第 4 章　程序的基本结构与基本语句

结构化程序设计方法由 E.Dijkstra 等人提出，是人们公认的具有“良好风格”的程序设计方法之一，它使用顺序、分支、循环三种基本控制结构构造程序。C 语言完全支持结构化程序设计方法，并提供了相应的语句。本章介绍算法及其描述、C 程序的三种基本控制结构和基本语句。

4.1　程序与基本语句

4.1.1　程序

什么是计算机程序？图灵奖获得者、瑞士计算机科学家 N.Wirth 教授曾提出“程序=数据结构+算法”，可见，数据结构和算法是构成程序的两个重要组成部分。

1. 数据结构

数据结构主要强调两个方面的内容：

- 计算机待解决问题的数据元素之间的关系，即数据元素之间的逻辑结构和数据在计算机内存中的存储结构。
- 与这些结构相适应的操作。

在程序设计中，算法的设计与实现，依赖于数据的逻辑结构和存储结构，数据结构的选择直接决定了算法的质量，而算法则决定了程序的质量。

学习语言，一定要重视数据结构方面的信息，否则，就不能深入理解 C 语言。例如 int、float 等数据类型在内存中的存放方式。只有了解了这些结构，才能很好地理解相关算法在这些结构上所做的操作，在第 2 章的学习中，读者应当已经体会到这一点了。

2. 算法

（1）算法的概念

算法是计算机的灵魂。算法实际上就是处理某一个具体问题的方法和步骤，是有穷动作的序列。通常一个问题可以有多种算法，一个给定算法解决一个特定的问题。

算法具有下列 5 个重要特性：

1）输入。一个算法有零个或多个输入（即算法可以无输入），这些输入通常取自于某个特定的对象集合。

2）输出。一个算法有一个或多个输出（即算法必须要有输出），通常输出与输入之间有着某种特定的关系。

3）有穷性。一个算法必须（对任何合法的输入）在执行有穷步之后结束，且每一步都在有穷时间内完成。

4）确定性。算法中的每一条指令必须有确切的含义，不存在二义性，并且在任何条件下，对于相同的输入只能得到相同的输出。

5）可行性。算法描述的操作可以通过已经实现的基本操作执行有限次来实现。

算法与程序不同。程序是对一个算法使用某种程序设计语言的具体实现，原则上，任一算法可以用任何一种程序设计语言实现。算法的有穷性意味着不是所有的计算机程序都是算法。例如，操作系统是一个在无限循环中执行的程序，而不是一个算法，然而可以把操作系统的各个任务看成是一个单独的问题，每一个问题由操作系统中的一个子程序通过特定的算法来实现，得到输出

结果后便终止。

为了让读者理解如何设计一个算法，下面举一个简单的例子。

【例 4.1】 设计一个算法：对任意给定的 3 个整数 x、y、z，求出其最大值。

分析：这个算法先比较出 x 和 y，得到一个大的值，再用这个值与 z 比较，将两者中大的值作为结果输出即可。

可将这个算法描述为：

（1）输入变量 x、y 和 z 的值。

（2）比较 x 和 y。如果 x>y 则 x 存入 max；否则，y 存入 max。

（3）比较 max 与 z。如果 z>max，则将 z 存入 max。

（4）输出结果 max。

显然，这是一个有穷的动作序列，算法中的步骤是有限的；描述每一个动作的指令的含义是明确的，没有其他的理解（二义性）；这个算法对任意的 3 个整数都是有效的，例如输入是 8、2、0，则输出为 8；输入是 1、5、3，则输出是 5。

（2）算法的描述

我们在设计了一个算法之后，还必须清楚、准确地将所设计的求解步骤表达出来，即描述算法。算法不能像诗歌、散文那样富有诗情画意。算法完全是一种“实话实说”，把你构思好的解题步骤严密地、直接地表示出来。

表示算法的方法很多，常用的有自然语言、流程图、伪代码和计算机程序设计语言等。

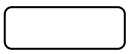
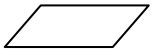
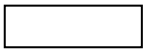
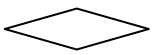
1）用自然语言记述算法。上面对例 4.1 算法的描述就是用汉语这样的自然语言来表示算法的。用自然语言描述算法，最大的优点是容易理解，缺点是容易出现二义性，并且算法通常都很冗长。而且对于分支、循环结构，自然语言表示法极不方便，不能清晰地显示出来。

2）用流程图来表示算法。流程图是人们经常用来描述算法的工具。流程图是指用规定的图形符号，通过方向性的线段连接而成的有方向的图形。表 4-1 列出了几种常用的流程图符号。图 4.1 是例 4.1 的流程图表示。

用流程图描述算法，具有直观、结构清晰、条理分明、通俗易懂、便于检查修改及交流等优点。流程图表示的算法既独立于任何特定的计算机，又独立于任何特定的计算机程序设计语言。这种独立性是很重要的，它不要求使用者熟悉特定的计算机及特定的程序设计语言。

流程图的缺陷是严密性不如程序设计语言，描述的灵活性不及自然语言。

表 4-1　常用的流程图符号

图 形	名 称	含 义
	起止框	表示程序开始或结束
	输入输出框	表示数据的输入输出，有一个入口和一个出口
	处理框	表示处理或运算的功能，有一个入口和一个出口
	判断框	表示判断和选择，有一个入口，两个或多个出口
	连接点	表示转向流程图的它处或从它处转入
	流程线	表示算法执行的路径，箭头代表方向

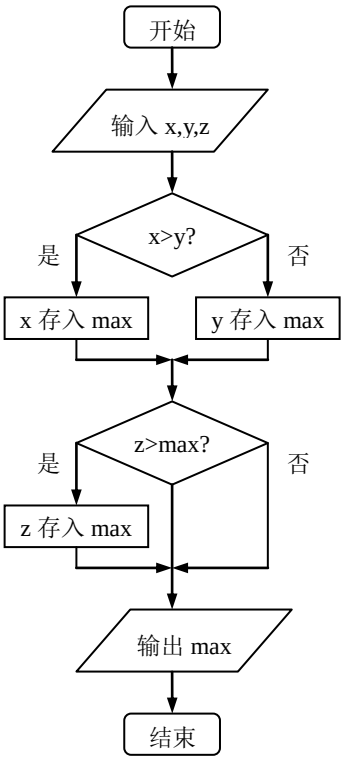


图 4.1　3 个值中求最大值的流程图

3）用伪代码来描述算法。伪代码是介于自然语言和程序设计语言之间的方法。计算机科学家对伪代码的书写形式没有做出严格的规定，它采用某种程序设计语言的基本语法，操作指令可以结合自然语言来设计。至于算法中自然语言的成分有多少，取决于算法的抽象级别。抽象级别高的伪代码自然语言多一些，抽象级别低的伪代码程序设计语言的语句多一些。只要具有一些程序设计语言基础的人都能阅读伪代码。

4）用程序设计语言来实现算法。事实上，用程序设计语言来表示算法，就是对算法的实现。用计算机程序设计语言表示算法显得清晰、简明、一步到位，写出的算法能由计算机处理。

例 4.1 的 C 语言程序如下：

```
#include <stdio.h>

int main()
{
    int x,y,z,max;

    scanf("%d%d%d",&x,&y,&z);
    if(x>y)
        max=x;
    else
        max=y;
    if(z>max)
        max=z;
    printf("max=%d\n",max);
```

```
    return 0;
}
```

程序运行结果截图如下（假设输入的 3 个整数分别为：4 3 6）：

3. 程序的评价

怎样评价一个程序设计的质量好坏呢？在本书中主要介绍两点，更多的评价标准在“数据结构”以及“C++面向对象程序设计”等课程中有介绍。

（1）正确性和健壮性

所谓“正确性”是指程序要能正常运行，对任何合法的输入能够得到正确的运行结果。

所谓“健壮性”是指对于错误的输入数据，程序要能辨别并做出处理，而不是产生错误动作或陷入瘫痪。

总之，一个好的程序，应该保证在运行过程中不管遇到什么情况都能正常工作，并输出相应的正确结果或出错信息才行。

因此，在编制一个程序前，应该认真地考虑运行后出现的各种可能情况。程序编好后，要上机输入各种数据，特别是临界值的数据，在运行中发现问题并及时修改，以保证正确性和健壮性。

（2）结构清晰，可读性好

所谓“可读性”是指程序条理清晰、简洁明快、易于理解，且在关键处加上注释，这样不但上机调试也方便，错误较少且很容易修改，这样的程序质量较高。

一个好的程序应该有较强的可读性，这是编写程序时应考虑的很重要的一个方面。一个程序不管有多复杂，编程者本人或其他阅读程序的人都应该容易理解，容易检查和修改。这样，对程序的使用和维护可带来很大的方便。

要得到一个可读性较好的程序，编程时应将一个较为复杂的任务分解成若干个较为简单的问题，并对这些简单的问题编写程序，要针对问题设计合适的结构，确定良好、精确的算法并且掌握熟练的编程技巧。

另外，在编程中要认真仔细选择变量名，清晰、明确的安排语句，并在程序适当的位置上加上必要的注释。

4.1.2 C 基本语句

任何复杂的 C 程序都可以由 3 种基本结构组成：顺序结构、分支结构、循环结构。本章接下来将分别介绍这 3 种基本结构的程序设计。

C 语言提供了多种语句来实现这些程序结构。C 语句可分为以下 5 类：

- 表达式语句；
- 控制语句；
- 函数调用语句；
- 复合语句；
- 空语句。

（1）表达式语句。表达式语句由表达式加上分号“;”组成。

其一般形式为：

表达式;

执行表达式语句就是计算表达式的值。

例如：

```
x=y+z;     //赋值语句
y+z;       //加法运算语句，但计算结果不能保留，无实际意义
i++;       //自增 1 语句，i 值增 1
```

（2）控制语句。控制语句用于控制程序的流程，以实现程序的各种结构方式。它们由特定的语句定义符组成。C 语言有 9 种控制语句，可分成以下 3 类：

①条件判断语句：if…else 语句、switch 语句；

②循环语句：for 语句、while 语句、do while 语句；

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

（3）函数调用语句。由函数名、实际参数加上分号“;”组成。

其一般形式为：

函数名(实际参数表);

执行函数语句就是调用函数体并把实际参数赋予函数定义中的形式参数，然后执行被调函数体中的语句，求取函数值 (在第 5 章函数中将详细介绍) 。

例如：

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

（4）复合语句。把多个语句用大括号{}括起来组成的一条语句称复合语句。在程序中应把复合语句看成是单条语句，而不是多条语句。

例如：

```
{   x=y+z;
    a=b+c;
    printf("%d%d",x,a);
}
```

是一条复合语句。

复合语句内的各条语句都必须以分号“;”结尾，在括号“}”外不能加分号。

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

例如：

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

本语句的功能是，只要从键盘输入的不是【Enter】则重新输入。这里的循环体为空语句。

4.2 顺序结构

顺序结构是一种最简单、最基本的程序结构。它的特点是，这个结构中的语句都是按照从上到下的顺序执行，可见，这种结构的语句只有一个入口和一个出口。流程图 4.2 就是顺序结构，它表示程序运行的语句和方向，它执行的顺序是语句 1、语句 2、语句 3。顺序结构的语句看起来很简单，但是它是所有程序的基础。组成顺序结构程序的几种基本语句：赋值语句、逗号表达式语句、格式化输入/输出语句、非格式化输入/输出语句。输入输出语句已经在第 3 章介绍，本节介绍赋值语句、逗号表达式语句。

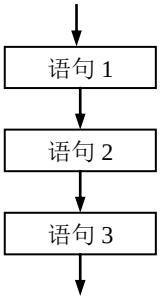


图 4.2　顺序结构流程

【任务 4.1】利用海伦公式求三角形面积

任务描述：古希腊数学家海伦(Heron，约公元 75 年)，对各学科都有着广泛的兴趣，他的著作涉及几何学、力学和计算学。《测量学》一书是他关于数学的代表作，书中给出了各种几何图形、物体面积及体积的精确与近似的计算方法。其中，以根据三角形的三条边的边长来计算三角形面积的公式最为著名，若已知三角形 ABC 的边长为 a、b、c，那么它的面积公式是：$s_{\Delta abc}=\sqrt{p(p-a)(p-b)(p-c)}$，其中 p 为半周长，即 $p=\frac{a+b+c}{2}$。这个公式在《测量学》中首次出现，所以人们称此公式为海伦公式。

任务分析：

程序的流程如下：

1）在键盘上输入三角形三条边的边长 a、b、c；

2）计算出半周长 p；

3）利用海伦公式即可计算出三角形的面积 s；

4）输出三角形的面积。

用流程图表示如图 4.3 所示。

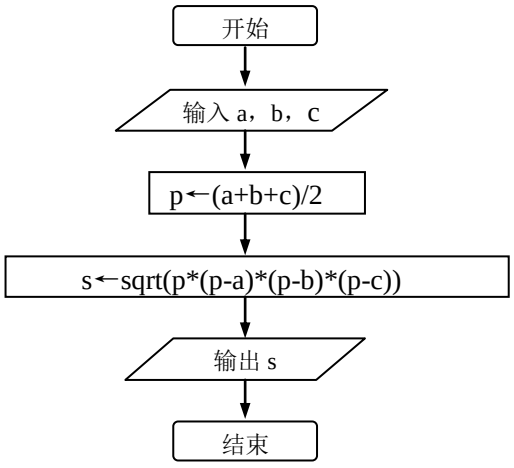


图 4.3　任务 4.1 算法流程图

根据算法流程的描述，要计算半周长 p 的值以及利用海伦公式计算三角形面积 s 的值，这就需要赋值运算。

4.2.1 赋值语句

1. 赋值运算符和赋值表达式

“=”即赋值运算符，是一个二元运算符，其作用是构造“赋值表达式”。

由赋值运算符将一个变量和一个表达式连接起来的式子称为赋值表达式。它的一般格式为：

变量名 = 表达式

赋值表达式的作用是将赋值运算符右边的表达式的值赋给左边的变量。值得一提的是，这里的表达式可以是常量（如 a=5）、变量（如 b=a）及各种表达式（如 b=3*5）。赋值运算符的结合性为“自右向左”，赋值表达式本身的值取左边变量的值。列举若干赋值表达式：

```
a=3*5              //a 的值为 15，赋值表达式的值为 15
a=b=5              //等价于 a=(b=5)  a、b 的值为 5，赋值表达式的值为 5
a=5+c=6            //表达式错误，赋值运算符优先级低于算术、关系和逻辑运算符
a=5+(c=6)          //c 的值为 6，a 和赋值表达式的值均为 11
```

注意：

（1）在进行赋值运算时，若“=”两边的数据类型不一致，系统会自动把“=”右边表达式的数据类型转换成左边变量的类型，然后赋给左边的变量。例如：

```
int x;
x=3.14;      // 对 3.14 取整赋给 x，x 的值为 3
```

（2）赋值运算符“=”不是数学上的“等号”。

2. 复合赋值运算符

在 C 语言中，所有的二元算术运算符和位运算符均可与赋值运算符组合成复合赋值运算符，共有 10 个：

+ =、- =、* =、/ =、% =、<< =、>> =、& =、^ =、| =

其中，有关算术运算的复合赋值运算符最常用。复合赋值运算符的优先级和结合性与赋值运算符相同，其构造的赋值表达式格式为：

变量名 复合赋值运算符 表达式

等价于

变量名 = 变量名 运算符 （表达式）

例如：

```
a+=3          等价于     a=a+3
a*=b-5        等价于     a=a*(b-5)        // 小提示: 不等价于 a=a*b-5，括号不能省
a%=3+b        等价于     a=a%(3+b)
```

3. 赋值语句

赋值语句是 C 程序最基本的语句，用于实现程序中的计算功能，几乎每个有实用价值的程序中都包含赋值语句。赋值语句是由赋值表达式加一个分号（;）构成的一种表达式语句。

例如：

```
a=5;
a+=5;
a=b=5;    //等效于 b=5; 和 a=b;两个赋值语句
```

注意：赋值表达式是一种表达式，它可以出现在任何允许表达式的地方，而赋值语句则不能出现在表达式中。如：（x=y+1）>0 是个正确的表达式，而（x=y+1;）>0 则是不合法的。

思考：语句 a=b=c=5;合法，语句 int a=b=c=5;不合法，为什么？

4.2.2 逗号运算符与逗号表达式

逗号运算符“,”的优先级最低。用逗号运算符连接表达式就构成了逗号表达式，其一般格式为：

表达式 1，表达式 2，…，表达式 n

功能：从左到右依次求解每个表达式的值，并把最后一个表达式的值作为整个逗号表达式的值。例如，设：

int a=2, b;

则

a+=1, b=a+2

是一个逗号表达式，该逗号表达式的值为 5，即 b=a+2 的值。此时，a 的值为 3，b 的值为 5。

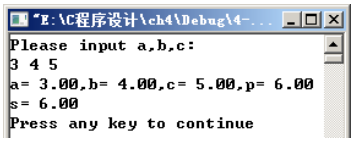
使用逗号表达式能够简约程序，但是逗号表达式不是必须的。

4.2.3 完成任务 4.1 的参考程序

参考程序：

```
#include <stdio.h>
#include <math.h>
int main()
{
    double a,b,c,p,s;
    printf("Please input a,b,c:\n");
    scanf("%lf%lf%lf",&a,&b,&c);

    p=1.0/2*(a+b+c);
    s=sqrt(p*(p-a)*(p-b)*(p-c));
    printf("a=%5.2f,b=%5.2f,c=%5.2f,p=%5.2f\n",a,b,c,p);
    printf("s=%5.2f\n",s);

    return 0;
}
```

程序运行结果截图如下（假设输入三条边分别是：3　4　5）：



思考：

这个程序在运行时会出现问题吗？问题出现在哪里？

4.2.4 顺序结构程序设计举例

【例 4.2】输入一个三位正整数，然后逆序输出。

分析：本题的关键是设计一个分离三位整数的个、十和百位的算法。设输入的三位正整数为 371。个位数可用对 10 求余的方法得到，如 371%10=1；百位数可用对 100 整除的方法得到，如

371/100=3；十位数既可通过将其变换为最高位后再整除的方法得到，如（371-3*100）/10=7，或者 371%100/10=7，也可通过将其变换为最低位再求余的方法得到，如（371/10）%10=7，流程图如图 4.4 所示。

参考程序：

```
#include<stdio.h>
int main( )
{
    int x, g,s,b;

    printf("请输入一个三位整数：");
    scanf("%3d",&x);          //语句①

    b=x/100;
    s=(x-b*100)/10;
    g=x%10;

    printf("%d 的逆序是%d%d%d\n",x,g,s,b);

    return 0;
 }
```

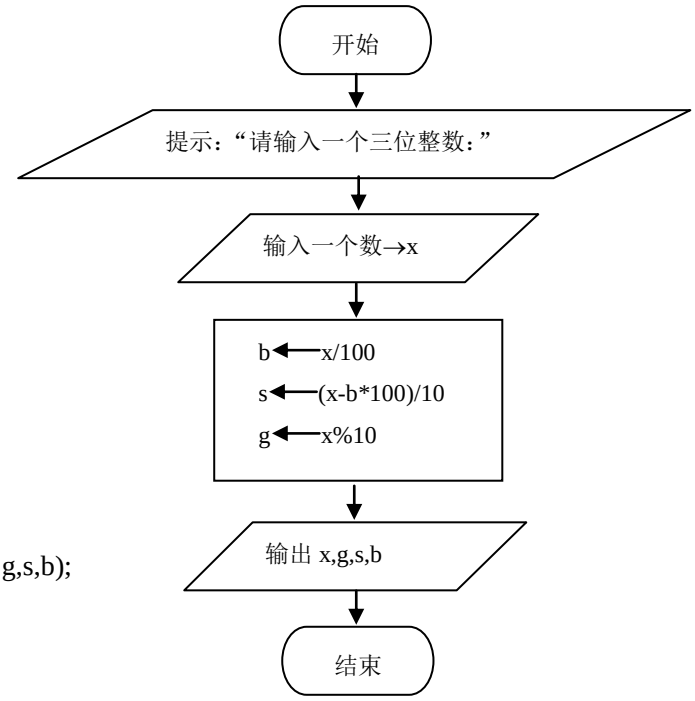
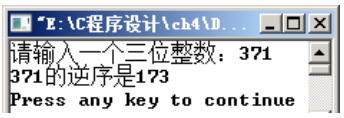


图 4.4　例 4.2 的流程图

执行语句①时，输入 4 位及以上的数，只取前 3 位。

程序运行结果截图如下（假设输入的整数是：371）：



4.3　分支结构

【任务 4.2】利用海伦公式求三角形面积（改进）

任务描述：输入三条边长值，利用海伦公式计算三角形的面积。

任务分析：程序运行时输入的三角形的 3 条边长必须满足：任何两边之和大于第三边。因此，在程序中需要对输入的三条边长是否满足三角形的要求进行判断。如果输入的三条边长满足三角形的要求，那么就计算这三条边形成的三角形面积，否则就应当不计算，并给出输入错误的提示。可以用流程图表示这样的判断如图 4.5 所示。

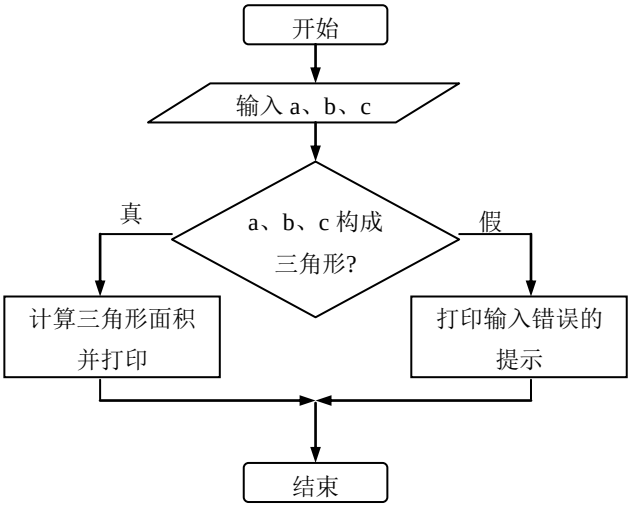


图 4.5 增加对输入数据合法性的判断

那么，如何用计算机语言进行判断呢？本节将介绍如何在程序中进行判断，如何使用 if 等分支语句进行分支结构的程序设计。

分支语句也称选择语句，它能控制程序根据给定条件，选择执行两个或两个以上分支程序段中的某一个。C 语言的分支语句包括 if 语句和 switch 语句。C 语言的 if 语句有 3 种基本形式。

4.3.1 单分支 if 语句

单分支 if 语句的格式是：

if(表达式) 语句

其语义是：如果表达式的值为真，则执行其后的语句，否则不执行该语句。其过程如图 4.6 所示。

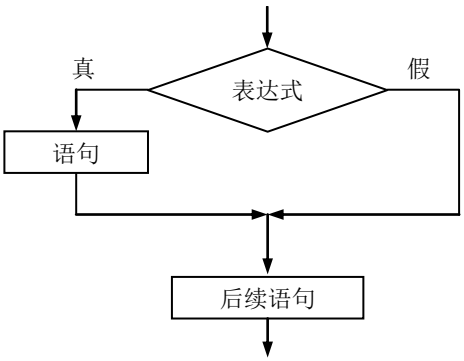


图 4.6 单分支 if 语句的执行过程

【例 4.3】输入两个整数，输出其中较大的数。

分析：

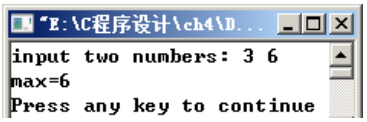
本例程序中，将输入两个数存入变量 a、b 中。把 a 先赋予变量 max，再用 if 语句判别 max 和 b 的大小，如果 max 小于 b，则把 b 赋予 max。因此 max 中总是大数，最后输出 max 的值。

参考程序：

```
#include <stdio.h>

int main()
```

```
{
    int a,b,max;

    printf("input two numbers: ");
    scanf("%d%d",&a,&b);

    max=a;
    if (max<b) max=b;

    printf("max=%d\n",max);

    return 0;
}
```

程序运行结果截图如下（假设输入的两个整数分别是：3 6）：



4.3.2 双分支 if 语句

双分支 if 语句的格式是：

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

其语义是：如果表达式的值为真，则执行语句 1，否则执行语句 2。

其执行过程如图 4.7 所示。

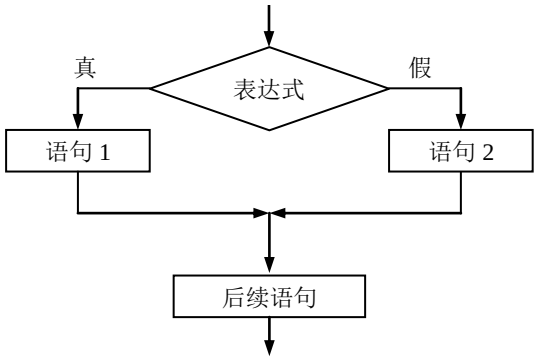


图 4.7　双分支 if 语句的执行过程

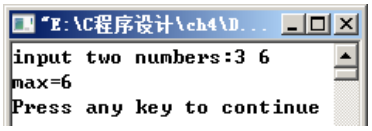
【例 4.4】输入两个整数，输出其中较大的数。改用 if…else 语句判别 a、b 的大小，若 a 大，则输出 a，否则输出 b。

参考程序：

```
#include <stdio.h>

int main()
```

```
{
    int a, b;

    printf("input two numbers:");
    scanf("%d%d",&a,&b);

    if(a>b)
        printf("max=%d\n",a);
    else
        printf("max=%d\n",b);

    return 0;
}
```

程序运行结果截图如下（假设输入的两个整数分别是：3 6）：

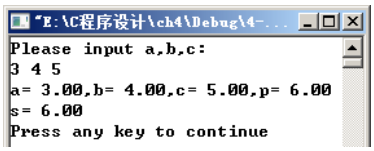


4.3.3 完成任务 4.2 的参考程序

参考程序：

```
#include<stdio.h>
#include<math.h>
int main()
{
    double a,b,c,p,s;

    printf("Please input a,b,c:\n");
    scanf("%lf%lf%lf",&a,&b,&c);

    if (a+b>c&&b+c>a&&c+a>b)
    {
        p = 1.0/2*(a+b+c);
        s = sqrt(p*(p-a)*(p-b)*(p-c));
        printf("a=%5.2f,b=%5.2f,c=%5.2f\n",a,b,c,p);
        printf("s=%5.2f\n",s);
    }
    else
        printf("input data error\n");
    return 0;
}
```

程序运行结果截图如下（假设输入三条边分别是：3　4　5）：



再次运行程序，输入三条边分别是：1　2　3，这时程序运行结果截图如下：

程序运行结果分析：第 1 次运行程序时，输入的三条边满足“任何两边之和大于第三边”，因此 if 条件为真，则计算这三条边形成的三角形面积并输出；第 2 次运行程序时，输入的三条边不满足条件，因此 if 条件为假，执行 else 后的语句，显示输入错误的提示。

在使用 if 语句中应注意以下问题：

（1）在 3 种形式的 if 语句中，在 if 关键字之后均为表达式。该表达式通常是逻辑表达式或关系表达式，但也可以是其它表达式，如赋值表达式等，甚至也可以是一个变量。

例如：

if(a=5) 语句;

if(b) 语句;

语法上都是允许的。

请注意，在：

if(a=5) …;

中表达式的值永远为非 0，所以其后的语句总是要执行的。如果表达式需要是一个判断两变量值是否相等的关系表达式，必须注意不要将判断等于号“==”写成了赋值号“=”，例如判断 x 是否等于 0 的条件应写成：

if(x==0)

（2）在 if 语句中，条件判断表达式必须用括号括起来，在语句之后必须加分号。

（3）在 if 语句的 3 种形式中，所有的语句应为单个语句，如果要想在满足条件时执行一组(多个)语句，则必须把这一组语句用{}括起来组成一条复合语句。但要注意的是在“}”之后不能再加分号。

例如：

```
if(a>b)
 {
      a++;
      b++;
 }
 else
 {
      a=0;
      b=10;
 }
```

【任务 4.3】百分制成绩转换成五级记分制成绩

任务描述：从键盘输入一个百分制成绩，转换成五级记分制成绩输出，即用五级记分制的等

级 A、B、C、D 和 E 分别表示 100~90、89~80、79~70、69~60、59~0 五类成绩。

任务分析：设输入的百分制成绩用变量 score 记录，输出的五级记分制等级用变量 grade 记录。只要判断输入的成绩 score 在 100~90、89~80、79~70、69~60、59~0 中的哪一段内，就能输出对应的等级 grade 是 A、B、C、D 或者 E。

多次使用单分支 if 语句格式可将任务 4.3 中的百分制转五级记分计分制，程序段描述如下：

```
if(score>=90) grade='A';                    //语句①
if(score>=80 && score<90) grade='B';
if(score>=70 && score<80) grade='C';
if(score>=60 && score<70) grade='D';
if(score<60) grade='E';
```

显然，这样多次使用单分支语句使得程序冗长，不够简洁。

若使用双分支 if 语句，只能将任务 4.3 中百分制转五级记分制的过程形式化描述为：

```
if(score>=90 )
    grade='A';
else
    {90 分以下成绩等级的转换}
```

显然，一个双分支 if 语句，哪怕是多个独立的双分支 if 语句也无法完成包含 5 个分支的任务 4.3。但多分支 if 结构或是嵌套的 if 语句可以实现任务 4.3。

4.3.4 多分支 if 语句和 if 语句的嵌套

1. 多分支 if 语句

当有多个分支选择时，可采用 if...else...if 语句，其一般形式为：

```
if(表达式 1)  语句 1;
else  if(表达式 2)  语句 2;
      else  if(表达式 3)  语句 3;
          ……
              else  if(表达式 m)  语句 m;
                    else  语句 n;
```

其语义是：依次判断表达式的值，当出现某个值为真时，则执行其对应的语句。然后跳到整个 if 语句之外继续执行后续程序。 如果所有的表达式均为假，则执行语句 n。然后继续执行后续程序。if-else-if 语句的执行过程如图 4.8 所示。

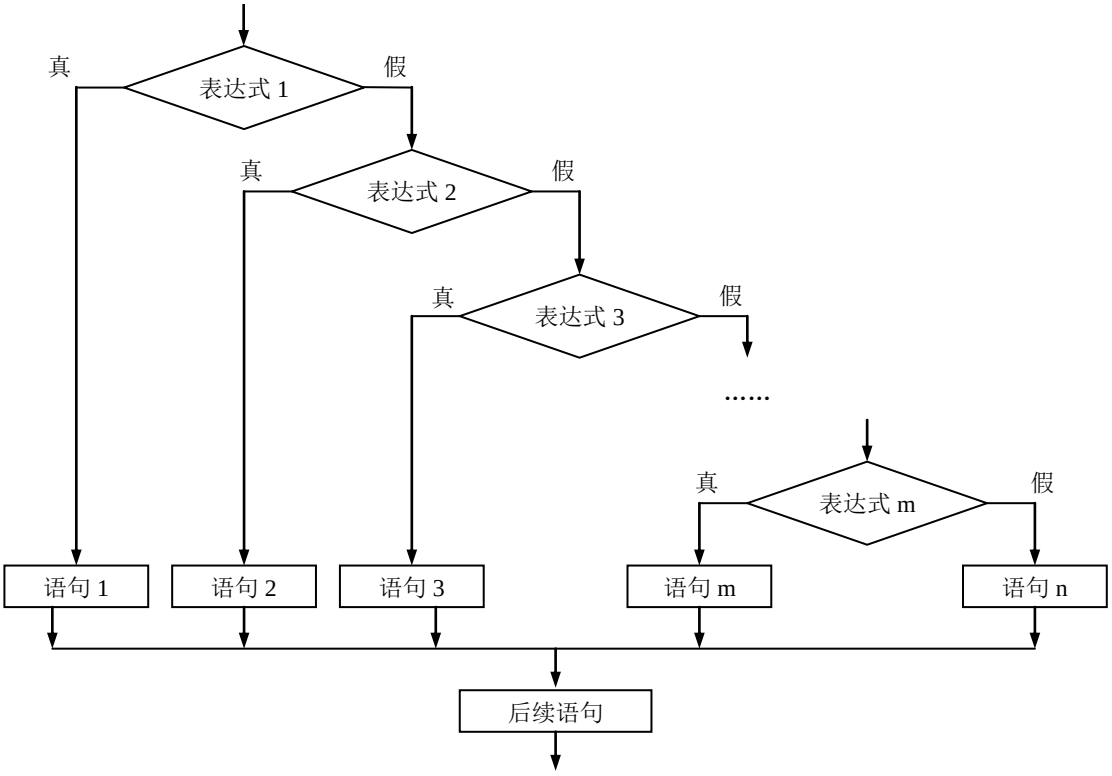


图 4.8　多分支 if 语句执行过程

2. if 语句的嵌套

当 if 语句中的执行语句又是 if 语句时，则构成了 if 语句嵌套的情形。

其一般形式可表示如下：

```
if(表达式)
    if 语句;
或者为
if(表达式)
    if 语句;
else
    if 语句;
```

在嵌套内的 if 语句可能又是 if-else 型的，这将会出现多个 if 和多个 else 重叠的情况，这时要特别注意 if 和 else 的配对问题。

例如，对于上面第一种形式，会出现以下一种情形：

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

其中的 else 究竟是与哪一个 if 配对呢?

为了避免这种二义性，C 语言规定，else 总是与它前面最近的没有配对的 if 配对。

使用多分支 if 语句，可将任务 4.3 中百分制转五级记分制

```
if(score>=90 )   grade='A';    //语句②
else   if(score>=80)   grade='B';
```

```
else    if(score>=70) grade='C';
          else if(score>=60) grade='D';
               else grade='E';
```

这里采用嵌套 if 语句描述的百分制转五级记分制代码②与采用单分支 if 语句描述的代码①相比，简化了条件的判断表达。

4.3.5 条件运算符与条件表达式

条件运算符“？：”是 C 语言唯一的三目运算符，由条件运算符构造的条件表达式的一般格式为：

表达式 1？表达式 2：表达式 3

其运算规则为：先计算表达式 1 的值，若不为 0，则计算表达式 2 的值（不计算表达式 3 的值），并将表达式 2 的值作为整个条件表达式的值；否则计表达式 3 的值（不计算表达式 2 的值），并将表达式 3 的值作为整个条件表达式的值。例如，下列条件表达式的值为 a、b 中较大的一个数值：

```
a>b?a:b
```

显然，下列 if 语句等价于：max= a>b?a:b; 。

```
if(a>b)
    max=a;
else
    max=b;
```

条件运算符的优先级较低，仅高于赋值运算符和逗号运算符。

条件表达式中的<表达式 1><表达式 2><表达式 3>可以是任意的 C 语言合法表达式，允许是又一个条件表达式。例如：下列条件表达式用于计算 a、b、c 三数中的最大值：

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

对于任务 4.3，借助条件表达式可将百分制转五级计分制的程序段简化为下列赋值语句：

```
grade=((score>=90)?'A':((score>=80)?'B':((score>=70)?'C':((score>=60)?'D':'E'))));
```

读者可自行完成借助条件表达式实现任务 4.3 的程序。

【例 4.5】输入一个字符，判别它是否小写字母，如果是，将它转换成大写字母；如果不是，不转换。然后输出最后得到的字符。

```
#include <stdio.h>
int main()
{
   char ch;

   scanf("%c",&ch);
   ch=(ch>='a' && ch<='z')?(ch-32):ch;   // 32 是小写字母和对应大写字母 ASCII 的差值

   printf("%c\n",ch);

   return 0;
}
```

程序运行结果截图如下（假设输入字符为：g）：

注意该程序与例 3.3 的区别。

4.3.6 switch 语句

switch 语句也称开关语句、多分支选择语句，其一般格式为：

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

其中，表达式是任一值为整型或字符型的合法表达式；常量表达式的值也只能是整型或字符型常量；各语句序列均是任选的，它可由一个或多个语句组成；关键字 break 也是任选的。default 分支可放在 switch 语句的任意位置，但通常作为 switch 语句的最后一个分支。

switch 语句的执行过程是，先求表达式的值，再依次与 case 后面的常量表达式比较，若与某一常量表达式的值相等，则从该 case 开始执行；否则，若有 default 分支，则从 default 分支开始执行，否则什么也不执行。switch 语句一旦从某个分支开始执行，则一直执行到 break 语句或 switch 语句的右花括号为止，而不是仅局限于执行某个分支。

使用 switch 语句，可将任务 4.3 中百分制转五级计分制的程序段描述如下：

```
int n=score/10;          //取 score 的十位，若 score 为 100，视十位为 10
switch(n)
{
    case 10:
    case 9:grade='A';break;          //L1
    case 8:grade='B';break;          //L2
    case 7:grade='C';break;          //L3
    case 6:grade='D';break;          //L4
    default:grade='E';               //L5
}
```

当输入的成绩 score 介于 100~90 时，上面程序段中的整型变量 n 可取值 10 或 9，均从 L1 行开始执行，遇 break 语句停止，所以 grade 值均为“A”；

当输入的成绩 score 介于 89~80 时，n 取值 8，只能从 L2 行开始执行，遇 break 语句停止，所以 grade 值为“B”；

当输入的成绩 score 介于 79~70、69~60 时，n 取值 7、6，分别从 L3、L4 行开始执行，遇 break 语句停止，grade 值分别为“C”和“D”；

当输入的成绩 score 介于 59~0 时，n 取值 5、4、3、2、1、0，均从 L7 行开始执行，遇 switch 语句的右花括号停止，grade 值均为“E”。

4.3.7 完成任务 4.3 的参考程序

（1）使用多分支 if 语句完成任务 4.3

参考程序：

```
#include<stdio.h>
int main( )
{
    int score;
    char grade;

    printf("输入一个成绩:");
    scanf("%d",&score);

    if(score>100||score<0)
    {
        printf("输入的百分制成绩无效\n");
        return 1;
    }

    if(score>=90 ) grade='A';
    else if(score>=80) grade='B';
        else if(score>=70) grade='C';
            else if(score>=60) grade='D';
                else grade='E';

    printf("%d 分，等级为%c\n",score,grade);

    return 0;
}
```

程序运行结果截图如下（假设输入成绩是：86）：

再次运行程序，输入成绩为 101，程序运行结果截图如下：

（2）使用条件表达式完成任务 4.3

参考程序：

```
#include<stdio.h>
int main( )
{
    int score;
```

```
    char grade;

    printf("请输入一个成绩:");
    scanf("%d",&score);

    if(score>100||score<0)
    {
        printf("输入的百分制成绩无效\n");
        return 1;
    }

    grade=((score>=90)?'A':((score>=80)?'B':((score>=70)?'C':((score>=60)?'D':'E'))));

    printf("%d 分，等级为%c\n",score,grade);

    return 0;
}
```

（3）使用 switch 语句完成任务 4.3

参考程序：

```
#include<stdio.h>
int main( )
{
    int score,n;
    char grade;

    printf("请输入一个成绩:");
    scanf("%d",&score);

    if(score>100||score<0)
    {
        printf("输入的百分制成绩无效\n");
        return 1;
    }

    n=score/10;
    switch(n)
    {
    case 10:
    case 9:grade='A';break;
    case 8:grade='B';break;
    case 7:grade='C';break;
    case 6:grade='D';break;
    default:grade='E';
    }
```

```
    printf("%d 分，等级为%c\n",score,grade);

    return 0;
}
```

4.3.8 分支结构程序设计举例

【例 4.6】编程计算下列分段函数：

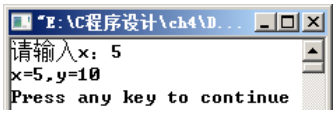
$$y=\begin{cases} -x & (x<0) \\ 0 & (x=0) \\ 2x & (x>0) \end{cases}$$

分析：分段函数的计算适合用 if 语句或条件表达式来实现。

参考程序：

```
#include<stdio.h>
int main()
{
    int x,y;
    printf("请输入 x：");
    scanf("%d",&x);

    if(x<0) y=-x;
    else if (x==0) y=0;
    else y=2*x;

    printf("x=%d,y=%d\n",x,y);

    return 0;
}
```

程序运行结果截图（假设输入的 x 是 5）如下：



思考：如何用条件表达式简化上述程序中的 if 语句？

【例 4.7】从键盘输入年份 year（4 位十进制数），判断其是否闰年并输出结果信息。

分析：

（1）判断闰年的条件是：能被 4 整除、但不能被 100 整除，或者能被 400 整除。用表达式可描述为：（year%4==0&&year%100!=0）||(year%400==0)；

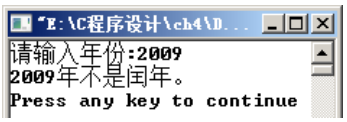
（2）定义一个变量 leap 存放是否闰年的信息，置初值为 0，表示非闰年。判断为闰年时，令 leap=1；

（3）根据 leap 的取值，输出是否闰年的结果信息。

参考程序：

```
#include <stdio.h>
int main()
{
    int year,leap=0;
    printf("请输入年份:");
    scanf("%d",&year);
    if((year%4==0 && year%100!=0)||(year%400==0))
        leap=1;
    if (leap)
        printf("%d 年是闰年。\n",year);
    else
        printf("%d 年不是闰年。\n",year);
    return 0;
}
```

程序的两次运行结果截图（输入年份分别为：2009 2012）如下：



【例 4.8】求一元二次方程 $a^2+bx+c=0$ 的根（设 a≠0）。

分析：

此问题的解决思路很明显，需要处理以下三种情况：

（1）$b^2-4ac>0$，方程有两个不相等的实根；

（2）$b^2-4ac=0$，方程有两个相等的实根；

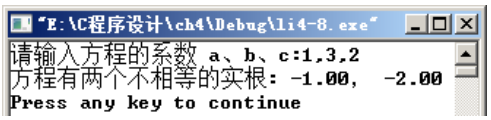
（3）$b^2-4ac<0$，方程有两个共轭复根，复根的形式为 $p+qi$ 和 $p-qi$，其中：实部 $p=-b/2a$，虚部 $q=\sqrt{-(b^2-4ac)}/2a$。

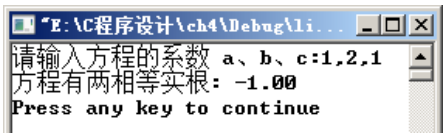
参考程序：

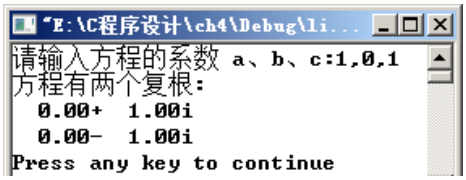
```
#include <stdio.h>
#include <math.h>
int main()
{
    double a,b,c,delta,x1,x2,realpart,imagpart;
    printf("请输入方程的系数 a、b、c:");
    scanf("%lf,%lf,%lf",&a,&b,&c);
    printf("方程有");
    delta=b*b-4*a*c;          //先计算，减少后面的重复计算
    if(delta>1e-6)
    {
        x1=(-b+sqrt(delta))/(2*a);
        x2=(-b-sqrt(delta))/(2*a);
        printf("两个不相等的实根:%6.2f，%6.2f\n",x1,x2);
    }
    else if(fabs(delta)<=1e-6) printf("两相等实根:%6.2f\n",-b/(2*a));      //A
    else
```

```
    {
        realpart=-b/(2*a);
        imagpart=sqrt(-delta)/(2*a);
        printf("两个复根:\n");
        printf("%6.2f+%6.2fi\n",realpart,imagpart);
        printf("%6.2f-%6.2fi\n",realpart,imagpart);
    }
    return 0;
}
```

运行程序三次，输入不同的系数，结果截图分别如下：







说明：

（1）程序中多次使用到求算术平方根的 sqrt()函数，该系统函数包含在“math.h”头文件中，所以程序一开始添加了“#include <math.h>”；

（2）因计算机中实数的计算和存储会有微小误差，在判断 delta 是否等于 0 时如果用 delta==0 做判定条件，可能导致 delta 本来为 0，却不满足 delta==0 条件，为此，我们用 fabs(delta)<=1e-6 作为判定 delta 为 0 的条件，见程序中的 A 行。

【例 4.9】已知某公司员工的基本工资为 800 元，绩效工资按每月工程利润(profit)提成，利润提成的比例(d)如下，编程计算员工工资。

profit <1000	没有提成
1000 <= profit < 2000	提成 10%
2000 <= profit < 5000	提成 15%
5000 <= profit < 10000	提成 20%
profit >= 10000	提成 25%

分析：

嵌套的 if 语句可以解决此问题。细心观察分析发现，利润提成比例 d 的变化点（如：1000、2000、5000 和 10000）都是 1000 的整数倍，所以利润整除 1000，其商一定是整数，可用作 switch 语句中常量表达式的取值。利润、利润/1000 及利润提成比例关系如下：

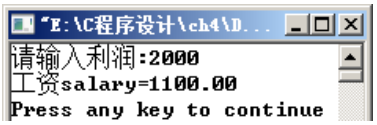
利润（profit）	利润/1000(profit/1000)	利润提成比例(d%)
profit <1000	0	0
1000 <= profit < 2000	1	10
2000 <= profit < 5000	2、3、4	15

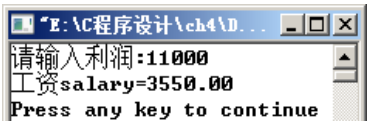
5000 <= profit < 10000	5、6、7、8、9	20
profit >= 10000	10、11、12、…	25

参考程序：

```
#include <stdio.h>
int main()
{
    long profit;                    // profit 存放利润，值为整数元
    int d;
    float salary=800 ;              //假设基本工资为 800

    printf ( "请输入利润:") ;
    scanf ("%ld",&profit ) ;

    if(profit<0)                    //利润不能为负
    {
        printf("输入的利润无效\n");
        return 1;
    }

    switch(profit/1000)
    {
    case 0 :d=0 ;break;
    case 1 :d=10;break;
    case 2 :
    case 3 :
    case 4 :d=15;break;
    case 5 :
    case 6 :
    case 7 :
    case 8 :
    case 9 :d=20;break;
    default:d=25;                   // 只要利润不低于 10000，利润提成率都为 25%
    }
    salary=salary+profit*d/100.0 ;

    printf("工资 salary=%.2f\n",salary);

    return 0;
}
```

运行程序两次，结果截图分别如下（分别输入利润值 2000 和 11000）：





4.4 循环语句

【任务 4.4】求 100 以内所有自然数的累加和

$$\sum_{n=1}^{100} n$$

任务描述：求 1+2+3+…+100，即 $\sum_{n=1}^{100} n$ 。

任务分析：根据任务描述，要求 100 以内所有自然数的累加和。首先需要定义一个存放累加和的变量 sum，并设初值为 0；若用顺序结构一个一个进行加法运算，需要百条相似的语句，如 sum=sum+1; sum=sum+2; ……sum=sum+100;，程序的结构性差。对于重复执行的某些操作，C 语言提供了循环语句来完成。编程者只需要使用较少的语句，而让计算机完成重复的、类似的工作，这样的程序结构将非常清晰。

循环结构是程序中一种很重要的结构。其特点是，在给定条件成立时，反复执行某程序段，直到条件不成立为止。给定的条件称为循环条件，反复执行的程序段称为循环体。

C 语言提供了多种循环语句，可以组成各种不同形式的循环结构。这些语句包括：

- for 语句；
- while 语句；
- do-while 语句。

4.4.1 for 语句

for 语句的一般格式为：

for(表达式 1；表达式 2；表达式 3)

　　语句

其中：表达式 1 用于初始化循环体中涉及的变量；表达式 2 用于判断循环是否结束，多为关系表达式或逻辑表达式；表达式 3 用于改变循环结束条件；语句可以是单条语句或复合语句（用花括号括{}起来的多条单语句），为 for 语句的循环体。

for 语句的执行过程如图 4.9 所示：

（1）计算表达式 1。

（2）计算表达式 2，若其值为真（非 0），则执行循环体，循环体执行结束后执行第(3)步；若其值为假（0），则跳过（3）（4）执行第(5)步。

（3）计算表达式 3。

（4）转回上面第(2)步继续执行。

（5）循环结束，执行 for 语句下面的一条语句。

for 语句中循环体有可能一次也不执行。

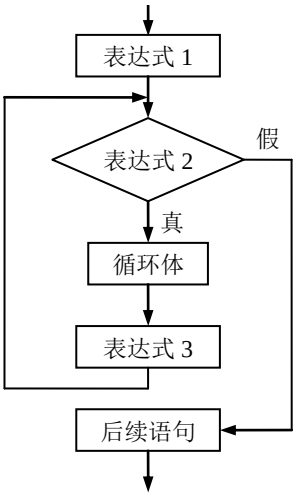


图 4.9　for 循环的执行过程

for 语句很好地体现了正确表达循环结构应当注意的 3 个问题：

- 循环控制变量的初始化；
- 循环的条件；
- 循环控制变量值的更新。

因此，for 语句也可以简化为：

for(循环变量赋初值；循环条件表达式；循环变量值更新表达式) 语句

注意：

for 语句的三个表达式可分别省略和同时省略，但括号里的两个分号不可缺少。当表达式 1 省略时，表示 for 语句本身不提供循环变量初始化，循环变量的初始化在 for 语句之前进行；表达式 3 省略时，表示 for 语句没有为下一次循环做准备，该工作应该在循环体内给出；表达式 2 省略时，相当于表达式 2 的值为 1，此时可能形成死循环，应该在循环体内增加终止循环的语句（如 break 语句，参 4.4.5 节）

对于任务 4.4，需要定义一个变量 n，通过自加 1 运算，使 n 取遍 100 以内的各自然数，通过 s=s+n 运算，将每个不同的 n 值加到 s 中；最后输出 s 的值。程序代码主体如下：

```
sum=0;
for( n=1; n<=100; n++ )          //n 为循环控制变量
    sum =sum+n;
```

也可以将语句 sum=0 放入 for 语句中：

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

4.4.2　while 语句

while 语句的一般格式为：

while(表达式) 语句

其中：表达式可以是任一合法 C 语言表达式，通常是关系表达式或逻辑表达式，用于表示循环条件；语句可以是任一单条语句或复合语句，是 while 语句的循环体。

while 语句的执行过程如图 4.10 所示：

计算表达式，若其值非 0，则执行循环体，反复以上过程，直到表达式的值为 0 为止。

while 语句中循环体有可能一次也不执行。

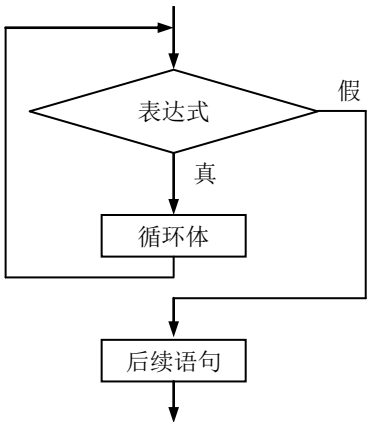


图 4.10　while 语句执行过程

任务 4.4 的程序代码主体，用 while 语句可以描述为：

```
sum=0,n=1;
while (n<=100)
{
    sum=sum+n;
    n++;
}
```

可见，相对于 for 语句的简化格式：

for(循环变量赋初值；循环条件表达式；循环变量值更新表达式) 语句

while 语句可理解为：

```
循环变量赋初值;
while(循环条件表达式)
{
    语句
    循环变量值更新表达式;
}
```

4.4.3　do…while 语句

由于 for 语句和 while 语句需要根据循环条件表达式的计算结果决定循环体的执行与否，所以存在循环体一次也不执行的可能性。下面将介绍的 do…while 语句，因先执行循环体，后判断循环条件，所以至少要执行一次循环体。

do…while 语句的一般格式为：

```
do
   语句
while（表达式）;          //分号不能缺少
```

其中：语句可以是任一单条语句或复合语句，是 do…while 语句的循环体；表达式可以是任一合法 C 语言表达式，通常是关系表达式或逻辑表达式，用于表示循环条件。

do…while 语句的执行过程如图 4.11 所示：

先执行循环体，再计算表达式；若表达式的值不为 0，则继续执行循环体，直到表达式的值为 0 为止。

do…while 语句中循环体至少执行一次。

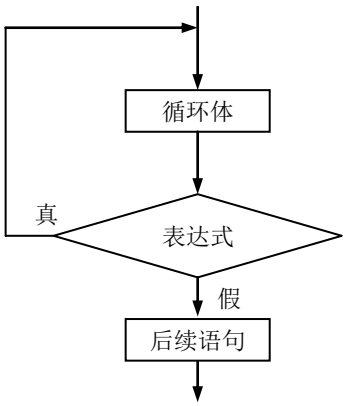


图 4.11　do-while 语句执行流程

任务 4.4 的程序代码主体，用 do…while 语句可以描述为：

```
sum=0,n=1;
do
{
    sum=sum+n;
    n++;                //为了程序简洁，省略大括号，也可用逗号表达式语句：sum+=n, n++;
} while(n<=100);
```

4.4.4　几种循环的比较

一般情况下，3 种循环可以互相代替，如上面的任务 4.4 既可以使用 for 循环又可以使用 while 循环和 do…while 循环。使用中还应注意：

（1）在 while 和 do…while 循环中，循环体应包含使循环趋于结束的语句。

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

（3）一般地，对于循环次数明确的问题应考虑用 for 语句实现，而对于次数不明确的循环问题可考虑用 while 语句实现。事实上，for 语句格式更灵活、功能更强大，适合循环次数明确或不明确的场合。

（4）3 种循环结构都可用 break 语句跳出循环，用 continue 语句结束本次循环。

思考：编辑运行下列程序各一次，输入 11，比较程序的输出结果。

```
#include <stdio.h>
int main()
{
   int i,sum=0;

   printf("i=");
   scanf("%d",&i);

   while(i<=10)
   {
      sum=sum+i;
      i++;
   }

   printf("sum=%d\n",sum);

   return 0;
}
```

```
#include <stdio.h>
int main()
{
   int i,sum=0;

   printf("i=");
   scanf("%d",&i);

   do
   {
      sum=sum+i;
      i++;
   }while(i<=10);

   printf("sum=%d\n",sum);

   return 0;
}
```

4.4.5 break 和 continue 语句

1. break 语句

break 语句不仅可以用在分支结构 switch（参见 4.3.6 节）中，也可以用在 3 个循环结构中。当 break 语句出现在 for、while 或 do…while 循环语句中时，可使程序终止循环体的执行，转去执行循环语句后面的语句，常与 if 语句连用。

【例 4.10】已知 $s=1+3+5+7+\ldots+n$，求 100 以内使 s 能被 81 整除的最小值 n。

参考程序：

```
#include <stdio.h>
int main()
{
    int i,s=0;

    for(i=1;i<100;i+=2)
    {
        s+=i;
        if(s%81==0) break;
    }
    if(i<100)
        printf("100 以内使 s 值能被 81 整除的最小值 n 为：%d\n",i);
    else
        printf("100 以内的奇数和不能被 81 整除\n");
```

```
    return 0;
}
```

下列两种情况，均可以使该程序中 for 语句的循环体终止执行：

①循环体中 s%81= =0 条件成立时，执行 break 语句终止循环；

②当循环变量 i 取值 101 时，不能使循环条件 i<100 成立，循环体终止执行。

情况一说明前面若干奇数的和能被 81 整除，可以输出满足题意的 n 值 i；情况二说明 100 以内不存在满足题意要求的 n 值。

程序的运行结果截图如下：

2. continue 语句

continue 语句只能用在循环语句的循环体中，作用是结束本次循环的循环体，提前进入下一次循环。对于 while 和 do…while 循环来说，若遇到 continue，则跳到该循环的表达式的位置；而对于 for 循环来说，则跳到该循环的表达式 3 处，而不是表达式 2 处。

【例 4.11】编程求 1~50 范围内能被 7 整除的所有整数。

参考程序：

```
#include <stdio.h>
int main()
{
    int x;

    printf("1~50 范围内能被 7 整除的整数有：\n");
    for(x=1;x<=50;x++)
    {
        if(x%7)                    //等价于 x%7!=0
            continue;              //语句①
        printf("%3d",x);       //语句②
    }
    printf("\n");

    return 0;
}
```

程序的语句①若能执行到，说明 x%7!=0 条件成立，即 x 不能被 7 整除，此时，循环体内的语句②不执行，提前进入下一次循环，即先 x++;再根据 x<=50 的值决定是否再次执行循环体。

程序的运行结果截图如下：

4.4.6 完成任务 4.4 的参考程序

（1）使用 for 语句完成任务 4.4

参考程序：

```
#include <stdio.h>
int main()
{
 int sum,n;

 sum=0;
 for(n=1;n<=100;n++)
        sum=sum+n;

 printf("1+2+3+...+100=%d\n",sum);

 return 0;
}
```

（2）使用 while 语句完成任务 4.4

参考程序：

```
#include <stdio.h>
int main()
{
 int sum,n;

 sum=0,n=1;
 while(n<=100)
 {
        sum=sum+n;
        n++;
 }

 printf("1+2+3+...+100=%d\n",sum);

 return 0;
}
```

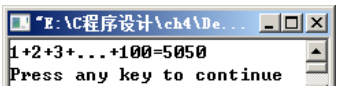
（3）使用 do…while 语句完成任务 4.4

参考程序：

```
#include <stdio.h>
int main()
{
 int sum,n;

 sum=0,n=1;
 do
 {
 sum=sum+n;
```

```
    n++;
    }while(n<=100);

    printf("1+2+3+...+100=%d\n",sum);

    return 0;
}
```

3 个参考程序的运行结果截图都是：



思考：如何修改上面的程序，完成下列计算：

$s=1\times2\times3\times4\times\ldots\times10$

$s=2+4+6+8+\ldots+100$

$s=101+102+103+104+\ldots+1000$

$s=1^2+2^2+3^2+4^2+\ldots+100^2$

【任务 4.5】打印图形

任务描述：根据输入的 n 的值，打印不同的图形（如下图 n=3 和 n=4 的情况）。

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

（n=3）　　（n=4）

任务分析：对于字符图形的打印输出问题，可以分析找出规律：每行中的打印操作是类似的，区别在于空格和星号的个数，然后重复每一行的操作。因而，每行中的打印操作可以用一个循环实现，然后再将这个循环操作作为循环体，循环执行 n 次（即 n 行）。

C 语言中可以在一个循环体内又完整地包含另一个循环，这称为循环的嵌套。前面介绍的几种类型的循环可以互相嵌套。例如可以在一个 for 循环中包含一个 while 循环，也可以在一个 while 循环中包含一个 for 循环。本节通过几个例子来说明循环嵌套的概念和使用。

4.4.7　循环的嵌套

首先介绍 for 循环语句的嵌套。

在一个 for 循环体内又完整地包含另一个 for 循环，称为 for 循环的嵌套。

如：

```
#include <stdio.h>
int main()
{
   int i,j;
   for( i=1; i<5; i++)
     {
        printf("%d", i);
        for( j=1; j<3; j++)
          printf("%d", j);
        printf("\n");
     }
   return 0;
}
```

外循环

内循环

外循环的循环体(复合语句)

内循环的循环体

上面这段程序的运行结果截图如下：

从运行结果可以看出二重嵌套循环的执行过程是：

当 i=1 时，执行外循环的循环体，包括：打印 i=1，接着执行内循环，打印 j=1、j=2。再执行换行操作后，i 变为 2，再次执行外循环的循环体。

3 种循环语句 for、while、do...while 可以互相嵌套自由组合。但是在使用循环嵌套时，应注意以下几个问题：

（1）在嵌套的各层循环中，应使用复合语句(即用一对大括号将循环体语句括起来)保证逻辑上的正确性；

（2）内层和外层循环控制变量不应同名，以免造成混乱；

（3）嵌套循环应采用右缩进格式书写，以保证层次的清晰性；

（4）循环嵌套不能交叉，即在一个循环体内必须完整地包含另一个循环。嵌套循环执行时，先由外层循环进入内层循环，并在内层循环终止之后接着执行外层循环，再由外层循环进入内层循环中，当外层循环全部终止时，程序结束。

【例 4.12】输出以下 4×5 的矩阵。

```
1   2   3   4   5
2   4   6   8   10
3   6   9   12  15
4   8   12  16  20
```

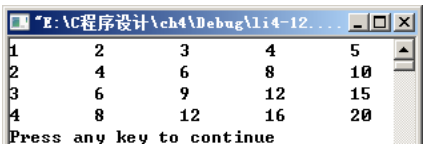
分析：此问题适合用循环的嵌套来处理，即用外循环来处理一行数据，用内循环来处理一列数据，为达到矩阵排列的效果，每输出 5 个数据就要输出一个换行符号。

参考程序：

```
#include <stdio.h>
int main()
{
    int i,j,n=0;              //i 存放行号,j 存放列号,n 存放已输出数据的个数
```

```
    for(i=1;i<=4;i++)                    //外循环
        for(j=1;j<=5;j++,n++)            //内循环
        {
            if(n&&n%5==0)
                printf ("\n");           //每输出 5 个数据就输出一个换行符号
            printf ("%d\t",i*j);         //输出数据，值为其行号和列号的乘积
        }
        printf("\n");

    return 0;
}
```

程序的运行结果截图如下：



思考：程序增加语句①后，运行结果如何？注意，当 break 出现在嵌套的循环结构中时，break 语句终止的是包含该 break 语句的最内层的循环，即只向外跳一层。若将语句①中的 break 换成 continue，运行结果又如何？

```
#include <stdio.h>
int main()
{
    int i,j,n=0;
    for(i=1;i<=4;i++)
        for(j=1;j<=5;j++,n++)
        {
            if(n&&n%5==0)
                printf ("\n");
            if (i==3 && j==1)
                break;              //语句①
            printf ("%d\t",i*j);
        }
        printf("\n");

    return 0;
}
```

4.4.8 完成任务 4.5 的参考程序

编程思路：

字符图形的打印输出问题需要考虑：

（1）图形共有多少行，用一个外循环来控制；

（2）每行的图形元素的起始位置，用输出空格来控制；

（3）每行有多少个图形元素，用一个内循环来控制。

字符图形打印的一般形式如下：

```
for(i=1;  ①  ; i++)
{
      for(j=1;  ②  ; j++)
          printf(" ") ;
      for(  ③  ;  ④  ;  ⑤  )
          printf("%c",  ⑥  ) ;
      printf("\n") ;
}
```

①控制图形行数。

②打印空格，控制每行中图形元素的起始位置。

③、④、⑤控制每行图形元素的多少。

⑥打印组成图形的字符。有一些复杂的图形在这里还要加上条件语句，在后面讲解。

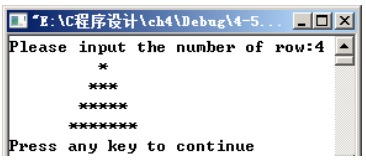
本例中每行“*”个数的变化规律总结如下：

行	每行中“*”个数与所在行的关系
1	2*1-1
2	2*2-1
3	2*3-1
4	2*4-1

参考程序：

```
#include <stdio.h>

int main()
{
    int i,j,k,n;

    printf("Please input the number of row:");
    scanf("%d",&n);

    for(i=1; i<=n;i++)
    {
        for(j=1;j<=10-i;j++)
            printf(" ") ;
        for( k=1;k<=2*i-1;k++)
            printf("*") ;
        printf("\n") ;
    }

    return 0;
}
```

程序的运行结果截图如下（假设输入为：4）：



4.4.9 循环结构程序设计举例

【例 4.13】输入一个大于 3 的整数 n，判定它是否是素数。

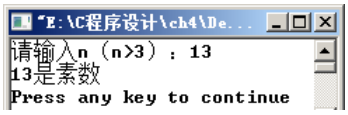
分析：

素数是只能被 1 或本身整除的数，若能判断 n 没有大于 2 和小于自身的因子，则可判定 n 是素质；若 n 能被 2~（n-1）之间某个整数 i 整除，则可判定 n 不是素数，不必继续被其它整数整除的判断，提前结束循环，此时，i 的值必定小于 n。

参考程序：

```
#include <stdio.h>
int main()
{
    int n,i;
    printf("请输入 n（n>3）：");
    scanf("%d",&n);
    for (i=2;i<=n-1;i++)              //语句①
        if(n%i==0)
            break;
    if(i<n)                           //语句②
        printf("%d 不是素数\n",n);
    else
        printf("%d 是素数\n",n);

    return 0;
}
```

程序的运行结果截图如下：



思考：判断一个整数 n 是否为素数时，是否一定需要判断 n 不能被 2~n-1 之间的任何一个整数整除？

【例 4.14】编写程序，利用以下公式计算 π 的近似值，直到最后一项的绝对值小于 10^{-8}。

$$\frac{\pi}{4}=1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\ldots$$

分析：

（1）这是一个求和问题，可以用循环语句解决，循环的结束条件是最后一项的绝对值小于 10^{-8}；

（2）参与求和的数据项正负符号间隔，且每一项的分母比前一项增加 2；

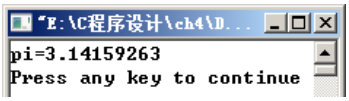
（3）最后一项的绝对值小于 10^{-8}，则有效位数超过 7 位，应使用 double 型。

参考程序：

```
#include <stdio.h>
```

```
#include <math.h>
int main()
{
    int s=1;                    //存放有符号的分子
    double pi=0,n=1,t=1; //pi 存放/4 前面项的和值,n 存放分母,t 存放当前项的值

    while(fabs(t)>=1e-8) //fabs()函数用于求绝对值，包含在 math.h 中
    {
        pi=pi+t;
        n=n+2;
        s=-s;
        t=s/n;
    }
    pi=pi*4;

    printf("pi=%10.8f\n",pi);

    return 0;
}
```

程序的运行结果截图如下:



【例 4.15】输入一行字符串，分别统计数字字符、字母和其他字符的个数。

分析：

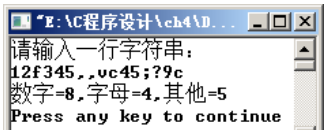
分别用三个变量作为三类字符的计数器。使用循环语句反复接受输入的字符，每接受一次字符，就判断它属性哪一类，同时并将该类的计数器加一。还需要注意的是从键盘上输入的一行字符串是以字符'\n'结尾的。

参考程序：

```
#include <stdio.h>
int main()
{
    int    digit=0,letter=0,other=0;
    char c;

    printf("请输入一行字符串：\n");
    do
    {
        scanf("%c",&c);     //从输入行中提取一个字符到变量 c 中
        if(c>='0'&&c<='9')
            digit++;
        else if(c>='a'&&c<='z'||c>='A'&&c<='Z')
            letter++;
```

```
        else other++;
    }while(c!='\n');

    printf("数字=%d,字母=%d,其他=%d\n",digit,letter,other);

    return 0;
}
```

程序的一次运行结果截图如下：



运行结果分析：输入的字符串中只有 4 个其他字符，但统计结果却是‘其他=5’。这是因为其他字符的个数统计中包含了行结束字符’\n’。

思考：如何修改程序，使得其他字符个数的统计不包括’\n’。

【例 4.16】编写程序输出 1000 以内的的所有完数。所谓“完数”是指与其因子之和相等的数。例如 6，因为 6=1+2+3，且 1、2、3 均是 6 的因子。

分析：

（1）用变量 s 保存一个数的因子和，因为 1 是任何一个数的因子，可直接作为 s 的初值；

（2）用变量 i 表示 1000 以内的一个数，初值为 2；

（3）用变量 j 表示 i 的试探因子，初值为 2，终值为 i/2，若有 i%j==0 成立，则说明 j 是 i 的因子，则 sum+=j。

参考程序：

```
#include <stdio.h>
int main()
{
    int i,j,sum;

    for(i=2;i<=1000;i++)
    {
        for(sum=1,j=2;j<=i/2;j++)        //语句①: 计算 i 的因子和
            if(i%j==0)
                sum+=j;
        if(sum==i)                       //若 i 是完数
        {
            printf("%d=1",i);        //输出 i=1,1 是完数 i 的第一个因子
            for(j=2;j<=i/2;j++)     //因语句①没保存 i 的因子，所以输出时需再求
                if(i%j==0)
                    printf("+%d",j); //按格式输出完数的其它因子
            printf("\n");
        }
    }
```

```
    return 0;
}
```

程序的运行结果截图如下：

【例 4.17】编程计算 1!+2!+3!+…10!

分析:

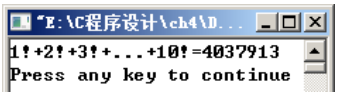
（1）若将任务 4.4 中的加法运算换成乘法运算，求累加和的问题相应就变成了求阶乘的问题；

（2）因为是连续的数的阶乘求和，所以，可以边求阶乘边求和。

参考程序：

```
#include <stdio.h>
int main()
{
    int i,p,s;

    for(i=2,p=1,s=1;i<=10;i++)
    {
        p*=i;              //求阶乘
        s+=p;              //求阶乘的和
    }

    printf("1!+2!+3!+...+10!=%d\n",s);

    return 0;
}
```

程序的运行结果截图如下：



思考：由于 n!的值按指数规律增长，当 n 较大时，应注意溢出问题的处理。即 n!值超出 int 型数据的表示范围怎么办？

常见的处理方法：

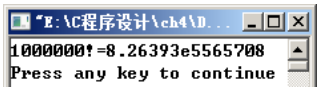
（1）使用表示范围更大的数据类型，如 double 或 long double；

（2）如果方法 1 仍然不能满足计算要求，可以采用尾数加指数的方法近似表示，即当计算出的阶乘值大于或等于 10 时，就除以 10，同时让指数加 1，最后将尾数和指数分别输出。例如，计算 1000000！的参考程序：

```
#include <stdio.h>
int main()
{
    int i,e=0;          //e 保存阶乘的指数
    double p=1;    //计算阶乘，保存尾数
```

```
    for(i=2;i<=1000000;i++)
    {
        p*=i;     //计算阶乘，保存到 p
        while(p>=10) p/=10.0,e++; //阶乘 p 反复除以 10，指数反复加 1
    }

    printf("1000000!=%g%c%d\n",p,'e',e);

    return 0;
}
```

程序的运行结果截图如下：



本章小结

本章的主要内容总结如下：

（1）C 语言程序的控制结构有顺序结构、分支结构和循环结构三种，C 语言提供相应的语句描述这些控制结构。

（2）分支语句包括 if 语句和 switch 语句，if 语句适用于两个分支的选择，嵌套后的 if 语句能描述多分支的选择，switch 语句本身就是一个多分支的选择语句。任一 switch 语句均可用 if 语句来实现，但反之不然（受常量表达式取值类型的限制）。

（3）循环语句包括 for 语句、while 语句和 do…while 语句。for 和 while 语句都是先判断循环条件，再执行循环体，所以循环体有可能执行若干次，也可能一次都不执行。而 do…while 语句是先执行循环体，后判断循环条件，所以循环体至少要执行一次，即使首次判断循环条件就不成立。三个循环语句使用频率由高到低依次为：for 语句、while 语句和 do…while 语句。

（4）实际编程时，循环语句中嵌套分支语句、分支语句中嵌套循环语句或循环语句中嵌套循环语句都很常见。

习题 4

一、选择题。

1．已有声明“int t=1;”，则执行“printf("%d",(t+5,t++));”时输出结果是（　　）。

A．1　　B．6　　C．2　　D．7

2．一元二次方程 $ax^2+bx+c=0$ 有两个相异实根的条件是 $a\neq 0$ 且 $b^2-4ac>0$，以下选项中能正确表示该条件的 C 语言表达式是（　　）。

A．a!=0,b*b-4*a*c>0　　B．a!=0||b b*b-4*a*c>0

C．a&& b*b-4*a*c>0　　D．!a&& b*b-4*a*c>0

3．已有声明“int x;”，实现“若 x 的值是奇数，则输出 x”这一功能的语句是（　　）

A．if(x/2) printf("%d",x);　　B．if(x%2) printf("%d",x);

C．if(x/2==1) printf("%d",x);　　D．if(x%2==0) printf("%d",x);

4．已有声明“int x,a=3,b=2;”，则执行赋值语句“x=a>b++?a++:b++;”后，变量 x、a、b 的值分别为（　　）。

A．3　4　3　　B．3　3　4
C．3　3　3　　D．4　3　4
5．以下关于 if 语句和 switch 语句的叙述错误的是（　　）
A．if 语句和 switch 语句都可以实现算法的选择结构
B．if 语句和 switch 语句都能实现多路（两路以上）选择
C．if 语句可以嵌套使用
D．switch 语句不能嵌套使用
6.以下 4 个程序段中有 3 个程序段的执行效果总是相同的，另一个执行效果不同的是(　　)。
A．if(a>b) c=a,a=b,b=c;　　B．if(a>b) { c=a,a=b,b=c; }
C．if(a>b){ c=a; a=b; b=c;}　　D．if(a>b) c=a; a=b; b=c;
7．有下列程序：

```
#include <stdio.h>
int main()
{    int a=15,b=21,m=0;
     switch(a%3)
     { case 0:m++;break;
       case 1:m++;
              switch(b%2)
              {   default:m++;
                  case 0:m++;break;
              }
     }
     printf("%d\n",m);
     return 0;
}
```

程序的输出结果是（　　）。
A．1　　B．2　　C．3　　D．4
8．以下关于 break 和 continue 语句的叙述正确的是（　　）。
A．break 和 continue 语句都可以出现在 switch 语句中
B．break 和 continue 语句都可以出现在循环语句的循环体中
C．在循环语句和 switch 语句之外允许出现 break 和 continue 语句
D．执行循环语句中的 break 和 continue 语句都将立即终止循环
9．以下关于 while 语句和 do…while 语句的描述错误的是（　　）
A．while 语句和 do…while 语句的循环体至少都会被执行一次
B．while 语句和 do…while 语句都可以使一段程序重复执行多遍
C．while 语句和 do…while 语句都包含了循环体
D．while 语句和 do…while 语句都包含了控制循环的表达式
10．执行以下程序后的结果是（　　）。

```
#include <stdio.h>
int main()
{    int x=3;
     do
     {
        printf("%d\t",x=x-3);
```

```
    }while(!x);
    return 0;
}
```

A．输出一个数 0　　B．输出一个数：3

C．输出 2 个数：0 和-3　　D．无限选好，反复输出数

二、填空题。

1．执行语句序列“int a,b;a=b=4;a+=b%3;”后，变量 a、b 的值分别是_____。

2．已有声明“int x=0,y=2;”，则执行语句“y=--x||++y;”后，变量 y 的值为_____。

3．若有声明“int a=0,b=1,c=2;”，执行语句“if(a>0&&++b>0)c++;else c--;”后，变量 a、b、c 的值分别是_____。

4．以下程序运行时，输出到屏幕的结果是_____。

```
#include <stdio.h>
int main()
{int a=1,b=1,d=10;
if(a)
    if(b)
        d=20;
    else
        d=30;
printf("%d\n",d);
return 0;
}
```

5．表示“当 x 取值在[1,10)范围内时 y 取值 1，否则 y 取值-1”所使用的 C 表达式为“y=_____? 1:-1”。

6．已有声明“int a=-3;”，则表达式“a>=0?a:-a”的值是_____。

7．如果一个循环结构的循环体至少要执行一遍，则最适合描述该循环结构的语句是_____。

8．除 goto 语句外，在循环结构中执行_____语句可提前结束本次循环直接进入下一次循环。

9．以下程序运行时，输出到屏幕的结果中第一行是_____，第二行是_____。

```
#include <stdio.h>
int main()
{int i,m=1,n=1;
printf("%4d%4d",m,n);
for(i=2;i<9;i++)
{
    n=n+m;
    m=n-m;
    if(i%3==0)printf("\n");
    printf("%4d",n);
}
```

return 0;

}

10．使用 VC++6.0 或者 Turbo C 系统编译 C 语言源程序后生成的文件名后缀是_____。

三、简答题。

1．什么是算法？算法的特性是什么？结构化程序设计的三种基本流程控制结构是什么？

2．用流程图表示求解下列问题的算法：

（1）输入两个数，交换后再输出。

（2）输入一个 3 位自然数，输出它的各位数之和。

（3）输出 100~200 之间能同时被 3 和 7 整除的数。