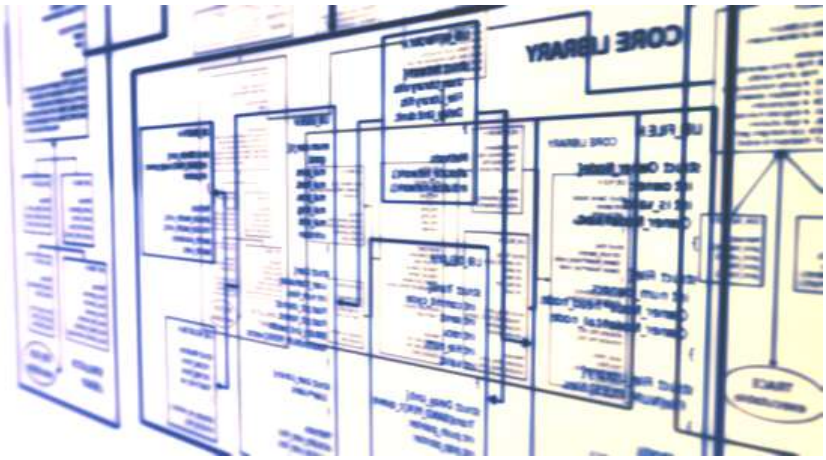
# Les algorithmes



## Baccalauréat S -Spécialité ISN

- 4.2 : Algorithme

Référentiel de formation

# Objectifs

L'apprenant doit être capable de...

- Comprendre un algorithme et expliquer ce qu'il fait.
- Modifier un algorithme existant pour obtenir un résultat différent.
- Concevoir un algorithme.
- Programmer un algorithme.
- S'interroger sur l'efficacité d'un algorithme.

# Prérequis

Représentation de l'information
Introduction à la programmation

# Principe

La complexification des systèmes informatiques a rendu nécessaire l'élaboration d'outils de descriptions grâce auxquels ont peut, de manière structurée et codifiée, expliciter :

- un besoin
- une démarche opératoire
- un raisonnement

L'algorithme est un de ces outils.

Il permet, en outre, de faciliter la gestion de projets complexes et dans le cas d'une application informatique le passage rapide vers la programmation car la plupart des langages informatiques utilisent un langage algorithmique.

Avec des langages tels que le C ou le Pascal, la résolution d'un problème informatique passe généralement par l'analyse descendante qui consiste à décomposer un problème en sous-problème jusqu'à descendre à des actions primitives.
On décompose ainsi un programme en un ensemble de sous-programmes appelés procédures qui coopèrent pour la résolution d'un problème.

**A chaque procédure ou fonction, on associe un algorithme identifié par son nom. Celui-ci doit être unique dans le programme complet.**

# Elaboration et règles

Un algorithme ne dépend pas

- **du langage terminal utilisé**
- **de la technologie qui devrait exécuter l'enchaînement séquentiel des actions**
- **Un algorithme ne peut être écrit que si le problème à résoudre est parfaitement maîtrisé. De plus, une erreur dans un algorithme se traduira par une erreur dans le développement final ou dans la compréhension du produit qu'il décrit.**

**Un algorithme s'écrit avec une feuille de papier. En aucun cas avec un ordinateur (sauf pour la mise en forme finale).**

Un algorithme :

- décrit une résolution de problème en mode pas à pas par des opérations simples connues par tous. A chaque pas, c'est la commande suivante qui est réalisée
- est délimité par les mots clés **DEBUT** et **FIN**
- comporte des variables dont il faut définir le format (entier, caractère, chaîne de caractères...)
- se compose d'instructions basées sur des mots simples en général des verbes d'action (afficher, saisir,
- peut comporter des sous-problèmes décrits par d'autres algorithmes

# Structures conditionnelles

Les structures conditionnelles permettent de réaliser des opérations en fonction d'une condition.

**SI** la réponse à la condition est de type logique (VRAI ou FAUX)
**ALORS** choisir ***SI ALORS SINON***
**SINON** choisir ***SELON...***.
**FINSI**

## SI ALORS SINON

Une structure conditionnelle permet de traiter une opération ou une suite d'opérations selon le résultat d'un test logique.

**La syntaxe normalisée est :**

```
SI <condition logique>
   ALORS
opération1
   SINON
opération2
FINSI
```

Si la ***condition logique*** est vraie alors opération1 est exécutée sinon c'est opération2

**Programmons :**

**En Delphi** : **IF** (condition) **THEN** instruction1 **ELSE** instruction2;
**En PHP :** **<?php if (**condition**)** {instruction1;} **ELSE** {instruction2;}**?>**
**En C :** **IF (**condition**)**{instruction1;} **ELSE** {instruction2;}

---

## SELON...

Lorsque la condition ne peut pas avoir de réponse binaire on utilisera la structure - plus rare - SELON QUE.

```
SELON QUE variable vaut

1:instruction1
   2:
instruction2
   3:
instruction3
   4:
instruction4
FINSELON
```

**Attention :** tous les types ne sont pas tolérés mais en général les entiers et les caractères sont admis;

**Exemple**

```
....
DEBUT
   afficher("Donner votre
âge : ")
   saisir(age)
   SELON QUE age vaut
```

```
        0..3 :
afficher("Bébé")
        4..12 :
afficher("Enfant")
        13..18 :
afficher("Adolescent")
        19..150 :
afficher("Adulte")
    FINSELON
FIN
```

Dans l'exemple précédent , la variable age est un entier compris entre 0 et 150. Selon sa valeur, le message affiché sera différent.
On remarquera la notation utilisée pour définir un intervalle de valeurs.

**Programmons :**

**En Delphi** : **CASE** variable **OF**
```
        val1:instruction1;
        val2:instruction2;
    END;
```
**En PHP :** **<?php**
```
    SWITCH (variable)
    {
        case valeur1:instruction1;BREAK;
        case valeur2:instruction2;BREAK;
    }
```

**En C :** **SWITCH (**variable**)**
```
    {
        case valeur1:instruction1;BREAK;
        case valeur2:instruction2;BREAK;
    }
```

# Structures itératives - Boucles

Une structure itérative permet de répéter une ou plusieurs opérations en fonction d'une condition logique atteinte ou non.
**Comment choisir entre les trois structures proposées ?**

**SI** nombre d'itérations connu
**ALORS** choisir boucle **POUR**
**SINON**
**SI** traitement exécuté au moins une fois
**ALORS** choisir boucle **REPETER**
**SINON** choisir boucle **TANT QUE**
**FINSI**
**FINSI**

### POUR.... FAIRE

Cette structure permet de répéter une suite d'opérations un certain nombre de fois. Elle n'a de sens que si le nombre d'itération est connu à l'avance.

```
POUR iteration=valeur_initiale
A valeur_finale
   FAIRE instructions
FINPOUR
```

**Programmons :**

**En Delphi** : **FOR** (variable=debut **TO** fin) **DO** instruction;

**En PHP : FOR** (variable**=**debut;variable**<**fin5132620031107381;variable**++){**inst ruction**;}**

**En C : FOR (int** variable**=**debut;variable**<**fin;variable**++){**instruction**;}**

**Remarque :** en C et PHP on peut aussi décrémenter (variable--) ou modifier le pas d'incrément ou de décrément variable+=2 par exemple)

### TANTQUE.... FAIRE

Cette structure répète une suite d'action tant que la condition est vérifiée.

```
....
TANT QUE <condition>
   FAIRE
       instructions
       mettre à jour
<condition>
 FINTQ
....
```

**REMARQUE :** si la condition est fausse dès le départ, les instructions ne seront jamais exécutées.

**ATTENTION :** il est indispensable que la condition soit mise à jour dans la boucle et que la condition finisse par ne pas être remplie sinon on ne sortira jamais de cette boucle.

**Programmons :**

**En Delphi** : **WHILE** (condition) **DO**
**BEGIN**
instruction;
mise_a_jour(condition);
**END;**

**En PHP : WHILE (condition){**instruction**;}**

**En C : WHILE (condition){**instruction**;}**

**REPETER.... JUSQU'A**

Cette structure répète une suite d'actions jusqu'à ce que la condition soit vérifiée.

```
....
REPETER
  instructions
  mettre à jour
<conditions>
JUSQU'A  <condition>
```

**REMARQUE :** les instructions seront exécutées au moins une fois.

**ATTENTION :** il est indispensable que la condition soit mise à jour dans la boucle et que la condition finisse par être remplie sinon on ne sortira jamais de cette boucle.

**Programmons :**

**En Delphi** : **REPEAT**

```
        instruction;
        mise_a_jour(condition);
    UNTIL (condition);
```

**En PHP : DO**

```
    {instructions;
    mise_a_jour(condition);
    }
  WHILE (condition);
```

**En C : DO**

```
    {instructions;
    mise_a_jour(condition);
    }
  WHILE (condition);
```

# Quelques conseils

Pour réaliser un algorithme il faut :

- avoir une écriture rigoureuse
- avoir un soucis de présentation et indenter, comme dans les exemples de ce cours, les lignes
- commenter

Il existe plusieurs manière de résoudre un problème, aussi faut-il toujours rechercher l'efficacité de l'algorithme.