第 3 章　程序控制结构

本章主要介绍实现结构化程序设计的 3 种基本结构，即顺序结构、选择结构和循环结构，以及实现这 3 种结构的相关语句、3 种结构程序的设计方法及条件运算符的应用。

程序中语句的执行顺序称为程序结构。计算机程序是由若干条语句组成的语句序列。如果程序中的语句是按照书写顺序执行的，称为顺序结构；如果某些语句是按照当时的某个条件来决定是否执行，称为选择结构；如果某些语句要反复执行多次，称为循环结构。

3.1　顺序结构程序设计

顺序结构的程序是自上而下顺序执行的各条语句。顺序结构的程序流程图如图 3.1 所示。

下面通过简单的顺序结构程序实例介绍顺序结构设计的方法。

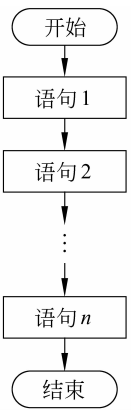

图 3.1　顺序结构流程图

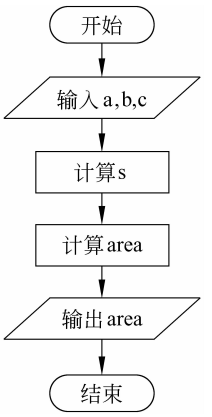

图 3.2　例 3.1 程序流程图

【例 3.1】 输入三角形的三条边长，求三角形面积。

假设输入的三条边 a、b、c 能构成三角形，求三角形面积公式如下：

$$\text{area} = \sqrt{s(s-a)(s-b)(s-c)}$$

其中，$s=(a+b+c)/2$。

分析：在计算三角形公式中出现了平方根函数，在 C 语言中，使用一些常用数学函数可以直接调用系统库文件 math.h(C++ 为 cmath)。常用的几个函数有 fabs()(实数绝对值函数)、sqrt()(平方根函数)，sin()(正弦函数)，log10()(以 10 为底的对数函数)和 rand()(随机函数)等，请参考附录 C。本例的程序流程图如图 3.2 所示。程序清单如下：

```
#include <iostream>
using namespace std;
#include <cmath>                          //包含数学函数 sqrt()所在的库文件,也称头文件
void main()
```

```
{
    float s,a,b,c,area;
    cin>>a>>b>>c;                          //给 a、b、c 赋初值
    s=1.0/2*(a+b+c);
    area=sqrt(s*(s-a)*(s-b)*(s-c)); //计算面积
    cout<<"area="<<area<<endl;          //输出结果
}
```

程序运行时输入：

```
3.5  4.6  5.1↙
```

运行结果为：

```
area=7.834539
```

3.2 选择结构程序设计

选择结构体现了程序的判断能力。在程序执行过程中能根据判断条件确定执行某个操作，这种程序结构称为选择结构，又称为分支结构。选择结构有 3 种形式，即单分支结构、双分支结构、多分支结构。C 语言提供了相应的语句，即 if-else 语句和 switch 语句，下面介绍实现 3 种选择结构的方法。

3.2.1 if 语句的 3 种形式

1. 简单单分支 if 语句

一般格式如下：

```
if(表达式)
    语句
```

单分支选择结构只考虑某个条件满足时执行相应的操作，否则顺序执行选择语句后的语句。

说明：

(1) 表达式可为任何类型，常用的是关系表达式和逻辑表达式。

(2) 语句可以是任何可执行语句(多于一条语句时，必须用一对{}括起来，形成复合语句)。

简单的单分支结构如图 3.3 所示。

【例 3.2】 输入一个学生成绩，如果及格则输出“good!”，否则什么也不做。流程图如图 3.4 所示。程序清单如下：

```
#include <iostream>
using namespace std;
void main()
{
    float g;
    cin>>g;                                //输入成绩
```

```
    if(g>=60)                          //判断是否及格
        cout<<"good!"<<endl;           //输出结果
}
```

运行程序时输入：

```
67↙
```

输出结果为：

```
good!
```

再次运行程序时输入：

```
50↙
```

输出结果为空。

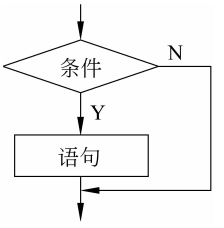

图 3.3　单分支结构流程图

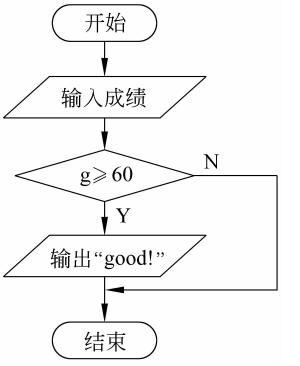

图 3.4　例 3.2 程序流程图

【例 3.3】 将两个整数 a 和 b 中的大数存入 a 中，小数存入 b 中。

分析：首先将 a、b 进行比较，如果 a 已经为大数，则无需变动；否则将两数对调，即将 a 存入 b 中，将 b 存入 a 中。对调时先设一个中间变量 temp 暂存数据，然后执行以下操作步骤：

(1) 将 a 赋给 temp，语句为 temp＝a;。

(2) 将 b 赋给 a，语句为 a＝b;。

(3) 将 temp 赋给 b(原来 a 的值)，语句为 b＝temp;。

程序清单如下：

```
#include <iostream>
using namespace std;
void main()
{
    int a,b,temp;
    cin>>a>>b;
    if(a<b)
    {
        temp=a;                        //通过变量 temp 完成两个数对调
        a=b;
        b=temp;
```

```
    }
    cout<<"a="<<a<<","<<"b="<<b<<endl;
}
```

运行程序时输入：

```
5↙ 9↙
```

运行结果为：

```
a=9,b=5
```

说明：实现两数对调时 3 条语句都要执行，因此，temp＝a；a＝b；b＝temp；构成复合语句，要用{}括起来。

2. 简单双分支 if-else 语句

一般格式如下：

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

双分支结构是按照条件判断出执行两个流程中的哪个流程的语句。如图 3.5 所示。

说明：

(1) 计算表达式的值，如果为真(非 0)，则执行语句 1，否则执行语句 2。

(2) 语句 1 和语句 2 可以是任何语句，也可以是空语句。如果语句多于一条，必须用一对{}括起来，形成复合语句。

(3) 表达式可以是任何类型，常用的是关系表达式或逻辑表达式。

(4) else 后面是 if 的子句，与 if 配对，不能单独出现。

可以使用 if-else 建立简单双分支结构，简单的 if-else 语句是指在 if 分支的语句体内不含有 if 语句。

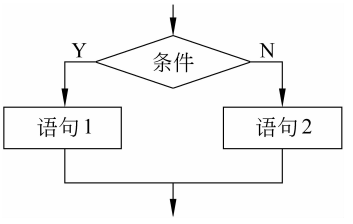

图 3.5　双分支结构流程图

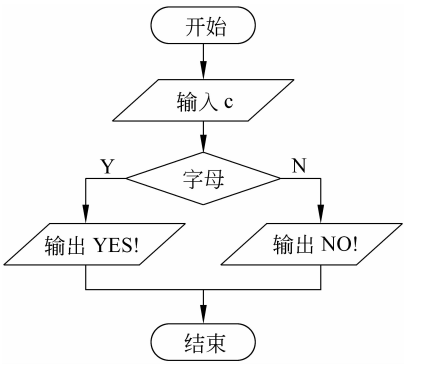

图 3.6　例 3.4 程序流程图

【例 3.4】 输入一个字符，若是英文字母则输出“YES!”，否则输出“NO!”。程序流程图如图 3.6 所示。

程序清单如下：

```
#include <iostream>
using namespace std;
void main()
{
    char c;
    cin>>c;
    if(c>='a'&&c<='z'||c>='A'&&c<='Z')
        cout<<"YES!"<<endl;
    else
        cout<<"NO!"<<endl;
}
```

运行程序时输入：

```
S↙
```

输出结果为：

```
YES!
```

再运行一次，输入：

```
8↙
```

输出结果为：

```
NO!
```

说明：书写 if-else 语句时，if 和 else 换行并对齐。

3. 嵌套 if-else 语句

if-else 均内嵌的分支结构的一般形式为：

```
if (表达式)
{
    …
    if (表达式)
        语句 1
    else
        语句 2
    …
}
else
{
    …
    if (表达式)
        语句 3
    else
        语句 4
    …
}
```

说明：

(1) 如果只有一条语句，可省略{}。例如，内嵌的“if（表达式）语句1 else 语句2”整体可看做一条复合if语句，上面的整个嵌套结构也可以看做一条复合语句。

(2) 当没有使用{}显式说明嵌套时，if-else的配对原则是：else总是与同一层最近的尚未配对的if语句配对。

【例3.5】 求如下所示分段函数 y 的值。

$$y=\begin{cases}-1 & x<0\\ 0 & x=0\\ 1 & x>0\end{cases}$$

分析：这是一个判断变量符号的问题，结果有3种可能，若x<0，则y=-1；若x=0，则y=0；若x>0，则y=1。在程序中内嵌一个双分支结构，其流程图如图3.7所示。程序清单如下：

```
#include <iostream>
using namespace std;
void main()
{
    int x,y;
    cin>>x;
    if(x<0)
        y=-1;
    else                                    //这里的else子句是与上面的if(x<0)配对
        if(x==0)
            y=0;
        else
            y=1;                            //这里的else子句是与最近的if(x==0)配对
    cout<<"x="<<x<<","<<"y="<<y<<endl;
}
```

运行程序时输入：

```
-5↙
```

输出结果为：

```
x=-5,y=-1
```

此程序是在else分支中内嵌if-else语句的，这是较为常用的设计方法，也可以在另一分支嵌套。它的表现形式如图3.8所示。程序清单如下：

```
#include <iostream>
using namespace std;
void main()
{
    int x,y;
    cin>>x;
```

```
    if(x>=0)
        if(x>0)
            y=1;
        else                                    //else 与 if(x>0)配对
            y=0;
    else                                        //else 与 if(x>=0)配对
        y=-1;
    cout<<"x="<<x<<","<<"y="<<y<<endl;
}
```

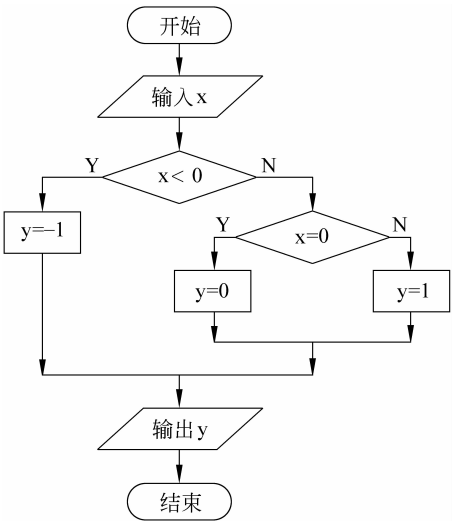

图 3.7　内嵌一个双分支结构

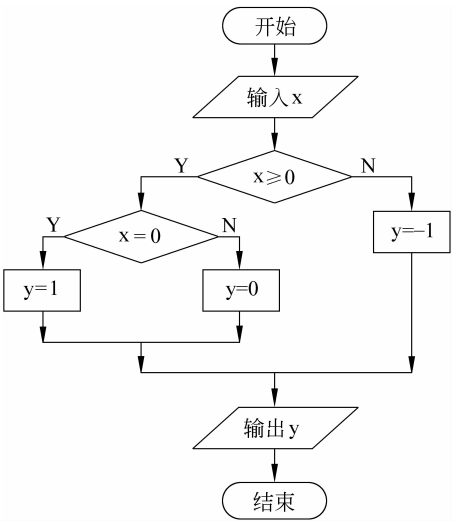

图 3.8　内嵌 if-else 语句的另一种形式

C 语言不限制内嵌层数，程序流程如果多于两个分支，称为多分支选择结构。多分支选择结构可以提出 n 个条件控制程序，最多有 $n+1$ 个执行语句(复合语句)。根据条件选择执行哪个语句。如果所有条件都不满足，则执行第 $n+1$ 个语句。多分支选择结构使用嵌套的 if-else 语句实现，一般格式如下：

```
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 语句 n+1
```

说明：多分支选择结构实际上是只在 else 分支上嵌套，且省略了{}的变体形式。

例如，将百分制成绩 s1 转换为 5 分制成绩 s2，可以用如 if-else-if 语句来完成。如图 3.9 所示。

```
if(s1<60) s2='E';
    else if(s1<70) s2='D';
        else if(s1<80) s2='C';
            else if(s1<90) s2='B';
```

```
else s2='A';
```

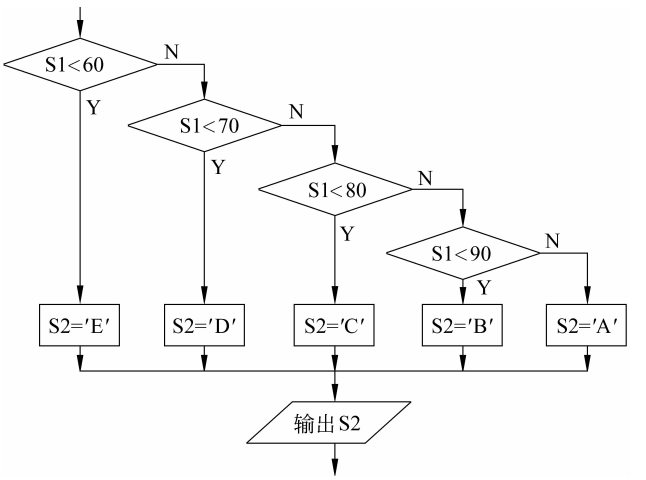

图 3.9　多分支嵌套结构流程图

注意：if-else 结构尽量缩格对齐。

说明：

(1) 习惯在 else 分支上嵌套，以上程序段中应注意的一点是 if 和 else 的配对关系。例如：

```
if(a>b)
    if(b<c)
        c=a;
    else
        c=b;
```

(2) 如果要使程序逻辑清晰，可将 if 和 else 的子句设计成复合语句(即用{}括起来)，这可保证 if 与 else 正确配对，例如：

```
if(a>b)
{
    if(b<c)
        c=a;
}
else
        c=b;
```

3.2.2　条件运算符？：

条件运算符“？：”是一种三目运算符，它的特点是有 3 个操作数。实际上条件运算符实现了简单的 if-else 的语句功能。表达式形式如下：

(表达式 1)?(表达式 2):(表达式 3)

其功能是：先计算表达式 1，如果表达式 1 的值是非 0(真)，则其结果取表达式 2 的值，否则，取表达式 3 的值。

使用条件表达式选择合适的数据给变量赋值非常方便，例如，当 a＝3，b＝4 时：

```
max=(a>b)?a:b;
```

变量 max 取变量 b 的值，为 4。

说明：

(1) 条件运算符的结合方向是自右向左，若有以下表达式：

```
a>b?a:c>d?c:d
```

则相当于：

```
a>b?a:(c>d?c:d)
```

(2) 3 个表达式的类型没有限制。如：

```
x?'a': 'b'
x>y?1:1.5
```

都是合法的条件表达式。

(3) 条件表达式可以作函数的参数，简化程序结构。如：

```
printf("max of %d,%d is %d\n",a,b,(a>b)?a:b);
```

【例 3.6】 输入一个英文字母，判断是否为大写字母，若是大写字母直接输出，否则转换成大写字母输出。程序清单如下：

```
#include <iostream>
using namespace std;
void main()
{
    char ch;
    cin>>ch;
    ch=(ch>='A'&& ch<='Z')?ch:(ch-32);
    cout<<ch<<endl;
}
```

程序运行时输入：

```
d↙
```

输出运行结果：

```
D
```

3.2.3 switch 语句实现多分支选择结构

在选择结构中，条件表达式的值若等于多个常量之一，可用 switch 多分支语句实现，以增加程序的可读性。

switch 语句的一般格式如下：

```
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 3: 语句组 3;break;
    ⋮
    case 常量表达式 n:语句组 n;break;
    [default :语句组 n+1;]
}
```

其功能是：以关键字 switch 后面的表达式为判断条件，在多分支中选择一个分支操作。switch 语句的执行过程如图 3.10 所示。

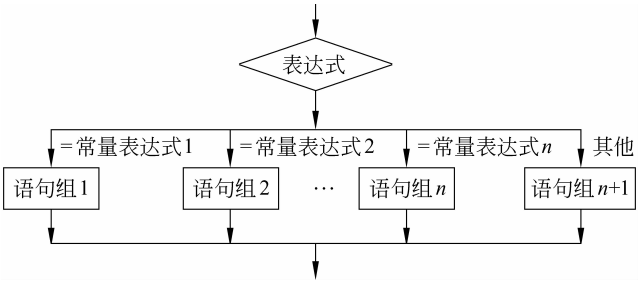

图 3.10　switch 语句流程图

说明：

(1) ANSI 标准允许 switch 后的表达式的值可以为整型、字符型和枚举型。当表达式的值等于其中一个 case 后面的常量表达式的值时，就执行此 case 后面的语句；若所有 case 中的常量表达式的值都没有相匹配的，就执行 default 后面的语句。

(2) switch 语句中的每一个 case 相当于一个语句标号，具体哪个 case 后面的语句能够执行，是根据 switch 后的表达式的值决定的。case 后的常量表达式可以是整型、字符型和枚举型的常量。注意各 case 后的常量表达式的值必须互不相同。

(3) “语句组”可以是一条或多条合法的 C 语言语句。

(4) break 是 C 语言的一种语句，其功能是中断正在执行的语句。在 switch 语句中的作用是：执行完某个语句组后，将退出该 switch 语句。如果省略了 break 语句，则执行完某个语句组后，将继续执行其后边的语句组。

【例 3.7】 输入 i，根据 i 的值输出信息。程序清单如下：

```
#include <iostream>
using namespace std;
void main()
{
    int i;
    cin>>i;                                          //输入控制变量 i
    switch(i)                                        //i 的值作为条件
    {
        case 1: cout<<"I am in case 1."<<endl;       //若 i 为常量 1,则执行此输出语句
            break;                                   //执行输出语句后跳出 switch 语句
        case 2: cout<<"I am in case 2."<<endl;       //若 i 为常量 2,则执行此输出语句
```