计算机程序设计语言（VC++）

第 4 章

数　　组

张晓如，华伟

《C++程序设计基础教程》
人民邮电出版社，2018.05

本章内容

4.1 数组的概念与定义

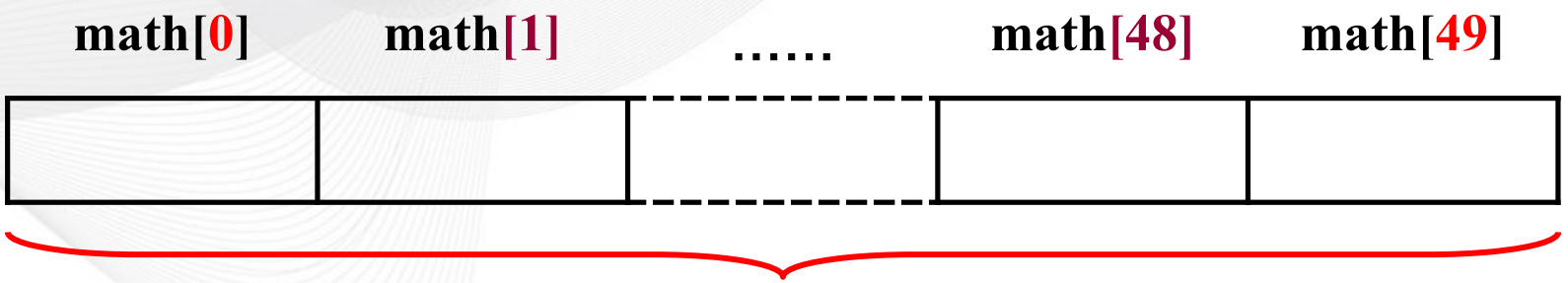
● 数组是有限个相同数据类型变量（元素）的集合。

4.1.1 一维数组

1. 定义格式：存储类型 数据类型 数组名 [数组大小];

- 默认存储类型：全局为静态，局部为自动；
- 数据类型：每个元素的数据类型；
- 数组大小：元素个数，通常为大于 0 的整型常量表达式

如： float math[50];



数组 math

- 数组名 math 是数组的首地址（ &math[0] ）

4.1 数组的概念与定义

2．一维数组的初始化

（1）以集合的形式列出所有元素的值。如：

```
int a1[5]={1,3,5,7,9};
```

a1[0]	a1[1]	a1[2]	a1[3]	a1[4]
1	3	5	7	9

（2）以集合的形式列出部分元素的值，其余元素的值为0。如：

```
int a2[5]={2,4,6 };
```

a2[0]	a2[1]	a2[2]	a2[3]	a2[4]
2	4	6	0	0

4.1 数组的概念与定义

- 定义和初始化数组时注意：
  - 定义一维数组时，若初始化，可省略数组大小。如：

    int a3[ ]={3,6,9};

    等同于

    int a3[3]={3,6,9}; // 根据列表中数据的个数确定数组大小
  - 列表中的数据个数可以小于或等于数组大小，但不能大于数组大小。如：

    int a4[5]={1,2,3,4,5,6}; // 语法错误

4.1 数组的概念与定义

3．一维数组的使用

- 数组使用是针对元素的，通常不能直接使用整个数组。如：

```
int a[5];
cin>>a;          // 语法错误
cout<<a;         // 逻辑错误
```

- 应该操作数组的各个元素，引用一维数组元素的格式如下：

  数组名 [元素位置]

  - 元素位置从 0 开始，到数组大小减 1 为止；
  - 元素位置通常为整型的变量或常量表达式；
  - 定义数组时用 [ ] 表示数组大小，定义后的数组用 [ ] 表示元素位置。

- 通常用循环语句操作数组，循环控制变量对应于元素位置。

4.1 数组的概念与定义

【例 4-1】保存从键盘输入的 10 个整数，并按 5 个一行的方式输出。

- 定义具有 10 个元素的整型数组 a，用于保存从键盘输入的数据。
- 通过循环语句从第一个元素到最后一个元素遍历数组，循环控制变量即为元素的下标，在遍历过程中输入每个元素。
- 再次通过循环语句遍历数组，输出每个元素，若输出的元素个数是 5 的倍数，则换行。

4.1 数组的概念与定义

【源程序代码】

```
int main(void) {
    int a[10],i;
    cout<<" 请输入十个整数： ";
    for(i=0;i<10;i++)
        cin>>a[i];
    cout<<" 输入的数据为： \n";
    for(i=0;i<10;i++) {
        cout<<a[i]<<'\t';
        if((i+1)%5==0)cout<<'\n';
    }
    cout<<'\n';
    return 0;
}
```

输入的数据为：
1 2 3 4 5
6 7 8 9 10

程序运行结果
请输入十个整数：1 2 3 4 5 6 7 8 9
10

4.1 数组的概念与定义

4.1.2 二维数组

1. 定义格式

存储类型 数据类型 数组名 [数组行数][数组列数];

- 行数和列数与一维数组大小有相同要求；
- 二维数组的大小（元素个数）为行数和列数之积。

如： int b[3][4];

第 1 行（ b[0] ）	b[0][0]	b[0][1]	b[0][2]	b[0][3]
第 2 行（ b[1] ）	b[1][0]	b[1][1]	b[1][2]	b[1][3]
第 3 行（ b[2] ）	b[2][0]	b[2][1]	b[2][2]	b[2][3]

4.1 数组的概念与定义

4.1.2 二维数组

- 二维数组看成特殊的一维数组。
  - 将二维数组的每一行都当作一个特殊元素；
  - 每个特殊元素是一个一维数组，其大小为列数。
- 二维数组的元素在内存中按先行后列的次序连续存放。
- N 行 M 列的二维数组与 [N×M] 的一维数组存储方式相同。

如：“int b[3][4];”时，其内存保存形式为：

b[0][0]	b[0][1]	b[0][2]	b[0][3]	b[1][0]	b[1][1]	b[1][2]	b[1][3]	b[2][0]	b[2][1]	b[2][2]	b[2][3]

第 1 行　　第 2 行　　第 3 行

4.1 数组的概念与定义

2. 二维数组初始化

（1）以行为单位，列出所有元素或部分元素的值，没有列出的元素值为 0 。

```
int b1[3][3] = { {1,2,3},{4,5,6},{7,8,9}};
int b2[3][3] = { {1,2}, {3,4,5}};
```

（2）按元素的存储顺序，列出全部或部分元素的值，没有列出的元素值为 0 。

```
int b3[3][3] = { 1,2,3,4,5,6,7,8,9};
int b4[3][3] = { 1,2,3,4,5};
```

（3）定义并初始化二维数组时可省略行数，但不能省略列数。

```
int b5[ ][3]={ 1,2,3,4,5,6,7,8};          // 数组 b5 为 3 行
int b6[3][ ]={ 1,2,3,4,5};                // 语法错误
```

4.1 数组的概念与定义

3. 二维数使用

- 二维数组的使用也是针对元素的，其一般格式如下：

  数组名 [行位置][列位置]

- 行位置与列位置皆从 0 开始，到行数或列数减 1 为止。
- 用两层嵌套的循环操作二维数组，一层控制行，另一层控制列，循环控制变量分别与行位置和列位置对应。

【例 4-2 】用下列数据初始化二维数组，并按矩阵的方式输出。

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15

4.1 数组的概念与定义

3. 二维数使用

【程序设计】

- 定义并初始化二维数组：
  - 以行为单位列出所有元素的值；

    int b[3][5]={{1,2,3,4,5},{6,7,8,9,10},{11,12,13,14,15}} ；
  - 或按内存顺序依次列出所有元素的值。

    int b[3][5]={1,2,3,4,5,6,7,8,9,10,11,12,13,14,15} ；
- 遍历数组：外循环控制行（ i ： 0→2 ），内循环控制列（ j ： 0→4 ）。

```
for ( i=0;i<3;i++ )
    for ( j=0;j<5;j++ )
```

- 元素间用水平制表符‘ \t’ 分隔，行间用换行符‘ \n’ 分隔。

4.1 数组的概念与定义

【源程序代码】

```
#include<iostream>
using namespace std;
int main(void) {
    int b[3][5]={{1,2,3,4,5},{6,7,8,9,10},{11,12,13,14,15}},i,j;
    for(i=0;i<3;i++) {                  // 外循环控制行
        for(j=0;j<5;j++)                // 内循环控制列
            cout<<b[i][j]<<'\t';        // i 为行，j 为列
        cout<<'\n';
    }
    cout<<'\n';
    return 0;
}
```

程序运行结果

```
1       2       3       4       5
6       7       8       9       10
11      12      13      14      15
```

4.1 数组的概念与定义

3. 二维数使用

【程序分析】

- 例 4-2 外循环控制行，内循环控制列，按先行后列的顺序输出。
- 若外循环控制列，内循环控制行，则可按先列后行的顺序，即实现转置输出。

```
for(i=0;i<5;i++) {                  // 外循环控制列
    for(j=0;j<3;j++)                // 内循环控制行
        cout<<b[j][i]<<'\t';   // j 为行，i 为列
    cout<<'\n';
}
```

程序运行结果

```
1       6
   11
2       7
   12
3       8
   13
4       9
```

4.2 字符数组与字符串

- 字符数组是数据类型为字符型的特殊数组。

4.2.1 字符数组的定义及初始化

1. 字符数组的定义：

存储类型 char 数组名 [数组大小];

如： char str[8];

则：

str[0]	str[1]	str[2]	str[3]	str[4]	str[5]	str[6]	str[7]

字符数组 str

字符串“ student” 的存储：

‘s’	‘t’	‘u’	‘d’	‘e’	‘n’	‘t’	‘\0’

字符串“ student”

4.2 字符数组与字符串

2. 字符数组初始化

（1）用列表对每个元素赋值，未列出值元素值为0（‘\0’）。

```
char s1[10]={'s' , 't' , 'u' , 'd' , 'e' , 'n' , 't'};
char s3[]={115,116,117,100,101,110,116};
```

s1/s3/s5	‘s’	‘t’	‘u’	‘d’	‘e’	‘n’	‘t’	‘\0’	‘\0’	‘\0’
s2	‘s’	‘t’	‘u’	‘d’	‘e’	‘n’	‘t’			
s4	‘s’	‘t’	‘u’	‘d’	‘e’	‘n’	‘t’	‘\0’		

（2）直接用字符串初始化字符数组。

```
char s3[10]="student";
char s4[]="student";
```

（3）将字符串置于列表中初始化字符数组。

```
char s5[10]={"student"};
```

```
char s6[7]={"student"};
```
定义**错误**。为什么？

4.2 字符数组与字符串

4.2.2 字符数组的使用

1. 字符数组遍历

- 通过循环语句实现；
- 循环控制变量对应于元素的位置；
- 循环结束为字符串结束标志。

【例 4-3】编程求字符串“I am a student.”的长度。

- 定义字符数组 s，并用字符串“I am a student.”初始化；
- 定义整型变量 len 为字符串长度，初值为 0；
- 用循环语句遍历数组 s（i：0→s[i] 为‘\0’），每经历一个元素 len 值加 1。

4.2 字符数组与字符串

【源程序代码】

```
#include<iostream>
using namespace std;
int main(void) {
    char s[100]="I am a student.";
    int len=0,i;
    for(i=0;s[i]!='\0';i++)
       len++;
    cout<<" 字符串长度为： "<<len<<endl;
    return 0;
}
```

程序分析

- 数组 s 的大小应 ≥16 ；
- s[i]!=‘\0’等同于 s[i]!=0 ，或 s[i] ；
- len 和 i 初值相同，同步变化，程序可简写。

程序运行结果

字符串长度为： 15

4.2 字符数组与字符串

4.2.2 字符数组的使用

2. 字符数组输入 / 输出

- 整体输出

  cout<< 字符数组名；

- 整体输入

cin>> 字符数组名；

cin.getline (字符数组名，数组大小);

- cin 输入时，键盘输入的空格字符是数据分隔符；
- cin.getline 输入时，将空格字符作为输入数据的一部分，第二个参数包含字符串结束标志。

- 例 4-3 中，同样输入 I am a student. 。若用 cin 输入时，字符串长度为 1 ；而用 cin.getline 输入时，字符串长度为 15 。

4.2 字符数组与字符串

4.2.3 字符串处理函数

整体使用字符数组（字符串）的库函数，头文件为cstring 。

1. 字符串拷贝函数

char * strcpy （ char * ， char * ）；

- 用于字符数组赋值，将第二个参数复制给第一个参数，如：

```
char s1[20],s2[20]="China";
strcpy(s1,s2);                    //s1 为“ China”
```

2. 字符串拼接函数

char * strcat （ char * ， char * ）；

- 将第二个参数拼接到第一个参数的后面，如：

```
char s1[20]=" teacher  ",s2[ ]=" student ";
strcat(s1,s2);                    //s1 为“ teacher student”
```

4.2 字符数组与字符串

4.2.3 字符串处理函数

3. 字符串比较函数

int strcmp（ char * ， char * ）；

- 比较两个字符数组（字符串）大小；若相等返回 0 ，前大后小返回，前小后大返回－1 ；
- 比较规则：比较对应字符的 ASCII 码值的大小，如：

```
char s1[10]="ac",s2[20]="abc",s3[30]="abc\0xyz";
int i=strcmp(s1,s2),j= strcmp(s2,s3); // i 为 1 ， j 为 0 。
```

4. 求字符串长度函数

int strlen （ char * ）；

- 求字符串的长度，如 strlen(s3) 的值为 3 。数组 s3 的大小为 30 ；字符串“ abc\0xyz” 的长度为 3 ，大小为 8 。

4.3 数组与指针

4.3.1 指针变量的运算

- 指针变量（指针）能参与**赋值运算**、**部分算术运算**、**关系运算**和**逻辑运算**。
  - 指针所指的位置必须明确，其操作的内存空间要合法；
  - 分清操作对象是指针本身，还是指针所指的内存空间。

1. 赋值运算

- 对指针本身（ p ）赋值运算是改变指针所指的位置；
- 对指针所指对象（ *p ）赋值运算是改变指针所指内存空间的内容。如：

```
int a[5]={1,2,3,4,5},*p1,*p2;
p1=a,p2=p1;              // 指针赋值，p1 、 p2 均指向 a[0]
*p2=10;          // 指针所指对象赋值， a[0] 的值赋成 10
```

4.3 数组与指针

2. 算术运算

- 指针加 / 减整数，表示其后 / 前整数个存储单元的地址：

```
int a[5]={1,3,5,7,9},*p1=&a[3],*p2,*p3;          //p1 指向 a[3]
p2=p1+1,p3=p1-2;                                 // p2 指向 a[4] ， p3 指向 a[1]
p2--,p3++;                                       // p2 指向 a[3] ， p3 指向 a[2]
```

3. 关系运算

- 用于判断指针所指的位置关系：
  - 当关系成立时，结果为 true （ 1 ）；
  - 当关系不成立时，其结果为 false （ 0 ）。如：

```
int a[5]={1,3,5,7,9},*p1=a,*p2=a,*p3=a+2;
```

则 p1>=p2 、 p1<p3为真； p1!=p2、 p1>p2 为假。

4.3 数组与指针

4.3.2 一维数组与指针

- 定义指针变量指向数组首元素，以指针变量名代替数组名，实现数组操作。（指针所指位置不变）
- 指针变量从前到后依次指向数组各元素，通过指针的取内容运算得到对应元素。（指针所指位置不断变化）

【例 4-5】用下列数据初始化一维数组，并通过指针变量求元素的最大值。

8.2 6.5 3 9.7 12 2.8 7.6 15 10.3

- 定义实型指针 p 指向数组 b 的首元素， max 表示最大值；
- 以 p 代替 b ，通过循环语句输出数组的各元素；
- 指针 p 从第二个元素开始遍历数组，遍历过程中将比 max 大的元素赋给 max 。

4.3 数组与指针

【源程序代码】

```
#include<iostream>
using namespace std;
int main(void) {
    double b[]={8.2,6.5,3,9.7,12,2.8,7.6,15.6,10.3};
    double *p=b,max=b[0];                                    // A
    cout<<" 数组为 : \n";
    for(int i=0;i<9;i++){                                    // 输出数组
        cout<<p[i]<<'\t';    // B ，指针变量名 p 代替数组名 b
        if((i+1)%5==0)cout<<'\n';
    }
    cout<<endl;
```

A 行，p 指向 b 的首元素，b 即 &b[0]

B 行，p[i] 等同于 b[i] ，可以写成 *(b+i)、*(p+i) 等形式

4.3 数组与指针

【源程序代码】

```
    p++;  // p 指向 b[1] ，等同于 p =&b[1]
    for(i=0;i<9;i++){  // 求最大元素
        if(*p>max)max=*p;  // *p 是指针 p 所指元素的值
        p++;      // p 后移一个元素
    }
    cout<<" 元素的最大值为： "<<max<<"\n";
    return 0;
}
```

- 使用指针操作数组时，指针类型必须与数组类型一致。

程序运行结果

```
数组为：
8.2   6.5   3   9.7   12
2.8   7.6   15.6   10.3
元素的最大值为：15.6
```

4.3 数组与指针

4.3.4 字符数组与指针

- 字符型指针变量指向字符串
  - 定义时用字符串对其初始化；
  - 用字符串对指针变量赋值。

如：

```
char *s1=" C++ Program",*s2;
s2=" This is a string.";
```

- 直接引用字符型指针变量所指的字符数组

```
char str[50],*s3=str;
cin.getline(s3,50);                    // 输入字符数组
cout<<s3;                              // 输出字符数组
strcpy(s3,s1);                         // 复制字符数组
*s3=*s2;          // 字符赋值，等同 s3[0]=s2[0] ， str[0]=*s2
```

4.3 数组与指针

【例 4-7 】设计一个程序，将字符串中的字符逆序。如将“ I am a student.” 逆序为“ .tneduts a ma I” 。

- 数组 str 存储字符串 , 指针 s1 指向首元素 ,s2 指向尾元素。
- 当 s1 在 s2 前面时，将 s1 和 s2 所指的元素互换；然后 s1 后移一个元素， s2 前移一个元素。

【源程序代码】

```
int main(void) {
    char str[100],*s1=str,*s2=str,t;
    cout<<" 请输入一个字符串： ";
    cin.getline(s1,100);
    cout<<" 输入的字符串是： ";
    cout<<s2<<endl;                    // s2 等同 str 、 s1
```

4.3 数组与指针

【源程序代码】

```
    while(*s2) s2++;      // s2 指向结束标志 ( 循环条件的含义 ?)
    s2--;                                  // 前移一位，指向尾元素
    while(s1<s2) {                              // 当 s1 在 s2 前面时
        t=*s1,*s1=*s2,*s2=t;                  // 交换 s1 和 s2 所指元
          素
        s1++,s2--;  // s1 后移、 s2 前移 ( 指针前有没有 * 的区
          别 ?)
    }
    cout<<"逆序后的字符串是：";
    cout<<str<<endl;
}
```

s1 、 s2 ?

*str ?

程序运行结果

```
请输入一个字符串： I am a
  student.
输入的字符串是： I am a
  student.
逆序后的字符串是： .tneduts a ma
```

4.3 数组与指针

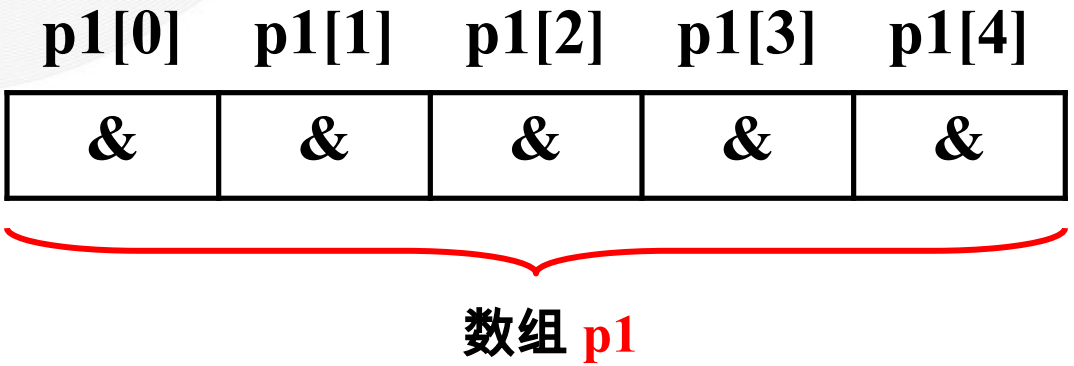
4.3.5 指针数组

- 各元素为指针变量的数组。
- 普通数组中存储的是普通数据（数值），指针数组中存储的是地址。
- 定义：

  存储类型 数据类型 * 数组名 [数组大小];

如：

float *p1[5] ;



4.4 数组与函数

- 按指针（地址）方式传递数组；
- 用指针使用数组。

4.4.1 一维数组与函数

1. 传递普通一维数组

- 函数原型说明

函数类型 函数名（数据类型 * 指针名 ,int 变量名）；

或

函数类型 函数名（数据类型 指针名 [ ],int 变量名）；

  - 第一个参数传递数组的首地址；
  - 第二个参数传递数组的元素个数。

- 函数调用

函数名 (数组名 , 数组大小);

【例 4-9 】设计一个程序 , 实现整型一维数组的输入 / 输出 , 要求数组的输入和输出通过两个函数实现。

```
void input(int *,int);    // 输入函数
void output(int [],int); // 输出函数
```

主函数：定义数组，调用函数实现输入 / 输出操作。

【源程序代码】

```
#define N 8
int main(){
    int a[N];
    cout<<“ 请输入数组： \n";       input(a,N);
    cout<<" 输入的数组为： \n";  output(a,N);  cout<<endl;
    return 0;
}
```

【源程序代码】

```
void input(int* p,int n){    // 指针变量传递数组首地址
      for(int i=0;i<n;i++){
                  cin>>*p;                    // 输入指针 p 所指元素
                  p++;                                  // 指针后移
            }
    }
    void output(int p[],int n) { // int *p
          for(int i=0;i<n;i++){
                  cout<<p[i]<<'\t';
                  if((i+1)%5==0)cout<<"\n";
            }
    }
```

4.4 数组与函数

4.4.1 一维数组与函数

2. 传递字符数组

对于字符数组，默认为到结束标志为止（隐含于字符串中），故可省略第二个参数。

- 函数原型

函数类型 函数名 (char* 指针名);

或

函数类型 函数名 (char 数组名 [ ]);

- 函数调用

函数名 (数组名);

【例 4-16】从键盘输入一个带空格的字符串，并将空格后的第一个小写英文字母改为大写英文字母。例如，从键盘输入“I am forever 25 years old.”时，输出“I Am Forever 25 Years Old.”。

- 定义函数 f 实现字符的转换，函数通过字符型指针传递字符串。形参：字符型指针，实参：字符数组名称。
- 通过循环语句遍历字符数组，将遍历过程中遇到符合题目要求的小写字母转换成大写字母。
- 大小写英文字母的转换
  - 小写字母 c 转换成大写字母的方法： c-=32 ；
  - 大写字母 c 转换成小写字母的方法： c+=32 。

【源程序代码】

```
void f(char *s){
    for(int i=0;s[i];i++)          // 遍历字符串，s[i] ： s[i]!='\0'
      if(s[i]==' ')                // 当前字符 s[i] 是空格
        if(s[i+1]>='a'&&s[i+1]<='z') // 下个字符是小写字母
          s[i+1]-=32;              // 小写字母转换为大写字母
}
int main(){ char str[100];
    cout<<" 请输入一个带空格的字符串 :";cin.getline(str,100);
    f(str); cout<<" 修改后的字符串为 : "; cout<<str<<endl;
    return 0;
}
```

cin>>str; ?

```
程序运行结果
请输入一个带空格的字符串：I am forever 25 years
  old.
修改后的字符串为： I Am Forever 25 Years Old.
```

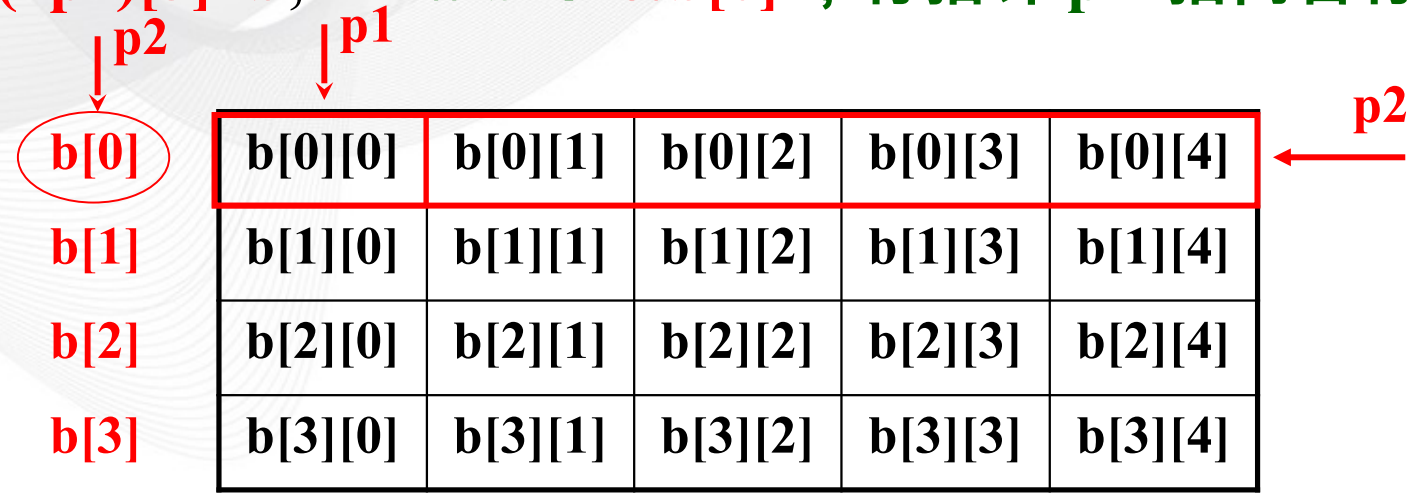
4.4 数组与函数

4.4.2 二维数组与函数

1. 二维数组的元素指针与行指针

- 指向元素的指针简称元素指针。（指向一个元素）
- 指向二维数组某行的指针简称行指针。（指向一行）
  - 行指针定义：数据类型 (* 指针变量名)[二维数组列数];

```
float b[4][5],*p1=&b[0][0]; // 元素指针 p1 指向首元素
float(*p2)[5]=b;            // b : &b[0] , 行指针 p2 指向首行
```



4.4 数组与函数

2. 行指针传递二维数组

- 函数原型

函数类型 函数名 (数据类型 指针变量名 (*)[N], int 变量名);

或

函数类型 函数名 (数据类型 指针变量名 [ ] [N], int 变量名);

  - 行指针中的 N 为常量，与二维数组的列数相同；
  - 第二个参数为二维数组的行数。

- 函数调用

函数名 (二维数组名，二维数组行数);

3. 行指针使用二维数组

- 用指向二维数组首行的行指针名代替二维数组名，实现二维数组的操作。

4.4 数组与函数

4.4.2 二维数组与函数

【例 4-11 】设计一个程序，求二维数组各元素的和，要求定义 sum 函数求和， print 函数输出二维数组。

```
int sum(int (*p)[5],int n); //  行指针 p 和数组行数 n
void show(int  p[][5],int n); //  行指针 p 和数组行数 n

int main()
{    int b[3][5]={{2,5,8,6,1},{4,12,9,5,9},{7,3,11,9,10}};
     show( b , 3 );
     cout<<" 各元素的和为 ： "<<sum(  b , 3 )<<"\n";
     return 0;
}
```

4.4 数组与函数

【源程序代码】

```
int sum(int p[][5],int n) { // 外围元素和， p 为行指针、 n 为行数
    int s=0,i,j;
    for(i=0;i<n;i++)
        for(j=0;j<5;j++)s+= p[i][j];
    return s;
    }
    void show(int (*p)[5],int n) { // 输出二维数组 , 指针等同 int p[][5]
        int i,j;
        for(i=0;i<n;i++) {
            for(j=0;j<5;j++)cout<<p[i][j]<<'\t';
            cout<<'\n';
        }
    }
```

4.5 程序举例

【例 4-12】设计一个程序，将一维数组中的元素从小到大排序，即升序排列。（选择排序法）

- 对于具有 n 个元素的一维数组 a ，共需进行 n-1 趟排序（外循环： i=0;i<n-1;i++ ）。
  - 第 1 趟排序，将最小元素放到第一个位置（ a[0] 位置）；
  - 第 2 趟排序，将次小元素，即剩余元素中的最小元素放到 a[1] 位置；
  - 以此类推，第 n-1 趟排序（最后一趟），将次大元素放到 a[n-2] 位置；最后剩下的元素（最大元素），自动进入 a[n-1] 位置。
- 每趟排序（内循环），将当前元素 a[i] 与其后的所有元素 a[j] 比较（ j=i+1 ;j<n;j++ ），若 a[i]>a[j] ，则 a[i]↔a[j] 。

4.5 程序举例

【源程序代码】

```
int main(){
    int a[10]={5,9,2,6,10,8,1,7,4,3};
    for(int i=0;i<9;i++)
        for(int j=i+1;j<10;j++)
          if( a[i]>a[j] ){int t=a[i];a[i]=a[j];a[j]=t;}
    for(i=0;i<10;){
        cout<<a[i]<<'\t';
        i++;
        if(i%5==0)cout<<'\n';
    }
    cout<<'\n';
    return 0;
}
```

降序 a[i]<a[j]

最多进行多少次交换？

n×(n-1)/2

程序运行结果

```
1   2   3   4   5
6   7   8   9   10
```

4.5 程序举例

■ 间接选择排序法

在每趟排序时，只记录当前最小（升序）或最大（降序）元素的位置，该趟排序结束后，若最小或最大元素不在当前位置，则交换，即每趟排序最多交换一个元素。

【源程序代码】

```
for(int i=0;i<9;i++) {                          //k 为最小元素所在位置
    int k=i;  // 设当前元素为最小元素，即最小元素位置 k 为 i
    for(int j=i+1;j<10;j++)                              // 查找最小元素
      if(a[k]>a[j])k=j;                          // 记录最小元素所在位置
    if(k!=i) {                          // 最小元素不在当前位置时交换元素
        int t=a[i];  a[i]=a[k];  a[k]=t;
    }
}
```

4.6 习题

1．找出一维数组中值最大的元素及其下标，最大元素可能不止一个。例如，{3，5，2，7，6，1，7，4，7，5}中的最大元素为7，其下标分别为3、6、8。具体要求如下。

（1）定义函数 int max（int *，int），返回数组元素的最大值。

（2）在主函数中，用测试数据初始化数组，并调用 max 函数完成测试。

2．求键盘输入的 N 个实数的方差。求方差的公式如下。

$$D=\frac{1}{n}\sum_{i=1}^{n}(x_i-\bar{x})^2 \quad \text{其中} \quad \bar{x}=\left(\sum_{i=1}^{n}x_i\right)/n$$

程序设计的要求如下。

（1）定义函数 void f（double p[]，int n，double &ave），求出数组 p 中 n 个元素的平均值，并通过参数 ave 带回主函数。

4.6 习题

3．求下列二维数组各元素的和。

2.6 5 8 6.3 1

4 12 9 4.5 9.6

7.2 3.8 11 7.9 10

程序设计的要求如下。

（1）定义函数 void print（float p[][5]，int n）；输出二维数组。

（2）定义函数 void fun（float（*p）[5]，int n，float *s）；求二维数组各元素的和，并通过参数 s 带回到主函数。

（3）在主函数中，用测试数据初始化二维数组，并调用 print 函数将其输出，调用 fun 函数得到各元素的和并输出。

4.6 习题

4．杨辉三角形是由正整数构成的一个矩阵，每行除最左侧与最右侧的元素为 1 外，其他元素等于其左上方与正上方两个数之和，如下所示。

```
1
1  1
1  2  1
1  3  3  1
1  4  6  4  1
……
```

（1）定义函数 void create（int p[][N]，int n），将杨辉三角的前 n 行保存到二维数组的下三角中。

（2）定义函数 void print（int（*p）[N]，int n），输出杨辉三角形。

（3）在主函数中调用上述函数，得到一个 N 阶的杨辉三角形。

4.6 习题

5．设计一个程序实现字符串的复制。具体要求如下。

（1）定义函数 char* copy（char*s1，char*s2），将 s2 赋值给 s1，并返回 s1。

（2）在主函数中输入两个字符串，测试 copy 函数。

6．设计一个程序，求键盘输入的一串字符中单词的个数。例如，“ I am a boy.” 中有 4 个单词。具体要求如下。

（1）定义函数 bool is_alphabet(char c)，若字符 c 是英文字母返回 true，否则返回 false。

（2）定义函数 int count（char*s），借助 is_alphabet 函数，求出字符串 s 中的单词个数并返回。算法提示：遍历字符串 s，若英文字母的下一个字符不是英文字母，则得到一个单词。

（3）在主函数中输入一个字符串，测试 count 函数。