

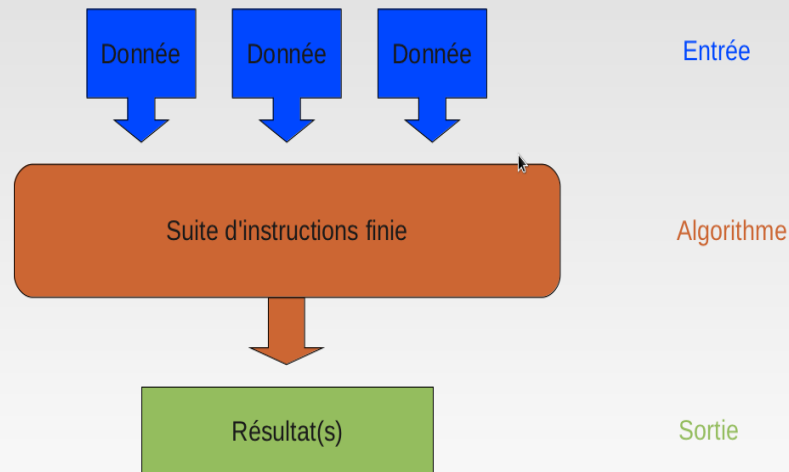
# Algorithmique

## Informatique

L'informatique est le domaine concernant le traitement automatique de l'information. Ce traitement est réalisé par des algorithmes.

## Algorithme

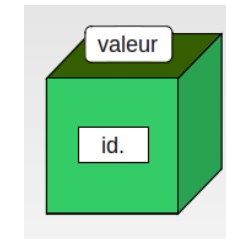
On pourrait définir un algorithme comme une **suite finie d'instructions** permettant, à partir de **données** de départ (d'informations), d'obtenir un **résultat** recherché.



## Variables

Les données (en entrée ou créées par l'algorithme) sont stockées dans des **variables** (qui correspondent à des emplacements de la mémoire). Une variable peut être vue comme une boîte pouvant contenir des objets (des valeurs). Les variables ont un nom (un identificateur) qui permet de les manipuler.

Faire une **affectation** consiste à mettre une valeur dans une variable.



L'affectation est schématisée par  $\leftarrow$  ou  $\rightarrow$  ou  $=$  suivant les machines.

Ainsi :

$a \leftarrow 13$       ou       $13 \rightarrow a$       ou       $a = 13$

signifie

« la variable portant le nom « a » prend la valeur 13 »

## Proposition de structure d'un algorithme

### Variables

on donne ici la liste des variables qui seront utilisées : pour les données en entrée, pour les calculs intermédiaires et pour les résultats à afficher.

### Traitement

on donne ici la liste des instructions :

- récupérer les données en entrée (instruction **Demander** ou **Lire**) ;
- faire des affectations (instruction  $\leftarrow$ ), des calculs ...

### Sortie

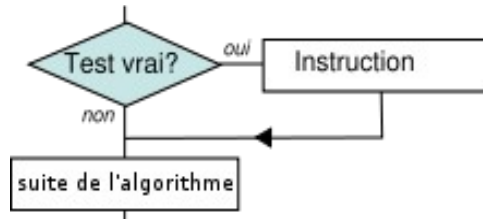
Afficher à l'écran (ou enregistrer dans un fichier) des résultats obtenus ou des messages (instruction **Afficher**).

Les trois premières instructions : **Demander (Lire)**, **Affecter**, **Afficher**.

## Les tests (structures conditionnelles)

Il est parfois nécessaire qu'un algorithme ait besoin de s'adapter à différentes situations. On utilise alors des tests. Il y en a deux sortes :

Forme 1 : **si** condition vérifiée **alors** instructions à effectuer



La condition est une comparaison, telles que, par exemple :

temperature = 0    ou    hauteur < 3    ou    x ≥ - 1

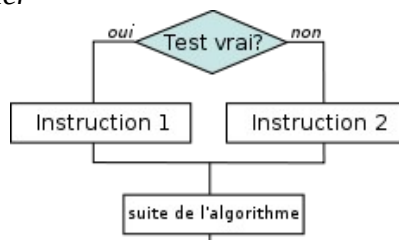
Voici un exemple volontairement simpliste :

**Si il pleut alors je prends mon parapluie**

qu'on pourrait traduire ainsi :

**Si temps=pluie alors prendreparapluie=oui**

Forme 2 : **si** condition vérifiée **alors** instructions à effectuer **sinon** autres instructions à effectuer



Exemple :

**Si** j'ai assez d'argent **alors** je m'achète la TX3000 **sinon** j'achète la Z20 qu'on pourrait traduire ainsi (en supposant que la TX3000 coûte 500 €) :

**Si** argent > 500 **alors** achat = « TX3000 » **sinon** achat = « Z20 »

Exemple 1 :

```
Variables
    les nombres i, p
Traitement
    i ← 5
    p ← 25
Si
    i < 10
Alors
    p ← 32
Sortie
Afficher p
```

L'algorithme affichera 32 car la condition  $i < 10$  est vérifiée ( $5 < 10$ ).

Exemple 2 :

```
Variables
    les nombres b, f, s
Traitement
Lire f
    b ← 3
    s ← 20
Si
    b + f > s
Alors
    s ← b + f
Sinon
    s ← s + 1
Sortie
Afficher s
```

Si l'utilisateur entre dans la machine (instruction Lire) la valeur 5 pour f alors l'algorithme affichera 21 car la condition «  $b + f > s$  » n'est pas vérifiée ( $3 + 5 \not> 20$ ) donc il exécute «  $s \leftarrow s + 1$  » (donc  $s \leftarrow 21$ ).

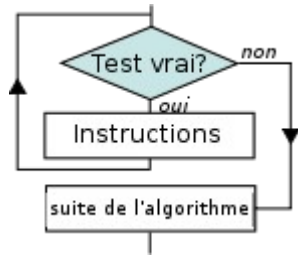
Si l'utilisateur entre la valeur 26 alors l'algorithme affichera 29 car  $b + f = 3 + 26 = 29 > 20$  donc il exécute l'instruction «  $s \leftarrow b + f$  ».

## Les boucles (répétitions d'instructions)

Nous ne parlerons ici que des boucles TantQue mais il existe aussi des boucles RépéterJusqu'à (peu utilisée) et des boucles Pour (qui sont très fréquentes).

L'idée est ici de demander à la machine d'effectuer une répétition d'instructions tant qu'une condition est vérifiée.

**TantQue** condition vérifiée **Faire** instructions **FinTantQue**



Remarque : le mot Faire n'existe pas dans les langages de programmation.

Exemple 1 :

```
Variables
    le nombre i
Traitement
    i ← 1
    TantQue i ≤ 100
    Faire
        Afficher « Je dois être attentif en classe. »
        i ← i + 1
    FinTantQue
Fin
```

Cet algorithme affiche le message « Je dois être attentif en classe. » et augmente la valeur de i à 2. Comme i est encore inférieur ou égal à 100, la machine ré-affiche le message et augmente i à 2 et ainsi de suite 100 fois. Cet algorithme affiche donc 100 fois le texte « Je dois être attentif en classe. » et sert donc à faire des punitions !

Exemple 2 : (hasard(1 ; 6) choisit un nombre entier entre 1 et 6)

```
Variables
    les nombres h et n
Traitement
    h ← hasard(1..6)
    Lire n
    TantQue n ≠ h
    Faire
        Afficher « Mauvaise réponse. »
        Lire n
    FinTantQue
    Afficher « Bonne réponse. »
Fin
```

Ici, la machine choisit un nombre entier au hasard entre 1 et 6 (appelé h). L'utilisateur entre des valeurs (dans n) tant qu'il n'a pas trouvé la valeur de h.

Exemple 3 :

```
Variables
    les nombres i, p et n
Traitement
    Lire n
    i ← 1
    p ← 1
    TantQue i < n
    Faire
        i ← i + 1
        p ← p × i
    FinTantQue
    Afficher p
Fin
```

Cet algorithme donne le nombre d'anagramme d'un mot de n lettres. Par exemple, on peut prouver que « MOT » a  $1 \times 2 \times 3 = 6$  anagrammes et que « MATHS » a  $1 \times 2 \times 3 \times 4 \times 5 = 120$  anagrammes.