第一次作业

尹朝阳 物理学系

1 题目 1

1.1 题目描述

请采用 Fortran90 自由格式编写程序，求解实系数方程 $ax^2+bx+c=0$ 的解。要求：系数 a,b,c 为程序执行时键盘输入，a,b,c 为任意实数；所有的解均输出到屏幕。

1.2 程序描述

题目要求解实系数方程 $ax^2+bx+c=0$，观察发现该方程的最高次幂不超过 2，可以分为三种情况求解。

- 最高次系数为 0：此时 $a=0,b=0$
- 最高次系数为 1：此时 $a=0,b\neq 0$
- 最高次系数为 2：此时 $a\neq 0$

其中，对于第一、二种情况，直接进行求解即可。

对于二次型方程，常规解法为

$$x_1=\frac{-b-\sqrt{b^2-4ac}}{2a},\quad x_2=\frac{-b+\sqrt{b^2-4ac}}{2a},$$

然而如果考虑到计算机将代码转为机器码再执行，其中的进制转换将导致一定的精度丢失，即产生浮点数误差。在 a 很小时，该误差被放大到不可忽视的程度，故需要改进算法：

$$x_1=\frac{q}{a},\quad x_2=\frac{c}{q}$$

其中

$$q=-\frac{1}{2}[b+sgn(b)\sqrt{b^2-4ac}]$$

采用该解法可以较好地减小精度造成的误差。

本程序采用两个源文件分别对应两种解法，/SolveEquation/Find_Roots_of_QuadraticEquations_version1.f90 用前一种解法，在 a 较小时误差较大。在第二次课后，又重写了/SolveEquation/Find_Roots_of_QuadraticEquations_version2.f90，采用第二种解法。

1.3 伪代码

Algorithm 1 Find roots of the equation (version 2)

Input : three parameters a, b and c

Output: the root(s) of the equation

if $a \leftarrow 0$ *and* $b \leftarrow 0$ *and* $c \leftarrow 0$ **then**
  $x \leftarrow$ any number
end
if $a \leftarrow 0$ *and* $b \leftarrow 0$ *and* $c \neq 0$ **then**
  No solution
end
if $a \leftarrow 0$ *and* $b \neq 0$ **then**
  $x \leftarrow -\frac{c}{b}$
end
if $a \neq 0$ **then**
  if $b^2 - 4ac \geq 0$ **then**
    $x_1 \leftarrow \frac{q}{a}$, $x_2 \leftarrow \frac{c}{q}$ when $q \leftarrow -\frac{1}{2}[b + sgn(b)\sqrt{b^2 - 4ac}]$
  else
    No solution
  end
end

2 题目 2

2.1 题目描述

24 点游戏是儿时玩的主要益智类游戏之一，玩法为：从一副扑克中抽取 4 张牌，对 4 张牌使用加减乘除中的任何方法，使计算结果为 24。例如，2,3,4,6，通过 (((4+6)-2)*3)=24，最快算出 24 者胜。

请采用 Fortran90 编程求解 24 点游戏的解。

2.2 程序描述

本程序要求输入 4 个数字，判断是否有一种解法，能够用 +，-，*，/ 四个运算符最终计算出 24 点，如果有，在屏幕上输出其中一个解。

由于 4 个运算符之间有优先级，且加括号的行为可以改变运算优先级，本程序采用完全括号表达式进行计算，即对于每一个运算符，在需要操作的数两边加上括号，通过外围括号的个数表示当前运算符的级别，如第一个要操作的运算符处于最里面，最后一个操作符两边只有一对括号。

此时只需要对于三个运算符的顺序进行排列，一共有 6 种情况，不重复的有 5 种，分别为

- (((A _ B) _ C) _ D)
- ((A _ (B _ C)) _ D)
- (A _ ((B _ C) _ D))
- (A _ (B _ (C _ D)))
- ((A _ B) _ (C _ D))

其中字母表示的是输入的数字，“_”表示的是运算符。

程序中用 caseA()、caseB()、caseC()、caseD()、caseE() 这 5 个函数分别表示这五种情况。此外定义 calculate() 函数进行两个数之间的运算；在给定运算顺序的情况下定义 FindSolution() 函数进行 4 个输入数字的全排列，三个位置上的运算符均有可能是“加减乘除”中的一种，故 FindSolution() 函数还对运算符进行穷举。

本程序的源文件路径为/Game_24Point/Game_24Point.f90。

本程序的中文注释采用 uft-8 编码。

2.3 伪代码

由于伪代码较长，详见末尾。

2.4 测试用例

用例1

```
> gfortran "Game_24Point.f90" -o "Game_24Point.exe"
> ./Game_24Point.exe
Please input 4 numbers range from 1 to 10:1 2 3 4
(((1 + 2) + 3) * 4)
```

用例2

```
> gfortran "Game_24Point.f90" -o "Game_24Point.exe"
> ./Game_24Point.exe
Please input 4 numbers range from 1 to 10:3 8 3 8
(8 / (3 - (8 / 3)))
```

用例3

```
> gfortran "Game_24Point.f90" -o "Game_24Point.exe"
> ./Game_24Point.exe
Please input 4 numbers range from 1 to 10:1 1 1 1
No solution for this group of numbers you input.
Please try again.
```

附：第二题伪代码

Algorithm 2 Game_24Point

Input : 4 numbers from 1 to 10

Output: one (or none) solution for the game 24POINT

```
found = FindSolution (4numbers)
if found ← 1 then
    at least one solution
else if found ← 0 then
    Output: no solution
end

function FindSolution(a,b,c,d) // sort 4 numbers and choose 3 operators
for every sort of 4 numbers do
    for every choice of 3 operators do
        if caseA = 24 then
            Output: (((A _ B) _ C) _ D)
        else if caseB = 24 then
            Output: ((A _ (B _ C)) _ D)
        else if caseC = 24 then
            Output: (A _ ((B _ C) _ D))
        else if caseD = 24 then
            Output: (A _ (B _ (C _ D)))
        else if caseE = 24 then
            Output: ((A _ B) _ (C _ D))
        end
    end
end

function calculate(A, B, operator) // calculate the expression "A (operator) B"
return (A (operator) B)

function caseA() // (((A _ B) _ C) _ D)
calculate(operator1); calculate(operator2); calculate(operator3)

function caseB() // ((A _ (B _ C)) _ D)
calculate(operator2); calculate(operator1); calculate(operator3)

function caseC() // (A _ ((B _ C) _ D))
calculate(operator2); calculate(operator3); calculate(operator1)

function caseD() // (A _ (B _ (C _ D)))
calculate(operator3), calculate(operator2), calculate(operator1)

function caseE() // ((A _ B) _ (C _ D))
calculate(operator1), calculate(operator3), calculate(operator2)
```