Introduction aux algorithmes

1. L'algorithme et son environnement

1.1. Définition de l'algorithme

Le mot algorithme vient du nom d'un mathématicien, Al Khawarismi dont le traité d'algèbre décrit des procédés de calcul à suivre étape par étape pour résoudre des problèmes qui se ramènent souvent à la résolution d'équations.
Exemple : On considère l'équation $4x + 3 = 2x - 7$
Suivre les instructions ci-contre :

- retrancher 3 dans les deux membres
- retrancher 2x dans les deux membres
- diviser par 2 les deux membres
- écrire la solution de l'équation

Un algorithme *est une suite d'instructions élémentaires qui s'appliquent dans un ordre déterminé à des données et qui fournissent en un nombre fini d'étapes des résultats.*

Exemple
Faire un bon café

1. Faire chauffer de l'eau
    1.1 prendre une casserole
    1.2 mettre de l'eau dans la casserole
        1.2.1 mettre la casserole sous le robinet
        1.2.2 ouvrir le robinet
        1.2.3 remplir la casserole
        1.2.4 fermer le robinet
    1.3 déposer la casserole sur un feu de la cuisinière
    1.4 allumer un feu de la cuisinière
    1.5 attendre l'ébullition
    1.6 éteindre le feu de la cuisinière
2. Mettre le café
3. Remuer
4. Servir

Chaque instruction peut être divisée en une suite d'instructions élémentaires

1.2. Définition du programme

Les instructions doivent être formulées dans un langage compréhensible par l'exécutant. Dans le cas d'un humain, il s'agira du langage courant (langue maternelle) ; mais dans le cas d'une machine, il faudra recourir à un langage de **programmation** (pascal, fortron, C, java...).
Un programme est la traduction d'un algorithme dans un langage de programmation compréhensible par la machine.

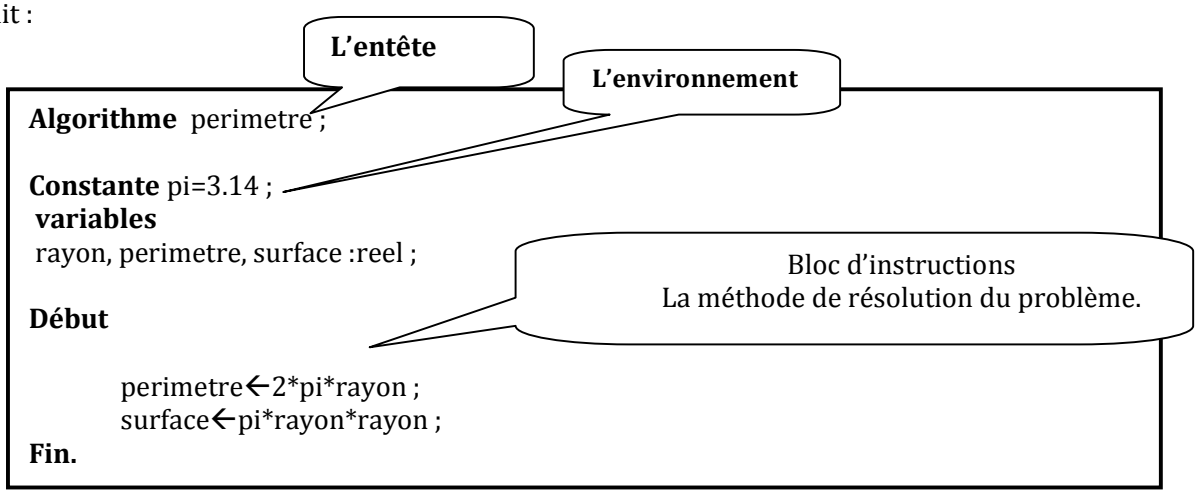
1.3. La structure d'un algorithme

Pour décrire les traitements et les objets manipulés par ces traitements, nous utilisons un langage algorithmique.
La résolution d'un problème est décrite par :

- Un ***en-tête*** qui sert à identifier la tache à résoudre.
- L'***environnement*** de résolution du problème.
- L'***algorithme*** qui décrit la méthode de résolution du problème.

L'entête est introduit par le mot clé ***algorithme*** suivi par un nom de la tache considérée.
Pour définir les objets de l'environnement, les langages de programmation utilisent des instructions déclaratives. Dans le langage que nous utilisons, l'environnement est représenté par le mot réservé **Variables**.

Exemple :
Le calcul du périmètre et de la surface d'un cercle dans le langage algorithmique est définit comme suit :



2. La déclaration d'un objet

Lorsqu'on déclare un objet dans son environnement, on le définit par certaines des ces caractéristiques en fonction de sa nature.
Selon la nature, un objet est soit une constante soit une variable.
On déclare une variable en définissant son nom et son type. Sa valeur, appelée à varier, ne la caractérise pas.
Une constante nommée est caractérisée par un nom et sa valeur qui ne peut en aucun cas être modifiée.
Le nom d'un objet doit respecter **une syntaxe particulière**. Il est constitué d'une suite de caractères et de chiffres commençant **par une lettre** ; il peut aussi contenir le caractère de soulignement.
Exemple :
rayon, valeur_absolue, x1 sont des noms corrects.
1x n'est par un nom correct car il doit commencer par une lettre.
Racine carrée n'est par un nom correct car il contient l'espace.
Dans l'écriture d'un algorithme, on prendra l'habitude de préciser les noms des objets utilisés en indiquant leurs types (nombre, chaîne de caractère, etc.). Cette étape est appelée déclaration des variables.
Pour déclarer une variable dans un algorithme, on utilise le mot clé « **variable »** selon la syntaxe suivante :

```
Variable
Nom_variable : type de donnée ;
```

Pour déclarer une variable dans un programme en pascal, on utilise le mot clé « **var »** selon la syntaxe suivante :

```
Var
Nom_variable : type de donnée ;
```

Pour déclarer une constante dans un algorithme, on utilise le mot clé « **constante »** selon la syntaxe suivante :

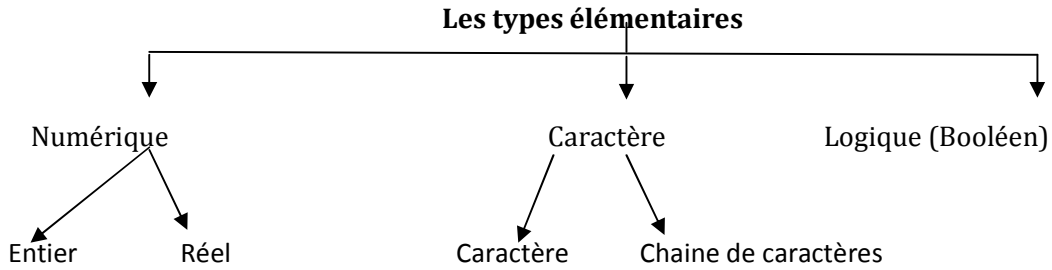
Constante
Nom_constante =valeur ;

Pour déclarer une constante dans un programme en pascal, on utilise le mot clé « **constante »** selon la syntaxe suivante :

Const
Nom_constante =valeur ;

2.1. Les types de données élémentaires

Un type est élémentaire, s'il définit un ensemble de valeurs simple. Les types prédéfinis disponibles dans les langages de programmation sont les types : numériques (entiers et réels), les caractères, les chaines de caractères et le type logique (booléen).



Type	Caractéristiques	Opérateurs Mathématique
Entier	- Définit un domaine de valeur limité, Contrairement à l'ensemble des entiers qui est infini. Le domaine de valeurs du type entier est un intervalle fermé bornés	– L'addition (+). – La soustraction (-) – La multiplication (*) – La division entière (euclidienne) notée par le mot clé ***div*** – L'opérateur modulo (reste de la division entière) par le mot clé ***mod***
Réel	Le type réel prend ses valeurs dans un sous ensemble des réels. On utilise le point ou la virgule ?? comme marque décimale.	– L'addition (+). – La soustraction (-) – La multiplication (*) – La division(/) – Les fonctions mathématiques prédéfinies. Exemple : sqrt().
Caractère	Comporte les lettres, les chiffres, les signes de ponctuations (. , ?, !, etc), les symboles utilisés en tant qu'opérateurs (+, *, -, /, <, =, etc), les caractères spéciaux (%, @, $, &, etc) Un caractère est encadré par 2 apostrophes.	–
Chaine de caractères	Est une suite de caractères. une chaine est encadrée par 2 apostrophes. Toute apostrophe figurant dans une chaine est doublée.	– Les fonctions prédéfinies (Exemple : Length())

Logique (Booléen)	Les deux valeurs de ce type sont notées ***vrai*** et ***faux.***	– La conjonction (***et)*** – La disjonction (***ou)*** – La négation (***non)*** – Les opérateurs de relation (=, <, >, ≠, ≤, ≥)

3. Les instructions élémentaires

Une instruction élémentaire est une instruction exprimée par un ordre simple. Toute instruction est définie par :

- Une syntaxe qui précise le formalisme d'écriture adopté pour formuler l'ordre.
- Une sémantique qui définit les actions effectuées par la machine en réponse à cet ordre.

3.1. L'affectation

Une **variable** est comme une boîte, repérée par un nom, qui va contenir une information. Pour utiliser le contenu de cette boîte, il suffit de l'appeler par son nom.
L'affectation permet de stocker, dans une variable, une valeur décrite par une expression.

Affectation du nombre 5 à la variable x

Syntaxe :

En langage algorithmique, l'instruction d'affectation est donnée par la syntaxe suivante :

X←E ;

Avec X : nom de variable.
←: le symbole de l'affectation.
E : une expression.

On dit que la variable X reçoit la valeur de l'expression E.

En langage Pascal, l'instruction d'affectation est symbolisée par **« := »** selon la syntaxe suivante :

X :=E ;

Sémantique :

L'exécution de l'affectation s'effectue en deux étapes :

1- L'expression E est évaluée, soit v la valeur obtenue.
2- La valeur v est stockée dans la variable X.

L'affectation n'est valide que si le type de l'expression est compatible avec le type de la variable X.

Exemples :

Soit A une variable de type entier dont l'état à un instant donné est schématisé par :

A | 32 | L'exécution de l'opération A←5 a pour effet : A | 5 |

La valeur 5 écrase la valeur 32 qui est alors définitivement perdue.
Si B est une variable de type entier,

B | 9 | A | 5 |

L'exécution de l'opération B←A a pour effet :

B	5

On remarque que cette affectation est sans effet sur la variable A.

Si X est une variable de type réel, l'instruction X←A est valide. Elle respecte l'inclusion de l'ensemble des entiers dans l'ensemble des réels. Le processeur convertit la valeur 5 en réel avant de l'affecter à la variable X.

Par contre l'affectation A←X provoque une erreur pour incompatibilité de type.

L'exécution de l'opération B←(B-2)*A a pour effet :

A	5	B	15

La valeur initiale de la variable B (5) est utilisée dans le calcul de l'expression dont la valeur est ensuite affectée à cette variable.

Incrémentation et décrémentation

Avec la variable A de type entier, l'instruction A←A+1 est appelée opération d'incrémentation. On dit qu'on incrémente A. l'opération inverse de l'incrémentation est la décrémentation A←A-1.

3.2. Instruction de lecture

L'instruction de lecture permet à l'utilisateur de transmettre des données au processeur

Syntaxe :

En langage algorithmique, l'instruction de lecture est donnée par la syntaxe suivante :

Lire (X) ;

Avec X : nom de variable.

On peut regrouper plusieurs ordres de lecture en un seul : lire (A, B, C) est équivalente à lire(A) ; lire(B) ; lire(C).

En langage Pascal, l'instruction de lecture est donnée par la syntaxe suivante :

read(X) ;

Sémantique :

La valeur transmise par l'utilisateur, par l'intermédiaire du dispositif d'entrée (clavier) est affectée à la variable X.

3.3. L'instruction d'écriture

L'instruction d'écriture permet au processus de communiquer à l'utilisateur des messages ou des résultats de traitement.

Syntaxe :

En langage algorithmique, l'instruction d'écriture est donnée par la syntaxe suivante :

Ecrire(E) ;

Avec E : Expression.

On peut regrouper plusieurs ordres d'écriture en un seul : écrire (E1, E2, E3) est équivalent à écrire(E1) ; écrire(E2) ; écrire(E3).

En langage Pascal, l'instruction d'écriture est donnée par la syntaxe suivante :

Write(E) ;

Sémantique :

L'exécution de l'instruction écrire(E) s'effectue en 2 étapes :

1- L'expression E est évaluée.
2- Sa valeur est communiquée à l'utilisateur par l'intermédiaire du dispositif de sortie (écran ou imprimante).

Exemples :

Ecrire (2*x-3) a pour effet d'afficher la valeur de l'expression 2*x-3 ; si x vaut 10, par exemple, cette instruction a pour effet d'afficher la valeur 27.

Ecrire (' Donner un entier supérieur à 10') a pour effet d'afficher la constante chaine :

Donner un entier supérieur à 10

4. Exemple récapitulatif

L'exemple suivant, représente un algorithme (ainsi que le programme équivalent en langage Pascal) permettant de calculer et d'afficher le périmètre et la surface d'un cercle dont le rayon est saisi par l'utilisateur.

```
Algorithme perimetre_surface_cercle ;
Constante
pi=3.14 ;
variable
r, p, s :reel ;
Début
Ecrire ('donner la valeur du  rayon du cercle') ;
Lire (r) ;
p←2*pi*r ;
s←pi*r*r*;
Ecrire('le périmètre du cercle est :', p) ;
Ecrire('la surface du cercle est :', s) ;
Fin.
```

```
Program perimetre_surface_cercle ;
Const
pi=3.14 ;
Var
r, p, s :real ;
Begin
Write ('donner la valeur du  rayon du cercle') ;
Read (r) ;
p :=2*pi*r ;
s :=pi*r*r*;
Write('le périmètre du cercle est :', p) ;
Write('la surface du cercle est :', s) ;
End.
```