第3章　口令验证模块的开发

本章概述：

本章讲授了关系运算符和关系表达式，并在分支语句中熟练使用关系表达式进行分支结构的程序设计。学习本章后，读者应能够熟练运用分支语句来解决问题。

教学重点：

- ●关系运算符和关系表达式
- ●逻辑运算
- ●双分支语句
- ●多分支语句

教学难点：

- ●逻辑运算
- ●双分支语句
- ●多分支语句

学习建议：

理解双分支和多分支语句这两种语句的语法及逻辑含义，并多做练习。

在很多应用程序中，为了提高程序使用的安全性，常常会在进入系统之前提供一个口令验证的功能，以保障只有系统的合法用户才能进入。我们不妨也给计算器加上一个这样的功能。

3.1　任务说明

任务描述：编写是 login 函数，实现计算器的密码验证功能。

任务要求：

(1)在进入计算器主菜单前，加入一个输入密码的提示。

(2)根据提示输入一个整数密码。

(3)如果密码输入正确，则允许进入计算器主菜单，如果输入不正确提示“密码输入错误”，不显示主菜单。

3.2　任务分析

☞ 开发思路

口令验证模块的开发思路如下：

(1)输入一个口令。

(2)判断该口令是否正确，如果正确，显示菜单部分，如果不正确，提示不正确。

要想实现以上步骤，那么我们要考虑到下面两个问题：首先，如何判断输入的口令与所预设的口令(比如 123)是否匹配。然后，根据口令的匹配情况给出相应的用户回馈信息。

☞ 新知识

下面就从这两个问题入手，来逐一地进行解决。

1.关系运算

假设预设的口令是“123”，那么如何判断输入的口令是否等于 123 呢？这里需要使用关系运算符来判断二者之间的关系。在关系运算符中，判断两个量是否相等的运算符是“＝＝”，这点与我们数学中的表示方法不太一样。数学中，我们用一个等号来表示两侧相等的关系，而在 C 语言中，由于一个等号是表示赋值运算的，为了与之区别，使用双等号(＝＝)表示两个表达式相等的关系。

那么判断口令是否为 123 的表达式为：“password＝＝123”。

上面由关系运算符构成的表达式就是关系表达式。关系表达式的运算结果有两种，一种是“真”，一种为“假”。当 password 与 123 相等时，结果为真；反之为假。在 C 语言中，“真”用“1”

来表示，“假”用“0”来表示。另外要注意一点，所有非 0 的值都是“真”的。

2.逻辑运算

假设预设密码有两个：123 和 111，那么单单使用上面的关系运算符是不够的，还需要使用逻辑表达式。判断口令是否与以上预设密码中的一个相匹配，其实就是要判断“password＝＝123”或者“password＝＝111”，在 C 语言中，或者的关系使用逻辑运算符“‖”来表示。

判断两个密码匹配的表达式为：“password＝＝123 ‖ password＝＝111”。

3.分支语句

上面我们使用关系运算和逻辑运算解决了密码的匹配判断问题。接下来，需要解决的问题就是在匹配和不匹配两种情形下，给出用户相应的反馈。如密码不对，提示 “密码不正确，不能使用本系统”，如密码正确，提示“密码正确，欢迎使用本系统”的字样。

这种决策功能在 C 语言中需要使用分支语句来完成。这里我们使用最常用的 if-else 引导的分支语句来解决。使用分支语句完成密码反馈的语句为：

```
if(password＝＝123 ‖ password＝＝111)
    printf("密码正确，欢迎使用本系统！");
else
    printf("密码不正确，不能使用本系统！");
```

3.3　任务实施

☞ 算法设计

根据上面的分析，我们给出该模块的 N-S 图如图 3-1 所示。

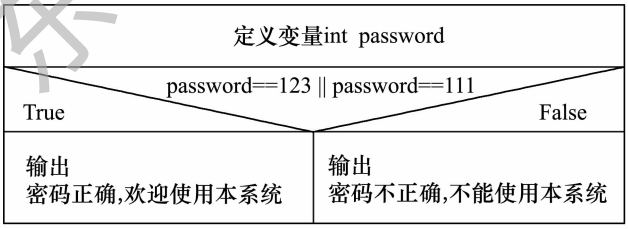


图 3-1　密码验证模块的 N-S 图

☞ 算法实现

calculator.c 中的 login 函数

```
/＊口令验证模块的代码＊/
＃include＜stdio.h＞                    //包含所需要的预编译头文件 stdio.h
void login();                           //声明口令验证函数
int main()                              //主函数
```

```
{
    login();                                    //调用口令验证函数
    return 0;
}
void login()                                    //定义(编写)口令验证函数
{
    int password;                               //定义个整型变量,表示用户输入口令
    printf ("请输入口令:");                      //提示输入的语句
    scanf ("%d",&password);                     //输入用户口令
    if (password==123 || password==111)         //使用 if-else 语句判断口令是否正确
        printf("密码正确,欢迎使用本系统!");      //口令正确时的输出
    else
        printf("密码不正确,不能使用本系统!");    //口令不正确时的输出
}
```

☞ **程序运行**

```
请输入口令:123
密码正确,欢迎使用本系统!
```

3.4 知识点详解

3.4.1 关系运算符和关系表达式

1.关系运算符

关系运算符都是双目运算符,其功能是用来对两个操作数的大小进行比较。C 语言提供了 6 种关系运算符,其运算规则如表 3-1 所示。

表 3-1　　C 语言中的关系运算符

运算符	意义	举例
<	小于	a < b
<=	小于或等于	c <= 5
>	大于	b > c
>=	大于或等于	b >=0
==	等于	c==b
! =	不等于	c ! =10

在这 6 个运算符中,前四个优先级相同,后两个优先级相同。前四个运算符的优先级高于后两个运算符的优先级。

2.关系表达式

关系运算符组成的关系表达式的值是逻辑值,即“真”或“假”。例如,a > 5 的值要么是真,

要么是假，取决于 a 的值。如果 a 值为 8，则 a ＞ 5 为真。在 C 语言中没有逻辑类型的量，规定“真”用 1 表示，“假”用 0 表示。于是，a ＞ 5 值为 1。这里的 1 就是数字 1。例如，表达式“(a ＞ 5)＋2”是合法的，其值为 3。

这是 C 语言不同于其他语言之处。

关系运算符常用来组成关系表达式作为某些语句的条件，故称条件表达式。

关于等于运算符“＝＝”是由两个代数式中的等号组成的，有时容易写成一个等号与代数式中的等号“＝”相混。在 C 语言中，一个等号是赋值运算符，它与等于运算符截然不同，请一定注意区别。

3.4.2　逻辑运算

逻辑运算符是用来对操作数进行逻辑操作的。C 语言提供了如下的逻辑运算符。

单目的逻辑运算符：“!”表示逻辑求反或逻辑非，如，!(a＋b)，! q 和! 9 等。

双目的逻辑运算符：“&&”表示逻辑与，即对两个操作数进行逻辑求与。例如，a&&b 、3&&0 和 7&&a 等。“‖”表示逻辑或，即对两个操作数进行逻辑求或。例如，(a＋b)‖ c ，6‖(c＋d)和 5‖7 等。

逻辑求反是对真求反后为假，对假求反后为真。

逻辑求与是指两个操作数只有都是真时，求与后才是真。否则，求与后为假。

逻辑求或是指两个操作数中只要有一个为真，求或后就为真。只有两个都是假时，求或后才是假。

操作数的真与假是这样规定的：非零为“真”，零为“假”。

逻辑运算结果的真与假是这样规定的：“真”用 1 表示，“假”用 0 表示。

例如：

int a＝5，b＝0；

! a 的值为 0，因为 a 为真，则! a 为假。

a&&b 的值为 0，因为 b 为假，则 a&&b 为假。

a‖b 的值为 1，因为 a 为真，则 a‖b 为真。

! a‖b 的值为 0，因为! a 和 b 都为假，则! a‖b 为假。

3.4.3　程序的三种基本结构

通常的计算机程序总是由若干条语句组成。从执行方式上看，从第一条语句到最后一条语句完全按顺序执行，是简单的顺序结构。若在程序执行过程当中，根据用户的输入或中间结果去执行若干不同的任务则为选择(或分支)结构。如果在程序的某处，需要根据某项条件重复地执行某项任务若干次或直到满足或不满足某条件为止，这就构成循环结构。大多数情况下，程序都不会是简单的顺序结构，而是顺序、选择(或分支)、循环三种结构的复杂组合。

C 语言中，有一组相关的控制语句，用以实现选择(或分支)结构与循环结构。

选择(或分支)控制语句：if，switch－case。

循环控制语句：for，while，do-while。

转移控制语句：break，continue，goto。

3.4.4　单分支 if 语句

在程序的三种基本结构中，第二种即为选择结构，其基本特点是：程序的流程由多路分支组成，在程序的一次执行过程中，根据不同的情况，只有一条支路被选中执行，而其他分支上的语句被直接跳过。

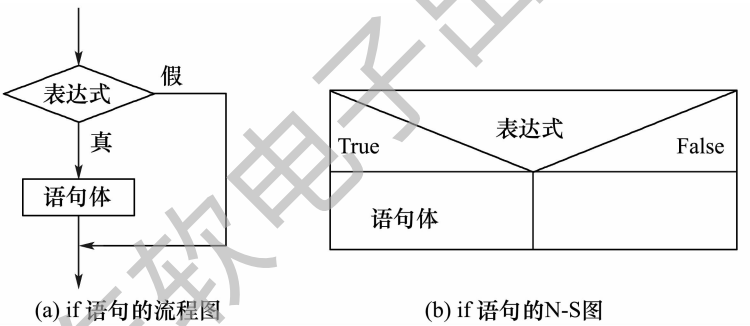
C 语言中，提供了由 if 语句和 switch 语句引导的选择结构，if 语句用于两者选一的情况，而 switch 用于多分支选一的情形。

单分支语句，一般来说就是根据用户设置的条件表达式的值，决定某一操作是否执行。单分支语句就相当于我们常说的“如果……就（那么）……”。C 语言为我们提供了 if 语句来实现单分支结构。该语句的语法格式为：

```
if(条件表达式)
{
语句体;
}
```

其语义为：如果条件表达式的值为真（非 0），则执行语句体；否则跳过语句体继续执行其后面的语句。语句体可以包括零条、一条或多条语句。当语句体为一条语句时，花括号可以省略。

if 语句的执行过程流程图和 N-S 图如图 3-2 所示。



(a) if 语句的流程图　　(b) if 语句的N-S图

图 3-2

【例 3-1】　从键盘输入一个整数，判断该数是否为偶数。

☞ 编程点拨

解决这个问题大致需要三个步骤：

(1)输入一个整数 number，保存在一个整型变量中。

(2)判断 number 是否为偶数；偶数即为能被 2 整除的数，换句话说就是与 2 相除余数为 0 的数。我们可以用前面学过的算数运算符“%”和关系运算符“==”表示，即为：“number%2==0”。

(3)输出判断结果。

根据以上分析，可以用如图 3-3 所示的 N-S 图来表示。

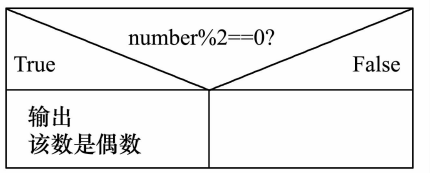


图 3-3　判断一个数是否为偶数的 N-S 图

程序清单 3-1 even.c

```
/*判断一个数是否为偶数*/
#include<stdio.h>                    //包含所需要的预编译头文件 stdio.h
int main( )                          //主函数
{
    int number;                      //定义一个整数 number
    printf("请输入一个整数:");        //输入提示
    scanf("%d",&number);             //输入一个整数
    if(number%2==0)                  //判断该数是否为偶数
        printf("该数是偶数\n");       //满足偶数条件输出"该数是偶数"
    return 0;
}
```

☞ **程序运行**

```
请输入一个整数:6
该数是偶数
```

3.4.5　双分支 if-else 语句

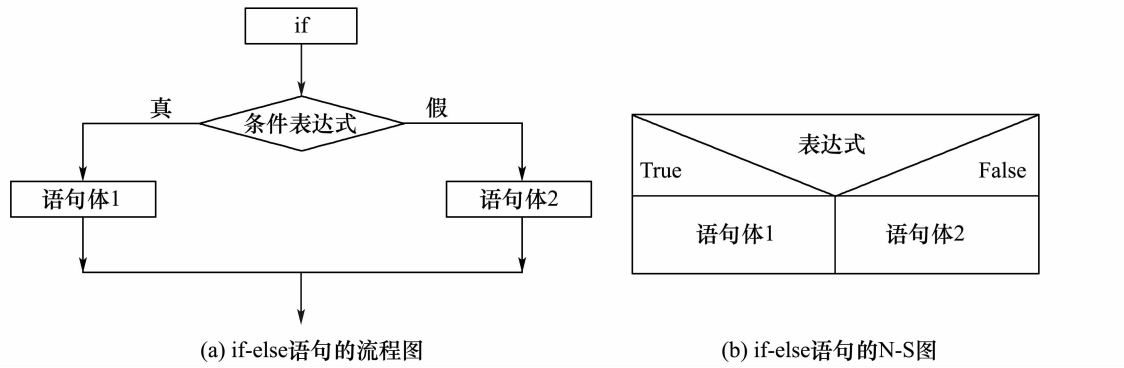
双分支语句相当于"如果……就(那么)……否则……"。例如:如果是休息日,我就出游;否则我就要去上课。在 C 语言中用 if-else 语句来实现双分支结构,也就是根据用户设置的条件表达式的值,选择两个操作中的一个来执行。

双分支语句的语法格式为:

```
if(条件表达式)
{
    语句体 1;
}
else
{
    语句体 2;
}
```

其语义为:如果表达式的值为真(非 0)时,则执行语句体 1;否则执行语句体 2。

if-else 语句的流程图和 N-S 图如图 3-4 所示。



(a) if-else语句的流程图　　(b) if-else语句的N-S图

图 3-4

【例 3-2】 从键盘输入一个百分制的学生分数，判断该分是否及格。

☞ 编程点拨

解决这个问题大致需要三个步骤：

(1)输入一个学生的成绩 score。

(2)判断该分数是否大于或等于 60 分，表达式为“score >= 60”。

(3)输出判断结果。

根据以上分析，可以用如图 3-5 所示的 N-S 图来表示。

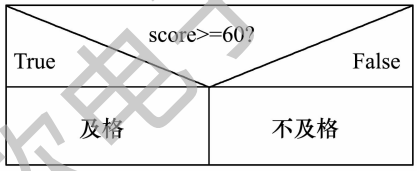


图 3-5 判断一个数是否为整数的 N-S 图

程序清单 3-2 score.c

```
/*判断一个成绩是否及格*/
#include<stdio.h>                    //包含所需要的预编译头文件 stdio.h
int main( )                          //主函数
{
    int score;                       //定义一个学生分数 score
    printf("请输入一个百分制分数:");    //输入提示
    scanf("%d",&score);              //输入 score 的值
    if (score >= 60 && score <= 100) //判断该分数是否及格
        printf("及格!\n");           //满足及格条件输出“及格”
    else
        printf("不及格!\n");         //不满足及格条件输出“不及格”
    return 0;
}
```

☞ **程序运行**

请输入一个百分制分数:60

及格!

3.4.6　多分支 if-else if 语句

单/双分支语句只能用来表示一种或两种选择的情况,而在实际应用中经常遇到多种选择的情况。在 C 语言中用 if-else if 语句、if-else 嵌套和 switch 语句来实现多分支选择的情况。用 if else if 语句可以实现多分支选择,具体语句格式如下:

```
if (表达式 1) {语句体 1;}
else if (表达式 2) {语句体 2;}
else if (表达式 3) {语句体 3;}
...
else if (表达式 n) {语句体 n;}
else {语句体 n+1;}
```

其语义为:如果表达式 1 的值为真(非 0)时,则执行语句体 1;否则如果表达式 2 的值为真(非 0)时,则执行语句体 2;否则如果表达式 3 的值为真(非 0)时,则执行语句体 3;……;否则条件表达式 n 的值为真(非 0)时,则执行语句体 n;否则执行语句体 n+1。

多分支语句的执行过程的流程图和 N-S 图如图 3-6 所示。

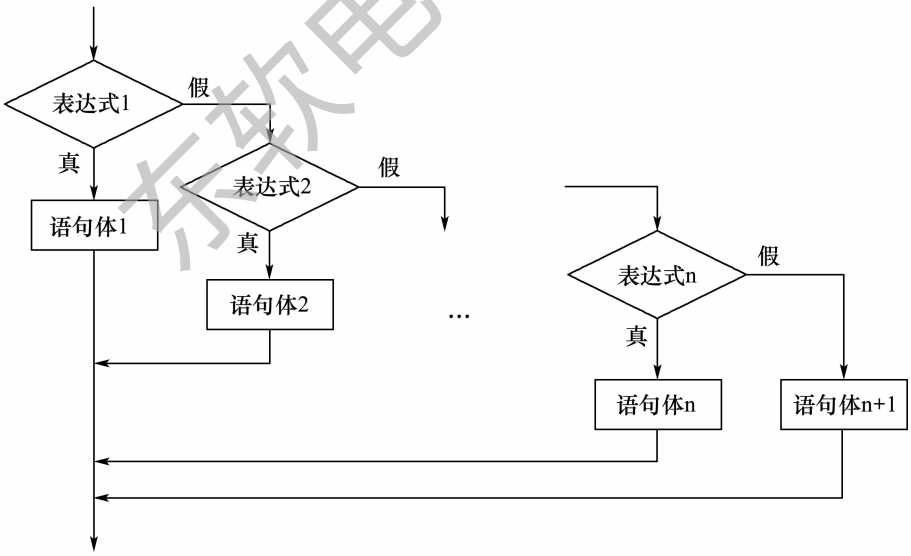


图 3-6(a)　多分支语句的流程图

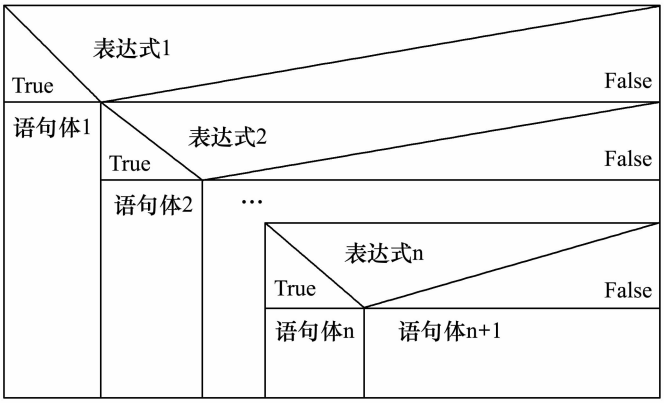


图 3-6(b) 多分支语句的 N-S 图

这种 if 语句在执行时，依次判断各条件表达式的值，当出现某个条件表达式的值为真时，则执行其对应的语句体，然后跳到整个 if 语句之后继续执行程序。如果所有的表达式均为假，则执行语句体 n＋1。然后继续执行整个 if 语句后面的语句。

【例 3-3】 从键盘输入一个百分制的学生分数，判断该分数的等级。分数与对应的等级如表 3-2 所示。

表 3-2 分数与等级对应表

分数	等级
100～90	A
89～80	B
79～70	C
69～60	D
60 以下	E

☞ 编程点拨

解决这个问题大致需要三个步骤：

(1)输入一个学生的成绩 score。

(2)逐步判断该分数的等级，若分数在 90 分以上(含 90 分)，则为“A”级；否则，若成绩高于或等于 80 分，等为“B”级；否则，若分数在高于或等于 70 分，则等级为“C”级；否则，若分数高于或等于 60 分，则等级为“D”级，如果不满足以上条件，则分数等级为“E” 级。用 if-else if 多分支语句表示为：

```
if(score>=90)
    printf("等级为 A! \n");
else if(score>=80)
    printf("等级为 B! \n");
else if(score>=70)
    printf("等级为 C! \n");
```

```
else if(score>=60)
    printf("等级为 D! \n");
else
    printf("等级为 E! \n");
```

注意：

在判断等级 B 时，我们使用了表达式“score >= 80”来判断，为什么不使用“if (score >= 80 && score < 90)”呢？是因为在上一个判断条件“if (score >= 90)”中，我们已经将所有 90 分以上的情形筛选出去了，所以这里面就不需要进行额外的附加小于 90 分的条件了，以下的条件判断以此类推。

(3)输出判断结果。

根据以上分析，可以用如图 3-7 所示的 N-S 图来表示。

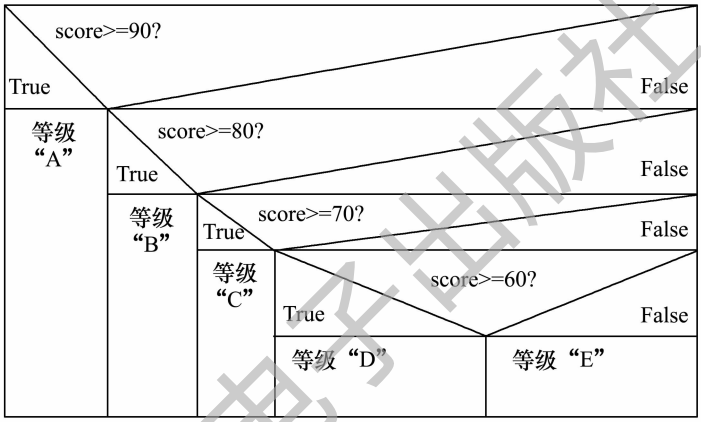


图 3-7　判断成绩等级的 N-S 图

程序清单 3-3 score.c

```
/*判断一个成绩对应的等级*/
#include<stdio.h>                    //包含所需要的预编译头文件 stdio.h
int main()                           //主函数
{
    int score;                       //定义一个学生分数 score
    printf("请输入一个百分制分数:");    //输入提示
    scanf("%d",&score);              //输入一个整数
    if(score >= 90)                  //判断该分数是否为 A 级
        printf("等级为 A! \n");        //满足 A 级条件,输出 A 级
    else if(score >= 80)             //判断该分数是否为 B 级
        printf("等级为 B! \n");        //满足 B 级条件,输出 B 级
    else if(score >= 70)             //判断该分数是否为 C 级
        printf("等级为 C! \n");        //满足 C 级条件,输出 C 级
    else if(score >= 60)             //判断该分数是否为 D 级
        printf("等级为 D! \n");        //满足 D 级条件,输出 D 级
    else
```

```
        printf("等级为 E! \n");              //不满足以上任何一个条件,输出 E 级
    return 0;
}
```

☞ **程序运行**

请输入一个百分制分数:76
等级为 C!

3.4.7　if 语句的嵌套

利用 if-else 嵌套实现多分支选择,具体语句格式如下:

```
if (表达式)
    if (表达式)
        语句体;
    else
        语句体;
else
    if (表达式)
        语句体;
    else
        语句体;
```

在 C 语言中允许使用 if-else 嵌套实现多分支选择结构,也就是在 if 或 else 子句中包含 if-else 语句的情况。上述语句如果由多条语句构成,要用“{}”括起来,构成复合语句。下面我们看个例子:

【例 3-4】 用 if-else 嵌套求三个数中的最大值。

☞ **编程点拨**

在三个数中,要求出一个最大值,就要首先比较其中的两个数,得到一个较大的值;然后再与第三个数进行比较,得到三个数中的最大值。其 N-S 图如图 3-8 所示。

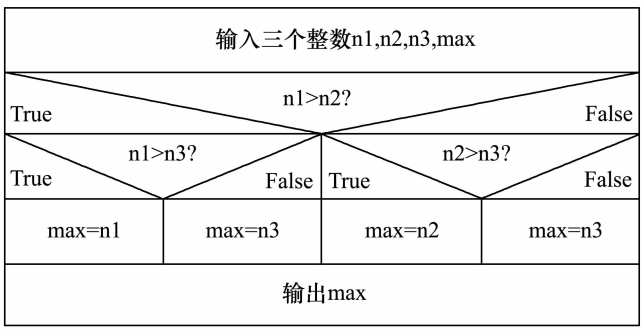


图 3-8　求三个数最大值的 N-S 图

程序清单 3-4 max.c

```
/*求三个数中的最大值*/
#include<stdio.h>                          //包含所需要的预编译头文件 stdio.h
int main()                                 //主函数
{
    int n1, n2, n3, max;                   //定义三个整数和一个变量 max 来记录最大值
    printf("请输入三个数:");                //输入提示
    scanf("%d%d%d",&n1,&n2,&n3);           //输入三个整数
    if(n1 > n2)                            //n1 与 n2 比较
    {
        if(n1 > n3)                        //n1 与 n3 比较
            max=n1;
        else
            max=n3;
    }
    else
    {
        if(n2 > n3)                        //n2 与 n3 比较
            max=n2;
        else
            max=n3;
    }
    printf("三个数的最大值是%d\n:",max);    //输出最大值
    return 0;
}
```

☞ **程序运行**

```
请输入三个数:23 5 100
三个数的最大值是 100
```

3.5　项目完善

在第 2 章的除法模块中,我们忽略了一种特殊情形——除数为零。当除数为零时,系统会出现“divide or mod by zero”的错误提示,那么如何在程序设计中避免这种问题的出现呢? 我们可以使用 if 分支语句来完成。具体步骤提示如下:

(1)在 calculator.c 文件中的 divide 函数中,按照图 3-9 所示的 N-S 图进行更新该模块的功能。在输入 number1 和 number2 之后使用 if-else 语句来完成除数为零的判定,改进除法模块中除数为零的问题。

(2)在主函数中调用 divide 函数。

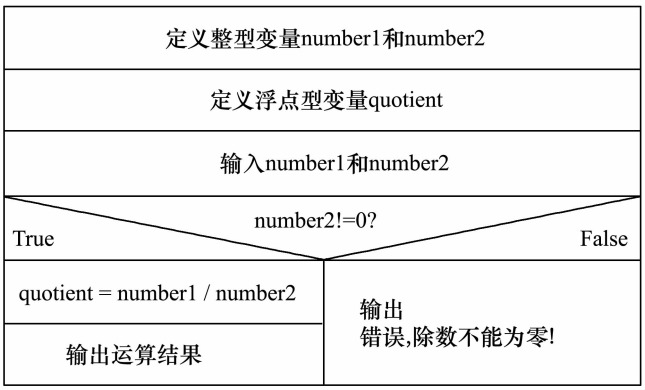


图 3-9 改进后的除法模块 N-S 图

自测题

1. 当 a=3,b=4,c=5 时,写出下列各式的值。

a < b 的值为_____,a==c 的值为_____,a && b 的值为_____,! a && b 的值为_____,a || c 的值为_____,! a || c 的值为_____,a+b > c && b==c 的值为_____。

2. 某市不同车牌的出租车 3 公里的起步价和计费分别为:夏利 7 元,3 公里以外 2.1 元/公里;富康 8 元,3 公里以外 2.4 元/公里;桑塔纳 9 元,3 公里以外 2.7 元/公里。编程:从键盘输入乘车的车型及公里数,输出应付的车资。

3. Write a program that asks the user to type in two integer values at the terminal. Test these two number to determine if the first is evenly divisible by the second, and then display an appropriate message at the terminal.