Rappels sur les suites - Algorithme

Table des matières

1 Suite : généralités

1.1 Définition

Définition 1 : Une suite (u_n) est une fonction définie de $\mathbb{N}$ (ou éventuellement $\mathbb{N} - [\![0,k]\!]$) dans $\mathbb{R}$. À un rang donné n, on associe un nombre réel noté u_n.

$$\begin{array}{rccc}(u_n) & : & \mathbb{N} \text{ ou } \mathbb{N}-[\![0,k]\!] & \longrightarrow & \mathbb{R} \\ & & n & \longmapsto & u_n\end{array}$$

Remarque :
- $\mathbb{N} - [\![0,k]\!]$ est l'ensemble $\mathbb{N}$ privé des premiers naturels jusqu'à k
- u_n est appelé le terme général de la suite (u_n).
- Bien faire la différence entre la suite noté (u_n) et le terme général noté u_n
- Si une suite est définie à partir du rang p, on la note $(u_n)_{n\geqslant p}$

Exemples :
- (u_n) : 2 ; 5 ; 8 ; 11 ; 14 ; 17 ; … suite arithmétique
- (v_n) : 3 ; 6 ; 12 ; 24 ; 48 ; 96 ; … suite géométrique

1.2 Exemples de suites

a) On peut définir une suite de **façon explicite** : $u_n = f(n)$

$$u_n = \frac{1}{n} \quad n \in \mathbb{N}^*, \quad v_n = \sqrt{n-3} \quad n \geqslant 3$$

b) On peut aussi définir une suite de **façon récurrente** à un ou plusieurs termes :

- À un terme : $u_{n+1} = f(u_n)$

$$\begin{cases} u_0 = 4 \\ u_{n+1} = 0,75u_n + 2 \end{cases}$$

(u_n) : 4 ; 5 ; 5,75 ; 6,3125 ; …

Pour calculer u_n, n étant donné

Variables : N, I entiers
U réel
Entrées et initialisation
Lire N
$4 \to U$ *on rentre u_0*
Traitement
pour I variant de 1 à N **faire**
$0,75U + 2 \to U$ *relation*
fin
Sorties : Afficher U

N	5	10	20	30
U	7,050 8	7,774 7	7,987 3	7,999 9

La suite semble croissante et converger vers 8

- À deux termes : $u_{n+2} = f(u_{n+1}, u_n)$

$$\begin{cases} u_0 = 1, \quad u_1 = 1 \\ u_{n+2} = u_{n+1} + u_n \end{cases}$$

(u_n) : 1 ; 1 ; 2 ; 3 ; 5 ; 8 ; 13 ; …

Pour calculer u_n, n étant donné

Variables : N, I entiers
U, V, W réels
Entrées et initialisation
Lire N
$1 \to V$ *on rentre u_0*
$1 \to U$ *on rentre u_1*
Traitement
pour I variant de 2 à N **faire**
$U + V \to W$ *relation*
$V \to U$, $W \to V$ *on passe au rang supérieur*
fin
Sorties : Afficher V

N	10	15	20	30
V	89	987	10 946	1 346 269

c) On peut encore définir une suite par l'intermédiaire d'une autre suite ou par une somme de termes, etc...

(u_n) étant définie, on définit la suite (v_n) par : $v_n = u_n - 4$

$$w_n = \sum_{i=1}^{n} \frac{1}{i} = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$$

Si on veut déterminer une valeur approchée d'un terme particulier de (w_n), on peut écrire le programme suivant :

Par exemple, on trouve les valeurs suivantes pour w_5, w_{10}, w_{50}.

Si l'on veut trouver le résultat exact en fraction avec la TI 82, écrire :
"Disp W ▷Frac"

On trouve les valeurs suivantes :

- $w_5 = \dfrac{137}{60} \simeq 2,283$
- $w_{10} \simeq 2,923$, $w_{50} \simeq 4,499$

Variables : N, I entiers
W réel
Entrées et initialisation
  Lire N
  $0 \to W$
Traitement
  pour I variant de 1 à N **faire**
    $W + \dfrac{1}{I} \to W$
  fin
Sorties : Afficher W

d) On peut aussi définir une suite par une assertion explicite sans pour autant être capable de préciser la valeur d'un terme quelconque.

Par exemple la suite (d_n) qui au rang $n \geqslant 1$ associe d_n la n ième décimale du nombre $\pi = 3,141\,592\ldots$: $d_1 = 1$, $d_2 = 4$, $d_3 = 1$, $d_4 = 5$, $d_5 = 9$, $d_6 = 2\ldots$

1.3 Variation ou monotonie d'une suite

Définition 2 : Soit (u_n) une suite numérique. On dit que :

- la suite (u_n) est strictement **croissante** (à partir d'un certain rang k) lorsque
$$u_{n+1} > u_n \quad \text{pour tout entier} \quad n \geqslant k$$
- la suite (u_n) est strictement **décroissante** (à partir d'un certain rang k) lorsque
$$u_{n+1} < u_n \quad \text{pour tout entier} \quad n \geqslant k$$
- la suite (u_n) est **monotone** (à partir d'un certain rang k) si elle est croissante ou décroissante à partir d'un certain rang k
- la suite (u_n) est **stationnaire** s'il existe un k tel que
$$u_{n+1} = u_n \quad \text{pour tout entier} \quad n \geqslant k$$
- la suite (u_n) est **constante** lorsque $u_{n+1} = u_n$ pour tout entier n du domaine de définition

Remarque :

Il existe des suites qui ne sont ni croissantes ni décroissantes : $u_n = (-1)^n$

Les premiers termes de la suite n'entrent pas nécessairement en compte dans la variation d'une suite. Ils peuvent cependant donner une indication pour la monotonie de la suite

1.4 Comment montrer la monotonie d'une suite

Règle 1 : Pour montrer la monotonie d'une suite,

- on étudie le signe de la quantité $u_{n+1} - u_n$
  si la quantité est positive (resp négative) à partir d'un certain rang k, la suite est croissante (resp décroissante) pour $n \geqslant k$
- si tous les termes de la suite sont strictement positifs à partir d'un certain rang k, on compare la quantité $\dfrac{u_{n+1}}{u_n}$ à 1
  si la quantité est supérieure à 1 (resp inférieure à 1) à partir d'un certain rang k, la suite est croissante (resp décroissante) pour $n \geqslant k$
- si la suite est définie de façon explicite, on étudie les variations de la fonction f sur $\mathbb{R}_+$
- (voir chapitre suivant) on utilise un raisonnement par récurrence

Exemples :

- Montrer que la suite (u_n) définie pour tout n par : $u_n = n^2 - n$ est croissante.
  Étudions le signe de la quantité : $u_{n+1} - u_n$

$$\begin{aligned} u_{n+1} - u_n &= (n+1)^2 - (n+1) - (n^2 - n) \\ &= n^2 + 2n + 1 - n - 1 - n^2 + n \\ &= 2n \end{aligned}$$

  Or pour tout $n \in \mathbb{N}$, on a $2n \geqslant 0$, donc $u_{n+1} - u_n \geqslant 0$. La suite (u_n) est croissante à partir du rang 0.

- Montrer que la suite (u_n) définie pour tout $n \in \mathbb{N}^*$ par : $u_n = \dfrac{2^n}{n}$ est croissante.

  Comme pour tout $n \in \mathbb{N}^* \quad u_n > 0$, comparons le rapport $\dfrac{u_{n+1}}{u_n}$ à 1 :

$$\frac{u_{n+1}}{u_n} = \frac{\dfrac{2^{n+1}}{n+1}}{\dfrac{2^n}{n}} = \frac{2^{n+1}}{n+1} \times \frac{n}{2^n} = \frac{2n}{n+1}$$

  Or $n \geqslant 1$, en ajoutant n de chaque côté de l'inégalité, $2n \geqslant n+1$, donc :

$$\frac{2n}{n+1} \geqslant 1$$

  Comme $\forall n \geqslant 1 \quad \dfrac{u_{n+1}}{u_n} \geqslant 1$, la suite (u_n) est croissante à partir du rang 1.

- Montrer que la suite (u_n) définie pour tout $n \geqslant 2$ par : $u_n = \dfrac{2n+1}{n-1}$ est décroissante.

  On étudie la fonction associée f définie sur $I = [2; +\infty[$ par $f(x) = \dfrac{2x+1}{x-1}$.
  Cette fonction est dérivable sur I, donc

$$f'(x) = \frac{2(x-1) - (2x+1)}{(x-1)^2} = \frac{-3}{(x-1)^2} \quad \text{donc} \quad f'(x) < 0 \quad x \in I$$

  La fonction f est donc décroissante sur I, donc la suite (u_n) est décroissante

1.5 Visualisation d'une suite

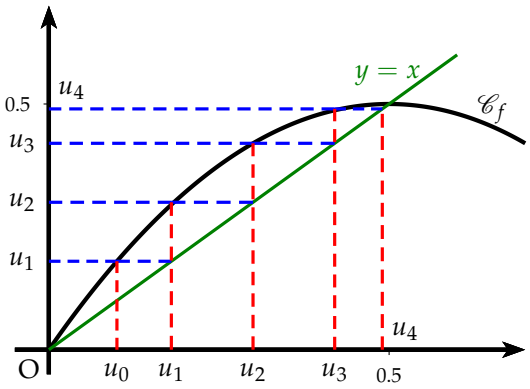
Pour visualiser une suite définie par récurrence $u_{n+1} = f(u_n)$, il suffit de tracer la courbe de la fonction associée f et la droite $y = x$. La droite sert à reporter les termes de la suite sur l'axe des abscisses.

Soit la suite (u_n) définie par :

$$\begin{cases} u_0 = 0,1 \\ u_{n+1} = 2u_n(1-u_n) \end{cases}$$

On obtient alors le graphe suivant, après avoir tracé la courbe $\mathscr{C}_f$ de la fonction f définie par :

$$f(x) = 2x(1-x)$$

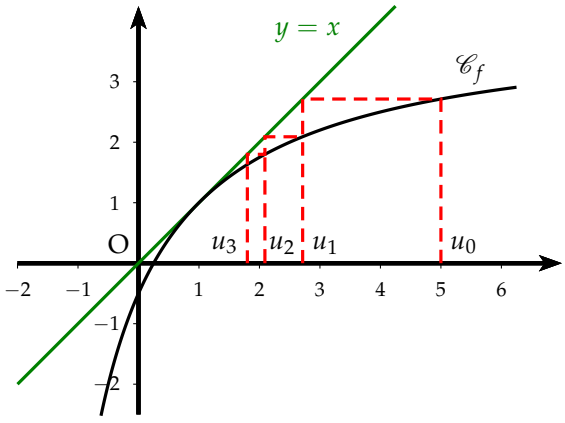


Exemple : Soit (u_n) la suite définie par :

$$\begin{cases} u_0 = 5 \\ u_{n+1} = \dfrac{4u_n - 1}{u_n + 2} \quad \forall n \in \mathbb{N} \end{cases}$$

f est la fonction définie sur l'intervalle $]-2\,;\,+\infty[$ par $f(x) = \dfrac{4x-1}{x+2}$.

Après avoir tracé la courbe $\mathscr{C}_f$ de la fonction f et la droite d'équation $y = x$, placer u_0 sur l'axe des abscisses, construire u_1, u_2 et u_3 en laissant apparents les traits de construction. Quelles conjectures peut-on émettre sur le sens de variation et sur la convergence de la suite (u_n) ?



D'après ce graphique, on peut conjecturer :

- que la suite est décroissante
- qu'elle semble converger vers 1 qui est l'abscisse du point d'intersection entre la droite et la courbe

Pour la représentation avec la TI 82, selectionner le mode "Suit" et le format "'Esc". Rentrer la suite puis règler la fenêtre. appuyer sur "graphe" puis sur "trace".

rentrer la suite avec la touche "$f(x)$"

$n_{min} = 0$

$u(n) = (4u(n-1)-1)/(u(n-1)+2)$

$u(n_{min}) = 5$

Pour la fenêtre :

$n_{min} = 0$, $n_{max} = 3$,

PremPoint = 1, pas = 1,

$X_{min} = -2$, $X_{max} = 6$, $X_{grad} = 1$

$Y_{min} = -2$, $Y_{max} = 4$, $Y_{grad} = 1$

2 Suite arithmétique (rappels)

2.1 Définition

Définition 3 : Une suite arithmétique (u_n) est définie par :

- un premier terme u_0 ou u_p
- une relation de récurrence : $u_{n+1} = u_n + r$ r étant la raison de la suite

Remarque : Une suite arithmétique correspond à une progression linéaire

Exemple : $(u_n) : \begin{cases} u_0 = 2 \\ u_{n+1} = u_n + 3 \end{cases}$ $(u_n) : 2 ; 5 ; 8 ; 11 ; 14 ; 17 ; \ldots$

2.2 Comment la reconnaît-on ?

Une suite (u_n) est arithmétique si la différence entre deux termes consécutifs est constante. Cette constante est alors la raison.

$\forall n \geqslant p \qquad u_{n+1} - u_n = r \quad \Leftrightarrow \quad (u_n)$ est une suite arithmétique de raison r

Exemple : Soit la suite (u_n) définie par $u_n = 4n - 1$. Montrer que la suite (u_n) est arithmétique.

$\forall n \in \mathbb{N}$, $u_{n+1} - u_n = 4(n+1) - 1 - (4n - 1) = 4$. La suite (u_n) est arithmétique.

2.3 Expression du terme général en fonction de n

Règle 2 : Soit (u_n) une suite arithmétique de raison r

- Si le premier terme est u_0, alors : $u_n = u_0 + nr$
- Si le premier terme est u_p, alors : $u_n = u_p + (n - p)r$

2.4 Somme des premiers termes

Théorème 1 : D'une façon générale, la somme des premiers termes d'une suite arithmétique obéit à :

$$S_n = \text{Nbre de termes} \times \frac{\Sigma \text{ termes extrêmes}}{2}$$

$S_n = 1 + 2 + 3 + \cdots + n$ alors $S_n = \dfrac{n(n+1)}{2}$

$S_n = u_0 + u_1 + u_2 + \cdots + u_n$ alors $S_n = (n+1)\left(\dfrac{u_0 + u_n}{2}\right)$

Exemple : Calculer la somme des termes suivants :

$$S = 8 + 13 + 18 + \cdots + 2008 + 2013$$

Il s'agit de la somme des termes d'une suite arithmétique de raison 5 et de termes extrêmes 8 et 2013.
Le nombre de termes est : $\dfrac{2013 - 8}{5} + 1 = 402$. (Règle des piquets et des intervalles)

$$S = 402 \times \frac{8 + 2013}{2} = 406\,221$$

Algorithme : Vérification

On écrit l'algorithme suivant permettant de calculer la somme S.

Comme la suite arithmétique est croissante, on peut proposer la boucle conditionnelle ci-contre. On remarquera que le critère d'arrêt est $u \leqslant 2008$ et non $u \leqslant 2013$ car quand on finit la boucle avec ce critère on calcule le terme suivant soit 2013 que l'on ajoute à la somme S.

On retrouve bien le résultat calculé.

Variables : I entier U, S réels
Entrées et initialisation
  $8 \to U$
  $8 \to S$
Traitement
  tant que $U \leqslant 2008$ **faire**
    $U + 5 \to U$
    $S + U \to S$
  fin
Sorties : Afficher S

3 Suite géométrique (rappels)

3.1 Définition

Définition 4 : Une suite géométrique (u_n) est définie par :

- un premier terme u_0 ou u_p
- une relation de récurrence : $u_{n+1} = q \times u_n$ q étant la raison de la suite

Remarque : Une suite géométrique correspond à une progression exponentielle.

Exemple : $(u_n) : \begin{cases} u_0 = 3 \\ u_{n+1} = 2u_n \end{cases}$ $(u_n) : 3\,;\,6\,;\,12\,;\,24\,;\,48\,;\,96\,;\,\ldots$

3.2 Comment la reconnaît-on ?

Une suite est géométrique si le quotient entre deux termes consécutifs est constant. Cette constante est alors la raison.

$$\forall n \geqslant p \qquad \frac{u_{n+1}}{u_n} = q \quad \Rightarrow \quad (u_n) \text{ est une suite géométrique de raison } q$$

Exemple : Soit la suite $(u_n) : \begin{cases} u_0 = 5 \\ u_{n+1} = 3u_n - 2 \end{cases}$

On pose la suite $v_n = u_n - 1$. Montrer que la suite (v_n) est géométrique.

$$v_{n+1} = u_{n+1} - 1 = 3u_n - 2 - 1 = 3(u_n - 1) = 3v_n \quad \text{donc } \forall n \in \mathbb{N},\ \frac{v_{n+1}}{v_n} = 3$$

La suite (v_n) est géométrique de raison 3 et de premier terme $v_0 = 4$.

3.3 Expression du terme général en fonction de n

Règle 3 : Soit (u_n) une suite géométrique de raison q

- Si le premier terme est u_0, alors : $u_n = q^n \times u_0$
- Si le premier terme est u_p, alors : $u_n = q^{n-p} \times u_p$

3.4 Somme des premiers termes

Théorème 2 : D'une façon générale, la somme des premiers termes d'une suite géométrique ($q \neq 1$) obéit à :

$$S_n = 1^{\text{er}} \text{ terme} \times \frac{1 - q^{\text{nbre termes}}}{1 - q}$$

$$S_n = u_0 + u_1 + u_2 + \cdots + u_n \quad \text{alors} \quad S_n = u_0 \times \frac{1 - q^{n+1}}{1 - q}$$

Exemple : Calculer la somme des termes suivants :

$$S = 0,02 - 0,1 + 0,5 - 2,5 + \cdots + 312,5$$

Il s'agit de la somme des termes d'une suite géométrique (u_n) de raison $q = -5$ de premier terme $u_0 = 0,02$.

On doit avoir : $u_n = 0,02(-5)^n = 312,5 \Rightarrow (-5)^n = \dfrac{312,5}{0,02} = 15\,625 = 5^6$.

Il y a alors 7 termes. $S = 0,02 \times \dfrac{1 - (-5)^7}{1 - (-5)} = \dfrac{5^7 + 1}{300} = 260,42$

3.5 Limite d'une suite géométrique

Théorème 3 : Soit une suite (u_n) définie par : $u_n = q^n$ (avec $q \in \mathbb{R}$)

- Si $q > 1$ alors (u_n) est divergente et $\lim\limits_{n \to +\infty} q^n = +\infty$
- Si $q = 1$ alors (u_n) est constante (donc convergente vers 1)
- Si $-1 < q < 1$ alors (u_n) est convergente et $\lim\limits_{n \to +\infty} q^n = 0$
- Si $q \leqslant -1$ alors (u_n) est divergente et n'a pas de limite

Exemples :

- (u_n) suite géométrique : $u_0 = -2$ et $q = 1,5$. La suite (u_n) converge-t-elle ?
  $\lim\limits_{n \to +\infty} 1,5^n = +\infty$ car $1,5 > 1$. La suite (u_n) diverge et $\lim\limits_{n \to +\infty} u_n = -\infty$.
- (v_n) suite géométrique : $v_0 = 4$ et $q = \dfrac{3}{4}$. La suite (v_n) converge-t-elle ?
  $\lim\limits_{n \to +\infty} \left(\dfrac{3}{4}\right)^n = 0$ car $-1 < \dfrac{3}{4} < 1$. La suite (u_n) converge vers 0.

4 Algorithme

4.1 Introduction

Définition 5 : Un algorithme est une suite d'instructions, qui une fois exécutée correctement, conduit à un résultat donné.
Pour fonctionner, un algorithme doit donc contenir uniquement des instructions compréhensibles par celui qui devra l'exécuter.

La maîtrise de l'algorithmique requiert deux qualités :

- il faut avoir une certaine **intuition**, car aucune recette ne permet de savoir à priori quelles instructions permettront d'obtenir le résultat voulu.
- il faut être **méthodique et rigoureux**. En effet, chaque fois qu'on écrit une série d'instructions qu'on croit justes, il faut systématiquement se mettre à la place de la machine qui va les exécuter, pour vérifier si le résultat obtenu est bien celui que l'on voulait.

Les ordinateurs, quels qu'ils soient, sont fondamentalement capables de comprendre quatre catégories d'ordres seulement (en programmation, on n'emploiera pas le terme d'ordre, mais plutôt celui d'instructions). Ces quatre familles d'instructions sont :

- l'affectation de variables
- la lecture / écriture
- les tests
- les boucles

Un algorithme exprime les instructions résolvant un problème donné indépendamment des particularités de tel ou tel langage de programmation.

Apprendre un algorithme, c'est apprendre à manier la structure logique d'un programme informatique.

« Un langage de programmation est une convention pour donner des ordres à un ordinateur.» (Pascal, Fortran, C++, PHP, Mathematica etc.)

4.2 Conventions pour écrire un algorithme

Historiquement, plusieurs types de notations ont représenté des algorithmes.

- Il y a eu notamment **une représentation graphique**, avec des carrés, des losanges, etc. qu'on appelait des organigrammes. Cependant dès que l'algorithme commence à grossir un peu, ce n'est plus pratique.
- On utilise généralement une série de conventions appelée « **pseudo-code** », qui ressemble à un langage de programmation authentique dont on aurait évacué la plupart des problèmes de syntaxe. Ce pseudo-code est susceptible de varier légèrement d'un livre (ou d'un enseignant) à un autre. C'est bien normal : le pseudo-code, encore une fois, est purement conventionnel ; aucune machine n'est censée le reconnaître.

4.3 Les variables

4.3.1 Définition

Définition 6 : Dès que l'on a besoin de stocker une information au cours d'un programme, on utilise une **variable**.

Pour employer une image, une variable est une boîte, que le programme (l'ordinateur) va repérer par une étiquette. Pour avoir accès au contenu de la boîte, il suffit de la désigner par son étiquette.

4.3.2 Déclaration des variables

La première chose à faire avant de pouvoir utiliser une variable est de créer la boîte et de lui coller une étiquette. C'est ce qu'on appelle la déclaration des variables.

Le nom de la variable (l'étiquette de la boîte) obéit à des impératifs changeant selon les langages. Toutefois, une règle absolue est qu'un nom de variable peut comporter des lettres et des chiffres, mais qu'il exclut la plupart des signes de ponctuation, en particulier les espaces. Un nom de variable correct commence également impérativement par une lettre. Quant au nombre maximal de signes pour un nom de variable, il dépend du langage utilisé.

Une fois le nom choisi, il faut déterminer le type de la variable. On peut distinguer 3 types de variable (il y en a d'autres !) :

- Type alphanumérique : du texte.
- Type numérique : un nombre (entier, décimal, réel).
- Type liste : ensemble de nombres ordonné.

4.4 Affectation d'une variable numérique

La seule chose qu'on puisse faire avec une variable, c'est l'affecter, c'est-à-dire lui attribuer une valeur.

$\boxed{a \leftarrow 24}$	Attribue la valeur 24 à la variable a	$\boxed{24 \rightarrow a}$
$\boxed{a \leftarrow b}$	Attribue la valeur de b à la variable a	$\boxed{b \rightarrow a}$

On peut aussi affecter une variable à l'aide d'une opération :

$\boxed{c \leftarrow a+4}$	Attribue la valeur $a+4$ à la variable c	$\boxed{a+4 \rightarrow a}$

On peut changer la valeur d'une variable avec elle-même :

$\boxed{b \leftarrow b+1}$	Incrémente (augmente) de 1 la variable b	$\boxed{b+1 \rightarrow b}$

Les opérateurs numériques sont :

- L'addition $+$
- La soustraction $-$
- La multiplication $*$

- La division /
- La puissance ^

Les opérateurs logiques sont :

- ET intersection de deux ensembles
- OU (non exclusif) union de deux ensembles
- NON complémentaire d'un ensemble

⚠ Remarque : :

- En informatique une variable ne peut prendre à un moment donné qu'une valeur et une seule, contrairement à une équation qui peut éventuellement avoir plusieurs solutions.
- Souvent l'affectation d'une variable se note : $B = A$ ou $B := A$ qui veut dire que B prend la valeur A, qui est alors différent de $A = B$ ou $A := B$ où A prend la valeur de B.

4.5 Lecture et écriture d'une variable

Définition 7 : **Lire** ou **Entrer** une variable signifie que l'utilisateur doit rentrer une valeur pour que le programme la lise.

Ecrire ou **Afficher** une variable signifie que le programme renvoie la valeur de la variable que le programme a trouvée.

4.6 Les tests

Il y a deux formes pour un test : soit la forme complète, soit la forme simplifiée :

Forme complète	**Forme simplifiée**
Si condition **alors** Instructions 1 **Sinon** Instructions 2 **FinSi**	**Si** condition **alors** Instructions **FinSi** Si la condition n'est pas vérifiée le programme ne fait rien.

La condition portant sur une variable peut être :

- Une valeur à atteindre.
- Une comparaison avec une autre variable (égalité, inégalité, non égalité).

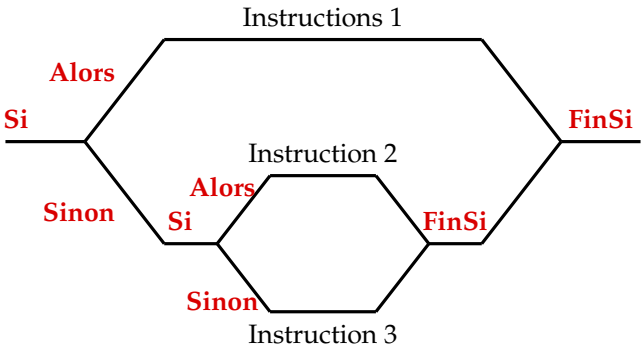
On peut aussi mettre un test qui se décompose en plusieurs conditions reliées par un opérateur logique :

- condition 1 ET condition 2 : les deux conditions doivent être vérifiées en même temps.
- condition 1 OU condition 2 : au moins l'une des deux conditions doit être vérifiée.

On peut aussi imbriquer un ou plusieurs autres tests à l'intérieur d'un test. On a alors le schéma suivant :

```
Si condition 1 alors
    Instructions 1
Sinon Si condition 2 alors
        Instructions 2
      Sinon
        Instruction 3
      FinSi
FinSi
```

On pourrait schématiser cette situation par :



C'est par exemple le cas lorsque l'on cherche les solutions de :

$$Ax^2 + Bx + C = 0$$

On peut écrire le programme ci-contre.

On peut observer que l'on a d'abord un "Si" simplifié pour tester si l'équation est du premier degré ou non. Si cette condition est vérifiée alors le programme est terminé. On lui demande d'aller à la fin sans poursuivre le programme.

Ensuite, on a deux "Si" imbriqués pour tester les trois cas du signe de Δ.

On peut remarquer que lorsque l'on veut afficher du texte, il faut mettre celui-ci entre guillemets.

Variables : A, B, C, X, Y, Δ réels
Entrées et initialisation
  Lire A, B, C
  $B^2 - 4AC \to \Delta$
Traitement et sorties
  si $A = 0$ **alors**
    $-\dfrac{C}{B} \to X$
    Afficher X
    Aller à fin du programme
  fin
  si $\Delta > 0$ **alors**
    $\dfrac{-B + \sqrt{\Delta}}{2A} \to X$
    $\dfrac{-B - \sqrt{\Delta}}{2A} \to Y$
    Afficher X, Y
  sinon
    si $\Delta = 0$ **alors**
      $-\dfrac{B}{2A} \to X$
      Afficher X
    sinon
      Afficher "Pas de Solution"
    fin
  fin

4.7 Les boucles

4.7.1 Définition

Définition 8 : Une **boucle** est une structure répétitive ou itérative, c'est à dire que la boucle effectue n fois un calcul sous le contrôle d'une condition d'arrêt.

4.7.2 La boucle conditionnelle

La boucle conditionnelle obéit au schéma suivant :

```
Tant que condition
    Instructions
FinTantque
```

⚠ Dans le cas où la condition est toujours vérifiée, l'ordinateur tourne alors dans la boucle et n'en sort plus. La « **boucle infinie** » est l'une des hantises les plus redoutées des programmeurs.
Cette faute de programmation est courante chez tout apprenti programmeur.

Exemple :

On veut trouver l'entier N pour que :

$$1 + 2 + \cdots + N > 10^P$$

On fait une boucle contrôlée par la condition $< 10^P$ pour trouver la somme des N premiers naturels en incrémentant N pour compter le nombre de boucles effectuées.

- Si $P = 3$, on trouve $N = 45$
- Si $P = 6$, on trouve $N = 1414$

Variables : N, P entiers
U réel
Entrées et initialisation
Lire P
$0 \to N$
$0 \to U$
Traitement
tant que $U \leqslant 10^P$ **faire**
$N + 1 \to N$
$U + N \to U$
fin
Sorties : Afficher N

4.7.3 Boucler en comptant

Si l'on connaît à l'avance le nombre d'incrémentation, on a alors la structure suivante :

Pour compteur = initial **à** final **par** valeur du pas
Instructions
FinPour

Exemple :

On veut programmer $K!$ "factorielle K" :
$N = K! = 1 \times 2 \times 3 \times \cdots \times K$
On incrémente alors un compteur I (par défaut le pas est égal à 1) allant de 2 à K
On obtient alors $N = K!$

- Si $K = 8$, on trouve $N = 40\,320$
- Si $K = 12$, on trouve :
  $N = 479\,001\,600$

Variables : K, I, N entiers
Entrées et initialisation
Lire K
$1 \to N$
Traitement
pour I de 1 à K **faire**
$N \times I \to N$
fin
Sorties : Afficher N