**程序设计教学大纲**

第9节

2018年1月15日星期日

**一、为什么要使用数组**

　　通过前面几章的学习，我们已经可以编写程序来解决各种相当复杂的问题了，但是当需要处理的数据比较多时，仅依靠前面的知识是不够的，即使简单的问题也可能需要比较复杂的程序来处理。请看下面的例子：

　　例题：输入50个学生的某门课程的成绩，打印出低于平均分的学生序号与成绩。

　　【分析】在解决这个问题时，虽然可以通过一个变量来累加读入的50个成绩求出学生的总分，进而求出平均分。但因为只有读入最后一个学生的分数后才能求得平均分，并且要求打印出低于平均分的学生序号和成绩，故必须把50个学生的成绩都保留起来，然后逐个和平均分比较，把低于平均分的成绩打印出来。如果，用简单变量a1，a2,…，a50存储这些数据，要用50个变量保存输入的数据，程序片断如下：

　　cin>>a1>>a2>>…>>a10;

　　…

　　cin>>a41>>a42>>…>>a50;

注意，如果真正要像上面这样编写程序，则上面的所有省略号必须用完整的语句写出来。可以看出，这样的程序是多么繁琐。如果说处理的数据规模达到成千上万，上面的例子单单读入就会异常复杂，电脑的优势没有得到体现。

　　从以上的讨论可以看出，如果只使用简单变量处理大量数据，就必须使用大量只能单独处理的变量，即使是简单问题也需要编写冗长的程序。

　　可以看出，我们需要把一大批具有相同性质的数据组合成一个新类型的变量，可以用简单的程序（比如循环50次）对这个新变量的各个分量进行相同的处理，每个分量仍然保留单个变量的所有性质（在上面的例子中，各分量是整型变量或实型变量的性质）。

　　如果能像数学中使用下标变量ai形式表示这50个数，则问题就容易实现。在C++语言中，具有下标性质的数据类型是数组。如果使用数组，上面的问题就变得十分简单、清晰。例如，读入50个学生的成绩，只需写如下语句即可：

　　 for (int i=1;i<=50;++i)

　　 cin>>a[i];

　　在这里引用了带下标的变量（分量变量称为数组元素）a[i]来代替a1,a2…,a50,方括号中的i称为下标，当循环变量i=1时a[i]就是a[1]；当i=2时a[i]就是a[2]……；当i=50时a[i]就是a[50]。输入的时候，让i从1变化到50，循环体内输入语句中的a[i]也就分别代表了a1,a2…,a50这50个带下标的变量。这样上述问题的程序可写为：

　　tot = 0; // tot存储50个学生的总分

　　for (int i=1;i<=50;++i) // 循环读入每一个学生的成绩，并把它累加到总分中

　　{

　　 cin>>a[i];

　　 tot+=a[i];

　　}

　　ave= tot/50; //计算平均分

　　for (int i=1;i<=50;++i)

　　　if (a[i]<ave) cout<<"No. "<<i<<" "<<a[i];

　　//如果第i个同学成绩小于平均分，则将输出这个学生的序号和成绩。

要在程序中使用下标变量，必须先说明这些下标变量的整体为数组，即数组是若干个同名（如上面的下标变量的名字都为a）下标变量的集合，这些变量的类型全部一致。

**二、一维数组的定义**

　　当数组中每个元素只带有一个下标时，我们称这样的数组为一维数组。

　　数组的定义格式如下：

　　　类型标识符 数组名[常量表达式]

　　说明：

　　①数组名的命名规则与变量名的命名规则一致。

　　②常量表达式表示数组元素的个数。可以是常量和符号常量，但不能是变量。

　　例如：

　　int a[10]; //数组a定义是合法的

int b[n]; //数组b定义是非法的

其中，a是一维数组的数组名，该数组有10个元素，依次表示为：a[0],a[1],a[2],a[3],a[4],a[5]，a[6],a[7],a[8],a[9]。需要注意的是：a[10]不属于该数组的空间范围。当在说明部分定义了一个数组变量之后,C++编译程序为所定义的数组在内存空间开辟一串连续的存储单元，每个数组第一个元素的下标都是0，因此第一个元素为第0个数组元素。例如：上例中的a数组在内存的存储如表所示：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| a[0] | a[1] | a[2] | a[3] | a[4] | a[5] | a[6] | a[7] | a[8] | a[9] |

a数组共有10个元素组成，在内存中10个数组元素共占10个连续的存储单元。a数组最小下标为0，最大下标9。按定义a数组所有元素都是整型变量。

**三、一组数组的引用**

通过给出的数组名称和这个元素在数组中的位置编号(即下标)，程序可以引用这个数组中的任何一个元素。

**一维数组元素的引用格式：**

**数组名[下标]**

例如：若i、j都是int型变量，则

a[5]

a[i+j]

a[i++]

都是合法的元素。

**说明：**

(1)下标可以是任意值为整型的表达式，该表达式里可以包含变量和函数调用。引用时，下标值应在数组定义的下标值范围内。

(2)数组的精妙在于下标可以是变量，通过对下标变量值的灵活控制，达到灵活处理数组元素的目的。

(3)C++语言只能逐个引用数组元素，而不能一次引用整个数组。

　　(4)数组元素可以像同类型的普通变量那样使用，对其进行赋值和运算的操作，和普通变量完全相同。

**例如： c[10]=34;实现了给c[10]赋值为34。**

**四、一维数组的初始化**

数组的初始化可以在定义时一并完成。格式：

　类型标识符 数组名[常量表达式]={值1，值2，…}

　　例如：

　　　int a[5]={1,2,3,4,5}

　　说明：

　　(1)在初值列表中可以写出全部数组元素的值，也可以写出部分。例如，以下方式可以对数组进行初始化：

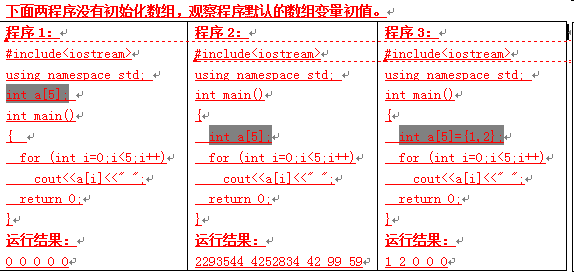
　　int x[10]={0,1,2,3,4};

　　该方法一次仅对数组的前5个元素依次进行初始化。

　　(2)对数组元素全部初始化为0，可以简写为：{0}。

　　例如：

int a[5]={0}; 将数组a的5个元素都初始化为0。



**【说明】**  
 程序1、程序2和程序3的区别在于数组定义在int main()之外与之内，程序1中数组定义放在int main()之外，其初始值是0值。程序2中数组定义放在int main()之内，其初始值是随机的。程序2中数组定义放在int main()之内，只给a[0]、a[1]赋初值，但后面的a[2]～a[4]元素自动赋0值。 **五、数组越界**  
 C++语言规定，使用数组时，要注意：  
 (1)、数组元素的下标值为正整数。  
 (2)、在定义元素个数的下标范围内使用。  
 然而，当在程序中把下标写成负数、大于数组元素的个数时，程序编译的时候是不会出错的。例如：  
 int a[10];  
 a[-3]=5;  
 a[20]=15;  
 a[10]=20;  
 int k=a[30]  
 这些语句的语法是正确的，能够通过程序的编译。然而，它们要访问的数组元素并不在数组的存储空间的，这种现象叫数组越界。例如下面程序：

#include<iostream>  
using namespace std;   
int main()  
{  
 int a[5];  
 for (int i=0;i<=10;i++)  
 {  
 a[i]=i;  
 cout<<a[i]<<" ";  
 }  
 return 0;  
} **【说明】**  
 该程序能够通过编译，也能运行出结果，程序的问题是定义a[5]，使用时数组下标超过了4。C++语言中，数组越界访问系统时不一定会给出任何的提示，也就是说，程序可以超出数组边界进行读/写，从而造成内存的混乱。  
 数组越界是实际编程中常见的错误，而且这类错误往往难以捕捉。因为越界语句本身并不一定导致程序立即出错，可能在遇到某些数据时才导致错误，有时由于越界，意外地改变了变量或指令，导致在调试器里调试的时候，程序不按照应当的次序运行的怪现象。

**练习题：**

练习1：输入n个数,要求程序按输入时的逆序把这n个数打印出来，已知整数不超过100个。也就是说，按输入相反顺序打印这n个数。

练习2：输入一系列票数，1 5代表1号选手，5票；直到输入0 0结束输入，然后统计各个选手的选票数。

练习3：将a数组中第一个元素移到数组末尾,其余数据依次往前平移一个位置。

练习4：录入一段话，分别统计这句话中的大写英文字母个数，小写英文字母个数、数字个数以及空格、逗号等符号个数。