

第3章 分支结构程序设计

人在一生中需要做出许多选择,小到考虑晚上吃什么,大到决定高考志愿填报的学校。只有通过一次次的选择,才能有无限种可能。因此,要根据自己的情况,在所面对的一次次选择中做出最佳的选择。

程序的执行也不是一成不变的,往往会要求程序能够在不同的场合有不同的动作,这时就需要在代码中使用条件语句来做出不同的选择。比如说,登录洛谷网时,网站后台会将登录者提交的用户名和密码在后台数据库中看看是否匹配,如果能够匹配就登录成功,否则就登录失败^①。这一章就来介绍如何让程序做出选择。图3-1所示为本章思维导图。

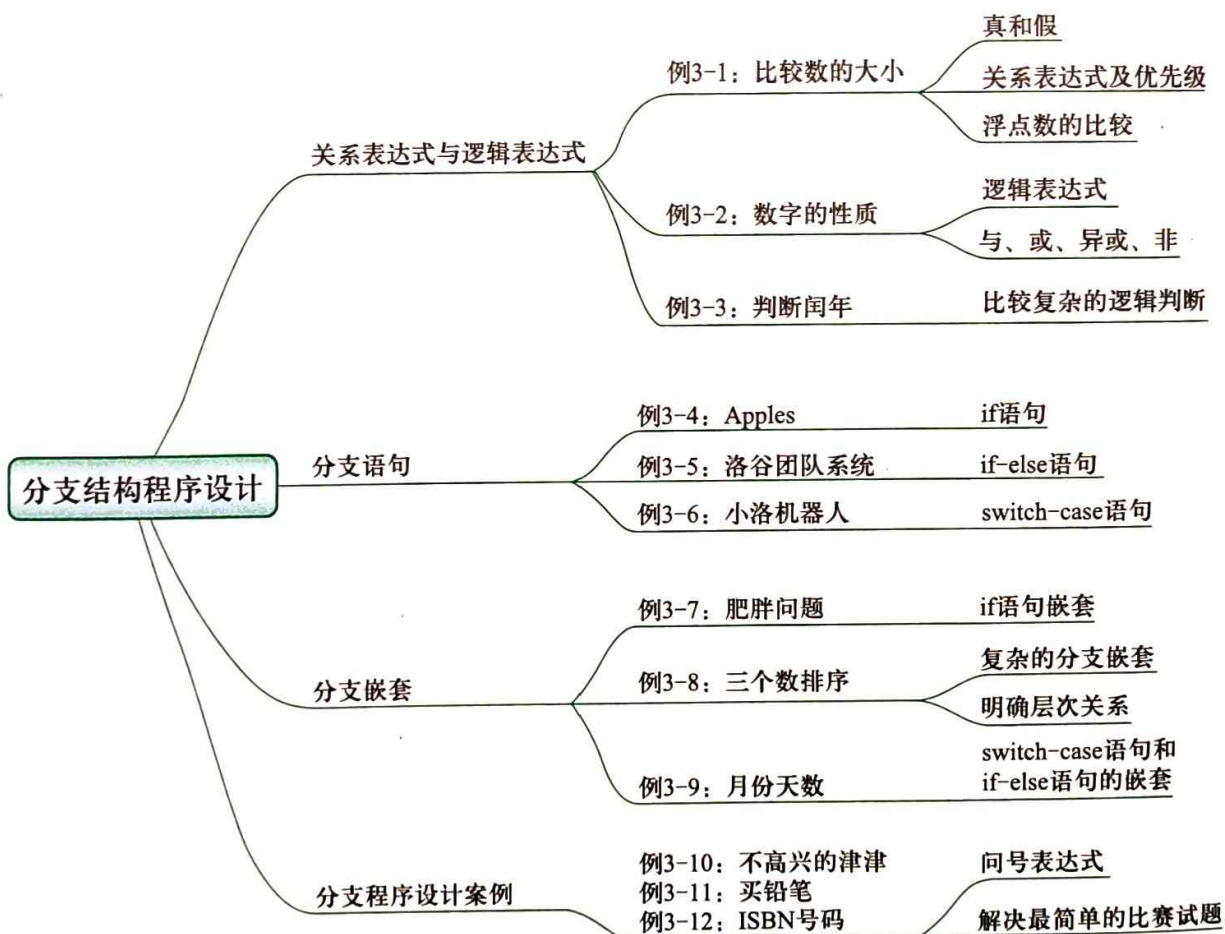


图3-1 本章思维导图

^① 为了保证安全性,实际的登录验证机制要复杂得多。

3.1 关系表达式与逻辑表达式

现实生活中,虽然很多事情真假难辨,但是在计算机中真假是黑白分明的。计算机的世界由 0 和 1 组成,0 代表假,1 代表真。这一节中将明确一些元素之间的关系,不会介绍新的编程知识点。

例 3-1 输入两个整数 a 和 b ,想知道:

- 1) a 是否大于 b ?
- 2) a 是否小于或等于 b ?
- 3) a 是否不等于 b ?

输出 3 个整数,用空格隔开。对于每一个询问,如果成立(条件为真)输出 1,否则输出 0。

分析:假设 a 为 2, b 为 3,那么很显然 $a > b$ 不成立, $a \leq b$ 成立, $a \neq b$ 成立。如果要写程序,代码如下,其输出是 0 1 1。注意,如果没有括号,则会编译错误。

```
#include <iostream>
using namespace std;
int main(){
    int a, b;
    cin >> a >> b;
    cout << (a > b) << ' ';
    cout << (a <= b) << ' ';
    cout << (a != b) << endl;
    return 0;
}
```

可以发现, $>$ 是大于, $>=$ 是大于或等于, $!=$ 是不等于。注意,如果要比较两个数是否相等,使用的是 $==$ 而不是 $=$, 因为一个等号是赋值的意思。由这些关系运算符组成的不是 0 就是 1 的表达式被称为**关系表达式**。用 1 来代表真(成立),用 0 来代表假(不成立)。在关系运算符左右两边放表达式(数字或者变量等),就会计算这两边的关系,返回 1 或者 0。

需要注意的是,如果关系表达式和算数表达式符号(如 $+$ 、 $-$ 、 $*$ 、 $/$ 、 $\%$)一起使用,需要注意优先级问题。关系表达式的地位比算术表达式还低,具体的优先级如图 3-2 所示:

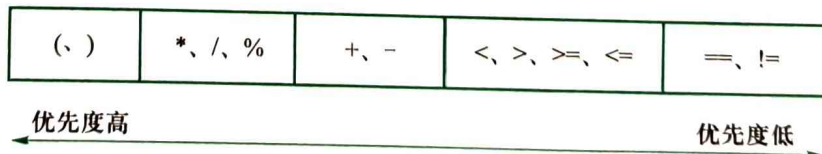


图 3-2 运算符优先级

括号的优先级最高,所以计算机会先去计算括号里面的内容,其次是乘除取余,然后是加减。而大于或小于之类的关系运算符优先级还要低一些,相等或不等的优先级更低,所以读者一定要注意运算顺序。

有一点特殊的情况:一般不会用 $==$ 来判断两个浮点数是否相等,这是因为浮点数可能会

产生精度误差。正确的方式是比较这两个数的差值是否小于一定程度。例如,假设 $\text{fabs}(a-b) < 1e-6$ 成立,就可以认为浮点数 a 和 b 相等。至于差值要小到多少取决于实际需求,一般来说不超过 $1e-6$ 就可以了。

例 3-2 数的性质(洛谷 P5710)。一些数字可能拥有以下的性质。

性质 1:是偶数。

性质 2:大于 4 且不大于 12。

小 A 喜欢这两个性质同时成立的数字;Uim 喜欢这两个性质至少符合其中一种性质的数字;八尾勇喜欢刚好有符合其中一个性质的数字;正妹喜欢不符合这两个性质的数字。现在输入一个一个数字 $x(0 \leq x \leq 100)$,要求输出这 4 个人是否喜欢这个数字,如果喜欢,则输出 1;否则,输出 0,用空格分隔。

分析:如果需要像题目这样将多个条件复合成一个条件进行判断,在汉语语境下类似“且”“或”“不”的情况,就需要使用逻辑运算符进行连接,这样的表达式称为**逻辑表达式**。和关系表达式一样,逻辑表达式也是取值 0 或者 1。

为了简化问题,可以把性质 1 和性质 2 存成 `bool` 类型的变量。`bool` 类型变量占用 1 字节,只能表示 0 或者 1 这两种取值(当然使用 `char`、`int` 这些类型来存储也是可以的)。然后再对这两种性质进行逻辑计算,代码如下:

```
#include <iostream>
using namespace std;
int main(){
    int x; bool p1, p2;
    cin >> x;
    p1 = x % 2 == 0;
    p2 = 4 < x && x <= 12;
    cout << (p1 && p2) << ' '; // 两个性质同时成立
    cout << (p1 || p2) << ' '; // 两个性质至少一个成立
    cout << (p1 ^ p2) << ' '; // 两个性质刚好一个成立
    cout << (!p1 && !p2); // 两个性质同时不成立
    // cout << !(p1 || p2); // 也可以这么写
}
```

当输入“5”时, $p1$ 不成立, $p2$ 成立,输出是“0 1 1 0”;而输入“13”时, $p1$ 和 $p2$ 都不成立,输出是“0 0 0 1”。读者可以尝试构造其他的输入并观察输出结果。

$p1$ 变量中, $\%$ 的优先级比 $==$ 的优先级更大,所以会先计算 $x\%2$ 的值,然后再计算这个值是否和 0 相等。如果 x 除 2 得到的余数等于 0,那么 $p1$ 成立,其值为 1,否则其值为 0。而 $p2$ 变量中,关系运算符的优先级要高于逻辑运算符,所以会先分别计算 $4 < x$ 和 $x \leq 12$ 的结果,然后再判断这两个条件是否同时成立。

在 C++ 中可以使用以下几种逻辑运算符。

- 1) 与运算符:用 `&&` 来判断两个条件是否同时成立。
- 2) 或运算符:用 `||` 来判断两个条件是否至少有一个成立。

3) 异或运算符^①:用 $\wedge$ 表示两个条件是否刚好一个成立、另一个不成立。

4) 非运算符:用 $!$ 可以将一个条件取反(将其颠倒)。

逻辑运算产生的结果也是0(不成立)或者1(成立),具体的运算规则如图3-3所示。

p1	p2	p1&p2
0	0	0
1	0	0
0	1	0
1	1	1

(a) 与运算符

p1	p2	p1 p2
0	0	0
1	0	1
0	1	1
1	1	1

(b) 或运算符

p1	p2	p1^p2
0	0	0
1	0	1
0	1	1
1	1	0

(c) 异或运算符

p1	!p1
0	1
1	0

(d) !非运算符

图3-3 逻辑运算规则

和运算表达式一样,可以使用逻辑运算符构造出更加复杂的逻辑表达式。比如,如果想判断两个性质是否至少一个成立,还可以直接写成 $x\%2==0||4<x\&\&x\leq 12$, 如果感觉比较混乱,这时可以加上括号使条件变得清晰,也就是 $(x\%2==0)||((4<x)\&\&(x\leq 12))$ 。请再次注意运算符优先级,以及不要把 $==$ 错误地输入成 $=$ 。图3-4所示为运算符优先级补充。

(,)	!, -(负号), ++, --	*, /, %	+, -(加减运算)	<<, >>(左右位移)	<, >, >=, <=	==, !=	&&	
优先度高				优先度低				

图3-4 运算符优先级补充

即使不加括号,上面的式子也是对的,因为根据运算优先级,会先计算运算表达式(取余),然后关系表达式(大于或小于),最后才是逻辑表达式(除了!);而 $\&\&$ 的优先级要比 $||$ 高,所以会先计算后面两项。

例3-3 闰年判断(洛谷 P5711)。输入一个年份(大于1582的整数^②),判断这一年是否为闰年,如果是,输出1;否则,输出0。

① 严格来说,异或运算符是位运算符而不是逻辑运算符,但是经常会用来做逻辑运算,所以一起在这里讲了。

② 现行公历制度是1582年颁行后的格里历。

分析:这是一个常识,被4整除是闰年,被100整除不是闰年,而被400整除又是闰年。但是这种说法并不严谨。比如说,2000年同时满足这3个条件(被4、100、400整除),但是第一和第三条说它是闰年,第二条又说它不是,所以它到底是不是闰年呢?

将这三个条件分别定义为p1、p2、p3。如果p3成立,那么它肯定是闰年。如果p3不成立,那么当p1成立且p2不成立时它也是闰年。这两种情况只要满足一种,它都是闰年。根据之前的分析,可以写成 $p3 \parallel !p3 \&\& (p1 \&\& !p2)$,其中后面的!p3是可以省略的,因为即使去掉也不影响结果,此时表达式可以写成 $p3 \parallel p1 \&\& !p2$ 。带入判断是否整除,可以写出如下程序:

```
#include <iostream>
using namespace std;
int main() {
    int x; bool p;
    cin >> x;
    p = (x % 400 == 0) || (x % 4 == 0) && (x % 100 != 0);
    //p = !(x % 400) || !(x % 4) && x % 100;
    cout << p << endl;
    return 0;
}
```

也可以使用注释中的写法。对于&&、||和!来说,参与计算的元素如果非零(无论正数还是负数),都会被视为1。例如,如果要判定x不是100的倍数时,x%100非零,那么它就会被视为1;而判定x是4的倍数时,x%4为零,需要将它取反才能使该条件为1,因此 $x\%4==0$ 和 $!(x\%4)$ 是等价的。

3.2 分支语句

例 3-4 Apples(洛谷 P5712)。八尾勇喜欢吃苹果,她今天吃掉了 $x(0 \leq x \leq 100)$ 个苹果。英语课上学到了apple这个词语,想用它来造句。如果她吃了1个苹果,就输出“Today, I ate 1 apple.”;如果她没有吃,那么就把1换成0;如果她吃了不止一个苹果,别忘了apple这个单词后面要加上代表复数的s。请帮她完成这个句子。

分析:本例中,要写出一个正确的英语句子。根据输入数据不同程序要做出不同的表现——如果输入不是0或1,就多输出一个s。代码如下:

```
#include <iostream>
using namespace std;
int main() {
    int x;
    cin >> x;
    cout << "Today, I ate " << x << " apple";
    if (x != 0 && x != 1) { // 也可写成!(x==0||x==1)
        cout << "s";
    }
}
```

```

    cout << "." << endl;
    return 0;
}

```

在这里,使用 if 语句来控制程序在指定条件下需要做什么事情。if 语句的一种用法如下:

```

if ( 成立条件表达式 ) {
    当条件成立时需要执行的语句;
}

```

首先输入 x , 知道吃了几个苹果。然后输出 Today, I ate x apple, 因为无论 x 是多少, 都要输出这样的句子。

然后, 当 x 既不是 0, 也不是 1 时, 说明要加上复数的 s 。这就是 if 语句的作用。如果条件成立, 大括号里面的语句就会执行 (在这里就是输出一个 s); 如果条件不成立, 就什么也不做, 跳过大括号中的部分。这里的“成立条件表达式”一般是指上面介绍过的关系表达式或者逻辑表达式——如果表达式是 1, 或者只要不是 0, 条件都会成立; 如果是 0, 条件就不成立了。

最后不要忘记加上结尾的句号。本程序的流程图如图 3-5 所示。

有了 if 语句, 程序就可以根据不同的情况做出不同的动作。不过有时候会希望当某个条件成立时做一些事情, 不成立时做另外一些事情, 如果这样, 可以使用 if-else 语句。

在接触 if 语句时, 写出来的代码开始有了“层次感”——代码中输出 s 的那条语句相比于其他语句要往右移动了一点, 这就是缩进。建议使用 4 个空格作为每个层级的缩进, 这样可以一目了然地知道那些语句是条件成立时才执行的。

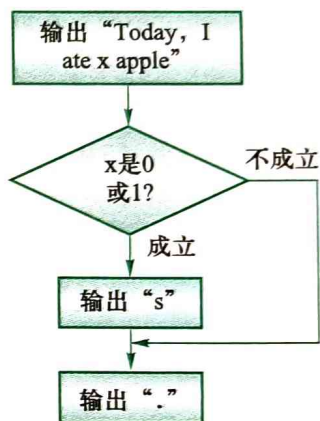


图 3-5 if 语句的流程

例 3-5 洛谷团队系统 (洛谷 P5713)。在洛谷上使用团队系统可以非常方便地添加自己的题目。如果在自己的计算机上配置题目和测试数据, 每题需要花费 5min 时间; 而在洛谷团队中上传私有题目, 每题只需要花费 3min, 但是上传题目之前还需要一次性花费 11min 创建与配置团队。现在要配置 n ($n \leq 100$) 道题目, 如果本地配置花费的总时间短, 请输出 “Local”, 否则输出 “Luogu”。

分析: 可以很容易地列出两种方式下分别消耗的时间——本地上传需要花费 $5n$ min 的时间, 而上传洛谷需要花费 $11+3n$ min 时间。如果前者小于后者, 则选择在本地配置, 否则选择洛谷上传。代码如下:

```

#include <iostream>
using namespace std;
int main() {
    int n;
    cin >> n;
    if ((5 * n) < (11 + 3 * n)) {
        cout << "Local" << endl;
    }
}

```

```

    } else {
        cout << "Luogu" << endl;
    }
    return 0;
}

```

这里是 if 语句的另外一种用法,配合上 else,就可以在条件成立的情况下做什么事情,否则,当条件不成立时做另外的一些事情,用法如下:

```

if ( 成立条件表达式 ) {
    当条件成立时需要执行的语句;
} else {
    当条件不成立时需要执行的语句;
}

```

特别地,如果需要执行的语句只有一条语句(也就是只有结尾的一个分号),那么大括号可以不要(甚至换行也可以不要),而空格和换行也都不是硬性要求(必要的分割还是要的)。在这里,判断语句还能不分行,写成 `if((5*n)<(11+3*n)) cout<<"local"<<endl;else cout<<"Luogu"<<endl;` 也是可以的,但是这样可读性比较差,因此不推荐。

整个过程的流程图如图 3-6 所示。

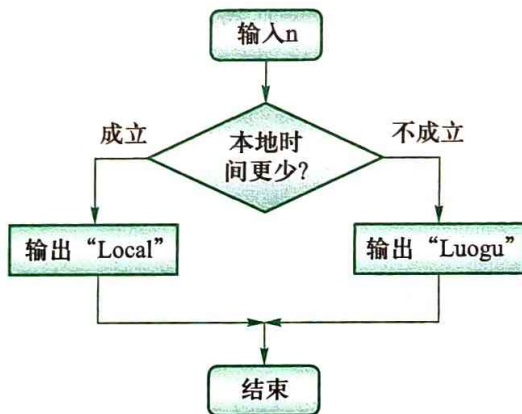


图 3-6 if-else 语句的流程图

使用大括号括起来的语句称为代码块。代码块里面的代码相比于外面应当有统一的缩进(建议 4 个空格)。开头大括号可以像例子中给出的那样不换行,也可以另起一行,看个人习惯。总之,写程序应当有良好的代码规范,比如换行、空格、缩进等,这也是为了使代码可读性强,更容易理解。

例 3-6 小洛机器人。小洛机器人是洛谷自行研发的人工智能聊天机器人。不过,目前它只支持最基本的几个功能,需要给它提供以下指令(一个字符),它才会按照指令给出对应的回复。

- 1) 输入“G”:打招呼,小洛会回复“Hello, my master!”,还会在下一行加上一句“I’m Xiaoluo.”。
- 2) 输入“N”:自我介绍,小洛只会回复“I’m Xiaoluo.”。
- 3) 输入“S”:唱歌,小洛会哼唱“Teinei teinei teinei~”。
- 4) 输入“B”或者“Q”:告别,小洛会说“Bye bye!”。
- 5) 其他任何字符:小洛无法理解,只能回复“Sorry...”。

作为小洛机器人的总设计师,请编写程序实现以上功能。

分析:这里的分支语句不像前面的 if 那样只需要判断是否成立了,而是需要根据读到的字母指令来做出不同的操作。可以使用多个 if 语句来判断多个条件,某个条件成立时执行相应的动作。不过,也可以使用更简单的办法,代码如下:

```
#include <iostream>
using namespace std;
int main() {
    char opt;
    cin >> opt;
    switch (opt) {
        case 'G': cout << "Hello, my master!" << endl;
        case 'N': cout << "I'm Xiaoluo." << endl; break;
        case 'S': cout << "Teinei teinei teinei~" << endl; break;
        case 'B': case 'Q':
            cout << "Bye bye!" << endl;
            break;
        default: cout << "Sorry.." << endl;
    }
    return 0;
}
```

这里使用了 switch-case 语句,可以判断一个变量是什么值,根据不同的值来进行操作,其一般结构如下:

```
switch (变量名) {
    case 变量可能的情况 1: 执行语句 1; break;
    case 变量可能的情况 2: 执行语句 2; break;
    ...
    default: 执行语句 n;
}
```

读入变量 opt,然后使用 switch 语句来看看它可能有哪些取值。当发现 opt 的值是 N 时,输出对应的自我介绍语句,然后遇到了 break,就跳出了 switch;如果发现值是 G 时,会输出对应的打招呼语句,由于这里没有 break,它就会接着运行下一条,也就是输出自我介绍,此时发现 break 后跳出;如果发现 opt 的值是 B 或者 Q 时,就会输出告别语句然后跳出;如果 opt 的值不是上面的几种情况,就会运行 default 后面的语句,也就是抱歉。需要再次强调的是,如果某一种情况运行完后没有 break,它就会接着运行下一种情况的语句。

switch-case 语句的 case 需要的只能是常量,不能是变量。这些常量类型可以是整数,也可以是字符类型等,但是不能使用浮点数(如果用浮点数,则还是要用多重 if 嵌套作为分支判断,最好不要直接用 == 判断浮点数相等)

3.3 分支嵌套

例 3-7 肥胖问题(洛谷 P5714)。BMI 指数是国际上常用的衡量人体胖瘦程度的一个标准,其算法是 m/h^2 ($40 \leq m \leq 120, 1.4 \leq h \leq 2.0$),其中 m 是指体重(单位:kg), h 是指身高(单位:m)。不同体型范围与判定结果如下:

- 1) 小于 18.5kg/m^2 : 体重过轻,输出 “Underweight”。
- 2) 大于或等于 18.5kg/m^2 且小于 24kg/m^2 : 正常体重,输出 “Normal”。
- 3) 大于或等于 24kg/m^2 : 肥胖,不仅要输出 BMI 值(使用 cout 的默认精度),还要输出 “Overweight”。

现在给出体重和身高数据,需要根据 BMI 指数判断体型状态并输出对应的判断。

分析: 本例中需要计算 BMI 指数,直接按照给出的公式计算即可,关键是如何根据不同的区间来分类(这回是分成了 3 组),这里还是使用 if 语句。if 语句可以嵌套,组成更复杂的条件分支,代码如下:

```
#include <iostream>
using namespace std;
int main() {
    double m, h, BMI;
    cin >> m >> h;
    BMI = m / h / h;
    if (BMI < 18.5)
        cout << "Underweight";
    else if (BMI < 24)
        cout << "Normal";
    else {
        cout << BMI << endl;
        cout << "Overweight" << endl;
    }
    return 0;
}
```

首先判断是否体重过轻,先判断 BMI 是否小于 18.5,如果成立就判断为体重过轻;剩下的部分,再次判断是否是小于 24,如果成立就判断为正常(体重过轻的情况在这里已经被筛选出去了,留下的都是正常或者超重);如果不成立,说明 BMI 大于或等于 24(已经把体重过轻和正常筛选出去了,剩下的都是胖子),达到了肥胖标准。

在本例中,如果某个条件成立所需要执行的语句只有一条语句,在代码结构清晰的情况下,可以不加上大括号,但是在该语句前面加上 4 个空格的缩进为宜。



例 3-8 三个数排序(洛谷 P5715)。给出三个整数 a 、 b 、 c ($0 \leq a, b, c \leq 100$), 要求把这三个整数从小到大排序。

解法 1: 将 3 个整数从小到大排列, 可能有 $[abc][acb][bac][bca][cab][cba]$ 这 6 种排列。枚举这 6 种排列的关系(小中大), 就可以依次判断属于那种情况了。比如, 当 $a \leq b \leq c$ 时, 判断条件就是 $a \leq b \&\& b \leq c$ 。在 C++ 中, 不可以连着写成 $\text{if}(a \leq b \leq c)$ 。

```
#include <cstdio>
using namespace std;
int main() {
    int a, b, c;
    scanf("%d%d%d", &a, &b, &c);
    if (a <= b && b <= c) printf("%d %d %d\n", a, b, c);
    else if (a <= c && c <= b) printf("%d %d %d\n", a, c, b);
    else if (b <= a && a <= c) printf("%d %d %d\n", b, a, c);
    else if (b <= c && c <= a) printf("%d %d %d\n", b, c, a);
    else if (c <= a && a <= b) printf("%d %d %d\n", c, a, b);
    else /*if (c <= b && b <= a)*/ printf("%d %d %d\n", c, b, a);
    return 0;
}
```

需要注意的是, 这 6 个条件并不是互斥的。如果不加上 `else`, 而且输入数据是 $a=b=c$ 时, 这些条件同时满足, 就会输出多次; 将小于或等于换替换成小于也是不可以的, 这还因为, 如果出现数据相等的情况, 那么这些条件一个都不满足, 将导致没有输出。因此, 必须在每种情况结束后加上 `else`, 不管输入是什么, 都能够刚好进入到其中一个分支中。

最后一个 `else` 这里, 出现了 `/* ... */` 的语句, 这也是注释。这种注释比较自由地控制需要注释部分的起始点和终止点。这种写法可以对一行内的内容进行注释, 也可以一下子注释多行内容。程序运行时, 注释内的内容将会被忽略。本程序中的注释语句可加可不加, 因为枚举了前 5 种情况, 剩下的情况肯定就是这一种了。

解法 2: 另外一种做法是, 首先拎出最大的一个数字, 分为 3 种情况—— a 最大、 b 最大或 c 最大。对于每一种情况, 还能分成两种情况, 就是剩下两个变量的大小关系。这样还是有 6 种情况。按照这样的思路, 可以写出这样的代码:

```
#include <cstdio>
using namespace std;
int main() {
    int a, b, c;
    scanf("%d%d%d", &a, &b, &c);
    if (a >= b && a >= c)
        if (b >= c) printf("%d %d %d\n", c, b, a);
        else printf("%d %d %d\n", b, c, a); // 这个else搭配哪个if呢?
    else if (b >= a && b >= c)
        if (a >= c) printf("%d %d %d\n", c, a, b);
```

```

        else printf("%d %d %d\n", a, c, b);
    else // if (c >= a && c >= b) 本句可加可不加
        if (a >= b) printf("%d %d %d\n", b, a, c);
        else printf("%d %d %d\n", a, b, c);
    return 0;
}

```

和上面一种解法一样,对于 if 语句下属的语句,都要留出缩进;此外判断 c 是否是最大的逻辑表达式是可加可不加的,因为不是 a 最大,也不是 b 最大,那自然就是 c 最大。

还有一个问题:观察第 8 行的 else,它前面有两个 if 语句,那么这个 else 语句和哪个 if 语句搭配呢?是第一个,还是最接近的一个?不一定,还是观察层级结构:最里层的 else 对应最里层的 if,次里层的 else 对应次里层的 if,所以这个 else 对应的是第 7 行的 if(因为是最里层的)。这里使用缩进格式,从而清楚地显示了复杂条件分支语句的关系。如果没有良好的缩进习惯,编程人员也容易被绕晕^①。

如果要求将这个 else 和最前面的那个 if 配对,而不是和上面一行的 if 配对,该怎么办呢?只需要将上一行的那个 if 连同它的条件成立执行语句使用大括号括起来即可,就像下面这样:

```

if (条件 1) {
    if (条件 2)
        条件 2 成立执行语句;
} else
    条件 1 不成立执行语句;

```

 **例 3-9** 月份天数(洛谷 P5716)。输入年份和月份,输出这一年的这一月有多少天。需要考虑闰年。

分析:不管是哪一年,除了 2 月之外的其他月份天数都是固定的,因此可以使用 switch-case 语句来先判断月份是大月还是小月——可能是 31 天的大月,也有可能是 30 天的小月;但是如果是 2 月,那么情况就不一样了,天数跟是否是闰年有关。如何判断闰年在本章已经介绍过了,在这里直接嵌套判断闰年的逻辑表达式,如果是闰年输出 29,否则输出 28。不要忘记加上 break 跳出分支语句。

```

#include <iostream>
using namespace std;
int main() {
    int y, m;
    cin >> y >> m;
    switch (m) {
        case 1: case 3: case 5: case 7: case 8: case 10: case 12:
            cout << 31 << endl; break;
        case 4: case 6: case 9: case 11:

```

① 在 Python 语言中,是需要强制使用缩进来表示层次关系的。

```

        cout << 30 << endl; break;
    case 2:
        if (!(y % 400) || !(y % 4) && y % 100)
            cout << 29 << endl;
        else
            cout << 28 << endl;
        break;
    default: break;
}
return 0;
}

```

如果能保证输入数据一定是合法的,那么分支语句中的 default 不写也是可以的,毕竟所有的情况都能够取得到。

3.4 分支程序设计案例

例 3-10 不高兴的津津(洛谷 P1085,NOIP2004 普及组)。津津除了上学之外,还要参加各科课外补习班。津津如果一天学习超过 8h 就会不高兴,而且上得越久就会越不高兴。假设津津不会因为其他事不高兴,并且她的不高兴不会持续到第二天。已知津津下周每天的上学时间和补习班学习时间,看看下周她会不会不高兴;如果会,那么哪天最不高兴。

分析: 程序中需要记录学习时间最长的一天,并记下来是星期几。首先定义变量 maxtime 记录最长的学习时间,将其初始化为 8 的原因是当学习时间超过 8h 才会去“打破”这个记录,否则这一天就过去了。maxday 变量用于记录星期几的时候打破纪录的。

然后就是整齐划一地处理每一天了,看起来很长但实际上是同样重复的内容。每次读入两个数,然后判断这两个数都能否打破纪录。如果能够打破纪录就更新 maxtime 和 maxday 这两个变量。所有 7 组数据读入处理后,直接输出答案即可。

```

#include <iostream>
using namespace std;
int main() {
    int t1, t2, maxtime = 8, maxday = 0;
    cin >> t1 >> t2;
    if (t1 + t2 > maxtime) maxtime = t1 + t2, maxday = 1;
    cin >> t1 >> t2;
    if (t1 + t2 > maxtime) maxtime = t1 + t2, maxday = 2;
    cin >> t1 >> t2;
    if (t1 + t2 > maxtime) maxtime = t1 + t2, maxday = 3;
    cin >> t1 >> t2;
    if (t1 + t2 > maxtime) maxtime = t1 + t2, maxday = 4;
    cin >> t1 >> t2;
    if (t1 + t2 > maxtime) maxtime = t1 + t2, maxday = 5;
}

```

```

cin >> t1 >> t2;
if (t1 + t2 > maxtime) maxtime = t1 + t2, maxday = 6;
cin >> t1 >> t2;
if (t1 + t2 > maxtime) maxtime = t1 + t2, maxday = 7;
cout << maxday;
return 0;
}

```

写这么多重复的语句很烦琐。下1章将会介绍循环语句,这样只需要把这7天的处理语句写一遍就可以解决问题。

例 3-11 买铅笔(洛谷 P1909,NOIP2016 普及组)。P 老师要买 n 支铅笔。商店一共有 3 种包装的铅笔,不同包装内的铅笔数量和总价有可能不同且已知。P 老师决定只买同一种包装的铅笔。由于铅笔的包装不能拆开,因此 P 老师可能需要购买超过 n 支铅笔才够。请问:在商店每种包装的数量都足够的情况下,要买够至少 n 支铅笔最少需要花费多少钱? 所有输入数据不超过 10000。

分析:计算出每种类型的笔需要买几包,然后乘上单价就是总价格。接着通过打擂台的方式比较选择哪种包装的笔最为便宜。假设需要 n 支笔,每包有 n_i 支,那么如果 n 可以整除 n_i (也就是 $n \% n_i == 0$),那么就需要 n / n_i 包;如果不能整除,那么还需要多买一包,也就是 $n / n_i + 1$ 包。可以使用 if 语句来进行判断,但是还有更简单的写法:

```

#include <iostream>
using namespace std;
int main() {
    int n, n1, n2, n3, p1, p2, p3, t1, t2, t3, total;
    cin >> n >> n1 >> p1 >> n2 >> p2 >> n3 >> p3;
    t1 = !(n % n1) ? n / n1 * p1 : (n / n1 + 1) * p1;
    t2 = !(n % n2) ? n / n2 * p2 : (n / n2 + 1) * p2;
    t3 = !(n % n3) ? n / n3 * p3 : (n / n3 + 1) * p3;
    total = t1; // 假设第一种是最省钱的方案
    if (t2 < total) total = t2;
    if (t3 < total) total = t3;
    cout << total << endl;
    return 0;
}

```

这里使用了问号表达式。问号表达式的形式是 $S1 ? S2 : S3$,意思是如果 $S1$ 条件成立,那么它的值就是 $S2$,否则就是 $S3$ 。本例中, $t1 = !(n \% n1) ? n / n1 * p1 : (n / n1 + 1) * p1$ 的意思就是如果条件 $!(n \% n1)$ 成立(等同于 $n \% n1 == 0$ 成立),那么 $t1$ 就是 $n / n1 * p1$;否则 $t1$ 就是 $(n / n1 + 1) * p1$,后面两条以此类推。问号表达式的优先级相当低,所以可以先将问号表达式的各项先计算出来,然后再进行判断。

本题还可以使用 `cmath` 头文件中的 `ceil()` 函数,进行上取整运算,直接得到需要购买几包铅笔。

例 3-12 (选读) ISBN(洛谷 P1055, NOIP2008 普及组)。每一本正式出版的图书都有一个国际标准书号 (ISBN)。其规定格式如 x-xxx-xxxxx-x, 前面 9 位都是数字, 最后一位是识别码。当前面 9 位已经确定后, 识别码的计算方法是: 第 1 位数字乘 1 加上第 2 位数字乘 2, 以此类推一直加到第 9 位, 其和对 11 取余数。如果余数是 10 那么识别码就是 X。现在要求编写程序判断输入的 ISBN 的识别码是否正确, 如果正确则输出 “Right”, 否则修正识别码并输出。

分析: 将每位数字读入, 然后计算识别码。如果读到的识别码是 X 且计算得到的识别码是 10, 或者读到的识别码的那位数字和计算得到的数字相等, 则识别码正确; 否则识别码错误, 需要照原样输出 ISBN, 然后输出正确的识别码, 代码如下:

```
#include <cstdio>
using namespace std;
int main() {
    char a, b, c, d, e, f, g, h, i, j;
    int check;
    scanf("%c-%c%c%c-%c%c%c%c%c-%c", &a, &b, &c, &d, &e, &f, &g, &h, &i, &j);
    check = (a-'0')*1 + (b-'0')*2 + (c-'0')*3 + (d-'0')*4 + (e-'0')*5
            + (f-'0')*6 + (g-'0')*7 + (h-'0')*8 + (i-'0')*9;
    check %= 11;
    if(j=='X'&&check==10 || check==j-'0')
        printf("Right\n");
    else
        printf("%c-%c%c%c-%c%c%c%c%c-%c", a, b, c, d, e, f, g, h, i, check==10?
            'X':check+'0');
    return 0;
}
```

这里复习了 scanf 读入和字符处理。读者还记得 ASCII 是单个字符和数字的映射吗? 使用 scanf 读入 a 到 j 的所有每一位 (注意是 char 类型, 读入的是单个字符, 而不是 int 类型)。读到的单个字符, 比如 '0' (注意是字符 '0'), 对应的整数是 48, 所以不能直接对这个字符进行识别码计算, 而是要减去 '0' 后才能变成数字 0, 这样才可以参与计算。输出的时候也要注意识别码是不是 10, 如果是, 那么还需要另外输出 'X'。

3.5 课后习题与实验

习题 3-1 当 $a=3, b=4, c=5$, 判断以下表达式是否成立?

- (1) $a < b || b > c || a > b$
- (2) $a > c || b > a \&\& c > b$
- (3) $b - a == c - b$
- (4) $a * b - c > a * c - b || a * b + b * c == b * b * (c - a)$

习题 3-2 当 $a=1, b=0, c=1$, 判断以下表达式是否成立?

- (1) $!a || !b$
- (2) $(a \&\& !a) || (b || !b)$

- (3) `a&& b&& c|| !a|| !c`
 (4) `a&& (b&& c|| a&& c)`
 (5) `!b&& (c&& (a&& (!c|| (!b|| (!a))))`

习题 3-3 对于整数变量 x , 写出与判断以下性质对应的表达式:

- (1) x 是否为偶数。
- (2) x 是否为 4 位整数。
- (3) x 是否为完全平方数(提示:数据类型转换)。
- (4) x 是否同时是奇数、完全立方数,而且是 3 位整数。
- (5) x 是否是水仙花数(也就是说 x 是各位数的立方和等于 x 的 3 位整数)。

习题 3-4 小玉家的电费(洛谷 P1422)。某地用电标准如下:月用电量在 150 千瓦时及以下部分的电价为 0.4463 元/千瓦时,月用电量在 151~400 千瓦时部分的电价为 0.4663 元/千瓦时,月用电量在 401 千瓦时及以上部分的电价为 0.5663 元/千瓦时。已知当月用电量(整数且不大于 10000),根据电价规定计算出应交的电费。

习题 3-5 小鱼的航程(洛谷 P1424)。有一只小鱼,它平日每天游泳 250km,周末休息(实行双休日),假设从周 x ($1 \leq x \leq 7$) 开始算起,过了 n ($n \leq 10^7$) 天以后,小鱼一共累计游泳了多少 km 呢?

习题 3-6 三角函数(洛谷 P1888)。输入一组勾股数 a 、 b 、 c ($a \neq b \neq c$),且不大于 10^9 ,用分数格式输出其较小锐角的正弦值。(提示:如果要求约分,则需要分子和分母各除以它们的最大公约数,学完下一章就知道怎么求了,现阶段不要求约分。)

习题 3-7 陶陶摘苹果(洛谷 P1046,NOIP2005 普及组)。一棵苹果树结出 10 个苹果。陶陶有个 30cm 高的板凳,当她不能直接用手摘到苹果的时候,就会踩到板凳上再试试。现在已知 10 个苹果到地面的高度,以及陶陶把手伸直的时候能够达到的最大高度,请帮陶陶算一下她能够摘到的苹果的数目。假设她碰到苹果,苹果就会掉下来。

习题 3-8 三角形分类(洛谷 P5717)。给出 3 条线段 a 、 b 、 c 的长度,均是不大于 10000 的整数。打算把这 3 条线段拼成一个三角形,它可以是什么三角形呢?

- (1) 如果三条线段不能组成一个三角形,输出 “Not triangle”。
- (2) 如果是直角三角形,输出 “Right triangle”。
- (3) 如果是锐角三角形,输出 “Acute triangle”。
- (4) 如果是钝角三角形,输出 “Obtuse triangle”。
- (5) 如果是等腰三角形,输出 “Isosceles triangle”。
- (6) 如果是等边三角形,输出 “Equilateral triangle”。

如果这个三角形符合以上多个条件,请分别输出,并用换行符隔开。

习题 3-9 (选做)ABC(洛谷 P4414,COCI2006)。已知 3 个整数 A 、 B 、 C 满足 $A < B < C$,题目会告知这 3 个数字,但是不会按照排序后的顺序直接给出(可能会乱序),而是另外给一个顺序,要求按照给定的顺序重新输出它们。例如,当输入是 “6 4 2 C A B” 时,即 $A=2$, $B=4$, $C=6$,要求按照 C 、 A 、 B 的顺序输出,因此应当输出 “6 2 4”。