Algorithmique

1 Algorithme

1.1 Définitions

Le mot "algorithme" vient du nom de l'auteur persan Al-Khuwarizmi Mohammed (né vers 780 - mort vers 850) qui a écrit en langue arabe le plus ancien traité d'algèbre (de l'arabe "al jabar" qui signifie "la transposition" et du grec "arithmos" qui signifie "nombre") dans lequel il décrivait des procédés de calcul à suivre étape par étape pour résoudre des problèmes ramenés à des équations.

Un algorithme peut se définir comme étant l'ensemble des règles et instructions à suivre, effectivement exécutables, permettant d'obtenir un résultat clairement défini en un nombre fini d'étapes.

Il ne faut pas confondre "algorithme et "programme". Un algorithme peut s'exprimer avec une notation indépendante de tout langage de programmation alors qu'un programme est écrit dans un langage particulier compréhensible par une machine. Un algorithme n'est pas compréhensible par un ordinateur qui ne peut qu'exécuter un programme. De plus, un algorithme doit toujours donner une réponse après un nombre fini d'opérations, alors que l'exécution d'un programme peut conduire à une boucle infinie et ne jamais s'arrêter.

Un algorithme se compose de **données** et d'**instructions**.

Les données peuvent être de différents types, par exemple les types **nombre**, **texte** (chaînes de caractères), ou le type **logique** (vrai ou faux).

Les instructions à donner pour une exécution automatique doivent être suffisamment simples pour être traduites dans un langage de programmation.

1.2 Instructions élémentaires

1.2.1 Entrée, sortie

L'entrée (ou la lecture) des données peut se faire en interrogeant l'utilisateur ou par extraction à partir d'un fichier. La sortie (ou l'écriture) peut se faire par un affichage sur l'écran, une impression sur papier ou une écriture dans un fichier. Les résultats obtenus peuvent aussi rester en mémoire.

La sortie permet à l'utilisateur de voir les valeurs des variables après le traitement (ou en cours de traitement si on veut contrôler l'exécution du programme).

Ces instructions ne seront pas traitées dans le cours de mathématiques.

1.2.2 Affectation, calcul

L'affectation d'une donnée dans une variable consiste en la création d'une zone de mémoire (dans la machine) à laquelle on donne un nom et où on inscrit une valeur.

Ensuite diverses formules permettent d'effectuer des calcul à l'aide d'opérations élémentaires, par exemple pour déterminer la valeur d'une fonction pour une valeur donnée de la variable.

1.3 Instruction conditionnelle, boucle et itération

1.3.1 Instruction conditionnelle

On utilise l'instruction "Si ... alors ... " , (If ... then ...),
ou bien l'instruction "Si ... alors ... Sinon" (If ... Then ... Else).

```
Si <condition> alors
        ... Instructions A ...
Sinon
        ... Instructions B ...
```

1.3.2 Boucles itératives

On utilise l'instruction "Pour" (For). Cette instruction s'emploie pour répéter n fois une suite d'instructions, lorsque n est connu à l'avance (par exemple pour déterminer un tableau de n valeurs d'une fonction, pour le calcul de n termes d'une suite).

```
Pour i variant de 1 à 100 avec un pas de 1
    Début
    ... Instructions ...
    Fin
```

1.3.3 Boucles itératives conditionnelles

On utilise l'instruction "Tant que" (While). Cette instruction s'emploie pour répéter une suite d'instructions lorsque le nombre de répétitions est inconnu. La suite d'instructions est répétée tant qu'une certaine condition est vraie.

```
X ← ...
Début Tant que <condition sur X>
    ... Instructions ...
Fin Tant que
```

Attention à toujours vérifier que la boucle ne va pas se répéter sans fin. Par exemple, l'algorithme suivant n'est pas correct :

```
i ← 1
Début Tant que i > 0
    i ← i + 1
Fin Tant que
```

Et celui-ci ?

```
x ← 1
y ← x + 1
Début Tant que ( y − x = 1)
    x ← 10 × x
    y ← x + 1
Fin Tant que
```

Ce dernier algorithme ressemble au précédent.

Pourtant, si on le programme avec une calculatrice, le programme s'arrête et on obtient une valeur finale pour x (10^{14} ou autre suivant la calculatrice).

Programme Casio

```
1 → A
2 → B
While B − A = 1
10 × A → A
A + 1 → B
WhileEnd
A ◢
```

Programme Texas

```
1 → A
2 → B
While B − A = 1
10 × A → A
A + 1 → B
End
Disp A
```

Comment peut-on expliquer ces résultats ?

Ce programme va bien se terminer car dans une calculatrice, la taille des nombres est limitée. Il existe donc un grand nombre K, par exemple $K = 10^{14}$, tel que 1 est négligeable par rapport à K et alors $K + 1$ est égal à K (pour la calculatrice !).

Remarque : la suite de nombres ainsi construite, 1, 2, 10, 11, 20, 21, ... est croissante et majorée par K, donc elle est convergente.

2 Algorithmes et suites

2.1 Rang à partir duquel un terme est supérieur à un réel donné

On considère l'algorithme suivant où u et n sont deux variables dont les valeurs initiales sont respectivement 2000 et 0 :

```
Tant que u < 2500
    u ← u × 1,025
    n ← n + 1
Fin Tant que
```

On programme cet algorithme et on l'exécute.

1. Ecrire la suite des quatre premières valeurs de u et de n.
2. Déterminer les valeurs finales de u et de n.
3. Donner un problème dont la solution est donnée par cet algorithme.

2.2 Limite d'une suite arithmético-géométrique

On considère la suite (u_n) définie par $u_1 = \frac{3}{4}$ et $u_{n+1} = \frac{1}{2}u_n + \frac{1}{4}$. On peut démontrer que la suite (u_n) est décroissante et convergente de limite $\frac{1}{2}$. Expliquer ce que fait l'algorithme suivant, écrire le programme sur une calculatrice et donner les valeurs finales de u et n.

Les variables u et n ont pour valeurs initiales respectivement 0,75 et 1.

```
Tant que u − 0,5 > 0,0001
    u ← 0,5 × u + 0,25
    n ← n + 1
Fin Tant que
```

2.3 Calcul des termes d'une suite

On considère la suite (u_n) définie par $u_0 = 3$ et $u_{n+1} = \frac{1}{2}(u_n + \frac{2}{u_n})$. On démontre que cette suite est décroissante et minorée, donc convergente. Ecrire un algorithme qui permet de calculer les termes de u_1 jusqu'à u_{20}. Ecrire le programme sur une calculatrice et émettre une conjecture sur la limite de cette suite.