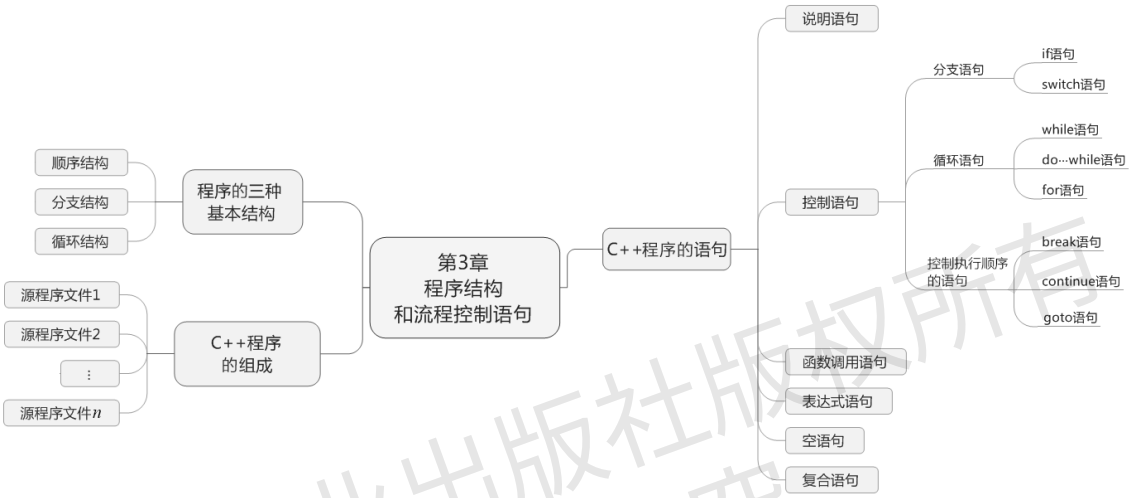
# 第3章

# 程序结构和流程控制语句




本章知识点导图

通过本章的学习，应掌握程序的三种基本结构，即顺序结构、分支结构和循环结构；掌握C++中实现这三种基本结构的控制语句的格式、功能和执行过程；能使用这些控制语句编写具有顺序、分支和循环三种基本结构的程序。



## 3.1 程序的三种基本结构和语句

### 3.1.1 程序的三种基本结构

尽管C++是面向对象的程序设计语言，但组成C++程序的函数仍是由若干基本结构组合而成的。每种基本结构可以包含一条或多条语句。程序有三种基本结构，即顺序结构、分支结构和循环结构。

#### 1. 顺序结构

按程序中语句的顺序依次执行的结构称为顺序结构，它是最简单的一种基本结构，如图 3.1 所示。在图 3.1 中，按语句的顺序，先执行 $S_1$ 操作，再执行 $S_2$ 操作。图 3.1（a）为顺序结构流程图，图 3.1（b）为其 N-S 流程图。

#### 2. 分支结构

在两种可能的操作中按一定条件选取一个执行的结构称为分支结构，如图 3.2 所示。在图 3.2 中，当条件 B 成立（真）时，执行 $S_1$ 操作，否则执行 $S_2$ 操作。图 3.2（a）为分支结构流程图，图 3.2（b）

为其 N-S 流程图。

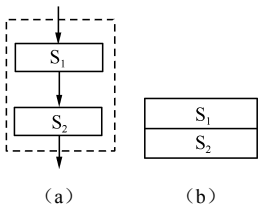



图 3.1　顺序结构

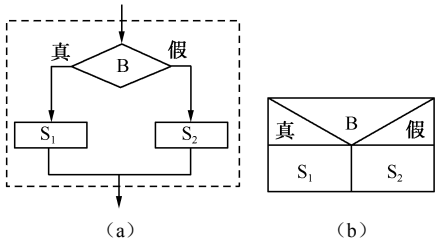



图 3.2　分支结构

由分支结构可以派生一种多分支结构，如图 3.3 所示。在图 3.3 中，依次判断条件 $B_i$（$i$=1，2，…，$n$）是否成立，当 $B_i$ 成立时，就执行相应的 $S_i$ 操作；当所有条件都不成立时，就执行 $S_{n+1}$ 操作。

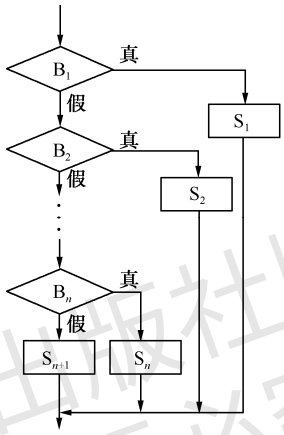



图 3.3　多分支结构

### 3．循环结构

循环结构有两种形式，即当型循环结构和直到型循环结构。

（1）当型循环结构。当某条件成立时，重复执行一个操作，直到条件不成立的结构称为当型循环结构，如图 3.4 所示。图 3.4（a）为当型循环结构流程图，图 3.4（b）为其 N-S 流程图。在图 3.4 中，当条件 B 成立（真）时，重复执行 S 操作，直到条件 B 不成立（假）时才停止执行 S 操作，转而执行其他操作。

（2）直到型循环结构。重复执行一个操作，直到某条件不成立的结构称为直到型循环结构，如图 3.5 所示。图 3.5（a）为直到型循环结构流程图，图 3.5（b）为其 N-S 流程图。在图 3.5 中，先执行 S 操作，再判断条件 B 是否成立，若条件 B 成立（真），则再次执行 S 操作，如此重复，直到条件 B 不成立（假）时停止执行 S 操作，转而执行其他操作。

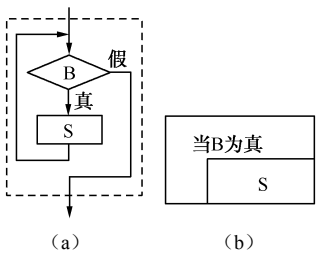



图 3.4　当型循环结构

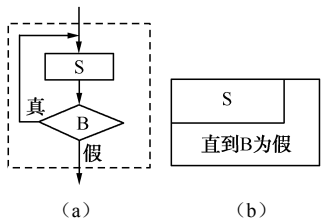



图 3.5　直到型循环结构

**说明：**在其他语言（如 Pascal）中，也有以条件 B 为真作为退出循环的条件的直到型循环结构。

三种基本结构都具有以下共同特征。

（1）单入口和单出口，即只有一个入口和一个出口。

（2）没有无用的部分，即结构中所有部分都有被执行的机会。

（3）不存在“死循环”（无终止的循环），即执行时间是有限的。

### 3.1.2 C++程序的组成

C++程序的组成如图 3.6 所示，即一个 C++程序可以由若干源程序文件组成，一个源程序文件可以由若干函数和编译预处理命令组成，一个函数由函数说明和函数体组成，函数体由变量定义和若干执行语句组成。语句是组成程序的基本单元。

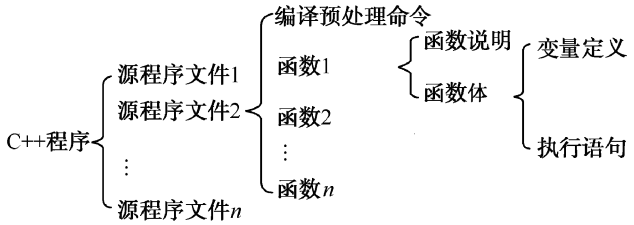



图 3.6 C++程序的组成

### 3.1.3 C++程序的语句

C++程序的语句可以分成以下六大类。

#### 1．说明语句

在 C++中，把对数据类型的定义和描述、对变量和符号常量的定义性说明统称为说明语句。例如：

```
int  a,b,c;                    //定义整型变量 a、b、c
```

说明语句在程序的执行过程中，并没有对数据进行操作，而是仅向编译系统提供一些说明性的信息。例如，定义变量语句 int a,b,c;告诉编译系统为变量 a、b、c 各分配 4B 的存储空间用于存放变量的值。在 C++中，说明语句作为语句来对待，它可以出现在函数中允许出现语句的任何位置，也可以出现在函数定义外。

#### 2．控制语句

完成一定控制功能的语句（有可能改变程序执行顺序的语句）称为控制语句。控制语句包括条件语句、开关语句、循环语句、转向语句、从函数返回语句等。例如：

```
if (x>y) z=x; else z=y;
```

该条件语句表示若 x>y，则 z=x，否则 z=y。根据 x、y 值的大小决定 z 的取值。

#### 3．函数调用语句

在一次函数调用后加一个分号构成的语句称为函数调用语句。例如，例 1.3 中的 add(a,b);，其中，add(a,b)为求 a、b 两个变量的和的函数。

#### 4．表达式语句

在一个表达式的后面加一个分号构成的语句称为表达式语句。例如，由一个赋值表达式加一个

分号构成一条赋值表达式语句：

```
y=x*x+2*x-1.5+sin(x);
```

#### 5．空语句

只有一个分号的语句称为空语句，即：

```
;
```

空语句不做任何操作，主要用于指明被转向的控制点或在特殊情况下作为循环语句的循环体。

#### 6．复合语句

用花括号把一条或多条语句括起来后构成的语句称为复合语句（语句块）。C++把复合语句作为一条语句来处理，复合语句可以出现在允许出现一条语句的任何位置。复合语句中的左花括号表明复合语句的开始，右花括号表明复合语句的结束，右花括号后不再需要分号。例如：

```
{t=a;a=b;b=t;}
```

复合语句主要用在控制语句中。

程序的三种基本结构都是通过语句来实现的。由于顺序结构比较简单，已在第 2 章中有所介绍，所以下面只介绍能实现分支结构、循环结构的分支语句和循环语句。

## 3.2　分支语句

分支语句用于实现分支结构程序设计。分支语句分为两路分支结构和多路分支结构两种，两路分支结构可用 if 语句实现，多路分支结构可用嵌套的 if 语句和 switch 语句实现。

### 3.2.1　if 语句

if 语句即条件语句，它根据给定的条件决定执行两个分支程序段中的某一个分支程序段。

#### 1．if 语句的三种形式

C++中的 if 语句有以下三种形式。

（1）单选 if 语句。单选 if 语句的格式为：

```
if (<表达式>)
    <语句>
```

单选 if 语句的执行流程为：当表达式的值为真（非 0）时，执行语句；否则不执行语句，如图 3.7 所示。图 3.7（a）为单选 if 语句流程图，图 3.7（b）为其 N-S 流程图。

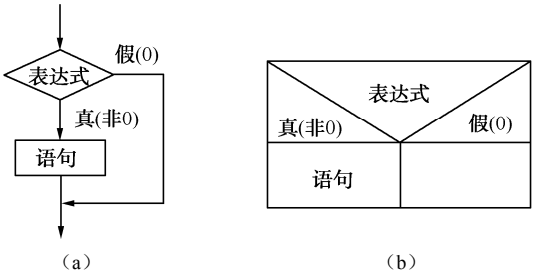



图 3.7　单选 if 语句执行流程图

**说明：**

① 表达式一般为关系表达式或逻辑表达式，但也可为其他表达式，必须用一对圆括号“()”括起来。

② 语句可以是单条语句，也可以是多条语句（此时必须用花括号“{}”将多条语句括起来，构成一条复合语句）。

**【例 3.1】** 输入两个整数 a 和 b，并输出其中较大的一个数。

求两个数中较大值的流程图如图 3.8 所示。

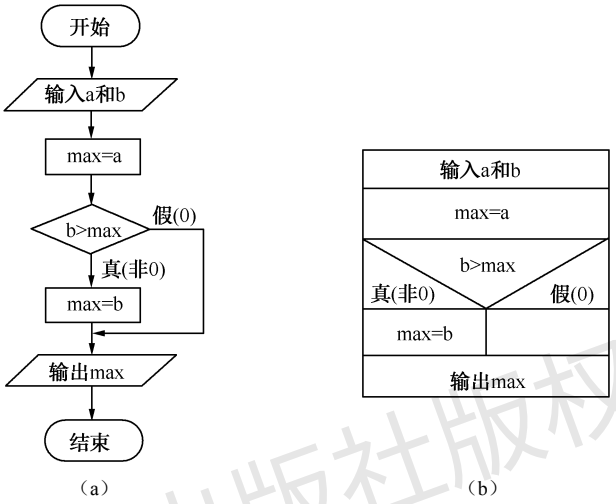



图 3.8 求两个数中较大值的流程图

程序如下：

```
#include <iostream>
using namespace std;
int main( )
{
    int a,b,max;
    cout<<"Input a,b:";
    cin>>a>>b;
    max=a;
    if (b>max)
        max=b;
    cout<<"max="<<max<<endl;
    return 0;
}
```

程序执行后显示：

```
Input a,b: 3        8
max=8
```

（2）双选 if 语句。双选 if 语句的格式为：

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

双选 if 语句的执行流程为：当表达式的值为真（非 0）时，执行语句 1；否则执行语句 2，如图 3.9 所示。图 3.9（a）为双选 if 语句流程图，图 3.9（b）为其 N-S 流程图。

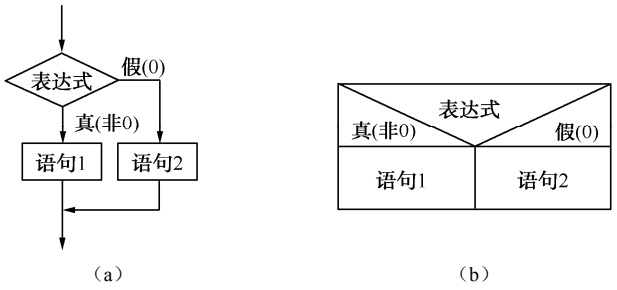



图 3.9　双选条件语句执行流程图

**【例 3.2】**　输入两个整数 a 和 b，输出其中较大的一个数。

求两个数中较大值的流程图如图 3.10 所示。

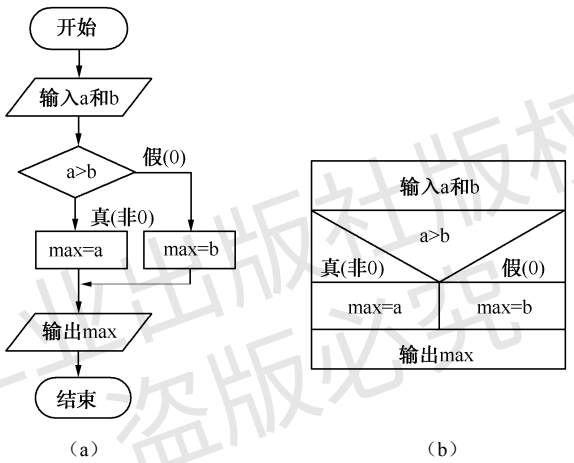



图 3.10　求两个数中较大值的流程图

程序如下：

```
#include <iostream>
using namespace std;
int main( )
{    int a,b,max;
     cout<<"Input a,b:";
     cin>>a>>b;
     if (a>b)
          max=a;
     else
        max=b;
     cout<<"max="<<max<<endl;
     return 0;
}
```

程序执行后显示：

```
Input a,b: 3          8
max=8
```

（3）多选 if 语句。多选 if 语句的格式为：

```
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 n-1>)
   <语句 n-1>
else
   <语句 n>
```

**【例 3.3】** 设有下列分段函数：

$$y=\begin{cases}x+1 & x<0\\ x^2-5 & 0\leqslant x<10\\ x^3 & x\geqslant 10\end{cases}$$


本题微课视频

编一程序，输入 $x$，并输出 $y$ 的值。

分段函数流程图如图 3.11 所示。

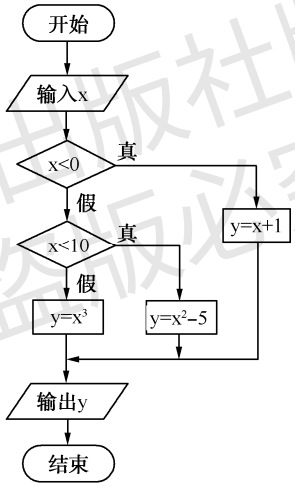



图 3.11　分段函数流程图 1

程序如下：

```
#include <iostream>
using namespace std;
int main( )
{    float x,y;
     cout<<"Input x:";
     cin>>x;
     if (x<0)
        y=x+1;
     else   if (x<10)
        y=x*x-5;
     else
        y=x*x*x;
```

```
        cout<<"y="<<y<<endl;
        return 0;
    }
```

程序执行后显示：

```
Input x: 3
y=4
```

### 2．if 语句的嵌套

一个 if 语句中包含一个或多个 if 语句的情况称为 if 语句的嵌套，其一般格式为：

```
if (<表达式 1>)
   if (<表达式 2>)
       <语句 1>
   else
       <语句 2>
else
   if (<表达式 3>)
       <语句 3>
   else
       <语句 4>
```

**【例 3.4】** 设有下列分段函数：

$$y=\begin{cases} x+1 & x<0 \\ x^2-5 & 0\leqslant x<10 \\ x^3 & x\geqslant 10 \end{cases}$$

编一程序，输入 $x$，并输出 $y$ 的值。

分段函数流程图如图 3.12 所示。

程序如下：

```
#include <iostream>
using namespace std;
int main( )
{    float x,y;
     cout<<"Input x:";
     cin>>x;
     if (x>=0)
        if (x>=10)
           y=x*x*x;
        else
           y=x*x-5;
     else
        y=x+1;
     cout<<"y="<<y<<endl;
     return 0;
}
```

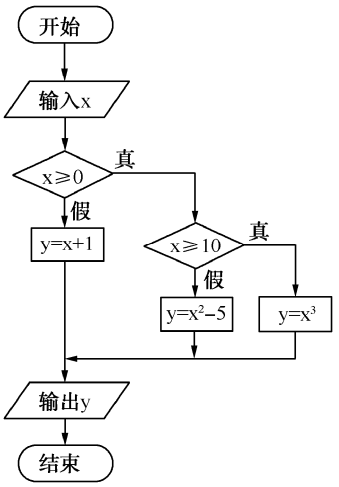



图 3.12　分段函数流程图 2

程序执行后显示：

```
Input x:-3
y=-2
```

在该程序中，内层的 if 语句嵌套在外层的 if 语句的 if 部分。

if 语句在嵌套使用时，应当注意 else 与 if 的配对关系。C++规定：else 总是与其前面最近的还没有配对的 if 进行配对。例如：

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

等价于：

```
if (<表达式 1>)
  {if (<表达式 2>)
      <语句 1>
   else
      <语句 2>
  }
```

如果要改变这种约定，则应用花括号构成复合语句。例如：

```
if (<表达式 1>)
  {if (<表达式 2>)
      <语句 1>
  }
else
    <语句 2>
```

此时，else 与第一个 if 配对。

**【例 3.5】** 求三个整数 a、b、c 中的最大者，a、b、c 由键盘输入。

求三个数中最大数的 N-S 流程图如图 3.13 所示。

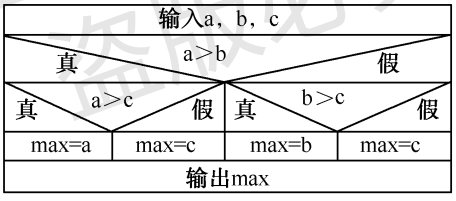



图 3.13　求三个数中最大数的 N-S 流程图

程序如下：

```
#include <iostream>
using namespace std;
int main( )
{    int a,b,c,max;
     cout<<"Input a,b,c:";
     cin>>a>>b>>c;
     if (a>b)
        if (a>c)
           max=a;
        else
           max=c;
     else
        if (b>c)
           max=b;
```

```
        else
            max=c;
    cout<<"max="<<max<<endl;
    return 0;
}
```

程序执行后显示：

```
Input a,b,c:      1      4      5
max=5
```

### 3.2.2 条件运算符和条件表达式

C++中提供了一个条件运算符“? :”，它是三目运算符（三元运算符），即要求有三个操作对象。由条件运算符构成的条件表达式的一般格式为：

```
<表达式 1>?<表达式 2>:<表达式 3>
```

条件运算符的执行过程为：先计算表达式 1 的值，如果表达式 1 的值为非 0（真），则计算表达式 2 的值，条件表达式的值就是表达式 2 的值；如果表达式 1 的值为 0（假），则计算表达式 3 的值，条件表达式的值就是表达式 3 的值。其中，表达式 1 一般为关系表达式或逻辑表达式，但也可以为其他表达式。

当 if 语句的两个分支中都执行给同一个变量赋值的赋值语句时，可以用条件表达式来代替 if 语句，因此也可以用条件表达式构成分支结构程序。

**【例 3.6】** 输入两个整数 a 和 b，输出其中较大的一个数。

程序如下：

```
#include <iostream>
using namespace std;
int main( )
{   int a,b,max;
    cout<<"Input a,b:";
    cin>>a>>b;
    max=(a>b)?a:b;
    cout<< "max="<< max<<endl;
    return 0;
}
```

程序执行后显示：

```
Input a,b:9      4
max=9
```

条件运算符的优先级高于赋值运算符和逗号运算符的优先级，低于算术运算符、关系运算符和逻辑运算符的优先级。因此，赋值表达式 m=(a>b)?a:b 中的括号可以省略，即可写成 m=a>b?a:b。

### 3.2.3 switch 语句

#### 1．switch 语句的格式及用法

if 语句允许程序在运行时按照表达式的值从两个可能的操作中选择一个来执行，但在程序要求进行多分支选择时，虽然用 if 语句可以实现，但需要多重嵌套，使用不方便，此时可以用 switch 语句来实现。

switch 语句即开关语句，它根据给定的条件决定执行多个分支程序段中的哪个分支程序段。

switch 语句的一般格式为：

```
switch (<表达式>)
{   case <常量表达式 1>:〔<语句 1>〕
    case <常量表达式 2>:〔<语句 2>〕
    …
    case <常量表达式 n-1>:〔<语句 n-1>〕
    〔default:<语句 n>〕
}
```

switch 语句的执行流程为：先计算表达式的值，然后将它与 case 后的常量表达式逐个进行比较，若与某一个常量表达式的值相等，就执行此 case 后面的语句；若都不相等，则执行 default 后面的语句，若没有 default，则不做任何操作就结束。

**【例 3.7】** 输入 0～6 的整数，将其转换成对应的星期。

程序如下：

```
#include <iostream>
using namespace std;
int main( )
{   int a;
    cout<<"Input an integer(0～6):";
    cin>>a;
    switch (a)
    {   case 0:cout<<"Sunday\n";
        case 1:cout<<"Monday\n";
        case 2:cout<<"Tuesday\n";
        case 3:cout<<"Wednesday\n";
        case 4:cout<<"Thursday\n";
        case 5:cout<<"Friday\n";
        case 6:cout<<"Saturday\n";
        default:cout<<"Input data error.\n";
    }
    return 0;
}
```

**说明：**

（1）switch 后面括号内的表达式只能是整型表达式、字符型表达式或枚举型变量。与其对应，case 后面的常量表达式也应为整型常量表达式、字符型常量表达式或枚举型数据。

（2）每个 case 后面的常量表达式的值必须互不相同。

（3）各个 case 和 default 的出现次序不影响执行结果。

（4）一个 case 后面可以包含多条语句，并且这些语句不必用花括号括起来，程序会自动顺序执行该 case 后面的所有语句，一个 case 后面也可以没有任何语句。

### 2．break 语句在 switch 语句中的作用

在执行 switch 语句的过程中，每当执行完一个 case 后面的语句后，程序就会不加判断地自动执行下一个 case 后面的语句。每个 case 后面的常量表达式只起语句标号的作用，是 switch 语句中执行各语句的入口。例如，在例 3.7 中，若在运行程序时输入 4，则执行结果为：

```
Thursday
Friday
```

```
Saturday
Input data error.
```

因此，应该在执行完一个 case 分支后，使程序跳出 switch 语句，即终止执行 switch 语句，这可以用 break 语句来实现。例 3.7 的程序应改写为：

```
#include <iostream>
using namespace std;
int main( )
{    int a;
     cout<<"Input an integer(0～6):";
     cin>>a;
     switch (a)
     {   case 0:cout<<" Sunday\n";break;
         case 1:cout<<" Monday\n";break;
         case 2:cout<<" Tuesday\n";break;
         case 3:cout<<" Wednesday\n";break;
         case 4:cout<<" Thursday\n";break;
         case 5:cout<<" Friday\n";break;
         case 6:cout<<" Saturday\n";break;
         default:cout<<"Input data error.\n";
     }
     return 0;
}
```

程序执行后显示：

```
Input an integer(0～6): 5
Friday
```

从 switch 语句的执行过程可知，任何 switch 语句均可用 if 语句来实现，并不是任何 switch 语句均可用 switch 语句来实现，这是由于 switch 语句限定了表达式的取值类型，而 if 语句中的条件表达式可取任意类型的值。

**【例 3.8】** 商店打折售货，购货金额越大，折扣越大，具体标准为（*m*：购货金额/元，*d*：折扣率）：


本题微课视频

| | |
|---|---|
| *m*<250 | *d*=0% |
| 250≤*m*<500 | *d*=5% |
| 500≤*m*<1000 | *d*=7.5% |
| 1000≤*m*<2000 | *d*=10% |
| *m*≥2000 | *d*=15% |

通过键盘输入购货金额，计算实付的金额。

**分析**：首先应找出购货金额与折扣率间对应关系的变化规律。从题意可知，当购货金额 *m* 每变化 250 元或 250 元的倍数时，折扣率就会变化。用 *c*=*m*/250 来表示折扣率的分档情况，如表 3.1 所示。

**表 3.1　商店打折售货分档情况表**

| *m*/元 | *c*=*m*/250 | *d* |
|---|---|---|
| *m*<250 | 0 | 0% |
| 250≤*m*<500 | 1 | 5% |
| 500≤*m*<1000 | 2，3 | 7.5% |
| 1000≤*m*<2000 | 4，5，6，7 | 10% |
| *m*≥2000 | 8 | 15% |

根据购货金额确定好折扣率后，再计算实付金额。

程序如下：

```
#include <iostream>
using namespace std;
int main( )
{    int m,c;
     float d,f;
     cout<<"Input m:";
     cin>>m;
     if (m>=2000)
        c=8;
     else
        c=m/250;
     switch (c)
     {   case 0:d=0;break;
         case 1:d=5;break;
         case 2:
         case 3:d=7.5;break;
         case 4:
         case 5:
         case 6:
         case 7:d=10;break;
         case 8:d=15;break;
     }
     f=m*(1-d/100.0);
     cout<<"f="<<f<<endl;
     return 0;
}
```

程序执行后显示：

```
Input m:500
f=462.5
```



## 3.3 循环语句

在程序设计中，经常会遇到在某一条件成立时，重复执行某些操作。例如，求：

$$S=1+2+3+4+\cdots+n$$

显然，在程序中不可能依次列出 1～$n$ 个数，要完成以上的累加求和运算，可设两个整型变量 sum 和 i，sum 存放累加和，i 从 1 变化到 $n$，并按下列步骤进行操作。

（1）给 sum 赋值 0，i 赋值 1。

（2）令 sum=sum+i，i=i+1。

（3）若 i≤n，则重复执行步骤（2）。

（4）输出 sum 的值。

在以上步骤中，步骤（2）和步骤（3）是需要重复执行的操作。这种重复执行的操作可由程序中的循环结构来完成。

所谓循环结构，就是在给定条件成立的情况下，重复执行一个程序段；当给定条件不成立时，退出循环，再执行循环下面的程序。实现循环结构的语句称为循环语句。在 C++中，循环语句有 while 语句、do…while 语句和 for 语句。

### 3.3.1 while 语句

#### 1. while 语句的格式

while 语句用来实现当型循环结构，其一般格式为：

```
while  (<表达式>)
  <语句>
```

**说明：**

（1）表达式称为循环条件表达式，一般为关系表达式或逻辑表达式，也可为其他表达式，必须用一对圆括号“()”括起来。

（2）语句称为循环体，可以是单条语句，也可以是多条语句（此时必须用花括号“{ }”将多条语句括起来，构成一个复合语句）。

#### 2. while 语句的执行过程

while 语句的执行过程为：先计算表达式的值，当表达式的值为真（非 0）时，重复执行指定的语句；当表达式的值为假（0）时，结束循环，如图 3.14 所示。图 3.14（a）为 while 语句的流程图，图 3.14（b）为其 N-S 流程图。

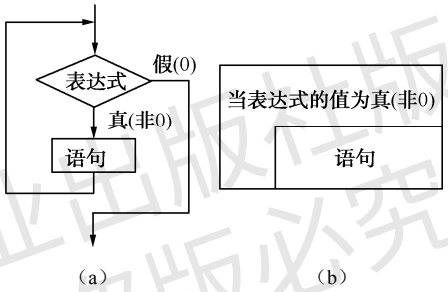



图 3.14　while 语句执行流程图

**【例 3.9】**　用 while 语句计算 $S=\sum_{i=1}^{n} i$，即求累加和 $S$=1+2+3+4+…+$n$。

用 while 语句求累加和的流程图如图 3.15 所示，程序如下：

```
#include <iostream>
using namespace std;
int main( )
{    int i,n,sum;
     cout<<"Input an integer:";
     cin>>n;
     sum=0;
     i=1;
     while (i<=n)
     {   sum=sum+i;
         i++;
     }
     cout<<"sum="<<sum<<endl;
     return 0;
}
```

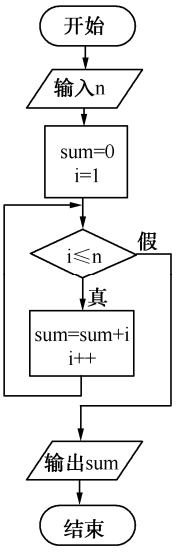



图 3.15　用 while 语句求累加和的流程图

程序执行后显示：

```
Input an integer:5
sum=15
```

**说明：**

（1）while 语句是先判断表达式 i<=n 是否成立，若条件成立，则将 sum 加 i 后赋给 sum，且 i 增加 1；若条件不成立，则不执行相应语句，退出循环。

（2）若表达式的值一开始就不成立，则语句一次也不执行。例如，当输入 n 为 0 时，i<=n 不成立，语句 sum=sum+i;和 i++;一次也不执行。

（3）在循环体中应有不断修改循环条件并最终使循环结束的语句，否则会形成死循环。例如，i++;语句，使 i 不断加 1，直到大于 n。

**【例 3.10】**　用 while 语句计算 $T=n!$，即求连乘积 $T=1\times2\times3\times4\times\cdots\times n$。

求连乘积流程图如图 3.16 所示，程序如下：

```
#include <iostream>
using namespace std;
int main( )
{    int i,n;
     float t;
     cout<<"Input an integer:";
     cin>>n;
     t=1.0;
     i=1;
     while (i<=n)
     {   t=t*i;
         i++;
     }
     cout<<"t="<<t<<endl;
     return 0;
}
```

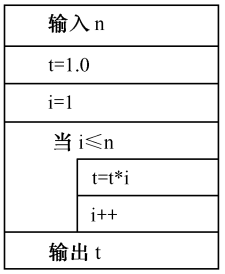



图 3.16　求连乘积流程图

程序执行后显示：

```
Input an integer:5
t=120
```

## 3.3.2 do…while 语句

### 1．do…while 语句的格式

do…while 语句用来实现直到型循环结构，其一般格式为：

```
do
   <语句>
while   (<表达式>);
```

**说明：**

（1）表达式称为循环条件表达式，一般为关系表达式或逻辑表达式，也可为其他表达式，必须用一对圆括号“()”括起来。

（2）语句称为循环体，可以是单条语句，也可以是多条语句（此时必须用花括号“{ }”将多条语句括起来，构成一个复合语句）。

（3）do…while 语句以分号结束。

### 2．do…while 语句的执行过程

do…while 语句的执行过程为：先执行语句，然后计算表达式的值，当表达式的值为真（非 0）时，重复执行指定的语句；当表达式的值为假（0）时，结束循环。do…while 执行流程图如图 3.17 所示，其中图 3.17（a）为 do…while 语句的流程图，图 3.17（b）为其 N-S 流程图。

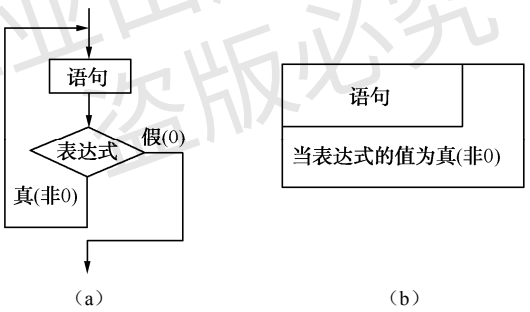



图 3.17　do…while 执行流程图

**【例 3.11】**　用 do…while 语句计算 $S=\sum_{i=1}^{n} i$，即求累加和 $S$=1+2+3+4+…+$n$。

用 do…while 语句求累加和的流程图如图 3.18 所示，程序如下：

```
#include <iostream>
using namespace std;
int main( )
{    int i,n,sum;
     cout<<"Input an integer:";
     cin>>n;
     sum=0;
     i=1;
     do
     {   sum=sum+i;
```

```
        i++;
    }
    while (i<=n);
    cout<<"sum="<<sum<<endl;
    return 0;
}
```

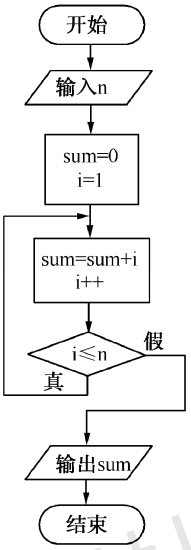



图 3.18　用 do…while 语句求累加和的流程图

**说明：**

（1）do…while 语句是先执行 sum=sum+i;和 i++;语句，后判断表达式 i<=n 是否成立。若条件成立，则继续执行循环体；若条件不成立，则不执行相应语句，退出循环。

（2）即使表达式的值一开始就不成立，语句仍要执行一次。例如，当输入 n 为 0 时，i<=n 不成立，语句 sum=sum+i;和 i++;也要执行一次。

（3）在循环体中，应有不断修改循环条件并最终使循环结束的语句，否则会形成死循环。

**【例 3.12】**　用 do…while 语句计算 $T=n!$，即求连乘积 $T=1\times2\times3\times4\times\cdots\times n$。

用 do…while 语句求连乘积的流程图如图 3.19 所示，程序如下：

```
#include <iostream>
using namespace std;
int main( )
{   int i,n;
    float t;
    cout<<"Input an integer:";
    cin>>n;
    t=1.0;
    i=1;
    do
    {   t=t*i;
        i++;
    }
    while (i<=n);
    cout<<"t="<<t<<endl;
    return 0;
}
```

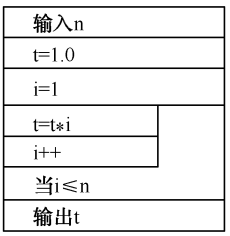



图 3.19　用 do…while 语句求连乘积的流程图

### 3.3.3 for 语句

#### 1. for 语句的一般格式

for 语句是一种功能较强的循环语句，其一般格式为：

```
for  (<表达式 1>;<表达式 2>;<表达式 3>)
     <语句>
```

**说明：**

（1）圆括号内的三个表达式之间用分号“;”隔开。

（2）表达式 1 称为循环初始化表达式，通常为赋值表达式，在简单情况下为循环变量赋初值。

（3）表达式 2 称为循环条件表达式，通常为关系表达式或逻辑表达式，在简单情况下为循环结束条件。

（4）表达式 3 称为循环增量表达式，通常为赋值表达式，在简单情况下为循环变量增量。

（5）语句部分为循环体，它可以是单条语句，也可以是多条语句（此时必须用花括号“{ }”将多条语句括起来，构成一个复合语句）。

#### 2. for 语句的执行过程

for 语句的执行过程如下。

（1）计算表达式 1 的值。

（2）计算表达式 2 的值，若表达式 2 的值为真（非 0），则转到步骤（3）；若表达式 2 的值为假（0），则结束循环。

（3）执行循环体语句。

（4）计算表达式 3 的值，返回步骤（2）继续执行。

for 语句执行流程图及其执行过程如图 3.20 和图 3.21 所示。

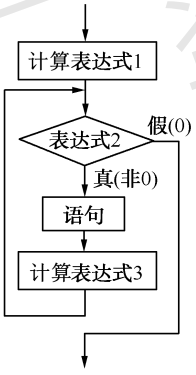



图 3.20　for 语句执行流程图

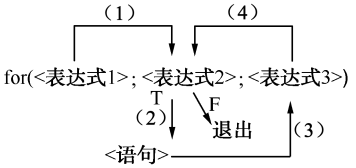



图 3.21　for 语句执行过程

for 语句可以等效于下列 while 语句：

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

**【例 3.13】**　用 for 语句计算 $S=\sum_{i=1}^{n} i$，即求累加和 $S$=1+2+3+4+…+$n$。

用 for 语句求累加和流程图如图 3.22 所示，程序如下：

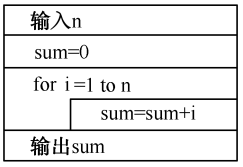



图 3.22　用 for 语句求累加和流程图

```
#include <iostream>
using namespace std;
int main( )
{    int i,n,sum;
     cout<<"Input an integer:";
     cin>>n;
     sum=0;
     for (i=1;i<=n;i++)
        sum=sum+i;
     cout<<"sum="<<sum<<endl;
     return 0;
}
```

在例 3.13 中，表达式 1，即 i=1，完成对循环变量 i 的初始化赋值工作，使 i 的初值为 1。表达式 2，即 i<=n，判断循环变量 i 的值是否小于或等于 n，若不成立则结束循环；若成立则执行 sum=sum+i; 语句，再执行表达式 3。表达式 3，即 i++，使循环变量 i 加 1，然后转表达式 2 继续判断 i<=n 是否成立。

**说明：**

（1）for 语句中的 3 个表达式都可以省略，但其中的两个分号不可以省略。

（2）若省略表达式 1，则应在 for 语句之前给循环变量赋初值。例如：

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

（3）若省略表达式 2，则不判断循环条件，循环无终止地进行下去，形成死循环，即认为表达式 2 始终为真，因此表达式 2 通常不能省略。

（4）若省略表达式 3，则在循环体中应有不断修改循环条件的语句。例如：

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

（5）若省略表达式 1 和表达式 3，则只有表达式 2，即只给出循环条件。例如：

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

此时，for 语句和 while 语句完全相同，上述语句相当于：

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

（6）表达式 1 和表达式 3 可以是一个简单的表达式，也可以是其他表达式，当然可以是逗号表达式，即用逗号“,”隔开的多个简单表达式，它们的运算顺序是按从左到右的顺序进行的。例如：

```
for (i=0,j=0;i+j<40;i++,j+=10)
   cout<<i<<'\t'<<j<<endl;
```

【例 3.14】　用 for 语句计算 $T=n!$，即求连乘积 $T=1\times2\times3\times4\times\cdots\times n$。

程序如下：

```
#include <iostream>
using namespace std;
int main( )
{    int i,n;
     float t;
     cout<<"Input an integer:";
     cin>>n;
     t=1.0;
     for (i=1;i<=n;i++)
        t=t*i;
     cout<<"t="<<t<<endl;
     return 0;
}
```

【例 3.15】　计算 $S=\sum_{k=1}^{20}\frac{1}{k(k+1)}$，即求 $S=\frac{1}{1\times2}+\frac{1}{2\times3}+\cdots+\frac{1}{19\times20}+\frac{1}{20\times21}$。

**分析**：求解该题仍采用求累加和的思想，即用循环语句将级数中各项值：t=1.0/(i*(i+1))依次加入累加和 sum 中。

程序如下：

```
#include <iostream>
using namespace std;
int main( )
{    int i;
     float t,sum;
     sum=0;
     for (i=1;i<=20;i++)
     {   t=1.0/(i*(i+1));
         sum=sum+t;
     }
     cout<<"S="<<sum<<endl;
     return 0;
}
```

程序执行后输出：

```
S=0.952381
```

### 3.3.4　三种循环语句的比较

对于任何一个循环结构，在一般情况下，均可用三种循环语句中的任何一种来实现。但对不同的循环结构，使用不同的循环语句，不仅可以优化程序的结构，还可以精简程序、提高程序执行效率。

（1）while 语句和 for 语句都是先判断循环条件表达式的值，后执行循环体语句；而 do…while 语句则是先执行循环体语句，后判断循环条件表达式的值。

（2）while 语句、do…while 语句和 for 语句都是在循环条件表达式为真时，重复执行循环体语句，在循环条件表达式为假时，结束循环。

（3）当第一次执行 while 语句或 for 语句时，若循环条件表达式为假，则一次也不执行循环体语

句；而当第一次执行 do…while 语句时，即使循环条件表达式为假，也要执行一次循环体语句。也就是说，do…while 语句至少执行一次循环体，而 while 语句和 for 语句有可能一次也不执行循环体。

（4）while 语句和 for 语句用来构成当型循环结构；do…while 语句用来构成直到型循环结构。

（5）在循环体语句至少执行一次的情况下，三种循环语句构成的循环结构可以相互转换。

实际上，用得最多的是 for 语句，其次是 while 语句，do…while 语句相对于前两种语句用得较少。

### 3.3.5 循环语句的嵌套

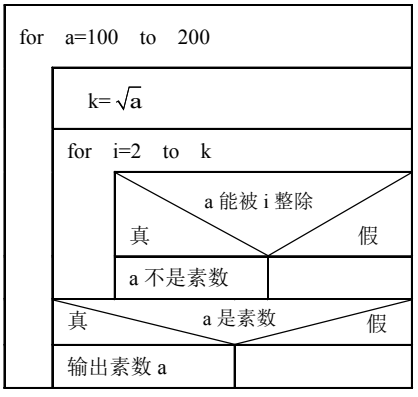



图 3.23 求 100～200 中的所有素数的流程

在程序设计中，如果一个循环语句的循环体包含循环语句，则称其为循环语句的嵌套。循环语句的嵌套又称多重循环。当一个循环语句的循环体中只包含一层循环语句时，称双重循环；若第二层循环语句的循环体中还包含一层循环语句，则称三重循环，依次类推。三种循环语句可以互相嵌套。

**【例 3.16】** 求 100～200 中的所有素数，输出时一行打印 5 个素数。

**分析：**判断一个数 a 是否为素数，只需将它除以 2～$\sqrt{a}$（取整）即可，如果都不能整除，那么 a 就是素数。求 100～200 中的所有素数的流程图如图 3.23 所示，程序如下：


本题微课视频

```
#include <iostream>
#include <cmath>
#include <iomanip>
using namespace std;
int main( )
{   int a,k,i,n;
    n=0;
    for (a=100;a<=200;a++)                    //a 从 100 循环到 200
    {   k=sqrt(a);
        for (i=2;i<=k;i++)                    //判断 a 是否为素数
          if (a%i==0)                         //a 不是素数
            break;
        if (i>k)                              //a 是素数
        {   cout<<setw(12)<<a;                //输出素数
            n=n+1;                            //统计素数的个数
            if (n%5==0)                       //控制每行输出 5 个素数
              cout<<endl;
        }
    }
    cout<<endl;
    return 0;
}
```

程序执行后输出：

```
101         103         107         109         113
127         131         137         139         149
151         157         163         167         173
179         181         191         193         197
199
```

在程序中，调用求解平方根的 sqrt( )库函数需要包含数学函数的头文件 cmath，因此，在程序中使用文件包含命令# include <cmath>。

在例 3.16 中，用内循环 for (i=2;i<=k;i++)判断 a 是否为素数。若 a 不能被 i（i=2～k）整除，则说明 a 是素数，内循环正常结束，此时 i=k+1，输出素数 a；若 a 能被某一个 i（i=2～k）整除，则说明 a 不是素数，使用 break 语句终止内循环的执行，此时 i<=k，不输出 a。用外循环 for (a=100;a<=200;a++){…}来判断 100～200 中哪些数是素数。因此程序使用了循环语句的嵌套。

## 3.4 控制执行顺序的语句

前面已介绍的 C++语句都是根据其在程序中的先后次序，从主函数开始依次执行各语句的。这里要介绍另一类语句，当执行该类语句时，会改变程序的执行顺序，即不依次执行紧跟其后的语句，而是跳到另一条语句处接着执行。从表面上看，循环语句或条件语句也改变了程序的执行顺序，但由于整个循环只是一个语句（条件语句也一样），所以它们仍然是顺序执行的。

### 3.4.1 break 语句

前面已经介绍过使用 break 语句终止 switch 语句的执行，实际上，break 语句也可以用于 while、do…while、for 循环语句中，使它们终止执行，即用于从循环体内跳出，从而提前结束循环。

break 语句的一般格式为：

```
break;
```

若 break 语句出现在嵌套的循环语句中，那么哪一层循环有 break 语句，就跳出该层的循环。break 语句不能用于循环语句和 switch 语句之外的语句中。

### 3.4.2 continue 语句

continue 语句能用于 while、do…while、for 循环语句中，当执行 continue 语句时，将控制转移到 while 和 do…while 语句的循环判断条件处，或 for 语句的增量表达式处，即结束本次循环，重新开始下一次循环。

continue 语句的一般格式为：

```
continue;
```

**【例 3.17】** 输入 10 个整数，统计其中正数的个数及正数的和。

程序如下：

```
#include <iostream>
using namespace std;
int main( )
{    int a,i,k=0,s=0;                    //k 存放正数的个数，s 存放正数的和
     cout<<"Input 10 integers:";
     for (i=1;i<=10;i++)
     {   cin>>a;                         //输入整数到变量 a 中
         if (a<=0)
            continue;                    //若 a 为负数或零，则转到 for 语句中的表达式 3 处执行
         k++;                            //若 a 为正数则 k 加 1，并将 a 累加到 s 中
         s+=a;
     }
     cout<<"k="<<k<<'\t'<<"s="<<s<<'\n';
```

```
    return 0;
}
```

程序执行后显示：

```
Input 10 integers:1 2 3 4 5 6 7 8 9 0
k=9      s=45
```

程序在执行时，用循环语句依次先输入 10 个整数，每次输入后均判断该数是否为负数或零，若为负数或零，则执行 continue 语句，转到 for 语句中的表达式 3 处执行 i++，开始新一轮循环。若为正数，则执行 k++与 s+=a，将 k 加 1，将 a 累加到 s 中。最后输出正数的个数及正数的和。

若 continue 语句出现在嵌套的循环语句中，则 continue 语句只对当前循环起作用。continue 语句不能用于循环语句之外的语句中。

continue 语句和 break 语句的区别是：continue 语句只结束本次循环，而不结束整个循环的执行；而 break 语句则是结束整个循环，不管循环条件是否成立。例如，设有如下程序：

```
#include <iostream>
using namespace std;
int main( )
{      int i;
    for (i=100;i<=200;i++)
    {   if   (i%3==0)
            continue;
        cout<<i<<'\t';
    }
    return 0;
}
```

运行该程序后，输出为：

```
100   101   103   104 …   199   200
```

该程序输出 100～200 中不能被 3 整除的数。

若把程序中的 continue 语句改为 break 语句，即将程序改为：

```
#include <iostream>
using namespace std;
int main( )
{      int i;
    for (i=100;i<=200;i++)
    {   if   (i%3==0)
            break;
        cout<<i<<"     ";
    }
    return 0;
}
```

则运行该程序后，先输出不能被 3 整除的 100、101，当循环变量 i=102 时，由于 i 能被 3 整除，所以执行 break 语句终止循环。因此，程序执行后输出：

```
100       101
```

### 3.4.3 语句标号和 goto 语句

#### 1. 语句标号

前面介绍的语句都是无标号语句，即在语句前面没有标号的语句。C++还允许使用带标号的语句，即在语句前带一个标号。例如：

```
label_10 : a=3
```

C++中带标号语句的一般格式为：

```
<语句标号>:<语句>
```

语句标号指示语句在程序中的位置，常常作为转移语句（goto 语句）的转移目标。

语句标号用标识符来表示，它的命名规则与标识符的命名规则相同，即由字母、数字和下画线组成，且第一个字符必须为字母或下画线，不能用整数作为标号。在 C++中，语句标号可以直接使用，不必先定义后使用，这一点与变量不同。

#### 2. goto 语句

goto 语句将改变程序的流程，无条件地转移到指定语句标号的语句处执行。

goto 语句的一般格式为：

```
goto <语句标号>;
```

其中语句标号是用户指定的，出现在本函数的一条语句的前面。

goto 语句的使用仅局限于函数内，不允许在一个函数中使用 goto 语句转移到其他函数中。goto 语句可以从条件语句或循环语句里面转移到条件语句或循环语句外面，但不允许从条件语句或循环语句外面转移到条件语句或循环语句里面。从结构化程序设计的角度出发，在进行程序设计时，应尽量避免使用 goto 语句。goto 语句可与 if 语句一起构成循环结构。

### 3.4.4 exit()函数和 abort()函数

exit()函数和 abort()函数都是 C++的库函数，其功能都是终止程序的执行，并将控制返回给操作系统。通常，exit()函数用于正常终止程序的执行，而 abort()函数用于异常终止程序的执行。显然，当执行两个函数中的任意一个时，都将改变程序的执行顺序。当使用这两个函数中的任意一个时，都应包含头文件 cstdlib。

#### 1. exit()函数

exit()函数的格式为：

```
exit(<表达式>);
```

其中，表达式的值只能是整型数。通常把表达式的值作为终止程序执行的原因。在执行该函数时，将无条件地终止程序的执行而不管该函数处于程序的什么位置，并将控制返回给操作系统。通常表达式的取值是一个常数：用 0 表示正常退出，用其他整数表示异常退出。

当执行 exit()函数时，系统要做终止程序执行前的收尾工作，如关闭该程序打开的文件、释放变量占用的存储空间（不包括动态分配的存储空间）等。

#### 2. abort()函数

abort()函数的格式为：

```
abort();
```

在调用该函数时，括号内不能有任何参数。在执行该函数时，系统不做结束程序前的收尾工作，直接终止程序的执行。因此通常使用 exit()函数来终止程序的执行。

除了上面介绍的语句和库函数可以改变程序的执行顺序，return 语句也可以改变程序的执行顺序。return 语句的执行过程会在第 5 章进行介绍。

## 3.5 程序设计举例

前面介绍了程序设计的三种基本结构，即顺序结构、分支结构和循环结构。其中分支结构程序主要是用分支语句（if、switch）实现的，而循环结构程序则主要是用循环语句（while、do…while、for）语句实现的。本节对分支语句与循环语句的应用进行了举例。

### 3.5.1 分支语句应用举例

分支语句用于实现分支结构程序设计。能够用分支结构程序解决的常见问题有：比较数的大小、找出若干数中的最大值与最小值、分段函数、解一元二次方程、判断闰年、多分支问题。分支语句有 if 语句和 switch 语句。

#### 1．if 语句

if 语句又有单选 if 语句、双选 if 语句与多选 if 语句，格式分别为：

```
单选 if 语句                双选 if 语句                多选 if 语句
if (<表达式>)               if (<表达式>)               if (<表达式 1>)
    <语句>                      <语句 1>                    <语句 1>
                            else                        else if <表达式 2>
                                <语句 2>                    <语句 2>
                                                            …
                                                        else
                                                            <语句 n>
```

if 语句可以嵌套使用，但应当注意 else 与 if 的配对关系。C++规定：else 总是与其前面最近的还没有配对的 if 进行配对。

**【例 3.18】** 编写程序，求一元二次方程 $ax^2+bx+c=0$ 的解。

**分析：**一元二次方程的解有以下几种情况。

（1）当 $a=0$ 时，若 $b=0$，则方程无解；若 $b\neq0$，则方程有单根。

（2）当 $a\neq0$ 时，若 $d=b^2-4ac=0$，则方程有两个相等的实根；若 $b^2-4ac>0$，则方程有两个不等的实根；若 $b^2-4ac<0$，则方程有两个共轭复根。

方程根 $x_1$、$x_2$ 的求解公式如图 3.24 所示。

$$a\begin{cases}=0,\ b\begin{cases}=0,\ \text{方程无解}\\ \neq0,\ x_1=-c/b\end{cases}\\ \neq0,\ d=b^2-4ac\begin{cases}=0,\ x_1=x_2=t_1\\ >0,\ x_1=t_1+t_2,\ x_2=t_1-t_2,\\ <0,\ x_1=t_1+t_2\mathrm{i},\ x_2=t_1-t_2\mathrm{i},\end{cases}\\ t_1=-b/(2a)\\ t_2=\sqrt{|d|}/(2a)\end{cases}$$

图 3.24 方程根 $x_1$、$x_2$ 的求解公式

根据求解公式，可将程序的结构框架按如下方式构建：判断 a 是否为 0，可以用双选 if 语句；判断 b 是否为 0，可以用嵌套双选 if 语句；判断 d，可用嵌套三选 if 语句。因此，程序的结构框架如下：

```
if (a==0)                          //双选 if 语句
    if (b==0)                      //嵌套双选 if 语句
      输出“方程无解”
```

```
    else
      x1=-c/b;
else
    if(d==0)                                    //嵌套三选 if 语句
     x1=x2=t1;
    else if(d>0)
      x1=t1+t2;x2=t1-t2;
    else
      x1=t1+t2i;x2=t1-t2i;
```

由上述程序结构框架可知，程序中应定义实型变量 a、b、c、d、t1、t2、x1、x2，用 cin 输入方程系数 a、b、c，计算并输出方程根 x1、x2。按程序的结构框架很容易写出下列程序：

```
#include <iostream>
#include <cmath>
using namespace std;
int main( )
{   float a,b,c,d,t1,t2,x1,x2;
    cout<<"Input a,b,c:";
    cin>>a>>b>>c;
    if (a==0.0)                                 //a=0，方程退化为一元一次方程
      if(b==0.0)                                //b=0，方程无解
        cout<<"Input data error!"<<endl;
      else                                      //b!=0，方程有单根
      {  x1=-c/b;
         cout<<"Single root:"<<x1<<endl;
      }
    else                                        //a!=0，方程为一元二次方程
    {     d=b*b-4*a*c;                          //计算 d=b2-4ac
          t1=-b/(2*a);
          t2=sqrt(fabs(d))/(2*a);
          if (d==0.0)                           //d=0，方程有两个相等的实根
          {  x1=t1;
             cout<<"Two equal real roots:"<<x1<<endl;
          }
          else if (d>0.0)                       //d>0，方程有两个不等的实根
          {  x1=t1+t2;
             x2=t1-t2;
             cout<<"Two distinct real roots:"<<x1<<','<<x2<<endl;
          }
          else                                   //d<0，方程有两个共轭复根
          {  cout<<"Complex roots:";
                cout<<t1<<'+'<<t2<<'i'<<','<<t1<<'－'<<t2<<'i'<<endl;
          }
     }
    return 0;
}
```

其中，sqrt()为求实数平方根的函数，fabs()为求实数绝对值的函数。当使用它们时，需要在程序中使用文件包含命令 # include <cmath>。

程序运行后显示：

```
Input a,b,c:2   -5   3
```

输出：

```
Two distinct real roots:1.5,1
```

由例 3.18 可知，C++程序设计的一般步骤如下。

（1）进行数学分析，建立求解数学模型。

（2）根据数学模型确定程序框架及所用语句。

（3）根据数学模型确定需要定义的变量及其数据类型。

（4）给变量输入数据。

（5）编写计算程序，执行程序得到运算结果。

（6）输出运算结果。

### 2．switch 语句

switch 语句的格式为：

```
switch (<表达式>)
  {case <常量表达式 1>:〔<语句 1>〕
   …
   case <常量表达式 n-1>:〔<语句 n-1>〕
   〔default:<语句 n>〕
  }
```

**注意：**在执行 switch 语句的过程中，一般用 break 语句结束 switch 语句的执行。

**【例 3.19】** 输入某一年的年份和月份，计算该月的天数。

**分析：**

（1）一年中的大月（1 月、3 月、5 月、7 月、8 月、10 月、12 月），每月的天数为 31 天。

（2）一年中的小月（4 月、6 月、9 月、11 月），每月的天数为 30 天。

（3）对于 2 月，则要判断该年是平年还是闰年，平年的 2 月为 28 天，闰年的 2 月为 29 天。

某年符合下面两个条件之一就是闰年：①年份能被 400 整除；②年份能被 4 整除，但不能被 100 整除。

已知年（year）、月（month）求天数（day）的求解公式如图 3.25 所示。

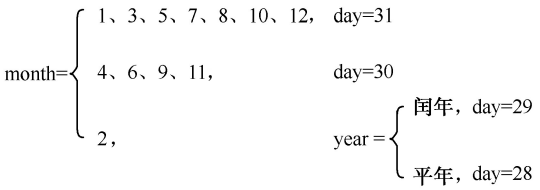



图 3.25　已知年、月求天数的求解公式

根据求解公式，可将程序的结构框架按如下方式构建：用 switch 语句对月份 month 进行多选判断，当 month=2 时，用双选 if 语句判断是否为闰年。程序的结构框架如下：

```
switch (month)
{   case   1、3、5、7、8、10、12 ：   day=31
    case   4、6、9、11 ：    day=30
    case   2 ：
           if (year==闰年)
               day=29
           else
               day=28
}
```

由上述程序的结构框架可知，程序中应定义整型变量 year、month、day，用 cin 输入 year、month，然后计算并输出 day。

程序如下：

```
#include <iostream>
using namespace std;
int main( )
{    int year,month,day;
     cout<<"Input year and month:";
     cin>>year>>month;
     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:if (year%400==0 || year%4==0 && year%100!=0)
                    day=29;
                else
                    day=28;
                break;
         default:cout<<"Input data error!"<<endl;day=0;break;
     }
     if (day!=0)
     cout<<"The day of "<<year<<','<<month<<" is "<<day<<endl;
     return 0;
}
```

程序运行后显示：

```
Input year and month:2002    1
```

输出：

```
The day of 2002,1 is 31
```

## 3.5.2 循环语句应用举例

循环语句用于实现循环结构程序设计。循环语句有 while 语句、do…while 语句和 for 语句。三种语句的格式如下：

| while 语句 | do…while 语句 | for 语句 |
| --- | --- | --- |
| while(<表达式>)<br><语句>; | do<br><语句><br>while (<表达式>); | for (<表达式 1>,<表达式 2>,<表达式 3>)<br><语句>; |

应注意三种循环语句的特点、区别及由它们构成的嵌套循环结构。

能够用循环结构程序解决的常见问题有：累加和、连乘积、求一批数的和及最大值与最小值、求数列的前 *n* 项、判断素数、求两个整数的最大公约数和最小公倍数、用迭代法求平方根、用穷举法求

不定方程组的整数解、打印图形等。

【例 3.20】 用公式 $e=1+\sum_{n=1}^{10}\frac{1}{n!}$，即 $e=1+\frac{1}{1!}+\frac{1}{2!}+\frac{1}{3!}+\cdots+\frac{1}{n!}$，求自然对数底 e 的近似值（$n$=5）。

分别用 while 语句、do…while 语句和 for 语句三种语句实现。

本例应用求累加和的方法求 e 的近似值，而求累加和的方法可归纳如下：

| 定义变量 | 算法 | 初值 |
|---|---|---|
| t：存放第 i 项分母值 | t=t*i; | t=1 |
| p：存放第 i 项值 | p=1/t; | p=1 |
| s：存放累加和 | s=s+p; | s=1 |
| i：循环变量 | i=i+1; | i=1 |

程序如下：

```
#include <iostream>
using namespace std;
int main( )
{int i=1;
 float s=1.0,t=1,p=1;
 while (i<=5)
      {t=t*i;
       p=1/t;
       s=s+p;
       i++;
      }
cout<<"e="<<s<<endl;
return 0;
}
```

```
#include <iostream>
using namespace std;
int main( )
{int i=1;
 float s=1.0,t=1, p=1;
 do
      {t=t*i;
       p=1/t;
       s=s+p;
       i++;
      } while (i<=5);
cout<<"e="<<s<<endl;
return 0;
}
```

```
#include <iostream>
using namespace std;
int main( )
{int i;
 float s=1.0,t=1,p=1;
 for (i=1; i<=5;i++)
      {t=t*i;
       p=1/t;
       s=s+p;
      }
cout<<"e="<<s<<endl;
return 0;
}
```

程序执行输出：

```
e=2.71667
```

在例 3.20 中，程序运行输出结果 e=2.71667 与自然对数的底 e=2.71828 误差较大。为了使误差控制在规定的范围 $\delta$ 内，应要求数列中最后一项的值小于 $\delta$，即 $\frac{1}{n!}<\delta$。

若取 $\delta$=0.00001，则循环的结束条件为 p<$\delta$=0.00001，而循环条件为 p>=0.00001。将上述程序改为：

```
#include <iostream>
using namespace std;
int main( )
{int i=1;
float s=1.0,t=1,p=1;
while (p>=0.00001)
      {t=t*i;
       p=1/t;
       s=s+p;
       i++;
      }
cout<<"e="<<s<<endl;
return 0;
}
```

```
#include <iostream>
using namespace std;
int main( )
{int i=1;
float s=1.0,t=1, p=1;
do
  {t=t*i;
  p=1/t;
  s=s+p;
  i++;
  }while (p>0.00001);
cout<<"e="<<s<<endl;
return 0;
}
```

```
#include <iostream>
using namespace std;
int main( )
{int i;
 float s=1.0, t=1, p=1;
 for (i=1; p>=0.00001;i++)
      {t=t*i;
       p=1/t;
       s=s+p;
      }
cout<<"e="<<s<<endl;
return 0;
}
```

程序执行输出：

```
e=2.71828
```

从例 3.20 可以看出，累加和问题用三种语句都可以实现。当读者学会这三种语句后，用哪种语句都可以编写循环结构程序。

**【例 3.21】** $s=\frac{1}{0!}-\frac{1}{1!}+\frac{1}{2!}-\frac{1}{3!}+\frac{1}{4!}+\cdots+\frac{(-1)^n}{n!}$，用累加和的方法求 $s$ 的值，直到最后一项 $\left|\frac{(-1)^n}{n!}\right|<0.00001$。

**分析**：本例与例 3.20 的唯一区别是，第 $i$ 项值由 p=$\frac{1}{\mathrm{i}!}$ 改成 p=$\frac{(-1)^{\mathrm{i}}}{\mathrm{i}!}$。因此，需要定义一个能表示分子值的变量 t1，其初值为 1，循环算法为 t1=(−1)* t1；第 i 项值 p= t1/t。

循环结束条件是 $\left|\frac{(-1)^n}{n!}\right|=\frac{1}{n!}<0.00001$，即循环条件是 1/t>=0.00001。用 do…while 语句实现的程序如下：

```
#include <iostream>
using namespace std;
int main( )
{    int i, t1=1;
     float s=1.0, t=1,p;
     i=1;
     do
     {    t=t*i;
          t1=(−1)*t1;
          p=t1/t;
          s=s+p;
          i++;
     } while (1/t>0.00001);
     cout<<"s="<<s<<endl;
     return 0;
}
```

程序执行后输出：

```
s=0.367879
```

**【例 3.22】** 裴波那契数列的前几个数为 1，1，2，3，5，8，…其规律为：


本题微课视频

$f_1=1$　　（$n=1$）
$f_2=1$　　（$n=2$）
$f_n=f_{n-1}+f_{n-2}$　　（$n\geqslant 3$）

编写程序求此数列的前 40 个数。

**分析**：可设两个变量 f1 和 f2，它们的初值为 f1=1，即数列的第 1 项；f2=1，即数列的第 2 项，用一个循环结构来求数列的前 40 项，每次处理两项，共循环 20 次。进入循环后，先输出 f1、f2，然后令 f1=f1+f2，即可求得第 3 项，再令 f2=f2+f1，注意此时的 f1 已经是第 3 项了，即可求得第 4 项；当进入下一次循环时，先输出第 3 项和 4 项，然后按上述方法求得第 5 项和 6 项，依次类推，即可求得前 40 项。

程序如下：

```
#include <iostream>
#include <iomanip>
using namespace std;
int main( )
{    long int f1,f2;
     int i;
     f1=1;f2=1;
     for (i=1;i<=20;i++)
     {   cout<<setw(12)<<f1<<setw(12)<<f2;
         if (i%2==0)
            cout<<endl;
         f1=f1+f2;
         f2=f2+f1;
     }
     return 0;
}
```

在例 3.22 中，if 语句的作用是在一行中输出 4 个数。

程序运行后输出：

| | | | |
|---|---|---|---|
| 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 | 832040 | 1346269 | 2178309 |
| 3524578 | 5702887 | 9227465 | 14930352 |
| 24157817 | 39088169 | 63245986 | 102334155 |

**【例 3.23】** 100 名学生种 100 棵树，其中高中生每人种 3 棵树，初中生每人种 2 棵树，小学生每 3 人种 1 棵树，问高中生、初中生、小学生各有多少人？

**分析**：设有 $i$ 名高中生、$j$ 名初中生、$k$ 名小学生，根据题意可以列出如下方程组：

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

三个变量只列出两个方程式，为不定方程组，此时可以用双重循环来求解，外循环的循环变量为 i，因为高中生每人种 3 棵树，所以 i 的最大值为 33；内循环的循环变量为 j，因为初中生每人种 2 棵树，所以 j 的最大值为 50。这种方法称为枚举法或穷举法。

种树问题流程图如图 3.26 所示。

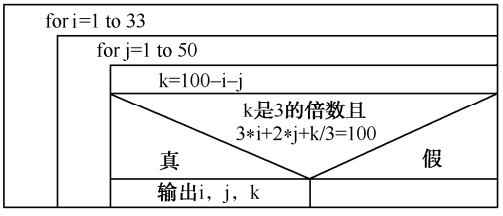



图 3.26　种树问题流程图

程序如下：

```
#include <iostream>
using namespace std;
int main( )
{    int i,j,k;
     for (i=1;i<=33;i++)                                  //高中生人数 i，从 1 循环到 33
        for (j=1;j<=50;j++)                               //初中生人数 j，从 1 循环到 50
        {   k=100−i−j;                                    //计算小学生人数 k=100−i−j
            if ((k%3==0) && (3*i+2*j+k/3==100))           //小学生人数必须是 3 的倍数
                                                          //并且总共种树 100 棵
            cout<<i<<'\t'<<j<<'\t'<<k<<endl;              //输出一个解
        }
     return 0;
}
```

程序运行后输出：

```
 5        32       63
 10       24       66
 15       16       69
 20       8        72
```

**【例 3.24】** 编写程序，按下列格式打印九九乘法表。

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

**分析**：用双循环完成，外层用 for 语句构成循环，用于控制输出的行数（共 9 行）；内层也用 for 语句构成循环，用于控制每行的输出。

打印九九乘法表流程图如图 3.27 所示，程序如下：

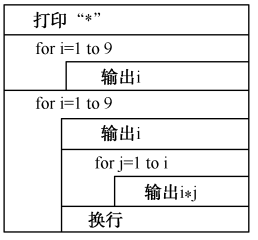



图 3.27　打印九九乘法表流程图

```
#include <iostream>
using namespace std;
int main( )
{    int i,j;
     cout<<"*"<<'\t';
     for (i=1;i<=9;i++)                    //输出标题行“*  1  2  3  4 … 9”
        cout<<i<<'\t';
     cout<<endl;
     for (i=1;i<=9;i++)                    //外循环控制第一个乘数 i 从 1 变到 9
     {   cout<<i<<'\t';
         for (j=1;j<=i;j++)                //内循环控制第二个乘数 j 从 1 变到 i
            cout<<i*j<<'\t';               //计算并输出每项 i*j
         cout<<endl;                       //输出完一行后换行
     }
     return 0;
}
```

# 本 章 小 结

一个 C++程序由若干源程序文件组成，一个源程序文件可以由编译预处理命令和若干函数组成，一个函数由函数说明和函数体组成，函数体由变量定义和若干执行语句组成。语句是组成程序的基本单元。

### 1．程序的三种基本结构

组成 C++程序的函数是由若干基本结构组成的。C++有三种基本结构，即顺序结构、分支结构和循环结构，各种控制结构是通过语句来实现的。

### 2．C++中的语句

C++中的语句可以分成六大类，即说明语句、控制语句、函数调用语句、表达式语句、空语句和复合语句，其中控制语句主要有分支语句与循环语句。

### 3．分支语句

分支语句用于实现分支结构程序设计。分支语句有 if 语句和 switch 语句。

（1）if 语句。

if 语句可以嵌套使用，但应当注意 else 与 if 的配对关系。C++规定：else 总是与其前面最近的还没有配对的 if 进行配对。

（2）条件运算符和条件表达式。

由条件运算符（? :）可以构成条件表达式。

（3）switch 语句。

在执行 switch 语句的过程中，每当执行完一个 case 后面的语句后，程序就会不加判断地自动执行下一个 case 后面的语句。如果要结束 switch 语句的执行，可用 break 语句来实现。

能够用分支结构程序解决的常见问题有：比较数的大小、找出若干数中的最大值与最小值、分段函数、解一元二次方程、判断闰年、多分支问题。

### 4．循环语句

循环语句用于实现循环结构程序设计。循环语句有 while 语句、do…while 语句和 for 语句。

（1）while 语句。

while 语句用来实现当型循环结构。

（2）do…while 语句。

do…while 语句用来实现直到型循环结构。

（3）for 语句。

for 语句是功能较强的一种循环语句。

应注意三种循环语句的特点、区别及由它们构成的嵌套循环结构。

能够用循环结构程序解决的常见问题有：累加和、连乘积、求一批数的和及最大值与最小值、求数列的前 *n* 项、判断素数、求两个整数的最大公约数和最小公倍数、用迭代法求平方根、用穷举法求不定方程组的整数解、打印图形等。

### 5．控制执行顺序的语句

（1）break 语句。

break 语句只能用在循环语句和 switch 语句中，其功能是终止循环语句和 switch 语句的执行。

（2）continue 语句。

continue 语句只能用在循环语句中，其功能是结束本次循环，重新开始下一次循环。

（3）goto 语句。

C++允许语句前带有一个标号，该标号称为语句标号。

goto 语句的功能是改变程序的执行流程，无条件地转移到指定语句标号的语句处执行。从结构化程序设计的角度出发，在进行程序设计时，应尽量避免使用 goto 语句。

### 6．本章重点、难点

**重点：**程序的三种基本结构，各种控制语句的格式、功能、执行过程及使用方法。

**难点：**使用 if 语句、while 语句、for 语句编写分支程序与循环程序。

## 习　　题

3.1　程序的三种基本结构是什么？

3.2　C++中的语句分为哪几类？

3.3　怎样区分表达式和语句？

3.4　程序的多路分支可通过哪两种语句来实现？说出用这两种语句实现多路分支的区别。

3.5　在使用 switch 语句时应注意哪些问题？

3.6　用于实现循环结构的循环语句有哪三种？分别用于实现哪两种循环结构？这三种循环语句在使用上有何区别？

3.7　分支程序与循环程序常用于解决哪些实际问题？

3.8　当 continue 语句与 break 语句用于循环结构时，在使用上有何区别？

3.9　程序的正常终止与异常终止有何区别，分别用什么函数来实现？在使用这些函数时应包含什么头文件？

3.10　设有程序段：

```
x=-1;
if  (a!=0)  {if  (a>0)  x=1;}  else    x=0;
```

写出该程序段表示的数学函数关系。

3.11　写出下列程序的运行结果。

```
#include <iostream>
using namespace std;
int main( )
{    int a=2,b=-1,c=2;
     if (a<b)
     if (b<0) c=0;
     else c=c+1;
     cout<<c<<endl;
     return 0;
}
```

3.12　写出下列程序的运行结果。

```
#include <iostream>
using namespace std;
int main( )
{    int i=10;
     switch (i)
        {  case    9:i=i+1;
           case  10:i=i+1;
           case  11:i=i+1;
           default :i=i+1;
        }
     cout<<i<<endl;
     return 0;
}
```

3.13　设计一个程序，判断通过键盘输入的整数的正负性和奇偶性。

3.14　有下列函数：

$$y=\begin{cases}-x+2.5 & (x<2)\\ 2-1.5(x-3)^2 & (2\leqslant x<4)\\ \dfrac{x}{2}-1.5 & (x\geqslant 4)\end{cases}$$

设计一个程序，通过键盘输入 $x$ 的值，并输出 $y$ 的值。

3.15　设计一个程序，通过键盘输入 a、b、c 三个整数，将它们按照从大到小的顺序输出。

3.16　输入平面直角坐标系中一点的坐标值（$x$，$y$），判断该点是在哪一个象限中或哪一条坐标轴上。

3.17　简单计算器。设计一个程序计算表达式 data1 op data2 的值，其中 data1、data2 为两个实数，op 为运算符（+、−、*、/），并且都由键盘输入。

3.18　奖金税率如下（$a$ 代表奖金，$r$ 代表税率）：

| | |
|---|---|
| $a<500$ | $r=0$ |
| $500\leqslant a<1000$ | $r=3\%$ |
| $1000\leqslant a<2000$ | $r=5\%$ |
| $2000\leqslant a<5000$ | $r=8\%$ |
| $a\geqslant 5000$ | $r=12\%$ |

输入一个奖金数，求税率、应交税款及实得奖金数。

3.19　写出下列程序的运行结果。

```
#include <iostream>
using namespace std;
int main( )
{    int n=4;
     while (--n)
        cout<<n<<'\t';
     cout<<endl;
      return 0;
}
```

3.20　写出下列程序的运行结果。

```
#include <iostream>
using namespace std;
int main( )
{    int x=3;
     do
       {cout<<x<<'\t';
       }
     while (!(x--));
     cout<<endl;
      return 0;
}
```

3.21　写出下列程序的运行结果。

```
#include <iostream>
using namespace std;
int main( )
{    int i=0,j=0,k=0,m;
     for (m=0;m<4;m++)
     switch (m)
     {   case 0:i=m++;
         case 1:j=m++;
         case 2:k=m++;
         case 3:m++;
     }
     cout<<i<<'\t'<<j<<'\t'<<k<<'\t'<<m<<endl;
      return 0;
}
```

3.22　写出下列程序的运行结果。

```
#include <iostream>
using namespace std;
int main( )
{     int n=0,m=0;
      for (int i=0;i<3;i++)
          for (int j=0;j<3;j++)
              if (j>=i) n++;m++;
     cout<<n<<'\n'<<m<<'\n';
     return 0;
}
```

3.23　求 $\sum_{n=1}^{100}\frac{1}{n}$ 的值，即求 $1+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+\cdots+\frac{1}{100}$ 的值。

3.24　编程计算 $y=1+\frac{1}{x}+\frac{1}{x^2}+\frac{1}{x^3}+\cdots$ 的值（$x>1$），直到最后一项小于 $10^{-4}$。

3.25　输入两个正整数 $m$ 和 $n$，求其最大公约数和最小公倍数。

3.26　某月 10 天内的气温（℃）为：−5、3、4、0、2、7、0、5、−1、2，编程统计气温在 0℃以上、0℃和 0℃以下的天气各多少天？并计算这 10 天的平均气温值。

3.27　求 $\sum_{n=1}^{10}n!$，即求 1!+2!+3!+4!+…+10!。

3.28　设用 100 元买 100 支笔，其中钢笔每支 3 元，圆珠笔每支 2 元，铅笔每支 0.5 元，问钢笔、

圆珠笔和铅笔可以各买多少支（每种笔至少买 1 支）？

3.29　编程显示如图 3.28 所示的菱形。

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

图 3.28　编程输出菱形

## 实　验　A

### 1．实验目的

（1）初步学会 VC++开发环境中单步执行程序的方法。

（2）掌握 if 语句的格式与使用方法，学会两路分支程序的设计方法。

（3）掌握嵌套 if 语句的格式与使用方法，学会多路分支程序的设计方法。

（4）掌握 switch 语句的格式与使用方法，学会多路分支程序的设计方法。

### 2．实验内容

（1）演示单步执行程序的方法。

（2）设计一个程序，判断通过键盘输入的整数的正负性和奇偶性。

实验数据：−3 与 5。

（3）有下列函数：

$$y=\begin{cases}-x+3.5 & (x<5)\\ 20-3.5(x+3)^2 & (5\leqslant x<10)\\ \dfrac{x}{2}-3.5+\sin x & (x\geqslant 10)\end{cases}$$

设计一个程序，通过键盘输入 $x$ 的值，并输出 $y$ 的值。

实验数据：6 与 11。

（4）编写程序，将百分制成绩转换成等级制成绩。

转换方法为：

| | |
|---|---|
| 90～100 | A |
| 80～89 | B |
| 70～79 | C |
| 60～69 | D |
| 0～59 | E |

通过键盘输入百分制成绩，输出成绩等级。

实验数据：75

### 3．实验要求

（1）编写实验程序。

（2）在 VC++运行环境中输入源程序。
（3）单步执行程序。
（4）编译运行源程序。
（5）输入测试数据进行程序测试。
（6）写出运行结果。

# 实 验 B

### 1．实验目的

（1）掌握 while 语句的格式与使用方法，学会当型循环程序的设计方法。
（2）掌握 for 语句的格式与使用方法，学会当型循环程序的设计方法。
（3）掌握 do…while 语句的格式与使用方法，学会直到型循环程序的设计方法。
（4）学会求常用级数的编程方法。

### 2．实验内容

（1）输入一行字符，分别统计其中英文字母、空格、数字字符和其他字符的个数。
提示：用 cin.get(c)函数通过键盘输入一个字符给变量 c，直到输入换行字符'\n'。
实验数据：

```
I am Student 1234
```

（2）设有一个数列，前四项为 0、0、2、5，以后每项分别是其前四项之和，编程求此数列的前 20 项。
（3）求π的近似值的公式为：

$$\frac{\pi}{2}=\frac{2}{1}\times\frac{2}{3}\times\frac{4}{3}\times\frac{4}{5}\times\cdots\times\frac{2n}{2n-1}\times\frac{2n}{2n+1}\times\cdots$$

其中，$n$=1，2，3，…设计一个程序，求出当 $n$=1000 时π的近似值。
（4）求出 1～599 中能被 3 整除且至少有一位数字为 5 的所有整数。例如，15、51、513 均是满足条件的整数。
**提示**：将 1～599 中三位整数 i 分解成个位、十位、百位，分别存放在变量 a、b、c 中。然后判断 a、b、c 中是否有 5。将三位整数 i（设 i=513）分解成个位、十位、百位的方法是：

```
c=i%10;          //c= i%10=513%10=3
a=i/10;          //a= i/10=51
b=a%10;          //b=a%10=51%10=1
a=a/10;          //a=a%10=51/10=5
```

### 3．实验要求

（1）编写实验程序。
（2）在 VC++运行环境中输入源程序。
（3）编译运行源程序。
（4）输入测试数据进行程序测试。
（5）写出运行结果。