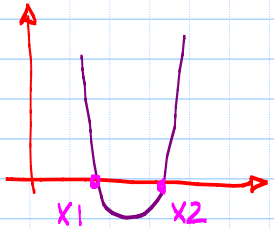
PROBLEMA PARA RESOLVER CON FUNCIONES

Note Title 9/7/2009

PROBLEMA:
Encontrar las 2 raíces de una función polinómica cuadrática (parábola).

función cuadrática:
$f(x) = ax^2 + bx + c$



discriminante

$$\begin{matrix} x_1 \\ x_2 \end{matrix} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

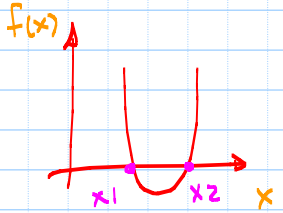
→ fórmula para encontrar las 2 raíces, x_1 y x_2

a no debe ser 0.0.

Casos:

$$D = b^2 - 4ac$$

1) $D > 0.0$

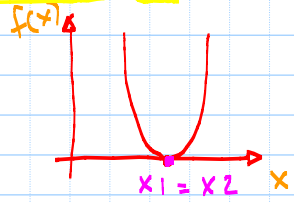


$$x_1 = \frac{-b + \sqrt{D}}{2a}$$

donde $D = b^2 - 4ac$

$$x_2 = \frac{-b - \sqrt{D}}{2a}$$

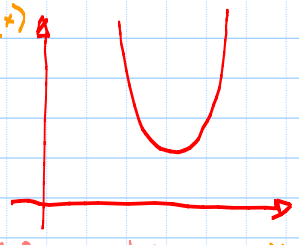
2) $D = 0.0$



$$x_1 = \frac{-b + \sqrt{D}}{2a} = \frac{-b}{2a}$$

$$x_2 = \frac{-b - \sqrt{D}}{2a} = \frac{-b}{2a}$$

3) $D < 0.0$



resp. compleja

$$x_1 = \frac{-b + \sqrt{D}}{2a} = \frac{-b}{2a} + \frac{\sqrt{D}}{2a}\,i = PR + PI\,i$$

Parte Real (PR)

Parte Imaginaria (PI)

$$PR = \frac{-b}{2a}$$

$$PI = \frac{\sqrt{D}}{2a}$$

$$x_2 = \frac{-b}{2a} - \frac{\sqrt{D}}{2a}\,i = PR - PI\,i$$

resp. compleja

PR → parte real

PI → parte imaginaria

Si se resuelve el problema con una sola función main la solución luce como sigue:

```
#include <iostream>
using namespace std;
#include <cmath>
int main ( )
{
    double a, b, c, D, x1, x2, PR, PI;

    cout << "Entre los 3 coeficientes de la función parabólica:";
    cin >> a >> b >> c;
(1) while (a == 0.0)
    {
        cout << "El coefic. a no puede ser cero. Re-entre: ";
        cin >> a;
    }

(2) D = b * b - 4.0 * a * c;  // D = pow(b, 2.0) - 4.0*a*c;
    if (D >= 0.0)  // Casos 1 y 2 , D >= 0
    {
(3)     x1 = (-b + sqrt(D)) / (2.0 * a);
        x2 = (-b - sqrt(D)) / (2.0 * a);

        cout << "x1 = " << x1
             << ", x2 = " << x2 << "\n\n";
    }else  // Caso 3, D < 0.0
    {
(4)     PR = -b / (2.0 * a);
        PI = sqrt(-D) / (2.0 * a);

        cout << "x1 = " << PR << " + " << PI << 'i'
             << ", x2 = " << PR << " - " << PI << "i\n\n";
    }
    return 0;
}
```

Se quiere resolver el mismo problema con las mismas 4 funciones, además de main:

(1) Función que solicite los coeficientes PideCoeficientes

(2) Calcule el discriminante Discriminante

(3) Calcule e imprima las raíces x1 y x2 cuando el discrim. es mayor ó igual que 0.0 Calc1

(4) Calcule e imprima las raíces x1 y x2 cuando el discrim. es menor que 0.0 Calc2

```
#include <iostream>
using namespace std;

#include <cmath>
// Prototipos ⟶
int main ( )
{
    double a, b, c, D;

(1) PideCoeficientes ( a, b, c );
(2) D = Discriminante ( a, b, c );
    if (D >= 0.0)
        Calc1 ( a, b, D ); ] (3)
    else
        Calc2 ( a, b, D ); ] (4)

    return 0;
}
```

```
void PideCoeficientes (double &, double &, double &);
double Discriminante (double, double, double);
void Calc1 (double, double, double);
void Calc2 (double, double, double);
```

```
void    PideCoeficientes (double &a, double &b, double &c)
{
   cout<< "Entre los 3 coeficientes de la función "
       << "parabólica: ";
   cin >> a >> b >> c;

   while (a == 0.0)
   {
      cout<< "El coefic. a no puede ser cero. "
          << "Re-entre: ";
      cin >> a;
   }
}
```

```
double Discriminante (double a, double b, double c)
{
   double D;
   D = b*b - 4.0*a*c;
   return D;
}
```

ó

```
double Discriminante (double a, double b, double c)
{
   return b*b - 4.0*a*c;
}
```

```
void Calc1 (double a, double b, double D)
{ double x1, x2;
   x1 = (-b + sqrt(D)) / (2.0*a);
   x2 = (-b - sqrt(D)) / (2.0*a);

   cout<< "x1 = " << x1
       << ", x2 = " << x2 << "\n\n";

}
```

```
void Calc2 (double a, double b, double D)
{ double PR, PI;
   PR = -b / (2.0*a);
   PI = sqrt(-D) / (2.0*a);

   cout << "x1 = " << PR << " + " << PI << 'i'
        << ", x2 = " << PR << " - " << PI << "i\n\n";

}
```

Ahora, suponga que las funciones Calc1 y Calc2 se la describen distinto a como se hizo anteriormente, como sigue:

(3) Calcule ~~e imprima~~ las raíces x1 y x2 cuando el discrim. es mayor ó igual que 0.0
(Ahora le llamaremos Calc1M a esta función)

(4) Calcule ~~e imprima~~ las raíces x1 y x2 cuando el discrim. es menor que 0.0
(Ahora le llamaremos Calc2M a esta función).

```
#include <iostream>
using namespace std;

#include <cmath>
// Prototipos
    ( )

void PideCoeficientes (double &,
         double &, double &);
double Discriminante (double,
         double, double);
void Calc1M (double, double,
    double, double &, double &);
void Calc2M (double, double,
    double, double &, double &);

int main
{
    double a, b, c, D, x1, x2, PR, PI;

(1) [PideCoeficientes ( a, b, c );
(2) [D = Discriminante ( a, b, c );
    if (D >= 0.0)
(3) {
        Calc1M( a, b, D, x1, x2);
        cout << "x1 = " << x1
             << ", x2 = " << x2 << "\n\n";
    }
    else
(4) {   Calc2M( a, b, D, PR, PI);
        cout << " x1 = " << PR << " + " << PI << 'i'
             << ", x2 = " << PR << " - " << PI << "i\n\n";
    }
    return 0;
}
```

Note los cambios que la nueva descripción de Calc1M y Calc2M provocan en main

```
// La nueva función Calc1M luce como sigue:
void Calc1M(double a, double b, double D, double &x1, double &x2)
{
    x1 = (-b + sqrt(D))/(2.0 * a);
    x2 = (-b - sqrt(D))/(2.0 * a);
}
```

```
// La nueva función Calc2M luce como sigue:
void Calc2M( double a, double b, double D, double &PR, double &PI)
{
    PR = -b/ (2.0 * a);
    PI = sqrt (-D)/(2.0 * a);
}
```

Ahora se quiere resolver el mismo problema solo con las sigts. 3 funciones además de main:

(1) Función que solicite los coeficientes PideCoeficientes

(2) Calcule el discriminante Discriminante

(3) Haga los cálculos de las raíces y las imprima Calculos

```
#include <iostream>
using namespace std;

#include <cmath>

int main ( )          // El main ahora luce así:
{
    double a, b, c, D;

(1) [PideCoeficientes ( a, b, c );
(2) [D = Discriminante ( a, b, c );

(3) Calculos (a, b, D);

    return 0;
}
```

```
//Las funciones PideCoeficientes y Discriminante
// quedan idénticas a las vis. anteriores
// por lo que NO se ha escrito nuevamente su código.


//La función Cálculos ahora luce así:
(3) void Calculos (double a, double b, double D)
{
    double X1, X2, PR, PI;

    if (D >= 0.0) // Casos 1 y 2, D >=0
    {
        X1 = (-b + sqrt(D)) / (2.0 * a);
        X2 = (-b - sqrt(D)) / (2.0 * a);

        cout << "X1 = " << X1
             << ", X2 = " << X2 << "\n\n";
    }else // Caso 3, D < 0.0
    {
        PR = -b / (2.0 * a);
        PI = sqrt(-D) / (2.0 * a);

        cout << "X1 = " << PR << " + " << PI << 'i'
             << ", X2 = " << PR << " - " << PI << "i\n\n";
    }
}

// El prototipo de la función Calculos es:

void Calculos (double, double, double);
```

Si se cambia la descripción de la función Calculos a la sigt:

(3) Función que calcule e imprima las 2 raíces, pero llamando, respectivamente, a las siguientes 2 funciones:

(3a) Calcule e imprima las raíces x1 y x2 cuando el discrim. es mayor ó igual que 0.0
Calc1

(3b) Calcule e imprima las raíces x1 y x2 cuando el discrim. es menor que 0.0
Calc2

```
//La función Cálculos, Calc1 y Calc2
// lucirían ent. como se muestra a continuación... main, PideCoeficientes y
// Discriminante se quedan igual que en el
// ejemplo anterior por lo que NO se
// repite el código aquí...

//La función Cálculos ahora luce así:
(3) void Calculos (double a, double b, double D)
{
    if (D >= 0.0) // Casos 1 y 2, D >=0

        Calc1 (a, b, D);
    else // Caso 3, D < 0.0
        Calc2 (a, b, D);
}
```

```
// y las funciones Calc1 y Calc2 deben estar
// desarrolladas en este programa:
(3a) void Calc1 (double a, double b, double D)
{ double x1, x2;
   x1 = (-b + sqrt(D)) / (2.0 * a);
   x2 = (-b - sqrt(D)) / (2.0 * a);

   cout << "x1 = " << x1
        << ", x2 = " << x2 << "\n\n";

}

(3b) void Calc2 (double a, double b, double D)
{
   double PR, PI;
   PR = -b / (2.0 * a);
   PI = sqrt(-D) / (2.0 * a);

   cout << "x1 = " << PR << " + " << PI << 'i'
        << ", x2 = " << PR << " - " << PI << "i\n\n";
}
```

NOTA: En todos estos ejercicios, la forma más elegante de imprimir las raíces cuando discriminante es menor que 0.0 no es la brindada en estas soluciones. Uds. pueden explorar formas más elegantes de imprimir las raíces complejas. (lo hicimos en clase, pero no se muestra aquí)

Dic. 6, 2010
Rev. 1, 09/09/2009