第 5 章　选择结构程序设计

应用程序的执行过程中可能会遇到多种不同的情况，程序必须依据指定的条件执行不同的操作，所有可能的情况都必须在程序编制过程中考虑好，用来实现这种情况的语句通常称为选择结构语句或分支结构语句(结构化程序设计的第 2 种结构)。C 语言中用来实现选择结构的语句有两种：if 语句和 switch 语句。

学习重点和难点：

- 关系运算符与关系表达式。
- 逻辑运算符与逻辑表达式。
- if 语句及其几种格式。
- switch 语句实现多分支选择。

学习本章后读者将会有按条件选择不同功能处理程序的编程能力。

5.1　本 章 引 例

第 4 章引例程序有个不足，就是若输入的边长 a，b，c 不构成三角形时，海伦公式就不适用的。学习本章后这类问题就能得到较好的解决。这里给出另外一个引例。

【例 5-1】 出租车计价程序：设某城市普通出租车收费标准：起步里程为 3 千米，起步价 10 元；超过起步里程后 10 千米内(即 4～10 千米)，每千米 2 元；超过 10 千米以上的部分每千米 3 元；营运过程中，因路阻或因乘客要求临时停车等产生等待时间的，按每 5 分钟 2 元收取(不足 5 分钟则不收费)。运价计费尾数四舍五入，保留到元(即不含小数)。编写程序，根据实际行驶里程(千米，截断式只保留 1 位小数)与等待时间(分钟)，计算并输出乘客应付的出租车费(元)。

分析： 行驶里程费与等待时间费分开计算后累加，里程费按规则分段计算，等待分钟不含小数，等待时间费直接按 5 的倍数计算。图 5-1 是本例的 N-S 流程图。

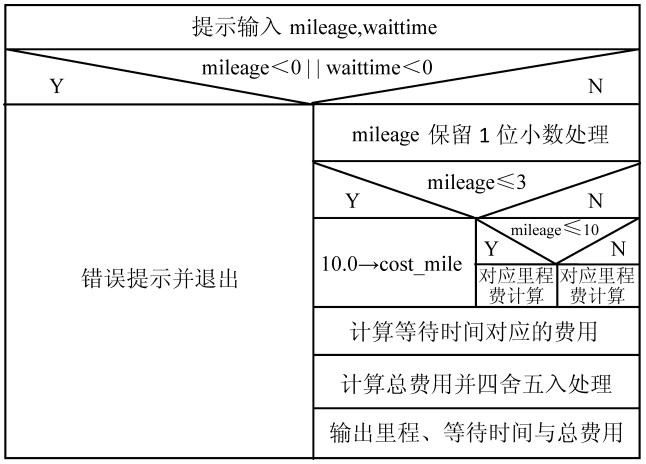


图 5-1　例 5-1 的 N-S 流程图

```
#include <stdio.h>
int main(void)
{ int waittime,cost_time,cost_all;
  double mileage,cost_mile;                                    //①变量定义部分
  printf("请输入里程数(千米,含1位小数):"); scanf("%lf",&mileage);    //②输入里程数
  printf("请输入等待时间(分钟):"); scanf("%d",&waittime);          //输入等待时间
  if (mileage<0 || waittime<0) {                //③按条件计算等待的数据处理部分
    printf("输入的里程数或等待时间(分钟)有误。");
    return 1;
  }
  mileage=((int)(mileage*10))/10.0;          //截断式处理,只保留1位小数
  if (mileage<=3.0)
    cost_mile=10.0;
  else if (mileage<=10.0)
    cost_mile=10.0+(mileage-3.0)*2.0;
  else
    cost_mile=10.0+(10.0-3.0)*2.0+(mileage-10.0)*3.0;
  cost_time=(waittime/5) * 2;
  cost_all=(int)(cost_mile+cost_time+0.5); //加0.5后转成int,实现四舍五入功能
  printf("本次乘坐里程数(千米):%5.1lf,等待时间(分钟):%d,费用共计:%d元\n",mileage,
waittime,cost_all);                                            //④输出结果
  return 0;
}
```

运行结果:

```
请输入里程数(千米, 含1位小数): 12.5
请输入等待时间(分钟): 6
本次乘坐里程数(千米):  12.5, 等待时间(分钟): 6, 费用共计: 34 元
```

学习本章内容后,编写的程序逻辑会更缜密,程序面对不同输入等情况的应对处理将更周全。简言之,编写的程序将会更健壮、更适用。

5.2 关系运算符和表达式

在程序中经常需要比较两个量的大小关系,以决定程序下一步的工作。比较两个量的运算符称为关系运算符。

5.2.1 关系运算符及其优先级

在C语言中有如表5-1所示的6种关系运算符。

表5-1 C语言关系运算符

符号	说　明
<	小于,如x<y(说明:x和y为变量、常量、各类表达式等)
<=	小于或等于
>	大于

续表

符号	说　明
>=	大于或等于
==	等于
!=	不等于(注意:“<>”不是C语言的不等于运算符)

关系运算符都是双目运算符,其结合性均为左结合。关系运算符的优先级低于算术运算符,高于赋值运算符。在6个关系运算符中,<、<=、>、>=的优先级相同,高于==和!=,==和!=的优先级相同(注意:关系运算符还分优先等级在其他语言中是少有的)。

5.2.2 关系表达式

关系表达式的一般形式为:

表达式　关系运算符　表达式

例如:

```
a+b>c-d  x>3/2  'a'+1<c  -i-5*j==k+1
```

都是合法的关系表达式。由于表达式也可以又是关系表达式,因此也允许出现嵌套的情况。例如:

```
a>(b>c)  a!=(c==d)
```

关系表达式的值是“真”和“假”,用1和0表示。如5>0的值为真,即为1。

(a=3)>(b=5),由于3>5不成立,故其值为假,即为0。

5>2>7>8在C语言中是允许的,相当于(((5>2)>7)>8)≡((1>7)>8) ≡(0>8),其值为0。

【例5-2】 输出一些关系表达式的运算值。

```
int main(void){
    char c='k';int i=1,j=2,k=3;float x=3e+5,y=0.85;
    printf("%d,%d\n",'a'+5<c,-i-2*j>=k+1);        //请注意各运算的优先级
    printf("%d,%d\n",1<j<5,x-5.25<=x+y);
    printf("%d,%d\n",i+j+k==-2*j,k==j==i+5);
}
```

运行结果如下。

```
1,0
1,1
0,0
```

说明:在本例中求出了各种关系表达式的值。字符变量是以它对应的ASCII码参与运算的。对于含多个关系运算符的表达式,如k==j==i+5,根据运算符的左结合性,先计算k==j,该式不成立,其值为0,再计算0==i+5,也不成立,故表达式值为0。

5.3 逻辑运算符和表达式

组合多种条件构成复合或复杂条件时,需要用到逻辑运算与逻辑表达式。

5.3.1 逻辑运算符及其优先级

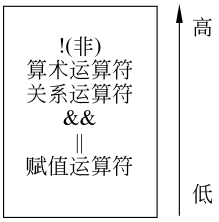
C语言中提供了三种逻辑运算符:

(1) && 与运算,如 x&&y(说明:x 和 y 为变量、常量、关系表达式或其他表达式等)。

(2) ‖ 或运算,如 x‖y。

(3) ! 非运算,如!x。

与运算符 && 和或运算符 ‖ 均为双目运算符,具有左结合性。非运算符! 为单目运算符,具有右结合性。逻辑运算符和其他运算符优先级的关系可表示为:!(非)→&&(与)→‖(或)。



"&&"和"‖"低于关系运算符,"!"高于算术运算符。

按照运算符的优先顺序可以得出:

a>b && c>d 等价于(a>b)&&(c>d)。

!b==c‖d<a 等价于((!b)==c)‖(d<a)。

a+b>c&&x+y<b 等价于((a+b)>c)&&((x+y)<b)。

5.3.2 逻辑运算及其取值

逻辑运算的值为"真"和"假"两种,用 1 和 0 来表示。其求值规则如下。

(1) **与运算 &&**。参与运算的两个量都为真时,结果才为真,否则为假。例如:

```
5.0>4.9 && 4>3
```

由于 5.0>4.9 为真,4>3 也为真,相与的结果也为真。

(2) **或运算‖**。参与运算的两个量只要有一个为真,结果就为真。两个量都为假时,结果为假。例如:

```
5.0>4.9‖5>8
```

由于 5.0>4.9 为真,相或的结果也就为真。

(3) **非运算!**。参与的运算量为真时,结果为假;参与的运算量为假时,结果为真。

例如,!(5.0>4.9)的结果为假,即为 0。

虽然 C 编译在运算得出逻辑运算结果值时,以 1 代表"真",0 代表"假"。**但在判断一个**

量是为“真”还是为“假”时，以 0 代表“假”，以非 0 的数值作为“真”。例如，由于 5 和 3 均为非 0，因此 5&&3 的值为“真”，即为 1。又如，5‖0 的值为“真”，即为 1。

逻辑运算!、&&、‖的“真值表”如表 5-2～表 5-4 所示，其中，x、y 为各种取值的运算量，可以是各种类型的常量、变量，各种表达式等。

表 5-2　逻辑运算真值表(1)

x	y	!x	x&&y	x‖y
真	真	假	真	真
真	假	假	假	真
假	真	真	假	真
假	假	真	假	假

表 5-3　逻辑运算真值表(2)

x	y	!x	x&&y	x‖y
1	1	0	1	1
1	0	0	0	1
0	1	1	0	1
0	0	1	0	0

表 5-4　逻辑运算真值表(3)

x	y	!x	x&&y	x‖y
非 0	非 0	0	1	1
非 0	0	0	0	1
0	非 0	1	0	1
0	0	1	0	0

注意：在 C 语言中要特别注意的是：运算量为 0 表示“假”，非 0 表示“真”；逻辑表达式的运算结果：“假”或“不成立”结果为 0，“真”或“成立”结果为 1。

5.3.3　逻辑表达式

逻辑表达式的一般形式为：

表达式 逻辑运算符 表达式

其中的表达式也可以是逻辑表达式，从而组成了嵌套的情形。例如：

```
(a&&b)&&c
```

根据逻辑运算符的左结合性，上式也可写为：

```
a&&b&&c
```

逻辑表达式的值是式中各种逻辑运算的最后值，以 1 和 0 分别代表“真”和“假”。

【例 5-3】 输出一些逻辑表达式的运算值。

```
int main(void){
    char c='k';  int i=1,j=2,k=3;  float x=3e+5,y=0.85;
    printf("%d,%d\n",!x*!y,!!!x);
    printf("%d,%d\n",x‖i&&j-3,i<j&&x<y);
    printf("%d,%d\n",i==5&&c&&(j=8),x+y‖i+j+k);
}
```

运行结果如下。

```
0,0
1,0
0,1
```

说明：本例中!x和!y分别为0,!x*!y也为0,故其输出值为0。由于x为非0,故!!!x的逻辑值为0。对于式x‖i&&j-3,先计算j-3的值为非0,再求i&&j-3的逻辑值为1,故x‖i&&j-3的逻辑值为1。对于式i<j&&x<y,由于i<j的值为1,而x<y为0,故表达式的值为1与0相与,最后为0,对于式i==5&&c&&(j=8),由于i==5为假,即值为0,该表达式由两个与运算组成,所以整个表达式的值为0。对于式x+y‖i+j+k,由于x+y的值为非0,故整个或表达式的值为1。

注意：逻辑表达式求解时有个特性,并非所有的逻辑运算符都被执行,只在必须执行下一个逻辑运算符才能求出表达式的结果时,才执行该运算符,也即在已明确表达式的真或假值时,后续对结果没有影响力的运算将不执行了。

(1) 如a&&b&&c,**只在a为真时,才判别b的值;只在a、b都为真时,才判别c的值**。也即a为假时,表达式为假,就不判断b与c了;a为真,b为假时,表达式亦为假,就不判断c了。

(2) 如a‖b‖c,**只在a为假时,才判别b的值;只在a、b都为假时,才判别c的值**。也即a为真时,表达式为真,就不判断b与c了;a为假,b为真时,表达式亦为真,就不判断c了。例如:

```
a=1;b=2;c=3;d=4;m=1;n=1; (m=a>b)&&(n=c>d);  /*结果 m=0,n=1,而非 n=0*/
```

例如,int x=5; 表达式0‖++x的值为1,表达式求值后x为6;而int x=5; 表达式0&&++x的值为0,表达式求值后x为5,而非x为6(因为++x未执行)。

5.4 if语句的用法

用if语句可以构成选择结构。它根据给定的条件进行判断,以决定执行某个分支程序段。C语言的if语句有三种基本形式。

5.4.1 if语句的三种形式

1. 第一种形式

其基本形式为:

```
if(表达式) 语句;
```

其语义是:如果表达式的值为真,则执行其后的语句(可为复合语句),否则不执行该语句,其执行逻辑如图5-2所示。

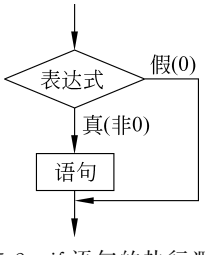


图5-2 if语句的执行逻辑

注意：if后的表达式可以是关系表达式、逻辑表达式、数值表达式等,类型也可以任意。

if(!表达式)等价于if(表达式==0),而if(表达式)等价于if(表达式!=0)。

【例 5-4】 输入两个整数,输出其最大者(方法 1)。

```
int main(void){
    int a,b,max;
    printf("input two numbers: ");scanf("%d%d",&a,&b);
    max=a;                          //先假设 a 为最大(这种做法常用)
    if (max<b) max=b;               //然后用当前的最大值 max 与 b 做比较
    printf("max=%d\n",max);         //输出两者的最大值
}
```

运行结果如下。

```
input two numbers: 12 56
max=56
```

说明:本例程序中,输入两个数 a 和 b。把 a 先赋予变量 max,再用 if 语句判别 max 和 b 的大小,如 max 小于 b,则把 b 赋予 max。最后 max 中是 a、b 两者的大数,最后输出 max 的值。

2. 第二种形式

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

其语义是:如果表达式的值为真,则执行语句 1,否则执行语句 2。其执行的逻辑过程如图 5-3 所示。

【例 5-5】 输入两个整数,输出其最大者(方法 2)。

```
int main(void){
    int a, b; printf("input two numbers: ");
    scanf("%d%d",&a,&b);
    if(a>b)       //直接比较,根据比较情况得到最大值
        printf("max=%d\n",a);
    else printf("max=%d\n",b);
}
```

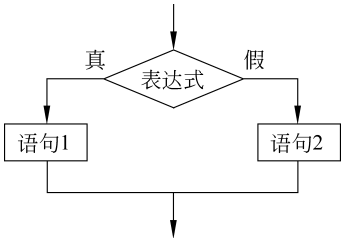


图 5-3 if…else 语句的执行逻辑

说明:输入两个整数,输出其中的大数。改用 if…else 语句判别 a、b 的大小,若 a 大,则输出 a,否则输出 b。

3. 第三种形式

前两种形式的 if 语句一般都用于两个分支的情况。当有多个分支选择时,可采用 if…else…if 语句,其一般形式为:

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

写成缩进嵌套形式

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

其语义是：依次判断表达式的值，当出现某个值为真时，则执行其对应的语句，然后跳到整个 if 语句之后继续执行程序。如果所有的表达式均为假，则执行语句 n。然后继续执行后续程序。if…else…if 语句的执行过程如图 5-4 所示，关于 if 语句嵌套的内容见 5.4.2 节。

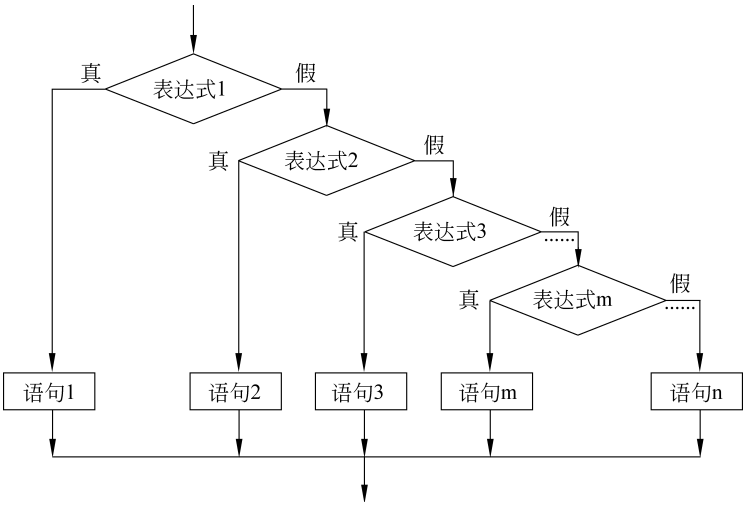


图 5-4　if…else…if 语句的执行过程

【例 5-6】 输入一个字符，判断它所属的类别。

```
#include<stdio.h>
int main(void){
    char c; printf("input a character: ");
    c=getchar();
    if(c<32)                          //或 if(iscntrl(c))使用判断字符的库函数,下同
        printf("This is a control character\n");
    else if(c>='0'&&c<='9')           //或 else if(isdigit(c))
        printf("This is a digit\n");
    else if(c>='A'&&c<='Z')           //或 else if(isupper(c))
        printf("This is a capital letter\n");
    else if(c>='a'&&c<='z')           //或 else if(islower(c))
        printf("This is a small letter\n");
    else
        printf("This is an other character\n");
}
```

运行结果如下。

```
input a character: X
This is a capital letter
```

说明：本例要求判别键盘输入字符的类别。可以根据输入字符的 ASCII 码来判别类型。由 ASCII 码表可知：ASCII 值小于 32 的为控制字符；在“0”和“9”之间的为数字；在“A”和“Z”之间的为大写字母；在“a”和“z”之间的为小写字母；其余则为其他字符。也可以如程序注释那样，用判断字符的库函数来表达所属字符条件。

这是一个多分支选择的问题，用 if…else…if 语句编程，判断输入字符 ASCII 码所在的范围，分别给出不同的输出。例如，输入为“g”，输出显示“This is a small letter”（它为小写字母）。

在使用 if 语句时还应注意以下问题。

（1）在三种形式的 if 语句中，在 if 关键字之后均为表达式。**该表达式通常是逻辑表达式或关系表达式，但也可以是其他表达式，如算术表达式、赋值表达式等，甚至也可以是一个变量**。

例如：

```
if(a=5) 语句;
if(b) 语句;
```

都是允许的。**只要表达式的值为非 0，即为“真”，都能执行到其后的语句**。

如在“if(a＝5)…；”中表达式的值永远为非 0，所以其后的语句总是要执行的，当然这种情况在程序中不一定会出现，但在语法上是合法的。

又如，有程序段：

```
if(a=b)          //请注意：a=b 为赋值表达式，而不是关系表达式
    printf("%d",a);
else
    printf("a=0");
```

本程序段的语义是，把 b 值赋予 a，如果为非 0 则输出该值，否则输出“a＝0”字符串。这种用法在程序中是经常出现的。

注意：“＝”与“＝＝”使用中的不同，例如，“int a＝0，b＝1；if(a＝b) printf("a equal to b?\n");”与“int a＝0，b＝1；if(a＝＝b) printf("a equal to b\n");”是完全不同的。

（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**。

例如，if x＝＝y printf("x 等于 y") 是错误的，应改为

```
if (x==y) printf("x 等于 y");
```

（3）在 if 语句的三种形式中，所有的语句应为单个语句，**如果要想在满足条件时执行一组（多个）语句，则必须把这一组语句用{}括起来组成一个复合语句**。但要注意的是在右大括号“}”之后不能再加分号（;）。例如：

```
if(a>b)
{  a++; b++;
}  //注意：此处不能有";"号
else { a=0; b=10; }
```

5.4.2 if 语句的嵌套

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

其一般形式可表示如下。

```
if(表达式)
    if 语句;
```

或者为

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

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

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

其中的else究竟是与哪一个if配对呢?应该理解为:

```
if(表达式 1)                    ┐
    if(表达式 2)    ┐           │
        语句 1;     ├内 if 语句 ├外 if 语句
    else            │           │
        语句 2;     ┘           ┘
```

还是应理解为:

```
if(表达式 1)                    ┐
    if(表达式 2)    ┐           │
        语句 1;     ├内 if 语句 ├外 if 语句
else                │           │
    语句 2;                     ┘
```

为了避免这种二义性,**C语言规定:else总是与它前面最近的还未配对的if子句相配对**。因此对上述例子应按前一种情况理解。当然完全可以通过加“{…}”来实现后一种逻辑的表达,如下所示。

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

【例5-7】 比较并显示两个数的大小关系(if…else形式实现)。

```
int main(void){
    int a,b;  printf("please input A,B: ");
    scanf("%d%d",&a,&b);
    if(a!=b)
```