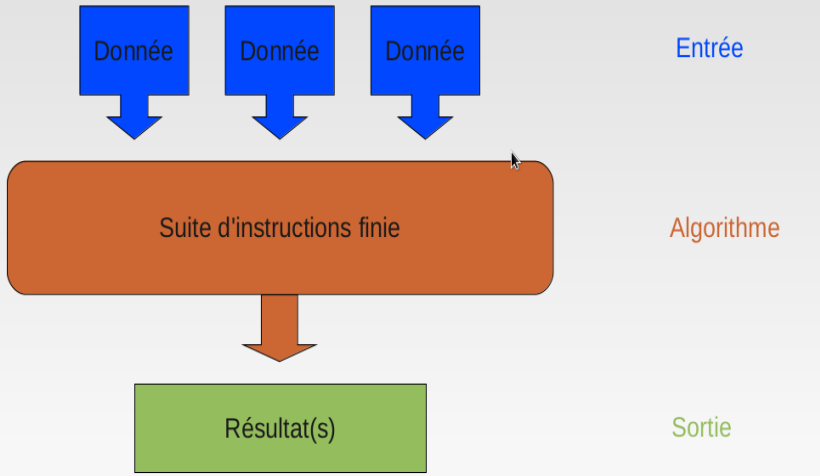
Algorithmique

Informatique

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

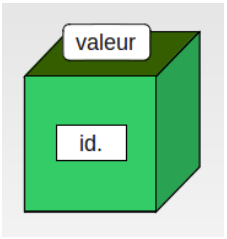
Algorithme

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 (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 consiste à mettre une valeur dans une variable.



L'affectation est schématisée par ← ou → ou = suivant les machines.
Ainsi :
a ← 13 ou 13 → a ou a = 13
signifie
« la variable portant le nom « a » prend la valeur 13 »

Proposition de structure d'un algorithme

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.

on donne ici la liste des instructions :
– récupérer les données en entrée (instruction ou) ;
– faire des affectations (instruction ←), des calculs …

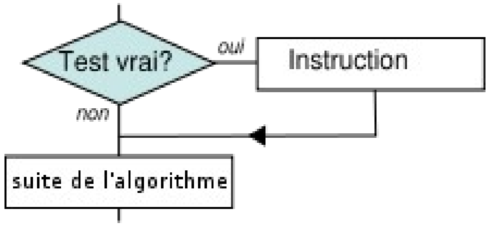
Afficher à l'écran (ou enregistrer dans un fichier) des résultats obtenus ou des messages (instruction).

Les trois premières instructions : ( ), , .

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 :

<u>Forme 1 :</u> *ond i t nd ẃ rеf t ees du fẁ ndu av vseœ ẁ sf*



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

temperature = 0 ou hauteur < 3 ou *l* ≥ - 1

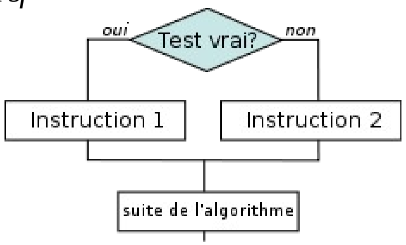
Voici un exemple volontairement simpliste :

t pj pẁ snjfd iu avnd ẏ ∓ ∓ ẁpt s

qu'on pourrait traduire ainsi :

a ẏu c j ẁpt s jfd i f sj ∓ ∓ ẁpt sa ẁt

<u>Forme 2 :</u> *ond i t nd ẃ rеf t ees du fẁ ndu av vseœ ẁ sf ẁ fa v du fẁ ndu av vseœ ẁ sf*



Exemple :

j'ai assez d'argent je m'achète la TX3000 j'achète la Z20

qu'on pourrait traduire ainsi (en supposant que la TX3000 coûte 500 €) :

∓ cd c ccc o= c =c vcvc cccccci no Vovo= c =vc vcvcccc

<u>Exemple 1 :</u>

```
ar  n   i
      les nombres i, p
  rn   o
      i ← 5
      p ← 25
      Si
            i < 10
      Alors
            p ← 32
  V n
              p
```

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

<u>Exemple 2 :</u>

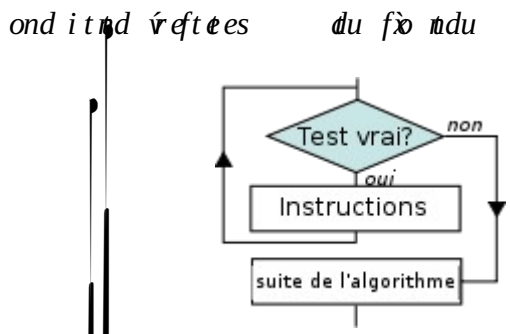
```
ar  n   i
      les nombres b, f, s
  rn   o
      Lire f
      b ← 3
      s ← 20
      Si
            b + f > s
      Alors
            s ← b + f
      Sinon
            s ← s + 1
  V n
              s
```

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 ≯ 20) donc il exécute « s ← s + 1 » (donc s ← 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 ← 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.



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

Exemple 1 :

```
Variables
    le nombre i
Début
    i ← 1
        Tant que i ⩽ 100
            « Je dois être attentif en classe. »
            i ← i + 1
```

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
Début
    h ← hasard(1..6)
        n
            n ≠ h
                « Mauvaise réponse. »
            n
        « Bonne réponse. »
```

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
Début
        n
    i ← 1
    p ← 1
            i < n
        i ← i + 1
        p ← p × i
          p
```

Cet algorithme donne le nombre d'anagramme d'un mot de n lettres.
Par exemple, on peut prouver que « MOT » a $1\times2\times3=6$ anagrammes et que « MATHS » a $1\times2\times3\times4\times5=120$ anagrammes.