北京交通大学
Beijing Jiaotong University

程序设计分组训练

习题课1

北京交通大学计算机科学与信息技术学院

授课人：田　媚

实验1-A

int (*p)[4] 和 int *p[4]的区别！！！

- **int *p[4]**
  - “[]”的优先级别大于“*”的优先级别;
  - “[]”的优先级别高，所以它首先是个大小为4的数组即p[4];
  - 剩下的“int *”作为补充说明，既说明该数组的每一个元素为指向一个整型类型的指针;
  - **int *p[4]声明的是一个长度为4的指针数组。**

实验1-A

int (*p)[4] 和 int *p[4]的区别！！！

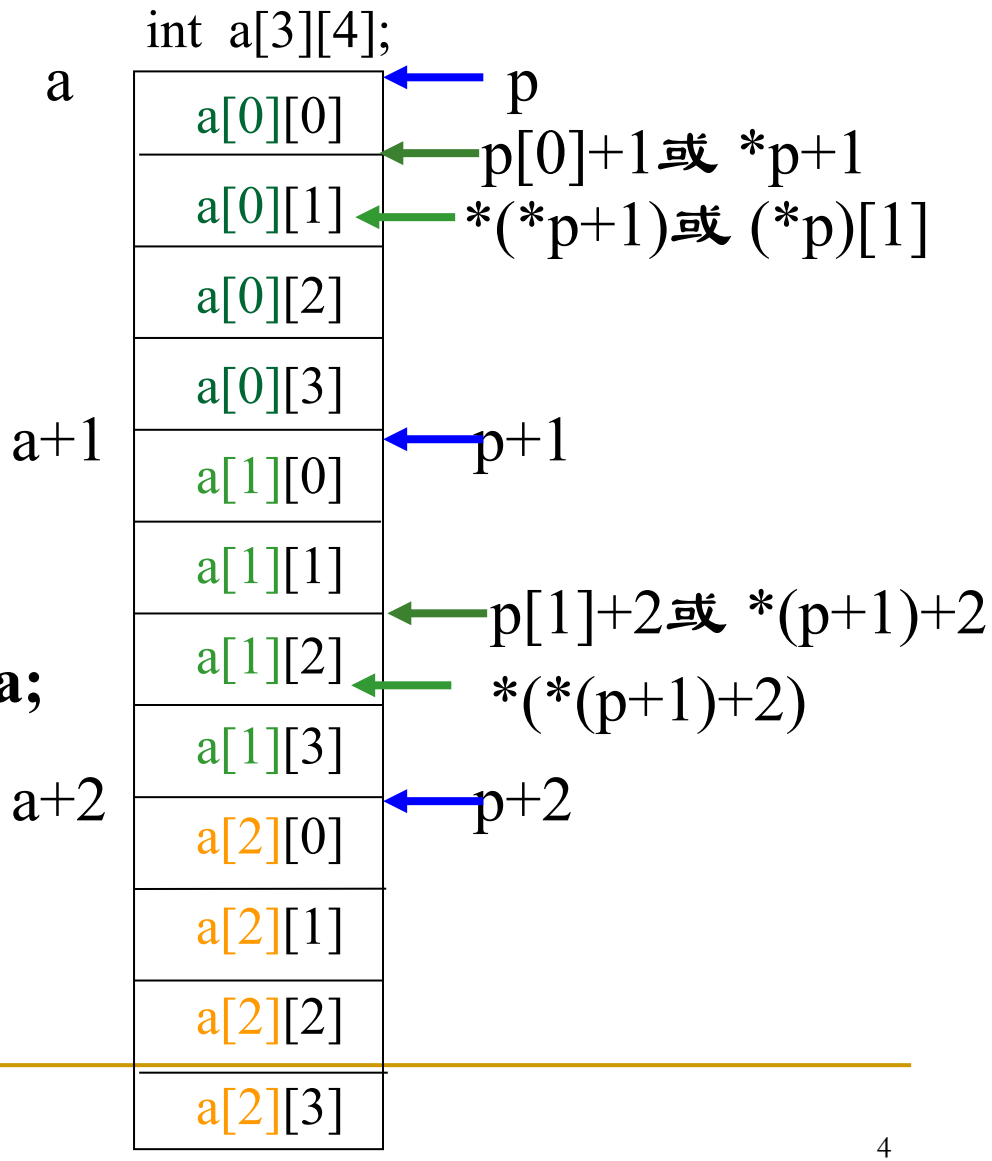
- **int (*p)[4]**
  - “()”的优先级别大于“[]”的优先级别;
  - “()”的优先级别高，首先是个指针，即*p;
  - 剩下的“int [4]”作为补充说明，即说明指针p指向一个长度为4的数组;
  - **int (*p)[4]声明的是指向长度为4的int数组的指针。**

指向一维数组的指针变量

- 定义形式： 数据类型 (*指针名)[一维数组维数];

例 int (*p)[4];

( )不能少
int (*p)[4]与int *p[4]不同
p的值是一维数组的
首地址，p是行指针



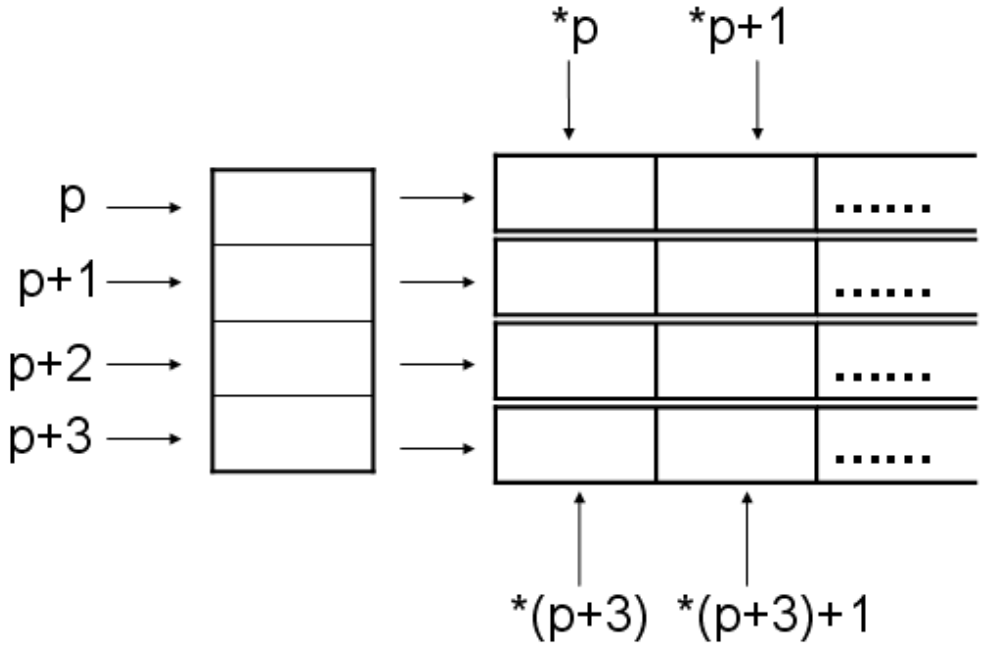
可让p指向二维数组某一行

如 int a[3][4], (*p)[4]=a;

一维数组的指针变量维数
和二维数组列数必须相同

对 *(*(p+i)+j)的理解：

p+i	相当于a数组的第i行的首地址；
*(p+i)	相当于a数组的第i行第0个元素的地址； 等价于a[i]；
*(p+i)+j	相当于a数组第i行第j列的元素的地址； 等价于&a[i][j] 等价于p[i]+j 等价于a[i]+j
((p+i)+j)	等价于a[i][j]



实验1-A

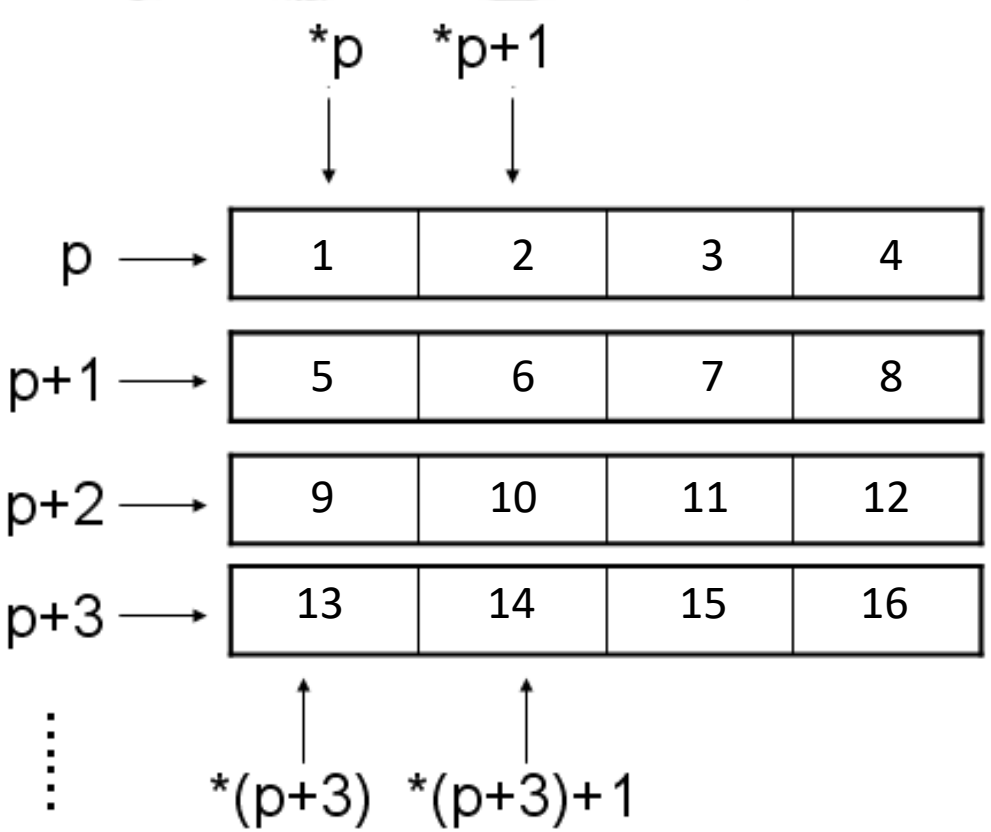
【问题】C语言中指针变量加1，意味着指针向后移动几个字节?

- **移动字节数与指针指向的数据类型有关**
  - char*型指针加1，向后移动1个字节;
  - int*型指针加1，向后移动4个字节;
  - 类比来看，任意指针变量加1，会移动和指针所指向数据类型所占用空间相同的字节;

- int *p，p + 1 相当于 p + sizeof(int)
- Int (*p)[4]，p + 1 相当于 p + sizeof(int[4]);

实验1-A

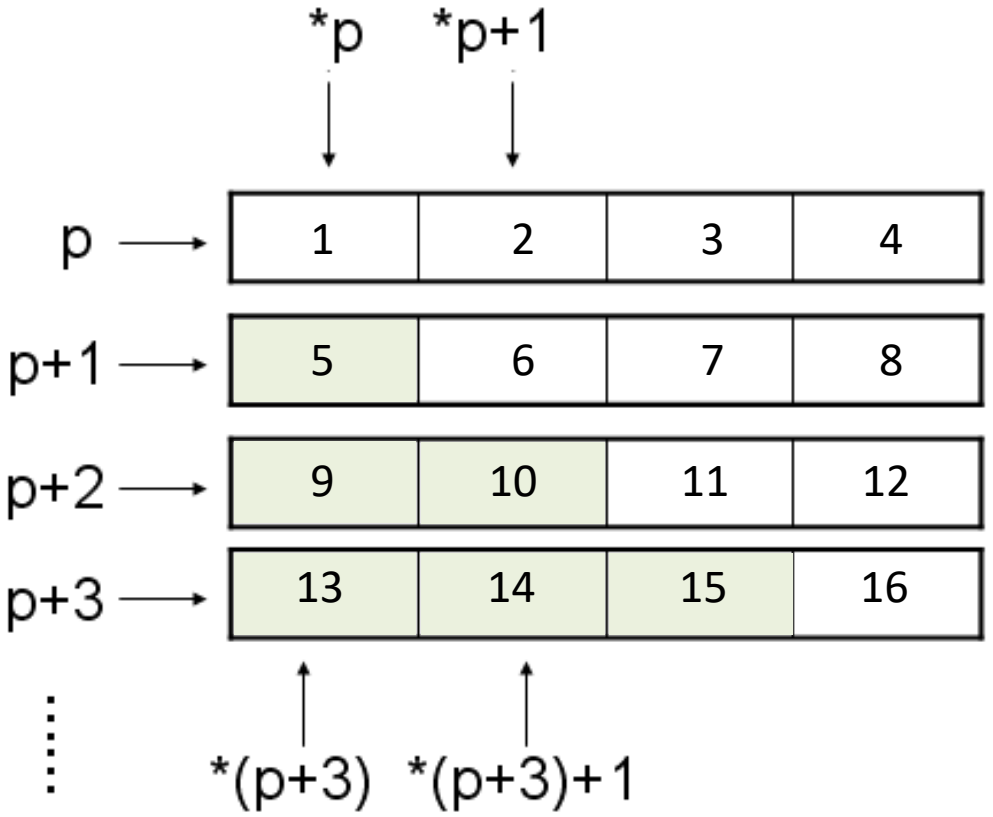
```
……
int a[4][4] = {{1,2,3,4}, {5,6,7,8},
{9,10,11,12},{13,14,15,16}};

int i, j, temp;
int (*p)[4] = a;
int *q = a[0];

for (i = 0; i < 4; i++)
{
   for(j = i + 1; j < 4; j++)
   {
      temp = *(*(p + j) + i);
      *(*(p + j) + i) = *(q + j);
      *(q + j) = temp;
   }
   q = q + i * 3;
}
……
```



实验1-A

```
……
for (i = 0; i < 4; i++)
{
   for(j = i + 1; j < 4; j++)
   {
      temp = *(*(p + j) + i);
      *(*(p + j) + i) = *(q + j);
      *(q + j) = temp;
……
```

i	j	*(*(p+j)+i)
0	1	5
0	2	9
0	3	13
1	2	10
1	3	14
2	3	15

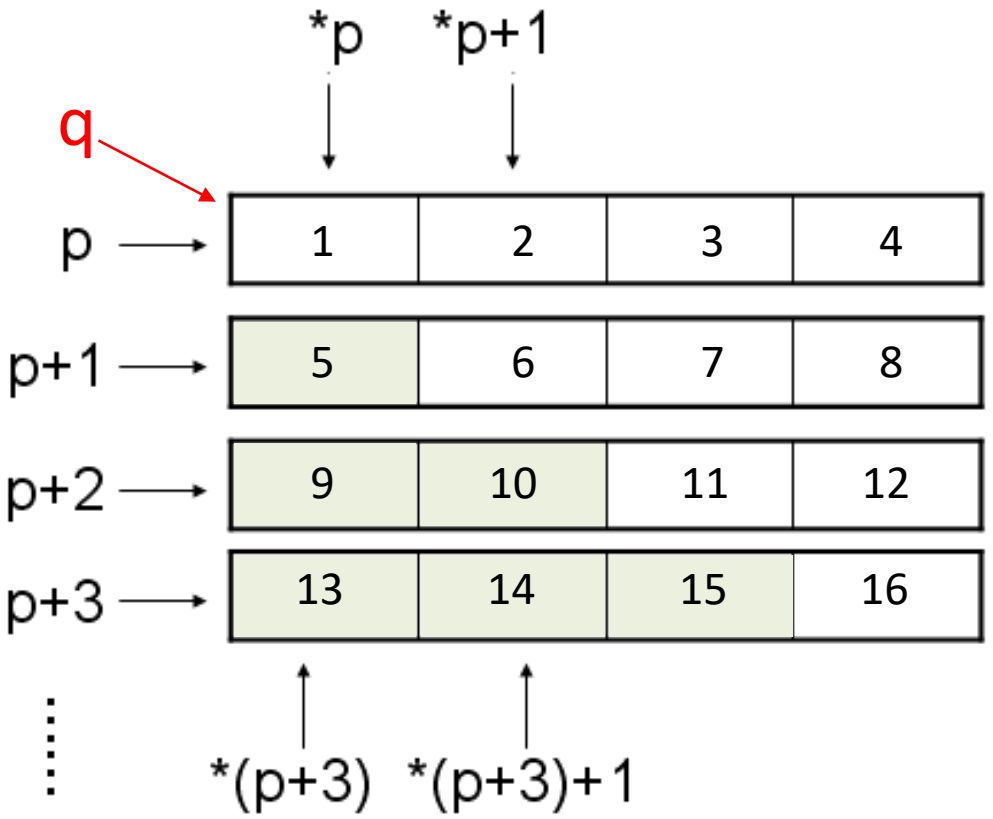


可以看出，在指针p的使用上，没有问题！

实验1-A

```
……
for (i = 0; i < 4; i++)
{
    for(j = i + 1; j < 4; j++)
    {
        temp = *(*(p + j) + i);
        *(*(p + j) + i) = *(q + j);
        *(q + j) = temp;
……
```

i	j	*(*(p+j)+i)	*(q+j)
0	1	5	2
0	2	9	3
0	3	13	4
1	2	10	3
1	3	14	4
2	3	15	4

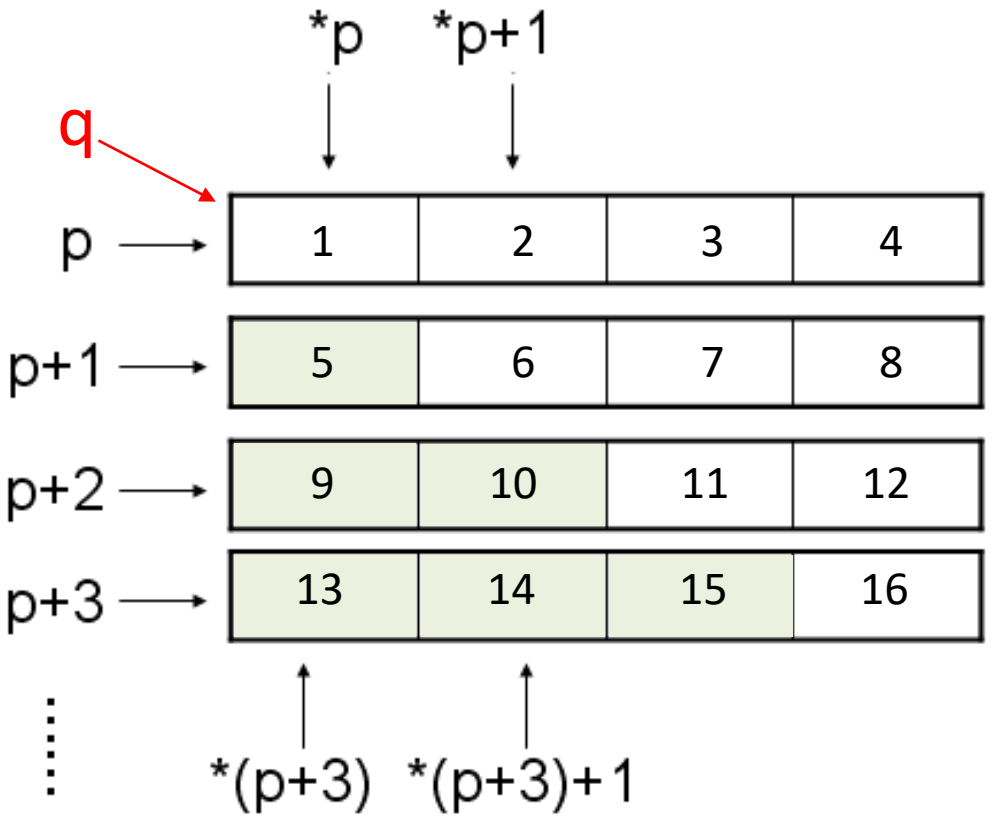


可以看出，程序正确运行的关键是在第一层循环外及时调整q指针的值！

实验1-A

```
……
for (i = 0; i < 4; i++)
{
   for(j = i + 1; j < 4; j++)
   {
      ……
   }
   q = q+i*4;
}
……
```

i	j	*(*(p+j)+i)	*(q+j)
0	1	5	2
0	2	9	3
0	3	13	4
1	2	10	3
1	3	14	4
2	3	15	8

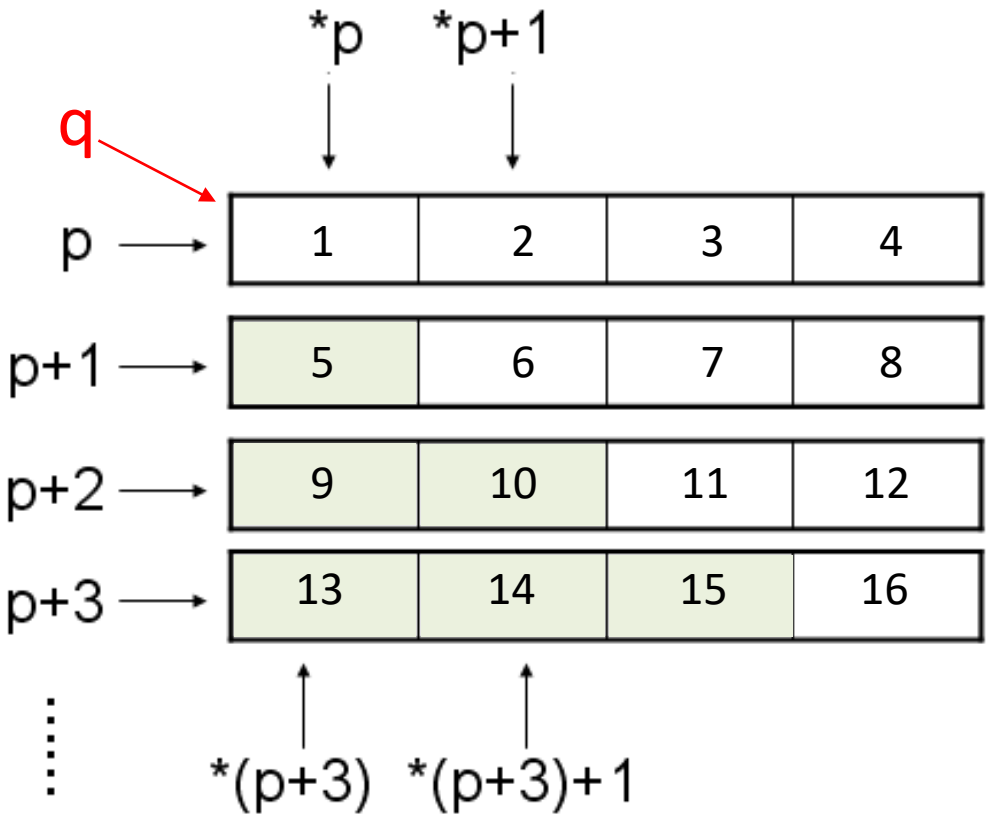


可以看出，错误就在于对指针q值的修改上！

实验1-A

```
……
for (i = 0; i < 4; i++)
{
   for(j = i + 1; j < 4; j++)
   {
      ……
   }
   q = q+4;
}
……
```

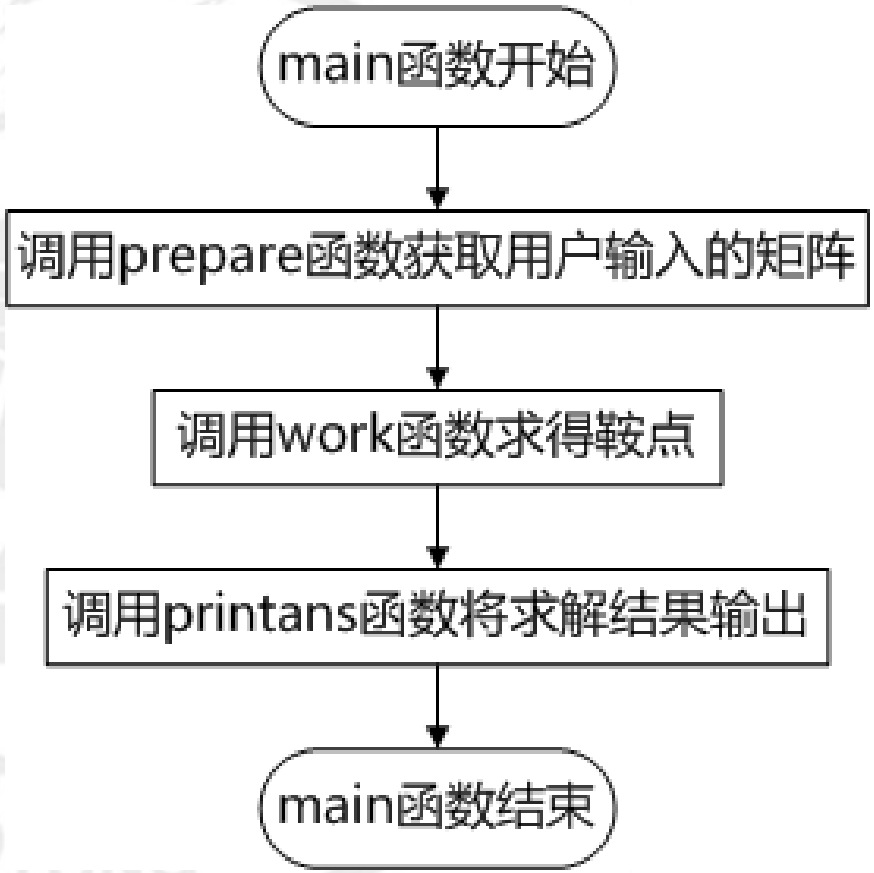
i	j	*(*(p+j)+i)	*(q+j)
0	1	5	2
0	2	9	3
0	3	13	4
1	2	10	7
1	3	14	8
2	3	15	12



当第一重循环完成后，将q指针指向二维数组的下一行，就可以正确地实现程序功能！

实验1-B

```
/*
*函数名称：_tmain
*函数功能：实现矩阵鞍点求解
*输入参数：无
*返回值：无
*版本信息：
*/
int main()
{
        prepare();
        work();
        printans();
        return 0;
}
```



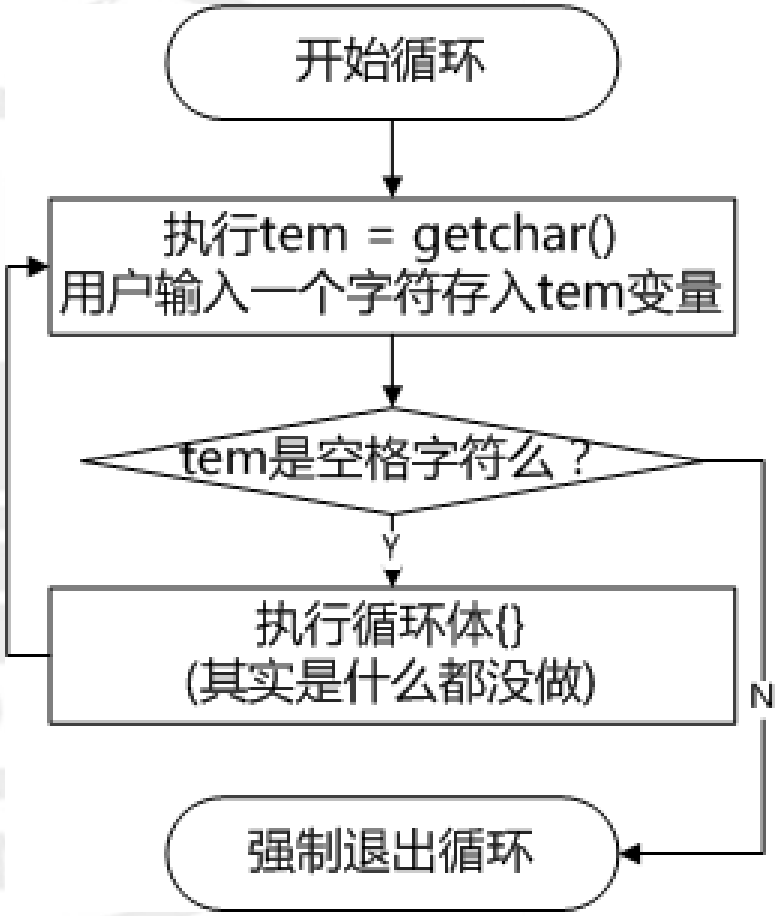
实验1-B

- **理解语句：** while(tem = getchar(), tem == ' '){};
  - 循环体内是一个逗号表达式；
  - 表达式值为非零，执行循环体；
  - 表达式值为零，退出循环体；
  - **逗号表达式：表达式1，表达式2。**
  - **逗号表达式的求解过程是：先求解表达式1，再求解表达式2。整个逗号表达式的值是表达式2的值。**

实验1-B

while(tem = getchar(), tem == ' '){};

```
……
While(1){
    tem = getchar();

    if(tem != ‘ ‘)
    {
        break;
    }
}
……
```



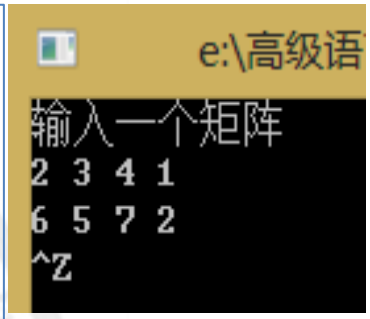
实验1-B

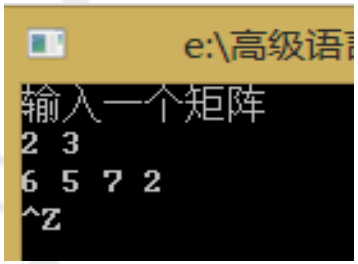
```
……
while(1)
{
   //如果用户输入空格，就空循环，直到用户输入非空格的字符
   while(tem = getchar(), tem == ' '){};

   if((tem == '\n') || (tem == EOF))
   {
      //用户输入回车或EOF时结束对矩阵一行数据的输入
      break;
   }

   //将用户输入的字符转换为数值存入matrix数组
   matrix[lenx][leny] = tem - '0';

   //将列标指向下一列
   leny = leny + 1;
}
……
```

- 如果用户输入两位数，程序会将其看做是两个一位数，分别存入matrix的两个单元
- 如果用户输入‘0-9’之外的字符，程序仍能正常运行，matrix里也会被存入数值

实验1-B

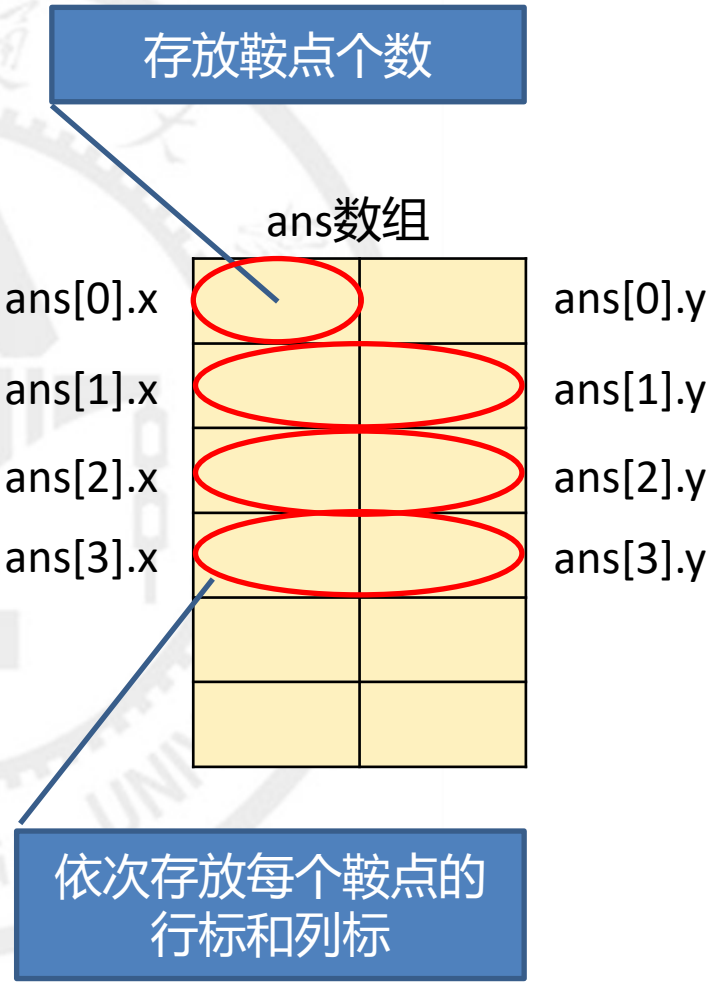
```
// matrix定义为全局变量，矩阵中所有元素值初始化为0
int matrix[1000][1000];
……
while(1)
{
   // Intem变量用来存放上一行用户输入的元素个数，如果用户输入的各行元素个数不同，那么用元素最多的那一行元素的个数作为矩阵的列数。缺少元素的行的元素由数组初始化定义为0.
   if(leny > Intem)
                        {
                                Intem = leny;
                        }
}
……
```



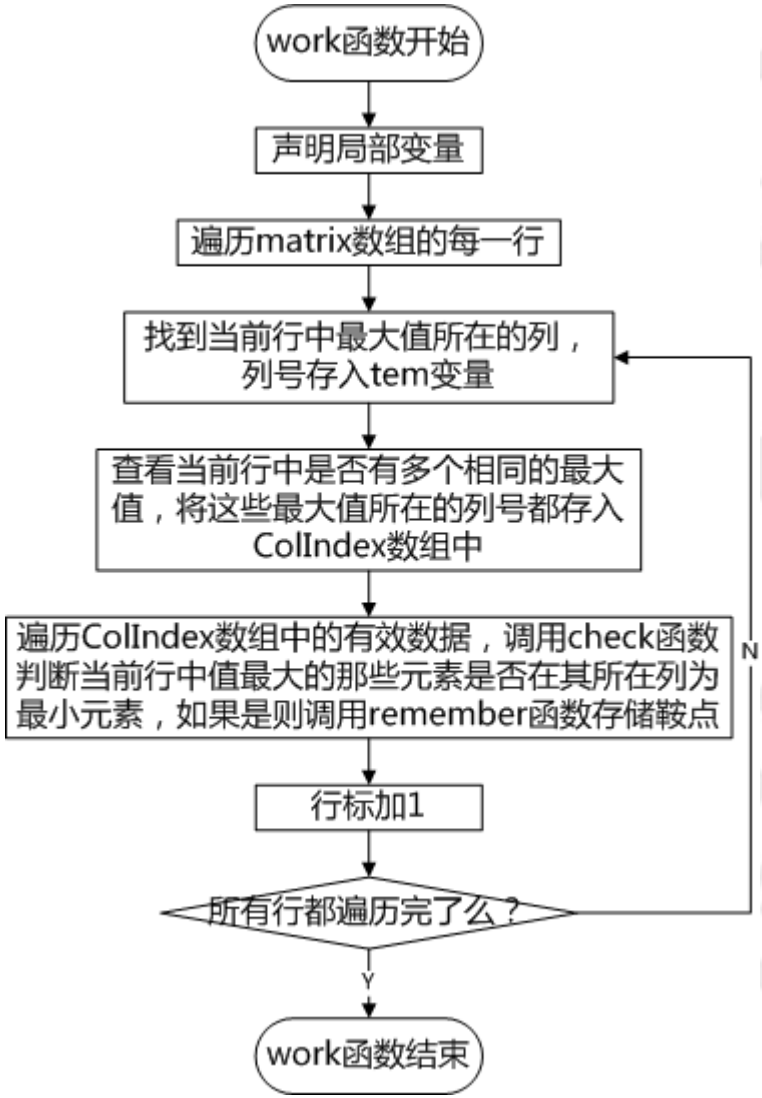


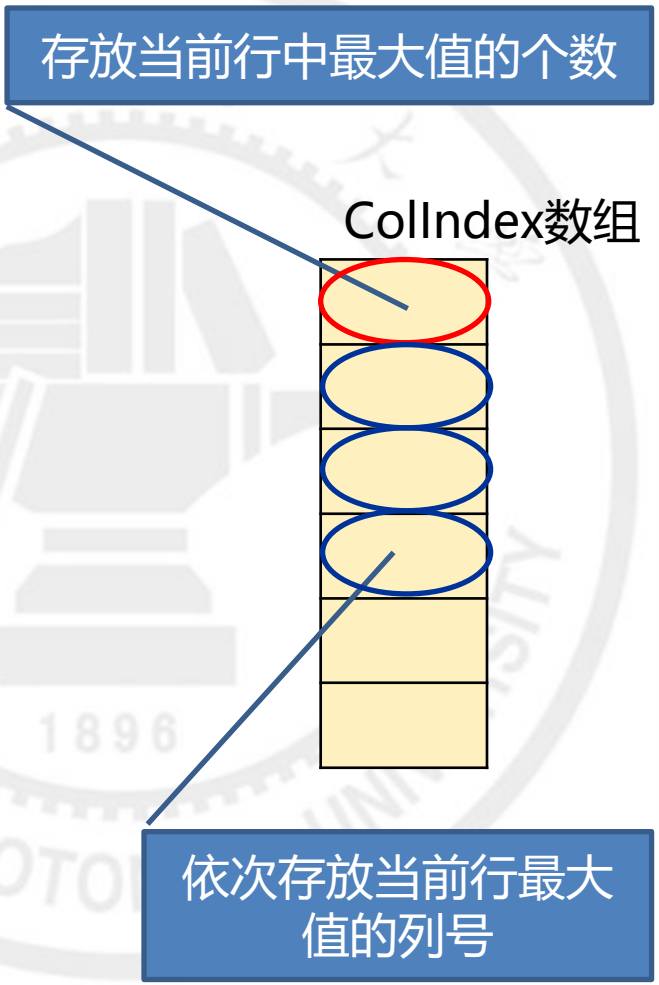
实验1-B

```
/*
*函数名称：remember
*函数功能：将检测到的鞍点存入结果数组
*输入参数：int x：要记录的鞍点行号
          int y：要记录的鞍点列号
*返回值：无
*版本信息：create by hansheng,2018-10-28
*/
void remember(int x, int y)
{
        ans[0].x = ans[0].x + 1;
        ans[ans[0].x].x = x;
        ans[ans[0].x].y = y;
}
```

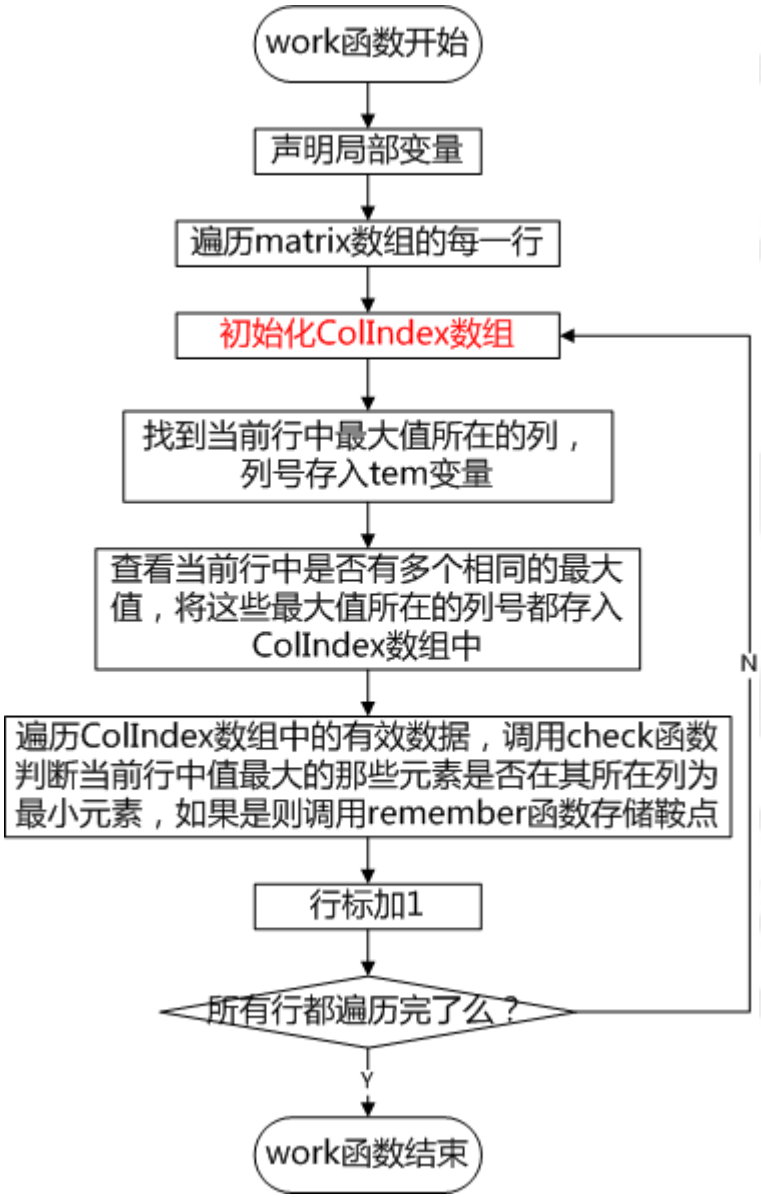


实验1-B


work函数开始
声明局部变量
遍历matrix数组的每一行
找到当前行中最大值所在的列，列号存入tem变量
查看当前行中是否有多个相同的最大值，将这些最大值所在的列号都存入ColIndex数组中
遍历ColIndex数组中的有效数据，调用check函数判断当前行中值最大的那些元素是否在其所在列为最小元素，如果是则调用remember函数存储鞍点
行标加1
所有行都遍历完了么？
N
Y
work函数结束


存放当前行中最大值的个数
ColIndex数组
依次存放当前行最大值的列号

实验1-B



在CodeForLab1B所给代码中，未在遍历每行时对矩阵中该行最大元素个数进行初始化，导致其进入

for(j = 1; j <= ColIndex[0]; j = j + 1)

时的循环次数出现问题（循环次数可能过多）

故在“tem = 0;”前加语句“ColIndex[0]=0;”，从而对ColIndex[0]进行初始化来解决此问题。

北京交通大學
Beijing Jiaotong University

希望同学们与老师一起

努力学习

收获知识与成功体验！