Algorithmie

I. Introduction

Un Algorithme est une suite d'instructions à résoudre un problème.

Exemple :

Problème : j'ai envie de faire pipi

Solution : Aller aux toilettes

II. Pseudo code / Algobox

```
Début algo
        Se lever
        Aller aux toilettes
        Se déshabiller
        PISSER
        Tirer la chasse
        Se rhabiller
        Se laver les mains
Fin algo

Variables
        Ingredient1
        Ingredient2
        Ingredient3
Début Algo
        Sortir ingredient1
        Sortir ingredient2
        Mélanger ingredient1 avec ingredient2 donne ingredient3
        Mettre ingredient3 au four
        Sortir la préparation
Fin algo

Variables
        chiffre1 : entier
        chiffre2 : entier
        resultat : entier
Debut Algo
        chiffre1 = Saisir « Le premier chiffre ? »
        chiffre2 = Saisir « Le deuxième chiffre ? »
        resultat = chiffre1 * chiffre2
        Afficher resultat
Fin algo
```

Cf 1-multiplication.alg

III. Variables

Les variables dans un programme sont typées. Types primitifs :

- Chaines de caractères
- Entier
- Décimal

- Booléan

Dans les langages comme PHP ou javascript (faiblement typé), ce n'est pas obligatoire de typer les variables.

Dans d'autres langages comme Java (fortement typé), c'est obligatoire.

IV. Structures conditionnelles

1. Le SI/SINON (le IF/ELSE in english)

Tester une condition, puis exécuter des instructions si la condition est vraie, avec la possibilité d'exécuter d'autres instructions est fausse.

```
Variables
        age : entier
Debut Algo
        age = Saisir « Quel est votre âge ? » : LIRE age
        SI age >= 18
                Afficher « Vous êtes majeur »
        SINON
                Afficher « Vous êtes mineur »
        FIN SI
Fin algo
```

Cf 2-isMajeur.alg

On peut cumuler les conditions grâce aux opérateurs logiques ET et OU.
On peut prioriser les OU grâce aux parenthèses.

```
Exemple :
Debut Algo
        SI 5 == 5 ET 6 < 10
                // on rentre
                Afficher « 5 est égal à 5 ET 6 est inférieur à 10
        FIN SI

        SI 5 == 5 OU 6 > 10
                // on rentre
                Afficher « 5 est égal à 5 OU 6 est supérieur à 10
        FIN

        SI 5 == 5 ET 3 > 5 OU 9 == 9
                // on rentre : le premier ensemble lié par un ET est faux, mais ensuite le deuxième est vrai. Comme on a un OU, l'ensemble des conditions est vrai.
        FIN SI

        SI 5 == 5 ET (3 > 5 OU 9 == 8)
                // on rente pa
        FIN SI
Fin algo
```

2. Le SELON (switch in english)

```
Variable :
        note : entier
Debut Algo
        note = saisir « Quelle note as-tu eue ? »
        SELON note
                CAS 5 :
                        Afficher « c'est pas très bien »
                CAS 10 :
                CAS 11 :
                CAS 12 :
                        Afficher « c'est moyen »
                CAS 20 :
                        Afficher « pas mal »
                DEFAUT :
                        Afficher « laisse moi tranquille avec tes notes de merde »
        FIN SELON
Fin algo
```

3. Le SINON SI

Au lieu d'écrire ce code pas lisible (et degueu, soyons clair)

```
Debut algo
        SI note == 5
                Afficher « »
        SINON
                SI note == 10 ou note == 11 ou note == 12
                        Afficher « »
                SINON
                        SI note == 20
                                Afficher « »
                        SINON
                                Afficher « »
                        FIN SI
                FIN SI
        FIN SI
Fin algo
```

```
Debut algo
        SI note == 5
                Afficher « »
        SINON SI note == 10 ou note == 11 ou note == 12
                Afficher « »
        SINON SI note == 20
                Afficher « »
        SINON
                Afficher « »
        FIN SI
Fin algo
```

4. Utilisation des booléens et condition ternaire

- Retourner une condition / ensemble de conditions

```
Variables
        majeur : booleen
Debut fonction estMajeur(age : entier)
        // version 1
        Si age >= 18
                majeur = vrai
        SINON
                majeur = faux
        FIN SI

        Retourner majeur

        // version 2
        SI age >= 18
                retourner vrai
        SINON
                retourner faux
        FIN SI

        // version 3 : la condition étant elle-même la valeur vrai ou faux
        // on peut retourner directement la condition
        retourner age >= 18
Fin fonction
```

- Affecter une valeur grâce à une condition ternaire

Une condition ternaire permet d'affecter une valeur à une variable en fonction d'une condition, le tout en une seule instruction.

```
Variables
Debut algo
        Lire age
        majeur = estMajeur(age)

        // version 1
        SI majeur == vrai
                texte = « Vous êtes majeur »
        SINON
                texte = « vous êtes mineur »
        FIN SI

        afficher texte

        // version 2
        SI majeur
                texte = « Vous êtes majeur »
        SINON
                texte = « vous êtes mineur »
        FIN SI

        afficher texte

        // version 3 : opérateur ternaire : « ? » et « : »
```

```
        // v1
        texte = (majeur == vrai) ? « Vous êtes majeur » : « Vous êtes mineur »
        // v2
        texte = majeur ? « Vous êtes majeur » : « Vous êtes mineur »

        afficher texte
Fin algo
```

V. La concaténation

```
Variables :

        Chaine1, chaine2, chaine3 : chaines de caractères

Debut Algo
        Chaine1 = «Bonjour»
        Chaine2 = «tout»
        Chaine3 = «le monde»
        Afficher chaine1 CONCAT chaine2 CONCAT chaine3
        // affiche «bonjourtoutle monde»
        Afficher chaine1 CONCAT « » CONCAT chaine2 CONCAT « » CONCAT chaine3 CONCAT « !»
        // affiche «bonjour tout le monde !»
Fin algo
```

1. Exercice

Ecrire l'algo qui demande à l'utilisateur un chiffre entre 1 et 9, sinon message erreur et fin programme.

S'il a bien saisi un chiffre correct, afficher la table de multiplication sous ce format :

```
1 * 2 = 2
2 * 2 = 4
…
9 * 2 = 18
```

```
Variables :
        Chiffre, resultat, resultat2, resultat3 : entier
Debut algo
        chiffre = saisir « Veuillez saisir un chiffre entre 1 et 9 »

        SI chiffre < 1 OU chiffre > 9
                Afficher « Mauvaise saisie »
        SINON
                // affichage de la table
                resultat = 1 * chiffre
                afficher « 1 * » CONCAT chiffre CONCAT « = » CONCAT resultat

                resultat2 = 2 * chiffre
                afficher « 2 * » CONCAT chiffre CONCAT « = » CONCAT resultat2

                resultat3 = 3 * chiffre
                afficher « 3 * » CONCAT chiffre CONCAT « = » CONCAT resultat3

                // la même pour les 6 lignes restantes de la table de multiplication
```

```
        FIN SI
Fin algo
```

Cf 3-table-multiplication.alg

VI. Structures itératives

POUR (FOR) / TANT QUE (WHILE) / REPETER JUSQU'A (DO WHILE)

- Boucle POUR

```
Debut algo
        // cette boucle affiche 950 fois le message
        POUR i de 0 à 949 PAR PAS DE 1
                Afficher « Je dois pas tricher »
        FIN POUR

        // cette boucle afficher 475 fois le message
        POUR i de 0 à 949 PAR PAS DE 2
                Afficher « Je dois pas tricher »
        FIN POUR
Fin algo
```

- TANT QUE

```
Debut Algo
        // même exemple que pour la boucle POUR (attention le contexte fait que la boucle pour est
à privilégier ici)
        i = 0
        TANT QUE i <= 949
                Afficher « Je dois pas tricher »
                i = i + 1 // incrémenter i : la nouvelle valeur est égale à la valeur actuelle + 1
        FIN TANT QUE

        // exemple où la boucle TANT QUE est préconisée
        chiffre = saisir « Veuillez saisir un chiffre entre 1 et 9 »
        TANT QUE chiffre < 1 OU chiffre > 9
                chiffre = saisir « Veuillez saisir un chiffre entre 1 et 9 »
        FIN TANT

        // autre exemple avec nombre de tours inconnus
        somme = 0
        chiffreAleatoire = RANDOM(1, 10)
        somme = somme + chiffreAleatoire

        i = 0
        TANT QUE somme < 100
                chiffreAleatoire = RANDOM(1, 10)
                somme = somme + chiffreAleatoire
                i = i + 1
        FIN TANT QUE

        Afficher « Nombre de tours de boucle : » CONCAT i
```

Fin algo

- REPETER JUSQU'A

```
Debut algo
        i = 0
        REPETER
                Afficher « Je dois pas tricher »
                i = i + 1
        TANT QUE i <= 949

        REPETER
                chiffre = saisir « Veuillez saisir un chiffre entre 1 et 9 »
        TANT QUE chiffre < 1 OU chiffre > 9
Fin algo
```

Boucle POUR : à utiliser quand on connait précisément le nombre d'itération à faire
Boucle TANT QUE : quand on ne sait pas le nombre d'itération à l'avance et qu'il est possible qu'il n'y en ait aucune
Boucle REPETER : quand on ne sait pas à l'avance le nombre d'itération mais qu'il y en a au moins une

- Table de multiplication avec boucle

```
Variables :
        chiffre, resultat: entier
Debut algo
        REPETER
                chiffre = saisir « Veuillez saisir un chiffre entre 1 et 9 »
        TANT QUE chiffre < 1 OU chiffre > 9

        // arrivé à cette ligne de code, la variable chiffre est forcément comprise entre 1 et 9

        // affichage de la table : 9 itérations (une pour chaque ligne de la table de multiplication)
        POUR i de 1 à 9 PAR PAS de 1
                resultat = i * chiffre
                afficher i CONCAT « * » CONCAT chiffre CONCAT « = » CONCAT resultat
         FIN POUR
Fin algo
```

- Exercices

1 - Demander à l'utilisateur de saisir 5 notes
Afficher la moyenne de ces notes

- Version sans boucle
- Version avec boucle

```
Variables
        note1 : entier
        note2 : entier
        note3 : entier
        note4 : entier
        note5 : entier
```

```
    note : entier
    somme : entier
    moyenne : nombre
Debut algo
    // 1- version sans boucle
    Lire note1
    Lire note2
    Lire note3
    Lire note4
    Lire note5
    somme = note1 + note2 + note3 + note4 + note5
    moyenne = somme / 5
    afficher moyenne

    // sans variable intermédiaire
    moyenne = (note1 + note2 + note3 + note4 + note5) / 5
    afficher moyenne

    // afficher directement la moyenne
    Afficher (note1 + note2 + note3 + note4 + note5) / 5

    // 1bis – Code qui permet de visualiser qu'une boucle est faisable
    somme = 0

    Lire note
    somme = somme + note

    Lire note
    somme = somme + note

    Lire note
    somme = somme + note

    Lire note
    somme = somme + note

    Lire note
    somme = somme + note

    moyenne = somme / 5

    // 2 – version avec boucle
    somme = 0

    POUR i de 0 à 4 PAR PAS de 1
        Lire note
        somme = somme + note
    FIN POUR

    moyenne = somme / 5

Fin algo
```

2- Demander à l'utilisateur de saisir des notes.
C'est lui qui décide quand la saisie s'arrête en tapant -1
Afficher la moyenne de ces notes

```
Variable
Debut algo
        // VERSION REPETER
        somme = 0
        nbNotes = 0

        REPETER
                Lire note
                SI note != -1
                        nbNotes = nbNotes + 1
                        somme = somme + note
                FIN SI
        TANT QUE note != -1

        moyenne = somme / nbNotes

        // VERSION TANT QUE
        somme = 0
        nbNotes = 0

        Lire note
        TANT QUE note > -1
                nbNotes = nbNotes + 1
                somme = somme + note
                Lire note
        FIN TANT QUE

        moyenne = somme / nbNotes
Fin algo
```

VII. Les tableaux (ou array)

Les tableaux peuvent avoir une liste de valeurs. On va accéder à chacune des valeurs du tableau, grâce à un indice, le premier élément étant accessible à l'indice 0.

```
Variables
        notes : tableau

        // langage fortement typé : un tableau ne peut avoir qu'un seul type de valeur
        notes : tableau de entier

        i : entier
        valeur : entier
        note : entier
Debut Algo
        // ajouter un élément dans un tableau : push
```

```
        Ajouter 20 dans notes
        Ajouter 13 dans notes
        Ajouter 10 dans notes

        // à cette ligne de code, le tableau notes possède trois éléments

        // accéder à un élément d'un tableau
        // afficher le deuxième élément
        Afficher notes[1]
        // on compare la deuxième note
        SI notes[1] == 13

        FIN SI

        // modifier un élément
        notes[2] = 11

        // afficher/parcourir tous les éléments d'un tableau
        POUR i de 0 à 2 PAR PAS de 1
        FIN POUR

        // parcourir un tableau avec indice max dynamique
        POUR i de 0 à LONGUEUR(notes)-1
                Afficher notes[i] CONCAT « <br> »
        FIN POUR

        // parcourir avec un foreach (POUR CHAQUE)
        POUR CHAQUE valeur DANS notes
                Afficher valeur
        FIN POUR CHAQUE

        POUR CHAQUE note DANS notes
                Afficher note
        FIN POUR CHAQUE
Fin algo
```

VIII. Les fonctions

1. Introduction

Une fonction est une série d'instructions à laquelle on va donner un nom, afin de pouvoir l'utiliser à plusieurs reprises sans avoir à la réécrire.

Exemple :

```
Variables
Debut Fonction direCoucou
        Afficher « Coucou »
        Afficher « toi ça va»
Fin fonction

Variables
Debut algo
```

```
    direCoucou()
    direCoucou()
Fin algo
```

2. Variables locales

Les variables locales sont des variables qui n'existent que dans leur contexte, ici dans le contexte de la fonction. Elles sont déclarées au moment où on exécute la fonction, et elles sont détruites à la fin de la fonction.

Dans l'algo principal, il peut y avoir des variables qui ont le même, il n'y aucune interaction/conflit entre les deux.

```
Variable
    note1 : entier // variables locales
    note2 : entier
    somme : entier
Debut fonction addition
    note1 = 5
    note2 = 10
    somme = note1 + note2
Fin fonction

Variables
    // ici somme est dans l'algo principal, pas de conflit avec la variable « somme » de la fonction
    somme : entier
Debut algo
    addition()
    addition()
Fin algo
```

3. Retour d'une valeur

Une fonction peut retourner une valeur, c'est une valeur qui pourra être dans l'utilisateur à l'endroit où la fonction a été appelée.

Une fonction se termine soit à la fin des instructions, soit à la première instruction « retourner ».

```
Variable
    note1 : entier // variables locales
    note2 : entier
    somme : entier
Debut fonction addition
    note1 = 5
    note2 = 15
    somme = note1 + note2
    // la fonction s'arrête immédiatement après un retourner
    retourner somme
Fin fonction

Variable
    date : chaine de caractères
Debut fonction getCurrentDate
    date = day() CONCAT « / » CONCAT month() CONCAT « / » CONCAT year()
    retourner date
```

```
Fin fonction


Variables
        total : entier
Debut algo
        addition() // valeur de retour pas utilisée

        // afficher une valeur retourner par une fonction
        afficher addition() // affiche 20

        total = addition()
        afficher total // affiche 20

        today = getCurrentDate()
        afficher today

        today2 = getCurrentDate()
        afficher today2
Fin algo
```

4. Paramètres d'entrées

Une fonction peut avoir des paramètres d'entrées, ce sont des valeurs attendues lors de l'appel de la fonction. Les paramètres peuvent être obligatoire (par défaut), ou facultatifs.

```
Variables
        somme : entier
Debut fonction addition(nombre1 : entier, nombre2 : entier)
        somme = nombre1 + nombre2
        retourner somme
Fin fonction

Variables
        calcul : entier
Debut algo
        note1 = 45
        note2 = 3
        addition(note1, note2)

        afficher addition(34, 89) // affiche 123

        calcul = addition(10, 5)
        afficher calcul // affiche 15

        SI addition(3, 2) > 4
                // condition vraie, on rentre ici
        FIN SI
Fin algo
```

5. Exercices

- Créer une fonction qui multiplie deux nombres (passés en paramètre) entre eux et retourne le résultat. Utiliser cette fonction dans un algo principal et afficher le résultat.

```
Variables
        produit : entier
Debut fonction multiplication (nb1 : entier, nb2 : entier)
        produit = nb1 * nb2
        retourner produit
Fin fonction

Variables
        nombre1 : entier
        nombre2 : entier
Debut Algo
        Lire nombre1
        Lire nombre2
        Afficher multiplication(nombre1, nombre2)

        resultat = multiplication(22, 3)
        ficher resultat
Fin algo
```

- Créer une fonction qui prend un âge en paramètre et retourne si oui ou non cet âge est celui d'une personne majeure. Utilisez cette fonction dans un algo principal, et afficher « Vous êtes majeur » si renvoie vrai, « Vous êtes mineur ».

```
Variables
Debut fonction estMajeur(age : entier)
        // doit retourner un booléen
        SI age >= 18
                retourner vrai
        SINON
                retourner faux
        FIN SI
Fin fonction

Variables
        age : entier
        majeur : booleen
Debut algo
        // utiliser correctement ce booléen pour afficher le message
        Lire age
        majeur = estMajeur(age)

        Si majeur == vrai
                Afficher « vous êtes majeur »
        SINON
                Afficher « Vous êtes mineur »
        FIN SI
Fin algo
```

- Créer une fonction qui affiche tous les éléments d'un tableau

```
Variables
        note : entier
Debut fonction afficheTab(notes : tableau)
```

```
        POUR CHAQUE note DANS notes
                afficher note
        FIN POUR CHAQUE
Fin fonction
Variables
        listeNotes : tableau
        note : entier
Debut algo
        Lire note
        Ajouter note dans listeNotes

        Lire note
        Ajouter note dans listeNotes

        afficheTab(listeNotes)
Fin algo
```

6. Paramètres facultatifs

On peut passer des paramètres facultatifs à une fonction : il suffit que ce paramètre ait une valeur par défaut dans la signature de la fonction.

Attention : les paramètres facultatifs sont forcément définis en dernières positions.

Exemple :

```
Fonction direCoucouA(prenom : chaine, upper : booleen = vrai, finCoucou :chaine= « ! »)
        SI upper == vrai
                Afficher « Coucou » CONCAT UPPER(prenom) CONCAT finCoucou
        SINON
                Afficher « Coucou » CONCAT prenom CONCAT finCoucou
        FIN SI

Fin fonction


Algo
        // le prénom sera affiché en majuscule car on n'a pas passé le deuxième paramètre
        // il a donc pour valeur par défaut « vrai » (défini dans la signature de la fonction)
        direCoucouA(« Fab »)

        nom = « Toto »
        direCoucouA(nom)

        nom = « Jean »
        direCoucouA(nom, faux)

        nom = « Gérard »
        direCoucouA(nom, vrai, « … »)
Fin algo
```

7. Fonctions récursives

Une fonction récursive, c'est une fonction qui s'appelle elle-même. Tout comme une boucle, il faut une condition d'arrêt, sinon on a une boucle infinie.

Analyse :

Factorielle de 5 : 5! = 1 * 2 * 3 * 4 * 5

1 * 2 = ?
? * 3 = ?
? * 4 = ?
? * 5 = ?

? * 1 = 1
1 * 2 = 2
2 * 3 = 6
6 * 4 = 24
24 * 5 = 120

- Sans récursivité

```
Algo
    Lire entier
    resultat = 1
    POUR i de 1 à entier
        resultat = resultat * i
    FIN POUR

    Afficher resultat
Fin Algo
```

- Avec récursivité

```
Fonction factorielle(chiffre : entier)
    SI chiffre == 1
        // on s'arrête
        retourner 1
    SINON
        // on continue
        // la factorielle d'un chiffre dépend de la factorielle précédente
        // 5 ! = 4 ! * 5
        // 4 ! = 3 ! * 4
        // 3 ! = 2 ! * 3
        // 2 ! = 1 ! * 2
        // 1 ! = 1

        resultat = factorielle(chiffre  - 1) * chiffre
        retourner resultat
    FIN SI
Fin fonction

Fontion afficherPasTriche(nbAffichages)
    SI nbAffichages == 1
        Afficher « Pas triche »
    SINON
        affichePasTriche(nbAffichages - 1)
    FIN SI
Fin Fonction
```

IX. Les structures

Les structures sont des types de variable que le développeur va lui-même définir. Elles permettent de regrouper des données se référant au même concept dans une seule variable, ce qui aura pour effet d'améliorer la lisibilité du code et sa maintenance/évolutivité. Avec ce type de variables, on commence à se rapprocher d'un développement objet (même si on n'y est pas encore 😊)

- Sans structure

```
Variables

Debut algo
        nom = « toto »
        email = « toto@pouet.fr »
        age = 25

        nom2 = « abdel »
        email2 = « abdel@mail.fr »
        age2 = 32

        afficher nom CONCAT email
        isMajeur(age2)
Fin algo
```

- Avec structure

```
Variable
        Personne : structure {
                nom : chaine
                email : chaine
                age : entier
        }

        personne1 : Personne
        personne2 : Personne
Debut algo
        personne1.nom = « Toto »
        personne1.email = « toto@pouet.fr »
        personne1.age = 25

        personne2.nom = « abdel »
        personne2.email = « abdel@mail.fr »
        personne2.age = 32

        afficher personne1.nom CONCAT personne1.email
        isMajeur(personne2.age)
Fin algo
```

X. « Connexion » à une base de données

```
Variables
        players : tableau de Player
        Player : structure {
                nom : chaine,
                points : entier
        }
Debut algo
        // récupération des données en bdd
        Connexion au serveur de bdd (« localhost », « root », « 1234 »)
        Choix de la base (« formation_m2i »)
        players = Requête SQL (« SELECT name, points FROM player ORDER BY points DESC »)

        // affichage des joueurs dans la page
        POUR i de 0 à LENGTH(players)-1
                Afficher players[i].name CONCAT « : » CONCAT players[i].points CONCAT « <br> »
        FIN POUR

        POUR CHAQUE player DANS players
                Afficher player.name CONCAT « : » CONCAT player.points CONCAT « <br> »
        FIN POUR CHAQUE
Fin algo
```

XI. Exercices

1. Exercice 1

Demander à l'utilisateur deux nombres, et afficher le résultat de la soustraction.

```
Variables
        nombre1 : nombre
        nombre2 : nombre
        difference : nombre
Début algo
        nombre1 = saisir « Veuillez saisir le premier nombre »
        nombre2 = saisir « Veuillez saisir le deuxième nombre »
        difference = nombre1 – nombre2
        afficher difference
Fin algo
```

2. Exercice 2

Demander à l'utilisateur deux nombres compris entre 5 et 50, et afficher le résultat de la multiplication.

```
Variables
        nombre1 : nombre
        nombre2 : nombre

Début algo
        // nombre 1 : répéter ou tant que ?
        REPETER
                nombre1 = saisir « Veuillez saisir le premier nombre »
        TANT QUE nombre1 < 5 OU nombre1 > 50

        nombre1 = saisir « Veuillez saisir le premier nombre »
        TANT QUE nombre1 < 5 OU nombre1 > 50
                nombre1 = saisir « Erreur, veuillez ressaisir nombre1 »
        FIN TAN QUE

        // idem nombre 2
        REPETER
                nombre2 = saisir « Veuillez saisir le deuxième nombre »
        TANT QUE nombre2 < 5 OU nombre2 > 50

        Afficher nombre1 * nombre2
Fin algo
```

3. Exercice 3

Demander à l'utilisateur deux nombres et afficher le résultat de la division. Vérifiez d'abord si la division est possible.

```
Variables
        numerateur : nombre
```

```
        denominateur : nombre
        quotient : nombre
Début algo
        numerateur = saisir « Saisissez le premier nombre »
        denominateur = saisir « Saisissez le deuxième nombre »
        TANT QUE denominateur == 0
                denominateur = saisir « Division par zéro impossible, ressaisir le deuxième nombre »
        FIN TAN QUE

        quotient = numerateur / denominateur
        afficher quotient
Fin algo
```

Cf. 4-exercice-quotient.alg

4. Exercice 4

Demander à l'utilisateur son âge, et afficher s'il est majeur ou mineur. S'il est mineur, afficher combien d'années il reste avant qu'il soit majeur.

```
Variables
        age : entier
        anneesRestantes : entier
Début Algo
        age = saisissez « Quel est votre âge »
        SI age < 18
                // en une ligne
                Afficher « Vous êtes mineur il vous reste » CONCAT (18 - age) CONCAT « années
avant d'être majeur »

                // détaillé
                Afficher « Vous êtes mineur »
                anneesRestantes = 18 – age
                Afficher « Nombre d'années restantes avant majorité : » CONCAT anneesRestantes
        SINON
                Afficher « Vous êtes majeur »
        FIN SINON
Fin Algo
```

5. Exercice 5

Demander à l'utilisateur un nom de fruit, lui afficher

- « Cool » dans le cas où il dit « fraise »
- « Pas cool » dans le cas où il dit « banane »
- « Etrange » dans le cas où il dit « kiwi »
- « Comme tu veux » dans tous les autres cas

```
Variables
        fruit : chaine de caractères
Début algo
        Lire fruit
```

```
    SELON fruit
        CAS « fraise » :
            Afficher « cool »
            Break
        CAS « banane » :
            Afficher « Pas cool »
            Break
        CAS « kiwi »
            Afficher « Etrange »
            Break
        DEFAUT :
            Afficher « Comme tu veux »
    FIN SELON
Fin algo
```

6. Exercice 6

Demander à l'utilisateur de saisir des notes, et de taper -1 pour stopper.
Stocker chacune des notes saisies dans un tableau. Ensuite (une fois sorti de la boucle),
Afficher chacune des notes précédées de son indice.
Enfin, afficher la moyenne sous ce format : « Moyenne = 39 / 3 = 13 »

```
Variables
Début algo
    // stocker les notes dans le tableau
    Lire note
    TANT QUE note != -1
        Ajouter note dans notes
        Lire note
    FIN TANT QUE

    // afficher les notes, précédées de leur indice dans le tableau
    POUR i de 0 à LONGUEUR(notes) - 1
        Afficher i CONCAT « » CONCAT notes[i]
    FIN POUR CHAQUE

    // calculer la moyenne
    somme = 0
    POUR CHAQUE n DANS notes
        somme = somme + n
    FIN POUR CHAQUE
    Moyenne = somme / LONGUEUR(notes)

    // afficher la moyenne
    Afficher « Moyenne = » CONCAT somme CONCAT « / » CONCAT LONGUEUR(notes) CONCAT
    « = » CONCAT moyenne

    // version une boucle
    somme = 0
    i = 0
    Lire note
    TANT QUE note != -1
```

```
            Ajouter note dans notes
            somme = somme + note
            Afficher i CONCAT « » CONCAT note
            i = i + 1
            Lire note
        FIN TANT QUE

        moyenne = somme / LONGUEUR(notes)
        Afficher « Moyenne = » CONCAT somme CONCAT « / » CONCAT LONGUEUR(notes) CONCAT
        « = » CONCAT moyenne

Fin algo
```

7. Exercice 7

Demander à l'utilisateur son année de naissance, puis son mois de naissance, et enfin son jour de naissance. Calculer alors précisément son âge en années.

anneeActuelle = year()
moisActuel = month()
jourActuel = day()

anneActuelle – anneeNaissance = age (mais pas précis, car l'anniversaire n'est peut-être pas encore passé)

```
Variables
        anneeActuelle : entier
        moisActuel : entier
        jourActuel : entier
        anneeNaissance : entier
        moisNaissance : entier
        jourNaissance : entier
        age : entier
Début Algo
        anneeActuelle = year()
        moisActuel = month()
        jourActuel = day()
        Lire anneeNaissance
        Lire moisNaissance
        Lire jourNaissance

        // version 1
        SI moisActuel < moisNaissance
            // anniv pas passé
            age = anneeActuelle - anneeNaissance - 1
        FIN SI

        SI moisActuel > moisNaissance
            // anniv passé
            age = anneeActuelle - anneeNaissance
        FIN SI
```

```
SI moisActuel == moisNaissance
        SI jourActuel < jourNaissance
                // pas passé
                age = anneeActuelle - anneeNaissance - 1
        FIN SI
        SI jourActuel >= jourNaissance
                // passé
                age = anneeActuelle - anneeNaissance
        FIN SI
FIN SI

// version 2
SI moisActuel < moisNaissance
        age = anneeActuelle - anneeNaissance – 1
SINON SI moisActuel > moisNaissance
        age = anneeActuelle - anneeNaissance
SINON SI moisActuel == moisNaissance
        SI jourActuel < jourNaissance
                age = anneeActuelle - anneeNaissance - 1
        SINON
                age = anneeActuelle - anneeNaissance
        FIN SI
FIN SI

// version 3
age = anneeActuelle - anneeNaissance
SI moisActuel < moisNaissance
        age = age – 1
SINON SI moisActuel == moisNaissance ET jourActuel < jourNaissance
        age = age - 1
FIN SI
```

```
        // version 4
        age = anneeActuelle - anneeNaissance
        SI moisActuel < moisNaissance OU (moisActuel == moisNaissance ET jourActuel < jourNaissance)
                age = age - 1
        FIN SI
Fin Algo
```

8. Exercice 8

Demander à l'utilisateur de saisir un nombre de notes indéterminé (-1 pour stop), en stockant ces notes dans un tableau.

Afficher ensuite la note la plus basse.

Faire deux boucles : une pour remplir le tableau, une autre pour trouver la note la plus basse.

```
Variables
        note : entier
        notes : tableau
        noteMin : entier
Debut algo
        Lire note
        TANT QUE note != -1
                Ajouter note dans notes
                Lire note
        FIN TANT QUE

        // V1 : good mais la première itération ne sert à rien : elle compare la première note avec elle-même
        // on part du principe que la note la plus petite est la première, puis on compare avec le reste
        noteMin = notes[0]
        POUR CHAQUE note DANS notes
                SI note < noteMin
                        noteMin = note
                FIN SI
        FIN POUR CHAQUE

        // V2 : avec un POUR, on débute l'indice à 1 pour éviter de comparer la 1ere avec elle-même
        noteMin = notes[0]
        POUR i de 1 à LONGUEUR(notes) – 1
                SI notes[i] < noteMin
                        noteMin = notes[i]
                FIN SI
        FIN POUR

        Afficher noteMin
Fin algo
```

9. Exercice 9

Demander à l'utilisateur de saisir un nombre de notes indéterminé (-1 pour stop), en stockant ces notes dans un tableau.

Afficher ensuite la note la plus haute.

Faire deux boucles : une pour remplir le tableau, une autre pour trouver la note la plus haute.

```
Variables
        note : entier
        notes : tableau
        noteMax : entier
Debut algo
        Lire note
        TANT QUE note != -1
                Ajouter note dans notes
                Lire note
        FIN TANT QUE

        noteMax  = notes[0]
        POUR i de 1 à LONGUEUR(notes) – 1
                SI notes[i] > noteMax
                        noteMax  = notes[i]
                FIN SI
        FIN POUR

        Afficher noteMax
Fin algo
```

10. Exercice 10

A partir de cet exercice, créez la fonction et utilisez-la dans un algo principal.

Créer une fonction qui soustrait un nombre à un autre et qui retourne la différence.

```
Variables
        resultat : entier
Debut fonction soustraction(nombre1 : entier, nombre2 : entier)
        resultat = nombre1 – nombre2
        retourner resultat
Fin fonction

Variables
        difference : entier
Debut algo
        nb1 = 50
        nb2 = 26
        // affiche 24
        afficher soustraction(nb1, nb2)

        nb3 = 34
        nb4 = 12
        difference = soustraction(nb3, nb4)
        // affiche 22
```

```
    afficher difference
Fin algo
```

11. Exercice 11

Créer une fonction qui dit si oui ou non une chaine de caractère (passée en paramètre) possède 5 caractères. LENGTH(chaine)

```
Variables
    nbChars : entier
Debut fonction hasFiveChars(chaine : chaine de caractère)
    nbChars = LENGTH(chaine)

    // v1
    SI nbChars == 5
        Retourner vrai
    SINON
        Retourner faux
    FIN SI

    // ou v2 : retourne directement la valeur de la condition
    retourner nbChars == 5
Fin fonction

Variables
    word : chaine
Debut algo
    word = « Bonjour »
    SI hasFiveChars(word) == vrai
        Afficher word CONCAT « a 5 caractères »
    SINON
        Afficher word CONCAT « n'a pas 5 caractères »
    FIN SI
Fin algo
```

12. Exercice 12

Créer une fonction qui prend en paramètre un tableau, et qui retourne un nouveau tableau dont tous les éléments du premier ont été inversés. [1,3,2] => [2,3,1]

```
Variables
    newTab : tableau
Debut fonction reverseTab(tab : tableau)
    // pour des tableaux dont on connait le nombre d'éléments
    newTab[0] = tab[2]
    newTab[1] = tab[1]
    newTab[2] = tab[0]

    newTab[0] = tab[6]
    newTab[1] = tab[5]
    newTab[2] = tab[4]
    newTab[3] = tab[3]
    newTab[4] = tab[2]
    newTab[5] = tab[1]
```

```
        newTab[6] = tab[0]

        // inverser les éléments d'un tableau pour toute taille de tableaux sans exception
        indexMax = LENGTH(tab) - 1
        POUR i de 0 à indexMax
                newTab[i] = tab[indexMax-i]
        FIN POUR

        Retourner newTab
Fin fonction

Variables
        tabWords : tableau
        newTab : entier
Algo
        Ajouter « coucou » dans tabWords
        Ajouter « salut » dans tabWords
        Ajouter « bonjour » dans tabWords

        // debug tabWords : [« coucou », « salut », « bonjour »]

        newTab = reverseTab(tabWords)
        // debug newTab : [« bonjour », « salut », « coucou »]
Fin algo
```

13. Exercice 13

Créer une fonction qui prend en paramètre un tableau d'entier et qui retourne la valeur la plus petite.

```
Variables
        minimum : entier
Fonction getSmallest(entiers : tableau)
        minimum = entiers[0]
        POUR i de 1 à LONGUEUR(entiers) - 1
                SI entiers[i] < minimum
                        minimum = entiers[i]
                FIN SI
        FIN POUR

        Retourner minimum
Fin fonction

Variable
        notes, min  : entier
Algo
        Ajouter 20 dans notes
        Ajouter 2 dans notes
        Ajouter 5 dans notes

        // on cherche dans le tableau « notes » l'élément le plus petit grâce à la fonction getSmallest
        min = getSmallest(notes)
        afficher min
```

Fin algo

14. Exercice 14

Créer une fonction servirBoisson qui affiche un l'écran « Voici un café », ou « Voici un thé » ou « Voici un cola ». La fonction prend une chaine de caractère en paramètre : c'est la boisson que l'on veut.

Si le paramètre ne correspond ni à café, thé ou cola, il faut afficher « Désolé, on ne sert pas cette boisson ». Si le paramètre n'est pas passé, il faut servir par défaut un café.

```
Fonction servirBoisson(boisson : chaine= « café »)
        SELON boisson
                CAS « café » :
                        Afficher « Voici un café »
                CAS « thé » :
                        Afficher « Voici un thé »
                CAS « cola » :
                        Afficher « Voici un cola »
                DEFAUT :
                        Afficher « Désolé, on ne sert pas de » CONCAT boisson
        FIN SELON
Fin fonction

Variables
        boisson : chaine
Algo
        // demander une boisson à un client
        Afficher « Tu veux quoi ? »
        Lire boisson

        servirBoisson(boisson) // affiche un des 4 messages en fonction de la saisie utilisateur

        // un client ne parle pas et claque des doigts
        servirBoisson() //  affiche « Voici un café »

        servirBoisson(« thé ») // afficher « Voici un thé »
Fin algo
```

15. Exercice

- Ecrire l'algorithme qui affiche si un mot est un palindrome (kayak, non)
- Adapter ce code pour créer une fonction isPalindrome, qui prend en paramètre une chaine de caractère et retourne si, oui ou non, cette chaine est un palindrome.

Analyse

« salut »
« s » avec « t » // on sait dès la première que le mot n'est pas un palindrome

« kayak »
« k » avec « k »
« a » avec « a »

« y » avec « y »
Correspond donc à tester les indices entre eux :
0 avec 4
1 avec 3
2 avec 2

« bonjour »

0 avec 6
1 avec 5
2 avec 4
3 avec 3 // comparer une lettre avec elle-même est inutile
4 avec 2 // comparer la suite est inutile car déjà fait dans l'autre sens
5 avec 1
6 avec 0

```
Variables
        i : entier
        isPalin : booleen
        indexMax : entier
Fonction isPalindrome(mot : chaine)
        i = 0
        isPalin = vrai
        indexMax = LENGTH(mot)
        TANT QUE i < indexMax ET isPalin == vrai
                SI mot[i] != mot[indexMax - 1 - i]
                        isPalin = faux
                FIN SI
                i = i + 1
        FIN TANT QUE

        retourner isPalin
Fin fonction

Variables
        mot : chaine
Algo
        mot = « bonjour »
        // la variable chaine est-elle un palindrome ?

        // V1 avec une boucle POUR : ça marche, mais potentiellement on fait des itérations inutiles.
        // si dès le premier tour on sait que ce n'est pas un palindrome, il est inutile de faire le reste
        des comparaisons
        isPalin = vrai
        POUR i de 0 à LENGTH(mot)-1
                SI mot[i] != mot[LENGTH(mot)-1-i]
                        isPalin = faux
                FIN SI
        FIN POUR

        SI isPalin
```

```
        Afficher mot CONCAT « est un palindrome »
    SINON
        Afficher mot CONCAT « n'est pas un palindrome »
    FIN SI

    // V2 avec un TANT QUE pour pouvoir s'arrêter dès que l'on sait que papalindrome
    Lire mot
    i = 0
    isPalin = vrai
    TANT QUE i < LENGTH(mot) ET isPalin == vrai
        SI mot[i] != mot[LENGTH(mot)-1-i]
            isPalin = faux
        FIN SI
        i = i + 1
    FIN TANT QUE

    // V3 avec une fonction (dans laquelle on externalise la V2)
    Lire mot
    isMotPalin = isPalindrome(mot)

    SI isMotPalin
        Afficher mot CONCAT « est un palindrome »
    Sinon
        Afficher mot CONCAT « n'est pas un palindrome »
    FIN SI

    Lire mot2
    isMot2Palin = isPalindrome(mot2)
Fin algo
```

16. Exercice 16

Ecrire l'algorithme qui :

- Ajoute des produits respectant cette structure, dans un tableau
  ```
  Produit : structure {
      Nom : chaine
      Prix : entier
  }
  ```
- Il faut 4 produits : vélo 120€, Cadre 5€, Chaise 10€, Pull 20€
- Affiche ces produits en vente en les préfixant par un indice
- Donner la main à l'utilisateur pour lui demander quel produit il veut acheter : il choisit un produit grâce à son indice. Tant qu'il n'a pas saisi -1, on lui redonne la main pour qu'il puisse choisir d'autres produits (gestion de panier)
- Quand il a fini il faut afficher le prix total
- Puis lui donner la main pour qu'il nous dise avec combien il va payer
  - S'il ne donne pas assez d'argent, on recommence l'étape précédent

S'il donne pile poil l'argent demandé ou plus, on lui dit merci et on lui rend si nécessaire la monnaie

```
Variables
    Produit : structure {
        Nom : chaine
        Prix : entier
    }
```

```
    produit1 : Produit
    produit2 : Produit
    produit3 : Produit
    produit4 : Produit
    catalogue : tableau de Produit
    choix : entier
    panier : tableau de Produit
    prod : Produit
    prixTotal : entier // camelCase
    thune : entier
    monnaie : entier
Algo
    /*
    produit1.nom = « Vélo »
    produit1.prix = 120
    produit2.nom = « Cadre »
    produit2.prix = 5
    produit3.nom = « Chaise »
    produit3.prix = 10
    produit4.nom = « Pull »
    produit4.prix = 20

    ajouter produit1 dans catalogue
    ajouter produit2 dans catalogue
    ajouter produit3 dans catalogue
    ajouter produit4 dans catalogue
    */

    // sans variable intermédiaire
    ajouter creerProduit(« Vélo », 120) dans catalogue
    ajouter creerProduit(« Cadre », 5) dans catalogue
    ajouter creerProduit(« Chaise », 10) dans catalogue

    // avec variable intermédiaire
    produit4 = creerProduit(« Pull », 20)
    ajouter produit4 dans catalogue

    // afficher le catalogue à l'écran
    POUR i de 0 à LENGTH(catalogue)-1
        Afficher i CONCAT catalogue[i].nom CONCAT catalogue[i].prix
    FIN POUR

    // ajouter les produits choisis dans le panier
    Lire choix
    TANT QUE choix != -1
        ajouter catalogue[choix] dans panier
        Lire choix
    FIN TANT QUE

    // afficher le prix total
    prixTotal = 0
    POUR CHAQUE prod DANS panier
```

```
            prixTotal = prixTotal + prod.prix
        FIN POUR CHAQUE

        Afficher « Le prix total est : » CONCAT prixTotal

        // avec combien d'argent il va payer ?
        Afficher « Vous souhaitez payer avec combien de thunes ? »
        Lire thune
        TANT QUE thune < prixTotal
            Afficher « Insuffisant, vous souhaitez payer avec combien de thunes ? »
            Lire thune
        FIN TANT QUE

        // dire au revoir et donner la monnaie si nécessaire
        SI thune > prixTotal
            monnaie = thune - prixTotal
            Afficher « Voici votre monnaie : » CONCAT monnaie
        FIN SI

        Afficher « merci orvoir »
Fin algo

Variable
        product : Produit
Fonction creerProduit(nom : chaine, prix : entier)
        product.nom = nom
        product.prix = prix
        retourner product
FIN fonction
```

17. Exercice 17

Créer une grille de 8 cases sur 8. Placer 3 bateaux dessus : un de 2 cases, un de 3 cases et un de 4 cases. Donner la main à l'utilisateur pour qu'il tire sur une case : dans l'eau, touché, coulé. On lui redonne la main tant que tous les bateaux ne sont pas coulés.

- placerBateau(nbCases, sens)
- verifPlacement(nbCases, sens)
- attaquer(x, y)
- verifJeu()

```
Variables
        age : entier // déclaration
        sexe : booleen // déclaration
Debut algo
        // initialisation : on donne une valeur à une variable
        // donner une valeur « en dur »
        age = 17

        // demander à l'utilisateur de saisir une valeur
```

```
        Lire age

        SI age < 18
                Afficher « Trop jeune »
        FIN SI

        SI age > 30
                Afficher « trop vieux »
        FIN SI

        // initialisation
        sexe = vrai

        SI sexe == vrai
                Afficher « Femme »
        SINON
                Afficher « Homme »
        FIN SI
Fin algo
```

Exercices supplémentaires :

- Demander à l'utilisateur son prénom (texte)

S'il s'appelle « toto », afficher « bienvenue ».
S'il ne s'appelle pas « toto », afficher « au revoir »

- Demander à l'utilisateur de saisir un fruit

Si il tape banane, afficher « ok voilà une banane »
Si il tape fraise, afficher « voici une fraise »
Si il ne tape aucun des deux, afficher « je n'ai pas en stock »

- Demander de saisir une note :

Afficher « pas bien » si la note est comprise entre 0 et 9
Afficher « moyen » si la note est comprise en 10 et 13
Afficher « très bien si la note est comprise entre 14 et 19
Afficher « excellent » si la note est égale à 20

- Afficher 10 fois « bonjour »

- Afficher les chiffres de 1 à 20

```
POUR i de 1 à 20 PAR PAS de 1
        Afficher i
FIN POUR
```

- Demander à l'utilisateur de saisir un chiffre entre 1 et 5

Tant qu'il ne saisit pas un chiffre entre 1 et 5 (par exemple 6 ou 0), il faut lui redonner la main pour qu'il saisisse à nouveau un chiffre entre 1 et 5
Afficher le carré de ce chiffre : il tape 5, il faut afficher le résultat de 5 * 5

```
Variable
        chiffre : entier
```

```
        resultat : entier
Début algo
        chiffre = Saisir « Veuillez saisir un chiffre 1 et 5 »
        TANT QUE chiffre < 1 OU chiffre > 5
                chiffre = Saisir « Veuillez saisir un chiffre 1 et 5 »
        FIN TANT QUE

        resultat = chiffre * chiffre
        afficher resultat
Fin algo
```