第3章

程序设计结构

本章学习目标

- 了解并掌握程序设计的三种基本结构；
- 理解并掌握算法的概念及其特点；
- 了解 continue 与 break 语句的使用；
- 理解并掌握三种结构的混合应用。

本章主要讲述程序设计的三种基本结构：顺序结构、选择结构和循环结构，以及利用这三种结构完成复杂的程序设计；两个特殊的语句 continue 和 break，需要掌握这两个特殊语句的应用环境以及含义；同时讲述算法的概念以及相关的特点，要求读者对算法有一个明确的认识。

3.1 算法的基本控制结构

所谓的程序就是规定了计算机执行的动作和动作的顺序。一个程序应包括以下两方面的内容：

（1）对数据的描述。在程序中要指定数据的类型和数据在内存中的组织形式，即数据结构。

（2）对操作的描述。即程序的操作步骤，也就是我们常说的算法的概念。

数据是操作的对象，操作的目的是对数据进行加工处理，以得到期望的结果。作为程序设计人员，必须认真考虑并设计数据结构和操作步骤。

1. 算法的概念

算法就是解决问题的步骤序列。对于同一个问题可以有不同的解决方法和步骤，也就是说相同问题可能有多种不同的算法，一般应当从众多的可行的算法中选择简单、精练、运算快且内存开销小的算法。

2. 算法的特点

算法是用来解决问题的方法，一个好的算法应满足以下几个特征：

- 可行性，指的是算法中的每一步都是计算机可以执行的，并能得到有效的结果。

- 确定性，指的是算法中的每一步必须有明确定义，不能让人产生任何歧义。
- 有穷性，指的是算法必须在执行有限步骤后正常结束，而不能是无限地执行下去（即陷入死循环）。
- 至少有一个输出，可以有若干个输入。输入信息就是算法所要加工的对象，输出信息就是算法所解决问题的最终结果。大多数算法需要输入信息，这些输入信息可以是通过键盘输入的数据，也可以是程序其他部分传递给算法的数据。同时，所有算法都至少要有一个输出（明确最终算法的结果）。

3. 算法描述的三种基本结构

对算法的理论研究和实践表明，任何算法的描述都可以分解为三种基本结构或者是它们的组合，这三种基本结构是顺序结构、分支结构（选择结构）和循环结构（重复结构）。下面分别详细介绍这三种基本结构。

3.2 顺序结构

所谓的顺序结构就是按照语句出现的先后顺序依次运行。

【例 3-1】 顺序结构应用案例 1。

题目：要求用户通过键盘输入一直角三角形的底长和高，然后计算出此直角三角形的面积。

```
#include <iostream.h>
void main()
{
  float x,h,area;
  cout <<"please input two numbers:"<< endl;
  cin >> x >> h;
  cout <<"x = "<< x <<", h = "<< h << endl;
  area = 1.0/2 * x * h;
  cout <<"area = "<< area << endl;
}
```

其运行结果如图 3-1 所示。

【例 3-2】 顺序结构应用案例 2。

题目：通过程序设计实现求任意两个实型数据的和。

```
#include <iostream.h>
void main()
{
  float x,y,add;
  cout <<"please input two numbers:"<< endl;
  cin >> x >> y;
  cout <<"x = "<< x <<", y = "<< y << endl;
  add = x + y;
  cout << x <<" + "<< y <<" = "<< add << endl;
}
```

其运行结果如图 3-2 所示。

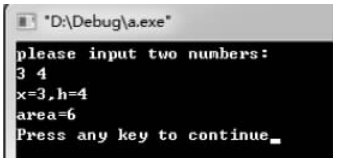

图 3-1　例 3-1 运行结果

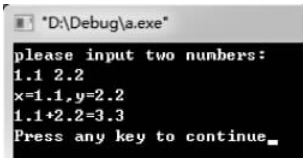

图 3-2　例 3-2 运行结果

3.3　分支结构

分支结构也称选择结构，是通过分支语句来实现的程序设计结构，其执行过程是根据给定条件决定选择哪一条语句来执行。主要细化包括单分支语句、双分支语句和多分支语句。分别用 if 语句、if…else 语句、if…else 的嵌套以及 switch 语句来实现。

3.3.1　单分支结构

在 C++中利用 if 语句来实现单分支结构，if 语句也称为条件语句，其功能是根据给定的条件选择程序的执行方向。if 语句的基本语法格式为：

```
if(表达式)
  语句
```

说明：

(1) if 是 C++语言中的关键字，后面紧邻的是表达式，该表达式可以是 C++中任何合法的表达式。

(2) 计算过程：首先计算表达式，表达式值为非零(真)则执行语句，若表达式值为零(假)则跳过语句，执行 if 语句的后续语句。

(3) 语句要求是一条语句，若一条语句不能完成功能需要多条语句时，则应采用复合语句。

【例 3-3】 单分支结构应用案例 1。

题目：通过键盘输入任意两个整数，输出较大的数。

```
#include <iostream.h>
void main()
{
  int a,b,max;
  cout <<"请输入两个数字:";
  cin >> a >> b;
  if(a > b)
    max = a;
  if(a <= b)
    max = b;
  cout <<"两个数中较大的是:"<< max << endl;
}
```

其运行结果如图 3-3 所示。

【例 3-4】 单分支结构应用案例 2。

题目：通过键盘输入任意两个整数，要求第一个数中放大数，第二个数中放小数。

```
#include <iostream.h>
void main()
{
  int a,b,t;
  cout <<"请输入两个数字:";
  cin >> a >> b;
  cout <<"交换前的结果: "<< a <<","<< b << endl;
  if(a < b)
  {
   t = a;
   a = b;
   b = t;
  }
  cout <<"交换后的结果: " << a <<","<< b << endl;
}
```

其运行结果如图 3-4 所示。

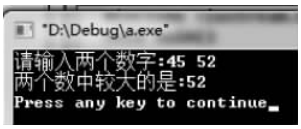

图 3-3 例 3-3 运行结果

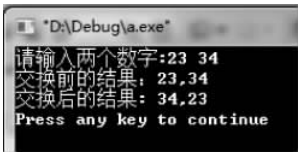

图 3-4 例 3-4 运行结果

3.3.2 双分支结构

在 C++语言中利用 if…else 语句能够实现双分支结构。其语句的基本语法格式为：

```
if(表达式)
    语句 1
else
    语句 2
```

说明：

(1) if…else 是 C++中的关键字，if 后的表达式可以是 C++中任意合法的表达式。

(2) 执行过程：

① 计算表达式，若表达式结果为非零则执行步骤②，否则执行步骤③。

② 执行语句 1，接着执行步骤④。

③ 执行语句 2，接着执行步骤④。

④ 执行分支语句的后续语句。

(3) 语句 1、语句 2 要求是一条语句，若一条语句不能把功能完成，则需要多条语句时需要使用复合语句。

(4) else 语句不能单独使用,必须与 if 配对使用。

【例 3-5】 双分支结构应用案例 1。

题目:利用双分支结构改写例 3-3,实现两个数中输出较大数。

```
#include <iostream.h>
void main()
{
  int a,b,max;
  cout <<"请输入两个数字:";
  cin >> a >> b;
  if(a > b)
      max = a;
   else
      max = b;
   cout <<"两个数中较大的是:"<< max << endl;
  }
```

其运行结果如图 3-5 所示。

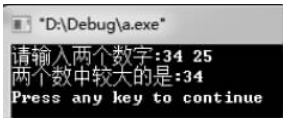

图 3-5 例 3-5 运行结果

【例 3-6】 双分支结构应用案例 2。

题目:通过键盘输入任意一个年份,判断该年份是否为闰年(闰年的条件是:年份可以被 4 整除但是不能被 100 整除,或者年份可以被 400 整除)。

```
#include <iostream.h>
void main()
{
  int year;
  cout <<"请输入一个年份(四位): ";
  cin >> year;
  if((year % 4 == 0 && year % 100!= 0)||(year % 400 == 0))
    cout << year <<"是闰年!"<< endl;
  else
    cout << year <<"不是闰年!"<< endl;
}
```

其运行结果如图 3-6 所示。

图 3-6 例 3-6 运行结果

3.3.3 多分支结构

在 C++语言中多分支结构有两种形式:一种是分支结构中内嵌分支结构,另一种是 switch 语句。

1. 分支结构内嵌的多分支结构

在分支语句中，内嵌的语句可以是任意语句。因此，分支语句中也可以是分支语句，形成分支的嵌套结构，称为嵌套的条件语句。其一般格式为：

```
if(表达式 1)          if(表达式 1)
  if(表达式 2)          if(表达式 2)
     语句 1                   语句 1
                         else
                               语句 2

if(表达式 1)          if(表达式 1)
   语句 1                    语句 1
else                   else
  if(表达式 2)            if(表达式 2)
      语句 2                     语句 2
                           else
                                  语句 3
```

注意：在 C++语言中，只有 if 语句或者 if…else 语句，没有单独的 else 语句，在 C++语言中规定 else 总是与它上面紧邻的没有 else 配对的 if 配对。

【例 3-7】 多分支结构的应用案例 1。

题目：将键盘输入的百分制成绩转换成五级计分制的成绩输出。五级计分制成绩确定规则：'A'(90～100)、'B'(80～89)、'C'(70～79)、'D'(60～69)、'E'(60 分以下，不包括 60)。

```
#include <iostream.h>
void main()
{ int score;
  char grade;
  cout <<"请输入一个分数值(0～100):";
  cin >> score;
  if(score >= 90&& score <= 100)
      grade = 'A';
  else if(score >= 80 && score <= 89)
      grade = 'B';
  else if(score >= 70 && score <= 79)
      grade = 'C';
  else if(score >= 60 && score <= 69)
      grade = 'D';
  else
      grade = 'E';
  cout << score <<"分所处的等级为:"<< grade << endl;
}
```

其运行结果如图 3-7 所示。

(a) (b)

图 3-7 例 3-7 运行结果

【例 3-8】 多分支结构的应用案例 2。

题目：用户通过键盘输入任意一个年份与月份，自动显示该年的当月所包含的天数。

```
#include <iostream.h>
void main()
{
   int year,month,day;
   cout <<"请输入一个年份(四位): ";
   cin >> year;
   cout <<"请输入一个月份: ";
   cin >> month;
   if(month == 1 ||month == 3 ||month == 5 ||month == 7 ||month == 8 ||month == 10 ||month == 12)
       day = 31;
   else if(month == 4||month == 6||month == 9||month == 11)
       day = 30;
   else
    {
      if(year % 4 == 0 &&year % 100!= 0)
        day = 29;
      else
        day = 28;
   }
  cout << year <<"年"<< month <<"月有"<< day <<"天"<< endl;
}
```

其运行结果如图 3-8 所示。

图 3-8 例 3-8 运行结果

2. switch 语句

switch 语句是开关语句，也称为多分支结构。它可以根据给定的条件，从多个分支语句中选择执行其中某一个分支。其语句格式为：

```
switch(表达式)
{
   case 常量表达式 1:[语句序列 1];[break;]
   case 常量表达式 2:[语句序列 2];[break;]
    ⋮
   case 常量表达式 n:[语句序列 n];[break;]
   [default:语句序列]
}
```

说明：

(1) 表达式可以是 C++语言中合法的任意表达式，但是表达式的最终结果必须是整型数据、字符型数据或枚举类型数据。

(2) 常量表达式只能是由字符型常量、整型常量或者枚举类型常量组成的表达式；语句序列是可选的，可以是一条或多条语句组成。

(3) 关键字 break 也是可选的。

(4) default 分支放在开关语句的任何位置，但通常作为开关语句的最后一个分支。default 分支若放在开关语句最后可以省略 break，若放在开关语句的其他位置则后面必须

有 break。

(5) 其执行过程是：先求表达式的值，再依次与 case 后面的常量表达式比较，若与某一常量表达式的值相等，则转去执行该 case 后的语句序列，一直执行到 break 语句或开关语句的右花括号位置。如果表达式的值与 case 后的任意一个常量表达式的值均不相等，则看是否有 default 分支，有则执行该分支语句，没有则什么都不做，结束开关语句。

注意：

- 当省略 case 后面的语句序列时，则可实现多个入口，执行同一语句序列。
- case 与后面的常量表达式之间要有空格。
- case 后的常量不能相同，但是顺序是任意的。
- case 后面的语句可以是多条语句，这些语句可以不用{}括起来。

【例 3-9】 多分支结构应用案例 3。

题目：修改例 3-7，利用开关语句实现成绩的等级。

```
#include <iostream.h>
void main()
{ int score;
  char grade;
  cout <<"please input a score:";
  cin >> score;
  switch(score/10)
  {
    case 10:
    case 9:grade = 'A';break;     //若 score/10 结果为 10,则执行 case 9 后的语句序列
    case 8:grade = 'B';break;
    case 7:grade = 'C';break;
    case 6:grade = 'D';break;
    default:grade = 'E';
  }
 cout <<"the grade of score is:"<< grade << endl;
}
```

其运行结果如图 3-9 所示。

该例题中，读者可以去掉程序中的 break，演示一下程序，看看运行结果，试着理解 break 在此结构中的作用。

【例 3-10】 多分支结构应用案例 4。

题目：设计一个小型计算器，能够实现加、减、乘、除和乘方的运算。

```
#include <iostream.h>
#include <math.h>
void main()
{
  float x1,x2;
  char op;
  cout <<"请输入两个数值:";
  cin >> x1 >> x2;
  cout <<"请输入一个运算符:";
  cin >> op;
```

```
  switch(op)                          //字符类型表达式
   {
     case '+':cout << x1 + x2 << endl;break;
     case '-':cout << x1 - x2 << endl;break;
     case '*':cout << x1 * x2 << endl;break;
     case '/':cout << x1/x2 << endl;break;
     case '^':cout << pow(x1,x2)<< endl;break;
     default:cout <<"the error of operator!"<< endl;
    }
 }
```

其运行结果如图 3-10 所示。

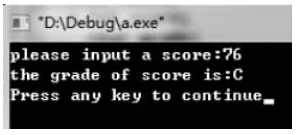

图 3-9　例 3-9 运行结果

图 3-10　例 3-10 运行结果

【例 3-11】 多分支结构应用案例 5。

题目：应用枚举类型值进行输入值的判断，通过输入 0 显示 male，输入 1 显示 female。

```
#include <iostream.h>
void main()
{
 enum sex{male,female}s;
 int n;
 cout <<"请输入一个整数(0—male,1—female): ";
 cin >> n;
 switch(n)
 {
   case 0:s = male;break;                          //0 对应 male,1 对应 female
   case 1:s = female;break;
   default:cout <<"您的输入错误!\n";
 }
 switch(s)
  {
    case male:cout <<"male\n";break;               //注意 break 语句的使用
    case female:cout <<"female\n";
  }
}
```

其运行结果如图 3-11 所示。

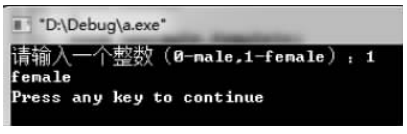

图 3-11　例 3-11 运行结果

3.4 循环结构

循环结构也称为重复结构，其语法要求是在一定的条件下反复执行同一动作，直到该条件失效。在C++语言中，循环结构包括for语句、while语句和do…while语句。

3.4.1 for语句

for语句的语法格式为：

```
for(表达式1;表达式2;表达式3)
    循环体
```

说明：

(1) for是C++语言中的关键词，不能省略。

(2) 表达式1、表达式2和表达式3可以是C++任意合法的表达式，这三个表达式均可以省略，但是分号不允许省略；循环体原则上要求是一条语句，若需要多条语句实现功能时需要采用复合语句。

(3) 其执行过程如下：

① 计算表达式1；

② 计算表达式2，若表达式2的结果为非0，则执行步骤③，否则转向步骤④；

③ 执行语句，计算表达式3，转到步骤②；

④ 结束循环，执行for语句的后续语句。

【例3-12】 循环结构应用案例1。

题目：用for循环实现求1～100之间所有偶数的和。

```
#include <iostream.h>
void main()
{
  int i = 0,sum = 0;
  for(;i <= 100;)                          //表达式1和表达式3省略
    {
      sum += i;
      i = i + 2;
    }
  cout <<"sum = "<< sum << endl;
}
```

其运行结果如图3-12所示。

【例3-13】 循环结构应用案例2。

题目：输出所有的“水仙花数”。所谓“水仙花数”是指一个三位数，其各位数字立方和等于该数本身。(例如，$1^3+5^3+3^3=153$，153是水仙花数)

```
#include <iostream.h>
void main()
```

```
{
   int i,j,k,n;
   cout<<"水仙花数有: ";
   for(n=100;n<1000;n++)
    {
      i=n/100;
      j=n/10-i*10;
      k=n%10;
      if(n==i*i*i+j*j*j+k*k*k)
          cout<<n<<" ";
      }
   cout<<endl;
}
```

其运行结果如图 3-13 所示。

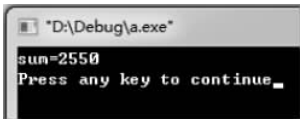

图 3-12　例 3-12 运行结果

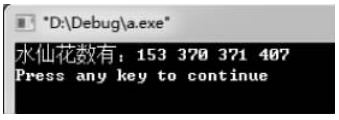

图 3-13　例 3-13 运行结果

3.4.2　while 语句

while 语句的格式:

```
while(表达式)
    循环体
```

说明:

(1) while 是 C++语言中的关键词,不能省略。

(2) 表达式是 C++中任意合法表达式;循环体是 C++中任意语句,如果是由多条语句组成的,需要用复合语句实现。

(3) 执行过程如下:

① 计算表达式,若表达式结果为非 0,则执行步骤②,否则执行步骤③。

② 执行循环体。

③ 停止循环,执行循环语句的后续语句。

【例 3-14】　循环结构应用案例 3。

题目:用 while 循环实现求 1～100 之间所有偶数的和。

```
#include<iostream.h>
void main()
{
  int i=0,sum=0;
  while(i<=100)
    {
      sum+=i;
      i=i+2;
```

```
    }
  cout <<"sum = "<< sum << endl;
}
```

其运行结果如图 3-14 所示。

【例 3-15】 循环结构应用案例 4。

题目：编写程序，实现求任意两个正整数的最大公约数和最小公倍数。

```
#include < iostream.h >
void main()
{
 int m,n,r,temp,p;
 cout <<"请输入两个数值:";
 cin >> m >> n;
 if(m < n)
 {
     temp = m;
     m = n;
     n = temp;
 }
 p = m * n;
 while(n!= 0)
  {
    r = m % n;
    m = n;
    n = r;
  }
 cout <<"最大公约数是: "<< m << endl;
 cout <<"最小公倍数是: "<< p/m << endl;
}
```

其运行结果如图 3-15 所示。

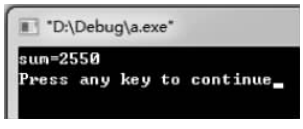

图 3-14 例 3-14 运行结果

图 3-15 例 3-15 运行结果

3.4.3 do…while 语句

do…while 语句的语法格式为：

```
do
   循环体
while(表达式);
```

说明：

(1) do 和 while 是 C++语言中的关键字，不能省略；同时表达式后面的分号不能省略。

(2) 表达式可以是C++中任意合法表达式,循环体要求是一条语句,若需要多条语句时,需要使用复合语句。

(3) 其执行过程如下:

① 进入循环开始执行循环体。

② 计算表达式,若表达式结果为非0,则执行步骤①,否则执行步骤③。

③ 退出循环,执行循环以后的后续语句。

【例 3-16】 循环结构应用案例5。

题目:用do…while循环结构实现求1～100之间所有偶数的和。

```
#include <iostream.h>
void main()
{
  int i = 0,sum = 0;
  do
  {
    sum += i;
    i = i + 2;
   }while(i <= 100);
   cout <<"sum = "<< sum << endl;
}
```

其运行结果如图3-16所示。

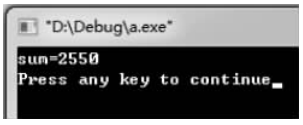

图3-16 例3-16运行结果

【例 3-17】 循环结构应用案例6。

题目:制作一个小游戏,要求:系统自动生成0～50之间的随机数x,用户去猜其具体的数值。

要求:

① 若用户猜的数值大于该数,则提示大于该数。

② 若用户猜的数值小于该数,则提示小于该数。

```
#include <iostream.h>
#include <stdlib.h>
void main()
{
  int min = 0,max = 50;
  int x,y;
  x = rand() % 50;
  cout <<"系统已经生成随机数(0～50),请您输入您猜测的数据: ";
  do
  {
    cin >> y;
    if(y > x)
      {
        max = y;
        cout <<"当前数值范围为: "<< min <<" -- "<< max << endl;
      }
    else if(y < x)
      {
```

```
            min = y;
            cout <<"当前数值范围为: "<< min <<" -- "<< max << endl;
        }
      else
          cout <<"您猜对了,您非常棒!"<< endl;
    }while(true);
}
```

其运行结果如图 3-17 所示。

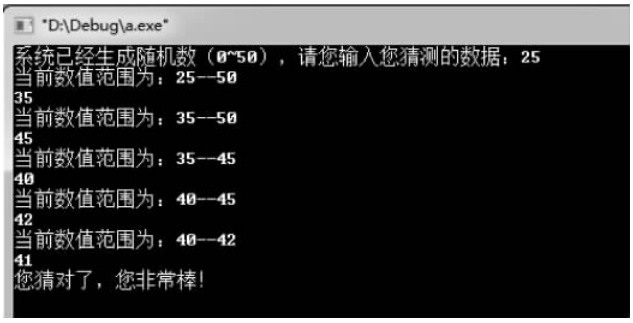

图 3-17 例 3-17 运行结果

总之,通过例 3-12、例 3-14、例 3-16 可以看出,在某些情况下,三种循环语句 for、while 和 do…while 是可以互相替换的。

3.5 其他控制语句

goto 语句也称为无条件转向语句,它可以将程序的执行流程转到程序中的任意位置,通常是从它所在的地方转移到带有标号的语句处。goto 语句与条件语句组合,可形成当型循环和直到型循环。但是对于规模庞大的程序来说,无限制地使用 goto 语句,则会导致程序流程过于复杂,程序跳转混乱,降低程序的可读性和可维护性等。因此,在 C++语言中又提供了功能受到限制的转向语句 break 和 continue 来替代 goto 语句。

1. break 语句

break 语句的语法格式为:

```
break;
```

说明:

(1) break 是 C++语言中的关键词,该语句只用在 switch 或循环语句中。

(2) break 语句用在开关语句 switch 中的某个分支语句中,其作用是结束开关语句的执行,并把控制转移到该开关语句之后的第一个语句执行。break 语句用在循环语句的循环体中,当执行到 break 语句时,直接结束该循环语句的执行,把控制转移到紧跟该循环语句之后的语句执行,具体案例在循环结构中介绍。

【例 3-18】 break 语句的应用案例。

题目：编程实现模拟 ATM 机的执行流程。

```
#include <iostream.h>
#include <vector>
void main()
{
  int password,Id;
  cout <<" ********** 进入自动提款系统 ********** "<< endl;
  cout <<"\n 请输入密码：";
  cin >> password;
  if(password == 142536)
      cout <<"\n 欢迎您使用 ATM 系统,请按键选择您所需要的服务"<< endl;
  else
  {
    cout <<"\n 您的密码错误,请重新输入"<< endl;
    exit(1);
   }
 cout <<"\n 1：查询"<< endl;
 cout <<"\n 2：取款"<< endl;
 cout <<"\n 3：存款"<< endl;
 cout <<"\n 4：退出"<< endl;
 cout <<"\n 请输入您的选择：";
 cin >> Id;
 switch(Id)
 {
   case 1：cout <<"进行查询操作中……"<< endl;break;
   case 2：cout <<"进行取款操作中……"<< endl;break;
   case 3：cout <<"进行存款操作中……"<< endl;break;
   case 4：exit(1);
   default:cout <<"您的输入有误!"<< endl;
 }
}
```

其运行结果如图 3-18 所示。

图 3-18　例 3-18 运行结果

2. continue 语句

continue 语句的格式：

```
continue;
```

说明：

(1) continue 语句只用在循环语句的循环体中，用于结束本次循环的循环体，提前进入下一次循环。

(2) 对于 while 和 do…while 循环来说，若遇到 continue 语句，则跳到该循环的表达式的位置；而对于 for 循环来说，则跳到该循环的表达式处。

【例 3-19】 continue 语句的应用案例。

题目：continue 语句应用在循环语句中，验证 continue 语句的功能。

```
#include <iostream.h>
void main()
{
  int x = 1,n = 10;
  while(n-- >= 0)
  {
    if(x > 4)
      continue;                          //若 x>4 成立则结束本次循环做下一循环
    cout << x++<<" ";
  }
  cout << endl;
  cout <<"x = "<< x <<",n = "<< n << endl;          //注意 n 的值
}
```

其运行结果如图 3-19 所示。

图 3-19 例 3-19 运行结果

3. goto 语句

goto 语句的语法格式为：

```
goto 语句标号;
```

说明：

(1) 语句标号是采用标识符来标识程序中某一条语句的，标号无须定义可以直接使用。其格式为：

语句标号：C++语句；

(2) C++语句可以是任意合法的语句，包括空语句。

(3) goto 语句的执行，当程序执行到该语句时，无条件地转移到标有语句标号的位置处执行。goto 语句主要有以下两种用途：

- 从循环体内转移到循环体外，但可用 break 和 continue 替代。只是需要从多层循环体内跳到外层循环体外时才用到 goto 语句。但是这种语法不符合结构化程序设计原则，不提倡使用。

注意：*不允许从循环语句的外层转移到循环语句的内层。*

- 与 if 语句一起构成循环。

【例 3-20】 goto 语句的应用案例。

题目：利用 goto 语句实现求 1～100 之内偶数的和。

```
#include <iostream.h>
void main()
{
  int i,sum = 0;
  i = 0;
  a:i = i + 2;
  sum += i;
  if(i<100)
     goto a;
  cout <<"sum = "<< sum << endl;
}
```

其运行结果如图 3-20 所示。

注意：if 后的条件表达式。

图 3-20　例 3-20 运行结果

4. exit( )函数

exit 函数是 C++标准库 cstdlib 中的函数，其函数原型为：

```
void exit(int status);
```

说明：

(1) 函数功能：执行该函数时，将终止当前程序的执行并将控制权返还给操作系统。

(2) status 为终止程序的原因，0 表示正常退出，非 0 表示异常退出。

具体应用可以参见例 3-18，这里就不再重复举例了。

3.6　多种结构的嵌套

do…while 语句、for 语句和 while 语句都是循环语句，它们之间在某些条件下是可以相通的。首先对三种循环语句简单做一比较：

- for 和 while 语句都是先判断循环条件，循环体有可能会执行若干次，也可能一次都不执行。而 do…while 语句是先执行循环体，后判断循环条件，所以循环体至少要执行一次。因此，对于至少要执行一次循环的程序段，需要使用 do…while 语句，而对于其他的循环结构的程序段，可以使用 for 和 while 语句。
- 由于 for 语句有三个表达式，可分别用于循环变量初始化、循环结束条件和循环控制变量的更新，所以用起来更加清晰、明了。其次是 while 语句，而 do…while 语句相对于前两种语句用得相对较少一些。
- 由于循环的内嵌语句可以使用 C++语句中的任意合法语句，因此，循环语句的内嵌语句也可以是一个循环语句，这种情况称为循环的嵌套。

【例 3-21】 结构嵌套的应用案例 1。

题目：若一个数恰好等于它的因子之和，则这个数称为完数。编写程序输出 100 以内

的所有完数。(如 1+2+3=6,而 1、2、3 是 6 的因子,所以说 6 是完数)

```
#include <iostream.h>
void main()
{
   int i,j,s;
   for(i=2;i<=100;i++)
    {
      s=0;
      for(j=1;j<i;j++)
        if(i%j==0)
          s+=j;
      if(s==i)
        cout<<i<<"是完数。"<<endl;
   }
}
```

其运行结果如图 3-21 所示。

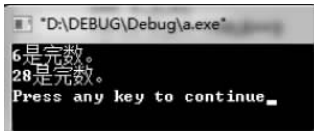

图 3-21 例 3-21 运行结果

【例 3-22】 结构嵌套的应用案例 2。

题目:求 15 个学生英语课程的平均分。

```
#include <iostream.h>
void main()
{
  int i;
  float sum=0,ave,score[15];
  cout<<"请输入 15 个学生的高数成绩:";
  for(i=0;i<15;i++)
     cin>>score[i];
  for(i=0;i<15;i++)
    sum+=score[i];
  ave=sum/15;
  cout<<"这 15 个学生高数课程的平均分为:"<<ave<<endl;
}
```

其运行结果如图 3-22 所示。

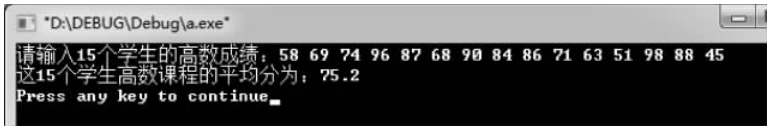

图 3-22 例 3-22 运行结果

【例 3-23】 结构嵌套的应用案例 3。

题目:结构嵌套中 break 和 continue 语句的应用。

```
#include <iostream.h>
void main()
{
  int i,x=1,y=0;
  for(i=0;i<10;i++)
```

```
    {
      x += 3;
      if(x > 5)
        {
          cout <<" ** x = "<< x <<"y = "<< y << endl;
          continue;
        }
      y = x + 5;
      cout <<" x = "<< x <<" y = "<< y << endl;
    }
}
```

其运行结果如图 3-23 所示。

若将上例中的 continue 语句改成 break 语句，则运行结果如图 3-24 所示。

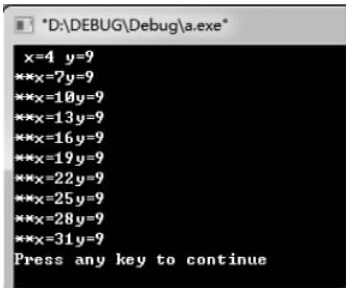

图 3-23 例 3-23 运行结果

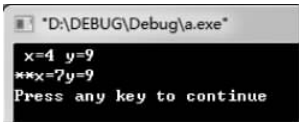

图 3-24 将 continue 语句改成 break 语句后的运行结果

【例 3-24】 结构嵌套的应用案例 4。

题目：有 n 个数，已按由小到大顺序排列好，要求输入一个数，把它插入到原有数列中，而且仍然保持有序，同时输出新的数列。

分析：通常插入算法应包含以下 4 个主要步骤：

(1) 确定插入位置。

(2) 把从最后一个元素到插入位置的每一个元素中的值，依次向后移动一个位置，即把 a[n]中的值放入 a[n+1]中，把 a[n-1]中的值放入 a[n]中，以此类推，直到把 a[i]中的值放入 a[i+1]中。

(3) 在确定的插入位置上放入 x 的值。

(4) 元素的个数增 1。

```
#include < iostream.h >
void main()
{
  int i,n,j;
  int a[11] = {12,27,35,41,53,67,74,80,96,100};
  cout <<"原数列为：" << endl;
  for(i = 0;i < 10;i++)
     cout << a[i]<<'\t';
  cout << endl;
```

```
  cout <<"输入插入数: "<< endl;
  cin >> n;
  j = 9;
  while(j > = 0&&n < a[j])
   {
     a[j + 1] = a[j];
     j -- ;
   }
  a[j + 1] = n;
  cout <<"插入后的数组: "<< endl;
  for(i = 0;i < 11;i++)
     cout << a[i]<<'\t';
  cout << endl;
}
```

其运行结果如图 3-25 所示。

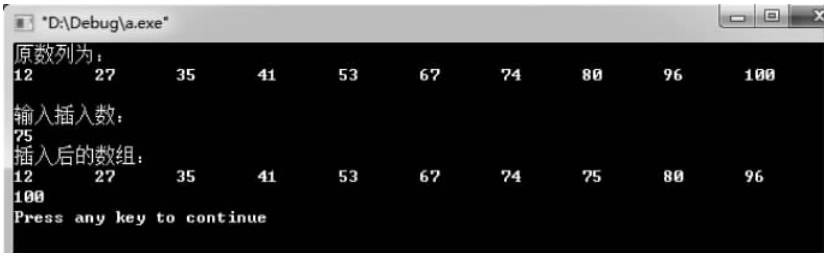

图 3-25 例 3-24 运行结果

3.7 综合案例——公司人员管理系统 3

在公司人员管理系统中,涉及员工的添加、删除、修改以及查询等操作。下面分别来描述各个功能的实现。

```
//员工的添加
void Company::add()
{
  Person * p;
  int duty;
  char Name[10];
  double Amount,T;
  cout <<"\n -------------- 新增员工 --------------- "<< endl;
  ++ID;
  cout <<"输入岗位信息(1 - 公司经理,2 - 销售经理,3 - 销售员,4 - 技术员): ";
  cin >> duty;
  cout <<"输入姓名: ";
  cin >> Name;
  if(duty == 3)
  {
    cout <<"本月销售额: ";
```

```
    cin >> Amount;
  }
  else if(duty == 4)
  {
    cout <<"本月工作时间(0 - 168 小时): ";
    cin >> T;
  }
switch(duty)
{
  case 1:p = new Manager(ID,Name,duty);break;
  case 2:p = new SalesManager(ID,Name,duty);break;
  case 3:p = new Sales(ID,Name,duty,Amount);break;
  case 4:p = new Technician(ID,Name,duty,T);break;
}
p -> next = 0;
if(Worker)                                          //若节点已经存在
{
   Person * p2;
   p2 = Worker;
   while(p2 -> next)
   {
      p2 = p2 -> next;
    }
    p2 -> next = p;                                 //连接节点
}
else
{
   Worker = p;
}
}
//员工的删除
void Company::delet()
{
  int No;
  cout <<"\n------------ 删除员工 ------------\n";
  cout <<"ID: ";
  cin >> No;
  Person * p1, * p2;
  p1 = Worker;
  while(p1)
  {
    if(p1 -> No == No)
      break;
    else
    {
      p2 = p1;
      p1 = p1 -> next;
     }
  }
  if(p1!= NULL)
  {
```

```
 if(p1 == Worker)
 {
   Worker = p1 -> next;
   delete p1;
 }
 else
 {
   p2 -> next = p1 -> next;
   delete p1;
  }
  cout <<"找到该员工信息并删除\n";
  }
  else
  cout <<"未找到!!!\n";
  }
//员工信息的修改
void Company::modify()
{
  int No,duty;
  char Name[10];
  double Amount,T;
  cout <<"\n---------- 修改员工信息 ----------\n";
  cout <<"ID: ";
  cin >> No;
  Person * p1, * p2;
  p1 = Worker;
  while(p1)
  {
    if(p1 -> No == No)
      break;
    else
    {
     p2 = p1;
     p1 = p1 -> next;
   }
  }
  if(p1!= NULL)
  {
    p1 -> output();
    cout <<"调整岗位(1 - 公司经理,2 - 销售经理,3 - 销售员,4 - 技术员): ";
    cin >> duty;
    if(p1 -> duty!= duty)
    {
     cout <<"输入姓名: ";
cin >> Name;
if(duty == 3)
{
   cout <<"本月销售额: ";
   cin >> Amount;
 }
else if(duty == 4)
```

```
{
  cout <<"本月工作时间(0 - 168 小时): ";
  cin >> T;
}
Person * p3;
switch(duty)
{
  case 1:p3 = new Manager(p1 - > No,Name,duty);break;
  case 2:p3 = new SalesManager(p1 - > No,Name,duty);break;
  case 3:p3 = new Sales(p1 - > No,Name,duty,Amount);break;
  case 4:p3 = new Technician(p1 - > No,Name,duty,T);break;
}
p3 - > next = p1 - > next;
if(p1 == Worker)
   Worker = p3;
else
   p2 - > next = p3;
delete p1;
}
else
{
 cout <<"输入姓名: ";
 cin >> p1 - > Name;
 if(duty == 3)
 {
  cout <<"本月销售额: ";
  cin >> Amount;
  ((Sales * )p1) - > setAmount(Amount);
 }
 else if(duty == 4)
 {
  cout <<"本月工作时间(0～168 小时): ";
  cin >> T;
  ((Technician * )p1) - > setT(T);}
 }
cout <<"修改成功!\n";
}
else
  cout <<"未找到该员工!"<< endl;
}
//查询员工信息
void Company::query()
{
  double sum = 0;
  cout <<" ---------- 查询员工本月销售信息 -------- \n";
  Person * p = Worker;
  while(p)
  {
   if(p - > duty == 3)
     sum += ((Sales * )p) - > getAmount();
   p = p - > next;
```

```
  }
p = Worker;
double sum2 = 0;
while(p)
{
   if(p -> duty == 2)
      ( (SalesManager * )p) -> setAmount(sum);
   p -> output();
   sum2 += p -> earning;
   p = p -> next;
}
cout <<"本月盈利: "<< sum * 0.20 - sum2 << endl;
cout <<"按照 20 % 利润计算\n";
}
```

3.8 小结

本章重点介绍了编程中所需要的三种基本结构：顺序结构、选择结构以及循环结构，这三种结构的嵌套可以解决任何复杂的问题。选择结构还细分为单分支结构、双分支结构以及多分支结构，需要注意的是双分支结构中一般在 else 子句中嵌套单分支或者多分支，程序结构更清晰；switch 语句的多分支结构在使用过程中要注意的是 break 语句的使用。在循环结构中有三种：for 语句、while 语句以及 do…while 语句，每一种语句都有自己的特点。此外，还介绍了一些特殊的语句：break 语句、continue 语句以及 goto 语句，由于 goto 语句为无条件转向语句，在程序中出现得多了会引起程序运行的混乱，所以尽量避免使用该语句。

习题 3

1. 输入某学生成绩，若成绩为 85 分以上，则输出 very good；若成绩为 60～85 分，则输出 good；若成绩低于 60 分，则输出 no good。

2. 输入三个整数，按从小到大的顺序输出它们的值。

3. 输入三角形的三条边，判别它们能否形成三角形，若能，则判断是等边、等腰三角形，还是一般三角形。

4. 输入百分制成绩，并把它转换成五级分制，转换公式为：

$$\text{grade(级别)}=\begin{cases}\text{A(优秀)} & 90\sim100\\ \text{B(良好)} & 80\sim89\\ \text{C(中等)} & 70\sim79\\ \text{D(合格)} & 60\sim69\end{cases}$$

5. 求 1000 以内的所有完数。所谓完数，是指一个数恰好等于它的所有因子之和。例如，因为 6＝1＋2＋3，所以 6 为完数。