第 3 章　运算符和表达式

【本章概述】

C 语言中的运算符有算术运算符、关系运算符、逻辑运算符、赋值运算符、条件运算符、位运算符、逗号运算符、指针运算符、强制类型转换运算符、分量运算符、下标运算符以及求字节数运算符等多种类型。而表达式也有算术表达式、逻辑表达式、赋值表达式等多种类型，这些内容是 C 语言的基础，需要认真学习和领会。

【学习要求】

- 掌握：算术运算符、关系运算符、逻辑运算符等常见运算符。
- 掌握：各种运算符的优先级。
- 掌握：各种表达式的组成及运算过程。
- 重点：运算符的优先级及表达式分析计算。
- 难点：自增、自减运算符。

3.1　算术运算符和算术表达式

3.1.1　算术运算符

C 语言提供了 7 种算术运算符，如表 3-1 所示。

表 3-1　算术运算符

类　型	含　义	示　例	优先级	结 合 方 向
＋	加	5＋8	4	从左到右
－	减或取负	6－7 或－4	4	为减号时从左到右，取负时从右到左
*	乘	12 * 4	3	从左到右
/	除	45/7	3	从左到右
%	取余	54%8	3	从左到右
＋＋	自增	i＋＋或＋＋i	2	从右到左
－－	自减	j－－或－－j	2	从右到左

自增 1 运算符记为＋＋，其功能是使变量的值自增 1。自减 1 运算符记为－－，其功能是使变量值自减 1。

自增 1，自减 1 运算符均为单目运算，都具有右结合性，有以下几种形式：

(1) ＋＋i：i 自增 1 后再参与其他运算。

(2) －－i：i 自减 1 后再参与其他运算。

(3) i＋＋：i 参与其他运算后其值再自增 1。

(4) i－－：i 参与其他运算后其值再自减 1。

在理解和使用上容易出错的是i++和i--，特别是当它们出现在较复杂的表达式或语句中时，常常难于弄清，因此应仔细分析。

对于除法运算符/而言，若两侧均为字符型或整型量，则结果必定为整数；若有一侧是浮点型，则相除的结果必定为双精度数值。因此算术表达式7/2的结果为3，而7.0/2的结果为3.500000。

对取余数运算符%而言，要求运算符两侧的运算数必须是整型（数值、变量或表达式）或者是字符型（数值、变量或表达式），如9%2、'a'%10等。

【例3.1】 算术运算符的使用。

```
#include <stdio.h>
void main()
{
    int i=8;
    printf("%d ",++i);
    printf("%d ",--i);
    printf("%d ",i++);
    printf("%d ",i--);
    printf("%d ",-i++);
    printf("%d\n",-i--);
}
```

i的初值为8，第1个printf中i先加1后输出结果为9；第2个printf中i先减1后输出结果为8；第3个printf中先输出i的值8之后再加1(i为9)；第4个printf先输出i的值9之后再减1(i为8)；第5个printf中先输出i的值－8之后再加1(i为9)，第6个printf中先输出i的值－9之后再减1(i为8)。程序运行结果如下：

```
9 8 8 9 -8 -9
```

【例3.2】 算术运算符的使用。

```
#include <stdio.h>
void main()
{
    int i=5,j=5,p,q;
    p=(i++)+(i++)+(i++);
    q=(++j)+(++j)+(++j);
    printf("%d,%d,%d,%d\n",p,q,i,j);
}
```

这个程序中，对p=(i++)+(i++)+(i++)应理解为：首先是3个i相加，故p值为15；然后i再自增1三次，相当于加3，故i的最后值为8。q=(++j)+(++j)+(++j)应理解为q=((++j)+(++j))+(++j)，即j先自增1两次，相当于加2变为7，两个7相加为14，然后再执行最后一个++j运算，此时j再自增1一次变为8，前面的结果14加8结果为22，j的最后值为8。

程序运行结果如下：

15,22,8,8

在执行类似 q=(++j)+(++j)+…+(++j)的运算中,只要右侧++j 的表达式多于两个,都是先对前两个++j 表达式计算(j 连续自增 1 两次,求和),然后顺序计算每个++j 表达式(j 先自增 1 一次,求和)。如 j=5,则 q=(++j)+(++j)+(++j)+(++j)+(++j)+(++j)运算执行后的结果是:q=52,j=11。

3.1.2 算术表达式

算术表达式是由算术运算符和括号将运算对象连接起来的式子,其中运算对象可以是常量、变量、函数和数组元素等内容。算术表达式的一般组成形式为

表达式 1 算术运算符 表达式 2…

例如:

```
a*x*x+b*x+c
s=s+i
```

【例 3.3】 若 a=5,b=3,c=2,x=1.5,则算术表达式 a*x*x+b*x+c 的结果是多少?

编程分析:对于表达式结果的求解,需要从表达式的优先级入手。整体上来看,对于表达式的求解是按照从左到右的顺序进行的;对于具体的某个运算对象而言,若左右两侧均有运算符,则需要考虑哪一侧的优先级高,如第二个 x,其左侧为*,右侧为+,而*的优先级要比+的优先级高,因此需要先结合*,然后再结合+。同理,对 b 而言,先结合右侧的*,然后再将结果与左侧的结果相加,最终结果为 5*1.5*1.5+3*1.5+2=17.75。

【例 3.4】 已知 int a=10,b=5,c=4;,计算表达式 a*b/c-1.25+'a'的值。

编程分析:b 左侧为*,右侧为/,结合方向为从左向右;c 左侧为/,右侧为-,按运算符优先级结合,a*b/c 的计算结果为 12。1.25 是单精度数,结果的数据类型转换为双精度。字符'a'在运算过程中用其 ASCII 码值表示,由于左侧计算的结果是双精度型,因此计算过程中将字符'a'的 ASCII 码值转换为双精度型参与运算,整个表达式的计算结果为 107.750000。

注意:

(1) 当运算符/两侧的操作数为整型时,计算结果为整型,如 19/2=9。

(2) 运算符%两侧的操作数必须是基本整型、短整型或长整型,而不能是浮点型。

(3) 运算符++、--只能用于变量,而不能用于常量或表达式。例如,i++、--j 均是正确的,而++10、(a+b)++则是错误的。

(4) C 语言中,运算符的运算优先级共分为 15 级。1 级最高,15 级最低。在表达式中,优先级较高的先于优先级较低的进行运算。而在一个运算量两侧的运算符优先级相同时,则按运算符的结合性所规定的结合方向处理。

(5) 运算符的结合性:C 语言中各运算符的结合性分为两种,即左结合性(自左至右)和右结合性(自右至左)。例如,算术运算符的结合性是自左至右,即先左后右。如有表达式 x-y+z,则 y 应先与-号结合,执行 x-y 运算,然后再执行+z 的运算。这种自左至右的

结合方向就称为左结合性。而自右至左的结合方向称为右结合性。最典型的右结合性运算符是赋值运算符。如 x=y=z,由于=的右结合性,应先执行 y=z 再执行 x=(y=z)运算。C 语言运算符中有不少为右结合性,应注意区别,以避免理解错误。

3.2 赋值运算符和赋值表达式

3.2.1 赋值运算符

C 语言提供了 11 种赋值运算符,如表 3-2 所示。

表 3-2 赋值运算符

类 型	含 义	示 例	优先级	结合方向
=	赋值	a=b+3	14	从右到左
+=	加等于	a+=b	14	从右到左
-=	减等于	a-=2	14	从右到左
*=	乘等于	a*=3	14	从右到左
/=	除等于	a/=(a+3)	14	从右到左
%=	取余等于	a%=b	14	从右到左
>>=	右移等于	a>>=1	14	从右到左
<<=	左移等于	a<<=2	14	从右到左
&=	按位与等于	a&=b	14	从右到左
^=	按位异或等于	a^=b	14	从右到左
|=	按位或等于	a|=b	14	从右到左

=运算符是一种基本赋值运算符,而其他赋值运算符则是复合赋值运算符,是对左侧的变量先进行某种运算之后再将结果赋值给该变量。

3.2.2 赋值表达式

赋值表达式是由赋值运算符和括号将运算对象连接起来的式子,其中运算对象可以是常量、变量、函数和数组元素等内容。赋值表达式的一般组成形式为

变量名称 赋值运算符 表达式

例如:

```
z=x+y
y=sin(angle*3.14159/180)
```

对于变量自身参与运算过程并将计算结果重新赋值给该变量的,如 a=a+5,可以将+和=运算符合在一起,构成复合赋值运算符,记为+=,因此上式可以写成 a+=5。表格中其他复合赋值运算符均是同一道理。

在赋值运算符=之前加上其他双目运算符可构成复合赋值运算符。构成复合赋值表达式的一般形式为

变量名称 双目运算符=表达式

它等效于

变量名称=变量名称 运算符 表达式

例如：

```
a+=5        等价于 a=a+5
x*=y+7      等价于 x=x*(y+7)
r%=p        等价于 r=r%p
```

对于复合赋值运算符这种写法，初学者可能不习惯，但它十分有利于编译处理，能提高编译效率并产生质量较高的目标代码。

【例 3.5】 已知 int a=5,b=3,x=10,计算如下表达式的值。

```
a=a*8                表达式值为 40,a=40
b%=2                 表达式值为 1,b=1
x*=(a+b)             表达式值为 410,x=410
a=b=c=5              表达式值为 5,a、b、c 值为 5
a=(b=5)              表达值为 5,b=5,a=5
a=5+(c=6)            表达式值为 11,c=6,a=11
a=(b=4)+(c=6)        表达式值为 10,a=10,b=4,c=6
a=(b=10)/(c=2)       表达式值为 5,a=5,b=10,c=2
```

注意：

(1) 对于简单赋值运算符和复合赋值运算符，等号左侧的操作数只能是变量，而不能是常量或表达式，如 a=5 是正确的，而 10=3+2 以及 a+b=8 都是错误的。

(2) 赋值运算符具有右结合性，因此 a=b=c=10 是正确的，等价于 a=(b=(c=10))。

(3) 当=两侧的数据类型不同时，将要进行数据类型的转换。例如，int a=3.5，此处等号左侧是整型变量 a，而右侧是单精度浮点数 3.5，赋值过程中将右侧的单精度数截取其整数部分赋给整型变量 a，因此 a 的数值为 3。具体转换规则如下：

① 实型赋给整型变量，舍去小数部分，如前例。

② 整型赋给实型变量，数值不变，但以浮点数形式存放，即增加小数部分(小数部分的值为 0)，如 float f=123，则 f 的结果为 123.000000。

③ 字符型赋给整型变量，由于内存中字符型占 1 字节，而整型占 4 字节，故将字符的 ASCII 码值放到整型变量的最低字节中，其余三个字节为 0。例如，int a='a'，则取该字符的 ASCII 码值赋给整型变量 a，则 a 的结果为 97。

④ 整型赋给字符型变量，只把最低字节值赋予字符变量。例如，char c=12345，12345 的十六进制为 0X3039，舍去高字节内容 30 后，取其低字节内容(0X39)赋给字符变量 c，则 c 的结果为 57。

3.2.3 赋值语句

在赋值表达式的基础上添加“;”就构成了赋值语句。例如，x=(a=4)+8;，计算时先计算右侧括号中的内容，然后与 8 相加，并将结果赋给变量 x。

【例 3.6】 已知华氏温度与摄氏温度之间的转换公式为 $c=5(f-32)/9$，编写程序将输入的华氏温度转换为摄氏温度输出。

编程分析：题目中已经说明了二者之间的转换公式，输入待转换的华氏温度数值，转换为摄氏温度的过程可以用算术表达式来表示，计算结果由赋值运算符＝来完成，最后输出摄氏温度值，编写过程中注意除号运算符的计算结果。

参考程序如下：

```
#include "stdio.h"
void main()
{
    int f;
    float c;
    scanf("%d",&f);                    /*输入华氏温度*/
    c=5/9*(f-32);
    printf("c=%f\n",c);                /*输出结果*/
}
```

输入 40 时，程序运行结果如下：

```
input a real number to F:40
c=0.000000
```

从输出内容来看，并不是期望的结果，究其原因，主要是等号右侧的/运算符造成的，由于其两侧均为整型数值(5 和 9)，二者相除的结果必定为整型数，此处结果为 0，因此导致整个表达式的结果为 0。此处可以将 5/9 更改为 5.0/9 或(float)5/9 等形式。

3.3 关系运算符和关系表达式

3.3.1 关系运算符

C 语言提供了 6 种关系运算符，如表 3-3 所示。

表 3-3 关系运算符

类 型	含 义	示 例	优先级	结合方向
＜	小于	5＜8	6	从左到右
＜＝	小于等于	a＜＝b	6	从左到右
＞	大于	a＞b＋1	6	从左到右
＞＝	大于等于	5＜＝8－2	6	从左到右
！＝	不等于	a！＝3	7	从左到右
＝＝	等于	a＝＝5	7	从左到右

3.3.2 关系表达式

关系表达式是由关系运算符和括号将运算对象连接起来的式子，其中运算对象可以是常量、变量、函数和数组元素等内容。关系表达式的一般组成形式为

表达式 1　关系运算符　表达式 2 …

关系表达式成立则结果为 1，关系不成立则结果为 0。例如：

```
5<8                    关系成立，表达式的值为 1
30>31                  关系不成立，表达式的值为 0
(3+7)!=(2+8)           关系不成立，表达式的值为 0
```

若有 int i=5，则(i＋＝3)＞6，i 的值为 8，8＞6，因此该表达式的值为 1。

由于关系运算符具有从左向右结合的特性，因此表达式 5＞2＞7＞8 在 C 语言中是允许的，表达式的值为 0。

【例 3.7】 关系表达式的应用。

若有 int a＝3，b＝2，c＝1，d，f；，则

```
a>b                    //表达式值为 1
(a>b)==c               //表达式值为 1
b+c<a                  //表达式值为 0
d=a>b                  //d=1
f=a>b>c                //f=0
```

注意：

(1) 表达式 1 和表达式 2 还可以是常量或变量的形式，也可以是赋值表达式、逻辑表达式和关系表达式等表达式嵌套的形式。

(2) 关系表达式的值为 0 或 1。

3.4　逻辑运算符和逻辑表达式

3.4.1　逻辑运算符

C 语言提供了 3 种逻辑运算符，如表 3-4 所示。

表 3-4　逻辑运算符

类　型	含　义	示　例	优先级	结合方向
!	逻辑非(取反)	! a	2	从右到左
&&	逻辑与(并且)	(5＞3)&&12%7	11	从左到右
||	逻辑或(或者)	y/4 || (x＋3)＝＝5	12	从左到右

其中逻辑与运算符 && 和逻辑或运算符||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。

```
!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)
```

逻辑运算的值为“真”和“假”两种，用 1 和 0 来表示。其求值规则如表 3-5 所示。

例如，由于 5 和 3 均为非 0 值，因此 5&&3 的值为“真”，即为 1。又如 5||0 的值为“真”，即为 1。

表 3-5 逻辑运算表

a	b	!a	!b	a && b	a||b
真	真	假	假	真	真
真	假	假	真	假	真
假	真	真	假	假	真
假	假	真	真	假	假

3.4.2 逻辑表达式

逻辑表达式是由逻辑运算符和括号将运算对象连接起来的式子，逻辑表达式的一般形式为

表达式 1 逻辑运算符 表达式 2

其中运算对象可以是常量、变量和函数的形式，也可以是关系表达式和算术表达式等表达式嵌套的形式，逻辑表达式的结果为 1 或 0。

【例 3.8】 输入年份值，判断并输出该年的天数。

编程分析：判断某年的天数，实际是判断该年是闰年还是平年，而闰年的判断条件是：该年能被 4 整除，但不能被 100 整除；或者该年能被 400 整除。由此看出该条件由两部分组成，且这两部分是逻辑或关系，用逻辑运算符||连接，只要有一个成立即可。能被 4 整除但不能被 100 整除是并列的关系，用逻辑与运算符 && 连接。

参考程序如下：

```
#include "stdio.h"
void main()
{
    int y;
    printf("input a year number:");                    //提示输入
    scanf("%d",&y);                                    //输入年份值
    if((y%4==0 && y%100!=0)|| y%400==0)
        printf("Year:%d Days=366\n",y);
    else
        printf("Year:%d Days=365\n",y);
}
```

程序运行结果如下：

```
input a year number:2013
Year:2013 Days=365
```

注意：

(1) 在逻辑运算值时，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”。如 3 && 0.5，结果为 1。

(2) 在逻辑与 && 和逻辑或|| 运算中，存在一种短路效应。如对于逻辑表达式 a&&b&&c，如果表达式 a 的逻辑值为“假”，由于逻辑表达式 a&&b 中逻辑与 && 运算符的运算规则，则整个表达式的结果为“假”，即 b 和 c 表达式均不再计算；但如果 a 的逻辑值

为“真”，此时整个表达式的逻辑值尚无法求出，因此还要求取 b 表达式的值，若此时 b 的表达式的值为“假”，则不再求取 c 的数值，否则在 a 和 b 表达式均为“真”的情况下还要计算 c 表达式的值。同理，对于逻辑表达式 a||b||c，在 a 为“真”的情况下不再求取 b 和 c 的值，否则一直进行到 b 或 c 表达式的值为“真”时为止。

【例 3.9】 计算逻辑表达式的值。

```
#include "stdio.h"
void main()
{
    int i=1,j=2,k=3;
    float x=200,y=0.85;
    printf("%d,%d\n",i==5&&'c'&&(j=8),x+y||i+j+k);
}
```

分析：对于表达式 i==5&&'c'&&(j=8)，由于 i==5 为假，逻辑值为 0，不再求取该表达式的第二部分和第三部分，可以判定该表达式的值为 0。对于表达式 x+y||i+j+k，由于x+y 的值为非 0，故该表达式的值为 1，也不用计算 i+j+k 表达式的值。程序运行结果如下：

```
0,1
```

3.5 条件运算符和条件表达式

3.5.1 条件运算符

C 语言提供了 1 种条件运算符，如表 3-6 所示。

表 3-6 条件运算符

类 型	含 义	示 例	优先级	结合方向
?:	条件运算	a>b?a:b	13	从右到左

条件运算符是三目运算符，具有右结合性。

3.5.2 条件表达式

条件表达式是由条件运算符和有关表达式、变量或常量等组成的式子。条件表达式的一般形式为

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，否则以表达式 3 的值作为条件表达式的值。如存在两个整型变量 a=5，b=10，求二者之间的较大值，可以用条件表达式表示为 a>b?a:b。

在双分支选择结构中，如果每个分支的执行语句部分均是对同一个变量进行赋值或输出操作，一般为如下的形式：

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

此时完全可以转换为条件表达式的形式，不但使程序简洁，也提高了运行效率。例如，求两个数之间较大值的双重分支结构为

```
if(a>b)
    max=a;
else
    max=b;
```

改写成用条件表达式语句表示的形式为

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

同理，对于 a、b、c 三个数求最大值的语句可以表示为

```
max=(max=a>b?a:b)>c? max:c;
```

注意：

(1) 条件运算符中“?”和“：”是一对运算符，不能分开单独使用。

(2) 条件运算符的结合方向是自右至左。因此 a＞b?a:c＞d?c:d 应理解为 a＞b?a:(c＞d?c:d)。

3.6 逗号运算符和逗号表达式

3.6.1 逗号运算符

C 语言提供了 1 种逗号运算符，如表 3-7 所示。

表 3-7 逗号运算符

类 型	含 义	示 例	优先级	结合方向
,	逗号	a＞b,c! =0,x	15	从左到右

3.6.2 逗号表达式

逗号表达式是由逗号运算符和有关变量、常量和表达式等组成的式子。逗号表达式的一般形式为

表达式 1,表达式 2, 表达式 3, …

求解的顺序是自左向右进行，先求解表达式 1 的值，然后求解表达式 2 的值，依此类推，整个逗号表达式的值是最后一个表达式的值。常用于循环 for 语句的第一个表达式中，作为循环变量赋初值或定义新变量并初始化。例如：

```
for(int i=0,sum=0; i<=100; i++)
```

【例 3.10】 计算如下逗号表达式的值。

```
a=3*5,a*4                    //a=15,表达式值 60
a=3*5,a*4,a+5                //a=15,表达式值 20
x=(a=3,6*3)                  //赋值表达式,表达式值 18,x=18
x=a=3,6*a                    //逗号表达式,表达式值 18,x=3
```

【例 3.11】 计算如下逗号表达式的值。

```
#include "stdio.h"
void main()
{
    int a=2,b=4,c=6,x,y;
    y=(x=a+b),(b+c);
    printf("y=%d,x=%d",y,x);
}
```

y=(x=a+b),(b+c)语句整体上是一个逗号表达式语句,由 y=(x=a+b)和(b+c)两个表达式语句组成,语句执行过程中,先执行第一个表达式的 x=a+b 部分,结果为 6 并将结果赋给 y,然后计算 b+c 表达式的值,结果 10,即整个表达式的结果为 10。但要打印输出的是 y 和 x 变量的数值,程序运行结果如下:

```
y=6,x=6
```

注意:

(1) 并不是所有出现逗号的地方都是逗号表达式,如在变量说明中,函数参数表中逗号只是用作各变量之间的间隔符。

(2) 逗号表达式中各个表达式也可以是逗号表达式的形式,即

```
(表达式 1,表达式 2),表达式 3
```

构成表达式嵌套的形式。

(3) 通常是要分别求逗号表达式内各表达式的值,并不一定要求整个逗号表达式的值。

习　　题

1. 单项选择题

(1) 若有说明语句 char c='\72';,则变量 c(　　)。

A. 包含 1 个字符　　B. 包含 2 个字符

C. 包含 3 个字符　　D. 说明不合法,c 值不确定

(2) 下列数据中属于"字符串常量"的是(　　)。

A. ABC　　B. "ABC"　　C. 'ABC'　　D. 'A'

(3) C 语言中,运算对象必须是整型的运算符是(　　)。

A. /　　B. %　　C. +　　D. -

(4) 若有以下定义: char a; int b; float c; double d;则表达式 a*b+d-c 值的类型为(　　)。

A. float　　B. int　　C. char　　D. double

(5) 执行语句 x=(a=3,b=a--)后,x、a、b 的值依次是(　　)。

A. 3,3,2　　B. 3,2,2　　C. 3,2,3　　D. 2,3,2

(6) 若有代数式 3ae/bc,则不正确的 C 语言表达式是(　　)。

A. a/b/c*e*3　　B. 3*a*e/b/c　　C. 3*a*e/b*c　　D. a*e/b/c*3

(7) 设整型变量 n 的值为 2,执行语句 n+=n--=n*n;后,n 的值是(　　)。

A. 0　　B. 4　　C. -4　　D. 2

(8) 已知 a=5,b=8,c=10,d=0;表达式的值为真的是(　　)。

A. a*2>8+2　　B. a&&d　　C. (a*2-c)||d　　D. a-b<c*d

(9) 以下程序运行后的输出结果是(　　)。

```
void main()
{
    int m=12,n=34;
    printf("%d%d",m++,++n); printf("%d%d\n",n++,++m);
}
```

A. 12353514　　B. 12353513　　C. 12343514　　D. 12343513

(10) 设整型变量 s、t、c1、c2、c3、c4 的值均为 2,则执行语句(s=c1==c2)||(t=c3>c4)后,s、t 的值为(　　)。

A. 1,2　　B. 1,1　　C. 0,1　　D. 1,0

(11) 设有定义 int a=2,b=3,c=4;,则以下选项中值为 0 的表达式是(　　)。

A. (! a==1)&&(! b==0)　　B. (a<b)&&!c||1

C. a&&b　　D. a||(b+b)&&(c-a)

(12) 为表示关系 X≥Y≥Z,应使用 C 语言表达式是(　　)。

A. (X≥Y)&&(Y≥Z)　　B. X>=Y>=Z

C. (X>=Y)‖(Y>=Z)　　D. (X>=Y)&&(Y>=Z)

(13) 表达式 (int)3.6*3 的值为(　　)。

A. 9　　B. 10　　C. 10.8　　D. 18

(14) 以下语句的输出结果是(　　)。

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

A. 1,0,4　　B. 1,0,3　　C. 0,0,3　　D. 0,0,4

(15) 已知 int x=3,y=4,z=5;,表达式!(x+y)+z-1&&y+z/2 的值是(　　)。

A. 6　　B. 0　　C. 2　　D. 1

2. 阅读程序写结果。

(1) 若 x 和 a 均是 int 型变量,计算表达式 x=(a=4,6*2) 后 x 的值为_____。

(2) 若有 int a=6,则计算 a+=a-=a*a 表达式后 a 的值为_____。

(3) 若有 int m=5, y=2;,则计算表达式 y+=y-=m*=y 后 y 的值是_____。

(4) 若有 int b=7;float a=2.5,c=4.7;,下面表达式的值为_____。

```
a+(int)(b/3*(int)(a+c)/2)%4
```

(5) 若 x 为 int 型,请以最简单的形式写出与逻辑表达式!x 等价的 C 语言关系表达式:_____。

(6) 以下程序运行后的输出结果是_____。

```
void main()
{
    int p=30;
    printf("%d\n",(p/3>0?p/10:p%3));
}
```

(7) 设 y 是 int 型变量,请写出判断 y 为奇数的关系表达式:_____。

第 4 章　C 语言程序的基本结构

【本章概述】

程序设计的最终产品是程序和文档。程序是利用计算机语言来解决某个问题而设计的一系列操作指令和数据的集合，语句是程序的基本组成单位，一个程序由若干个语句组成。C 语言是一种结构化程序设计语言，主要有顺序结构、选择结构和循环结构 3 种基本结构，更复杂的程序就是由这 3 种基本结构相互嵌套组成的。

【学习要求】

- 了解：C 语言语句的类型。
- 掌握：ANSI 和 N-S 流程图的绘制方法。
- 掌握：选择结构的基本形式。
- 掌握：选择结构、顺序结构的基本形式和执行过程。
- 掌握：循环控制语句的作用。
- 重点：3 种基本结构的形式及组成。
- 难点：循环嵌套及循环控制语句的作用。

4.1　结构化程序设计方法与算法

4.1.1　结构化程序设计方法

结构化程序设计(structured programming)是进行以模块功能和处理过程设计为主的详细设计的基本原则。其概念最早由迪克斯特拉(E. W. Dijkstra)在 1965 年提出的，是软件发展的一个重要的里程碑。其主要观点是采用自顶向下、逐步求精的程序设计方法；使用 3 种基本控制结构构造程序，任何程序都可由顺序、选择和循环 3 种基本控制结构来构造。结构化程序设计是以模块化设计为中心，将待开发的软件系统划分为若干个相互独立的模块，这样使完成每一个模块的工作变单纯而明确，为设计一些较大的软件打下了良好的基础。

C 语言就是一种结构化的程序设计语言。它层次清晰，便于按模块化方式组织程序，易于调试和维护。

程序设计的基本目标是用算法对问题的原始数据进行处理，从而获得所期望的效果。但这仅仅是程序设计的基本要求。要全面提高程序的质量，提高编程效率，使程序具有良好的可读性、可靠性、可维护性以及良好的结构。编制出好的程序，应当是每位程序设计工作者追求的目标，而要做到这一点，就必须掌握正确的程序设计方法和技巧。

在结构化程序设计思想提出之前，程序设计强调程序的效率。结构化程序的概念首先是针对以往编程过程中无限制地使用转移语句而提出的，转移语句可以使程序的控制流程

强制性地转向程序的任一处，如果一个程序中多处出现这种转移情况，将会导致程序流程无序可循，程序结构杂乱无章，这样的程序是令人难以理解和接受的，并且容易出错。结构化程序设计方法是程序设计的先进方法和工具，其主要原则可以概括为：自顶向下，逐步求精，模块化，限制使用 goto 语句。与程序的效率相比，在结构化程序设计中，人们更重视程序的可理解性，即程序的易读性。

算法的实现过程是由一系列操作组成的，这些操作之间的执行次序就是程序的控制结构。1996 年，计算机科学家 Bohm 和 Jacopini 证明了这样的事实：任何简单或复杂的算法都可以由顺序结构、选择结构和循环结构这 3 种基本结构组合而成。这 3 种结构就被称为程序设计的 3 种基本结构，也是结构化程序设计必须采用的结构。这三种基本结构的含义如下：

(1) 顺序结构：是一种简单的程序设计，它是最基本、最常用的结构。顺序结构是指语句的执行顺序和它在程序中出现的次序是一致的，即一条语句执行完后紧跟着执行它下面的那条语句。

(2) 选择结构：又称为分支结构，包括简单选择结构和多分支选择结构。

(3) 重复结构：又称循环结构，有当型循环结构(先判断后执行循环体)和直到型循环结构(先执行循环体后判断)两种形式。

结构化程序与非结构化程序之间最重要的区别就是：结构化程序是用模块化的方法设计的，每个模块只能有一个入口和一个出口，复杂结构应该用嵌套的基本控制结构进行组合嵌套来实现。模块化设计带来了生产力的巨大提升：首先，小模块可以很快、很容易地编写；其次，通用模块可以被重用，使后续的程序可以更快地开发；最后，程序的模块可以独立进行测试，有助于减少调试的时间。

按结构化程序设计方法设计出的程序具有两大明显的优点：程序易于理解、使用和维护；提高了编程工作效率，降低了软件开发成本。

程序的质量首先取决于它的结构，其次取决于程序员的编程风格。具有良好的设计风格应该是程序员所具备的基本素质，良好的程序设计风格不仅有助于提高程序的可靠性、可理解性、可测试性、可维护性和可重用性，而且也能够促进技术的交流，改善软件的质量。程序编写过程中，从表达清晰、便于阅读、理解和维护的角度出发，应遵循以下规则：

(1) 一个说明或一个语句占一行。

(2) 标识符，包括模块名、变量名、常量名、标号名和子程序名等，这些名字应能反映它所代表的实际东西，有一定实际意义，使读者能顾名思义。

(3) 必要的程序注释是程序设计者与程序阅读者之间沟通的重要手段，可以是功能性注释(用以描述其后的语句或程序段用于完成什么样的工作)，也可以是序言性注释(位于每个程序模块的开头部分，给出程序的整体说明)。

(4) 用{ }括起来的部分通常表示了程序的某一层次结构，可以是程序的复合语句，也可以是数组初始化、结构体、共用体或枚举的定义等内容。对于复合语句而言，{ }一般与该结构语句的第一个字母对齐，并单独占一行。

(5) 低一层次的语句或说明可比高一层次的语句或说明向右缩进若干字符后书写，即采用分层缩进的写法显示嵌套结构层次，整个程序呈现出锯齿形的缩进结构，这样可使程序的逻辑结构更加清晰，层次更加分明。

4.1.2 算法

人们使用计算机，就是要利用计算机处理各种不同的问题，一般要经过需求分析、设计算法、编写程序、上机调试与维护5个过程。解决问题中的这些具体的方法和步骤，其实就是解决一个问题的算法。

著名计算机科学家沃思(Niklaus Wirth)曾提出一个公式：

程序＝数据结构＋算法

其中，数据结构指对数据的描述，即程序中数据的类型和组织形式；算法(algorithm)是对拟解决问题的具体操作步骤的描述。解决一个问题可能有多种算法，例如，数学题常常有“一题多解”，也就是说，解决一个问题的算法可能不止一种。这时，应该通过分析、比较，挑选一种最优的算法。编程时除了采用结构化编程思想之外，还要考虑解决问题的步骤和方式是否简洁和正确，否则有可能造成简单问题复杂化，导致程序的效率较低。因此，程序设计人员必须认真考虑和设计数据结构与操作步骤。

程序设计是指编写程序的全过程，而算法是解决一个问题所采取的方法和步骤。实际上，要编制一个完整的程序，除了数据结构和算法之外，还应该采用结构化程序设计思想，并且选用一种计算机语言来实现，因此，可以在沃思提出的公式的基础上扩展成以下公式：

程序设计＝算法＋数据结构＋计算机语言＋程序设计方法

在这4个组成部分中，算法是一个程序的灵魂，数据结构是加工对象，计算机语言是工具，程序设计方法是良好程序的基础。对于初学者来说，重要的是理解程序设计的基本思想和方法，学习高级语言的重点，掌握分析问题、解决问题的方法，锻炼分析、分解直至最终归纳整理出算法的能力。与之相对应，C语言的语法是工具，是算法的一个具体实现。所以在高级语言的学习中，一方面应熟练掌握该语言的语法，因为它是算法实现的基础，另一方面必须认识到算法的重要性，加强思维训练，以写出高质量的程序。

1. 算法的描述

算法，简单地讲，就是解决问题的流程安排，即先做什么，后做什么。精确地讲，算法是被精确定义的一系列规则，这些规则规定了解决特定问题的一系列操作顺序，以便在有限步骤内产生出所有问题的解答。人的思想要用语言来表达，而算法是人求解问题的思想方法，是对解题过程的精确描述，同样也需要用语言来表示。用于描述算法的工具很多，如自然语言法、传统流程图法、N-S流程图法和伪代码法等，理论上都可以用来表示算法，但是效率有很大差异。

1) 自然语言描述法

自然语言是人们日常使用的语言，可以是汉语、英语等多种形式。自然语言描述法是非常直观的一种表示方法，它所表示的含义往往不太严格，需要根据上下文进行理解，用该方法所描述的算法简单易懂，但存在结构冗长、容易出现歧义的缺点。例如，“他说他昨天出去玩了。”这里的两个“他”就构成了歧义——是指说话者本人还是其他人？自然语言描述法一般只用于简单问题的算法描述。

【例4.1】 计算1＋2＋3＋…＋100。

算法一：

S1：计算1＋2的结果，结果为3；

S2：利用 S1 的结果与 3 相加，结果为 6；

⋮

S98：利用 S97 的结果与 99 相加，结果为 4950；

S99：利用 S98 的结果与 100 相加，结果为 5050；

S100：输出计算结果。

该算法共由 100 个步骤完成，虽然易于理解，但内容比较烦琐，阅读和理解算法要花费很长时间，而要实现的功能却十分简单。在此，可以对该算法进行优化，将求和的过程改写成循环的方式，即添加用于计数的循环变量 i 和用于保存和的变量 s，在每次累加之前先判断循环变量 i 的数值是否超出界限。

算法二：

S1：定义循环变量 i=1，用于保存和的变量 s=0；

S2：判断 i 的数值是否小于等于 100，若是则执行 S3，否则跳转到 S4 执行；

S3：将 i 的数值累加到 s，然后变量 i 自身加 1，转到 S2 执行；

S4：输出 s 的数值。

2）传统流程图描述法

自然语言描述的算法虽然简单易懂，但只适用于简单问题算法的描述，对于复杂问题的描述，采用自然语言描述法则文字冗长，特别是对于有较多分支结构、选择结构和嵌套的问题，用自然语言来描述更是错综复杂。因此，美国国家标准化协会（American National Standard Institute，ANSI）规定了一些常用的流程图符号，用于表示程序的执行步骤与控制流向，主要有图 4-1 所示的几种符号，将这些符号组合起来，就可以表示比较复杂的算法。

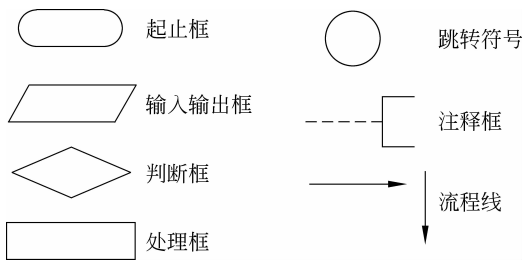

图 4-1　ANSI 流程图基本符号

说明：

- 起止框用于流程图的开始和结束位置，标志算法的开始和结束，每个算法都有一个开始和结束标志。
- 输入输出框用于标志算法中需要输入的数据和输出算法的结果。
- 判断框用于对给定的条件进行判断，根据对条件计算的结果是否成立来决定如何执行后续的操作，有一个入口，可以有一个或两个出口。
- 处理框用于表示算法中的各种具体操作。
- 跳转符号用于绘制较大流程图时控制流程的转向，由于空间或功能的限制，较大的流程图有时需要转向其他位置的入口点。
- 注释框用于对流程图中的内容进行注释，做到直观、易于理解。
- 流程线用于表明流程的走向，表示具体操作的执行顺序。

人们经过长期的实践，在应用和总结的基础上，提出了结构化程序设计的思想，将算法的描述归纳为顺序结构、选择结构和循环结构3种基本结构的顺序组合，共同点是只有一个入口和一个出口。使用3种基本结构描述的算法是结构化的算法，按照结构化算法编写出来的程序具有良好的可读性和可维护性。

3种基本结构的流程图如图4-2所示。

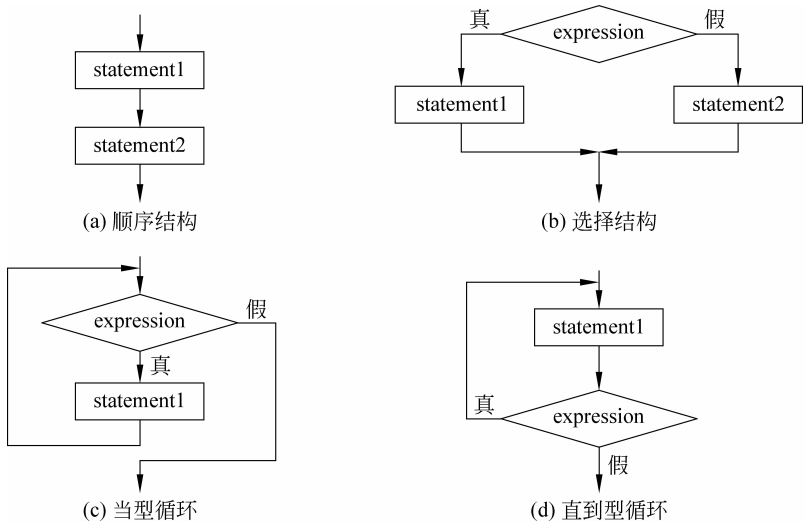

图4-2 3种基本结构的流程图

【例4.2】 用传统流程图表示1＋2＋…＋100的算法。

编程分析： 根据题意，是要求100个自然数的和，在此采用循环的方式，定义循环变量i和用于保存求和结果的变量s，每次执行循环语句(求和、变量加1)前先判断i的数值是否小于等于100，大于100则结束循环并输出求和结果。流程图如图4-3所示。

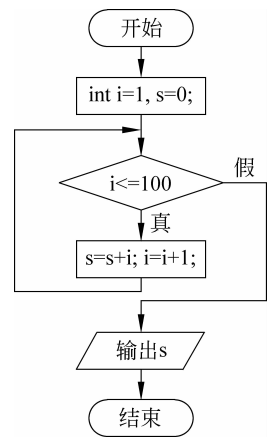

图4-3 例4.2流程图

参考程序如下：

```
#include "stdio.h"
void main()
{
    int i=1,s=0;
    while(i<=100)
    {
        s=s+i;
        i++;
    }
    printf("%d\n",s);
}
```

【例4.3】 判断输入年份的天数，用ANSI流程图表示其算法。

编程分析： 闰年的判断条件是：该年份是4的整倍数但不是100的整倍数，或者是400的整倍数。整倍数的判断是用两个整数相除并判断余数是否为0来实现的，判断年份n为

闰年的 C 语言表达式为(n%4==0 && n%100!=0)||n%400==0。流程图如图 4-4 所示。

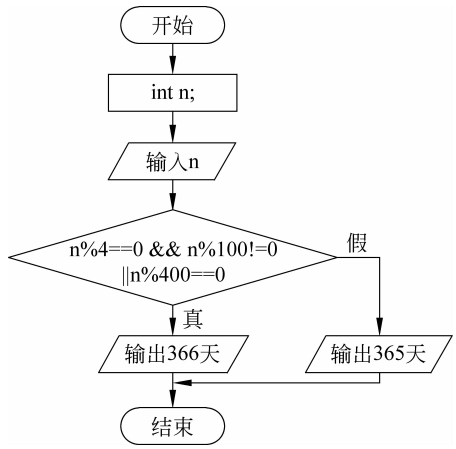

图 4-4　例 4.3 流程图

参考程序如下：

```
#include "stdio.h"
void main()
{
    int n;
    scanf("%d",&n);                    /* 此处必须带 & 符号,表示输入到 n 的内存地址中 */
    if(n%4==0 && n%100!=0 ||n%400==0)
        printf("366 days\n");          /* \n 表示输出后光标到下一行 */
    else
        printf("365 days\n");
}
```

3）N-S 流程图描述法

在结构化程序设计中，可以由 3 种基本结构顺序组合成各种复杂的算法结构，因此，ANSI 结构流程图中的流程线就成了多余，于是 1973 年美国学者 I. Nassi 和 B. Shneiderman 提出了一种新的程序控制流程图的表示方法，即 N-S 结构图，它使用矩形框来表示 3 种基本结构，由 3 种基本结构的矩形框嵌套，可以组成各种复杂的程序算法。N-S 结构的 3 种基本流程图如图 4-5 所示。

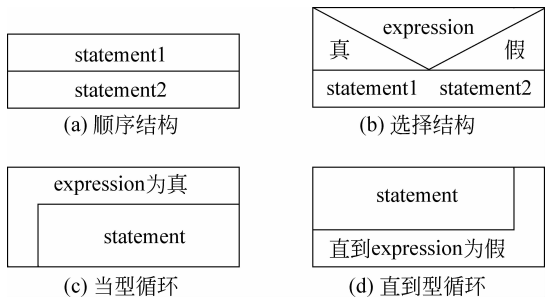

图 4-5　N-S 结构的流程图

【例 4.4】 输出两个整数中的较大值，用 N-S 结构的流程图表示算法。

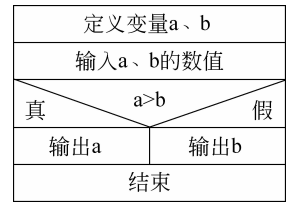

图 4-6 例 4.4 流程图

编程分析：对于两个数求较大值，就是根据其数值大小用比较的方式得出，如果前者大于后者，则较大值为前者，否则，较大值为后者。在 C 语言中可以使用关系运算符(＞、＜、＞＝、＜＝等)来表示数据之间的大小关系。流程图如图 4-6 所示。

参考程序如下：

```
#include "stdio.h"
void main()
{
    int a,b;
    scanf("%d,%d",&a,&b);
    if(a>b)
        printf("bigger=%d\n",a);
    else
        printf("bigger=%d\n",b);
}
```

程序执行时，请输入两个数字，二者之间用逗号分隔，如 23,56，程序运行结果如下：

```
23,65
bigger=65
```

2. 算法的特点

尽管算法因求解问题的不同而千变万化、简繁各异，但它们都必须具备以下 5 个特征，在表示一个算法时需要从这些特征出发，以使得设计的算法切实可行。

(1) 有穷性：一个算法必须保证执行有限步之后结束。如果一个算法不具备有穷性，但具有算法的其他特性，则称为计算方法。比如对无极调和级数的计算就不是一个算法，而是一个计算方法。但是在确定了项数或者精度之后，再进行的计算就是有穷的了，这时对调和级数的计算(如求和)就变成了算法。

(2) 确切性：算法的每一步骤必须有确切的定义，即每种运算所执行的操作都必须是确定的、无二义性的。比如在配制某种溶液时候，应明确指出“5 升水加 100 克药粉”，而不能是“一桶水加适量药粉”这种不确定的说法。

(3) 输入：一个算法有 0 个或多个输入，以刻划运算对象的初始情况，所谓 0 个输入是指算法本身给定了初始条件。

(4) 输出：一个算法有一个或多个输出，以反映对输入数据加工后的结果。没有输出的算法是毫无意义的。

(5) 可行性：算法原则上能够精确地运行，且人们用笔和纸做有限次运算后即可完成。

4.2 C 程序语句

C 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。在 C 程序中，共有 5 种类型的语句，分别是表达式语句、函数调用语句、控制语句、复合语句和空语句。