一．静态数组：

**1.用指针访问一维数组的方法**

int a[5],\*p=a;

**循环控制:**

(1)用i

(2)用p=a;p<a+10;p++

**元素访问:**

(1)p不变,a[i],p[i],\*(a+i),\*(p+i)

(2)p改变,\*p

**地址访问：**

(1)p不变, &a[i],&p[i],a+i,p+i

(2)p改变, p

注意:

(1)数组名是指针常量,不能改变

(2)**若指针变化,再一次访问数组时,注意指针复位**

**2.用指针访问二维数组的方法（一级指针、二级指针）**

**（1）用一级指针变量操作二维数组**

将二维数组视为按行拉长的一维数组

**定义与初始化:** int a[3][4], \*p=a[0];或int \*p=&a[0][0];

**循环控制:** for(i=0;i<12;i++)

或 for(p=a[0];p<a[0]+12;p++)

**元素访问:** p不变 \*(p+i)

p改变 \*p

**指针移动:** 按元素存放顺序移动

**（2）用指向行的指针变量操作二维数组(数组指针)**

**定义形式为：** 类型 （\*变量名）[行长]；

**定义与初始化:** int a[3][4], (\*p)[4]=a;

**循环控制:** for(i=0;i<3;i++)

for(j=0;i<4;j++)

**元素访问:** p不变 \*(\*(p+i)+j) 或 \*(p[i]+j) 或 p[i][j] （常用p不变）

p改变 \*(\*p+j)

**指针移动:** 按行移动

**（3）用指针数组操作二维数组**

**定义与初始化:** int \*p[3]={a[0],a[1],a[2]};

**循环控制:** for(i=0;i<3;i++)

for(j=0;i<4;j++)

**元素访问:** p[i]不变 \*(\*(p+i)+j) 或 \*(p[i]+j) 或 p[i][j] （常用p[i]不变）

p[i]改变 \*(p[i])

**指针移动:**每个指针p[i]都横向移动,但p不能改变.

**例如:** for(i=0;i<3;i++)

{cout<<endl;

for(j=0;j<4;j++)

cout<<\*(p[i]+j)<<" ";

}

**（4）用二级指针操作二维数组**

如：int i,j,a[3][4];

int \*p1[3]={a[0],a[1],a[2]};

int \*\*p2=p1;

二．动态分配与回收（使用new和delete）

int \*p1,\*p2, (\*p3)[4],\*p4;

p1=new int; //为整型变量分配堆内存

\*p1=1; //为整型变量赋值

p2=new int[2]; //为一维数组分配堆内存

p2[i]=2; //一维数组第i个元素

p3=new int[3][4]; //为二维数组分配堆内存，行指针

p3[i][j]=2; //二维数组第i,j个元素

p4=new int[2\*3]; //为二维数组分配堆内存，一级指针

p4[i\*3+j]=2; //二维数组第i,j个元素p3[i][j]

delete p1; //释放简单变量空间

delete []p2; //释放数组空间

delete []p3; //释放数组空间

delete []p4; //释放数组空间

注意：new将动态分配的首地址赋值给一个指针变量，该指针变量的指向不应被改变，用于delete释放堆内存。

C++输入输出缓冲区问题

缓冲区是内存的一部分，用来缓冲输入或输出的数据，以减少磁盘读写次数，提高效率；当C++进行读写操作时，不一定马上执行输入、输出操作，而是当清空缓冲区时才被读写。打印机就是一个典型的例子，有打印缓冲区！

一．缓冲区类型

* 全缓冲（磁盘读写、socket）
* 行缓冲（键盘输入，遇到换行符才结束）
* 无缓冲（标准错误）cerr<<”error!”;

二．刷新缓冲区

* 缓冲区满（不确定性）
* 执行flush语句
* 执行endl语句(cout<<endl;)
* 输入输出交替进行
* 关闭文件（程序结束）