第3章 C语言的控制结构

通过前两章的学习，我们了解和掌握了面向结构程序设计的思想及基础知识。在本章中将介绍结构化程序设计的3种基本结构，即顺序结构、选择结构和循环结构，重点是实现选择结构和循环结构的程序设计方法。

本章要点

- 理解结构化程序设计的思想及方法。
- 掌握实现选择结构的程序设计方法：if、switch语句。
- 掌握实现循环结构的程序设计方法：while、do…while、for语句。

3.1 结构化程序设计

面向结构程序设计由E. W. dijkstra在1969年提出，是以模块化设计为中心，将待开发的软件系统划分为若干个相互独立的模块，这样使完成每一个模块的工作变得简单而明确，为设计一些大型软件打下了良好的基础。面向结构程序设计是一种程序设计方法，用3种基本的控制结构通过组合和嵌套实现任何单入口、单出口的程序——这也是面向结构程序设计基本原理。这3种程序控制结构分别是顺序结构、选择结构和循环结构。

3.1.1 结构化程序设计的方法

1. 自顶向下

在进行软件系统设计时应先考虑总体，后考虑细节；先考虑全局目标，后考虑局部目标，不要一开始就过多追求细节，要先从最上层总目标开始设计，逐步使问题具体化。

2. 逐步细化

对于复杂问题，应设计一些子目标作为过渡，逐步细化。

3. 模块化设计

一个复杂问题肯定是由若干个稍简单的问题构成的。模块化是把程序要解决的总目标分解为多个子目标，再进一步分解为具体的小目标，每一个小目标称为一个模块。

3.1.2 结构化程序设计的步骤

如果要开发一个软件系统,不论这个系统的规模有多大,我们通常需要按照下述规定流程进行开发与实现。

1. 分析问题

对于要解决的问题,首先必须分析清楚,明确题目的要求,列出所有已知量,找出题目的求解范围、解的精度等。

通过分析,获取问题所涉及的已有数据和最终要得到的数据。

2. 建立数学模型

在对实际问题进行分析之后,找出数据间的内在规律,在已有数据和最终要得到的数据之间建立数学模型(即数学表达式),可以用数学方法来解决该问题,最终才能利用计算机来解决。

3. 设计算法

在建立数学模型后还不能着手编写程序,必须根据数据的数据结构设计解决问题的算法(即解题步骤)。在选择算法时一般要注意以下几点:

(1) 算法的逻辑结构尽可能简单;

(2) 算法所要求的存储量应尽可能少,即算法的空间复杂度尽可能小;

(3) 避免不必要的循环和递归,减少算法的执行时间,即算法的时间复杂度尽可能小;

(4) 在满足题目条件要求下使所需的计算量最小。

4. 编写程序

采用某种计算机语言,将前面所涉及的数据和算法进行详细的描述;把整个程序看作一个整体,先全局后局部,自顶向下,一层一层分解处理,如果某些子问题的算法相同而参数不同,可以用子程序来表示。

5. 调试运行

将整个程序编译、调试后运行程序得出结论。

6. 程序测试

根据运行结果分析程序,通过几组数据验证程序的正确性。

7. 写出程序的文档

主要是对程序中的变量、函数或过程做必要的说明,解释编程思路,画出框图,讨论运行结果等。

3.1.3　结构化程序设计的特点

结构化程序设计中的3种基本结构都遵循具有唯一入口和唯一出口的原则，并且程序不会出现死循环，在程序的静态形式与动态执行流程之间具有良好的对应关系。

1. 优点

由于模块相互独立，因此在设计其中一个模块时不会受到其他模块的牵连，故可将原来较为复杂的问题简化为一系列简单模块的设计。模块的独立性还为扩充已有的系统、建立新系统带来不少的方便，因为可以充分利用现有的模块做积木式的扩展。

按照面向结构程序设计的观点，任何算法功能都可以通过由程序模块组成的3种基本程序结构的组合来实现。

结构化程序设计的基本思想是采用“自顶向下，逐步求精”的程序设计方法和“单入口、单出口”的控制结构。自顶向下、逐步求精的程序设计方法从问题本身开始，经过逐步细化，将解决问题的步骤分解为由基本程序结构模块组成的结构化程序框图；“单入口、单出口”的思想认为一个复杂的程序，如果它仅是由顺序、选择和循环3种基本程序结构通过组合、嵌套构成，那么这个新构造的程序一定是一个单入口、单出口的程序，据此很容易编写出结构良好、易于调试的程序。

2. 缺点

(1) 用户要求难以在系统分析阶段准确定义，致使系统在交付使用时产生许多问题。

(2) 用系统开发每个阶段的成果进行控制，不能适应事物变化的要求。

(3) 系统开发周期长。

3.1.4　结构化程序设计的3种基本控制结构

1. 顺序结构

顺序结构表示程序中的各语句按照它们出现的先后顺序依次被执行，每个语句都被执行且只执行一次，如图3-1所示。

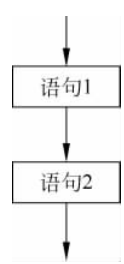

图3-1　顺序结构程序设计流程图

说明：执行完语句1即开始执行语句2。

2. 选择结构

选择结构表示程序的处理步骤出现了分支，它需要根据某一特定的判断条件的结果选择其中的一个分支执行。选择结构有单选择、双选择和多选择3种形式，如图3-2所示。

说明：

(1) 图中表达式的值为逻辑值，只有真、假两个可选值，通常我们规定非0为真、0为假。

(2) 单选择形式：如图3-2(a)所示，当表达式的值为真的时候执行语句，为假的时候不执行任何语句。

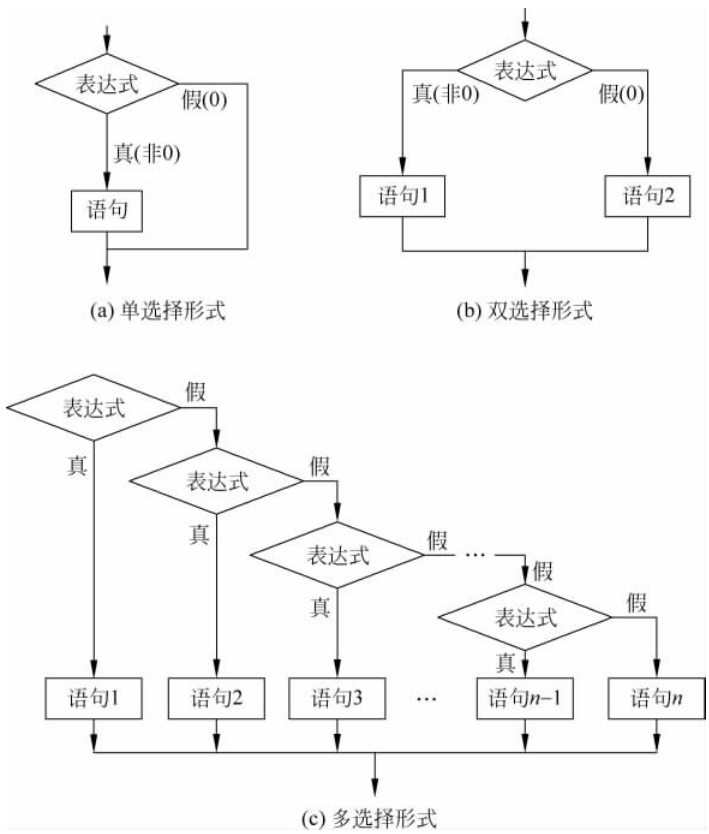

图 3-2　选择结构程序设计流程图

(3) 双选择形式：如图 3-2(b)所示，当表达式的值为真的时候执行语句 1，为假的时候执行语句 2。

(4) 多选择形式：如图 3-2(c)所示，遵循上面的判断原则对多个表达式依次进行判断，实现根据不同的判断情况来选择执行不同的语句。

(5) 对于上述 3 种形式，不管根据什么判断条件执行哪条语句，最后都只有一个出口，即去执行后续语句，遵循结构化程序设计的单入口、单出口原则。

3. 循环结构

循环结构表示程序反复执行同一组操作，直到某条件(即表达式的值)为假(或为真)时才终止循环，否则继续执行本组操作。

循环结构的基本形式有两种，即当型循环和直到型循环。

(1) 当型循环：如图 3-3(a)所示，表示先判断条件(即表达式的值)，当满足给定的条件(即表达式的值为非 0)时执行循环体，并且在循环终端处流程自动返回到循环入口；如果条件不满足(即表达式的值为 0)，则退出循环体，直接到达流程出口处。因为是“当条件满足时执行循环”，即先判断循环条件后执行，所以称之为当型循环。

(2) 直到型循环：如图 3-3(b)所示，表示从结构入口处直接执行循环体语句，在循环终端处判断条件(即表达式的值)，如果条件不满足(即表达式的值为非 0)，返回入口处，继续

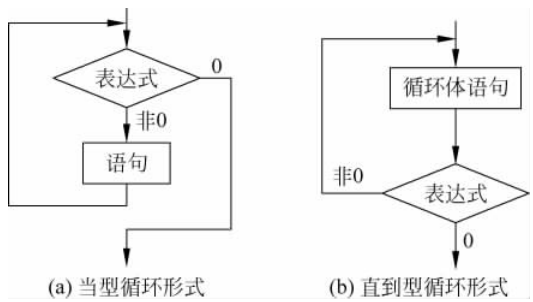

图 3-3　循环结构程序设计流程图

执行循环体，直到条件满足（即表达式的值为 0）时退出循环，到达流程出口处。因为是先执行循环体后判断，即“直到条件为真时为止”，所以称之为直到型循环。

要点说明：

① 采用循环结构来实现控制的前提条件有两个，首先，每次循环所要进行的操作在完全相同的情况下；其次，每次循环所要操作的数据必须能由上次循环的操作数据通过某种办法获取到。

② 采用循环控制变量来控制循环何时结束：我们需要清楚循环控制变量的 3 个值，即初值、终值和步长，通过它们来控制循环。

- 初值：即第一次循环操作时循环控制变量的值。
- 终值：即处理完最后一次循环操作需要退出循环时循环控制变量的值。
- 步长：本次循环操作与下一次循环操作的循环控制变量的增（或减）值。

3.2　顺序结构程序设计

前面介绍了程序设计的 3 种基本结构，即顺序结构、分支结构和循环结构，由 3 种结构组成各种复杂的程序，C 语言提供了多种语句来实现这些结构。本节讨论顺序结构程序设计以及 C 语言的基本语句。以顺序结构设计的程序自顶向下，按照每条语句书写的顺序依次执行，每条语句都执行，而且都只执行一次。在顺序结构中不涉及复杂的算法，只是由一些基本 C 语句组成，是一种最简单的程序设计结构。

【例 3-1】　从键盘输入任意 3 个数据，求其平均值。

算法设计：

（1）求平均值，故将变量定义为浮点型比较合适；

（2）输入数据，求平均值；

（3）输出平均值。

程序代码：

```
#include <stdio.h>
void main()
{
  float a,b,c,d;                          /* 变量的定义 */
  scanf("%f%f%f",&a,&b,&c);               /* 输入函数后加一个分号 */
```

```
    d=(a+b+c)/3;                          /*表达式后面加一个分号*/
    printf("%f",d);                       /*输出函数后面加一个分号*/
}
```

程序说明：

(1) 本程序的所有语句按照书写顺序逐条执行。

(2) 所有语句都以分号结尾，这种语句称为表达式语句。

除了表达式语句外，C 语言还有其他类型的语句，例如函数调用语句、控制语句、复合语句、空语句，下面分别做详细介绍。

使用任何高级语言编写的程序都是由若干个语句按照一定的结构组织在一起的，语句用来向计算机系统发出操作指令，每个语句经编译后形成若干个机器指令。一个完整的 C 语言程序包含若干条语句，大体结构如图 3-4 所示。

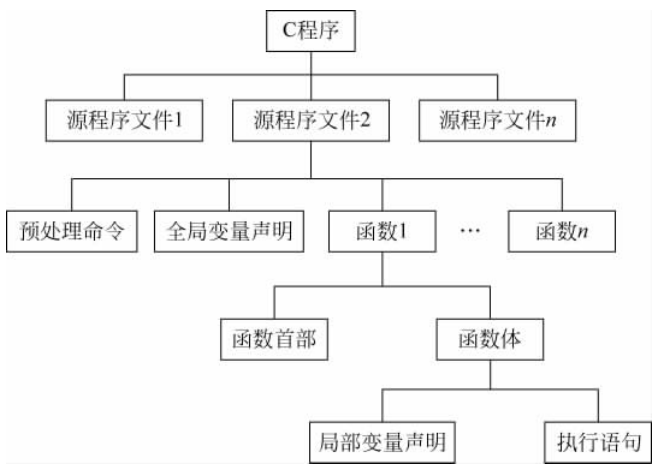

图 3-4 C 语言程序结构

其中 C 程序的执行部分是由语句组成的，也是程序的主体，程序的具体功能就是由各种语句实现的。

(1) 表达式语句：表达式语句由表达式加上分号“；”组成。

其一般形式为：

表达式;

说明：执行表达式语句就是计算表达式的值。

例如：

```
a=b+c;      赋值语句
a+b;        加法运算语句,但计算结果不能保留,无实际意义
i++;        自增1语句,i值增1,与"i=i+1;"等价
```

(2) 函数调用语句：由函数名、实际参数加上分号“；”组成。

其一般形式为：

函数名(实际参数表);

说明：执行函数语句就是调用函数体并把实际参数赋给函数定义中的形式参数，然后执行被调函数体中的语句，求取函数值(在后面函数中再详细介绍)。

例如：

```
printf("C Program");      调用库函数,输出字符串
```

(3) 控制语句：控制语句用于控制程序的流程，以实现程序的3种结构方式。C语言有9种控制语句，按其功能可以分成以下3类。

① 条件判断语句：if语句、switch语句；

② 循环执行语句：do…while语句、while语句、for语句；

③ 转向语句：break语句、goto语句、continue语句、return语句。

(4) 复合语句：把多个语句用括号{}括起来组成的语句称复合语句。

在程序中应把复合语句看成是单条语句，而不是多条语句。

例如：

```
while(i<=10)
{
  sum=sum+i;
  i++;
}
printf("%d",sum);
```

while循环体中“{}”内的语句为复合语句，复合语句内的各条语句都必须以分号(;)结尾，在括号“}”外不能再加分号。

(5) 空语句：只有分号“;”组成的语句称为空语句。空语句是什么也不执行的语句。在程序中空语句可用来做空循环体。

例如：

```
while(getchar()!='\n')
;
```

本语句的功能是当从键盘输入的字符不是回车时重新输入。

循环体内只有一个分号，为空语句。

(6) 赋值语句：在赋值表达式后面加上一个分号就是赋值语句，例如“sum=sum+i”是赋值表达式，而“sum=sum+i;”是赋值语句。赋值语句是C语言中最常用的一种语句，功能和特点都与赋值表达式相同。

其一般形式为：

变量=表达式;

在使用赋值语句时需要注意以下几点：

① 由于在赋值符“=”右边的表达式也可以是一个赋值表达式，因此，下述形式

```
变量=(变量=表达式);
```

是成立的，从而形成嵌套的情形。其展开之后的一般形式为：

```
变量=变量=…=表达式;
```

例如：

```
a = b = c = d = e = 5;
```

按照赋值运算符的右结合性，其实际上等效于：

```
e = 5;
d = e;
c = d;
b = c;
a = b;
```

② 注意在变量说明中给变量赋初值和赋值语句的区别。

给变量赋初值是变量说明的一部分，赋初值后的变量与其后的其他同类变量之间仍必须用逗号间隔，而赋值语句必须用分号结尾。

例如：

```
int a = 5,b,c;
```

③ 在变量说明中不允许连续给多个变量赋初值。

例如下述说明是错误的：

```
int a = b = c = 5;
```

必须写为：

```
int a = 5,b = 5,c = 5;
```

而赋值语句允许连续赋值，即可以写为“a＝b＝c＝5；”。

④ 注意赋值表达式和赋值语句的区别。

赋值表达式是一种表达式，它可以出现在任何允许表达式出现的地方，而赋值语句不能出现在任何地方，只能作为语句单独使用。

下述语句是合法的：

```
if((x = y + 5)> 0) z = x;
```

该语句的功能是，若表达式 $x＝y＋5$ 大于 0，则 $z＝x$。

下述语句是非法的：

```
if((x = y + 5;)> 0) z = x;
```

因为“x＝y＋5；”是语句，不能出现在表达式中。

【例 3-2】 从键盘输入一个小写字母，要求改用大写字母输出。

算法设计：

(1) 从键盘输入一个字符存入变量 s1 中；

(2) 利用 ASCII 值转换大小写(大写字母 A 的 ASCII 值为 65，小写字母 a 的 ASCII 值为 97，大小写字母相差 32)；

(3) 输出转换后数据。

程序代码：

```
#include <stdio.h>
 void main()
{
   char s1,s2;                          /*定义变量*/
   scanf("%c",&s1);
   s2 = s1 - 32;                        /*转换成大写字母*/
   printf("%c \n",s2);                  /*输出结果*/
}
```

运行情况：

```
输入数据：b
输出结果：B
```

【例 3-3】 从键盘输入任意 3 位整数，求其各位数字之和。

程序代码：

```
#include <stdio.h>
 void main()
{
   int n = 0,x = 0,y = 0,z = 0,sum = 0;
   scanf("%d",&n);                      /*输入一个 3 位整数*/
   x = n/100;                           /*求百位数上的数字*/
   y = (n/10) % 10;                     /*求十位数上的数字*/
   z = n % 10;                          /*求个位数上的数字*/
   sum = x + y + z;
   printf("sum = %d\n",sum);
}
```

运行情况：

```
输入：345
```

则结果为：sum＝12

【例 3-4】 求 $ax^2+bx+c=0$ 方程的根，a、b、c 由键盘输入，假设 $b^2-4ac>0$。

算法设计：

(1) 定义变量 a、b、c、disc、x1、x2、p、q；

(2) 输入 a、b、c；

(3) 计算 disc$=b^2-4ac$、$p=\dfrac{-b}{2a}$、$q=\dfrac{\sqrt{b^2-4ac}}{2a}$、x1$=p+q$、x2$=p-q$；

(4) 输出 x1、x2。

程序代码：

```
#include <math.h>
#include <stdio.h>
void main()
{
   float a,b,c,disc,x1,x2,p,q;
   scanf("%f %f %f",&a,&b,&c);
   disc = b * b - 4 * a * c;
```

```
    p=-b/(2*a);
    q=sqrt(disc)/(2*a);
    x1=p+q;x2=p-q;
    printf("\nx1=%5.2f\nx2=%5.2f\n",x1,x2);
}
```

运行情况：

从键盘输入“1 2 1”时得到两个相等的实根－1，输入“2 3 1”时得到两个不相等的实根，x1＝－0.50，x2＝－1.00。注意输入的数据必须保证 disc 大于 0。

3.3 选择结构程序设计

选择结构表示程序的处理步骤出现了分支，它需要根据某一特定的条件选择其中的一个分支执行。选择结构有单分支、双分支和多分支 3 种形式，其语句有两类，一类是 if 语句，另一类是 switch 语句，下面分别进行介绍。

3.3.1 单分支选择结构

在 C 语言中提供简单 if 语句来实现单分支选择结构。

if 语句又称为条件分支语句，它通过对给定条件的判断来决定所要执行的操作。

格式：

```
if (表达式)语句;
```

功能：首先计算表达式的值，若表达式的值为“真”(非 0)，则执行语句；若表达式的值为“假”(0)，不执行语句。其流程图如图 3-2(a)所示。

例如：

```
if(x<0)x=-x;
```

【例 3-5】 从键盘输入 3 个整数，按从大到小的顺序输出。

算法设计：

将从键盘输入的 3 个整数分别存入 a、b、c 几个整型变量中。先比较 b 与 a 的大小，如果 b 值比 a 值大就交换两者的值，否则不做任何操作，此时 a 变量里存放的肯定是 a、b 中最大的那个数；再比较 a 与 c 的大小，如果 c 值比 a 值大就交换两者的值，经过上述两次比较操作，变量 a 中存放的肯定是 3 个数中的最大值；最后比较 b 与 c 的大小，如果 c 值比 b 值大就交换两者的值，这样 a、b、c 中的值就变为从大到小的顺序了。

程序代码：

```
#include<math.h>
#include<stdio.h>
void main()
{
    int a,b,c,t;
    scanf("%d%d%d",&a,&b,&c);
    printf("a=%d,b=%d,c=%d\n",a,b,c);
```

```
    if (b>a) {t=a;a=b;b=t;}        /* 执行后 a 值比 b 值大 */
    if (c>a) {t=a;a=c;c=t;}        /* 执行后 a 值比 b、c 值大 */
    if (c>b) {t=b;b=c;c=t;}        /* 执行后 b 值比 c 值大 */
      printf(" %d %d %d\n",a,b,c);
}
```

运行情况：

```
输入数据：5 4 9
输出结果：9 5 4
```

3.3.2　双分支选择结构

在 C 语言中提供了简单分支 if 语句来实现双分支选择结构。

格式：

```
if (表达式)语句 1
else 语句 2
```

功能：首先计算表达式的值，若表达式的值为“真”(非 0)，则执行语句 1；若表达式的值为“假”(0)，则执行语句 2。其流程图如图 3-2(b)所示。

例如：

```
if(x>=0)y=x;
else y=-x;
```

【例 3-6】 输入一个整数(大于 1 小于 100)，判断其是否为同构数。同构数是指一个整数位于它的平方数右边，如 5、6 都为同构数。

算法设计：

输入一个整数存入变量 m 中(如 $m=25$)，并计算其平方数存入变量 pm 中(pm=625)。如果 m 小于 10，取其平方数 pm 的个位数；如果 m 大于 10，取其平方数 pm 的后两位数构成的数(如 25)，进而判断 m 是否为同构数。

程序代码：

```
#include<stdio.h>
 void main()
{
   int m,pm,y;
   scanf(" %d",&m);
   pm=m*m;
   if(m<10) y=pm%10;              /* 如 m 为 1 位整数,取其平方数的个位数 */
   else y=pm%100;                 /* 如 m 为两位整数,取其平方数的后两位数 */
   if(m==y) printf(" %d 是同构数\n",m);
   else  printf(" %d 不是同构数\n",m);;
}
```

运行情况：

```
输入数据：25
```

```
输出结果: 25 是同构数
输入数据: 15
输出结果: 15 不是同构数
```

3.3.3 多分支(多情况)选择结构

在C语言中,对多个分支选择时可采用if…else…if分支嵌套语句或switch语句来实现。

1. if…else…if分支嵌套语句

格式:

```
if (表达式 1) 语句 1
  else if (表达式 2) 语句 2
        else if (表达式 3) 语句 3
             ⋮
              else if (表达式 n - 1) 语句 n - 1
                else 语句 n
```

功能:首先计算表达式1的值,若为"真"(非0),执行语句1,否则进行下一步判断;若表达式2为真,执行语句2,否则进行下一步判断……最后所有表达式都为假时执行语句n,如图3-2(c)所示。

例如:

```
if (score > 89)        grade = 'A';
  else if (score > 79)  grade = 'B';
       else if (score > 69)  grade = 'C';
            else if (score > 59)  grade = 'D';
                 else     grade = 'E';
```

对于if语句的说明如下:

(1) if后面圆括号中的表达式一般是关系表达式或逻辑表达式,用于描述选择结构的条件,也可以是任意类型的表达式(例如常量表达式、赋值表达式、算术表达式等)。

例如:

```
if (2)  printf("OK!");
```

是合法的,因为表达式的值为2,非0,按"真"处理,执行结果输出"OK!"。

(2) 在if语句中,每个else前面有一个分号,整个语句结束的位置也有一个分号。这是由于分号是C语句中不可缺少的部分,这个分号是if语句中的内嵌语句所需要的。

(3) 在if和else后面可以只含有一个内嵌的操作语句,也可以含有多个操作语句,此时应该用大括号"{ }"将几个语句括起来,构成一个复合语句。注意,复合语句的"{"和"}"之后不能加分号。

【例3-7】 $y=\begin{cases}1(x>0)\\0(x=0)\\-1(x<0)\end{cases}$,输入一个$x$值,输出相应$y$值。

算法设计:

(1) 输入数据 x；

(2) 判断 x 的值，计算 y 值；

(3) 输出 y 的值。

程序代码：

```
#include <stdio.h>
void main()
{
  int x,y;
  scanf("%d",&x);
  if (x<0)  y=-1;
  else  if (x==0)  y=0;
      else  y=1;
  printf("x=%d,y=%d\n",x,y);
}
```

运行情况：

输入 4，则 $y=1$；输入 0，则 $y=0$；输入 -5，则 $y=-1$。

【例 3-8】 输入学生的百分制成绩，输出相应五级分制等级。

算法设计：

(1) 定义变量 score，输入百分制成绩；

(2) 判断 score 所在的范围，得出相应等级；

(3) 输出相应等级。

程序代码：

```
#include <stdio.h>
 void main()
{
   int score;
   char grade;
   scanf ("%d",&score);
   if (score>=90) grade = 'A';
     else if (score>=80) grade = 'B';
         else if (score>=70) grade = 'C';
             else if (score>=60) grade = 'D';
                 else   grade = 'E';
   printf("%c\n", grade);
}
```

运行情况：

输入 93，则输出'A'；输入 87，则输出'B'；输入 71，则输出'C'；输入 62，则输出'D'；输入 35，则输出'E'。

【例 3-9】 从键盘输入任意字符，判断是数字、大小写字母还是其他字符，输出其相应信息。

算法设计：

从键盘输入的是一个字符，可以用 getchar()函数完成字符的输入，判断输入的字符是

大写、小写或是其他字符,输出相应信息。

程序代码:

```
#include <stdio.h>
 void main()
{
   char c;
   printf("请输入字符: ");
   c = getchar();
   if(c >= '0'&&c <= '9')
       printf("这是一个数字\n");
   else if(c >= 'A'&&c <= 'Z')
            printf("这是一个大写字母\n");
        else if(c >= 'a'&&c <= 'z')
                 printf("这是一个小写字母\n");
             else
                 printf("这是一个其他字符\n");
}
```

运行情况:

若从键盘输入一个大写字母"A",则输出"这是一个大写字母"。

注意:getchar()每次只接受一个字符。

2. switch 语句

上面介绍的多分支结构用 if-else-if 语句解决,由于分支在很多时候结构层次比较多,编程时容易出现错误,在 C 语言中还提供了另外一种实现多分支结构的控制语句——switch 语句,用 switch 语句可以方便地解决多分支结构问题。

格式:

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

说明:

(1) switch 语句的表达式通常是一个整型或字符型变量,也允许是枚举型变量,其结果为相应的整数、字符或枚举常量。case 后的常量表达式必须是与表达式对应的整数、字符或枚举常量。

(2) switch 语句中所有 case 后面的常量表达式的值必须互不相同。

(3) switch 语句中的 case 和 default 的出现次序是任意的。

(4) switch 语句的执行过程是先用表达式的值与"case 常量表达式"逐个进行比较,在找到值相等的常量表达式时执行其后面的语句组,并且直到遇见 break 语句或整个 switch 语句终止时结束,所以必须恰当地运用 break 语句来终止 switch。

(5) 每个 case 的后面既可以是一个语句,也可以是多个语句,当是多个语句的时候不需要用花括号括起来。

(6) 多个 case 的后面可以共用一组执行语句,例如:

```
switch (n)
{
 case 1:
 case 2: x = 10;break;
   ⋮
}
```

它表示当 $n=1$ 或 $n=2$ 时都执行下面两个语句:

```
x = 10;
break;
```

(7) 将例 3-10 中的"break;"语句去掉后运行:

```
输入数据: 88          输入数据: 67
输出结果: B           输出结果: D
          C                     E
          D
          E
```

在该程序中没有 break 语句,因此执行结果出现错误。

【例 3-10】 将例 3-8 改用 switch 语句实现。

```
#include <stdio.h>
void main()
{
     int score,k;
     scanf("%d",&score);
     k = score /10;
     switch(k)
    {
         case  10:
         case   9:  printf("A\n");break;
         case   8:  printf("B\n");break;
         case   7:  printf("C\n");break;
         case   6:  printf("D\n");break;
          default:  printf("E\n");
     }
}
```

【例 3-11】 输入一个不多于 5 位的正整数,逆序输出各个数位上的数。

算法设计:

(1) 定义一个长整型变量 x,将输入数据存入 x;

(2) 判断 x 的位数,求出个数位的代码;

(3) 输出各个数位上的数。

程序代码:

```
#include <stdio.h>
 void main()
{
   long int x;
   int n,g,s,b,q,w;
   scanf("%ld",&x);
   if(x>=10000)n=5;
   else if(x>=1000)n=4;
        else if(x>=100)n=3;
            else if(x>=10)n=2;
                else n=1;
    g=x%10;s=x/10%10;b=x/100%10;q=x/1000%10;w=x/10000;
    switch(n)
    {
        case 5:printf("%d,%d,%d,%d,%d\n",g,s,b,q,w);break;
        case 4:printf("%d,%d,%d,%d\n",g,s,b,q);break;
        case 3:printf("%d,%d,%d\n",g,s,b);break;
        case 2:printf("%d,%d\n",g,s);break;
        case 1:printf("%d\n",g);
    }
}
```

运行情况：

```
输入数据：45678
输出数据：8 7 6 5 4
```

3.3.4 条件运算符和条件表达式

条件运算符由两个符号“?”和“：”组成，要求有 3 个操作对象，称三目(元)运算符，它是 C 语言中唯一的三目运算符。

条件表达式的格式：

表达式 1 ?表达式 2：表达式 3

说明：

(1) 通常情况下，表达式 1 是关系表达式或逻辑表达式，用于描述条件表达式中的条件，表达式 2 和表达式 3 可以是常量、变量或表达式。

例如：

```
(x==y)?'T':'F'
(a>b)?printf("%d",a):printf("%d",b)
```

等均为合法的条件表达式。

(2) 条件表达式的执行顺序：先求解表达式 1，若为非 0(真)则求解表达式 2，此时表达式 2 的值作为整个条件表达式的值；若表达式 1 的值为 0(假)，则求解表达式 3，表达式 3 的值就是整个条件表达式的值。

```
min=(a<b)?a:b;
```

执行结果就是将 a 和 b 中较小的值赋给 min。

(3) 条件运算符的优先级别仅高于赋值运算符,低于前面介绍过的所有运算符。因此"min=(a<b)? a:b;"可直接写成"min=a<b? a:b;",如果有"a>b? a:b+1"等效于"a>b? a:(b+1),"而不等效于"(a>b? a:b)+1"。

(4) 条件运算符的结合方向为"自右至左"。例如:

```
x>0?1:x<0?-1:0      等效于 x>0?1:(x<0?-1:0)
```

(5) 表达式1、表达式2和表达式3的类型可以不同,此时条件表达式的值的类型为它们中较高的类型。

【例 3-12】 从键盘输入一个字符,如果是大写字母,将它转换成小写字母并输出,如果是小写字母直接输出。

算法设计:

(1) 定义字符型变量 a、b;

(2) 判断输入字符的大小写,若是大写字母转换为小写字母,小写字母的ACSII码值比相应大写字母的ACSII码值大32;

(3) 输出转换后的字符。

程序代码:

```
#include <stdio.h>
void main()
{
  char ch;
  printf("Please enter a character: \n");
  scanf("%c",&ch);
  ch=(ch>='A'&&ch<='Z')?(ch+32):ch;
  printf("%c\n",ch);
}
```

程序结果:

```
Please enter a character:
A↙
a
```

3.4 循环结构程序设计

循环结构是程序中一种很重要的结构,其特点是在给定条件成立时反复执行某程序段,直到条件不成立为止。给定的条件称为循环条件,反复执行的程序段称为循环体。C语言提供了3种实现循环结构的循环语句,即while循环语句、do…while循环语句、for循环语句。

3.4.1 while 语句

格式:

```
while(条件表达式)
    {循环体}
```

功能：先计算条件表达式的值，当值为真(非0)时执行循环体语句，当条件表达式不成立(值为0)时结束循环，执行循环体外的语句，其执行过程如图3-3(a)所示。

【例3-13】 求 $sum=\sum_{n=1}^{100}n$。

算法设计：

累加运算是一个表达式(sum=sum+i)多次执行，执行100次，所以在累加运算中从1开始至100依次累加，在程序设计中将变量i作为循环控制变量。

(1) 设累加变量 sum=0；

(2) 循环控制变量i赋初值1，设定循环条件为i<=100(即循环控制变量终值为100)，循环语句为 sum=sum+i，每循环一次，循环控制变量增加1；

(3) 输出 sum 的值。

程序代码：

```
#include <stdio.h>
void main()
  {
     int i,sum = 0;
     i = 1;                          /* 为循环控制变量 i 赋初值 1 */
     while(i <= 100)                 /* 循环控制变量的终值为 100 */
    {
       sum = sum + i;
       i++;                          /* 每循环一次,循环控制变量增加 1 */
    }
    printf("%d\n",sum);
  }
```

在使用 while 语句时应注意以下几点：

(1) while 语句中的表达式一般是关系表达式或逻辑表达式，只要表达式的值为真(非0)即可继续循环。

(2) 循环体如包括一个以上的语句，则必须用{}括起来，组成复合语句。如果不包括花括号，则 while 语句的范围只到 while 后面的第一个分号处。

(3) 在循环体中应有使循环趋向于结束的语句，否则循环就是死循环，如上例中的“i++;”语句。

【例3-14】 求 $p=n!$(从键盘输入一个整数存入变量n)。

算法设计：

累乘运算是一个表达式($p=p*i$)多次执行，执行n次，所以在累乘运算中从1开始至n依次累乘，在程序设计中一般用变量i作为循环控制变量。

(1) 设定变量 $p=1$，$i=1$；

(2) 设定循环条件为i<=n，n由键盘输入，循环语句为“p=p*i;”；

(3) 输出 sum 的值。

程序代码：

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

```
{
   long p = 1;
   int i,n;
   scanf("%d",&n);
   i = 1;                          /* 为循环控制变量 i 赋初值 1 */
   while(i<=n)                     /* 循环控制变量的终值为 n */
  {
      p = p * i;
      i++;                         /* 每循环一次,循环控制变量增加 1 */
   }
   printf("%ld\n",p);
}
```

【例 3-15】 计算并输出整数 n 的所有因子之和(不包括 1 与自身)。

算法设计：

对从 2 到 $n-1$ 的所有整数 k 分别试除 n，如果能整除，则 k 等于 n 的因子进行累加。

(1) 设定变量“int n,k,s=0;”，由键盘输入整数存入变量 n。

(2) 设定循环条件为 $k<=n-1$(也可以是 $k<=n/2$，因为一个整数的真因子不会大于该数的一半)，循环语句为“if(n%k==0) s=s+k;”。

(3) 输出 s 的值。

程序代码：

```
#include <stdio.h>
void main()
{
   int n,k,s = 0;
   scanf("%d",&n);
   k = 2;                          /* 循环操作数据的初值从 2 开始 */
   while(k<n)                      /* 循环操作数据的终值为 n */
  {
      if(n%k == 0) s = s + k;
      k++;                         /* 每循环一次,循环操作数据的值增 1 */
  }
   printf("%d\n",s);
}
```

【例 3-16】 从键盘输入一批整数，分别统计奇数、偶数的个数，输入 0 时结束。

算法设计：

(1) 输入一个整数存入变量 n 中；

(2) 当 n 的值不为 0 时判断奇偶性并统计，当 n 的值为 0 时退出循环执行(4)；

(3) 再次输入一个整数存入变量 n 中，返回(2)；

(4) 输出结果。

程序代码：

```
#include <stdio.h>
 void main()
{
   int n,js = 0,os = 0;
```

```
    scanf("%d",&n);                    /*从键盘获取循环操作数据的初值*/
    while(n!=0)                        /*循环操作数据的终值为0*/
      { if(n%2!=0) js++;
        else os++;
        scanf("%d",&n);                /*从键盘获取下一个循环操作数据*/
      }
      printf("js=%d,os=%d\n",js,os);
    }
```

3.4.2 do…while 语句

格式：

```
do
  {
   循环体
  }while(条件表达式);
```

do 循环与 while 循环的不同在于它先执行循环中的语句，然后判断表达式是否为“真”，如果为“真”则继续循环，如果为“假”则终止循环，因此 do…while 循环至少要执行一次循环语句。其执行过程如图 3-3(b)所示。

【例 3-17】 用 do…while 语句求 100～200 之间的奇数和。

```
#include<stdio.h>
void main()
{
   int i,sum=0;
   i=100;
   do
   {
      if(i%2==1)
         sum=sum+i;
      i++;
   } while(i<=200);
   printf("%d\n",sum);
}
```

注：当循环体内有多条语句时要用“{}”把它们括起来。

【例 3-18】 while 和 do…while 循环比较。

下面两种编写方式有什么区别，请读者上机运行，运行时输入数据为 15，注意检验运行结果。

(1)

```
#include<stdio.h>
void main()
{
  int sum=0,i;
  scanf("%d",&i);
  while(i<=10)
```

```
  {
     sum = sum + i;
     i++;
  }
  printf("sum = %d",sum);
}
```

(2)

```
#include <stdio.h>
 void main()
{
   int sum = 0,i;
   scanf("%d",&i);
   do
   {
      sum = sum + i;
      i++;
   }while(i <= 10);
   printf("sum = %d",sum);
}
```

3.4.3　for 语句

在 C 语言中 for 语句的使用最为灵活，一般用于计数型循环，也可以用于条件型循环，用 while 语句和 do…while 语句所能解决的问题大部分可以用 for 语句来解决。

格式：

```
for(表达式 1; 表达式 2; 表达式 3)
  {循环体; }
```

执行过程如下：

(1) 先求解表达式 1。

(2) 求解表达式 2，若其值为真(非 0)，则执行 for 语句中的循环体，然后执行下面的第(3)步；若其值为假(0)，则结束循环，转到第(5)步。

(3) 求解表达式 3。

(4) 转回上面第(2)步继续执行。

(5) 循环结束，执行 for 语句下面的一个语句。

其执行过程可以用图 3-5 表示。

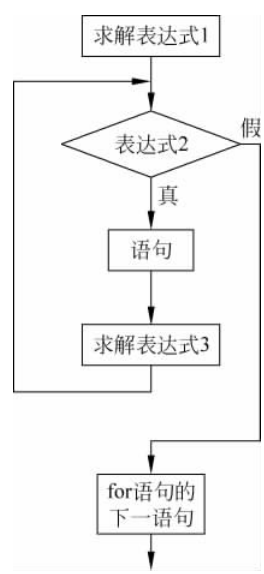

图 3-5　执行过程

for 语句也可以用下面的形式表示：

```
for(循环控制变量赋初值; 循环条件; 循环控制变量增量)
    {  循环体 }
```

循环控制变量赋初值总是一个赋值语句，它用来给循环控制变量赋初值；循环条件是一个关系表达式，它决定什么时候退出

循环；循环变量增量定义循环控制变量每循环一次后按什么方式变化，这 3 个部分之间用“；”分开。

例如：

```
for(i = 1; i <= 100; i++)
  sum = sum + i;
```

先给循环控制变量 i 赋初值 1，判断 i 是否小于等于 100，若为真，则执行语句“sum＝sum＋i”，之后 i 值增加 1。再重新判断 i 是否小于等于 100，重复执行语句，直到条件为假，即 $i>100$ 时结束循环。相当于：

```
i = 1;
while(i <= 100)
{    sum = sum + i;
     i++;
}
```

for 循环语句的一般形式对应以下 while 循环形式：

```
表达式 1;
 while(表达式 2)
{    语句;
     表达式 3;
}
```

注意：

(1) for 循环语句中的“表达式 1(循环控制变量赋初值)”、“表达式 2(循环条件)”和“表达式 3(循环控制变量增量)”都是可选项，即可以省略，但“；”不能省略。

(2) 省略“表达式 2(循环条件)”，若不做其他处理，便成为死循环。

例如：

```
for(i = 1;;i++)
  sum = sum + i;
```

相当于：

```
i = 1;
 while(1)
 {
     sum = sum + i;
     i++;
 }
```

(3) 若省略了“表达式 3”，则不对循环控制变量进行修改操作，这时可在语句体中加入修改循环控制变量的语句。

例如：

```
for(i = 1;i <= 100;)
{
   sum = sum + i;
```

```
    i++;
}
```

(4) 省略“表达式 1”和“表达式 3”。

例如：

```
for(;i<=100;)
{
    sum=sum+i;
    i++;
}
```

相当于：

```
while(i<=100)
{
    sum=sum+i;
    i++;
}
```

(5) 3 个表达式都可以省略。

例如：

```
for(; ; )语句
```

相当于：

```
while(1)语句
```

(6) 表达式 1 可以是设置循环控制变量初值的赋值表达式，也可以是其他表达式。

例如：

```
for(sum=0;i<=100;i++) sum=sum+i;
```

(7) 表达式 1 和表达式 3 可以是一个简单表达式，也可以是逗号表达式。

```
for(sum=0,i=1;i<=100;i++)
  sum=sum+i;
```

或：

```
for(i=0,j=100;i<=100;i++,j--)
 k=i+j;
```

(8) 表达式 2 一般是关系表达式或逻辑表达式，也可以是数值表达式或字符表达式，只要其值非零，就执行循环体。

例如：

```
for(i=0;(c=getchar())!='\n';i+=c);
```

又如：

```
for(;(c=getchar())!='\n';)
  printf("%c",c);
```

【例 3-19】 求 1～200 之间能被 3 整除但不能被 12 整除的数的个数及和。

算法设计：

(1) 定义整型变量 n 与 sum；

(2) 对于 1～200 之间的所有 3 的倍数，若不能被 12 整除则累加并计数；

(3) 输出结果。

程序代码：

```
#include <stdio.h>
 void main()
{
   int i,n = 0,sum = 0;
   for(i = 3;i <= 200;i = i + 3)
      if(i % 12!= 0) {sum = sum + i;n++;}
   printf("sum = %d k = %d\n",sum,n);
}
```

3.4.4 几种循环语句的比较

(1) 3 种循环语句都可以用来处理同一个问题，一般可以互相代替，但不提倡用 goto 型循环，故在此没做介绍。

(2) 在使用 while 和 do…while 循环时，循环体中应包括使循环趋于结束的语句，for 语句的功能最强。

(3) 在用 while 和 do…while 循环时，循环控制变量初始化的操作应在 while 和 do…while 语句之前完成，而 for 语句可以在表达式 1 中实现循环控制变量的初始化。

3.4.5 break 语句

break 语句通常用在循环语句和 switch 语句中。当 break 语句用在 switch 语句中时可使程序跳出 switch 结构执行 switch 以后的语句。

当 break 语句用于 do…while、for、while 循环语句中时，可使程序终止循环而执行循环后面的语句，通常 break 语句总是和 if 语句连在一起，即满足条件时便跳出循环。观察以下程序，注意 break 语句的功能。

【例 3-20】 判断整数 m 是否为素数。

算法设计：

(1) 输入任意数据存入整型变量 m 中；

(2) 若整数 m 不能被 2 到$\sqrt{m}$范围内的任意数整除，则 m 是素数；

(3) 输出是否为素数的信息。

程序代码：

```
#include <stdio.h>
#include <math.h>
 void main()
{
   int m,i,k;
```

```
    scanf("%d",&m);
    k=sqrt(m);
    for(i=2;i<=k;i++)
       if(m%i==0)break;
    if(i>=k+1)                          /*整个循环过程中都没出现能将m整除的i*/
       printf("%d is a prime number\n",m);
    else
       printf("%d is not a prime number\n",m);
}
```

注意：

(1) break 语句对 if-else 的条件语句不起作用。

(2) 在多层循环和 switch 结构中一个 break 语句只能向外跳一层。

(3) 在循环语句中一旦使用了 break 语句，在跳出循环后通常要做条件判断，从而判断是什么情况退出循环结构的。

3.4.6　continue 语句

continue 语句的作用是跳过循环体中剩余的语句而强行进入下一次循环。continue 语句只用在 for、while、do…while 等循环体中，常与 if 条件语句一起使用，用来加速循环。

【例 3-21】 输出 100～200 之间不能被 3 整除的数。

程序代码：

```
#include<stdio.h>
void main()
   {
     int n;
     for(n=100;n<=200;n++)
     { if( n%3==0)  continue;
       printf("%4d",n);
     }
    printf("\n");
   }
```

程序说明：

continue 语句和 break 语句的区别是 continue 语句只结束本次循环，而不是终止整个循环语句；break 语句是终止整个循环过程，不再判断循环条件是否成立。

3.4.7　多重循环

若一个循环体内又包含一个完整的循环结构，称为循环的嵌套。在内嵌的循环中还可以嵌套循环，这就是多重循环。在各种语言中对于循环的嵌套的概念都是一样的。

【例 3-22】 以下面的形式输出九九乘法表。

```
1×1=1
2×1=2  2×2=4
3×1=3  3×2=6  3×3=9
…
```

9×1=9　9×2=18　9×3=27　…　9×9=81

算法设计：

九九乘法表共9行，每行的列数与行数相同，所以循环中又涉及循环，用双重循环实现，并且内层循环的循环次数与外层循环变量的值相等。

程序代码：

```
#include <stdio.h>
void main()
{
     int i,j;
     for (i=1; i<=9; i++)
     {
       for(j=1; j<=i; j++)
          printf("%d×%d=%2d  ", i, j, i*j);
       printf("\n");
     }
}
```

3.4.8 程序举例

【例 3-23】 输入一个年份，判断它是否为闰年。

算法设计：

(1) 输入年份；

(2) 若年号能被400整除或能被4整除但不能被100整除，则该年号为闰年，否则不是闰年；

(3) 输出闰年。

程序代码：

```
#include <stdio.h>
void main()
{
   int year,leap;
   printf("Please enter year:\n");
   scanf("%d",&year);
   if (year%400==0 || year%4==0&&year%100!=0)
      printf("%d is a leap year.\n",year);
   else
      printf("%d is not a leap year.\n",year);
}
```

程序运行情况：

```
Please enter year:
2008↙
2008 is a leap year.
Please enter year:
1989↙
1989 is not a leap year.
```

【例 3-24】 求 $ax^2+bx+c=0$ 方程的解。

算法分析：

(1) $a=0$，不是二次方程。

(2) $b^2-4ac=0$，有两个相等实根。

(3) $b^2-4ac>0$，有两个不等实根。

(4) $b^2-4ac<0$，有两个虚根。

程序代码：

```
#include "math.h"
   void main()
  {
      float a,b,c,d,disc,x1,x2,realpart,imagepart;
      printf("Please enter a,b,c:\n");
      scanf("%f,%f,%f",&a,&b,&c);                          /* 方程系数 */
      printf("The equation");
      if (fabs(a)<=1e-6)                                   /* 判别 a=0 */
        printf("is not quadratic");
      else
      {
      disc=b*b-4*a*c;                                      /* 判别式 */
      if (fabs(a)<=1e-6)                                   /*判别式等于 0*/
        printf("has two equal roots:%8.4f\n",-b/(2*a));
      else if (disc>1e-6)                                  /*判别式大于 0*/
          {
            x1=(-b+sqrt(disc))/(2*a);
            x2=(-b-sqrt(disc))/(2*a);
            printf("has distinct real roots:");
            printf("x1=%8.4f,x2=%8.4f\n",x1,x2);
          }
          else                                             /*判别式小于 0*/
          {
            realpart=-b/(2*a);
            imagpart=sqrt(-disc)/(2*a);
            printf("has complex roots:\n");
            printf("%8.4f+%8.4fi\n",realpart,imagpart);
            printf("%8.4f-%8.4fi\n",realpart,imagpart);
           }
      }
}
```

运行结果：

```
Please enter a,b,c:
3,4,5
The equation has complex roots:
-0.6667+1.1055i
-0.6667-1.1055i
Please enter a,b,c:
1,2,1
```

```
The equation has two equalroots: -1.0000
```

程序说明：

程序中用 disc 代表 b^2-4ac，用 fabs(disc)<=1e-6 判别 disc 的值是否为 0，是因为实数 0 在机器中存储时存在微小的误差，往往是以一个非常接近 0 的实数存放，所以采取的办法是判别 disc 的绝对值(fabs(disc))是否小于一个很小的数，如果小于此数就认为 disc=0。

【例 3-25】 用“$\pi/4\approx1-1/3+1/5-1/7+\cdots$”公式求 π 的近似值，直到某一项的绝对值小于 10^{-6} 为止。

程序代码：

```
#include <math.h>
void main()
{
  int s;
  float n,t,pi;
  t = 1,pi = 0;n = 1.0;s = 1;
  while(fabs(t)> 1e - 6)
  {
     pi = pi + t;
     n = n + 2;
     s =- s;
     t = s/n;
   }
  pi = pi * 4;
  printf("pi = %10.6f\n",pi);
}
```

【例 3-26】 输出 100～200 之间的素数。

程序代码：

```
#include <math.h>
#include <stdio.h>
void main()
{
  int n,i,k,m = 0;
  for(n = 101; n<= 200; n = n + 2)                    /* 设置循环 */
  {
      k = sqrt(n);
      for(i = 2;i <= k;i++)
        if(n % i == 0) break;                         /* 终止循环 */
      if (i > k)
      {  m = m + 1;
          printf(" %4d",n);
          if(m % 5 == 0)printf("\n");                 /* 控制每行输出 5 个数 */
      }
  }
    printf("\n");
}
```

【例 3-27】 找出 1～100 之间的所有同构数。

同构数的概念及算法设计见例 3-6。

程序代码：

```
#include <stdio.h>
void main()
{ int m,pm,t;
  for(m=2;m<=100;m++)
  {
    pm=m*m;
    if(m<10) t=pm%10;
    else t=pm%100;
    if(m==t) printf("%d\n",m);
  }
}
```

本章小结

本章学习了程序的控制结构，顺序、选择和循环 3 种基本控制结构是程序设计的基本思想，任何一个复杂的程序设计问题都可以通过 3 种基本程序结构的组合、嵌套来实现。读者要充分理解和掌握用 C 语言实现选择、循环结构的各种语句的功能和句法要求，并会应用到解决实际问题中。

习题 3

1. 有 3 个整数 a、b、c，由键盘输入，输出其中的最大数。

2. 有一函数：

$$y=\begin{cases}x & x<1\\ 2x-1 & 1\leqslant x<10\\ 3x-11 & x\geqslant 10\end{cases}$$

编写程序，输入 x 值，输出相应表达式对应的 y 值。

3. 给出一个百分制成绩，要求输出成绩等级'A'、'B'、'C'、'D'、'E'。90 分以上为'A'，80～89 分为'B'，70～79 分为'C'，60～69 分为'D'，60 分以下为'E'。

4. 给出一个不多于 4 位的正整数，要求：

(1) 求出它是几位数；

(2) 分别输出每一位数字；

(3) 按逆序输出各位数字，例如原数为“1234”，则输出“4 3 2 1”。

5. 编写程序求 1－3＋5－7＋…－99＋101 的值。

6. 输入两个正整数 m 和 n，求其最大公约数和最小公倍数。

7. 求一个 4 位数的各位数字的立方和。

8. 求出斐波那契数列的前一项与后一项之比的极限的近似值，例如，当误差为 0.0001

时,函数值为 0.618056。

9. 计算以下公式的 y 值:$y=1/2!+1/4!+\cdots+1/m!$(m 是偶数,m 值由键盘输入)。

10. 求出 1000 以内前 20 个不能被 2、3、5、7 整除的数的和。

11. 找出 1000 以内的所有完数(一个数若恰好等于它的真因子(即除了本身以外的约数)之和,这个数就称为完数,如 6=1+2+3)。

12. 求 $s=1^k+2^k+3^k+\cdots+N^k$ 的值(即 1 的 K 次方到 N 的 K 次方的累加和),n 和 k 值由键盘输入。

13. 求给定正整数 n 以内的所有素数之积($n<28$),n 值由键盘输入。

14. 输出所有"水仙花数",所谓"水仙花数"是指一个 3 位数,其各位数字的立方和等于该数本身。例如,153 是一个水仙花数,因为 $153=1^3+5^3+3^3$。

15. 求 $1!+2!+3!+\cdots+n!$ 的和,从键盘输入 n 值,并输出运算结果。

16. 输出以下图案,要求用循环实现。

```
    *
  * * *
* * * * *
* * * * * * *
* * * * *
  * * *
    *
```

17. 输入一行字符,分别统计出其中的英文字符、空格、数字和其他字符个数。