第3章

流程控制语句

在程序设计语言中，流程控制语句用于确定程序的执行顺序。C++ 语言提供了三种基本控制结构。顺序结构是指按先后顺序依次执行程序中的语句；分支结构是指根据给定的条件有选择地执行程序中的语句；循环结构是指按给定规则重复地执行程序中的语句。本章对 C++ 语言的程序流程控制语句进行详细讲解。

3.1 程序流程描述的方法

3.1.1 程序流程的描述方式

在 C++ 语言中有五种常用的程序流程描述方式：自然语言、传统流程图、N-S 流程图、伪代码和程序设计语言。

1. 自然语言

在 1.1 节我们讲过什么是计算机程序，计算机程序是为了完成某项任务、解决某个问题，计算机要执行的一系列指令（步骤）。如果操作步骤都是顺序执行的，那么用自然语言描述通俗易懂，比较直观且容易理解。比如案例 3 中计算泳池栏杆造价用自然语言描述为：先输入泳池的半径，计算泳池的周长，然后用栏杆单价乘以泳池的周长，计算出泳池周围栏杆的造价，最后输出计算结果。显然，自然语言可以把操作步骤说明得很清晰，但是，如果程序中包含了分支结构和循环结构，并且执行步骤较多时，使用自然语言描述就会显得不那么清晰明了。

2. 传统流程图

程序传统流程图是用一系列特定图形符号（如表 3.1 所示）、流向线和文字说明描述程序的执行步骤，控制语句的执行顺序，是程序分析和过程描述的基本方式。这种方法能够克服复杂程序用自然语言描述不直观的缺点。

表 3.1　传统流程图常用的特定图形符号

序号	图形符号	名　称	含　义
1		起止框	程序的开始或结束
2		处理框	数据的各种处理和运算操作过程
3		输入/输出框	数据的输入和结果的输出
4		注释框	添加程序注释
5		流向线	程序的执行路径
6		判断框	根据条件的不同，选择不同的操作
7		连接点	转向流程图的其他位置或从其他位置转入

三种基本程序流程结构为：顺序结构、分支结构（单分支、二分支、多分支）和循环结构。循环结构是在一定条件下反复执行某段程序的流程结构，被反复执行的程序段称为循环体。三种基本结构可以用传统流程图明晰地表达，如图 3.1～图 3.6 所示。

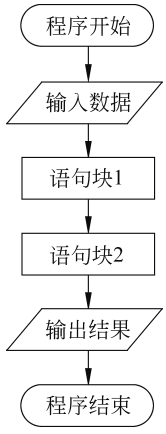

图 3.1　顺序结构

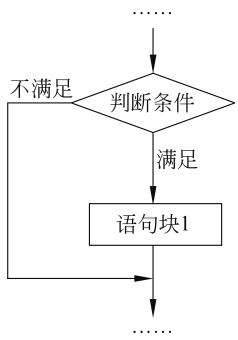

图 3.2　单分支结构

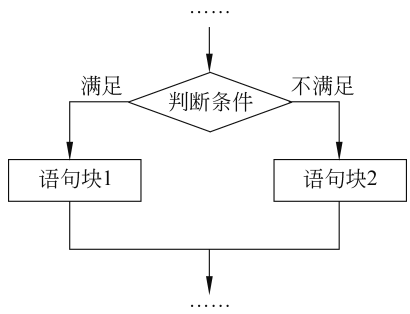

图 3.3　二分支结构

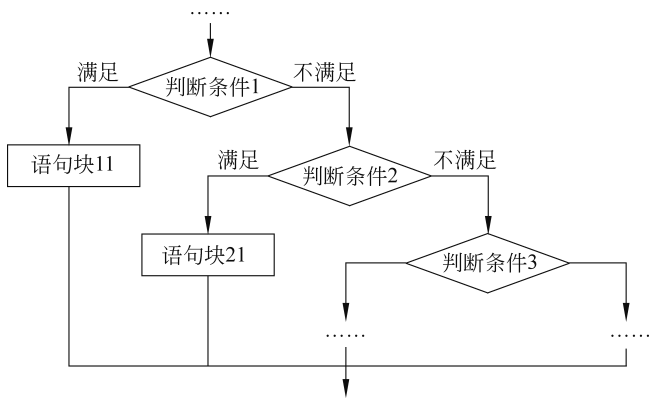

图 3.4　多分支结构

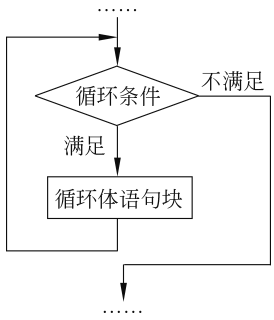

图 3.5　当型循环结构

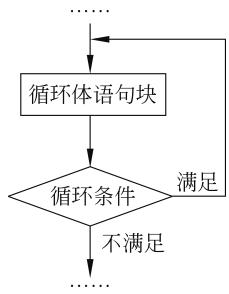

图 3.6　直到型循环结构

当型循环结构是指若条件满足时执行循环体语句；若条件不满足时，退出循环，执行循环体后面的语句。直到型循环结构是指先执行循环体语句块，如果条件满足，就继续执行循环体语句块，直到条件不满足的时候退出循环。

3. N-S 流程图

N-S 流程图又叫盒图，是美国学者 I. Nassi 和 B. Shneiderman 提出的一种在流程图中完全去掉流向线的图，将全部程序写在一个矩形框内，从而避免在描述大型复杂算法时，图中的流向线较多，影响了用户对程序的阅读和理解。N-S 流程图使用矩形框来表达各种处理步骤。顺序结构、分支结构及循环结构的 N-S 流程图如图 3.7～图 3.12 所示。

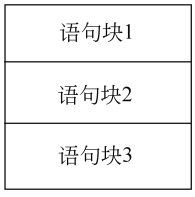

图 3.7　顺序结构

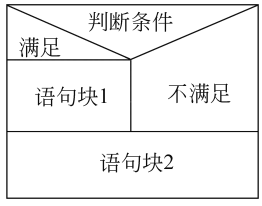

图 3.8　单分支结构

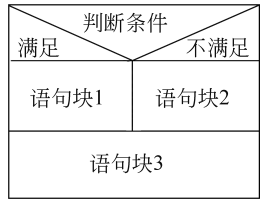

图 3.9　二分支结构

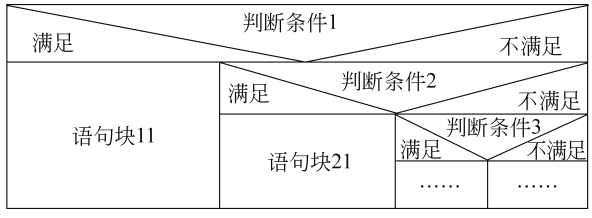

图 3.10　多分支结构

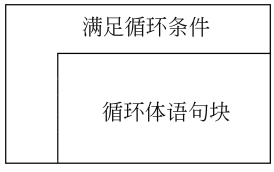

图 3.11　当型循环结构

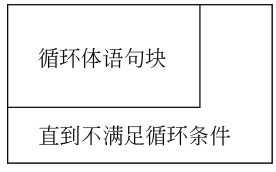

图 3.12　直到型循环结构

4. 伪代码

伪代码是在更简洁的自然语言描述中，用程序设计语言的流程控制结构来表示处理步骤的执行流程，用自然语言和各种符号来表示所进行的各种处理及所涉及的数据。它是介于程序代码和自然语言之间的一种描述方法。这样描述的程序书写紧凑、自由，在表达分支结构和循环结构时易于理解，同时也更有利于流程向程序的转换。顺序结构、分支结构及循环结构的伪代码描述形式如图 3.13～图 3.18 所示。

```
语句块 1;
语句块 2;
语句块 3;
……
```

图 3.13　顺序结构

```
if(条件)
    语句块 1;
……
```

图 3.14　单分支结构

```
if(条件)
    语句块 1;
else
    语句块 2;
……
```

图 3.15　二分支结构

```
if(条件 1)
    语句 11;
else if(条件 2)
    语句 21;
else if(条件 3)
    语句块 31;
else if ……
else  语句 n;
```

图 3.16　多分支结构

```
while(条件)
{
    语句块 1;
    语句块 2;
    ……
}
```

图 3.17　当型循环结构

```
do
{
    语句块 1;
    语句块 2;
    ……
}while(条件);
```

图 3.18　直到型循环结构

5. 程序设计语言

程序最终是需要在计算机上运行的，因此程序设计语言是程序的最终描述形式。无论用何种方法描述，都是为了将其更方便地转化为计算机程序。

3.1.2　判断素数的程序流程描述

1. 素数的定义

素数也称为质数，指的是在大于数字 1 的自然数中，除了可以被数字 1 以及该数字本身整除之外，再也不能够被其他任何自然数整除的数。如果还可以被其他自然数整除，则称该自然数为合数。

2. 自然语言描述

【例 3.1】 判断一个自然数 n 是否是素数(n＞＝2)，有如下步骤。

步骤 1：输入 n 的值。

步骤 2：设置变量 i 的值为 2。

步骤 3：求解 r＝n%i；若 r＝0 转向步骤 6，否则执行步骤 4。

步骤 4：变量 i 的值自加 1。

步骤 5：判断 i＜n 是否成立；若为真则转向步骤 3，否则执行步骤 6。

步骤 6：判断 i＜n 是否成立；若为真则输出“合数”，否则输出“素数”。

3. 用程序传统流程图描述

判断一个数 n 是否是素数(n＞＝2)，如图 3.19 所示。

4. 用 N-S 流程图描述

判断一个数 n 是否是素数(n＞＝2)，如图 3.20 所示。

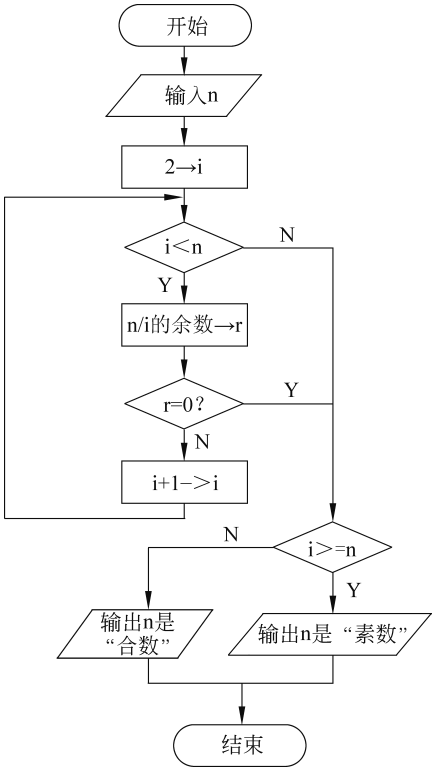

图 3.19　判断素数的传统流程图

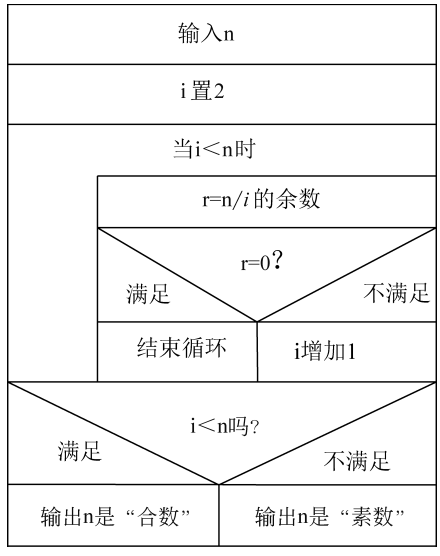

图 3.20　判断素数 N-S 流程图

5. 用伪代码描述

判断自然数 n 是否为素数的伪代码描述如下：

```
input n;
i 置为 2;
while(i<n)
{
    r=n/i 的余数;
    if(r=0)
```

```
    break;
    else
        i= i+1;
}
if(i<n)
    output  "n是合数";
else
    output  "n是素数";
```

6. C++ 语言写出程序

判断自然数 n 是否为素数的 C++ 语言代码描述如下：

```
#include<iostream>
using namespace std;
int main()
{
    int n,i,r;
    cout<<"输入大于或等于2的自然数n:";
    cin>>n;                              //输入 n
    i=2;
    while(i<n)
    {
        r=n%i;
        if(r==0)
            break;
        else
            i++;
    }
    if(i<n)
        cout<<n<<"是合数"<<endl;      //输出 n 是合数
    else
        cout<<n<<"是素数"<<endl;      //输出 n 是素数
}
```

【思考与练习】

1. 简答题

(1) 写出传统流程图中常用的特定图形符号，并说明其意义。

(2) 分别用自然语言、传统流程图、N-S 流程图、伪代码描述下面程序的执行过程：

输出 1000 以内的完全平方数，并要求每行输出 6 个数。完全平方数指的是一个整数可以等于另一个整数的平方。例如 1＝1×1，4＝2×2，9＝3×3 等，以此类推。若一个数能表示成某个整数的平方的形式，则称这个数为完全平方数，那么 1、4、9、16、……就是完全平方数。

2. 写程序功能

(1) 图 3.21 为一个程序的传统流程图，写出它所表示的函数。

(2) 图 3.22 为一个程序的传统流程图，写出程序的功能。

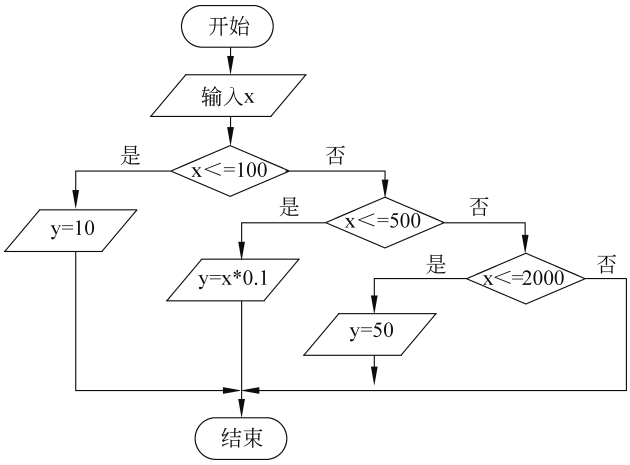

图 3.21　程序的传统流程图(一)

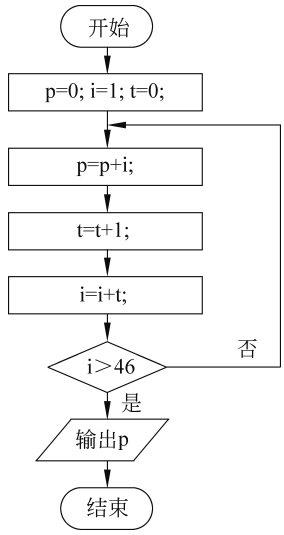

图 3.22　程序的传统流程图(二)

3.2　案例 7——阶梯水价

3.2.1　问题描述及程序代码

1. 问题描述

“阶梯水价”是对自来水用户实行分类计量收费和超定额后累进加价制的俗称。“阶梯水价”充分发挥了市场、价格因素在水资源配置、水需求调节等方面的作用。2020 年唐山居民生活年用水量价格标准如下。

第一阶梯：0～120(含)m^3，水价为 5 元/m^3(其中，水费为 3.5 元/m^3，污水处理费为 1.5 元/m^3)。

第二阶梯：120～180(含)m^3，水价为 6.75 元/m^3(其中，水费为 5.25 元/ m^3，污水处理费为 1.5 元/m^3)。

第三阶梯：180m^3 以上，水价为 12 元/m^3(其中，水费为 10.5 元/m^3，污水处理费为 1.5 元/ m^3)。题目要求编写程序，输入某居民用户 2020 年的用水量，输出该用户本年度应缴费金额。

测试用例 1：

```
输入
年用水量(立方米)：35
输出
您应缴水费：175 元
```

测试用例 2：

```
输入
年用水量(立方米)：135
输出
您应缴水费：701.25 元
```

测试用例 3：

```
输入
年用水量(立方米)：190
输出
您应缴水费：1125 元
```

2. 程序代码

```
#include<iostream>
using namespace std;
int main()
{
    const float price1=5.0;                                  //第一阶梯单价
    const float price2=6.75;                                 //第二阶梯单价
    const float price3=12;                                   //第三阶梯单价
    float bound1=120*price1;                                 //第二阶梯用水量的基数
    float bound2=120*price1+(180-120) *price2;               //第三阶梯用水量的基数
    float cons;                                              //年度用水量
    float total;                                             //缴费金额
    cout<<"请输入本年度用水量：";
    cin>>cons;
    if(cons<=120)                                            //如果小于或等于 120 立方米
        total=price1*cons;                                   //第一阶梯单价×用水量
    //120 立方米以内的单价属于第一阶梯单价,超出部分为第二阶梯单价
    else if(cons<=180)                                       //用水量在第二阶段
        total=bound1+(cons-120)*price2;
    //120 立方米以内的单价属于第一阶梯单价,120~180 立方米的单价属于第二阶梯单价
    //超出 180 立方米的属于第三阶梯单价
    else total=bound2+(cons-180)*price3;
    cout<<"您本年度需要缴费："<<total<<"元"<<endl;   //输出计算结果
}
```

3. 程序分析及运行结果

本程序根据输入用水量的多少计算缴费金额,因为用水量不同,水的单价也不一样,所以程序采用分支结构,使用条件语句完成。在 C++ 语言中分支结构的语句有两种,一种是条件语句,即 if 语句;另一种是开关语句,即 switch 语句。它们都可以用来实现多分支结构。分支结构具有一定的语句控制能力,可以根据输入的数据和给定的条件来决定执行哪些语句,不执行哪些语句。就像本案例一样,给定了三个条件,输入的数据满足不同的条件,就用到不同的计算公式。

3.2.2 条件语句

1. 条件语句的语法格式

条件语句的一般格式如下：

```
if(条件 1)
    语句块 1;
```

```
else if(条件 2)
    语句块 2;
else if(条件 3)
    语句块 3;
    ……
else if(条件 n)
    语句块 n;
else
    语句块 n+1;
```

if、else if 和 else 是关键字，条件 1、条件 2、条件 3、……、条件 n 是表达式，既可以是关系表达式、逻辑表达式，也可以是其他表达式。语句块 1、语句块 2、语句块 3、……、语句块 n、语句块 n＋1，每个语句块既可以是一条语句，也可以是用花括号“{}”括起来的多条语句。用花括号“{}”括起来的多条语句也叫复合语句，复合语句在语法上相当于一条语句，执行的时候会把复合语句每一条语句都执行完成。复合语句内可以有声明语句，有声明语句的复合语句叫分程序或语句块。语句块通常可以作为 if 体、else 体或循环体语句。

在条件语句中，至少有一个 if 条件，else if 可以有零个、一个或多个，else 可以有零个或一个。如果 else if 和 else 都省略，那么这种条件语句就是单分支语句。

2. 单分支

单分支的语句形式：

```
if(条件表达式) 语句块 1;
```

其传统流程图如图 3.2 所示。

【例 3.2】 编写程序完成输入一个数据 n，求 n 的绝对值并保存在 n 变量中，在显示器上输出数据 n 值。提示如果 n 是正数，则输出 n 本身；如果 n 是负数，那么对 n 取负数，然后还是输出 n。代码如下：

```
#include<iostream>
using namespace std;
int main()
{
    double n;
    cout<<"输入数据 n:";
    cin>>n;
    if(n<0)
    n=-n;
    cout<<n<<endl;
    return 0;
}
```

【例 3.3】 编写程序完成输入一个正整数 n，输出 n 的所有因数（当整数 a 除以整数 b(b≠0)的商为 c，c 正好是整数，余数为 0 时，就说 b 是 a 的因数）。比如数据 n 的值是 30，那么 1、2、3、5、6、10、15 和 30 都是 30 的因数，本案例用单分支结构完成。直观地理解，若是因数的时候输出因数，否则就不需要处理。程序包含循环结构和单分支结构，流程图描述如图 3.23 所示。

程序代码如下：

```
#include<iostream>
using namespace std;
int main()
{
    int n;
    cout<<"请输入一个整数:";
    cin>>n;
    for(int i=1;i<=n;i++)
        if(n%i==0)
            cout<<i<<"\t";
    cout<<endl;
    return 0;
}
```

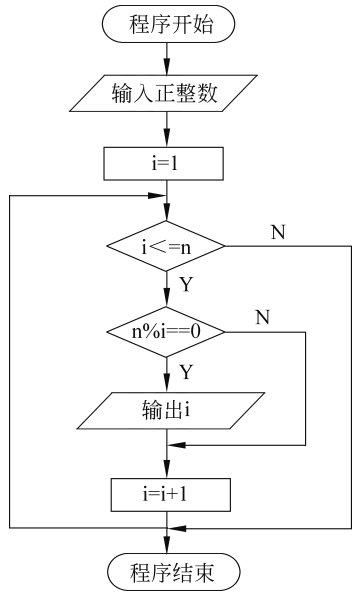

图 3.23　输出正整数 n 的因数流程图

3. 二分支

二分支的语句形式：

```
if(条件表达式)
    语句块 1;
else
    语句块 2;
```

其传统流程图如图 3.3 所示。

【例 3.4】 当学生用户登录学生基本信息管理系统时，需要验证用户名和登录密码。如果用户名和密码都正确，进入系统，显示“登录成功!”提示信息。若两者有一个不正确，则不能进入系统，且显示器上提示“用户名或密码错误!”。

说明：本案例属于二分支结构语句。假设用户名是“Feng”，密码是“PPNN13mod”（网上看到的一个很文艺的密码 PPNN13moddkstFeb1，这一大串字符看似很晕，实际是杜牧的一句诗，“娉娉裊裊十三余，豆蔻梢头二月初。”这个程序员是不是很有诗情画意？暂且借用这个密码）。

本程序定义字符串使用字符数组，对于字符串的比较使用 strcmp 函数，判断用户名正确且密码正确，使用逻辑与运算符。程序代码如下：

```
#include<iostream>
#include <string.h>
using namespace std;
int main()
{
    char userName[30];                    //用户名
    char pass[30];                        //密码
    cout<<"请输入用户名: ";
    cin>>userName;                        //输入用户名
    cout<<"请输入密码: ";
    cin>>pass;                            //输入密码
    //下面判断如果用户名是"Feng"并且密码是"PPNN13mod"
    if(!strcmp(userName,"Feng")&&!strcmp(pass,"PPNN13mod"))
          cout<<"登录成功!";
    else                                  //任何一个不正确
```

```
        cout<<"用户名或密码错误!";
    return 0;
}
```

因为根据上面程序的条件不能确切知道到底是用户名错误还是密码错误，现在对该代码进行修改。在二分支语句中嵌套了二分支语句，首先判断用户名是否正确，如果用户名正确，再看密码是否正确，只有二者都正确的情况下，才会提示“登录成功!”。

```
if(!strcmp(userName,"Feng"))          //用户名正确
    if(!strcmp(pass,"PPNN13mod"))     //密码正确
        cout<<"登录成功!";
    else                              //密码不正确
        cout<<"用户密码错误!";
else                                  //用户名不正确
    cout<<"用户名错误!";
```

分支嵌套说明：在分支嵌套中需要注意 else 子句如何与 if 子句的匹配问题。

【例 3.5】 if 子句和 else 子句的匹配问题。

```
if(x>0)
    if(x < 50)
          cout<<"OK!\n";
else
    cout<<"NOT OK!\n";
```

阅读本段代码，else 子句前面有两个 if 子句，如果把 else 子句看成和第一个 if 子句匹配，那么就是一个二分支结构中嵌套了一个单分支结构；如果把 else 子句看成和第二个 if 子句匹配，那么就是一个单分支结构中嵌套了一个二分支结构。这样对于给定的 x 值，输出结果就不一样了。比如 x 的值是－100，如果把 else 子句理解为和第一个 if 子句匹配，那么输出“NOT OK!”；如果把 else 子句理解为和第二个 if 子句匹配，那么什么也不输出。同理，如果 x 的值是 80，如果把 else 子句理解为和第一个 if 子句匹配，那么什么也不会输出；如果 else 子句和第二个 if 子句匹配，那么就会输出“NOT OK!”。程序在相同的输入情况下，不可能输出不同的结果。那么在出现这样语句的情况下，else 子句到底与哪个 if 子句匹配呢？记住 else 子句从属于最近的 if 子句，本案例中 else 子句和第二个 if 子句匹配。如果想让 else 子句与第一个 if 子句匹配，应用加花括号{}来实现，修改如下：

```
if(x>0)
{
    if(x < 50)
        cout<<" x is OK\n";
}
else
    cout<<" x is not OK\n";
```

4. 多分支

多分支结构的传统流程图如图 3.4 所示。多分支语句里至少有一个 else if 关键字，是在同一个表达式可能有多个选择的情况下，根据条件选择执行不同的语句块。案例 7 就是根据用水量的

多少，有三种计算缴费的标准，所以这是三分支语句，有时候可能有更多的分支。

【例 3.6】 输入一个百分制的学生成绩 score，请把百分制成绩转换成五分制的字符，转换规则如下：

如果输入的成绩不是 0～100 分，则输出“成绩无效”；否则，若成绩在 60 分以下输出“E”；若成绩大于或等于 60 分且小于 70 分输出“D”；若成绩大于或等于 70 分且小于 80 分输出“C”；若成绩大于或等于 80 分且小于 90 分输出“B”；若成绩大于或等于 90 分且小于或等于 100 分输出“A”。程序传统流程图如图 3.24 所示。

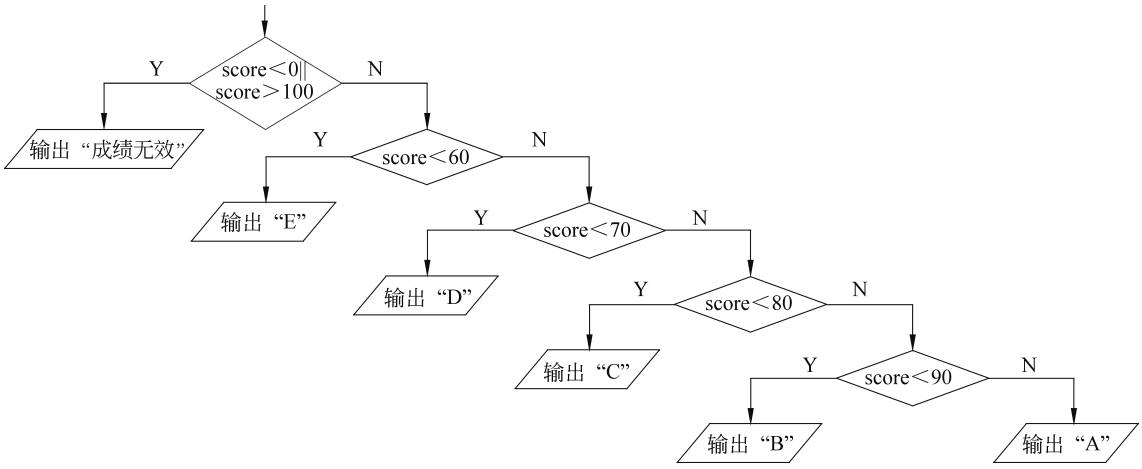

图 3.24 百分制转换五分制流程图

根据画好的传统流程图，再写程序代码就会感觉简单多了，代码如下：

```
#include<iostream>
using namespace std;
int main()
{
    int score;                              //声明成绩变量
    cout<<"请输入学生成绩[0~100]";          //提示信息
    cin>>score;                             //输入语句
    if(score<0||score>100)
        cout<<"成绩无效！\n";
    else if(score<60)
        cout<<"E\n";
    else if(score<70)
        cout<<"D\n";
    else if(score<80)
        cout<<"C\n";
    else if(score<90)
        cout<<"B\n";
    else
        cout<<"A\n";
    return 0;
}
```

多分支语句中含有多个 else if 子句，从代码上看程序的可读性较差，通常多分支语句可以转

换成开关语句，即用 switch 语句完成。

3.2.3　上机练习

【上机目的】

- 熟悉分支结构中条件语句的用法。
- 掌握条件语句的三种分支语句的格式。
- 掌握条件语句的嵌套和多分支语句之间的区别。

1. 上机调试阶梯水价案例，要求先把收费标准中每个阶段水价及水量标准从键盘输入，再输入用户的用水量，计算用户的缴费金额。这样程序就可以适用于不同年份、不同地区有阶段水价或水量标准调整的情况。

2. 用户登录的案例中，第一种方法可以“提示用户名或密码不正确”，第二种方法用 if 嵌套语句实现时，可以提示“用户名错误”或“密码错误”。请修改程序，完成当用户名和密码都错误时给出“用户名且密码错误”的提示信息。

3. 一元二次方程 $ax^2+bx+c=0$，a 是二次项系数，b 是一次项系数，c 是常数项。编写程序要求从键盘输入 a、b、c 三个数，如果 a 为 0，输出“不是一元二次方程!”，否则求出一元二次方程的解。若判别式大于 0，有两个不等实数根，输出两个不等实数根的值；若判别式等于 0，有两个相等的实数根，输出相等实数根的值；若判别式小于 0，有两个共轭复数根，输出两个共轭复数根。

4. 编写程序完成从键盘上输入一个字符，输出这个字符是字母、数字、空格还是其他字符。

【思考与练习】

1. 简答题

(1) else 子句如何与 if 子句相匹配？

(2) 分别用单分支结构、二分支结构和多分支结构写一个程序。

2. 单选题

(1) 有语句

```
int a=1,b=2,c=3;
if(a>c)
    b=a; a=c; c=b;
```

则 c 的值为(　　)。

A. 1　　B. 2　　C. 3　　D. 不一定

(2) 条件语句的格式：

```
if(条件表达式) 语句块 1;
else 语句块 2;
```

其中“条件表达式”为(　　)。

A. 是关系表达式　　B. 必须是逻辑表达式

C. 必须是关系表达式或逻辑表达式　　D. 可以是任何合法的表达式

3. 程序分析

(1) 写出运行结果。

```
int a=4,b=3,c=5,t=0;
if(a<b)t=a;a=b;b=t;
if(a<c)t=a;a=c;c=t;
cout<<a<<b<<c;
```

(2) 写出运行结果。

```
int a=4,b=3,c=5,t=0;
if(a<b){t=a;a=b;b=t;}
if(a<c){t=a;a=c;c=t;}
cout<<a<<b<<c;
```

3.3 案例 8——车牌查询

3.3.1 问题描述及程序代码

1. 问题描述

车牌是标识车辆身份的号牌,车的车牌号就像人的身份证号一样,其主要作用是通过车牌号可以知道该车辆的所属地区。车牌号的第一个是汉字表示省或直辖市,比如河北省是冀,河南省是豫,北京市是京等。汉字后面第一个字母表示本省的省辖市,河北省车牌号的第一个字母代表的城市如下:A—石家庄市,B—唐山市,C—秦皇岛市,D—邯郸市,E—邢台市,F—保定市,G—张家口市,H—承德市,J—沧州市,R—廊坊市,T—衡水市。现在输入河北省的车牌号,输出该车辆所属的城市。

测试用例 1:

```
输入车牌号: A001
输出: 石家庄市
```

测试用例 2:

```
输入车牌号: B6666
输出: 唐山市
```

测试用例 3:

```
输入车牌号: M9107
输出: 不是河北省的车辆
```

2. 程序代码

```
#include <iostream>
using namespace std;
int main()
{
    char vlpn[10];                          //车牌号码
    cout<<"请输入车牌号码:";                 //提示信息
```

```
    cin>>vlpn;
    char letter;                                    //第一个字母
    letter=vlpn[0];
    switch(letter)
    {
        case  'A': cout<<"石家庄市\n"; break;
        case  'B': cout<<"唐山市\n"; break;
        case  'C': cout<<"秦皇岛市\n"; break;
        case  'D': cout<<"邯郸市\n"; break;
        case  'E': cout<<"邢台市\n"; break;
        case  'F': cout<<"保定市\n"; break;
        case  'G': cout<<"张家口市\n"; break;
        case  'H': cout<<"承德市\n"; break;
        case  'J': cout<<"沧州市\n"; break;
        case  'R': cout<<"廊坊市\n"; break;
        case  'T': cout<<"衡水市\n"; break;
        default:  cout<<"不是河北省的车辆！\n";
    }
    return 0;
}
```

3. 程序分析及运行结果

本程序案例同样是属于多分支，是根据车牌号码首字母的不同，执行不同的输出语句。如果使用条件语句，也完全能够实现此功能，但是会有很多项 else if 子句，程序的可读性较差，这里使用了开关语句，即 switch 语句，程序代码就很清晰，一目了然。下面介绍开关语句的语法格式及其注意事项。

3.3.2　开关语句

1. 开关语句的语法格式

开关语句具有如下一般格式：

```
switch(整数表达式)
{
    case 整常量表达式 1: 语句序列 1;  break;
    case 整常量表达式 2: 语句序列 2;  break;
    case 整常量表达式 3: 语句序列 3;  break;
    ……
    case 整常量表达式 n: 语句序列 n;  break;
    default: 语句序列 n+1;
}
```

等价于条件语句：

```
if(整数表达式==整常量表达式 1)
    语句序列 1;
else if(整数表达式==整常量表达式 2)
    语句序列 2;
```

```
else if(整数表达式==整常量表达式 3)
    语句序列 3;
    ……
else if(整数表达式==整常量表达式 n)
   语句序列 n;
else
    语句序列 n+1;
```

开关语句的流程图如图 3.25 所示。

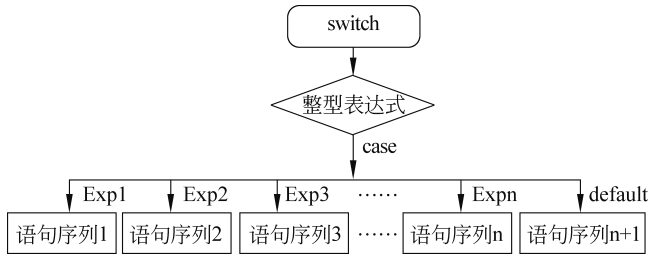

图 3.25　switch 语句流程图

根据本案例的问题描述和 switch 语句的流程图，画出本案例流程图如图 3.26 所示。

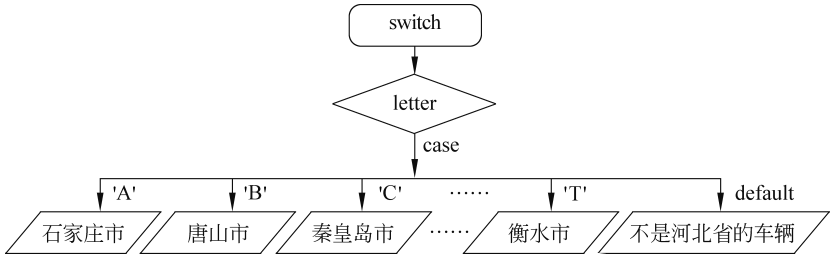

图 3.26　车牌查询程序流程图

2. 开关语句的几点说明

- switch 后面的表达式是整型表达式，一定要含有变量，而 case 后面的表达式必须是整型常量表达式，不允许出现变量，字符常量也是整型常量，因为字符类型是整型的子集。
- 开关语句可以嵌套使用。
- break 是转向语句，在 case 子句中既可以使用 break 语句跳出 switch 语句，也可以没有 break 语句。如果有 break 就结束 switch 语句；如果没有 break 语句就接着执行下一个 case 子句。
- default 子句既可以有，也可以没有，并且可以放在 switch 语句的任何位置。

3. 开关语句的应用案例

【例 3.7】 使用开关语句编程实现输入年、月、日，输出这一天是本年度的第多少天。

```
#include <iostream>
using namespace std;
int main()
{
```

```
    int year,month,day;                                    //年、月、日
    int sumdays;                                           //本月前过了多少天
    int days;                                              //本年度的第多少天
    cout<<"输入年月日: ";                                  //提示信息
    cin>>year>>month>>day;                                 //输入年月日
    if(year<0||month<1||month>12||day<0||day>31)           //判断日期的有效性
     {
         cout<<"无效日期! \n";
        return 0;
     }
    else
        switch(month)
        {
            case 1:sumdays=0;  break;                      //输入的月份是 1 月份
            case 2:sumdays=31;  break;                     //2 月份
            case 3:sumdays=59;  break;                     //3 月份,先按平年计算
            case 4:sumdays=90;  break;                     //4 月份
            case 5:sumdays=120;  break;                    //5 月份
            case 6:sumdays=151;  break;                    //6 月份
            case 7:sumdays=181;  break;                    //7 月份
            case 8:sumdays=212;  break;                    //8 月份
            case 9:sumdays=243;  break;                    //9 月份
            case 10:sumdays=273;  break;                   //10 月份
            case 11:sumdays=304;  break;                   //11 月份
            case 12:sumdays=334;                           //12 月份最后一个语句不需要 break;
        }
    //本月前的天数+日期的日
    days=sumdays+day;
    //下面判断是否闰年,月份是否在 3 月以后,
    //如果是闰年并且月份在 3 月以后,总天数加 1
    //闰年条件: 400 的整数倍或不是整百的数时是 4 的整数倍
    if(year%400==0||(year%100!=0&&year%4==0))              //如果是闰年
        if(month>2)                                        //如果月份是 2 月以后日期
            days++;                                        //天数加 1
    cout<<year<<"年"<<month<<"月"<<day<<"日是本年度的第"<<days<<"天。\n";
    return 0;
}
```

运行程序：

```
输入年月日: 2020 12 31
```

输出结果：

```
2020 年 12 月 31 日是本年度的第 366 天
```

3.3.3　上机练习

【上机目的】

- 掌握开关语句的语法格式。

- 掌握多分支条件语句和开关语句之间的区别。

1. 编写程序，利用开关语句实现实数的加、减、乘、除四则运算。输入数据的格式为：

操作数 1　运算符　操作数 2

2. 题目要求如例 3.6，将输入百分制成绩转换成五分制成绩，用开关语句完成多分支功能。

【思考与练习】

1. 单选题

(1) 下列关于开关语句的描述中，(　　)是正确的。

A. 开关语句中 default 子句既可以没有，也可以有一个

B. 开关语句中的每个语句序列中必须有 break 语句

C. 开关语句中 default 子句只能放在后面

D. 开关语句中 case 子句后面的表达式是整型表达式

(2) 下面有关 break 语句的描述中，(　　)是错误的。

A. break 语句可用于循环体内，它将退出本重循环

B. break 语句可用于开关语句中，它将退出本开关语句

C. break 语句可用于 if 体内，它将退出 if 语句

D. break 语句在一个循环体内可以出现多次

(3) 执行以下代码的结果是(　　)。

```
int i(5);
switch(i)
{
    case 0:  cout<<"0"<<' '
    default:  cout<<"default"<<' ';
    case 1:  cout<<"1"<<' ';
}
```

A. 屏幕无任何显示　B. default 1　C. default　D. 0 default 1

2. 程序分析

分析程序的执行结果。

```
#include <iostream>
using namespace std;
void main()
{
    int a(1),b(6),c(4),d(2);
    switch(a++)
    {
        case 1:c++;d++;
        case 2:switch(++b)
            {
                case 7:c++;
                case 8:d++;
            }
        case    3:c++;d++;
```

```
            break;
        case 4:c++;d++;
    }
    cout<<c<<','<<d<<endl;
}
```

3.4 案例 9——猜数小游戏

3.4.1 问题描述及程序代码

1. 问题描述

系统自动生成一个 1～20 的随机数，游戏玩家猜数，程序根据玩家猜数给出相应的提示，如果猜的数比生成的随机数大，提示“您猜大了!”；如果猜的数比随机数小，提示“您猜小了!”；如果猜的数正好是随机数，提示“恭喜您猜对了!”。然后根据所猜次数，程序给出评价，如果猜数少于 4 次，显示器上提示“太棒了！只用了 x 次”；如果猜数大于或等于 4 次小于 8 次，程序会提示“加油哦！用了 x 次”；如果需要猜测 8 次以上，提示“运气较差呀！用了 x 次”，x 表示实际猜测的次数，直到猜中后游戏结束。

根据问题描述，首先画出程序传统流程图，如图 3.27 所示。接着依据程序流程图能非常清晰地转化成程序代码。

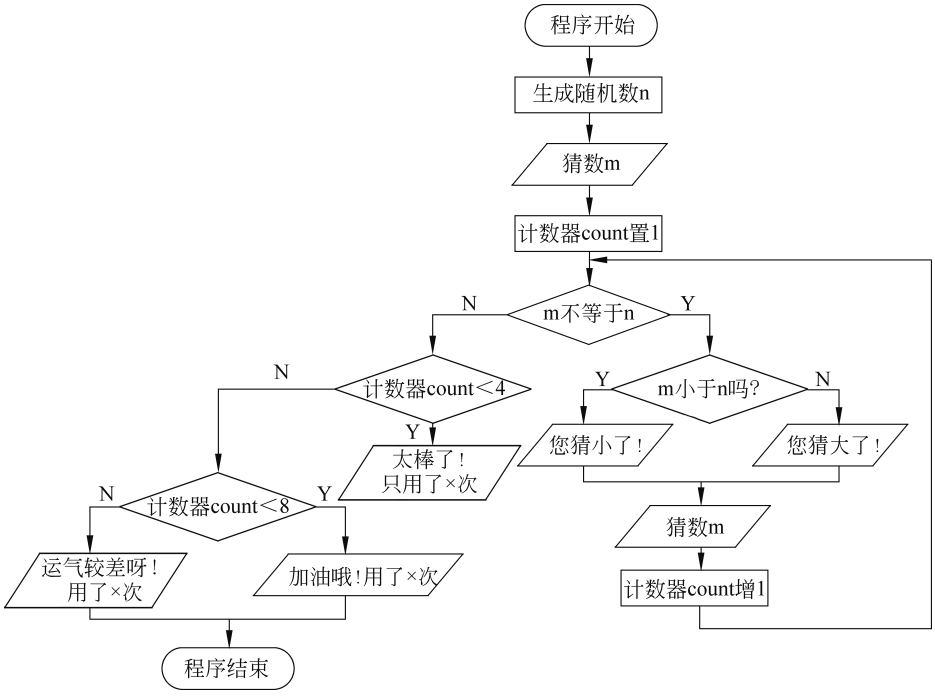

图 3.27　猜数小游戏流程图

从流程图中看，程序包含二分支结构和循环结构，可见多种结构之间是可以互相嵌套的。

2. 程序代码

```
#include <iostream>
#include <time.h>
using namespace std;
int main()
{
    srand(time(0));                                    //设置随机种子
    //生成一个随机数
    int n=rand()%20+1;                                 //闭区间[1,20]
    int m;                                             //声明猜数变量
    int count=1;                                       //计数器
    cout<<"输入一个 1～20 的整数:";
    cin>>m;                                            //输入数据
    while(m!=n)                                        //猜数不正确
    {                                                  //花括号里是循环体
        if(m<n)                                        //猜数小于随机数
            cout<<"您猜小了!"<<endl;
        else                                           //猜数大于随机数
            cout<<"您猜大了!"<<endl;
        cout<<"输入一个[1,20]区间的整数:";             //提示输入
        cin>>m;                                        //接着猜数
        count++;                                       //计数器增 1
     }
    if(count<4)                                        //猜数少于 4 次
        cout<<"太棒了! 只用了"<<count<<"次就猜对了!"<<endl;
    else if(count<8)                                   //猜数大于或等于 4 小于 8 次
        cout<<"加油啊! 用了"<<count<<"次!"<<endl;
    else                                               //猜数大于或等于 8 次
        cout<<"运气较差呀! 用了"<<count<<"次"<<endl;
    return 0;
}
```

3. 程序分析及运行结果

在本程序中,有两个新的知识点。首先介绍如何让系统生成一个随机数,本案例是通过调用随机数生成函数完成的。其次,输入数据并与系统生成的随机数进行比较的操作会重复执行,把这些具有规律性的重复操作的语句称为循环体,能否继续重复,取决于循环的终止条件,本案例中循环终止的条件是输入数据和系统生成的随机数相等。程序测试如图 3.28 所示。

```
输入一个[1,20]区间的整数:10
您猜小了!
输入一个[1,20]区间的整数:15
您猜大了!
输入一个[1,20]区间的整数:12
您猜大了!
输入一个[1,20]区间的整数:11
加油啊! 用了4次!
```

图 3.28　程序执行结果

3.4.2　生成随机数

1. 生成伪随机数

C++ 语言里的 rand()函数能够生成伪随机数,此函数不需要参数,其返回值是一个 0 至最大整数之间的任意整数。之所以称为伪随机数,因为默认是以 1 为随机种子(即起始值),随机数生