第 3 章 选择结构程序设计

选择结构是程序的三种逻辑结构之一，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)                 /* 分支控制 */
        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。

程序第一次执行结果：

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

程序第二次执行结果：

```
输入两门课的成绩：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、大于等于、小于、小于等于、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
>=	大于等于	ave>=90
<	小于	ave<90
<=	小于等于	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，第一次执行程序时，变量 ave 的值为 92，关系表达式 ave>=90 成立，则其值为 1；第二次执行程序时，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）关系运算符的优先级低于算术运算的优先级。

（3）关系运算符的优先级高于赋值运算的优先级。

4．关系运算的结合性

所有 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 语言的逻辑符有三个，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，并计算当 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 表示的是变量 x 的取值范围：x>20，并且 x≤100，使用逻辑与运算&&可描述 x 的取值关系：

```
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 时，则不再对其后端进行运算，整个与运算表达式的值为 0。例如，逻辑表达式 a&&b 求值时，若 a 为 0，则表达式的值为 0，对 b 不再求值；若 a 非 0，则需计算 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 时，则不再对其后端进行运算，整个或运算表达式的值为 1。例如，逻辑表达式 a||b 求值时，若 a 为 1，则表达式的值为 1，对 b 不再求值；若 a 为 0，则需计算 b 的值。

3.1.4 if 命令

if 命令是最基本的选择控制命令，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，花括号{}可以省略。

双分支 if 命令的执行过程如下：

首先对“表达式”求值，然后进行判断分支：若“表达式”为非 0 值(即“真”)，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“假”），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。

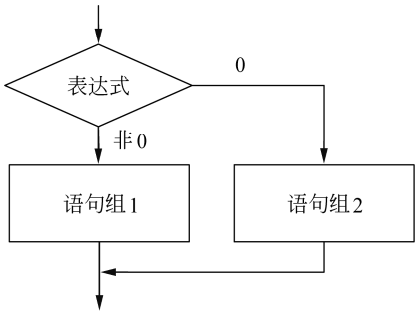

图 3-1　双分支 if 命令流程图

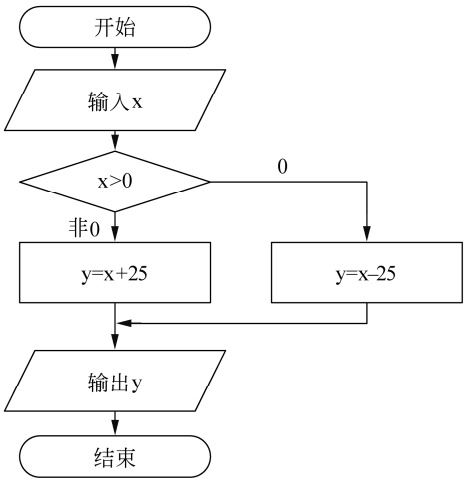

图 3-2　例 3-2 算法流程图

（2）实现程序

```
#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");      /* 每门课程成绩都不低于90  */
    else
        printf("加油!\n");        /* 至少有一门课程成绩不足90  */
   return 0;
}
```

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。

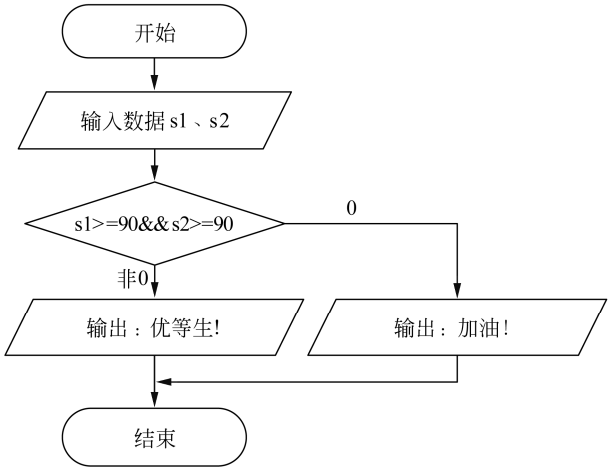

图 3-3　例 3-4 算法流程图

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。

第一次执行结果：

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

第二次执行结果：

```
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

```
#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
```

（4）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。为什么输入一个字符串时，只对其第一个字符进行输出处理？

2．单分支 if 命令

单分支 if 命令形式如下：

```
if(表达式)
  {语句组}
```

执行过程：首先对“表达式”求值，若表达式的值非 0（即真），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）程序实现

```
#include<stdio.h>
int main()
{
    int s1,s2,ave;
    printf("输入两门课的成绩：");
    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;
}
```


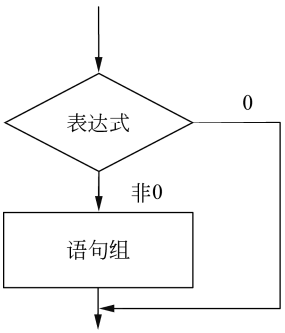

图 3-4　单分支 if 命令流程图

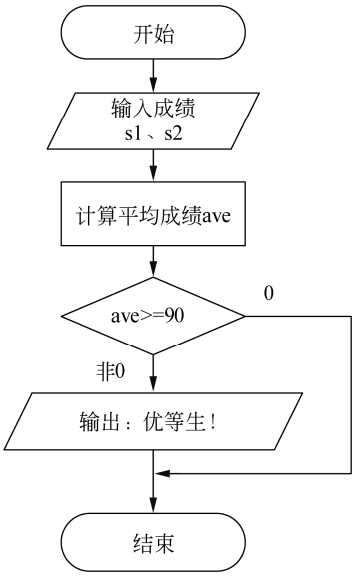

图 3-5　例 3-6 算法流程图

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。

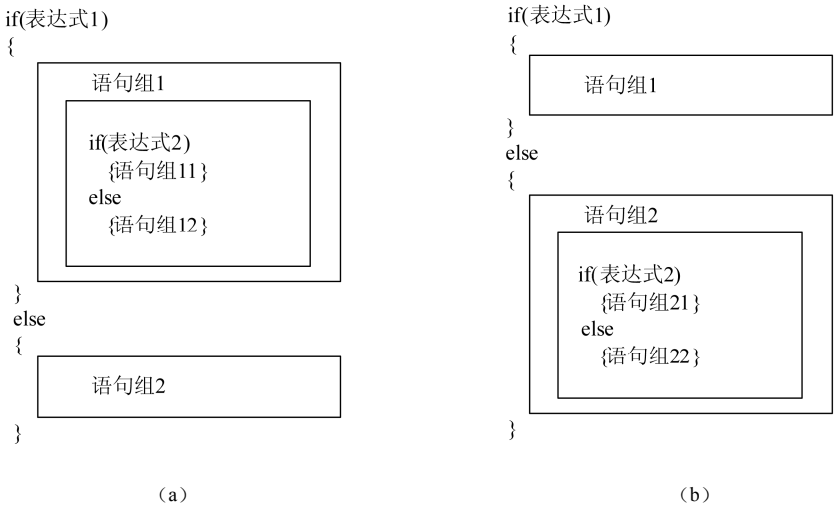

图 3-6　if 命令嵌套结构示意图

【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，则显示“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，然后再对 if 命令的嵌套情况进行说明。也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。

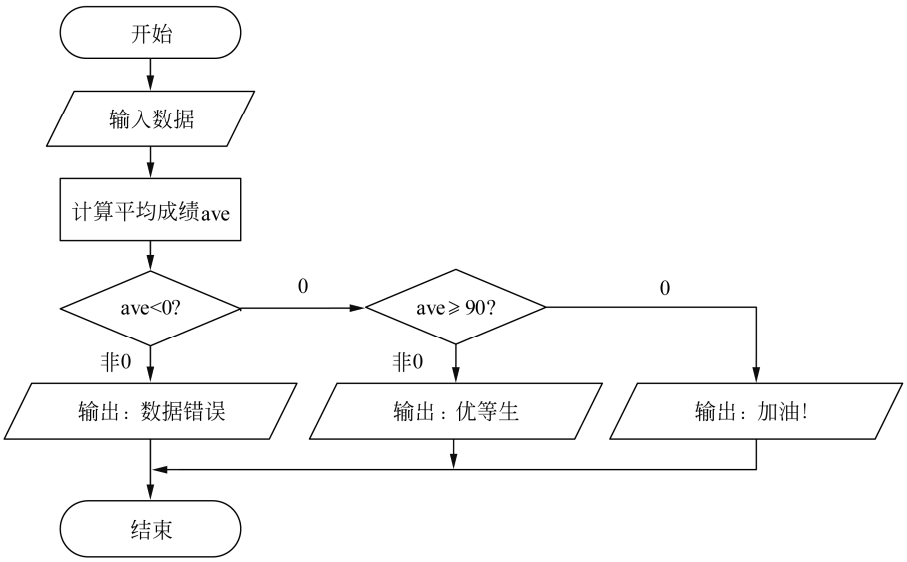

图 3-7　if 命令的嵌套结构举例

（2）实现程序

```
#include<stdio.h>
int main()
{
    int s1,s2,ave;
    printf("输入两门课的成绩：");
    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;
}
```

第一次执行结果：

```
输入两门课的成绩：70,-90↵
数据错误!
```

第二次执行结果：

```
输入两门课的成绩：90,95↵
优等生!
```

第三次执行结果：

```
输入两门课的成绩：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 与第一个 if 配对。程序段二整体上是一个单分支 if 语句，else 与第二个 if 配对，共同作为第一个 if 命令的语句体。关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}。如图 3-8 所示为四层 if 结构的流程图。

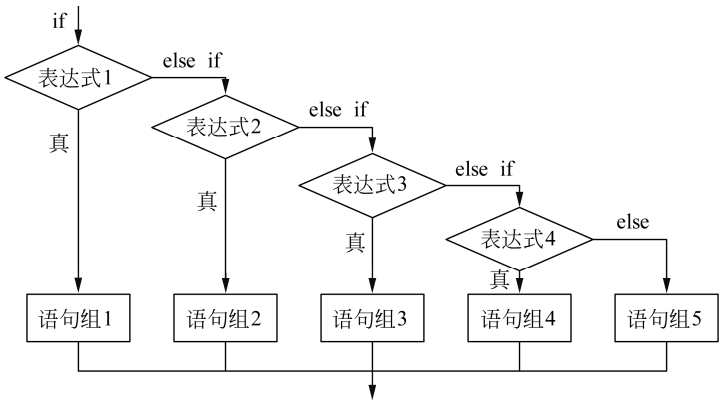

图 3-8　if-else if 语句的逻辑结构

【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

```
#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 命令判断而实现的，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就停止扫描过程，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 是 switch 命令执行过程的示意图。

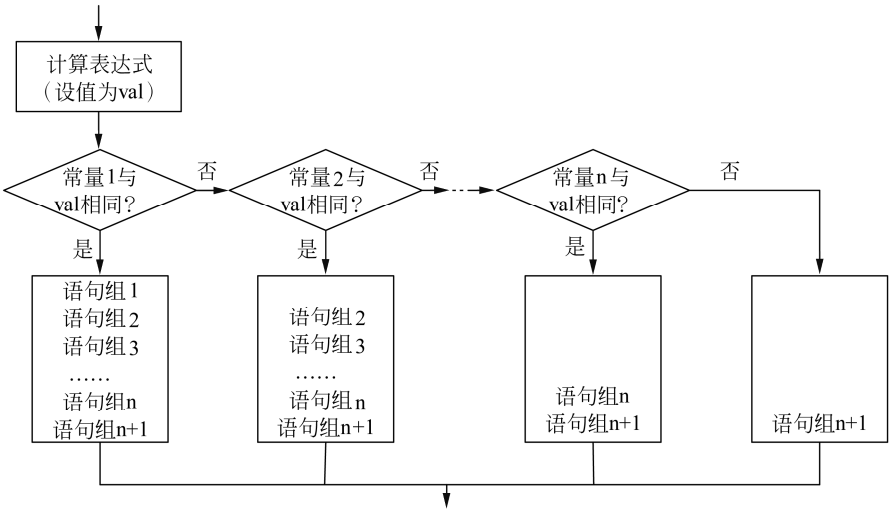

图 3-9　switch 命令执行过程示意图

使用 switch 命令时，允许缺省“default:”及其语句组。

【例 3-10】 switch 执行过程示例程序。

```
#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或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 语句，使程序立即执行 switch 语句的后续语句，即“printf("\n");”。

下面是关于 switch 语句的其他几点说明:

（1）任何一个 case 的语句组允许为空。

【例 3-11】 缺省 case 语句组的程序举例。

```
#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或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 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。

如下用法是正确的：

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

如下用法是错误的：

```
case a+5:
```

（4）switch 命令允许嵌套，即在 case 的语句组中允许再使用 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。地球绕太阳运行周期约为 365.24219 天，即一回归年。公历的平年只有 365 天，比回归年短约 0.24219 天，每四年累积约一天，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 天来，因此，每四百年中要减少三个闰年。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。

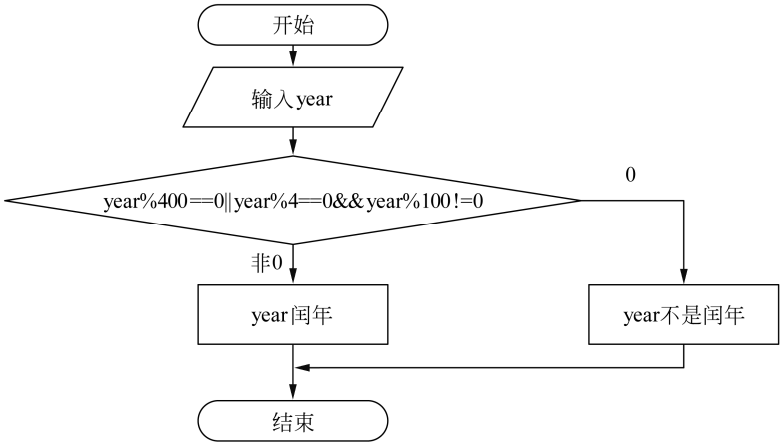

图 3-10　闰年问题算法流程图

2．实现程序

```
#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。

3. 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≠0）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，要注意打开 math.h 文件。

程序的算法流程图如图 3-11 所示。

2．实现程序

```
#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;
}
```

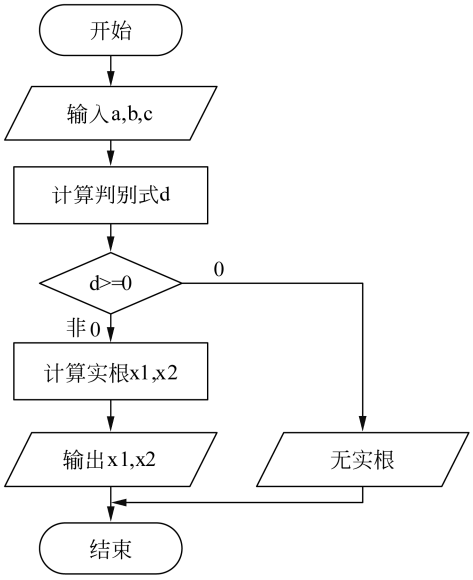

图 3-11　求解一元二次方程的算法流程图

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。

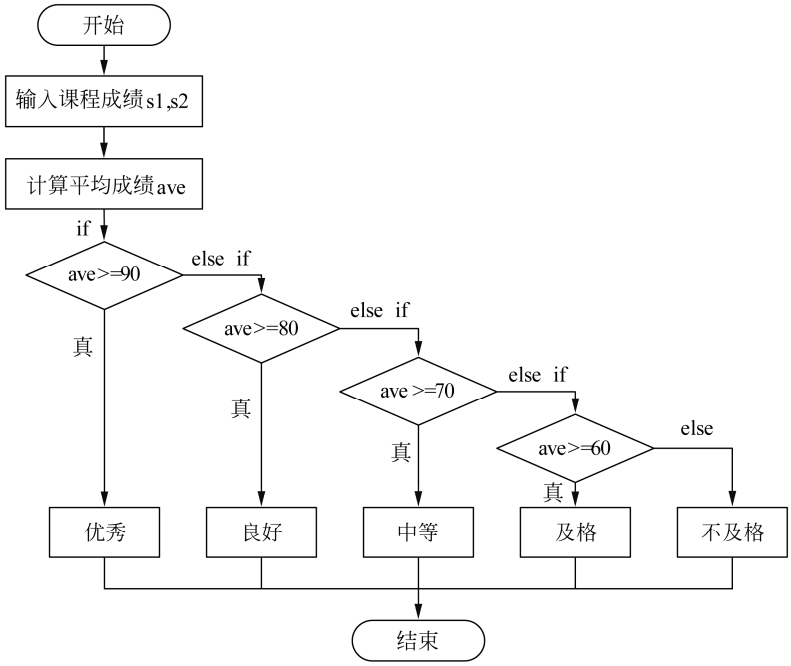

图 3-12 “学生成绩分等级显示”算法流程图

2．实现程序

```
#include<stdio.h>
int main()
{
    int s1,s2,ave;              /* s1、s2课程成绩、ave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
```

3. 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，即使输入了百分制之外的成绩值（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 结构也能方便地予以实现。希望读者使用 switch 结构改写上面的程序。

小　　结

本章介绍了程序中的分支结构知识，主要有 if 命令、switch 命令、用于条件表达的关系表达式和逻辑表达式，以及分支结构程序的典型实例。

（1）if 命令是最基本的分支控制命令，它有多种形式，通常用关系表达式和逻辑表达式进行分支条件表达。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 式的结构，而且更清晰。程序总是试图从满足条件的第一个 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，详细介绍了分支结构程序设计的方法和过程。

习　题　3

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

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

2．以下程序段的运行结果为_____。

```
int x=2,y=-1,z=2;
if(x<y)                    /* 第一个if */
  if(y<0)z=0;              /* 第二个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．a=1，b=2，c=3　　B．a=2，b=3，c=3

C．a=2，b=3，c=1　　D．a=2，b=3，c=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);
```

执行该段程序后，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时，则将其输出

二、编程题

1．按照如图 3-13 所示的流程图编写程序，并指出程序功能。

2．按照如图 3-14 所示的流程图编写程序，并指出程序功能。

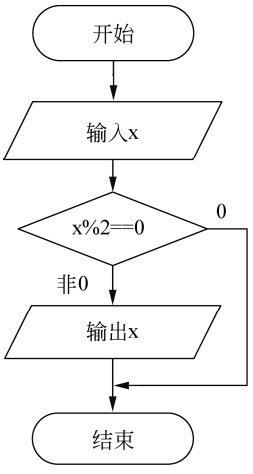

图 3-13　习题 1 算法流程图

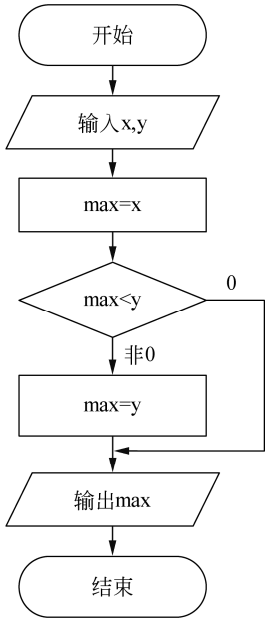

图 3-14　习题 2 算法流程图

3．计算邮费，邮件重量由键盘输入。邮件计费标准为：不超过 100 克时，每件 10 元；超过 100 克后，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．输入三个整数，然后按由大到小的顺序输出这三个数。

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。

实验 3　选择结构程序设计

一、实验目的

（1）学会使用关系表达式、逻辑表达式表示条件的方法。

（2）掌握 if 命令三种形式的用法。

（3）掌握 switch 命令的用法。

（4）学会具有嵌套的选择结构程序设计方法。

二、基本实验

1．if 命令及简单条件（单一关系表达式表示条件）练习

按以下要求修改例 3-1 的程序：

（1）学生的课程成绩可以是包括小数位的实数值。

（2）输出结果包括两类信息：

① 显示平均成绩；

② 若平均成绩不低于 90 分，则显示“优等生”；否则显示“加油!”。

以下为结果示例。

```
第一次执行程序：
输入两门课的成绩：95.5,81↵
Average: 83.3 加油!
第二次执行程序：
输入两门课的成绩：95.5,89↵
Average: 92.3 优等生!
```

2．if 命令及复杂条件（逻辑表达式表示条件）练习

修改上面的程序，将判定优等生的条件改为：若任意一门课程的成绩不低于 90，即判

定为“优等生”。

3．单分支 if 命令练习

修改上面的程序，使其只有在优等生情况时，才显示相关信息。

4．if-else if 命令练习

修改例 3-8 的程序，使程序执行后显示的结果与实验内容 1 的形式相同，即在原有显示信息之前，再增加平均成绩的信息。

5．switch 选择结构练习

例 3-14 使用 if-else if 结构实现“学生成绩分等级显示”的程序，事实上使用 switch 结构也能方便地解决这一问题。试用 switch 分支结构改写例 3-14 的程序。

提示：为减少 case 数量，可先将分布在[0,100]范围的平均成绩 ave（保留整数），通过表达式 ave/10 将其映射到[0,10]的较小范围内，如表 3-3 所示。在 switch 结构中，使用 switch(ave/10)形式进行分支控制，算法流程图如图 3-15 所示。

表 3-3 “ave/10”与等级对应表

ave（int 型）的值	ave/10 的值	对应等级
100	10	优秀（excellence）
小于 100、不小于 90	9	优秀（excellence）
小于 90、不小于 80	8	良好（all right）
小于 80、不小于 70	7	中等（middling）
小于 70、不小于 60	6	及格（pass）
小于 60	5、4、3、2、1、0	不及格（fail）

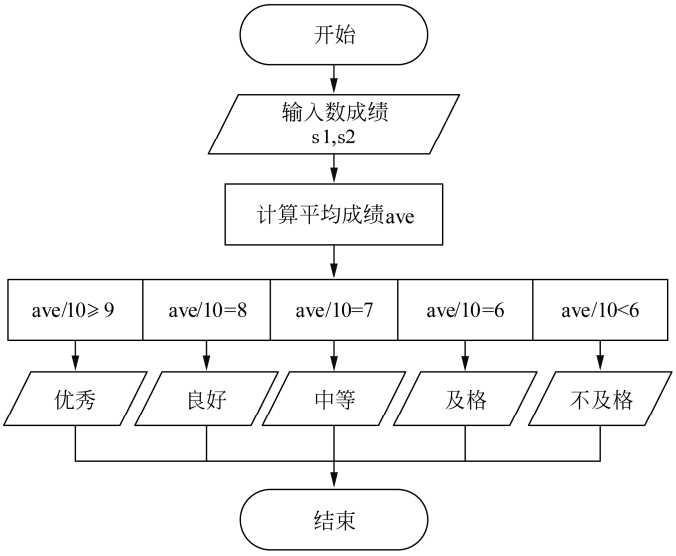

图 3-15　使用 switch 结构的算法流程图

参考程序：

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

```
{
    int s1,s2,ave;              /* s1、s2存储课程成绩，ave存储平均成绩*/
    printf("Score: ");
    scanf("%d,%d",&s1,&s2);     /* 输入课程成绩 */
    ave=(s1+s2)/2;              /* 计算平均成绩 */
    switch(ave/10)
    {
        case 10:                /* ave为100 */
        case 9:                 /* ave小于100、不小于90 */
            printf("Result: excellence\n");     /* 优秀时，输出等级结果 */
            break;
        case 8:                 /* ave小于90、不小于80 */
            printf("Result: all right\n");      /* 良好时，输出等级结果 */
            break;
        case 7:                 /* ave小于80、不小于70 */
            printf("Result: middling\n");       /* 中等时，输出等级结果 */
            break;
        case 6:                 /* ave小于70、不小于60 */
            printf("Result: pass\n");           /* 及格时，输出等级结果 */
            break;
        default:                /* ave小于60 */
            printf("Result: fail\n");           /* 不及格时，输出等级结果 */
    }
   return 0;
}
```

参考数据及结果：

```
Score: 77,98↵（此时，ave为87，ave/10为8，与“case 8:”匹配 */
Result: all right
Score: 89,92↵（此时，ave为90，ave/10为9，与“case 9:”匹配 */
Result: excellence
```

三、综合实验

编写学生成绩分等级程序。某学期有两门课程，每门课程都按百分制实行理论考核，按 A、B 两级实行实验考核。要求按照如下标准对学生的学习情况进行综合评价：

（1）综合评价结论由理论考核成绩和实验考核成绩共同确定。

（2）理论考核按照平均成绩分为如下 5 个等级：

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。

（3）当两门课的实验考核成绩均达到 A 级时，综合评定等级即为理论考核的等级，否

则按照理论考核的等级降一级认定。若理论考核等级为“不及格”时，不管实验考核成绩如何，综合评定等级仍为“不及格”。

实验参考如下。

（1）编程分析

① 本实验题目是前述学生成绩分等级显示问题的进一步扩展。理论成绩分等级的程序分析设计参考前述教学内容。本实验需要特别考虑的是，按照理论成绩分等后，要进一步对实验考核成绩进行判断，然后才能确定综合结论。可以考虑在 switch 语句中使用 if 语句处理。

② 输入数据的方式也应是程序设计时认真考虑的问题，合理的数据输入形式会使输入数据时清晰，而且不容易出错。例如，可以将每门课的理论考核成绩和实验考核成绩合为一体输入。假若有如下成绩：

a 课程：理论考核为 90，实验考核为 A。

b 课程：理论考核为 80，实验考核为 B。

则可以考虑使用如下方式输入数据：

90A,80B↵

（2）参考程序

```
#include<stdio.h>
int main(void)
{
    int a1,b1;        /* a1,b1分别存储a、b两门课程的理论考核成绩 */
    char a2,b2;       /* a2,b2分别存储a、b两门课程的实验考核成绩 */
    int ave;
    repeat:
    printf("请输入考核成绩: ");
    scanf("%d%c,%d%c",&a1,&a2,&b1,&b2);
    if(a1<0||b1<0||a1>100||b1>100)
         goto repeat;   /* 输入非法数据时，要求重新输入 */
    ave=(a1+b1)/2;
    switch(ave/10)
    {
        case 10:
        case 9:
             if(a2=='A'&&b2=='A')
                 printf("Result: excellence\n");  /* 理论≥90，实验均为A等 */
             else
                 printf("Result: all right\n");    /* 理论≥90，实验不为A等 */
             break;
        case 8:
             if(a2=='A'&&b2=='A')
                 printf("Result: all right\n");  /* 80≤理论<90，实验均为A等 */
             else
                 printf("Result: middling\n");  /* 80≤理论<90，实验不为A等 */
```

```
                break;
            case 7:
                if(a2=='A'&&b2=='A')
                    printf("Result: middling\n");  /* 70≤理论<80，实验均为A等 */
                else
                    printf("Result: pass\n");     /* 70≤理论<80，实验不为A等 */
                break;
            case 6:
                if(a2=='A'&&b2=='A')
                    printf("Result: pass\n");     /* 60≤理论<70，实验均为A等 */
                else
                    printf("Result: fail\n");      /* 60≤理论<70，实验不为A等 */
                break;
            default:
                printf("Result: fail\n");         /* 理论<60 */
        }
        return 0;
    }
```

关于参考程序的说明：

（1）该程序对数据输入的合法性进行了一定检验，能够确保理论考核的输入数据在 0～100 范围内，但对实验考核输入数据的合法性未作检验。程序中体现的实验考核成绩只有 A 和非 A 两种情况。

（2）若实验考核成绩为 A 级时，必须输入大写 A，否则将处理为非 A 级。当然，可以通过修改程序，无论输入 A 或 a 时，都认定为 A 级。