Introduction à l'algorithmique

DAKKAR Borhen-eddine

Lycée le Corbusier

BTS SN-IR

Table des matières

Table des matières

Qu'est ce qu'un algorithme?

- Un algorithme est un programme informatique qui représente une procédure de calcul.
- Un algorithme prend en entrée une valeur et donne en sortie une valeur.

- Un algorithme est défini comme une séquence d'instructions étape par étape qui décrit comment un calcul doit être effectué.
- Avant qu'un programme ne soit écrit, le programmeur doit avoir une compréhension claire du résultat souhaité et comment le programme proposé doit le produire.

Qu'est ce qu'un algorithme?

Pour illustrer le concept d'un algorithme, nous considérerons les exemples suivants:

Exemple 1

Nous cherchons un Algorithme qui calcule la somme de tous les nombres entiers de 1 à 100.

Exemple 2

Nous cherchons un algorithme qui prend en Entrée une suite de n nombres : $< N_1, N_2, ..., N_n >$ et nous donne comme Sortie une suite organisée de façon que : $N'_1 < N'_2 < ...N'_n >$.

Qu'est ce qu'un algorithme?

Revenons à notre premier exemple, nous pouvons resoudre ce problème en suivant ces deux méthodes :

Méthode 1

Nous allons organiser les nombres de 1 à 100 dans une colonne et faites leurs somme:

$$
\begin{array}{rr}
 & 1 \\
 & 2 \\
 & 3 \\
 & . \\
 & . \\
 & . \\
 & 98 \\
 & 99 \\
+ & 100 \\
\hline
= & 5050
\end{array}
$$

Qu'est ce qu'un algorithme?

Méthode 2

Nous allons utiliser la formule suivante:

$$somme = \frac{n * (a + b)}{2}$$

avec:

- n : le nombre de termes à ajouter.
- a : le premier nombre à ajouter.
- b : le dernier nombre à ajouter.

L'application numérique de cette formule est donne:

$$somme = \frac{100(1 + 100)}{2} = 5050 \quad (1)$$

Qu'est ce qu'un algorithme?

- Les deux précédentes méthodes paraissent intuitives pour nous. Ce qui n'est pas le cas pour un ordinateur.
- Une déclaration telle que "ajouter les nombres de 1 à 100" n'est pas compréhensible pour un ordinateur.
- Pour programmer un ordinateur, nous devons communiquer avec lui en utilisant un language conventionnel.
- Un ordinateur est une machine «répondant à un algorithme»; ce n'est pas une machine à "réponse intuitive".
- Un ordinateur à besoins d'un ensemble d'instructions étape par étape qui, collectivement, forment un algorithme.

Qu'est ce qu'un algorithme?

Afin qu'un ordinateur puisse exécuter notre algorithme. Il faut lui préciser les étapes à suivre:

- **Étape1:** Définir n égale à 100;
- **Étape2:** Définir a égale à 1;
- **Étape3:** Définir b égale à 100;
- **Étape4:** Calculer la

$$somme = \frac{n * (a + b)}{2};$$

- **Étape5:** Afficher la valeur somme;

Qu'est ce qu'un algorithme?

- Un programme, doit être écrit dans une langue comprise par l'ordinateur.
- Un algorithme peut être écrit ou décrit de différentes manières.
- Nous appelons un organigramme les images qui emploient des formes spécifiquement définies pour la description d'un algorithme.
- Nous appelons un pseudocode les phrases utilisées pour pour décrire un algorithme (les étapes de traitement).

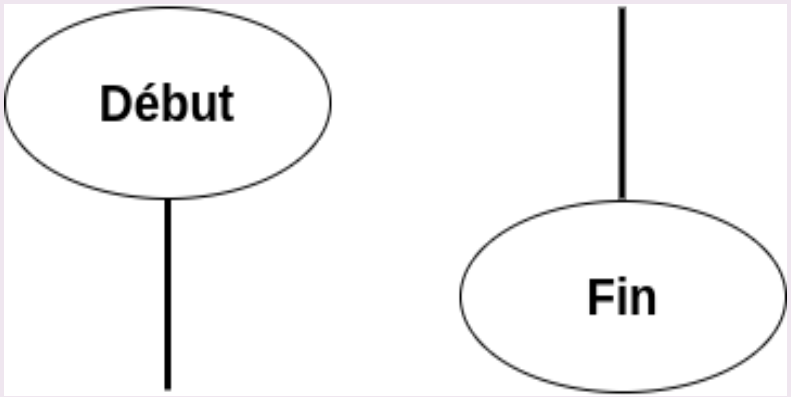
Table des matières

Les éléments d'un organigramme

- Un Organigramme (logigramme, algorigramme) est une représentation graphique d'un algorithme à l'aide des symboles.
- Il représente l'enchaînement des opérations et des décisions effectuées par un algorithme.
- Les symboles utilisés dans un organigrammes sont:

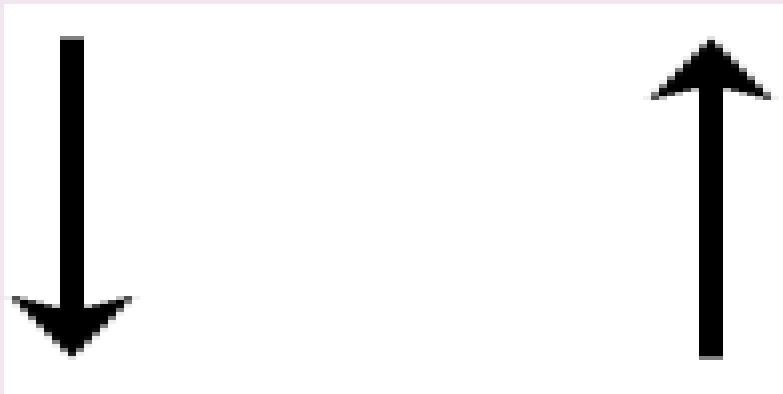
Tout algorithme à un début et une fin.

Début/Fin



Les flèches indiquent le sens de l'algorithme.

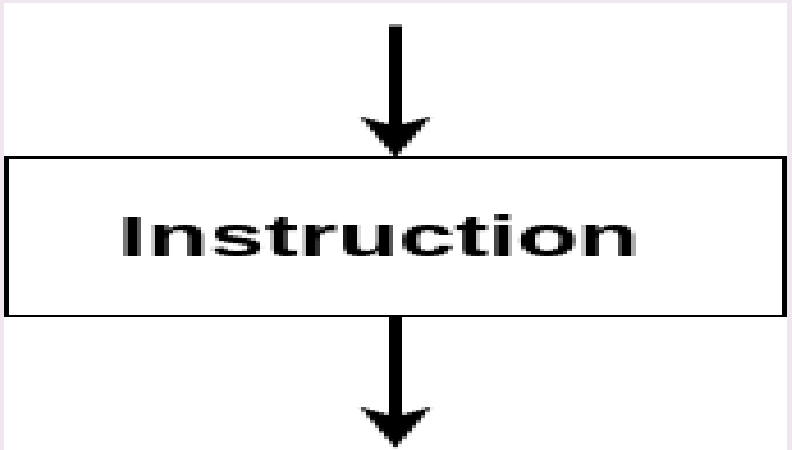
Les flèches



Les éléments d'un organigramme

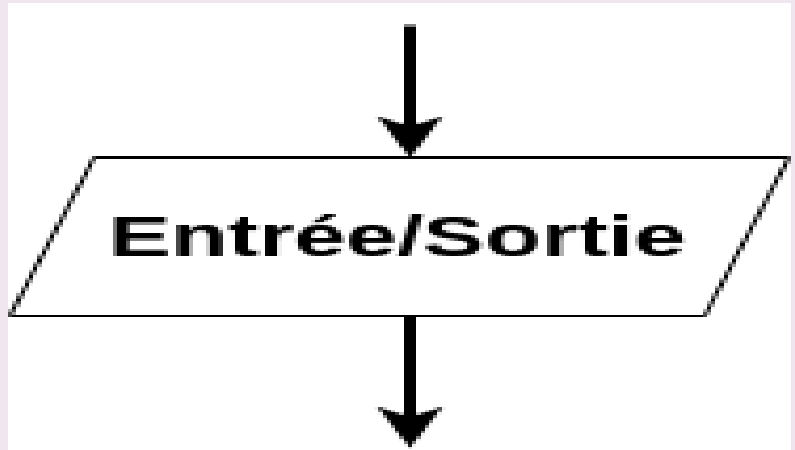
Les instructions sont symbolisées par un rectangle et un text à l'interieur.

Instruction



Un parallélogramme est utilisé pour représenter les entrées/sorties de l'agorithme.

Entrée/Sortie

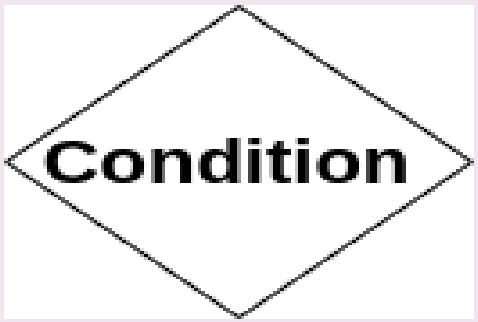


Les éléments d'un organigramme

Les losanges sont utilisés pour symbolisées les donditions.

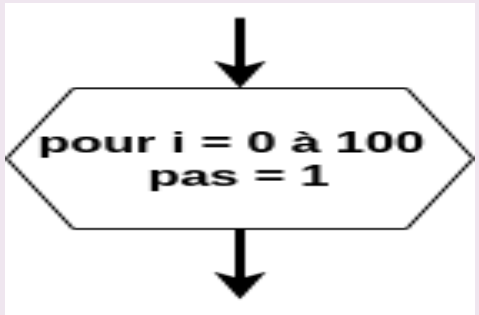
L'appel d'un sous programme est symbolisé par un rectangle avec deux traits à l'interieur.

Condition



Sous-programme

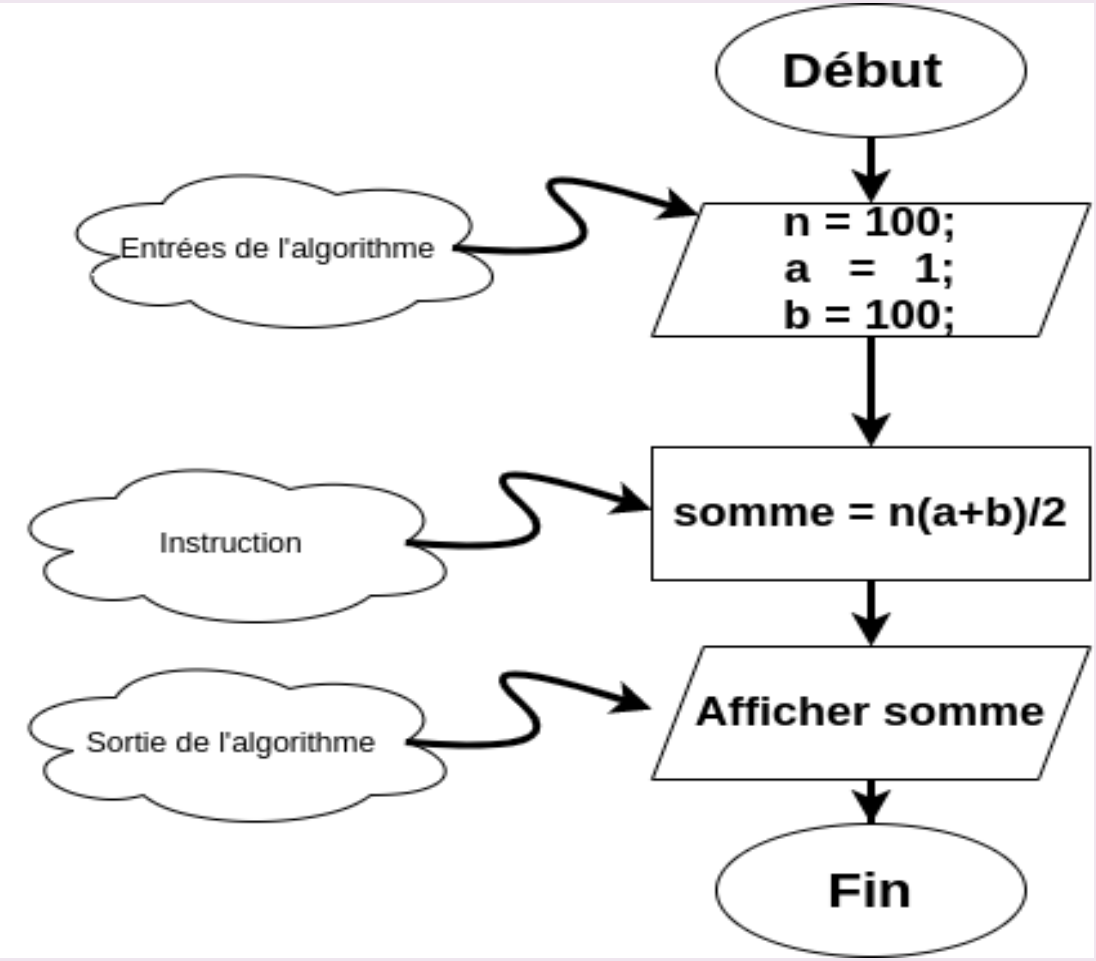
Boucle



Exemple organigrame

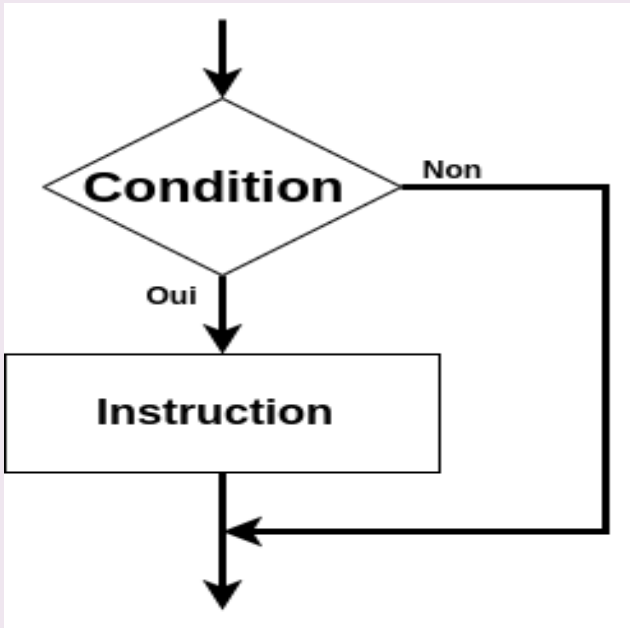
L'ogranigramme suivant représente l'algorithme qui calcule la somme de tous les nombres entiers de 1 à 100.

Somme de nombres entiers de 1 à 100

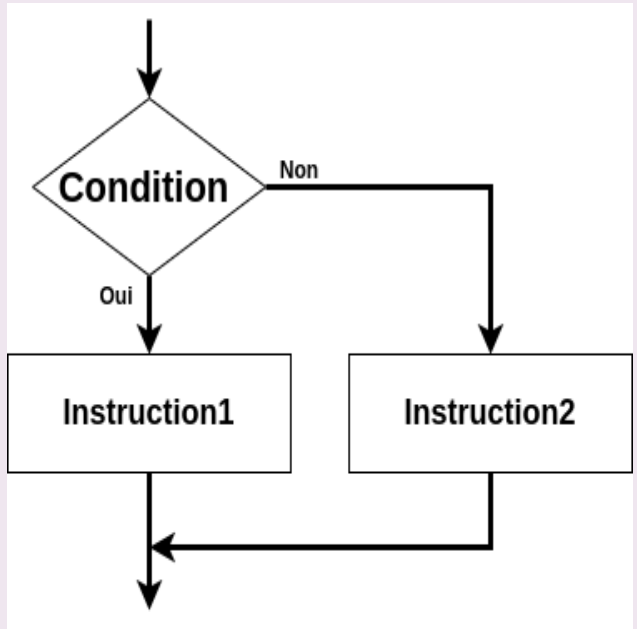


Structures de contrôle

Si ... alors (if...)

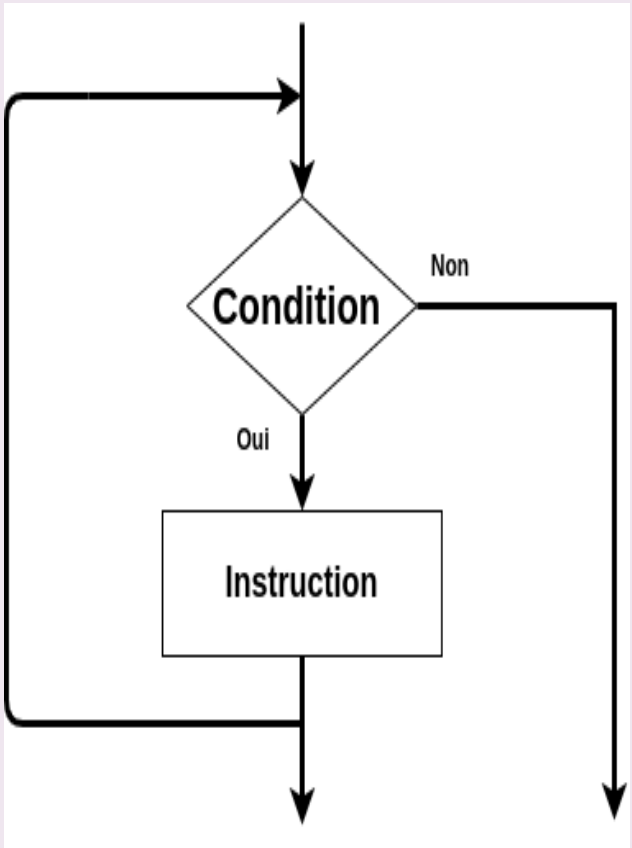


Si ... alors ... sinon (if... else if)

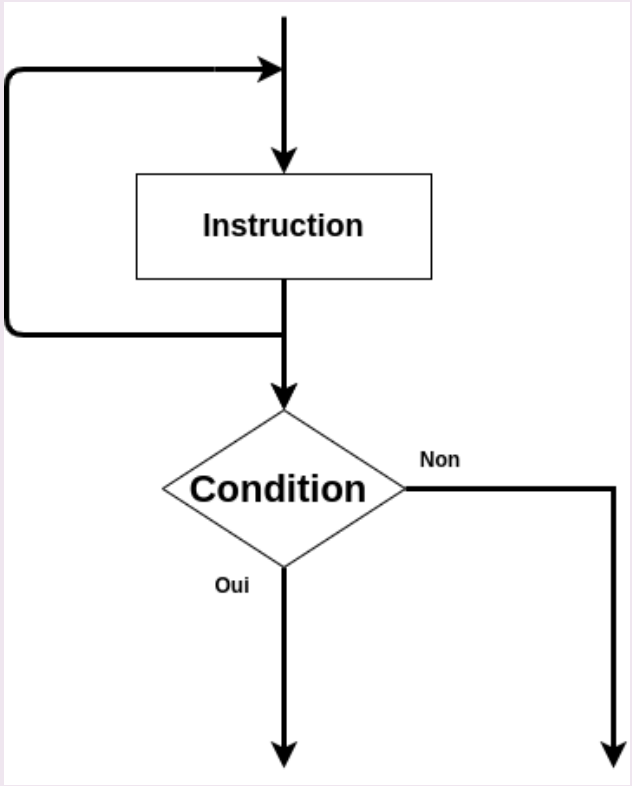


Structures de contrôle

Tant que ... faire (While ... do)



Répéter ... jusqu'à (Do ... while)



Exemlpe organigrame avec une condition Si (if)

L'exemple suivant montre un simple algorithme qui vérifie si l'identifiant (Id) d'un utilisateur à une longueur acceptable (elle doit être superieure à 10).
Les étapes de l'algorithme sont:

- **Étape1 :** lire l'Id (indentifiant).
- **Étape2 :** calculer NB le nombre de caractères de l'Id.
- **Étape3 :** vérifier si NB est superieur à 10.
- **Étape4 :** un message "L'identifiant est long" est afiché si la condition est vérifiée.
- **Étape5 :** un message "L'identifiant est court" est afiché dans le cas contraire.

Exemlpe organigrame avec une condition Si (if)

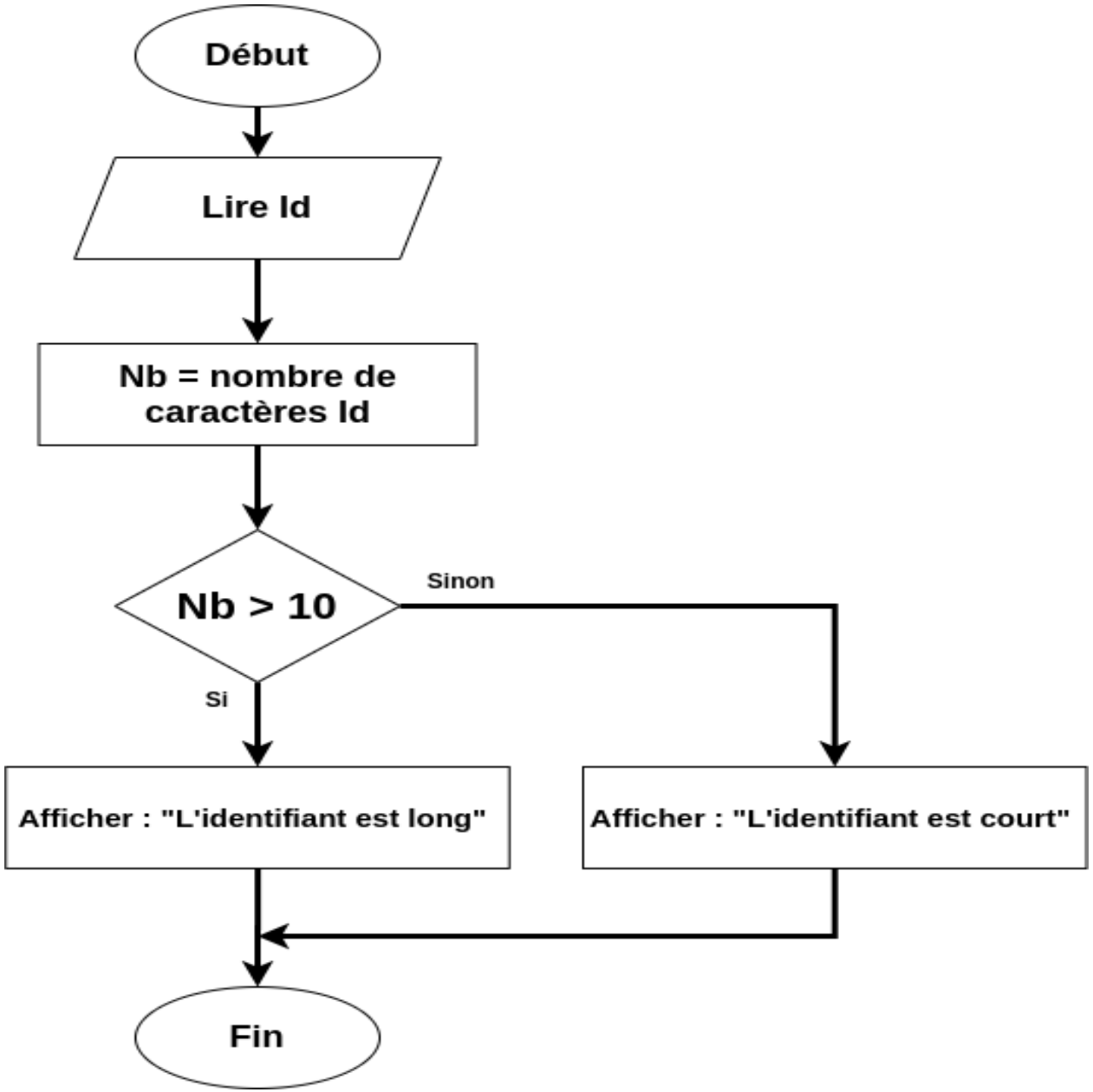


Table des matières

Qu'est ce qu'un pseudocode?

- Un pseudocode est une description d'un algorithme avec un vocabulaire simple et sans connaissance à priori du langage de programmation.
- Le pseudocode est un langage dans lequel on exprime clairement et formellement un algorithme.
- Dans un pseudocode nous exprimons les idées formelles de l'algorithme avec une langue près du langage naturel de ses usagers.
- Un pseudocode doit respecter une forme rigoureuse.

Algorithme 1 : Structure d'un algorithme

Début

Déclarations des variables
Initialisation des variables
Instructions de traitement

FIN

Opérations de base

Nous pouvons lister trois opérations (instructions) basiques dans un pseudocoe:
Les opérations primitives sont , et

- **Lire :** cette instruction effectue une lecture à partir d'un périphérique d'entrées (un clavier, ...).
- **Écrire :** cette instruction permet d'afficher une sortie de l'algorithme (une valeur sur un écran par exemple).
- **$\leftarrow$:** ce symbole effectue l'opérations d'affectation d'une valeur à une variable.

Opérations arithmétiques

Opérations arithmétiques	
Pseudocode	Description
a + b	Addition
a - b	Spustraction
a * b	Multiplication
a / b	Division
a % b (a mod b)	Reste de la division de a par b

Opérations arithmétiques	
Pseudocode	Description
a ET b	Vrai si a et b sont tous deux vrai
a OU b	Vrai si au moins a ou b est vrai
NON a	Multiplication

Opérations arithmétiques

Opérations arithmétiques	
Pseudocode	Description
a = b	Égal
a < b	Inferieur
a > b	Superieur
a <= b	Inférieur ou égal
a >= b	Supérieur ou égal
a != b	Différent de

Exemple de pseudocode

Algorithme 2 : Somme entiers de 1 à 100

Début

Entier : n, a, b

n $\leftarrow$ 100

a $\leftarrow$ 1

b $\leftarrow$ 100

somme $\leftarrow n * (a + B)/2$

Écrire ("La somme est égale à :", somme);

FIN

Structures de contrôle

Si ... alors (if ...)

Algorithme 3 : Boucle Si ... alors

Début
  Déclarations
  Initialisations
  Si condition **alors**
    Instructions
FIN

La condition est une expression booléenne, elle n'est executée que si la condition est vraie.

Si ... alors sinon (if ... else if)

Algorithme 4 : Boucle Si ... alors sinon

Début
  Déclarations
  Initialisations
  Si condition1 **alors**
    InstructionsA
  Sinon condition2 **alors**
    InstructionsB
FIN

Les "InstructionsA" sont exécutées si la "condition1" est vraie, sinon c'est les "InstructionsB" qui seront exécutées.

Structures de contrôle

Pour ... faire (For ... do)

Algorithme 5 : Boucle For ... do

```
Début
    Déclarations
    Initialisations
    Pour i de 0 à n faire
        Instructions
    Fin pour
FIN
```

Les variables "i" et "n" doivent être déclarées.

Tant que ... faire (While ... do)

Algorithme 6 : Boucle Tant que ... faire

```
Début
    Déclarations
    Initialisations
    Tant que condition faire
        Instructions
    Fin Tq
FIN
```

Les "Instructions" sont exécutées tant que la "condition" est vraie.

Structures de contrôle

Répéter ... Jusquà (Do ... While)

Algorithme 7 : Boucle Répéter ... Jusquà

Début
  Déclarations
  Initialisations
  Répéter i de 0 à n **faire**
    Instructions
  Jusquà condition
FIN

Les "Instructions" à l'interieur de la boucle sont exécutés jusqu'à l'enrichissement de "condition".

Exemple somme des entiers pairs

L'algotithme suivant clacule la somme des entiers pairs qui sont compris entre 0 et une valeur N lue.

Algorithme 8 : Somme des entiers pairs

Début

```
Entier : i, somme, N
somme ← 0
Lire (N)
Pour i de 0 à N faire
    Si i%2 alors
        somme ← somme + i
Fin pour
Écrire ("Somme des entiers pairs = :", somme)
```

FIN

Doublement de capital

- Nous cherchons un algorithme qui demande à l'utilisateur de lui fournir la valeur d'un capital **cap** qu'il souhaite placer, ainsi que le **taux** (annuel) auquel sera effectué le placement.
- L'algorithme doit afficher l'évolution annuelle de ce capital jusqu'à ce qu'il ait atteint ou dépassé le double du capital initial.
- Notez bien que nous utiliserons des variables de type réel.
- Faites dérouler l'algorrithme pour une valeur de capital de 10000 et un taux de 20%.

Doublement de capital solution 1

Algorithme 9 : Doublement de capital

Début

Réel : capIni, cap, taux
Écrire "donnez le capital à placer et le taux : "
Lire cap, taux
capIni $\leftarrow$ cap
Répéter
cap $\leftarrow$ cap * (1 + taux)
Écrire "capital un an plus tard : ", cap
Jusqu'à cap >= 2 * capIni
Écrire "fin du programme"

FIN

Doublement de capital solution 2

Algorithme 10 : Doublement de capital

Début

Réel : capIni, cap, taux
Écrire "donnez le capital à placer et le taux : "
Lire cap, taux
capIni $\leftarrow$ cap – On peut écrire aussi capIni := cap
Tant que cap $<$ 2 * capIni faire
    cap $\leftarrow$ cap * (1 + taux)
    Écrire "capital un an plus tard : ", cap
Fin TQ
Écrire "fin du programme"

FIN

Différence entre Tant que ... faire et répeter ... jusqu'à

Nous pouvons présenter la différence comme suit:

Répéter
    instruction
Jusqu'à condition

est équivalent à :

instruction
Tant que (non condition) **faire**
instruction

Différence entre Tant que ... faire et répeter ... jusqu'à

De même, l'ecriture :

Tant que condition **faire**
  instruction

est équivalente à :

Tant que condition **faire**
  instruction
Répéter
  si condition **alors**
    instruction
Jusqu'à condition

Exercice 1

Écrire un algorithme qui lit une suite de caractères, terminée par un point, et qui affiche le nombre de caractères lus (point non compris).

Notez bien que nous allons traiter des varible de type caractère.

Solution Exercice1

Algorithme 11 : Nombre de caractère

Début

Caractère c - - pour lire chacun des caractères

Entier nc - - pour compter le nombre de caractères lus

nc $\leftarrow$ 0

Répéter

  Lire c

  nc $\leftarrow$ nc + 1 - - on comptabilise tous les caractères, même le point

Jusqu'à c = ''.''

nc $\leftarrow$ nc - 1 - - pour tenir compte du point

Ecrire "vous avez fourni ", nc, "caractères ", suivis d'un point

Fin

Solution Exercice 1

Il existe une deuxième solution:

répéter
    Lire c
    si c != '.' **alors**
        nb $\leftarrow$ nb + 1
Jusqu'à c = '.'

Exercice 2

Écrire un programme qui lit 20 notes entières et qui indique le pourcentage de notes supérieures à 10.

Solution Exercice 2

Algorithme 12 : Pourcentage

```
Entier i  - - compteur de boucle sur les 20 notes
Entier nMoy  - - compteur du nombre de notes
supérieures à 10
Entier note  - - pour lire une note
Réel pourcent  - - pourcentage de notes supérieures à 10
nMoy ← 0
Ecrire "donnez 20 notes entières"
Pour i de 1 à 20
    Lire note  - - lecture d'une note
    Si note > 10 alors
     nMoy ← nMoy + 1
Si nMoy = 0 alors
    Ecrire "aucune note supérieure à 10"
sinon
     pourcent ← (100.0 * nMoy)/20
    Ecrire "il y a ", pourcent, "% de notes > à 10"
Fin
```

Références