Algorithmique

0. Introduction

■ DÉFINITION :

Un **algorithme** est une description structurée de la liste des actions effectuées par un programme.

Remarque : Il peut être représenté graphiquement, c'est alors un organigramme (ou algorigramme), ou écrit sous une forme littérale, appelée langage algorithmique. On l'appellera « l'algorithme » pour simplifier.

Attention

L'organigramme, qui n'est pas exigible, il permet cependant dans certains cas de mieux comprendre certaines procédures à effectuer.

1. Lecture et écriture

Cette partie correspond aux transferts d'informations entre l'utilisateur et la machine.

■ DÉFINITION :

La **lecture** est la saisie (transfert de l'utilisateur vers la machine), à l'aide du clavier.

■ DÉFINITION :

L'**écriture** est l'affichage (transfert de la machine vers l'utilisateur), à l'aide de l'écran.

2. Calculs et affectation des variables

■ DÉFINITION :

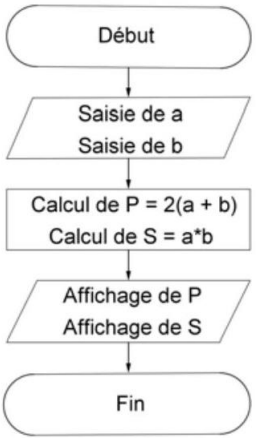
Les **calculs** correspondent au traitement interne à la machine.

■ DÉFINITION :

L'**affectation des variables** est le stockage de valeurs numériques (ou autres...), repérées par leur nom.

Exemple :

Calcul du périmètre et de la superficie d'un rectangle connaissant ses dimensions.



Exercice 1

Construire un organigramme et un algorithme (en langage naturel) permettant de calculer l'aire d'un disque connaissant son périmètre.

Exercice 2

Dans l'algorithme ci-dessous, montrer que l'on pourrait exprimer y directement en fonction de x avec une instruction ne comportant qu'une seule opération.

```
Lire x
a ⟵ x + 2
b ⟵ a * a
c ⟵ x - 2
d ⟵ c * c
y ⟵ (b - d)/4
Afficher y
```

Exercice 3

Construire l'organigramme et l'algorithme du problème suivant :

Calculer le prix HT d'un objet connaissant son prix TTC (le taux de TVA étant variable et donné par l'utilisateur).

Exercice 4

Écrire un algorithme permettant d'échanger les valeurs de deux variables.

Exercice 5

Un algorithme contient dans cet ordre les instructions suivantes :

« a prend la valeur 5 »,
« b prend la valeur 2 »,
« a prend la valeur b »,
« b prend la valeur a ».

Déterminer les valeurs finales de a et de b et faire une remarque concernant l'une des instructions.

Exercice 6

Écrire l'algorithme suivant et commenter le résultat : choisir un entier, calculer le carré de l'entier suivant, puis le carré de l'entier précédent, puis l'écart de ces deux carrés, et enfin le quotient de ce dernier résultat par l'entier de départ.

3. Tests

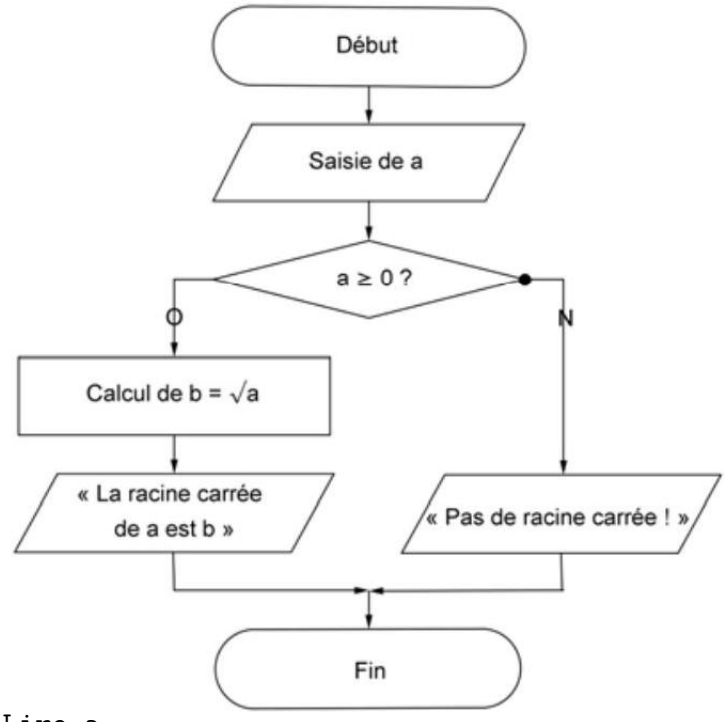
■ DÉFINITION :

Le test « SI... ALORS... [SINON] » permet un choix ayant deux réponses possibles (VRAI ou FAUX), avec des instructions à réaliser si la réponse est VRAI, et éventuellement si la réponse est FAUX, par l'intermédiaire d'un « SINON ».

Exemple : Calcul de la racine carrée d'un réel positif, et sans réaction lorsqu'il est négatif.

```
Lire a
Si (a ⩾ 0) Alors
  b ⟵ √a
  Afficher "La racine carrée de a est : "
  Afficher b
```

Exemple : Calcul de la racine carrée d'un réel positif, et affichage d'un message d'erreur lorsqu'il est négatif.




```
Lire a
Si (a ⩾ 0) Alors
  b ⟵ √a
  Afficher "La racine carrée de a est :"
  Afficher b
Sinon
  Afficher "Pas de racine carrée !"
```

Exemple :

Voici un algorithme du calcul de la valeur absolue d'un réel.
Réécrire cet algorithme sans utiliser l'instruction « Sinon ».

```
Lire a
Si (a > 0) Alors
  b ⟵ a
Sinon
  b ⟵ -a
Afficher "La valeur absolue de "
Afficher a
Afficher " est : "
Afficher b
```

Exercice 7

Construire un algorithme indiquant le plus grand de deux réels donnés par l'utilisateur. Prévoir un message si les deux nombres sont égaux.

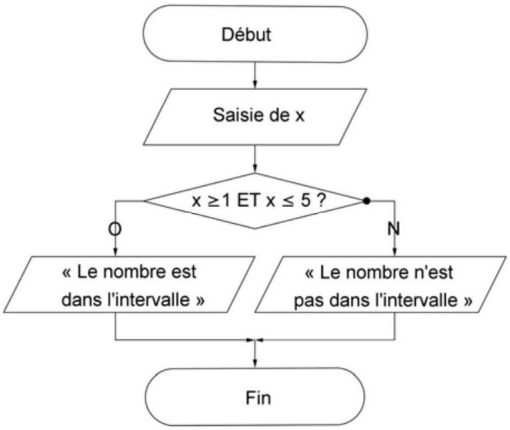
Exercice 8

Écrire un algorithme saisissant cinq réels donnés par l'utilisateur et indiquant, au fur et à mesure, le plus grand des réels saisis.

4. Tests avec connecteurs

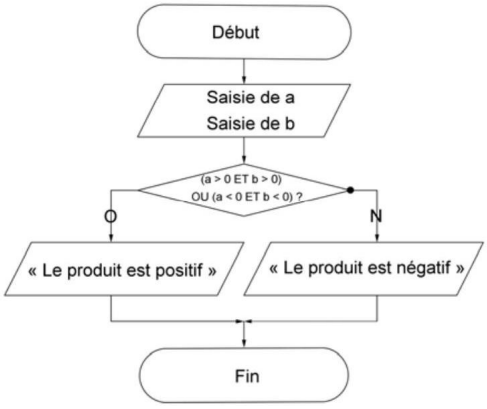
Remarque : Si A et B sont deux affirmations pouvant être vraies ou bien fausses, alors l'affirmation (A ET B) est vraie si et seulement si les deux affirmations sont vraies simultanément, et (A OU B) est vraie si et seulement si au moins l'une des deux affirmations est vraie.

Exemple : Déterminer si un nombre appartient à l'intervalle $[1\,;\,5]$.




```
Lire x
Si ((x ⩾ 1) ET (x ⩽ 5)) Alors
  Afficher "Le nombre est dans
l'intervalle"
Sinon
  Afficher "Le nombre n'est pas dans
l'intervalle"
```

Exemple : Détermination du signe du produit de deux nombres non nuls sans faire le calcul du produit.




```
Lire a
Lire b
Si ((a>0 ET b>0) OU (a<0 ET b<0)) Alors
  Afficher "Le produit est positif"
Sinon
  Afficher "Le produit est négatif"
```

Exercice 9

Écrire un algorithme permettant de vérifier si le carré du réel a est supérieur ou égal à 1, sans calculer effectivement le carré.

Exercice 10

Écrire un algorithme saisissant trois réels donnés par l'utilisateur et indiquant s'ils ont été saisis dans l'ordre croissant, décroissant ou dans le désordre.

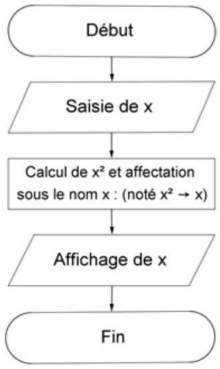
Exercice 11

Une année est bissextile lorsqu'elle est divisible par 4, sans être divisible par 100, sauf si elle est divisible par 400. Écrire un algorithme indiquant si une année, saisie par l'utilisateur, est bissextile ou non.

5. Boucle « Pour »

Remarque : On peut faire un calcul avec la valeur d'une variable, et affecter le résultat sous le même nom de variable (utile pour les suites définies par récurrence !).

Exemple : Calcul du carré d'un réel




```
Lire x
x ⟵ x * x
Afficher x
```

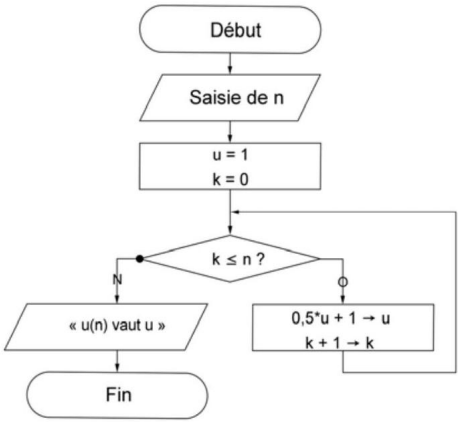
Exercice 12

Écrire un algorithme permettant d'effectuer trois fois de suite (sans utiliser de boucle) le calcul suivant : multiplier par 2 et ajouter 1, en utilisant une seule variable. Refaire l'algorithme, toujours avec une seule variable, et un seul calcul.

Exemple :

Détermination du terme u_n d'une suite (u_n) définie par récurrence : $u_0 = 1$ et, pour tout n,

$$u_{n+1} = 0,5u_n + 1$$

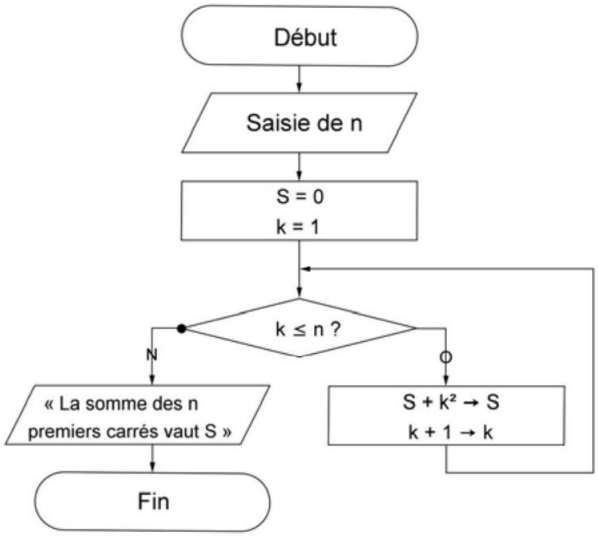



```
LIRE n
u ⟵ 1
Pour k allant de 1 à n
  u ⟵ 0.5 * u + 1
Afficher "u("
Afficher n
Afficher ") = "
Afficher u
```

Remarque : Pour traduire exactement cet organigramme en langage algorithmique, il faudrait effectuer un RENVOI avant le test, ce qui est faisable (instruction « Aller à » mais inutilement compliqué.
De plus, puisque la variable k prend successivement **toutes** les valeurs entières de 1 à n, on peut traduire plus simplement cet organigramme en langage algorithmique grâce à une boucle « Pour... De... À », qui utilise successivement les valeurs de la variable citée de la première à la dernière, avec un pas de 1, c'est-à-dire en ajoutant à chaque fois 1 à la variable, et sort de la boucle lorsque la dernière valeur est dépassée.

Remarque : À la sortie de la boucle, k vaut $n + 1$.

Exemple : Calcul de la somme des carrés des entiers entre 1 et n, entier donné par l'utilisateur

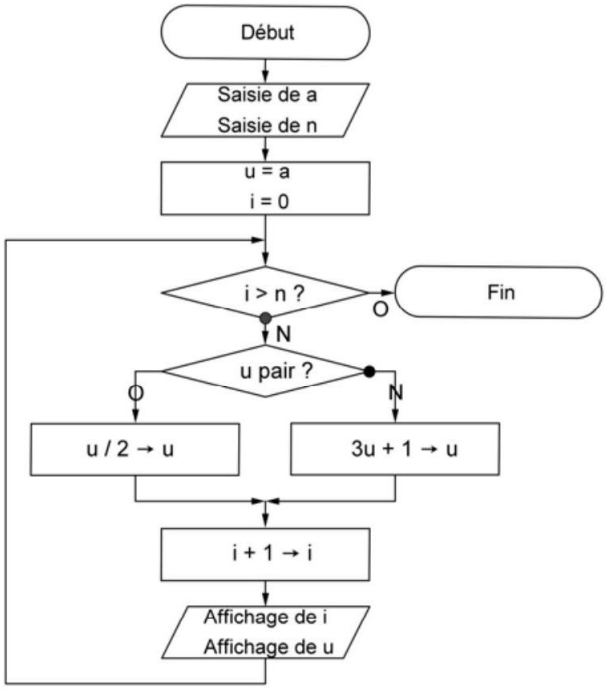



```
LIRE n
S ⟵ 0
Pour k allant de 1 à n
  S ⟵ S + k * k
Afficher "La somme des "
Afficher n
Afficher " premiers carrés vaut : "
Afficher S
```

Exemple : Détermination des premières valeurs de la suite dite *de Syracuse*
On considère la *suite de Syracuse de premier terme* $a \in \mathbb{R}$, définie par

$$\begin{cases} u_0 = a \\ \forall n \in \mathbb{N}, u_{n+1} = \begin{cases} \dfrac{u_n}{2} & \text{si } u_n \text{ est pair} \\ 3u_n + 1 & \text{si } u_n \text{ est impair} \end{cases} \end{cases}$$

L'algorithme suivant permet de calculer le $n$ième terme de cette suite :




```
Lire a
Lire n
u ⟵ a
Pour i allant de 1 à n
  Si (u%2 == 0) Alors
    u ⟵ u / 2
  Sinon
    u ⟵ 3 * u + 1
Afficher " -> "
Afficher u
```

Remarque : « u % 2 » renvoie le reste de la division de u par 2. S'il vaut 0 , c'est que u est un entier pair. Dans plusieurs exemples, les affichages finaux sont en dehors de la boucle car on affiche seulement le dernier terme. En revanche, parfois, les affichages sont dans la boucle car les valeurs sont affichées pour chaque valeur de n.

Exercice 13

Expliquer ce que fait l'algorithme ci-dessous, c'est-à-dire ce que représente P pour les nombres a et b.

```
Lire a
Lire b
P ⟵ 0
Pour k allant de 1 à a
  P ⟵ P + b
Afficher "P = "
Afficher P
```

Exercice 14

Écrire un algorithme calculant le produit de tous les entiers entre n et $2n$, où n est un entier positif donné par l'utilisateur.

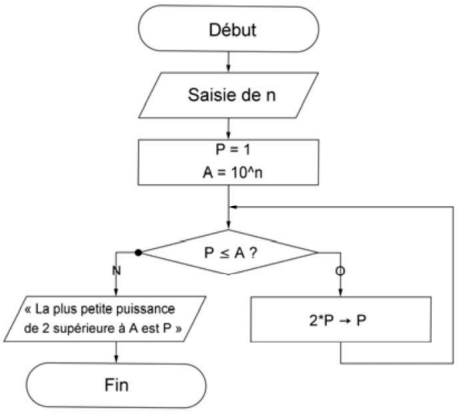
Exercice 15

Écrire un algorithme calculant le $n$ième terme de la suite de Fibonacci, définie par :

$$\begin{cases} u_0 = 1 \\ u_1 = 1 \\ \forall n \in \mathbb{N}, u_{n+2} = u_{n+1} + u_n \end{cases}$$

6. Boucle « Tant que »

Exemple : Détermination de la plus petite puissance de 2 supérieure à 10^n, l'entier n étant donné par l'utilisateur.




```
Lire n
P ⟵ 1
A ⟵ 10^n
Tant que (P<=A)
  P ⟵ 2*P
Afficher "La plus petite puissance de 2
supérieure à "
Afficher A
Afficher " est : "
Afficher P
```

Remarque : Cet organigramme ressemble beaucoup à ceux de certains exemples précédents, mais ici le nombre de tests effectué n'est pas connu à l'avance, donc on ne peut faire une « boucle POUR ». On utilise alors une boucle « TANT QUE », qui effectue des instructions tant que certaines conditions sont vérifiées, et qui s'arrête lorsque ce n'est plus le cas.

Exemple : Méthode de dichotomie pour déterminer un encadrement d'amplitude 10^{-p} de la solution dans $[1;2]$ de l'équation $x^2 = 2$.

```
a ⟵ 1
b ⟵ 2
Lire p
Tant que (b − a > 10^−p)
  m ⟵ (a+b) / 2
  Si ((m*m-2) * (b*b-2) > 0) Alors
    b ⟵ m
  Sinon
    a ⟵ m
Afficher a
Afficher " < solution < "
Afficher b
```

Exercice 16

Écrire un algorithme donnant la liste de tous les entiers positifs dont le cube est inférieur ou égal à un réel donné, grâce à une boucle « Tant que ».

Exercice 17

Écrire un algorithme calculant le nombre d'années nécessaire pour qu'une commune de 300 000 individus n'ait plus que 50 000 individus, sachant qu'elle perd 10 % de sa population par an.

Exercice 18

On appelle « partie entière » d'un réel le plus grand entier inférieur ou égal à ce réel. Écrire un algorithme calculant la partie entière d'un réel positif.

Remarque : En langage naturel, la partie entière de x se note Partie_Entière(x)

Exercice 19

Écrire un algorithme donnant le nombre d'entiers entre deux réels donnés.