Équation quadratique [ss05] - Exercice

Karine Zampieri, Stéphane Rivière

Unisciel algoprog Université Haute-Alsace

Version 16 mai 2018

Table des matières

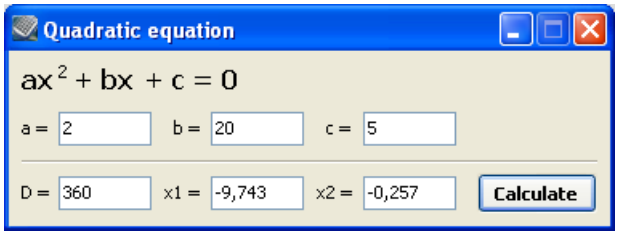
C - Équation quadratique (Solution)

Mots-Clés Algorithmes paramétrés ■
Requis Structures de base, Structures conditionnelles ■
Difficulté • ○ ○ (40 min à 50 min) ■

Objectif
Cet exercice détermine le nombre et les solutions de l'équation quadratique

$$a\,x^2 + b\,x + c = 0$$

en examinant tous les cas, même quand certains coefficients sont nuls.
(image : http://kalkules.com/index.php?page=tools&language=fr)



1 Équation quadratique / pgquadratik

1.1 Équation linéaire

Propriété

Dans $\mathbb{R}$, l'équation linéaire $a\,x + b = 0$ a pour solution :

- Si a n'est pas nul : une solution $(x = -b/a)$
- Sinon si b n'est pas nul : aucune solution
- Sinon (a et b nuls) : l'ensemble $\mathbb{R}$

Écrivez le **profil** d'une procédure `resoudreDegre1(a,b,n,x)` qui résout l'équation linéaire $a\,x + b = 0$ en restituant le nombre de solutions dans `n` (entier) et l'éventuelle solution dans `x` (réel).

Solution Paramètres

Entrants : `a` et `b`

Sortants : `n` et `x`

Écrivez le corps de la procédure.
Dans le cas de l'ensemble $\mathbb{R}$ solution, mettez -1 dans `n`.

Validez votre procédure avec la solution.

1.2 Équation quadratique

Propriété

L'équation quadratique $a\,x^2 + b\,x + c = 0$ a pour discriminant :

$$\Delta = b^2 - 4\,a\,c$$

Dans $\mathbb{R}$, ses solutions sont :

- Si Δ est positif : deux racines $(-b \pm \sqrt{\Delta})/(2a)$
- Sinon si Δ est nul : une racine double $(-b)/(2a)$
- Sinon (Δ est négatif) : aucune racine réelle

Écrivez le **profil** d'une procédure `resoudreDegre2(a,b,c,n,x1,x2)` qui résout l'équation quadratique $a\,x^2 + b\,x + c = 0$ avec `a` **supposé non nul**, en restituant le nombre de solutions dans `n` (entier) et les éventuelles solutions dans `x1` (réel) et `x2` (réel).

Solution Paramètres

Entrants : Les réels `a,b,c`

Sortants : Un entier `n` et les réels `x1,x2`

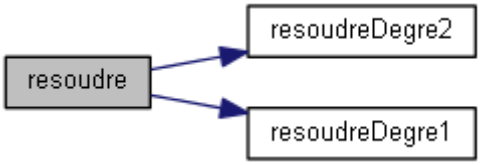
Écrivez le corps de la procédure.

Outil C
La fonction racine carrée `sqrt(x)` est définie dans la bibliothèque `<math.h>`.

Validez votre procédure avec la solution.

1.3 Résolution

Déduisez une procédure `resoudre(a,b,c,n,x1,x2)` qui résout l'équation quadratique $ax^2 + bx + c = 0$ dans $\mathbb{R}$ en restituant le nombre de solutions dans `n` (entier) et les éventuelles solutions dans `x1` (réel) et `x2` (réel).



Solution simple
On teste le premier coefficient `a` et on appelle la procédure `resoudreDegre2` (s'il est non nul) ou la procédure `resoudreDegre1` (cas sinon) avec les bons paramètres.

Validez votre procédure avec la solution.

Enfin écrivez une procédure `afficherSolutions(n,x1,x2)` qui, selon le nombre de solutions `n` (entier 2, 1, 0 ou négatif), affiche (où `[x]` désigne le contenu de `x`) :

```
==> Deux racines: [x1] et [x2] # cas n=2
==> Une racine: [x1] # cas n=1
==> Aucune solution dans R # cas n=0
==> L'ensemble R # cas autre
```

Validez votre procédure avec la solution.

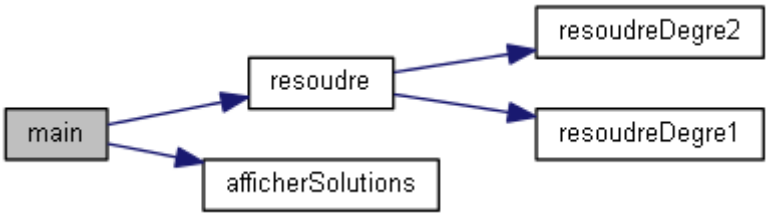
1.4 Programme principal

Écrivez une procédure `test_quadratik1` qui :

- Saisit les coefficients réels d'une équation quadratique.
- Calcule le nombre de solutions et les éventuelles solutions réelles.
- Puis affiche les solutions.

Affichez l'invite :

```
Coefficients a b c de l'équation?
```



Testez le programme sur les trois équations suivantes :

- L'équation : $-2x^2 + 7x + 15$ qui a deux racines réelles : -1.5 et 5
- L'équation : $x^2 - 2x + 1$ qui a une racine double : 1
- L'équation : $x^2 + 3x + 3$ qui n'a pas de racine réelles

Validez votre procédure avec la solution.

Cas de solutions très différentes
On pourra consulter l'exercice @[Équation quadratique] (dans les conditionnelles) qui traitent le cas de solutions très différentes. ■

1.5 Résolution dans les complexes

Dans le cas où le discriminant Δ est négatif,

- La partie réelle est donnée par la formule : $-b/(2a)$.
- Et la partie imaginaire par : $\pm(\sqrt{-\Delta}/(2a))i$ où i représente la valeur $\sqrt{-1}$.

Copiez/collez la procédure `resoudreDegre2` en la procédure `resoudreDegre2C`.
Modifiez-la de sorte à résoudre **toutes** les solutions de l'équation quadratique.
Dans ce cas, mettez l'entier -2 dans `n`.

De même, copiez/collez la procédure `resoudre` en la procédure `resoudreC` et modifiez-la de sorte à appeler la procédure de résolution dans $\mathbb{C}$.

Enfin, copier/collez la procédure `afficherSolutions` en la procédure `afficherSolutionsC` et modifiez-la de sorte qu'elle affiche :

```
==> Deux racines sur R: [x1] et [x2] # cas n=2
==> Une racine: [x1] # cas n=1
==> Aucune solution # cas n=0
==> L'ensemble R # cas n=-1 (degré 1)
==> Deux racines sur C: [x1]-[x2]i et [x1]+[x2]i
```

Validez vos procédures avec la solution.

Copiez/collez la procédure `test_quadratik1` en la procédure `test_quadratik2` puis modifiez-la de sorte à appeler les procédures de résolution dans $\mathbb{C}$.

Testez avec $(a,b,c) = (2,6,1)$, $(3,3,0)$, $(1,3,1)$, $(0,12,-3)$, $(3,6,3)$ et $(2,-4,3)$.

Validez votre procédure avec la solution.

Solution C @[pgquadratik.c]

```c
#include <stdio.h>
#include <math.h>


void resoudreDegre1(double a,double b,int* n,double* x)
{
   if (a!=0.0)
   {
      *n = 1;
      *x = -b/a;
   }
   else if (b!=0.0)
   {
      *n = 0;
   }
   else
   {
      *n = -1;
   }
}

void resoudreDegre2(double a,double b,double c,int* n,double* x1,double* x2)
{
  double delta = b*b-4*a*c;
  if (delta > 0.0)
  {
    *n = 2;
    *x1 = (-b+sqrt(delta))/(2*a);
    *x2 = (-b-sqrt(delta))/(2*a);
  }
  else
  {
    if (delta == 0.0)
    {
      *n = 1;
      *x1 = *x2 = (-b)/(2*a);
    }
    else
    {
      *n = 0;
    }
  }
```

```
}

void resoudre(double a,double b,double c,int* n,double* x1,double* x2)
{
  if (a != 0.0)
  {
    resoudreDegre2(a,b,c,n,x1,x2);
  }
  else
  {
    resoudreDegre1(b,c,n,x1);
  }
}

void afficherSolutions(int n,double x1,double x2)
{
  switch (n)
  {
    case 2:
      printf("==> Deux racines: %g et %g\n",x1,x2);
      break;
    case 1:
      printf("==> Une racine: %g\n",x1);
      break;
    case 0:
      printf("==> Aucune solution dans R\n");
      break;
    default:
      printf("==> L'ensemble R\n");
  }
}

void test_quadratik1()
{
  double a;
  double b;
  double c;
  printf("Coefficients a b c de l'equation? ");
  scanf("%lf%lf%lf",&a,&b,&c);
  int ns;
  double x1;
  double x2;
  resoudre(a,b,c,&ns,&x1,&x2);
  afficherSolutions(ns,x1,x2);
}

void resoudreDegre2C(double a,double b,double c,int* n,double* x1,double* x2)
{
   double delta = b*b-4*a*c;
   if (delta > 0.0)
   {
      *n = 2;
      *x1 = (-b+sqrt(delta))/(2*a);
      *x2 = (-b-sqrt(delta))/(2*a);
   }
   else if (delta == 0.0)
   {
      *n = 1;
```

```
      *x1 = *x2 = (-b)/(2*a);
   }
   else
   {
      *n = -2;
      *x1 = (-b)/(2*a);
      *x2 = sqrt(-delta)/(2*a);
   }
}

void resoudreC(double a,double b,double c,int* n,double* x1,double* x2)
{
  if (a != 0.0)
  {
    resoudreDegre2C(a,b,c,n,x1,x2);
  }
  else
  {
    resoudreDegre1(b,c,n,x1);
  }
}

void afficherSolutionsC(int n,double x1,double x2)
{
  switch (n)
  {
    case 2:
      printf("==> Deux racines sur R: %g et %g\n",x1,x2);
      break;
    case 1:
      printf("==> Une racine: %g\n",x1);
      break;
    case 0:
      printf("==> Aucune solution\n");
      break;
    case -1:
      printf("==> L'ensemble R\n");
      break;
    default:
      printf("==> Deux racines sur C: %g-%gi et %g+%gi\n",x1,x2,x1,x2);
  }
}

void test_quadratik2()
{
  double a;
  double b;
  double c;
  printf("Coefficients a b c de l'equation? ");
  scanf("%lf%lf%lf",&a,&b,&c);
  int ns;
  double x1;
  double x2;
  resoudreC(a,b,c,&ns,&x1,&x2);
  afficherSolutionsC(ns,x1,x2);
}

int main()
```

```
{
   test_quadratik1();
   test_quadratik2();
}
```

2 Références générales

Comprend [Chappelier-CPP1 :c02 :et01], [Maunoury-AL1 :c05 :ex05], [Lemaitre-CC1 :c5 :ex1], [Gottfried-CC1 :c7 :ex39..ex42] ■