实系数一元二次方程求根

问题描述

实系数一元二次方程是最简单的方程之一，其一般形式为$ax^2+bx+c=0$，其中a,b,c均为实数。该方程求解过程中一个重要的中间量是其判别式$\Delta=b^2-4ac$:

- 当$\Delta>0$时方程有两个不相等的实根，分别为$x_{1,2}=\frac{-b\pm\sqrt{\Delta}}{2a}$;
- 当$\Delta=0$时方程有两个相等的实根$x=-\frac{b}{2a}$;
- 当$\Delta<0$时方程有两个共轭复根$x_{1,2}=\frac{-b\pm i\sqrt{-\Delta}}{2a}$。

本程序即采用以上理论，利用**Fortran90**语言编写代码，实现了实系数一元二次方程求解。

方法描述

使用`read`语句使用户从键盘输入a,b,c三个参数，定义判别式`Judge`变量，利用选择结构，根据判别式与0的相对大小，分别利用上述求根公式求解出不等实根、相等实根、共轭复根情况的解。整个流程嵌套在一个循环结构内部，用户可以在一次求根结束后选择继续运算或退出程序。

伪代码

```
do
    Judge ← b^2 - 4ac   //定义判别式，abc为用户键盘输入
    x1 ← [-b+sqrt(Judge)]/(2a)
    x2 ← [-b-sqrt(Judge)]/(2a)  //代入求根公式

    if Judge > 0    //根据判别式判断根的情况
        then print x1,x2
    else if Judge = 0
        then print x1
    else
        temp ← complex(Judge)   //把判别式数据类型转换为复数，便于下面开方运算
        x1 ← [-b + sqrt(temp)] / (2a)
        x2 ← [-b - sqrt(temp)] / (2a)
        print x1,x2
    end
while conti = 'n'   //用户输入conti变量决定是否继续运行
```

I/O示例

```
lbx@lbx-virtual-machine:~/compu_phys/week1/FindRoot_dir$ gfortran -c FindRoot.f90
lbx@lbx-virtual-machine:~/compu_phys/week1/FindRoot_dir$ gfortran FindRoot.o -o FindRoot
lbx@lbx-virtual-machine:~/compu_phys/week1/FindRoot_dir$ ./FindRoot
 Input a,b and c of ax^2+bx+c=0
1 3 2
The equation has two different real roots:x1=-1.00000,x2=-2.00000
 Do you want to continue?(y/n)
y
 Input a,b and c of ax^2+bx+c=0
1 6 9
The equation has two identical real roots:x=-3.00000
 Do you want to continue?(y/n)
y
 Input a,b and c of ax^2+bx+c=0
1 1 4
The equation has two different complexroots:x1=-.50000+1.93649i,x2=-.50000-1.93649i
 Do you want to continue?(y/n)
n
```

可以看到三次输入分布代表不同方程类型，将输入输出列成表格为：

a	b	c	x_1	x_2
1	3	2	-1.00000	-1.00000
1	6	9	-3.00000	
1	1	4	$-0.50000+1.93649i$	$-0.50000-1.93649i$

源代码

完整源代码见附件。

```
temp=Complex(Judge,0.0)
x1=(-b+Sqrt(temp))/(2*a)
x2=(-b-Sqrt(temp))/(2*a)
```

这是源文件第21行代码。这里以0虚部将浮点数`Judge`构造为复数`temp`，是由于Fortran语言中`sqrt()`开方函数运算结果和输入参数的数据类型一致，因此想由实数得到复数则需进行类型转换。