第 3 章　C 语言程序结构及相关语句

从程序流程的角度看，程序可以分为 3 种基本结构：顺序结构、分支结构、循环结构。这 3 种基本结构可以组成所有复杂程序。C 语言提供了多种语句来实现这些程序结构。本章将介绍这些基本语句及其应用，为后面各章的学习打下基础。

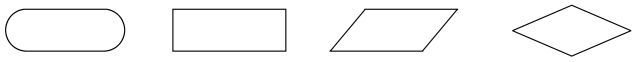
3.1　相关知识

3.1.1　算法描述方法

算法一般可以用以下两种方法进行描述。

（1）伪代码。采用近似高级语言但又不受语法约束的语言描述。

（2）流程图。传统的流程图常用的符号如图 3.1 所示，由这些框和流程线组成的流程图来表示算法，形象直观，简单方便，但在描述复杂算法时不易阅读。



(a) 开始或结束框　(b) 处理框　(c) 输入输出框　(d) 判断框

图 3.1　流程图常用符号

3.1.2　结构化程序

结构化程序设计方法是程序设计的先进方法，结构化程序设计是一种使用顺序、分支和循环共 3 种基本控制结构，并且使用这 3 种基本结构足以表达出各种其他形式的结构的程序设计方法。

（1）顺序结构。在执行时按照先后顺序逐条进行，没有分支，没有循环。例如后面介绍的赋值语句、输入输出语句等都可以构成顺序结构。顺序结构可用图 3.2 所示的流程图来表示。

（2）分支结构。分支结构也称为选择结构，根据不同的条件去执行不同分支中的语句，如后面章节中介绍的 if 语句、switch 语句等都可以构成分支结构。分支结构可用图 3.3 所示的流程图来表示。

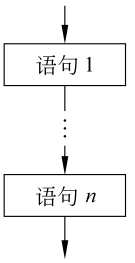


图 3.2　顺序结构流程图

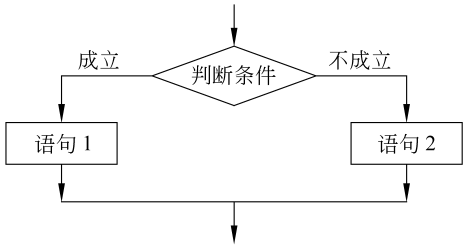
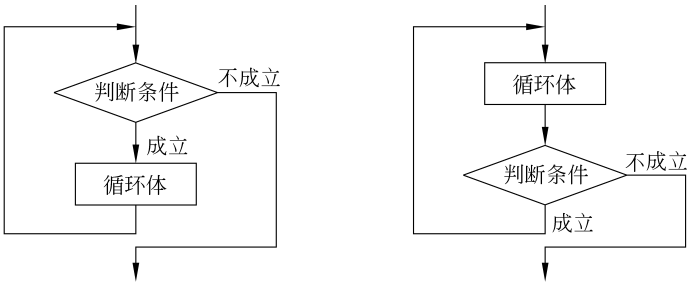


图 3.3　分支结构流程图

(3) 循环结构。根据各自的条件,使同一组语句重复执行多次或一次也不执行。循环结构包括当型循环(如图 3.4(a)所示)和直到型循环(如图 3.4(b)所示)。当型循环的特点是,当指定的条件满足时,就执行循环体,否则就不执行。直到型循环的特点是,执行循环体直到指定的条件不成立,就不再执行循环。



(a) 当型循环　　(b) 直到型循环

图 3.4　循环结构流程图

* 3.1.3　模块化结构

计算机在处理复杂任务时,常常需要把大任务分解为若干子任务,每个子任务又分解成很多个小子任务,每个小子任务只完成一项简单的功能。在程序设计时,用一个个模块来实现这些功能。这样的程序设计方法称为“模块化”程序设计方法,由一个个功能模块构成的程序结构就称为“模块化结构”。模块化结构可以大幅提高程序编制的效率。

C 语言是一种结构化程序设计语言。它直接提供了 3 种基本结构的语句;提供了定义“函数”的功能,用函数实现模块化结构。

3.2　顺 序 语 句

3.2.1　C 程序的语句

C 程序的执行部分由语句组成,程序的功能也由执行语句来实现。C 语句可分 5 类:表达式语句、函数调用语句、控制语句、复合语句和空语句。

1. 表达式语句

表达式语句由表达式加上分号(;)组成。一般形式如下:

```
表达式;
```

执行表达式语句就是计算表达式的值。例如:

```
c=a+b;
```

是赋值语句。

```
a+b;
```

是加法运算语句,计算结果不能保留,无实际意义。

```
i++;
```

是自增 1 语句,i 值增 1。

2. 函数调用语句

由函数名、实际参数表加上“;”组成。一般形式如下:

```
函数名(实际参数表);
```

执行函数语句是调用函数体并将实际参数赋予函数定义中的形式参数,然后执行被调用函数体中的语句,求取函数值。

3. 控制语句

控制语句用于控制程序的流程,它们由特定的语句定义符组成,用于实现程序的各种结构方式,在后面章节将进行详细介绍。

4. 复合语句

将任意多条语句用“{}”括起来组成的一个语句称为复合语句。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:

```
{
    k=i+j;
    a=b-c;
    printf("%d,%d",k,a);
}
```

是一条复合语句。复合语句内的各条语句都以“;”结尾,在“}”外不用加“;”。

5. 空语句

只由“;”组成的语句称为空语句。空语句什么也不执行。在程序中空语句可用作空循环体。例如:

```
while(getchar()!='\n');
```

本语句的功能是,只要从键盘输入的字符(用“getchar()”实现)不是回车符则重新输

入。这里的循环体为空语句。

3.2.2 数据输出语句

数据输出语句用于向标准输出设备显示器输出数据。在C语言中,所有数据输入输出都由库函数完成。因此都是函数语句。本节介绍printf()函数和putchar()函数。

1. 格式化输出函数printf()

printf()函数称为格式化输出函数,其中函数名最后一个字母f的含义是格式(format)。printf()函数的功能是按用户指定的格式,将指定的数据显示到显示器屏幕上。

1) printf()函数调用的一般形式

printf()函数为一个库函数,函数原型在头文件stdio.h中。printf()函数调用的一般形式如下:

```
printf("格式控制字符串",输出表)
```

其中,格式控制字符串用于指定输出格式。格式控制字符串由格式字符串和非格式字符串两种组成。格式字符串是以“%”开头的字符串,在“%”后面跟有各种格式字符,以说明输出数据的类型、长度、小数位数等。例如,%d表示按十进制整型输出,%ld表示按十进制长整型输出,%c表示按字符型输出,后面将专门给予讨论。

非格式字符串在输出时按原样显示,在显示中起提示作用。输出表中给出了各输出项,要求格式字符串和各输出项在数量和类型上应一一对应。

例3.1 printf()函数示例,程序演示见3011.mp4～3013.mp4。


```
/*文件路径名:e3_1\main.c*/
#include<stdio.h>                          /*包含库函数printf()所需要的信息*/

int main(void)                             /*主函数main()*/
{
    int a=65, b=66;                        /*定义变量*/

    printf("%d %d\n",a,b);                 /*格式化输出a和b*/
    printf("%d,%d\n",a,b);                 /*格式化输出a和b*/
    printf("%c,%c\n",a,b);                 /*格式化输出a和b*/
    printf("a=%d,b=%d\n",a,b);             /*格式化输出a和b*/

    return 0;                              /*返回值0,返回操作系统*/
}
```

程序运行时屏幕输出如下：

```
65 66
65,66
A,B
a=65,b=66
```

本例程序中 4 次输出了 a、b 的值，但由于格式控制串不同，输出的结果也不相同。第 1 个 printf()函数输出语句的格式控制字符串中，两个格式控制字符串%d 之间加了一个空格(非格式字符)，输出的 a、b 值之间有一个空格。第 2 个 printf()函数输出语句的格式控制字符串中加入的是非格式字符“,”，因此输出的 a、b 值之间加了一个“,”。第 3 个 printf()函数输出语句的格式字符串要求按字符型输出 a、b 值。第 4 个 printf()函数输出语句为了提示输出结果又增加了非格式字符串。

2) 格式字符串

格式字符串的一般形式如下：

```
%[标志][输出最小宽度][.精度][长度]类型
```

其中，“[]”中的项为可选项。各项的意义介绍如下。

① 类型。类型字符用于表示输出数据的类型，如表 3.1 所示。

表 3.1　printf()函数格式字符串中的类型字符

类 型 字 符	意　　义
d、i	d 和 i 都用于以十进制形式输出整数
o	以八进制形式输出无符号整数
x	以十六进制形式输出无符号整数
u	以十进制形式输出无符号整数
f	以小数形式输出单、双精度实数
e	以指数形式输出单、双精度实数
g	以%f 和%e 中较短的输出宽度输出单、双精度实数
c	输出单个字符
s	输出字符串

② 标志。标志字符有－、＋、空格、0 和＃这 5 种，其意义如表 3.2 所示。

表 3.2　printf()函数格式中的标志字符

标 志 字 符	意　　义
－	结果左对齐，右边填空格
＋	输出符号(正号或负号)
空格	输出值为正时冠以空格，为负时冠以负号

续表

标 志 字 符	意　　义
0	用 0 进行前位填充
#	对 c、s、d、u 类型无影响；对 o 类型，在输出时加前缀 0；对 x 类型，在输出时加前缀 0x；对 e、g、f 类型，当结果有小数时才给出小数点

③ 输出最小宽度。用十进制整数来表示输出的最少位数，如果实际位数多于指定的宽度，则按实际位数输出，如果实际位数少于指定的宽度则补以空格或 0。

④ 精度。精度格式符以“.”开头，后跟十进制整数。具体的意义是，如果输出实数，则表示小数的位数；如果输出的是字符串，则表示输出字符的最大个数。

⑤ 长度。长度格式符为 h、l 两种，h 表示按短整型量输出，l 表示按长整型量输出。

例 3.2　printf()函数格式字符串使用示例，程序演示见 3021.mp4～3023.mp4。


```
/* 文件路径名:e3_2\main.c */
#include<stdio.h>                          /* 包含库函数 printf()所需要的信息 */

int main(void)                             /* 主函数 main() */
{
    int a=168;                             /* 定义整型变量 */
    float b=1243.1698;                     /* 定义单精度实型变量 */
    double c=2421985.50168;                /* 定义双精度实型变量 */
    char d='a';                            /* 定义字符型变量 */

    printf("a=%d,%5d,%o,%x\n",a,a,a,a);          /* 输出 a */
    printf("b=%f,%lf,%5.4lf,%e\n",b,b,b,b);      /* 输出 b */
    printf("c=%lf,%f,%8.4lf\n",c,c,c);           /* 输出 c */
    printf("d=%c,%8c\n",d,d);                    /* 输出 d */

    return 0;                                  /* 返回值 0, 返回操作系统 */
}
```

程序运行时屏幕输出如下：

```
a=168,168,250,a8
b=1243.169800,1243.169800,1243.1698,1.243170e+003
c=2421985.501680,2421985.501680,2421985.5017
d=a,       a
```

本例第 1 个 printf()函数以 4 种格式输出整型变量 a 的值，其中%5d 要求输出宽度

为5，而a值为168只有3位，故补两个空格。第2个printf()函数以4种格式输出实型量b的值。其中%f和%lf格式的输出相同，说明符l对f类型无影响。%5.4lf指定输出宽度为5，精度为4，由于实际长度超过5，故应该按实际位数输出，小数位数超过4位部分被截去。第3个printf()函数输出双精度实数，%8.4lf由于指定精度为4位，故截去了超过4位的部分。第4个printf()函数输出字符量d，其中%8c指定输出宽度为8，故在输出字符'a'之前补加7个空格。

注意：不同的编译系统输出表的求值顺序不一定相同，可以从左到右，也可以从右到左。Visual C++ 6.0、Visual C++ 2022和Dev-C++ 5.11是按从右到左的顺序进行的。

例3.3 printf()函数输出表的求值顺序示例，程序演示见3031.mp4～3033.mp4。


```
/*文件路径名:e3_3\main.c*/
#include<stdio.h>                          /*包含库函数printf()所需要的信息*/

int main(void)                             /*主函数main()*/
{
    int i=6,a,b,c;                         /*定义变量*/
    printf("%d,%d,%d\n",a=++i,b=++i,c=++i);          /*输出各表达式的值*/

    return 0;                              /*返回值0,返回操作系统*/
}
```

在Visual C++ 6.0、Visual C++ 2022和Dev-C++ 5.11中，程序运行时屏幕输出如下：

```
9,8,7
```

2. 字符输出函数putchar()

putchar()函数用于输出字符，功能是在显示器上显示单个字符。函数原型在头文件stdio.h中，一般形式如下：

```
putchar(字符)
```

例如：

```
putchar('a');              /*用于输出小写字母a*/
putchar(x);                /*用于输出字符变量x的值*/
putchar('\n');             /*用于换行,对控制字符则执行控制功能,不在屏幕上显示*/
```

例3.4 putchar()函数使用示例，程序演示见3041.mp4～3043.mp4。

```
/* 文件路径名:e3_4\main.c */
#include<stdio.h>                          /* 标准输入输出头文件 */

int main(void)                             /* 主函数 main() */
{
    char a='A',b='o',c='m';                /* 定义字符型变量 */

    putchar(a); putchar(b); putchar(b); putchar('\t');
                                                    /* 输出字符, '\t'为制表符 */
    putchar(a); putchar(b); putchar('\n');          /* 输出字符, '\n'为换行符 */
    putchar(b); putchar(c); putchar('\n');          /* 输出字符, '\n'为换行符 */

    return 0;                                       /* 返回值 0, 返回操作系统 */
}
```

程序运行时屏幕输出如下：

```
Aoo    Ao
om
```

3.2.3 数据输入语句

C 语言的数据输入是由函数语句完成的。本节介绍从标准输入设备键盘上输入数据的函数 scanf()和 getchar()。

1. 格式化输入函数 scanf()

scanf()函数称为格式化输入函数，其功能是按用户指定的格式把从键盘输入的数据传递给指定的变量。

1) scanf()函数的一般形式

scanf()函数是一个库函数，它的函数原型在头文件 stdio.h 中。scanf()函数的一般形式如下：

```
scanf("格式控制字符串",地址表);
```

其中，格式控制字符串的作用与 printf()函数相同，但不显示非格式字符串，也就是不能显示提示字符串。地址表中给出各变量的地址。地址由取地址运算符“&”后跟变量名组成的。例如，&x 表示变量 x 的地址。此地址就是编译系统在内存中给 x 变量分配的地址。应该把变量的值和变量的地址这两个不同的概念区别开。变量的地址是 C 编译系统分配的，用户不必关心具体的地址是多少。在赋值表达式中给变量赋值，例如，x＝168

在赋值号左边是变量名，不能写地址，而 scanf()函数要求写变量的地址，如 &x。这两者在形式上是不同的。“&”是一个取地址运算符，&x 是一个表达式，其功能是求变量的地址。

例 3.5 scanf()函数使用示例。

```
/* 文件路径名:e3_5_1\main.c */
#include<stdio.h>                                   /* 标准输入输出头文件 */

int main(void)                                      /* 主函数 main() */
{
    int a,b;                                        /* 定义变量 */

    printf("输入 a,b:");                            /* 输入提示 */
    scanf("%d%d",&a,&b);                            /* 输入 a 和 b */
    printf("a=%d,b=%d\n",a,b);                      /* 输出 a 和 b */

    return 0;                                       /* 返回值 0, 返回操作系统 */
}
```

说明：在 Visual C++ 6.0 及 Dev-C++ 5.11 中，C 中关于格式化输入函数 scanf()，在 5.4.3 节中的字符串函数 strcpy()和 strcat()，以及 8.3.1 节中的文件打开函数 fopen()都能正常编译，但在版本比较新的 Visual C++（例如 Visual C++ 2022)中，在上述几个函数调用时会出现如下编译时的错误。

```
错误 C4996'scanf': This function or variable may be unsafe. Consider using scanf_s
instead. To disable deprecation, use _CRT_SECURE_NO_WARNINGS. See online help
for details.
```

程序演示见 30511.mp4～30513.mp4。

为什么会报错？这是因为微软公司认为函数 scanf()、strcpy()、strcat()和 fopen()不够安全，解决这类问题的最简捷方式是在文件开头添加编译预处理命令“# pragma warning(disable:4996)”禁止对代号为 4996 的警告，此处 pragma 的含义是编译指示，“warning(disable:4996)”的含义是禁止对代号为 4996 的警告，关于编译预处理命令 # pragma 的详细使用方法，请在网上查阅相关资料。

修改后的程序如下：

```
/* 文件路径名:e3_5_2\main.c */
#pragma warning(disable:4996)          /* 禁止对代号为 4996 的警告 */
#include<stdio.h>                      /* 标准输入输出头文件 */
```

```
int main(void)                          /*主函数 main()*/
{
    int a,b;                            /*定义变量*/

    printf("输入 a、b:");               /*输入提示*/
    scanf("%d%d",&a,&b);                /*输入 a 和 b*/
    printf("a=%d,b=%d\n",a,b);          /*输出 a 和 b*/

    return 0;                           /*返回值 0,返回操作系统*/
}
```

修改后的程序在 Visual C++ 6.0、Visual C++ 2022 及 Dev-C++ 5.11 中都能正常运行,程序演示见 30521.mp4～30523.mp4。

在本例中,由于 scanf()函数本身不能显示提示,所以先用 printf()语句在屏幕显示输入提示,请用户输入 a 和 b 的值。执行 scanf()语句,等待用户输入。用户输入 6、8 后按 Enter 键,然后执行 printf()函数。在 scanf()函数的格式控制字符串中由于没有非格式字符在 %d%d 之间作输入时的间隔,因此在输入时要用一个以上的空格、换行回车符或 Tab 符作为每两个输入数之间的间隔。

例如:

```
6 8
```

或

```
6
8
```

程序运行时屏幕输出如下:

```
输入 a、b:6 8
a=6,b=8
```

2) 格式字符串

格式字符串的一般形式如下:

```
%[*][输入数据宽度][长度]类型
```

其中,有"[]"的项为任选项。各项的意义如下。

① 类型。表示输入数据的类型,如表 3.3 所示。

表 3.3　scanf()函数格式字符串中的类型字符

类型字符	意　　义	类型字符	意　　义
d、i	d 和 i 都用于输入十进制整数	e、f	输入实型数(e 和 f 都适用于输入小数形式与输入指数形式)
o	输入八进制整数	c	输入单个字符
x	输入十六进制整数	s	输入字符串
u	输入无符号十进制整数		

② 星号(*)。表示该输入项读入后不赋予相应的变量,即跳过该输入值。例如

```
scanf("%d %* d %d",&a,&b);
```

当输入为“6 8 9”时,把 6 赋予 a,8 被跳过,9 赋予 b。

③ 宽度。用十进制整数指定输入的宽度(即字符数)。例如:

```
scanf("%5d",&a);
```

当输入

```
1234567890
```

后,只把 12345 赋予变量 a,其余部分被截去。又如:

```
scanf("%4d%4d",&a,&b);
```

当输入

```
1234567890
```

后,将把 1234 赋予 a,而把 5678 赋予 b。

④ 长度。长度格式符为 l 和 h,l 用于表示输入长整型数据(如%ld)和双精度浮点数(如%lf)。h 表示输入短整型数据。

使用 scanf()函数应注意以下几点。

① scanf()函数中没有精度控制,例如:

```
scanf("%5.2f",&a);
```

是非法的。用户不能企图用此语句输入小数为 2 位的实数。

② scanf()函数中要求给出变量地址,如给出变量名则会出错。例如:

```
scanf("%d",i);
```

是非法的,应改为

```
scanf("%d",&i);
```

才是合法的。

③ 在输入字符数据时,若格式控制字符串中无非格式字符,则认为所有输入的字符均为有效字符。例如:

```
scanf("%c%c",&a,&b);
```

若输入

```
d e
```

则把'd'赋予 a，' ' 赋予 b。只有当输入为

```
de
```

时，才能把'd'赋予 a，把'e'赋予 b。

如果在格式控制字符串中加入空格作为间隔，例如：

```
scanf("%c %c",&a,&b);
```

则所输入的两个字符之间可加空格。

例 3.6 scanf()函数使用示例，程序演示见 3061.mp4～3063.mp4。


```
/* 文件路径名:e3_6\main.c */
#pragma warning(disable:4996)              /* 禁止对代号为 4996 的警告 */
#include<stdio.h>                          /* 标准输入输出头文件 */

int main(void)                             /* 主函数 main() */
{
    char a,b;                              /* 定义变量 */

    printf("输入字符 a,b:");               /* 输入提示 */
    scanf("%c%c",&a,&b);                   /* 输入 a、b */
    printf("%c%c\n",a,b);                  /* 输出 a、b */

    return 0;                              /* 返回值 0，返回操作系统 */
}
```

本例中，scanf()函数的格式控制字符串"%c%c"中没有空格，输入 M N 时，a 的值为'M'，b 的值为' '。

程序运行时屏幕输出如下：

```
输入字符 a,b:M N
M
```

而输入改为 MN 时，a 的值为'M'，b 的值为'N'。

程序运行时屏幕输出如下：

```
输入字符 a,b:MN
MN
```

例 3.7 scanf()函数使用示例，程序演示见 3071.mp4～3073.mp4。

```
/* 文件路径名:e3_7\main.c */
#pragma warning(disable:4996)                    /* 禁止对代号为 4996 的警告 */
#include<stdio.h>                                /* 标准输入输出头文件 */

int main(void)                                   /* 主函数 main() */
{
    char a,b;                                    /* 定义变量 */

    printf("输入字符 a,b:");                     /* 输入提示 */
    scanf("%c %c",&a,&b);                        /* 输入 a、b */
    printf("%c%c\n",a,b);                        /* 输出 a、b */

    return 0;                                    /* 返回值 0, 返回操作系统 */
}
```

本例 scanf()函数的格式控制字符串"%c %c"之间有空格,输入的数据之间可以有空格间隔。

程序运行时屏幕输出如下:

```
输入字符 a,b:M N
MN
```

④ 如果格式控制字符串中有非格式字符,则输入时也要输入该非格式字符。

例如 scanf("%d,%d ",&a,&b);其中用非格式字符“,”作间隔符,故输入时应为

```
5,6
```

又如:

```
scanf("a=%d,b=%d,c=%d",&a,&b,&c);
```

则输入应为

```
a=5,b=6,c=7
```

2. 字符输入函数 getchar()

getchar()函数的功能是从键盘上输入一个字符。函数原型在头文件 stdio.h 中,一般形式如下:

```
getchar()
```

注意:getchar()函数只能接收单个字符,输入数字也按字符处理。输入多于一个字

符时，只接收第一个字符。

例 3.8 getchar()函数使用示例，程序演示见 3081.mp4～3083.mp4。


```
/* 文件路径名:e3_8\main.c */
#include<stdio.h>                    /* 标准输入输出头文件 */

int main(void)                       /* 主函数 main() */
{
    char c;                          /* 定义变量 */

    c=getchar();                     /* 输入 c */
    printf("%c\n",c);                /* 输出 c */

    return 0;                        /* 返回值 0, 返回操作系统 */
}
```

程序运行时屏幕输出如下：

```
A
A
```

如果用户输入“AB”，则将字符'A'赋值给 c，字符'B'将被丢失。

程序运行时屏幕输出如下：

```
AB
A
```

3.3 分支结构程序

3.3.1 关系运算符和表达式

1. 关系运算符

在程序中经常需比较两个量的大小关系，以便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。在 C 语言中有以下关系运算符：＜(小于)、＜＝(小于或等于)、＞(大于)、＞＝(大于或等于)、＝＝(等于)、!＝(不等于)。

关系运算符都是双目运算符，都具有左结合性。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，“＜”“＜＝”“＞”“＞＝”的优先级都为 6，“＝＝”和“!＝”的优先级都为 7，“＜”“＜＝”“＞”“＞＝”的优先级高于“＝＝”“!＝”的优先级。

2. 关系表达式

关系表达式一般形式如下：

```
表达式 关系运算符 表达式
```

例如：

```
a-b>c*d,x>3*2,'a'+6<c,6+8*j==k-1;
```

都是合法的关系表达式。

关系表达式的值是“真”和“假”，“真”用“1”和“假”用“0”表示。例如，8＞0 的值为“真”，即为 1。(a＝6)＞(b＝8)由于 6＞8 不成立，其值为假，即为 0。

例 3.9 关系表达式使用示例，程序演示见 3091.mp4～3093.mp4。


```
/*文件路径名:e3_9\main.c*/
#include<stdio.h>                    /*标准输入输出头文件*/

int main(void)                       /*主函数 main()*/
{
    int i=1,j=6,k=8;                 /*定义整型变量*/
    double x=3e+5,y=0.86;            /*定义实型变量*/

    printf("%d\n",-i+2*j>=k+1);      /*输出关系表达式的值*/
    printf("%d\n",x-1.98<=x+y);      /*输出关系表达式的值*/
    printf("%d\n",k==j==i+6);        /*输出关系表达式的值*/

    return 0;                        /*返回值 0, 返回操作系统*/
}
```

程序运行时屏幕输出如下：

```
1
1
0
```

本例中求出各关系运算符的值。对于含多个关系运算符的表达式，例如 k＝＝j＝＝i＋6，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＋6，也不成立，故表达式值为 0。

3.3.2 逻辑运算符和表达式

1. 逻辑运算符

在 C 语言中，有 3 种逻辑运算符：&&(与)、‖(或)、!(非)。

运算符“&&”和“‖”都是双目运算符。具有左结合性。“!”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如下。

(1) “!”的优先级为 2,“&&”的优先级为 11,“‖”的优先级为 12,“!”的优先级高于“&&”的优先级,“&&”的优先级高于“‖”的优先级。

(2) 逻辑运算符中的“&&”和“‖”的优先级低于关系运算符的优先级并且高于赋值运算符,“!”的优先级高于算术运算符的优先级,算术运算符的优先级高于关系运算符的优先级,如图 3.5 所示。

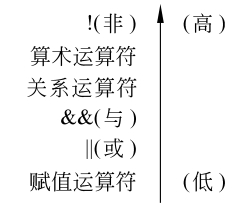


图 3.5　逻辑运算符优先级

根据运算符的优先顺序可得:

```
x>y&&u>v                    /*等价于  (x>y)&&(u>v)
!a==b‖c<d                   /*等价于  ((!a)==b)‖(c<d)
a+b>c&&d+e<f                /*等价于  ((a+b)>c)&&((d+e)<f)
```

2. 逻辑运算的值

逻辑运算的值也为“真”和“假”两种,用“1”表示“真”和用“0”表示“假”。求值规则如下。

(1) 与(&&)。参与运算的两个量都为真时,结果才为真,否则为假。例如,6>0 ‖ 8>1,由于 6>0 为真,8>1 也为真,所以“&&”运算的结果也为真。

(2) 或(‖)。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,6>0‖8<1,由于 6>0 为真,所以“‖”运算的结果也为真。

(3) 非(!)。参与运算量为真时,结果为假;参与运算量为假时,结果为真。例如,!(6>0)的结果为假。

说明:虽然 C 编译器在处理逻辑运算时,以“1”表示“真”,“0 ”表示“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 6 和 8 均为非“0”,因此 6 && 8 的值为“真”,即为 1。又如,6‖0 的值为“真”。

3. 逻辑表达式

逻辑表达式的一般形式如下:

```
表达式  逻辑运算符  表达式
```

其中的表达式可以又是逻辑表达式,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,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的值,以“1”表示“真”和“0”表示 “假”。

例 3.10　逻辑表达式使用示例,程序演示见 3101.mp4～3103.mp4。


```
/* 文件路径名:e3_10\main.c */
#include<stdio.h>                          /* 标准输入输出头文件 */
int main(void)                             /* 主函数 main() */
{
    char c='k';                            /* 定义字符型变量 */
    int i=1,j=6,k=8;                       /* 定义整型变量 */
    double x=3e+5,y=0.86;                  /* 定义实型变量 */

    printf("%d\n",!x*!y);                  /* 输出逻辑表达式的值 */
    printf("%d\n",i<j&&x<y);               /* 输出逻辑表达式的值 */
    printf("%d\n",i==6&&c&&(j=8));         /* 输出逻辑表达式的值 */

    return 0;                              /* 返回值 0, 返回操作系统 */
}
```

程序运行时屏幕输出如下：

```
0
0
0
```

本例中!x 和!y 都为 0，!x * !y 也为 0，故其输出值为 0。对于 i<j && x<y 式，由于 i<j 的值为 1，而 x<y 为 0，故表达式的值为 1，0 相与，最后为 0，对于 i==6 && c && (j=8)表达式，由于 i==6 为假，即值为 0，由于此表达式由两个与运算组成，整个表达式的值为 0。

3.3.3 if 语句

用 if 语句构成分支结构。根据给定的条件进行判断，以决定执行某分支程序段。C 语言的 if 语句共有 3 种基本形式。

1. if 语句第 1 种形式

第 1 种形式为基本形式，格式如下：

```
if(表达式) 语句
```

语义：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。执行过程流程如图 3.6 所示。

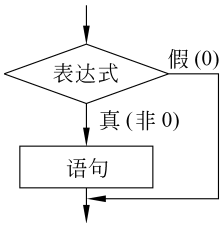


图 3.6　if 语句第 1 种形式执行过程流程

例 3.11　输入两个整数，输出其中的大值，程序演示见 3111.mp4～3113.mp4。


```
/* 文件路径名:e3_11\main.c */
#pragma warning(disable:4996)                    /* 禁止对代号为 4996 的警告 */
#include<stdio.h>                                /* 标准输入输出头文件 */

int main(void)                                   /* 主函数 main() */
{
    int a,b,max;                                 /* 定义变量 */

    printf("输入两个数:");                       /* 输入提示 */
    scanf("%d%d",&a,&b);                         /* 输入 a、b */
    max=a;                                       /* 假设 a 最大 */
    if (max<b) max=b;                            /* 如果 b 更大,则最大值为 b */
    printf("最大值:%d\n",max);                   /* 输出 a、b 的最大值 */

    return 0;                                    /* 返回值 0, 返回操作系统 */
}
```

程序运行时屏幕输出如下：

```
输入两个数:1 6
最大值:6
```

本例中，输入两个数 a、b。先将 a 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。这样 max 就是 a、b 的最大值，最后输出 max 的值。

2. if 语句第 2 种形式

第 2 种形式为 if…else 形式，格式如下：

```
if (表达式) 语句 1
else 语句 2
```

语义：如果表达式的值为真，则执行语句 1，否则执行语句 2。执行过程流程如图 3.7 所示。

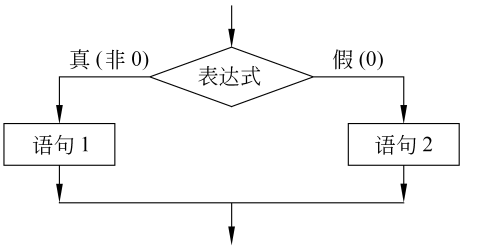


图 3.7　if 语句第 2 种形式执行过程流程

例 3.12　使用 if 语句第 2 种形式改写例 3.11，程序演示见 3121.mp4～3123.mp4。


```
/* 文件路径名:e3_12\main.c */
#pragma warning(disable:4996)                /* 禁止对代号为 4996 的警告 */
#include<stdio.h>                            /* 标准输入输出头文件 */

int main(void)                               /* 主函数 main() */
{
    int a,b;                                 /* 定义变量 */
    printf("输入两个数:");                    /* 输入提示 */
    scanf("%d%d",&a,&b);                     /* 输入 a、b */
    if (a<b) printf("最大值:%d\n",b);         /* 如果 a<b,则最大值为 b */
    else printf("最大值:%d\n",a);            /* 否则,最大值为 a */

    return 0;                                /* 返回值 0, 返回操作系统 */
}
```

程序运行时屏幕输出如下：

```
输入两个数:1 6
最大值:6
```

本例中，要求输入两个整数，输出其中的最大值。改用 if…else 语句判别 a、b 的大小，如果 b 大，则输出 b，否则输出 a。

例 3.13　写程序，判断某一年是否为闰年。

根据历法，闰年的条件为，年份能被 4 整除但不能被 100 整除或年份能被 400 整除。

具体程序实现如下，程序演示见 3131.mp4～3133.mp4。



```
/* 文件路径名:e3_13\main.c */
#pragma warning(disable:4996)                    /* 禁止对代号为 4996 的警告 */
#include<stdio.h>                                /* 标准输入输出头文件 */

int main(void)                                   /* 主函数 main() */
{
    int year;                                    /* 年份 */

    printf("输入年份:");                          /* 输入提示 */
    scanf("%d", &year);                          /* 输入 year */
    if (year%4==0&&year%100!=0                   /* 如果年份能被 4 整除但不能被 100 整除 */
        ||year%400==0) printf("%d 是闰年\n", year);
                                                 /* 或年份能被 400 整除,则为闰年 */
    else printf("%d 不是闰年\n", year);          /* 否则为平年 */

    return 0;                                    /* 返回值 0, 返回操作系统 */
}
```

程序运行时屏幕输出如下：

```
输入年份:2024
2024 不是闰年
```

本题使用 if else 语句直接写出闰年条件，程序简洁，可读性强。

3. if 语句第 3 种形式

第 3 种形式为 if…else…if 形式，前面第 2 种形式的 if 语句一般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，其一般形式如下：

```
if (表达式 1) 语句 1;
else if (表达式 2) 语句 2
…
else if (表达式 m) 语句 m
else 语句 m+1
```

语义：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 $m+1$。然后继续执行后续程序。if…else…if 语句的执行过程流程如图 3.8 所示。

说明：可以省略“else 语句 $m+1$”，省略“else 语句 $m+1$”后，如果所有的表达式均为假，则直接执行 if 语句的后续程序。