# 《计算概论A》课程 程序设计部分

# 指针（2）


李　戈

北京大学 信息科学技术学院 软件研究所

lige@sei.pku.edu.cn


北京大学

# 二维数组与指针


北京大学

# 二维数组的地址

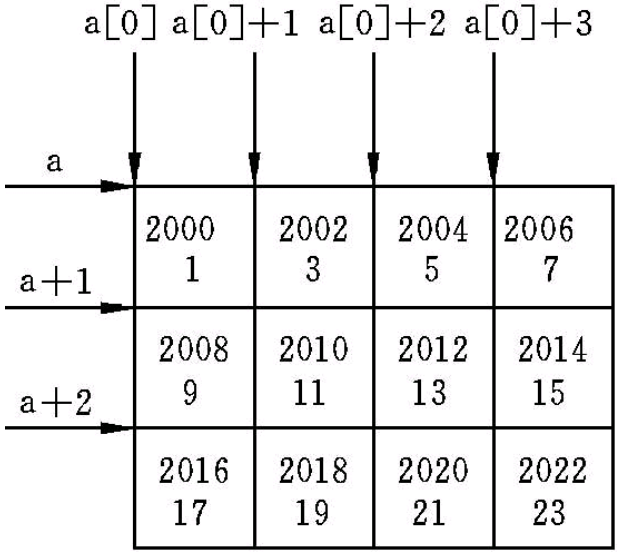
- 二维数组的地址
  - int a[3][4] = {{1,3,5,7},{9,11,13,15},{17,19,21,23}};
  - 由对一维数组的分析可知：

    “数组名是指向数组第一个元素的指针”；
  - 且二维数组的第一个元素是a[0]
    - a[0]是一个“包含四个整型元素”的一维数组；
  - 因此可以做出各种推断：
    - a与&a[0]等价；a[0]与&a[0][0]等价；
    - a[0]与*a等价；a[0][0]与**a等价；


北京大学

# 二维数组的地址

```
#include<iostream.h>
void main()
{
    int a[3][4]={{1,3,5,7,},{9,11,13,15},{17,19,21,23}};

    cout<<"        a = "<<a<<endl;
    cout<<"    &a[0] = "<<&a[0]<<endl<<endl;

    cout<<"      a+1 = "<<a+1<<endl;
    cout<<"  &a[0]+1 = "<<&a[0]+1<<endl<<endl;

    cout<<"       *a = "<<*a<<endl;
    cout<<"     a[0] = "<<a[0]<<endl;
    cout<<" &a[0][0] = "<<&a[0][0]<<endl<<endl;

    cout<<"     *a+1 = "<<*a+1<<endl;
    cout<<"   a[0]+1 = "<<a[0]+1<<endl;
    cout<<"&a[0][0]+1 = "<<&a[0][0]+1<<endl<<endl;
}
```



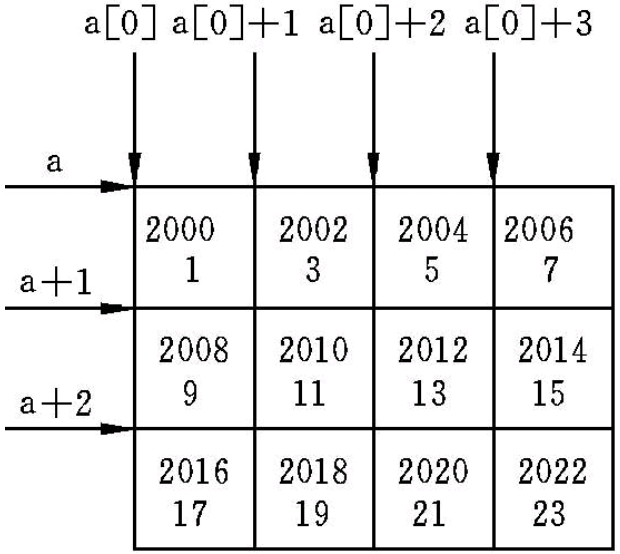
# 二维数组的地址

```
#include<iostream.h>
void main()
{
    int a[3][4]={{1,3,5,7,},{9,11,13,15},{17,19,21,23}};

    cout<<"        a = "<<a<<endl;
    cout<<"     &a[0] = "<<&a[0]<<endl<<endl

    cout<<"      a+1 = "<<a+1<<endl;
    cout<<"   &a[0]+1 = "<<&a[0]+1<<endl<<

    cout<<"       *a = "<<*a<<endl;
    cout<<"     a[0] = "<<a[0]<<endl;
    cout<<"  &a[0][0] = "<<&a[0][0]<<endl<<

    cout<<"     *a+1 = "<<*a+1<<endl;
    cout<<"    a[0]+1 = "<<a[0]+1<<endl;
    cout<<"&a[0][0]+1 = "<<&a[0][0]+1<<end
}
```

```
        a = 0x0013FF50
     &a[0] = 0x0013FF50

      a+1 = 0x0013FF60
   &a[0]+1 = 0x0013FF60

       *a = 0x0013FF50
     a[0] = 0x0013FF50
  &a[0][0] = 0x0013FF50

     *a+1 = 0x0013FF54
    a[0]+1 = 0x0013FF54
&a[0][0]+1 = 0x0013FF54

Press any key to continue
```

# 二维数组的地址

```
#include<iostream.h>
void main()
{
    int a[3][4]={{1,3,5,7,},{9,11,13,15},{17,19,21,23}};

    cout<<"        a = "<<a<<endl;
    cout<<"    &a[0] = "<<&a[0]<<endl<<endl;

    cout<<"      a+1 = "<<a+1<<endl;
    cout<<"  &a[0]+1 = "<<&a[0]+1<<endl<<endl;

    cout<<"     a[1] = "<<a[1]<<endl;
    cout<<"    &a[1] = "<<&a[1]<<endl;
    cout<<"   *(a+1) = "<<*(a+1)<<endl<<endl;

    cout<<"     *a+1 = "<<*a+1<<endl<<endl;

    cout<<"       &a = "<<&a<<endl;
    cout<<"     &a+1 = "<<&a+1<<endl;
}
```



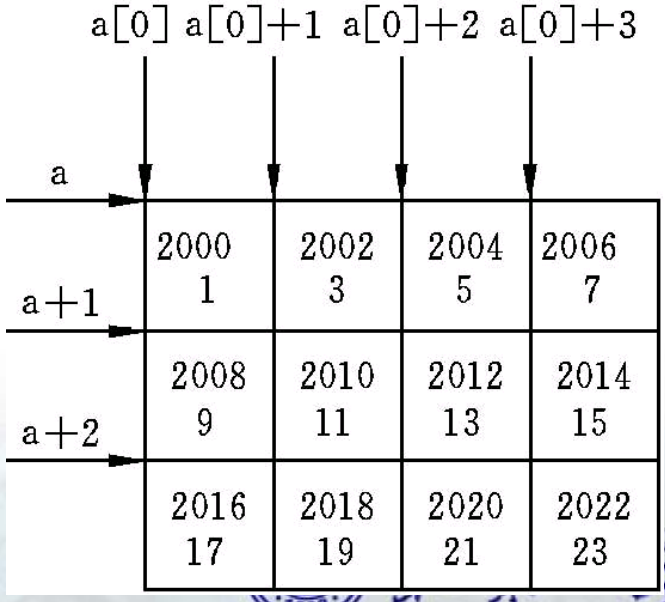
# 二维数组的地址

```
#include<iostream.h>
void main()
{
    int a[3][4]={{1,3,5,7,},{9,11,13,15},{17,19,21,23}};

    cout<<"       a = "<<a<<endl;
    cout<<"    &a[0] = "<<&a[0]<<endl<<endl;

    cout<<"     a+1 = "<<a+1<<endl;
    cout<<"  &a[0]+1 = "<<&a[0]+1<<endl<<endl;

    cout<<"     a[1] = "<<a[1]<<endl;
    cout<<"    &a[1] = "<<&a[1]<<endl;
    cout<<"   *(a+1) = "<<*(a+1)<<endl<<endl;

    cout<<"     *a+1 = "<<*a+1<<endl<<endl;

    cout<<"      &a = "<<&a<<endl;
    cout<<"     &a+1 = "<<&a+1<<endl;
}
```

```
       a = 0x0013FF50
    &a[0] = 0x0013FF50

     a+1 = 0x0013FF60
  &a[0]+1 = 0x0013FF60

     a[1] = 0x0013FF60
    &a[1] = 0x0013FF60
   *(a+1) = 0x0013FF60

     *a+1 = 0x0013FF54

      &a = 0x0013FF50
     &a+1 = 0x0013FF80
```

# 二维数组的地址

- 二维数组地址
  - int a[3][4] = {{1,3,5,7},{9,11,13,15},{17,19,21,23}};
  - 数组名a是“指向数组第一个元素”的指针；
  - “*a”等价于a[0]，相当于让a下沉了一级；
  - “&a”表示“指向二维数组”的指针，相当于上浮了一级；
- 几个有用的结论
  - a, a[0], &a[0][0]有相同的值；
  - a+1表示第1行的地址；
  - *(a+1)表示第1行第0列的地址；
  - *a+1表示第0行第1列的地址；
  - a[0]+1表示第0行第1列的地址；



# 字符串与指针

## —— 重温字符串

北京大学

# 程序分析（1）

```
#include <iostream>
using namespace std;
int main( ) {
    char h[] = "123";
    for(int i = 0; i < 10; i++)
        cin>>h[i];
    for(i = 0; i < 10; i++)
        cout<<h[i];
    cout<<endl;
    return 0;
}
```

可以运行，但危险！

它有可能占用了不允许突破的内存边界！

北京大学

# 程序分析（2）

```
#include<iostream>
using namespace std;
int main(){
   char h[ ] = "Peking";
    h[0]='a'; h[1]='b';
    h[2]='4'; h[3]='7';
    h[4]='c';
    cout<< h <<endl;
    return 0;
}
```

输出：ab47cg

北京大学

# 程序分析（3）

```
#include<iostream>
using namespace std;
int main()
{
   char h[]="123456";
   h = "abcdef";
   cout<<h<<endl;
   return 0;
}
```

error C2440: “=”: 无法从“const char [7]”转换为“char [7]”

# 程序分析（4）

```
#include<iostream>
using namespace std;
int main()
{
   char h[]="123456789";
   cin>>h;                    //键入12345678912345
   cout<<h<<endl;             //输出12345678912345
   return 0;                  //但，危险！
}
```

# 字符串与指针


北京大学

# 字符串与指针

- 指向数组的指针
  - **int a[10];  int *p;  p = a;**
- 指向字符串的指针
  - 指向字符串的指针变量：
  - **char a[10];  char *p;  p = a;**

北京大学

# 字符串指针举例

请说明一下程序完成了什么任务：

```
int main()
{
    char a[ ]= "How are you?", b[20];
    char *p1, *p2;
    for (p1 = a, p2 = b; *p1 != '\0'; p1++, p2++)
          *p2 = *p1;
    *p2= '\0';
    cout << "string a is :" << a<<endl;
    cout << "string b is :" << b<<endl;
    return 0;
}
```

北京大学

# 字符串指针举例

```
int main( )
{   char buffer[10] = "ABC";
    char *pc;
    pc = "hello";
    cout << pc << endl;
    pc++;
    cout << pc << endl;
    cout << *pc << endl;
    pc = buffer;
    cout << pc;
    return 0;
}
```

输出：

hello

ello

e

ABC

北京大学

```
#include<iostream.h>
int main()
{
    int a = 5;
    int *pa = &a;

    int b[6] = {1, 2, 3, 4, 5, 6};
    int *pb = b;

    char c[6] = {'h', 'e', 'l', 'l', 'o', '\0'};
    char *pc = c;

    cout<< a <<endl;
    cout<< pa <<endl<<endl;

    cout<< &b[0] <<endl;
    cout<< b <<endl;
    cout<< pb <<endl<<endl;

    cout<< &c[0] <<endl;
    cout<< c <<endl;
    cout<< pc <<endl;

    return 0;
}
```

```
#include<iostream.h>
int main()
{
    int a = 5;
    int *pa = &a;

    int b[6] = {1,2,3,4,5,6};
    int *pb = b;

    char c[6] = {'h','e','l','l','o','\0'};
    char *pc = c;

    cout<< a <<endl;
    cout<< pa <<endl<<endl;

    cout<< &b[0] <<endl;
    cout<< b <<endl;
    cout<< pb <<endl<<endl;

    cout<< &c[0] <<endl;
    cout<< c <<endl;
    cout<< pc <<endl;

    return 0;
}
```

```
5
0x0013FF7C

0x0013FF60
0x0013FF60
0x0013FF60

hello
hello
hello
Press any key to continue
```

```
#include<iostream.h>
int main()
{
    int a = 5;
    int *pa = &a;

    int b[6] = {1,2,3,4,5,6};
    int *pb = b;

    char c[6] = {'h','e','l','l','o','\0'};
    char *pc = c;

    cout<< a <<endl;
    cout<< pa <<endl<<endl;

    cout<< &b[0] <<endl;
    cout<< b <<endl;
    cout<< pb <<endl<<endl;

    cout<<static_cast<void*>(&c[0])<<endl;
    cout<<static_cast<void*>(c)<<endl;
    cout<<static_cast<void*>(pc)<<endl;

    return 0;
}
```

```
5
0x0013FF7C

0x0013FF60
0x0013FF60
0x0013FF60

0x0013FF54
0x0013FF54
0x0013FF54
```

# string 类型

北京大学

# string类型

- string类型
  - C++标准库中声明的一个字符串类
- 定义string类型变量
  - #include <string> //注意头文件名不是string.h
  - string string1;
  - string string2=″China″;
- 字符串变量的输入输出
  - cin>> string1;
  - cout<< string2;

北京大学

# string类型的运算

- string类型变量的赋值
  - string1="Canada";
  - string2=string1; //不要求string2和string1长度相同
- 用加号连接字符串
  - string string1="C++";
  - string string2="Language";
  - string1=string1 + string2;

    //连接后string1为"C++ Language"


北京大学

# string类型的运算

- 可以使用关系运算符

```
void main( )
{     string str1, str2, temp;
      cin>>str1>>str2;
      if(str1>str2){
            temp = str1;
            str1=str2;
            str1=temp;
      }
      cout<<str1<<“ ”<<str2<<endl;
}
```

北京大学

# string类型与字符数组

- 区别
  - string是“类”；
  - 在定义string类型变量时不需指定长度，长度随其中的字符串长度而改变。
  - string str1, str2 = “This is a test.”;
- 相似
  - 可以对字符串变量中某一字符进行操作：

```
string word="Then";
word[2]='a'; //修改后word的值为"Than"
```

北京大学

# 利用指针变量
# 引用多维数组中的元素


北京大学

# 遍历数组元素

```
#include<iostream>
using namespace std;
intmain( )
{
    int a[3][4] ={1, 3, 5, 7, 9, 11, 13, 15, 17,
    19, 21, 23};
    int *p;
    for(p=____; p<&a[0][0]+12; p++)
    {
        cout<<p<<" "<<*p<<endl;
    }
    return 0;
}
```

```
0x0013FF50 1
0x0013FF54 3
0x0013FF58 5
0x0013FF5C 7
0x0013FF60 9
0x0013FF64 11
0x0013FF68 13
0x0013FF6C 15
0x0013FF70 17
0x0013FF74 19
0x0013FF78 21
0x0013FF7C 23
```

# 遍历数组元素

```
#include<iostream>
using namespace std;
intmain( )
{
    int a[3][4] ={1, 3, 5, 7, 9, 11, 13, 15, 17,
    19, 21, 23};
    int *p;
    for(p=&a[0][0]; p<&a[0][0]+12; p++)
    {
        cout<<p<<" "<<*p<<endl;
    }
    return 0;
}
```

```
0x0013FF50 1
0x0013FF54 3
0x0013FF58 5
0x0013FF5C 7
0x0013FF60 9
0x0013FF64 11
0x0013FF68 13
0x0013FF6C 15
0x0013FF70 17
0x0013FF74 19
0x0013FF78 21
0x0013FF7C 23
```


北京大学

# 遍历每一个元素

- 举例

```
int main( ){
    int a[3][4]={1,3,5,7,9,11,13,15,17,19,21,23};
    int *p;
    for(p=&a[0][0]; p<&a[0][0]+12; p++){
        if (( p - &a[0][0] ) % 4 == 0 )
            cout<<endl;
        cout<<setw(4)<<*p;
    }
    return 0;
}
```

运行结果

```
1    3    5    7
9    11   13   15
17   19   21   23
```

北京大学

# 程序填空

- 输入 i, j; 输出a[i][j];

```
main( )
{
   int a[3][4]={1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23};
   int (*p)[4],  i,  j; //p应该如何定义，其基类型是什么？
   p = a;              //如果使用p = a，前面、后面如何填写？
   cin>>i>>j;
   cout<<setw(4)<<_____;  //利用p访问任一元素
}
```

北京大学

# 问题分析

- 从 p = a 开始
  - a 是a[3][4]的“第一个元素的地址”；
  - 所谓“第一个元素”是指一个“包含4个int型元素的一维数组”；
  - 所以，a是一个“包含4个int型元素的一维数组”的地址；
  - 因此，p的基类型应该是：

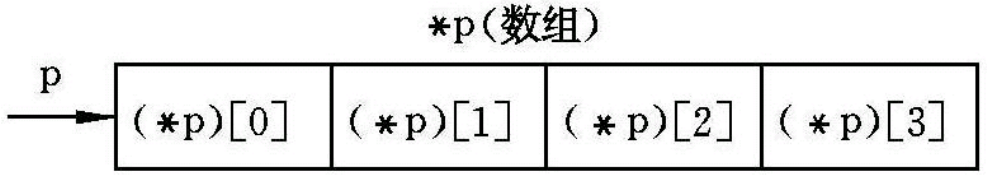
    “包含4个int型元素的一维数组”

北京大学

# 利用指针变量引用多维数组中的数组

- 问题
  - 如何定义一个指向“包含4个int型元素的一维数组”的指针变量？
- 解答
  - 变量定义语句：**int (*p)[4];**
  - 解释：
    - 对比 **int a[4];**

**int (*p)[4];**




北京大学

# 利用指针变量引用多维数组中的数组

- 输入 i, j; 输出a[i][j];

```
main( )
{
   int a[3][4]={1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23};
   int (*p)[4],  i,  j;
   p = a;
   cin>>i>>j;
   cout<<setw(4)<<*(*(p+i)+j);
}
```

北京大学

# 利用指针变量引用多维数组中的数组

- *(*( p + i ) + j ) 是什么？
  - p指向一个“包含4个int型元素的一维数组”；
    - a为二维数组中第一个元素的地址；
    - 因此p = a (p = &a[0])合法
  - p + i 是第i+1个“包含4个int型元素的一维数组”的地址。
  - p + i 等价于 &a[i];
  - *(p + i) 等价于a[i];
  - *(p + i) + j 等价于 a[i] + j
  - 因为：a[i] + j 等价于 &a[i][j]
  - *(*( p + i ) + j ) 等价于 a[i][j]

北京大学

# 程序填空

■ 输入 i, j; 输出a[i][j];

```
main( )
{
   int a[3][4]={1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23};
   int _______,  i,  j;
   p = a ;
   cin>>i>>j;
   cout<<setw(4)<<p[i][j];
}
```

北京大学

# 利用指针变量引用多维数组中的数组

- 输入 i, j; 输出a[i][j];

```
main( )
{
   int a[3][4]={1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23};
   int (*p)[4],  i,  j;
   p = a;
   cin>>i>>j;
   cout<<setw(4)<<p[i][j];
}
```

北京大学

# 指针数组


北京大学

# 指针数组

- 指针数组
  - 数组中的各个数组元素均为指针类型的数据
- 指针数组的定义
  - 定义普通数组：**int array[10];**
  - 定义指针数组：**int *pointer[10]**
  - 解释：
    - 因为：“**[ ]**”优先级高于“*****”
    - 所以：**pointer[10]**是数组，数组名为**pointer**
    - 数组名前面是数组类型“*****”
    - “*****”前是指针变量的类型“**int**”

北京大学

# 指针数组的用途举例

- 背景：
  - 图书馆有若干本书，每本书都有一个名字，它可以用字符串描述；
- 目标：
  - 要对所有的书名按字母排序
- 解决方案：
  - 把所有的书名“读入程序”，然后依次检查书名的第1、2、3…个字符的ASCII码，利用ASCII码排列大小。
- 问题：
  - 如何把所有的书名“读入程序”中呢？
  - 进一步的问题：读入程序中，需要有一个结构存储它们，这个结构是什么呢？

北京大学

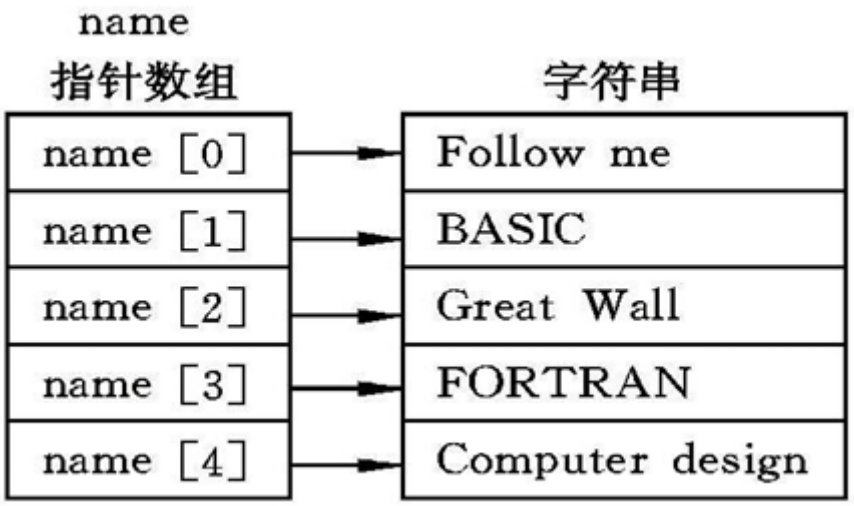
# 指针数组的用途举例

- 解决方案一
  - 每个书名都是一个字符串，可以用字符数组存储；
  - 要存储多个字符数组，可以选择使用二维数组；
  - 问题：
    - 在定义二维数组时，如何指定固定的列数？
- 解决方案二
  - 使用指针数组，针对每本书，设计一个指针变量，指向书名字符串；
  - 利用一个指针数组，存放所有指向书名的指针；

北京大学

# 指针数组的用途举例

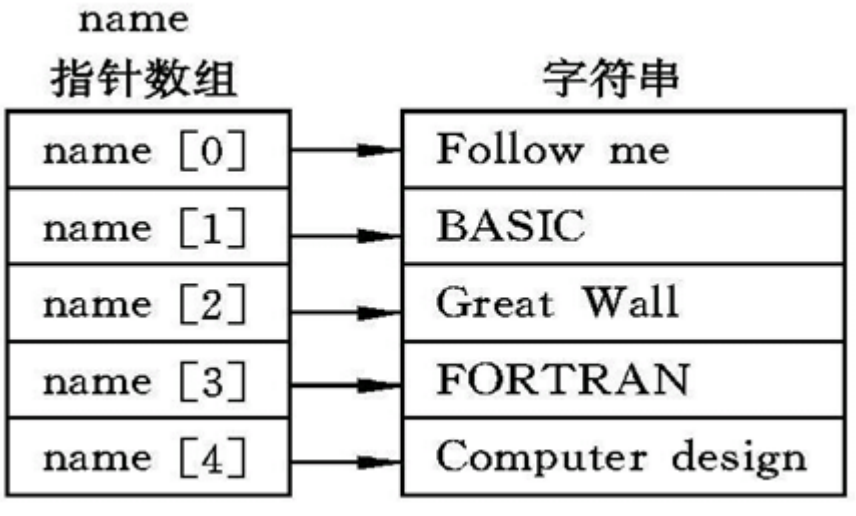
| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| F | o | l | l | o | w | | m | e | \0 | | | | | | |
| B | A | S | I | C | \0 | | | | | | | | | | |
| G | r | e | a | t | | w | a | l | l | \0 | | | | | |
| F | O | R | T | R | A | N | \0 | | | | | | | | |
| C | o | m | p | u | t | e | r | | d | e | s | i | g | n | \0 |

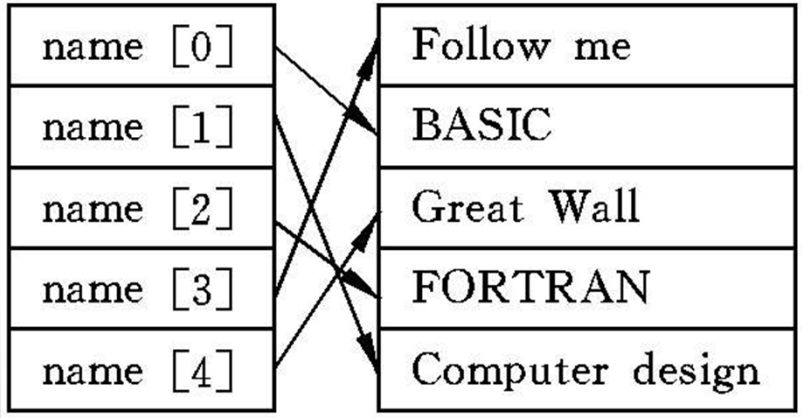



北京大学

# 指针数组的用途举例

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| F | o | l | l | o | w | | m | e | \0 | | | | | | |
| B | A | S | I | C | \0 | | | | | | | | | | |
| G | r | e | a | t | | w | a | l | l | \0 | | | | | |
| F | O | R | T | R | A | N | \0 | | | | | | | | |
| C | o | m | p | u | t | e | r | | d | e | s | i | g | n | \0 |




北京大学

# 指针数组的用途举例

- 指针数组的定义
  - **char *name[ ] =**

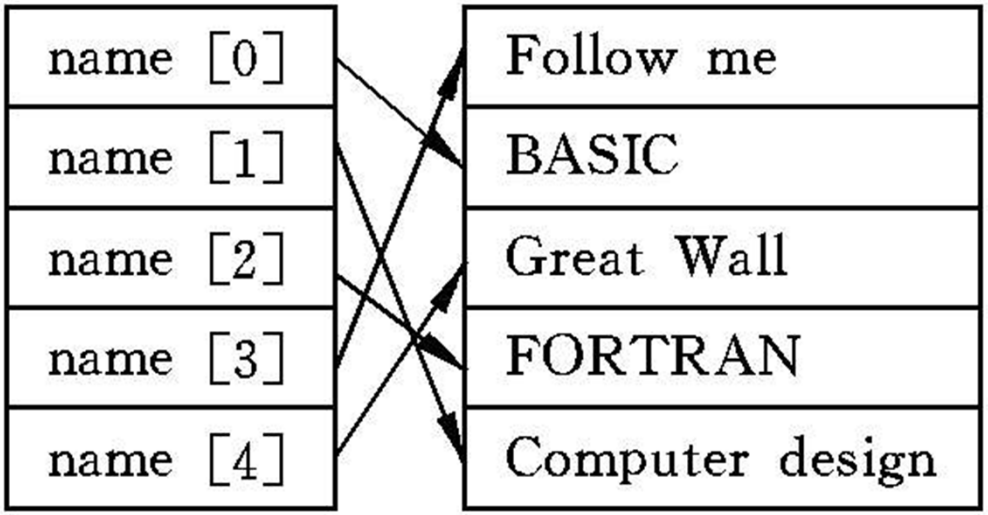
    **{"Follow me"，"BASIC"，"Great Wall"，"FORTRAN"，"Computer Design"};**
- 指针数组的访问
  - **name[i]**

    表示指向第i本书 书名 字符串的指针；

北京大学

# 指针数组的用途举例

```
#include<iostream.h>
#include<string.h>
void main()
{   char *name[ ] = {"Follow me", "BASIC", "Great Wall", "FORTRAN",
    "Computer design"};
    char *temp; int k;
    for (int i = 0; i < 4; i++)
    {       k = i;
            for (int j = i + 1; j<5; j++)
                    if(strcmp(name[k], name[j]) > 0) k = j;
            if (k != i)
            {       temp = name[i];
                    name[i] = name[k];
                    name[k] = temp;
            }
    }
    for (i= 0; i<5; i++)
            cout<<name[i]<<endl;
}
```



# string数组

- 可以用string定义字符串数组
  - string name[5];
  - string name[5]={"Zhang", "Li", "Fun", "Wang", "Tan"};
- name数组的状况：

| | | | | | |
|---|---|---|---|---|---|
| name[0] | Z | h | a | n | g |
| name[1] | L | i | | | |
| name[2] | F | u | n | | |
| name[3] | W | a | n | g | |
| name[4] | T | a | n | | |

北京大学

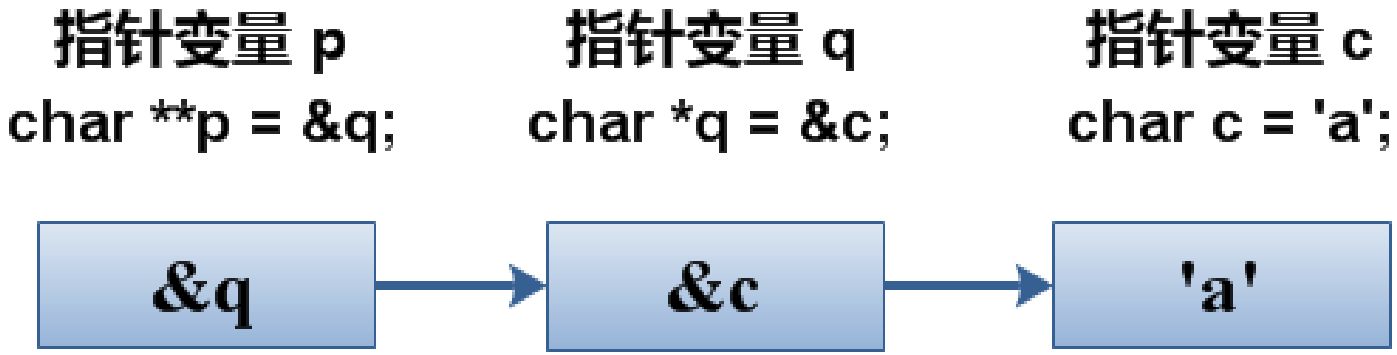
# 指向指针的指针

# 指向指针的指针

- 定义方式
  - **char c = ‘a’;**
  - **char *q = &c;**
  - **char * * p = &q;**

  定义一个：

  指向“指向**char**型数据的指针变量”的指针变量

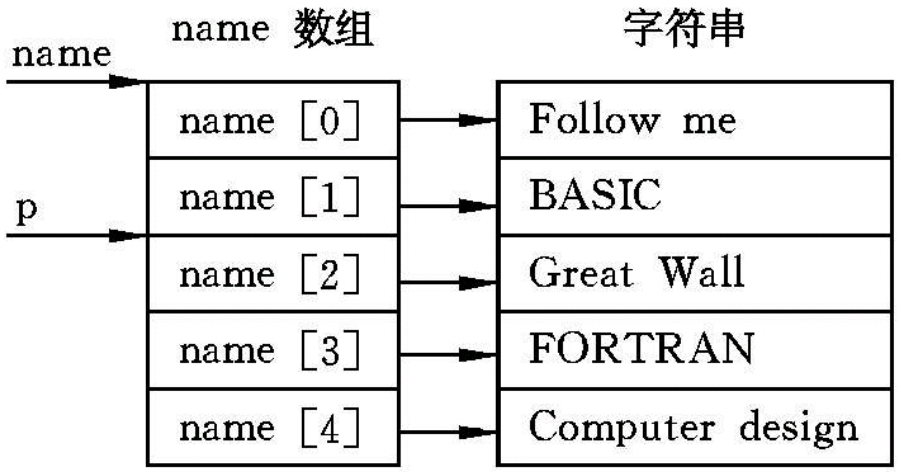



北京大学

# 指向指针的指针

- 推演
  - 理论上可以定义n多层，但实际意义不大


- 指向指针的指针 的 用途
  - 定义复杂的结构



# 程序填空

```
#include<iostream>
#include<string>
using namespace std;
void main(){
    char *name[ ] = {"Follow me", "BASIC", "Great
    Wall", "FORTRAN", "Computer design"};
    char _______;
    for (p = name; p<name+5; p++)
        cout<<*p<<endl;          //打印字符串name[i]
}
```

# 程序填空

```
#include<iostream>
#include<string>
using namespace std;
void main(){
    char *name[ ] = {"Follow me", "BASIC", "Great
    Wall", "FORTRAN", "Computer design"};
    char **p;
    for (p = name; p<name+5; p++)
        cout<<*p<<endl; //打印字符串name[i]
}
```

北京大学

# 程序填空

■ 输入 i, j; 输出a[i][j];

```
main( )
{
   int a[3][4]={1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23};
   int **p,  i,  j; //错误
   p = a ;
   cin>>i>>j;
   cout<<setw(4)<<p[i][j];
}
```

北京大学

# 程序填空

■ 输入 i, j; 输出a[i][j];

```
main( )
{
   int a[3][4]={1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23};
   int (*p)[4],  i,  j;
   p = a ;
   cin>>i>>j;
   cout<<setw(4)<<p[i][j];
}
```

北京大学

好好想想，有没有问题?

谢　谢　！

北京大学