

分支语句


知识点

bool数据类型

- 关系运算符

表 3-1 关系运算符

操作符	数学符号	名称	示例 (radius 为 5)	结果
<	<	小于	radius < 0	false
<=	≤	小于或等于	radius <= 0	false
>	>	大于	radius > 0	true
>=	≥	大于或等于	radius >= 0	true
==	=	等于	radius == 0	false
!=	≠	不等于	radius != 0	true

 警示：“等于”运算符是两个等号 (==)，而不是单个等号 (=)，后者是赋值运算符。

- 布尔变量：保存布尔值的变量

```
bool lightson = true; // 声明布尔型变量并赋初值为true
```

- "true"显示为1，"False"显示为0

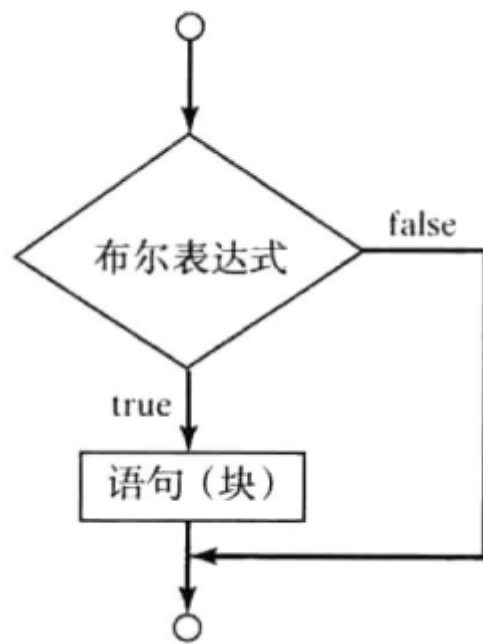
```
cout<<(4<5); // 显示输出为1，因为4<5是true
cout<<(4>5); // 显示输出为0，因为4>5是false
```

- 非零数值赋予一个bool变量，非0值为true, 0代表false

```
bool b1 = -1.5; // Same as bool b1 = true
bool b2 = 0; // Same as bool b2 = false
bool b3 = 1.5; // Same as bool b3 = true
```

if语句

```
if (boolean-expression)
{
    statement(s);
}
```



双分支的 if_else 语句

```

if (boolean-expression)
{
    statement(s)-for-the-true-case;
}
else
{
    statement(s)-for-the-false-case;
}

```

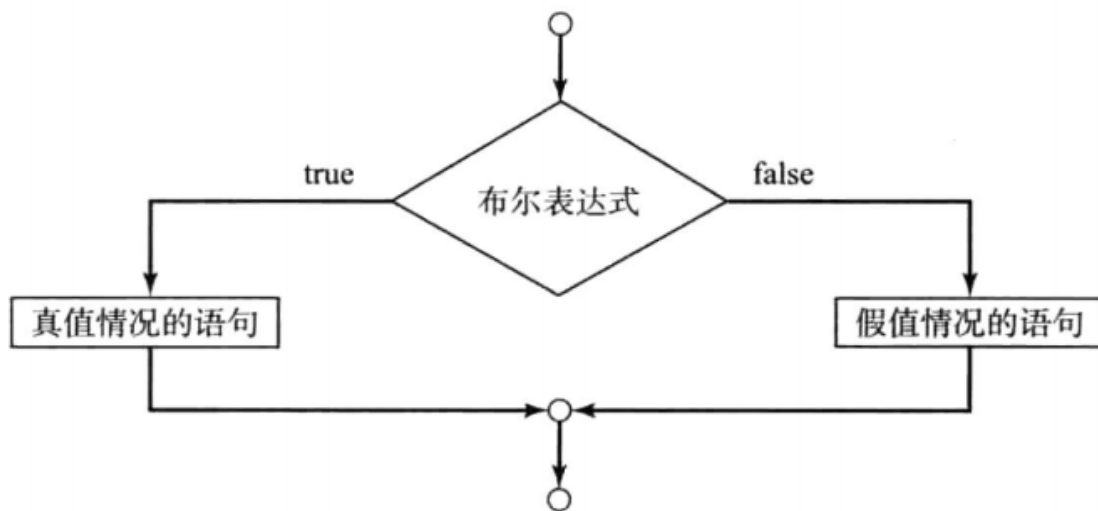


图 3-2 当布尔表达式求值为真时，if-else 语句执行针对真值情况的语句；
否则，执行针对假值情况的语句

通常，如果只有一条语句，外面一层大括号可以省略。

嵌套的if语句和多分支的if语句

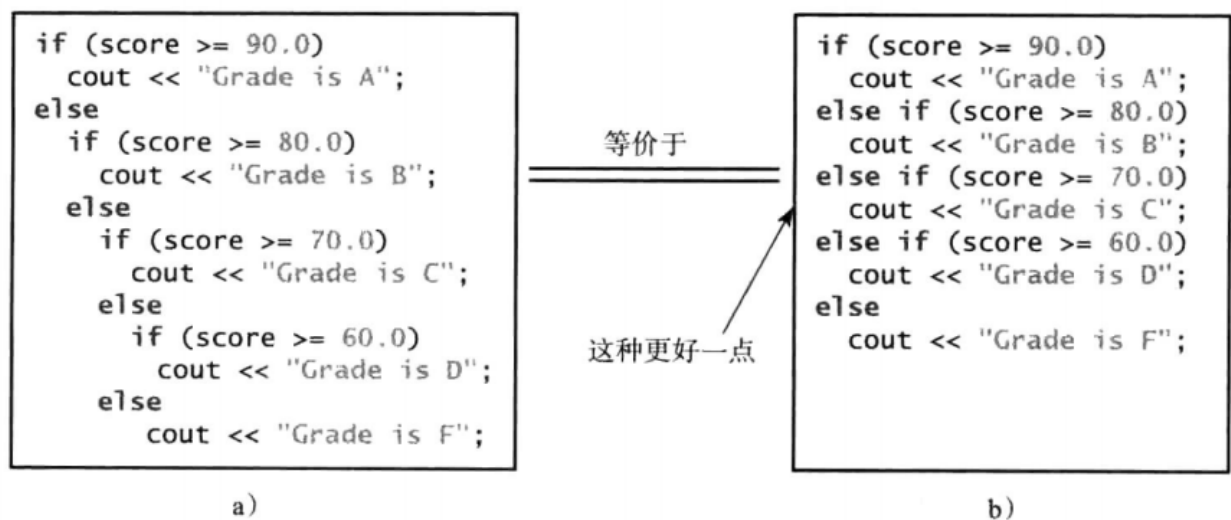
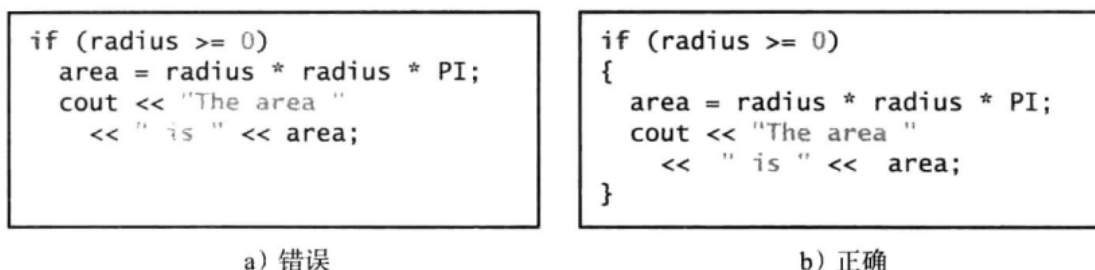


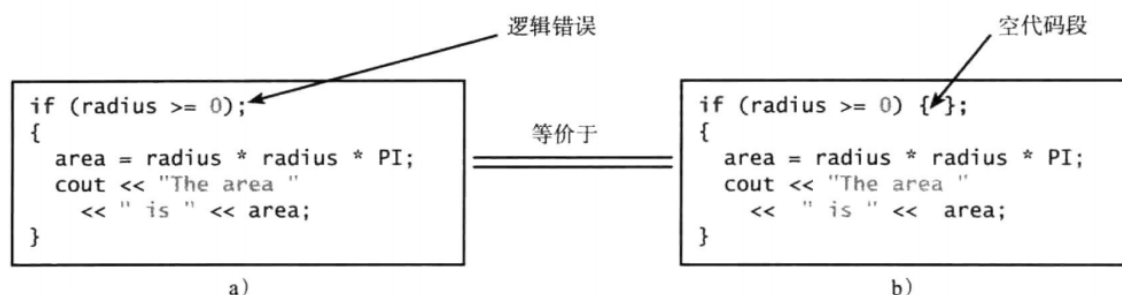
图 3-3 b 给出了多分支 if-else 语句的一种较好的代码风格

常见错误和陷阱

- 错误1：忘记必须括号



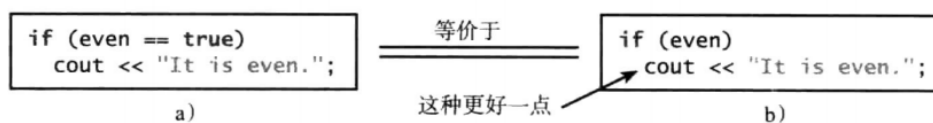
- 错误2：if行错误的分号



这个错误是很难查找，因为它既不是编译错误，也不是运行时错误，它是一个逻辑错误。a) 中的代码是等价的 b) 中一个空代码段。

- 错误3：错误使用=代替==
- 错误4：布尔值的冗余测试

为了测试条件 bool 变量是 true 还是 false，如 a) 用等价测试符 (==) 是冗余的。

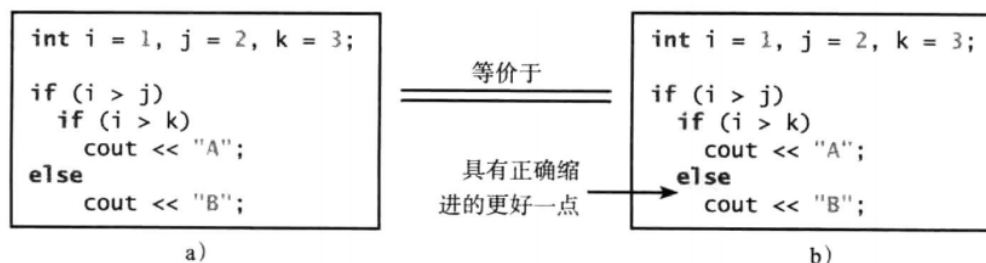


作为替代，直接测试 bool 变量，就像 b) 中一样。另外一个这样做的原因就是避免难以检测的错误。用 = 符号替代 == 来比较两个项，在测试条件句中是常见的错误。它可能会导致下面的错误语句：

```
if (even = true)
  cout << "It is even.";
```

这个语句把 true 赋值给 even，所以 even 的值一直是 true。所以，if 语句的条件一直是 true。

- 错误5：else位置歧义



else子句通常和同一程序段中最近的if子句配套

- 错误6：两个浮点值的相等性测试

浮点数有限制的精度，涉及浮点数的计算会导致舍入误差

，通常在比较两个double值的时候把 ϵ 设置为 10^{-14} ，比较两个float值的时候把 ϵ 设置为 10^{-7}

```
const double EPSILON = 1E-14;
double x = 1.0 - 0.1 - 0.1 - 0.1 - 0.1 - 0.1;
if (abs(x - 0.5) < EPSILON)
    cout << "x is approximately 0.5" << endl;
```

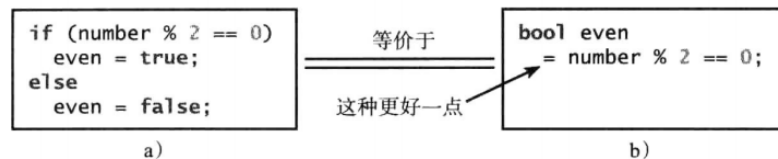
将显示：

x is approximately 0.5

cmath 库中的函数 `abs(a)` 可以用来返回一个数 `a` 的绝对值。

- 陷阱1：简化布尔变量赋值

通常，初学者在写代码中，容易像 a) 中那样，把一个条件测试语句赋值给一个 bool 变量：



这不是一个错误，但是应该写成如 b) 所示的样子更好一点。

- 陷阱2：避免在不同分支中的相同语句

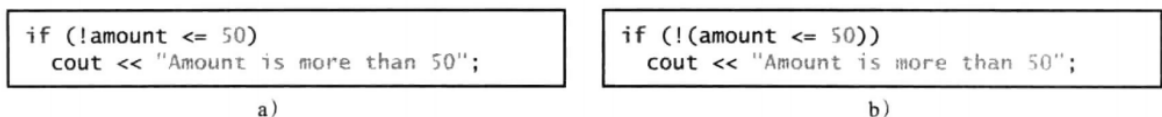
```
if (inState)
{
    tuition = 5000;
    cout << "The tuition is " << tuition << endl;
}
else
{
    tuition = 15000;
    cout << "The tuition is " << tuition << endl;
}
```

这并不是一个错，但是最好还是写成如下的样式：

```
if (inState)
{
    tuition = 5000;
}
else
{
    tuition = 15000;
}
cout << "The tuition is " << tuition << endl;
```

新代码把重复的部分移除了，让代码方便维护，因为这样修改了输出语句之后，只需要修改一处就可以了。

- 陷阱3：整数值可以被用作布尔值



生成随机数

- `rand()` 函数获得随机整数
- 种子 (seed) 控制随机数，默认情况下种子的值为1
- `srand(seed)` 改变种子的值，`time(0)` 使每次种子的值都不相同
- 生成一定区间的随机整数

1. 确定区间形式

2. 算出区间长度
3. 写出生成随机整数表达式

eg1. 区间 (m,n) :

1. 确定区间 $[m+1, n-1]$
2. 区间长度: $(n-1)-(m+1)+1 = n-m-1$
3. 表达式: $\text{rand}() \% (n-m-1) + m + 1$

eg2. 区间 $[m,n)$:

1. 确定区间: $[m, n-1]$
2. 区间长度: $n-1-m+1 = n-m$
3. 表达式: $\text{rand}() \% (n-m) + m$

eg3. 区间 (m,n]:

1. 确定区间: $[m+1, n]$
2. 区间长度: $n-(m+1)+1 = n-m$
3. 表达式: $\text{rand}() \% (n-m) + (m+1)$

eg4. 区间 $[m,n]$:

1. 确定区间: $[m,n]$
2. 区间长度: $n-m+1$
3. 表达式: $\text{rand}() \% (n-m+1) + m$

- 举例:

步骤 1: 生成两个一位的整数, 把数值赋予 number1 和 number2。

步骤 2: 如果 $\text{number1} < \text{number2}$, 把 number1 和 number2 互换位置。

步骤 3: 提示学生回答 “number1 - number2 的结果是多少?”。

步骤 4: 检查学生的回答, 并显示是否证明。

```
#include<iostream>
#include<ctime>//for time function
#include<cstdlib>//for rand and srand function
using namespace std;
int main(){
    //1.Generate 2 random single_digit intergers
    srand(time(0));
    int number1 = rand()%10;
    int number2 = rand()%10;

    //2.If number1<number2, swap number1 with number2
    if(number1 < number2){
        int temp = number1;
        number1 = number2;
        number2 = temp;
    }

    //3.Prompt the student to answer "What is number1 - number2?"
    cout<<"What is "<<number1<<" - "<<number2<<" ? ";
    int answer;
    cin>answer;
```

```
//4.Grade the answer and display the result
if(number1-number2==answer){
    cout<<"You are correct";
}
else
    cout<<"Your answer is wrong."<<number1<<" - "<<
    number2<<" should be "<<(number1-number2)<<endl;
return 0;
}
```

逻辑运算符

表 3-3 布尔运算符

运算符	名字	描述
!	逻辑非	逻辑取反
&&	逻辑与	逻辑合取
	逻辑或	逻辑析取

表 3-4 运算符!的真值表

p	$!p$	示例(假定 age=24, weight=140)
true	false	!(age > 18) 为 false, 因为 (age > 18) 为 true !(weight == 150) 为 true, 因为 (weight == 150) 为 false
false	true	


表 3-5 运算符&&的真值表

$p1$	$p2$	$p1 \&\& p2$	示例(假定 age=24, weight=140)
false	false	false	(age > 18) && (weight <= 140) 为 true, 因为 (age > 18) 和 (weight <= 140) 都不为 true
false	true	false	
true	false	false	(age > 18) && (weight > 140) 为 false, 因为 (weight > 140) 为 false
true	true	true	

表 3-6 运算符||的真值表

$p1$	$p2$	$p1 p2$	示例(假定 age=24, weight=140)
false	false	false	(age > 34) (weight <= 140) 为 true, 因为 (weight <= 140) 为 true
false	true	true	
true	false	true	(age > 34) (weight >= 150) 为 false, 因为 (age > 34) 和 (weight >= 150) 均为 false
true	true	true	

警示:

 警示: 在数学中, 表达式

$1 \leq \text{numberOfDaysInAMonth} \leq 31$

是正确的。然而, 在 C++ 中, 它是错误的, 因为 $1 \leq \text{numberOfDaysInAMonth}$ 得出的是 bool 值, 然后一个 bool 值 (1 为 true, 0 为 false) 同 31 比较, 这会导致一个逻辑错误。正确的表达式为

$(1 \leq \text{numberOfDaysInAMonth}) \&\& (\text{numberOfDaysInAMonth} \leq 31)$

Switch语句

- 语法

```
switch (switch-expression)
{
    case value1: statement(s)1;
                break;
    case value2: statement(s)2;
                break;
    ...
    case valueN: statement(s)N;
                break;
    default:    statement(s)-for-default;
}
```

*

- 遵循规则

switch 语句遵循如下规则：

- switch 表达式必须产生一个整型值，而且必须放在括号内。
- value1, ..., valueN 是整型常量表达式，即表达式中不能包含变量，如 $1 + x$ 就是非法的。这些值必须是整型值，不能是浮点型值。
- 当某个 case 语句中的值与 switch 表达式的值相等，则从此 case 语句开始执行后续语句，直至遇到一个 break 语句或者到达 switch 语句末尾。
- default 情况是可选的，它用于指出，在任何指定情况均与 switch 表达式不匹配时，执行什么动作。

- 关键字 break 是可选的，break 语句会立刻终止 switch 语句。

条件表达式

- 语法

很明显，条件表达式与前面的语句有着完全不同的形式，其中没有显式的 if 语句。条件表达式的语法如下所示：

```
boolean-expression ? expression1 : expression2;
```


• 举例

假如想让变量 num1 和 num2 中的较大者赋予变量 max，那么可以用条件表达式写一个很简单的语句：

```
max = num1 > num2 ? num1 : num2;
```


另一个例子，下面语句在 num 为偶数时输出信息 “ num is even”，否则输出 “ num is odd”。

```
cout << (num % 2 == 0 ? "num is even" : "num is odd") << endl;
```

运算符优先级和结合律

• 优先级

表 3-7 运算符优先级表

优先级	运算符
	var++ 和 var-- (后缀)
	+, - (一元加、减), ++var 和 --var (前缀)
	static_cast<type>(v), (type)(Casting)
	! (逻辑非)
	*, /, % (乘、除和模)
	+, - (二元加、减)
	<, <=, >, >= (关系)
	==, != (等于)
	&& (逻辑与)
	(逻辑或)
	=, +=, -=, *=, /=, %= (赋值)

• 结合律

除赋值运算符外所有二元运算符都是左边结合

由于 + 和 - 优先级相同，而且是左结合的，因此

$$a - b + c - d \text{ 等价于 } ((a - b) + c) - d$$

赋值运算符是右结合的 (right associative)，因此

$$a = b += c = 5 \text{ 等价于 } a = (b += (c = 5))$$

调试

习题

题目：

假设写一个程序来玩彩票。程序随机生成一个两位数作为中奖号码，提示用户输入一个两位数，然后通过比较看用户按照下面的规定是否赢了：

- 1) 如果用户的输入符合彩票数字的正确顺序，奖金是 \$10 000。
- 2) 如果用户输入的数字和彩票数字都相同，奖金是 \$3000。
- 3) 如果用户输入的数字和彩票数字中的一个相同，奖金是 \$1000。

注意，生成的数字的两个位置都有可能是 0。如果一个数字小于 10，我们就给这个数字前面补充一个 0 来形成一个两位数。举个例子，数字 8 被看做是 08，0 被看做是 00。程序清单 3-7 给出了完整的程序。

代码：

```

#include <iostream>
#include <ctime>
#include <cstdlib>
using namespace std;

int main() {
    srand(time(0));
    int number_1 = rand() % 100;
    int number_2;
    int a = number_1 / 10;
    int b = number_2 / 10;
    int c = number_1 % 10;
    int d = number_2 % 10;
    cin >> number_2;
    if (number_1 == number_2) {
        cout << "You'll gain $10000." << endl;
    } else if (a == d && c == b) {
        cout << "You'll gain $3000." << endl;
    } else if (!((a != b && a != d) && (c != b && c != d))) {
        cout << "You'll gain $1000." << endl;
    }

    return 0;
}
```

T2

题目:

*3.1 (代数: 解二次方程) 二次方程 $ax^2 + bx + c = 0$ 的两个根可由下列公式得出:

$$r_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \text{ 与 } r_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

其中 $b^2 - 4ac$ 称为二次方程的判别式。如果它为正, 则方程会有两个实根。如果它为零, 则方程有一个根。如果它是负的, 则方程无实根。

编写程序, 提示用户输入 a 、 b 、 c 的值, 输出基于判别式的结果。如果判别式为正, 则输出两个根。如果判别式为 0, 输出一个根。否则, 输出 "The equation has no real roots."

注意, 可以使用 `pow(x, 0.5)` 来计算 \sqrt{x} 。下面为一个运行样例:

```
Enter a, b, c: 1.0 3 1
The roots are -0.381966 and -2.61803
```

```
Enter a, b, c: 1 2.0 1
The root is -1
```

```
Enter a, b, c: 1 2 3
The equation has no real roots
```

代码:

```

#include <iostream>
#include <cmath>
using namespace std;
int main() {
    cout << "Enter a, b, c" << endl;
    double a, b, c;
    cin >> a >> b >> c ;
    double d = pow(b, 2) - 4 * a * c;

    if (d > 0) {
        double e = (b + pow(d, 0.5)) / (2 * a);
        double f = (b - pow(d, 0.5)) / (2 * a);
        cout << e << " " << f << endl;
    } else if (d == 0) {
        double g = (0 - b + pow(d, 0.5)) / (2 * a);
        cout << g << endl;
    } else
        cout << "The equation has no real roots." << endl;

    return 0;
}
```

练习出错点：把e和f放到了第一个if循环之前

T3

题目：

3.2（检验数字）编写程序，提示用户输入两个整数，检验第一个数是否能由第二个数整除。下面为一个运行样例：

Enter two integers: 2 3 <input type="button" value="Enter"/>
2 is not divisible by 3

Enter two integers: 22 2 <input type="button" value="Enter"/>
22 is divisible by 2

代码：

```
#include <iostream>
using namespace std;

int main() {
    cout << "Enter two intergers." << endl;
    int a, b;
    cin >> a >> b;
    switch (bool (a % b)) {
        case 0:
            cout << a << " is divisible by " << b;
            break;
        case 1:
            cout << a << " is not divisible by " << b;
            break;
        return 0;
    }
```

关键点：尝试用switch 代替if_else语句

T4

题目：

*3.3（代数：解 2×2 线性方程组）可以使用克莱姆法则解下列 2×2 方程组：

$$\begin{matrix} ax+by=e \\ cx+dy=f \end{matrix} \quad x=\frac{ed-bf}{ad-bc} \quad y=\frac{af-ec}{ad-bc}$$

编写程序，提示用户输入 a 、 b 、 c 、 d 、 e 和 f ，输出结果。如果 $ad-bc$ 为 0，则输出 “The equation has no solution”。

Enter a, b, c, d, e, f: 9.0 4.0 3.0 -5.0 -6.0 -21.0 <input type="button" value="Enter"/>
x is -2.0 and y is 3.0

Enter a, b, c, d, e, f: 1.0 2.0 2.0 4.0 4.0 5.0 <input type="button" value="Enter"/>
The equation has no solution

代码：

```

#include <iostream>
using namespace std;
int main() {
    cout << "Enter a,b,c,d,e,f." << endl;
    double a, b, c, d, e, f;
    cin >> a >> b >> c >> d >> e >> f;
    double g = a * d - b * c;
    if (g != 0) {
        double x = (e * d - b * f) / g;
        double y = (a * f - e * c) / g;
        cout << "x is " << x << ", y is " << y << endl;
    } else
        cout << "The equation has no solution." << endl;

    return 0;
}

```

关键点：简单的if_else语句，适当简化代码（变量g的出现）

T5

题目：

*3.9（求一个月的天数）编写一个程序，提示用户输入月份和年份，输出该月的天数。例如，如果用户输入月份为2，年份为2012，程序应该显示2012年2月有29天。如果用户输入月份为3，年份为2015，程序应该输出2015年3月有31天。

代码：

```

#include <iostream>
using namespace std;

int main()
{
    cout << "请输入月份和年份: " << endl;
    int month = 0, year = 0;
    cin >> month >> year;

    cout << year << "年" << month << "月有";
    switch (month)
    {
        case 1:
        case 3:
        case 5:
        case 7:
        case 8:
        case 10:
        case 12: cout << 31; break;
        case 4:
        case 6:
    }
}

```

```

        case 9:
        case 11: cout << 30; break;
        case 2: if (year % 4 == 0 && year % 100 != 0 || year % 400 ==
0) cout << 29; //判断是否为闰年
                else cout << 28; break;
        default:
                break;
    }
    cout << "天" << endl;

    return 0;
}

```

关键点：switch 与 if_else 条件语句的结合

T6

题目：

- *3.15 (游戏：剪刀、石头、布) 编写程序进行剪刀、石头、布这个游戏。(剪刀可以剪布，石头可以敲击剪刀，布可以包裹石头。) 程序随机生成数 0、1 或者 2，它们分别代表剪刀、石头与布。程序提示用户输入数字 0、1 或者 2，然后输出信息来告知用户或者计算机赢、输或者平。下面为一个运行样例：

```

scissor (0), rock (1), paper (2): 1
The computer is scissor. You are rock. You won

```

```

scissor (0), rock (1), paper (2): 2
The computer is paper. You are paper too. It is a draw

```

代码：

```

#include <iostream>
#include <ctime>
using namespace std;

int main() {
    cout << "scissor(0), rock(1), paper(2):" << endl;
    int computer_number, input_number;
    srand(time(0));
    computer_number = rand() % 3;
    cin >> input_number;

    cout << "The computer is ";
    switch (computer_number) {
        case 0:
            cout << "scissor. ";
            break;
        case 1:
            cout << "rock. ";
            break;
        case 2:

```

```

        cout << "paper. ";
        break;
    }

    cout << "You are ";
    switch (input_number) {
        case 0:
            cout << "scissor ";
            break;
        case 1:
            cout << "rock ";
            break;
        case 2:
            cout << "paper ";
            break;
        default:
            cout << "Inputting is wrong.";
            return 0;
    }
    if (
        (input_number == 0 && computer_number == 2) ||
        ( input_number == 1 && computer_number == 0 ) ||
        ( input_number == 2 && computer_number == 1 )) {
        cout << ". You won." << endl;
    }

    else if (input_number == computer_number) {
        cout << "too. It is a draw." << endl;
    } else
        cout << ". You lost." << endl;

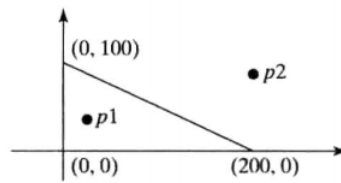
    return 0;
}

```

关键点：双重switch 与if_else语句的结合；注意运行样例字符串的表达细节

题目：

****3.23**（几何：点是否在三角形内？）假设一个正三角形放置在平面上如下所示。直角点放置在（0，0），其他两点放置在（200，0）与（0，100）。编写程序，提示用户输入带有x、y坐标的点，确定此点是否在三角形内部。



下面为几个运行样例：

Enter a point's x- and y-coordinates: 100.5 25.5 <input type="button" value="Enter"/>
The point is in the triangle

Enter a point's x- and y-coordinates: 100.5 50.5 <input type="button" value="Enter"/>
The point is not in the triangle

代码：

```
#include <iostream>
using namespace std;

int main()
{
    cout << "Enter the point's x- and y-coordinates:";
    double x, y;
    cin >> x >> y;

    /*点P在三角形ABC内部，可以通过以下三个条件判断：
        点P和点C在直线AB同侧
        点P和点B在直线AC同侧
        点P和点A在直线BC同侧
    如果以上三个条件同时满足，则点P在三角形ABC内部。*/
    //利用程序3-29可以判断点在直线的那一侧
    //顺时针遍历三个顶点时，第三个点一定位于剩余两点组成有向线段的右侧
    //所以，顺时针时，点P一直处于有向线段的右侧，点P才在三角形内部

    double x0, x1, y0, y1;
    //1.点(0,0)指向点(0,100)的有向线段
    x0 = 0, y0 = 0, x1 = 0, y1 = 100;
    bool flag1 = (x1 - x0) * (y - y0) - (x - x0) * (y1 - y0) < 0; //小于0则位于线段右侧
    //2.点(0,100)指向点(200,0)的有向线段
    x0 = 0, y0 = 100, x1 = 200, y1 = 0;
    bool flag2 = (x1 - x0) * (y - y0) - (x - x0) * (y1 - y0) < 0;
    //3.点(200,0)指向点(0,0)的有向线段
    x0 = 200, y0 = 0, x1 = 0, y1 = 0;
    bool flag3 = (x1 - x0) * (y - y0) - (x - x0) * (y1 - y0) < 0;

    if (flag3 && flag2 && flag1)
        cout << "The point is in the triangle" << endl;
```



```

else
    cout << "The point is not in the triangle" << endl;

return 0;
}

```

关键点：bool变量与if_else条件语句

T8

题目：

****3.25**（几何：两个矩形）编写程序，提示用户输入两个矩形的中心 x 、 y 、宽度与高度，检查第二个矩形是否在第一个矩形内或是否与第一个有重叠，如图 3-9 所示。对所有形况测试程序。

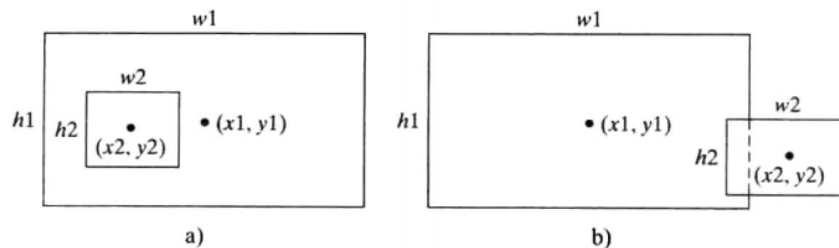


图 3-9 a) 一个矩形在另一个矩形内，b) 一个矩形与另一个有重叠

下面为几个运行样例：

```

Enter r1's center x-, y-coordinates, width, and height: 2.5 4 2.5 43
Enter r2's center x-, y-coordinates, width, and height: 1.5 5 0.5 3
r2 is inside r1

```

```

Enter r1's center x-, y-coordinates, width, and height: 1 2 3 5.5
Enter r2's center x-, y-coordinates, width, and height: 3 4 4.5 5
r2 overlaps r1

```

```

Enter r1's center x-, y-coordinates, width, and height: 1 2 3 3
Enter r2's center x-, y-coordinates, width, and height: 40 45 3 2
r2 does not overlap r1

```

代码：

```

#include <iostream>
using namespace std;
int main() {
    double x1, x2, y1, y2, w1, w2, h1, h2;
    cout << "Enter r1's center x-,y-coordinates, width, and height."
    << endl;
    cin >> x1 >> y1 >> w1 >> h1;
    cout << "Enter r2's center x-,y-coordinates, width, and height."
    << endl;
    cin >> x2 >> y2 >> w2 >> h2;
    if ((w1 - w2) / 2 <= abs(x1 - x2) < w1 / 2) &&

```

```

        ((h1 - h2) / 2 <= abs(h1 - h2) < h1 / 2) &&
        w1 > w2 &&
        h1 > h2) {
            cout << "r2 is inside r1." << endl;
        }

    else if ( ((abs(x1 - x2) <= (w1 + w2) / 2)) &&
              ((abs(y1 - y2) <= (w1 + w2) / 2))) {
        cout << "r2 overlaps r1." << endl;
    }

    else
        cout << "r2 does not overlap r1." << endl;
    return 0;
}

```

关键点：找准最简单的两种情况（不好表述的情况用else）；
多个逻辑运算符美观写法。