



Algorithmes

Algorithme

- Introduction à cette formation
 - Votre formateur ... Et Vous
 - Le matériel et logiciels
 - Le langage Python est utilisé comme support
 - L'organisation – horaires
 - Formation de 3,5 jours
 - La forme :
 - Un mélange de concepts avec application directe par un exemple simple
 - Des exercices
 - Une évaluation



Algorithme

- Les liens utiles

- <https://openclassrooms.com/fr/courses/1467201-algorithmique-pour-lapprenti-programmeur/1467284-quest-ce-quun-algorithme>
- <https://www.cours-gratuit.com/cours-algorithme/cours-et-exercices-complet-algorithmes-en-pdf>

- Les ordinateurs sont des bestioles binaires
- Qu'est ce que l'algorithmique ?
- Les variables, leur type, affectation
- Opérateurs et expressions
- Les tests
- Boucles
- Les tableaux
- Définir une fonction
- Lire et écrire dans un fichier

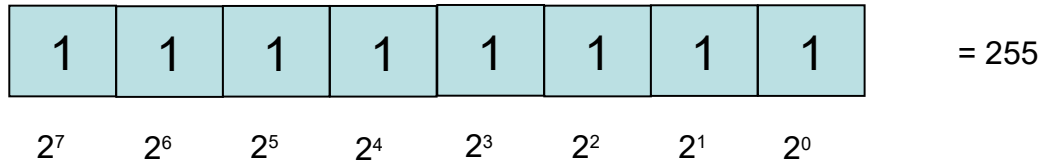
- L'humain à – en grande majorité – 10 doigts sur ses deux mains. Il a imaginé un système de numérotation en conséquence, une numérisation en base 10
- 8209 peut être décomposé ainsi :
 - $8 \times 1000 + 2 \times 100 + 0 \times 10 + 9 \times 1$ ce qui peut s'écrire
 - $8 \times 10^3 + 2 \times 10^2 + 0 \times 10^1 + 9 \times 10^0$

8	2	0	9
10^3	10^2	10^1	10^0

- Les babyloniens ont compté en base 60
- Les Shadoks comptent en base 4

<https://www.youtube.com/watch?v=IP9PaDs2xgQ>

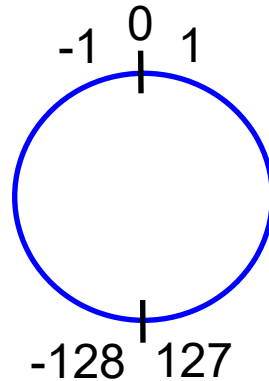
- Tout système électronique numérique a pour moyen de mémorisation une « case » ne contenant que le 0 ou 1 : Cette case se nomme un bit
- Ces cases ont été groupées d'abord par 4 puis maintenant par 8 : un octet (byte)
Un octet peut donc coder 2^8 soit 256 possibilités



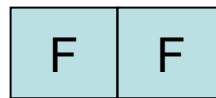
- Un nombre peut être vu/déclaré comme purement positif



- Un nombre peut être vu/déclaré comme signé

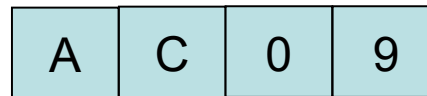


- Codage hexadécimal
Le codage en base 2 n'est pas pratique pour l'humain
- Un octet est vu comme 2 paquets de 4 bits
- Un paquet de 4 bits (un quartet) décrit 16 possibilités
- En hexadécimal chaque possibilité est symbolisée par un chiffre de 0 à F :
 - 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F



16^1 16^0

$$= 15 \times 16 + 15 \times 1 = 255$$



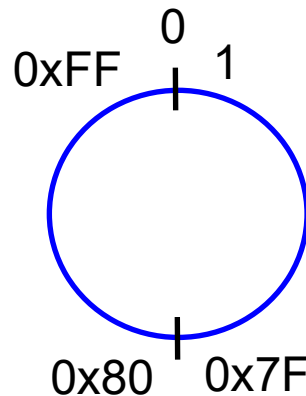
16^3 16^2 16^1 16^0

$$= 44041$$

- Un nombre peut être vu/déclaré comme purement positif



- Un nombre peut être vu/déclaré comme signé
Par convention c'est le bit de poids fort (le 8^{ème}) qui est à 1 pour un nombre négatif



Pour un byte signé :

0xFF vaut -1

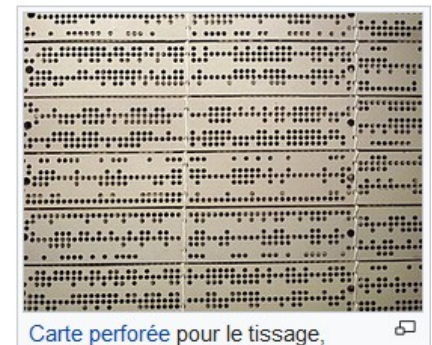
0x80 vaut -128

0x7F vaut +127

Exercice section 2

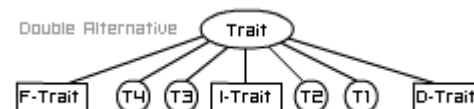
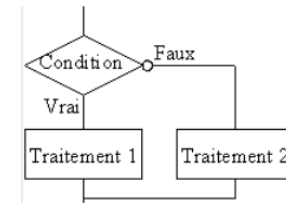


- Un **algorithme** est une suite finie et non ambiguë d'opérations ou d'instructions permettant de résoudre une classe de problèmes (source Wikipédia)
- Un algorithme, c'est une suite d'instructions, qui une fois exécutée correctement, conduit à un résultat donné.
 - un algorithme doit donc contenir uniquement des instructions compréhensibles par celui (ou la machine) qui devra l'exécuter
- Pour écrire efficacement un algorithme :
 - Se placer du côté de celui qui va l'exécuter
 - Avoir de l'intuition à défaut d'expérience
 - Avoir un esprit logique (pas mathématique)
 - Être méthodique et rigoureux



- Un algorithme en informatique concerne la manipulation de 4 concepts de base :
 - L'affectation de variables
 - La lecture/ écriture vers le monde extérieur
 - Les tests
 - Les boucles
- Un algorithme est indépendant des langages. C'est le passage de l'algorithme à la programmation qui utilise un langage.
- Conventions d'écriture d'un algorithme :
 - Organigrammes (logigramme, ordinogramme)
 - Arbres programmatiques
 - Pseudo-code

```
Fonction factorielle (n)
  r = 1
  Pour i de 1 jusqu'à n avec un pas de 1
    r = r*i
  Fin pour
  Retourner r
Fin Fonction
```



- Les variables sont des zones de stockage en mémoire
Ce sont des « boîtes » qui vont contenir des nombres, positifs ou négatifs, des nombres décimaux, du texte ...
- Toute boîte à une étiquette et une taille.
Une variable a :
 - Un nom
 - Un type (détermine la taille)



- Types possibles :
 - numériques

Type Numérique	Plage
Byte (octet)	0 à 255
Entier simple	-32 768 à 32 767
Entier long	-2 147 483 648 à 2 147 483 647
Réel simple	$-3,40 \times 10^{38}$ à $-1,40 \times 10^{45}$ pour les valeurs négatives $1,40 \times 10^{-45}$ à $3,40 \times 10^{38}$ pour les valeurs positives
Réel double	$1,79 \times 10^{308}$ à $-4,94 \times 10^{-324}$ pour les valeurs négatives $4,94 \times 10^{-324}$ à $1,79 \times 10^{308}$ pour les valeurs positives

- Types possibles :
 - Entier
 - Réel
 - Caractère
 - Chaîne de caractères (string)
 - Booléen. Vaut Vrai ou Faux
- Déclarer des variables en pseudo-code :

Entier age

Reel prixHT, tauxTVA

Chaîne nom

Caractère c

- Affecter une variable consiste à y mettre une valeur du bon type
- On dit aussi assigner une variable
- En pseudo :

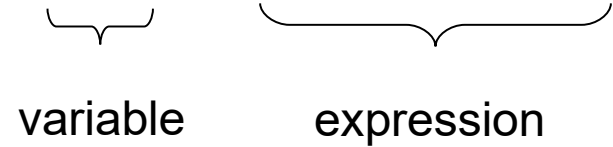
```
age ← 24  
nom ← "Albert"  
nom2 ← nom  
age ← age + 1
```

- Exercice section 3



- Dans l'instruction

Age \leftarrow Age + 1 + delta

The diagram shows the expression 'Age \leftarrow Age + 1 + delta'. A curly brace is placed under 'Age' and labeled 'variable'. Another curly brace is placed under 'Age + 1 + delta' and labeled 'expression'.

variable expression

- Une expression est un ensemble de valeurs, reliées par des **opérateurs**, et équivalent à **une seule valeur**

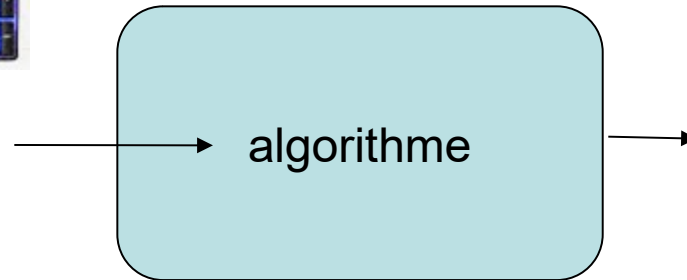
- Opérateurs numériques
 - +, -, *, /
 - % (ou mod) modulo, reste de la division entière 5 % 2 vaut 1
 - ^ pour la puissance ex 4^2
- Opérateur alphanumérique : & pour concaténer
 - A ← "Salut"
 - B ← " John"
 - C ← A & B
- Opérateurs logiques ou booléens
 - ET, OU, NON, XOR

- Dans la majorité des langages l'affectation \leftarrow se fait par un $=$
Pour autant nous ne sommes pas en math !

$X \leftarrow X + 1$

$X = X + 1$

le pseudo code se traduit par
ce qui n'a pas de sens en math !



Lit les
données
externes

Ecrit un
résultat en
externe



- Pseudo-code

```
Ecrire "Entrer votre age : "  
Lire age  
Ecrire "Entrer votre nom : "  
Lire nom  
Ecrire "Votre nom est: " & nom
```

- Dès que le programme rencontre une instruction Lire, l'exécution s'interrompt, attendant la frappe d'une valeur au clavier

- Exercice section 4



- Un test permet l'exécution d'une série d'instructions selon la valeur d'une **expression booléenne**
- En Pseudo-code

```
Si booléen Alors  
    Instructions  
Finsi
```

```
Si booléen Alors  
    Instructions 1  
Sinon  
    Instructions 2  
Finsi
```

- booléen est soit une **variable booléenne**, soit une **condition**

- Une Condition est une comparaison et comporte en général :
 - Une valeur
 - Un opérateur de comparaison
 - Une valeur
- Ex $\text{age} < 18$
- $'t' < 'z'$ vaut vrai
- $'Hello' < 'Arbre'$ vaut faux
- $'Hello' < 'arbre'$ vaut vrai
- Remarque :
 $18 < \text{age} < 30$ n'a pas de sens en algorithmique
 $18 < \text{age}$ ET $\text{age} < 30$ est correct
- Exercice section 5



- Il est possible de **composer** plusieurs conditions avec des opérateurs logiques ET , OU, NON, XOR
- Exemple : tester si age dans l'intervalle [10,20] se traduit par le pseudo
 $10 \leq \text{age} \text{ ET } \text{age} \leq 20$
- tester si age hors de l'intervalle [10,20] se traduit par le pseudo
 $\text{age} < 10 \text{ OU } \text{age} > 20$
- $\text{age} < 10 \text{ ET } \text{age} > 20$ est un bug !



- Tests imbriqués

```
Variable Temp en Entier
Début
Ecrire "Entrez la température de l'eau :"
Lire Temp
Si Temp <= 0 Alors
    Ecrire "C'est de la glace"
FinSi
Si Temp > 0 Et Temp < 100 Alors
    Ecrire "C'est du liquide"
Finsi
Si Temp > 100 Alors
    Ecrire "C'est de la vapeur"
Finsi
Fin
```

```
Variable Temp en Entier
Début
Ecrire "Entrez la température de l'eau :"
Lire Temp
Si Temp <= 0 Alors
    Ecrire "C'est de la glace"
Sinon
    Si Temp < 100 Alors
        Ecrire "C'est du liquide"
    Sinon
        Ecrire "C'est de la vapeur"
    Finsi
Finsi
Fin
```


- Transformation de Morgan :

l'inverse de A **ET** B

est

NON A **OU** NON B

l'inverse de A **OU** B

est

NON A **ET** NON B

Si A ET B Alors

Instructions 1

Sinon

Instructions 2

Finsi

équivalent à :

Si NON A OU NON B Alors

Instructions 2

Sinon

Instructions 1

Finsi

- Une boucle permet de **répéter** un ensemble d'instructions, de réaliser plusieurs **Itérations**

```
TantQue booléen
```

```
...
```

```
Instructions
```

```
...
```

```
FinTantQue
```

```
Caractère Rep
```

```
Début
```

```
Ecrire « Voulez vous un café ? (O/N) »
```

```
Lire Rep
```

```
TantQue Rep <> « O » ET Rep <> « N »
```

```
  Lire Rep
```

```
FinTantQue
```

```
Fin
```

Exercice
section 6



```
Pour compteur  $\leftarrow$  initial à Final Pas ValeurDuPas  
    instruction  
    instruction  
FinPour
```

```
Pour i  $\leftarrow$  0 à 10 Pas de 1  
    Afficher i « x 7 = » i*7  
FinPour
```

Exercice
section 7



Faire

instruction

instruction

Jusqu'à condition de sortie

Caractère c

Faire

Lire c

instructions

Jusqu'à c = 'q'

- Une variable simple est caractérisée par son nom et son type. Elle ne contient qu'une seule donnée.



- Une variable tableau est caractérisée par :
 - Son nom
 - Type des éléments
 - Sa taille fixe



Tableau entiers notes[10]

Entier a

notes[0] \leftarrow 7

A \leftarrow notes[0]

- Une liste est un tableau dont la taille peut être modifiée (augmentée ou diminuée) pendant son utilisation.

liste entiers notes[]

Entier a

notes[0] \leftarrow 7

A \leftarrow notes[0]

Exercice
section 8



- Utile pour :
 - Écrire une seule fois un algorithme qui sera utilisé plusieurs fois
 - Pour simplifier un algorithme principal, en déléguant à des algorithmes secondaires des parties de traitement



- Une fonction est définie par :
 - Une en-tête : nom de la fonction, paramètres et leur type, type de l'éventuelle valeur de retour
 - Un corps : algorithme qui utilise les paramètres d'entrée et qui réalise un traitement, fournit éventuellement une valeur de retour

- Définition

```
entier fact(entier n)
Début
    entier i
    entier f ← 1
    Pour i de 1 à n faire
        f ← f * i
    FinPour
    retourner f
Fin
```



- Utilisation

```
entier a, b ← 5
a ← fact(15)
a ← fact(b)
```



Exercice
section 9



- Heureusement il existe des fonctions prédéfinies :
 - Fonctions mathématiques, trigonométriques, algébriques
 - Fonctions sur les chaînes de caractères :
 - **Len(chaîne)** : renvoie le nombre de caractères d'une chaîne
 - **Mid(chaîne,n1,n2)** : renvoie un extrait de la chaîne, commençant au caractère n1 et faisant n2 caractères de long.
 - **Left(chaîne,n)** : renvoie les n caractères les plus à gauche dans chaîne.
 - **Right(chaîne,n)** : renvoie les n caractères les plus à droite dans chaîne
 - **Trouve(chaîne1,chaîne2)** : renvoie un nombre correspondant à la position de chaîne2 dans chaîne1. Si chaîne2 n'est pas comprise dans chaîne1, la fonction renvoie zéro.

- Servent à stocker des informations de manière permanente, entre deux exécutions d'un programme
- C'est l'OS (Operating système) qui s'occupe de l'aspect physique du fichier.
- Les programmes utilisent des interfaces de programmation (API) pour accéder au contenu des fichiers.
- Le contenu d'un fichier peut être :
 - Du texte (d'un certain encoding, UTF-8, UTF-16, Unicode ...)
 - Du binaire
- Quand le contenu est de type texte, il peut être organisé en d'innombrables façons :
 - Texte tabulé, séparé par des ;
 - XML
 - ...

- Il existe plusieurs types d'accès à un fichier :
 - Séquentiel : lecture ligne à ligne du fichier
 - Direct : lire directement une information du fichier
 - Indexé
- C'est l'accès séquentiel qui existe le plus souvent dans les langages.

- Pour utiliser un fichier, il faut d'abord l'ouvrir et choisir le mode :
 - En lecture
 - En écriture
 - Pour ajout à la fin

L'ouverture fournit un identifiant à utiliser pour les autres opérations

```
Chaîne lec, nom, prenom  
Ouvrir "fic.txt" identifiant monFic en lecture  
lec ← LireFichier monFic  
nom ← Mid(lec,1,20)  
prenom ← Mid(lec,21,20)
```

- Utiliser une boucle pour parcourir un fichier
- EOF() rend vrai quand la lecture est terminée
- Un fichier doit être fermé

```
Chaîne lec  
Ouvrir "fic.txt" identifiant monFic en lecture  
TantQue Non EOF(monFic)  
    lec ← LireFichier monFic  
    ...  
Fin TantQue  
Fermer monFic
```

Exercice
section 10

