第5章　选择结构程序设计

微课视频

◇ 学习意义

选择控制结构是结构化程序设计所采用的三种基本控制结构之一，另外两种是顺序控制和循环控制。有人曾经证明：任何程序都可用顺序、选择、循环三种控制结构来实现，而结构化程序设计的研究成果表明：只用这三种控制结构编写的程序易于保证正确性。在编制程序时，有时并不能保证程序一定执行某些指令，而是要根据一定的外部条件来判断哪些指令要执行。如菜谱中要加工番茄，可能有这样的步骤：如果是用番茄，则去皮、切碎，开始放入，如果是用番茄酱，就在最后放入。这里并不知道具体操作时执行哪段指令，但菜谱给出了不同条件下的处理方式，计算机程序也是如此，可以根据不同的条件执行不同的代码，这就是选择结构。程序总是为解决某个实际问题而设计的，而问题往往包含多个方面，不同的情况需要有不同的处理方法，所以选择结构在实际应用程序中可以说是无处不在，离开了选择结构很多情况将无法处理，因此，正确掌握选择结构程序设计方法对于编写实际应用程序尤为重要。本章将首先介绍与选择结构有关的关系运算符、逻辑运算符和条件运算符的使用方法，然后重点介绍 C 语言中的两个选择控制语句 if 和 switch，最后通过几个实例程序帮助读者更好地掌握选择结构程序设计的方法。

◇ 学习目标

（1）理解选择结构的含义；

（2）掌握 C 语言语句的分类；

（3）掌握关系运算符、逻辑运算符和条件运算符的用法；

（4）掌握 if、switch 语句的使用方法。

◇ 难点提示

（1）由关系运算符、逻辑运算符组成复杂的条件表达式；

（2）switch 语句实现选择结构程序设计。

微课视频

5.1 C语言程序中语句的分类

C语言程序的执行部分是由语句组成的。程序的功能也是由执行语句实现的。C语言中的语句可以分为以下五类。

1. 表达式语句

表达式语句由表达式加上分号“;”组成。其一般形式为：

```
表达式;
```

例如，a=10是赋值表达式，而a=10;则是赋值语句；k++是表达式，k++;则是表达式语句。

2. 函数调用语句

由函数名、实际参数加上分号“;”组成。其一般形式为：

```
函数名(实际参数表);
```

执行函数语句就是调用函数体并把实际参数赋予函数定义中的形式参数，然后执行被调函数体中的语句，求取函数值(在第8章中再详细介绍)。

例如，printf("C Program")是函数调用；printf("C Program");是函数调用语句，其功能是输出字符串"C Program"。

3. 复合语句

把多条语句用花括号{}括起来组成的一条语句称为复合语句。在程序中应把复合语句看成单条语句，而不是多条语句。例如：

```
{
  c=a+b;
  z=x+y;
  printf("c=%d, z=%d", c, z);
}
```

是一条复合语句。复合语句内的各条语句都必须以分号“;”结尾，在括号“}”外不要加分号。

复合语句又称分程序，在复合语句内可以定义变量，但定义的变量只能在复合语句内使用。例如，下面程序中的阴影部分是复合语句。在该复合语句中，定义了一个变量z。

```
#include <stdio.h>
int main( )
{
  int x=10, y=20, z;
```

```
    z=x+y;
    {
        int z;
        z=x*y;
        printf("z=%d\n", z);          //输出复合语句中 z 的值
    }
    printf("z=%d\n", z);              //输出复合语句外 z 的值
    return 0;
}
```

输出的结果将为：

```
z=200
z=30
```

为什么会输出这样的结果？通过后面第 8 章的学习读者将会明白。复合语句主要使用在选择和循环控制结构中。

4. 空语句

只有分号“;”组成的语句称为空语句。空语句是什么也不执行的语句。在程序中空语句可用来作空循环体。例如，while(getchar( )!='\n');的功能是，只要从键盘输入的字符不是回车则重新输入。这里的循环体为空语句。

5. 控制语句

用来实现一定的控制功能的语句称为控制语句。控制语句用于控制程序的流程，以实现程序的各种结构方式。它们由特定的语句定义符组成。C 语言用控制语句来实现选择结构和循环结构。C 语言有九种控制语句，可分成以下三类。

(1) 条件判断语句——if 语句、switch 语句。

(2) 循环执行语句——do while 语句、while 语句、for 语句。

(3) 转向语句——break 语句、goto 语句、continue 语句、return 语句。

5.2 关系运算符、逻辑运算符、条件运算符

在学习各种控制语句之前，必须学会关系运算符、逻辑运算符和条件运算符的用法。它们是选择结构和循环结构程序设计的基础。

微课视频

5.2.1 关系运算符和关系表达式

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。C 语言提供了 6 种关系运算符，它们的含义及优先级关系见表 5-1。

表 5-1 关系运算符

关系运算符	含 义	优 先 级	结 合 性
>	大于	这些关系运算符等优先级，但比下面的优先级高	左结合性
>=（>和=之间没有空格）	大于或等于		
<	小于		
<=（<和=之间没有空格）	小于或等于		
==（两个=之间没有空格）	等于	这些关系运算符等优先级，但比上面的优先级低	
!=（!和=之间没有空格）	不等于		

用关系运算符连接起来的式子称为关系表达式。其一般形式为：

```
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。由于表达式又可以是关系表达式，因此也允许出现嵌套的情况，例如，a>(b>c)，a!=(c==d)等。关系表达式的值是“真”和“假”，分别用 1 和 0 表示。例如，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。

⚠注意：

- C 语言用 0 表示假，非 0 表示真。
- 一个关系表达式的值不是 0 就是 1，0 表示假，1 表示真。

关系运算符都是双目运算符，其优先级高于与、或以及赋值运算符，但低于算术运算符和移位运算符（如图 5-1 所示）。

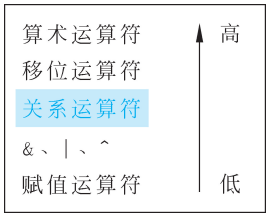

图 5-1 关系运算符的优先级

例如：

c > a+b 含义是 c >(a+b)

a > b !=c 含义是(a > b) !=c

a==b < c 含义是 a==(b < c)

a=b > c 含义是 a=(b > c)

a >> 2 < c+d 含义是(a >> 2) <(c+d)

a & 4 > b | c 含义是(a &(4 > b)) | c

关系运算符具有左结合性，相同优先级的关系运算符连用时，按照从左向右的顺序计算表达式的值。例如，如果有：

```
a=1; b=2; c=3;
d=a !=c==a < b < c;
```

则 d 的值是 1。因为=的优先级最低，所以 d=a!=c==a<b<c；等价于 d=(a!=c==a<b<c);，又因为<的优先级高于!=和==，所以它又等价于 d=(a!=c==(a<b<c));，又因为!=和==是同一优先级，具有左结合性，所以它又等价于 d=((a!=c)==((a<b)<c));，这时候最先计算 a<b 的值为 1，再计算 1<c 的值是 1。接下来计算 a!=c 的值

是1,再计算1==1的值是1,所以d的值是1。

⚠注意:在使用关系运算符时,应避免对实数进行相等或不等的判断。例如,关系表达式1.0/3.0*3.0==1.0的值是0,可改写为:fabs(1.0/3.0*3.0-1.0)<1e-6。

微课视频

5.2.2 逻辑运算符和逻辑表达式

C语言提供的逻辑运算符有三种,其含义及优先级关系如表5-2所示。

表5-2 逻辑运算符

逻辑运算符	含 义	结合性	优先级关系
!	单目运算符,逻辑非,表示相反	右结合性	↑ 高
&&(两个&之间没有空格)	双目运算符,逻辑与,表示并且	左结合性	
||(两个|之间没有空格)	双目运算符,逻辑或,表示或者		↓ 低

用逻辑运算符连接起来的式子称为逻辑表达式。其一般格式为:

```
表达式  逻辑运算符  表达式
```

例如,a<b && b<c,x>10 || x<-10,!x && !y都是合法的逻辑表达式。由于表达式又可以是逻辑表达式,因此也允许出现嵌套的情况,例如,x>y && x>z && x<0等。逻辑表达式的值也是"真"或"假",分别用1和0表示。

"||"运算符两边的式子只要有一个式子为真,整个逻辑表达式的值就是真(即为1),否则整个逻辑表达式的值就是假(即为0)。

"&&"运算符两边的式子只有都是真时,整个逻辑表达式的值才是真(即为1),否则整个逻辑表达式的值就是假(即为0)。

"!"运算符是单目运算符,当其右边的式子是真时,整个逻辑表达式的值是假(即为0),否则整个逻辑表达式的值就是真(即为1)。

表5-3列出了逻辑运算符的运算规则。

表5-3 逻辑运算符的运算规则

A	B	!A	!B	A && B	A || B
假	假	1	1	0	0
假	真	1	0	0	1
真	假	0	1	0	1
真	真	0	0	1	1

逻辑运算符中的!是单目运算符,因此其优先级较高,它与++、--和sizeof同优先级,比算术运算符的优先级要高。另两个逻辑运算符是双目运算符,优先级较低,比关系运算符和位运算符都要低,但比赋值运算符的优先级高(如图5-2所示)。

例如:

a<=x && x<=b等价于(a<=x)&&(x<=b)

a > b && x > y 等价于(a > b)&&(x > y)
a==b || x==y 等价于(a==b)||(x==y)
!a || a > b 等价于(!a) || (a > b)
!a > b 等价于(!a) > b
c=a || b 等价于 c=(a || b)
a | 7 && b & 8 等价于(a | 7)&&(b & 8)
a >> 2 && b << 1 等价于(a >> 2)&&(b << 1)

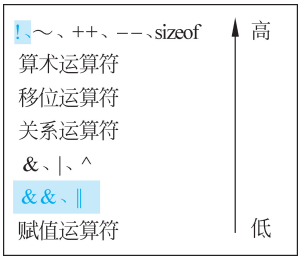

图 5-2　逻辑运算符的优先级

三个逻辑运算符中,!的优先级最高,&& 的优先级高于||的优先级。!具有右结合性,而 && 和||具有左结合性。例如,如果有:

```
a=4; b=5;
c=b>3 && 2 || 8<b - !a;
```

则 c 的值是 1。因为>、<、-、&&、||、!的优先级高于=,所以 c=b>3&&2||8<b-!a;等价于 c=(b>3&&2||8<b-!a);,又因为 &&、||的优先级低于>、<、-、!,所以又等价于 c=((b>3) &&2||(8<b-!a));,又因为 && 的优先级高于||,所以又等价于 c=(((b>3)&&2)||(8<b-!a));,又因为-的优先级高于<,! 的优先级高于-,所以最后等价于 c=(((b>3)&&2)||(8<(b-(!a))));。所以先计算(b>3) && 2 的值,因为 b=5,所以 b>3 的值为 1,1 && 2 的值为 1,而对于 8<(b-(!a))来说,因为!a 为 0,b-0 的值为 5,8<5 的值为 0,所以其值为 0,最后 1||0 的值就为 1,再赋给 c,所以 c 的值为 1。

有一点需要说明,在逻辑表达式求值的过程中,并不是所有的逻辑运算符都被执行,只有在必要的情况下才会执行。例如:

- 在求解 a && b && c 的值时,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。如果 a 为假,则不会判别 b 和 c,因为整个表达式的值已经确定了。
- 在求解 a || b || c 的值时,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,如果 a 为真,就不会判别 b 和 c,因为整个表达式的值已经确定了。

例如:

```
a=1; b=2; c=3; d=4; m=1; n=1;
```

则(m=a>b)&&(n=c>d)的结果将使得 m 的值为 0,但 n 的值仍为 1。因为先计算 m=a>b,则 m 的值为 0(因 a<b),则整个表达式的值就为 0,&& 后面的表达式根本就不计算,所以 n 的值不变,仍为 1。

5.2.3　条件运算符和条件表达式

微课视频

条件运算符为?和:,它是 C 语言提供的唯一一个三目运算符,即有三个参与运算的量。由条件运算符连接的式子称为条件表达式。条件表达式的一般格式为:

```
表达式 1 ? 表达式 2 : 表达式 3
```

其求值规则为:如果表达式 1 的值为真,则以表达式 2 的值作为条件表达式的值,否则以表

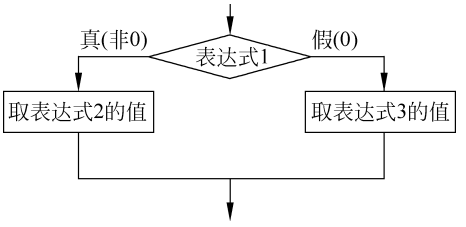

图 5-3　条件表达式流程图

达式 3 的值作为整个条件表达式的值，其执行过程如图 5-3 所示。例如，5>3 ? 6 : 20 的值是 6，5<3 ? 6 : 20 的值是 20。

条件表达式通常用于赋值语句之中。例如，将变量 a、b 中的最大值赋给变量 max 可以写成如下语句。

```
max=a > b ? a : b;
```

使用条件表达式时，还应注意以下几点：

- 条件运算符的运算优先级特别低，仅高于赋值运算符和逗号运算符，而比其他运算符都低。因此 max=(a>b) ? a : b;可以去掉括号而写为：

```
max=a > b ? a : b;
```

- 条件运算符可嵌套。例如，x>0 ? 1 : (x<0 ? -1 : 0)。其表达式的值是：如果 x 是正数，则为 1，如果是负数，则为-1，如果为零，则为 0。
- 条件运算符的结合方向是自右至左。如 a>b ? a : c>d ? c : d 等价于 a>b ? a: (c>d ? c : d)。
- 条件运算符?和：是一对运算符，不能分开单独使用。
- 表达式 1、表达式 2、表达式 3 的类型可以不相同，表达式值取较高的类型。例如，x ? 'a' : 'b'的含义是 x 为 0，表达式值为'b'；x 不为 0，表达式值为'a'。x>y ? 1 : 1.5 的含义是 x 大于 y，表达式值为 1.0；x 小于或等于 y，表达式值为 1.5。

下面例 5-1 中的程序是利用条件运算符对第 4 章中例 4-13 小写字母转盘程序的改写。

【例 5-1】　小写字母转盘。

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

int main( )
{
    char ch, ch1, ch2;                          //变量定义

    ch=getche( );                               //读取一字符
    putchar('\n');                              //换行
    ch1=ch=='a'? 'z' : ch-1;                    //求前驱字母
    ch2=ch=='z'? 'a' : ch+1;                    //求后继字母
    printf("ch1=%c, ch2=%c\n", ch1, ch2);       //显示结果
    return 0;
}
```

运行结果(假设输入字母为 w)：

```
ch1=v, ch2=x
```

到现在为止，已经学习了三十多个运算符了。掌握它们的优先级关系特别重要。机械

地记忆这三十多个运算符的优先级关系是痛苦的。下面给出的优先级的记忆规则将会使这件事变得轻松许多。

（1）总体上讲，单目运算符都是同等优先级的，具有右结合性，并且优先级比双目运算符和三目运算符都高。

（2）三目运算符的优先级比双目运算符要低，但高于赋值运算符和逗号运算符。

（3）逗号运算符的优先级最低，其次是赋值运算符。

（4）只有单目运算符、赋值运算符和条件运算符具有右结合性，其他运算符都是左结合性。

（5）双目运算符中，算术运算符的优先级最高，逻辑运算符的优先级最低。

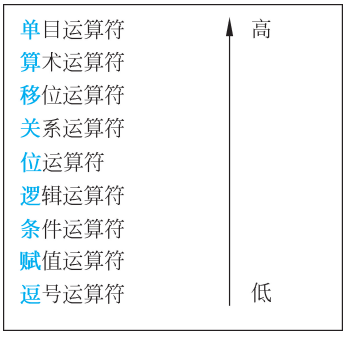

图 5-4　各类运算符的优先级关系

图 5-4 给出了各类运算符之间的优先关系，详细内容见附录 D。

运算符优先级的口诀是：“单算移关，位逻条赋，逗！”。

5.3 选择结构的程序设计

在程序的三种基本结构中，第二种为选择结构，其基本特点是：程序的流程由多路分支组成，在程序的一次执行过程中，根据不同的情况，只有一条分支被选中执行，而其他分支上的语句则被直接跳过。

C 语言中提供了 if 语句和 switch 语句来实现选择结构，if 语句用于两者选一的情况，而 switch 语句用于多分支选一的情形。

5.3.1　if 语句

微课视频

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。

1. 简单 if 语句形式

语句格式为：

```
if(表达式)
  语句;
```

其语句功能是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，而执行 if 语句后面的语句。其对应的流程图如图 5-5 所示。

例如，下面的程序段是输入两个整数，输出其中的大数。

```
int a, b, max;
```

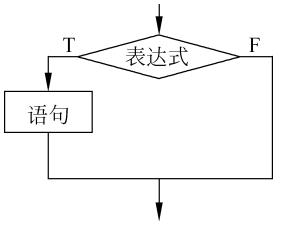

图 5-5　简单 if 语句流程图

```
printf("input two numbers: ");
scanf("%d%d", &a, &b);
max=a;
if(max < b)
   max=b;
printf("max=%d", max);
```

本例程序中,输入两个数 a 和 b。把 a 先赋给变量 max,再用 if 语句判别 max 和 b 的大小,如果 max 小于 b,则把 b 赋给 max。因此 max 中总是大数,最后输出 max 的值。

2. if-else 形式

语句格式为:

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

其语句功能是如果表达式的值为真,则执行语句 1,否则执行语句 2,其对应的流程图如图 5-6 所示。

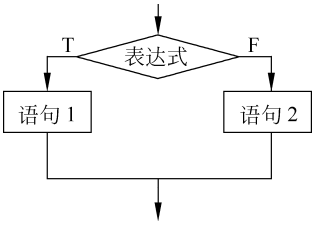

图 5-6　if-else 语句流程图

例如,下面的程序段同样是输出两个整数中的最大数。

```
int a, b;
printf("input two numbers: ");
scanf("%d%d", &a, &b);
if(a > b)
   printf("max=%d\n", a);
else
   printf("max=%d\n", b);
```

本程序中,改用 if-else 语句判别 a 和 b 的大小,若 a 大,则输出 a,否则输出 b。

3. if-else-if 形式

语句格式为:

```
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                 语句 n;]
```

其语句功能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 语句之外继续执行程序。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，其对应的流程图如图 5-7 所示。

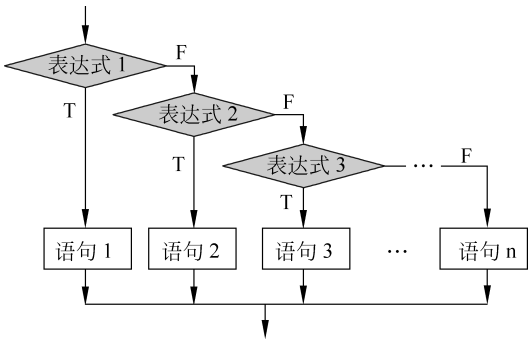

图 5-7　if-else-if 语句流程图

例如，下面的程序段是判断输入字符的种类。

```
char c;
printf("Enter a character: ");
c=getchar( );
if(c < 0x20)   printf("The character is a control character\n");
else if(c >= '0' && c <= '9')   printf("The character is a digit\n");
else if(c >= 'A' && c <= 'Z')   printf("The character is a capital letter\n");
else if(c >= 'a' && c <= 'z')   printf("The character is a lower letter\n");
else printf("The character is other character\n");
```

本程序中，根据输入字符的 ASCII 码来判别字符类型。由 ASCII 码表可知 ASCII 值小于 32(十六进制 0x20)的为控制字符。在'0'和'9'之间的为数字，在'A'和'Z'之间为大写字母，在'a'和'z'之间为小写字母，其余则为其他字符。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'，输出显示它为小写字符。

在使用 if 语句时还应注意以下几点：

(1) 在 if 语句中，条件判断表达式必须用括号括起来。

(2) 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，但也可以是其他任何表达式，如赋值表达式等，甚至也可以是一个变量。只要表达式非零时，表达式的值就为真，否则就是假。例如，下面的 if 语句都是合法的。

```
if(a=5) 语句;          //表达式的值永远为非 0,所以其后的语句总是要执行的
if(b)   语句;          //等价于 if(b !=0) 语句
```

(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}之后不能再加分号。例如，下面的程序段左边是正确的，而右边则是错误的，因为在 if 与 else 之间有两条语句，必须用{ }括起来组成一个复合语句。

正确的 if 语句	错误的 if 语句

if(a > b)
{
  a++;
  b++;
}
else
{
  a=0;
  b=1;
}
	if(a > b)
	  a++;
	  b++;
	else
	{
	  a=0;
	  b=1;
	}

⚠**注意**：在 if 和 else 之间如果只有一条语句，则可不用{ }括起来，但如果多于一条语句则必须用{ }括起来，否则会产生编译错误。

(4) 在简单的 if 语句中(即只有 if，没有 else)，如果在满足条件时执行的是多条语句，原则上应用{ }括起来，但如果没有用{ }括起来，尽管不会产生编译错，但程序的逻辑将出现异常。例如，下面程序的目的是当 x 小于 y 时，交换 x 和 y 的值。左边的程序是正确的，而右边的程序表现异常，原因就是 if 语句的后面漏掉了{ }。

正确的程序	逻辑异常的程序

#include <stdio.h>
 	
int main( )
{
  int x=4, y=2, t=0;
 	
  if(x < y)
  {
  t=x;
  x=y;
  y=t;
  }
  printf("x=%d, y=%d\n", x, y);
  return 0;
}
	#include <stdio.h>
	
	int main( )
	{
	  int x=4, y=2, t=0;
	
	  if(x < y)
	  t=x;
	  x=y;
	  y=t;
	  printf("x=%d, y=%d\n", x, y);
	  return 0;
	}

运行结果：x=4, y=2	运行结果：x=2, y=0

(5) 在 if 语句中，如果表达式是一个判断两个数是否相等的关系表达式，要当心不要将==写成了赋值运算符=。下面程序的意图是当 x 等于 0 时，输出 x=0，否则输出 x!=0。左边的程序是正确的，右边的程序则是错误的，因为将 x==0 误写成了 x=0。程序中 x 的值是 0，因此 x==0 的值是 1，而 x=0 的值是 0。

结果正确的程序	结果错误的程序

#include <stdio.h>
 	
int main( )
{
  int x=0;
 	
  if(x==0)
  printf("x=0\n");
  else
  printf("x !=0\n");
  return 0;
}
	#include <stdio.h>
	
	int main( )
	{
	  int x=0;
	
	  if(x=0)
	  printf("x=0\n");
	  else
	  printf("x !=0\n");
	  return 0;
	}

运行结果：x=0	运行结果：x !=0

(6) if 语句允许嵌套，即 if 语句中的执行语句又是 if 语句。其一般形式可表示为：

```
if(表达式)
  if 语句;
```

或

```
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？**C 语言规定，在省略{ }时，else 总是和它上面离它最近的未配对的 if 配对**(如图 5-8 所示)。因此，else 与第二个 if 配对，上述例子理解的形式应为：

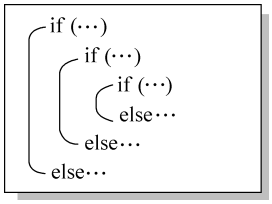

图 5-8　if-else 配对原则

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

如果希望 else 与第一个 if 配对，那么必须在 else 前加上{ }，即写成如下形式。

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

```
    }
    else
        语句 2;
```

试比较下面程序一和程序二之间的差异。

程　序　一	程　序　二

#include <stdio.h>
 	
int main( )
{
  int a=1, b=-1;
 	
  if(a > 0)
  if(b > 0)
  a++;
  else
  a--;
  printf("a=%d\n");
  return 0;
}
	#include <stdio.h>
	
	int main( )
	{
	  int a=1, b=-1;
	
	  if (a > 0)
	  {
	  if (b > 0)
	  a++;
	  }
	  else
	  a--;
	  printf("a=%d\n");
	  return 0;
	}

运行结果：a=0	运行结果：a=1

微课视频

5.3.2　switch 语句

C 语言还提供了另一种用于多分支选择的 switch 语句，实际上，switch 语句是 if-else 语句的变形。在某些情况下，使用 switch 语句要比 if 语句更为简洁、易读。其一般格式为：

```
switch(表达式)
{
  case 常量表达式 C1: 语句组 1;
                      break;
  case 常量表达式 C2: 语句组 2;
                      break;
          ⋮
  case 常量表达式 Cn: 语句组 n;
                      break;
  [default:           语句组;
                      break;]
}
```

其执行过程为：

- 当 switch 后面“表达式”的值与某个 case 后面的“常量表达式”的值相同时，就执行

case 后面的语句(组)；当执行到 break 语句时，跳出 switch 语句，转向执行 switch 语句的下一条语句。

- 如果没有任何一个 case 后面的“常量表达式”的值与“表达式”的值匹配，则执行 default 后面的语句(组)，然后再执行 switch 语句的下一条语句。

switch 语句执行的流程图如图 5-9 所示。

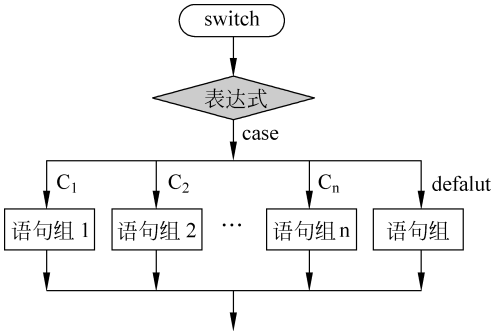

图 5-9 switch 语句流程图

在使用 switch 语句时还应注意以下几点：

(1) switch 后面的“表达式”，必须是一个整型表达式，而且每个 case 后的“常量表达式”的类型应该与 switch 后面的“表达式”的类型一致。例如，下面的程序是错误的。

```
float  a, b=4.0;
scanf("%f", &a);
switch(a)              //错误,不可为浮点型表达式
{
  case 1:  b=b+1;  break;
  case 2:  b=b-1;  break;
}
printf("b=%f\n",b);
```

(2) case 后面语句(组)可加{ }也可以不加{ }，但一般不加{ }。例如：

```
switch(i)
{
  case 1:  { b=b+1;  break; }     // { }可加可不加
  case 2:  b=b-1;  break;
}
```

(3) 每个 case 后面“常量表达式”的值必须各不相同，否则会出现相互矛盾的现象(即对表达式的同一值，不允许有两种或两种以上的执行方案)。例如，下面的程序是错误的。

```
int  a, b=4;
scanf("%d", &a);
switch(a)
{
  case 1:  b=b+2;  break;
  case 2:  b=b*2;  break;
  case 1:  b=b+2;  break;          //错误,case 1 在前面已使用,可改为 case 3
```

```
}
printf("b=%d\n", b);
```

(4) 每个 case 后面必须是“常量表达式”,表达式中不能包含变量。初学者使用 switch 语句时很容易犯此类错误,所以这一点务必要注意。例如,下面程序的功能是欲判断用户输入的字符类别,但程序是错误的,因为 case 后面的表达式不是常量表达式。

```
char c;
printf("Enter a character: ");
c=getchar( );
switch(c)
{
  case c < 0x20 :                    //错误,case 后面跟着变量
                printf("The character is a control character\n");
                break;
  case c >= '0' && c <= '9' :
                printf("The character is a digit\n");
                break;
  case c >= 'A' && c <= 'Z' :
                printf("The character is a capital letter\n");
                break;
  case c >= 'a' && c <= 'z' :
                printf("The character is a lower letter\n");
                break;
  default :
                printf("The character is other character\n");
                break;
}
```

要实现该程序的功能,应使用 if-else-if 形式的分支结构,其程序参见 5.3.1 节。当然,如果非要使用 switch 语句来实现,则只能将每种字符列举出来,这种方法显然是非常烦琐的,所以 if 语句的功能并不是简单地用 switch 来代替就行,有时 switch 语句根本就无法实现 if 语句的功能。

(5) case 后面的“常量表达式”仅起语句标号作用,并不进行条件判断。系统一旦找到入口标号,就从此标号开始执行,不再进行标号判断,所以必须加上 break 语句,以便结束 switch 语句。比较下面程序一、程序二的运行结果。

程　序　一	程　序　二

#include <stdio.h>
 	
int main( )
{
  char ch;
 	
  ch=getch( );
  switch(ch)
  {
	#include <stdio.h>
	
	int main( )
	{
	  char ch;
	
	  ch=getch( );
	  switch(ch)
	  {

程 序 一	程 序 二

case 'Y': printf("Yes\n"); break;
case 'N': printf("No\n"); break;
case 'A': printf("All\n"); break;
default : printf("Yes,No or All\n");
}
return 0;
}
	case 'Y': printf("Yes\n"); break;
	case 'N': printf("No\n");
	case 'A': printf("All\n"); break;
	default : printf("Yes,No or All\n");
	}
	return 0;
	}

运行结果(假设输入 N): No	运行结果(假设输入 N): No All

对于程序二来说,因为 ch 的值是'N',因此从 case 'N'处开始执行,首先执行 printf 函数调用,输出 No,因为后面没有 break,所以接着执行下一行语句,即 case 'A'后面的 printf 函数调用,输出 All,当遇到 break 时,就跳出 switch 语句。

(6) 多个 case 子句,可共用同一语句(组)。例如,下面的程序中,当 a 的值是 1、2、3 时,将 b 的值加 2,当 a 的值是 4、5、6 时,将 b 的值减 2,否则将 b 的值乘以 2。

```
int  a, b=4;
scanf("%d", &a);
switch(a)
{
  case 1:
  case 2:
  case 3:  b+=2;     break;
  case 4:
  case 5:
  case 6:  b -=2;    break;
  default:  b *=2;   break;
}
printf("b=%d\n", b);
```

(7) case 子句和 default 子句如果都带有 break 子句,那么它们之间顺序的变化不会影响 switch 语句的功能。下面的两个程序是等价的。

程 序 一	程 序 二

#include <stdio.h>
 	
int main( )
{
char ch;
 	
ch=getch( );
switch(ch)
{
	#include <stdio.h>
	
	int main( )
	{
	char ch;
	
	ch=getch( );
	switch(ch)
	{

程　序　一	程　序　二

case 'Y': printf("Yes\n"); break;
case 'N': printf("No\n"); break;
case 'A': printf("All\n"); break;
default : printf("Yes,No or All\n");
break;
}
return 0;
}
	case 'Y': printf("Yes\n"); break;
	default : printf("Yes,No or All\n");
	break;
	case 'N': printf("No\n"); break;
	case 'A': printf("All\n"); break;
	}
	return 0;
	}

(8) case 子句和 default 子句如果有的带 break 子句,而有的没有带 break 子句,那么它们之间顺序的变化可能会影响输出的结果。比较下面程序一、程序二的运行结果。

程　序　一	程　序　二

#include <stdio.h>
 	
int main( )
{
char ch;
 	
ch=getch( );
switch(ch)
{
case 'Y': printf("Yes\n"); break;
case 'N': printf("No\n"); break;
case 'A': printf("All\n"); break;
default : printf("Yes,No or All\n");
}
return 0;
}
	#include <stdio.h>
	
	int main( )
	{
	char ch;
	
	ch=getch( );
	switch(ch)
	{
	case 'Y': printf("Yes\n"); break;
	default : printf("Yes,No or All\n");
	case 'N': printf("No\n"); break;
	case 'A': printf("All\n"); break;
	}
	return 0;
	}

运行结果(假设输入 B): Yes,No or All	运行结果(假设输入 B): Yes,No or All No

(9) switch 语句可以嵌套。例如,下面程序的 switch 语句中又嵌套了一个 switch 语句。

```
int main( )
{
  int x=1, y=0, a=0, b=0;
  switch(x)
  {
    case 1: switch( y )
            {
              case 0: a++; break;
              case 1: b++; break;
```

```
            }
        case  2:  a++;  b++; break;
        case  3:  a++;  b++;
    }
    printf("\na=%d, b=%d", a, b);
    return 0;
}
```

程序运行的结果为：

```
a=2,b=1
```

5.4 选择结构程序设计举例

微课视频

【例 5-2】 已知某公司员工的保底薪水为 500，某月所接工程的利润 profit（整数）与利润提成的关系如表 5-4 所示（计量单位：元）。计算员工的当月薪水。

表 5-4　工程利润与利润提成的关系

工程利润 profit	提 成 比 率
profit≤1000	没有提成
1000<profit≤2000	提成 10%
2000<profit≤5000	提成 15%
5000<profit≤10000	提成 20%
10000<profit	提成 25%

程序应该这样来设计：

（1）首先要定义一个变量 profit 用来存放员工所接工程的利润。

（2）其次提示用户输入员工所接工程的利润，并调用 scanf 函数接受用户输入员工所接工程的利润。

（3）然后根据表 5-4 的规则，计算该员工当月的提成比率 ratio。

（4）最后计算该员工当月的薪水 salary（保底薪水＋所接工程的利润×提成比率），并输出结果。

画出 N-S 流程图表示算法如图 5-10 所示。

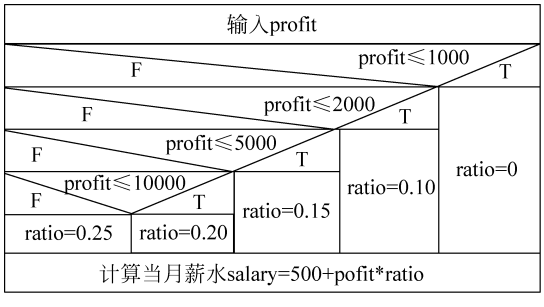

图 5-10　计算当月薪水 N-S 流程图

具体程序如下：

```
1    #include <stdio.h>
2    int main( )
3    {
4       long   profit;                          //所接工程的利润
5       float  ratio;                           //提成比率
6       float  salary=500;                      //薪水,初始值为保底薪水 500
7
8       printf("Input profit: ");               //提示输入所接工程的利润
9       scanf("%ld", &profit);                  //输入所接工程的利润
10
11      //计算提成比率
12      if(profit <=1000)
13         ratio=0;
14      else if(profit <=2000)
15            ratio=(float)0.10;
16      else if(profit <=5000)
17            ratio=(float)0.15;
18      else if(profit <=10000)
19            ratio=(float)0.20;
20      else   ratio=(float)0.25;
21
22      salary+=profit * ratio;                 //计算当月薪水
23      printf("salary=%.2f\n", salary);        //输出结果
24      return 0;
25   }
```

运行结果(假设输入 4000)：

```
Input profit: 4000↙
salary=1100.00
```

程序解释：

如果在调用 scanf 函数时，用户输入 4000，则 profit 的值是 4000，因此第 12 行的 profit<=1000 的值为假，不会执行第 13 行的语句，然后程序试图执行第 14 行语句，但 profit <=2000 的值也为假，因此不会执行第 15 行的语句，接着程序试图执行第 16 行语句，profit <=5000 的值为真，因此会执行第 17 行的语句，提成比率 ratio 为 0.15，并跳出 if 语句。最后，根据“保底薪水+所接工程的利润×提成比率”计算当月的薪水 salary=500+4000×0.15=1100 元，并输出结果。

为了更好地理解 if 语句的用法，可以将例 5-2 改写成下面这样。但读者要仔细比较一下它们的不同之处。

```
#include <stdio.h>
int main( )
{
    long   profit;                          //所接工程的利润
    float  ratio;                           //提成比率
    float  salary=500;                      //薪水,初始值为保底薪水 500

    printf("Input profit: ");               //提示输入所接工程的利润
    scanf("%ld", &profit);                  //输入所接工程的利润

    //计算提成比率
    if(profit <=1000)
        ratio=0;
    if(1000 < profit && profit <=2000)
        ratio=(float)0.10;
    if(2000 < profit && profit <=5000)
        ratio=(float)0.15;
    if(5000 < profit && profit <=10000)
        ratio=(float)0.20;
    if(10000 < profit)
        ratio=(float)0.25;

    salary+=profit * ratio;                 //计算当月薪水
    printf("salary=%.2f\n", salary);        //输出结果
    return 0;
}
```

【例 5-3】 用 switch 语句来实现例 5-2。

微课视频

算法设计要点：

为使用 switch 语句，必须将利润 profit 与提成的关系转换成某些整数与提成的关系。分析本题可知，提成的变化点都是 1000 的整数倍(1000,2000,5000,…)，如果将利润 profit 整除 1000，则当：

profit≤1000	对应 0,1
1000<profit≤2000	对应 1,2
2000<profit≤5000	对应 2,3,4,5
5000<profit≤10000	对应 5,6,7,8,9,10
10000<profit	对应 10,11,12,…

为解决相邻两个区间的重叠问题，最简单的方法就是：利润 profit 先减 1(最小增量)，然后再整除 1000 即可。

profit≤1000	对应 0
1000<profit≤2000	对应 1
2000<profit≤5000	对应 2,3,4
5000<profit≤10000	对应 5,6,7,8,9
10000<profit	对应 10,11,12,…

具体程序如下：

```
#include <stdio.h>

int main( )
{
    long   profit;                          //所接工程的利润
    int    grade;
    float  ratio;                           //提成比率
    float  salary=500;                      //薪水,初始值为保底薪水 500

    printf("Input profit: ");               //提示输入所接工程的利润
    scanf("%ld", &profit);                  //输入所接工程的利润

    //将利润-1、再整除 1000,转换成 switch 语句中的 case 标号
    grade=(profit-1)/1000;
    switch(grade)                           //计算提成比率
    {
      case  0:  ratio=0;          break;    // profit≤1000
      case  1:  ratio=(float)0.10;  break;  // 1000<profit≤2000
      case  2:
      case  3:
      case  4:  ratio=(float)0.15;  break;  // 2000<profit≤5000
      case  5:
      case  6:
      case  7:
      case  8:
      case  9:  ratio=(float)0.20;  break;  // 5000<profit≤10000
      default:  ratio=(float)0.25;          // 10000<profit
    }
    salary+=profit * ratio;                 //计算当月薪水
    printf("salary=%.2f\n", salary);        //输出结果
    return 0;
}
```

微课视频

【例 5-4】 写一段程序，从键盘上输入年份 year(4 位十进制数)，判断其是否闰年。闰年的条件是：能被 4 整除，但不能被 100 整除，或者能被 400 整除。

算法设计要点：

(1) 如果 X 能被 Y 整除，则余数为 0，即如果 X%Y 的值等于 0，则表示 X 能被 Y 整除。

(2) 首先画出判别闰年算法的 N-S 流程图，如图 5-11 所示。将是否闰年的标志 leap 预置为 0(非闰年)，这样仅当 year 为闰年时，将 leap 置为 1 即可。最后判断 leap 是否为 1，若是，则输出“闰年”信息。这种处理两种状态值的方法，对优化算法和提高程序可读性非常有效，请读者仔细体会。

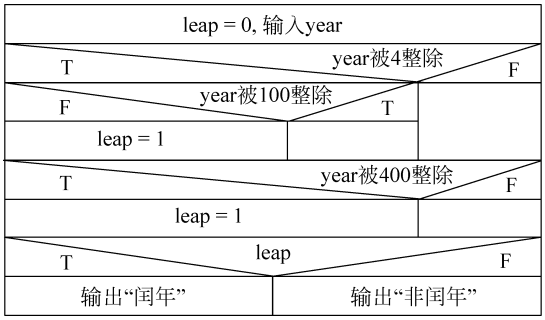

图 5-11 判别闰年算法 N-S 流程图

具体程序如下：

```
1   #include <stdio.h>
2
3   int main( )
4   {
5       int year, leap=0;                    // leap=0: 预置为非闰年
6
7       printf("Please input the year: ");   //提示输入年份
8       scanf("%d", &year);                  //输入年份
9
10      if(year % 4==0)                      //如果被 4 整除
11          if(year % 100 !=0)               //如果不被 100 整除
12              leap=1;                      //置为闰年
13      if(year % 400==0)                    //如果被 400 整除
14          leap=1;                          //置为闰年
15
16      //输出结果
17      if(leap)                             //如果是闰年(leap==1)
18          printf("%d is a leap year.\n", year);
19      else
20          printf("%d is not a leap year.\n", year);
21      return 0;
22  }
```

运行结果(假设输入 2020)：

```
Please input the year: 2020 ↙
2020 is a leap year.
```

程序解释：

- 程序的第 7、8 行用于输入年份。如输入值为 2020，则 year 的值为 2020。
- 第 10～14 行用于判别输入的年份 year 是否为闰年。如果 year 为 2020，则能被 4 整除，即对于第 10 行的 if 语句来说，因为 year%4==0 为真，所以执行第 11 行，因为 2020 不被 100 整除，即 year%100!=0 为真，所以执行第 12 行的语句，则 leap 置为

1(闰年标志)。又因 2020 不被 400 整除,即 year%400!=0,所以第 14 行不执行。

- 第 17 行中因为 leap 为 1,所以执行第 18 行的语句,输出其为闰年。
- 利用逻辑运算能描述复杂条件的特点,可将上述程序中的第 10～14 行优化为:

```
if(( year % 4==0 && year % 100 !=0 ) || ( year % 400==0 ) )
    leap=1;
```

或直接写为:

```
leep=( year % 4==0 && year % 100 !=0 ) || ( year % 400==0 );
```

微课视频

【例 5-5】 编写一程序,从键盘上输入任意两个数和一个运算符(+,-,*,/),计算其运算的结果并输出。

算法设计要点:

首先输入两个数和一个运算符号,然后根据运算符号来进行相应的运算,但是在做除法运算时,应先判别除数是否为 0,如果为 0,运算非法,给出提示信息。如果运算符号不是+、-、*、/,则同样是非法的,也应给出提示信息。其他情况,输出运算的结果。

具体程序如下:

```
1   #include <stdio.h>
2
3   int main( )
4   {
5     float a, b;                                   //存放两个数的变量
6     int tag=0;                                    //运算合法的标志,0 合法,1 非法
7     char ch;                                      //运算符变量
8     float result;                                 //运算结果变量
9
10    printf("input two number: ");                 //提示输入两个数
11    scanf("%f%f", &a, &b);                        //输入两个数
12    fflush(stdin);                                //清键盘缓冲区
13    printf("input arithmetic label(+ - * /): ");  //提示输入运算符
14    scanf("%c", &ch);                             //输入运算符
15
16    switch(ch)                                    //根据运算符来进行相关的运算
17    {
18      case  '+':  result=a+b;  break;             //加法运算
19      case  '-':  result=a-b;  break;             //减法运算
20      case  '*':  result=a*b;  break;             //乘法运算
21      case  '/':  if(!b)                          //除法运算,判除数是否为 0
22                  {
23                    printf("divisor is zero!\n"); //显示除数为 0
24                    tag=1;                        //置运算非法标志
25                  }
26                  else                            //除数非 0
```

```
27                          result=a/b;                          //计算商
28                        break;
29        default:        printf("illegal arithmetic label\n");   //非法的运算符
30                        tag=1;                                 //置运算非法标志
31    }
32    if(!tag)                                                   //运算合法,显示运算结果
33      printf("%.2f %c %.2f=%.2f\n", a, ch, b, result);
34    return 0;
35  }
```

运行结果(假设输入的两个数为：20　30,运算符为：+)：

```
input two number: 20 30↙
input arithmetic label(+ - * /): +↙
20.00+30.00=50.00
```

程序解释：

- 程序的第 5 行定义了两个浮点型变量 a 和 b,用于存放要计算的两个数。
- 第 6 行定义一变量 tag,其值用于运算合法的标志,0：合法,1：非法。
- 第 7 行定义一变量 ch 用于存放运算符。
- 第 8 行定义一变量 result 用于存放运算结果。
- 第 10～14 行输入计算的数据及运算符。
- 第 16～31 行根据运算符的类型进行相应的运算。但在做除法运算时,首先要判断除数是否为 0,如果为 0,则置运算非法标志,即 tag 置 1,否则做运算(见第 21～28 行)。如果运算符不是+、-、*、/,则置运算非法标志,即 tag 置 1(见第 29、30 行)。
- 第 32、33 行表示如果运算合法,则显示相应的运算结果。

微课视频

5.5 本章小结及常见错误列举

C 语言程序的执行部分是由语句组成的。程序的功能也是由执行语句实现的。C 语言中的语句可以分为表达式语句、函数调用语句、复合语句、空语句及控制语句五类。

关系表达式和逻辑表达式是两种重要的表达式,主要用于条件执行的判断和循环执行的判断。

C 语言提供了多种形式的条件语句以构成选择结构。

(1) if 语句主要用于单向选择。

(2) if-else 语句主要用于双向选择。

(3) if-else-if 语句和 switch 语句主要用于多向选择。

任何一种选择结构都可以用 if 语句来实现,但并非所有的 if 语句都有等价的 switch 语句。switch 语句只能用来实现以相等关系作为选择条件的选择结构。

本章主要讨论了选择结构程序设计的有关方法,重点介绍了 if 语句和 switch 语句,在

学习这两种控制语句的过程中,又接触到了一些新的C语言关键字,它们是:

if	else	switch	case	default	break

下面列举了一些在选择结构程序设计中常见的错误。

(1) 忘记必要的逻辑运算符。

例如:

```
if(a > b > c)
    ⋮
```

本意为如果a>b并且b>c,由于在数学中使用a>b>c的形式,也就把它照搬到计算机程序中。而在C语言程序中,表达式a>b>c的求值是先求a>b,得到一个逻辑值0或1,再拿这个数与c进行比较,结果当然是不对的。对于这种情况,应该使用逻辑表达式,写成:

```
if(a > b && b > c)
    ⋮
```

(2) 在关系表达式中误用=来表示==。

C语言用两个连续的赋值运算符来表示“相等”关系运算符。如果将=当作==使用,通常不会有语法错,但却隐含着不易发现的逻辑错误。

(3) 应该用复合语句时忘记写花括号{ }。

例如:

```
if(a > b)
  temp=a;
  a=b;
  b=temp;
```

由于没有写花括号,if的影响只限于temp=a;一条语句,而不管(a>b)是否为真,都将执行后两个赋值语句。正确的写法为:

```
if(a > b) {
  temp=a;
  a=b;
  b=temp;
}
```

(4) 在不该加分号的地方加分号。

例如:

```
if(a==b);
  c=a+b;
```

本意是如果a等于b则执行c=a+b,但由于if(a+b)后面跟有分号,c=a+b;在任何情况下都要执行。因为if后加分号相当于后跟一个空语句,这种错误是因为习惯在每行的末尾都加分号所致,正确的写法为:

```
if(a==b)
  c=a+b;
```

(5) 在 else 之前的语句丢失分号。

例如：

```
if(a > b)
  max=a     //漏了分号,正确的写法为: max=a;
else
  max=b;
```

(6) 将 &&、||误输入为 &、|。

在 C 语言中，&& 是逻辑运算符，而 & 可以是位运算符，也可以是取地址运算符；|| 是逻辑运算符，而 | 是位运算符。如果将 &&、|| 误输入成 &、|，也不会有语法错误，但隐含着逻辑错误。

(7) ==、!=、<=、>=运算符中间多了空格。

像==、!=、<=、>=这些运算符虽然由两个或两个以上的字符组成，但它们是整体，构成这些运算符的字符之间不能有空格。

(8) case 子句后面的程序段中漏掉了 break 子句。

仔细比较下面两段程序，左边的程序正常，右边的程序表现异常，因为 case 子句的后面漏掉了 break 子句。

```
ch=getch( );
switch(ch)
{
  case  'Y': printf("Yes\n");
              break;
  case  'N': printf("No\n");
              break;
}
```

```
ch=getch( );
switch(ch)
{
  case  'Y': printf("Yes\n");
              //漏掉了 break;
  case  'N': printf("No\n");
              break;
}
```

(9) case 后面跟着变量表达式。

switch 语句中，case 后面必须是常量表达式，不能是变量或变量表达式。下面的程序是错误的。

```
switch(a)
{
  case a > 0 :    printf("a > 0\n");   break;
  case a < 0 :    printf("a < 0\n");   break;
  case a==0 :     printf("a=0\n");     break;
}
```

(10) switch 后面的表达式为浮点类型。

C 语言规定，switch 后面的表达式必须是整型，不能是浮点类型。例如，下面的程序是错误的。

```
float f;
```

```
int a=4;
switch(f)
{
  case 1 :  a++;  break;
  case 2 :  a--;  break;
}
```

(11) if 或 switch 后面的表达式忘记了( )。

C 语言规定,if 或 switch 后面的表达式一定要加( ),否则会产生编译错。例如,下面的程序是错误的。

```
if  a > 10      //漏掉了( ),应改为: if(a > 10)
  b++;
```

习题 5

1. 填空题

(1) C 语言中的语句可以分为_____、_____、_____、_____、_____五类。

(2) C 语言用_____表示假,_____表示真。

(3) C 语言提供的三种逻辑运算符是_____、_____和_____。

(4) 关系运算符具有_____结合性,相同优先级的关系运算符连用时,按照_____的顺序计算表达式的值。

(5) 对于一个关系表达式的值,_____表示假,_____表示真。

(6) 对于 C 语言运算符的优先级,_____运算符优先级最高,_____运算符的优先级最低。

(7) C 语言中用于选择结构的控制语句有_____语句和_____语句两种,前者用于_____的情况,而后者用于_____的情形。

(8) 当 a=3, b=2, c=1 时,表达式 f=a>b>c 的值是_____。

(9) 当 m=2,n=1,a=1,b=2,c=3 时,执行完 d=(m=a!=b)&&(n=b>c)后; n 的值为 _____,m 的值为_____。

(10) 条件"2<x<3 或 x<-10"的 C 语言表达式是_____。

(11) 若 a=6,b=4,c=2,则表达式!(a-b)+c-1&&b+c/2 的值是_____。

(12) 设 x,y,z 均为 int 型变量,请写出描述"x,y 和 z 中有两个为负数"的表达式_____。

(13) 设有变量定义: a=5,c=4; 则(--a==++c) ? -a : C++的值是_____,此时 c 的存储单元的值为_____。

(14) 已知 A=7.5, B=2, C=3.6,表达式 A>B&&C>A ||A<B&&!C>B 的值是_____。

(15) 若 a=1,b=4,c=3,则表达式!(a<b)||!c&&1 的值是_____。

2. 选择题

(1) 有如下程序,该程序的输出结果是(　　)。

```
int main( )
{
  int x=1, a=0, b=0;
  switch( x ) {
    case 0: b++;
    case 1: a++;
    case 2: a++;  b++;
  }
  printf("a=%d, b=%d\n", a, b);
  return 0;
}
```

A. a=2, b=1　　B. a=1, b=1　　C. a=1, b=0　　D. a=2, b=2

(2) 若有如下程序,该程序的输出结果是(　　)。

```
int main( )
{
  float x=2.0, y;
  if(x<0.0) y=0.0;
  else if(x<10.0) y=1.0/x;
  else y=1.0;
  printf("%f\n", y);
  return 0;
}
```

A. 0.000000　　B. 0.250000　　C. 0.500000　　D. 1.000000

(3) 设有:int a=1, b=2, c=3, d=4, m=2, n=2;执行(m=a>b)&&(n=c>d)后 n 的值是(　　)。

A. 1　　B. 2　　C. 3　　D. 4

(4) 对 if 语句中表达式的类型,下面正确的描述是(　　)。

A. 必须是关系表达式

B. 必须是关系表达式或逻辑表达式

C. 必须是关系表达式或算术表达式

D. 可以是任意表达式

(5) 多重 if-else 语句嵌套使用时,寻找与 else 配对的 if 方法是(　　)。

A. 缩排位置相同的 if　　B. 其上最近的 if

C. 下面最近的 if　　D. 其上最近的未配对的 if

(6) 以下错误的 if 语句是(　　)。

A. if(x>y)　z=x;

B. if(x==y)　z=0;

C. if(x!=y)printf("%d", x)　else　printf("%d", y);

D. if(x<y)　{ x++;　y--; }

(7) 以下程序的输出为(　　)。

```
int main( )
{
  int a=20, b=30, c=40;
  if(a > b)   a=b,
  b=c;   c=a;
  printf("a=%d, b=%d, c=%d", a, b, c);
  return 0;
}
```

A. a=20，b=30，c=20　　B. a=20，b=40，c=20

C. a=30，b=40，c=20　　D. a=30，b=40，c=30

(8) 对于条件表达式(k)?(i++):(i--)来说,其中的表达式 k 等价于(　　)。

A. k==0　　B. k==1　　C. k !=0　　D. k !=1

(9) 下面程序运行结果为(　　)。

```
int main( )
{
  char c='a';
  if('a'< c <='z')   printf("LOW");
  else   printf("UP");
  return 0;
}
```

A. LOW　　B. UP　　C. LOWUP　　D. 程序语法错误

(10) 对下述程序,正确的判断是(　　)。

```
int main( )
{
  int a, b;
  scanf("%d, %d", &a, &b);
  if(a > b)   a=b;   b=a;
  else   a++;   b++;
  printf("%d, %d", a, b);
  return 0;
}
```

A. 有语法错误不能通过编译　　B. 若输入 4,5 则输出 5,6

C. 若输入 5,4 则输出 4,5　　D. 若输入 5,4 则输出 5,5

(11) 逻辑运算符两侧运算对象的数据类型(　　)。

A. 只能是 0 或 1　　B. 只能是 0 或非 0 正数

C. 只能是整型或字符型数据　　D. 可以是任何类型的数据

(12) 以下关于运算符优先顺序的描述中正确的是(　　)。

A. 关系运算符<算术运算符<赋值运算符<逻辑运算符

B. 逻辑运算符<关系运算符<算术运算符<赋值运算符

C. 赋值运算符<逻辑运算符<关系运算符<算术运算符

D. 算术运算符<关系运算符<赋值运算符<逻辑运算符

(13) 下列运算符中优先级最高的是(　　)。

A. < B. + C. && D. !=

(14) 若希望当 A 的值为奇数时，表达式的值为“真”，A 的值为偶数时，表达式的值为“假”，则以下不能满足要求的表达式是(　　)。

A. A % 2==1 B. !(A % 2==0)

C. !(A % 2) D. A % 2

(15) 判断 char 型变量 cl 是否为小写字母的正确表达式是(　　)。

A. 'a'<=cl<='z' B. (cl>=a) && (cl<=z)

C. ('a'>=cl)||('z'<=cl) D. (cl>='a') && (cl<='z')

(16) 已知 int x=10, y=20, z=30; 以下语句执行后 x, y, z 的值是(　　)。

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

A. x=10, y=20, z=30 B. x=20, y=30, z=30

C. x=20, y=30, z=10 D. x=20, y=30, z=20

(17) 请阅读以下程序：

```
int main( )
{
  int a=5, b=0, c=0;
  if(a=b+c)  printf("***\n");
  else       printf("$ $ $\n");
  return 0;
}
```

以上程序(　　)。

A. 有语法错不能通过编译 B. 可以通过编译但不能通过连接

C. 输出 *** D. 输出 $ $ $

(18) 请阅读以下程序，其运行结果是(　　)。

```
int main( )
{
  char c='A';
  if('0'<=c<='9')    printf("YES");
  else printf("NO");
  return 0;
}
```

A. YES B. NO C. YESNO D. 语句错误

(19) 当 a=1, b=3, c=5, d=4 时，执行完下面一段程序后 x 的值是(　　)。

```
if(a < b)
if(c < d)   x=1;
else
  if(a < c)
      if(b < d)   x=2;
      else   x=3;
  else   x=6;
```

```
else   x=7;
```

A. 1　　B. 2　　C. 3　　D. 6

(20) 已知 x=43，ch='A'，y=0；则表达式(x>=y&&ch<'B'&&!y)的值是(　　)。

A. 0　　B. 语法错　　C. 1　　D. “假”

(21) 有如下程序，正确的输出结果是(　　)。

```
int main( )
{
  int a=15, b=21, m=0;
  switch(a % 3)
  {
    case 0: m++; break;
    case 1: m++;
        switch(b % 2)
          {
            default: m++;
            case 0: m++; break;
          }
  }
  printf("%d\n", m);
  return 0;
}
```

A. 1　　B. 2　　C. 3　　D. 4

(22) 阅读以下程序，如果从键盘上输入 5，则正确的输出结果是(　　)。

```
int main( )
{
  int x;
  scanf("%d", &x);
  if(x-- < 5)     printf("%d", x);
  else   printf("%d", x++);
  return 0;
}
```

A. 3　　B. 4　　C. 5　　D. 6

(23) 若 a、b、c1、c2、x、y 均是整型变量，正确的 switch 语句是(　　)。

A.
```
switch(a+b)
  {
   case 1: y=a+b;   break;
   case 0: y=a-b;   break;
   case 3: y=b-a;   break;
  }
```

B.
```
switch(a*a+b*b)
  {
   case 3;
   case 1: y=a+b; break;
  }
```

C.
```
switch a
  {
   case c1: y=a-b;   break
   case c2: x=a*d;   break
```

D.
```
switch(a-b)
  {
   default: y=a*b;   break
   case 3: case 4: x=a+b;    break
```

```
        default: x=a+b;
    }
```

```
        case 10: case 11: y=a-b; break;
    }
```

(24) 与 y=(x>0?1:x<0?-1:0);的功能相同的 if 语句是(　　)。

A.
```
if(x>0)   y=1;
 else if(x<0)   y=-1;
 else   y=0;
```

B.
```
if(x)
 if(x>0)   y=1;
 else if(x<0)   y=-1;
```

C.
```
y=-1
if(x)
if(x>0)   y=1;
else if(x==0)   y=0;
else   y=-1;
```

D.
```
y=0;
if(x >=0)
if(x>0)   y=1;
else   y=-1;
```

(25) 若有定义：float w; int a, b; 则合法的 switch 语句是(　　)。

A.
```
switch(w){
 case 1.0:printf("*\n");
 case 2.0:printf("**\n");
}
```

B.
```
switch(a) {
 case 1: printf("*\n");
 case 2: printf("**\n");
}
```

C.
```
switch(b) {
 case 1:printf("*\n");
 default:printf("\n");
 case a:printf("**\n");
}
```

D.
```
switch(a+b); {
 case 1: printf("*\n");
 case 2: printf("**\n");
 default: printf("\n");
}
```

(26) 执行以下程序段后的输出结果是(　　)。

```
int w=3, z=7, x=10;
printf("%d ", x>10 ? x=100 : x-10);
printf("%d ", w++ || z++);
printf("%d ", !w>z);
printf("%d\n", w && z);
```

A. 0 1 1 1　　B. 1 1 1 1　　C. 0 1 0 1　　D. 0 1 0 0

(27) 以下程序的运行结果是(　　)。

```
int main( )
{
  int k=4, a=3, b=2, c=1;
  printf("\n%d\n", k<a ? k : c<b ? c : a);
  return 0;
}
```

A. 4　　B. 3　　C. 2　　D. 1

(28) 假定 w、x、y、z、m 均为 int 型变量，有如下程序段：

```
w=1; x=2; y=3; z=4;
m=(w<x)?w : x;  m=(m<y)?m : y;  m=(m<z)?m: z;
```

则该程序运行后，m 的值是(　　)。

A. 4　　B. 3　　C. 2　　D. 1

(29) 为表示关系 x≥y≥z,应使用 C 语言表达式(　　)。

A. (x>=y)&&(y>=z)　　B. (x>=y)AND(y>=z)

C. (x>=y>=z)　　D. (x>=y)||(y>=z)

(30) 以下程序的输出结果是(　　)。

```
int main( )
{
  int a=-1, b=4, k;
  k=(++a<0)&&!(b--<=0);
  printf("%d%d%d\n", k, a, b);
  return 0;
}
```

A. 104　　B. 103　　C. 003　　D. 004

3. 程序填空题

(1) 执行下面程序时,若从键盘上输入 8,则输出为 9,请填空。

```
int main( )
{
  int x;
  scanf("%d", &x);
  if(________> 8)
    printf("%d\n",++x);
  else printf("%d\n", x--);
  return 0;
}
```

(2) 执行下面程序时输出为 1,请填空。

```
int main( )
{
  int a=4, b=3, c=2, d=1;
  printf("%d\n",(a < b ? a : d < c ?________: b));
  return 0;
}
```

(3) 下面程序的功能是根据输入的百分制成绩 score,转换成相应的五分制成绩 grade 并打印输出。转换的标准为:当 90≤score≤100 时,grade 为 A;当 80≤score<90 时,grade 为 B;当 70≤score<80 时,grade 为 C;当 60≤score<70 时,grade 为 D;当 score<60 时,grade 为 E;请填空。

```
#include <stdio.h>
int main( )
{
  int score, mark;
  scanf("%d", ________);
  mark=________;
  switch(mark)  {
    default:printf("%d--E", score);
```

```
        _________
    case 10:
    case _________:printf("%d--A", score);   break;
    case _________:printf("%d--B", score);   break;
    case _________:printf("%d--C", score);   break;
    case _________:printf("%d--D", score);   break;
  }
  return 0;
}
```

(4) 以下程序是对四个数 a、b、c、d 从大到小的顺序输出,请填空。

```
#include <stdio.h>
int main( )
{
  int a, b, c, d, t;
  scanf("%d%d%d%d", &a, &b, &c, &d);
  if(a<b) {t=a; a=b; b=t;}
  if(_________){t=c; c=d; d=t;}
  if(a<c) {t=a; a=c; c=t;}
  if(_________){t=b; b=c; c=t;}
  if(b<d) {t=b; b=d; d=t;}
  if(c<d) {t=c; c=d; d=t;}
  printf("%d %d %d %d\n", a, b, c, d);
  return 0;
}
```

(5) 以下程序的功能是判断输入的年份是否是闰年,请填空。

```
#include <stdio.h>
int main( )
{
  int y, f;
  scanf("%d", &y);
  if(y%400==0) f=1;
  else if(_________) f=1;
  else _________;
  if(f)printf("%d is not ", y);
  printf("a leap year\n");
  return 0;
}
```

(6) 输入一个字符,如果它是一个大写字母,则把它变成小写字母;如果它是一个小写字母,则把它变成大写字母;其他字符不变,请填空。

```
#include <stdio.h>
int main( )
{
  char ch;
  scanf("%c", &ch);
  if(_________) ch=ch+32;
  else if(ch>='a' && ch<='z') _________;
  printf("%c", ch);
  return 0;
}
```

4. 编程题

(1) 编一程序判断输入整数的正负性和奇偶性。

(2) 编程判断输入数据的符号属性。

$$\text{sign}=\begin{cases}1 & x>0\\0 & x=0\\-1 & x<0\end{cases}$$

输入 x,打印出 sign 的值。

(3) 输入任意三个数 num1、num2、num3,按从小到大的顺序排序输出。

(4) 在屏幕上显示一张如下所示的时间表。

```
***** Time *****
1   morning
2   afternoon
3   night
Please enter your choice:
```

操作人员根据提示进行选择,程序根据输入的时间序号显示相应的问候信息,选择 1 时显示“Good morning”,选择 2 时显示“Good afternoon”,选择 3 时显示“Good night”,对于其他的选择显示“Selection error!”,用 switch 语句编程实现。

(5) 输入一个年份和月份,打印出该月份有多少天(考虑闰年),用 switch 语句编程。

(6) 运输公司对用户计算运费。路程(以 s 表示,单位为 km)越远,每千米运费越低。标准如下:

$s<250$	没有折扣
$250\leqslant s<500$	2%折扣
$500\leqslant s<1000$	5%折扣
$1000\leqslant s<2000$	8%折扣
$2000\leqslant s<3000$	10%折扣
$3000\leqslant s$	15%折扣

设每千米每吨货物的基本运费为 p,货物重为 w,距离为 s,折扣为 d,则总运费 f 的计算公式为:

$$f=p\times w\times s\times(1-d)$$

编一程序用于计算总运费。要求用 switch 语句来实现。

(7) 编一程序,对于给定的一个百分比制成绩,输出相应的五分制成绩。设 90 分以上为‘A’,80～89 分为‘B’,70～79 分为‘C’,60～69 分为‘D’,60 分以下为‘E’(用 switch 语句实现)。

(8) 编程实现:输入一个整数,判断它能否被 3,5,7 整除,并输出以下信息之一。

- 能同时被 3,5,7 整除;
- 能被其中两个数(要指出哪两个)整除;
- 能被其中一个数(要指出哪一个)整除;
- 不能被 3,5,7 任一个整除。