

Algorithmie

Qu'est-ce qu'un algorithme ?

C'est une méthode permettant de résoudre un problème de manière systématique.

Les algorithmes pour l'informatique doivent donc s'appuyer sur ce que « sait faire » l'ordinateur. En pratique, ces capacités dépendent du langage et de l'environnement de programmation utilisés.

Dans le cadre de la programmation, l'algorithme va permettre d'écrire de manière compréhensible par tous et de manière suffisamment proche des langages de programmation ce que l'ordinateur va devoir faire.

Un algorithme est rédigé dans un langage à mi-chemin entre le français et les langages de programmation : le pseudo-code.

L'écriture du pseudo-code ne fait l'objet d'aucune norme, la syntaxe qui vous sera proposée par la suite est une possibilité parmi d'autres.

Elements d'un algorithme

Les variables

Bases de la programmation, permettent d'associer un nom à une valeur.

Pour créer notre variable (on dira qu'on déclare notre variable)

```
variable [nom de la variable] : [type de la variable]
```

Types de variable possibles : Boolean (Vrai ou Faux), Nombre, Caractère, Chaîne de caractères

Une variable sera toujours déclarer au début de notre programme.

Pour donner une valeur à notre variable (on dira qu'on affecte la valeur)

```
[nom de la variable] ← [valeur]
```

Il n'y a que deux cas d'utilisation possible pour les variables : l'affectation et la lecture.

Opérateurs arithmétiques

- + Addition
- Soustraction
- * Multiplication
- / Division

```
[nom de la variable] ← 12 + 4  
[nom de la variable] ← 4 + [autre variable]
```

Ce sont les règles de base des mathématiques qui s'appliquent.

Saisi et affichage

Pour afficher :

```
Afficher [nom de la variable]
Afficher "J'affiche du texte"
```

Pour saisir une valeur :

```
Saisir [nom de la variable]
```

Les Test

Un test en algorithmie est similaire à un aiguillage : seul un chemin sera parcouru par le programme.

Il existe deux formes possibles pour les tests.

```
Si condition Alors
    Instruction
FinSi

Si condition Alors
    Instruction 1
Sinon
    Instruction 2
FinSi
```

On applique la notion d'indentation, c'est à dire le fait de décaler les instructions concernés par le test.

Exemple pour illustrer :

```
Si la rue à droite est autorisée à la circulation Alors
    Tournez à droite
    Avancez
    Prenez la deuxième à gauche
Sinon
    Continuez jusqu'à la prochaine rue à droite
    Prenez cette rue
    Prenez la première à droite
FinSi
```

Il est également possible de placer des tests à l'intérieur d'autres tests (on parle alors de tests imbriqués). On peut imbriquer des tests sans limite de profondeur (un test dans un test dans un test ...).

Variante : Le SinonSi

Ecriture alternative aux tests imbriqués, le Sinon et le Si peuvent être fusionnés en un SinonSi.

Exemple:

```
Si Temperature <= 0 Alors
    Afficher "C'est de la glace"
SinonSi Temperature < 100 Alors
    Afficher "C'est du liquide"
Sinon
    Afficher "C'est de la vapeur"
FinSi
```

 On peut mettre autant de SinonSi que l'on veut

Conditions : Opérateurs de comparaison

 inférieur à

 inférieur ou égal à

 égal à

 supérieur ou égal à

 supérieur à

 différent de

Une condition est une comparaison qui une fois évaluée donnera un résultat booléen (Vrai ou Faux)

```
5 < 7 est vrai
5 = 7 est faux
```

Les conditions doivent être "simple", c'est à dire que contenir qu'une comparaison.

`4 < 5 < 7 est mathématiquement correcte mais pas valable en programmation`

Conditions : Opérateurs logiques

Pour exprimer des conditions "complexes", il faut les décomposer en conditions simples et les associer grâce aux opérateurs logiques.

- ET
- OU
- NON

Reprenons l'exemple précédent et appliquons les opérateurs logiques

```
4 < 5 < 7 pourra s'écrire
4 < 5 ET 5 < 7
```

Rappel sur les booléen :

Fonctionnement du ET

`4 < 5 ET 5 < 7`

Comparaison 1	Comparaison 2	Résultat
Vrai	Vrai	Vrai
Vrai	Faux	Faux
Faux	Faux	Faux
Faux	Vrai	Faux

Fonctionnement du OU

4 < 5 OU 5 < 7

Comparaison 1	Comparaison 2	Résultat
Vrai	Vrai	Vrai
Vrai	Faux	Vrai
Faux	Faux	Faux
Faux	Vrai	Vrai

Fonctionnement du NON

NON (4 < 5)

Comparaison 1	Résultat
Vrai	Faux
Faux	Vrai

i Il n'y a pas de limite au nombre de conditions que l'on peut comparer grâce aux opérateurs logiques. Dans le cas de conditions composées, la présence des parenthèses possède une influence sur le résultat.

Les boucles

Les boucles servent à automatiser la répétition d'actions redondante de manières simple.

Boucle TantQue

```
TantQue booléen
    Instructions
FinTantQue
```

Lorsque le programme arrive sur la ligne du TantQue, il évalue le booléen (c'est à dire qu'il vérifie sa valeur). Si le booléen est VRAI, il exécute les instructions jusqu'à rencontrer la ligne FinTantQue puis retourne à la ligne du TantQue et recommence (évalue le booléen ...) jusqu'à ce que le booléen soit évalué à FAUX.

On peut synthétiser la boucle en "Tant que la condition est vraie faire ...".

⚠ Attention aux erreurs. Ecrire une condition qui ne sera jamais VRAI implique que votre programme ne rentrera jamais dans votre boucle. Au contraire, si votre condition n'est jamais FAUX, alors votre programme ne sortira jamais de la boucle.

Boucle Pour

La boucle Pour permet de définir un nombre déterminé de passages en indiquant la valeur de début de notre boucle, la valeur de fin et (optionnel) le pas.

```
Pour Compteur ← Initial à Final faire
    Instructions
Compteur Suivant

Pour Compteur ← Initial à Final Pas valeurDuPas faire
    Instructions
Compteur Suivant
```

 Si le Pas n'est pas indiqué, alors la valeur par défaut sera 1.

Exemple :

```
Pour Truc ← 1 à 15 faire
    Afficher Truc
Truc Suivant
```

Cette exemple affichera 1 puis 2 puis 3 ... jusqu'a 15.


Choisir entre TantQue et Pour

Pour savoir qu'elle est la meilleur boucle à choisir pour notre programme, il faut regarder si l'on connait le nombre de tours que l'on doit faire (on choisira Pour) ou si l'on "attends" un événement (on choisira TantQue).

Boucle Jusqu'a

Elle fonctionne exactement comme la boucle TantQue mais effectue au moins un passage dans la boucle avant d'évalué la condition.

```
Faire
    Instructions
Jusqu'a Booléen
```

 La boucle TantQue effectuera entre 0 et n tours tandis que la boucle Jusqu'a effectuera entre 1 et n tours.

 Comme pour les tests, Il est possible d'imbriqués les boucles.

Les Tableaux

Si dans notre programme nous avons besoin de 15 valeurs, actuellement nous devons déclarer 15 variables. Imaginez si vous devez gérer 1000 valeurs.

La solution : Les Tableaux. Ils servent à gérer une grande quantité de valeurs en une seule variable où chaque valeur sera désigné par un numéro (appelé indice).

Comme pour les variables, il faut déclarer nos tableaux en précisant son nom et le plus grand indice qu'il contiendra :

```
Tableau Note[11] : Nombre
```

 La première "case" d'un Tableau contient l'indice 0

⚠ Lors de la déclaration, le plus grand indice du tableau doit donc être (nombre d'élément que l'on souhaite - 1)

Parcourir un Tableau avec les boucles

```
Pour i ← 0 à 11 faire  
    Afficher Note[i]  
i Suivant
```