

第4章 选择结构程序设计

1. 什么是算术运算? 什么是关系运算? 什么是逻辑运算?

解: 略。

2. C语言中如何表示“真”和“假”? 系统如何判断一个量的“真”和“假”?

解: 对于逻辑表达式, 若其值为“真”, 则以 1 表示, 若其值为“假”, 则以 0 表示。但是在判断一个逻辑量的值时, 系统会以 0 作为“假”, 以非 0 作为“真”。例如 $3 \&\& 5$ 的值为“真”, 系统给出 $3 \&\& 5$ 的值为 1。

3. 写出下面各逻辑表达式的值。设 $a=3, b=4, c=5$ 。

(1) $a+b>c \&\& b==c$

(2) $a||b+c \&\& b-c$

(3) $!(a>b) \&\& !c||1$

(4) $!(x=a) \&\& (y=b) \&\& 0$

(5) $!(a+b)+c-1 \&\& b+c/2$

解:

(1) 0

(2) 1

(3) 1

(4) 0

(5) 1

4. 有 3 个整数 a, b, c , 由键盘输入, 输出其中最大的数。

解:

方法一: N-S 图见图 4.1。

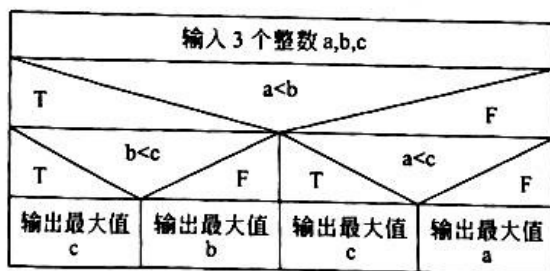


图 4.1

程序如下:

```
#include <stdio.h>
int main()
{
```

```

int a,b,c;
printf("请输入3个整数:");
scanf("%d,%d,%d",&a,&b,&c);
if (a<b)
    if (b<c)
        printf("max=%d\n",c);
    else
        printf("max=%d\n",b);
else if (a<c)
    printf("max=%d\n",c);
else
    printf("max=%d\n",a);
return 0;
}

```

运行结果:

```

请输入3个整数:12,34,9
max=34

```

方法二: 使用条件表达式,可以使程序更简明、清晰。

```

#include <stdio.h>
int main()
{
    int a,b,c,temp,max;
    printf("请输入3个整数:");
    scanf("%d,%d,%d",&a,&b,&c);
    temp=(a>b)? a:b; //将a和b中的大者存入temp中
    max=(temp>c)? temp:c; //将a和b中的大者与c比较,取最大者
    printf("3个整数的最大数是%d\n",max);
    return 0;
}

```

运行结果:

```

请输入3个整数:12,34,9
3个整数的最大数是34

```

5. 从键盘输入一个小于1000的正数,要求输出它的平方根(如平方根不是整数,则输出其整数部分)。要求在输入数据后先对其进行检查是否为小于1000的正数。若不是,则要求重新输入。

解:

```

#include <stdio.h>
#include <math.h>
#define M 1000
int main()
{
    int i,k;

```



```

printf("请输入一个小于%d的整数 i:", M);
scanf("%d", &i);
if (i > M)
{ printf("输入的数据不符合要求, 请重新输入一个小于%d的整数 i:", M);
  scanf("%d", &i);
}
k = sqrt(i);
printf("%d 的平方根的整数部分是 %d\n", i, k);
return 0;
}

```

运行结果:

① 第一次: 输入正确数据。

```

请输入一个小于1000的整数 i: 345
345 的平方根的整数部分是: 18

```

② 第二次: 输入不正确数据。

```

请输入一个小于1000的整数 i: 1230
输入的数据不符合要求, 请重新输入一个小于1000的整数 i: 130
130 的平方根的整数部分是: 11

```

讨论: 题目要求输入的数小于 1000, 今为了增加程序的灵活性, 定义符号常量 M 为 1000, 如果题目要求输入的数小于 10000, 只须修改 `define` 指令即可, 不必修改主函数。

用 `if` 语句检查输入的数是否符合要求, 如果不符合要求应进行相应的处理。从上面的程序看来是很简单的, 但在实际应用中是很有用的。因为在程序提供用户使用后, 不能保证用户输入的数据都是符合要求的。假若用户输入了不符合要求的数据怎么办? 如果没有检查和补救措施, 程序是不能供实际使用的。

本程序的处理方法是: 提醒用户“输入的数据错了”, 要求重新输入。但只提醒一次, 再错了怎么办? 在学习了第 5 章循环之后, 可以将程序改为多次检查, 直到正确输入为止。程序如下:

```

#include <stdio.h>
#include <math.h>
#define M 1000
int main()
{
    int i, k;
    printf("请输入一个小于%d的整数 i:", M);
    scanf("%d", &i);
    while (i > M)
    { printf("输入的数据不符合要求, 请重新输入一个小于%d的整数 i:", M);
      scanf("%d", &i);
      k = sqrt(i);
    }
}

```



知否大学
— 微信公众号同名 —


```
printf("%d 的平方根的整数部分是%d\n", i, k);
return 0;
}
```

运行结果:

```
请输入一个小于1000的整数i:1230
输入的数不符合要求,请重新输入一个小于1000的整数i:1245
输入的数不符合要求,请重新输入一个小于1000的整数i:654
654的平方根的整数部分是: 25
```

多次输入不符合要求的数据,均通不过,直到输入符合要求的数据为止。

这种检查手段是很重要的,希望读者能真正掌握。本例只是示意性的,程序比较简单。有了此基础,读者根据此思路完全可以做到对任何条件进行检查处理,使程序能正常运行,万无一失。

6. 有一个函数:

$$y = \begin{cases} x & (x < 1) \\ 2x - 1 & (1 \leq x < 10) \\ 3x - 11 & (x \geq 10) \end{cases}$$

写程序,输入 x 的值,输出 y 相应的值。

解:

```
#include <stdio.h>
int main( )
{
    int x, y;
    printf("输入 x:");
    scanf("%d", &x);
    if(x < 1) // x < 1
    {
        y = x;
        printf("x = %3d, y = x = %d\n", x, y);
    }
    else if(x < 10) // 1 ≤ x < 10
    {
        y = 2 * x - 1;
        printf("x = %d, y = 2 * x - 1 = %d\n", x, y);
    }
    else // x ≥ 10
    {
        y = 3 * x - 11;
        printf("x = %d, y = 3 * x - 11 = %d\n", x, y);
    }
    return 0;
}
```

运行结果:

①

```
输入x:4
x=4, y=2*x-1=7
```



知否大学
— 微信公众号同名 —

②

输入x:-1
x=-1, y=x=-1

③

输入x:28
x=28, y=3*x-11=49

7. 有一函数:

$$y = \begin{cases} -1 & (x < 0) \\ 0 & (x = 0) \\ 1 & (x > 0) \end{cases}$$

有人分别编写了以下两个程序,请分析它们是否能实现题目要求。不要急于上机运行程序,先分析上面两个程序的逻辑,画出它们的流程图,分析它们的运行情况。然后上机运行程序,观察并分析结果。

(1)

```
#include <stdio.h>
int main( )
{
    int x,y;
    printf("enter x:");
    scanf("%d",&x);
    y=-1;
    if(x!=0)
        if(x>0)
            y=1;
    else
        y=0;
    printf("x=%d,y=%d\n",x,y);
    return 0;
}
```

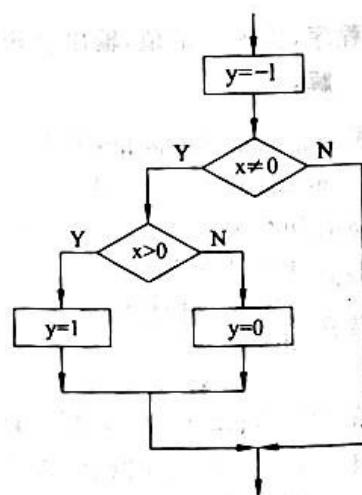


图 4.2

解: 程序(1)的流程图见图 4.2。

它不能实现题目的要求。如果输入的 $x < 0$, 则输出 $y = 0$ 。请注意 else 与 if 的配对关系。程序(1)中的 else 子句是和第 9 行的内嵌的 if 语句配对, 而不与第 8 行的 if 语句配对。运行结果:

enter x:-6
x=-6, y=0

x 的值为 -6, 输出 $y = 0$, 结果显然不对。

(2)

```
#include <stdio.h>
int main( )
{
```



知否大学
- 微信公众号同名 -

```

int x,y;
printf("enter x:");
scanf("%d",&x);
y=0;
if(x>=0)
    if(x>0) y=1;
    else y=-1;
printf("x=%d,y=%d\n",x,y);
return 0;
}

```

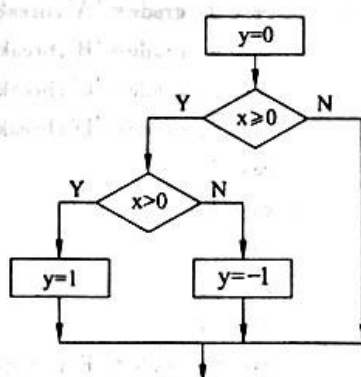


图 4.3

解：程序(2)的流程图见图 4.3。

它也不能实现题目的要求。如果输入的 $x < 0$ ，则输出 $y = 0$ 。

运行结果：

```

please enter x:-4
x=-4,y=0

```

x 的值为 -4 ，输出 $y = 0$ ，结果显然不对。程序(2)中的 `else` 子句是和第 9 行的内嵌的 `if` 语句配对，而不与第 8 行的 `if` 语句配对。

一定要注意 `if` 与 `else` 的配对关系。配对关系不随 `if` 和 `else` 所出现的列的位置而改变，例如程序(2)中的 `else` 与第 8 行的 `if` 写在同一列，但 `else` 并不因此而与第 8 行的 `if` 语句配对，它只和在它前面的离它最近的 `if` 配对。

请和教材第 4 章例 4.5 程序对比分析，进一步理解 `if-else` 的配对规则。

为了使逻辑关系清晰，避免出错，一般把内嵌的 `if` 语句放在外层的 `else` 子句中（如例 4.5 中程序 1 那样），这样由于有外层的 `else` 相隔，内嵌的 `else` 不会被误认为和外层的 `if` 配对，而只能与内嵌的 `if` 配对，这样就不会搞混，若像本习题的程序(1)和程序(2)那样写就很容易出错。

可与本章例 4.5 中介绍的程序进行对比分析。

8. 给出一百分制成绩，要求输出成绩等级 'A'、'B'、'C'、'D'、'E'。90 分以上为 'A'，80~89 分为 'B'，70~79 分为 'C'，60~69 分为 'D'，60 分以下为 'E'。

解：

```

#include <stdio.h>
int main( )
{
    float score;
    char grade;
    printf("请输入学生成绩:");
    scanf("%f",&score);
    while (score>100||score<0)
    {
        printf("\n 输入有误,请重输");
        scanf("%f",&score);
    }
    switch((int)(score/10))
    {
        case 10:

```



找课后习题答案
下载「知否大学」APP


```

    case 9: grade='A'; break;
    case 8: grade='B'; break;
    case 7: grade='C'; break;
    case 6: grade='D'; break;
    case 5:
    case 4:
    case 3:
    case 2:
    case 1:
    case 0: grade='E';
    }
    printf("成绩是 %5.1f, 相应的等级是%c\n", score, grade);
    return 0;
}

```


运行结果:

①

请输入学生成绩: 98.5
成绩是 98.5, 相应的等级是A

②

请输入学生成绩: 58
成绩是 58.0, 相应的等级是E

 说明: 对输入的数据进行检查, 如小于 0 或大于 100, 要求重新输入。\$(int)(score/10)\$ 的作用是将 \$(score/10)\$ 的值进行强制类型转换, 得到一个整型值。例如, 当 \$score\$ 的值为 78 时, \$(int)(score/10)\$ 的值为 7。然后在 \$switch\$ 语句中执行 \$case\ 7\$ 中的语句, 使 \$grade='C'\$。

9. 给一个不多于 5 位的正整数, 要求:

- ① 求出它是几位数;
- ② 分别输出每一位数字;
- ③ 按逆序输出各位数字, 例如原数为 321, 应输出 123。

解:

```

#include <stdio.h>
#include <math.h>
int main()
{
    int num, indiv, ten, hundred, thousand, ten_thousand, place;
    //分别代表个位、十位、百位、千位、万位和位数

    printf("请输入一个整数(0~99999):");
    scanf("%d", &num);
    if (num > 9999)
        place = 5;
    else if (num > 999)
        place = 4;
}

```



```

else if (num>99)
    place=3;
else if (num>9)
    place=2;
else place=1;
printf("位数:%d\n",place);
printf("每位数字为:");
ten_thousand=num/10000;
thousand=(int)(num-ten_thousand*10000)/1000;
hundred=(int)(num-ten_thousand*10000-thousand*1000)/100;
ten=(int)(num-ten_thousand*10000-thousand*1000-hundred*100)/10;
indiv=(int)(num-ten_thousand*10000-thousand*1000-hundred*100-ten*10);
switch(place)
{
    case 5: printf("%d,%d,%d,%d,%d",ten_thousand,thousand,hundred,ten,indiv);
            printf("\n反序数字为:");
            printf("%d%d%d%d%d\n",indiv,ten,hundred,thousand,ten_thousand);
            break;
    case 4: printf("%d,%d,%d,%d",thousand,hundred,ten,indiv);
            printf("\n反序数字为:");
            printf("%d%d%d%d\n",indiv,ten,hundred,thousand);
            break;
    case 3: printf("%d,%d,%d",hundred,ten,indiv);
            printf("\n反序数字为:");
            printf("%d%d%d\n",indiv,ten,hundred);
            break;
    case 2: printf("%d,%d",ten,indiv);
            printf("\n反序数字为:");
            printf("%d%d\n",indiv,ten);
            break;
    case 1: printf("%d",indiv);
            printf("\n反序数字为:");
            printf("%d\n",indiv);
            break;
}
return 0;
}

```

运行结果:

```

请输入一个整数(0~99999):98423
位数:5
每位数字为:9,8,4,2,3
反序数字为:32489

```

10. 企业发放的奖金根据利润提成。利润 I 低于或等于 100 000 元的,奖金可提成 10%; 利润高于 100 000 元,低于 200 000 元 ($100\,000 < I \leq 200\,000$) 时,低于 100 000 元的部分按 10% 提成,高于 100 000 元的部分,可提成 7.5%; $200\,000 < I \leq 400\,000$ 时,低于 200 000 元的部分仍

按上述办法提成(下同)。高于 200 000 元的部分按 5% 提成; $400\,000 < I \leq 600\,000$ 元时, 高于 400 000 元的部分按 3% 提成; $600\,000 < I \leq 1\,000\,000$ 时, 高于 600 000 元的部分按 1.5% 提成; $I > 1\,000\,000$ 时, 超过 1 000 000 元的部分按 1% 提成。从键盘输入当月利润 I , 求应发奖金总数。

要求:

- (1) 用 if 语句编程序。
- (2) 用 switch 语句编程序。

解:

- (1) 用 if 语句编程序。

```
#include <stdio.h>
int main( )
{
    int i;
    double bonus, bon1, bon2, bon4, bon6, bon10;
    bon1 = 100000 * 0.1;
    bon2 = bon1 + 100000 * 0.075;
    bon4 = bon2 + 100000 * 0.05;
    bon6 = bon4 + 100000 * 0.03;
    bon10 = bon6 + 400000 * 0.015;
    printf("请输入利润 i:");
    scanf("%d", &i);
    if (i <= 100000)
        bonus = i * 0.1;
    else if (i <= 200000)
        bonus = bon1 + (i - 100000) * 0.075;
    else if (i <= 400000)
        bonus = bon2 + (i - 200000) * 0.05;
    else if (i <= 600000)
        bonus = bon4 + (i - 400000) * 0.03;
    else if (i <= 1000000)
        bonus = bon6 + (i - 600000) * 0.015;
    else
        bonus = bon10 + (i - 1000000) * 0.01;
    printf("奖金是: %10.2f\n", bonus);
    return 0;
}
```

运行结果:

```
请输入利润 i: 234000
奖金是: 19200.00
```

此题的关键在于正确写出每一区间的奖金计算公式。例如利润在 100 000~200 000 元时, 奖金应由两部分组成:

- ① 利润为 100 000 元时应得的奖金, 即 $100\,000 \text{ 元} \times 0.1$ 。

② 100 000 元以上部分应得的奖金, 即 $(\text{num}-100\,000) \times 0.075$ 元。

同理, 200 000~400 000 元这个区间的奖金也应由两部分组成:

① 利润为 200 000 元时应得的奖金, 即 $100\,000 \times 0.1 + 100\,000 \times 0.075$ 。

② 200 000 元以上部分应得的奖金, 即 $(\text{num}-200\,000) \times 0.05$ 元。

程序中先把 100 000 元、200 000 元、400 000 元、600 000 元、1 000 000 元各关键点的奖金计算出来, 即 bon1 , bon2 , bon4 , bon6 和 bon10 。然后再加上各区间附加部分的奖金即可。

(2) 用 switch 语句编程。

N-S 图见图 4.4。

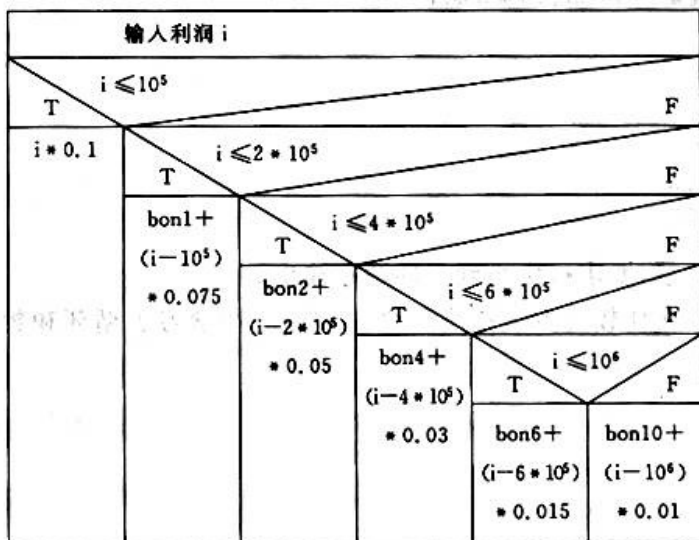


图 4.4

```
#include <stdio.h>
int main( )
{
    int i;
    double bonus, bon1, bon2, bon4, bon6, bon10;
    int branch;
    bon1 = 100000 * 0.1;
    bon2 = bon1 + 100000 * 0.075;
    bon4 = bon2 + 200000 * 0.05;
    bon6 = bon4 + 200000 * 0.03;
    bon10 = bon6 + 400000 * 0.015;
    printf("请输入利润 i:");
    scanf("%d", &i);
    branch = i / 100000;
    if (branch > 10) branch = 10;
    switch(branch)
    {
        case 0: bonus = i * 0.1; break;
        case 1: bonus = bon1 + (i - 100000) * 0.075; break;
```



找课后习题答案



知否大学 APP


```

    case 2:
    case 3: bonus = bon2 + (i - 200000) * 0.05; break;
    case 4:
    case 5: bonus = bon4 + (i - 400000) * 0.03; break;
    case 6:
    case 7:
    case 8:
    case 9: bonus = bon6 + (i - 600000) * 0.015; break;
    case 10: bonus = bon10 + (i - 1000000) * 0.01;
    }
    printf("奖金是 %10.2f\n", bonus);
    return 0;
}

```

运行结果:

```

请输入利润 i: 156890
奖金是 14266.75

```

11. 输入 4 个整数, 要求按由小到大的顺序输出。

解: 此题采用依次比较的方法排出其大小顺序。在学习了循环和数组以后, 可以掌握更多的排序方法。

程序如下:

```

#include <stdio.h>
int main()
{
    int t, a, b, c, d;
    printf("请输入 4 个数:");
    scanf("%d, %d, %d, %d", &a, &b, &c, &d);
    printf("a = %d, b = %d, c = %d, d = %d\n", a, b, c, d);
    if (a > b)
        { t = a; a = b; b = t; }
    if (a > c)
        { t = a; a = c; c = t; }
    if (a > d)
        { t = a; a = d; d = t; }
    if (b > c)
        { t = b; b = c; c = t; }
    if (b > d)
        { t = b; b = d; d = t; }
    if (c > d)
        { t = c; c = d; d = t; }
    printf("排序结果如下: \n");
    printf("%d %d %d %d\n", a, b, c, d);
    return 0;
}

```


运行结果:

```
请输入4个数:6,8,1,4
a=6,b=8,c=1,d=4
排序结果如下:
1 4 6 8
```

12. 有4个圆塔,圆心分别为 $(2,2)$ 、 $(-2,2)$ 、 $(-2,-2)$ 、 $(2,-2)$,圆半径为1,见图4.5。这4个塔的高度为10m,塔以外无建筑物。今输入任一点的坐标,求该点的建筑高度(塔外的高度为零)。

解: N-S图见图4.6。

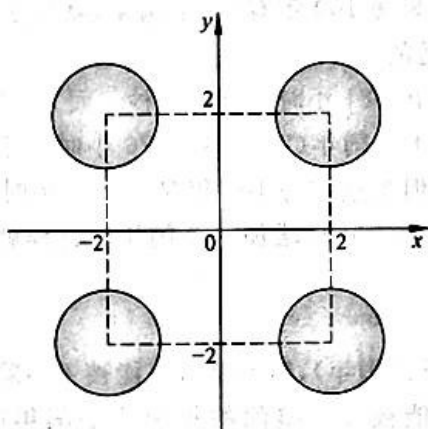


图 4.5

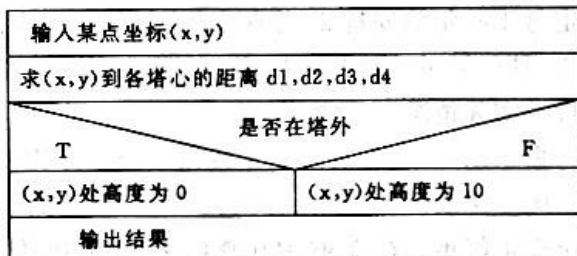


图 4.6

程序如下:

```
#include <stdio.h>
int main( )
{
    int h=10;
    float x1=2,y1=2,x2=-2,y2=2,x3=-2,y3=-2,x4=2,y4=-2,x,y,d1,d2,d3,d4;
    printf("请输入一个点(x,y):");
    scanf("%f,%f",&x,&y);
    d1=(x-x4)*(x-x4)+(y-y4)*(y-y4); //求该点到各中心点距离
    d2=(x-x1)*(x-x1)+(y-y1)*(y-y1);
    d3=(x-x2)*(x-x2)+(y-y2)*(y-y2);
    d4=(x-x3)*(x-x3)+(y-y3)*(y-y3);
    if (d1>1 && d2>1 && d3>1 && d4>1) h=0; //判断该点是否在塔外
    printf("该点高度为 %d\n",h);
    return 0;
}
```

运行结果:

①

```
请输入一个点(x,y):0.5,0.7
该点高度为 0
```

②

请输入一个点(x,y):2.1,2.3
该点高度为 10

关于闰年问题的说明:

在教材第 4 章中举了计算闰年的例子,有的读者对闰年规则不清楚,纷纷来信询问。因此,有必要在此对闰年的规定作一些说明:

地球绕太阳转一周的实际时间为 365 天 5 小时 48 分 46 秒。如果一年只有 365 天,每年就多出 5 个多小时。4 年多出的 23 小时 15 分 4 秒,差不多等于一天。于是决定每 4 年增加 1 天。但是,它比一天 24 小时又少了约 45 分钟。如果每 100 年有 25 个闰年,就少了 18 时 43 分 20 秒,这就差不多等于一天了,这显然是不合适的。

可以算出,每年多出 5 小时 48 分 46 秒,100 年就多出 581 小时 16 分 40 秒。而 25 个闰年需要 $25 \times 24 = 600$ 小时。581 小时 16 分 40 秒只够 24 个闰年($24 \times 24 = 576$ 小时),于是决定每 100 年只安排 24 个闰年(世纪年不作为闰年)。但是这样每 100 年又多出 5 小时 16 分 40 秒($581 \text{ 小时 } 16 \text{ 分 } 40 \text{ 秒} - 576 \text{ 小时}$),于是又决定每 400 年增加一个闰年。这样就比较接近实际情况了。

根据以上情况,决定闰年按以下规则计算:

闰年应能被 4 整除(如 2004 年是闰年,而 2001 年不是闰年),但不是所有能被 4 整除的年份都是闰年。在能被 100 整除的年份中,只有同时能被 400 整除的年份才是闰年(如 2000 年是闰年),能被 100 整除而不能被 400 整除的年份(如 1800、1900、2100)不是闰年。这是国际公认的规则。只说“能被 4 整除的年份是闰年”是不准确的。

教材中介绍的方法和程序是正确的。