Logique algorithmique

1 Introduction

Dans certains cas, les algorithmes doivent prendre une décision.

Si le chiffre d'affaire du représentant est supérieur à 100, alors il bénéficie d'une prime de 10 euros sinon, il n'obtient aucune prime.

De même, certaines instructions peuvent être répétées plusieurs fois.

Tant que les notes ne sont pas toutes enregistrées, on enregistre la note suivante

Ou

On enregistre une note jusqu'à ce que toutes les notes soient enregistrées.

Ou

Pour note n=1 à note n=10, enregistrer une note (la n-ième)

Il y a des instructions spécifiques pour ces cas là.

2 Alternatives (si)

2.1 Introduction

Dans certains, l'algorithme est censé faire un choix entre deux (ou plus) possibilités.

Exemple :

Je veux afficher le plus grand de deux nombres saisis.

La réponse en français est :

Si le nombre 1 est plus grand que le nombre 2, alors afficher le nombre 1, sinon afficher le nombre 2.

Formalisons :

```
Si (nombre1>nombre2) alors
        Afficher (nombre1)
Sinon
        Afficher (nombre2)
FinSi
```

On écrira donc l'alternative sous la forme :

```
Si (condition) alors
        {1er groupe d'instructions, exécutées si condition est Vraie}
Sinon
        {2nd groupe d'instructions, exécutées si condition est Fausse}
FinSi
```

Rappel : une condition est une comparaison de deux données (variable ou constante) ou plusieurs comparaisons reliées par des connecteurs logiques (ET, OU).

2.2 Cas d'utilisation

On utilise des alternatives dans différents cas :

Effectuer un choix :

```
Si (chiffreAffaire>1000) Alors
        // Prime de 10% du chiffre d'affaire
        Prime <- chiffreAffaire * 0,1
Sinon
        // Prime de 5% du chiffre d'affaire
        Prime <- chiffreAffaire * 0,05
FinSi
```

Contrôler une valeur

```
Si (choixSaisi = 'O') alors
        Réponse <- 'OUI'
Sinon
        Réponse <- 'NON'
FinSi
```

2.3 Exemples sur les algo vus aux chapitres précédents

Calcul de la surface du cercle ; on ne peut calculer de surface que si le diamètre est positif (et non nul)

```
Algo surfaceCirculaire
Variables
        PI = 3.141592635
        surface, diamètre : réel
Début
        // Affichage et saisie des données
        Écrire ("Donnez le diamètre du cercle")
        Lire (diamètre)

        // Calcul de la surface si le diamètre est positif
        Si (diamètre >= 0) alors
                surface <- PI*diamètre*diamètre/4
                Écrire ("La surface du cercle est de :" , surface)
        Sinon
                Écrire ("Le diamètre est négatif ou nul, impossible de calculer une surface")
        FinSi
Fin
```

Échange de deux caractères ; l'échange n'est fait que si les deux caractères sont différents

```
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 si les valeurs sont différentes
        Si (a<>b) alors
                tampon <- a
                a<-b
                b<-tampon

                // 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)
        Sinon
                Afficher ("les valeurs sont identiques, pas besoin de les échanger")
        FinSi
Fin
```

Calcul du volume du tube ; on ne peut calculer un volume que si le diamètre extérieur est supérieur au diamètre intérieur.

```
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)

        Si (diametreExt< diametreInt) alors
                // 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)
        Sinon
                Afficher ("Le diamètre intérieur est supérieur ou égal au diamètre extérieur : impossible de
                calculer un volume")
        FinSi
Fin
```

2.4 Conclusion

Si (condition) alors, sinon, finSi est l'instruction majeure pour créer des traitements ou 'chemins' alternatifs lorsque l'algo doit prendre une décision.

L'instruction Sinon n'est pas obligatoire mais un Si se fini toujours par un FinSi comme un début se fini par un Fin (répétez 15 fois aussi vite que possible … si, si, allez-y. Aller, si).

3 Répétitives ou itératives

3.1 Introduction

Dans certains cas (assez courants), on est amené à exécuter plusieurs fois la même action.

Exemple :

Calculer et afficher le tableau d'amortissement d'un emprunt à remboursement constant.

On écrira un tableau comportant autant de lignes que d'année de remboursement.

Chaque ligne du tableau est calculée selon le principe suivant :

Première colonne : capital emprunté ; c'est le capital restant à rembourser de la ligne précédente.

Seconde colonne : intérêt d'emprunt ; c'est le capital emprunté multiplié par le taux de remboursement.

Troisième colonne : remboursement ; c'est le capital initial divisé par le nombre d'années de remboursement.

Quatrième colonne : annuité : c'est la somme des intérêts et du remboursement.

Premier jet de l'algo :

```
Algo empruntremboursementConstant
Var
        capitalEmprunté, capitalRestant, remboursement : réel
        txInteretPourcent, txInteret, interet : réel
        durée, année : entier
Début
        Afficher ("donnez le capital emprunté")
        Saisir (capital)
        Afficher ("donnez le taux d'intérêt (en pourcentage)")
        Saisir (txInteretPourcent)
        Afficher ("donnez la durée de l'emprunt")
```

```
        Saisir (durée)

        // Afficher les entêtes de colonnes
        Afficher ("Année, Capital, Intérêt, Capital remboursé, Annuité")

        // Calcul des conditions de départ
        rembousement <- capital / durée
        txInteret <- txInteret/100
        année <- 1

        // Calcul d'une ligne
        interet <- capitalEmprunté *txInteret
        annuité <- interet+remboursement
        // Afficher les résultats en colonnes
        Afficher (année, capitalRestant, interet, capital remboursé, annuité")
        capitalRestant <- capitalEmprunté-remboursement
Fin
```

Cet algo est correct mais on ne calcule qu'une seule ligne. Que faut-il faire pour calculer les suivantes ? Peut-on répéter la séquence suivante ? pour chaque année de la durée de remboursement ?

```
        // Calcul de la ligne 2
        année <- 2
        interet <- capitalRestant * txInteret
        annuité <- interet + remboursement
        // Afficher les résultats en colonnes
        Afficher (année, capitalRestant, interet, capital remboursé, annuité")
```

Mais le capital restant dû de l'année suivante est le capital restant à devoir à la fin de l'année que l'on vient de calculer. Donc calculons le.

```
        capitalRestant <- capitalRestant - remboursement
```

Oui mais il faut modifier l'algo à chaque fois que l'on change de durée (qu'elle soit saisie ou non).

En fait, en français on dit que pour chaque année, on calcule les intérêts du capital restant dû, le remboursement, l'annuité puis on affiche le tout. Donc en algo, ça donne :

```
        capitalRestant <- capitalEmprunté
        Pour année de 1 à durée
                interet <- capitalRestant*txInteret
                annuité <- interet+remboursement
                Afficher (année, capitalRestant, interet, capital remboursé, annuité")
                capitalRestant <- capitalRestant - remboursement
        finPour
```

Remarquez l'initialisation du capital restant qui, la première année est égal au capital emprunté.

On a donc ici une nouvelle instruction qui permet de répéter plusieurs fois un même traitement. Il en existe plusieurs.

3.2 Pour – FinPour

Format :

```
Pour (variableDeComptage de borneDeDépart à borneD'Arrivée)
        {Instructions à exécuter plusieurs fois}
FinPour
```

Cette instruction permet de construire une boucle lorsque l'on connaît le nombre de termes, le nombre de répétitions à faire.

Voir exemple ci dessus.

3.3 Répéter – Jusqu'à

Format :

```
Répéter
        {Instructions à exécuter plusieurs fois}
Jusqu'à (conditionDeFinDeBoucle)
```

Cette instruction permet de répéter un nombre indéterminé au départ de fois la boucle. Seule la condition de sortie permet de finir la boucle.

On répète les traitements jusqu'à ce que la condition soit vraie.

Exemple :

Répéter le calcul des annuités jusqu'à ce qu'on n'ait plus rien à rembourser

Attention, la formulation de la phrase est importante.

Répéter signifie qu'on effectue les traitements au moins une fois. De plus la condition de fin est bien placée après les traitements et elle vérifie s'il faut s'arrêter ou non.

3.4 TantQue – FinTantQue

Format :

```
TantQue (conditiond'executionDesTraitements) Faire
        {Instructions à exécuter plusieurs fois}
FinTantQue
```

Cette instruction permet d'exécuter un nombre indéterminé au départ de fois la boucle. Seule la condition d'entrée permet de commencer la boucle et de voir si on doit la continuer.

On exécute les traitements tant que la condition est vraie.

Exemple :

Tant que l'on a quelque chose à rembourser, on effectue le calcul de l'annuité

Attention, la formulation de la phrase est importante.

TantQue signifie qu'on effectue les traitements que si la condition est vérifiée, c'est à dire qu'avant le traitement, si la condition n'est pas vérifiée, le traitement n'est pas fait et on passe à la suite.

De plus la condition de fin est bien placée avant les traitements et elle vérifie s'il faut les faire ou non.

3.5 Exemples

3.5.1 Procédure d'accueil d'un client

Lorsqu'un (*lorsque=Si*) client fait appel à l'organisation pour avoir des renseignements.

L'organisation renvoie un catalogue et des informations publicitaires

Éventuellement (*éventuellement=Si*), le client nous adresse une commande traitée par l'organisation.

Plus tard (=autre envoie d'un client), le client effectue le paient qui est enregistré et encaissé.

```
Si (client demande renseignements) Alors
        Envoie d'un catalogue au client
Sinon
        Si (client envoie une commande) Alors
                Traiter la commande du client
        Sinon
                Si (client envoie un paiement) Alors
                        Enregistrer paiement du client
                        Encaisser paiement du client
                FinSi
```

```
        FinSi
    FinSi
```

3.5.2 Second exemple : le contrôle des factures impayées

Seconde procédure

(*Seconde procédure*) Régulièrement, on analyse toutes les factures non payées. Si la date de la facture est vielle de plus d'un mois, c'est que le client tarde à payer. L'organisation envoie alors au client une lettre de relance. On note qu'une lettre a été envoyée pour que le client n'en reçoive q'une seule.

```
Pour numéro_de_facture de 1 à 999999
        Si (la date de la facture est ancienne et aucune lettre envoyée) Alors
                Envoyer lettre de relance
                Noter dans la facture, qu'une lettre a été envoyée
        FinSi
FinPour
```

Ou mieux :

```
Répéter
        Lire une facture
        Si (la date de la facture est ancienne et aucune lettre envoyée) Alors
                Envoyer lettre de relance
                Noter dans la facture, qu'une lettre a été envoyée
        FinSi
Jusqu'à (fin de la liste des factures)
```

4 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
Les tests	Prendre des décisions	Les alternatives ou conditionnelles	Si, sinon, finsi
Les boucles	Faire plusieurs fois le même travail	Les répétitives ou itérations	Répéter, jusqu'à ; tant que, faire, fin tant que ; pour, fin pour

Instructions et leur structures

```
Si (conditions de réalisation) Alors
        <Actions à réaliser si les conditions sont vraies (remplies)>
Sinon
        <Actions à réaliser si les conditions ne sont fausses (pas réalisées)>
FinSi

Pour <nom d'un compteur> de <borne initiale> à <borne finale>
        <Actions à réaliser tant que le compteur est entre la borne initiale et la borne finale>
FinPour

TantQue (condition de répétition)
        <Actions à réaliser tant que les conditions de travail sont vraies>
FinTantQue

Répéter
        <Actions à réaliser jusqu'à ce que les conditions de fin soient vraies>
Jusqu'à (conditions de fin)
```

Important : notez le retrait d'écriture, il indique que chaque instruction, vue ci-dessus, correspond à un début et une fin de bloc d"actions à réaliser".

5 Applications

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

5.1 Calcul du maximum

```
Algorithme maximum()
/* Description : Cet algorithme effectue la comparaison entre deux entiers.
        Il affiche l'entier maximum des deux */
Var     // déclaration des variables
        nombre1, nombre2 : entier, nombres à comparer
Début   // corps
        Saisir ("Entrez un entier", nombre1)            // lecture du premier entier
        Saisir ("Entrez un second entier", nombre2)     // lecture du second entier
        Si (nombre1>nombre2) alors
                Afficher ("Le maximum est : ", nombre1)
        Sinon
                Afficher ("Le maximum est : ", nombre2)
        Fin si
Fin
```

5.2 Calcul de la moyenne des notes d'un élève

L'algo doit faire la saisie des nom et prénom de l'élève, celle de ses notes s'il y en a puis afficher leur nombre, le total et la moyenne de celles-ci, à la fin.

```
Algorithme moyenneElève()
Variables
        nom, prénom : chaîne
                // nom et prénom : informations élève à rappeler en fin de calcul de la moyenne
        continuer : caractère
                // continuer : indicateur qui permet de savoir ce que veut faire l'utilisateur
        note, nombre, total : réel
                // note : la dernière note saisie
                // nombre : nombre de notes saisies, nécessaire pour effectuer le calcul de la moyenne
                // total : somme de l'ensemble des notes
Début
        // Initialisations
        nombre <-0
        // Saisie des informations élève
        Afficher ("donnez les nom et prénom de l'élève")
        Saisir (nom, prénom)
        // Saisie initiale de continuer
        Répéter
                Afficher ("y a-t-il des notes à saisir (o/n) ? ")
                Saisir (continuer)
        Jusqu'à (continuer="O" ou continuer="o" ou continuer="N" ou continuer=n")

        // Boucle de saisie des notes
        TantQue (continuer="O" ou continuer="o") faire
                // Saisie de la note
                Afficher ("Donnez la note ", nombre+1," : ")
                Saisir (note)
                // Calculs
                nombre <- nombre+1  // incrémenter le nbre de notes saisies
                total<-total+note  // accumuler la note dans le total
                // Saisie de la continuation
                Répéter
                        Afficher ("y a-t-il d'autres notes à saisir (o/n) ? ")
                        Saisir (continuer)
                Jusqu'à (continuer="O" ou continuer="o" ou continuer="N" ou continuer=n")
        FinTantQue

        // Affichage des informations élève
        Afficher ("Élève : ", nom, ", ", prénom, " : ")
        // Affichage des résultats
        Si (nombre>0) Alors
                Afficher ("nombres de notes : ", nombre)
                Afficher ("total des notes : ", total)
                Afficher ("moyenne des notes", total/nombre)
        Sinon
```

```
                Afficher ("aucune note saisie")
        FinSi

        // Affichage de fin du programme
        Afficher ("Au revoir")
Fin
```

Transformer cet algorithme pour effectuer la saisie tant qu'il y a des élèves à saisir.

Noter la boucle de contrôle de saisie autour de la saisie de continuer pour être sûr d'avoir saisi un O, o, N ou n.

5.3 Programme de calcul de la racine carrée d'un nombre

```
Algorithme racine()
        /* Description : Cet algorithme calcule et affiche la racine carrée d'un nombre.
                Il s'arrête lorsqu'on donne un nombre inférieur ou égal à zéro
        */
Variable  // déclaration des variables
        nombre, racine : réel
Début   // corps
        Saisir ("entrez un nombre positif (0 ou négatif pour sortir du programme)", nombre)     // lecture du nombre
        TantQue (nombre>0)
                racine <- RacineCarrée(nombre)
                Afficher("la racine carrée de ",nombre, " est ", racine)
        FinTantQue
        Ecrire("Au revoir")
Fin
// fin de l'algorithme
```

5.4 Calcul de la racine carrée à l'aide de la dichotomie

Par principe, la racine carrée d'un nombre est inférieure à ce nombre si il est supérieur à 1 et supérieure si il est inférieur à 1.

On ne tiendra pas compte des erreurs de calcul comme racine de -1

La dichotomie est une méthode de recherche de solution d'une équation.

Pour rechercher la racine d'un nombre, nous utiliserons l'équation suivante :

Nombre – (racine(nombre))² = 0

Mais on ne connaît pas racine(nombre) donc on le calcul en recherchant une valeur approchée.

Cette valeur approchée est recherchée comme suit.

On prend le milieu d'entre le nombre et 1 et on calcule milieu*milieu.

Si milieu*milieu est supérieur à nombre, alors la racine se trouve entre notre milieu et la borne inférieure sinon, la racine se trouve entre milieu et le nombre.

Si on n'a pas retrouvé la racine du premier coup, on recommence en calculant le milieu d'entre milieu et la borne concernée.

Dans notre cas, on utilisera l'équation

nombre – milieu² <= précision

sinon on n'a pas fini de calculer…. La précision est donnée en nombre de décimales.

```
Fonction calculRacineCarrée(nombre : réel, précisionNbDécimales : entier): réel
Variables
        // borneInf, borneSup bornes inférieure et supérieure ; précision : nombre 1 précédé de précisionNbDécimales décimales
        borneInf, borneSup, milieu, précision : réel
Début
        // Initialisations
        borneInf <-1
        borneSup <-1

        // Calcul de la précision
        précision <- 1
        Pour i de 1 à précisionNbDécimales
                précision <- précision/10
        FinPour

        // Ordonner les bornes :
        Si ( nombre > 1 ) Alors
```

```
                borneSup <- nombre
        Sinon
                borneInf <- nombre
        FinSi

        // Recherche du 1er milieu
        milieu <- (borneInf + borneSup)/2

        TantQue (nombre - milieu * milieu > précision) faire
                // Ordonner les bornes
                Si (milieu*milieu > nombre) Alors
                        borneSup <- milieu
                Sinon
                        borneInf <- milieu
                Finsi

                // Recherche du (nouveau) milieu
                milieu <- (borneInf + borneSup)/2
        FinTantQue

        // Retourner le résultat (la racine)
        Retourner milieu
fin
```

Analysez le texte et observez l'algo. Si vous comprenez mieux l'algo que le texte, vous avez tout gagné

5.5 Amortissements

```
Algo CalculAmortissement()
Variables
        mode, continuer : caractère
        tauxPourcent, taux : réel
        capitalInit, capitalRestant, intérêt, remboursement, annuité : réel
        totalIntérêt, totalAnnuité : réel
        n, durée : entier
Début
        // Initialisation globale des variables
        continuer <- 'o'
        Répéter
                // Initialisation locale des variables
                tauxPourcent <- 0
                // Saisie des données
                Afficher ("Choisissez un type d'amortissement")
                Répéter
                        Afficher ("Annuité constante=A ; Remboursement constant=R")
                        Saisir (mode)
                Jusqu'à (mode = "A" ou mode = "R")
                Afficher ("Donnez le capital initial")
                Saisir (capitalInit)
                Afficher ("Donnez la durée de l'emprunt")
                Saisir (durée)
                Afficher ("Donnez le taux (en %) pour un amortissement par annuités constantes")
                Saisir (tauxPourcent)

                // Calculs préliminaires : taux en valeur absolue, annuité (cste)
                taux<-tauxPourcent/100
                Si (mode = "A") alors
                        annuité<- capitalInit * taux/100 * (1-(1+ taux/100)^(-durée))
                        remboursement <-0
                Sinon
                        annuité<- 0
                        remboursement <- capitalInitial/durée
                FinSi

                // Préparation de l'affichage des résultats : affichage de l'entête des colonnes
                Si (mode = "A") alors
                        Afficher ("Amortissement par annuités constantes")
                Sinon
                        Afficher("Amortissement par remboursements constants")
                FinSi
```

```
                    Afficher ("Année \tCapital \tIntérêt \tRemboursement \tAnnuité")

                    // Calculs et affichage des résultat au fur et à mesure ; n est l'année en cours
                    capitalRestant <- capitalInitial
                    Pour (n=1 à durée)
                            intérêt <- capitalRestant * taux
                            Si (mode = "A") Alors
                                    // anuité = constante
                                    remboursement <- annuité – intérêt
                            Sinon
                                    annuité <- intérêt +remboursement
                                    // remboursement = constante
                            FinSi
                            Afficher (n, "\t", capitalRestant, "\t", intérêt, "\t", remboursement, "\t", annuité)

                            // Calcul des totaux de récapitulation en fin d'algo
                            totalAnnuité <- totalAnnuité+annuité
                            totalIntérêt <- totalIntérêt <- intérêt

                            // Préparation du calcul de la ligne suivante
                            capitalRestant <- capitalRestant - remboursement
                    FinPour
                    // Affichage de fin de calcul
                    Afficher ("Le total des intérêts se monte à : ", totalIntérêt)
                    Afficher ("Le total des annuités s'élève à : ", totalAnnuité

                    // Recommencer ?
                    Afficher ("Voulez-vous effectuer une nouvelle simulation ? (o/n) ")
                    Saisir (continuer)
            Jusqu'à (continuer <> 'o' et continuer <>'O')

            // Traitements de fin du programme
            Afficher ("À bientôt")
Fin
```