第 3 章　程序流程控制

学习目标

(1) 掌握顺序结构的概念。

(2) 掌握分支和多分支语句的语法格式及用法。

(3) 理解循环的概念，掌握 while、do-while、for 三种循环语句的语法格式、用法及区别。

(4) 掌握选择结构和循环结构嵌套的含义及用法。

结构化程序设计是一种程序设计的原则和方法，按照这种原则和方法设计出的程序具有结构清晰、可读性好、易于修改和容易验证等优点。任何算法都可以由 3 种基本结构——顺序结构、选择结构和循环结构组成。

3.1　顺序结构程序设计

用C++ 编写程序时，实现顺序结构的方法非常简单，只需要将语句顺序排列即可。例如交换两个整数值的程序段：

```
t=x;
x=y;
y=t;
```

就是顺序结构。

3.2　选择结构程序设计

选择结构又称分支结构，它的作用是根据指定的条件，来决定某些操作是执行还是不执行，或者决定从给定的若干操作中选择部分代码执行。C++ 的选择结构通过 if 语句和 switch 语句来实现。

3.2.1　if 语句

if 语句用来判断给定的条件是否满足，根据判定结果的真或假来决定执行两组操作之一。C++ 中 if 语句有三种形式：单分支 if 语句、双分支 if 语句、多分支 if 语句。

1. 单分支 if 语句

基本格式为：

```
if(表达式)
    语句
```

执行这一结构时，首先对表达式的值进行判断。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。其执行流程如图 3.1 所示。

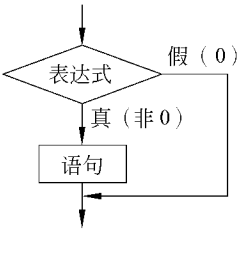

图 3.1　if 结构的流程

【例 3-1】　在两个数中取大数。

程序代码如下：

```
//e3_1.cpp
#include<iostream.h>
void main()
{
    int num1, num2, max;
    cout<<"input 2 numbers:";
    cin>>num1>>num2;
    max=num1;
    if(max<num2)max=num2;
    cout<<"max="<<max<<endl;
}
```

程序运行结果：

```
input 2 numbers:
5 3
max=5
```

程序说明：

本例中，输入两个整数 num1，num2。把 num1 先赋予变量 max，再用 if 语句判别 max 和 num2 的大小，如果 max 小于 num2，则把 num2 赋予 max。因此 max 总是两者中的大数，最后输出 max 的值。

2. 双分支 if 语句：if-else

if-else 结构构造了一种双分支结构，如图 3.2 所示。

基本格式为：

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

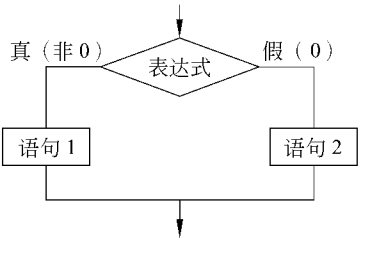

图 3.2　if-else 结构

执行该结构时，首先对表达式的值进行判断。如果表达式的值为“真”，则执行语句 1；否则执行语句 2。

【例 3-2】　在两个数中取大数。

程序代码如下：

```
//e3_2.cpp
#include<iostream.h>
void main()
{
    int num1, num2;
    cout<<"input 2 numbers:" ;
    cin>>num1>>num2;
    if(num1>num2)
        cout<<"max="<<num1<<endl;
    else
```

```
        cout<<"max="<<num2<<endl;
}
```

程序运行结果：

```
input 2 numbers:
5 3
max=5
```

程序说明：

本例与例 3-1 功能相同，输入两个整数 num1 和 num2，判断两者的大小，若 num1 大，则输出 num1，否则输出 num2。

改用 if-else 语句实现，比 if 结构易于理解且格式清晰。

3. 多分支 if 语句：if-else if

if-else if 是一种多分支选择结构。其格式为：

```
if(表达式 1)
    语句 1
else if(表达式 2)
        语句 2
        ⋮
     else  if(表达式 n)
              语句 n
          else 语句 n+ 1
```

执行这一结构时，依次判断表达式的值，当出现某个表达式值为“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。如果所有的表达式均为假，则执行语句 n+1。然后继续执行后续程序。if-else if 结构的执行过程如图 3.3 所示。

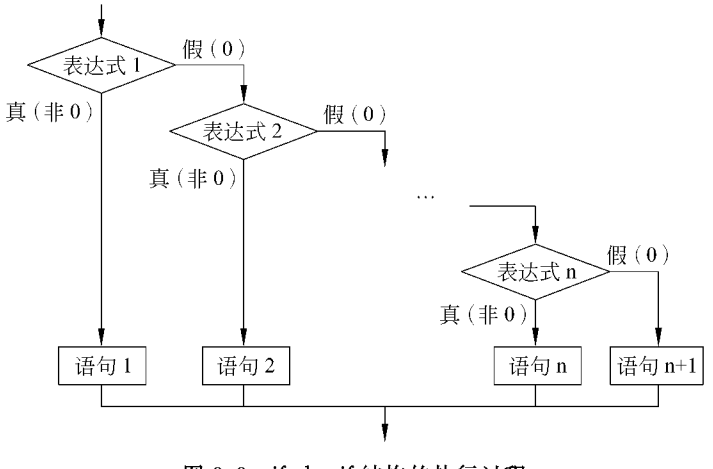

图 3.3 if-else if 结构的执行过程

【例 3-3】 将学生成绩由百分制转化为等级制。规则如下：

(1) 85 分(含)以上为 A 级。

(2) 70 分(含)以上且 85 分以下为 B 级。

(3) 60 分(含)以上且 70 分以下为 C 级。

(4) 60 分以下为 D 级。

程序代码如下：

```
//e3_3.cpp
#include<iostrea.h>
void main()
{
    float score;
    cout<<please input a score: ";
    cin>>score;
    if(score>=85)
        cout<<"the score "<<score<<"is A\n";
    else if(score>=75)
            cout<<"the score "<<score<<"is B\n";;
         else if(score>=60)
                 cout<<"the score "<<score<<"is C\n";;
              else
                 cout<<"the score "<<score<<"is D\n";;
}
```

程序运行结果：

```
please input a score: 89
the score 89.000000 is A
```

程序说明：

这是一个多分支选择的问题，用 if-else if 语句实现，通过判断输入的成绩变量 score 所在的范围，分别给出不同的输出。

使用 if 语句还应注意下列几个问题。

(1) 三种形式的 if 语句中，在关键字 if 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。

例如：

```
if(x==y&&a==b)  cout<<"x=y, a=b"<<endl;
```

执行时 if 语句先对表达式求解，若表达式值为“真”，则执行后面的语句。

表达式也可以是其他表达式，如赋值表达式等，甚至也可以是一个变量。例如：

```
if(x=3) 语句;
if(a) 语句;
```

都是合法的，只要表达式的值为“真”，则执行其后的语句。

(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。空语句也是允许的，表示什么也不做。如 if(b>0);。

(3) 若在条件满足时需要执行多条语句，必须把多条语句用花括号括起来组成一个复合语句；若在条件满足时执行 1 条语句，则花括号可以省略，以上例子都属于此情况；注意在右括号后不要加分号。

例如：

```
if(x>1)
```

```
    {x++;y--;}
else
    {x--;y++;}
```

(4) 正确地使用缩进格式,有助于更好地理解程序,尤其是在使用 if 语句嵌套时。同时,在容易引起混淆的地方添加花括号来保证逻辑关系的正确性。

3.2.2 switch 语句

switch 是一种多分支选择结构,也称为标号分支结构。其格式为:

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

switch 结构在执行时,首先计算 switch 判断表达式的值,并按照计算结果依次寻找 case 子结构中与之相等的常量表达式,若找到,则执行该 case 子结构后的语句序列;若找不到与之相等的常量表达式,则执行 default 后的语句序列。default 子句不是必需的。若结构中无 default 子结构且没有相符的 case 子结构时,则什么也不做。执行流程如图 3.4 所示。

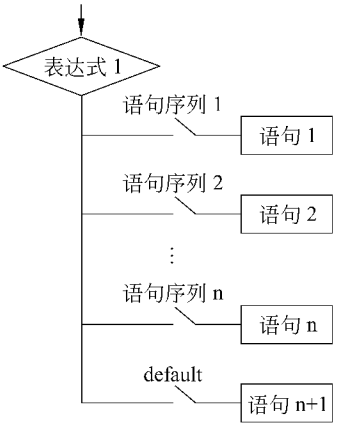

图 3.4 switch 结构的流程

在使用 switch 结构时,应注意以下几个问题。

(1) case 后的各常量表达式值不能相同。

(2) case 后允许有多个语句,可以不用花括号括起来。

(3) 程序执行至与 switch 表达式值匹配的 case 子结构后的语句序列 m 后,不是立即退出 switch 结构,而是继续执行语句序列 m+1,直至语句序列 n+1。若需要在执行语句序列 m 后,立即退出 switch 结构,则在每个语句序列后加一条 break 语句,break 语句用于跳出 switch 结构。

(4) switch 结构也可以嵌套。

【例 3-4】 输入 1～7 中的数字,将其转换成相应的星期英文单词。

程序代码如下:

```
//e3_4.cpp
#include<iostream.h>
void main()
{
    int num;
    cin>>num;
    switch(num)
    {
        case 1: cout<<"Monday\n"; break;
        case 2: cout<<"Tuesday\n"; break;
```

```
        case 3: cout<<"Wednesday\n"; break;
        case 4: cout<<"Tursday\n"; break;
        case 5: cout<<"Friday\n"; break;
        case 6: cout<<"Saturday\n";break;
        case 7: cout<<"Sunday\n";break;
        default: cout<<"error\n";
    }
}
```

程序运行结果：

```
1↵
Monday
```

若输入 1～7 之外的数字则显示 error。

程序说明：

在本例中，每一个 case 子结构最后都有一条 break 语句，用于跳出 switch 流程。程序中若无 break 语句，运行时，若输入数字 1，则运行结果为：

```
Monday
Tuesday
Wednesday
Tursday
Friday
Saturday
Sunday
error
```

在使用 switch 结构时，一定要注意 break 语句的位置。

3.3　循环结构程序设计

3.3.1　while 语句

while 用来实现“当型”循环结构。其格式为：

```
while(表达式)
    语句
```

在执行 while 语句时，先对表达式进行计算，若值为“真”(非 0)，则执行循环体语句；否则跳过循环体语句，执行 while 结构后面的语句。每执行完一次循环体语句，都对表达式进行一次计算和判断。若表达式值为 0，则立即跳出循环体。流程如图 3.5 所示。

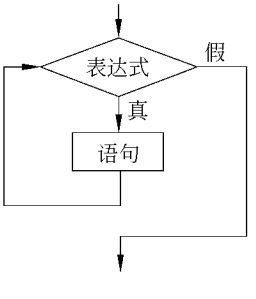

图 3.5　while 结构的流程

【例 3-5】　计算 sum＝1＋2＋3＋4＋…＋100。

程序代码如下：

```
//e3_5.cpp
#include<iostream.h>
void main()
{
    int sum=0, i;
    i=1;
```

```
    while(i<=100)
    {
        sum=sum+i;
        i++;
    }
    cout<<"sum="<<sum<<endl;
}
```

程序运行结果：

```
sum=5050
```

程序说明：

程序中，循环变量 i 初值为 1，循环体语句是复合语句{sum＝sum＋i;i++ ;}，每累加一次，i 的值增 1，为下一次累加做准备。当 i=101 时，跳出循环体，执行 while 循环后面的 cout 语句。

使用 while 循环要注意以下几点。

(1) while 是一个入口条件循环，是否执行循环体在进入循环之前决定的，因此循环体有可能一次都不被执行。

(2) 循环体部分如果是多条语句，必须用花括号将多条语句构成一个复合语句。while 语句在语法上是一条语句，即使使用的是一条复合语句。

3.3.2 do-while 语句

do-while 也是一种循环结构，称为“直到型”循环，其格式为：

```
do
    循环体语句
while(表达式);
```

程序执行时，当流程到达关键字 do 后，立即执行一次循环体，然后对表达式进行判断。若表达式值为“真”，则重复执行循环体语句；否则退出。do-while 结构至少要执行一次循环体。其执行过程如图 3.6 所示。

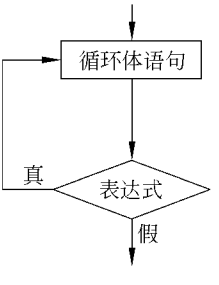

图 3.6 do-while 结构的流程

例 3-5 改写后的程序代码如下。

```
#include<iostream.h>
void main()
{
    int sum=0, i;
    i=1;
    do{
        sum=sum+i;
        i++;
    }while(i<=100);
    cout<<"sum="<<sum<<endl;
}
```

3.3.3 for 循环

C++ 中，for 语句最为灵活，它可以完全替代 while 语句。for 循环格式为：

```
for(表达式 1;表达式 2;表达式 3)
```

循环体语句

执行过程：先执行表达式 1(表达式 1 在整个循环中只执行一次)，接着重复执行下面的过程：计算表达式 2 的值，若其值为“真”，则执行一次循环体语句，然后执行表达式 3；再执行表达式 2，判断其值是否为“真”……直到表达式 2 的值为“假”，跳出循环。执行过程如图 3.7 所示。

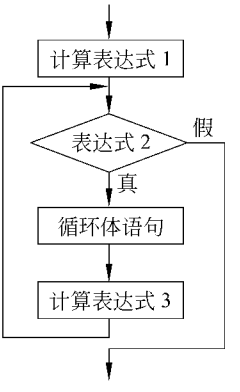

图 3.7　for 结构的流程

例 3-5 改写后的程序代码如下：

```
#include<iostream.h>
void main()
{
    int sum=0, i;
    for(i=1;i<=100;i++)
        sum=sum+i;
    cout<<"sum="<<sum<<endl;
}
```

用 for 结构来表示循环，条理清晰。while 循环变量 i 的修正要在循环体中设置，而 for 结构用表达式 3 即可使循环变量发生改变。

3.3.4　循环的嵌套

一个循环体内包含另一个完整的循环，叫循环的嵌套。其中内层循环还可以再嵌套循环，即多重循环。

while 循环、do-while 循环、for 循环都可以相互嵌套。嵌套时，外层循环执行一次，内层循环执行一个周期。

【例 3-6】 打印九九乘法表。

九九乘法表共 9 行，9 列。先打印 9 行，然后再打印每一行的每一列。用变量 m 控制行数，n 控制列数。

程序代码如下：

```
//e3_6.cpp
#include<iostream.h>
void main()
{
    int m, n;
    for(m=1;m<=9;m++)
    {
        for(n=1;n<=9;n++)
            cout<<m*n<<"\t";
        cout<<endl;
    }
}
```

程序运行结果：

```
1    2    3    4    5    6    7    8    9
2    4    6    8    10   12   14   16   18
3    6    9    12   15   18   21   24   27
```

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

程序说明：

语句 cout << endl;用来在每行的第 9 列后换行。

3.3.5 几种循环的比较

(1) 3 种循环都可以用来处理同一个问题，一般可以互相代替。

(2) while 和 do-while 循环，循环体中应包括使循环趋于结束的语句，否则循环会永远执行下去。前者的循环体可能一次也不执行，后者的循环体至少执行一次。for 语句功能最强。

(3) 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。

3.4 几种控制语句

3.4.1 break 语句

break 语句通常用在循环和 switch 结构中。当 break 用于 switch 结构中时，可使程序跳出 switch 结构而执行 switch 结构以后的语句；当 break 语句用于 do-while、for、while 循环中时，可使程序中途退出循环结构，而执行循环体后面的语句。通常 break 语句总是与 if 语句联在一起使用，即满足条件时便跳出循环。

【例 3-7】 判断一个大于 3 的正整数 m 是否为素数。

判定一个数 m 是否为素数的方法是，看在 2 到 m/2 中能否找到可以整除(余数为 0)m 的数 n。若能找到 n，则 m 不是素数；若找不到，则 m 为素数。用 for 结构穷举 2 到 m/2 之间的数 n，去除 m，若中途找到一个可以整除 m 的 n，则退出循环。

程序代码如下：

```
//e3_7.cpp
#include<iostream.h>
void main()
{
    int m, n;
    cout<<"please input the number m:\n";
    cin>>m;
    for(n=2;n<=m/2;n++)
        if(m%n==0) break;
    if(n>m/2)
        cout<<m<<" is a prime\n";
    else
        cout<<m<<" is not a prime number\n";
}
```

程序运行结果：

```
please enter the number m:
67
67 is a prime number
```

若输入 25，结果为：

```
please enter the number m:
25
25 is not a prime number.
```

程序说明：

break 语句在程序中的作用是跳出循环体，执行 if-else 语句。

3.4.2　continue 语句

continue 语句的作用是跳过本次循环中剩余的语句，强行执行是否执行下一次循环的判定。continue 语句只用在 for、while 和 do-while 等循环体中，常与 if 条件语句一起使用，用来加速循环。

【例 3-8】 打印 3～100 内的素数。

本例基于上例，用一个嵌套的 for 循环处理。

程序代码如下：

```
//e3_8.cpp
#include<iostream.h>
void main()
{
    int m, n;
    cout<<"the prime number is:\n";
    for(m=3;m<100;m+=2)
    {
        for(n=2;n<=m/2;n++)
            if(m%n==0) break;
        if(n<m/2)continue;
            cout<<m<<"\t";
    }
}
```

程序运行结果：

```
3    5    7    11    13    17    19    23    29    31
37   41   43   47    53    59    61    67    71    73
79   83   83   97
```

程序说明：

当找到一个 m 是素数时，要用 break 跳出内层循环，同时外层循环中要跳过输出语句（用 continue 实现），直接进入下一次外层循环（对下一个数进行测试）。

3.4.3　exit 函数和 abort 函数

exit 和 abort 是C++ 的两个库函数，其作用均为中止整个程序的运行。所不同的是，前者在