第3章 算法和控制语句

教学目标

(1) 初步理解算法的概念和特点。
(2) 掌握常用问题的算法。
(3) 掌握用流程图表示算法。
(4) 了解结构化程序设计的方法。
(5) 掌握格式化输入输出函数的用法。
(6) 掌握分支结构的程序设计,理解分支语句的嵌套。
(7) 掌握循环结构的程序设计及其相互嵌套。
(8) 理解 break 和 continue 的控制。
(9) 能够编写较复杂的程序。

3.1 问题引导

使用计算机,就是要利用计算机处理各种不同的问题,因此就必须事先对各类问题进行分析,确定解决问题的具体方法和步骤,再编制好一组使计算机执行的指令(即程序),交给计算机按照指定的步骤去工作。这些具体的方法和步骤,其实就是解决一个问题的算法。根据算法,依据某种规则编写计算机执行的命令序列,就是编制程序,而书写时所遵守的规则,就是语法规则。对于面向过程的程序设计语言,主要关注的是算法。掌握了算法,也就为以后的程序设计打下了坚实的基础。程序设计的关键之一,是解题的方法和步骤。学习C语言的重点,就是掌握分析问题、解决问题的方法,所以在C语言的学习中,一方面应熟练掌握该语言的语法,因为它是算法实现的基础,另一方面必须认识到算法的重要性,加强思维训练,以写出高质量的程序。下面从常见的问题出发来分析计算机如何解决这些问题。

[**问题 1**]**判断 3 角形的形状**:已知 3 个正数,这 3 个数能否构成一个三角形的 3 条边;如果能构成三角形,判断所构成三角形的形状。

[**问题 2**]**学习委员的烦恼**:小王作为班上的学习委员,每门课程考试后,任课老师都会让他统计成绩。老师并不关心每个人的具体成绩,而只关心参加考试的人数、平均分、最低分和最高分这 4 项指标,如何用C语言编写程序来帮小王完成这项任务。

[**问题 3**]**确定小偷**:甲、乙、丙、丁 4 人为偷窃嫌疑犯,只有一个是真正的小偷,在审讯过程,4 人都有可能说真话或假话。

甲:乙没有偷、丁偷的;

乙：我没有偷，丙偷的；
丙：甲没有偷，乙偷的；
丁：我没有偷；
请推断谁是小偷。

3.1.1 算法的概念

算法就是解决问题的方法和步骤。通常一个算法是由一系列的求解步骤来完成的。著名计算机科学家沃斯(Nikilaus Wirth)提出一个著名的公式：

数据结构＋算法＝程序

后来人们对沃斯公式进行了改进，认为程序除了包含数据结构和算法外，程序设计方法和开发工具也非常重要，改进后的沃斯公式为：

数据结构＋算法＋程序设计方法＋开发工具＝程序

数据结构就是数据的类型和数据的组织形式，它是处理的对象；而算法是程序的灵魂。它是解决“做什么”和“怎么做”的问题。在编写程序时，如果不了解算法就编写不出程序。一般来说，先给出问题的粗略算法(计算步骤)，再根据问题的内容进行逐步细化，添加必要的细节，使之成为较为详细的描述。程序设计方法是指导程序设计各阶段工作的原理和原则，以及依此提出的设计技术。程序设计方法学的目标是设计出可靠、易读而且代价合理的程序，目前主要有面向过程的程序设计方法和面向对象的程序设计方法。开发工具就是语言编程环境，好的编程环境能起到事半功倍的效果。

3.1.2 算法的表示

一个算法可以用不同的方法表示，常用的表示方法有：自然语言、传统流程图、N-S 流程图、伪代码、计算机语言等。本书主要采用传统流程图和计算机语言表示算法。

(1) 传统流程图：它用图形框表示各种操作，用图形表示算法。其特点是直观形象、易于理解，特别适合于初学者。国家标准 GB 1526—89 规定了一些常用的流程图符号，如图 3-1 所示。

图 3-1　常用流程图符号

(2) 计算机语言表示：一个算法的最终目的是要用计算机来求解，只有用计算机编写的语言程序才能被计算机执行，因此用流程图描述一个算法后必须将其转化为计算机语言。用计算机语言表示算法必须严格遵守所用语言的语法规则。例如，求自然数和的 C 语言程序如下：

```
#include<stdio.h>
#include<stlib.h>
int main()
{
    int i,n,s=0;
    scanf("%d",&n);
```

```
    i = 1;
    while(i <= n)
    {
        s = s + i;
        i++;
    }
    printf("s = %d\n",s);
    system("pause");
    return 0;
}
```

3.1.3 基本算法举例

例 3.1 用流程图表示[问题 1,判断三角形的形状]。

分析:首先输入 3 个数,并判断是否都是正数;如果都是正数,再判断是否满足两边之和大于第三边(或两边之差小于第三边),然后根据三边的关系判断三角形的类型。其流程如图 3-2 所示。

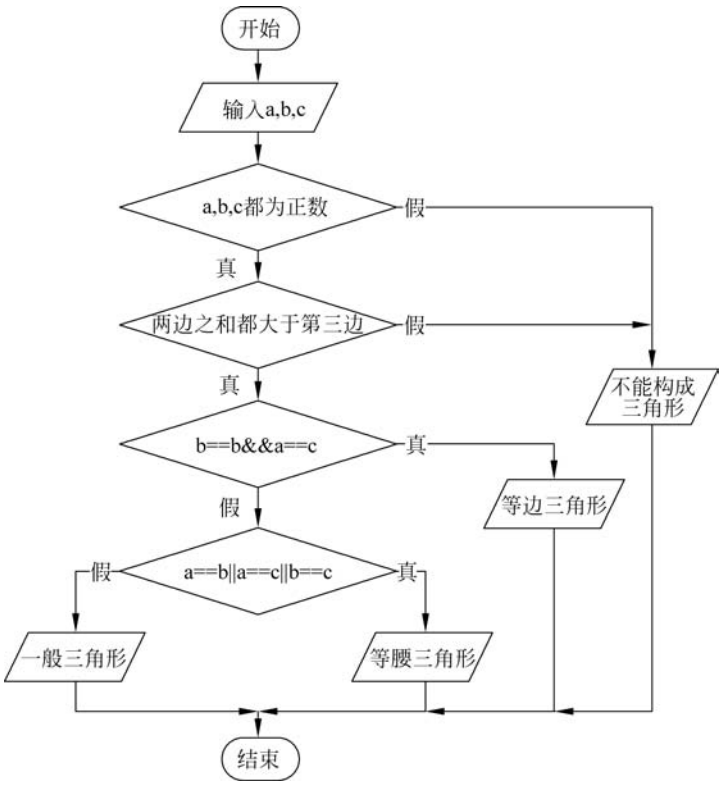

图 3-2 判断三角形类型的流程图

例 3.2 求 $n!$。

分析:从数学知识可知:$n!=1\times2\times3\times\cdots\times(n-1)\times n$,这是一个累计相乘,也就是说前一步的乘积可以作为后一步的一个乘数,反复进行,直到 n 为止。其流程如图 3-3 所示。

例 3.3 判断某一年是否为闰年。

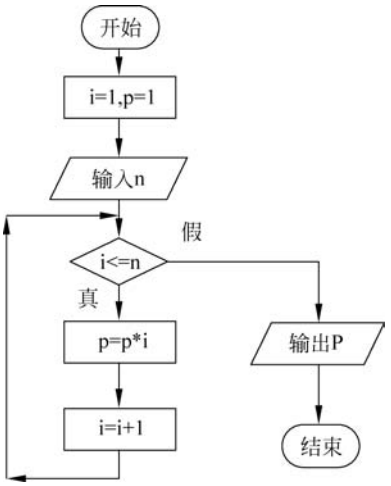

图 3-3　求 $n!$ 的流程图

分析：要判断某一年是否为闰年，用 4 来除年份，看是否能被整除，同时该年份不能被 100 整除；或者能被 100 整除，还要被 400 整除；这样的年份就是闰年，否则不是闰年。其流程如图 3-4 所示。

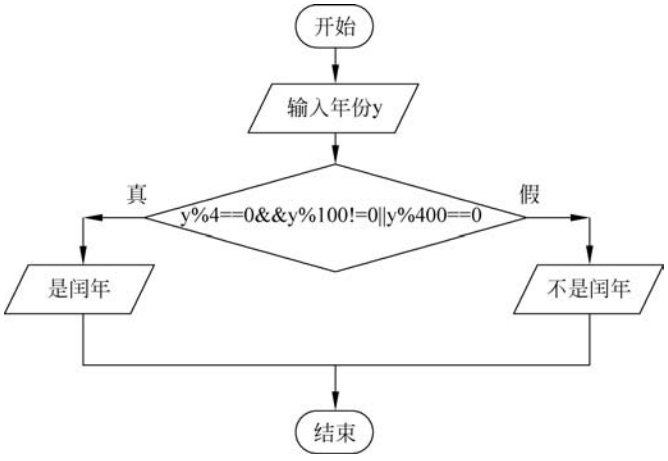

图 3-4　判断是否为闰年流程图

例 3.4　输入一个正整数，将它反位组成一个数输出（如输入 12345，输出 54321）。

分析：要输出反位数，首先要从低位数开始取出每一位数，可以用除以 10 取余数的方法得到；另外由于不知道输入的数具体的位数，可以采用不断整除 10 的方法逐步减少位数，直到为 0，这就是典型的数位分离的算法。其流程如图 3-5 所示。

例 3.5　判断一个整数 n 是否为素数。

分析：要判断一个数 n 是否为素数，很容易想到素数只能被 1 和 n 整除，而不能被其他数整除，因此可用 2 到 $n-1$ 之间的数来除 n，如果能被其中的一个数整除，n 就不是素数。但为了提高算法的效率，可采用这样来求解：只要 n 不能被 2 到$\sqrt{n}$之间的整数整除，n 就是素数。其流程如图 3-6 所示。

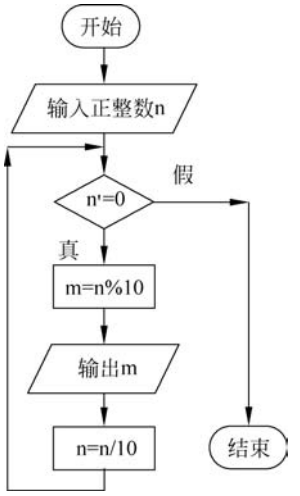

图 3-5　求正整数的反位数流程图

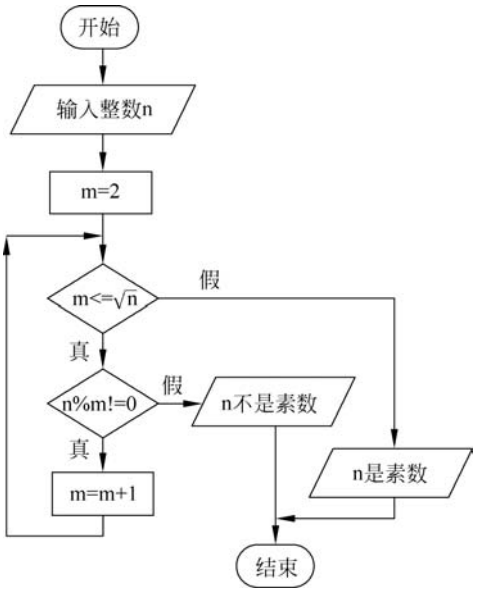

图 3-6　判断是否为素数

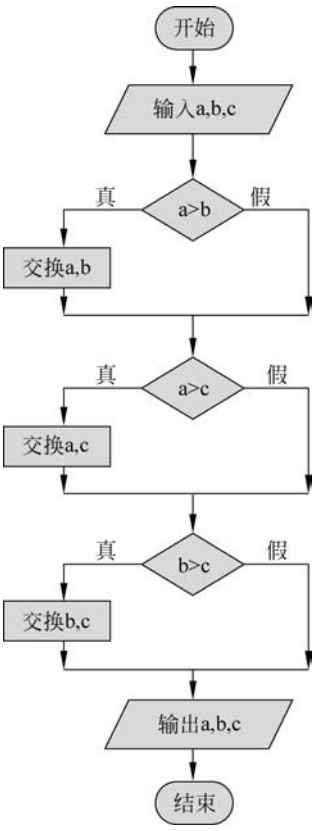

图 3-7　3 个数排序流程图

例 3.6　对 3 个整数 a、b、c 按从小到大排序。

分析：对 a、b、c 从小到大排序，就是要求 a 存放最小的数，b 存放第二小的数，c 存放最大的数。一般先将 a 和 b 进行比较，得到两者之间较小的数，存放在 a 中，再将 a 和 c 进行比较，将较小的数存放在 a 中，这时 a 就是 3 个数中的最小数。再将 b 和 c 比较，将较小的数给 b，较大的数给 c，这样 a、b、c 就按从小到大排序了。其流程如图 3-7 所示。

从上述实例可以看出算法具有如下特点。

(1) 有穷性：一个算法应包含有限的操作步骤。也就是说，在执行若干操作步骤之后，算法将结束，而且每一步都在合理的时间内完成。

(2) 确定性：算法中的每一步都应当有确切的含义，而不应当是含糊的、模棱两可的。对相同的输入必须得出相同的结果。

(3) 有零个或多个输入：算法可以有多个输入也可以没有输入，依实际的需要而定。

(4) 有一个或多个输出：算法的目的是求解，这些“解”只有通过输出才能得到。

(5) 可行性：算法中的每一步操作都应当能有效地执行并得到确定的结果。

3.1.4 三种基本结构

1966 年，Bohra 和 Jacopini 提出：任何复杂的算法都可以由顺序结构、选择(分支)结构和循环结构这 3 种基本结构组成。因此，在构造一个算法的时候，仅以这 3 种基本结构为单元，遵守 3 种基本结构的规范。3 种基本结构之间可以并列，可以相互包含，但不允许交叉，不允许从一个结构直接转到另一个结构的内部去。正因为整个算法都是由 3 种基本结构组成的，就像用模块构建的一样，所以结构清晰，易于正确性验证，易于纠错，这种方法就是结构化方法。遵循这种方法的程序设计，就是结构化程序设计。

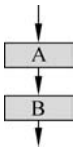

图 3-8　顺序结构

1. 顺序结构

顺序结构根据操作的先后顺序执行，如图 3-8 所示，在执行完 A 所指定的操作后，再执行 B 所指定的操作。

2. 选择(分支)结构

选择(分支)结构根据某个给定条件进行判断，条件为真或假时分别执行不同的操作。其基本形状有两种，分别如图 3-9 和图 3-10 所示。图 3-9 的执行过程为：当条件为真时执行 A，否则执行 B；图 3-10 的执行过程为：当条件为真时执行 A，否则什么也不做。

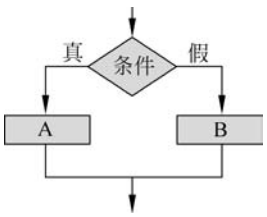

图 3-9　选择结构(1)

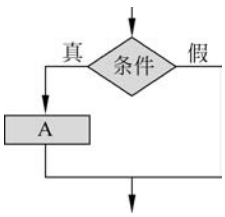

图 3-10　选择结构(2)

3. 循环结构

循环结构又称为重复结构，根据条件的真或假反复执行某些操作。循环结构又分为当型循环和直到型循环两种类型。

(1) 当型循环：当条件为真时，执行循环操作序列，然后判断条件是否为真，若为真则继续执行，如此反复；若条件为假时，就跳出循环，不执行循环序列，如图 3-11 所示。

(2) 直到型循环：先执行操作序列，再判断条件是否为真，若为真则继续执行循环操作序列，若为假，就跳出循环序列，如图 3-12 所示。

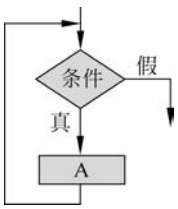

图 3-11　当型循环

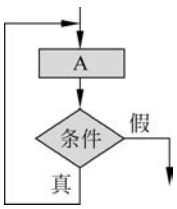

图 3-12　直到型循环

以上 3 种基本结构，都有如下共同的特点。

(1) 只有一个入口和一个出口。

(2) 结构内的每一部分都有可能被执行到。

(3) 结构内不存在“死循环”。

实践证明,由以上 3 种结构组成的算法结构,可以解决任何复杂的问题,在程序设计中要熟练掌握。除了用传统流程图表示算法外,还可以采用“伪代码”和“N-S 流程图”表示。

3.2 C 语言的标准输入和输出

在程序的运行过程中,往往需要用户输入一些数据,而程序运行所得到的计算结果又需要输出给用户,由此实现人与计算机之间的交互,因此在程序设计中,输入输出是必不可少的重要内容。在 C 语言中,没有专门的输入输出语句,所有的输入输出操作都是通过对标准输入/输出库函数的调用实现的。常用的输入输出函数有 scanf()、printf()、getchar()、putchar()、gets()和 puts(),下面分别介绍。

3.2.1 格式化输入输出

1. 格式化输出函数 printf()

printf()函数是一个标准库函数,它把信息输出到标准输出设备显示器上,它的函数原型在头文件 stdio.h 中,因此在使用该函数时,要求在程序开头加上 #include <stdio.h>,以保证不发生编译出错。printf()函数的调用格式如下:

```
printf("控制字符串",输出项列表)
```

输出项可以是常量、变量、表达式,其类型与个数必须与控制字符串中格式字符的类型、个数一致;当有多个输出项时,各项之间用逗号分隔。

控制字符串必须用双引号括起,由格式说明和普通字符两部分组成。

1) 格式说明

一般格式为:

```
%[<修饰符>]<格式字符>
```

格式字符规定了对应输出项的输出格式,常用的格式字符如表 3-1 所示。

表 3-1 C 语言常用输出格式字符

格式字符	含 义	举 例	结 果
c	按字符输出	char a=65; printf("%c",a);	A
d	按十进制整数输出	int a=567; printf ("%d",a);	567
u	按十进制无符号整数输出	int a=-1; printf("%u",a);	4294967295
f	按浮点数输出	float a=567.789; printf("%f",a);	567.789000
E 或 e	按指数形式输出	double a=567.789; printf("%e",a);	5.677890e+002
o	按八进制输出	int a=65; printf("%o",a);	101
X 或 x	按十六进制输出	int a=255; printf("%x",a);	ff
s	按字符串输出	printf("%s","ABC");	ABC
g	按 e、f 格式中较短的一种输出	float a=567.789; printf("%g",a);	567.789
p	输出变量在内存中的起始地址	int a; printf ("%p",&a);	0060FEFC

修饰符是可选的，用于确定数据输出的宽度、精度、小数位数、对齐方式等，用于产生更规范整齐的输出，当没有修饰符时，以上各项按系统缺省设定显示（小数按 6 位小数输出）。C 语言常用修饰符如表 3-2 所示。

表 3-2　C 语言常用修饰符

修饰符	含　义
m	输出数据域宽，数据长度＜m，左补空格；否则按实际输出
.n	对实数，指定小数点后位数（四舍五入）；对字符串，指定实际输出位数
－	输出数据在域内左对齐（默认为右对齐）
＋	指定在有符号数的正数前显示正号（＋）
0	输出数值时指定左面不使用的空位置自动填 0
#	在八进制和十六进制数前显示前导 0，0x
l	在 d、o、x、u 前，指定输出精度为 long 型；在 e、f、g 前，指定输出精度为 double 型

2）普通字符

普通字符包括可打印字符和转义字符。可打印字符主要是一些说明字符，这些字符按原样显示在屏幕上，如果有汉字系统支持，也可以输出汉字。对于不可打印的转义字符，它们是一些控制字符，控制产生特殊的输出效果。

例 3.7　printf()的用法。

```
#include<stdio.h>
#include<stlib.h>
int main()
{
    int x=1234;
    float f=123.456;
    double m=123.456;
    char ch='a';
    char a[]="Hello,world!"; /*定义一个字符数组来保存一个字符串*/
    int y=3,z=4;
    printf("%d %d\n",y,z);
    printf("y=%d,z=%d\n",y,z);
    printf("%8d,%2d\n",x,x);
    printf("%f,%8f,%8.1f,%.2f,%.2e\n",f,f,f,f,f);
    printf("%lf\n",m);
    printf("%3c\n",ch);
    printf("%s\n%15s\n%10.5s\n%2.5s\n%.3s\n",a,a,a,a,a);
    system("pause");
    return 0;
}
```

程序运行结果：

```
3 4
y=3,z=4
    1234,1234
123.456001,123.456001, 123.5,123.46,1.23e+002
123.456000
```

```
  a
Hello,world!
  Hello,world!
    Hello
Hello
Hel
```

2. 格式化输入函数 scanf()

scanf()函数是一个标准库函数,它是从标准输入设备——键盘上输入信息,它的函数原型在头文件 stdio.h 中,与 printf()函数相同,在使用该函数时,要求在程序开头加上 #include<stdio.h>,以保证不发生编译出错。scanf()函数的一般形式为:

```
scanf("控制字符串",地址表列);
```

其中,控制字符串的作用与 printf()函数相同,但不能显示非格式字符串,也就是不能显示提示字符串。地址表列中给出各变量的地址。地址是由地址运算符 & 后跟变量名组成的。例如,&a,&b 分别表示变量 a 和变量 b 的地址。这个地址就是编译系统在内存中给 a,b 变量分配的地址。在 C 语言中,使用了地址这个概念,这是与其他语言不同的。应该把变量的值和变量的地址这两个不同的概念区别开来。变量的地址是 C 编译系统分配的,用户不必关心具体的地址是多少。

控制字符串的两个组成部分为格式说明和普通字符。

1) 格式说明

格式说明规定了输入项中的变量以何种类型的数据格式被输入,形式是:

```
% [ <修饰符> ] <格式字符>
```

各修饰符是可选的,可以没有,这些修饰符是:

(1) 字段宽度。

例如:

```
scanf( "%3d",&a );
```

按宽度 3 输入一个整数赋给变量 a。例如,输入 12345,则 a=123。

(2) l 和 h。

可以和 d、o、x 一起使用,加 l 表示输入数据为长整数,加 h 表示输入数据为短整数,例如:

```
scanf("%10ld %hd",&x,&i);
```

则 x 按宽度为 10 的长整型读入,而 i 按短整数读入。例如,输入 12345678904321,则 x=1234567890,i=4321。

(3) 抑制字符 * 。

* 表示按规定格式输入但不赋予相应变量,作用是跳过相应的数据。例如:

```
scanf("%d %*d %d",&x,&y,&z);
```

执行该语句,若输入为 1□2□3(□表示空格,以后不再说明)。

结果为 x = 1,y = 3,z 未赋值,2 被跳过。上述语句等价于:

```
scanf("%d %*d %d",&x,&y);
```

2）普通字符

普通字符包括空格、转义字符和可打印(显示)字符。

(1) 空格。

在有多个输入项时，一般用空格或回车作为分隔符，若以空格作分隔符，则当输入项中包含字符类型时，可能产生非预期的结果，例如：

```
scanf("%d%c",&a,&ch);
```

输入 32□q

这时输出 a=32，ch=□，这是因为分隔符空格被读入并赋给 ch；若期望 a = 32，ch = q，可使用如下语句(在两个格式符之间插入一个空格符)：

```
scanf ( "%d□%c",&a,&ch );
```

此处%d 后的空格，就可跳过字符 q 前的所有空格，保证非空格数据的正确录入。

(2) 转义字符\ n 和\ t。

例如：

```
scanf ( "%d %d",&a,&b );
scanf ( "%d %d %d",&x,&y,&z );
```

输入为：

```
1□2□3 回车
4□5□6 回车
```

结果为：a = 1，b = 2，x = 3，y = 4，z = 5

若将上述语句改为

```
scanf("%d %d\n",&a,&b );
scanf("%d %d %d",&x,&y,&z );
```

对同样的输入，其结果为 a = 1，b = 2，x = 3，y = 4，z = 5，虽然在第一个 scanf 的最后有一个\n，但第二个 scanf 语句仍然依次读入数据，这是因为\n 为缺省分隔符之一，若 scanf("%d%d%d\n",&x,&y,&z)，则必须在最后输\n 字符，因此建议在 scanf()的格式字符串中最好不要加入不必要的字符。

(3) 可打印(显示)字符。

例如：

```
scanf("%d,%d,%c",&a,&b,&ch);
```

当输入为：1,2,q 回车

即：a = 1，b = 2，ch = q

若输入为 1□2□q 回车，除 a = 1 正确赋值外，对 b 与 ch 的赋值都将以失败告终。也就是说，这些不打印字符应是输入数据分隔符，scanf 在读入时自动去除与可打印字符相同的字符。

(4) 精度要求。

在用 scanf()函数输入浮点数时，不要在格式控制中指定小数部分的位数，否则编译时

会报错。例如 scanf("%5.2f",&f),编译器会报告错误。

(5) 在 ACM 程序设计大赛中,需要进行多组输入,直到按 Ctrl+Z 键后按 Enter 键才结束,一般可以利用带返回值的 scanf()函数来实现。具体用法是:scanf("%d",&a)!=0 或 scanf("%d",&a)!=EOF 作为循环条件,实现反复输入。

3.2.2 其他输入输出

1. putchar()函数

向标准输出设备输出一个字符,其调用格式为:

```
putchar(ch);
```

其中 ch 为一个字符变量或常量。putchar()函数的作用等同于 printf("%c",ch)。

例 3.8 putchar()函数应用。

```
#include<stdio.h>
#include<stlib.h>
int main()
{
    char c;
    c='B';
    putchar(c);        /*变量作为参数*/
    putchar('\x42');   /*字符作为参数*/
    putchar(0x42);     /*十六进制表示的 ASCII 码作为参数*/
    system("pause");
    return 0;
}
```

程序运行结果:

```
BBB
```

从本例中的连续 3 个字符输出函数语句可以了解字符变量的不同赋值方法。

2. getchar()函数

getchar()函数的功能为从键盘输入的一个字符,它不带任何参数。其调用格式为:

```
getchar();
```

getchar()函数的返回值就是从键盘上输入的字符,一般可以用一个字符变量得到这个返回值。

例 3.9 getchar()函数的应用。

```
#include<stdio.h>
#include<stlib.h>
int main()
{
    char ch;
    ch=getchar();
    putchar(ch);
    printf("%d\n",ch);
    system("pause");
```

```
    return 0;
}
```

程序运行结果：

```
输入字符 A 回车
A65
```

3. gets()函数

gets()函数的功能是从键盘上输入一个字符串，按 Enter 键结束输入(由于在 C 语言中没有字符串变量，把字符串保存在数组中，在第 5 章会讲到)。其调用格式为：

```
gets(a);                    /* 这里 a 为字符数组,定义形式为 char a[20]; */
```

例 3.10 gets()函数的应用。

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
    char a[20];
    gets(a);
    printf("%s\n",a);
    system("pause");
    return 0;
}
输入:Welcome to Nanjing 回车
输出:Welcome to Nanjing
```

4. puts()函数

puts()函数的功能是向屏幕输出一个字符串，并换行。

例 3.11 puts()函数的应用。

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
    char a[20];
    gets(a);
    puts(a);
    system("pause");
    return 0;
}
输入:Welcome to Nanjing 回车
输出:Welcome to Nanjing
```

说明：

(1) 在使用 getchar()函数输入时，要用 Enter 键来表示结束。

(2) getchar()得到的字符只能是第一个输入的字符，它可以赋值给字符变量，也可以不给任何变量，还可以作为表达式的一部分。

(3) gets()函数可以从键盘上输入一个带空格的字符串，按 Enter 键结束输入。

(4) puts()函数输出字符串后自动换行。

3.2.3 C 语言语句

和其他计算机编程语言一样,C 语言的语句也是用来完成一定操作,并通过编译产生计算机指令。C 语言的语句可分为以下 5 类。

(1) 控制语句:用来完成控制功能。

① if…else	分支语句
② for()、while()、do…while	循环语句
③ continue	结束本次循环语句
④ break	终止循环或多分支语句
⑤ switch	多分支语句(开关语句)
⑥ return	返回语句

(2) 函数调用语句:用来实现函数调用,由函数调用加一个分号构成。例如:

```
printf("Hello,World!\n");
```

(3) 表达式语句:由一个表达式加一个分号构成,最典型的是赋值表达式加一个分号构成赋值语句。例如:

```
x = 100/3;
```

(4) 空语句:由一个分号构成,例如:

```
;
```

它什么也不做,有时用来做循环语句的循环体,表示空循环;有时为了程序结构清晰。

(5) 复合语句:当一个语句不能完成某一功能,需要用多个语句才能实现,这时用花括号{}把这些语句括起来,构成复合语句,例如:

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

3.2.4 顺序结构程序设计

顺序结构的程序是按照语句出现的先后顺序来执行的,是程序设计中最简单的一种结构。下面通过一些简单的例子来说明顺序结构程序设计。

例 3.12 编写一个程序,从键盘上输入以秒为单位的时间,表示成小时分钟秒的形式。

```
/*
程序名称:ex3-12
  程序功能:小时分秒表示
*/
#include<stdio.h>
#include<stdlib.h>
int main()
```

```
{
    int s;
    scanf("%d",&s);
    printf("%d小时",s/3600);
    printf("%d分钟",(s%3600)/60);
    printf("%d秒\n",(s%3600)%60);
    system("pause");
    return 0;
}
```

程序运行结果：

```
输入 5000 回车
1 小时 23 分 20 秒
```

程序完全是按语句出现的先后顺序执行的，中间不存在任何控制流程的转移。

例 3.13 计算如图 3-13 所示的铁环的重量(已知铁的密度为 7.86g/cm^3)。

分析：要计算铁环的重量，就必须知道内径 $d2$ 和外径 $d1$，以及铁环的厚度 h，计算出体积，再乘以密度，就是铁环的重量。

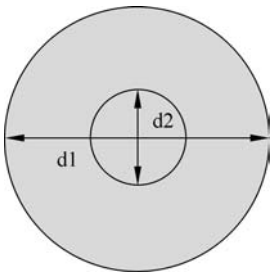

图 3-13 铁环示意图

```
/*
  程序名称:ex3-13
  程序功能:计算铁环的重量
*/
#include<stdio.h>
#include<stdlib.h>
#define RUO 7.86
#define PI 3.14159
int main()
{
    double d1,d2,h,s1,s2,w;
    scanf("%lf %lf %lf",&d1,&d2,&h);
    s1=PI*(d1/2)*(d1/2);
    s2=PI*(d2/2)*(d2/2);
    w=(s1-s2)*h*RUO;
    printf("铁环的重量为:%.2lf\n",w);
    system("pause");
    return 0;
}
```

程序运行结果：

```
输入 3.0□1.0□0.5
铁环的重量为:24.69
```

3.3 条件语句

条件语句是用来判断给定的条件是否满足(表达式值是否为 0)，并根据判断的结果(真或假)决定执行的语句，选择结构就是用条件语句来实现的。

3.3.1 if 语句

这是最简单的条件语句,实现基本的分支操作。

(1) 格式:

```
if(表达式)
{
    语句序列
}
```

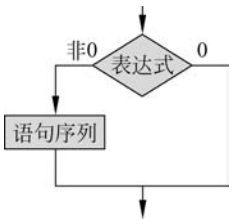

图 3-14 if 语句流程图

(2) 执行过程:先计算 if 后表达式的值,如果其值为非 0(真),则执行紧跟在 if 后面的语句序列;否则跳过该结构,执行后面的语句。其流程如图 3-14 所示。

说明:

① 表达式一般以关系或逻辑表达式为主,也可以是任何类型的表达式,只要值是非 0 就是真,是 0 就是假。

② 语句序列是实现分支操作的语句,既可以是单条语句,也可以是复合语句。当为单条语句时,表示复合语句的花括号{ }可以省略。

③ if 语句表达式必须书写在小括号()内,如果省略编译会出现语法错误。

例 3.14 从键盘输入一个整数,如果该整数为奇数则将其乘 3 加 1 后输出,如果为偶数则直接输出。

分析:要判断整数是否为奇数,只需要用该数除以 2 取余数,如果余数为 1 是奇数;是 0 为偶数。其流程如图 3-15 所示。

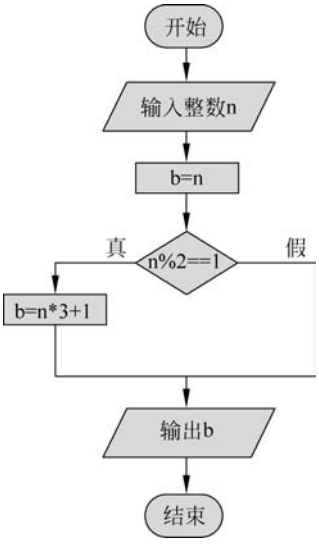

图 3-15 数据变换流程图

```
/*
  程序名称:ex3-14
  程序功能:处理整数
*/
#include<stdio.h>
#include<stdlib.h>
int main()
{
    int n,b;
    scanf("%d",&n);
    b=n;
    if(n%2==1) /*也可以用 if(n%2!=0)*/
        b=n*3+1;
    printf("处理的结果是:%d\n",b);
    system("pause");
    return 0;
}
```

程序运行结果:

```
① 输入 12
处理的结果是:12
```

② 输入 13
处理的结果是:40

例 3.15 从键盘输入 3 个整数 a、b、c,对这 3 个数从小到大排序。

根据例 3.6 的分析和流程,其程序如下:

```
/*
  程序名称:ex3-15
  程序功能:3 个整数排序
*/
#include<stdio.h>
#include<stdlib.h>
int main()
{
    int a,b,c,t;
    scanf("%d %d %d",&a,&b,&c);
    if(a>b)
    {
        t=a;
        a=b;   找到 a、b 两个数中较小的给 a
        b=t;
    }
    if(a>c)
    {
        t=a;
        a=c;   将 a 与 c 比较,较小的给 a,这时 a 就是 3 个数中最小的
        c=t;
    }
    if(b>c)
    {
        t=b;
        b=c;   将 b 与 c 比较,较小的给 b,这时 c 就是 3 个数中最大的
        c=t;
    }
    printf("%d,%d,%d\n",a,b,c);
    system("pause");
    return 0;
}
```

程序运行结果:

```
输入 45□21□70 回车
21,45,70
```

说明:本例在排序的过程中,要交换两个数,一般的做法是:设置一个临时变量(如本例中的变量 t)将一个变量的值先存放在这个临时变量里(如 $t=a$),再将要交换的变量 b 的值存放在 a 里,然后把 t 的值给 b。

思考:

① 如何实现 3 个整数的从大到小排序?

② 如何实现 3 个浮点数的排序?

③ 不用临时变量,如何实现两个数交换?

3.3.2 if…else 语句

if…else 语句是典型的条件语句,其流程如图 3-16 所示。

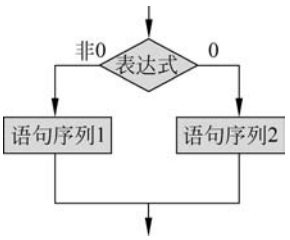

图 3-16 if…else 流程图

(1) 格式:

```
if(表达式)
{
    语句序列 1
}
else
{
    语句序列 2
}
```

(2) 执行过程:先计算 if 后表达式的值,如果其值为非 0(真),则执行语句序列 1,执行完以后,再执行该结构后面的语句;如果表达式的值为 0(假),则执行语句序列 2,执行完以后,再执行后面的语句。

例 3.16 从键盘上输入两个整数,求它们的最大值。其流程如图 3-17 所示。

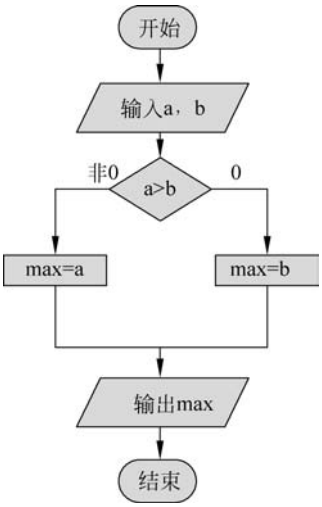

图 3-17 求最大值流程图

```
/*
  程序名称:ex3-16
  程序功能:求两数的最大值
*/
#include<stdio.h>
#include<stdlib.h>
int main()
{
    int a,b,max;
    scanf("%d %d",&a,&b);
    if(a>b)
        max=a;
    else
        max=b;
    printf("最大值:%d\n",max);
    system("pause");
    return 0;
}
```

程序运行结果:

```
输入 34□56 回车
最大值:56
```

说明:上面程序中,由于每个分支的语句序列都只有一个语句,因此把表示复合语句的{}省略了。

例 3.17 从键盘输入一个整数,如果该整数为奇数,则将其乘 3 加 1 后输出;如果为偶数,则除以 2 输出。其流程如图 3-18 所示。

```
/*
  程序名称:ex3-17
  程序功能:数据变换
*/
#include<stdio.h>
#include<stdlib.h>
int main()
{
    int n,b;
    scanf("%d",&n);
    if(n%2==1)
        b=3*n+1;
    else
        b=n/2;
    printf("变换后的数为:%d\n",b);
    system("pause");
    return 0;
}
```

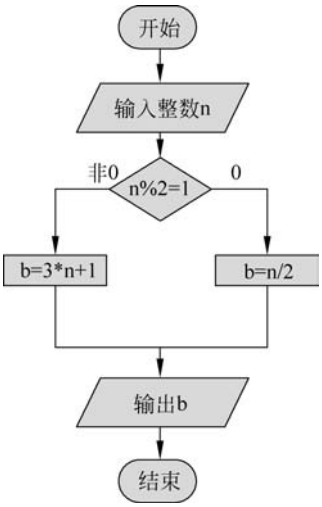

图 3-18　数据变换流程图

程序运行结果：

```
① 输入 24 回车
变换后的数为:12
② 输入 25 回车
变换后的数为:76
```

3.3.3　if…else if 语句

实际应用中常常面对更多的选择，这时，将 if…else 扩展一下，就得到 if…else if 结构。

(1) 格式：

```
if(表达式 1)
{
    语句序列 1
}
else if(表达式 2)
{
    语句序列 2
}
else if(表达式 3)
{
    语句序列 3
}
…
else
{
    语句序列 n
}
```

(2) 执行过程：先计算表达式 1 的值，若其值为非 0，则执行语句序列 1；否则计算表达式 2 的值，若表达式 2 的值为非 0，则执行语句序列 2；否则计算表达式 3 的值，若表达式 3

的值为非 0,则执行语句序列 3……直到计算表达式 $n-1$ 的值,若表达式 $n-1$ 的值为非 0,则执行语句序列 $n-1$,否则执行语句序列 n。其流程如图 3-19 所示。

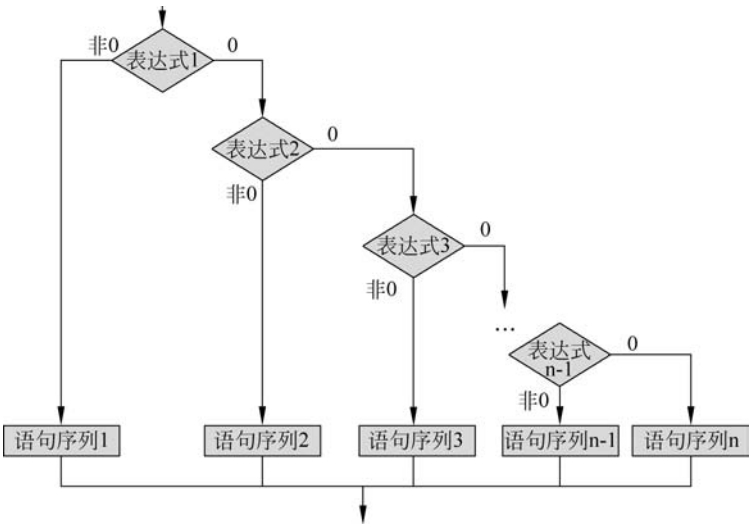

图 3-19　if…else if 流程图

说明:

① 语句序列为单条语句,表示复合语句的{}可以省略。

② if…else if 满足完全排斥的特性,绝不会出现某次执行了其中两路分支以上的情况。

③ if…else if 语句中的 else if 可以有有限多个,取决于编程的实际需求。

例 3.18　从键盘上输入字符,判断输入字符的类型。

分析:键盘上的字符有字母、数字、控制字符以及其他字符。如果是字母其范围在 A～Z 或 a～z;数字字符其范围在 0～9;控制字符其 ASCII 码小于 32;除了以上情况外,就是其他字符。其流程如图 3-20 所示。

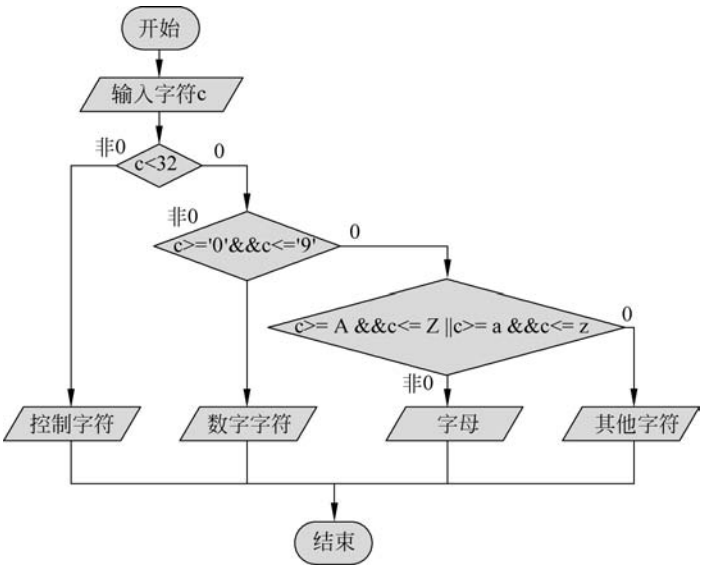

图 3-20　判断输入字符的类型的流程图

```
/*
  程序名称:ex3-18
  程序功能:判断字符的类型
*/
#include<stdio.h>
#include<stdlib.h>
int main()
{
    char c;
    printf("输入一个字符:");
    c=getchar();
    if(c<32) printf("是控制字符\n");
    else if(c>='0'&&c<='9') printf("是数字字符\n");
    else if(c>='A'&&c<='Z'||c>='a'&&c<='z') printf("是字母\n");
    else printf("是其他字符\n");
    system("pause");
    return 0;
}
```

程序运行结果：

① 输入一个字符:ctrl 回车,是控制字符
② 输入一个字符:A 回车,是字母
③ 输入一个字符:9 回车,是数字字符
④ 输入一个字符:/ 回车,是其他字符

例 3.19 已知 2021 年 7 月 1 日为星期四,从键盘上输入 1～31 的整数,按下述格式输出该日是星期几的信息在对应栏下。

```
2021 年 7 月日历
Sun Mon Tue Wed Thu Fri Sat
-----------------------
            1
```

分析：根据日历的特点,周日～周六分别用 0～6 的整数表示,知道每月 1 号是星期几后,用“(输入日期+每月 1 号星期几-1)%7”得到的值,就是星期几,再按照指定的格式输出。

```
/*
  程序名称:ex3-19
  程序功能:输出日历
*/
#include<stdio.h>
#include<stdlib.h>
int main()
{
    int date,weekday,original_date=6;
    scanf("%d",&date);
    if(date<1||date>31)
    {
        printf("数据输入错误!\n");
        exit(0);
```

```
    }
    weekday = (date + original_date - 1) % 7;
    printf("2021 年 5 月日历\n");
    printf(" ----------------------------------\n");
    printf("Sun Mon Tue Wed Thu Fri Sat \n");
    printf(" ----------------------------------\n");
    if(weekday == 0)
        printf(" % 2d\n",date);
    else if(weekday == 1)
        printf(" % 7d\n",date);
    else if(weekday == 2)
        printf(" % 12d\n",date);
    else if(weekday == 3)
        printf(" % 17d\n",date);
    else if(weekday == 4)
        printf(" % 22d\n",date);
    else if(weekday == 5)
        printf(" % 27d\n",date);
    else
        printf(" % 32d\n",date);
        system("pause");
    return 0;
}
```

程序运行结果：

```
输入 20 回车
2021 年 7 月日历
 ------------------------------------------------------
Sun     Mon     Tue     Wed     Thu     Fri     Sat
 ------------------------------------------------------
                20
```

例 3.20 输入学生的成绩,输出学生的等级：90～100(优)、80～89(良)、70～79(中)、60～69(及格)、60 以下(不及格)。

分析：输入学生成绩在 0～100,其流程如图 3-21 所示。

```
/*
  程序名称:ex3 - 20
  程序功能:学生成绩等级划分
*/
#include < stdio.h >
#include < stdlib.h >
int main()
{
    int cj;
    scanf(" % d",&cj);
    if(cj < 0||cj > 100)
    {
        printf("数据输入错误\n");
        exit(0);
```

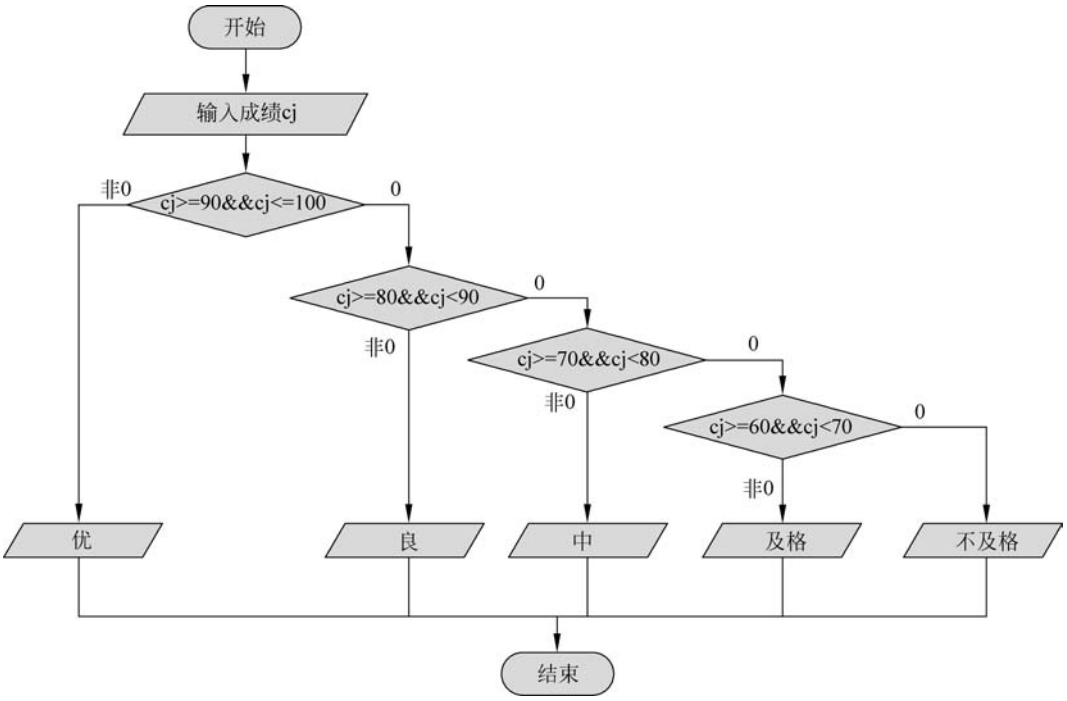

图 3-21　学生成绩等级流程图

```
    }
    if(cj>=90&&cj<=100)
        printf("优\n");
    else if (cj>=80&&cj<90)
        printf("良\n");
    else if(cj>=70&&cj<80)
        printf("中\n");
    else if(cj>=60&&cj<70)
        printf("及格\n");
    else
        printf("不及格\n");
    system("pause");
    return 0;
}
```

程序运行结果：

```
① 输入 101 回车
数据输入错误!
② 输入 85 回车
良
```

3.3.4　条件语句的嵌套

在 if 条件语句中又包含一个或多个 if 条件语句称为条件语句的嵌套。主要有以下几种形式。

(1) 嵌套具有 else 子句的 if 语句。

```
if (表达式 1)
    if (表达式 2)
        语句序列 1
    else                    } 内嵌 if
        语句序列 2
```

(2) 嵌套不含 else 子句的 if 语句。

```
if (表达式 1)
    语句序列 1
else
    if(表达式 2)
        语句序列 2          } 内嵌 if
```

(3) 一般形式。

```
if (表达式 1)
   if (表达式 2)
        语句序列 1
   else                     } 内嵌 if
        语句序列 2
else
   if(表达式 3)
        语句序列 3
   else                     } 内嵌 if
        语句序列 4
```

说明：if 和 else 必须正确配对，其配对原则是：当缺省{ }时，else 总是和它上面离它最近的未配对的 if 配对。例如：

```
if(…)
if(…)
if(…)
…
else…
else…
else…
```

不正确的配对可能会出现逻辑错误甚至语法错误。例如：

```
if (a == b)
        if(b == c)
             printf("a == b == c");
         else
             printf("a!= b");
```

这时，else 是和第二个 if 配对，若希望 else 与第一个 if 配对，则可修改为

```
if (a == b)
{
       if(b == c)
          printf("a == b == c");
```

```
}else
  printf("a!= b");
```

所以,加{}可以改变 else 和 if 的配对。

例 3.21 判断两个数的大小关系。

```
/*
  程序名称:ex3-21
  程序功能:判断两个数的大小关系
*/
#include<stdio.h>
#include<stdlib.h>
int main()
{
      int x,y;
    printf("输入两个整数 x,y:");
    scanf("%d,%d",&x,&y);
    if(x!= y)
        if(x>y)
            printf("x>y\n");
        else
            printf("x<y\n");
    else
        printf("x==y\n");
    system("pause");
    return 0;
}
```

程序运行结果:

```
① 输入两个整数 x,y:12,23 回车
x<y
② 输入两个整数 x,y:20,10 回车
x>y
③ 输入两个整数 x,y:12,12 回车
x==y
```

例 3.22 [问题 1]判断三角形的形状:从键盘输入 3 个数,判断这 3 个数能否构成一个三角形的 3 条边;如果能构成三角形,判断所构成三角形的形状。

```
/*
  程序名称:ex3-22
  程序功能:判断三角形的类型
*/
#include<stdio.h>
#include<stdlib.h>
int main()
{
        double a,b,c;
        scanf("%lf %lf %lf",&a,&b,&c);
        if(a<=0||b<=0||c<=0)
        {
```

```
            printf("三角形的边长不能为 0 或负,请重新!\n");
            exit(0);
        }
        if(a + b <= c||a + c <= b||b + c <= a)
        {
            printf("不能构成三角形,请重新!\n");
            exit(0);

        }
        else if(a == b&&b == c)
            printf("是等边三角形\n");
        else if(a == b||b == c||a == c)
            printf("是等腰三角形\n");
        else
            printf("是一般三角形\n");
    system("pause");
    return 0;
}
```

程序运行结果:

```
输入: 0 1 2
      三角形的边长不能为 0 或负,请重新!
输入: 1 2 3
      不能构成三角形,请重新!
输入: 2 2 2
      是等边三角形
输入: 2 2 3
      是等腰三角形
输入: 3 4 6
      是一般三角形
```

思考:如何判断是锐角三角形、直角三角形或钝角三角形?

3.3.5 条件语句的应用

上面讲述了条件语句的格式、执行过程以及应用时的注意事项。下面通过一些实例讲述条件语句的应用。

例 3.23 输入年份,判断是否为闰年。

根据例 3.3 的分析和图 3-4 所示流程,其程序如下:

```
/*
  程序名称:ex3 - 23
  程序功能:判断闰年
*/
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int year;
    scanf("%d",&year);
```

```
    if(year <= 0)
    {
        printf("数据输入错误!\n");
        exit(0);
    }
    if(year % 4 == 0&&year % 100!= 0||year % 400 == 0)
        printf(" % d 是闰年!\n",year);
    else
        printf(" % d 不是闰年!\n",year);
    system("pause");
    return 0;
}
```

程序运行结果：

```
① 输入 2013 回车
2013 不是闰年!
② 输入 2012 回车
2012 是闰年!
③ 输入 - 100 回车
数据输入错误!
```

例 3.24 从键盘输入一元二次方程 $ax^2+bx+c=0$ 的系数 a、b、c，求它的根。

分析：根据数学知识，一般用求根公式 $x_{1,2}=\dfrac{-b\pm\sqrt{b^2-4ac}}{2a}$ 求解，但必须满足是一元二次方程才能使用公式求解，因此要判断 a 的值是否为 0，判别式 $\Delta=b^2-4ac\geqslant 0$。其流程如图 3-22 所示。

```
/*
  程序名称:ex3 - 24
  程序功能:求解一元二次方程
*/
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int main()
{
    double a,b,c,delta,x1,x2,p,q;
    scanf(" % lf  % lf  % lf",&a,&b,&c);
    if(a == 0)
        printf("不是一元二次方程!\n");
    else
    {
        delta = b * b - 4 * a * c;
        if(delta == 0)
        {
            printf("方程有两个相等的实数根!\n");
            x1 = - b/(2 * a);
            x2 = x1;
            printf(" % .2lf ,     % .2lf\n",x1,x2);
        }
```

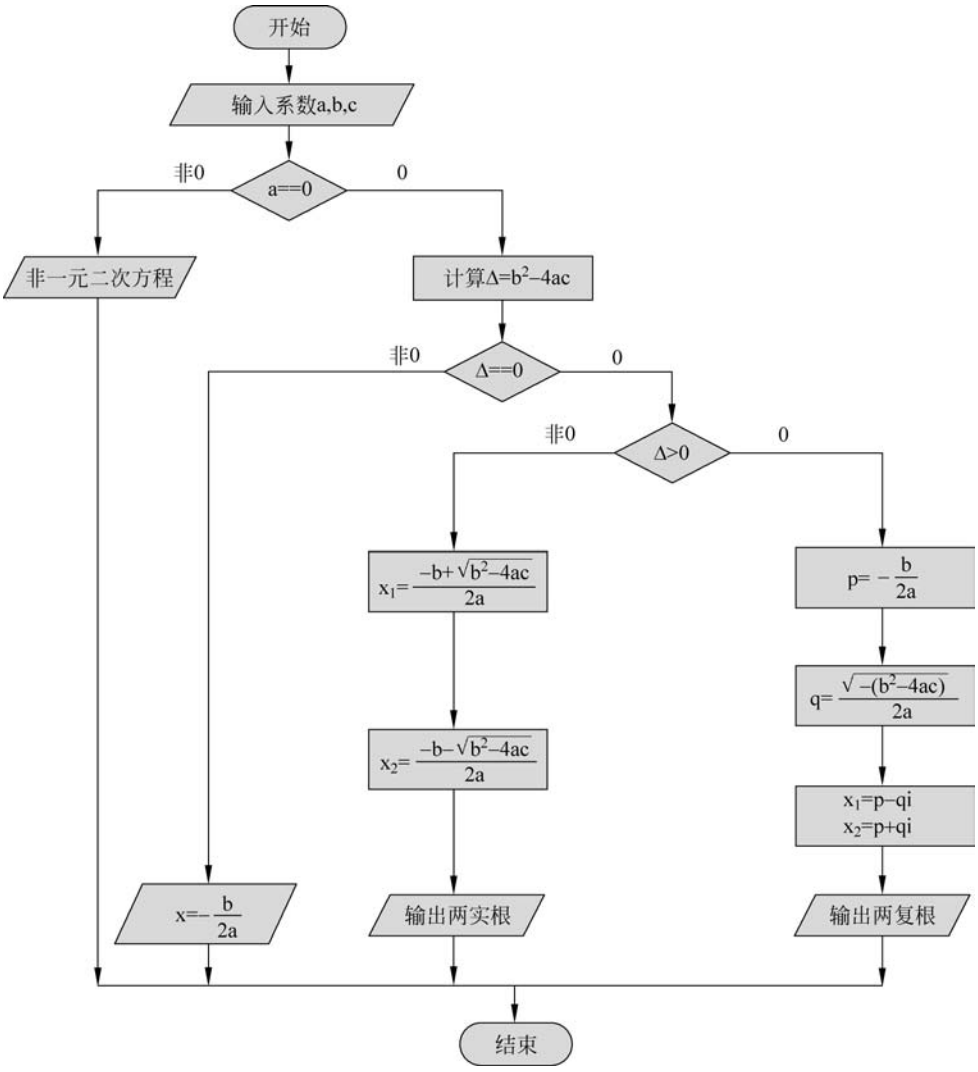

图 3-22　求一元二次方程的根的流程图

```
else if(delta > 0)
{
    printf("方程有两个不相等的实数根!\n");
    x1 = - b/(2 * a) + sqrt(delta)/(2 * a);
    x2 = - b/(2 * a) - sqrt(delta)/(2 * a);
    printf(" % .2lf ,     % .2lf\n",x1,x2);
}
else
{
    printf("方程有两个不相等的复数根!\n");
    p = - b/(2 * a);
    q = sqrt( - delta)/(2 * a);
    printf(" % .2lf +  % .2lfi\n",p,q);
    printf(" % .2lf -  % .2lfi\n",p,q);
}
```

```
    }
system("pause");
    return 0;
}
```

程序运行结果：

```
① 输入 0□2□3 回车
  不是一元二次方程
② 输入 1□2□1 回车
  方程有两个相等的实数根!
  -1.00 -1.00
③ 输入 3□5□2
方程有两个不相等的实数根!
  -0.67 -1.00
④ 输入 1□2□3 回车
方程有两个不相等的复数根!
-1.00+1.41i -1.00-1.41i
```

说明：由于开平方根函数 sqrt()的声明包含在 math.h 中，因此在程序的前面要增加#include<math.h>。一般数学处理函数的声明都在 math.h 中，因此用到数学函数都要增加一条这样的包含语句。另外，在有的编译器中，还必须要求 sqrt()函数的参数为浮点型数据，才能编译通过。

3.4 多分支语句

多分支语句是根据某一条件在很多选项中选择其中的一个来执行。在 C 语言中，实现多分支可用 if…else if 和 switch 语句，switch 语句可使程序更加简洁、清晰。

3.4.1 switch 多分支语句

(1) 多分支语句 switch 的格式：

```
switch( 表达式)
{
    case E1:
                语句序列 1;
    case E2:
                语句序列 2;
          …
    case En:
                语句序列 n;
    [default:
                默认语句序列;]
}
```

(2) 执行过程：首先执行 switch 后面括号中表达式的值，然后将该值与常量 E1、E2……En 的值进行逐一比较，若与某个值相等，则执行 case 子句后的语句序列；若没有相同值，则转向 default 子句，执行默认语句序列。其流程如图 3-23 所示。

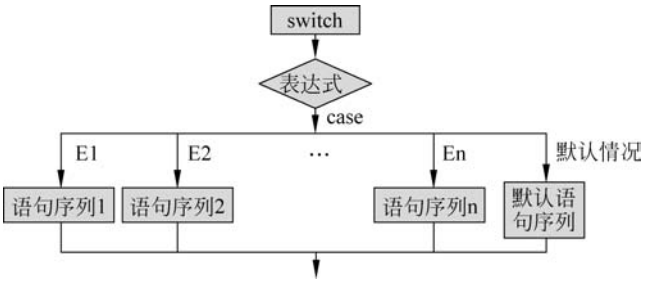

图 3-23　switch 语句流程图

说明：

① switch 后面表达式的值必须是整型或字符型。

② E1,E2,…,En 是常量表达式,且值必须互不相同。

③ 每个 case 语句的冒号后面可以是 0 条或多条语句,多条语句时,可以不加{}。

④ 各 case 的顺序可以是任意的。

⑤ 允许多个 case 语句使用同一语句序列。例如：

```
case 1:
case 2:
case 3:printf("Hello,World!\n");
```

⑥ default 语句不是必需的,但建议使用,一般放在最后。

⑦ 每个 case 后面语句序列里的 break 语句可有可无,但执行效果不同。当有 break 语句时,遇到 break 语句,程序就跳出这一层 switch 语句结构,转到其结构后面的语句执行;当没有 break 语句时,程序就会一直执行下去,直到遇到 break 语句或该 switch 结构结束。

例 3.25　用 switch…case 语句重新编写例 3.18 的程序。

分析：由于学生的成绩有很多种情况,本题的关键就是要把这些成绩变换到少数几种情况,可以用学生成绩整除 10 得到。

```
/*
  程序名称:ex3-25
  程序功能:用 switch 语句求学生成绩等级
*/
#include<stdio.h>
#include<stdlib.h>
int main()
{
    int score;
    scanf("%d",&score);
    if(score<0||score>100)
    {
        printf("输入数据错误\n");
        exit(0);
    }
    switch(score/10)
    {
        case 10:
```

```
        case 9:printf("优秀\n");break;
        case 8:printf("良好\n");break;
        case 7:printf("中等\n");break;
        case 6:printf("及格\n");break;
        default:printf("不及格\n");break;
    }
    system("pause");
    return 0;
}
```

和例 3.20 的程序对比可知：用 switch 编写的程序要简单清晰很多。

3.4.2 多分支语句的嵌套

一个 switch 语句中嵌套一个或多个 switch 语句、if 语句称为 switch 语句的嵌套。一般有以下几种形式。

1. switch 语句嵌套 switch 语句

```
switch(…)
{
    …
    switch(…)
    {
        语句序列 1
    }
    语句序列 2
}
```

在这种结构中，内层的 switch 语句的 break 语句只能跳出本层的 switch 结构，然后执行语句序列 2。

2. switch 语句嵌套 if 语句

```
switch(…)
{
    …
    if(…)
    {
        …
    }
    else
    {
        …
    }
    语句序列
}
```

3. if 语句嵌套 switch 语句

```
if(…)
{
    …
```

```
switch(…)
    {
        …
    }
else
{
    switch(…)
    {
        …
    }
}
```

3.4.3 多分支语句应用

多分支语句可以用 switch 语句和 if…else if 语句来实现,它们绝大多数情况下可以相互转换。例如,根据学生成绩,给出学生成绩的等级,既可以用 if…else if 语句来实现,又可以用 switch 语句来实现。下面列举一些它们的应用。

例 3.26 已知银行整存整取存款不同期限的年利率分别为:

$$\text{年息}=\begin{cases}2.25\% & \text{期限 1 年}\\ 2.79\% & \text{期限 2 年}\\ 3.33\% & \text{期限 3 年}\\ 3.60\% & \text{期限 5 年}\\ 4.14\% & \text{期限 8 年}\\ 0.30\% & \text{活期}\end{cases}$$

要求输入本金和期限,求到期后能从银行得到的利息与本金的合计。

分析:银行根据存期的不同,储户享受不同的利率,一共有 6 种不同的存期,就相当于 6 个分支,从键盘上输入本金和存期,可以根据如下公式计算储户到期后从银行获得的总金额:

$$\text{总金额}=\text{本金}\times\text{年利率}\times\text{存期}+\text{本金}$$

```
/*
  程序名称:ex3-26
  程序功能:根据存期和本金计算总金额
*/
#include<stdio.h>
#include<stdlib.h>
int main()
{
    int year;
    double money,rate,total;
    printf("输入存款和存期:");
    scanf("%lf %d",&money,&year);
    switch(year)
    {
        case 1:rate=0.0225;break;
        case 2:rate=0.0279;break;
        case 3:rate=0.0333;break;
```

```
            case 5:rate = 0.0360;break;
            case 8:rate = 0.0414;break;
            default:rate = 0.003;break;
        }
        total = money + money * rate * year;
        printf("从银行获得的总金额为:%.2lf\n",total);
        system("pause");
        return 0;
    }
```

运行结果：

```
① 输入存款和存期:10000□3 回车
从银行获得的总金额为:10999.00
② 输入存款和存期:10000□4 回车
从银行获得的总金额为:10120.00
```

例 3.27 从键盘上输入年份和月份，求该月有多少天？

分析：一年中每月的天数，除了 2 月份受是否闰年的影响要变化外，其他月份的天数都不变。因此，根据输入的年份判断是否是闰年，如果是闰年，那么 2 月份就有 29 天；否则 2 月份就有 28 天。

```
/*
  程序名称:ex3-27
  程序功能:输入年份和月份,求该月天数
*/
#include<stdio.h>
#include<stdlib.h>
int main()
{
    int year,month,day,leapyear;
    scanf("%d %d",&year,&month);
    if(year<0||month<1||month>12)
    {
        printf("输入的数据错误!\n");
        exit(0);
    }
    leapyear = year%4 == 0&&year%100!= 0||year%400 == 0; /*是否为闰年*/
    switch(month)
    {
        case 1:
        case 3:
        case 5:
        case 7:
        case 8:
        case 10:
        case 12:day = 31;break;
        case 4:
        case 6:
        case 9:
        case 11:day = 30;break;
```

```
            case 2:day = 28 + leapyear;break;
        }
        printf(" % d 年 % d 月的天数为: % d\n",year,month,day);
        system("pause");
        return 0;
}
```

运行结果:

① 输入 2013 3 回车
2013 年 3 月的天数为:31
② 输入 2020 2 回车
2020 年 2 月的天数为:29

3.5 循 环 语 句

程序中经常需要对某些操作对象进行同样的操作,这种操作就是循环(或重复)。循环语句是程序中的一个基本语句。循环语句可以在编程时只写很少的语句,让计算机反复执行,从而完成大量同类型的操作。C 语言中提供了 while、do…while、for 三种语句构成循环结构。

3.5.1 while 循环语句

while 语句构成的循环是当型循环。

(1) 格式:

```
while(表达式)
{
    语句序列
}
```

(2) 执行过程:当表达式的值为非 0 时,执行{}中的语句序列,然后再计算表达式的值;若为非 0,继续执行{}中的语句序列,如此反复,直到表达式的值为 0,则执行结构后面的语句,其流程如图 3-24 所示。

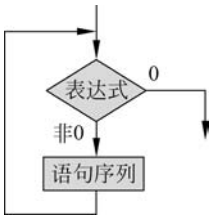

图 3-24 while 循环流程图

说明:

① 语句序列称为循环体,当为一条语句时,表示复合语句的{}可以省略。

② 表达式可以为任何类型。

③ 其特点是先判断,后执行。若条件不成立,有可能一次也不执行。

④ 语句序列中必须有改变 while 后面括号的表达式值的语句,否则有可能是死循环。

例 3.28 学生成绩统计:小王作为班上的学习委员,每门课程考试后,任课老师都会让他统计成绩。老师并不关心每个人的具体成绩,而只关心参加考试的人数、平均分、最低分和最高分这 4 项指标,编写程序来帮小王完成这项任务。

分析：这是累加求和，求成绩平均以及找出最大值和最小值，首先要定义存放全班总分 s、平均值 avg、最大值 max 和最小值 min 等变量，并给赋初值（一般和为 0，乘积为 1），输入数 x，判断 x 是否为负数，如果为负则结束循环，否则就累加；同时用一个整型变量 k 统计人数。其流程如图 3-25 所示。

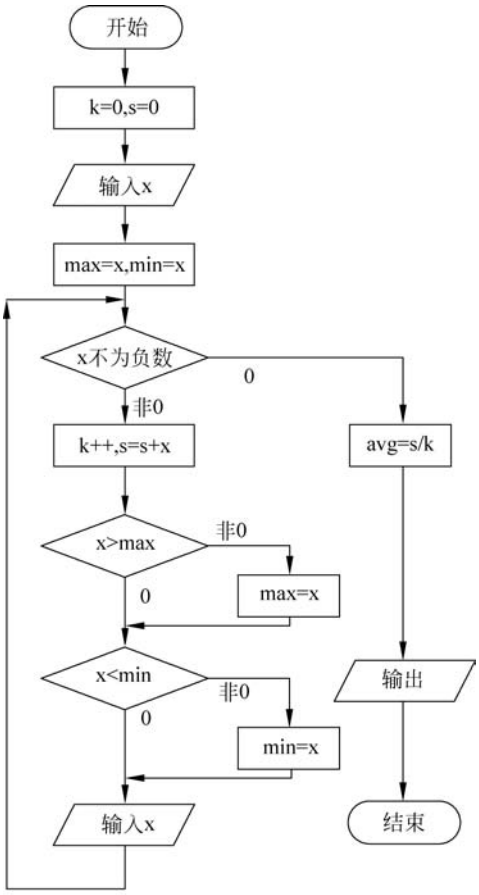

图 3-25　统计学生成绩流程图

```
/*
   程序名称:ex3-28
   程序功能:统计学生考试人数,并求最高分、最低分和平均分
*/
#include<stdio.h>
#include<stdlib.h>
int main()
{
    int k=0,x,max,min,s=0;
    double avg=0;
    scanf("%d",&x);
    max=x;
    min=x;
    while(x>=0)
    {
        s=s+x;
        k++;
        if(x>max) max=x;
        if(x<min) min=x;
        scanf("%d",&x);
    }
    if(k>0)
    {
    avg=(double)s/k;
    printf("学生人数=%d,平均分=%.2lf,最高分=%d,最低分=%d\n",k,avg,max,min);
    }
    system("pause");
    return 0;
}
```

程序运行结果：

```
输入 60 65 70 80 85 56 75 88 90 10 -1 回车(数字之间用空格分开或直接回车)
学生人数=10,平均分=67.90,最高分=90,最低分=10
```

例 3.29　输入一个正整数，将它反位组成一个新的数输出（如输入 12345，组成 54321 输出）。

分析：要组成一个反位数，首先要进行数位分离，即除以 10 取余，再整除 10，直到被除数为 0。同时每次对分离出来的数位与前一次分离出来的数位乘 10 相加。最后得到的这个数就是反位数。流程如图 3-26 所示。

```
/*
  程序名称:ex3-29
  程序功能:输入整数,求反位数
*/
#include<stdio.h>
#include<stdlib.h>
int main()
{
    int n,m,t=0;
    scanf("%d",&n);
    if(n<=0)
    {
        printf("数据输入错误!\n");
        exit(0);
    }
    while(n!=0)
    {
        m=n%10; /*取出最低位数*/
        t=t*10+m;
        n=n/10;
    }
    printf("反位数为:%d\n",t);
    system("pause");
    return 0;
}
```

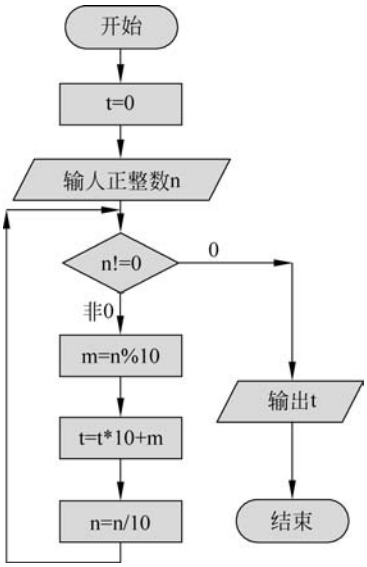

图 3-26　求反序数流程图

运行结果:

① 输入 12345 回车
反位数为:54321
② 输入 -1234 回车
数据输入错误!

一般情况下,while 型循环语句最适合于知道控制循环的条件为某个表达式的值,而且该表达式的值会在循环中被改变。所以 while 语句又称为“先判断,后执行”循环。

3.5.2　do…while 循环语句

do…while 语句构成的循环是直到型循环。

(1) 格式:

```
do
{
    语句序列
} while(表达式);
```

(2) 执行过程:先执行{}中的语句序列一次,再计算 while 括号中的表达式,如果表达式的值非 0,则继续执行{}中语句序列,直到表达式的值为 0,结束循环,执行该结构后面的语句。其流程如图 3-27 所示。

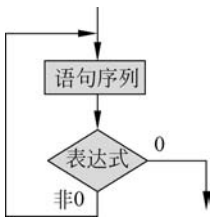

图 3-27　do…while 流程图

说明：

① 语句序列称为循环体，当为一条语句时，表示复合语句的{}可以省略。

② 表达式可以为任何类型。

③ 其特点是先执行，后判断，若条件不成立，也要执行一次。

④ 语句序列中必须有改变 while 后面括号的表达式值的语句，否则有可能是死循环。

⑤ while(表达式)后面的；不能少。

例 3.30 用 do…while 语句重新编写例 3.26 的程序。

```
/*
  程序名称:ex3-30
  程序功能:用 do…while 编写统计学生成绩的程序
*/
#include<stdio.h>
#include<stdlib.h>
int main()
{
    int k=0,x,max,min,s=0;
    double avg=0;
    max=0;
    min=100;
    do
    {
        scanf("%d",&x);
        if(x>=0)
        {
            s=s+x;
            k++;
            if(x>max) max=x;
            if(x<min) min=x;
        }
    }while(x>=0);
    if(k>0)
    {
    avg=(double)s/k;
    printf("学生人数=%d\n 班级平均分=%.2lf\n 最高分=%d\n 最低分=%d\n",k,avg,
max,min);
    }
system("pause");
return 0;
}
```

运行结果：

```
输入 60 65 70 80 85 56 75 88 90 10 -1 回车(数字之间用空格分开或直接回车)
学生人数=10
平均分=67.90
最高分=90
最低分=10
```

与用 while 语句编写的程序运行的结果完全一样。

例 3.31 从键盘上输入两个整数,求它们的最大公约数。

分析:求最大公约数有很多方法,其中辗转相除求最大公约数是编程中常用的方法。其流程如图 3-28 所示。

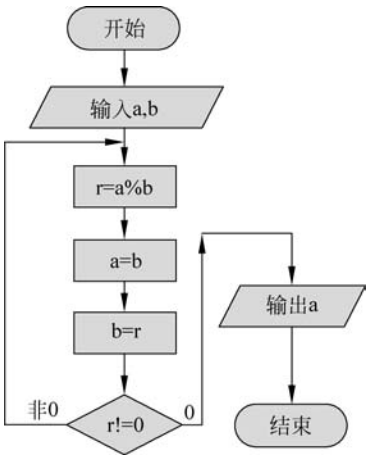

图 3-28 求最大公约数流程图

```
/*
  程序名称:ex3 - 31
  程序功能:求两个数的最大公约数
*/
#include < stdio.h >
#include < stdlib.h >
int main()
{
    int a,b,r;
    scanf(" % d % d",&a,&b);
    do
    {
        r = a % b;
        a = b;
        b = r;
    }while(r!= 0);
    printf("最大公约数为: % d\n",a);
    system("pause");
    return 0;
}
```

程序运行结果:

```
输入 8□12 回车
最大公约数为:4
```

从上面的例题可以看出,while 语句和 do…while 语句可以相互转换,其主要区别在于 while 语句是先判断后执行,只要不满足条件,循环体语句根本不会被执行;而 do…while 语句是先执行后判断,不管条件是否满足,循环体语句总会执行一次。例如:

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

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

当输入小于或等于 10 的数时，两个程序的运行结果是一样的。但输入 11，则左边程序的输出是 11，而右边程序的输出是 0。

3.5.3 for 循环语句

for 语句是循环控制结构中使用较为广泛的一种循环控制语句，特别适合已知循环次数的情况。

(1) 格式：

```
for (<表达式 1>;<表达式 2>;<表达式 3>)
{
    语句序列
}
```

(2) 执行流程：先计算表达式 1 的值，该表达式是首次进入循环前执行一次；然后计算表达式 2 的值，如果表达式 2 的值为非 0，则执行一次{}中的语句序列，否则不执行{}中的语句序列，结束循环转到{}后面的语句执行。执行完一次{}中的语句序列后，计算表达式 3 的值，然后再计算表达式 2 的值，整个流程由此一直重复下去。其流程如图 3-29 所示。

说明：

① 语句序列称为循环体。

② 当语句序列只有单条语句，表示复合语句的{}可以省略。

③ 表达式 1 一般为赋值表达式，给控制变量赋初值；如果省略表达式 1，这时 for 语句为如下格式：

```
表达式 1
for (;<表达式 2>;<表达式 3>)
{
    语句序列
}
```

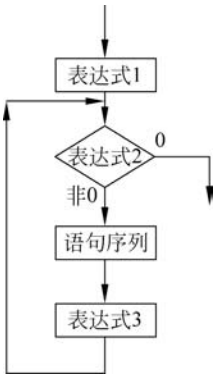

图 3-29 for 语句流程图

④ 表达式 2 一般为关系表达式或逻辑表达式，称为循环控制条件；如果省略表达式 2，for 语句格式为：

```
for (表达式 1;;<表达式 3>)
{
    语句序列
}
```

这时相当于条件永远是真(表达式 2 的值为非 0)，语句序列中必须有结束循环的语句，否则就会形成死循环。

⑤ 表达式 3 一般为赋值表达式，给控制变量增量或减量；如果省略表达式 3，for 语句格式为：

```
for (表达式 1;<表达式 2>;)
{
    语句序列
    表达式 3
}
```

⑥ for(;;)也是合法的 for 语句,这时与无限的 while 语句等价。

例 3.32 输入正整数 $n(n<13)$,计算 $1!+2!+\cdots+n!$。

分析:本题首先要计算阶乘,但前一数的阶乘与下一个数相乘可得下一个数的阶乘,$(i+1)!=i!\times(i+1)$,其流程如图 3-30 所示。

```
/*
  程序名称:ex3-32
  程序功能:阶乘的和
*/
#include<stdio.h>
#include<stdlib.h>
int main()
{
    int i,sum=1,p=1,n;      /*sum=1 是避免 n 为 0*/
    scanf("%d",&n);
    if(n<0)
    {
        printf("负数没有阶乘!\n");
        exit(0);
    }
    for(i=2;i<=n;i++)
    {
        p=p*i;
        sum=sum+p;
    }
    printf("阶乘的和为:%d\n",sum);

    system("pause");
    return 0;
}
```

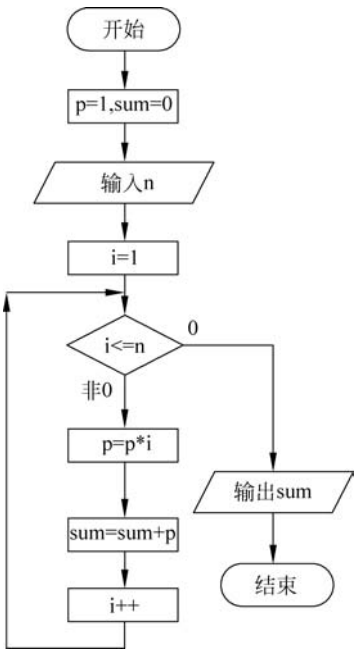

图 3-30 求阶乘的和的流程图

程序运行的结果:

① 输入 5 回车
阶乘的和为:153
② 输入 - 2 回车
负数没有阶乘

思考:当 n 大于等于 13,会出现什么结果?有什么解决的办法?

例 3.33 从键盘输入一个正整数,判断该数是否为素数。

根据例 3.5 的算法流程,其程序如下:

```
/*
  程序名称:ex3-33
  程序功能:判断素数
*/
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
int main()
{
```

```
    int i,m,n;
    scanf("%d",&n);
    if(n<=0)
    {
        printf("输入数据错误!\n");
        exit(0);
    }
    m=sqrt(1.0*n);
    for(i=2;i<=m;i++)
        if(n%i==0)break;          /*break跳出循环*/
    if(i>m&&n!=1)
        printf("%d是素数\n",n);
    else
        printf("%d不是素数\n",n);
    system("pause");
    return 0;
}
```

程序运行结果：

① 输入17回车
17是素数
② 输入56回车
56不是素数
③ 输入-23回车
输入数据错误

思考：为什么要判断$i>m\&\&n!=1$?

例3.34 求所有三位数的水仙花数。

分析：三位数的水仙花数是这样一个数：对于一个三位整数，其各位数的立方和等于该数。如$153=1^3+5^3+3^3$，则153就是水仙花数。因此对每一个三位数进行数位分离。个位数等于该数除以10取余数，百位数等于该数整除100，十位数=(该数-100×百位数)/10。

```
/*
  程序名称:ex3-34
  程序功能:求水仙花数
*/
#include<stdio.h>
#include<stdlib.h>
int main()
{
    int i,m,n,k;
    for(i=100;i<1000;i++)
    {
        m=i/100;              /*求百位数*/
        n=(i-100*m)/10;       /*求十位数*/
k=i%10;                       /*求个位数*/
        if(i==m*m*m+n*n*n+k*k*k)
```

```
            printf("%d",i);
        }
        system("pause");
        return 0;
}
```


程序运行结果：

```
153 370 371 407
```

思考：求十位数字还有哪些方法？

3.5.4 循环语句的嵌套

一个循环语句的循环体中又包含循环语句，称为循环语句的嵌套。被嵌套的循环当然还可以嵌套循环，形成多重循环结构，实际编程中循环的嵌套应用非常广泛。一般有以下几种形式：

```
1. while()
   { …
              while()
              { …
              }
         …
   }
2. do
   { …
              do
              { …
              }while();
   …
   }while();
3. while()
   { …
              do
              { …
              }while();
       ….
   }
4. for(;;)
   { …
              do
              { …
              }while();
         …
           while()
           { …
           }
         …
    }
```

说明：

① 三种循环可互相嵌套，层数不限。

② 外层循环可包含两个以上内循环，但不能相互交叉。

③ 嵌套循环的执行流程：外层循环执行一层，内层循环要执行完。

④ 嵌套循环的跳转：只能跳转出本层循环。

⑤ 禁止从外层跳入内层、禁止跳入同层的另一循环和向上跳转。

例 3.35 编写 C 语言程序，按下面格式输出乘法九九表。

乘法九九表

```
-----------------------------------
    1   2   3   4   5   6   7   8   9
-----------------------------------
1   1   2   3   4   5   6   7   8   9
2   2   4   6   8   10  12  14  16  18
3   3   6   9   12  15  18  21  24  27
4   4   8   12  16  20  24  28  32  36
5   5   10  15  20  25  30  35  40  45
6   6   12  18  24  30  36  42  48  54
7   7   14  21  28  35  42  49  56  63
8   8   16  24  32  40  48  56  64  72
9   9   18  27  36  45  54  63  72  81
-----------------------------------
```

分析：外循环 i 控制行输出，内循环 j 控制列输出，输出结果为 i * j，其流程如图 3-31 所示。

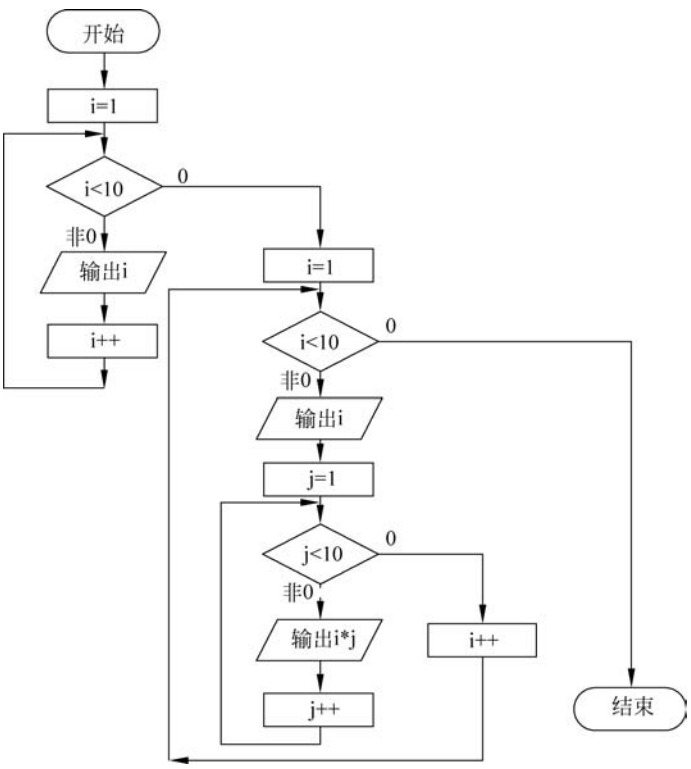

图 3-31 乘法九九表流程图

```
/*
  程序名称:ex3-35
  程序功能:输出乘法九九表
*/
#include<stdio.h>
#include<stdlib.h>
int main()
{   int i,j;
    printf("\n----------------------------------------\n");
        printf(" ");          /*保证后面的输出对齐*/
    for(i=1;i<10;i++)
        printf("%4d",i);
    printf("\n----------------------------------------\n");
    for(i=1;i<10;i++)
    {
        printf("%d",i);
        for(j=1;j<10;j++)
            printf("%4d",i*j);
        printf("\n");
    }
    printf("----------------------------------------\n");
    system("pause");
    return 0;
}
```

程序运行结果:

```
----------------------------------
   1   2  3  4  5  6  7   8   9
----------------------------------
1   1   2  3  4  5  6  7   8   9
2   2   4  6  8  10 12 14  16  18
3   3   6  9  12 15 18 21  24  27
4   4   8  12 16 20 24 28  32  36
5   5   10 15 20 25 30 35  40  45
6   6   12 18 24 30 36 42  48  54
7   7   14 21 28 35 42 49  56  63
8   8   16 24 32 40 48 56  64  72
9   9   18 27 36 45 54 63  72  81
----------------------------------
```

例 3.36 百钱买百鸡:一百元钱买了一百只鸡,其中母鸡一只 5 元钱、公鸡一只 3 元钱、小鸡一只 0.5 元钱,问每一种鸡都必须要买的情况下,计算所有的购买方法。

分析:这是一道古典数学问题,设 i、j、k 分别为购买母鸡、公鸡和小鸡的数量,可列出三元一次方程组,解三个未知数,两个方程的不定方程,由于每一种都要买,则 i 的取值范围为 1～19、j 的取值范围为 1～31、k 的取值范围为 1～98,对于这个问题可以用穷举的方法,遍历 i、j、k 的所有可能组合,最后得到问题的解。在计算机中用循环来解决这一类问题。其流程如图 3-32 所示。

$$\begin{cases} i+j+k=100 \\ 5i+3j+k/2=100 \end{cases}$$

```
/*
  程序名称:ex3-36
  程序功能:百钱买百鸡
*/
#include<stdio.h>
#include<stdlib.h>
int main()
{
    int i,j,k;
    for(i=1;i<=19;i++)
        for(j=1;j<=31;j++)
            for(k=1;k<=98;k++)
                if(i+j+k==100&&5*i+3*j+0.5*k==100)
                    printf("母鸡数:%d,公鸡数:%d,小鸡数:%d\n",i,j,k);
    system("pause");
    return 0;
}
```

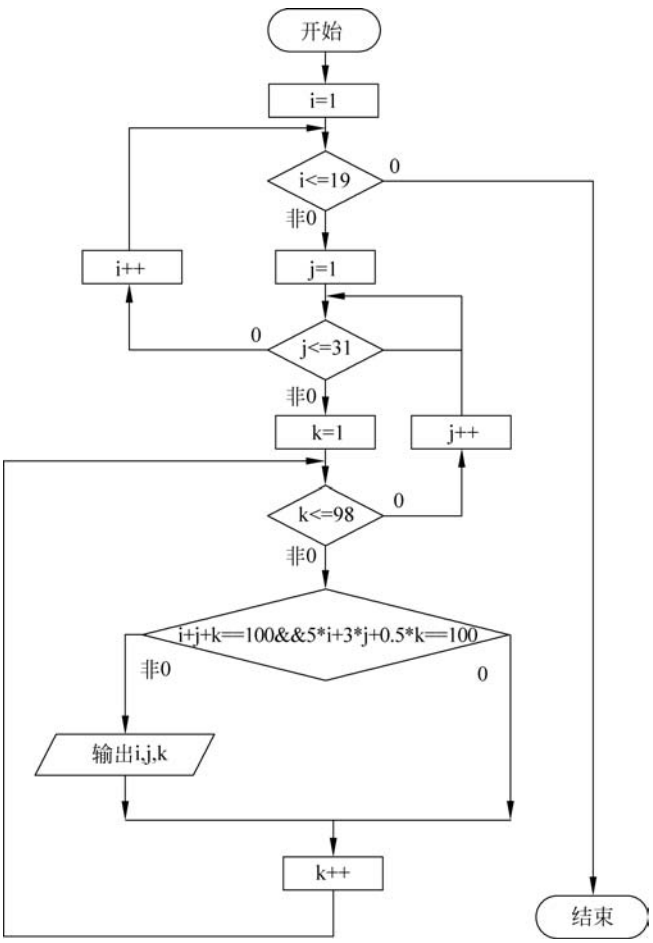

图 3-32　百钱买百鸡流程图

程序运行结果：

```
母鸡数:5,公鸡数:11,小鸡数:84
母鸡数:10,公鸡数:2,小鸡数:88
```


如果把条件修改一下，可以不用三重循环，而用二重循环就可以解决。因为公鸡和母鸡数确定后，小鸡数也就确定了，即 $k=100-i-j$。

循环语句允许嵌套，分支语句允许嵌套，循环与分支语句之间也可以嵌套，程序就是这样由简单的语句构造成各种复杂结构。由于任何复杂的程序都是由顺序、分支和循环三种基本结构组成的，因此熟练掌握分支结构和循环结构，对于解决较复杂的问题非常重要。对于复杂的程序，经常要使用表示复合语句的{}，许多初学者往往写了左侧的花括号，而忘记右侧的花括号，以致造成错误，所以采用缩进书写是减少花括号丢失的一个好办法。

3.6 转移语句

当要改变程序的执行顺序，可以用转移语句来实现，在 C 语言中主要有 goto、break 和 continue 三种转移语句。

3.6.1 goto 语句

goto 语句是无条件转移语句，其一般形式为

```
goto 语句标号
```

语句标号用标识符表示，用来表示程序的某个位置。goto 语句的功能就是无条件地把程序转到语句标号所在的位置开始执行。

计算机语言中的 goto 语句曾引起过很大的争议，许多人主张不使用 goto 语句，理由是它使程序难于控制，也大大降低了程序的可读性。而另外一些人主张保留 goto 语句，因为它解决一些特定的问题时会很方便。对于一个熟练的程序员，如果可以使程序更清晰的话，可有保留地使用。对于初学者最好不使用 goto 语句。本书不对 goto 语句进行进一步讨论。

3.6.2 continue 和 break 语句

当需要在循环体中提前跳出循环，或者在满足某种条件下，不执行循环中剩下的语句而立即从头开始新的一轮循环，这时就要用到 break 或 continue 语句。

1. continue 语句

只能用于循环结构中，当遇到 continue 语句，程序就跳过循环体中位于该语句后的所有语句，提前结束本次循环并开始新一轮循环。

(1) 格式：

```
continue;
```

(2) 功能：结束本次循环，开始下一次循环。其流程如图 3-33 所示。

说明：continue 只能用在循环结构中，而不能用于其他控制结构。

例 3.37 输出 100～200 不能被 3 整除的数。

```
/*
  程序名称:ex3-37
  程序功能:输出不能被 3 整除的数
*/
#include<stdio.h>
#include<stdlib.h>
int main()
{
    int k;
    for(k=100;k<=200;k++)
    {
        if(k%3==0)
            continue;
        printf("%d",k);
    }
    system("pause");
    return 0;
}
```

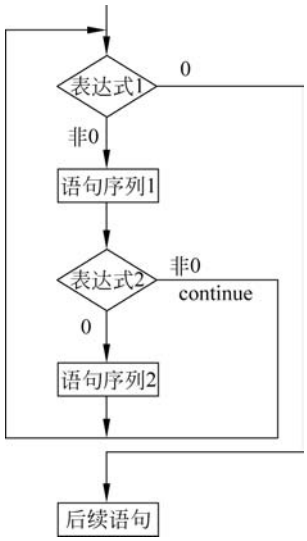

图 3-33 continue 流程图

运行结果为：

```
100  101  103  104  106  107  109  110  112  113  115  116  118  119  121  122
124  125  127  128  130  131  133  134  136  137  139  140  142  143  145  146
148  149  151  152  154  155  157  158  160  161  163  164  166  167  169  170
172  173  175  176  178  179  181  182  184  185  187  188  190  191  193  194
196  197  199  200
```

2. break 语句

在前面学习 switch 语句时，break 语句的作用是在 case 子句执行完后，通过 break 语句跳出 switch 结构。在循环语句中，break 语句的作用是结束本层循环，转而执行本层循环语句后的语句(即跳出本层循环)。其流程如图 3-34 所示。

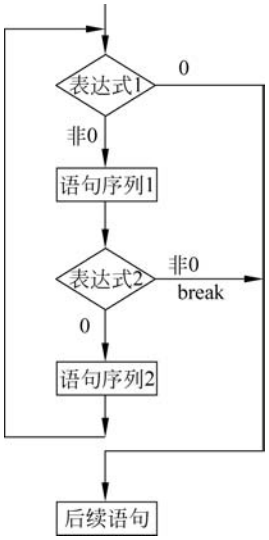

图 3-34 break 流程图

(1) 格式：

```
break;
```

(2) 功能：跳出 switch 结构或结束本层循环。

说明：break 语句只能用于 switch 或循环结构中。

例 3.38 输出半径为 1～10 的圆的面积，若面积超过 100，则不输出。

```
/*
  程序名称:ex3-38
  程序功能:输出圆的面积
*/

#include<stdio.h>
#include<stdlib.h>
#define PI 3.1415926
int main()
```

```
{
    int r;
    double area;
    for(r = 1;r < = 10;r++)
    {
        area = PI * r * r;
        if(area > 100.0)
            break;
        printf(" %.2lf",area);
    }
    system("pause");
    return 0;
}
```

程序运行结果为：

```
3.14 12.57 28.27 50.27 78.54
```

3.7 综 合 应 用

例 3.39 验证哥德巴赫猜想：任意一个充分大的偶数，可以用两个素数之和表示。例如：

```
4 = 2 + 2
6 = 3 + 3
    …
98 = 19 + 79
```

分析：哥德巴赫猜想是世界著名的数学难题。自从计算机出现后，人们就开始用计算机去尝试解各种各样的数学难题。先不考虑怎样判断一个数是否为素数，而是从整体上对这个问题进行考虑，可以这样做：读入一个偶数 n，将它分成 p 和 q，使 $n=p+q$。可以令 p 从 2 开始，每次加 1，而令 $q=n-p$，如果 p、q 均为素数，则正为所求，否则令 $p=p+1$ 再试。为了判断 p、q 是否是素数，设置两个标志量 p_flag 和 q_flag，初始值为 0，若 p 是素数，令 p_flag=1，若 q 是素数，令 q_flag = 1。流程如图 3-35 所示。

```
/*
  程序名称:ex3-39
  程序功能:验证哥德巴赫猜想
*/
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
int main()
{
    int i,p,q,n,p_flag,q_flag;
    scanf(" %d",&n);
    if((n%2==1)||n<4)
```

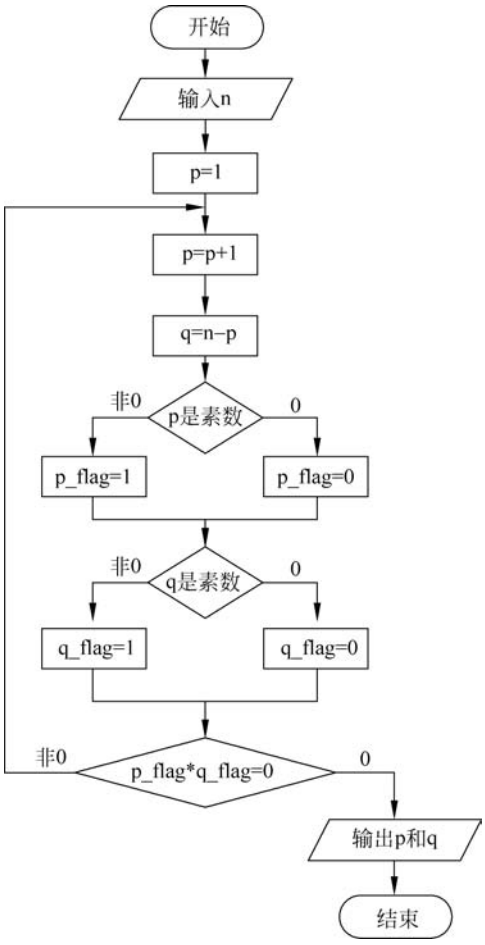

图 3-35　哥德巴赫猜想流程图

```
{
    printf("数据输入出错\n");
    exit(0);
}
p = 1;
do
{
    p = p + 1;
    q = n - p;
    p_flag = 1;
    for(i = 2;i <= sqrt(1.0 * p);i++)
    {
        if(p % i == 0)
        {
            p_flag = 0;
            break;
        }
    }
    q_flag = 1;
```

```
        for(i = 2;i <= sqrt(1.0 * q);i++)
        {
            if(q % i == 0)
            {
                q_flag = 0;
                break;
            }
        }
    }while(p_flag * q_flag == 0);
    printf(" % d =  % d +  % d\n",n,p,q);
    system("pause");
    return 0;
}
```

程序运行结果:

```
输入 20 回车
20 = 3 + 17
```

思考:该算法有什么地方需要改进?

例 3.40 求斐波那契(Fibonacci)数列前 40 项,该数列的通项公式如下:

$$\begin{cases} F_1 = 1 & (n = 1) \\ F_2 = 1 & (n = 2) \\ F_n = F_{n-1} + F_{n-2} & (n > 2) \end{cases}$$

分析:这是一道古典数学问题,有一对小兔子,生长期是一个月,再经过一个月后生了一对小兔子,出生的小兔子经过一个月也成为成年兔子,再经过一个月后也生了一对小兔子,当然原来的成年兔子每个月都要生一对小兔子,如此下去……假设所有兔子都不死,问题就变成求每个月兔子的总对数。就可以得到数列 1,1,2,3,5,8,13,…,即从第 3 个数开始,该数是前面两个数之和。其流程如图 3-36 所示。

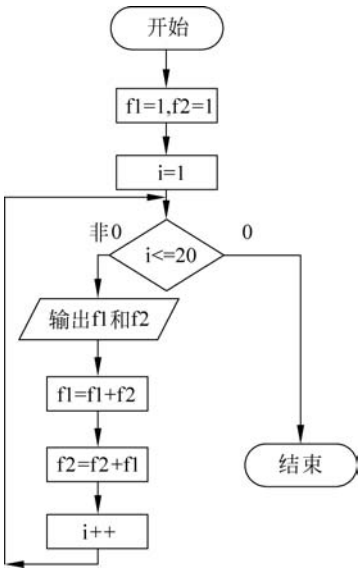

图 3-36 Fibonacci 数列流程图

```
/*
  程序名称:ex3 - 40
  程序功能:求 Fibonacci 数列
*/
#include < stdio.h >
#include < stdlib.h >
int main()
{
    int i,f1 = 1,f2 = 1;
    for(i = 1;i <= 20;i++)
    {
        printf(" % 12d  % 12d",f1,f2);
        if(i % 2 == 0)        /* 每行输出 4 个数 */
            printf("\n");
        f1 = f1 + f2;
        f2 = f2 + f1;
    }
    system("pause");
    return 0;
}
```

程序运行结果：

```
       1        1        2        3
       5        8       13       21
      34       55       89      144
     233      377      610      987
    1597     2584     4181     6765
   10946    17711    28657    46368
   75025   121393   196418   317811
  514229   833040  1346269  2178309
 3524578  5702887  9227465 14930352
24157817 39088169 63245986 102334155
```

思考：题目要求计算前 40 项，为什么只循环了 20 次？

例 3.41 判断一个正整数是否为回文数。回文数是这样的数：一个正整数从左往右读和从右往左读都是一样的数(如 121，123321)。

分析：要判断正整数 n 是否为回文数，把 n 进行数位分离，原来的最高位变为最低位，原来的最低位变为最高位，组成一个新的数 m，比较 m 和 n 是否相等，若相等则是回文数，否则不是回文数。其流程如图 3-37 所示。

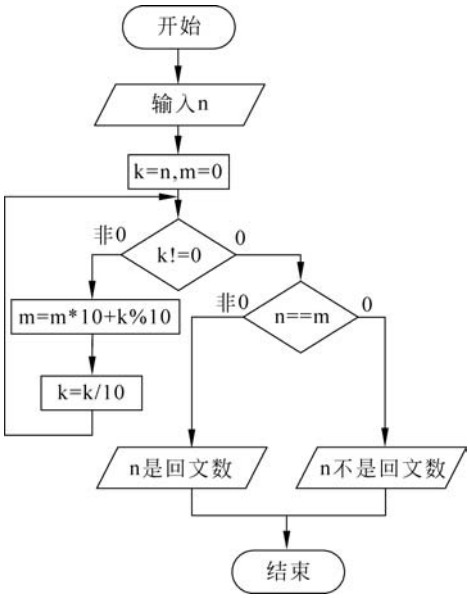

图 3-37 判断回文数流程图

```
/*
  程序名称:ex3-41
  程序功能:判断回文数
*/
#include<stdio.h>
#include<stdlib.h>
int main()
{
```

```
    int n,m = 0,k;
    do
    {
        scanf(" % d",&n);
        if(n < = 0)
            printf("数据输入错误,请重新输入!\n");
    }while(n < = 0);
    k = n;
    while(k!= 0)
    {
        m = m * 10 + k % 10;
        k = k/10;
    }
    if(n == m)
        printf(" % d 是回文数!\n",n);
    else
        printf(" % d 不是回文数!\n",n);
    system("pause");
    return 0;
}
```

程序运行结果:

① 输入 - 123
输入数据错误,请重新输入!
② 输入 123
123 不是回文数!
③ 输入 123321
123321 是回文数!

例 3.42 求分数数列$\frac{2}{1},-\frac{3}{2},\frac{5}{3},-\frac{8}{5},\cdots$,前 n 项之和。

分析:对于求分数数列的和,要分析分子和分母的构成以及前后项之间的关系,从这个数列来看,后一项的分子是前一项分子和分母之和,后一项的分母是前一项的分子,而且正负项间隔出现。其流程如图 3-38 所示。

```
/*
  程序名称:ex3 - 42
  程序功能:分数数列求和
*/
#include < stdio.h >
#include < stdlib.h >
int main()
{
    int a = 2,b = 1,s = 1,i,n;              /* s 表示首项的符号变量 */
    double sum = 0.0;
    scanf(" % d",&n);
    for(i = 1;i < = n;i++)
    {
        sum = sum + s * a/(double)b;        /* 也可以写成 sum = sum + s * 1.0 * a/b */
        a = a + b;
```

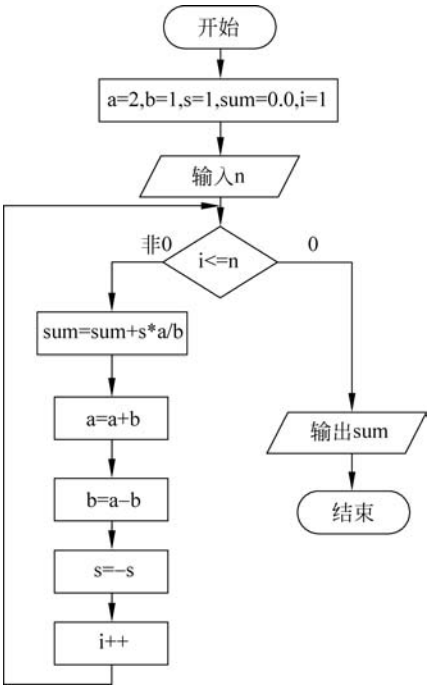

图 3-38　求分数数列之和的流程图

```
        b = a - b;
        s = - s;
    }
    printf("数列的和为: % lf\n",sum);
    system("pause");
    return 0;
}
```

程序运行结果：

```
输入:20
数列的和为:0.577922
```

注意：s= -s 是用来实现项之间的正负间隔，由于整数相除的结果是整数，因此要进行数据类型转换。

例 3.43　求一元高次方程 $2x^3-4x^2+3x+6=0$ 在 1.5 附近的根。

分析：对于高次方程，由于没有公式进行求根，只能采取迭代的方法，一般用牛顿迭代法，设 $f(x)=2x^3-4x^2+3x+6$，$f'(x)$ 为 $f(x)$ 的导数，其迭代公式为：$x_{n+1}=x_n-\dfrac{f(x_n)}{f'(x_n)}$，当 $|x_{n+1}-x_n|<\varepsilon$ 时，则 x_{n-1} 为方程的根。其流程如图 3-39 所示。

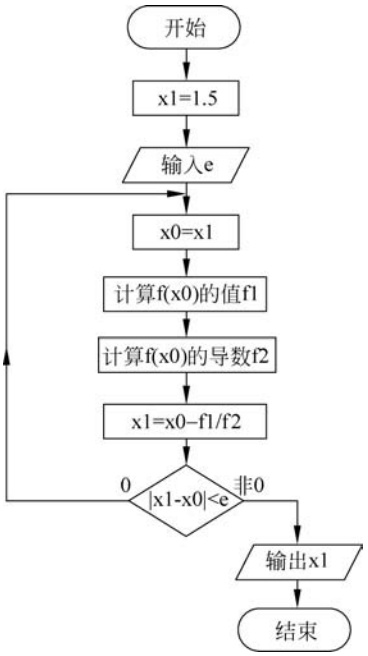

图 3-39　求高次方程根的流程图

```
/*
  程序名称:ex3-43
  程序功能:求高次方程的根
*/
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
int main()
{
    double x0,x1=1.5,e,f1,f2;
    scanf("%lf",&e);
    do
    {
        x0=x1;
        f1=2*x0*x0*x0-4*x0*x0+3*x0+6;
        f2=6*x0*x0-8*x0+3;
        x1=x0-f1/f2;
    }while(fabs(x1-x0)>=e);
    printf("方程的根为:%lf\n",x1);
    system("pause");
    return 0;
}
```

程序运行结果:

```
输入 0.000001
方程的根为:- 0.801207
```

例 3.44 确定小偷:甲、乙、丙、丁 4 人为偷窃嫌疑犯,只有一个是真正的小偷,在审讯过程,4 人都有可能说真话或假话:

甲:乙没有偷、丁偷的;

乙:我没有偷,丙偷的;

丙:甲没有偷,乙偷的;

丁:我没有偷;

请推断谁是小偷。

分析:设 a,b,c,d 代表甲、乙、丙、丁 4 人为偷窃嫌疑犯,a=1 表示甲是小偷,a=0 表示甲不是小偷;b=1 表示乙是小偷,b=0 表示乙不是小偷;c=1 表示丙是小偷,c=0 表示丙不是小偷;d=1 表示丁是小偷,d=0 表示丁不是小偷;因此 4 人所说的话可表示为如下表达式:

```
甲:((!b&&d)==1)||((b&&!d)==1)
乙:((!b&&c)==1)||((b&&!c)==1)
丙:((!a&&b)==1)||((a&&!b)==1)
丁:d==0||d==1
```

另外,由于只有一个是真正的小偷,则还要满足:a+b+c+d==1。

```
/*
  程序名称:ex3-44
```

```
  程序功能:确定小偷
*/
#include<stdio.h>
#include<stdlib.h>
int main()
{
    int a,b,c,d;
    for(a=0;a<=1;a++)
        for(b=0;b<=1;b++)
            for(c=0;c<=1;c++)
                for(d=0;d<=1;d++)
if(a+b+c+d==1&&((!b&&d)||(b&&!d))==1&&((!b&&c)||(b&&!c))==1&&((!a&&b)||(a&&!b))==
=1&&(d||!d)==1)
                    {
                        if (a==1)
                            printf("甲是小偷!\n");
                        else if(b==1)
                            printf("乙是小偷!\n");
                        else if(c==1)
                            printf("丙是小偷!\n");
                        else if(d==1)
                            printf("丁是小偷!\n");
                    }
    system("pause");
    return 0;
}
```

本章小结

本章主要讲述了以下内容。

(1) 算法的基本概念、算法的特点、算法的表示方法和常用的算法,算法是程序设计的灵魂,这是学好C语言程序设计的关键所在。C语言是结构化程序设计语言,由3种基本结构组成。

(2) C语言不提供输入输出语句,C语言的输入输出都是由函数来完成的。格式化输入输出函数scanf()和printf(),格式字符和修饰字符的功能和使用。C语言的语句必须用分号结束。

(3) 顺序结构的程序是最简单的程序,它是按照语句的先后顺序执行的。

(4) 分支结构的程序设计是按照表达式的值非0和0来执行不同的语句序列。由if语句、if…else语句、if…else if语句、switch…case语句来实现分支结构程序设计,if…else if语句和switch…case语句设计的程序可相互转换。特别注意,case后面的表达式的值必须是整数或字符。break可以跳出一层switch结构。

(5) 循环结构是C语言程序设计中最重要的知识之一,很多的程序都要用循环来实现,C语言中的循环主要用while、do…while、for来实现,它们之间可以相互嵌套构成更复杂的程序,break和continue语句可以控制循环的走向。

习 题 3

1. 单项选择题

(1) 已知有如下定义和输入语句,若要求 a1、a2、c1、c2 的值分别为 10、20、A 和 B,当从第一列开始输入数据时,正确的数据输入方式是(　　)。

```
int a1,a2; char c1,c2;
scanf("%d%c%d%c",&a1,&c1,&a2,&c2);
```

A. 10□A20B↙　　B. 10□A□20□B↙

C. 10A20B↙　　D. 10A20□B↙

(2) 执行下列程序片段时输出的结果是(　　)。

```
int x = 13,y = 5;
printf("%d",x% = (y/ = 2));
```

A. 3　　B. 2　　C. 1　　D. 0

(3) 若定义 x 为 double 型变量,则能正确输入 x 值的语句是(　　)。

A. scanf("%f",x);　　B. scanf("%f",&x);

C. scanf("%lf",&x);　　D. scanf("%5.1f",&x);

(4) 有输入语句 scanf("a=%d,b=%d,c=%d",&a,&b,&c);为使变量 a 的值为 1,b 的值为 3,c 的值为 2,则正确的数据输入方式是(　　)。

A. 132↙　　B. 1,3,2↙

C. a=1 b=3 c=2↙　　D. a=1,b=3,c=2↙

(5) 逻辑运算符两侧运算对象的数据类型(　　)。

A. 只能是 0 或 1　　B. 只能是 0 或非 0 正数

C. 只能是整型或字符型数据　　D. 可以是任何类型的数据

(6) 能正确表示“当 x 的取值在[1,10]和[200,210]内为真,否则为假”的表达式是(　　)。

A. (x>=1) && (x<=10) && (x>=200) && (x<=210)

B. (x>=1) || (x<=10) || (x>=200) || (x<=210)

C. (x>=1) && (x<=10) || (x>=200) && (x<=210)

D. (x>=1) || (x<=10) && (x>=200) || (x<=210)

(7) C 语言对嵌套 if 语句的规定是:当缺省{ }时,else 总是与(　　)。

A. 其之前最近的 if 配对

B. 第一个 if 配对

C. 缩进位置相同的 if 配对

D. 其之前最近的且尚未配对的 if 配对

(8) 设:int a=1,b=2,c=3,d=4,m=2,n=2;执行(m=a>b) && (n=c>d)后 n 的值为(　　)。

A. 1　　B. 2　　C. 3　　D. 4

(9) 下述程序的输出结果是(　　)。

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
    int a=0,b=0,c=0;
    if (++a>0 || ++b>0)
        ++c;
    printf("%d,%d,%d",a,b,c);
    system("pause");
    return 0;
}
```

A. 0,0,0　　B. 1,1,1　　C. 1,0,1　　D. 0,1,1

(10) 以下程序的输出结果是(　　)。

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
    int x=1,y=0,a=0,b=0;
    switch(x)
    {
      case 1:switch (y)
          {
                case 0 : a++; break;
                case 1 : b++; break;
            }
    case 2:a++; b++; break;
    case 3:a++; b++;
    }
    printf("a=%d,b=%d",a,b);
    system("pause");
    return 0;
}
```

A. a=1,b=0　　B. a=2,b=1　　C. a=1,b=1　　D. a=2,b=2

(11) 有如下程序段：

```
int k=2;
while(k=0) {printf("%d",k);k--;}
```

则下面描述中正确的是(　　)。

A. while 循环执行 2 次　　B. 循环是无限循环

C. 循环体语句一次也不执行　　D. 循环体语句执行一次

(12) 若运行以下程序时，输入 2473↙，则程序的运行结果是(　　)。

```
#include<stdio.h>
#include<stdlib.h>
int main()
```

```
{
    int c;
    while ((c = getchar()) != '\n')
        switch (c - '2')
        {
            case 0 :
            case 1 : putchar (c + 4);
            case 2 : putchar (c + 4); break;
            case 3 : putchar (c + 3);
            default : putchar (c + 2); break;
    }
    printf("\n");
    system("pause");
    return 0;
}
```

A. 668977　　B. 668966　　C. 66778777　　D. 6688766

(13) 以下程序段的循环次数是(　　)。

```
for (i = 2; i == 0; ) printf(" % d",i -- );
```

A. 无限次　　B. 0 次　　C. 1 次　　D. 2 次

(14) 下面程序的输出结果是(　　)。

```
# include < stdio. h >
# include < stdlib. h >
int main()
{
    int x = 9;
    for (; x > 0; x -- )
    {
        if (x % 3 == 0)
        {
            printf(" % d", -- x);
            continue;
        }
    }
    system("pause");
    return 0;
}
```

A. 741　　B. 852　　C. 963　　D. 875421

(15) 下面程序的输出结果是(　　)。

```
# include < stdio. h >
# include < stdlib. h >
int main()
{
    int k = 0,m = 0,i,j;
    for (i = 0; i < 2; i++)
    {
        for (j = 0; j < 3; j++)
```

```
            k++;
        k-=j;
    }
    m = i+j;
    printf("k=%d,m=%d",k,m);
    system("pause");
    return 0;
}
```

A. k=0,m=3　　B. k=0,m=5　　C. k=1,m=3　　D. k=1,m=5

2. 填空题

(1) 一个表达式要构成一个C语句,必须_____。

(2) 复合语句是用一对_____界定的语句块。

(3) 写出数学式 $y=\begin{cases}1(x<0)\\0(x=0)\\-1(x>0)\end{cases}$ 的C语言表达式_____。

(4) 将条件"y能被4整除但不能被100整除,或y能被400整除"写成逻辑表达式_____。

(5) C语言的语法规定:缺省复合语句符号时,else子句总是与_____的if相结合,与书写格式无关。

(6) switch语句中,如果没有与该值相等的标号,并且存在default标号,则从_____开始执行,直到switch语句结束。

(7) C语言的3个循环语句分别是_____语句、_____语句和_____语句。

(8) 至少执行一次循环体的循环语句是_____。

(9) continue语句的作用是结束_____循环。

(10) 用break语句可以使程序流程跳出switch语句体,也可以在循环结构内中止_____循环体。

3. 阅读程序,指出结果

(1) 若运行时输入100↙,以下程序的运行结果是_____。

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
    int a;
    scanf("%d",&a);
    printf("%s",(a%2!=0)?"No":"Yes");
    system("pause");
    return 0;
}
```

(2) 以下程序的运行结果是_____。

```
#include<stdio.h>
#include<stdlib.h>
int main()
```

```
{
    int a = 2,b = 7,c = 5;
    switch (a > 0)
    {
        case 1: switch (b < 0)
            {
                case 0: printf("@"); break;
                case 1: printf("!"); break;
            }
        case 0: switch (c == 5)
                    {
                    case 0: printf("*"); break;
                    case 1: printf("#"); break;
                    default : printf("#"); break;
                }
        default : printf("&");
    }
    printf("\n");
    system("pause");
    return 0;
}
```

(3) 阅读下面程序,输入字母 A 后,其运行结果是_____。

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    char ch;
    ch = getchar();
    switch(ch)
    {
        case 65:printf("%c",'A');
        case 66:printf("%c",'B');
        default:printf("%s\n","other");
    }
    system("pause");
    return 0;
}
```

(4) 下面程序运行的结果是_____。

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int k = 1,n = 263;
    do { k *= n%10; n/=10; } while (n);
    printf("%d\n",k);
    system("pause");
    return 0;
}
```

(5) 下面程序运行的结果是 _____。

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
    int x,i;
    for (i=1; i<=100; i++)
    {
        x=i;
        if (++x%2==0)
            if (++x%3==0)
                if(++x%7==0)
                    printf("%d ",x);
    }
    system("pause");
    return 0;
}
```

(6) 下面程序运行的结果是_____。

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
    int i,b,k=0;
    for (i=1; i<=5; i++)
    {
        b=i%2;
        while (b--==0) k++;
    }
    printf("%d,%d",k,b);
    system("pause");
    return 0;
}
```

(7) 下面程序运行的结果是_____。

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
    int a,b;
    for (a=1,b=1; a<=100; a++)
    {
        if (b>=20) break;
        if (b%3==1) { b+=3; continue; }
        b-=5;
    }
    printf("%d\n",a);
    system("pause");
    return 0;
}
```

(8) 下面程序运行的结果是_____。

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
    int i=5;
    do
    {
        switch(i%2)
            {
              case 0 : i--; break;
              case 1 : i--; continue;
        }
        i--; i--;
        printf("%d",i);
    }while(i>0);
    system("pause");
    return 0;
}
```

(9) 下面程序运行的结果是_____。

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
    int i,j;
    for(i=0;i<3;i++,i++)
    {
      for(j=4; j>=0; j--)
      {
        if((j+i)%2)
        {
            j--;
            printf("%d,",j);
            continue;
        }
        --i;
        j--;
        printf("%d,",j);
        }
    }
    system("pause");
    return 0;
}
```

(10) 下面程序运行的结果是_____。

```
#include<stdio.h>
#include<stdlib.h>
int main()
```

```
{
    int a = 10,y = 0;
    do
    {
        a += 2;
        y += a;
        if (y > 50) break;
    } while (a == 14);
    printf("a = %d y = %d\n",a,y);
    system("pause");
    return 0;
}
```

(11) 下面程序运行的结果是_____。

```
#include < stdio.h >
#include < stdlib.h >
int main()
{
    int i,j,k = 19;
    while (i = k - 1)
    {
        k -= 3;
        if (k % 5 == 0)
        {
            i++;
            continue;
    }
    else if (k < 5) break;
    i++;
    }
    printf("i = %d,k = %d\n",i,k);
    system("pause");
    return 0;
}
```

(12) 下面程序运行的结果是_____。

```
#include < stdio.h >
#include < stdlib.h >
int main()
{ int y = 2,a = 1;
    while (y-- != -1)
        do {
            a * = y;
            a++;
        } while (y-- );
    printf("%d,%d\n",a,y);
    system("pause");
    return 0;
}
```

4. 程序填空题

(1) 以下程序输出 x、y、z 3 个数中的最小值,请填空使程序完整。

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
    int x=4,y=5,z=8;
    int u,v;
    u = x<y ?_____________;
    v = u<z ?_____________;
    printf("%d",v);
    system("pause");
    return 0;
}
```

(2) 下面程序接受键盘上的输入,直到按↙键为止。这些字符被原样输出,但若有连续的一个以上的空格时只输出一个空格。请填空使程序完整。

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
    char cx,front='\0';
    while (_____________!='\n')
    {
        if (cx!=' ') putchar(cx);
        if (cx==' ')
            if (_____________)
                putchar(_____________)
        front=cx;
    }
    system("pause");
    return 0;
}
```

(3) 以下程序的功能是:从键盘上输入若干学生的成绩,统计并输出最高成绩和最低成绩,当输入负数时结束输入。请填空使程序完整。

```
#include<stdio.h>
#include<stdlib.h>
int main (    )
{
    double s;
    double gmax,gmin;
    scanf("%f",&s);
    gmax=s;
    gmin=s;
    while(_____________)
    {
        if(s>gmax)
```

```
            gmax = s;
        if(_____)
            gmin = s;
        scanf("%lf",&s);
    }
    printf("\ngmax = %f\ngmin = %f\n",gmax,gmin);
    system("pause");
    return 0;
}
```

(4) 下面程序的功能是：输出1～100每位数的乘积大于每位数的和的数。请填空使程序完整。

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
    int n,k = 1,s = 0,m;
    for (n = 1; n<=100; n++)
    {
        k = 1; s = 0;
        _____;
        while (_____)
        {
            k* = m%10;
            s+= m%10;
            _____;
        }
        if (k>s) printf("%d",n);
    }
    system("pause");
    return 0;
}
```

(5) 已知如下公式：

$$\frac{p}{2}=1+\frac{1}{3}+\frac{1}{3}\cdot\frac{2}{5}+\frac{1}{3}\cdot\frac{2}{5}\cdot\frac{3}{7}+\frac{1}{3}\cdot\frac{2}{5}\cdot\frac{3}{7}\cdot\frac{4}{9}+\cdots$$

下面程序的功能是：根据上述公式输出满足精度要求的eps的π值。请填空使程序完整。

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
    double s = 1.0,eps,t = 1.0;
    int n;
    scanf ("%lf",&eps);
    for (n = 1;_____; n++)
    {
        t = _____;
        s+= t;
```

```
    }
    printf("%lf\n",_____________);
    system("pause");
    return 0;
}
```

(6) 下面程序段的功能是：计算 1000！的末尾有多少个零。请填空使程序完整。

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
    int i,k,m;
    for (k=0,i=5; i<=1000; i+=5)
    {
        m = i;
        while (_____________)
        {
            k++;
            m=m/5;
        }
    }
    printf("%d\n",k);
    system("pause");
    return 0;
}
```

(7) 下面程序按公式 $\sum_{k=1}^{100}k+\sum_{k=1}^{50}k^2+\sum_{k=1}^{10}\frac{1}{k}$ 求和并输出结果。请填空使程序完整。

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
    _____________;
    int k;
    for (k=1; k<=100; k++)
        s+=k;
    for (k=1; k<=50; k++)
        s+=k*k;
    for (k=1; k<=10; k++)
        s+=_____________;
    printf("sum=%lf\n",s);
    system("pause");
    return 0;
}
```

5. 编程题

(1) 有一函数：

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

编写一程序，输入 x，输出 y 值。

(2) 从键盘上输入 3 个整数，求最小的数。

(3) 输入年月日，判断是这年的第几天。

(4) 企业发放的奖金根据利润提成：利润低于或等于 10 万元时，奖金可提 10%；利润高于 10 万元低于 20 万元时，低于 10 万元的部分按 10%提成，高于 10 万元的部分可提成 7.5%；利润在 20～40 万之间时，高于 20 万元的部分可提成 5%；利润在 40～60 万之间时，高于 40 万元的部分可提成 3%；利润在 60～100 万之间时，高于 60 万元的部分可提成 1.5%；利润高于 100 万元时，超过 100 万元的部分按 1%提成。从键盘输入当月利润，求应发放奖金总数。

(5) 输入字符，并以 Enter 键结束。将其中的小写字母转换成大写字母，而其他字符不变。

(6) 输入一个正整数，求它的所有素数因子。

(7) 从键盘输入正整数 a，求 s=a+aa+aaa+…+a…a。

(8) 九头鸟(传说中的一种怪鸟，它有九个头，两只脚)、鸡和兔子关在一个笼子里，它们的头的数量是 100，脚的数量也是 100。编程计算其中九头鸟、鸡和兔子各有多少只？

(9) 用二分法求方程 $2x^3-4x^2+3x-6=0$ 在区间(−10,10)之间的根。

(10) 编写一个程序，计算 $x-\frac{1}{2}\cdot\frac{x^3}{4}+\frac{1}{2}\cdot\frac{3}{4}\cdot\frac{x^5}{6}-\frac{1}{2}\cdot\frac{3}{4}\cdot\frac{5}{6}\cdot\frac{x^7}{8}+\cdots$ 的近似值(直到最后一项的绝对值小于 eps)。

(11) 取出一个无符号的十进制整数中所有奇数数字，按原来的顺序组成一个新的数。