## **百钱百鸡**

### **案例描述**

中国古代数学家张丘建在他的《算经》中提出了一个著名的“百钱百鸡问题”：一只公鸡值五钱，一只母鸡值三钱，三只小鸡值一钱，现在要用百钱买百鸡，请问公鸡、母鸡、小鸡各多少只？

### 案例分析

如果用一百钱只买一种鸡，那么，公鸡最多20只，母鸡最多33只，小鸡最多300只。但题目要求买100只，所以小鸡的数量在0~100之间，公鸡数量在0~20之间，母鸡数量在0~33之间。我们把公鸡，母鸡和小鸡的数量分别设为cock、hen、chicken，通过上述分析可知：

（1）0<=cock<=20；

（2）0<=hen<=33；

（3）0<=chicken<=100；

（4）cock+hen+chicken=100；

（5）5\*cock+3\*hen+chicken/3=100。

与此同时，可知母鸡、小鸡和公鸡的数量相互限制，这里可以使用三层循环嵌套来解决此问题。在实现案例之前，先来学习完成程序需要的知识。

### 必备知识

#### 循环的嵌套

有时为了解决一个较为复杂的问题，需要在一个循环中再定义一个循环，这样的方式被称作循环嵌套。在C语言中，while、do…while、for循环语句都可以进行嵌套，并且它们之间也可以互相嵌套。 常用的几种嵌套语句如表3-1所示。

1. 常用的几种循环嵌套

|  |  |  |
| --- | --- | --- |
| **常用的循环嵌套形式** | | |
| while()  {  while()  {….}  } | do  {  do{…}  while();  }while(); | for( ; ;)  {…  for( ; ;)  {…}  } |
| while()  {  do{…}  while();  } | for( ; ;)  { …  while(){…}  …  } | do  {…  for( ; ;)  {…}  …  } |

其中for循环嵌套是最常见的循环嵌套，其语法格式如下所示：

for(初始化表达式; 循环条件; 操作表达式)

{

……

for(初始化表达式; 循环条件; 操作表达式)

{

执行语句;

……

}

……

}

### 案例实现

#### 1.案例设计

（1）先定义三个整型变量分别用来存储公鸡、母鸡和小鸡；

（2）第一层for循环控制公鸡的数量，第二层for循环控制母鸡的数量，第三层for循环控制小鸡的数量；

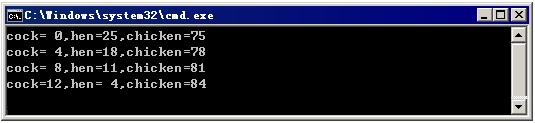
（3）根据这三层循环我们可以得到很多种方案，但是其中有很多是不符合条件的，我们要把合理的方案筛选出来，即把满足“cock+hen+chicken=100”和“5\*cock+3\*hen+chicken/3=100”的方案输出。

#### 2.完整代码；

1. #include <stdio.h>
2. int main()
3. {
4. int cock,hen,chicken;
5. for(cock = 0; cock <= 20; cock++) //控制公鸡的数量
6. for (hen = 0; hen <= 33; hen++) //控制母鸡的数量
7. for (chicken = 0; chicken <= 100; chicken++) //控制小鸡的数量
8. {
9. if((5\*cock+3\*hen+chicken/3.0 == 100)&&(cock+hen+chicken == 100))
10. printf("cock=%2d,hen=%2d,chicken=%2d\n",cock,hen,chicken);
11. //将满足条件的方案，直接输出到屏幕上
12. }
13. return 0;
14. }

代码第9行把条件五改成了“5\*cock+3\*hen+chicken/3.0=100”，这是因为C语言中两个整数相除得到结果仍为整数，“/”两边如果有一个数是float类型时，所得结果为float型，在以后编程时要注意对“/”两边数据类型进行处理。

运行结果如图3-17所示。



运行结果

🕮**多学一招：算法优化**

上述算法需要尝试21\*34\*101=72114次，为了提高效率，可以对算法进行优化。当公鸡和母鸡的数量确定后，小鸡数量固定为100-cock-hen，此时约束条件只剩条件五了，代码如下：

1. #include <stdio.h>
2. int main()
3. {
4. int cock,hen,chicken;
5. for(cock = 0; cock <= 20; cock++)
6. for (hen = 0; hen <= 33; hen++)
7. {
8. chicken = 100 - cock - hen;
9. if(5\*cock+3\*hen+chicken/3.0 == 100)
10. printf("cock=%2d,hen=%2d,chicken=%2d\n",cock,hen,chicken);
11. }
12. return 0;
13. }

此算法只需尝试21\*34=714次，大大缩短运算时间。

## **回文素数**

### **案例描述**

若整数i从左向右读与从右向左读是相同的数，且i为素数，此时称其为回文素数。所谓素数是指只能由1和它本身整除的整数。

对于偶数位的整数，只有11是回文素数。也就是说，除了11以外，所有的2位整数都不是回文素数。所有的4位整数、6位整数、8位整数中也不存在回文素数。但是三位回文素数有很多，比如：101、131、151、181、191、313等。本案例要求通过编程求出所有小于1000的回文素数。

### 案例分析

因为要对所有1000以内的整数进行判断，所以此处适合用循环结构语句；又因为要判断是否为素数以及判断是否为回文素数，所以一定会用到选择结构语句。此案例综合了本章这两个重要的知识点，请灵活运用学过的知识解决此案例。

### 案例实现

#### 1.案例设计

（1）先采用穷举法对1000以内所有整数进行遍历，判断其是否为素数。判断一个数是否为素数的关键在于，判定整数能否被1和它自身之外的其他整数所整除，如果都不能整除，则此数为素数；

（2）如果此数为素数，则继续判断此数是两位数还是三位数；

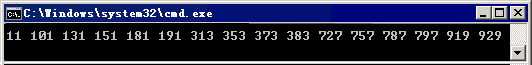
（3）如果为两位数，则判断其十位和个位是否相同，如果相同则说明此数为回文素数；如果是三位数，则判断其百位和个位是否相同，如果相同则说明此数为回文素数；

（4）最后将所有小于1000的回文素数打印输出到屏幕上即可。

#### 2.完整代码

1. #include <stdio.h>
2. int main()
3. {
4. int flag;
5. //定义整型变量flag用来记录是否为素数，1代表是，0代表不是
6. int n;
7. int i;
8. for (n = 10; n < 1000; n++)
9. {
10. for (i = 2; i < n; i++)
11. {
12. flag = 1; //flag默认为1
13. if (n % i == 0)
14. {
15. flag = 0; //如果不符合素数要求，则把flag置为0
16. break; //并跳出循环
17. }
18. }
19. if(flag == 1) //判断是否是素数
20. {
22. if (n / 100 == 0) //判断是否是两位数
23. {
24. if (n / 10 == n % 10) //判断十位和各位是否相同
25. {
26. printf("%4d", n);
27. }
28. }
29. else
30. {
31. if (n / 100 == n % 10) //判断百位和个位是否相同
32. {
33. printf("%4d", n);
34. }
35. }
36. }
37. }
38. printf("\n");
39. return 0;
40. }

运行结果如图3-21所示。



运行结果

## **听说过自守数么**

### **案例描述**

如果某个数的平方的末尾几位等于该数自身，那么就称这个数为自守数。例如，0和1的平方的个位数仍然是0和1，所以0和1是自守数，称为平凡自守数。很显然，5和6是一位自守数，因为5×5=25，6×6=36。而25和76是两位自守数，因为25×25=625，76×76=5776，当然还有三位自守数，四位自守数等等，在此不再一一介绍。

自守数有一个特性，以它为后几位的两个数相乘，乘积的后几位仍是这个自守数。因为5是自守数，所以以5为个位数的两个数相乘，[乘积](http://baike.baidu.com/view/563968.htm)的个位仍然是5；76是自守数，所以以76为后两位数的两个数相乘，其结果的后两位仍是76，如176×576=101376。

案例要求编程求出0~10000内的所有自守数，并依次输出到屏幕上。

### 案例分析

根据自守数的定义可知，案例的关键是知道当前所求数的位数。由于数字范围控制在了10000以内，所以直接求出此数的平方后再截取最后相应的位数，和原数比较，判断是否相同即可。

### 案例实现

#### 1.案例设计

（1）用for循环遍历1~10000之间的所有整数；

（2）求出当前循环中此数的平方和此数的位数；

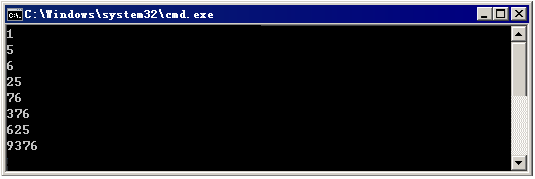
（3）通过对此数的平方取余求出此数的尾数；

（4）判断尾数是否和此数相等，如果相等则为自守数，将此数输出到屏幕上。

#### 2.完整代码

1. #include <stdio.h>
2. #include <math.h>
3. int main()
4. {
5. int i, a, k, m; //此数为i
6. for(i = 1; i < 10000; i++) //在1~10000的范围内依次检验
7. {
8. a = i; //把i的值赋给a
9. a \*= a; //求出a的平方即i的平方
10. k = log10(i) + 1; //求出i的位数k
11. for(m = 1; k; k--)
12. m \*= 10;
13. a %= m; //通过取余求出尾数
14. if(a == i) //判断i的平方的尾数是否等于i
15. printf("%d\n", i); //输出所有自守数
16. }
17. return 0;
18. }

运行结果如图3-20所示。



运行结果

## **最大值和最小值**

### **案例描述**

最值问题可谓是经典中的经典了，说它是每个程序员都该掌握的知识一点也不为过。本案例要求先输入数组的大小和各个数组元素，然后求出数组中的最大值和最小值以及它们所在的位置，最后把它们依次输出到屏幕上。

### 案例分析

本案例是应用一维数组的典型案例。C语言中规定，只能逐个引用数组中的元素，而不能引用整个数组。所以在对数组元素进行判断时，只能通过循环对数组元素逐个引用，通过判断每一个元素值的大小，找出其中最大和最小的元素。

为了更好地完成此案例，请先认真学习一维数组的相关知识。

### 必备知识

#### 1. 一维数组的定义与初始化

一维数组也称向量，它用以组织具有一维顺序关系的一组同类型数据。在C语言中，一维数组的定义方式如下所示：

数据类型 数组名[常量表达式];

在上述语法格式中，类型说明符表示数组中所有元素的类型，常量表达式指的是数组的长度，也就是数组中存放元素的个数。例如：

int array[5];

上述代码定义了一个数组，编译器为数组分配存储空间。其中，int是数组的类型，array是数组的名称，方括号中的5是数组的长度。值得注意的是：数组占据的内存空间是连续的，这样，很容易计算数组占据的内存大小和每个元素对应的内存首地址。例如对上例来说，其占据的内存大小为：5\*sizeof(int)。

完成数组的定义后，编译器根据数组定义语句中提供的数据类型和数组长度给数组变量分配适当的内存空间。这时，如果想使用数组操作数据，还需要对数组进行初始化。数组初始化元素值的常见方式有三种，具体如下：

（1）直接对数组中的所有元素赋初值，示例代码如下：

int i[5]={1,2,3,4,5};

上述代码定义了一个长度为5的数组i，数组中元素的值依次为1、2、3、4、5。

（2）只对数组中的一部分元素赋值，示例代码如下：

int i[5]={1,2,3};

在上述代码中，定义了一个int类型的数组，但在初始化时，只对数组中的前三个元素进行了赋值，其它元素的值会被默认设置为0。

（3）对数组全部元素赋值，但不指定长度，示例代码如下：

int i[]={1,2,3,4};

在上述代码中，数组i中的元素有4个，系统会根据赋值号右边初始值列表中给出的初值个数自动设置数组的长度。因此，数组i的长度为4。

注意：

1、数组的下标是用方括号括起来的，而不是圆括号。

2、数据类型不仅可以是int、float、char等基本类型，也可以是后续章节将要介绍的指针、结构体等类型。

3、数组名的命名规则与变量名的命名规则相同。

4、数组定义中，常量表达式的值可以是符号常量，示例如下：

int a[N]; //假设预编译命令#define N 4, 下标是符号常量

#### 2. 一维数组的引用

在程序中，经常需要访问数组中的一些元素，因为数组名的本质是存放该数组在内存中的地址的常量，所以无法进行任何计算，这时可以通过数组名和下标来引用数组中的元素。一维数组元素的引用方式如下所示：

数组名[下标];

在上述方式中，下标指的是数组元素的位置，数组元素的下标是从0开始的。例如，引用数组x中第三个元素的方式为：x[2]。

注意：

数组的下标都有一个范围，即“0~[数组长度-1]”，假设数组的长度为6，其下标范围为0～5。当访问数组中的元素时，下标不能超出这个范围，否则程序会报错。

### 案例实现

#### 1.案例设计

（1）先输入数组的大小；

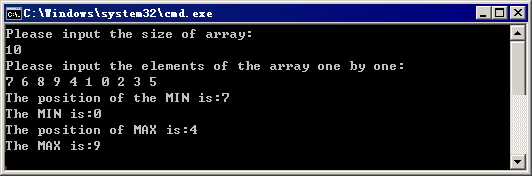
（2）利用for循环依次输入数组中的各个元素；

（3）分别求出数组元素中的最大值和最小值，并输出到屏幕上。

#### 2.完整代码

1. #include <stdio.h>
2. int main()
3. {
4. int a[50]; //定义数组存放元素
5. int MAX, MIN; //定义最大值和最小值变量
6. int i, n;
7. int j = 0;
8. int k = 0;
9. printf("Please input the size of array:\n");
10. scanf("%d", &n); //输入数组的元素个数
11. printf("Please input the elements of the array one by one:\n");
12. for (i = 0; i < n; i++) //依次输入数组中的元素
13. scanf("%d", &a[i]);
14. MIN = a[0]; //数组首元素默认为最小值
15. for (i = 1; i < n; i++) //找出数组元素中的最小值
16. { if (a[i] < MIN) //如果有比MIN小的元素
17. {
18. MIN = a[i]; //就把此元素赋值给MIN
19. j = i + 1; //将存储最小值的位置赋给j
20. }
21. }
22. MAX = a[0]; //数组首元素默认为最大值
23. for (i = 1; i < n; i++) //找出数组元素中的最大值
24. {
25. if (a[i] > MAX) //如果有比MAX大的元素
26. {
27. MAX = a[i]; //就把此元素赋值给MAX
28. k = i + 1; //将存储最大值的位置赋给k
29. }
30. }
31. printf("The position of the MIN is:%d\n", j); //输出最小值所在的位置
32. printf("The MIN is:%d\n", MIN);
33. printf("The position of MAX is:%d\n", k); //输出最大值所在的位置
34. printf("The MAX is:%d\n", MAX);
35. return 0;
36. }

运行结果如图5-1所示。



运行结果

## **猜宝游戏**

### 案例描述

学生时代的生活虽然单一，但也有许多小游戏贯穿其中，给平淡的校园生活带来一丝欢乐，猜硬币就是这些游戏之一。某个课间，甲和乙一起玩猜硬币的游戏：初始时，甲的左手握着一枚硬币，游戏开始后，甲进行有限次或真或假的交换，最后由乙来猜测这两只手中是否有硬币。

本案例要求编写程序，实现游戏过程。

### 案例分析

由于该案例比较主观，并且甲的手法和乙的眼力都能影响游戏的结果，因此本案例的目的在于模拟游戏过程。

因为游戏要执行有限次，所以需要首先确定交换进行的次数，通过循环执行每次交换；又因为每次交换是真是假并不确定，所以至少需要实现两个交换函数，一个函数真正地实现两个手中硬币的交换，另一个只需表面完成交换。而每次是否真正地交换硬币也是随机的，因此使用随机数发生器来决定每次选择执行的函数。

本案例中将涉及到指针的相关使用方式，下面先来学习这些知识。

### 必备知识

#### 指针作为函数参数

在C语言中，实参和形参之间的数据传递是单向的值传递，即只能由实参传递给形参，而不能由形参传递给实参。这与C语言中内存的分配方式有关。当发生函数调用时，系统会使用形参对应的实参为形参赋值，此时的形参以及该函数中的变量都存放在函数调用过程中系统在栈区开辟的空间里，栈区随着函数的调用而被分配，随着函数的结束而被释放，在此过程中，栈区对主调函数不可见，因此主调函数并不能读取栈中形参的数据。若要将栈中的数据传递给主调函数，只能用关键字“return”来实现。

但并非所有从主调函数传入被调函数的数据都是不需要改变的。在第四章学习函数时曾讲到过返回值，利用返回值可以将在被调函数中修改的数据返回给主调函数，但是C语言中返回值只能返回一个数据，往往不能达到要求；函数中也曾学到过全局变量，但是这种方式违背模块化程序设计的原则，与函数的思想背道而驰。

本节将学习一种新的方法，即使用指针变量作为函数的形参，通过传递地址的方式，使形参和实参都指向主调函数中数据所在地址，从而使被调函数可以对主调函数中的数据进行操作。

#### 指针的交换

根据指针可以获得变量的地址，也可以得到变量的信息，所以指针交换包含两个方面，一是指针指向交换，二是指针所指地址中存储数据的改变。

（1）指针指向交换

若要交换指针的指向，首先需要申请一个指针变量，记录其中一个指针原来的指向，再使该指针指向另外一个指针，使另外一个指针指向该指针原来的指向。假设p和q都是int\*型的指针，则其指向交换示意图如图6-4所示。



1. 指针交换指向
2. 具体的实现方法如下：

int \*tmp=NULL; //创建辅助变量指针

tmp=p; //使用辅助指针记录指针p的指向

p=q; //使指针p记录指针q的指向

q=tmp; //使指针q指向p原来指向的地址

1. （2）数据的交换
2. 若要交换指针指向的空间中的数据，首先需要获取数据，获取数据的方法在案例一中已经学习，即使用“\*”运算符取值。假设p和q都是int\*型的指针，则数据交换示意图如图6-5所示。
3. 
4. 数据交换
5. 具体的实现方法如下：

int tmp=0; //创建辅助变量

tmp=\*p; //使用辅助变量记录指针p指向地址中的数据

\*p=\*q; //将q指向地址中的数据放到p所指地址中

\*q=tmp; //将p中原来的数据放到q所指地址中

### 案例实现

#### 案例设计

（1）使用基类型的变量作为形参，构造交换函数；

（2）使用指针变量作为形参，在函数体中交换指针的指向；

（3）使用指针变量作为形参，在函数体中交换指针变量所指内存中存储的数据；

（4）使用随机数生成器确定交换发生的次数，选择每轮要执行的交换方法；

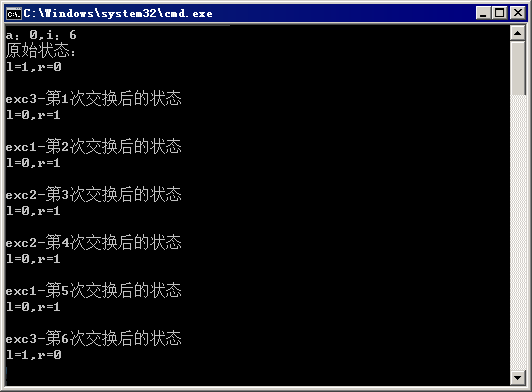
（5）使用while循环语句控制交换进行的轮数；

（6）使用switch语句根据产生的随机数选择本轮执行的交换方法。

#### 完整代码

1. #include <stdio.h>
2. #include <stdlib.h>
3. //函数声明
4. void exc1(int l, int r);
5. void exc2(int\* l, int\* r);
6. void exc3(int\* l, int\* r);
7. //游戏模拟
8. //使用随机函数获取交换的次数，和每次交换所选择的函数
9. int main()
10. {
11. int a = 0, i = 0, j;
12. int l = 1, r = 0;
13. srand((unsigned int)time(NULL));
14. i = 5 + (int)(rand() % 5); //随机设置交换次数
15. j = i;
16. printf("a：%d,i：%d\n",a,i);
17. printf("原始状态：\n");
18. printf("l=%d,r=%d\n\n", l, r);
19. while (i>0)
20. {
21. i--;
22. a = 1 + (int)(rand() % 3);
23. switch (a)
24. {
25. case 1:
26. exc1(l, r);
27. printf("exc1-第%d次交换后的状态\n", j - i);
28. printf("l=%d,r=%d\n\n", l, r);
29. break;
30. case 2:
31. exc2(&l, &r);
32. printf("exc2-第%d次交换后的状态\n", j - i);
33. printf("l=%d,r=%d\n\n", l, r);
34. break;
35. case 3:
36. exc3(&l, &r);
37. printf("exc3-第%d次交换后的状态\n", j - i);
38. printf("l=%d,r=%d\n\n", l, r);
39. break;
40. default:
41. break;
42. }
43. }
44. return 0;
45. }
46. //函数定义
47. void exc1(int l, int r)
48. {
49. int tmp;
50. tmp = l; //交换形参的值
51. l = r;
52. r = tmp;
53. }
54. void exc2(int\* l, int\* r)
55. {
56. int\* tmp;
57. tmp = l; //交换形参的值
58. l = r;
59. r = tmp;
60. }
61. void exc3(int\* l, int\* r)
62. {
63. int tmp;
64. tmp = \*l; //交换形参变量指向内容的值;
65. \*l = \*r;
66. \*r = tmp;
67. }

运行结果如图所示。



运行结果

## 到底有多少单词

### 案例描述

要求编程求出一句话中到底有多少单词。首先在屏幕上输入一句话，每个单词之间用一个空格隔开，要求第一个字符和最后一个字符都不能为空格；然后统计出这句话的单词数量，并把结果输出到屏幕上。

### 案例分析

在程序中一句话可用一串字符表示，输入一串字符需要用到刚刚学过的gets()函数；一句话中单词的数量可以根据获取的字符串中空格的数量确定。综上，本案例的代码应实现以下功能：

（1）使用字符数组变量接收gets()函数获取的字符串；

（2）计算字符数组中的空格数量，推算出此句话中的单词数量；

（3）将统计结果输出。

### 案例实现

#### 1.案例设计

（1）首先使用gets()函数将输入的字符串保存在str字符数组中；

（2）然后使用if语句判断用户输入的第一个字符是否为结束符：如果是，则要给出“这句话没有单词”的提示；如果不是，说明字符串正常，开始执行else语句里的代码；

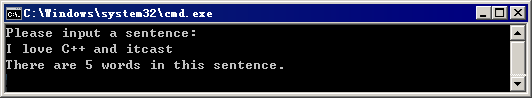
（3）采用for循环遍历字符数组中的每个字符，如果遇到结束符，立即结束循环；如果没有遇到结束符，则判断字符是否为空格。如果是，则把单词数量加1；反之不做任何操作，继续进行循环，判断下一个字符；

（4）最后把总单词数输出到屏幕上。

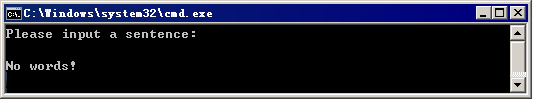
#### 2.完整代码

1. #include <stdio.h>
2. int main()
3. {
4. char str[50]; //定义保存字符串的数组
5. int i, count=1; //count表示单词的个数
6. char blank; //表示空格
7. printf ("Please input a sentence:\n");
8. gets(str); //输入字符串
9. if(str[0]=='\0') //判断如果字符串为空的情况
10. printf("No words!\n");
11. else
12. {
13. for(i=0;str[i]!='\0';i++) //循环判断每一个字符
14. {
15. blank=str[i]; //得到数组中的字符元素
16. if(blank==' ') //判断是不是空格
17. count++; //如果是则加1
18. }
19. printf("There are %d words in this sentence.\n",count);
20. }
21. return 0;
22. }

运行结果如图所示。



运行结果



运行结果

## 求学生平均成绩

### 案例描述

一个小组中有3个学生，每个学生有3门课程的成绩需要统计。案例要求通过编程依次输入学生的学号、姓名和三门课程的成绩，计算出平均成绩并依次把每个学生的学号、姓名和平均成绩打印在屏幕上。

### 案例分析

很显然，用刚刚学过的结构体存放学生的信息是最好的选择。一个结构体变量可以存储一组数据，如一个学生的学号、姓名和成绩等。但是本案例中有3个学生的信息需要存储，因此需要采用结构体数组。与前面讲解的数组不同，结构体数组中的每个元素都是结构体类型的，它们都具有若干个成员项。接下来将针对结构体数组的定义、引用及初始化方式进行讲解。

### 必备知识

#### 1.结构体数组的定义

假设一个班有20个学生，如果需要描述这20个学生的信息，可以定义一个长度为20的Student类型数组，与定义结构体变量一样，可以采用三种方式定义结构体数组stus。

1. （1）先定义结构体类型，后定义结构体数组，具体示例如下：

struct Student

{

int num;

char name[10];

char sex;

};

struct Student stus[20];

1. （2）在定义结构体类型的同时定义结构体数组，具体示例如下：

struct Student

{

int num;

char name[10];

char sex;

}stus[20];

1. （3）直接定义结构体数组，具体示例如下：

struct

{

int num;

char name[10];

char sex;

}stus[20];

#### 2.结构体数组的初始化

结构体数组与数组类似，都通过为元素赋值的方式完成初始化。由于结构体数组中的每个元素都是一个结构体变量，因此，在为每个元素赋值的时候，需要将其成员的值依次放到一对大括号中。

例如，定义一个结构体数组students，该数组有3个元素，并且每个元素有num、name、sex三个成员，可以采用下列两种方式对结构体数组students初始化。

（1）先定义结构体数组类型，然后初始化结构体数组，具体示例如下：

struct Student

{

int num;

char name[10];

char sex;

};

struct Student students[3] = { {20160101, "Zhang San",'M'},

{20160102, "Li Si",'W'}

{20160103, "Zhao Wu",'M'}

};

（2）在定义结构体数组的同时，对结构体数组初始化，具体示例如下：

struct Student

{

int num;

char name[10];

char sex;

}students[3] = { {20160101, "Zhang San",'M'},

{20160102, "Li Si",'W'},

{20160103, "Zhao Wu",'M'}

};

当然，使用这种方式初始化结构体数组时，也可以不指定结构体数组的长度，系统在编译时，会自动根据初始化的值决定结构体数组的长度。因此以下的初始化方式也是合法的。

struct Student

{

int num;

char name[10];

char sex;

}students[] = { {20160101, "Zhang San",'M'},

{20160102, "Li Si",'W'},

{20160103, "Zhao Wu",'M'}

};

#### 3.结构体数组的引用

结构体数组的引用是指对结构体数组元素的引用，由于每个结构体数组元素都是一个结构体变量，因此，结构体数组元素的引用方式与结构体变量类似，其语法格式如下所示：

数组元素名称.成员名;

例如，要引用本案例第2个知识点结构体数组students[]中第一个元素的num成员，可以采用下列方式：

students[0].num;

#### 4.结构体数组指针

指针可以指向结构体数组，即将结构体数组的起始地址赋给指针变量，这种指针就是结构体数组指针。下面语句定义了Student结构体的一个数组和该数组的指针。

struct Student stu1[10],\*p=&stu1;

在上述代码中，p是指向Student结构体数组的指针，从定义上看，它和结构体指针没什么区别，只不过指向的是结构体数组。

### 案例实现

#### 1.案例设计

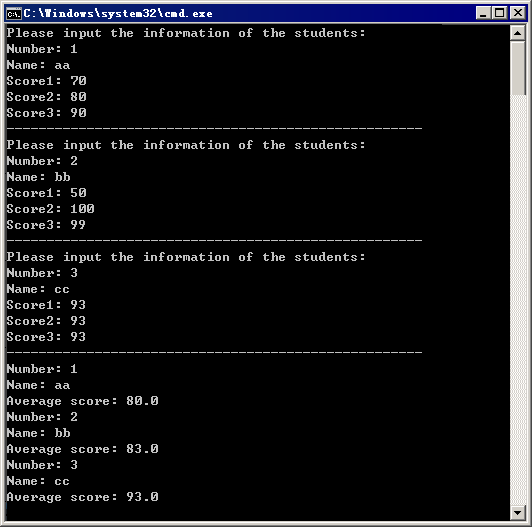
（1）定义一个结构体数组，用来存放3个学生的信息；

（2）用for循环依次读取3个学生的信息，再用for循环分别计算成绩总和并求出平均值。

#### 2.完整代码

1. #include <stdio.h>
2. struct student //定义结构体数组
3. {
4. char num[6];
5. char name[10];
6. int score[3];
7. float average;
8. }stu[5];
9. int main()
10. {
11. int i, j, k;
12. float sum;
13. for (i = 0; i < 3; i++) //通过循环依次输入三个学生的信息
14. {
15. printf("Please input the information of the students:\n");
16. printf("Number: ");
17. scanf("%s",stu[i].num); //输入学号
18. printf("Name: ");
19. scanf("%s",stu[i].name); //输入姓名
20. sum = 0;
21. for (j = 0; j < 3; j++) //输入三门成绩
22. {
23. printf("Score%d: ", j + 1);
24. scanf("%d", &stu[i].score[j]);
25. sum += stu[i].score[j]; //累加成绩
26. }
27. stu[i].average = sum / 3; //算出平均成绩
28. printf("----------------------------------------------------\n");
29. }
30. for (k = 0; k < 3; k++) //最后输出三个学生的信息以及平均成绩
31. {
32. printf("Number: %s\n", stu[k].num);
33. printf("Name: %s\n", stu[k].name);
34. printf("Average score: %.1f\n", stu[k].average);
35. }
36. return 0;
37. }

运行结果如图所示。



## 综合案例——图书管理系统

### 案例描述

随着科技的发展，计算机的普及，计算机软件在诸多领域都得到了广泛的应用。如今，管理系统不再是大公司的专利，许多小型的管理系统，如餐厅的餐饮管理系统、超市的收银系统、学校的学生选课系统等都已逐步普及到了我们的生活中。

本案例要求实现一个基于单链表的图书管理系统，该系统可实现图书信息的增加、浏览、查询、更新、删除这五项功能，并能将链表中存储的数据保存到文件中。

### 案例分析

本案例要求实现基于单链表的图书管理系统，并能实现增删改查这几项基本功能。第9章的案例6对单链表的定义与基础操作进行了详细地讲解，掌握单链表的基础操作是完成本案例的前提。另外图书信息应包含多项数据，所以链表结点中应为结构体类型的数据。当需要将链表中的数据保存到本地时，可以使用文件读写将数据写入创建的文件中。

### 案例实现

#### 案例设计

本案例包含多项功能，为了保证代码结构完整，脉络清晰，本案例的每个功能将被模块化为一个函数，在主函数中根据用户的选择，调用对应的功能。

本案例的功能函数如下：

（1）图书信息录入。该函数应实现增加数据的功能，其实质为链表结点的添加；

（2）图书信息浏览。该函数应实现链表中书籍信息的输出，其实质为链表的遍历；

（3）图书信息查询。该函数应能根据用户输入的某项信息，查找判断链表中是否存在相应记录，并将查找结果输出；

（4）图书信息修改。该函数应能根据用户输入的某项信息，找到对应记录，并修改记录中保存的信息；

（5）图书信息删除。该函数借助查询功能，查找链表中的数据，并将找到的数据对应的结点从链表中删除。

（6）图书信息保存。该函数应能将链表中的数据写入文件。

（7）创建书单。上述（1）~（6）这6个功能都依赖于链表，本案例设计使用链表来存储图书信息，在执行各项功能之前应先实现一个链表。

（8）菜单函数。本函数可展示功能菜单，提供用户与程序交互的入口。

#### 完整代码

StuManage.h//头文件

1. #include <stdio.h>
2. typedef struct book
3. {
4. char bnum[10]; //书籍编号
5. char bname[30]; //书名
6. char bauthor[20]; //作者
7. char bclassfy[10]; //类别编号
8. float bprice; //价格
9. struct book\* next; //链表指针
10. }BookInfo;
11. BookInfo\* CreateBooksList(); //创建链表
12. void Insert(BookInfo\* head); //插入
13. void Delete(BookInfo\* head); //删除
14. void Print(BookInfo\* head); //浏览
15. void Search(BookInfo\* head); //查询
16. void Update(BookInfo\* head); //修改
17. void Save(BookInfo\* head); //保存
18. int menu(); //菜单

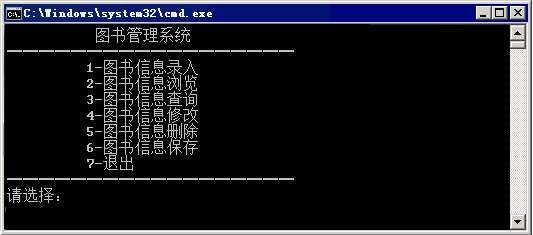
StuManage.c//函数定义

1. #define \_CRT\_SECURE\_NO\_WARNINGS
2. #include "StuManage.h"
3. #include <stdlib.h>
4. #include <string.h>
5. //创建书单
6. BookInfo\* CreateBooksList()
7. {
8. BookInfo\* head;
9. head = (BookInfo\*)malloc(sizeof(BookInfo)); //为头结点分配空间
10. head->next = NULL; //初始化头指针
11. return head;
12. }
13. //插入记录
14. void Insert(BookInfo\* head)
15. {
16. BookInfo \*b, \*p;
17. char flag = 'Y';
18. p = head;
19. while (p->next != NULL)
20. p = p->next;
21. //开辟新空间，存储书籍信息，并加入链表
22. while (flag == 'Y' || flag == 'y')
23. {
24. b = (BookInfo\*)malloc(sizeof(BookInfo)); //开辟新空间
25. printf("请输入图书编号："); //获取书籍信息
26. fflush(stdin); //清空缓冲区
27. scanf("%s", b->bnum);
28. printf("请输入书名：");
29. fflush(stdin);
30. scanf("%s", b->bname);
31. printf("请输入作者：");
32. fflush(stdin);
33. scanf("%s", b->bauthor);
34. printf("请输入类别编号：");
35. fflush(stdin);
36. scanf("%s", b->bclassfy);
37. printf("请输入图书价格：");
38. fflush(stdin);
39. scanf("%f", &b->bprice);
40. p->next = b; //将新增加的结点加入链表
41. p = b; //指针p向后移动，指向尾结点
42. b->next = NULL;
43. printf("添加成功！\n继续添加？(Y/N):");
44. fflush(stdin);
45. scanf("%c", &flag);
46. if (flag == 'N' || flag == 'n')break;
47. else if (flag == 'Y' || flag == 'y')continue;
48. }
49. return;
50. }
51. //删除记录
52. void Delete(BookInfo\* head)
53. {
54. BookInfo \*b, \*p;
55. char tmp[30];
56. int flag; //标志位，判断是否找到了要删除的书籍
57. flag = 0;
58. b = head;
59. p = head;
60. printf("请输入要删除的书籍名：");
61. fflush(stdin);
62. scanf("%s", tmp);
63. //遍历链表
64. while (p != NULL)
65. {
66. if (strcmp(p->bname, tmp) == 0)
67. {
68. flag = 1;
69. break;
70. }
71. p = p->next;
72. }
73. if (flag == 1)
74. {
75. for (; b->next != p;)
76. b = b->next;
77. b->next = p->next;
78. free(p);
79. printf("删除成功！\n");
80. }
81. else
82. printf("该书不存在！");
83. return;
84. }
85. //浏览书单
86. void Print(BookInfo\* head)
87. {
88. BookInfo \*p;
89. if (head == NULL || head->next == NULL) //判断链表是否为空
90. {
91. printf("无记录！\n");
92. return;
93. }
94. p = head;
95. printf("┏━━━┳━━━━━┳━━━━━┳━━━━━┳━━━━┓\n");
96. printf("┃ 编号 ┃ 书名 ┃ 作者 ┃ 类别编号 ┃ 价格 ┃\n");
97. printf("┣━━━╋━━━━━╋━━━━━╋━━━━━╋━━━━┫\n");
98. //遍历链表，输出书籍信息
99. while (p->next != NULL)
100. {
101. p = p->next;
102. printf("┃%-6s┃%-10s┃%-10s┃%-10s┃%.2lf ┃\n", p->bnum,
103. p->bname, p->bauthor, p->bclassfy, p->bprice);
104. printf("┗━━━┻━━━━━┻━━━━━┻━━━━━┻━━━━┛\n");
105. }
106. }
107. //查找书籍
108. void Search(BookInfo\* head)
109. {
110. BookInfo \*p;
111. char tmp[30];
112. int flag = 0;
113. p = head;
114. if (head == NULL || head->next == NULL)
115. printf("清单为空！\n");
116. else
117. {
118. printf("请输入书名：");
119. fflush(stdin);
120. scanf("%s", tmp);
121. while (p->next != NULL)
122. {
123. p = p->next;
124. if (strcmp(p->bname, tmp) == 0)
125. {
126. flag = 1; //书籍已找到
127. printf("编号：%s\n书名：《%s》\n作者：%s\n分类：%s\n价格：%.2f\n",
128. p->bnum, p->bname, p->bauthor, p->bclassfy, p->bprice);
129. return;
130. }
131. if (p->next == NULL)
132. printf("\n查询完毕！");
133. }
134. if (flag == 0)
135. printf("没有找到《%s》！\n", tmp);
136. }
137. return;
138. }
139. //修改信息
140. void Update(BookInfo\* head)
141. {
142. BookInfo \*p;
143. int flag = 0;
144. char tmp[30];
145. p = head;
146. printf("请输入书名：");
147. fflush(stdin);
148. scanf("%s", tmp);
149. while (p->next != NULL)
150. {
151. p = p->next;
152. if (strcmp(p->bname, tmp) == 0)
153. {
154. flag = 1; //标志找到所要修改的书籍
155. printf("请输入编号：");
156. fflush(stdin);
157. scanf("%s", p->bnum);
158. printf("请输入书名：");
159. fflush(stdin);
160. scanf("%s", p->bname);
161. printf("请输入作者：");
162. fflush(stdin);
163. scanf("%s", p->bauthor);
164. printf("请输入类别编号：");
165. fflush(stdin);
166. scanf("%s", p->bclassfy);
167. printf("请输入价格：");
168. fflush(stdin);
169. scanf("%f", &p->bprice);
170. }
171. }
172. if (flag == 0)
173. printf("没有找到《%s》！\n", tmp);
174. return;
175. }
176. //保存书单到文件
177. void Save(BookInfo\* head)
178. {
179. BookInfo \*p;
180. FILE \*fp;
181. p = head;
182. //以只写的方式打开文件
183. fp = fopen("C:\\Users\\admin\\Desktop\\bookslist.txt", "w");
184. fprintf(fp, "┏━━━┳━━━━━┳━━━━━┳━━━━━┳━━━━┓\n");
185. fprintf(fp, "┃ 编号 ┃ 书名 ┃ 作者 ┃ 类别编号 ┃ 价格 ┃\n");
186. fprintf(fp, "┣━━━╋━━━━━╋━━━━━╋━━━━━╋━━━━┫\n");
187. while (p->next != NULL)
188. {
189. p = p->next;
190. fprintf(fp, "┃%-6s┃%-10s┃%-10s┃%-10s┃%.2lf ┃\n",
191. p->bnum, p->bname, p->bauthor, p->bclassfy, p->bprice);
192. fprintf(fp, "┗━━━┻━━━━━┻━━━━━┻━━━━━┻━━━━┛\n");
193. }
194. fclose(fp);
195. printf("保存成功！\n");
196. printf("数据已成功保存到C:\\Users\\admin\\Desktop\\bookslist.txt\n");
197. }
198. //菜单
199. int menu()
200. {
201. int sec;
202. printf(" 图书管理系统 \n");
203. printf("━━━━━━━━━━━━━━━━━━ \n");
204. printf(" 1-图书信息录入\n");
205. printf(" 2-图书信息浏览\n");
206. printf(" 3-图书信息查询\n");
207. printf(" 4-图书信息修改\n");
208. printf(" 5-图书信息删除\n");
209. printf(" 6-图书信息保存\n");
210. printf(" 7-退出\n");
211. printf("━━━━━━━━━━━━━━━━━━ \n");
212. printf("请选择：");
213. fflush(stdin);
214. scanf("%d", &sec);
215. while (sec > 7 || sec < 0)
216. {
217. printf("选择有误！\n请重新输入：");
218. scanf("%d", &sec);
219. }
220. return sec;
221. }

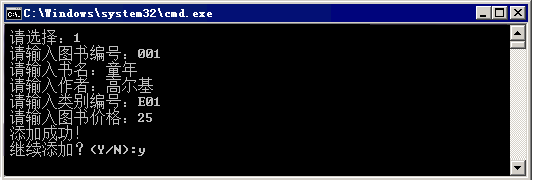
main.c//测试文件

1. #include "StuManage.h"
2. #include <Windows.h>
3. int main()
4. {
5. BookInfo \*head;
6. int sel;
7. head = NULL;
8. for (;;)
9. {
10. sel = menu(); //输出菜单，并获取选择的功能
11. switch (sel)
12. {
13. case 1:
14. if (head == NULL)
15. head = CreateBooksList();
16. Insert(head);
17. break;
18. case 2:Print(head); break;
19. case 3:Search(head); break;
20. case 4:Update(head); break;
21. case 5:Delete(head); break;
22. case 6:Save(head); break;
23. case 7:exit(0); break;
24. default:break;
25. }
26. }
27. return 0;
28. }

程序的运行结果如图10-13和图10-14所示。



1. 【案例5】运行结果——menu()函数



1. 【案例5】运行结果——图书信息录入

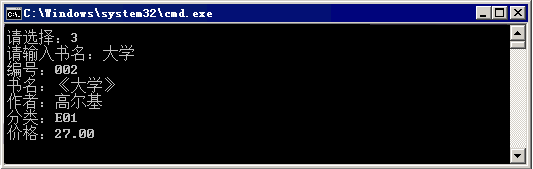
如图10-15所示，在一本书的信息添加成功之后，程序会询问用户是否继续添加，若用户选择“Y”或“y”，程序将继续调用Insert()函数，执行信息录入功能；若用户选择“N”或“n”，程序将返回选择菜单。

输入“y”再添加两条记录，之后输入“n”返回选择菜单，选择功能2，浏览书单，则程序运行结果如图10-15所示。



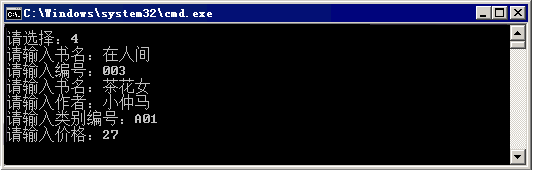
1. 【案例5】运行结果——图书信息浏览

若要查询某条记录，选择功能3，根据提示输入要查询的书名。程序运行结果如图10-16所示。



1. 【案例5】运行结果——图书信息查询

若要修改某条记录，选择功能4，根据提示输入要修改的书名，再依次输入新的信息。程序运行结果如图10-17所示。



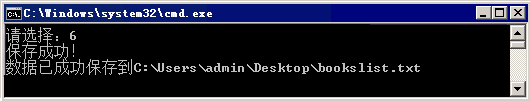
1. 【案例5】运行结果——图书信息修改

若要删除某条记录，选择功能5，根据提示输入要删除的书名，程序将调用函数实现该书面对应记录的信息删除。程序运行结果如图10-18所示。



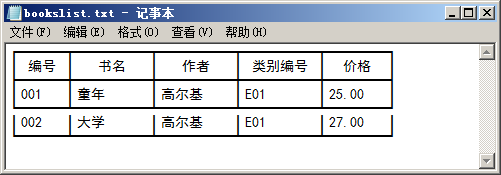
1. 【案例5】运行结果——图书信息删除

若要将链表中的记录保存到文件中，选择功能6，程序将调用Save()函数实现信息的本地存储。程序运行结果如图10-19。



1. 【案例5】运行结果——图书信息保存

此时在桌面上应存在一个名为“bookslist”的文本文件，打开该文件，文件中的内容如图10-20所示。



【案例5】运行结果——图书信息保存