第3章 选择结构程序设计

选择结构是程序的3种逻辑结构之一，在C语言程序中使用if命令和switch命令实现选择结构。本章系统介绍选择结构程序设计知识，主要内容包括用于表示条件的关系表达式和逻辑表达式、if命令和switch命令的结构及执行过程、选择结构程序设计的基本方法等。

任何选择处理都是有条件的，合理、正确地表达和使用选择条件是选择结构程序设计的重要内容。

3.1 if选择结构

在第1章关于选择结构算法的知识中讨论了判定“优等生”问题的选择结构算法（算法流程图见图1-5），其中分支选择的条件是ave≥90（ave表示平均成绩），该条件成立时显示“优等生”，否则显示“加油!”。本节从此算法的实现程序开始逐步介绍if选择结构的相关知识。

3.1.1 if选择结构程序示例

【例3-1】 输入一个学生的两门课程的成绩，若平均成绩不低于90，则显示“优等生”，否则显示“加油!”。

程序如下：

```
#include<stdio.h>
int main()
{
    int s1,s2,ave;                          /*s1、s2为课程成绩,ave为平均成绩*/
    printf("输入两门课程的成绩: ");
    scanf("%d,%d",&s1,&s2);                 /*输入课程成绩s1、s2*/
    ave=(s1+s2)/2;                          /*计算平均成绩ave*/
    if(ave>=90)                             /*选择控制*/
        printf("优等生\n");                 /*ave不低于90时执行该语句*/
    else
        printf("加油!\n");                  /*ave不足90时执行该语句*/
    return 0;
}
```

程序解析：

该程序中的if-else命令用于实现选择控制，选择条件是ave>=90。当ave>=90成立时

执行语句“printf("优等生\n");”,输出字符串“优等生”;否则执行语句“printf("加油!\n");”,输出字符串“加油!”。本例中决定分支的条件 ave>=90 称为关系表达式。

以下是程序的执行实例,希望读者根据具体数据分析程序的选择控制过程。

程序第 1 次执行结果:

```
输入两门课程的成绩: 88,96 ↵(此时表达式 ave>=90 成立)
优等生
```

程序第 2 次执行结果:

```
输入两门课程的成绩: 77,85 ↵(此时表达式 ave>=90 不成立)
加油!
```

3.1.2 关系表达式

关系表达式是由关系运算符连接运算对象而构成的表达式。在选择结构中,进行分支选择的条件常使用关系表达式。例如,在上述示例程序中使用 ave>=90 作为选择控制条件,其中的>=符号称为关系运算符。

1. 关系运算符

C 语言有 6 种关系运算,分别表示两个对象进行大小比较的 6 种情况,即大于、大于或等于、小于、小于或等于、等于、不等于,其运算符及含义如表 3-1 所示。

表 3-1 关系运算符及其含义

关系运算符	含 义	实 例
>	大于	ave>90
>=	大于或等于	ave>=90
<	小于	ave<90
<=	小于或等于	ave<=90
==	等于	ave==90
!=	不等于	ave!=90

2. 关系表达式的值

关系表达式只有两个取值,或者是 1,或者是 0。当关系表达式所表示的“关系”成立时,其值为 1;否则其值为 0。例如在上述程序中,第 1 次执行程序时变量 ave 的值为 92,关系表达式 ave>=90 成立,则其值为 1;第 2 次执行程序时变量 ave 的值为 81,关系表达式 ave>=90 不成立,则其值为 0。

3. 关系运算的优先级

(1) 关系运算>、>=、<以及<=的优先级相同,关系运算==和!=的优先级相同,前面一组运算符的优先级高于后面一组运算符的优先级。

(2) 关系运算的优先级低于算术运算的优先级。

(3) 关系运算的优先级高于赋值运算的优先级。

4. 关系运算的结合性

6 种关系运算都是左结合的。例如关系表达式 a<5>b 与(a<5)>b 等价,若 a=-2、

b=2,则其值为 0。

3.1.3 逻辑表达式

逻辑表达式是由逻辑运算符连接运算对象而构成的表达式,它在程序中常用于表示复杂条件。例如上述“优等生”问题,如果要求每门课程的成绩都不低于 90 时判定为“优等生”,那么程序中判断优等生的条件就要发生变化,满足优等生的条件是关系表达式 s1>=90 与 s2>=90 都要成立。以下是该复杂条件的逻辑表达式,其中 && 称为逻辑与运算。

```
s1 >= 90&&s2 >= 90
```

1. 逻辑运算符

C 语言的逻辑符有 3 个,分别为逻辑与运算符“&&”、逻辑或运算符“||”以及逻辑非运算符“!”。

当两个条件为“并且”关系时,使用与运算“&&”表示。实例如上。

当两个条件为“或者”关系时,使用或运算“||”表示。例如关于上述优等生问题,若任意一门课程的成绩不低于 90,即判定为“优等生”,则判定“优等生”的逻辑表达式如下。

```
s1 >= 90||s2 >= 90
```

当对某个条件进行否定时使用非运算“!”表示。例如,表达式 s1>=90 可用如下“!”运算表示。

```
!(s1 < 90)
```

2. 逻辑表达式的值

逻辑表达式只有 1 和 0 两个取值。当逻辑表达式所表示的条件成立时(条件为“真”),其值为 1;否则(条件为“假”)其值为 0。设 a、b 为表示条件的表达式,则对应于 a、b 的各种取值时,逻辑表达式!a、a&&b 和 a||b 的结果值如表 3-2 所示。该表称为逻辑运算真值表,它清楚地描述了逻辑运算 &&、||、!的功能。

表 3-2 逻辑运算真值表

a	b	!a	a&&b	a||b
1(真)	1(真)	0(假)	1(真)	1(真)
1(真)	0(假)		0(假)	1(真)
0(假)	1(真)	1(真)	0(假)	1(真)
0(假)	0(假)		0(假)	0(假)

【例 3-2】 将数学关系式 20<x≤100 用逻辑表达式表示,并计算 x=50 时逻辑表达式的值。

求解:

(1) 将数学关系式表示为逻辑表达式。

数学关系式 20<x≤100 表示变量 x 的取值范围为 x>20,而且 x≤100,使用逻辑与运算 && 可描述 x 的取值关系如下:

```
x > 20&&x <= 100
```

(2) 逻辑表达式求值。

当 x=50 时,表达式 x>20 为 1、表达式 x<=100 为 1,则逻辑表达式 x>20&&x<=100 的值为 1。

3. 逻辑运算符的优先级和结合性

(1) 优先级:"!"高于"&&","&&"高于"‖"。

(2) 优先级:"!"高于算术运算符,"&&""‖"低于关系运算符。

(3) 结合性:"&&""‖"是左结合的,"!"是右结合的。

4. 逻辑运算的特点

(1) 由与运算"&&"构成的逻辑表达式自左至右求值,若运算符"&&"前端表达式的值为 0,则对其后端不再运算,整个与运算表达式的值为 0。例如对逻辑表达式 a&&b 求值,若 a 为 0,则表达式的值为 0,对 b 不再求值;若 a 为非 0,则要计算 b 的值。

(2) 由或运算"‖"构成的逻辑表达式自左至右求值,若运算符"‖"前端表达式的值为 1,则对其后端不再运算,整个或运算表达式的值为 1。例如对逻辑表达式 a‖b 求值,若 a 为 1,则表达式的值为 1,对 b 不再求值;若 a 为 0,则要计算 b 的值。

3.1.4 if 命令

if 命令是最基本的分支控制命令,在具体应用中有多种不同的使用形式,但不管何种形式,都要首先判断给定的条件,然后决定下一步要执行程序的哪些语句。

1. 双分支 if 命令

双分支 if 命令的一般形式如下:

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

其中,"表达式"是 if 命令进行分支处理的条件;"语句组"是若干个 C 语句,当它只有一个语句时大括号"{}"可以省略。

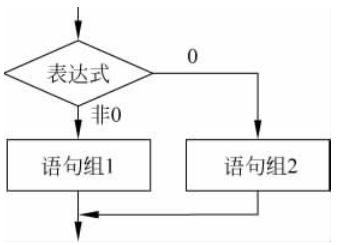

图 3-1 双分支 if 命令流程图

双分支 if 命令的执行过程如下:

首先对"表达式"求值,然后进行分支判断。若"表达式"为非 0 值(即条件成立),则执行{语句组 1},然后执行紧接{语句组 2}之后的语句;否则(即条件不成立)执行{语句组 2},然后继续向下执行其他语句。分支控制过程的流程图如图 3-1 所示。

【例 3-3】 计算下面分段函数的值(关系表达式作为选择条件)。

$$y=\begin{cases} x+25 & (x>0) \\ x-25 & (x\leqslant 0) \end{cases}$$

1) 算法设计

用流程图描述算法,如图 3-2 所示。

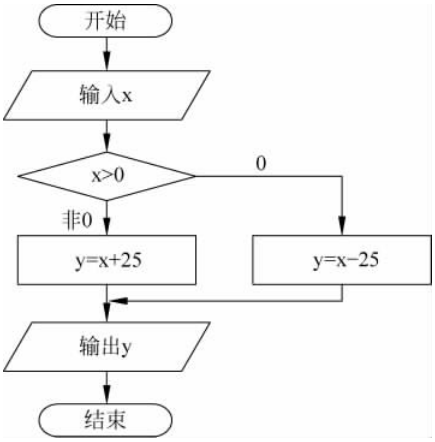

图 3-2　例 3-3 算法流程图

2）实现程序

程序如下：

```
#include<stdio.h>
int main()
{
    int x,y;
    printf("x=");
    scanf("%d",&x);                    /*输入 x 的值*/
    if(x>0)                            /*判断 x>0 是否成立*/
        y=x+25;                        /*x>0 成立时*/
    else
        y=x-25;                        /*x>0 不成立时*/
    printf("y= %d\n",y);               /*输出 y 的值*/
    return 0;
}
```

【例 3-4】 输入一个学生的两门课程的成绩，若每门课程的成绩都不低于 90，则显示“优等生”，否则显示“加油!”（逻辑表达式作为选择条件）。

程序如下：

```
#include<stdio.h>
int main()
{
    int s1,s2;
    printf("输入课程成绩: ");
    scanf("%d, %d",&s1,&s2);
    if(s1>=90&&s2>=90)
        printf("优等生\n");            /*每门课程的成绩都不低于 90*/
    else
        printf("加油!\n");            /*至少有一门课程的成绩不足 90*/
    return 0;
}
```

在该程序中，逻辑表达式 s1>=90&&s2>=90 作为 if 命令的选择条件，只有该条件成立时才会显示“优等生”。程序的算法流程图如图 3-3 所示。

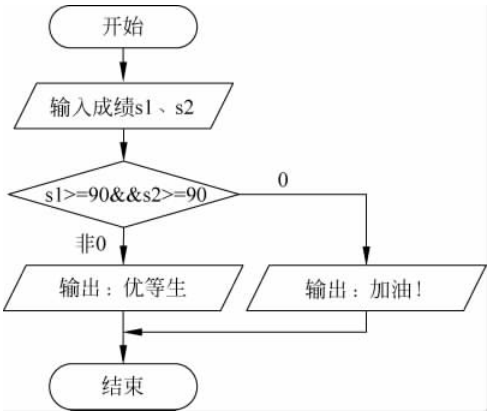

图 3-3　例 3-4 算法流程图

以下是程序的执行实例,希望读者根据具体数据,对 if 命令的分支控制过程进行分析。

第 1 次执行结果:

```
输入课程成绩: 88,96 ↵(此时表达式 s1 >= 90&&s2 >= 90 不成立)
加油!
```

第 2 次执行结果:

```
输入课程成绩: 95,90 ↵(此时表达式 s1 >= 90&&s2 >= 90 成立)
优等生
```

【例 3-5】 从键盘输入一个字符,若其为大写英文字母,则在屏幕上输出它的小写形式,否则原样输出该字符。

1) 问题分析

(1) 输入字符后首先要判断它是否为大写字母。设输入量为 ch,则下面的逻辑表达式成立(其值为 1)时 ch 为大写字母:

```
ch >= 'A'&&ch <= 'Z'
```

(2) 当 ch 为大写字母时,其对应的小写字母的 ASCII 码值为 ch+32,以%c 格式输出该表达式,即显示为小写字母。

2) 算法设计

(1) 输入字符并存储到 ch 中。

(2) 判断逻辑表达式 ch>='A'&&ch<='Z',若其成立,则以%c 格式输出 ch+32,否则输出 ch。

3) 实现程序

程序如下:

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

```
    ch = getchar();                          /* 输入字符 */
    printf("Output: ");
    if(ch >= 'A'&&ch <= 'Z')                 /* 判断 ch 是否为大写字母 */
        printf(" %c\n",ch + 32);             /* ch 为大写字母时输出其小写形式 */
    else
        printf(" %c\n",ch);                  /* ch 不是大写字母时原样输出 */
    return 0;
}
```

下面是程序两次运行的结果：

```
Input: T↵
Output: t
Input:example↵
Output:e
```

● 问题思考

在上面的运行结果中，当输入字符串 example 时输出结果为 e。为什么在输入一个字符串时只对其第 1 个字符进行输出处理？

2. 单分支 if 命令

单分支 if 命令是双分支 if 命令的一种简化结构，其一般形式如下：

```
if(表达式)
  {语句组}
```

其中，“表达式”和“语句组”的含义与在双分支 if 命令中的含义相同。

单分支 if 命令被执行后，首先对“表达式”求值，若表达式的值为非 0（即条件成立），则执行{语句组}，然后继续向下执行其他语句；否则不执行{语句组}，而直接执行{语句组}之下的语句。简而言之，该 if 命令的功能是根据条件表达式的值决定是否执行{语句组}，其控制过程的流程图如图 3-4 所示。

【例 3-6】 输入一个学生的两门课程的成绩，若平均成绩不低于 90，则显示“优等生”。

1）算法设计

这是例 3-1 的一个简化问题，算法流程图如图 3-5 所示。

2）实现程序

程序如下：

```
#include <stdio.h>
int main()
{
    int s1,s2,ave;
    printf("输入两门课程的成绩: ");
    scanf(" %d, %d",&s1,&s2);                /* 输入课程成绩 s1、s2 */
    ave = (s1 + s2)/2;                       /* 计算平均成绩 ave */
    if(ave >= 90)                            /* 判断 ave >= 90 是否成立 */
        printf("优等生\n");                  /* ave >= 90 成立时执行该语句 */
    return 0;
}
```

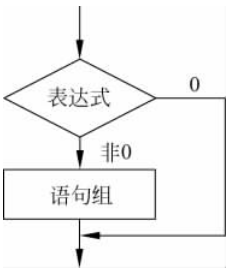

图 3-4　单分支 if 命令流程图

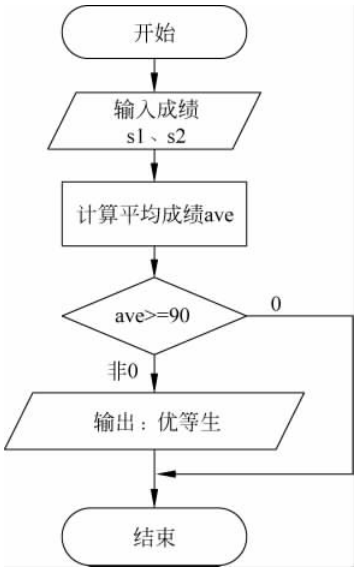

图 3-5　例 3-6 算法流程图

执行程序时，若输入的成绩数据满足优等生的条件，则显示“优等生”，否则立即结束，不会显示任何信息。

3. if 命令的嵌套结构

当一个 if 命令的{语句组}内又使用了 if 命令时就形成了 if 命令的嵌套结构，这种结构用于多重条件判断的情况。图 3-6 是 if 命令嵌套结构示意图，该图只表示了嵌套的两种情况。

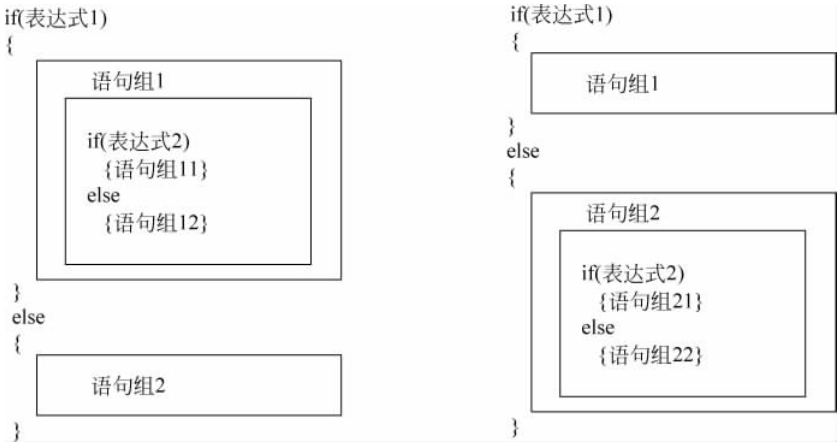

图 3-6　if 命令嵌套结构示意图

【例 3-7】 输入一个学生的两门课程的成绩，若平均成绩小于 0，则显示“数据错误!”；否则，若平均成绩不低于 90，显示“优等生”，低于 90 则显示“加油!”。

1) 算法设计

根据平均成绩的计算结果，问题处理将有以下两个大的分支。

分支一：平均成绩小于 0，显示“数据错误!”；

分支二：平均成绩不小于 0，进一步进行小分支处理。

算法流程图如图 3-7 所示。对照流程图，先给出具体的实现程序，然后对 if 命令的嵌套情况进行说明。希望读者能够借助程序流程图加深对 if 命令嵌套结构的认识。

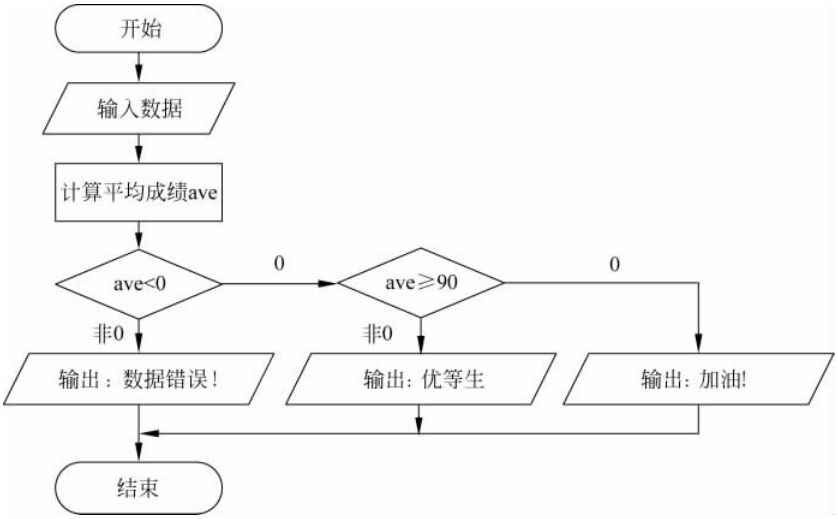

图 3-7　if 命令的嵌套结构举例

2）实现程序

程序如下：

```
#include<stdio.h>
int main()
{
    int s1,s2,ave;
    printf("输入两门课程的成绩: ");
    scanf("%d,%d",&s1,&s2);                    /*输入两门课程的成绩*/
    ave=(s1+s2)/2;                             /*计算平均成绩,存储到变量 ave 中*/
①   if(ave<0)                                  /*外层 if*/
②       printf("数据错误!\n");
③   else                                       /*与外层 if 配对*/
④       if(ave>=90)                            /*内层 if*/
⑤           printf("优等生\n");
⑥       else                                   /*与内层 if 配对*/
⑦           printf("加油!\n");
    return 0;
}
```

执行结果：

```
输入两门课程的成绩: 70,-90↵
数据错误!
```

再次执行：

```
输入两门课程的成绩: 90,95↵
```

```
优等生
```

再次执行：

```
输入两门课程的成绩: 70,65 ↵
加油!
```

为便于说明程序嵌套结构，部分语句加了编号，这些编号信息不属于源程序的内容。程序中有两个 if 命令，①和③构成外层的 if 命令，②位于外层 if 命令的{语句组 1}内，④、⑤、⑥、⑦位于外层 if 命令的{语句组 2}内，④和⑥构成的 if 命令嵌套在外层 if 命令中。

使用嵌套的 if 命令首先应避免出现嵌套混乱。C 语言规定，在 else 语句无明确配对结构时，else 与其前最近的一个尚未配对的 if 配对。下面是两个程序段，可以进一步说明 else 和 if 的配对情况。

程序段一：

```
if(x>20)
{
    if(y>100)
        printf("Good");
}
else
    printf("Bad");
```

程序段二：

```
if(x>20)
    if(y>100)
        printf("Good");
    else
        printf("Bad");
```

程序段一整体上是一个 if-else 的结构，else 与第 1 个 if 配对。程序段二整体上是一个单分支 if 语句，else 与第 2 个 if 配对，共同作为第 1 个 if 命令的语句体。对于这两个程序段的执行结果，希望读者分析并进行验证。

4. if-else if 结构

if-else if 结构属于 if-else 结构的嵌套形式，它的一般结构如下：

```
if(表达式 1)
    {语句组 1}
else if(表达式 2)
    {语句组 2}
else if(表达式 3)
    {语句组 3}
     ⋮
else if(表达式 n)
    {语句组 n}
else
    {语句组 n+1}
```

该结构中的每一个“表达式”都是一个“条件”，该结构的功能是根据表达式的值决定是否执行它所属的语句组。if-else if 结构的具体执行过程为从上到下逐个对条件进行判断，一旦条件满足就执行与它有关的语句组，其下的所有条件都不再判断，当然它们的语句组也不被执行；当任何一个条件都不满足时执行{语句组 n+1}。图 3-8 所示为 4 层 if 结构的流程图。

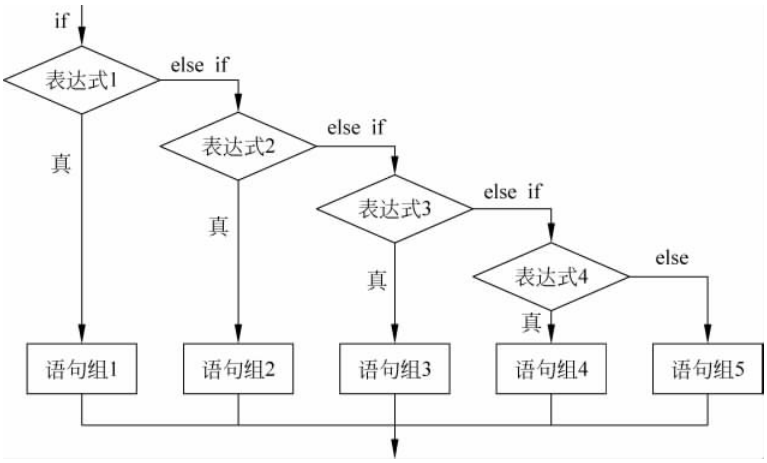

图 3-8　if-else if 语句的逻辑结构

【例 3-8】 用 if-else if 结构改写例 3-7 的程序。

程序如下：

```
#include<stdio.h>
int main()
{
    int s1,s2,ave;
    printf("输入两门课程的成绩: ");
    scanf("%d,%d",&s1,&s2);
    ave=(s1+s2)/2;
    if(ave<0)
        printf("数据错误!\n");          /*ave<0 成立时*/
    else if(ave>=90)
        printf("优等生\n");             /*ave<0 不成立,而 ave>=90 成立时*/
    else
        printf("加油!\n");              /*ave<0 不成立,ave>=90 也不成立时*/
    return 0;
}
```

希望读者对照例 3-7 的程序，通过具体数据分析该程序的执行过程。

3.1.5　条件运算

条件运算是 C 语言中唯一的一个三目运算，运算符由“?”和“:”构成，它根据条件从两个表达式中选择一个进行计算取值。有些简单的 if-else 结构可通过条件运算实现。

1. 条件运算表达式

由条件运算符构成的表达式称为条件运算表达式，其一般形式如下。

```
表达式 1?表达式 2:表达式 3
```

例如：

```
5?19 + 6:21
```

条件运算表达式按以下过程求值：

(1) 计算“表达式 1”的值；

(2) 当“表达式 1”的值为非 0 时，取“表达式 2”的值为条件运算表达式的值，否则取“表达式 3”的值为条件运算表达式的值。

由此可以求得上面的条件运算表达式的值为 25。

2. 条件运算的优先级和结合性

条件运算的优先级高于赋值运算，而低于关系运算。

【例 3-9】 用条件运算计算下面分段函数的值。

$$y=\begin{cases} x+25 & (x>0) \\ x-25 & (x\leqslant 0) \end{cases}$$

1) 问题分析

使用 x>0 作为计算 y 值的条件，即可得到计算 y 值的条件运算表达式。

```
y = (x > 0)?(x + 25):(x - 25)
```

或者：

```
y = x > 0?x + 25:x - 25
```

当然，也可以使用 x≤0 作为计算 y 值的条件，具体表达式由读者分析给出。

2) 算法设计

(1) 输入 x；

(2) 计算表达式(x>0)?(x+25):(x-25)的值，并存储在 y 变量中；

(3) 输出 y。

3) 实现程序

程序如下：

```
#include <stdio.h>
int main()
{
    int x,y;
    printf("x = ");
    scanf(" %d",&x);                    /* 输入 x */
    y = (x > 0)?(x + 25):(x - 25);      /* 计算(x > 0)?(x + 25):(x - 25),存储在 y 中 */
    printf("y = %d\n",y);               /* 输出 y */
    return 0;
}
```

本节例 3-3 中，该分段函数的求值是通过 if 命令判断来实现的，希望读者注意比较这两个程序的异同。

3.2 switch 选择结构

switch 选择结构是多分支选择的常见形式，由 switch 命令实现，具有结构清晰的特点。switch 命令的一般格式如下：

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

switch 命令的执行过程如下：

首先计算 switch 表达式的值，然后从第 1 个 case 行开始，由上至下依次扫描 case 行，比较 case 行中的常量值与 switch 表达式的值是否相同，一旦遇到相同的情况，即停止扫描过程，并从该 case 行的语句组开始，依次向下执行各语句组，直至遇到强制中断命令 break 或执行完最后一个语句组为止。当所有 case 都不符合要求时执行 default 下的语句组。

在扫描 case 行时，case 的语句组不起任何作用；开始执行语句后，其下的所有 case 行以及 default 行都被忽略掉，不再有任何作用。图 3-9 所示为 switch 命令执行过程的示意图。

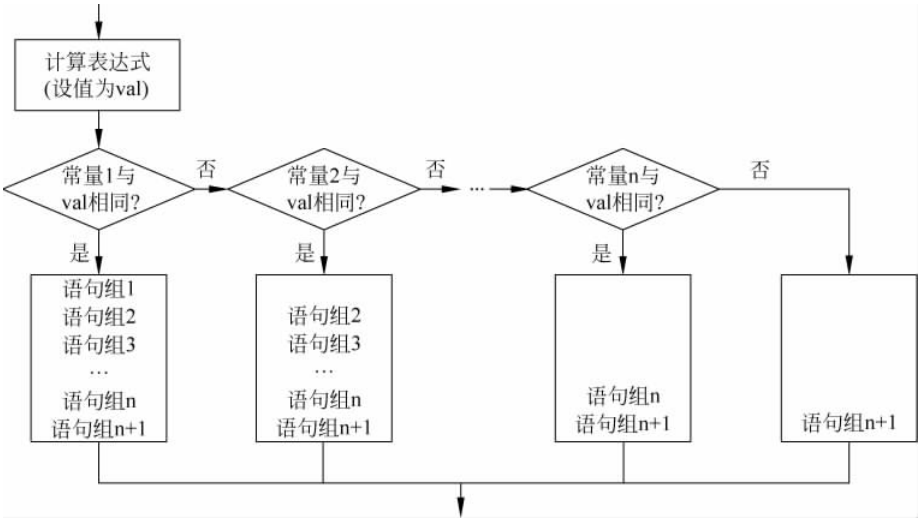

图 3-9　switch 命令执行过程示意图

使用 switch 命令时允许省略 default：及其语句组。

【例 3-10】 switch 执行过程示例程序。

程序如下:

```
#include<stdio.h>
int main()
{
    int i;
    scanf("%d",&i);                    /*为变量 i 输入数据*/
    switch(i)
    {
        case 0:
            printf("zero ");           /*i 为 0 时输出*/
        case 1:
            printf("one ");            /*i 为 0 或 1 时输出*/
            break;                     /*终止 switch 语句*/
        case 2:
            printf("two ");            /*i 为 2 时输出*/
        case 3:
            printf("three ");          /*i 为 2 或 3 时输出*/
        case 4:
            printf("four ");           /*i 为 2、3 或 4 时输出*/
            break;                     /*终止 switch 语句*/
        default:
            printf("other ");          /*i 不为 0、1、2、3、4 时输出*/
    }
    printf("\n");
    return 0;
}
```

分别用 0、1、3、6 的值为 i 提供数据,执行 4 次程序,结果如下:

i=0 时,输出 zero one

i=1 时,输出 one

i=3 时,输出 three four

i=6 时,输出 other

程序中的 break 为中断命令,其功能是终止 switch 命令,使程序立即执行 switch 命令的后续语句,即“printf("\n");”语句。

下面是关于 switch 命令的其他几点说明。

(1) 任何一个 case 的语句组允许为空。当某个 case 的语句组为空时,表示它与下面的 case 执行相同的语句组。

下面是一个示例程序。

【例 3-11】 省略 case 语句组的程序举例。

程序如下:

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

```
int main()
{
    char result;
    scanf (" %c",&result);                  /* 为变量 result 输入字符 */
    switch(result)
    {
        case 'A':
        case 'B':
        case 'C':
        printf ("Good!\n");                 /* result 为 A、B 或 C 时执行该语句 */
            break;                          /* 终止 switch 语句 */
        case 'D':
        case 'E':
            printf ("Bad!\n");              /* result 为 D 或 E 时执行该语句 */
            break;                          /* 终止 switch 语句 */
        default:
            printf ("Error!\n");            /* result 不是 A、B、C、D、E 时执行该语句 */
    }
    return 0;
}
```

该程序执行后，输入 A 或 B 或 C 时显示"Good!"；输入 D 或 E 时显示"Bad!"；其他输入则显示"Error!"。

(2) switch 命令的"表达式"通常为整数型值或字符型值，case 中常量的类型应与之相应。

(3) case 中的"常量"位置允许是常数表达式，但不允许是变量表达式。

以下用法是正确的：

```
case 5 + 8:
case 'a' + 6:
```

以下用法是错误的：

```
case a + 5:
```

(4) switch 命令允许嵌套，即在 case 的语句组中允许使用 switch 命令。

3.3 选择结构程序举例

【例 3-12】 编写一个程序，根据公历历法的闰年规律判定某个年份是否为闰年。

1) 问题分析与算法设计

闰年是为了弥补因人为历法造成的年度天数与地球实际公转周期的时间差而设立的，补上时间差的年份即为闰年。地球绕太阳运行的周期约为 365.242 19 天，即一回归年。公历的平年只有 365 天，比回归年约短 0.242 19 天，每四年累计约一天，把这一天加于 2 月末(2 月 29 日)，使当年延长为 366 天，这一年就为闰年。

按照每四年一个闰年计算，平均每年就要多算出 0.0078 天，经过四百年就会多出大约

3 天,因此每四百年中要减少 3 个闰年。所以规定公历年份是整百数的,必须是 400 的倍数才是闰年,不是 400 的倍数的就是平年。例如 1700 年、1800 年和 1900 年为平年,2000 年为闰年。

简而言之,公历历法闰年的遵循规律为四年一闰,百年不闰,四百年再闰。

(1) 设用变量 year 表示年份,写出满足闰年条件的逻辑表达式。

当 year 是 400 的整倍数时为闰年,条件表示为:

```
year % 400 == 0
```

当 year 是 4 的整倍数,但不是 100 的整倍数时为闰年,条件表示为:

```
year % 4 == 0&&year % 100!= 0
```

对于年份 year,满足上述任何一个条件均为闰年。因此,满足闰年条件的逻辑表达式如下:

```
year % 400 == 0 || year % 4 == 0&&year % 100!= 0
```

当上述表达式成立时即为闰年。

(2) 输入 year,根据上述逻辑表达式的值即可得到 year 是否为闰年的结论。

其算法流程图如图 3-10 所示。

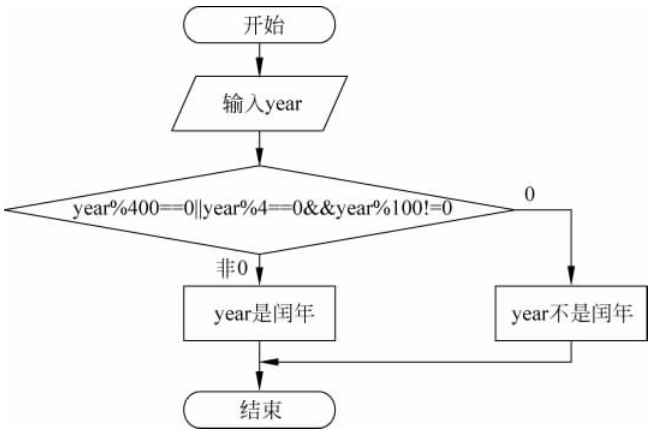

图 3-10　闰年问题算法流程图

2) 实现程序

程序如下:

```
#include<stdio.h>
int main()
{
    int year;                                          /*定义存储年份值的变量*/
    printf("Input year: ");
    scanf("%d",&year);                                 /*输入年份值*/
    if(year % 400 == 0 || year % 4 == 0&&year % 100!= 0)  /*判断闰年*/
        printf("%d is a leap year.\n",year);           /*是闰年时*/
    else
```

```
        printf(" %d is not a leap year.\n",year);     /* 不是闰年时 */
    return 0;
}
```

执行结果：

```
Input year: 1995 ↵
1995 is not a leap year.
```

再次运行

```
Input year: 2000 ↵
2000 is a leap year.
```

该程序主要是学习在 if 语句中使用综合条件的方法，将多种条件组合成逻辑表达式可以简化程序设计。

● 问题思考

下面的程序也能对闰年年份进行判断，功能与上面的程序相同，但程序更简洁，它将 printf()函数和条件运算结合起来，巧妙地解决了闰年判断问题。希望读者运行程序，并结合程序执行结果对程序进行解读分析。

```
#include<stdio.h>
int main()
{
    int year;
    printf("year = ");
    scanf(" %d",&year);
    printf(" %s\n",year % (year % 100?4:400)?"NO":"YES");
    return 0;
}
```

【例 3-13】 设计求解一元二次方程 $ax^2+bx+c=0(a\neq0)$的通用程序，并运行程序，对下面的两个方程求解。

方程一：　　$3x^2+9x-1=0$

方程二：　　$2x^2-4x+3=0$

1）问题分析与算法设计

（1）一元二次方程若有实根，则计算并输出实根，$x_{1,2}=\dfrac{-b\pm\sqrt{b^2-4ac}}{2a}$；否则输出无实根信息。

（2）程序的输入量为方程的系数 a、b、c。输入不同的系数，则求解不同的方程。

（3）程序中要使用数学函数 sqrt()，因此注意打开 math.h 文件。

程序的算法流程图如图 3-11 所示。

2）实现程序

程序如下：

```
#include<stdio.h>
#include<math.h>
int main()
```

```
{
    float a,b,c;                                /*定义存储方程式系数的变量*/
    float x1,x2,d;                              /*x1、x2 存储方程根,d 存储判别式的值*/
    printf("Input a,b,c: ");
    scanf("%f,%f,%f",&a,&b,&c);                 /*输入方程式的系数值*/
    d=b*b-4.0*a*c;                              /*计算判别式的值*/
    if(d>=0.0)                                  /*当方程有实根时求方程的两个实根*/
    {
        x1=(-b+sqrt(d))/(2.0*a);                /*计算 x1*/
        x2=(-b-sqrt(d))/(2.0*a);                /*计算 x2*/
        printf("x1=%f,x2=%f\n",x1,x2);          /*输出 x1、x2*/
    }
    else                                        /*当方程无实根时输出无实根信息*/
        printf("no real root.\n");
    return 0;
}
```

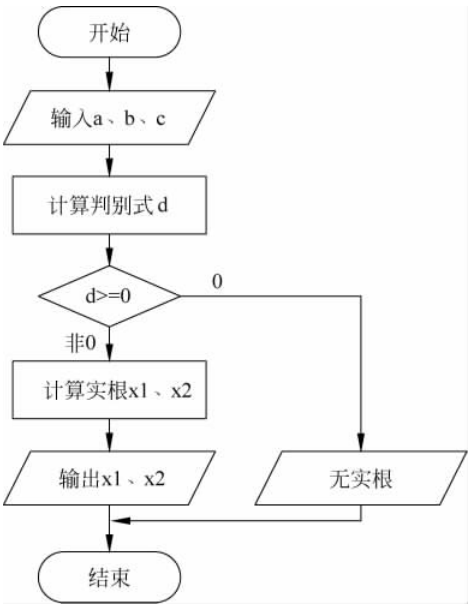

图 3-11　求解一元二次方程的算法流程图

下面是求解方程一的执行结果：

```
Input a,b,c: 3,9,-1↵(输入方程一的系数)
x1=0.107275,x2=-3.107175
```

下面是求解方程二的执行结果：

```
Input a,b,c: 2,-4,3↵(输入方程二的系数)
no real root.
```

【例 3-14】 学生成绩分等级显示。某班学生有两门课程,按百分制成绩(无小数位)进行考核。要求输入一个学生的两门课程的成绩,然后按平均成绩分等级显示考核结果。考

核结果的等级标准如下。

优秀(excellence)：平均成绩≥90；

良好(all right)：80≤平均成绩<90；

中等(middling)：70≤平均成绩<80；

及格(pass)：60≤平均成绩<70；

不及格(fail)：平均成绩<60。

1) 问题分析与算法设计

实现学生成绩分等级显示的基本处理过程可以分成两个阶段。

(1) 输入成绩并计算平均成绩。

(2) 分等级显示。成绩分为 5 个等级，每个等级对应不同的条件。类似的多分支问题通常使用 if-else if 结构或者 switch 结构进行逻辑控制。本例使用 if-else if 结构，算法流程图如图 3-12 所示。

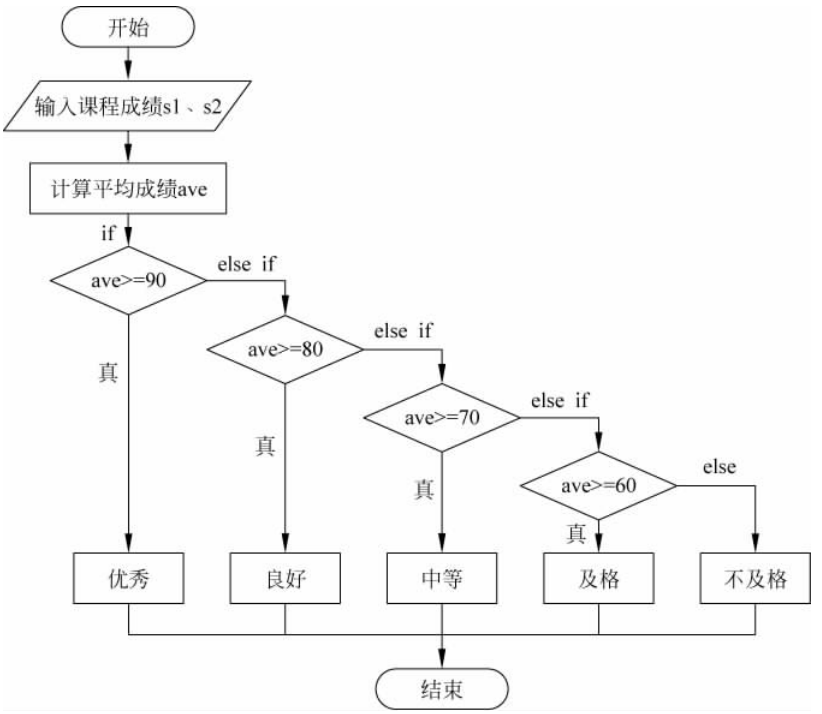

图 3-12 “学生成绩分等级显示”算法流程图

2) 实现程序

程序如下：

```
#include<stdio.h>
int main()
{
    int s1,s2,ave;                          /*s1、s2 为课程成绩,ave 为平均成绩*/
    printf("Score: ");
    scanf("%d,%d",&s1,&s2);                 /*输入课程成绩*/
```

```
    ave = (s1 + s2)/2;                          /* 计算平均成绩 */
    if(ave >= 90)
        printf("Result: excellence\n");         /* 优秀 */
    else if(ave >= 80)
        printf("Result: all right\n");          /* 良好 */
    else if(ave >= 70)
        printf("Result: middling\n");           /* 中等 */
    else if(ave >= 60)
        printf("Result: pass\n");               /* 及格 */
    else
        printf("Result: fail\n");               /* 不及格 */
    return 0;
}
```

程序执行结果:

```
Score: 77,98 ↵(此时 ave 为 87)
Result: all right
Score: 89,92 ↵(此时,ave 为 90)
Result: excellence
```

● 问题思考

(1) 对输入数据进行合法性检查是算法评价的一项重要指标(算法的健壮性)。上面的程序未检查输入数据的合法性,即使输入了百分制以外的成绩值(例如 900、−90 等),也会进行等级判定,这显然是不合理的。希望读者完善程序,使得只有输入数据合法时才会进行等级判定,输出相应结果。

(2) "学生成绩分等级显示"是一个多分支处理问题,上面的程序使用 if-else if 结构实现分支控制。类似的多分支处理问题使用 switch 结构也能方便地实现。希望读者使用 switch 结构改写上面的程序。

小　　结

本章介绍了选择结构知识,主要有选择结构控制命令 if 和 switch、用于条件表达的关系表达式和逻辑表达式,以及选择结构程序的典型实例。

(1) if 命令是最基本的选择结构控制命令,它有多种形式,通常用关系表达式和逻辑表达式表示选择控制条件。任何一种 if 命令的语句体中都可以出现其他的 if 结构,这种结构称为 if 命令的嵌套结构。

(2) switch 命令专门用于多路分支控制,适用于 if-else if 形式的结构,而且更清晰。程序总是试图从满足条件的第 1 个 case 子句开始执行其后的所有语句,而不再对其后的 case 进行判断,因此必要时使用 break 命令中断 switch 命令的运行。

(3) 本章最后通过闰年问题、求解一元二次方程和学生成绩分等级显示等典型实例详细介绍了选择结构程序设计的方法和过程。

习　题　三

一、选择题

1. 下面是由 if 构成的一个程序段：

```
if(a < b)
{   if(d == c)x = 1; }
else
    x = 2;
```

该程序段所表示的逻辑关系对应的表达式是_____。

A. $x=\begin{cases}1 & (a<b \text{ 且 } c=d)\\ 2 & (a\geqslant b \text{ 且 } c\neq d)\end{cases}$　　B. $x=\begin{cases}1 & (a<b \text{ 且 } c=d)\\ 2 & (a<b \text{ 且 } c\neq d)\end{cases}$

C. $x=\begin{cases}1 & (a<b \text{ 且 } c=d)\\ 2 & (c\neq d)\end{cases}$　　D. $x=\begin{cases}1 & (a<b \text{ 且 } c=d)\\ 2 & (a\geqslant b)\end{cases}$

2. 以下程序段的运行结果为_____。

```
int x = 2,y = - 1,z = 2;
if(x < y)                    /* 第 1 个 if */
  if(y < 0)z = 0;            /* 第 2 个 if */
  else z += 1;
printf(" %d\n",z);
```

A. 3　　B. 2　　C. 1　　D. 0

3. 有程序段如下：

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

执行该程序段后变量 a、b、c 的值是_____。

A. 1、2、3　　B. 2、3、3

C. 2、3、1　　D. 2、3、2

4. 执行下面的程序段后 a 和 b 的值分别为_____。

```
int a = 3,b = 5,c;
c = (a > -- b)?a++ :b -- ;
```

A. 3、2　　B. 3、3　　C. 4、4　　D. 4、5

5. 以下程序段的输出结果是_____。

```
int x,y,temp;
x = 1,y = 2;
if(1)
{
   if(x < y)
  {
      temp = x;
```

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

A. x=1,y=1　　B. x=2,y=2

C. x=1,y=2　　D. x=2,y=1

6. 程序段如下：

```
int x,y,z,max;
x = 1,y = 2,z = 3;
max = x;
if(z > y)
{
    if(z > x)
        max = z;
  }
  else
      if(y > x)
          max = y;
  printf("max = %d\n",max);
```

执行该程序段后的输出结果为_____。

A. max=1　　B. max=2

C. max=3　　D. 不确定

7. 有变量定义如下：

```
char ch = 'A';
```

则下列表达式的值是_____。

```
ch = (ch>'A'&& ch<= 'Z')?(ch + 32):ch;
```

A. A　　B. a　　C. Z　　D. z

8. 有程序段如下：

```
int year = 2000;
printf(" %s\n",year % (year % 100?4:400)?"NO":"YES");
```

执行上面的 printf()语句后输出结果是_____。

A. NO　　B. YES

C. NO:YES　　D. 不确定

9. 有语句如下：

```
int n;
scanf(" %d",&n);
```

要求当 n 是奇数时将其显示输出。以下语句中符合要求的是_____。

A. if(n%2==0)printf("%d\n",n);

B. if(n%2)printf("%d\n",n);

C. if(n%2==1)printf("%d\n",n);

D. if(n/2==1)printf("%d\n",n);

10. 有程序段如下：

```
int flag;
char ch;
scanf(" %c",&ch);
flag = ch >= '0'&&ch <= '9';
if(flag)
    printf(" %c\n",ch);
```

以下关于程序段执行结果的叙述中正确的是_____。

A. 当输入一个字符时立即将该字符输出

B. 对输入的任何数值立即输出

C. 若输入一个数字字符，则立即将其输出

D. 若输入的字符是非数字字符，则将其输出

二、编程题

1. 按照图 3-13 所示的流程图编写程序，并指出程序的功能。

2. 按照图 3-14 所示的流程图编写程序，并指出程序的功能。

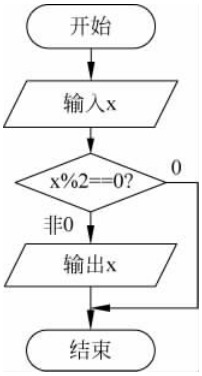

图 3-13　编程题 1 算法的流程图

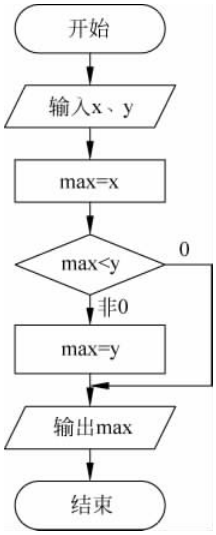

图 3-14　编程题 2 算法的流程图

3. 计算邮费，邮件的重量由键盘输入。邮件的计费标准为不超过 100 克时每件 10 元，超过 100 克时超出部分每克计费 0.5 元。

4. 按照货物重量计算运费并输出结果。物流公司按照货物重量分段计费，标准如下：

货物重量不超过 50 吨时运费为 80 元/吨，货物重量超过 50 吨但不超过 100 吨时超出部分运费为 75 元/吨，货物重量超过 100 吨时超出部分运费为 70 元/吨。

5. 求分段函数 y 的值，其中 x 的值由键盘输入。

$$y=\begin{cases}x & (x\leqslant 0)\\ 2x & (0<x<1)\\ 3x^2-6x+7 & (x\geqslant 1)\end{cases}$$

6. 输入 3 个整数,然后按由大到小的顺序输出这 3 个数。

7. 由键盘输入一个整数,判断其能否既被 3 整除又被 5 整除。

8. 编写程序,其功能是输入 1、2、3、4、5、6、7 中的任何一个数字,将对应显示 Monday、Tuesday、Wednesday、Thursday、Friday、Saturday、Sunday; 输入其他数字,则显示 No!。

9. 编写完整求解一元二次方程 $ax^2+bx+c=0$ 的程序。

10. 由键盘输入一个字符,判断它是字母、数字还是其他符号。

11. 输入一个不多于 3 位的正整数,编写程序,实现以下功能:

(1) 求出它是几位数。

(2) 按逆序打印出各位数字,例如原数为 321,应输出 123。