第5章

数组、指针和引用

人们经常需要使用大量集中在一起的数据来工作，C++通过数组处理来满足这种要求。数组是一组具有相同数据类型的数据的有序集合，可以是一维的，也可以是多维的，许多重要的应用都是基于数组的。

C++具有在程序运行时获得变量地址和操纵地址的能力，这种用来操纵地址的特殊类型就是指针。指针可以用于数组，可以用做函数参数，也可以用于内存访问和堆内存操作。指针的功能最强，但也最危险。

使用引用可以确定把参数传递给函数的方法。通过引用的学习，要求正确掌握引用的语法，明确引用与指针的区别。

5.1 数　　组

数组是一组具有相同数据类型的数据的有序集合。数组是一个存储单元，元素的值就存放在其中。数组的数据类型也就是它的元素的数据类型，如整型数组，其中的每个元素都存储一个整数。数组的下标用于标识数组元素的位置。根据元素排列方式的不同，数组可分为一维数组、二维数组等。不同维数的数组，其下标访问方式也不同。通过数组的学习，要求理解数组下标含义，掌握数组初始化的方法，学会把数组作为函数的参数，学会二维数组的使用。

5.1.1 一维数组

案例引入 冒泡法排序。

从键盘输入10个整数到一个一维数组中，使用冒泡法对其进行从小到大排序，输出排序结果。要求将每趟排序的结果输出。

源代码展示

实现冒泡法排序算法的代码如下所示。

```
#include <iostream.h>
int main(int argc,char * argv[])
{
    int a[10];
    int i,j,k,temp;
      cout<<"请输入 10 个整数:\n";
    for(i=0;i<10;i++)                    //输入 10 个整数
        cin>>a[i];
    for(i=0;i<9;i++)                     //使用冒泡算法从小到大排序
    {
        for(j=0;j<9-i;j++)
            if(a[j]>a[j+1])
            {
                temp=a[j];
                a[j]=a[j+1];
                a[j+1]=temp;
            }
        cout<<"第"<<i+1<<"趟排序结果:";
        for(k=0;k<=9;k++)                //输出每趟排序结果
            cout<<a[k]<<" ";
        cout<<endl;
    }
    cout<<"排序后为:\n";
    for(i=0;i<=9;i++)                    //输出排序完成后结果
        cout<<a[i]<<" ";
    cout<<endl;
    return 0;
}
```

运行结果

执行程序,输出每趟排序和全部排序后的结果,如图 5-1 所示。

从图 5-1 可以看出,输入 10 个整数,进行了 9 趟排序后,输出了排序结果。

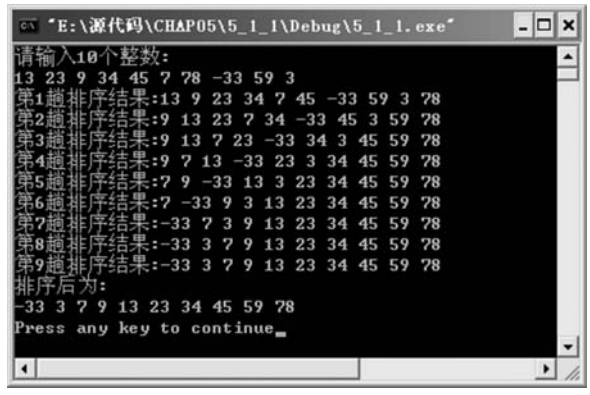

图 5-1　一维数组示例程序运行结果

程序分析

在程序中，一维数组 a 被分配了 10 个整型元素的内存空间，并通过键盘向该数组中输入了 10 个整数，然后使用冒泡算法进行从小到大排序，最后输出排序后的结果。

从程序运行的结果可以看出，每趟排序后都将一个最大的数放到数组的后面。冒泡算法的基本思想如下。

(1)在 a[0]～a[9]中，依次比较两个相邻元素的值，若 a[j]＞a[j＋1]，则交换 a[j]与 a[j＋1]，j 的值为 0，1，2，…，8。经过这样一趟冒泡，将这 10 个数中最大的数放在了 a[9]中。第 1 趟排序各次比较结果如图 5-2 所示。

13　23　9　34　45　7　78　−33　59　3　　第1次比较13

13　23　9　34　45　7　78　−33　59　3　　第2次比较23

13　9　23　34　45　7　78　−33　59　3　　第3次比较23

13　9　23　34　45　7　78　−33　59　3　　第4次比较34

13　9　23　34　45　7　78　−33　59　3　　第5次比较45

13　9　23　34　7　45　78　−33　59　3　　第6次比较45

13　9　23　34　7　45　78　−33　59　3　　第7次比较78

13　9　23　34　7　45　−33　78　59　3　　第8次比较78

13　9　23　34　7　45　−33　59　78　3　　第9次比较78

13　9　23　34　7　45　−33　59　3　78　　第1趟比较结束

图 5-2　第 1 趟各次比较情况

(2)第 1 趟比较结束后，将 10 个数中的最大数交换到 a[9]，然后对 a[0]～a[8]再进行一趟冒泡，又将该范围内的最大值换到了 a[8]中。

(3)依次进行下去，最多只进行 9 趟冒泡，就可完成排序。

(4)如果在某趟冒泡过程中没有交换相邻两个数的值，则说明排序已完成，可以提前结束排序。

可以推知，如果有 n 个数排序，则要进行 N－1 趟比较。在第 1 趟比较中要进行 N－1 次两两比较，在第 i 趟比较中要进行 N－i 次两两比较。综上所述，用 N－S 图描述的 N 个整数冒泡排序算法如图 5-3 所示。

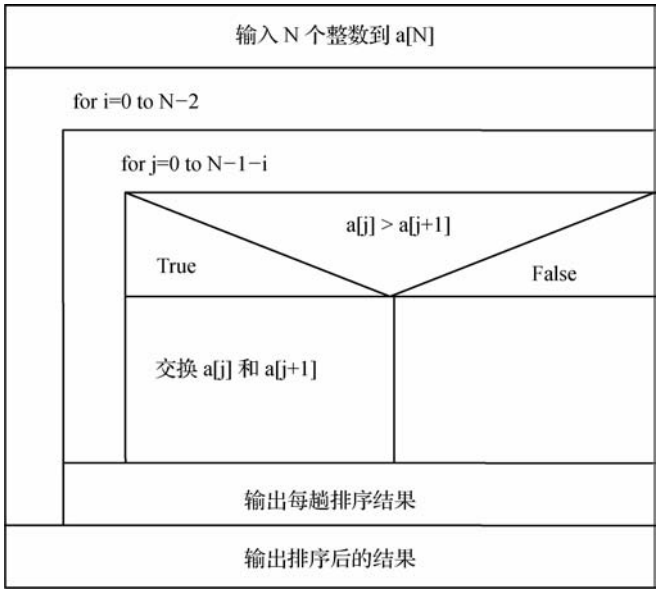

图 5-3　冒泡排序算法 N-S 图

知识讲解

数组是一组具有相同数据类型的数据的有序集合。其中，一维数组中的元素是线性排列的，用下标 0，1，2…标识它的每个元素。如果一维数组有 n 个元素，则其下标为 0～(n－1)。

1）一维数组的定义

一维数组的定义格式如下。

```
类型说明符 数组名[常量表达式];
```

例如：

```
int a[10];
```

表示定义了一个整型数组，数组名为 a，有 10 个元素。但该数组未进行初始化。

2）一维数组的初始化

对数组元素的初始化可以在定义数组时进行，例如：

```
int a[10]={0,1,2,3,4,5,6,7,8,9};
```

即将数组元素的初值依次放在一对花括号内。

经过上面的定义和初始化后，a[0]＝0，a[1]＝1，a[2]＝2……依次类推。

当然，也可以只给一部分元素赋值，例如：

```
int a[10]={0,1,2,3,4};
```

花括号内只提供了 5 个初始值，此时只给数组的前 5 个元素赋初始值，后 5 个元素的值为 0。

由于在对全部数组元素赋初始值时，数据的个数已经确定，因此，可以不用指定数组的长度。例如：

```
int a[5]={0,1,2,3,4};
```

也可以写成：

```
int a[]={0,1,2,3,4};
```

5.1.2　二维数组

案例引入　**实现矩阵的转置。**

将用二维数组表示的矩阵的行和列互换，如下面一个 2×3 的矩阵：

$$\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\end{bmatrix}$$

转置后变成 3×2 的矩阵：

$$\begin{bmatrix}1 & 4\\2 & 5\\3 & 6\end{bmatrix}$$

源代码展示

实现矩阵转置的代码如下所示。

```
#include <iostream.h>
int main(int argc,char * argv[])
{
    int a[2][3];
    int b[3][2];
    int i,j;
    cout<<"请输入 2×3 矩阵的值:\n";
    for(i=0;i<2;i++)                    //输入 2×3 矩阵的值
        for(j=0;j<3;j++)
            cin>>a[i][j];
    for(i=0;i<2;i++)                    //对 2×3 矩阵进行转置
        for(j=0;j<3;j++)
            b[j][i]=a[i][j];
    cout<<"转置后的矩阵为:\n";
    for(i=0;i<3;i++)                    //输出转置后的矩阵
    {
```

```
            for(j=0;j<2;j++)
                cout<<b[i][j]<<" ";
            cout<<endl;
        }
        return 0;
    }
```

运行结果

程序运行结果如图 5-4 所示。

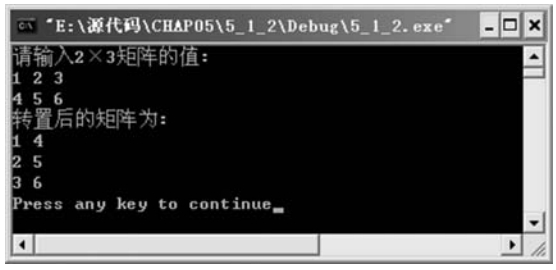

图 5-4　二维数组示例程序运行结果

从图 5-4 可以看出，输入 2×3 的矩阵，转置后输出 3×2 的矩阵。

程序分析

在程序中，首先声明了一个 2×3 的二维数组 a 和一个 3×2 的二维数组 b，并通过键盘向二维数组 a 中输入 6 个值；然后将数组 a 进行转置，转置后的元素存放到数组 b 中，最后将数组 b 输出。

知识讲解

二维数组通常用来表示按行和列格式存放信息的数值表。要访问表中的某个元素，需要指定两个下标：第一个下标表示该元素所在的行，第二个下标表示该元素所在的列。

1）二维数组的定义

二维数组的定义格式如下。

```
类型说明符 数组名[常量表达式 1][常量表达式 2];
```

其中，常量表达式 1 表示二维数组的行数，常量表达式 2 表示二维数组的列数。

例如：

```
int a[2][3];
```

表示定义了一个二维数组，数组名为 a，有 6 个元素。它在逻辑上的空间形式为 2 行 3 列，每

个元素的数据类型为整型。数组 a 的各个元素如下所示。

a[0][0]　a[0][1]　a[0][2]

a[1][0]　a[1][1]　a[1][2]

2)二维数组的初始化

可以按照以下 4 种方式给二维数组初始化。

(1)分行给二维数组赋初值。例如：

```
int a[2][3]={{1,2,3},{4,5,6}};
```

(2)可以将所有数据写在一个花括号内，按数组排列的顺序对各元素赋初值。例如：

```
int a[2][3]={1,2,3,4,5,6};
```

(3)对部分元素赋初值。例如：

```
int a[2][3]={{1},{2}};
```

此时，二维数组 a 中的元素为：

1 0 0

2 0 0

也可以对各行中的某一元素赋初值。例如：

```
int a[2][3]={{1},{0,2}};
```

此时，二维数组 a 中的元素为：

1 0 0

0 2 0

还可以只对某几行元素赋初值。例如：

```
int a[2][3]={{1}};
```

此时，二维数组 a 中的元素为：

1 0 0

0 0 0

(4)如果对全部元素都赋初值，则定义数组时对第一维的长度可以不指定，但第二维的长度不能省略。例如：

```
int a[2][3]={1,2,3,4,5,6};
```

等价于：

```
int a[][3]={1,2,3,4,5,6};
```

可以只对部分元素赋初值而省略第一维的长度，但应分行赋初值。

例如：

```
int a[][3]={{1},{4,5,6}};
```

此时，二维数组 a 中的元素为：

1 0 0

4 5 6

5.1.3 数组名作函数参数

案例引入 **用选择法对数组中的整数进行排序。**

从键盘输入 10 个整数到一个数组中，用选择法对这 10 个整数进行排序，并按由小到大的顺序输出。要求输出每趟排序的结果。

源代码展示

实现选择法排序的代码如下所示。

```
#include "stdafx.h"
#include <iostream.h>
int main(int argc,char * argv[])
{
    void select_sort(int arr[],int n);          //声明函数
    int a[10],i;
    cout<<"请输入 10 个整数"<<endl;
    //从键盘输入 10 个整数到数组 a 中
    for(i=0;i<10;i++)
        cin>>a[i];
    cout<<endl;
    select_sort(a,10);                          //调用函数，实参是数组 a
    cout<<"选择法排序后结果为:"<<endl;
    //输出排序后结果
    for(i=0;i<10;i++)
        cout<<a[i]<<" ";
    cout<<endl;
    return 0;
}
//选择排序函数，形参 arr 是数组名
void select_sort(int arr[],int n)
{
    int i,j,k,m,temp;
    for(i=0;i<n-1;i++)        //进行 n-1 趟排序
```

```
    {
        k=i;                          //将余下的第一个数的下标用变量 k 保存
    //每趟排序,将余下数中最小的数与余下的第一个数交换
    for(j=i+1;j<n;j++)
        if(arr[j]<arr[k])
            k=j;
        temp=arr[k];
        arr[k]=arr[i];
        arr[i]=temp;
        //输出每趟排序结果
        cout<<"第"<<i+1<<"趟排序结果:";
        for(m=0;m<10;m++)
            cout<<arr[m]<<" ";
        cout<<endl;
    }
}
```

运行结果

程序运行结果如图 5-5 所示。

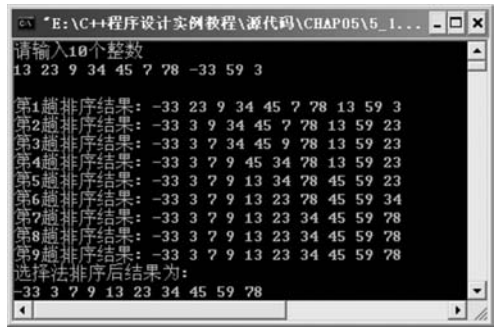

图 5-5　数组名作为函数参数示例程序运行结果

从图 5-5 可以看出,输入 10 个整数,使用选择法进行了 9 趟排序后,输出了排序结果。

程序分析

在程序中定义了函数 select_sort(int arr[],int n),形参 arr 是数组名,函数使用选择法进行排序。第一趟排序将 10 个数中最小的数与 a[0]进行交换(即－33 与 13 交换),第二趟排序将余下的 9 个数中最小的数与 a[1]交换(即 3 与 23 交换),每趟排序都从余下的数中找出最小的,并与余下的第一个数进行交换。

在调用函数 select_sort(a,10)前后,数组 a 中元素的位置发生了变化。在调用函数前,数组 a(实参)中的元素是无序的;调用函数后,形参数组 arr 已用选择法进行了排序,形参数

组 arr 的改变也使实参数组 a 随之改变。

知识讲解

当用数组名作函数参数时，由于数组名代表数组首元素的地址，所以在调用函数时是将实参数组首元素的地址传递给形参数组名，这样实参数组和形参数组便占用同一段内存地址单元。假如，实参数组 a 的起始地址为 2000，将 a 传递给形参数组 b 时，b 的起始地址也是 2000。此时，a[0]与 b[0]表示同一个单元，a[1]与 b[1]也表示同一个单元，依次类推。因此，当形参数组 b 中的元素值发生变化时，实参数组 a 中的元素值也随着改变。

这与用变量作函数参数不同，用变量作函数参数时，在调用函数时只是将实参变量的值传递给形参变量，而没有将地址传递给形参变量，所以即使形参的值发生变化，也不会影响实参变量的值。

5.1.4 字符数组

案例引入 **输出 3 个字符串中的最大者。**

从键盘输入 3 个长度不超过 10 的字符串，用 3 个字符数组存放，然后进行判断，将最大的字符串输出。

源代码展示

实现输出最大字符串的代码如下所示。

```
#include <iostream.h>
#include <string.h>
int main(int argc,char * argv[])
{
    char maxStr[10];        //定义字符数组 maxStr,保存最大字符串
    char str[3][10];        //定义二维字符数组
    int i;
    cout<<"输入 3 个字符串:"<<endl;
    for(i=0;i<3;i++)  //循环输入 3 个字符串,存入字符串数组 str 中
        cin>>str[i];
    //将前两个字符串比较,大者存入字符数组 maxStr 中
    if(strcmp(str[0],str[1])>0)
    {
        strcpy(maxStr,str[0]);
    }
    else
    {
        strcpy(maxStr,str[1]);
```

```
    }
//将最后一个字符串和 maxStr 中的字符串比较
    if(strcmp(str[2],maxStr)>0)
    {
//如果最后一个字符串大,则将其复制到 maxStr 中
        strcpy(maxStr,str[2]);
    }
    cout<<"最大的字符串为:"<<maxStr<<endl;
    return 0;
}
```

运行结果

程序运行结果如图 5-6 所示。

图 5-6　字符数组示例程序运行结果

从图 5-6 可以看出,输入 3 个字符串,进行比较后,将最大的字符串输出。

程序分析

在程序中,首先声明了一个字符数组 maxStr,用于保存最大字符串;接着声明了一个二维字符数组,即存放字符串的数组 str,用于接收从键盘输入的 3 个字符串。在进行比较时,先将前两个字符串比较,将较大的字符串暂时存入字符数组 maxStr 中;再将最后一个字符串和 maxStr 中的字符串比较,如果最后一个字符串大,则将其复制到 maxStr 中。程序结束后,maxStr 中保存的字符串就是最大的字符串。

知识讲解

1)字符数组的定义与初始化

当数组中的元素全都由字符组成时,该数组称为字符数组。在C++中,用一个一维的字符数组表示字符串。数组的每个元素保存字符串的一个字符,在字符串结尾附加一个空字符,用\0'表示。因此,如果一个字符串有 n 个字符,则用于保存该字符串的数组长度应为 n+1。下面的代码声明了一个长度为 10 的字符数组 maxStr:

```
char maxStr[10];
```

则该数组能够保存的字符数最多为 9。

字符数组在初始化时，不需要逐个字符地赋值，可以用一个字符串对字符数组进行初始化。例如：

```
maxStr[10]="Hello";
```

此时，内存中字符数组 maxStr 的内容如下所示。

H	e	l	l	o	\0				

如果初值个数小于数组长度，则只将这些字符赋给数组中前面那些元素，其余的元素自动设置为空字符。

注意：字符串的长度不包括结尾符。

2)字符串处理函数

常用的字符串处理函数包括以下几个。

(1)计算字符串长度的函数 strlen()，其语法格式如下。

```
strlen(字符数组)
```

功能：计算字符串的长度，即字符个数，不包含字符串结尾标记'\0'。

例如：

```
char s1[10]="Hello";
strlen(s1);
```

字符数组的长度为 5。

(2)字符串复制函数 strcpy()，其语法格式如下。

```
strcpy(字符数组 1,字符数组 2)
```

功能：将字符数组 2 中的字符串复制到字符数组 1 中。字符串结束标记'\0'也一并复制。

例如：

```
char s1[10],s2[10]="Hello";
strcpy(s1,s2);
```

字符数组 s1 中的内容为"Hello"(包括结束标记'\0')。

注意：在使用 strcpy()函数时，要求字符数组 1 的长度不能小于字符数组 2 的长度。

(3)字符串连接函数 strcat()，其语法格式如下。

```
strcat(字符数组 1,字符数组 2)
```

功能：将字符数组 2 中的字符串连接到字符数组 1 后面，结果存放在字符数组 1 中。

例如：

```
char s1[20]="Hello ";
char s2[10]="World";
```

```
strcat(s1,s2);
```

连接后的新字符串为"Hello World"。

注意：在使用 strcat()函数时，要求字符数组 1 长度足够大，能够存放连接后的字符串。另外，连接后，第一个字符串后的\0'没有了，只在新字符串的最后保留\0'。

(4)字符串比较函数 strcmp()，其语法格式如下。

```
strcmp(字符串 1,字符串 2)
```

功能：将字符串 1 和字符串 2 中的字符从左到右逐个按照其 ASCII 码值大小进行比较，直到出现不相同的字符或者遇到\0'。如果字符串 1 大于字符串 2，返回一个正整数；如果字符串 1 小于字符串 2，返回一个负整数；否则返回 0。

5.2 指　　针

指针是 C 和C++中的一个重要概念，正确、灵活地使用它，可以使程序更加简洁、高效。通过指针的学习，要求能够理解指针的含义，并正确使用指针。

5.2.1 指针的使用

案例引入 输入两个整数，并将它们按从大到小的顺序输出。

使用指针变量实现：从键盘输入两个数给整型变量 a 和 b，并将它们按先大后小的顺序输出。

源代码展示

实现该案例的代码如下所示。

```
#include <iostream.h>
#include <string.h>
int main(int argc,char * argv[])
{
    cout<<"输入两个整数:"<<endl;
    int a,b;                    //定义整型变量
    int *p1,*p2,*p;
    cin>>a>>b;                  //输入两个整数
    p1=&a;                      //p1 指向 a
    p2=&b;                      //p2 指向 b
    cout<<"*p1 的值为"<<*p1<<" *p2 的值为"<<*p2<<endl;
                                //输出 *p1 和 *p2 的值
```

```
    //如果 a 小于 b,则交换 p1 和 p2 的值
    if(a<b)
    {
        p=p1;
        p1=p2;
        p2=p;
    }
    cout<<"最大数为:"<<*p1<<" 最小数为:"<<*p2<<endl;
    return 0;
}
```

运行结果

程序运行结果如图 5-7 所示。

图 5-7　指针示例程序运行结果

从图 5-7 可以看出,输入两个整数 40 和 60,通过指针变量将这两个整数进行交换,按先大后小的顺序输出。

程序分析

程序开始定义了 3 个指针变量 p1、p2 和 p,然后让 p1 指向变量 a,让 p2 指向变量 b。如果 a 小于 b,将 p1 的值和 p2 的值交换。交换前后内存情况分别如图 5-8 和图 5-9 所示。

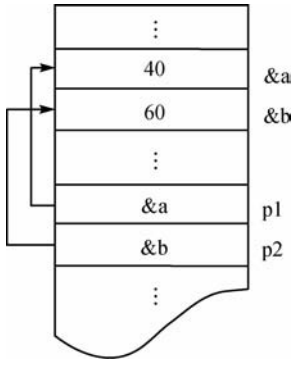

图 5-8　交换前内存情况

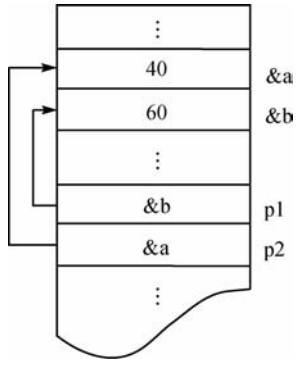

图 5-9　交换后内存情况

在交换过程中，不是交换整型变量 a 和 b 的值，而是交换指针变量 p1 和 p2 的值，变量 a 和变量 b 的内容并没有交换。变量 p1 和 p2 的值变了，p1 的值交换前是 &a(即 a 的地址)，交换后变成了 &b(即 b 的地址)；p2 的值则由 &b 变成了 &a。因此，交换后输出 * p1 和 * p2 时，最终输出的是变量 b 和 a 的值，即 60 和 40。

知识讲解

1)指针的概念

对于程序中定义的一个变量，在编译时系统会根据该变量的数据类型给它分配一个内存单元，分配一定长度的空间。如 Visual C++ 6.0 为整型变量分配 4 字节的存储空间，为字符型变量分配 1 字节存储空间。内存的每个字节单元都有一个编号，称为“地址”。例如，下面的代码定义了两个整型变量 a 和 b，并进行了初始化：

```
int a=2,b=3;
```

编译时，系统为变量 a 分配了 1000、1001、1002 和 1003 这 4 个字节单元，为变量 b 分配了 1004、1005、1006 和 1007 这 4 个字节单元，如图 5-10 所示。

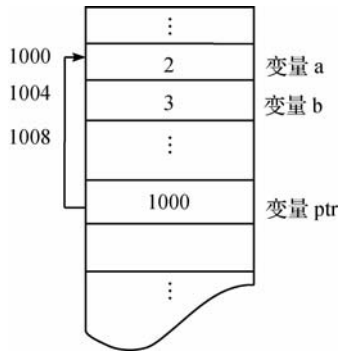

图 5-10　变量 a 和 b 的内存空间

在程序中，一般通过变量名对内存单元进行存取操作。程序编译后将变量名转换为变量的地址，对变量值的存取都是通过地址进行的。这种按变量地址存取变量值的方式称为直接存取方式，或直接访问方式。

还可以采用另一种方式存取数据，即间接访问方式，方法是将变量 a 的地址存放在另一个变量中。这个用于存放地址的变量是一个特殊的变量。例如，定义一个保存整型变量 a 的地址的变量 ptr：

```
ptr=&a;
```

其中，“&”是取地址运算符，&a 是 a 的地址。语句执行后，变量 ptr 的值就是整型变量 a 所占用的内存单元的起始地址 1000。

采用间接方式取变量 a 的值时，先要找到变量 ptr，从中取出值(变量 a 的地址)1000，然后从 1000 开始的 4 个字节单元中取出变量 a 的值(即 2)。这个过程的示意图见图 5-10。

2)指针变量

一个变量的地址称为该变量的指针,如整型变量 a 的首地址是 1000,因此,1000 就是整型变量 a 的指针。存放变量地址的变量就是指针变量,如上述的变量 ptr 就是一个指针变量,它存放了变量 a 的地址 1000,指向了整型变量 a,图 5-10 中的单箭头表示了这种指向关系。在C++中,用“*”表示指向,如 ptr 是一个指针变量,而 *ptr 表示 ptr 所指向的变量,因此,下面两个语句的作用相同:

```
a=2;
*ptr=2;
```

后一条语句的含义是将整数 2 赋值给指针变量 ptr 所指向的变量(也就是 a)。

(1)定义指针变量。定义指针变量的语法如下。

```
数据类型 *指针变量名;
```

例如:

```
int *ptr;
```

开头的 int 表示所定义的 ptr 是指向整型数据的指针变量。

(2)指针的赋值。将被指向的变量的地址赋值给指针变量即可,例如:

```
ptr=&a;              //将整型变量 a 的地址存放到指针变量 ptr 中
```

(3)引用指针变量。引用指针变量用到下面两个与指针变量相关的运算符。

- &:取地址。
- *:指针运算符。

例如,&a 表示变量 a 的地址,*ptr 表示指针变量 ptr 所指向的变量。

5.2.2 指针的运算

案例引入 **指针的关系运算。**

定义两个整型变量,分别用两个指针指向它们,然后进行指针的关系运算。

源代码展示

实现该案例的代码如下所示。

```
#include <iostream.h>
int main(int argc,char * argv[])
{
    int a=10,b=10,*p1,*p2;
    p1=&a;
    p2=&b;
```

```
    if((p1!=p2))
        cout<<"指针 p1 和 p2 指向不同的内存地址"<<endl;
    if(*p1==*p2)
        cout<<"指针 p1 和 P2 指向的内存地址单元的值相同"<<endl;
    return 0;
}
```

运行结果

程序运行结果如图 5-11 所示。

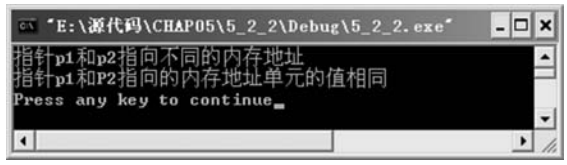

图 5-11　指针关系运算示例程序运行结果

从图 5-11 可以看出，程序输出了指针变量 p1 和 p2 以及它们所指向变量的值进行关系运算的结果。

程序分析

程序用指针 p1 和 p2 分别指向整型变量 a 和 b 的地址，这两个指针指向的内存地址不同，但它们指向的内存地址单元的内容相同(即变量 a 和变量 b 的值)相同，都是 10。

知识讲解

1)指针的关系运算

两个指针进行关系运算时，必须指向同样的数据类型，在指向不同数据类型的指针之间进行关系运算是没有意义的。指针和整型变量之间进行关系运算也是没有意义的，如将指针 p1 和变量 a 进行关系运算是没有意义的。但是指针和整数 0 之间可以进行关系运算。例如，判断指针是否为零指针的关系运算为：

```
p1==0;
```

或者

```
p1!=0;
```

2)指针的算术运算

指针的算术运算包括加和减两种。

(1)指针和整数进行加、减运算，例如：

```
p1+n;   // p1 为指针，n 为整数
```

(2)指针加 1 和减 1,例如:

```
p1++;   // p1 为指针,进行自增操作
p1--;   // p1 为指针,进行自减操作
```

(3)指针相减,例如:

```
p1-p2;   // p1、p2 均为指针
```

5.3 指针与数组

数组是内存中一块连续的空间,存储一组相同类型的数据,程序可通过下标访问数组中的元素,下标表示元素在数组中的位置。如果将数组中第一个元素(下标为 0)的地址保存在某个指针变量中,则对数组元素的访问就转变为指针操作。

5.3.1 指向数组元素的指针

案例引入 使用指针访问数组元素。

假设一个整型数组 a 中有 10 个元素,要求使用指针访问数组 a 的元素。

源代码展示

实现该案例的代码如下所示。

```
#include <iostream.h>
#include <string.h>
int main(int argc,char * argv[])
{
    int i,a[10];
    int *p;
    cout<<"输入 10 个整数:"<<endl;
    for(i=0;i<10;i++)
    {
        cin>>a[i];              //输入 10 个整数
    }
    p=a;   //将数组 a 的首元素的地址赋给指针变量 p,使 p 指向数组 a 的首元素 a[0]
    cout<<"使用指针变量输出数组 a 中元素:"<<endl;
    for(i=0;i<10;i++)
    {
        cout<<*p<<" ";      //通过指针变量输出这 10 个整数
```

```
        p++;                    //p 的先后次序为 a[0]~a[9]
    }
    cout<<endl;
    return 0;
}
```

运行结果

程序运行结果如图 5-12 所示。

图 5-12　指向数组元素的指针示例程序运行结果

从图 5-12 可以看出，输入 10 个整数到数组 a 中，再通过指针变量将数组 a 中的元素输出。

程序分析

数组包含若干个元素，每个元素在内存中都有相应的地址。指针变量既然可以指向变量，当然也可以指向数组元素。在程序中，定义了数组 a 和指针变量 p。先给数组 a 赋值，然后通过语句

```
p=a;
```

将数组 a 的首元素的地址赋给指针变量 p。在C++ 中，数组名代表数组中第一个元素（即下标为 0 的元素）的地址。因此，下面的语句与“p=a;”等价：

```
p=&a[0];
```

赋值后 p 指向数组 a 的首元素 a[0]，此时输出 * p 得到的是 a[0]的值，即数组 a 的第一个元素值。接着使用语句

```
p++;
```

使 p 指向下一个元素，即 a[1]，如果使用 * p，则得到下一个元素 a[1]的值。通过循环执行“p++;”，* p 便会不断输出数组 a 的下一个元素，即 * p 就是 a[0]，*（p+1）就是 a[1]……*（p+i）就是 a[i]。

知识讲解

指针变量不仅可以指向一维数组中的元素，也可以指向多维数组中的元素。以二维数

组为例，假设一个二维数组定义如下：

```
int a[2][3]={{1,2,3},{4,5,6}};
```

其中，a 是数组名。该二维数组 a 包含两行，即两个元素 a[0]和 a[1]。其中，每个元素又是一个一维数组，每个一维数组包含 3 个元素，例如，a[0]所代表的一维数组包含的 3 个元素分别为：a[0][0]、a[0][1]和 a[0][2]，如图 5-13 所示。

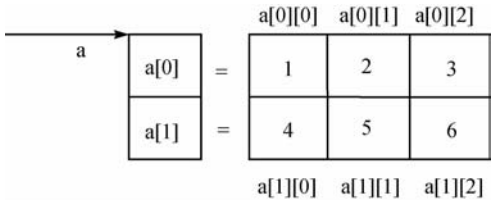

图 5-13　二维数组 a

a[0]和 a[1]是一维数组名，由于数组名代表数组中首元素地址，因此，a[0]代表一维数组 a[0]中第 0 列元素的地址，即 &a[0][0]。同理，a[1]代表一维数组 a[1]中第 0 列元素的地址，即 &a[1][0]。

下面的语句定义了一个指向整型变量的指针变量 p，并对 p 进行了赋初值。

```
int *p;                          //声明指针变量 p
p=a[0];                          //使 p 指向 a[0][0]
```

上面的语句把 a[0]赋给了 p，因为 a[0]是 a[0][0]的地址(即 &a[0][0])，所以对 p 的赋初值也可写成下面形式。

```
p=&a[0][0];                          //使 p 指向 a[0][0]
```

思考：如果写成“p=a”，结果如何？分析：a 是二维数组名，代表的是二维数组首元素的地址，二维数组首元素是 a[0]，因此 a 的值为 &a[0]，即 a 指向 a[0]。这与 p 指向的数据类型不匹配(p 指向整型数据)，所以会出现编译错误。

给 p 赋值后，使用下面的语句可以输出二维数组 a 中的元素。

```
for(i=0;i<2;i++)
    for(j=0;j<3;j++)
    {
        cout<<"a["<<i<<"]["<<j<<"]="<< *p<<endl;
        p++;
    }
```

循环开始时，p 指向 a[0][0]，此时输出 *p，也即输出的是 a[0][0]的值(即 1)；执行语句“p++;”后，p 指向下一个元素 a[0][1]，输出 *p 时，输出的是 a[0][1]的值(即 2)；通过循环执行语句“p++;”，*p 便会不断输出二维数组 a 的下一个元素，最后得到的输出结果如图 5-14 所示。

图 5-14　用指针变量输出二维数组中元素的结果

5.3.2　指针数组

案例分析　**将若干字符串按字母顺序输出。**

假设有一个字符串数组，包含 5 个元素，要求使用指针数组将其按字母顺序输出。

源代码展示

实现该案例的代码如下所示。

```
#include <iostream.h>
#include <string.h>
int main(int argc,char * argv[])
{
    char *name[5]={"C++","C#","C","VC++","Java"};  //定义指针数组
    int i,j,k;
    char *temp;
      cout<<"指针数组初始值:"<<endl;
    for(i=0;i<5;i++)                    //输出指针数组初始值
          cout<<name[i]<<" ";
    cout<<endl;
    for(i=0;i<4;i++)                    //用选择法对字符串进行排序
    {
        k=i;
        for(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;
          }
```

```
    }
    cout<<"按字母反序后输出:"<<endl;
    for(i=0;i<5;i++)                    //按字母反序后输出指针数组的值
        cout<<name[i]<<" ";
    cout<<endl;
    return 0;
}
```

运行结果

程序运行结果如图 5-15 所示。

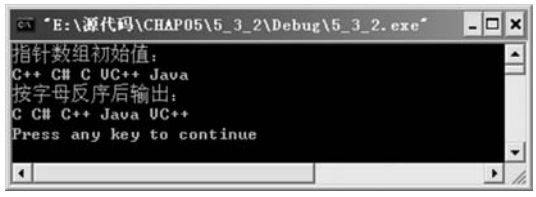

图 5-15　指针数组示例程序运行结果

从图 5-15 可以看出，程序先输出指针数组的初始值，然后按字母顺序排序，并输出排序后的结果。

程序分析

上述程序首先定义指针数组 name，它包含 5 个元素，其初始值分别为字符串"C++"、"C#"、"C"、"V C++"和"Java"的起始地址，即该指针数组包含 5 个字符串指针。通过数组的下标可以将指针数组中的所有字符串输出。然后用选择排序法对这 5 个字符串进行排序，strcmp()为字符串比较函数，name[k]和 name[j]表示第 k 个和第 j 个字符串的起始地址。如果字符串 name[k]大于 name[j]，则函数值为正，否则为负。一共需要进行 4 趟排序，每趟排序时都从当前待排序的多个字符串中查找最小字符串，并与当前待排序的第一个字符串进行交换(即将最小的字符串交换到最前面)，以此类推，直到排序结束。最后将排序结果输出。

知识讲解

指针数组表示其元素为指针的数组，它是指针的集合，它的每一个元素都是指针变量，如图 5-16 所示。

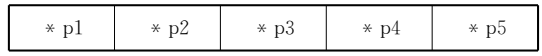

图 5-16　指针数组

C++中声明指针数组的语法格式如下。

```
数据类型 *指针数组名[常量表达式]
```

其中，数据类型是指数组中作为元素的各个指针所指向的类型，在同一个指针数组中各个指针元素指向的数据类型相同；指针数组名表示数组的首地址，是一个标识符；常量表达式表示指针数组的元素个数。

指针数组在使用前必须先赋值，也可以通过初始化赋值，例如：

```
char *name[5]={"C++","C#","C","VC++","Java",}
```

5.4 指针与函数

指针用法灵活，常常和函数一起使用。本节主要介绍指针作为函数参数、函数指针和指针函数的应用。

5.4.1 指针作为函数参数

案例引入　将大写字母转为小写。

从键盘输入一个大写字母，编写一个函数将其转换为小写字母并输出。要求转换函数中使用指针作为形参。

源代码展示

实现该案例的代码如下所示。

```
#include "stdafx.h"
#include <iostream.h>
int main(int argc,char * argv[])
{
    void lower(char *c);                    //函数声明
    char upper;
    cout<<"请输入大写字母:"<<endl;
    cin>>upper;
    lower(&upper);                          //调用函数
    cout<<"转换后字母为:"<<endl;
    cout<<upper<<endl;
    return 0;
}
void lower(char *c)
{
    if((*c>='A')&&(*c<='Z'))
        *c=*c-'A'+'a';
}
```

运行结果

程序运行结果如图 5-17 所示。

图 5-17　指针作为函数参数示例程序运行结果

从图 5-17 可以看出，输入大写字母，调用函数将其转换成小写字母后输出。

程序分析

在 lower()函数中，形参指针 c 接收从主调函数传递的实参 &upper(即大写字母的地址)，将该地址中的大写字母转换成小写字母，转换后的结果仍然保存在该地址中。

知识讲解

在函数的参数列表中，可以使用指针类型的参数。调用函数时，传递的实参可以是一个指针变量，或者一个变量的地址。在调用过程中，对形参所指向的地址内容的修改会影响实参。

当实参是一个变量的地址时，被调用函数的形参的数据类型必须与实参传递的数据类型相同。

由于数组名是一个指针，因此，数组名也可以作为函数的参数。

5.4.2　函数指针

案例引入　交换两个数。

定义一个函数指针，该指针指向实现两个整数值交换的 swap()函数，并在 main()函数中调用该函数指针，输出交换后的结果。

源代码展示

实现该案例的代码如下所示。

```
#include "stdafx.h"
#include <iostream.h>
int main(int argc,char * argv[])
{
    void swap(int *x,int *y);          //函数声明
    void(*p)(int *,int *);             //定义指向函数的指针变量 p
```

```
    p=swap;                                  //使 p 指向函数 swap()
    int a,b;
    cout<<"请输入两个整数"<<endl;
    cin>>a>>b;
    cout<<"交换前:"<<endl;
    cout<<a<<" "<<b<<endl;
    (*p)(&a,&b);                             //用函数指针变量调用函数
    cout<<"交换后:"<<endl;
    cout<<a<<" "<<b<<endl;
    return 0;
}
void swap(int *x,int *y)                     //定义函数,交换两个整数
{
    int temp=*x;
    *x=*y;
    *y=temp;
}
```

运行结果

程序运行结果如图 5-18 所示。

图 5-18　函数指针示例程序运行结果

从图 5-18 可以看出,通过函数指针变量调用 swap()函数,将两个数交换后输出。

程序分析

上述程序定义了函数指针 p,该指针指向 swap()函数。在使用函数指针 p 前为其进行了赋值,并在 main()函数中通过该指针调用 swap()函数,将两个整数进行交换。程序中语句"(*p)(&a,&b);"的作用等效于语句"swap(int *a,int *b);"。

知识讲解

在C++中允许指针指向函数,该指针保存函数的首地址。通常,定义函数指针的语法格式如下。

```
数据类型(*函数指针名)(形参表)
```

其中，数据类型表示函数指针所指向的函数的返回值类型；形参表指定该函数指针所指向函数的形参类型和个数。例如，下面的语句声明了一个函数指针 p，指向一个返回值类型为整型，且有两个整型参数的函数：

```
int(*p)(int,int);
```

注意：在定义指向函数的指针变量 p 时，(*p)两侧的括号不能省略。*p 表示 p 是指针变量，指向一个变量；(*p)(int,int)表示 p 是函数指针变量，指向一个函数。

定义函数指针 p 后，在使用前必须先给它赋值，使它指向一个函数的入口地址。由于函数名代表函数的入口地址，因此只需将函数名赋给函数指针 p 即可。赋值的语法为：

```
函数指针名=函数名;
```

5.4.3 指针函数

案例引入 从若干字符串中输出最大字符串。

定义一个包含 5 个字符串的指针数组，从这些字符串中查找最大字符串并输出。

源代码展示

实现该案例的代码如下所示。

```
#include "stdafx.h"
#include <iostream.h>
#include <string.h>
int main(int argc,char * argv[])
{
    char *max(char *x,char *y);                    //函数声明
    char *name[5]={"C++","C#","C","VC++","Java"};  //定义指针数组
    char *maxStr=name[0];
    for(int i=1;i<5;i++)
    {
        maxStr=max(maxStr,name[i]);                //调用指针函数
    }
    cout<<"最大的字符串为:"<<maxStr<<endl;
    return 0;
}
char *max(char *x,char *y)                         //返回两个字符串的大者
```

```
{
    if(strcmp(x,y)>0)
        return x;
    else
        return y;
}
```

运行结果

程序运行结果如图 5-19 所示。

图 5-19　指针函数示例程序运行结果

从图 5-19 可以看出，通过循环调用指针函数 max()，最终找到最大字符串并输出。

程序分析

上述程序定义了一个返回类型为字符型的指针函数 max()，该函数的参数是两个字符型指针。函数 max()的返回值是一个指向字符的指针，在主函数 main()中定义了一个字符型指针 maxStr 用于接收返回值。每调用一次函数 max()，可得到两个字符串中的较大者，循环 4 次后，便可从 5 个字符串中找到最大字符串。

知识讲解

除了使用 void 表示函数无返回值外，函数的返回值类型还可以是整型、字符型等。如果函数的返回值为指针类型，则该函数称为指针函数。在C++ 中定义指针函数的语法格式为：

```
数据类型 * 函数名(形参列表)
```

其中，数据类型是函数返回的指针所指向数据的类型；* 函数名代表一个指针型的函数。例如，下面的语句声明了一个指针函数：

```
int *a(int x,int y);
```

其中，a 是函数名，如果在主函数 main()中调用它，可得到一个指向整型数据的指针。

5.5 引　　用

虽然在程序中使用指针比较灵活和高效，但使用指针并不方便，如果使用不当，会造成不易发现的错误。为此，C++ 引入了引用的概念。本节主要介绍引用的定义和引用作为函

数参数的相关内容。

5.5.1 引用的定义

案例引入 引用的简单使用。

通过案例说明引用和变量之间的关系。

源代码展示

实现该案例的代码如下所示。

```
#include "stdafx.h"
#include <iostream.h>
int main(int argc,char * argv[])
{
    int a=10;
    int &b=a;           //声明 b 是 a 的引用
    cout<<"a="<<a<<",b="<<b<<endl;
    a=100;              //a 的值发生变化,b 的值也一起变化
    cout<<"a="<<a<<",b="<<b<<endl;
    b=200;              //b 的值发生变化,a 的值也一起变化
    cout<<"a="<<a<<",b="<<b<<endl;
    return 0;
}
```

运行结果

程序运行结果如图 5-20 所示。

图 5-20 引用的定义示例程序运行结果

从图 5-20 可以看出,将 b 声明为对 a 的引用后,a 或 b 的值都会随对方的值的改变而改变。

程序分析

上述程序中 a 的初始值为 10,b 是 a 的引用(即 b 是 a 的一个别名),它的值也应该是 10。当 a 的值变为 100 时,b 的值也随之变为 100。当 b 的值变为 200,a 的值也会随之变为 200。

知识讲解

定义引用就是为一个变量起一个别名，施加于别名上的操作产生的效果等同于直接施加于别名所代表的变量上。如上面的程序，给变量 a 起个别名 b，语句为：

```
int a;              //定义一个整型变量 a
int &b=a;           //声明 b 是对变量 a 的引用
```

上述语句中，“&”不是取地址运算符，而是一个引用修饰符。

注意： 引用必须在声明时就初始化，引用声明后，必须始终与其代表的变量相联系，不能再作为其他变量的引用。

在C++中，引用与指针都可用于间接访问其他对象，但它们之间存在以下区别。

(1)指针的值是某个变量的内存地址，需要开辟内存单元存放变量的地址；而引用与初始化它的变量具有相同的内存地址，不需要为其另外开辟内存单元，它们使用内存中的同一个地址单元。

(2)指针可以指向数组的地址，可能替代数组使用；引用不能替代数组，只能代表数组中的某一个元素。

5.5.2　引用作为函数参数

案例引入 实现两个数的交换。

定义一个实现两个数交换的函数，使用引用作为函数的形参。

源代码展示

实现该案例的代码如下所示。

```
#include "stdafx.h"
#include <iostream.h>
int main(int argc,char * argv[])
{
    void swap(int &,int &);          //函数声明
    int a,b;
    cout<<"请输入两个整数"<<endl;
    cin>>a>>b;
    cout<<"交换前:"<<endl;
    cout<<"a="<<a<<",b="<<b<<endl;
    swap(a,b);                       //调用函数交换两个整数
    cout<<"交换后:"<<endl;
    cout<<"a="<<a<<",b="<<b<<endl;
    return 0;
```

```
}
void swap(int &x,int &y)        //函数 swap()的形参是引用类型
{
    int temp;
    temp=x;
    x=y;
    y=temp;
}
```

运行结果

程序运行结果如图 5-21 所示。

图 5-21　引用作为函数参数示例程序运行结果

从图 5-21 可以看出,交换前变量 a 和 b 的值分别为 10 和 20,调用函数 swap()交换后,a 和 b 的值分别为 20 和 10。

程序分析

上述程序定义了一个函数 swap(),其形参 x 和 y 是整型变量的引用。在这里 &x 不是变量 x 的地址,而是指明 x 是一个引用型变量,& 是引用型变量的修饰符。在主函数 main()中调用 swap()函数时,将实参变量 a 和 b 传递给形参。a 的名字传给引用变量 x,x 就是 a 的别名;b 的名字传给引用变量 y,y 就是 b 的别名。这样 a 和 x 代表同一个变量(即共享同一地址单元),b 和 y 也代表同一个变量。在 swap()函数中,将 x 和 y 的值交换后,a 和 b 的值也一同发生了改变。

知识讲解

使用引用作为函数的形参与使用指针作为函数的形参存在以下区别。

(1)使用指针作为函数的形参时,需要为其重新分配内存单元,其内容是地址;而使用引用作为函数的形参时,由于引用不是一个独立的变量,所以它不单独占用内存单元。

(2)如果使用引用作为函数形参,则在主函数中调用该函数时,如上述程序在 main()函数中调用 swap()函数,实参不用变量的地址,直接使用变量名。

5.6 习　　题

一、选择题

1. 若有以下定义，则说法错误的是(　　)。

```
int a=100, *p=&a;
```

A. 声明变量 p，其中 * 表示 p 是一个指针变量
B. 变量 p 经初始化，获得变量 a 的地址
C. 变量 p 只可以指向一个整形变量
D. 变量 p 的值为 100

2. 若有以下定义，则赋值正确的是(　　)。

```
int a,b,*p;
float c,*q;
```

A. p=&c　　B. q=p　　C. p=NULL　　D. q=new int

3. 如果 x 是整型变量，则合法的形式是(　　)。

A. &(x+5)　　B. *x　　C. &*x　　D. *&x

4. 若有语句“int a[10]={0,1,2,3,4,5,6,7,8,9},*p=a;”，则下列(　　)不是对数组 a 中元素的正确引用(其中 0≤i<10)。

A. p[i]　　B. *(*(a+i))　　C. a[p-a]　　D. *(&a[i])

5. 如下语句的输出结果为(　　)。

```
int **pp,*p,a=10,b=20;
pp=&p;              //二级指针 pp 取一级指针 p 地址
p=&a;               //一级指针 p 取变量 a 地址
p=&b;               //一级指针 p 取变量 b 地址
cout<<*p<<","<<**pp<<endl;
```

A. 10,20　　B. 10,10　　C. 20,10　　D. 20,20

6. 若有程序段“static char line[]="Visual C++";char *point=line;”则 point 的值为(　　)。

A. "Visual C++"　B. line 的首地址　　C. Visual　　D. \0

7. 相同数据类型的数组名和指针变量均表示地址，以下说法不正确的是(　　)。

A. 数组名代表的地址不变，指针变量存放的地址可变
B. 数组名代表的存储空间不变，指针变量指向的存储空间长度可变
C. A 和 B 的说法均正确
D. 没有差别

8. 若有以下定义，则选项中操作错误的是(　　)。

```
int a[5]={1,3,5,7,9}, *p=new int [5];
```

A. p=a　　B. *p=a　　C. *(p+1)=a[1]　　D. *p=*a

9. 若有以下定义，则释放指针所指内存空间的操作是(　　)。

```
int *p=new int [10];
```

A. delete []p　　B. delete *p　　C. delete p　　D. delete p[]

10. 执行下列程序段后，字符串 str1 的值是(　　)。

```
char str1[8],str2[]="AA\0BB";
strcpy(str1,str2);
```

A. AA\0BB　　B. AA BB　　C. AA　　D. AA\0BB\0

二、填空题

1. 数组定义时有 3 个要素：数组名、数组元素的_____和数组元素的_____。数组中的元素是按元素在数组中的位置进行访问的，是通过_____进行的，称为_____或_____访问。为了在声明数组时对数组大小的修改更为方便，总是将_____用于声明数组长度。

2. 计算机内存是一维编址的，多维数组在内存中的存储_____，C/C++ 多维数组在内存中的排列是_____方式，即越_____的下标变化_____。设数组 a 有 m 行 n 列，每个元素占内存 u 个字节，则 a[i][j]的首地址为_____ +_____。

3. 指针变量保存了另一个变量的_____值。不能任意给指针变量赋地址值，只能赋给它_____和_____的地址。使用变量名来访问变量，是按_____来直接存取变量，称为_____方式；而借助指针变量取得另一个变量的地址，访问该变量的方式称为_____方式。

三、程序题

1. 写出下列程序的运行结果。

```
#include<iostream>
int main()
{
    int x=100,y=200;
    int *p=&x;
    (*p)++;
    p=&y;
    *p=x;
    cout<<x<<","<<*p<<endl;
}
```

2. 写出下列程序的运行结果。

```
#include<iostream>
int main()
{
    int a[]={10,20,30,40,50,60,70,80,90,100};
    int i,*p,sum=0;
    p=a;
    for(i=1;i<10;i=i+2)
        sum+=p[i];
    cout<<"sum="<<sum<<endl;
}
```

3. 写出下列程序的运行结果。

```
#include<iostream>
const int N=5;
int main()
{
    char* course[]={"Mathematics","English","Data  structure","C++ Pro-
                    gramming","Internet" };
    int i,j,k;
    char *temp;
    for(i=0;i<N-1;i++)
    {
        k=i;
        for(j=i+1;j<N;j++)
            if(strcmp(course[k],course[j])>0)
                k=j;
            if(k!=i)
            {
                temp=course[i];
                course[i]=course[k];
                course[k]=temp;
            }
    }
    for(i=0;i<N;i++)
        cout<<course[i]<<endl;
}
```

四、简答题

1. 运算符“*”和“&”的作用分别是什么？

2. 设 a 为数组名，那么“a++;”是否合法？为什么？

3. 指针作为函数的参数时，它传递的是什么？实参要用什么？而使用引用作为函数的参数时，实参要用什么？何时只能用指针而不能用引用？

五、编程题

1. 打印杨辉三角形(10 行)。

2. 写一个函数，将一个 3 阶方阵转置。

3. 使用指针编写函数 strcat()，实现两个字符串的首尾连接(将字符串 str2 接到 str1 的后面，str1 最后面的'\0'被取消)。