



中國人民大學

RENMIN UNIVERSITY OF CHINA

信息学院

SCHOOL OF INFORMATION

程序设计1荣誉课程

3. 程序控制语句与枚举法

授课教师：游伟 副教授

授课时间：周一08:00 – 09:30, 周四16:00 – 17:30 (明德新闻楼0201)

上机时间：周四18:00 – 21:00 (理工配楼二层5号机房)

课程主页：<https://rucsesec.github.io/programming>

引子：博饼游戏

【背景】博饼是闽南地区盛行的中秋传统民俗活动。游戏时，需要6个骰子和一个大瓷碗。一般会进行多轮投掷，根据每轮投掷的骰子点数特征累计积分。如果某一轮投掷，出现规定特征以外的点数，那么游戏结束，否则继续进行。多轮投掷结束后，根据累计积分判定奖次。具体的点数特征和积分见下页。

【任务】编写程序，进行博饼游戏的积分统计。

【输入格式】第一行是一个正整数 n ($1 \leq n < 256$)，代表最多进行 n 轮投掷；
往后至多 n 行，每行6个正整数，代表每个骰子的点数

【输出格式】一个正整数，代表最终的累计积分（十六进制格式输出）

【输入样例】 6

【输出样例】： 20

4 1 4 4 5 4

1 1 1 1 1 1

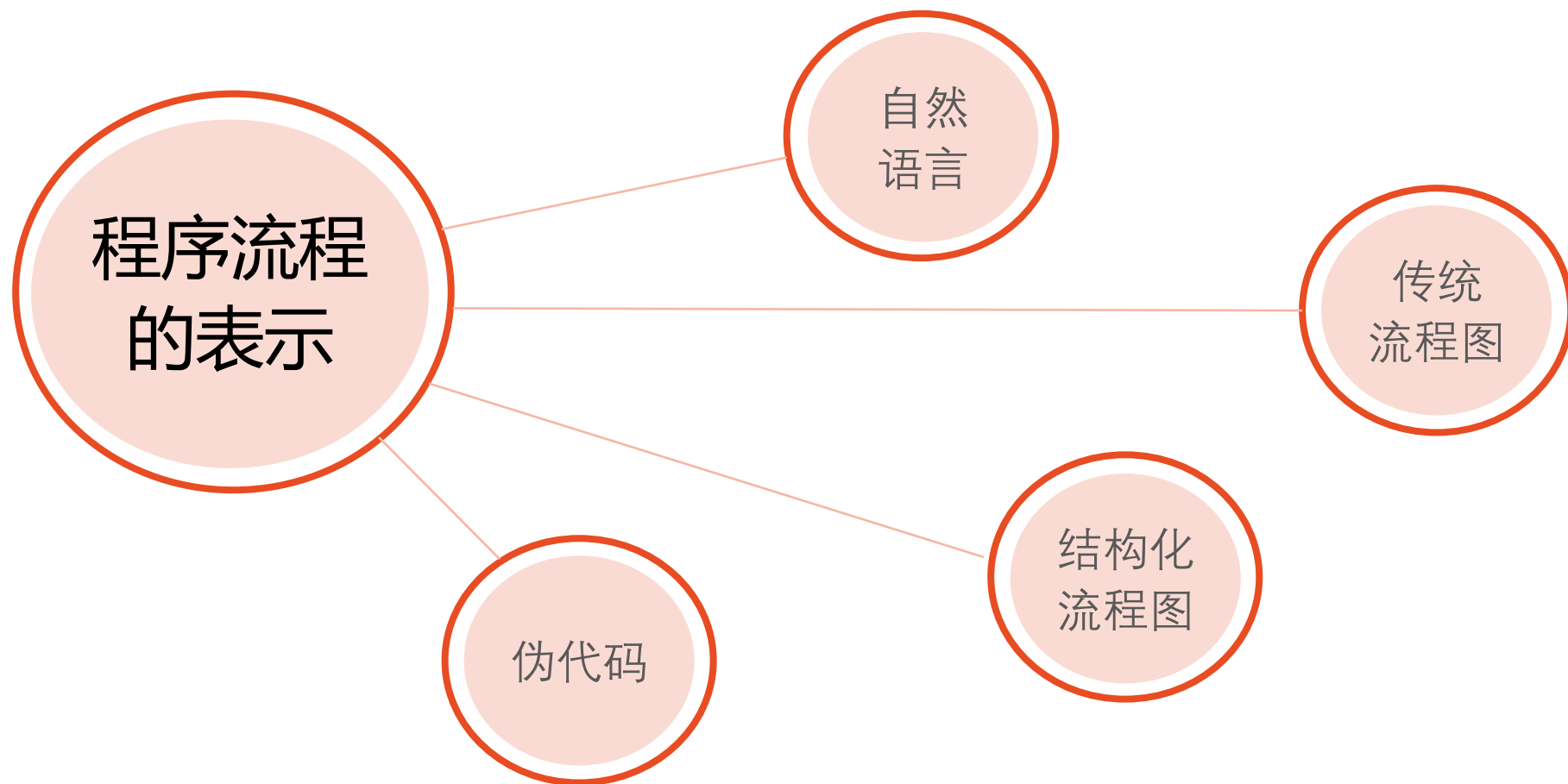
引子：博饼游戏

官级	别名	特征	积分
状元	插金花		2048
状元	红六勃		1024
状元	遍地锦		512
状元	黑六勃		256
状元	五红		128
状元	五子		64
状元	四红		32
榜眼	对堂		16
探花	三红		8
进士	四进		4
举人	二举		2
秀才	一秀		1
平民	出局	其它	-1

目录

1. 程序流程的表示
2. 关系运算/逻辑运算/条件运算
3. 分支语句
4. 循环语句
5. 枚举法解题

3.1 程序流程的表示



3.1.1 几个示例

■ 示例1：求 $1 \times 2 \times 3 \times 4 \times 5$

算法步骤

S1: 先求1乘以2，得到结果2

S2: 将步骤1得到的乘积2再乘以3，得到结果6

S3: 将6再乘以4，得24

S4: 将24再乘以5，得120



若题目改为: 求 $1 \times 3 \times 5 \times 7 \times 9 \times 11$

算法步骤

S1: 令 $p=1$ ，或写成 $1 \Rightarrow p$ (表示将1存放在变量 p 中)

S2: 令 $i=3$ 或写成 $3 \Rightarrow i$ (表示将2存放在变量 i 中)

S3: 使 p 与 i 相乘，乘积仍放在变量 p 中，可表示为: $p*i \Rightarrow p$

S4: 使 i 的值加 2 即 $i+2 \Rightarrow i$

S5: 若 $i \leq 11$ ，返回S3；否则，结束
或者 若 $i > 11$ ，结束；否则，返回S3

用这种方法表示的算法具有一般性、通用性和灵活性

3.1.1 几个示例

■ 示例2：求5!

P: 表示被乘数

i: 表示乘数

算法步骤

S1: $1 \Rightarrow p$

S2: $2 \Rightarrow i$

S3: $p * i \Rightarrow p$

S4: $i + 1 \Rightarrow i$

S5: 如果 $i \leq 5$, 则返回S3; 否则结束

3.1.1 几个示例

■ 示例3：求 $1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \dots + \frac{1}{99} - \frac{1}{100}$

sign：表示当前项的数值符号

term：表示当前项的值

sum：表示当前项的累加和

deno：表示当前项的分母

算法步骤

S1: sign=1

S2: sum=1

S3: deno=2

S4: sign=(-1) * sign

S5: term=sign * (1/deno)

S6: sum=sum+term

S7: deno=deno+1

S8: 若deno ≤ 100返回S4；否则算法结束

3.1.1 几个示例

■ 示例4：给出一个大于或等于3的正整数，判断它是不是一个素数

解题思路：素数(prime)是指除了1和该数本身之外，不能被其他任何整数整除的数。

算法步骤

S1: 输入n的值

S2: $i=2$ (i作为除数)

S3: n被i除，得余数r

S4: 如果 $r=0$ ，表示n能被i整除，则输出n“不是素数”，算法结束；否则执行S5

S5: $i+1 \Rightarrow i$

S6: 如果 $i \leq \sqrt{n}$ ，返回S3；否则输出n的值以及“是素数”，然后结束

实际上，n不必被 $2 \sim (n-1)$ 之间的整数除，只须被 $2 \sim n/2$ 间整数除即可，甚至只须被 $2 \sim \sqrt{n}$ 之间的整数除即可。

3.1.1 几个示例

- 示例5：有50个学生，输出成绩在80分以上的学生的学号和成绩

n : 表示学生学号

下标 i : 表示第几个学生

n_1 : 表示第一个学生的学号

n_i : 表示第 i 个学生的学号

g : 表示学生的成绩

g_1 : 表示第一个学生的成绩

g_i : 表示第 i 个学生的成绩

算法步骤

S1: $1 \Rightarrow i$

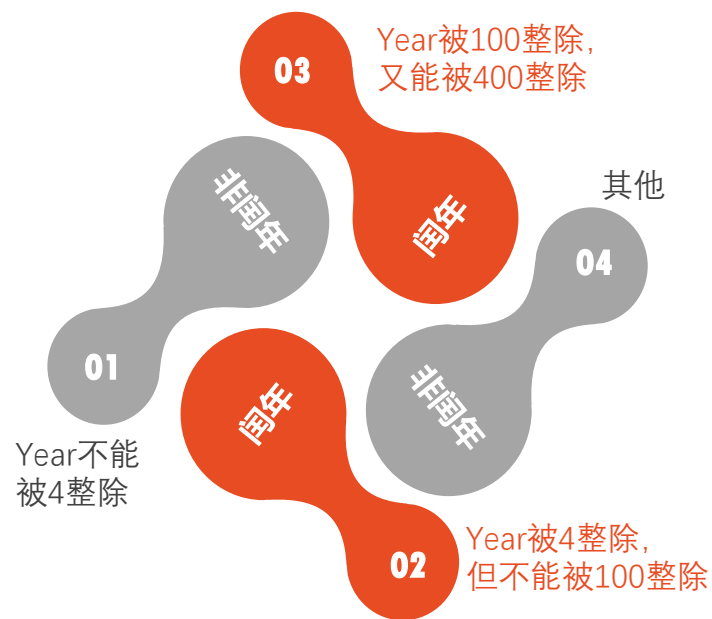
S2: 如果 $g_i \geq 80$, 则输出 n_i 和 g_i , 否则不输出

S3: $i+1 \Rightarrow i$

S4: 如果 $i \leq 50$, 返回到S2, 继续执行, 否则, 算法结束

3.1.1 几个示例

- 示例6：判定2000—2500年中的每一年是否为闰年，并输出结果



算法步骤

S1: 2000=>year

S2: 若year不能被4整除, 则输出year 的值和“不是闰年”。然后转到S6, 检查下一个年份

S3: 若year能被4整除, 不能被100整除, 则输出year的值和“是闰年”。然后转到S6

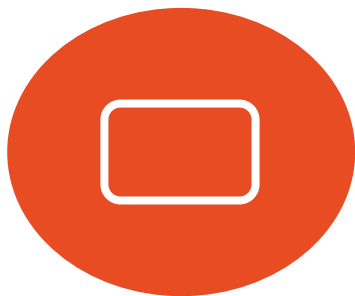
S4: 若year能被400整除, 输出year的值和“是闰年”, 然后转到S6

S5: 输出year的值和“不是闰年”

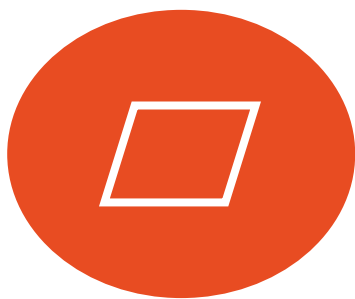
S6: year+1=>year

S7: 当year≤2500时, 转S2继续执行, 否则算法停止

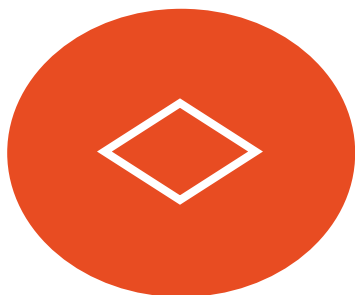
3.1.2 传统流程图



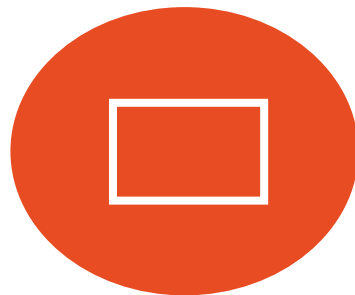
起止框



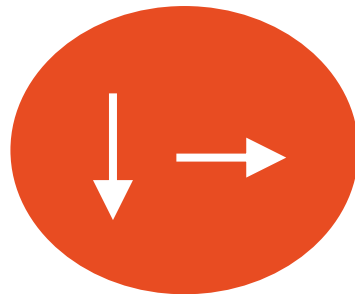
输入输出框



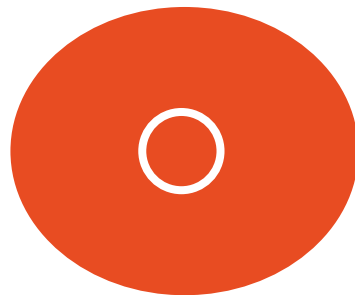
判断框



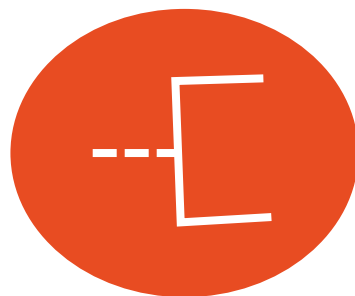
处理框



流程线



连接点



注释框

3.1.2 传统流程图

■ 示例1：求 $1 \times 2 \times 3 \times 4 \times 5$ （用流程图表示）

算法步骤

S1: $1 \Rightarrow p$

S2: $2 \Rightarrow i$

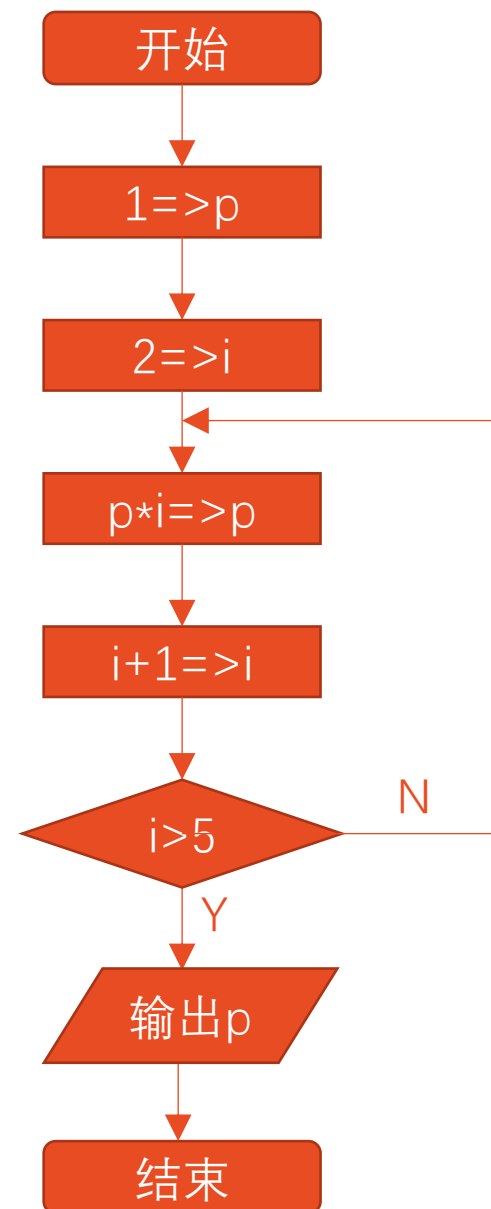
S3: $p * i \Rightarrow p$

S4: $i + 1 \Rightarrow i$

S5: 如果 $i \leq 5$ ，则返回S3；否则结束

P: 表示被乘数

i: 表示乘数



3.1.2 传统流程图

■ 示例3：求 $1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \dots + \frac{1}{99} - \frac{1}{100}$
(用流程图表示)

sign：表示当前项的数值符号

term：表示当前项的值

sum：表示当前项的累加和

deno：表示当前项的分母

算法步骤

S1: sign=1

S2: sum=1

S3: deno=2

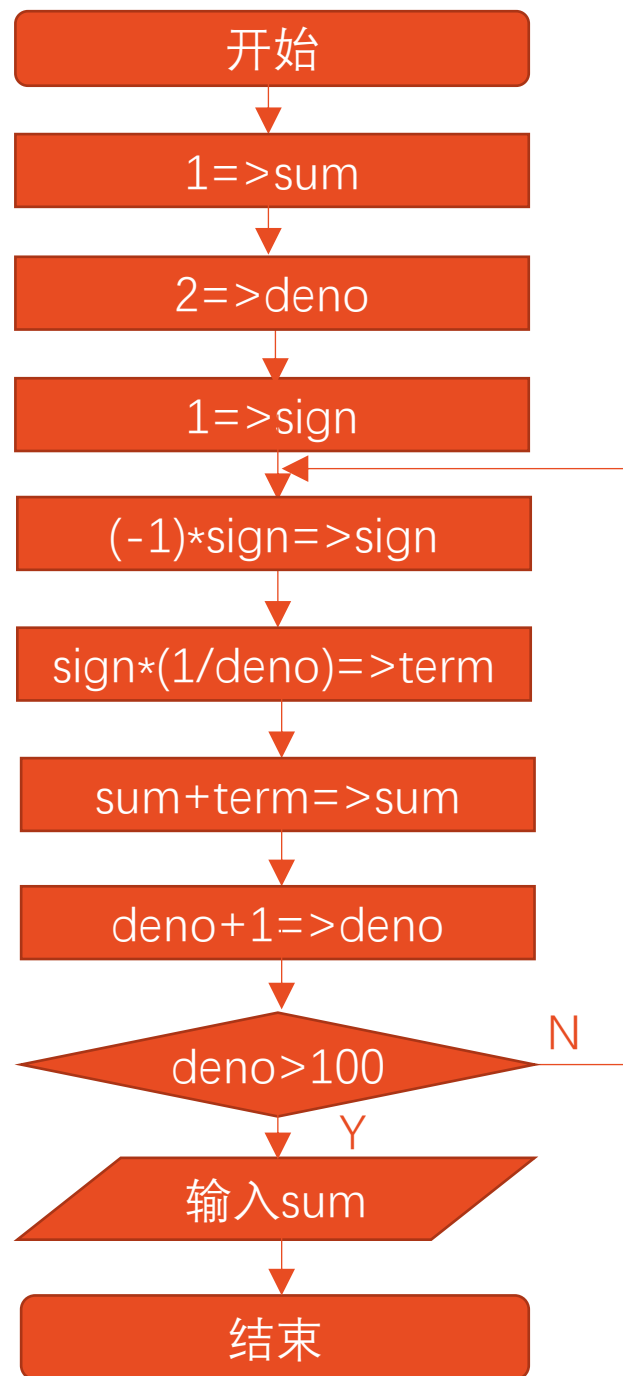
S4: sign=(-1)*sign

S5: term=sign*(1/deno)

S6: sum=sum+term

S7: deno=deno+1

S8: 若deno ≤ 100返回S4；否则算法结束



3.1.2 传统流程图

■ 示例4：给出一个大于或等于3的正整数，判断它是不是一个素数（用流程图表示）

算法步骤

S1: 输入n的值

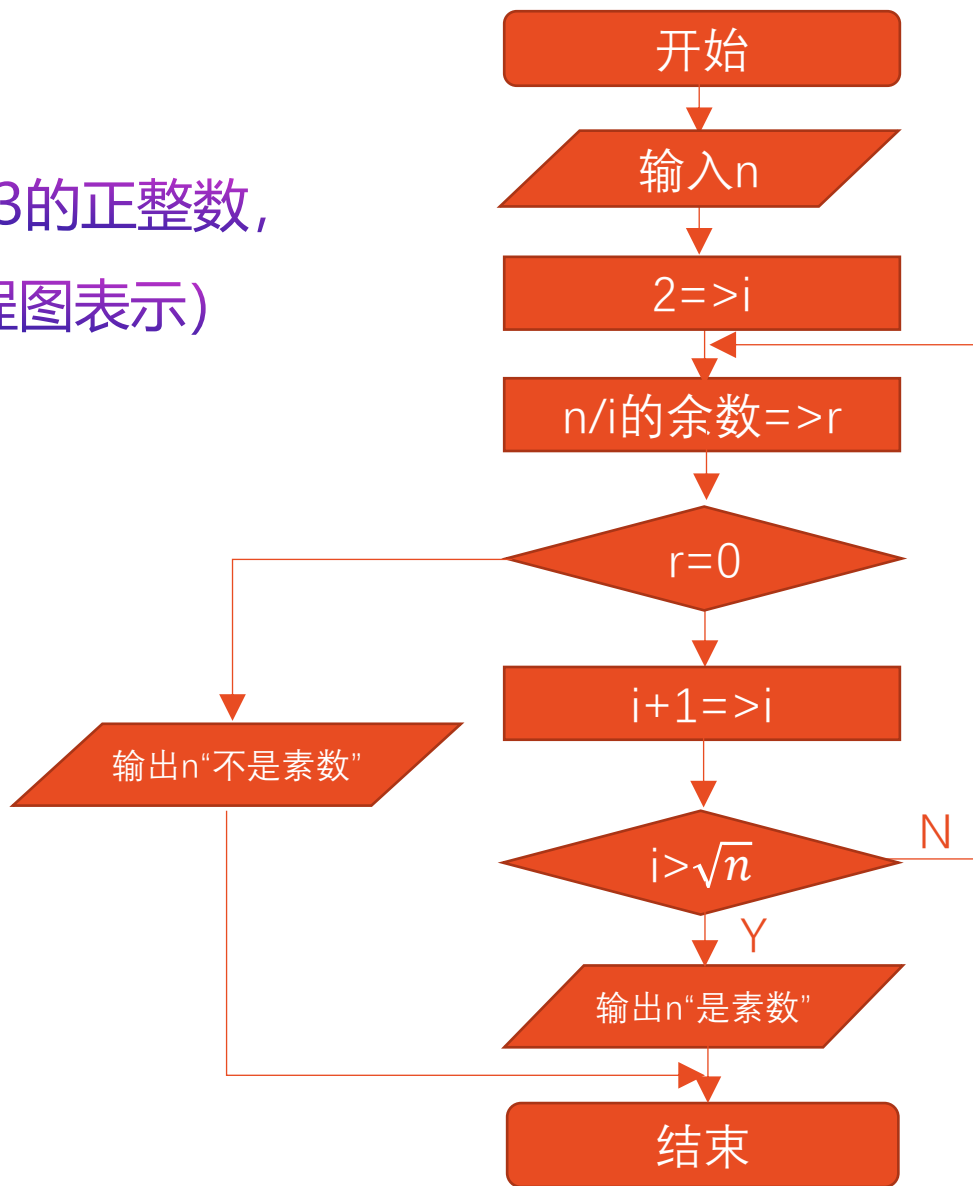
S2: $i=2$ （i作为除数）

S3: n被i除，得余数r

S4: 如果 $r=0$ ，表示n能被i整除，则输出n“不是素数”，算法结束；否则执行S5

S5: $i+1 \Rightarrow i$

S6: 如果 $i \leq \sqrt{n}$ ，返回S3；否则输出n的值以及“是素数”，然后结束



3.1.2 传统流程图

■ 示例5：有50个学生，输出成绩在80分以上的学生的学号和成绩（用流程图表示）

下标 i ：表示第几个学生

n ：表示学生学号

n_1 ：表示第一个学生的学号， n_i ：表示第 i 个学生的学号

g ：表示学生的成绩

g_1 ：表示第一个学生的成绩， g_i ：表示第 i 个学生的成绩

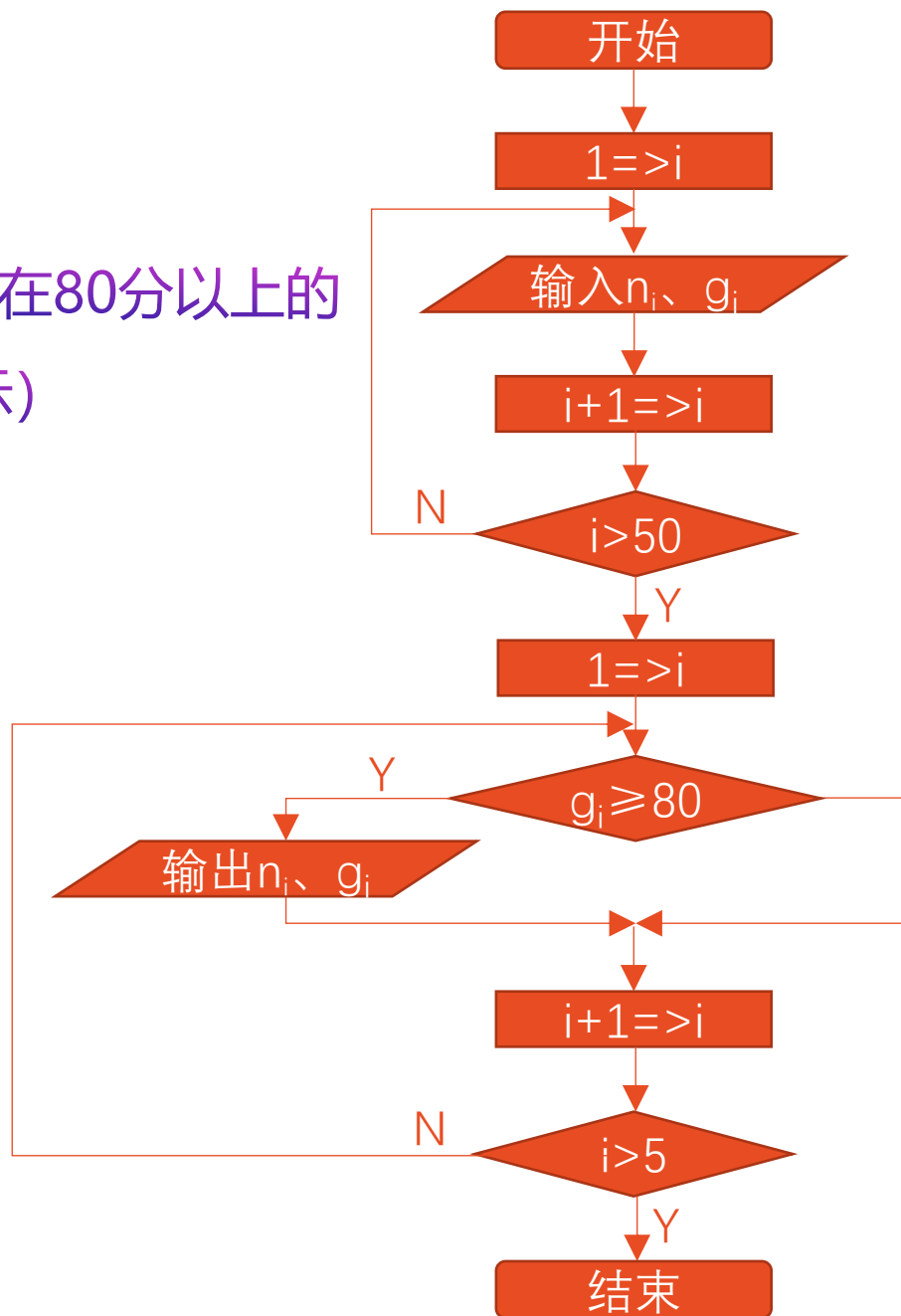
算法步骤

S1: $1 \Rightarrow i$

S2: 如果 $g_i \geq 80$ ，则输出 n_i 和 g_i ，否则不输出

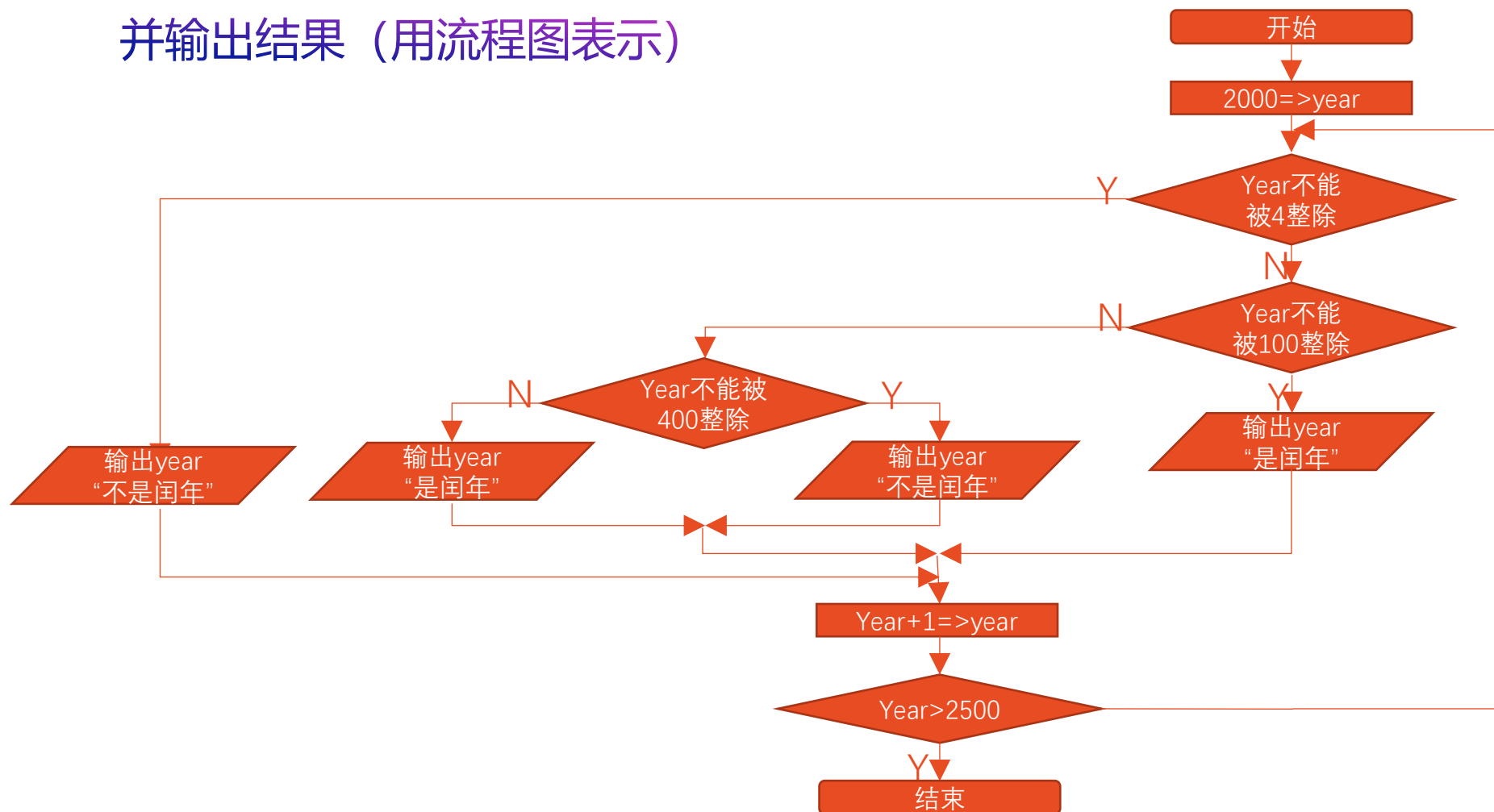
S3: $i+1 \Rightarrow i$

S4: 如果 $i \leq 50$ ，返回到S2，继续执行，否则，算法结束



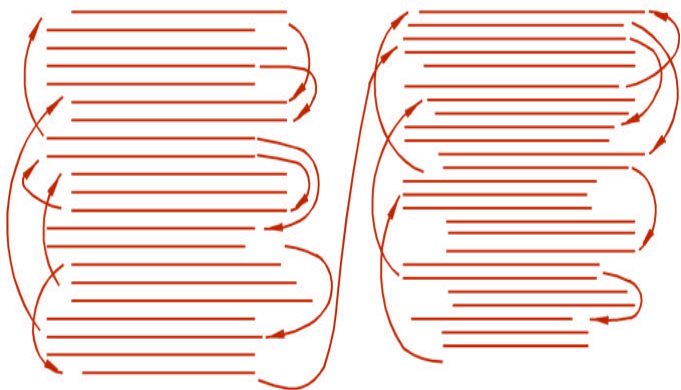
3.1.2 传统流程图

- 示例6：判定2000—2500年中的每一年是否为闰年，并输出结果（用流程图表示）



3.1.3 结构化流程图

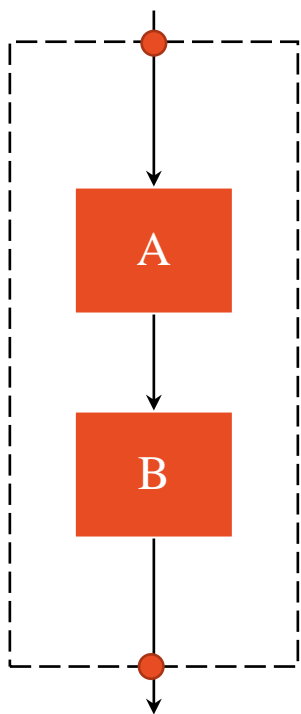
■ 传统流程图的弊端



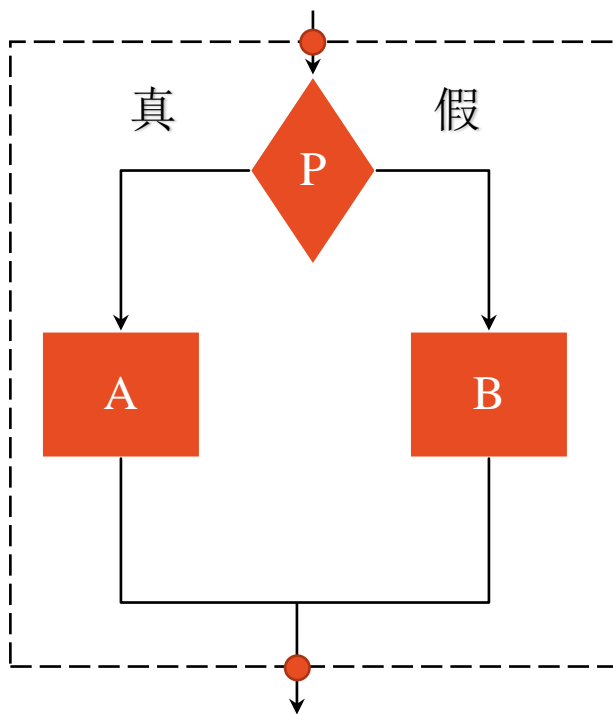
传统流程图用流程线指出各框的执行顺序，对流程线的使用没有严格限制。因此，使用者可不受限制地使流程随意地转来转去，使流程图变得毫无规律，阅读时要花很大精力去追踪流程，使人难以理解算法的逻辑。

3.1.3 结构化流程图

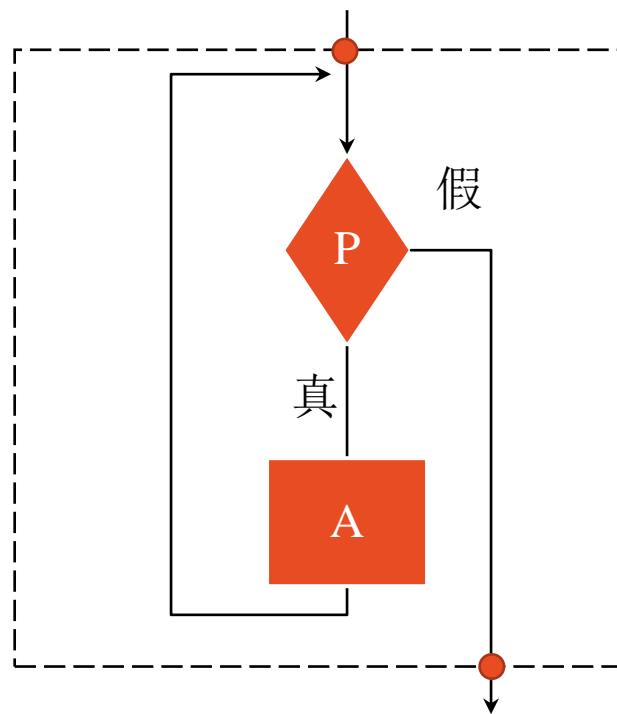
■ 三种基本结构



顺序结构







选择结构



循环结构

3.1.3 结构化流程图

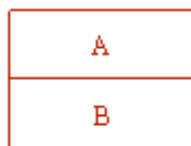
■ 三种基本结构的特点

-  1 只有一个入口
-  2 只有一个出口
-  3 结构内的每一部分都有机会被执行到
-  4 结构内不存在“死循环”

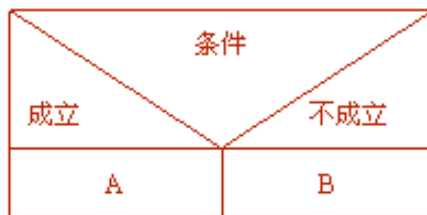
3.1.3 结构化流程图

■ N-S流程图表示法

- 完全去除了带箭头的流程线
- 全部流程写在一个矩形框内



(a) 顺序结构



(b) 分支结构



(c) 循环结构 (条件在前)



(d) 循环结构 (条件在后)

3.1.4 伪代码

- 伪代码：用介于自然语言和计算机语言之间的文字和符号，来描述程序流程
- 它如同一篇文章一样，自上而下地写下来。每一行(或几行)表示一个基本操作。
- 它不用图形符号，因此书写方便，格式紧凑，修改方便，容易看懂，也便于向计算机语言算法(即程序)过渡

3.1.4 伪代码

■ 示例2：求5!（用伪代码表示）

P: 表示被乘数

i: 表示乘数

```
#include <stdio.h>
int main()
{
    int i,p;
    p=1;
    i=2;
    while(i<=5) {
        p=p*i;
        i=i+1;
    }
    printf("%d\n",p);
    return 0;
}
```

C 代 码

begin (算法开始)

1=>p

2=>i

while i≤5

{ p*i=>p

i+1=>i

}

print p

end (算法结束)

伪 代 码

3.1.4 伪代码

- 示例3：求 $1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \dots + \frac{1}{99} - \frac{1}{100}$
(用流程图表示)

```
#include <stdio.h>
int main()
{
    int sign=1;
    double deno=2.0,sum=1.0,term;
    while(deno<=100)
    {
        sign=-sign;
        term=sign/deno;
        sum=sum+term;
        deno=deno+1;
    }
    printf("%f\n",sum);
    return 0;
}
```

C 代 码

```
begin      (算法开始)

1=>sign
1=>sum
2=>deno
while deno≤100
{  (-1)*sign=>sign
   sign*(1/deno)=>term
   sum+term=>sum
   deno+1=>deno
}
print sum
end        (算法结束)
```

伪 代 码

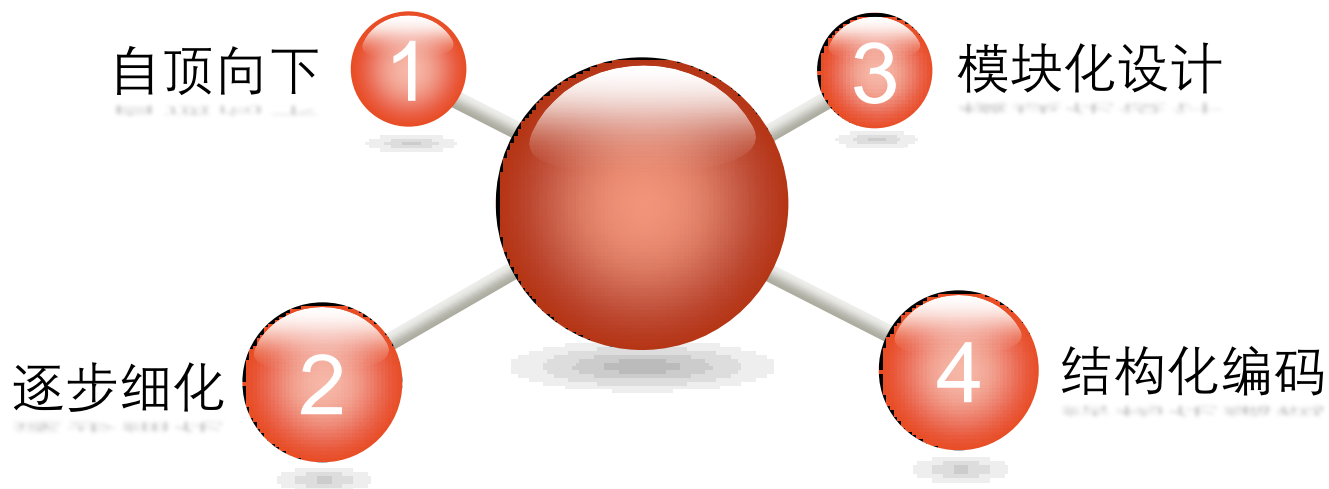
sign: 表示当前项的数值符号

term: 表示当前项的值

sum: 表示当前项的累加和

deno: 表示当前项的分母

结构化程序设计方法



思考

用流程图和伪代码的方法，
表示博饼游戏的积分累流程

C语言运算符概览

算术运算符:	+ - * / % ++ --
赋值运算符:	= 及其扩展
求字节数 :	sizeof
强制类型转换:	(类型)
函数调用符运算符:	()
关系运算符:	< <= == > >= !=
逻辑运算符:	! &&
条件运算符:	?:
下标运算符:	[]
位运算符 :	<< >> ~ & ^
指针运算符:	* &
分量运算符:	. ->
逗号运算符:	,

3.2 关系运算/逻辑运算/条件运算

- 关系运算：将两个数值进行比较，判断其比较的结果是否符合给定的条件

- 关系表达式只能表达一些简单的条件
- 每个判断只是对一个条件进行测试

- 逻辑运算：通过逻辑运算符把简单的条件组合起来，能够形成更加复杂的条件

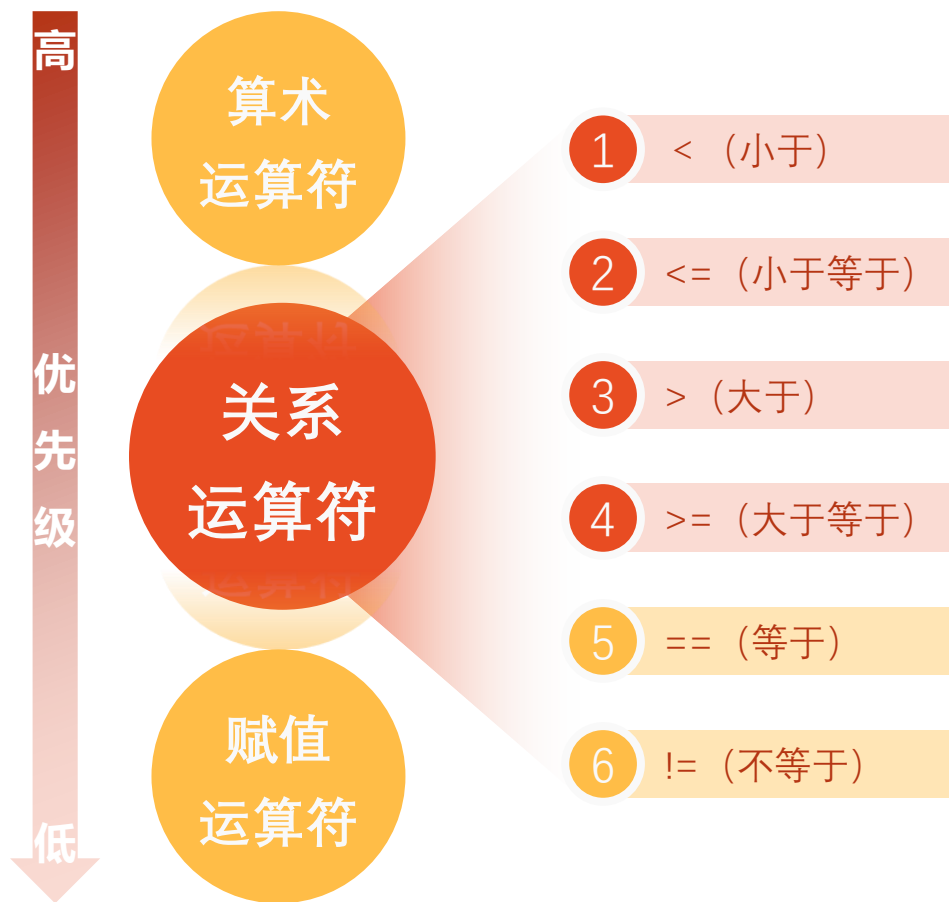
- 例： $10 > y > 5$ 的逻辑表达式 $(y > 5) \ \&\& \ (y < 10)$

- 例： $x < -10$ 或者 $x > 0$ 的逻辑表达式 $(x < -10) \ || \ (x > 0)$

- 条件运算：C语言中唯一的一个三目运行符，根据不同的结果来决定计算哪个子表达式

3.2.1 关系运算符与关系表达式

■ 运算符优先级



- 前 4 种关系运算符的优先级别相同，后 2 种也相同。前 4 种高于后 2 种。
- 关系运算符的优先级低于算术运算符。
- 关系运算符的优先级高于赋值运算符。

$c > a + b$ 等效于 $c > (a + b)$

(关系运算符的优先级低于算术运算符)

$a > b == c$ 等效于 $(a > b) == c$

(大于运算符 > 的优先级高于相等运算符 ==)

$a == b < c$ 等效于 $a == (b < c)$

(小于运算符 < 的优先级高于相等运算符 ==)

$a = b > c$ 等效于 $a = (b > c)$

(关系运算符的优先级高于赋值运算符)

3.2.1 关系运算符与关系表达式

- 用关系运算符将两个数值或数值表达式连接起来的式子，称为关系表达式
- 关系表达式的值是一个逻辑值，即“真”或“假”
- 在C的逻辑运算中，以“1”代表“真”，以“0”代表“假”

若 $a=3$, $b=2$, $c=1$, 则:

$d=a>b$, 由于 $a>b$ 为真, 因此关系表达式 $a>b$ 的值为1, 所以赋值后 d 的值为1。

$f=a>b>c$, 则 f 的值为0。因为“ $>$ ”运算符是自左至右的结合方向, 先执行“ $a>b$ ”得值为1, 再执行关系运算“ $1>c$ ”, 得值0, 赋给 f , 所以 f 的值为0

3.2.2 逻辑运算符与逻辑表达式

■ 运算符优先级

运算符	含义	举例	说明
!	逻辑非(NOT)	!a	如果a为假, 则!a为真;如果a为真, 则!a为假
&&	逻辑与(AND)	a && b	如果a和b都为真, 则结果为真, 否则为假
	逻辑或(OR)	a b	如果a和b有一个以上为真, 则结果为真, 二者都为假时, 结果为假

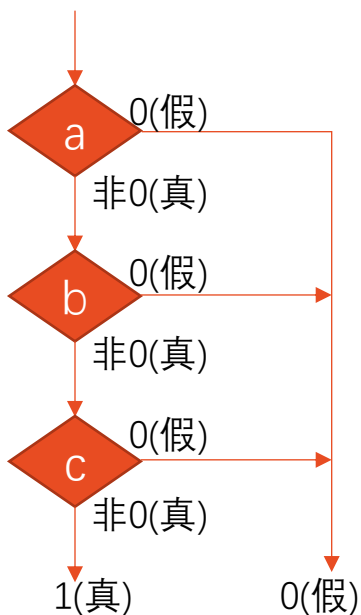
- “&&”和“||”是双目运算符, 要求有两个运算对象(操作数); “!”是单目运算符, 只要有一个运算对象
- 优先次序: !(非)→&&(与)→||(或), 即“!”为三者中最高的; 逻辑运算符中的“&&”和“||”低于关系运算符, “!”高于算术运算符
- 逻辑运算结果不是0就是1, 不可能是其他数值。而在逻辑表达式中作为参加逻辑运算的运算对象可以是0(“假”)或任何非0的数值(按“真”对待)

a	b	!a	!b	a && b	a b
真 (非0)	真 (非0)	假 (0)	假 (0)	真 (1)	真 (1)
真 (非0)	假 (0)	假 (0)	真 (1)	假 (0)	真 (1)
假 (0)	真 (非0)	真 (1)	假 (0)	假 (0)	真 (1)
假 (0)	假 (0)	真 (1)	真 (1)	假 (0)	假 (0)

3.2.2 逻辑运算符与逻辑表达式

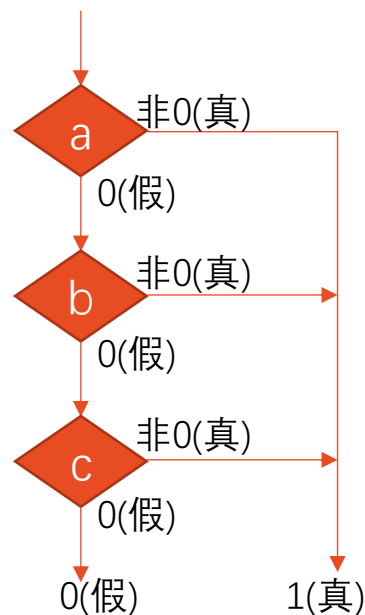
■ 在逻辑表达式的求解中，并不是所有的逻辑运算符都被执行，只是在必须执行下一个逻辑运算符才能求出表达式的解时，才执行该运算符。

- $a \ \&\& \ b \ \&\& \ c$ 。只有a为真(非0)时，才需要判别b的值。只有当a和b都为真时才需要判别c的值。



逻辑
表达式

- $a \ || \ b \ || \ c$ 。只要a为真(非0)，就不必判断b和c。只有a为假，才判别b。a和b都为假才判别c。



3.2.3 条件运算符和条件表达式

```
if (a>b)
```

```
    max=a;
```

```
else
```

```
    max=b;
```



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

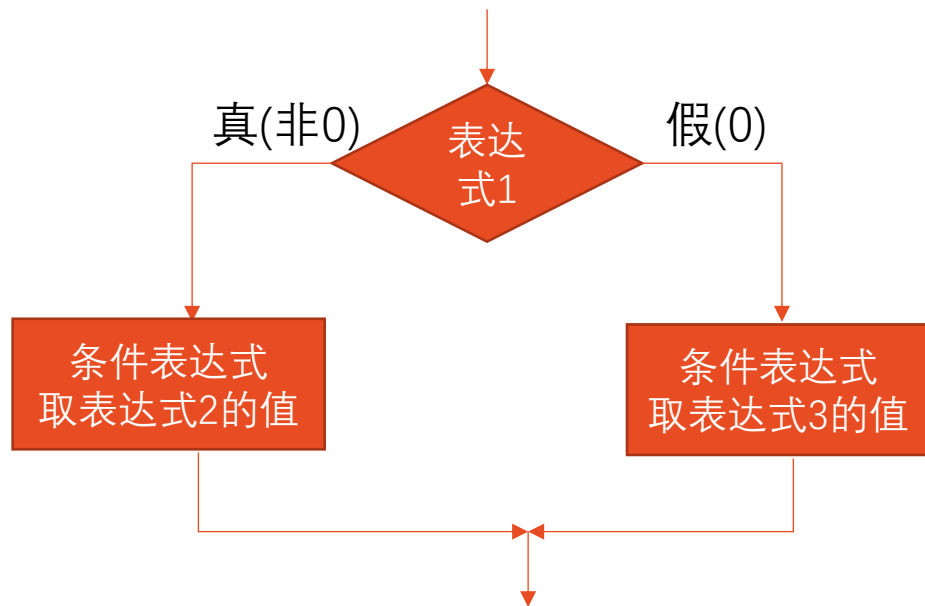
或

```
a>b ? (max=a) : (max=b);  
//表达式2和表达式3是赋值表达式
```

表达式1 ? 表达式2 : 表达式3

条件运算符由两个符号(?)和(:)组成, 必须一起使用。要求有3个操作对象, 称为三目(元)运算符, 它是C语言中唯一的一个三目运算符。

条件运算符的执行顺序: 先求解表达式1, 若为非0(真)则求解表达式2, 此时表达式2的值就作为整个条件表达式的值。若表达式1的值为0(假), 则求解表达式3, 表达式3的值就是整个条件表达式的值。



3.2.3 条件运算符和条件表达式

■ 示例：输入一个字符，判别它是否为大写字母，如果是，将它转换成小写字母；如果不是，不转换。然后输出最后得到的字符

```
1. #include <stdio.h>
2. int main()
3. {
4.     char ch;
5.     scanf("%c",&ch);
6.     ch=(ch>='A'&&ch<='Z')?(ch+32):ch;
7.     printf("%c\n",ch);
8.     return 0;
9. }
```



条件表达式“(ch>='A'&&ch<='Z')?(ch+32):ch”的作用是：如果字符变量ch的值为大写字母，则条件表达式的值为(ch+32)，即相应小写字母，32是小写字母和大写字母ASCII的差值。如果ch的值不是大写字母，则条件表达式的值为ch，即不进行转换。