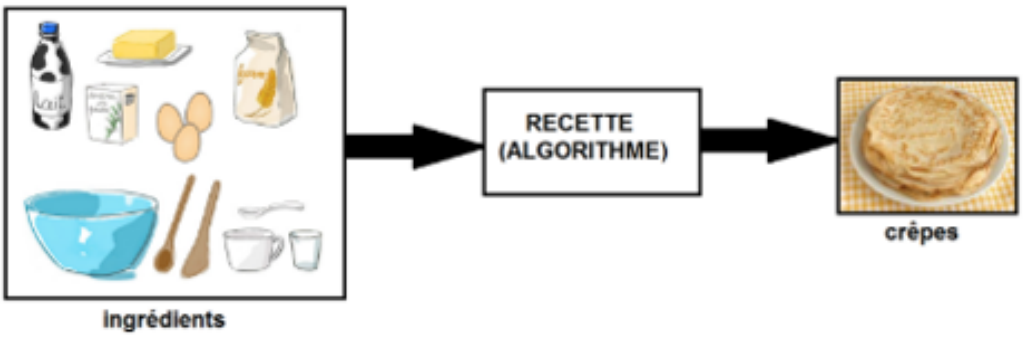
Algorithmie: Recherche d'occurrences

1 Qu'est ce qu'un algorithme ?

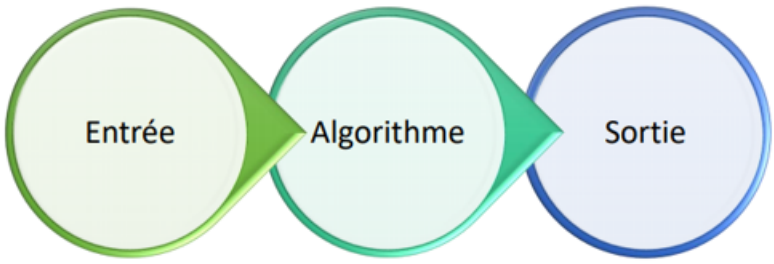
1.1 Définition

Définition 4.1

Un algorithme est une procédure pas-à-pas de résolution d'un problème s'exécutant en un temps fini.



Un délicieux algorithme :-)



La plupart des algorithmes transforment des données d'entrée en des données de sortie.

1.2 Les points essentiels d'un algorithme

La preuve : Il faut être en mesure de prouver que l'algorithme résoud bien le problème pour lequel il a été écrit

L'arrêt : Il faut prouver que l'algorithme s'arrête en un nombre fini d'étapes, et ceci dans tous les cas de figure.

Le choix des structures de données : La façon d'organiser et d'utiliser les données influence l'écriture et la performance de l'algorithme.

Son efficacité : Il est très important d'avoir une idée du comportement de l'algorithme lorsque la taille des données passée en argument augmente.

1.3 Le pseudo code

1.3.1 Définition

En programmation, le pseudo-code, également appelé LDA (pour Langage de Description d'Algorithmes) est une façon de décrire un algorithme en langage presque naturel, sans référence à un langage de programmation en particulier.
L'écriture en pseudo-code permet de comprendre les difficultés de la mise en œuvre de l'algorithme et de développer une démarche structurée dans la construction de celui-ci.
En effet, son aspect descriptif permet de décrire avec plus ou moins de détail l'algorithme, permettant de ce fait de commencer par une vision très large en s'affranchissant de la syntaxe des langages de programmation.

Il n'existe pas de réelle convention standardisée pour le pseudo-code. Nous utiliserons le pseudo-code utilisé à L'Ecole Supérieure d'Informatique.
Les mots-clés seront écrits en majuscules et soulignés Les variables ne contiendront aucun espace, aucun caractère accentué, ne commenceront pas par un chiffre et seront écrites en minuscules Les commentaires seront précédés par //.

1.3.2 Le pseudo-code

Afficher un texte : ECRIRE "Veuillez entrer les nom et prénom"

Afficher une variable ECRIRE nom,prenom

Afficher une variable et du texte ECRIRE "Les contenus de val1 et val2",val1," et ",val2

Déclarer les variables et affectation (internes et externes) :

```
    val1:entier     // déclaration de variable , ici val1
                    //est du type entier
val2:reel
flag:logique
nom,prenom:chaine
val1 <- 5    // Affectation interne, ici val1 prend  //la valeur 5
val2 <-5.5
flag <- VRAI
LIRE nom, prenom  // Affectation externe:  //l'utilisateur entre les valeurs
```

Les opérateurs arithmétiques :

*	multiplication
/	division
+	addition ou concaténation
-	soustraction
MOD	reste d'une division
DIV	division entière
**	exposant

>	plus grand que (greater than)
<	plus petit que (less than)
>=	plus grand ou égal à (greater than or loose-equals)
<=	plus petit ou égal à (less than or loose-equals)
=	égalité
!=	inégalité

Les opérateurs logiques :

ET	et
OU	ou inclusif
XOU	ou exclusif
NON	inverse de

Les boucles conditionnelles

```
SI (val1 > 5) ALORS
   ECRIRE "La valeur",val1,"est supérieure à 5"
FINSI
SI (val1 > 5) ALORS
   ECRIRE "La valeur",val1,"est supérieure à 5"
SINON
   ECRIRE "La valeur",val1,"est égale ou inférieure à 5"
FINSI
SI (val1 > 5) ALORS
   ECRIRE "La valeur",val1,"est supérieure à 5"
SINONSI (val1=5) ALORS
   ECRIRE "La valeur",val1,"est égale à 5"
SINON
   ECRIRE "La valeur",val1,"est inférieure à 5"
FINSI
```

Les boucles :

```
TANT QUE (cpt < val1)
      ECRIRE "Tour n°",cpt
   Cpt <- Cpt + 1
FINTANT

POUR(i DE 0 A 5 PAR 1) FAIRE
    ECRIRE "Tour n°",i
FINPOUR
```

Les tableaux :

```
montab1:tableau[0,15]  de entier  // pour déclarer un tableau (une liste  // en pyt
montab2:tableau[0,5]  de chaine
montab3:tableau[0,9]  de reel
montab4:tableau[0,126]  de logique

montab:tableau[0,5] de entier   //pour initialiser le tableau avec des 0
i:entier
i <- 0
TANT QUE(i <= 5) FAIRE
 montab[i] <-0
 i <-i + 1
FINTANT


POUR(i DE 0 A 5 PAR 1 FAIRE)     // pour parcourir un tableau
 ECRIRE "Veuillez entrer une valeur"
 LIRE montab[i]
FINPOUR
  //Affichage du tableau
POUR(i DE 0 A 5 PAR 1 FAIRE)
 ECRIRE montab[i]
FINPOUR
```

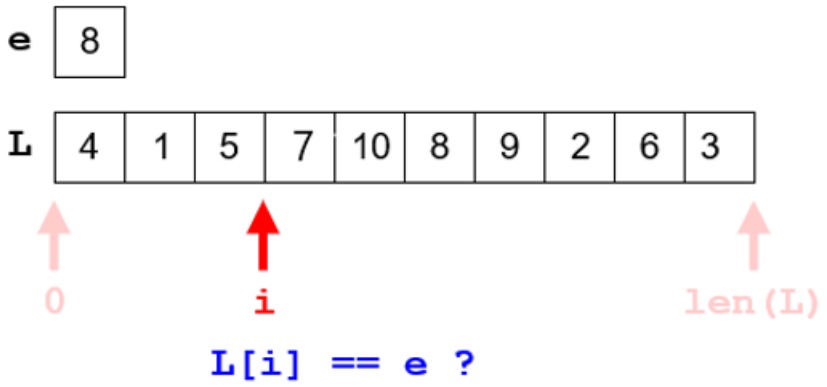
Les fonctions ou procédures :

```
FONCTION politesse()      //Procédure sans arguments
ECRIRE "Bonjour"
FIN FONCTION

politesse()    // pour appeler la fonction ou la procédure politesse

FONCTION calcul(val1:entier, val2:entier)
   res:entier
   res = val1 + val2
   RETOURNER res
FIN FUNCTION

val1, val2, res:entier   //Programme principal
LIRE val1, val2
res = calcul(val1,val2)
```

2 recherche d'occurrences

La recherche d'une occurrence!consiste à déterminer la position (l'indice i) d'un élément e présent dans la liste L. On parle d'occurrence de e dans L.

Exemple : Je recherche l'occurrence de 8 dans la liste L :



Exercice 4.1

Ecrire un algorithme de recherche d'une occurrence utilisant une boucle POUR.
L'algorithme est une fonction qui prend en argument un tableau d'entiers et une valeur entière, et qui renvoie l'occurrence de cette valeur (FAUX si la valeur n'est pas présente dans le tableau).

Exercice 4.2

Ecrire un algorithme de recherche d'une occurrence en utilisant une boucle TANT QUE.
L'algorithme est une fonction qui prend en paramètres un tableau d'entiers et une valeur entière, et qui renvoie l'occurrence de cette valeur (FAUX si la valeur n'est pas présente dans le tableau)