的数组。

```C++
#include <iostream>

int main()
{
    int* array = new int[50];

    int** a2d = new int*[50]; // 这里存储的是一个指针对象的缓冲区

    std::cin.get();
}
```

这里只是分配了一个可以存储200字节指针的内存块，并没有初始化。
然后我们可以遍历并设置每个指针指向一个数组，这样就能得到一个包含50个数组的内存位置的数组

```C++
for (int i = 0; i < 50; i++)
        a2d[i] = new int[50];
```

三维数组（了解即可）：

```C++
int*** a3d = new int**[50];

for (int i = 0; i < 50; i++)
{
    a3d[i] = new int*[50];
    for (int j = 0; j < 50; j++)
        a3d[i][j] = new int[50];  // 两次解引用
    //  int** ptr = a3d[i];
    //  ptr[j] = new int[50];
}
a3d[0][0][0] = 0;
```

还是回到二维数组：

```C++
int** a2d = new int*[50]; // 这里存储的是一个指针对象的缓冲区

for (int i = 0; i < 50; i++)
    a2d[i] = new int[50];

a2d[0][0] = 0;
a2d[0][1] = 0;
a2d[0][2] = 0;
```

要注意的是，第一列是指针的索引，而不是整数的索引，处理的其实是

```C++
int** a2d = new int*[50]; // 这里存储的是一个指针对象的缓冲区
```

而右边的第二列才是在处理内部数据。

因为是堆分配的，所以需要`delete[]`来释放内存。但是要注意的是，`delete[] a2d`只是一个保存实际整型数组的指针的数组，如果删除了会造成存储实际数组的那50个数组的200字节内存空间泄漏，因为我们无法访问到它们了。所以需要：

```C++
for(int i = 0;i < 50;i++)
    delete[] a2d[i];
delete[] a2d;
```

2. 缓存不命中

```C++
int** a2d = new int*[5]; //创建一个5*5的二维数组

for (int i = 0; i < 5; i++)
    a2d[i] = new int[5];
```

按上面的方法处理数组的数组会造成*memory fragmentation*（内存碎片）问题，我们不是在一个连续的内存缓冲区，即在一行中保存这25个整数，实际上是创建了5个单独的缓冲区，每个缓冲区有5个整数，除非你采取某些自定义分配方式或者采用内存池分配，它们会被分配到内存中完全随机的位置。所以我们读取完一行要跳到下一行读取数组内容时会导致*cache miss*（缓存不命中），这意味着我们会浪费时间从RAM中获取数据。实际上用这种方式遍历这25个整数要比我们只分配一个一维数组要慢得多（一维数组内存分配都在一行），最重要的一点是当你在编程和优化时，你可以优化的最重要的事情之一就是优化你的内存访问。所以如果你能将要访问的内存存储在一起，那么你在定位数据时会有更多的*cache hits*（缓存命中）以及更少的*cache miss*，你的程序会更快。

你能想出一个更好的方法来存储25个整数吗?

你可以把它们存储在一个一维数组里：

C++

```cpp
int* array = new int[5 * 5];
for(int y=0;y<5;y++)
    for(int x=0;x<5;x++)
    {
        array[x + y * 5] = 2;  // y维每+1，就向前跳5个数，相当于网格中跳一行
    }
```