

INTRODUCTION

L'attribution principale d'un gestionnaire informaticien est la mise en place d'application permettant d'automatiser une ou plusieurs activités d'une entreprise donnée. Pour y parvenir l'informaticien devra procéder d'abord à une analyse de l'activité, voire du domaine à informatiser (cette analyse fait l'objet d'un autre cours) et ensuite à la programmation proprement dite. Cette programmation exige de la part de l'informaticien la maîtrise de deux outils qui sont :

- **Une logique de programmation** : celle-ci lui permettra de penser et de matérialiser sur feuille toutes les étapes qu'il doit parcourir ou toutes les tâches qu'il doit exécuter afin d'obtenir le résultat escompté. Cette logique sera acquise grâce à l'algorithme.

- **Un langage de programmation** : Celui-ci lui permettra de converser avec la machine, de **traduire** l'algorithme en un langage compréhensible par la machine. Il ressort de là que le véritable travail à effectuer par l'informaticien n'est pas la traduction mais la matérialisation des différentes tâches à exécuter, c'est cette matérialisation des différentes tâches dans leur ordre qui constitue la partie algorithmique d'un programme. **Mais c'est quoi l'algorithme ?**

Dans la vie de tous les jours nous faisons tous des algorithmes. Si nous énumérons toutes les tâches à exécuter depuis le réveil jusqu'à se trouver dans une salle de cours, nous venons d'écrire un algorithme.

Si pendant une vente nous sommes capables d'exécuter un ensemble de tâches et communiquer au client le prix à payer, nous faisons de l'algorithme.

La différence ici avec l'algorithme du point de vue informatique est le langage utilisé pour écrire cet algorithme et qui n'est pas un langage compréhensible par la machine ni un langage de tous les jours.

CHAPITRE 1 : ENVIRONNEMENT D'UN ALGORITHME

Objectif : *Etre capable de définir un objet, d'énumérer et de déclarer tous les objets intervenant dans un problème posé*

Connaissances Préalables : *Aucune*

I. DEFINITIONS

1. Algorithme :

C'est un ensemble d'instructions ordonnées qui une fois exécuté produit un résultat donné, escompté, attendu. Il ressort de cette définition qu'avant d'écrire un algorithme quel qu'il soit le programmeur doit poser l'une des principales questions ci-dessous :

- **Quel résultat veut-on obtenir ?**
- **Que doit faire l'algorithme à écrire.**

Après avoir répondu à cette question il doit ensuite se poser la question :

Comment y arriver?

Exemple : Nous voulons écrire un algorithme qui doit nous afficher le reste à payer par un étudiant pour sa scolarité.

Quel résultat doit être affiché ? Le reste à payer

Comment y parvenir ? Connaître le montant à payer pour sa scolarité et le total déjà payé.

On peut dire que connaissant ces deux informations on va faire l'opération suivante :

Montant de la scolarité – total déjà payé (et non le contraire)

Remarquez bien qu'on ne pouvait pas faire la différence si l'on ne connaît pas encore les deux informations (les deux Montants) et c'est là que le mot **ORDONNÉ** prend toute son importance dans la définition d'un algorithme.

2. Instruction :

Chacune des tâches exécutées dans un algorithme pour aboutir au résultat (y compris le résultat lui-même) est appelée instruction. En réalité une instruction est une ligne de l'algorithme, un ordre qu'on donne à la machine et cet ordre exécuté peut permettre à :

- L'utilisateur de lire un résultat affiché par la machine.
- L'utilisateur de faire entrer des informations dans la machine.
- La machine de faire une opération et de conserver le résultat dans une zone.
- La machine de faire des comparaisons etc. etc....

Dans l'exemple précédent les différentes instructions pouvant permettre d'atteindre le résultat demandé peuvent se présenter de la façon suivante :

- **Connaître (ou communiquer à la machine)** le montant de la scolarité et le total déjà payé.
- **Demander à la machine de faire** la différence et d'**afficher** la valeur trouvée.

On peut constater que chacune de ces lignes contient l'ordre, l'instruction proprement dite, la tâche qui sera exécutée et qui est :

- **Connaître ou communiquer.**
- **Faire** (calculer) et **afficher**.

A la suite de l'ordre on retrouve les éléments concernés par cet ordre à savoir :

- **Montant de la scolarité et Total déjà payé** pour l'ordre **Connaître**
- **Résultat de l'opération de soustraction**

Chacun de ces éléments est appelé **Objet de l'algorithme**

3. Les éléments d'un algorithme

Comme nous venons de le voir, l'univers d'un algorithme est constitué par des objets sur lesquels portent les tâches à exécuter.

- a) Comment représenter un objet.


En algorithmes un objet est représenté par un ensemble de caractères en un seul mot, (significatif ou non), le premier caractère étant une lettre.

Certaines représentations de ce fait ne sauraient être admises.

Le tableau ci-dessous vous donne quelques représentations en s'appuyant sur notre exercice

Objet	Acceptée	Non Acceptée
Montant de la scolarité	MONTSCOL MONT-SCOL MONTANTSCOLARITE	MONT SCOL MONTANT SCOL MONTANT SCOLARITE
Total déjà payé	TOTAL TOTALDPAYE TOTALDEJAPAYE	TOTAL PAYE TOTAL D PAYE TOTAL DEJA PAYE
Résultat	RESULTAT	RESUL TAT

b) Comment définir un objet

Un objet en algorithmes est une zone réservée, une boîte que le programmeur se propose ou propose à l'ordinateur pour conserver (de façon temporaire) une seule valeur à un instant T. Nous pouvons schématiser cette zone ou boîte par un rectangle  et c'est elle qui sera reconnue par exemple comme MONTSCOL. Lorsque la valeur contenue dans cette boîte ne change pas d'une occurrence à l'autre (ici d'un étudiant à un autre) on dira que l'objet est une constante ; dans le cas contraire on dira que l'objet est une variable. De ce qui précède on peut alors définir une variable comme **un espace logique prévu pour contenir plusieurs valeurs mais ne conserve qu'une et une seule valeur à un instant T.**

Exercice 1

Pour calculer la moyenne de chacun des élèves de CP2 l'instituteur utilise les objets suivants

NOM DE L'ELEVE ; PRENOM ; CLASSE ; MATIERE ; MOYENNE DANS LA MATIERE ; MOYENNE SEMESTRIELLE.

Donnez un nom à chacun des objets tout en précisant si l'objet est une constante ou une variable.

Exercice 2

Les objets relatifs à une facture sont : Numéro de facture ; date de facture ; nom de l'article ; quantité ; PU ; Montant HT ; TAUX DE TVA ; MONTANT TTC.

Travail à Faire : Même question que précédemment.

A ce stade de notre cours il convient de donner les différentes parties d'un algorithme.

➤ **En-tête** : constitué d'au moins une ligne sur laquelle on donne le nom de l'algorithme (en un mot) Exemple : Algorithme CALCULREST.

➤ **Partie Déclarative** qui consiste à déclarer tous les objets que l'on utilisera dans la 3^e partie.

➤ **Partie traitement ou corps** : c'est la partie la plus importante car contient toutes les instructions permettant d'aboutir au résultat escompté.

**AUCUN OBJET NE DOIT ETRE UTILISE DANS CETTE PARTIE
SANS ETRE DECLARE AU PREALABLE.**

C'est dans cette dernière partie que nous allons mettre en pratique les 4 catégories d'ordre ou d'instruction que nous aurons à étudier dans ce cours : ces 4 catégories d'ordre sont également les principales notions que tout étudiant en Informatique Développeur d'Application doit maîtriser avant de prétendre se lancer dans le développement des applications, ce sont :

- L'AFFECTATION
- LA LECTURE ET L'ECRITURE
- LES TEST ou CONDITION
- LES BOUCLES

Avant d'y arriver, permettez que nous terminions ce chapitre en apprenant à dire à l'ordinateur, à déclarer à l'ordinateur nos objets étant donné que ce sont ces objets qui seront utilisés dans chacune de ces catégories d'ordre

II. DECLARATION DES OBJETS

1. Pourquoi déclarer les objets ?

Comme nous l'avons vu plus haut, la déclaration constitue la 2^{ème} partie d'un algorithme. Elle consiste à dire à l'ordinateur

- Les différents objets que l'on désire utiliser dans le programme
- Les différents caractères qu'on souhaite conserver ou déposer dans ces différentes boîtes. Par exemple si nous disons à l'ordinateur que nous avons un objet dans lequel on souhaite faire entrer des nombres tels que 125 ; 200... l'ordinateur n'acceptera pas qu'on y entre des lettres telles que A ; B ; C... Vous voyez l'importance de la déclaration ?
- Les différentes opérations dans lesquelles ces objets pourront intervenir. Et oui $125 + 200$ vous savez ce que cela donne même si vous n'êtes pas fort en Maths ; mais $AZ + BR$ donne quoi ? Même un docteur en maths posera beaucoup de questions sur A Z B R avant de donner une réponse. Vous voyez l'importance de la déclaration ?

Non vous ne voyez pas encore, par contre vous réalisez qu'une déclaration de variable n'a rien de romantique par rapport à une déclaration d'amour même si tous les chefs d'entreprises vont la préférer à une déclaration d'impôts.

2. Comment déclarer les objets

On entend par syntaxe la manière dont on emploie un ordre ou une commande

- a) Cas des variables
 - Syntaxe générale

VAR Nom de la variable : type

- Interprétation

Nom de variable : C'est le nom donné à la variable

Type : Indique à l'ordinateur les caractères (Chiffres, lettres, signe de ponctuation) que la variable peut accepter et les différentes opérations possibles sur cette variable.

- Exemple de déclaration

VAR MONT : Numérique

VAR NOM : Alphanumérique

VAR NOMBRE : Entier

VAR Existe : Booléen

Le premier exemple (Numérique) et le deuxième (Alphanumérique) regroupent encore plusieurs types ; ce qui permet de dire qu'il y a une infinité de types de données. L'objet de ce cours n'est pas de vous dresser une liste exhaustive de tous les types existants mais de vous donner ceux dont vous aurez besoin dans l'écriture de vos algorithmes tout au moins pendant le cycle BTS. C'est pour cela que nous vous présentons ci-dessous un tableau donnant les différentes valeurs que peut accepter une variable en fonction de son type.

TYPES	VALEURS ou CARACTERES POSSIBLES
Byte (octet)	0 à 255
Entier simple	-32768 à +32767
Entier long	-2 147 483 648 à 2 147 483 648
Réel simple	-3,40*10 ³⁸ à -1,40*10 ⁴⁵ pour les valeurs négatives 1,40*10 ⁴⁵ à 3,40*10 ³⁸ pour les valeurs positives
Réel double	-1,79*10 ³⁰⁸ à -4,94*10 ⁻³²⁴ pour les valeurs négatives 4,94*10 ⁻³²⁴ à 1,79*10 ³⁰⁸ pour les valeurs positives
Caractère	1 caractère (Alphabétique ou pas)
Chaîne de caractères	Ensemble de caractères (Alphabétique ou pas)
Booléen	VRAI ou FAUX / FERMER ou OUVERT

Que faut-il retenir ? Qu'est-ce qui est fréquemment utilisé à notre niveau ?

➤ Si la variable doit contenir uniquement des valeurs entières, le type approprié est **entier**. C'est le cas des variables nombre, page, numéro, ordre

➤ Si la variable peut contenir ne serait-ce qu'une seule valeur décimale en plus des valeurs entières le type approprié est **réel** que nous désignons encore par Numérique (Num).

➤ Si la variable doit pouvoir conserver au moins 1 caractère qui n'est pas un chiffre, le type approprié est **Caractère** ou **Chaîne de caractère** (CC) que nous désignons encore par Alphanumérique (AN).

➤ Si la variable ne peut contenir que deux valeurs exclusives l'une de l'autre le type approprié est **Booléen**.

Ce sont là les types les plus utilisés en algorithmes ; par contre chaque langage a sa spécificité et certains langages renferment plus de types que d'autres. Parmi ces types on peut citer :

- MONETAIRE : (pour tout ce qui est montant)
- DATE : (pour tout ce qui est date)

Exercice 3

Dans un algorithme on utilise les variables contenues dans le tableau ci-dessous, complétez ce tableau en vous référant aux valeurs qu'on souhaite stocker dans ces variables.

Variables	Exemple de Valeurs possibles	Type
VARIAB1	12 ; 14 ; 25 ; -12 ; -15 ; 1,10	
VARIAB2	A12 ; 14 ; -12 ; 25	
VARIAB3	VRAI ; FAUX	
VARIAB4	ALI ; YAO KONAN ; K.JULES	
VARIAB5	12,5 ; 15,5 ; 18	
VARIAB6	12 ; 10 ; 14 ; 18 ; 20 ; -25	

b) Cas des constantes

Dans le cas des constantes la déclaration est plus aisée. En effet si une boîte est constante cela signifie que le contenu de la zone ne change pas, cela signifie aussi qu'on connaît ce contenu et par conséquent c'est cette valeur qu'il faut attribuer à la zone lors de la déclaration

- * Syntaxe : **CONST nom de la constante = Valeur**
- * Exemple : **CONST TAUX TVA = 0.18**

CONST NOMSOC = "Ministère"

Pourquoi Ministère est entre quote (" ") et pas 0.18

L'ordinateur reconnaît les chiffres (de 0 à 9) et le nom d'un objet commence par une lettre. Il faut alors que l'ordinateur puisse reconnaître NOMSOC comme le nom d'un objet et que Ministère est la valeur de cet objet et non un autre objet lui-même. Pour résoudre cette difficulté et dire à l'ordinateur que NOMSOC est un objet constant dont le contenu n'est pas numérique mais chaîne de caractère, on met cette chaîne de caractère entre griffe (" ") d'où **NOMSOC = "Ministère"**.

Exercice 4

Observez le reçu qui vous permet de suivre ce cours et donnez à travers le tableau ci-dessous

- Les objets présents
- VARIABLE ou CONSTANTE
- Le type de chaque objet

INFORMATION	OBJET PROPOSE	VARIABLE OU CONSTANTE	TYPE

Exercice 5

Conformément au tableau que vous avez rempli, faite la déclaration de tous les objets.

Exercice 6

Les déclarations suivantes sont-elles correctes (Oui ou Non) ? Justifiez votre réponse en cas de **Non**

a/ CONST Nbr1=-258

Test = True

Nbr2 =32,741

b/ CONST X = 593,47

U ='a'

X = -25, 01

c/ VAR x : -200

y : Entier

CONST z : Numérique

**SI VOUS AVEZ RENCONTRE DE DIFFICULTES DANS
CERTAINS EXERCICES REPRENEZ-LES AVANT DE
CONTINUER AVEC LE CHAPITRE 2**

CHAPITRE 2 : LES AFFECTATIONS

***Objectif :** Etre capable de donner le contenu d'une variable ou la réaction de l'ordinateur après une affectation quelle qu'elle soit.*

***Connaissances Préalables :** Savoir définir et déclarer les Objets d'un algorithme*

Définition et Fonctionnement

1. Affectation

L'affectation consiste à attribuer à une variable une valeur : on dit également qu'on attribue une valeur à un identificateur parce que justement le nom d'une variable est aussi appelé Identificateur.

2. Syntaxe de l'affectation

Nom de variable ← Valeur

3. Signification

Cette instruction demande à l'ordinateur de remplacer le contenu de la boîte représentée par **Nom de variable**, par "valeur". On peut donc déjà dire que l'affectation est la première possibilité pour donner une valeur ou un contenu à une variable après sa déclaration ; la 2^e possibilité, nous la verrons, sera l'instruction d'entrée ou la saisie. Cela n'est pourtant pas si simple, mais peut le devenir si vous aviez compris l'importance de déclarer une variable, si vous aviez compris que celle-ci est une boîte ou sac logique ; et dès lors vous accepteriez avec nous qu'il n'est pas possible de mettre n'importe quoi dans ce sac. Imaginez votre portefeuille qui peut contenir vos jetons (parce que les billets vous n'en avez pas encore !!!) et au lieu des jetons ce sont des cailloux que vous voulez y mettre ; bien sûr en forçant ça ira mais quel sens ? Quelle utilité ? Par

contre même si vous forcez, vous ne pourrez pas y mettre votre ordinateur portable qui a lui, un autre sac plus grand que le portefeuille. Vous voyez là où je veux en venir ? C'est aux règles à respecter pour que l'ordinateur puisse exécuter l'affectation

4. Les règles de l'affectation.

La règle principale est que le **Nom de variable** et la **valeur** doivent être de même type mais il faut avoir en esprit l'adage qui dit "Qui peut le plus peut le moins" cela signifie tout simplement que ce qui peut entrer dans le portefeuille peut indubitablement entrer dans le sac prévu pour l'ordinateur. Dans notre contexte on dira que si une variable est déclarée Alphanumérique, alors elle peut contenir toute sorte de caractères à savoir :

- Des chiffres uniquement
- Des lettres uniquement
- La fusion des deux

Et quand la variable est de type réel, que peut-elle contenir ?

Votre réponse ici

5. Conséquence d'une affectation

Lorsque les règles de l'affectation sont respectées c'est à dire que l'ordinateur ne trouve pas une erreur dans l'affectation, **le contenu de la variable est écrasé par la nouvelle valeur qu'on lui affecte**. Cette valeur affectée peut être


- Une constante numérique (0 ; 10 ; 15,5 etc.)
- Une expression entre griffe ("Monsieur")
- Une autre variable (Montant ou Qté1+Qté2) et dans ce cas c'est le contenu de cette variable qui représente la valeur


EXERCICES**Exercice 7**

A partir du tableau ci-dessous, précisez le contenu de la variable ou des variables au départ (Ecrire Erreur en cas d'affectation impossible).

VARIABLE DE DEPART	TYPE	CONTENU	AFFECTATION	NOUVEAU CONTENU OU REACTION ORDI.
Coucou	Entier	12	COUCOU ← 15,5	
Coucou	Réel	12	COUCOU ← 15,5	
Coucou	Réel	15,5	COUCOU ← 12	
Bébé	Réel	10	Bébé ← "Carine"	
Bébé	AN		Bébé ← "Carine"	
Bébé	AN	Carine	Bébé ← "Ali G. "	
NOM	AN	Durant	NOM ← PRENOM	
PRENOM	AN	Jules		
NOM	AN	Durant	NOM ← PRENOM	
PRENOM	AN	Jules		
QTE	Réel	0		
PU	Réel	5000	MONT ← QTE*PU	
MONT	Réel	5000		
NBR	Entier	5	NBR ← NBR+1	

Ces différentes affectations sans grande signification ont principalement pour objectif de montrer au lecteur 2 choses :

 Il n'est pas nécessaire pour l'ordinateur que le nom de variable ait un sens ; l'essentiel est que ce nom soit constitué d'un ensemble de caractères et écrit en un seul mot

 Quel que soit le contenu d'une variable celui-ci disparaît, est écrasé si l'affectation ne comporte pas d'erreur.

Exercice 8

Donnez le contenu de toutes les variables présentes après la dernière instruction

a/ RIEN ← 12	b/ A ← 0
RIEN ← 14	B ← 0
RIEN ← RIEN + 1	B ← 100
	B ← B+A

Ces 2 groupes d'instruction, une fois encore en dehors de l'objectif pédagogique sont parfaitement idiots car à quoi sert d'affecter à une variable une valeur qu'on ne veut pas utiliser ou qu'on écrase à l'instruction suivante ?

Dans le tableau précédent nous avons utilisé les signes * et + : ils sont appelés des opérateurs. Nous donnons ici ceux qui sont les plus utilisés en programmation et qu'on peut regrouper en opérateurs Arithmétiques ou Numériques, opérateurs de comparaison puis des opérateurs Alphanumériques comprenant entre autres des fonctions prédéfinies dont nous parlerons encore dans le chapitre consacré aux sous-programmes.

Opérateurs Arithmétiques	Opérateurs de Comparaison	Opérateurs Logiques	Opérateurs alphanumériques
Addition (+)	Plus petit que (<)	FAUX/VRAI	(Concaténation (& ou //))
Soustraction (-)	Plus grand que (>)	ET/OU/NON	Extraction (SSCHAINE)
Multiplication (*)	Inférieure ou égal (<=)		Nbre de caract.(Longueur)
Division (/)	Supérieur ou égal (>=)		Position (Rang)
Puissance (^)	Différent (<>)		Valeur Numérique (Code)
Division Entière (DIV)			Caractère d'1 Nbre (CAR)
Reste d'1 divisi. (MOD)			Transformation (CVCHAINE/CVNOMBRE)

Les opérateurs alphanumériques permettent de concaténer deux chaînes de caractères mais dans cette colonne, comme le montre le tableau un ensemble de fonctions sont employées avec les chaînes de caractères dont nous présentons ici les plus utilisées

Concaténation NOM ← "JULES"
 PRENOM ← "YAO"
Ensemble ← NOM & PRENOM
Dans Ensemble on aura **JULESYAO**

Longueur

C'est le nombre de caractères d'une chaîne de caractères ou du contenu d'une variable. On écrit **Longueur (Chaîne)**

Exemple : Longueur ("NOM") = 3

Longueur (NOM) = 5 car la variable NOM contient JULES qui compte 5 caractères.

Extraction

Elle permet d'extraire à partir d'une **position**, un **nombre de caractères** à l'intérieur d'une chaîne de caractères ou du contenu d'une variable

Exemple : SSCHAINE ("Couper Décaler",2,9) =ouper Déc (un espace est égal à 1 caractère)

Vérifiez que SSCHAINE (NOM,3,2) correspond à LE

Rang

Cette fonction recherche dans une chaîne de caractères ou dans le contenu d'une variable une sous chaîne à partir d'une position. Si la sous chaîne est trouvée c'est la position ou le rang du premier caractère de cette sous chaîne qui est affichée, dans le cas contraire c'est la valeur 0 (zéro) qui est affichée.

Exemple : Rang("Algorithme", "rithme",2)=5

Rang("Algorithme", "rithme",7)=0

Avec les opérateurs numériques il serait intéressant de faire une comparaison entre Informatique et Mathématique bien que ces opérateurs seront utilisés avec les mêmes règles dans les 2 cas.

- La multiplication et la division sont prioritaires par rapport à l'addition et la soustraction.
- L'utilisation des parenthèses sert à modifier cette priorité.

Exemple : $10*2+4 = 24$

$$(10*2)+4 = 24$$

$$\text{Mais } 10*(2+4) = 60$$

Les opérateurs de comparaison seront approfondis dans le chapitre suivant, mais en attendant sachez qu'on peut comparer également 2 valeurs alphabétiques c'est-à-dire voir si « A » est inférieur à «B» et ce sont les codes numériques des 2 caractères qui seront comparés

En informatique et plus précisément en algorithme le signe de l'affectation nous l'avons vu est \leftarrow . C'est par abus qu'on le remplace par le signe (=) d'autant plus que c'est ce signe qui est utilisé par la plupart des langages, mais cette égalité n'a pas le même sens qu'en Mathématique où $A=B$ signifie que $B=A$; alors qu'en informatique nous l'avons vu avec le tableau que le contenu de l'élément à gauche sera celui de l'élément à droite. Ainsi dans l'affectation $A \leftarrow B$, C'est le contenu de A qui sera modifié prenant ainsi la valeur de B et dans l'affectation $B \leftarrow A$, c'est le contraire.

CHAPITRE 3 : INSTRUCTIONS DE LECTURE / ECRITURE

Objectif: Etre capable d'écrire sans faute un algorithme simple de lecture/écriture

Connaissances Préalables : Savoir déclarer un objet

Les instructions de lecture/écriture encore appelées saisie/affichage sont à la base de la conversation effective entre l'homme et la machine ; la lecture interrompt l'exécution du programme, permettant ainsi à l'utilisateur de taper des données et l'écriture permet de lire la réponse affichée par cette machine.

I. INSTRUCTION DE LECTURE ou SAISIE

Nous avons vu que l'affectation était la première possibilité permettant de modifier le contenu d'une variable et nous avons même prévenu qu'il y a une deuxième possibilité ; et voilà nous y sommes ; c'est la **lecture**.

1. Syntaxe

SAISIR NOM DE VARIABLE

Exemple : SAISIR NOMSOC

2. Interprétation

Nous avons dit également (et ces rappels c'est pour vous éviter de remonter en arrière) qu'un algorithme ou mieux un programme est un ensemble d'instructions ordonnées dont l'exécution permet d'aboutir au résultat escompté : mais l'exécution c'est quoi ? C'est lorsque la machine (plutôt le pointeur ou le curseur) passe sur chacune des lignes ou instruction de votre programme pour faire ce que vous lui demandez, recommandez ou exigez. Ainsi lorsqu'elle arrive sur l'instruction '**SAISIR** NOMSOC' il y a un arrêt temporaire de l'exécution, l'ordinateur attend que vous tapiez une valeur

(chiffre, lettre, etc. etc.) au clavier et que vous confirmez la fin de votre frappe ou de votre saisie par une frappe sur la touche Entrée ou Enter ou Return ; et ce n'est qu'après cela que l'exécution du programme se poursuivra avec l'instruction qui suit '**SAISIR NOMSOC**'. Ce que vous avez tapé au clavier reste alors le contenu (ou le nouveau contenu) de la variable et ceci jusqu'à ce que :

- Soit le curseur revienne encore sur cette instruction (on verra cela dans le chapitre des boucles) permettant ainsi à l'utilisateur d'entrer une nouvelle valeur.
- Soit qu'une affectation ait lieu sur la variable modifiant ainsi son contenu.

Ainsi nous pouvons retenir que l'instruction '**SAISIR** Nom de variable' permet à l'utilisateur de faire entrer une valeur dans 'Nom variable'.

3. Règle principale à observer

La syntaxe montre que l'instruction ou l'ordre de lecture porte **OBLIGATOIREMENT** sur une variable et par conséquent cette variable doit avoir été (vous l'avez deviné j'espère) déclarée.

II. Instruction d'écriture ou AFFICHAGE

Comme le nom l'indique l'instruction d'écriture va permettre à la machine d'écrire ou d'afficher quelque chose à l'écran et donc à l'utilisateur de pouvoir lire ce que la machine a écrit. Ici encore ça va car c'est un ordre donné à la machine, donc la machine va écrire ; alors que plus haut c'est l'ordre de lecture mais la machine ne lit rien : c'est l'utilisateur qui lit. De grâce ne vous trompez pas !

1. Syntaxe

AFFICHER VALEUR

Remarquez ici la différence avec la lecture !!! (Nous avons écrit **valeur** à la place de **nom de variable**), c'est pour dire que l'écriture peut concerner aussi bien une variable qu'une expression ou même une constante.

2. Interprétation

Nous allons interpréter la syntaxe à partir de trois exemples en fonction de ce que nous venons de dire.

Exemple1 : **AFFICHER NOMSOC**

Ici valeur est une variable et ce qui sera affiché est le contenu de cette variable (CIE, SODECI ou Autre). Le résultat serait le même si NOMSOC était déclaré comme une constante

Exemple2 : **AFFICHER "NOMSOC" ;**

Du fait des griffes (" ") la machine considère qu'il s'agit d'une expression et dans ce cas c'est exactement ce qui est à l'intérieur des griffes qui sera affiché à l'écran et dans notre cas c'est NOMSOC

Exemple3 : **AFFICHER "Le montant de la facture est", Montant**

Dans cet exemple on a utilisé à la fois l'expression et la variable. Si dans la boîte ou la zone mémoire représentée par la variable Montant on avait 50 000, ce qui sera affiché dès que l'ordinateur passe sur cette instruction est

Le montant de la facture est 50 000

Avec ces instructions de lecture / Ecriture, nous pouvons à présent écrire nos premiers algorithmes. Mais pas trop vite, au lieu de vous demander d'écrire un programme, nous allons le faire pour vous, histoire de tester si vous avez compris ce qu'on appelle **exécution d'un programme – Affectation – Lecture – Ecriture**.

Exercice 9a

Quel résultat affichera la machine après l'exécution de ce programme

```
Programme test1
Var VAL1, VAL2 : Réel
Début
    VAL1 ← 100
    VAL2 ← 2*val1
FIN
```

Réponse (voir la partie du corrigé ou en classe mais réfléchissez d'abord)

Exercice 9b

Même question mais cette fois-ci après VAL2 ← 2*val1 on a écrit les deux instructions suivantes

Afficher VAL1

Afficher VAL2

Avec cet exemple vous avez vu les 3 parties d'un algorithme (qu'on vous rappelle ici)

- L'en-tête
- La déclaration des constantes et variables
- Le corps du programme qui commence par **DEBUT** et finit avec **FIN**

Exercice 10

Ecrire un programme qui demande à l'utilisateur d'entrer un nombre et qui affiche le carré de ce nombre.

Exercice 11

Ecrire un programme qui lit le prix HT d'un article, le nombre d'articles et le taux de TVA, et qui fournit le prix TTC correspondant. Faire en sorte que des messages apparaissent pour les différentes saisies à effectuer.

Exercice 12

Ecrire un algorithme qui affiche l'âge d'un individu à partir de son année de naissance tapée au clavier.

Exercice 13

Etant donné deux nombres saisis au clavier, affichez la nouvelle valeur de chacune des 2 variables après avoir effectué une permutation.

PRINCIPE : Au départ vous avez par exemple saisi dans la variable VAR1 la valeur 10 et dans la variable VAR2 la valeur 20 : Vous devez faire en sorte qu'à l'affichage (après permutation) les valeurs contenues dans VAR1 et VAR2 soient respectivement 20 et 10.

Avant de passer au chapitre suivant nous vous proposons ici un exercice hors du commun. Si vous trouvez la réponse vous pouvez dire « je suis sur la bonne voie » et dans le cas contraire relisez encore bien la partie du cours sur la définition d'un algorithme, sur les différentes étapes d'un algorithme et sur la déclaration des objets et si cela ne vous avance en rien, ne vous décourager pas : contactez-nous (mais comment ? vous le savez ! comment avez-vous fait pour recevoir ce support)

Des trois démarches ci-dessous, choisissez la bonne lors de l'écriture d'un algorithme.

1^{ère} démarche

- 1) Décomposer le travail en suite d'opérations qui fourniront les résultats en sortie
- 2) Définir les données d'entrée

2^e démarche

- 1) Définir les données d'entrée
- 2) Décomposez le travail en suite d'opérations élémentaires qui fourniront des résultats
- 3) Définir les résultats ou données de sortie

3^e démarche

- 1) Définir les données d'entrée et les résultats ou données de sortie
- 2) Décomposer le travail en une suite d'opérations élémentaires sur des données d'entrée.

CHAPITRE 4 : LES TESTS OU STRUCTURES CONDITIONNELLES

Objectif : *Etre capable d'écrire un algorithme dont l'exécution d'au moins une instruction est conditionnée, d'utiliser l'une ou l'autre des 4 structures décisionnelles pour résoudre un problème posé*

Connaissances Préalables :

Savoir écrire un algorithme simple de lecture / écriture

Vous savez maintenant que nous faisons de l'algorithme tous les jours, ne serait-ce qu'en indiquant le chemin à quelqu'un, tâche qui consiste en un ensemble d'instructions que nous pouvons résumer comme ci-dessous :

- Aller tout droit
- Au feu tournez à gauche
- Et voilà la pharmacie

L'ordre est si important que le résultat ne serait pas le même si on avait dit :

- Au feu tournez à gauche
- Aller tout droit
- Et voilà la pharmacie

Ce que nous faisons aussi tous les jours c'est de conditionner certaines de nos tâches c'est à dire exécuter certaines tâches selon une condition donnée. C'est le cas de ce que nous faisons tous les jours depuis notre réveil jusqu'à nous retrouver à notre bureau. Le réveil est sûr (encore faudrait il que Dieu le tout puissant nous le permette) mais la suite c'est quoi ?

- 1) Je me lève
- 2) Je me brosse
- 3) Je me lave
- 4) Je m'habille
- 5) Je vais au travail

Très peu de lecteurs se retrouveront dans cette logique, c'est normal parce que nous n'avons pas tous les mêmes habitudes : mais malgré cela on peut accepter que ces 5 tâches dans leur ordre vont permettre à l'individu de se retrouver au bureau. Et si c'était un dimanche ? Qu'en serait-il du point 5 ? L'algorithme devient alors

- 1) Je me lève
- 2) Je me brosse
- 3) Je me lave
- 4) Je m'habille

Si jour n'est pas un dimanche

- 5) Je vais au travail

Il en est de même si nous considérons encore le cas d'une indication de chemin avec incertitude. Nous sommes parfois amenés à dire :

- Aller tout droit
- Au prochain carrefour s'il y a un feu
- Vous tournez à gauche et voilà la pharmacie
- Dans le cas contraire c'est au carrefour sur prochain que vous tournez

Nous disons que l'instruction 'Tournez à gauche' est conditionnée, de même que l'instruction 5 (je vais au travail)

Nous verrons que ces instructions conditionnées peuvent être utilisées dans les 4 types de structures conditionnelles que sont :

- La structure conditionnelle simple
- La structure conditionnelle complète
- La conditionnelle imbriquée
- La structure à choix multiple

I. STRUCTURE CONDITIONNELLE SIMPLE

1. Syntaxe générale

SI CONDITION Alors

ACTION

FSI

2. Interprétation

- <ACTION> est une ou plusieurs instructions c'est à dire une ou plusieurs lignes de notre algorithme mais plus important ce sont des instructions qui ne seront exécutées que si la condition posée est vérifiée, mais qu'est-ce qu'une condition en informatique.
- <CONDITION> est la comparaison de deux valeurs ou plus précisément la comparaison du contenu d'une variable à une valeur : valeur ici est très large, mais retenez déjà qu'elle a les trois sens utilisés dans l'instruction **Afficher Valeur** vue précédemment mais en plus elle peut être un intervalle.

Pour faire ces comparaisons nous utilisons les opérateurs logiques et booléens dont on a parlé plus haut et qui sont principalement :

= **Egal à**

<> **Différent de**

< **Strictement inférieur à**

> **Strictement supérieur à**

<= **Inférieur ou égal à**

>= **Supérieur ou égal à**

Et / Ou / Non

Exemple

Dans l'algorithme précédent, il était question d'aller au travail seulement si le jour n'était pas dimanche ou encore si le jour est différent de dimanche. On écrira :

SI JOUR <> "Dimanche" alors

Ici Jour est une variable parce que justement son contenu peut être Lundi, Mardi.....Dimanche. Ces 7 contenus possibles sont écrits avec des lettres et non des chiffres par conséquent Jour est une variable de type Alphanumérique ; ce qui explique qu'on puisse le comparer à une expression sans oublier d'utiliser des quotes.

Quelle interprétation serait faite par la machine si on enlevait les quotes c'est-à-dire si on écrivait SI JOUR <> Dimanche. Si vous n'arrivez pas à répondre revoyez encore la partie déclaration des variables et la définition d'un objet.

Exercice 14

Ecrire un algorithme qui affiche l'âge d'un enfant à partir de son année de naissance saisie au clavier, l'algorithme doit afficher le message « peut déjà aller à l'école » Si l'enfant a plus de 2 ans

Cet exercice est le même que l'exercice N°12 jusqu'au calcul de l'âge, mais au moment où l'exercice N°12 prend fin cet exercice continue avec la ligne ci-dessous :

SI AGE > 2 alors

AFFICHER ("peut déjà aller à l'école")

FSI

On le voit bien, l'instruction ECRIRE ('peut déjà aller à l'école') qui va permettre l'affichage à l'écran d'un message ne sera exécutée qu'à condition que l'âge soit supérieur à 2.

Que faut-il retenir ?

- Une condition commence par SI
- Une variable ne s'écrit pas en deux mots encore moins en une phrase mais en un seul mot
- Une comparaison se fait grâce aux opérateurs de comparaison (= <= >= < >)

- Une variable peut être comparée à une valeur, un ensemble de valeurs ou encore à une ou plusieurs variables.

Exercice 15

Des expressions ci-dessous, lesquelles ne sont pas admises dans l'écriture d'un algorithme

- a) SI AGE > 2
- b) Si age est supé à 2
- c) SI AGEENFANT > 2
- d) Si enfant a plus de 2
- e) Si jour est dimanche
- f) SI JOUR = "Dimanche"
- g) SINB1 <> NB2
- h) Si le nombre1 est différent de nombr2

Remarque1 : La condition posée peut être vérifiée ou non c'est à dire être VRAIE ou FAUSSE, elle peut donc être une expression booléenne.

Remarque2 : Que se passe-t'il si la condition n'est pas vérifiée ?

Dans certains cas il y a d'autres tâches à exécuter et on parle alors de structure alternative.

II. Structure conditionnelle complète

1. Syntaxe générale

SI CONDITION Alors

«ACTION1»

SINON

«ACTION 2»

FSI

2. Interprétation

- <ACTION1> et <ACTION2> sont 2 blocs d'instructions qui seront exécutés de façon exclusive c'est à dire que les deux ne seront jamais exécutés pour une même condition ; où c'est l'un ou c'est l'autre et c'est pour cette raison qu'on parle **d'alternative**.
- Si <CONDITION> est vérifiée <ACTION1> est exécutée c'est à dire toutes les instructions entre SI et SINON et ensuite l'ordinateur passe sur l'instruction placée après FSI
- Si <CONDITION> n'est pas vérifiée c'est <ACTION2> qui sera exécutée c'est à dire toutes les instructions entre le SINON et FSI et ensuite l'ordinateur comme précédemment passe sur l'instruction placée après FSI.

3. Exemples

Exemple1 : Reprenons notre programme relatif aux tâches exécutées après le réveil chaque matin en disant que si le jour est un dimanche on reste à la maison.

L'algorithme devient :

SI JOUR <>"Dimanche"

AFFICHER "Je vais au travail"

SINON

AFFICHER "je reste à la maison"

FSI

Exemple2 : Reprenez l'exercice 14 mais cette fois pour l'âge strictement supérieur à 3 on affichera le message "peut être inscrit au CP1" et dans le cas contraire "reste au jardin"

Exercice 16

On désire écrire un algorithme qui demandera à l'utilisateur d'entrer deux nombres et l'ordinateur affichera le plus grand de ces 2 nombres. Traiter cet exercice

- a) Avec la structure conditionnelle simple
- b) Avec la structure conditionnelle complète

Exercice 17

Exerçons-nous à comprendre le programme écrit par quelqu'un d'autre

Observez cette partie du corps d'un algorithme.

SI JOUR = "LUNDI" alors

1. AFFICHER "JOUR DE TRAVAIL"

2. AFFICHER "JE VAIS AU TRAVAIL"

SINON

3. AFFICHER "JE SUIS À LA PRESIDENCE"

FSI

4. AFFICHER "JE SUIS CHEZ MOI"

Questions

- 1) A quelle condition chacune des instructions (1, 2, 3 et 4) sera exécutée
- 2) Quel est, ou quels sont le(s) message(s) qui sera (seront) affiché(s) si le jour est dimanche.
- 3) Même question si le jour est Mardi

Exercice 18

Ecrire un algorithme qui calcule et affiche la moyenne annuelle d'un étudiant à partir de ses deux moyennes semestrielles ; on suppose que la moyenne du 2^e semestre compte pour coefficient 2 et celle du 1^{er} semestre pour coefficient 1. On affichera le message « passe en classe supérieure » si la moyenne est égale ou supérieure à 10 et le message « redouble » dans le cas contraire.

Exercice 19

Proposez un algorithme produisant le même résultat que l'algorithme ci-dessous

SI (BON>MAUVAIS+4) OU (Gentil="Ok") alors

BON ← BON+1

SINON

BON ← BON-1

FINSI

Exercice 20

Ecrire l'algorithme permettant de saisir deux nombres et d'afficher le plus grand des deux (on suppose que les deux nombres saisis sont différents).

Exercice 21

Reprendre l'exercice précédent mais affichez cette fois-ci en plus, le message « les deux nombres sont égaux » en cas d'égalité.

Si vous avez résolu ce dernier exercice je vous dis BRAVO vous êtes en avance par rapport au cours et si vous avez des difficultés à le traiter, alors ne vous attristez pas, suivez-nous.

Nous vous avons prévenu que pour être « Bon » en algorithme vous devez maîtriser ses 4 notions qui sont l'affectation, les instructions de lecture / écriture, les tests et les boucles ; Malheureusement les difficultés ne sont pas identiques sinon on penserait qu'on est presque au bout de notre peine vu qu'on est sur la 3^e notion (les test) : donc armez vous de courage et de patience et surtout essayez de ne pas oublier tout ce que vous avez déjà compris jusqu'ici et plus précisément sur le test ; parce que maintenant ça va être un peu plus compliqué et vous pouvez vous en douter car dans la vie courante ce n'est pas toujours de 2 choses l'une ou l'autre, cela peut être

également 3 choses, 4 choses ou encore plus. Si on estime que du Lundi au Vendredi je vais au travail ; le Samedi, je suis avec les amis et le Dimanche je suis en famille, l'algorithme à écrire pourrait être :

SI JOUR ="Lundi" ou JOUR ="Mardi" ouou JOUR = "vendredi" alors

AFFICHER ("Je vais au travail")

SINON

SI JOUR = "Samedi" alors

AFFICHER ("Je suis avec les amis")

SINON

AFFICHER ("Je suis en famille")

FSI

FSI

Un SI après un SINON, ça c'est nouveau !!! On dit qu'il y a imbrication de SI ou encore on parle de Condition imbriquée.

III. Structure conditionnelle imbriquée

1. Utilisation

Elle est utilisée lorsqu'une condition à elle seule ne permet pas de prendre une décision ou d'exécuter une instruction donnée.

Dans l'exemple précédent il y avait 3 instructions ou trois messages à afficher, un excluant les autres : Ces messages et les conditions correspondantes sont

- Je vais au travail SI Jour est compris entre Lundi et Vendredi
- Je suis avec les amis SI Jour est Samedi
- Je suis en famille SI Jour est Dimanche

2. Interprétation

Que signifie le premier SINON pour l'ordinateur ?

Tout simplement que le jour n'est ni Lundi ni Mardi ni Mercredi ni Jeudi ni vendredi.

Le jour est quoi alors ?

Nous avons encore 2 autres possibilités : samedi ou Dimanche or l'instruction à exécuter le samedi n'est pas la même que celle du dimanche ; c'est pour cette raison que connaissant déjà une condition (jour n'est ni lundi, mardi.....vendredi) nous ne pouvons encore préciser à la machine le message à afficher ou l'instruction à exécuter ; nous sommes obligés de poser une nouvelle condition d'où la présence d'un autre SI après le SINON pour demander si le jour est samedi (et dans ce cas c'est le message 'je suis avec les amis') qui doit être affiché.

Et si le jour n'est pas Samedi ? Franchement de vous à moi le Jour est quoi alors ?

On n'a pas encore un 8^e Jour de la semaine et cela ne viendra probablement jamais, c'est donc forcément Dimanche ; ce qui explique qu'après le 2^{ième} SINON il n'est plus nécessaire de poser une autre condition.

3. Différence avec les deux premières structures

Avant de vous dire s'il y a différence ou pas essayez d'écrire le même programme en utilisant la structure conditionnelle. Vous pouvez aussi consulter la correction si vous êtes paresseux.

En obtenant le même résultat nous pouvons dire qu'il n'y a pas fondamentalement de différence avec les deux précédentes structures et ce qu'il convient de retenir est qu'on peut résoudre le même problème en utilisant l'une ou l'autre des 3 structures de test, et que la structure imbriquée peut être conseillée lorsqu'on a plus de 2 actions, structure permettant de supprimer progressivement (de mettre à l'écart) les différentes possibilités jusqu'à ce qu'il n'en reste qu'une seule et à ce moment il y aura un dernier SINON qui n'aura plus de SI mais directement la dernière action qui reste à exécuter.

Exercice 22

Ecrire un algorithme qui affiche le montant à payer par un étudiant à partir de la saisie de son Nom, Prénom, Montant normal de la scolarité et statut qui peut être O (Orienté) P (Privé) F (Forfait) en tenant compte des conditions suivantes.

- Privé : Aucune réduction n'est accordée
- Orienté : 100% de réduction accordée
- Forfait : 50% de réduction accordée

Dans l'exemple que nous avons choisi pour expliquer les tests imbriqués nous avons utilisé un des opérateurs déjà mentionnés plus haut ; il s'agit de **OU**.

SI JOUR= "Lundi" OU SI JOUR= "Mardi".....SI JOUR= "Vendredi". Remarquez que l'opérateur **Et** n'aurait pas de sens car quel est ce jour qui serait à la fois Lundi et Mardi ?

Donnons des exemples et leur signification.

a) SI NB1 \neq 0 et NB1 \neq NB2 alors

Signifie « Si NB1 est différent de Zéro (0) et NB1 est encore différent de NB2 ». En d'autres termes les instructions conditionnées par cette ligne seront exécutées si les deux conditions qu'elle regroupe sont vérifiées ; et dès que l'une des deux n'est pas vérifiée ces instructions ne seront pas exécutées. Cette condition peut être encore écrite de la façon suivante :

SI NB1 \neq 0 alors

SI NB1 \neq NB2 alors

C'est exactement la même chose mais moins compréhensible.

En fait la programmation, cela peut devenir un jeu, un plaisir si l'on maîtrise les tests et les boucles que nous allons voir (mais pas tout de suite), si l'on comprend comment l'ordinateur interprète chacune des lignes du programme et surtout si l'on sait que l'ordinateur est un robot, un exécutant

des ordres plus que tout esclave et enfin un travailleur médiocre car il ne fera ni plus ni moins que ce qu'on lui demande de faire; mais si l'on ne comprend pas cela, la programmation peut alors devenir de la peste malheureusement, on voudra l'éviter et si on la fuit il faut également perdre cette envie, ce goût d'écrire un jour une application.

b) SI $NB1 \neq 0$ ou $NB2 \neq NB1$

Les instructions conditionnées par cette ligne seront exécutées si l'une au moins des conditions est vérifiée c'est à dire

- SI $NB1$ est différent de 0 quelque soit $NB2$.
- SI $NB2$ est différent de $NB1$ quelque soit $NB1$.

Pour que cette condition soit fausse il faut que $NB1$ soit égal à 0 et que $NB2$ soit égal à $NB1$.

L'utilisation de ces opérateurs logiques ne constitue pas de véritables difficultés surtout pour tous ceux qui connaissent les tables de vérité et c'est votre cas.

Donnez le contraire des conditions suivantes ?

1/ SI $A > 0$

2/ SI $A > 0$ et $B = 0$

3/ SI $CATEG = 'A'$ ou $NBRENF > 4$

4/ SI $NBRE = 10$ ou $NOM = "FIN"$

5/ SI $A > 10$ et $A < 100$

Exercice 23

Le salaire net à payer à un employé dans la société RIA se calcule en ajoutant au salaire brut une prime qui est fonction de la catégorie de l'employé. Ainsi la prime est égale à :

- $0,1 \times \text{salaire brut}$ pour la catégorie 1
- $0,09 \times \text{salaire brut}$ pour la catégorie 2
- $0,07 \times \text{salaire brut}$ pour la catégorie 3

- 0 pour toutes les autres catégories.

Ecrire l'algorithme permettant de solutionner ce problème.

Exercice 24

Etant donné la température de l'eau affichez un message qui dit son état sachant que cet état est :

- De la glace pour température inférieure ou égale à 0
- du liquide pour température comprise entre 0 et 100
- de la vapeur pour température supérieure à 100

Ces deux exercices (et surtout le No 23) nous permettent de constater que l'utilisation des tests imbriqués n'est pas toujours aisée car on peut avoir plus d'une dizaine de cas possible et on risque de se perdre dans les imbrications. Fort heureusement les informaticiens (qui ne supportent pas voir ceux qui font leur premier pas en programmation souffrir ou déçus d'avoir choisi cette branche de l'informatique) ont pensé à une autre façon beaucoup plus aisée de traiter ces genres de problèmes où nous avons beaucoup de condition : c'est la **structure de choix**.

IV. STRUCTURE A CHOIX MULTIPLE

1. Format Général

SELON < NOM DE VARIABLE > FAIRE

VALEUR1 : < ACTION1 >

VALEUR2 : < ACTION2 >

.

.

.

VALEURn : < ACTIONn >

SINON

< ACTION PAR DEFAULT >

FSELON

2. Interprétation

- Nom de variable : c'est le nom que vous donnez à votre variable (A, B, NOM, NBRE...).
- VALEUR1, VALEUR2 VALEURn sont les différentes valeurs que peut prendre la variable.
- <ACTION1>.....<ACTIONn> sont les actions à exécuter si la variable prend respectivement VALEUR1, VALEUR2 VALEURn
- SINON : est facultatif et permet de préciser dans le cas échéant l'action que l'ordinateur doit exécuter si le contenu de la variable n'est égal à aucune des valeurs précisées (VALEUR1 à VALEURn). Dans certains sujets toutes les possibilités sont prises en compte dans les VALEUR1, 2, à n et dans ce cas le SINON n'a plus sa raison d'être.

Cette structure, comme nous l'avons dit permet d'éviter les tests imbriqués et par conséquent, bien que cette structure permette une meilleure lecture de l'algorithme, le programmeur (surtout celui qui préfère la complexité à la simplicité) n'est pas obligé de l'utiliser. Cette structure correspond encore à ceci :

SI VARIABLE = VALEUR1 alors

 <ACTION1>

FSI

SI VARIABLE = VALEUR2 alors

 <ACTION2>

FSI

;

;

SI VARIABLE = VALEURn alors

 <ACTIONn>

FSI

SI VARIABLE <>VALEUR1 et VARIABLE <> VALEUR2..... et
VARIABLE<> VALEURn alors

<ACTION PAR DEFAULT>

FSI

Remarque : VALEUR1 à VALEURn peut être effectivement précisée
(10 ; 20 ; A ; B) mais peut être également un domaine ou intervalle de valeurs
par exemple de 0 à 9 et là on écrira 0..9 : <ACTION> ;

Cela peut être également l'une ou l'autre de plusieurs valeurs non
consécutives par exemple 4 ou 6 ou 9 et là on écrira 4,6,9 : <ACTION> ;

Enfin Valeur1 comme toutes les autres valeurs peut être également un
caractère alphabétique et là on écrira 'A' : <ACTION> si A est une des
valeurs possibles pour la variable

Exercice 25

Reprendre l'Exercice No 23 en utilisant une structure à choix multiple.

Exercice 26

Ecrire un algorithme qui calcule l'âge d'un enfant et informe
l'utilisateur de la catégorie de cet enfant qui peut être :

- « Poussin » de 6 à 7 ans
- « Pupille » de 8 à 9 ans
- « Minimes » de 10 à 11 ans
- « Cadet » à partir de 12 ans.

Nous vous conseillons de traiter cet algorithme de 3 manières sous les
numéros 26a, 26b, 26c en utilisant trois structures différentes

Conclusion sur les TESTS

Nous avons essayé de traiter toutes les formes de structure décisionnelle (conditionnelle, alternative, imbrication, Structure de choix). Dans la pratique il est rare qu'une structure soit imposée car c'est le résultat qui compte mais attention !!! Pour le moment on cherche à résoudre des problèmes en écrivant un certain nombre d'instructions, quant à savoir la meilleure solution c'est probablement celui où on écrit moins. Donc nous vous conseillons de vous exercer afin de maîtriser cette 3^e notion (les Test) des 4 qui font l'objet de ce cours, mais une 3^e notion qu'on peut considérer comme la toute première difficulté dans l'écriture d'un algorithme. C'est pour cette raison que nous vous proposons dans les pages qui suivent une série d'exercices dont certains sont corrigés, d'autres sont accompagnés d'explication pour leur résolution et d'autres feront l'objet de vos propres recherches. Avant de passer à cette série d'exercices nous voudrions conclure le cours sur les TESTS par quelques règles ou remarques ou rappels.

- 1) Une condition est une comparaison.
- 2) Ce qui est valable en Mathématique ou a un sens en Français n'est pas forcément applicable en Informatique. Ainsi SI $0 < A < 100$ (qui signifie si A est compris entre 0 et 100) ne veut rien dire en programmation où l'on écrira par exemple SI $A > 0$ et $A < 100$.
- 3) Dans un algorithme toute condition écrite doit pouvoir être VRAI au moins une fois. Ainsi SI $A > 10$ et $A < 0$ ne pouvant être jamais réalisée c'est une condition qui ne doit figurer dans aucun algorithme.
- 4) La présence de parenthèses dans des conditions composées et employant à la fois les opérateurs ET, OU a une grande influence sur le résultat comme dans le cas de l'utilisation des multiplications et additions avec les variables numériques

EXERCICES DE SYNTHÈSE

Exercice 27 (BTS EPP Partie pratique)

Une compagnie de chemin de fer pratique ses tarifs en fonction de la catégorie du voyageur ;

- Voyageur normal (N) : plein tarif
- Abonné de la compagnie (A) : 40% de réduction
- Enfant de moins de 10 ans (E) : gratuit
- Employé de la compagnie (T) : gratuit
- Parent d'un employé (P) : 50% de réduction
- Personne âgée 70 ans (V) : 30% de réduction

Travail à faire

Concevez une application qui permet, connaissant le tarif normal du voyage, de calculer le prix à payer par un voyageur donné.

Exercice 28 (BTS EPP Partie pratique)

Un transporteur routier vous demande d'écrire un algorithme qui lui permette de calculer les frais de transport des colis qu'il véhicule sachant que :

- La taxe de base applicable à tout colis est de 6500
- Si le colis pèse plus de 60kg, une surtaxe de 650 par kilo supplémentaire est ajoutée.
- Si une des dimensions (largeur, longueur, hauteur) du colis dépasse 1 mètre, 1350 f sont perçus.
- 10% du total obtenu sont ajoutés pour tout trajet supérieur à 100 km

Etant donné la largeur, la hauteur, la longueur, le poids du colis et la distance à parcourir, écrire un algorithme qui calcule le prix à payer pour le transport d'un colis.

Exercice 29 (BTS EPP Partie théorique)

Un concessionnaire automobile désire qu'on écrive à l'intention de ses clients un algorithme qui calcule les frais mensuels d'utilisation des voitures qu'il vend. Etant donné le nombre de kilomètres que parcourt le client en une année, le type de carburant utilisé (**G** pour diesel et **E** pour essence), et la cylindrée de la voiture, écrire l'algorithme sachant que :

- Si la voiture est à essence et la cylindrée est supérieure à 2000 cm³, le coût du carburant est calculé en tenant compte que la voiture consomme 10 litres aux 100 km.
- Si la voiture est à essence et la cylindrée est inférieure à 2000 cm³, le coût du carburant est calculé en tenant compte que la voiture consomme 8 litres aux 100 km.
- Si la voiture est à diesel, le coût du carburant est calculé en tenant compte que la voiture consomme 8 litres aux 100 km.
- Un surcoût de 70% pour les frais d'entretien est appliqué aux véhicules diesel. Un surcoût de 60% pour les frais d'entretien est appliqué aux véhicules essence.

Exercice 30 (BTS EPP Partie pratique)

Dans une entreprise nommée IVOIRE PRESTIGE de la place le calcul des salaires est soumis à l'impôt

- De 1 à 90 000 → 2 % d'impôt
- De 90 001 à 500 000 → 10 % d'impôt
- Plus de 500 000 → 25 % d'impôt

Sachant que l'impôt se déduit du salaire de base catégoriel, écrire un algorithme qui permet de calculer le salaire net d'un employé.

Exercice 31 (BTS EPP Partie pratique)

Concevoir un programme pour calculer le montant d'impôt à payer, selon le revenu mensuel :

- De 0 à 90 000, 0 % d'impôt.
- De 90 001 à 500 000, 10 % d'impôt.
- Plus de 500 000, 25 % d'impôt

Prévoir une réduction des impôts due selon l'âge du déclarant :

- Moins de 18 ans, afficher un message d'erreur (« Non imposable !!! »)
- De 18 à 25 ans, subvention de 5 000 F (mais pas de remboursement si la subvention est plus grande que l'impôt).
- Plus de 25 ans, aucune subvention.

Exercice 32 (BTS EPP Partie pratique)

Nous appelons « Menu Principal » l'association d'une valeur d'une variable de type scalaire aux différents choix d'un algorithme.

A partir d'un « Menu Principal » affiché à l'écran :

- Effectuer la somme de trois nombres réels,
- Effectuer le produit de trois nombres réels,
- Effectuer la moyenne de trois nombres réels

Travail à faire

Concevez une application qui permet à partir d'un Menu Principal de faire :

- la somme de trois nombres réels,
- le produit de trois nombres réels,
- la moyenne de trois nombres réels

Exercice 33

Ecrire un programme qui affiche la saison en fonction du mois saisi au clavier sachant que nous sommes :

- En Hiver pour les valeurs 12,1,2
- Au Printemps pour les valeurs 3,4,5
- En Eté pour les valeurs 6,7,8
- En Automne pour les valeurs 9,10,11

Exercice 34

Concevez un algorithme qui affiche si un habitant du pôle nord doit payer ou non l'impôt. Dans cette partie du monde les règles appliquées sont :

- Les hommes de plus de 20 ans paient l'impôt
- Les femmes paient l'impôt si elles ont entre 18 et 35 ans
- Tous les autres ne paient pas l'impôt

Exercice 35

Ecrivez un algorithme qui demande à l'utilisateur d'entrer un numéro de jour, de mois et d'année et qui affiche si la date entrée est valide ou non

Exercice 36

Proposez un programme qui permet de résoudre l'équation $ax^2+bx+c=0$

Exercice 37

Les élections législatives, dans un pays africain obéissent à la règle suivante :

- Lorsque l'un des candidats obtient plus de 50 % des suffrages, il est élu dès le premier tour.
- En cas de deuxième tour, peuvent participer uniquement les candidats ayant obtenu ou moins 12,5 % des voix au premier tour.

Vous devez écrire un algorithme qui permette la saisie des scores de quatre candidats au premier tour. Cet algorithme traitera ensuite un seul candidat (le premier de la liste par exemple) et dira s'il est élu, battu, s'il se trouve en ballottage favorable (il participe au second tour en étant arrivé en tête à l'issue du premier tour) ou défavorable (il participe au second tour sans avoir été en tête au premier tour).

CHAPITRE 5 : LES STRUCTURES DE BOUCLE

***Objectif :** Être capable d'écrire un algorithme dont l'exécution peut être reprise plusieurs fois par l'ordinateur, que le nombre de fois soit défini ou non.*

***Connaissances Préalables :** Être capable d'utiliser l'une ou l'autre des 4 structures décisionnelles pour résoudre un problème posé*

INTRODUCTION

La plupart des tâches exécutées dans une entreprise sont répétitives. Si nous nous plaçons dans le cadre scolaire, à chaque inscription on écrit (on dira en informatique qu'on saisit ou encore on enregistre) les données relatives à l'étudiant inscrit c'est à dire son nom, prénom et autres informations (nous savons écrire un tel algorithme). Quand un 2^e étudiant s'inscrit on fera la même chose et cela sera ainsi même pour le centième étudiant ; une des raisons de la structure de boucle est de nous aider à ne pas écrire 100 fois ou n fois la même instruction « Saisir NOM » mais à l'écrire une seule fois et obliger la machine à passer sur cette instruction autant de fois que nous désirons afin de saisir autant de nom que nous souhaitons. C'est dire qu'on maintiendra les instructions déjà écrites pour traiter un seul étudiant, mais en plus on ajoutera d'autres instructions qui nous permettront de repasser plusieurs fois sur les instructions préalablement écrites ; on dira qu'on boucle sur ces instructions. Les instructions ou commandes permettant de boucler, de reprendre les mêmes instructions un certain nombre de fois sont au nombre de 3 à savoir :

- LA STRUCTURE REPETER
- LA STRUCTURE TANT QUE
- LA STRUCTURE POUR

I. STRUCTURE REPETER

1. Format Général

REPETER

< ACTION >

JUSQU'A <CONDITION>

2. Interprétation

<ACTION> (vous le savez déjà) est une ou plusieurs instructions ou mieux c'est ce qui se fait pour une unité (un étudiant, un produit etc. etc.) et qui sera reprise.

<CONDITION> nous savons également ce qu'est une condition (au cas où vous l'auriez oublié, reportez vous à la structure conditionnelle) : ce qui est nouveau ici c'est le **SI** qui disparaît c'est à dire qu'on n'écrira pas « JUSQU'A **SI** A = 10 » mais on écrira JUSQU'A A = 10

Mais ça veut dire quoi ça ?

Cela signifie que SI A prend la valeur 10, on sortira de la boucle pour se positionner sur l'instruction écrite immédiatement après JUSQU'A ; autrement dit <ACTION> sera répétée, reprise tant que la valeur de A ne sera pas 10.

Conséquence de la structure de boucle. Comme nous l'avons vu dans la conclusion sur les TESTS, une condition posée doit pouvoir être réalisée au moins une fois sinon on ne pourra jamais sortir de la boucle. La condition posée ici aura pour conséquence l'ajout d'autres instructions à <ACTION> qui s'exécutait pour une unité. Ces instructions ajoutées aux anciennes vont alors constituer la nouvelle « ACTION » de la boucle. Le nombre de fois que <ACTION> sera reprise dépendra de la condition posée et nous montrerons cela par quelques exemples.

3. Exemple de boucle

a) On veut écrire un algorithme permettant de saisir 10 NOM et PRENOM

b) On dit ici que le **nombre** de NOM et PRENOM à saisir est connu ; en l'occurrence 10 : On déclare alors une variable appelée Nombre ou Compteur ou autre et après la saisie d'un NOM et PRENOM on incrémente (on ajoute 1 à) cette variable et l'algorithme devient

VAR Nom,Prenom : CC

Compteur : Entier

REPETER

AFFICHER ("Entrez le NOM")

SAISIR (NOM)

AFFICHER ("Entrez le PRENOM")

SAISIR (PRENOM)

Compteur \longleftarrow Compteur + 1

JUSQU'A Compteur = 10

Cet algorithme, dès que 10 NOM et PRENOM sont saisis le curseur sort de la boucle pour se retrouver sur l'instruction qui suit immédiatement JUSQU'A.

Pour ceux qui pensent avoir compris répondez aux questions suivantes

- 1) Si on écrivait l'instruction COMPTEUR \longleftarrow COMPTEUR + 1 après JUSQU'A quel serait le résultat ?
- 2) Si avant REPETER on avait l'instruction COMPTEUR \longleftarrow 10, quel serait le résultat. ?

Pour les réponses aux deux questions, soyez présents au cours car c'est maintenant que vous aller commencer à écrire de vrais algorithmes.

c) On veut écrire un algorithme qui permet cette fois-ci de saisir autant de nom et prénom que l'on désire.

Ici c'est l'utilisateur qui décidera de mettre fin ou non à la répétition en faisant une action qui bien évidemment est prévue dans l'algorithme et qui permet de sortir de la boucle.

Cet algorithme peut être écrit de la façon suivante.

VAR NOM, PRENOM, REP : CC

REPETER

AFFICHER ("Entrez le NOM")

SAISIR (NOM)

AFFICHER ("Entrez le PRENOM")

SAISIR (PRENOM)

AFFICHER ("Voulez-vous continuer O/N ? ")

SAISIR (REP)

JUSQU'A REP= 'N'

Ici l'utilisateur saisit autant de noms qu'il désire et à chaque fois l'ordinateur lui demande s'il veut continuer. Il lui suffit alors de taper la touche N (qui signifie ici Non) pour arrêter c'est-à-dire sortir de la boucle.

Ce même algorithme peut être encore écrit de la façon suivante :

VAR NOM, PRENOM, REP : CC

REPETER

AFFICHER ("Entrez le NOM ou taper la touche Entrée pour arrêter")

SAISIR (NOM)

SI NOM<>" " alors

AFFICHER ("Entrez le PRENOM")

SAISIR (PRENOM)

FSI

JUSQU'A NOM= " "

Ici l'utilisateur saisit également autant de noms qu'il désire mais la condition de sortie de la boucle n'est pas de taper la lettre N mais de ne rien taper à la place du nom.

Pour ceux qui pensent avoir compris complétez l'algorithme pour que la machine affiche le nombre total de NOM saisi (RDV au cours)

La structure de boucle comme nous l'avons dit dans l'introduction est faite principalement pour ces genres d'algorithme que nous venons d'écrire, mais quand elle est comprise elle est alors utilisée à n'importe quel endroit d'un algorithme où sa présence est nécessaire, c'est ce qui permet d'avoir dans le même algorithme une infinité de boucle surtout lorsque l'on doit obliger l'utilisateur à saisir une valeur faisant partie d'un ensemble qu'une variable peut prendre : c'est ce que nous verrons avec les exemples c et d.

d) Ecrire un algorithme qui saisit 2 nombres nécessairement différents

- Solution 1

VAR x, y : N

REPETER

AFFICHER ("Entrez 2 nombres différents")

SAISIR (x, y)

JUSQU'A $x \neq y$

- Solution 2

VAR x, y : N

REPETER

AFFICHER ("Entrez le premier nombre")

SAISIR (x)

AFFICHER ("Entrez le deuxième nombre")

LIRE (y)

JUSQU'A $x \neq y$

- Solution 3

VAR x, y : N

AFFICHER ("Entrez le première nombre")

SAISIR (x)

REPETER

 AFFICHER ("Entrez le deuxième nombre")

 SAISIR (y)

JUSQU'A x <> y

Toutes ces 3 possibilités permettent de SAISIR 2 nombres différents mais quelle est la différence entre la solution 1 et la solution 2 ? Quelle est la différence entre la solution 2 et la solution 3 ?

Exercices

On ne va pas se compliquer la vie. Si on se réfère au début de ce chapitre on comprendra qu'on peut prendre les exercices déjà traités et les adapter à la structure de boucle. Vous avez donc des exercices pour vous entrainer avant qu'on ne vous en propose de nouveaux et si vous avez l'embarras du choix traiter les exercices suivants

Exercice 38 (26R) (Comprenez reprise de l'exercice26 en Répétitive)

Reprendre l'exercice 26 en supposant que nous avons 50 enfants et permettre à l'ordinateur d'afficher le nombre total d'enfants pour chaque catégorie

Exercice 39 (27R)

Reprendre l'exercice 27 pour un nombre indéterminé de voyageurs et donner en fin de traitement le nombre total de voyageurs traités

Exercice 40 (28R)

Reprendre l'exercice 28 pour un nombre indéterminé de colis et donner en fin de traitement le nombre total de colis traités

Exercice 41 (32R)

Reprendre l'exercice 32 jusqu'à ce qu'un des nombres saisis soit nul et le message suivant s'affichera « Désolé, un de vos nombres est nul ; Tapez la touche **Entrée** pour sortir ».

Exercice 42 (34R)

Reprendre l'exercice 34 ; mais cette fois-ci l'utilisateur communique à l'ordinateur le nombre total d'habitants à traiter, effectue les saisies et l'ordinateur devra afficher le nombre total d'hommes payant l'impôt et le pourcentage correspondant

Exercice 43 (35R)

Reprendre l'exercice 35 mais cette fois-ci on fera 10 fois la saisie. Toute fois dès qu'on saisit successivement 3 dates invalides le programme s'arrête après avoir affiché l'un ou l'autre des messages suivants :

- « Saisie non achevée car Erreurs successives de date »
- « Saisie achevée sans 3 Erreurs successives de date »
- Saisie achevée mais avec 3 Erreurs successives de date »

Exercice 44

Etant données des couples (quantité commandée, prix unitaire) communiquées au terminal (clavier), écrire un algorithme calculant le montant total de la facture ; l'algorithme devra arrêter la lecture des données dès que l'on communique soit une quantité négative, soit un prix unitaire négatif, ou dès que le montant total atteint un plafond (communiqué au début de la lecture des données).

Exercice 45

Tout achat chez AGGROS est matérialisé par une facture comportant un numéro, le nom du client, la date achat et les différents articles payés (Nom, Qté , Pu). Lorsque le montant de l'achat est supérieur à 500 000, le client bénéficie d'une remise de 10%.

Proposer un algorithme permettant de saisir 10 factures et afficher :

- le montant à payer après chaque facture
- Le nombre total de factures ayant bénéficié d'une remise
- Le Numéro de facture dont le montant est le plus élevé ainsi que le nom du client correspondant

Exercice 46

Ecrire un algorithme qui saisit le nom de chacun des 30 étudiants d'une classe ainsi que les différentes notes obtenues par chacun d'eux. Sachant que le nombre de notes peut varier d'un étudiant à l'autre et que toutes les notes sont de coefficient 1, on vous demande d'afficher :

- La moyenne de chaque étudiant
- La moyenne du premier et son nom
- La moyenne du dernier et son nom
- La moyenne de la classe

Exercice 47

Médecin sans frontière est un organisme qui travaille partout dans le monde. Présentement, l'organisme a des problèmes de conversion de température. Il voudrait convertir une température exprimée en degré Celsius en son équivalent en Fahrenheit et vis versa. La formule est la suivante :

$$\text{Fahrenheit} = (\text{Celsius} / 5) * 9 + 32$$

Ecrire l'algorithme qui permet de réaliser cette conversion pour 15 valeurs. L'utilisateur a le choix d'une conversion dans l'une ou l'autre unité

Exercice 48

Ecrire un algorithme qui demande un nombre de départ, et qui calcule et affiche la somme des entiers jusqu'à ce nombre. Par exemple, si l'on entre 5, le programme doit calculer :

$$1 + 2 + 3 + 4 + 5 = 15$$

Exercice 49

Ecrire un algorithme qui demande un nombre de départ, et qui calcule sa factorielle.

NB : la factorielle de 8, notée 8 !, vaut $1 \times 2 \times 3 \times 4 \times 5 \times 6 \times 7 \times 8$

II. STRUCTURE TANT QUE

C'est une autre forme de boucle et pour l'expliquer nous prenons la solution 2 proposée pour saisir 2 nombres différents.

Dans cette solution on répétait l'action c'est à dire la saisie de x et y jusqu'à ce que x soit différent de y ; sous une autre forme ou compréhension, cela signifie que tant que x n'est pas différent de y on reprendra la saisie de x et de y ou encore tant que x est égal à y on reprendra la saisie de x et y : ce qu'on peut présenter de la façon ci-dessous :

TANT QUE X=Y

SAISIR (X)

SAISIR (Y)

Comme la présence de JUSQU'A qui indique la dernière ligne faisant partie des instructions à répéter, la boucle TANT QUE a également FINTANTQUE (ou FTQ) pour indiquer la dernière instruction du bloc.

1. Format général

TANT QUE <CONDITION> faire

<ACTION>

FTQ

2. Interprétation

C'est la même que dans la structure répétitive mais ce qu'il faut remarquer ici est que la condition permet ici de rester dans la boucle et par conséquent elle doit être formulée de telle sorte qu'elle soit contraire à la condition utilisée dans la boucle répétitive et qui permet elle d'en sortir. Le tableau ci-dessous exprime quelques conditions et leur contraire.

CONDITION	CONTRAIRE
$X = Y$	$X \neq Y$
$X > Y$	$X \leq Y$
$X < Y$	$X \geq Y$

3. Différence capitale entre les 2 types de structure.

Dans la structure répétitive le curseur passera (on vient de le dire) sur la condition après être passé sur <ACTION> et si au premier tour la condition de sortie de la boucle est vérifiée <ACTION> aura été exécutée une seule fois.

Dans la structure itérative le curseur rencontre la condition avant d'atteindre l'action ; ainsi si cette condition n'est pas vérifiée le curseur n'entrera pas dans la boucle et <ACTION> ne sera pas exécutée même une seule fois.

4. Exemple d'algorithme

Reprenons la solution 2 en structure TANTQUE : on écrira :

TANT QUE $x = y$ faire

AFFICHER ("Entrez le première nombre")

SAISIR (x)

AFFICHER ("Entrez le deuxième nombre")

SAISIR (y)

FTQ

Reprenons la solution 3 en structure TANTQUE : on écrira :

```
VAR x, y : N
    début
        AFFICHER ("Entrez le premier nombre")
        SAISIR (x)
        TANT QUE x = y faire
            AFFICHER ("Entrez le deuxième nombre")
            SAISIR (y)
    FTQ
```

C'est bien là une erreur à ne pas commettre ; en effet dès qu'on a déclaré deux variables et qu'on saisit une sans l'autre, il y a toutes les chances que le contenu des deux soit différent et dans notre cas ici on ne pourra plus entrer dans la boucle pour saisir Y parce que la condition permettant d'y entrer ($X = Y$) ne sera pas vérifiée. Que fallait-il faire alors ? Saisir les deux variables à l'extérieur et ouvrir ensuite la boucle pour permettre une éventuelle reprise de saisie au cas où les deux seraient de contenus identiques (cela est bien évidemment plus long et c'est normal car dans plusieurs possibilités il y a toujours une moins meilleure) et cela donne :

```
VAR x, y : N
    début
        AFFICHER ("Entrez le premier nombre")
        SAISIR (x)
        AFFICHER ("Entrez le deuxième nombre")
        SAISIR (y)
        TANT QUE x = y faire
            AFFICHER ("Entrez le deuxième nombre")
            SAISIR (y)
    FTQ
```


Exercices

Comme précédemment, il serait intéressant de reprendre les exercices déjà traités avec la structure répétitive avant de passer à de nouveaux exercices ; c'est pour cela que nous vous proposons de reprendre cette fois-ci en structure TANT QUE les exercices 26R, 27R, 32R, 35R, 44, 46, 47, 48, 49.

III. STRUCTURE POUR

1. Format général

Pour variable \longleftarrow Val initiale à Val finale faire
 <ACTION>

Fpour

2. Interprétation

Contrairement aux deux structures de boucles précédentes, la condition de sortie ne se pose pas de façon explicite en terme de comparaison, mais en réalité il s'agit toujours d'une comparaison dans laquelle la variable prend une première valeur (**valeur initiale**) à laquelle le chiffre 1 sera ajouté (on parle d'incrément) après chaque exécution de <ACTION> jusqu'à ce que la valeur de la variable atteigne **valeur finale**.

3. Différences capitales avec les 2 structures précédentes

La différence capitale est l'absence de l'instruction de l'incrément (variable := variable + 1) : En effet, dans cette structure l'incrément est automatique. Cela signifie également que Valeur initiale est forcément inférieure ou égale à valeur finale : ce qui n'est pas le cas dans les 2 autres parce que la décrémentation est possible. Enfin une autre différence est qu'il y a une valeur finale qui doit être connue (5, 10 ou N à condition que N ait été déclaré et saisi).

Si nous prenons l'exercice d'application permettant de saisir 10 NOMS et PRENOMS, cette solution en structure POUR devient :

VAR Nom,Prenom : CC

Compteur : Entier

POUR Compteur ← 1 à 10 faire

AFFICHER ("Entrez le NOM")

SAISIR (NOM)

AFFICHER ("Entrez le PRENOM")

SAISIR (PRENOM)

FPOUR

Exercices Reprendre en structure **Pour** les exercices 26R, 35R, 46, 47, 48, 49

Exercice 50 Table de multiplication

Proposez un algorithme qui demande à l'utilisateur d'entrer un nombre entier strictement supérieur à zéro et inférieur ou égal à 12. L'ordinateur affichera ensuite la table de multiplication du nombre entier. Par exemple si l'utilisateur entre 3, l'ordinateur affichera

3 x 1 =3

3 x 2 =6

3 x 3 =9

3 x 12 =36

Exercice 51 : Amortissement d'un Matériel (BTS EPP Partie pratique)

En admettant que l'amortissement d'un bien se calcule par la méthode linéaire dont la formule est la suivante :

Dépréciation Annuelle = Valeur Initiale / Durée de vie

Nouvelle Valeur Après un an = Valeur Ancienne – Dépréciation Annuelle

Ecrire l'algorithme qui calcule et affiche :

Les années successives, les dépréciations et les nouvelles valeurs d'un matériel connaissant sa valeur initiale, sa durée de vie et l'année d'acquisition.

Exercice 52 VOYELLES

Ecrire un algorithme qui lit une suite de caractères et qui comptabilise le nombre de fois où chacune des 6 voyelles y apparaît. Un format d'affichage peut être le suivant

Votre texte comporte

2 fois la lettre a

8 fois la lettre e

3 fois la lettre i

4 fois la lettre o

5 fois la lettre u

2 fois la lettre y

Exercice 53 Combien de mots ?

Ecrire un algorithme qui lit une phrase et affiche le nombre de mots contenus dans cette phrase.

La structure **pour** trouve toute son importance dans la conservation de différentes valeurs de la même variable donnant naissance à la notion de tableau qui constitue la dernière partie de ce support et aussi du programme de première année en Informatique de gestion. Avant d'aborder cette dernière partie très importante nous allons voir le chapitre concernant les sous-programmes.

CHAPITRE 6 : LES SOUS-PROGRAMMES

***Objectif :** Être capable d'écrire un algorithme pouvant être l'appelé ou l'appelant d'un autre*

***Connaissances Préalables :** Savoir écrire un algorithme faisant intervenir des structures conditionnelles et/ou des structures de boucles*

INTRODUCTION

Pourquoi un sous-programme ? A quoi sert-il ?

N'est-ce pas déjà compliqué d'écrire des programmes ? Pourquoi ajouter encore des sous-programmes ? Rassurer vous ce n'est pas dans l'intention de vous compliquer encore plus les choses mais pour vous permettre d'écrire moins. En effet les programmes écrits jusqu'à présent sont plus ou moins longs et la même action pouvait être écrite plusieurs fois, c'est le cas par exemple de l'algorithme qui saisit deux nombres et permet à l'utilisateur de choisir d'exécuter l'une ou l'autre des actions suivantes :

Le produit des deux nombres

La somme des deux nombres

La moyenne des deux nombres

Dans cet algorithme, une instruction permet d'effectuer la somme. La même somme sera utilisée pour déterminer la moyenne.

Un autre exemple est celui de la saisie de réponse par **oui** ou **non** (et le contrôle ou la contrainte qu'elle implique) qui peuvent être répétés plusieurs fois dans le même algorithme. C'est ainsi qu'on peut être amené dans un algorithme à écrire un ensemble d'instructions dont la présence est nécessaire à plusieurs endroits. Le premier avantage des sous-programmes est de permettre au programmeur d'écrire cet ensemble d'instructions une seule fois et de pouvoir l'appeler à tout autre endroit de l'algorithme. Par voie de

conséquence le nombre de lignes du programme est réduit entraînant un deuxième avantage des sous-programmes qui est la meilleure lisibilité du programme.

1. Définition

Un sous-programme est une action d'un programme, autonome de ce programme, réalisant une tâche précise et pouvant être appelée à tout moment et à n'importe quel endroit de ce programme. De ce fait ce sous-programme sera dit l'appelé alors que le programme principal sera dit l'appelant. Ces sous-programmes se présentent sous forme de **procédure** et **fonction**

2. Procédure

a) Définition

Une procédure est une action d'un programme située à l'intérieur de celui-ci et obéissant à des règles précises.

On peut encore la définir comme un bloc d'instructions nommé et déclaré dans l'entête de l'algorithme et appelé dans le corps de ce dernier chaque fois que le programmeur en a besoin.

b) Déclaration ou présentation

La présentation d'une procédure ressemble à celle d'un programme et constitue la déclaration de cette procédure

Procédure Nom_procédure (paramètres ou arguments : type)

.....

.....

Début

Fin Procédure

Interprétation :

- **Nom_procédure** est le nom par lequel nous désignons notre procédure
- **Paramètre** constitue un ensemble d'arguments avec leur type.

c) Variables globales et locales

Avec l'utilisation des procédures et fonctions, les variables peuvent être classées en deux (2) grandes catégories en fonction de l'endroit où elles ont été déclarées. On a ainsi les variables dites globales et les variables dites locales.

Une variable globale est une variable déclarée dans le programme principal et peut être utilisée dans toutes les procédures du programme sans être à nouveau déclarée

Une variable locale est une variable propre à une procédure ; lorsqu'on en a besoin dans une autre procédure, il faut à nouveau la déclarer. Une telle variable est créée avec et disparaît avec la procédure d'où le nom de local.

d) Appel de la procédure

Comment peut-on appeler une personne ? A quoi sert de donner un nom à une personne, à une variable ou encore à une procédure ? Vous imaginez déjà qu'on appellera une procédure par son nom.

Syntaxe : nom procédure (liste des paramètres)

e) Applications

❖ Application 1

Nous allons considérer l'exemple pris en introduction en permettant à l'utilisateur après avoir entré 2 nombres de choisir l'exécution de l'une ou l'autre des opérations suivantes :

Le produit des deux nombres

La somme des deux nombres

La moyenne des deux nombres

Résolution

Algorithme principal

* Déclaration des variables globales

VAR x,y : réel

choix : entier

* Déclaration de la procédure Somme

Procédure Somme

Var S : réel

Début

S ← x+y

Fin

* Déclaration de la procédure Moyenne

Procédure Moyenne

Var M : réel

Début

Somme

M ← S/10

AFFICHER ("La moyenne des nombres saisis est : ", M)

Fin

* Déclaration de la procédure Produit

Procédure Produit

Var P : réel

Début

P ← x*y

AFFICHER (" LE produit des nombres saisis est : ", P)

Fin

DEBUT (* corps du programme principal*)

(*Saisie des nombres*)

AFFICHER ("Entrer le premier nombre ")

SAISIR (x)
AFFICHER ("Entrer le deuxième nombre ")
SAISIR (y)

(*Affichage du menu*)
AFFICHER ("Tapez 1 pour afficher la somme")
AFFICHER ("Tapez 2 pour afficher la Moyenne")
AFFICHER ("Tapez 3 pour afficher le produit")
AFFICHER ("Faites votre choix")
SAISIR (choix)
SELON choix de
 1 : somme
 AFFICHER ("La somme est : ", S)
 2 : Moyenne
 3 : Produit
FinSuivant
Fin

Fonctionnement de l'ensemble

A l'intérieur du programme principal figurent un ensemble de noms ou termes écrits en un seul mot et soulignés : Ce sont des procédures. Dans notre cas ici lorsque l'utilisateur tape 3, le curseur ira rechercher la procédure ou l'action nommée produit et va l'exécuter. Lorsque l'exécution s'achève le curseur ne continue pas sur la ligne qui suit **Fin** de ladite procédure mais repart dans le programme principal (et ça vous êtes censé le savoir pour avoir étudié la structure de choix) et puisqu'il n'y a plus autre instruction après produit, le curseur se retrouvera après FinSuivant. Par contre dans le cas de la somme, le curseur va ensuite exécuter l'instruction d'affichage qui suit Somme avant de revenir à FinSuivant.

On peut donc dire que les sous-programmes constituent une façon de décomposer un problème en différentes parties.

L'importance des procédures est plus ressentie lorsqu'il s'agit d'écrire une application. En effet plusieurs programmes seront écrits et dans le même programme on est très souvent amené à exécuter la même tâche plusieurs fois.

Si nous prenons l'exemple d'un programme de mise à jour des étudiants (bien entendu vous n'avez pas encore fait le cours sur les fichiers mais vous avez une idée de ce qu'on appelle mise à jour et si ce n'est pas le cas tâchez d'être présent au cours), l'utilisateur souhaiterait qu'après un clic sur un bouton nommé '**nouveau**', que les anciennes saisies s'effacent pour qu'il puisse entrer de nouvelles ; il en est de même après un clic sur les boutons '**Enregistrer**' , '**Supprimer**'. En clair la même action (mettre les zones de saisies à blanc) peut constituer la fin de chacune de ces trois actions ou procédures d'un même programme. Plutôt que de l'écrire trois (3) fois, on pourra écrire une procédure '**Effacezone**' qu'on appellera à chacun de ces endroits.

❖ Application 2

A vous de jouer cette fois-ci. Il s'agit de stocker dans un tableau 10 nombres et ensuite l'ordinateur demandera à l'utilisateur de choisir l'une ou l'autre des opérations suivantes :

Le minimum des nombres

La somme des nombres

Le produit des nombres

La moyenne des nombres

On vous demande de résoudre ce problème de deux manières :

- Sans les procédures
- Avec les procédures

Vous pouvez traiter l'exercice maintenant si vous avez déjà compris ce qui n'est pas encore expliqué (les tableaux) sinon vous patientez et vous traiterez le sujet quand on sera sur ce chapitre. Nous proposons cet exercice pour dire que les notions de procédure et fonction peuvent être utilisées dans toutes les parties du cours en particulier dans la résolution des problèmes relatifs aux tableaux et fichiers.

3. Fonction

a) Présentation et définition

Dans l'écriture d'un programme certains résultats sont très complexes à obtenir à travers les instructions telles que : écrire un programme qui saisit un mot et qui affiche le nombre de caractères de ce mot ou encore la position où se trouve un (1) caractère dans une chaîne de caractères ou encore le sinus ou cosinus d'un angle. C'est pour cette raison que les calculatrices comportent des fonctions. Il en est de même pour tous les langages. Ces fonctions prédéfinies ont donc essentiellement pour rôle d'alléger le travail du programmeur en lui épargnant de longs et pénibles algorithmes.

Par définition une fonction est un bloc d'instructions qui retourne obligatoirement une et une seule valeur résultat à l'algorithme appelant. cette valeur pourra être affichée ou exploitée dans une instruction

b) Déclaration

Fonction Nom_fonction (arguments : type) : type

Var /* ici déclaration des variables locales

.....

.....

Début

Corps de la fonction

Renvoyer résultat

Fin Procédure

Les fonctions sont ainsi comme les procédures, constituées de trois (3) parties :

- Le nom qui est un mot réservé. Ne cherchez donc pas à inventer un mot, le langage ne l'acceptera pas.
- Deux (2) parenthèses, une ouvrante, une fermante
- Les arguments ou paramètres c'est à dire une liste de valeurs

pour l'exécution de la fonction. Le nombre de ces arguments peut être un(1), deux(2), etc. ou encore zéro(0) et dans ce dernier cas on n'aura rien dans la parenthèse. Ces contraintes ne dépendent pas du programmeur mais du langage de programmation.

Ainsi pour une fonction comme sinus qui requière un argument en l'occurrence la valeur de l'angle, si on lui donne deux (2) ou aucun le langage affichera une erreur à l'exécution.

c) Exemples

Nous présentons ici quelques fonctions prédéfinies, bien entendu ce n'est pas toutes ces fonctions que vous aurez à utiliser dans vos algorithmes et probablement certaines ne seront jamais utilisées jusqu'à ce que vous abandonniez l'informatique ou qu'elle vous abandonne.

Fonction	Rôle	
Abs(x)	Valeur absolue de x	
Frac (x)	partie décimale de x	
Exp (x)	valeur exponentielle de x	
Ent (x)	partie entière de x	
Round(x)	Arrondi x à l'entier le plus proche	
Sqrt (x)	Racine carré de x	
sqr(x)	carré de x	
Chr(x)	caractère dont le code ASCII est X	
Uppcase (Caractère1)	le caractère majuscule du caractère 1	
Modulo(x,y)	Reste de division de X par Y	
Chr (code ASCII)	Le caractère du code ASCII	
Sschaine(chaîne,c1,c2)	Extraction ou Mid(cc,c1,c2)	
Longueur(cc) ou len	Nombre de caractères de cc	
Rang(chaîne,chaîne2,c)	Où se trouve chaîne2 dans chaîne à partir de c caractère	
Pred(arg)	Prédécesseur de (arg)	
Succ(arg)	Successeur de arg	
Arctan(x)	Tangente inverse de x	
Cos(x)	Cosinus de x	
Sin(x)	Sinus de x	
Ln(x)	Logarithme népérien de x	
Left(cc,nb)	Les nb caractères à gauche	
Right(cc,nb)	Les nb caractères à droite	
Trouve(cc,chaîne)	Où se trouve Chaîne dans cc	

CHAPITRE 7 : LES TABLEAUX

Objectif: *Etre capable de stocker des valeurs dans un tableau et de manipuler ce tableau en effectuant des opérations allant des plus simples (somme, produit, moyenne) aux plus complexes (Classement).*

Connaissances Préalables: *Savoir utiliser la structure de boucle en particulier la structure Pour*

INTRODUCTION

Pourquoi un tableau ? A quoi sert un tableau ?

Toutes les variables que nous avons utilisées jusqu'à présent permettaient de stocker toutes les valeurs pouvant être prises par ces variables ; mais à un instant donné, une variable ne pouvait contenir qu'une seule valeur c'est-à-dire que la valeur n vient écraser, effacer la valeur $n-1$. Dans le cas de la saisie de 10 noms par exemple le 2^{ème} nom saisi écrase le 1^{er}, le 3^{ème} écrase le 2^{ème} si bien qu'au 10^{ème} nom on n'a plus la valeur des 9 premiers noms saisis tout simplement parce que pour toutes les saisies on n'utilise qu'une seule zone que nous schématisons généralement par

Le tableau vient corriger cette insuffisance en créant plusieurs zones ou cases contiguës, chacune contenant une valeur de la même variable.

--	--	--	--	--	--	--	--	--	--

I. ENVIRONNEMENT TABLEAU

1. Définition d'un tableau

Un tableau est une structure linéaire de données de même nature ou type. Cela ne veut peut être rien dire pour vous alors retenez ceci :

Une variable pouvant contenir à un même instant un ensemble de valeurs chacune occupant une position ou une case(1, 2 ,3.....N) est appelée tableau ou encore variable indicée. Chaque valeur est alors repérée par sa position encore appelée son indice et allant de 0 à N ou encore de 1 à N où N indique le nombre total de cases.

2. Exemple et Notation

12	2	4	15	8	6	12	2	6	17
1	2	3	4	5	6	7	8	9	10

Ici le tableau a 10 valeurs donc 10 indices ou positions

La valeur se trouvant à l'indice 1 est 12

La valeur se trouvant à l'indice 2 est 2

La valeur se trouvant à l'indice 10 est 17

Puisque ce sont des nombres qui sont dans ce tableau on peut dire qu'il s'agit d'une variable Note ou Moyenne ou âge et un des noms qu'on peut donner à ce tableau serait TABNOTE ou (TABMoy) ou (Tage)

3. Déclaration d'un tableau

Dans ce qui précède nous devons comprendre qu'un tableau est avant tout une variable et par conséquent doit être déclaré. La différence avec les variables précédentes est que le tableau contient plusieurs cases ; on est donc amené à donner le nombre maximum de cases dans le tableau. Par ailleurs les différentes valeurs susceptibles d'être placées dans le tableau indiqueront le type du tableau.

Partant de ces 2 éléments le format général de la déclaration d'un tableau devient alors

VAR *NOMTABLEAU* : TABLEAU [*Imin* .. *IMax*] DE TYPE

4. Interpretation

NOMTABLEAU : c'est le nom qu'on donne au tableau.

Exemple TABNOM, TABNOTE, TABMoy ou encore tout simplement NOM, NOTE, Moy.

Imin et *IMax* indiquent respectivement le nombre minimum et maximum de cases dans le tableau.

Type : indique le type du tableau (entier ; réel ; chaîne de caractère ; etc.)

Une autre forme de déclaration vient du fait que l'indice minimum a généralement la valeur 1 et c'est l'indice maximum qui varie et qui par conséquence doit être nécessairement précisé.

Entraînement :

- Déclarer un tableau contenant 30 noms
- Déclarer un tableau contenant l'âge de 20 employés
- Déclarer un tableau contenant la moyenne de chacun des 30 étudiants d'une classe

ATTENTION ARRETEZ-VOUS ICI SINON VOUS ALLEZ
DECOUVRIR LA SOLUTION

- VAR TABNOM : Tableau[1..30] de CC
- VAR TABAGE : Tableau[1 ..20] d'entier
- VAR TABMoy : Tableau [1.. 30] de réel

Ces différentes déclarations montrent qu'on connaît le nombre maximum de cases du tableau ; or il arrive fréquemment que ce nombre ne soit pas connu c'est-à-dire désigné par une variable n ou nb ; on parle alors de tableau **dynamique**. Comment pensez-vous qu'on peut déclarer un tel tableau dont le nombre maximal de case est nb ? Réfléchissez.

II. Opérations sur les tableaux

Elles peuvent être subdivisées en deux grands groupes qui sont :

A. Opérations de base

Ce sont celles déjà rencontrées avec les variables simples et qui permettent dans un premier temps d'introduire des valeurs dans le tableau (opération de création ou de remplissage ou de renseignement du tableau) ; sans elle aucune autre opération ne serait possible sur les tableaux. Dans un deuxième temps, il s'agit de pouvoir afficher (édition ou affichage du tableau) les valeurs contenues dans le tableau ou une partie du tableau.

Dans ces 2 cas comme dans les autres opérations que nous verrons, il s'agira de faire l'opération pour plusieurs cases du tableau d'où la notion effective de boucle.

1. Création ou remplissage d'un tableau

Le format général est le suivant :

Algorithme

Pour *Indice* allant de *Imin* à *IMax* faire

SAISIR (NOMTABLEAU [Indice])

Finpour

Exercice d'application : Nous voulons remplir un tableau de noms contenant 30 cases.

PROGRAMME TABLEAUNOM

VAR TABNOM : Tableau[1..30] de CC

I : entier

Début

Pour i allant de 1 à 30 faire

AFFICHER ("Entrer le nom numéro", i)

SAISIR (TABNOM [i])

Fpour

Fin

2. Interprétation :

Pour i allant de 1 à 30 (peut encore s'écrire : Pour i ← 1 à 30) signifie que l'indice (i) va prendre successivement la valeur 1, 2, 3 jusqu'à 30 et pour chacune de ces valeurs l'instruction saisir (TABNOM[i]) va permettre de saisir un nom et ce nom sera stocké dans la case correspondant à i.

Ainsi au 1^{er} tour i = 1, le nom saisi sera conservé dans la case N°1

Au 2^{ème} tour, i=2, le nom saisi sera conservé dans la case N°2 jusqu'à ce que la 30^{ème} case soit remplie

i est une variable et doit être forcément déclarée comme un entier. On préfère souvent i (pour Indice) mais toute autre lettre ou ensemble de lettre (compteur) est aussi acceptable.

3. Affichage ou édition d'un tableau

Le format de l'affichage est le même que celui du remplissage à la différence qu'il faut remplacer l'instruction de création (**Saisir**) par l'instruction d'affichage (**Afficher**)

Pour Indice allant de Imin à IMax faire

Afficher (NOMTABLEAU [Indice])

Fpour

Exercice 54

Ecrivons un algorithme qui lit 30 noms et affiche les noms ainsi lus

(**Voir cours**)

B. Manipulation de tableaux

1. Somme et moyenne des éléments d'un tableau .

Il s'agit d'additionner (de faire un cumul) de toutes les valeurs contenues dans un tableau ; et pour avoir traité déjà certains exercices sur les boucles vous savez qu'en algorithme on n'écrit pas $\text{somme} = \text{val1} + \text{val2} + \text{val3} + \dots + \text{valn}$ (sinon on déclarera val1, val2, val3 ... valn), mais tout simplement $\text{somme} = \text{somme} + \text{val}$; ainsi un seul élément (val) est déclaré.

Exercice 55

Ecrire un algorithme qui stocke dans un tableau les moyennes des 30 étudiants d'une classe et qui affiche la moyenne de ces moyennes

2. Produit des éléments (valeurs) d'un tableau

Le principe est le même que celui de la somme à la différence que tout nombre multiplié par zéro (0) donne 0. Pour éviter donc que le produit (le résultat) soit zéro. Il faut, après avoir déclaré une variable ProduitMoy (produit

des moyennes) et avant de rentrer dans la boucle affecter la valeur 1 à ProduitMoy.

Reprendre l'exercice 49 mais cette fois en affichant en plus de la moyenne le produit des moyennes saisies.

3. Fusion

L'opération consiste à obtenir un tableau à partir de deux autres et peut se présenter sous différentes formes. Il peut s'agir :

- De la somme des éléments de deux tableaux

Tab1	2	8	5	12	6
Tab2	-2	4	8	12	5
Tabresultat	0	12	13	24	11

- De la fusion de 2 tableaux ordonnés avec conservation des doublons.

Tab1

HERMAN	MARTIN	SOLANGE	SOW
--------	--------	---------	-----

Tab2

ANGBO	CHRISTIAN	JOSEPHINE	RAOUL
-------	-----------	-----------	-------

Tabresult

ANGBO	CHRISTIAN	HERMAN	JOSEPHINE	MARTIN	RAOUL	SOLANGE	SOW
-------	-----------	--------	-----------	--------	-------	---------	-----

Ou encore avec un tableau d'entier

2	8	5	12	6	Tab1					
-2	4	8	12	7	Tab2					
-2	2	4	5	6	7	8	8	12	12	Tabresult

- De la fusion de 2 tableaux ordonnés sans doublons

2	8	5	12	6	Tab1			
-2	4	8	12	7	Tab2			
-2	2	4	5	6	7	8	12	Tabresult

4. Eclatement

C'est le contraire de la fusion. A partir d'un tableau il faut obtenir deux autres tableaux. Les critères de séparation peuvent être :

- Tableau des nombres Pairs et Impairs
- Tableau des nombres positifs et négatifs

5. Ajout ou insertion d'un élément

Il s'agit ici d'ajouter un élément à un tableau : Cela nécessite donc que le tableau ait une case vide et de surcroît si le tableau est trié il faudra insérer l'élément au bon endroit.

6. Suppression d'un élément

Il s'agit ici d'affecter du vide à une case et de passer à une réorganisation du tableau de telle sorte que si l'élément à l'indice i est supprimé, alors l'élément à l'indice $i+1$ prend sa place ainsi de suite de telle sorte que c'est la dernière case qui reste vide.

7. Recherche d'un élément particulier dans un tableau (le plus petit ; le plus grand ; une valeur quelconque etc.

Pour la recherche d'un élément quelconque, il n'y a aucune difficulté : Il suffit de passer sur chacune des cases et vérifier si le contenu de la case est le même que l'élément recherché. En ce qui concerne le plus grand ou plus petit, ce n'est pas non plus sorcier vu tout ce que vous avez déjà fait avant d'arriver ici. Souvenez-vous seulement qu'il faut déclarer une zone **MAX** (ou **MIN** ou les 2 en fonction du sujet). Dans ces cases vous placez le premier élément du tableau, ensuite il faudra passer sur chacune des autres cases à partir de la 2^{ème} jusqu'à la dernière et à chaque fois que l'élément rencontré sera supérieur à **MAX**, c'est lui qui devient MAX donc une affectation sera effectuée... Vous connaissez la suite

8. Tri de Tableau

Il existe plusieurs méthodes de tri, mais dans tous les cas le tri va consister à présenter les éléments d'un tableau dans un ordre précis (croissant ou décroissant) que les éléments soient de type entier, réel ou chaîne de caractères

Exercices

Exercice 56

Etant donné un tableau à 10 éléments réels, affichez

- La moyenne du tableau
- La valeur la plus petite du tableau
- Le nombre d'éléments ayant une valeur supérieure à la moyenne

Exercice 57

Etant donné un tableau à N éléments réels, écrire l'algorithme permettant d'afficher l'un ou l'autre des résultats suivants en fonction du choix effectué par l'utilisateur (faire en sorte que les libellés apparaissent clairement).

- La moyenne du tableau
- La valeur la plus petite
- Le nombre de notes supérieures ou égales à la moyenne et le pourcentage correspondant.

Exercice 58

Ecrire un algorithme qui stocke dans un tableau un certain nombre de noms. Ensuite l'utilisateur pourra saisir un nom quelconque et la machine devra afficher le nombre de fois que ce nom existe dans le tableau. Cette opération de recherche doit pouvoir se répéter jusqu'à ce que l'utilisateur tape le mot «FIN» à la place du nom pour arrêter.

Exercice 59

Un mot palindrome est un mot dont son miroir et lui n'en font qu'un.
Exemple : *KAYAK* = *KAYAK* et *ESSE* = *ESSE*, etc.

Ecrire l'algorithme qui permet de dire si un mot donné est un palindrome c'est-à-dire identique à son miroir.

Exercice 60

Soit un tableau de dimension N, proposez un algorithme qui permet à l'utilisateur de remplir le tableau et d'afficher les valeurs saisies. L'algorithme

devra permettre à l'utilisateur de choisir l'exécution de l'une ou l'autre des opérations suivantes :

- Recherche d'une valeur précise
- Recherche d'une valeur avec affichage de l'indice
- Recherche d'une valeur avec affichage des indices
- Inversion du tableau et avec affichage du tableau inversé
- Tri du tableau avec choix et affichage du tableau trié
- Suppression d'une valeur à l'indice I

N.B Chacune de ces opérations constituera une procédure

Exercice 61 Eclatement d'un tableau

Ecrire un algorithme qui permet d'éclater un tableau à N éléments en deux autres tableaux. Les deux tableaux seront :

-Soit tableau à valeurs positives et tableau à valeurs négatives

-Soit tableau des nombres pairs et tableau des nombres impairs

L'algorithme sera écrit en utilisant ou non la notion de procédure mais dans tous les cas il revient à l'utilisateur de choisir le type d'éclatement souhaité.

Exercice 62 Tri d'un tableau et vérification du contenu

Ecrire un algorithme qui remplit un tableau à N éléments réels. Ensuite le tableau sera trié dans l'ordre croissant et l'ordinateur devra afficher le tableau trié et dire si les éléments triés sont consécutifs.

III. Tableaux à deux dimensions ou « Matrice »

1. Définitions

Un tableau à deux dimensions A est à interpréter comme un tableau (unidimensionnel) de dimension L dont chaque composante est un tableau (unidimensionnel) de dimension C.

On appelle L le nombre de ligne du tableau et C le nombre de colonne du tableau. L et C sont alors les deux dimensions du tableau. Un tableau à deux dimensions contient donc $L \times C$ composantes.

On dit qu'un tableau à deux dimensions est carré, si L est égal à C.

En faisant le rapprochement avec les mathématiques, on peut dire que "A est un vecteur de L vecteurs de dimension C", ou mieux : "A est une matrice de dimensions L et C".

Plus simplement on parlera d'un tableau à deux dimensions lorsque chacune des valeurs du tableau ne peut être repérée qu'en précisant deux coordonnées

Considérons un tableau contenant les 5 NOTES de chacun des 4 élèves d'une classe :

Les notes					
Elèves	1	2	3	4	5
1	15	14	9	10	18
2	19	14	10	19	15
3	15	14	14	10	12
4	10	10	12	14	14

Sur une ligne nous retrouvons les 5 notes d'un élève et dans une colonne la note obtenue par chacun des 4 élèves pour un devoir donné.

2. Déclaration

Syntaxe

Tableau <NomTabl>[<DimLigne>,<DimCol>] : <Type>

Exemple sur notre tableau de note

VAR Tnotes : tableau[1..4,1..5] de type réel

3. Saisie des éléments d'un tableau à deux dimensions

Avec Pour	Avec TANQUE
Début POUR I←1 A N FAIRE POUR J← 1 A M FAIRE SAISIR (TNOTES [I,J]) FPOUR FPOUR Fin	Début I ← 1 TANQUE (I < N) FAIRE J ← 1 TANTQUE (J<M) FAIRE SAISIR (TNOTES [I,J]) J ← J + 1 FTQ I ← I + 1 FTQ Fin

Remarque 1: Utilisez de préférence la structure POUR à la structure TANT QUE en ce qui concerne les tableaux

Remarque 2 : N et M représentent respectivement 4 et 5 conformément au tableau déclaré ci-dessous

4. Comment reconnaître un tableau à deux dimensions

A partir du tableau précédent nous pouvons encore conseiller de considérer les éléments manipulés (saisie et/ou affichage) et les structures de boucle à utiliser. Si dans cette analyse nous constatons qu'on aura au moins deux boucles dont l'une est à l'intérieur de l'autre et que les valeurs concernant la même variable doivent être disponibles à un même instant T, nous pouvons penser à un tableau à deux dimensions.

Exercices

Exercice 63

Ecrivez un algorithme de remplissage d'un tableau de 6 sur 13, avec des zéros.

Exercice 64

Quel résultat produira l'algorithme ci-dessous ?

Var

X : tableau[1..2,1..3] de type entier

I, j, val : Entier

Début

Val \leftarrow 1

Pour i \leftarrow 1 a 2 faire

 Pour j \leftarrow 1 a 3 faire

 X [i,j] \leftarrow Val

 Val \leftarrow Val + 1

 Fpour

Fpour

Pour I \leftarrow 1 a 2 faire

 Pour j \leftarrow 1 a 3 faire

 Afficher (X [i,j])

 Fpour

Fpour

Fin

<u>Exercice 65</u> Quel résultat produira cet algorithme ?	<u>Exercice 66</u> Quel résultat produira cet algorithme ?
Var X: tableau [1..2,1..3] de type] : Entier I, j, val : Entier Début Val ← 1 Pour i ← 1 à 2 faire Pour j ← 1 à 3 faire X [i, j] ← val Val ← val + 1 Fpour Fpour Pour j ← 1 à 3 faire Pour i ← 0 à 1 faire Afficher (X [i, j]) Fpour Fpour Fin	Var T : tableau [1..3,1..2] de type Entier K, m : Entier Début Pour k ← 1 à 3 faire Pour m ← 1 à 2 faire T [k, m] ← k+m Fpour Fpour Pour k ← 1 à 3 faire Pour m ← 1 à 2 faire Afficher (T [k, m]) Fpour Fpour Fin

Exercice 67

Soit un tableau T a deux dimensions (12,8) préalablement rempli de valeurs numériques.

Ecrire un algorithme qui recherche la plus grande valeur au sein de ce tableau.

Exercice 68

Soit un tableau de N lignes et de M colonnes. Ecrire un programme qui permet de le remplir avec des valeurs saisies au clavier et qui :

- Affiche le tableau rempli
- Calcule et affiche la somme de tous les éléments du tableau
- La somme sur chaque ligne
- La somme sur chaque colonne

CHAPITRE 8 : LES ENREGISTREMENTS ET FICHIERS

Objectif: A partir de la structure d'un enregistrement ou encore d'un fichier, l'étudiant doit être capable d'écrire l'algorithme permettant :

- ✓ De mettre à jour le fichier (Créer, modifier, supprimer)
- ✓ De consulter les informations contenues dans le fichier (consultation non sélective)
- ✓ De consulter des informations obéissant à un même critère de sélection (Consultation Sélective).

Connaissances Préalables: Savoir utiliser la structure de boucle et la structure décisionnelle

INTRODUCTION

Pourquoi encore les fichiers ?

Comme si tout n'était déjà pas compliqué !!!

Les structures de données telles que les tableaux utilisés précédemment permettaient de stocker les données manipulées dans la mémoire centrale c'est à dire dans une zone volatile de la machine. En mémoire, la durée de vie d'une variable est égale à celle d'exécution du programme. Nous comprenons facilement que cette solution n'est pas envisageable pour les informations d'une entreprise. En effet les données concernant un employé, un produit, un fournisseur ou un contrat etc. doivent être mémorisées plus longtemps que la durée d'exécution d'un programme. Le seul moyen connu en informatique pour mémoriser sur une longue période ces données est de recourir aux fichiers.

I. Rappel sur les fichiers

1. Définition

Les fichiers sont des espaces logiques servant à stocker des informations de manière permanente, ils utilisent comme support les mémoires de masse ou mémoires auxiliaires telles que les Disques Durs, les clés USB, les Disquettes, les Bandes magnétiques, les CD, etc.

2. Les catégories de fichiers

Il existe deux grandes catégories de fichiers

➤ Ceux qui sont disposés sous forme de lignes successives. Ces lignes, malgré leur spécificité au niveau des valeurs, ont la même structure c'est-à-dire que si la ligne N contient dans l'ordre les informations Numéro, Nom, Prénom et âge, il en sera de même pour toutes les autres lignes avant ou après la ligne N. Ces différentes lignes sont alors appelées **ENREGISTREMENT**.

Exemple : CARNET D'ADRESSE

ANNUAIRE TELEPHONIQUE

COORDONNEES DES ETUDIANTS

Ce type de fichier est utilisé lorsqu'on doit stocker des informations assimilables à une Base de Données et est appelé **FICHIER TEXTE**.

➤ Ceux qui ne sont pas disposés sous forme de lignes. Ici les octets sont écrits les uns à la suite des autres. Ils sont appelés **FICHIER BINAIRE**.

Un exemple est ce cours que vous êtes entrain de lire ou encore un fichier son, un fichier image, un programme écrit, un programme exécutable, etc. etc.

Une autre différence entre les 2 fichiers est que dans le fichier texte les données sont écrites sous forme de (vous ne devinez jamais !!!) texte (oui c'est cela), en fait cela veut dire qu'en les visualisant avec un éditeur de texte, on comprend les informations affichées alors que dans le cas de fichier binaire il y a un codage non compréhensible par l'utilisateur.

Dans le cadre de ce cours, nous nous intéresserons évidemment au fichier disposé sous forme d'enregistrement. Mais là encore nous pouvons rencontrer 2 types de représentation ou structure

Dans l'un les champs sont les uns à la suite des autres, la fin d'un enregistrement marque le début d'un autre.

Exemple: "YAO"; "KONAN"; "07-60-60-60" "BARTHELEMY";
"OLOMIDE"; "07-07-60-50"

Ce type est dit de structure **délimitée**

L'avantage principal : gain d'espace mémoire (espace mémoire non perdu).

L'inconvénient : Lenteur dans la recherche d'une information.

Dans le 2^{ème} type, les valeurs des champs sont disposées de façon lisible, claire

Exemple: YAO KONAN 07-60-60-60
 BARTHELEMY OLOMIDE 07-60-70-50

On parle de structure à champ de **largeur fixe**. L'inconvénient ici est la perte d'espace. (Si vous n'avez pas compris soyez au cours). C'est ce qui explique qu'au début de l'informatique et surtout au moment où les mémoires de stockages étaient très coûteuses, ce type de structure n'était pas trop utilisé. Par contre, depuis des dizaines d'années elle demeure la structure la plus utilisée surtout avec l'avènement des bases de données.

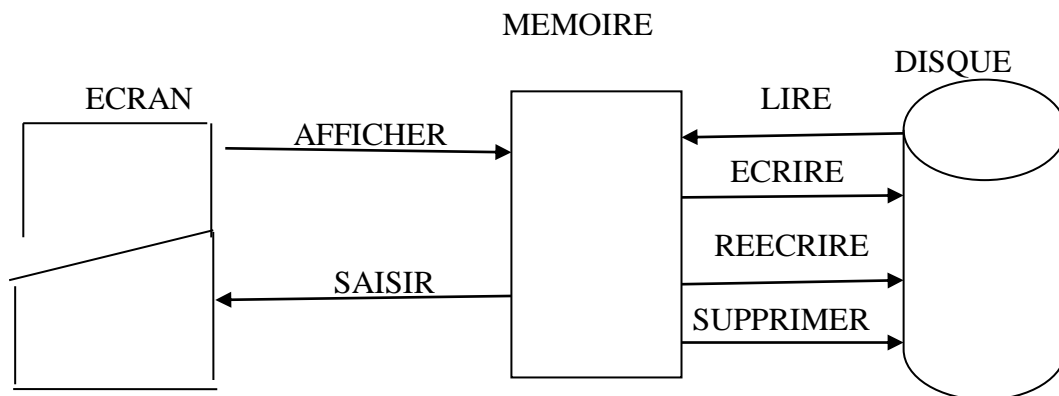
Oui c'est bien beau tout cela mais nous on fait quoi avec tout cela ?

Patiencez, je sais que ce qui vous intéresse c'est comment écrire votre algorithme lorsque les fichiers interviennent, mais commencez d'abord par faire quelques suppositions ou déductions qui découlent de tout ce que vous saviez en algorithme et de la définition de fichier qu'on vient de vous rappeler. Ah oui oui oui en algo je saisisais des données, j'affichais aussi d'autres données mais toutes disparaissaient parce qu'il n'y avait pas de fichier, donc maintenant qu'il y a fichier on va m'apprendre probablement comment faire pour garder (en informatique on dira **conserver, enregistrer**

ou stocker) ces données dans un fichier (**programme de création**) ou alors ces données qui sont déjà dans le fichier, comment faire pour les modifier (**programme de modification**) ou les supprimer (**programme de suppression**) ou tout simplement les regarder, les voir (en informatique on dit **consultation ou visualisation** pour ne pas confondre avec la consultation médicale !!!). En réalité ce que nous devons chercher à connaître ce sont les instructions qui vont permettre d'échanger des informations entre l'ordinateur (plus précisément la mémoire) et le support non volatile (généralement le disque dur). Ces instructions sont :

ECRIRE - REECRIRE - SUPPRIMER - LIRE.

3. Schématisation



Dans les parties qui vont suivre, après avoir étudié les instructions concernant les enregistrements et fichiers, nous verrons comment les utiliser pour écrire

- ✓ Les programmes de mise à jour (Création-Modification-Suppression)
- ✓ Les programmes de consultation

II. LES ENREGISTREMENTS

1. Définition

Un enregistrement est une structure de données qui regroupe un certain nombre d'informations de même type ou de types différents mais qualifiant le même objet.

Exemple :

Une personne peut être qualifiée par les informations nom, prénom, âge,...

Un courrier peut être caractérisée par date expédition, objet, type,....

2. Déclaration et syntaxe

La déclaration d'un enregistrement se fera de la façon suivante

Identificateur : ENREGISTREMENT

Champ 1 : type

Champ 2 : type

...

Champ N : type

FinEnregistrement

N.B : ce qui est en gras va dépendre de l'enregistrement à déclarer

Exemple de l'enregistrement étudiant qui contient nom, prénom, date naissance (voir schéma plus bas)

Etudiant = ENREGISTREMENT

Nom : chaîne de caractères

Prénom : chaîne de caractères

DateNais : ENREGISTREMENT

Jour : entier

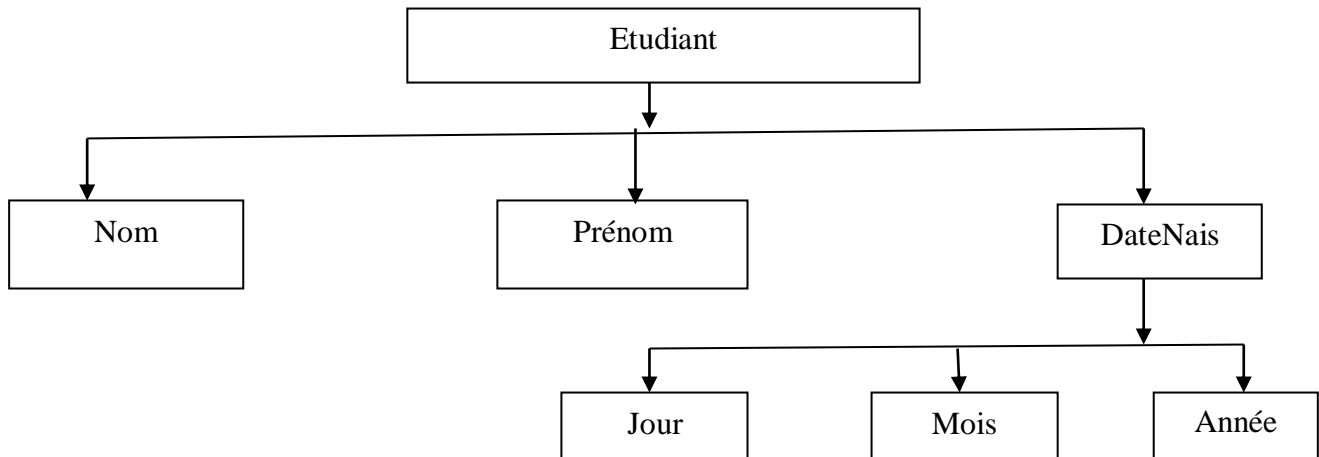
Mois : entier

Année : entier

FinEnregistrement

FinEnregistrement

Une telle déclaration permet de dire qu'un enregistrement peut en contenir un ou plusieurs autres qu'on peut appeler « Sous-Enregistrement ». L'enregistrement que nous venons de déclarer peut être représenté sous forme de structure hiérarchique suivante.



3. Access au champ d'un enregistrement

Pour accéder à un champ d'un enregistrement, il faut indiquer le nom du champ précédé du nom de l'enregistrement, les deux noms étant séparés par un point.

Identificateur. champ

Exemple :

Etudiant. Nom ← "KONATE"

Etudiant. prénom ← "pierre"

Etudiant. DateNais. Jour ← 07

Etudiant.DateNais.Mois ← 05

Etudiant.DateNais.annee ← 2007

Toutes les opérations classiques d'affectation, de comparaison, de calcul etc. sont aussi utilisables sur les données de type enregistrement.

En réalité un enregistrement n'a pas une existence propre, il n'existe qu'à l'intérieur d'un fichier, c'est ce qui explique qu'on ne peut déclarer un fichier sans déclarer l'enregistrement. Passons donc sur la partie concernant les deux.

III. LES INSTRUCTIONS SUR LES FICHIERS ET ENREGISTREMENTS

1. Sur les fichiers

a) Déclaration des fichiers

La syntaxe générale est la suivante

Fichier nom du fichier (type **organisation**)

Enregistrement nom de l'enregistrement

Rubrique 1 : type

Rubrique 2 : type

‘ ‘ ‘ ‘ ‘ ‘ ‘ ‘

‘ ‘ ‘ ‘ ‘ ‘ ‘ ‘2

Rubrique n : type

Fin enregistrement

Exemple : déclarer le fichier des lecteurs contenant :

MATRICULE 7 C AN

NOM 20 C AN

PRENOM 30 C AN

DATE ABONNEMENT 10 C AN

NOMBRE DE PRET accordé 1 C ENTIER

MAIL 80 C AN

TYPE ABONNEMENT 1 C A

b) Ouverture

Ouvrir nom du fichier (mode)

Le mode sera :

✓ L (Lecture) s' il s'agit d'un programme de consultation ou d'édition

- ✓ E (Ecriture) s'il s'agit d'un programme de saisie sans recherche
- ✓ L/E (Lecture/Ecriture) si il s'agit d'un programme de mise à jour c'est-à-dire Création, Modification ou Suppression

Exemple : ouvrir le fichier précédemment déclaré dans un programme :

- De modification
- De suppression
- De consultation

c) Fermeture

Elle est plus simple que l'ouverture en ce sens qu'elle ne dépend pas du type de programme à écrire

Fermer nom de fichier

Exemple : fermer Fichierlecteur

2. Sur les Enregistrements

a) Ecriture

Ecrire nom de l'enregistrement

b) Réécriture

Reécrire nom enregistrement

c) Suppression

Supprimer nom enregistrement

d) **Lecture** : qu'est ce que c'est ? En quoi cela consiste

Lire NomEnregistrement signifie en organisation séquentielle prendre un Enregistrement c'est-à-dire que le système à la lecture de cette instruction garde en mémoire un enregistrement, de ce fait toutes les valeurs de cet enregistrement sont disponibles et peuvent être utilisées dans des opérations (arithmétiques, de comparaison etc), en particulier la valeur du champ identifiant pourra être comparée à une valeur nouvellement saisie afin d'éviter des redondances. C'est cette instruction de lecture qui sera donc utilisée pour

rechercher l'existence ou non d'un enregistrement (Voir programmes de modification, suppression et consultation).

Cette procédure de recherche d'un enregistrement se présente ainsi

- Saisie de la valeur recherchée

AFFICHER: ''Entrez la Valeur recherchée''

SAISIR VALRECH

- Recherche

REPETER

LIRE NomEnregistrement

JUSQU'A VALRECH = ValeurdansFichier ou FinFichier

- Test de Vérification

SI VALRECH= ValeurdansFichier alors

AFFICHER ' Cette valeur est déjà saisie''

La suite dépend de l'objectif à atteindre (Voir cours)

Après avoir étudié brièvement les instructions ou commandes rencontrées dans les programmes à fichier, nous allons à présent écrire les différents programmes

IV. PROGRAMME DE MISE A JOUR EN ORGANISATION SEQUENTIELLE

1. Les étapes : Cas du programme de création

- Déclarer le fichier
- Déclarer les variables
- Ouvrir le fichier
- Boucle de traitement si elle existe
- Saisir les informations
- Valider les saisies
- Ecrire l'enregistrement sur disque
- Reprendre éventuellement le traitement

2. L'écriture effective des programmes

Le fichier que nous utiliserons dans toute cette partie est le fichier des écritures comptables contenant :

- Le numéro de l'écriture : 11 C N
- La date de l'écriture : 10 C AN
- Le montant : 11 C N
- Le code (débit ou crédit) : 1C A (D ou C)

a) Programme de création

Ecrire un algorithme permettant de saisir et d'enregistrer toutes les écritures comptables.

Résolution

Programme creEcriture

- **Déclaration fichier**

FICHIER = Fiécriture(Organisation séquentielle)

Enregistrement = écriture (ou encore Ecriture = Enregistrement)

Noécriture : numérique

Date écriture : chaîne de caractères

Montant : numérique

Code : caractère

Fin enregistrement

- **Déclaration variable**

Var rep : caractère

DEBUT

- **Ouverture fichier**

Ouvrir Fiécriture (E)

- **Corps du programme**

REPETER

AFFICHE: 'saisir les données de l'écriture'

```
SAISIR noecrit, dataécrit, montant, code
AFFICHE ‘’Enregistrer O/N ?’’
SAISIR rep
Si rep = ‘’O’’ Alors
    écrire écriture
Fsi
AFFICHER ‘’continuer O/N ?’’
SAISIR rep
Jusqu'à rep = ‘’N’’
```

- **Fermeture fichier**

```
FERMER Fiécriture
```

FIN

Cet algorithme de création que nous venons d'écrire va nous permettre d'enregistrer autant d'écriture que désire l'utilisateur c'est-à-dire qu'au fur et à mesure que l'utilisateur répond ‘’O’’ à la question ‘’enregistrer O/N ? Un enregistrement sera écrit dans le fichier FIECRIT conformément aux valeurs saisies et finalement le contenu de ce fichier FIECRIT pourra se présenter de la façon suivante

NOECRIT	DATECRIT	MONTANT	CODE
1	01/12/1999	1500 000	D
2	01/12/1999	1500 000	C
3	01/12/1999	250 000	D
4	02/12/1999	500 000	D

Il est impératif que le fichier conserve des informations avant de pouvoir exécuter ou écrire les autres programmes concernant les fichiers à savoir modification, suppression et édition. De tous ces programmes les plus importants sont la consultation et l'édition. Pourquoi ??? Voir cours.

Avant d'en arriver là nous allons proposer les autres programmes de mise à jour.

b) Programme de Modification

Programme ModEcriture

- **Déclaration fichier**

FICHIER = Fiécriture(Organisation séquentielle)

Enregistrement = écriture (ou encore Ecriture = Enregistrement)

Noécriture : numérique

Date écriture : chaîne de caractères

Montant : numérique

Code : caractère

Fin enregistrement

- **Déclaration variable**

Var rep : caractère

VALRECH : Numerique

DEBUT

- **Ouverture fichier**

Ouvrir Fiécriture (L/E)

- **Corps du programme**

REPETER

AFFICHER: "Entrez le Numéro de l'écriture"

SAISIR VALRECH

- **Recherche**

REPETER

LIRE Ecriture

JUSQU'A VALRECH = Noécriture ou FinFichier

- **Test de Vérification**

SI VALRECH= Noécriture alors

AFFICHER ' Cette valeur est déjà saisie'

- Affichage des autres champs de l'enregistrement
AFFICHER écriture .datecrit
AFFICHER écriture. montant
AFFICHER écriture. Code
- Modification eventuelle de l'une ou l'autre des valeurs
AFFICHE: ''Entrez les nouvelles valeurs ou Tapez Entrée''
SAISIR écriture.dataécrit,écriture. montant, écriture.code
- Enregistrement des nouvelles saisies
AFFICHER ''Enregistrer O/N ?''
SAISIR rep
Si rep = ''O'' Alors
 REécrire écriture
Fsi
SINON
 AFFICHER ('' Cette valeur n'existe pas, Modification impossible)
FINSI
- Question pour passer éventuellement à une autre modification
AFFICHER ''Autre Modification O/N ?''
SAISIR rep
JUSQU'A REP = ''N''
- **Fermeture fichier**
FERMER Fiécriture

FIN

c) Programme de suppression

Programme SupEcriture

- **Déclaration fichier**

FICHER = Fiécriture(Organisation séquentielle)

Enregistrement = écriture (ou encore Ecriture = Enregistrement)

Noécriture : numérique

Date écriture : chaîne de caractères

Montant : numérique

Code : caractère

Fin enregistrement

- **Déclaration variable**

Var rep : caractère

VALRECH : Numerique

DEBUT

- **Ouverture fichier**

Ouvrir Fiécriture (L/E)

- **Corps du programme**

REPETER

AFFICHER: "Entrez le Numéro de l'écriture"

SAISIR VALRECH

- **Recherche**

REPETER

LIRE Ecriture

JUSQU'A VALRECH = Noécriture ou FinFichier

- **Test de Vérification**

SI VALRECH= Noécriture alors

AFFICHER "Cette valeur est déjà saisie"

- **Affichage des autres champs de l'enregistrement**

AFFICHER écriture .datecrit

AFFICHER écriture. montant

AFFICHER écriture. Code

- Question pour supprimer ou non l'Enregistrement.

Contrairement à la modification la suppression ne concerne pas l'une ou l'autre des valeurs de l'enregistrement mais tout l'enregistrement

AFFICHER "Voulez-vous Supprimer cet Enregistrement O/N ?"

SAISIR rep

Si rep = "O" Alors

Supprimer écriture

Fsi

SINON (de SI plus haut)

AFFICHER ("Cette valeur n'existe pas, Suppression impossible")

FINSI

- Question pour passer éventuellement à une autre suppression

AFFICHER "Autre Suppression O/N ?"

SAISIR rep

JUSQU'A REP = "N"

- **Fermeture fichier**

FERMER Fécriture

FIN

Remarque : Une valeur modifiée par erreur est encore disponible et peut donc être à nouveau modifiée, ce qui n'est pas le cas pour la suppression qui fait que l'enregistrement n'est plus disponible. Il est donc important de demander une confirmation avant d'effectuer l'opération de suppression.

Comment insérer cette confirmation dans ce programme ? Pas besoin d'être un génie pour cela mais soyez au cours.

V. PROGRAMME DE CONSULTATION GLOBALE OU D’AFFICHAGE A L’ECRAN

1. Principe :

il s’agit d’afficher à l’écran tous les enregistrements d’un fichier.

Exemple :

AFFICHER la liste des écritures passées ;

AFFICHER la liste des étudiants

Dans un cas comme dans l’autre l’algorithme doit permettre de parcourir tout le fichier depuis le 1^{er} enregistrement jusqu’au dernier en ayant à chaque fois en mémoire 1 enregistrement. Comment avoir en mémoire 1 enregistrement en algorithme ? C’est la LECTURE

Comment afficher cet enregistrement en algorithme ? C’est l’AFFICHAGE

Ce sont donc les 2 ordres les plus importants (ordre de lecture et d’affichage) pour les programmes de consultation.

2. Ecriture du programme

Nous désirons afficher le contenu du fichier de la façon suivante :

NOECRITURE, DATECRITURE, MONTANT, CODE.

Programme consEcriture

- **Déclaration fichier**

FICHIER = Fiécriture(Organisation séquentielle)

Enregistrement = écriture (ou encore Ecriture = Enregistrement)

Noécriture : numérique

Date écriture : chaîne de caractères

Montant : numérique

Code : caractère

Fin enregistrement

DEBUT

- **Ouverture fichier**

OUVRIR Fiécriture (L)

- **Corps du programme**
- **Première lecture pour se rassurer que le fichier n'est pas vide**

LIRE écriture

SI fin-fichier Alors

AFFICHER "aucune écriture n'est encore passée"

SINON

- **Affichage de l'en-tête**

AFFICHER "Numero, Date écriture, montant, Code"

REPETER

- **Affichage du premier enregistrement lu**

AFFICHER noecrit, datecrit, montant, code

- **Lecture des autres enregistrements**

LIRE écriture

JUSQU'A fin fichier

FSI

Fermer fichier

FIN

Remarque 1: nous aurons plusieurs enregistrements à afficher à l'écran et pas à la même ligne. On peut alors introduire dans cet algorithme la notion de ligne permettant d'afficher par exemple :

Le 1^{er} enregistrement a la ligne L

Le 2^{ièm} enregistrement a la ligne L + 1

Le 3^{ièm} enregistrement a la ligne L + 1 + 1

Le 4^{ièm} enregistrement a la ligne L + 1 + 1 + 1

Ainsi de suite. Cette façon de faire n'est pas identique d'un langage de programmation à l'autre. C'est pour cette raison que nous n'en parlerons pas ici. Par contre au niveau de l'affichage sur feuille (édition à l'imprimante) cette notion de ligne sera utilisée

Remarque 2 : il est également possible comme on le verra dans l'édition à l'imprimante d'utiliser l'ordre d'affectation après avoir lu un enregistrement

M-NOECRIT ← NOECRIT. Cela nous ramènera à allonger inutilement l'algorithme.

VI. PROGRAMME DE CONSULTATION SELECTIVE

1. Principe :

Il s'agit ici d'afficher à l'écran un ensemble d'enregistrements répondant à un ou plusieurs critères.

Exemple :

AFFICHER la liste des écritures débit ;

AFFICHER la liste des écritures débit du mois de Novembre 2012;

AFFICHER la liste des écritures débit du mois de Novembre 2012 et dont le montant dépasse 1 000 000 Frs;

Ces programmes ne contiennent aucune difficulté pour celui qui a compris le programme de consultation générale. En effet l'affichage cette fois-ci sera conditionné et la condition à écrire est fonction du résultat attendu. Ainsi dans le cas de l'affichage des écritures débit, le test sera :

Si code='D' alors

AFFICHER noecrit, datecrit, montant, code

Finsi

Et dans le cas de l'affichage des écritures débit du mois de Novembre 2012

Qu'allons-nous poser comme test ???

2. Ecriture du programme

Nous voulons afficher la liste des écritures débit du mois de Novembre 2012 et dont le montant dépasse 1 000 000 Frs;

Réservée pour la proposition de Solution

[illegible]

Exercices

. **Cas 1** : Cet exercice permet d'**ajouter** ou d'**insérer** des Enregistrements dans le fichier nommé **fetud** (Il s'agit donc d'un programme de création)
Vous pouvez aisément déduire la structure du fichier à partir des données saisies

Algorithme creationEtudiant

*Déclaration de la structure du fichier

Fetud = Fichier

Type etudiant = Enregistrement (ou Etudiant = Enregistrement)

mat:chaîne(11)

nom : chaîne(20)

ddn : date

FinEnregistrement

*Déclaration des variables

Var rep:caractère

Début

Répéter

Ecrire('Entrer le Matricule ')

lire (Etudiant.mat)

Ecrire('Entrer le Nom ')

lire (Etudiant.nom)

Ecrire('Entrer la date naissance ')

lire (Etudiant.ddn)

Ecrire('Enregistrer O/N ? ')

lire(rep);

Si rep='O' alors

Ecrire (fetud,Etudiant)

Fsi

Ecrire ('Autre etudiant O/N ?');

lire(rep);

jusqu'à rep='N';

Fin

Cas 2 : Cet exercice transforme un schéma en une structure d'enregistrement

Différences avec le précédent :

- ✓ L'énoncé ne parle pas de fichier mais seulement d'enregistrement
- ✓ Dans la structure de l'enregistrement la date de naissance (ddn) est décomposée en jour, mois et année et de ce fait peut être considérée comme un sous-enregistrement.
- ✓ Il s'agit d'un seul étudiant donc pas de boucle

A partir de ces informations et des saisies effectuées, proposez un schéma qui correspond à la structure d'un enregistrement

Ecriture du programme !!! Attention ne triche pas, écris pour toi d'abord

Algorithme Etudiant

- Déclaration de la structure des enregistrements **Date et Étudiant**

Type ddn = Enregistrement (ou ddn = Enregistrement)

 Jour : Entier

 Mois : Entier

 Année : Entier

FinEnregistrement

Étudiant = Enregistrement

 mat:chaîne(11)

 nom : chaîne(20)

 ddn : date

FinEnregistrement

- Déclaration des variables

Var f : Étudiant

- Traitement

Début

 Afficher 'Entrer le Matricule '

 Saisir f.mat

 Afficher 'Entrer le Nom '

 Saisir f.nom

 Afficher 'Entrer le jour de naissance '

Saisir f.ddn.jour
Afficher 'Entrer le mois de naissance '
Saisir f.ddn.mois
Afficher 'Entrer l'année de naissance '
Saisir f.ddn.année

Fin

Cas 3 : A traiter absolument

Le fichier du personnel d'une société de la place contient les informations suivantes

Matricule	8 C AN
Nom et prénom	50 C AN
Date embauche	10 C AN
Spécialité	30 C AN
Année de naissance	4 C N
Salaire	7 C N

Proposer un programme permettant de mettre à jour ce fichier. Cette mise à jour va consister soit à ajouter un nouvel enregistrement au fichier, soit à modifier l'enregistrement, soit encore à le supprimer. Dans ce dernier cas il est souhaitable que l'utilisateur confirme la suppression avant que l'enregistrement ne soit supprimé.

Afficher à la sortie du programme le nombre d'opérations par tâche

Exemple d'affichage :

Nombre d'enregistrements créés : 5

Nombre de modifications effectuées : 2

Nombre de suppressions effectuées : 2

Réservée pour la proposition de Solution

Cas 4

Les informations relatives à un étudiant d'une grande école sont :

<u>Propriétés</u>	<u>Longueur</u>
- Numéro matricule	05
- Nom étudiant	20
- Prénoms	30
- Sexe	09
- Cel	11

Réalisez une application qui permet, à partir d'un menu principal de :

- 1) Choisir le module «saisie ou Ajout» pour saisir 10 étudiants et 10 étudiantes.
- 2) Choisir le module «Tri» pour trier par ordre alphabétique la liste des étudiants
- 3) Faire afficher les plus jeunes étudiants
- 4) Faire afficher la liste des étudiants par réseaux cellulaires (ex : MTN, ORANGE, MOOV, KOZ)

Nous proposons dans les pages qui suivent des corrigés ou indications pour quelques exercices.

REFLECHISSEZ ENCORE

AVANT DE TOURNER

CETTE PAGE

CONCLUSION

Nous voici au terme de notre travail qui nous a permis de parcourir les différentes structures que l'on rencontre au cycle BTS. Malgré le sérieux mis à réaliser ce document ainsi que les nombreuses relectures, certaines erreurs peuvent être têtues au point de rester toujours en vie. Si ces erreurs se retrouvent au niveau de certains corrigés, n'hésitez pas à nous les faire connaître. Ces types d'erreurs sont principalement dues au fait que ce sont justement des algorithmes et non des traductions et c'est là toute la difficulté de la matière. En effet le concepteur, l'étudiant n'a pas la possibilité de tester un algorithme et très vite un point-virgule (;) à la place d'un deux points (:) ou une instruction omise donnera un autre sens à votre algorithme

Enfin nous vous recommandons une fois encore de traiter les exercices de ce document de façon approfondie afin que l'objectif soit atteint : ***Acquérir la logique de programmation pour un meilleur développement des Applications***