# Logique algorithmique

## 1 Introduction

### 1.1 Voici un algorithme

```
Algorithme nimbSimple()
Variable // déclaration des variables utilisées dans l'algorithme
        Joueur, ordi : entier
        total : entier
Début // début de l'algorithme
        // initialisation des variables et présentation du programme
        joueur <-0
        ordi <- 0
        total <-15
        Afficher ("bienvenue dans le nimb")
        Afficher ("Il y a ",total, " allumettes en jeu")
        Afficher ("le joueur commence")
        Répéter // début du jeu
                // affichage des condition de départ du tour joueur et question
                Afficher ("il reste", total, " allumettes")
                Saisir ("joueur, combien d'allumettes enlevez-vous ? ",joueur)
                total <- total-joueur // calcul du reste d'allumettes
                // calcul du résultat.
                Si (total<=0) Alors
                        Afficher ("le joueur a perdu")
                Sinon
                        // c'est au tour de l'ordi
                        Si (total<=1) Alors
                                Afficher ("l'ordinateur a perdu")
                                ordi<-total // l'ordi enlève le reste des allumettes
                        Sinon
                                ordi <- ent(total/3) // calcul du nombre d'allumettes à enlever
                                Si (ordi>1) Alors
                                        ordi<-3
                                Sinon
                                        ordi<-total-1
                                FinSi
                                Afficher ("l'ordinateur enlève ", ordi, " allumettes")
                        FinSi
                        total<- total-ordi // calcul du total restant
                FinSi
        Jusqu'à ((total<=0) ou (joueur=99))
Fin
```

### 1.2 Algorithme, qu'est ce que c'est ?

Une recette de cuisine, un guide d'utilisation de votre dernier lecteur MP3 ou d'un répondeur sont des exemples d'algorithme que vous avez probablement déjà exécuté.

Vous avez déjà indiqué le chemin à une personne, expliqué où cherché un objet à votre mère dans votre tanière qui sert de chambre ? Vous avez créé et fait exécuter des algorithmes.

L'algorithme est une suite d'instructions qui, exécutées correctement, doit donner le résultat attendu.

Si l'algorithme est juste, le résultat obtenu est celui que vous vouliez atteindre. Sinon, au mieux, le résultat est aléatoire ...

Coup de chance, un ordinateur exécutera vos ordres sans réfléchir et, contrairement au touriste, il franchira le pont en travaux.

Par ailleurs, si vous dites "aller à droite" avant de dire "aller tout droit jusqu'au carrefour" le chmin emprunté sera différent et donc le lieu final aussi.

De plus, un ordinateur étant une machine, elle risque de se manger le mur à la première instruction (' y a pas de carrefour au début ...).

Un algorithme est donc une décomposition d'un problème qui nécessite de mettre en place une logique dans la succession des actions élémentaires. Cette logique est appelée séquentielle

### 1.3 Algorithmique, langage ou langue ?

Si le touriste est allemand, vous ne lui indiquerez pas le chemin en chinois. De même, vous lui direz plutôt de prendre à droite que de prendre le cap 090.

Les instructions doivent être compréhensibles par la personne ou par le dispositif qui l'exécutera.

Pour que les rédacteurs d'un algorithme puissent se faire comprendre des lecteurs et exécutants, il convient de mettre en place un formalisme suffisant pour exprimer différents concepts ou actions.

Remarque : L'algorithmique est issue de la programmation et non l'inverse, donc le formalisme se rapproche des langages de programmation, tout en gardant une distance et une souplesse d'écriture suffisante pour pouvoir être traduit dans différents langages.

Les opérations élémentaires se rapprocheront donc de celles que l'on utilise avec une calculatrice (opérations de calcul, mise en mémoire, affichage et saisie de valeurs).

Cependant, certains langages n'utiliser que ponctuellement l'algorithmique car leur approche est différente de cette démarche séquentielle (langages objets ou évènementiels).

## 2 Logique algorithmique

### 2.1 Exemples

#### 2.1.1 Écrire le prénom de son interlocuteur sur une feuille

Pour écrire le prénom de son voisin sur une feuille, soit on connaît son voisin, soit on ne le connaît pas.

Dans ce dernier cas, avant d'écrire le prénom sur la feuille, il faut le lui demander (logique, non ?).

Il existe donc bien un ordre logique et séquentiel entre les opérations que nous souhaitons effectuer.

Cet ordre appartient à la notion de logique algorithmique

#### 2.1.2 Exemple : Calculer le volume d'un tube

Pour calculer le volume d'un tube, on va décomposer le problème en problèmes plus simples : Soit le tube (ci-contre) de diamètres extérieur et intérieurs de 10mm et 6mm et d'une hauteur de 5mm.

Calculons son volume.

Le volume est la surface de l'anneau, multipliée par la hauteur du tube (5).

A-t-on le volume ou reste-t-il des inconnues ?

La surface de l'anneau est la surface du cercle extérieur moins la surface du cercle intérieur

La surface du cercle extérieur est $\pi$*diamètre extérieur du tube (10) au carré divisé par 4

La surface du cercle intérieur est $\pi$*diamètre intérieur du tube (6) au carré divisé par 4

C'est bon, nous avons résolu le problème en le décomposant.

Mettons les choses dans l'ordre de calcul :

1. calcul de la surface du cercle extérieur= $\pi$*6*6/4
2. calcul de la surface du cercle intérieur= $\pi$*6*6/4
3. surface de l'anneau = surface du cercle extérieur - surface du cercle intérieur
4. volume du tube = surface de l'anneau * 5

Si le calcul se fait en commençant par la fin, on risque de ne pas pouvoir effectuer ce calcul (on ne met pas la charrue avant les bœufs !).

De plus, les opérations utilisables sont d'un niveau très "bas", ce cont les opérations mathématiques qui sont les instructions élémentaires.

### 2.1.3 Procédure de gestion des clients.

Le cycle des documents entre un client et une organisation est le suivant :

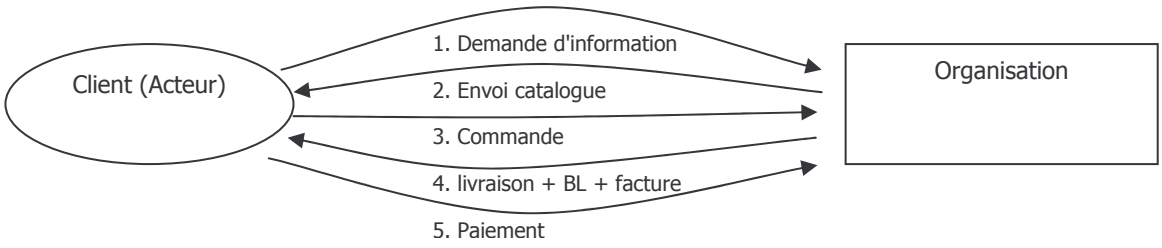
Lorsqu'un client fait appel à l'organisation pour avoir des renseignements.

L'organisation renvoie un catalogue et des informations publicitaires

Le client émet alors une commande de différents articles que l'organisation enregistre.

L'organisation prépare la commande, imprime le bon de livraison et la facture correspondants à la commande puis envoie le tout au client.

Le client effectue le paient qui est enregistré et encaissé.




Ceci est une suite d'actions élémentaires effectuées dans un ordre précis. Si on change l'ordre d'une de ces étapes, la succession de ces opérations est faussée.

Ceci est un algorithme informel.

## 2.2 Définition de l'algorithme

L'algorithmique est une méthode de décomposition d'un problème complexe en actions élémentaires visant à apporter une solution à ce problème.

Un algorithme est une suite logique et ordonnée d'actions élémentaires.

On peut alors s'apercevoir que tout est décomposable en algorithmes.

# 3 Données traitées : variables, types et expressions

Les données traitées en algorithmique sont aussi élémentaires que les actions disponibles et les traitements qu'on va leur faire subir (rassurez vous, elles ne souffrent pas, seuls nos cerveaux sont mis à rude épreuve et les ordis ne font que ce qu'on leur dit, un point, c'est tout).

## 3.1 Comme dans la vie courante

On écrit un algo "comme" dans la vie courante. Et dans celui-ci …

Comme dans la vie courante, nous utilisons des chiffres et des nombres pour effectuer des calculs. On disposera donc de chiffres et de nombres, entiers ou réels.

Exemples :

Entiers : 1, 2, 3, -1 ;

Bornes : 0 à 65 535 ou -32 767 à +32 767 ; 0 à 4 294 967 296 ou -2 147 483 648, +2 147 483 648 (il y a de la marge …)

Réels : 1, 2, 3, -1, 2.5, 3.141592653 ou le résultat de calculs 22/7 ou 355/113

(pi= Que j'aime à faire apprendre un nombre utile aux sages, Immortel Archimède, artiste, ingénieur, Qui de ton jugement peut priser la valeur ? Pour moi ton problème eut de pareils avantages, …)

Comme dans la vie courante, nous disposons de caractères pour former des mots puis des phrases qui ne sont que des chaînes de caractères. On disposera donc de chaînes de caractères, délimitées par des guillemets.

Exemples :

Caractères : "a", "b", "A", "é", "&", "§", "#", " " (espace), "" (vide)

Chaîne : "oui", "Oui", "OUI", "toto", "Que j'aime à faire apprendre un nombre utile aux sages"

Petit poème, plus complet
Que j'aime à faire apprendre un nombre utile aux sages !
Glorieux Archimède, artiste ingénieux,
Toi de qui Syracuse aime encore la gloire,
Soit ton nom conservé par de savants grimoires !
Jadis, mystérieux, un problème bloquait
Tout l'admirable procédé, l'œuvre grandiose
Que Pythagore découvrit aux anciens Grecs.
Ô, quadrature ! Vieux tourment du philosophe !
Insoluble rondeur, trop longtemps vous avez
Défié Pythagore et ses imitateurs.
Comment intégrer l'espace bien circulaire ?
Former un triangle auquel il équivaudra ?
Nouvelle invention : Archimède inscrira
Dedans un hexagone, appréciera son aire,
Fonction du rayon. Pas trop ne s'y tiendra
Dédoublera chaque élément antérieur ;
Toujours de l'orbe calculée approchera ;
Définira limite ; enfin, l'arc, le limiteur
De cet inquiétant cercle, ennemi trop rebelle !
Professeur, enseignez son problème avec zèle ! ...

Comptez les lettres, vous obtiendrez (à peu près) PI

Les 100 premières décimales de Pi :
3,141 592 653 589 793 238 462 643 383 279 502 884 197 169 399 375 105 820 974 944 592 307 816 406 286 208 998 628 034 825 342 117 067 (527 autre décimales visibles au palais de la découverte de Paris)

Comme dans la vie courante, nous disposons d'opérateurs arithmétiques (+ - / *) pour faire nos calculs. On disposera donc d'opérateurs arithmétiques que l'on utilisera de la même façon ou comme pour les calculatrices (sauf anciennes HP)

Exemples :

3*3, 22/7, 3+2*6-1, (3+2)*(6-1)

Comme dans la vie courante, nous disposons d'opérateurs logiques (=, <, >, …) pour construire des conditions et des connecteurs logiques (Et, Ou, Non) pour assembler nos conditions.

Exemples :

2,718<e et 2,719>e, nom="Tom" ou nom="Jerry"

Le résultat d'une comparaison et d'une expression logique est du type VRAI/FAUX ou OUI/NON. C'est un résultat binaire ou booléen.

On rencontre donc en algo différents types de données élémentaires et différents opérateurs. Ils servent à construire des expressions ou des conditions

## 3.2 Types des données

Les types élémentaires utilisés (et utilisables) en algo sont les suivants :

Entier : nombre sans décimales

Réel : nombre pouvant avoir une partie décimale

Caractère : un caractère isolé

Chaîne (de caractères) : une suite de caractères

Booléen : une valeur VRAI/FAUX ou OUI/NON (pour plus d'info, rechercher "algèbre de bool" sur le net)

## 3.3 Opérateurs de traitement

Opérateurs arithmétiques : calculs

+ : addition

- : soustraction

* : multiplication (ne pas utiliser le x)

/ : division (utiliser la barre de fraction, pas le signe ÷)

^ : puissance x^y = x à la puissance y (2^3 = 8 ; rarement utilisé quand y=2 ou 3).

Opérateurs logiques : comparaisons

= : égalité

<, <= ; >, >= ; <> : inférieur, inférieur ou égal ; supérieur, supérieur ou égal ; différent

Connecteurs logiques : connecte deux comparaisons

ET : les deux comparaisons connectées doivent être vraies pour que le résultat soit vrai

OU : les deux comparaisons connectées doivent être fausses pour que le résultat soit faux.

NON : inverse le résultat d'une comparaison

Note : on utilisera aussi des parenthèses pour marquer les priorités, comme en mathématique ou sur la majorité des calculatrices.

## 3.4 Expressions

Une expression est un calcul arithmétique, alphabétique ou logique selon le type général du résultat.

On parlera d'expression pour les expressions arithmétiques ou alphabétiques et de conditions pour les expressions logiques.

Une expression peut être de différents types :

Expression arithmétique ;

Résultat numérique, sert à manipuler des nombres

3, 3*2, 32*28, (30+2)*(30-2), a+1 sont des expressions arithmétiques.

Expression alphabétique ;

Résultat alphabétique, sert à manipuler des caractères ou des chaînes

"titi", " le canari", "titi"+" le canari" sont des expressions alphabétiques

Expressions logiques ;

Résultat booléen (VRAI ou FAUX et pas autre chose, en algo), sert à construire des conditions.

1=1, 3<4, 3>=4, a<>b ; a OU b, a ET b sont des expressions logiques

*Remarque : Ne pas dire "c'est une variable constante", mais dire "c'est une constante" (sinon ça fera bien rire vos collègues.)*

## 3.5 Variables

Reprenons le calcul du volume du tube :

1. calcul de la surface du cercle extérieur= π*10*10/4
2. calcul de la surface du cercle intérieur= π*6*6/4
3. surface de l'anneau = surface du cercle extérieur - surface du cercle intérieur
4. volume du tube = surface de l'anneau * 5

Dans cet algo, on effectue des calculs intermédiaires que l'on garde en mémoire pour pouvoir les réutiliser plus loin. C'est le rôle principal d'une variable.

Ainsi, on peut dire que "surface du cercle extérieur" est une variable.

Une variable possède différentes caractéristiques :

- un nom, appelé aussi identificateur, étiquette, …
- un type, appelé aussi domaine de valeur, définissant ce que peut contenir la variable
- une valeur qui peut varier au cours de l'algorithme.

Note Importante : la valeur d'une variable est inconnue au départ de l'algo. Il convient donc de l'initialiser.

Ainsi, on indiquera, AVANT de commencer l'algorithme, la liste des variables que l'on va utiliser dans un paragraphe consacré à la déclaration des variables.

### 3.5.1 Remplir une variable : affectation (affecter une valeur à une variable)

On rempli une variable avec une valeur.

On modifie sa valeur en ÉCRASANT L'ANCIENNE valeur.

On utilisera le signe <- pour désigner cette affectation (le signe = est un opérateur de comparaison)

Exemples :

a<- 15 // la variable a prend la valeur 14. 14 est une expression dont le résultat est numérique (14)

a<- 0 // l'ancienne valeur est écrasée et remplacée par 0

a<- 3*2 // a prend la valeur de l'expression 3*2, c'est à dire 6

nom <- 'titi' // nom prend la valeur titi

b<- a // b prend la valeur CONTENUE DANS a (6)

c<-a+1 // c est affecté de la valeur a+1, càd du contenu de a plus 1 (6+1, soit 7)

a<-a+1 // cette dernière signifie que a est incrémentée de 1 (soit 6+1, donc 7)

personnage <- nom + "le canari" // personnage contient : "titile canari"

personnage <- nom + " le canari" // personnage contient : "titi le canari"

a=b // CECI N'EST PAS UNE AFFECTATION mais une comparaison

La valeur de l'expression est calculée, puis le résultat est affecté à la variable.

Pour généraliser, on dira que l'on affecte le résultat d'une expression à la variable.

Nom_de_variable <- expression

### 3.5.2 Nommons les variables

On peut donner (presque) n'importe quel nom aux variables.

Donc utilisons des noms significatifs !

Exemple :

nom, description, prime, nombre1, nombre2, tauxInteret

Cependant, il est interdit d'utiliser des espaces ou autres caractères spéciaux (', ", @, /, §, …) dans les noms (les seuls autres caractères que a à z et 1 à 0 sont – (tiret) et _ (souligné))

Par habitude, on écrit souvent le nom d'une variable avec plusieurs mots (salaire brut). Ces mots sont collés ensemble et on met une majuscule pour marquer chaque mot sauf au début où un nom de variable commence toujours par une minuscule (salaireBrut).

Note : ce formalisme est celui que j'utilise dans ce cours mais ce n'est qu'une convention d'écriture et non une obligation. J'y dérogerai afin d'habituer le lecteur aux différentes conventions.

### 3.5.3 Exemples

Reprenons le calcul du volume du tube et formalisons. Nous obtenons le résultat suivant :

```
diametreExt<-10 // initialisation de la variable
diametreInt<- 6 // initialisation de la variable
hauteurTube <- 5 // initialisation de la variable
surfaceCercleExt <- π* diametreExt * diametreExt /4
surfaceCercleInt <- π* diametreInt * diametreInt /4
surfaceAnneau <- surfaceCercleExt – surfaceCercleInt
volumeTube <- surfaceAnneau * hauteurTube
```

## 3.6 Variable et constantes

Dans certains cas, il est plus pratique d'utiliser des valeurs par leur nom que par leur Valeur.

C'est souvent le cas de Pi ou du taux de TVA (s'il est unique)

On déclarera alors non des variables, mais des constantes comme suit :

```
Constante
        PI=3,1415926538979
```

Ou

```
Variable
        PI : réel <- 3.14159265358979323846264338327 95 // constante Pi
```

Ça revient presque au même que déclarer une variable et l'initialiser dans l'algorithme

```
Variable
        PI : réel
Début
        PI <- 3.14159265358979323846264338327 95 // constante Pi
```

Dans le premier cas, l'ordinateur, au moment de l'exécution de l'algorithme, remplacera, toutes les fois où apparaît le nom de la constante, le nom de la constante par sa valeur.

Dans les autres cas, la constante sera une variable comme les autres, au programmeur de veiller qu'elle reste inchangée.

Personnellement, je n'utilise pas de constantes, mais toujours la troisième solution, éventuellement la seconde car une constante n'est autre qu'une variable dont la valeur ne change pas. Cependant, je respecte la casse majuscule pour le nom de la variable.

## 3.7 Formalisons 1

Un algorithme sert à traiter des données.

Il est le résultat d'une suite ordonnée de traitements élémentaires.

Un algorithme informel est la description de ces traitements.

Un algorithme formel respecte une convention d'écriture, un langage et sera écrit comme suit :

```
Algorithme nom_de_l'algorithme
Variable
        // liste de définition des variables
        nom : chaîne
```

```
        salaire, tauxInteret : Réel
Début
        {liste des traitements}
Fin
```

On notera :

Le nom de l'algorithme, précédé du mot clé 'algorithme'

Le paragraphe où sont définies les variables, intitulé 'variables'

Le paragraphe où sont définis les traitements, encadré par 'début' et 'fin'

Le traitement de base : l'affectation (remplissage) des variables : variable <- expression

Autre vision :

Comme une dissertation, un algorithme est composé de plusieurs parties : un **titre**, une **introduction**, un **développement**, une **conclusion**.

**Le titre** est le nom de l'algorithme. Il peut comporter une liste de paramètres nécessaires à son fonctionnement.

**L'introduction**, ou Description, est une description globale comprenant celle du service rendu et des résultats qu'il produit, des modifications éventuelles effectuées sur les paramètres

**Le développement** est la liste des tâches (instructions) effectuées par l'algorithme. Il est composé de plusieurs sous parties (cf. plus loin).

**La conclusion** (optionnelle) décrit les résultats s'ils ne l'ont pas déjà été dans l'introduction.

Le **développement** est composé d'une partie déclarative et d'un corps (exécutif ;-)

La **partie déclarative** sert à définir les nom et types des variables manipulées dans le corps et est identifiée par le mot clé **var**.

Le **corps** contient l'ensemble des instructions à effectuer pour remplir le service défini. Il est entouré des mots clé **début** et **fin**.

Ces instructions sont exécutées dans l'ordre de l'algorithme et toute erreur dans le séquencement de ces instructions peut provoquer des erreurs dans le calcul du résultat.

NOTE : respectez les retraits en début de ligne, cela améliore la lisibilité, c'est déjà assez complexe comme ça, pas besoin, en plus, de 'crypter' un algo en écrivant comme un analphabète.

### 3.7.1 Exemples d'algorithmes formels simples

Calcul de la surface du cercle

```
Algo surfaceCirculaire
Variables
        Pi = 3.141592635
        Surface, diamètre : réel
Début
        Surface <- PI*diamètre*diamètre/4
Fin
```

Échange de deux caractères :

```
Algo echange
Variables
        a, b, tampon : caractère
Début
        // Initialisation
        a<-"A"
        b<-"B"
        // Échange
        tampon <- a
        a<-b
        b<-tampon
        // Fin de l'échange
```

```
Fin
```

Calcul du volume du tube :

```
Algo volumeTube
Var
        diametreExt, diametreInt, hauteurTube, volumeTube : réel
        surfaceCercleExt, surfaceCercleInt, surfaceAnneau
Début
        diametreExt <-10 // initialisation de la variable
        diametreInt <- 6 // initialisation de la variable
        hauteurTube <- 5 // initialisation de la variable
        surfaceCercleExt = π* diametreExt * diametreExt /4
        surfaceCercleInt = π* diametreInt * diametreInt /4
        surfaceAnneau = surfaceCercleExt – surfaceCercleInt
        volumeTube = surfaceAnneau * hauteurTube
Fin
```

Ne remarqueriez vous pas un problème ?

a) pour modifier les conditions, il faut changer l'algorithme
b) lorsque l'algorithme a fait son boulot, l'utilisateur n'en est pas informé

C'est alors que le lire, la saisie et le afficher sont arrivés (petite musique à l'harmonica en bruit de fond)

# 4 Les entrées/sorties

Les entrées/sorties sont des instructions élémentaires pour transmettre des données à un algorithme et de restitution des résultats, en cours d'exécution.

Pour bien les comprendre, il faut se placer comme si nous étions l'ordinateur et que nous n'avions, pour communiquer avec l'utilisateur, que deux petites fenêtres.

Ainsi, nous aurons une fenêtre LIRE pour lire ce qu'un utilisateur a écrit à l'aide du clavier et une fenêtre ECRIRE, sur laquelle on écrit les résultats présentés à l'utilisateur.

## 4.1 Entrée : saisir ou lire

Pour entrer une donnée, on utilisera l'instruction SAISIR.

SAISIR sera suivi du nom d'une (seule) variable entre parenthèse.

```
Saisir (prénom)
```

## 4.2 Sortie : afficher ou écrire

Pour afficher un résultat, on utilisera l'instruction AFFICHER

AFFICHER sera suivie d'une liste d'éléments textuels à afficher

Exemples :

```
Afficher ("bonjour")
Afficher (prénom)
Afficher ("bonjour ", prénom, ", comment allez-vous ?")
Afficher ("Salaire Mensuel : ", salaireAnnuel/12, "€")
```

On peut intercaler des textes et des expressions, les variables n'ont pas d'unité, elles doivent être gérées manuellement.

De la même façon, cette instruction ne gère pas les espaces entre les données affichée et il faut eaussi les gérer manuellement.

Exemples :

```
Afficher ("bonjour", prénom)
```

Donne le résultat : Bonjourtoto

et

```
Afficher ("bonjour ", prénom)
```

Donne le résultat : Bonjour toto

## 4.3 Remarque Importante ; Les alternatives d'écriture de saisir et afficher

Certains professionnels, professeurs et auteurs d'ouvrages autorisent de mettre plusieurs variables derrière SAISIR.

```
Saisir (nom, prénom)
```

Certains professeurs et auteurs d'ouvrages utilisent aussi LIRE de la même façon que SAISIR.

```
Lire (prénom)
```

D'autre encore considèrent que lire est une fonction qui retourne une valeur et utilisent alors lire de la façon suivante :

```
Variable <- lire()
```

Cette façon est trop proche d'un certain langage de programmation et je ne l'utiliserai qu'à titre d'exemple de notations.

Certains professionnels, professeurs et auteurs d'ouvrages autorisent de mettre un texte affiché avant la lecture des variables dans l'instruction SAISIR.

```
Saisir ("Donnez vos nom et prénom", nom, prénom)
```

Le texte est affiché puis les variables sont saisies. Cette instruction en regroupe deux (afficher puis saisir) ce qui est un peu complexe comme instruction. Je ne l'utiliserai qu'à titre d'exemple.

Certains professionnels, professeurs et auteurs d'ouvrages utilisent aussi ECRIRE de la même façon que AFFICHER.

```
Écrire ("bonjour ", prénom, " ", nom)
```

## 4.4 Formalisons avec les exemples vus plus haut

Calcul de la surface du cercle

```
Algo surfaceCirculaire
Variables
        PI = 3.141592635
        surface, diamètre : réel
Début
        // Affichage et saisie des données
                Écrire ("Donnez le diamètre du cercle")
                Lire (diamètre)
        // Traitement et calculs
                surface <- PI*diamètre*diamètre/4
        // Affichage des résultats
                Écrire ("La surface du cercle est de :" , surface)
Fin
```

Échange de deux caractères :

```
Algo echange
Variables
        a, b, tampon : caractère
Début
        // Initialisation : saisie des deux nombres
                Saisir ("Donner un 1er caractère pour la variable a  :", a)
                Saisir ("Donner un 2nd  caractère pour la variable b :", b)
        // Échange
                Afficher ("je commence l'échange")
                tampon <- a
                a<-b
                b<-tampon
                Afficher ("j'ai fini l'échange")
        // Fin de l'échange, affichage des résultats
                Afficher ("la variable a contient maintenant : ", a)
```

```
                Afficher ("la variable b contient maintenant : ", b)
                Afficher ("le tampon contient : ", tampon)
        Fin
```

Calcul du volume du tube :

```
Algo volumeTube
Var
        diametre_exterieur, diametre_exterieur, hauteur_tube : réel
        surface_cercle_extérieur, surface_cercle_interieur
Début
        // Saisie des informations :
        Afficher ("donnez les diamètres intérieurs puis extérieur et la hauteur du tube :")
        Saisir (diametreExt , diametreInt, hauteurTube)

        // Calculs
        surfaceCercleExt <- π* diametreExt * diametreExt /4
        surfaceCercleInt <- π* diametreInt * diametreInt /4
        surfaceAnneau <- surfaceCercleExt – surfaceCercleInt
        volumeTube <- surfaceAnneau * hauteurTube

        // Restitution des résultats
        Afficher ("le volume du tube est de : ", volumeTube)
Fin
```

### 4.5 Conclusion

Les entrées et sorties sont des instructions élémentaires qui permettent à un algorithme de communiquer à l'extérieur directement avec le clavier et l'écran.

En règle générale il y a deux moments d'utilisation de ces instructions dans un algorithme : en début d'algo ou de traitement (initialisation des variables) et en fin de traitement ou d'algo (restitution des résultats).

## 5 Résumé

| Groupe | Description banale | Description informaticienne | Instruction |
|---|---|---|---|
| La remarque | Commentaire pour expliquer l'algo | Ligne de remarque ou de commentaire | // … ; /* … */ |
| L'affectation | Remplir une boite avec une valeur | Affecter une valeur à une variable | … <- … |
| Entrées/Sorties | Capter ou restituer des informations | Les entrées/sorties, lecture/écriture | Lire, écrire ; Saisir, afficher |
| Alternatives | Choix d'un traitement, selon des conditions données à vérifier | Conditionnelle ou alternative | Si (condition) Alors<br>Traitement_si_condition_vraie<br>Sinon<br>Traitement_si_condition_fausse<br>FinSi |

Les expressions sont de deux types : arithmétique ou booléenne

Les expressions arithmétiques sont des calculs ou affectations

Les expressions booléennes sont des comparaisons

## 6 Applications

Dans certains exemples, j'ai utilisé différentes notations des entrées/sorties afin de présenter les différentes façons de les écrire.

## 6.1 Se faire cuire une omelette

Faites la liste des opérations nécessaires à la confection d'une omelette pour 4 personnes, du point où l'on casse les œufs jusqu'à la dégustation.

Les ingrédients sont les suivants : 2 œufs par personnes, du lait, une pincée de sel, une de poivre, une cuiller à soupe d'herbes de Provence, du fromage découpé en grains, des pommes de terre.

## 6.2 La chasse à l'ours-mouth géant

Explications

Dans cet exercice, nous utilisons 3 objets (unHomme, unOursMouth et unPiège). Chacun de ses objets est caractérisé par un certain nombre de comportements, appelés méthodes. Par exemple, l'objet homme possède une méthode tuer. Une méthode peut avoir besoin de paramètres particuliers que l'on écrit entre parenthèses.

On écrira : unHomme.tuer(unOursMouth) pour dire que la méthode tuer de l'objet unHomme est appelée avec le paramètre unOursMouth. Ce que fait la méthode est ici explicite dans son nom. Comment est écrite la méthode importe peu pour l'instant.

Classez la liste des tâches à exécuter pour chasser un ours-mouth géant par ordre d'exécution.

- unHomme.tuer(unOursMouth)
- unHomme.attendre(unOursMouth)
- unOursMouth.naître()
- unHomme.poser(unPiège)
- unHomme.fabriquer(unPiège.nouveau())
- unPiège.capturer(oursMouth)
- unHomme.naître()

## 6.3 La cueillette du pissenlit sauvage

Classez la liste des tâches à exécuter pour cueillir du pissenlit sauvage par ordre d'exécution.

- unPissenlit <- uneFemme.rechercher('Pissenlit')
- uneFemme.fabriquer(unPanier.nouveau())
- uneFemme.cueillir(unPissenlit)
- unEnvironnement.nouveau()
- unPissenlit.eclore(unEnvironnement)
- uneFemme.naître(unEnvironnement)
- uneFemme.porter(unPanier)

Note : la flèche <- représente une affectation de valeur :

a<- b aura pour résultat que a prend la valeur contenu dans b

## 6.4 Vive la neige

Un enfant souhaite fabriquer un bonhomme de neige. Voici ce que lui conseille un adulte :

1- Faire le corps
2- Faire la tête
3- Poser la tête sur le corps
4- Trouver un chapeau, une écharpe, deux cailloux, une carotte, une pipe, un balai
5- Placer le chapeau sur la tête
6- Enrouler l'écharpe entre la tête et le corps
7- Enfoncer les cailloux à l'emplacement des yeux
8- Planter la carotte à l'emplacement du nez
9- Planter la pipe à hauteur de la bouche
10- Planter le balai dans le bas du corps

A faire :

En examinant les conseils de l'adulte, pouvez-vous

- trouvez leur **but** ?
- déterminer quels sont les objets que l'on manipule ?
- **décomposer** la fabrication du bonhomme de neige ?

Reprendre ces actions, et les développer si nécessaire en **actions** plus **élémentaires** en utilisant le formalisme vu dans la chasse à l'ours-mouth géant.

## 6.5 Dis bonjour au monsieur

Ce programme va lire les noms et prénoms d'un utilisateur puis afficher "bonjour ..." suivi du prénom puis du nom de l'utilisateur.

```
Algorithme bonjour()
/* Description : Cet algorithme effectue la comparaison entre deux entiers.
        Il affiche l'entier maximum des deux */
Var     // déclaration des variables
        nom, prénom : texte, nom et prénom à afficher
Début   // corps
        Ecrire ("quel est votre nom ?")   // Affichage du texte de la question
        Lire (nom)                        // lecture du nom
        Ecrire ("et votre prénom ?")      // Affichage du texte de la question
        Lire (prénom)                     // lecture du prénom
        Ecrire("Bonjour, ", prénom, nom) // Affichage du texte complet : "Bonjour, " suivi des contenus de prénom et nom
Fin
```

## 6.6 Calcul de la surface d'un cercle, version autonome

```
Algorithme surfaceCirculaire()
/* Description :   Cet algorithme calcule la surface d'un cercle en fonction de son rayon : S=PI*R^2. */
Var     // déclaration des variables
        surface, rayon: réel, variables surface et rayon
        PI : réel <- 3.14159265358979323846264338327795 // constante Pi
Début   // corps
        Ecrire("Donnez le rayon du cercle")
        Lire(rayon)
        surface <- 2*PI* rayon ^2
        Ecrire("La surface du cercle est : ", surface)
Fin
```

## 6.7 Calcul de la moyenne de deux nombres

```
Algorithme moyenne()
/* description :   Cet algorithme calcule et affiche la moyenne de deux nombres lus au clavier */
Var
        nombre1, nombre2, moyenne : entier
Début   // corps
        Ecrire ("entrez un entier")
        Lire (nombre1)
        Ecrire ("entrez un second entier")
        Lire (nombre2)
        moyenne <- (nombre1 + nombre2)/2
        Ecrire("la moyenne est : ", moyenne)
Fin
```

## 6.8 Calcul des valeurs de la répartition secondaire

```
repartitionSecondaire()
/* Description
        Cet algorithme sert à calculer deux montants pour une répartition secondaire des charges indirectes
        Il résout un système de 2 équations du 1er degré à 2 inconnues :
                x=ay+b ; y=cx+d
        avec :
                pa : les pourcentage de répartition du premier centre sur le second (en valeur absolue),
                pc : le pourcentage de répartition du second centre sur le premier (en valeur absolue)
                b : le total des charges indirectes de la répartition primaire du premier centre auxiliaire à répartir
                d : le total des charges indirectes de la répartition primaire de second centre auxiliaire
        le résultat est : x = (ad+b)/(1-ac) ; y = cx+d
```

```
*/
Var         // déclaration des variables
            a,b,c,d, x, y : réels, données absolues
            pa,pb : réels, pourcentages
Début
            // saisie du premier centre
            Ecrire("Donnez le poucentage du premier centre (/ex 10,3)")
            Lire(pa)
            Ecrire("entrez le montant des charges du premier centre de la répartition primaire")
            Lire(b)
            // saisie du second centre
            Ecrire("Donnez le pourcentage du second centre (/ex 10,3)")
            Lire(pc)
            Ecrire("entrez le montant des charges du second centre de la répartition primaire")
            Lire(d)

            // calculs
            a <- pa/100       // calcul de a puis c
            c <- pc/100
            x <- =(ad+b)/(1-ac)          // calcul de x
            y <- cx+d                    // calcul de y

            // affichage des résultats
            Ecrire("Les montants à répartir sont : ")
            Ecrire("Pour le premier centre : ", x)
            Ecrire("Pour le second centre  : ", y)
Fin
```

Passons à la suite : Algorithme structuré