Algorithmes, feuille, crayon

Christophe BAL : projetmbc@gmail.com

Version du 2015-10-25

Mentions « légales »

À l'adresse `https://github.com/bc-writings-algo-general/algorithms_with_paper`, vous trouverez le code source LaTeX de ce document : voir le dossier `fr`.

1 Aperçu

Ce document souhaite proposer une prise de contact très douce avec les algorithmes grâce à des exercices courts ne faisant appel à aucune connaissance particulière du lycée. Deux publics sont visés.

1. Les lycéens dans le cadre du cours de mathématiques doivent apprendre à faire fonctionner mentalement un algorithme pour pouvoir ensuite l'analyser sans s'aider d'une traduction sous forme de programme [1].
2. Les programmeurs débutants doivent savoir faire fonctionner un algorithme sans le programmer. Ceci force à s'organiser mais aussi à analyser ce qui est fait [2].

2 Qu'appelle-t-on « algorithme » ?

Que nous dit Wikipédia France [3] à propos des algorithmes ?

> Un algorithme est une suite finie et non ambiguë d'opérations ou d'instructions permettant (à un ordinateur) de résoudre un problème.
>
> Wikipédia

Trois choses importantes sont à remarquer.

1. L'utilisation de « suite finie et non ambiguë d'opérations ou d'instructions » indique qu'un algorithme doit pouvoir se mettre sous forme d'un texte fini.
2. Lorsque l'on lit « permettant de résoudre un problème », ceci sous-entend qu'un algorithme doit se finir en un temps fini, éventuellement démesurément grand.
3. Enfin l'auteur a mis le mot « ordinateur » entre des parenthèses car il veut souligner le fait que l'on peut appliquer des algorithmes dans d'autres situations comme en pâtisserie pour appliquer une recette [4], ou bien pour monter très facilement un meuble de faible qualité, ou encore quand l'on répète une chorégraphie non improvisée... etc.

1. Très souvent en devoir, traduire les algorithmes à étudier est bien trop chronophage, et parfois certains algorithmes ne sont pas traduisibles pour les calculatrices standards car ils utilisent des notations symboliques ou des concepts graphiques.
2. Il est toujours gratifiant de taper des centaines de lignes de code pour faire ce que l'on souhaite, on peut même se prendre pour le roi du monde, mais il est bien plus utile à long terme de faire les choses avec intelligence et efficacité. L'auteur se permet cette remarque car il a été l'une des victimes de ce que les anglo-saxons pourraient nommer le *"direct coding"*, une pratique consistant à taper plus qu'à réfléchir à l'organisation du code.
3. Texte copié-collé le 30 Septembre 2015.
4. La cuisine de plats salés n'a pas été prise en exemple car elle demande moins de rigueur, et donc elle autorise qu'une recette ne soit pas suivie à la lettre.

Il se trouve que la définition de Wikipédia France devient bonne à condition de la modifier comme suit.

> “ Un algorithme est une suite finie et non ambiguë d'opérations ou d'instructions **reproductibles** permettant de résoudre **en un nombre fini d'étapes** un problème, en lien ou non avec l'informatique. ”

Pour finir voyons ce que nous dit le site de l'Encyclopédie Larousse en ligne [5].

> “ Étymologie : latin médiéval algorithmus, latinisation du nom d'un mathématicien de langue arabe [a], avec influence du grec arithmos, nombre.
>
> Ensemble de règles opératoires dont l'application permet de résoudre un problème énoncé au moyen d'un nombre fini d'opérations. Un algorithme peut être traduit, grâce à un langage de programmation, en un programme exécutable par un ordinateur.
>
> Encyclopédie Larousse en ligne ”
>
> *a.* Ce mathématicien se nomme Al-Khwarizmi.

Cette définition est bonne dans sa première partie mais elle devient plus discutable dans la seconde car même si en programmation l'étude d'algorithmes est primordial, ceci ne veut pas dire que tout algorithme est traduisible en un programme exécutable par un ordinateur [6].

3 Exercices de « lecture »

3.1 Des algorithmes via un langage courant comme chez son pâtissier

3.1.1 Manipulation de variables

[Algo.1] Que renvoie l'algorithme suivant ?

```
X prend la valeur 6.
X augmente de 1.
Renvoyer X.
```

[Algo.2] Que renvoie l'algorithme suivant ?

```
X prend la valeur 2.
X prend la valeur X + 4.
Renvoyer X.
```

[Algo.3] Que renvoie l'algorithme suivant ?

```
X prend la valeur 10.
X diminue de 1.
X prend la valeur X + 3.
X augmente de 5.
X prend la valeur 100 − X.
Renvoyer X.
```

[Algo.4] Que renvoie l'algorithme suivant ?

```
X prend la valeur 10.
X prend la valeur X − 4.
X prend la valeur X + 10.
X prend la valeur X + 3.
X prend la valeur X − 8.
X prend la valeur 199 768 − X.
```

[Algo.5] Qu'affiche l'algorithme suivant ?

```
X prend la valeur 7.
X prend la valeur X × 3.
X prend la valeur X − 11.
Afficher X.
```

[Algo.6] Qu'affiche l'algorithme suivant ?

```
X prend la valeur 7.
X prend la valeur X − 11.
X prend la valeur X × 3.
Afficher X.
```

5. Texte copié-collé le 30 Septembre 2015.

6. Les physiciens estiment à environ 10^{80} le nombre total d'atomes présents dans l'Univers. Admettons que cette approximation soit exacte. Imaginons un algorithme très simple qui demande d'écrire 10^{80} lettres X les unes après les autres. Bien que facile à traduire dans n'importe quel langage de programmation, on n'obtient pas pour autant un programme physiquement exécutable.

[Algo.7] Est-il vrai que l'algorithme suivant affichera le nombre décimal 0,33333333333 ?

```
X prend la valeur 9.
X prend la valeur X − 8.
X prend la valeur X/3.
Afficher X.
```

[Algo.8] Est-il vrai que l'algorithme suivant affichera le nombre décimal 0,25 ?

```
X prend la valeur 4.
X prend la valeur −X.
X prend la valeur X + 5.
X prend la valeur X/4.
```

3.1.2 Tester des situations

[Algo.9] Qu'affiche l'algorithme ci-dessous lorsque $a = -1$? Même question pour $a = 3$, puis pour $a = 2$?

```
Donnée
    Un réel a.
Actions
    Si a est supérieur ou égal à 2 alors :
        Afficher a².
    Sinon :
        Afficher a + 2.
    Fin du ou des test(s) Si
```

[Algo.10] Qu'affiche l'algorithme ci-dessous lorsque $a = -1$? Même question pour $a = 3$, puis pour $a = 2$?

```
Donnée
    Un réel a.
Actions
    Si a est supérieur ou égal à 2 alors :
        Afficher a².
    Sinon si a est inférieur ou égal à (−2) alors :
        Afficher a + 2.
    Fin du ou des test(s) Si
```

[Algo.11] Que renvoie l'algorithme ci-dessous lorsque $n = 10$? Même question pour $n = 7$?

```
Donnée
    Un naturel n.
Actions
    Si n est pair alors :
        n prend pour valeur le quotient de
        la division euclidienne de n par 2.
    Sinon :
        n prend pour valeur 3n + 1.
    Fin du ou des test(s) Si
    Renvoyer n.
```

[Algo.12] Cet algorithme renverra toujours la même chose que l'algorithme précédent n°11. Vrai ou faux ?

```
Donnée
    Un naturel n.
Actions
    Si n est pair alors :
        n prend pour valeur le quotient de
        la division euclidienne de n par 2.
    Sinon :
        n prend pour valeur 3n + 1.
        Renvoyer n.
    Fin du ou des test(s) Si
```

[Algo.13] Qu'affiche l'algorithme ci-dessous lorsque $n = 11$, $n = 33$, $n = 4$, $n = 70$, et enfin $n = 25$?

```
Donnée
    Un naturel n.
Actions
    Si n est un multiple de 2 alors :
        n prend pour valeur n/2.
    Sinon si n est un multiple de 3 alors :
        n prend pour valeur n/3.
    Sinon si n est un multiple de 5 alors :
        n prend pour valeur n/5.
    Sinon :
        n prend pour valeur 0.
    Fin du ou des test(s) Si
    Afficher n sous forme réduite.
```

[Algo.14] Qu'affiche l'algorithme ci-dessous lorsque $n = 11$, $n = 33$, $n = 4$, $n = 70$, et enfin $n = 25$?

```
Donnée
    Un naturel n.
Actions
    Si n est un multiple de 2 alors :
        n prend pour valeur n/2.
    Fin du ou des test(s) Si
    Si n est un multiple de 3 alors :
        n prend pour valeur n/3.
    Fin du ou des test(s) Si
    Si n est un multiple de 5 alors :
        n prend pour valeur n/5.
    Sinon :
        n prend pour valeur 0.
    Fin du ou des test(s) Si
    Afficher n sous forme réduite.
```

3.1.3 Répéter une action un nombre de fois « connu »

[Algo.15] Qu'affiche l'algorithme suivant ?

```
Pour TEST un naturel allant de 2
à 4 faire :
    Afficher TEST.
Fin de la boucle Pour
```

[Algo.16] Qu'affiche l'algorithme suivant ?

```
Pour CONCENTRE un naturel pair allant
de 3 à 12 faire :
    Afficher CONCENTRE.
Fin de la boucle Pour
```

[Algo.17] Qu'affiche l'algorithme ci-dessous pour $X = 6$?

```
Donnée
    Un réel X.
Actions
    Afficher X.
    Pour i un naturel allant de 1 à 3 faire :
        X prend la valeur X + 1.
    Fin de la boucle Pour
```

[Algo.18] Qu'affiche l'algorithme ci-dessous pour $X = 6$?

```
Donnée
    Un réel X.
Actions
    Pour i un naturel allant de 1 à 3 faire :
        Afficher X.
        X prend la valeur X + 1.
    Fin de la boucle Pour
```

[Algo.19] Qu'affiche l'algorithme ci-dessous pour $X = 6$?

```
Donnée
    Un réel X.
Actions
    Pour i un naturel allant de 1 à 3 faire :
        X prend la valeur X + 1.
        Afficher X.
    Fin de la boucle Pour
```

[Algo.20] Qu'affiche l'algorithme ci-dessous pour $X = 6$?

```
Donnée
    Un réel X.
Actions
    Pour i un naturel allant de 1 à 3 faire :
        X prend la valeur X + 1.
    Fin de la boucle Pour
    Afficher X.
```

[Algo.21] Qu'affiche l'algorithme ci-après lorsque $S = 6$?

```
Donnée
    Un réel S.
Actions
    Pour k un naturel allant de 1 à 3 faire :
        S prend la valeur S + k.
    Fin de la boucle Pour
    Afficher S.
```

[Algo.22] Qu'affiche l'algorithme ci-après lorsque $S = 11$?

```
Donnée
    Un réel S.
Actions
    Pour i un naturel allant de 1 à 3 faire :
        R est la valeur renvoyée par l'algorithme
        n°11 appliqué à S (voir la section 4.1.2).
        S prend la valeur R.
    Fin de la boucle Pour
    Afficher S.
```

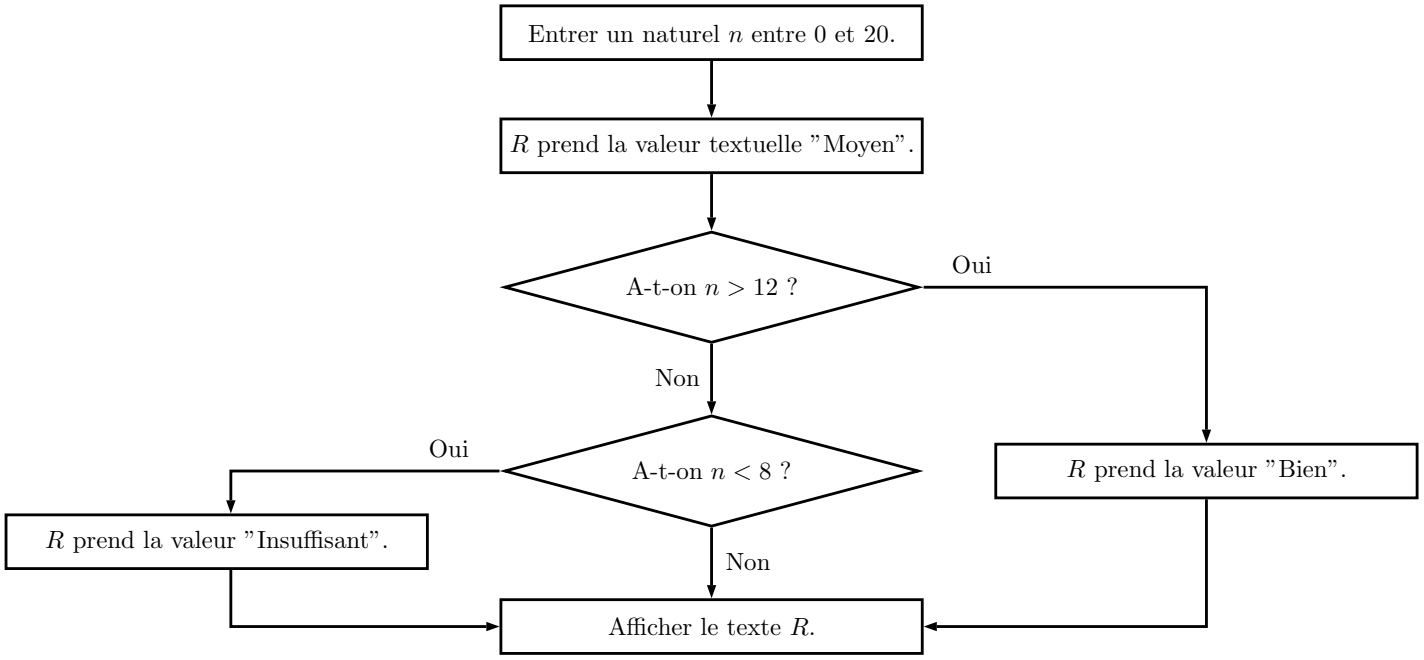
3.2 Certains algorithmes directement sous forme de diagrammes

Les diagrammes proposés en exercice dans cette section seront désignés par le néologisme « algorigrammes » [7]. Nous n'utiliserons pas les conventions usuelles.

7. Tous les diagrammes ont tous été réalisés avec le logiciel libre et gratuit LaTeXDraw que vous pourrez télécharger à l'adresse suivante : `http://latexdraw.sourceforge.net`.

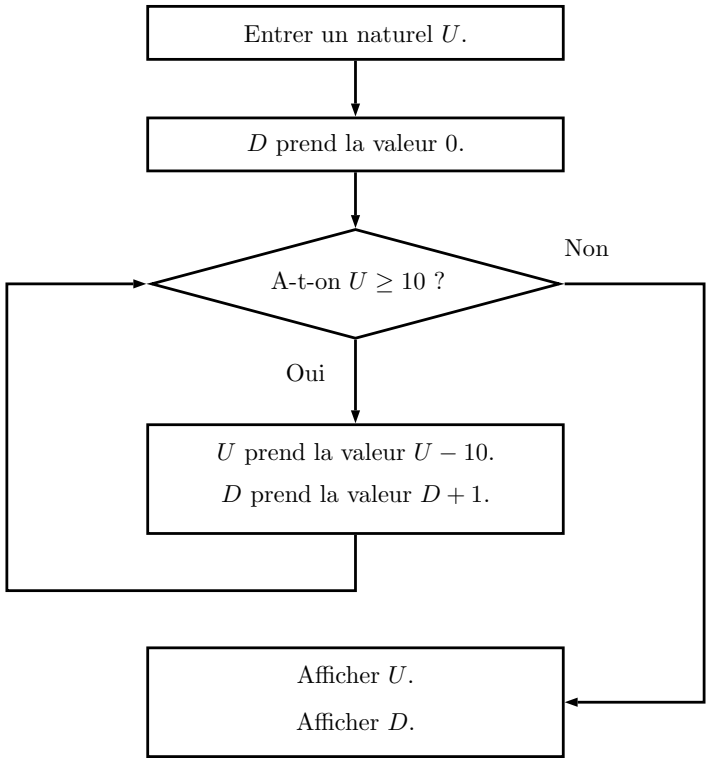
3.2.1 Utilisation de textes

Commençons par un cas simple qui devrait vous rappeler de bons ou mauvais souvenirs. Qu'affiche l'algorithme représenté par l'algorigramme ci-dessous lorsque $n = 10$, $n = 5$ puis $n = 18$?



3.2.2 Répéter une action un nombre de fois « inconnu »

Considérons l'algorigramme ci-dessous.



Traduire cet algorigramme en langage naturel puis déterminer l'affichage obtenu lorsque $U = 43$.

3.2.3 Répéter une action un nombre de fois « connu »

L'une des limitations des algorigrammes, tels que nous les avons présentés, est qu'il n'existe pas de convention pour indiquer une boucle du type `Pour ... un naturel allant de ... à ...` . Sauriez-vous malgré tout contourner cette limitation pour proposer un algorigramme représentant « presque » l'algorithme **[Algo.21]** ?

3.3 En utilisant un langage symbolique

3.3.1 Est-ce efficace ?

Nous redonnons ci-dessous la version « langage naturel » de l'algorithme n°14, puis nous proposons à côté une version symbolique de cet algorithme. Repérez les conventions utilisées. On constate que l'écriture « symbolique » est très économe en texte, et du coup il est très facile de la parcourir pour l'analyser.

[Algo.14]

```
Donnée
    Un naturel N.
Actions
    Si N est un multiple de 2 alors :
        N prend pour valeur N/2.
    Fin du ou des test(s) Si
    Si N est un multiple de 3 alors :
        N prend pour valeur N/3.
    Fin du ou des test(s) Si
    Si N est un multiple de 5 alors :
        N prend pour valeur N/5.
    Sinon :
        N prend pour valeur 0.
    Fin du ou des test(s) Si
    Afficher N sous forme réduite.
```

[Algo.14 "Symbolique"] Ci-dessous, $k \mid n$ signifie que le naturel k divise le naturel n. Par exemple, 3 divise 15 mais il ne divise pas 13.

Donnée : $n \in \mathbb{N}$
Début
- **Si** $2 \mid n$ **:**
  - $n \leftarrow \frac{n}{2}$
- **Si** $3 \mid n$ **:**
  - $n \leftarrow \frac{n}{3}$
- **Si** $5 \mid n$ **:**
  - $n \leftarrow \frac{n}{5}$
- **Afficher** n *sous forme réduite.*

3.3.2 Un retour en boucle

[Algo.23] Qu'affiche l'algorithme ci-dessous lorsque $n = 6$?

Donnée : n un naturel
Début
- $F = 1$
- **Pour** $i = 1$ **à** n **:**
  - $F \leftarrow F \times i$
- **Afficher** F.

[Algo.24] Qu'affiche l'algorithme ci-dessous lorsque $x = 6$ puis $x = 12$?

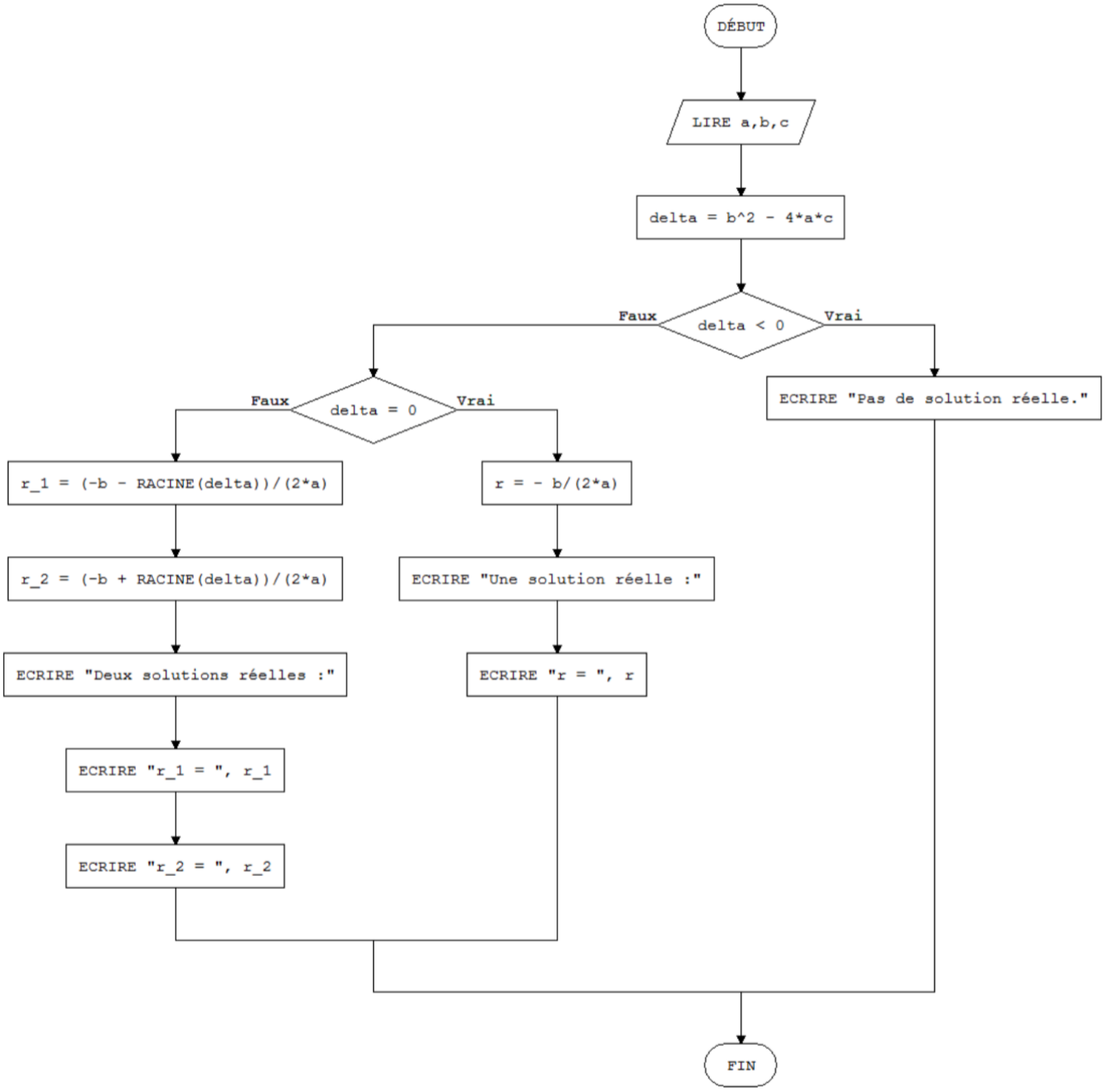
Donnée : un réel x.
Début
- **Tant Que** $x \leqslant 10$ **:**
  - $x \leftarrow x + 1$
  - **Afficher** x.

3.3.3 Traduire en version symbolique

Donner une version symbolique des algorithmes n°9, n°10 et n°11. Faire de même avec les deux algorigrammes présentés dans les sections 4.2.1 et 4.2.2.

Annexe : Les algorigrammes ou la folie des grandeurs

Cette annexe utilise le logiciel LARP qui est gratuit depuis le 1er Mars 2008, téléchargeable à l'adresse suivante `http://www.larp.marcolavoie.ca/fr/default.htm`, et qui malheureusement ne fonctionne que sous Windows©. LARP, qui est l'acronyme de Logiciel d'Algorithmes et de Résolution de Problèmes, permet au choix de taper des programmes ou bien d'utiliser des algorigrammes. La seconde option permet de produire l'algorigramme ci-dessous qui propose une méthode de résolution dans $\mathbb{R}$ d'une équation polynomiale du second degré [8] où nous supposons implicitement que $a \neq 0$.



LARP permet de produire directement une version « programme » de l'algorigramme précédent. Nous obtenons ce qui suit où de nouveau nous supposons implicitement que $a \neq 0$.

8. Nul besoin de connaître cette théorie qui est généralement abordée en classe de première. L'idée ici est juste de comparer « visuellement » différentes représentations d'un algorithme non simplistes.

```
DÉBUT
    LIRE a,b,c
    delta = b^2 - 4*a*c
    SI delta < 0 ALORS
        ECRIRE "Pas de solution réelle."
    SINON
        SI delta = 0 ALORS
            r = - b/(2*a)
            ECRIRE "Une solution réelle :"
            ECRIRE "r = ", r
        SINON
            r_1 = (-b - RACINE(delta))/(2*a)
            r_2 = (-b + RACINE(delta))/(2*a)
            ECRIRE "Deux solutions réelles :"
            ECRIRE "r_1 = ", r_1
            ECRIRE "r_2 = ", r_2
        FINSI
    FINSI
FIN
```

La comparaison des deux représentations précédentes montre qu'un algorigramme devient vite très encombrant. La version symbolique ci-après montre que l'on peut faire encore mieux.

Donnée : $(a, b, c) \in \mathbb{R}^3$ où l'on suppose $a \neq 0$
Début
- $\Delta = b^2 - 4ac$
- **Si** $\Delta < 0$ **:**
  - **Afficher** *"Pas de solution réelle."*
- **Sinon:**
  - **Si** $\Delta = 0$ **:**
    - $r = \dfrac{-b}{2a}$
    - **Afficher** *"Une solution réelle :"*
    - **Afficher** $"r = ", r$
  - **Sinon:**
    - $r_1 = \dfrac{-b - \sqrt{\Delta}}{2a}$ et $r_2 = \dfrac{-b + \sqrt{\Delta}}{2a}$
    - **Afficher** *"Deux solutions réelles :"*
    - **Afficher** $"r_1 = ", r_1$
    - **Afficher** $"r_2 = ", r_2$

En résumé, les algorigrammes sont un très bon outil quand l'on prend contact avec les algorithmes et la programmation mais ils montrent très vite leur limite quand on s'attaque à de « vrais » problèmes.