《计算概论A》课程 程序设计部分

指针（3）

李　戈

北京大学 信息科学技术学院 软件研究所

2010年12月17日

北京大学

利用指针变量
引用多维数组中的元素

分析程序（1）

```
#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
```

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

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
}
```

问题分析

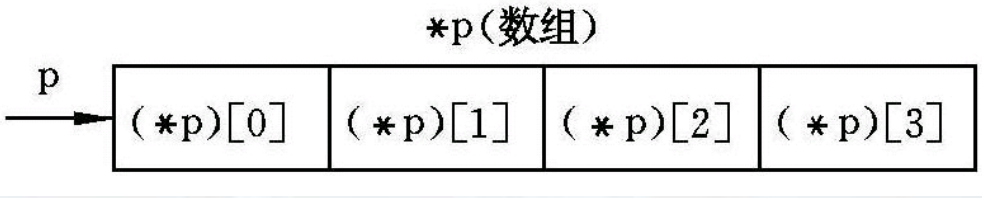
- 从 p = a 开始
  - a 是a[3][4]的“第一个元素的地址”；
  - 所谓“第一个元素”是指一个“包含4个int型元素的一维数组”；
  - 所以，a是一个“包含4个int型元素的一维数组”的地址;
  - 因此，p的基类型应该是:

    “包含4个int型元素的一维数组”

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);
}
```

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]

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 = _____;
   cin>>i>>j;
   cout<<setw(4)<<p[i][j];
}
```

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];
}
```

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**”

指针数组的用途举例

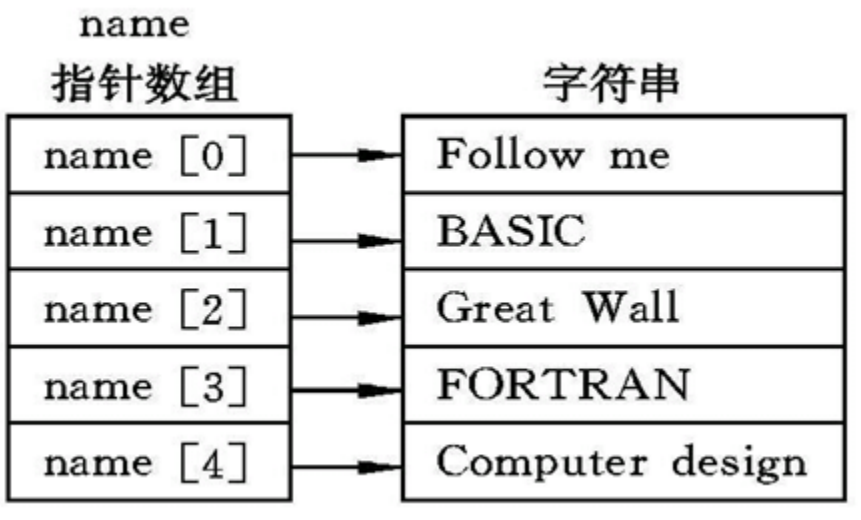
- 背景:
  - 图书馆有若干本书，每本书都有一个名字，它可以用字符串描述;
- 目标:
  - 要对所有的书名按字母排序
- 解决方案:
  - 把所有的书名“读入程序”，然后依次检查书名的第1、2、3…个字符的ASCII码，利用ASCII码排列大小。
- 问题:
  - 如何把所有的书名“读入程序”中呢？
  - 进一步的问题：读入程序中，需要有一个结构存储它们，这个结构是什么呢？

指针数组的用途举例（续）

- 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；

指针数组的用途举例（续）

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（续）

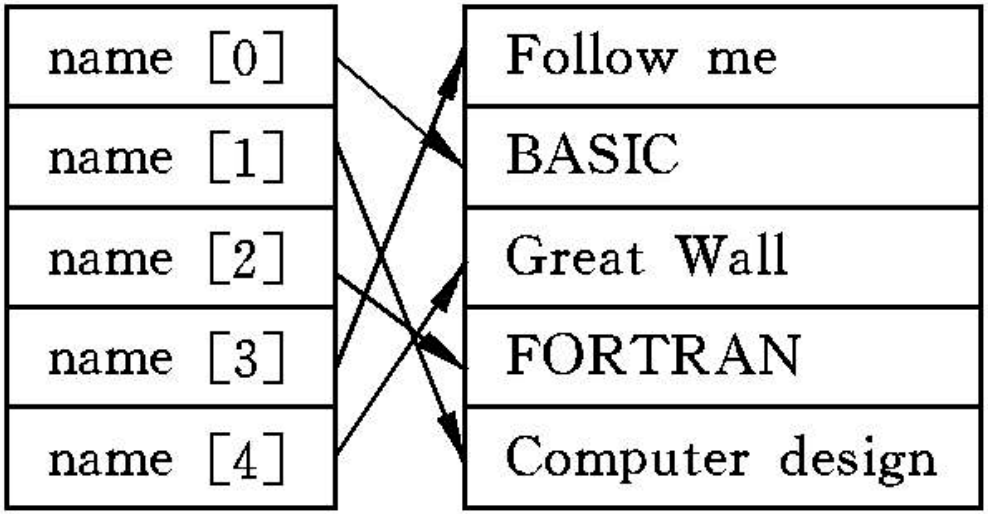
- 指针数组的定义
  - **char *name[ ] =**

    {"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;

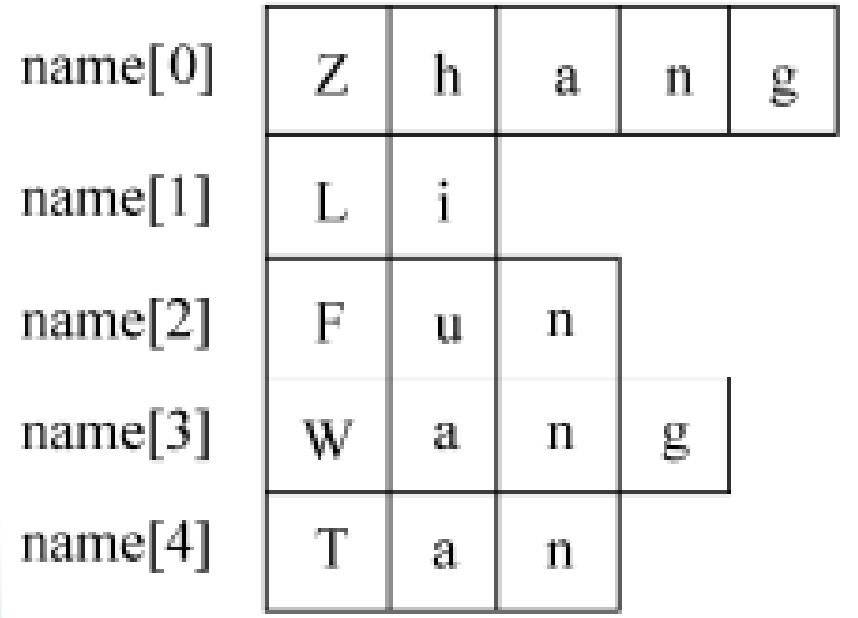
指针数组的用途举例（续）

```
#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:



指向指针的指针

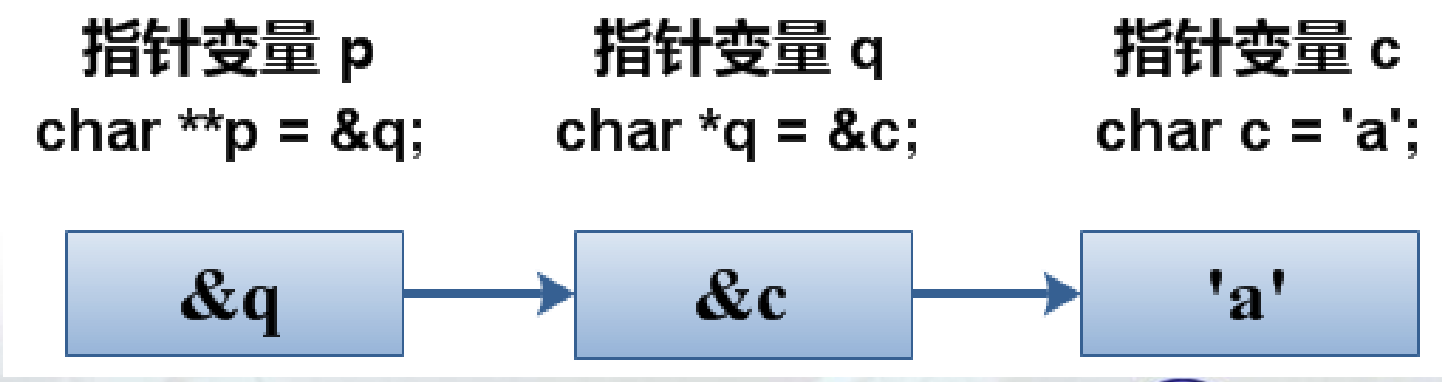
北京大学

指向指针的指针

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

  定义一个:

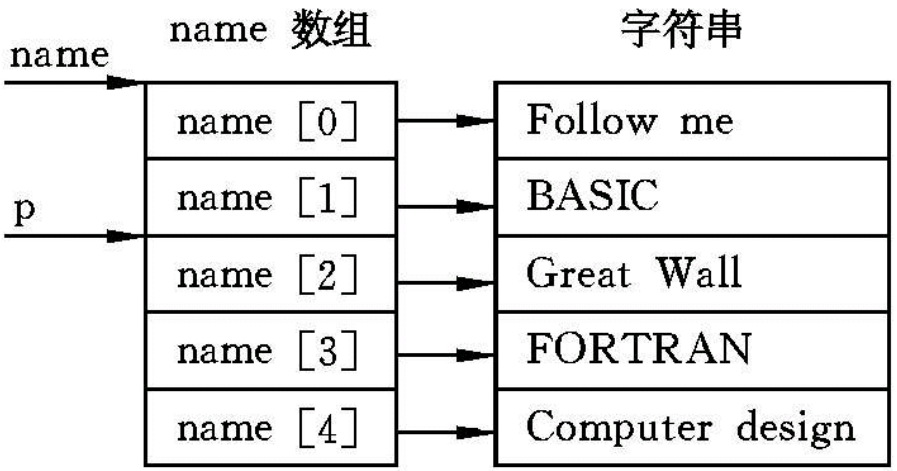
  指向“指向char型数据的指针变量”的指针变量



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]
}
```

指针与函数

北京大学

关于指针与函数

- 指针与函数参数
- 指针与函数返回值
- 指针与函数名

指针变量作为函数参数

- 输入A、B、C 3个整数，按由大到小顺序输出。

```
#include<stdio.h>
void main()
{
    int a,b,c,*p1,*p2,*p3;
    cin>>a>>b>>c;
    p1=&a; p2=&b; p3=&c;
    Rank(p1, p2, p3);
    cout<<a<<b<<c<<endl;
}
```

```
void Swap(int *pt1, int *pt2)
{   int temp;
    temp = *pt1;
    *pt1 = *pt2;
    *pt2 = temp;
}
void Rank(int *q1, int *q2, int *q3)
{   if(*q1<*q2)  Swap(q1,q2);
    if(*q1<*q3)  Swap(q1,q3);
    if(*q2<*q3)  Swap(q2,q3);
 }
```

多维数组名做函数参数

例：有一个3×4的矩阵，求所有元素中的最大值。

```
max-value(int (*array)[4])
{
    int max = array[0][0];
    for(int i=0; i<3; i++)
        for(int j=0; j<4; j++)
            if(array[i][j]>max)
                max = array[i][j];
            return max;
}
main( )
{
    int a[3][4] = {{1,3,5,7},{9,11,13,15},{2,4,6,8}};
    cout<<"The Max value is "<<max-value(a);
}
```

数组名实参 vs. 指针形参

- 可否将数组作为实参附给指针型形参？可以！

```
#include<iostream.h>
int main()
{
    int a[10] =
        {1,2,3,4,5,6,7,8,9,10};
    sum(a,10);
    return 0;
}
```

```
void sum(int *p, int n)
{
    int total = 0;
    for(int i=0;i<n;i++)
    {
        total += *p++;
    }
    cout<<total<<endl;
}
```

数组名实参 vs. 数组名形参

- 数组名可否用作形参以接收指针实参？可以！

```
#include<iostream.h>
int main()
{
    int a [10] =
          {1,2,3,4,5,6,7,8,9,10};
    int *p = a;
    sum(p,10);
    return 0;
}
```

```
void sum(int array[], int n)
```

等价于sum(int *array, int n)

```
{   int total = 0;
    for(int i=0;i<n;i++)
    {
          total += *array++;
    }
    cout<<total<<endl; }
```

C++编译都将形参数组名作为指针变量来处理！

值传递 vs. 地址传递

- 可否实现由小到大的排序？

```
#include<iostream>
using namespace std;
void main()
{
    int a[2] = {12, 5};
    rank(a[0], a[1]);
    cout<<"min to max: "
    <<a[0]<<", "<<a[1]<<endl;
}
```

```
void rank(int a, int b)
{
    int temp;
    if(a > b)
    {
        temp = a;
        a = b;
        b = temp;
    }
}
```

值传递 vs. 地址传递

- 如何实现由小到大的排序？

```
#include<iostream>
using namespace std;
int a[2] = {12, 5};
void main( )
{
    rank( );
    cout<<"min to max: "
    <<a[0]<<", "<<a[1]<<endl;
}
```

```
void rank( )
{
    int temp;
    if(a[0] > a[1])
    {
        temp = a[0];
        a[0] = a[1];
        a[1] = temp;
    }
}
```

值传递 vs. 地址传递

- 如何实现由小到大的排序？

```
#include<iostream>
using namespace std;
void main( )
{
    int a[2] = {12,5};
    rank(&a[0], &a[1]);
    cout<<"min to max:
    "<<a[0]<<", "<<a[1]<<endl;
}
```

```
void rank(int *a, int *b)
{
    int temp = 0;
    if(*a > *b)
    {
        temp = *a;
        *a = *b;
        *b = temp;
    }
}
```

值传递 vs. 地址传递

- 还能实现由小到大的排序？

```
#include<iostream.h>
    int a[2] = {12,5};
void main()
{
    rank(&a[0], &a[1]);
    cout<<"min to max:
    "<<a[0]<<", "<<a[1]<<endl;
}
```

```
void rank(int *a, int *b)
{
    int *temp = NULL;
    if(*a > *b)
    {
        temp = a;
        a = b;
        b = temp;
    }
}
```

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;

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"

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;
}
```

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″

好好想想，有没有问题?

谢　谢　！

北京大学