UNIVERSITE CATHOLIQUE DE L'AFRIQUE DE L'OUEST

Faculté des sciences de gestion

Science – Foi - Action

SAINT MICHEL

COLLEGE PRIVE

COURS ALGORITHME

Moustapha DIOP

Plan de Cours

Cours : Algorithme	Introduction Générale
Moustapha DIOP	*Janvier 2011 – Chapitre I : Introduction Générale*

1. Rappel

Un langage de programmation est un symbolisme permettant une communication entre l'utilisateur et l'ordinateur. Le langage commun comme sa syntaxe est très complexe. Un langage de programmation sera donc plus restrictif, on le définira en fournissant à l'utilisateur le vocabulaire autorisé, sa syntaxe et sa sémantique.

La programmation permet de résoudre des problèmes utilisateurs. La résolution d'un problème nécessite le passage par un certain nombre d'étapes :

- Analyse du problème
- Rédaction d'un Algorithme
- Ecriture du programme

2. Algorithme

L'algorithme consiste avant tout, à déterminer la démarche permettant d'obtenir à l'aide d'un ordinateur, la solution d'un problème donné. Cette programmation ne peut aboutir sans la définition des étapes par lesquelles il faut passer pour aboutir à la solution: c'est ce qu'on nomme généralement « la recherche algorithmique ». La traduction de cette analyse (recherche algorithmique) dans un langage de programmation donné n'en constitue que l'aboutissement et la codification suivant des règles précises.

Faute de disposer d'un moyen rationnel lui permettant d'exprimer sa démarche, le programmeur en est souvent réduit à penser dans le langage qu'il a appris en premier. Dans ces conditions, certaines spécificités du langage, ses particularités techniques, voire sa complexité, masquent nécessairement les aspects fondamentaux de l'algorithme.

Faire l'algorithme consiste alors à définir les étapes par lesquelles il faut passer pour la résolution d'un problème utilisateur, indépendamment des particularités de tel ou tel langage.

3. Définitions

a) Un algorithme est un ensemble de règles permettant de réaliser mécaniquement toutes les opérations particulières correspondant à un certain type de travail.

b) Un algorithme est la décomposition en opérations élémentaires admissibles par l'ordinateur d'un problème utilisateur.

4. Algorithme et Programmation

Le travail de l'ordinateur consiste à exécuter les instructions données par l'homme à une très grande vitesse. Ces instructions sont contenues dans des programmes dont la réalisation nécessite plusieurs étapes.

4.1. Définition exacte du problème

Il faudra alors définir très clairement les objectifs que l'on souhaite atteindre et leurs limitations éventuelles. Cette étude fait partie de la phase d'analyse d'un problème.

Puis il faut déterminer les méthodes de résolution possibles. En général, pour des problèmes complexes, différentes méthodes sont possibles et ne donneront pas des programmes équivalents. Le choix ne peut être fait que par des personnes connaissant parfaitement l'application envisagée et l'ordinateur. ~~Le résultat de cette phase est un « cahier de charges » du programme à réaliser.~~

4.2. Etude détaillée de l'algorithme

La méthode générale étant fixée, il reste à détailler l'algorithme correspondant : Il y aura nécessairement un algorithme pour chaque programme.

4.3. La programmation ou codification

L'algorithme étant réalisé, il ne reste plus qu'à le coder suivant un langage de programmation, puis exécuter le programme ainsi réalisé en ordinateur.

Un programme est constitué d'un ensemble de directives nommées **INSTRUCTIONS** qui spécifient :

- Les opérations élémentaires à exécuter

- La façon dont elles s'enchainent

5. Le Rôle de l'ordinateur dans une entreprise

L'ordinateur peut intervenir à plusieurs niveaux dans une entreprise. Il est utilisé pour des travaux de secrétariat communément appelé BUREAUTIQUE, pour les gros travaux de calcul scientifique, mais aussi intervenir d'une manière intéressante, pour aider à l'optimisation de la solution retenue. C'est ainsi que l'ordinateur a été utilisé pour calculer la trajectoire des obus pendant la seconde guerre mondiale, pour la facturation, pour la paie du personnel, etc…,

Cours : Algorithme	Notion de Variables
Moustapha DIOP	*Janvier 2011 – Chapitre II*

En programmation, nous avons toujours besoin de stocker des valeurs : Il peut s'agir des données issues du disque dur, fournies par l'utilisateur (par l'intermédiaire du clavier), des résultats intermédiaires ou définitifs. Ces endroits de la mémoire centrale où l'on stocke les informations portent un nom : les variables.

Définitions

DEFINITION 1 : en programmation, une variable est donc un nom qui sert à repérer un emplacement donné de la mémoire centrale

DEFINITION 2 : une variable est un emplacement mémoire nommé, de taille fixée prenant au cours du déroulement de l'algorithme un nombre indéfini de valeurs différentes.

Noms des variables

Les noms de variables doivent être significatifs. Les limitations dans le choix des noms de variables dépendront du langage choisi. D'une manière générale, dans tous les langages, un nom de variable est formé d'une ou de plusieurs lettres, les chiffres sont également autorisés. ATTENTION ! Pas d'espaces dans un nom de variable.

Un nom de variable commence impérativement par une lettre.

Déclaration de variables

La déclaration consiste à énumérer toutes les variables dont on aura besoin au cours de l'algorithme. Chaque déclaration doit comporter le nom de la variable (identification) et son type.

La 1ère chose à faire avant de pouvoir utiliser une variable est donc de la déclarer. Cette déclaration consiste à lui réserver une case mémoire à laquelle on donne un nom.

Les noms des variables comme dit un peu plus haut obéissent à des impératifs changeant selon les langages.

Les Types de variables

Les types numériques classiques

Le type de codage (type de variables) choisi pour un nombre va déterminer :

- Les valeurs maximales et minimales des nombres pouvant être stockés dans la variable
- La précision de ces nombres (dans le cas des nombres décimaux).

Tous les langages, quels qu'ils soient, offrent un bouquet de types numériques. On trouve des types suivants :

Byte	1 octet	0 à 255
Boolean	2 octets	True ou False
Integer	2 octets	-32 768 à 32 767
Long (entier long)	4 octets	-2 147 483 648 à 2 147 483 647
Single (virgule flottante, simple précision)	4 octets	$-3{,}402823E^{38}$ à $-1{,}401298E^{-45}$ pour les valeurs négatives ; $1{,}401298E^{-45}$ à $3{,}402823E^{38}$ pour les valeurs positives
Double (à virgule flottante en double précision)	8 octets	$-1{,}79769313486232E^{308}$ à $-4{,}94065645841247E^{-324}$ pour les valeurs négatives ; $4{,}94065645841247E^{-324}$ à $1{,}79769313486232E^{308}$ pour les valeurs positives
Currency (entier à décalage)	8 octets	-922 337 203 685 477,5808 à 922 337 203 685 477,5807 Décimal 14 octets +/-79 228 162 514 264 337 593 543 950 335 sans séparateur décimal
Date	8 octets	1er janvier 100 au 31 décembre 9999
Object	4 octets	Toute référence à des données de type Object
Variant (nombres)	16 octets	Toute valeur numérique, avec la même plage de valeurs qu'une donnée de type Double
Variant (caractères)	22 octets	+ longueur de la chaîne Même plage de valeurs qu'une donnée de type String de longueur variable

A la lecture de ce tableau, on est alors emmené à se poser la question suivante : Pourquoi, ne pas déclarer toutes les variables numériques en réel double ?

En vertu du principe de l'économie des moyens, un bon algorithme ne se contente pas seulement de marcher, il évite en même temps de gaspiller les ressources de la machine

Ex:

*Variable P**4** Entier Long*

Variables PHT, PTTC en réel simple

Dans tous les langages, toutes les variables d'un même type occupent en mémoire un nombre déterminé de bit (sauf le type chaine qui existe dans certains langages)

Autres types Numériques

Certains langages autorisent d'autres types numériques comme

- Le type monétaire
- Le type date (jour /mois/année)

Les types non numériques

Le type caractère

Ces variables peuvent contenir des informations autres que des types numériques. Ce sont alors des informations de type alphanumérique (également appelé de type caractère). Dans une variable de ce type, on stocke des caractères, qu'il s'agisse des lettres, des signes de ponctuation, d'espace ou de chiffres.

Le type booléen

Dans ce type de variable, on y stocke uniquement les valeurs logiques VRAI ou FAUX.

REMARQUE : il est évident que les opérations arithmétiques (addition, soustraction, multiplication, division) possibles avec les variables numériques n'ont aucun sens pour les variables de type caractère. Par contre les comparaisons seront possibles pour les 2 types

Le type d'une variable définit alors :

- La nature des informations qui seront représentées dans la variable (numériques, caractères,...)
- Le codage utilisé (NE, NEL, ….)
- Les limitations concernant les valeurs qui peuvent être représentées.

Structure générale d'un algorithme

Entête ***Algorithme*** Nom Algorithme
Déclaration
    Variable Identificateur : type
    Constante Identificateur = valeur

Corps de l'algorithme
Début
    Instruction 1
    Instruction 2
    ……
    Instruction n
Fin

Cours : Algorithme	L'Instruction d'Affectation
Moustapha DIOP	***Janvier 2011 – Chapitre III***

Nous pouvons renseigner une variable soit par lecture, soit par affectation, soit par calcul

Dans ce chapitre, nous allons aborder nos premières manipulations de variables : leur affecter des valeurs.

L'affectation consiste simplement à placer une valeur dans une variable – Par exemple une instruction qui permettra de dire : *affecter à A la valeur 5* autrement dit *ranger dans A la valeur 5*

La valeur à placer dans une variable pourrait également provenir d'une autre variable ou d'un calcul (cas des variables numériques)

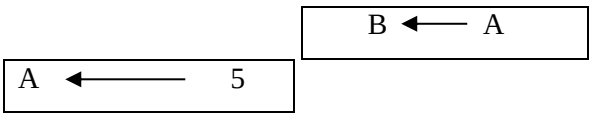
Affecter à A la valeur de B

Ecrire *Affecter à A la valeur 5*, est représenté par beaucoup de langage comme ceci *A=5*

1. Rôle de l'instruction d'affectation

Malgré sa simplicité, l'utilisation du signe = peut prêter à confusion. Nous utiliserons alors le symbole ← car il a le mérite de préciser la variable qui reçoit la valeur

Ex :



Ainsi, ces deux instructions ci-dessus apparaissent clairement comme différente l'une de l'autre.

La 1ère signifie : *Affecter à la variable A la valeur 5*

Tandis que la seconde signifie : *Affecter à la variable B la valeur de la variable A*

Cette représentation est plus claire que si on utilisait le signe =

A=5	B= A

Dans l'instruction d'affectation nous avons 2 parties

- A gauche du symbole ⟵ , on trouve le nom d'une variable destinée à recevoir une valeur
- A droite du symbole ⟵ , on trouve la valeur, ou quelque chose qui précise la valeur en question.

Ainsi B ⟵ A+3, détermine la valeur de A+3 et affecte la valeur à B, sans modifier A

Exercice : Quels sont les valeurs des variables A et B après exécution

	A	B
A ⟵ 1		
B ⟵ A+3		
A ⟵ 3		

2. Les opérations numériques

Ce sont les opérations arithmétiques, tout ce qu'il y a de classique

+ Addition

- Soustraction

* Multiplication

/ Division

^ Puissance

Exercice : Montrez comment évoluent les valeurs de A, B et C en supposant que A et B contiennent initialement 5 et 7

<u>Exercice</u> : Vous disposez de 2 verres. L'un contient de l'eau et l'autre du jus de fruit. Faites de sorte que l'eau se trouve dans le verre de jus et le jus dans le verre d'eau.

<u>Exercice</u> : Donnez les différentes valeurs de X dans les expressions suivantes

A ⟵ 1

B ⟵ 2

C ⟵ 3

*X ⟵ A+B*C*

*X ⟵ A*B+C*

X ⟵ A+C/B

X ⟵ C/A+B

<u>Exercice</u> : Donnez les valeurs des différentes variables dans les expressions suivantes :

	Réponse ⟶	X
A ⟵ 1		
B ⟵ 2		
C ⟵ 3		
X ⟵ A+B*C		
X ⟵ (A+B)*C		
X ⟵ A+C/B		
X ⟵ (A+C)/B		

Réponse

les valeurs de A, B, C ne changent pas, mais x prendra successivement les valeurs suivantes : 7- 9- 2,5- 2

Les variables non définies

1er Cas

Variables X, Y, Z numériques

X ⟵ 3

Z ⟵ X+Y

Y ⟵ 4

Nous voyons que la ligne Z ⟵ X+Y pose un problème parce que Y n'a pas de valeur jusqu'à cette ligne. Ce problème est résolu différemment par les traducteurs:

- certains ignorent l'erreur
- d'autres donnent un message d'erreur
- d'autres simplifient le problème en attribuant initialement la valeur 0 à la variable Y

2ème Cas

Variables X, Y, Z numériques

X ⟵ 3

Z ⟵ X+Y

Y ⟵ 4

Cours : Algorithme	Les instructions d'Ecriture et de Lecture
Moustapha DIOP	*Janvier 2011 – Chapitre IV*

L'instruction d'affectation vue précédemment permet la manipulation des variables. Mais pour qu'un programme présente un intérêt pratique, il devra pouvoir nous communiquer un certain nombre d'informations (résultats), par l'intermédiaire d'un périphérique. Ce sera le rôle de l'instruction d**'écriture.**

Nous pouvons aussi être amenés à transmettre des informations à notre programme, toujours par l'intermédiaire d'un périphérique. Cela se fera par l'instruction de **lecture**

1. L'instruction d'Ecriture

Cette instruction a pour rôle de nous fournir des résultats sous une forme directement compréhensible. Plus précisément elle écrit sur un périphérique les valeurs d'une ou de plusieurs variables

Ainsi, écrire A signifie écrire la valeur de la variable A

Ecrire x, y signifie écrire les valeurs de x puis de y

Présentation des résultats

Exemple de programme

Variable x, y : numériques

X ⟵ *1*

y ⟵ *5*

Écrire x, y

Son exécution | 1 5 | donnera valeurs disposées sur la même ligne

Ce résultat vu à prime abord ne donnera aucune information sur le 1 ou sur le 5 affiché.
L'utilisation des libellés permettrait de donner un sens aux valeurs affichées. Ces informations seront traitées plus loin ()

Autres dispositions

Variable x, y, z : numériques

X ⟵ *3*

Y ⟵ *15*

Z ⟵ *x+Y*

Ecrire x, y

Ecrire z

Son exécution fournira les résultats suivants

3 15

18

Exercice 1

Quel résultat produira ce programme

Variable val, double : numériques

Val ⟵ ***231***

Double ⟵ ***val * 2***

Ecrire double

Exercice 2 : écrire un programme qui calcule et écrit le carré de 547

2. L'Instruction de lecture

L'instruction d'écriture vue dans le paragraphe précédent, permet au programme de nous communiquer des résultats. De manière analogique, l'instruction de lecture va nous permettre de fournir des valeurs à notre programme. Plus précisément, cette instruction va chercher une valeur sur un périphérique et l'attribue à une variable.

Ainsi lire A : signifie prendre une valeur sur le périphérique et la ranger dans A

Lire x, y: signifie prendre deux valeurs et les ranger dans x et y

Dans l'exercice du paragraphe précédent permettant de calculer le double de 231, l'instruction de lecture à la place de l'affectation permettrait de calculer le double de n'importe quel nombre.

On aura :

Variables val, double : numériques
Début
Lire val
*Double ← val * 2*
Ecrire val
Ecrire double
Fin

Exercice : écrire un programme qui lit un nombre et qui écrit son carré

3. Utilisation de "Libellés" dans les instructions de Lecture ou d'Ecriture

Lorsque nous exécutons l'exercice précédent, en fournissant 12 à nombre, le programme affichera le nombre 144. Pour quelqu'un qui vient de jeter un coup d'œil à l'écran, ce nombre n'a aucune signification particulière.

C'est pour cela, il est souhaitable d'accompagner ces instructions de libellés

Exemple: écrire " le carré est", carré

L'exercice précédent sur le carré d'un nombre peut se présenter comme ceci après introduction des libelles.

Variable nombre, carré : numériques
Début
Ecrire "donner un nombre"
Lire nombre
*Carré ← nombre * nombre*
Ecrire "le carré est", carré
Fin

Cours : Algorithme	Les Structures de Contrôle
Moustapha DIOP	*Janvier 2011 – Chapitre III*

1. Les Structures Fondamentales

Jusqu'ici l'utilisation que nous avons faite des instructions précédentes à savoir affectation, lecture, écriture, était séquentielle ; les instructions s'y exécutent dans l'ordre où elles étaient écrites, c'est-à-dire de la 1ère à la dernière. Or, l'intérêt d'un programme en ordinateur et sa puissance proviennent de deux choix :

- La possibilité d'effectuer des choix dans le traitement réalisé : déterminer par exemple une remise dont le pourcentage dépendra du montant de la facture
- La possibilité de répéter plusieurs fois les mêmes instructions. Par exemple, les instructions d'édition d'une facture à un client dans un programme de facturation, seront répétées autant de fois qu'il y a de clients.

Donc un programme informatique comporte deux types d'instruction :

- Les instructions de base : ce sont elles qui permettent la manipulation des variables : affectation, lecture, écriture
- Les instructions de structuration de programme : elles servent à préciser comment doivent s'enchainer chronologiquement ces instructions de base.

2. Les Structures de choix

Supposons un programme permettant d'afficher la décision d'un conseil de classe, décision dépendant de la moyenne :

- Si moyenne >= 10 l'élève est admis

- dans le cas contraire l'élève redouble.

Donc notre programme affichera « Admis » ou « Redouble » suivant le cas

On écrira alors

```
Si moyenne    >= 10 alors
        Décision="Admis"
    Sinon
        Décision="Admis"
 Fin Si
 écrire décision
```

Si la condition est vraie, on exécutera l'instruction (ou les instructions) qui vient après le mot Alors, dans le cas contraire, on exécutera celle qui vient après le mot sinon

Ex : Soit un programme qui permet de lire une moyenne et de la recalculer dans les conditions suivantes :

Si moyenne > 10 l'élève a un bonus de un point

dans le cas contraire l'élève a un malus de un point

Exercice 1 :

Faites un algorithme qui permet d'afficher le nom et le net à payer d'un client sachant une remise est accordée dans les conditions suivantes :

- ***Si le montant de la facture > 100000 une remise de 15% est accordée***
- ***Pour tout autre cas pas de remise***

NB : La TVA est de 18%

Exercice 2

Ecrire un programme qui permet de calculer le net de la facture d'un client à partir du montant hors taxe dans les conditions suivantes :

- ***si le client est de type HTVA, il ne paie pas la tva, mais dans tous les autres cas, la TVA est de 18%***
- ***Une remise de 10% est accordée pour tout montant >250000***

Faites l'algorithme correspondant qui affichera ensuite le Nom et le Net de la facture

2.1. Cas particuliers : Absence de l'une des parties d'un choix

Supposons que dans un programme de facturation, on donne en guise de cadeau une chemise lorsque le montant de la facture est supérieur à 100 000F

Dans ce cas, si le montant de la facture n'est pas supérieure à 100 000, nous n'aurons aucune instruction à exécuter ; il serait alors judicieux et plus naturel de ne pas mentionner le sinon et d'écrire simplement : Si Le montant > 100 000 alors Cadeau ⟵" Chemise"

2.2. Les choix imbriqués

Il peut arriver que l'une des parties d'une structure de choix contienne à son tour une autre structure de choix. Dans ce cas, on dit que l'on a des structures imbriquées les unes dans les autres.

Exemple :

Faites l'algorithme qui permet d'afficher le nom et le net à payer d'un client dans les conditions suivantes :

- ***Si montant Brut de la facture est compris entre 25 000 et 50 000 taux de remise est de 3%***
- ***Au-delà de 50 000F taux de remise de 5%***

NB : Les clients Hors taxe ne paient pas la TVA

Exercice

Le conseil d'orientation de votre établissement a pris les mesures suivantes :

- ***Pour tout étudiant dont la moyenne est comprise entre 11 et 12 (12 comprise) et s'il est âgé de moins de 23 ans, il bénéficie d'une aide mensuelle de 15000F***
- ***Si la moyenne est comprise entre 12 et 14 (14 comprise), il bénéficie d'une demi-bourse de 22500F***
- ***Si la moyenne est >14, l'étudiant bénéficie d'une bourse mensuelle de 50000F***

Tous les étudiants boursiers (aides non comprises) reçoivent en plus un accompagnement de 75000F pour les fournitures.

Pour tout autre étudiant, pas de bourse, pas d'aide, ni accompagnement pour les fournitures scolaires

Faites l'algorithme qui permet de calculer le gain annuel d'un étudiant

3. La structure Répétitive

La répétition jusqu'à (ou boucle) est, au même titre que la structure de choix étudiée dans le paragraphe précédent, une structure fondamentale. Elle permet de répéter un ensemble d'instruction jusqu'à ce qu'une certaine condition soit réalisée.

Exemple

Les élèves d'une classe ont composé dans les matières suivantes : Français, Anglais, Maths, Compta. Sachant que les coefficients sont respectueusement 2, 2, 1, 3 écrire le programme qui permet de calculer, puis d'afficher le nom et la moyenne de chaque élève.

3.1. La sentinelle

La sentinelle dans une structure répétitive est un moyen de poursuivre l'exécution des instructions dans la boucle, ou d'en sortir.

Contrairement à la variable Rep de l'exercice précédent, la sentinelle est une variable de traitement des données de l'énoncé dont une valeur permet de mettre fin au traitement

Exemple : Soit un programme qui calcule le carré d'un nombre. En supposant que l'entrée d'une valeur 0 doit mettre fin à l'exécution, faites l'algorithme correspondant.

Variables a, c : numériques

Début

Répéter

Lire a

Si a >0 alors

*C ⟵ a*a*

Ecrire c

Fin si

Jusqu'à a = o

Fin

Les instructions entre répéter et jusqu'à doivent être répétées jusqu'à ce que la valeur de a soit nulle

Dans cet exemple, le nombre de répétitions de ces instructions n'est pas indiqué explicitement. Il dépendra ici des données que l'on fournira au programme.

D'autre part, pour chacune de ces répétitions, la condition a = 0 n'est examinée qu'après l'exécution de ces instructions

Exemple :

Soit un programme qui permet de lire en continu un nombre < 100

Un nombre >=100 met Fin à la lecture

EXECICE : écrire les instructions permettant de répéter la question

" Voulez-vous continuer ?" jusqu'à ce que la réponse soit égale à "N"

Les valeurs possibles sont "O" ou "N"

Réponse

Variable réponse : caractère

Répéter

Ecrire « voulez vous continuer ? »

Lire réponse

Jusqu'à réponse =" N"

3.2. La Structure Tant Que

La structure de boucle **tant que** comme **répéter… jusqu'à** est une structure qui permet de répéter une ou une suite d'instructions un certain nombre de fois. Mais à la différence de "répéter…. Jusqu'à", les instructions de la boucle peuvent ne pas être exécutées.

Par exemple, dans un programme où l'utilisateur doit répondre à une question dont les seules réponses possibles sont : O pour oui et N pour non

Réponse

On aura alors

Variable Rep : caractère

Début

Ecrire « voulez-vous continuer »

Tant que Rep <> " O" et Rep <> " N" Alors

Lire rep

Si rep <> "O" et Rep <> " N" alors

Ecrire « réponse incorrecte »

Fin si

Fin tant que

Fin

3.3. Le compteur

Les exemples précédents nous ont permis de répéter des instructions, un certain nombre de fois. Ignorant le nombre de répétition, nous avons alors posé parfois la question "Ya-t-il un Autre ? ". La réponse positive à cette question, permet de reprendre les instructions dans la boucle ou d'en sortir dans le cas contraire.

Mais si nous connaissons le nombre d'individus, nous devons répéter le traitement pour chacun. Nous devons alors disposer d'un compteur qui nous permet de compter le nombre de traitement effectué.

Ce compteur est une variable qui porte un nom : par exemple n. A chaque tour de boucle, sa valeur doit évoluer de 1, on parle d'incrémentation (n ⟵ n+1) ou diminuer de 1, on parle de décrémentation (n ⟵ n-1)

Un compteur doit toujours avoir une valeur de départ. Nous dirons alors que nous l'initialisons. Il peut être initialisé de deux manières différentes :

a. en lui donnant la plus petite valeur possible (ex n=0) ; dans ce cas pour faire évoluer n, nous l'incrémentons de 1 : nous aurons n ⟵ n+1

b. Nous pouvons donner à n sa valeur maximale (ex n=15). Dans ce cas sa valeur évolue par décrémentation nous aurons n ⟵ n-1

L'incrémentation ou la décrémentation du compteur est souvent suivie du test de sa valeur pour ne pas dépasser la valeur maximale ou minimale. Ce test n'est pas nécessaire si cette valeur n'est pas connue.

Il existe deux types de comptage : le comptage systématique et le comptage sélectif

3.3.1.Le comptage systématique

Le comptage systématique contrairement au comptage sélectif permet de compter toutes les occurrences.

Exemple :

Soit un programme qui permet de saisir les versements de 5 élèves : Nom (N) et le montant (M)

Autre Exemple

Soit un programme qui permet de calculer le carré d'un nombre puis de l'afficher, et de déterminer par la suite le nombre de valeurs entrées

Sachant que l'introduction d'une valeur 0 met fin au programme, faire l'algorithme correspondant

– nous aurons

Réponse

Variables a, c, n : numériques

n ⟵ O

Répéter

Ecrire « entrez un nombre »

Lire a

*C ⟵ a*a*

écrire "carré =", c

n ⟵ n+1

Jusqu'à a= 0

Écrire "vous avez entré ", n, " valeurs"

3.3.2.Le Comptage Sélectif

Le mécanisme de comptage sélectif sera le même que précédemment avec cette seule différence que l'on incrémente le compteur que lorsqu'une certaine condition est réalisée.

Exemple : Pendant le carême, les étudiants ont participé à la campagne de charité en versant dans les caisses une somme qui varie en fonction des participants. Ecrire un programme qui permet de saisir les informations utiles, puis d'afficher le nombre de filles qui y ont participé.

Soit n, le compteur de filles

Ex : Ecrire un programme qui permet de compter le nombre de « i » dans une phase fournie en différé et se terminant par un point

Cours : Algorithme	Accumulation
Moustapha DIOP	*Janvier 2011 – Chapitre VI*

Nous venons de voir dans le chapitre précédent, la notion de compteur qui consiste à compter le nombre de séquences. Cette notion est différente de celle qui consiste à compter son argent ou sa monnaie.

Dans ce cas, on ne compte pas le nombre de pièces, mais la somme totale que constituent ces pièces. En programmation, nous ferons cette distinction en utilisant le terme "accumulation". Il existe deux types d'accumulation :

- L'accumulation systématique
- L'accumulation sélective

Un accumulateur est une variable qui permet de cumuler les valeurs d'une autre variable

1 L'Accumulation Systématique

Soit à calculer le total des 3 notes obtenues par un élève dans une matière, valeurs lues en différé. Cette résolution est semblable à celle du compteur ; la différence réside dans le fait qu'avec le compteur nous augmentons sa valeur de 1, alors qu'avec l'accumulation, nous ajoutons une certaine valeur à la variable.

Un accumulateur cumule les données d'une variable. Ainsi, si **Total** désigne l'accumulateur, et **note** la variable qui gère les notes, alors nous écrirons l'instruction :

Total ⟵ Total+note

Celle-ci comparée à l'instruction suivante :

Cpt ⟵ cpt+1

Qui permet d'effectuer un comptage

De la même manière que nous avons initialisé le compteur, nous devrons initialiser l'accumulateur en lui donnant une valeur convenable :

Ex : Soit un programme qui permet de calculer le total des points obtenus par un étudiant en Maths sur 3 devoirs. Calculez le total des points obtenus :

Variables note, total, i : numériques
Debut
Total ⟵ *0*
Répéter pour i=1 à 3
Lire note
Total ⟵ *Total+note*
Ecrire "Total des points obtenu est ", Total

Exercice : Calculer et Ecrire la somme d'un nombre quelconque de valeurs suivies d'une valeur 0

Exercices : Lire 50 valeurs. Calculer et écrire leur moyenne en calculant d'abord leur somme

2 L'Accumulation Sélective

Le principe reste le même que précédemment avec cette différence que seules certaines valeurs sont prises en compte (condition).

Exercice : Lire 50 valeurs puis calculer la somme de celles qui sont positives

Exercices : Soit un programme qui permet de lire des valeurs de signe quelconque suivies de 0, puis de calculer et d'écrire la somme des valeurs positives et celle des valeurs négatives :

Cours : Algorithme	L'itération
Moustapha DIOP	***Janvier 2011 – Chapitre VII***

1. Définition

Nous faisons souvent dans un programme de l'itération sans le savoir : Elle a toujours lieu dans des boucles et prend tout son sens dans le comportement des variables. Soient les deux bouts de programmes suivants :

Répéter
    Lire a
    *C ← a*a*
    Ecrire c
Jusqu'à a=0

Programme 1

S ← 0
Répéter
    lire a
    s ← s+a
Jusqu'à a=0
Ecrire s

Programme 2

Ces deux programmes comportent une boucle "Répéter …….Jusqu'à". Cependant dans le *programme 1*, il s'agit de la répétition du même traitement : *Lire une valeur, puis calculer et afficher son carré.*

Dans ce programme, ce qui se passe au $5^{ème}$ tour est totalement indépendant du $4^{ème}$, $3^{ème}$, ….

Par contre dans le *programme 2*, la valeur prise par s au $5^{ème}$ tour, dépend de la valeur qu'elle prend à la fin du quatrième tour, du troisième tour etc…

Nous appellerons itération une telle boucle dans laquelle l'état d'une ou plusieurs variables à un tour, dépend de l'état au tour précédent. Exemple d'itération :

i ← i+1

S ← S+nbre

2 Exemple simple d'itération

Il existe plusieurs formes d'itération. C'est pourquoi il peut parfois être difficile de trouver l'itération permettant de résoudre un problème donné.

Supposons que nous ayons à calculer la somme des 100 premiers nombres entiers non nuls :
1+2+3+4+…..+99+100
Plusieurs possibilités s'offrent à nous :

1er cas

S ← 0

Répéter pour i=1 à 100

S ← S+i

2ème cas

S ← 0

i ← 1

Repeter

S ← S+i

i ← i+1

jusqu'à i >100

3ème cas

S ← 0

i ← 0

Repeter

S ← S+i

i ← i+1

jusqu'à i =100

Ces trois solutions utilisent le même principe : répéter l'instruction *S* *S+i* avec i prenant successivement les valeurs 1, 2, …. 100. Nous allons étudier en détail les différentes valeurs de i et S au cours de l'exécution du 3ème exemple.

Instant	*Valeur de i*	*Valeur de S*
Avant d'être dans la boucle	*0*	*0*
Après 1 tour dans la boucle	*1*	*1*
Après 2 tours dans la boucle	*2*	*1+2*
Après k tour dans la boucle	*K*	*1+2+……+k*
Après k+1 tour dans la boucle	*K+1*	*1+2+……+k+(k+1)*
Après 100 tours dans la boucle	*100*	*1+2+……+99+100*

Le passage d'un état au suivant est réalisé par les deux instructions :

i ← ***i+1***

S ← ***S+i***

Ce sont toujours ces deux instructions qui permettent de progresser, quel que soit l'état dont on part

~~Sachant que la factorielle d'un nombre n se représente comme suit :~~

~~n !=1*2*…….*n~~

~~en remplaçant le signe + par le signe * on pourrait bien utiliser le 3ème cas pour calculer la factorielle d'un nombre~~

nous aurons :

F ← *1*

I ← *1*

Répéter

i ⟵ *i+1*

F ⟵ *F*i*

Jusqu'à i=100

Cours : Algorithme	Les Tableaux
Moustapha DIOP	*Janvier 2011 – Chapitre VII*

1. Tableau à une dimension

1.1. Quand la notion de variable ne suffit plus

A supposer que nous ayons besoin de conserver les 10 notes d'un élève dans un programme. Jusque-là, nous n'avons utilisé qu'une seule variable. Mais si nous voulons disposer simultanément de ces notes, alors nous rencontrerons une diffi

culté.

En utilisant une seule variable, nous ne conserverons pas les 10 notes, car les nouvelles notes viendront écraser les anciennes. A la fin de l'exécution du programme, seule la dernière note est conservée.

L'autre solution consiste à déclarer autant de variables que de notes. Cette solution sera coûteuse en termes de temps, de mémoire et de variables déclarées

Note1, note2…….

Solution

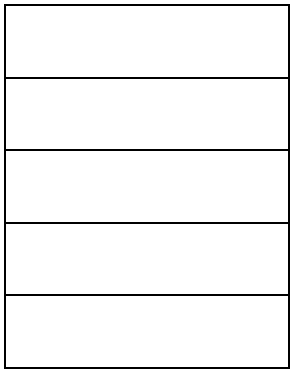
La solution réside à l'utilisation des tableaux et qui consistent :

- A l'utilisation d'un seul nom pour l'ensemble des 10 notes (TNotes),
- A repérer chaque note en utilisant le nom commun suivi entre parenthèses d'un n° entre 1et 10

Ainsi TNotes(1) désigne la 1ère note du tableau TNotes

TNotes(10) désigne la dernière note

Ex :



1.2. Affectations de valeurs à un tableau

L'affectation de valeur dans un tableau, se fait de la manière : Pour un tableau à 4 éléments appelé Tcoef, nous aurons :

Tableau Tcoef(4) : numérique

Debut

TCoef (1) ⟵ 2

TCoef (2) ⟵ 5

TCoef(3) ⟵ 2

TCoef(4) ⟵ 3

2	5	2	3

Son exécution place dans les quatre cases du tableau Tcoef les valeurs :

Considérons maintenant le programme suivant :

Tableau x (4) : numérique

Début

X(1) ⟵ 1

X(2) ⟵ 1

X(3) ⟵ 1

X(4) ⟵ 1

Fin

Son exécution place la valeur 1 dans chaque case de x

En fait, ce programme peut se simplifier si l'on considère qu'il nous suffit de répéter 4 fois une instruction de la forme x(i) ← 1 où i prend les valeurs 1, 2, 3, 4

On dira que i est l'indice du tableau. Cette nouvelle orientation donnera le programme suivant :

Tableau x(4) : numérique

Variable i : numérique

Répéter pour i = 1 à 4

X(i) ← 1

Ou bien

Tableau x(4) : numérique

Variable i : numérique

i ← 1

Répéter

X(i) ← 1

i ← i+1

Jusqu'à i=5

Remarque : il faut bien prendre soin de ne pas confondre l'indice qui sert à repérer un élément du tableau avec la valeur de cet élément. En d'autres termes, il ne faut pas confondre i avec x(i)

1.3. Lecture des éléments d'un tableau

Soit le programme suivant :

Tableau x(4) : numérique

Lire x(1)

Lire x(2)

Lire x(3)

Lire x(4)

Si nous l'exécutons en lui fournissant comme données 8, 15, 30, 13

Nous obtenons ces valeurs dans x

X

8	15	30	13

Ce programme peut se simplifier de la manière suivante

Tableau x(4) : numérique

Variable i : numérique

Répéter pour i =1à 4

Lire x(i)

ou Bien

Tableau x(4) : numérique
Variable i : numérique
i ← 1
Répéter
Lire X (i)
i ← i+1
Jusqu'à i=5

La simplification vous paraitra encore plus évidente si nous supposons que x comporte 1000 éléments.

1.4. Ecriture des éléments d'un tableau

Soit un programme qui permet de charger dans un tableau 6 nombres: 0 dans la 1ère case, 2 dans la dernière case et 1 dans les autres, puis d'afficher le contenu du tableau

Tableau nombre (6) : numérique
Variable i : numérique
Début
Nombre (1) ← 0
Répéter pour i = 2 à 5
Nombre (i) ← 1
Nombre (6) ← 2

Fin

Affichage du résultat

Répéter pour i = 1 à 6

Ecrire nombre (i)

Son exécution remplit ainsi le tableau nombre :

nombre

0	*1*	*1*	*1*	*1*	*2*

Puis on écrit les 6 valeurs. Avec cette convention habituelle (chaque écriture commence sur une nouvelle ligne), les résultats se présentent ainsi :

0

1

1

1

1

2

Exercice : Que fait ce programme et quels résultats fournira-t-il à l'exécution

Tableau nombre (5) : numérique

Variable i : numérique

Répéter pour i = 1 à 5

Nombre(i)⟵ i*i

Répéter pour i = 1 à 5

Ecrire nombre (i)

1.5. Utilisation de variable indicées dans des expressions

Soit ce programme ;

Variable j : numérique

Tableaux a(6), b(6) : numériques

Répéter pour j = 1 à 6

Lire a(j)

Répéter pour j = 1 à 6

b (j)⟵ a(j) +1

Répéter pou j = 1 à 6

écrire b(j)

Exercice 1 : Que fait ce programme et quels résultats fournira-t-il à l'éxécution (D)

Variable i : numérique
Tableau c(6) : numérique
Répéter pour i = 1 à 6
Lire c (i)
Répéter pour i = 1 à 6
c(i)⟵ c(i)*c(i)
Répéter pour i= 1 à 6
Écrire c(i)

Exercice 2 : que fournira l'exécution de ce programme

Tableau nombre (6) : numérique

Variable k : numérique

Nombre (1)⟵ 0

Début

Répéter pour k = + 2 à 6

Nombre (k) ⟵ nombre (k-1) + 2

Répéter pour k = 1 à 6

Ecrire nombre (k)

Fin

Exercice 3 : que fournira l'exécution de ce programme (D)

Tableau suite(8) : numérique

Variable i : numérique

Suite (1) ⟵ 1

Suite (2) ⟵ 6

Répéter pour i = 3 à 8

Suite (i) ⟵ suite (i-1) + suite (i-2)

Répéter pour i = 1 à 8

Ecrire suite (i)

2. Quelques opérations avec les tableaux

Les algorithmes que nous avons utilisés jusque-là peuvent s'appliquer sans difficultés aux tableaux.

2.1. Calcul de la somme des éléments d'un tableau

Soit T un tableau de 20 éléments réservé de la manière suivante :

Tableau T(20) : numérique

Pour calculer la somme nous aurons :

Som ⟵ 0

Répéter pour i = 1 à 20

Som ⟵ som +T(i)

Exemple : écrire les instructions permettant d'obtenir la somme des éléments positifs (pos) et la somme des éléments négatifs (Neg) d'un tableau T de 20 nombres

Cours : Algorithme	**Les Tableaux à deux dimensions**
Moustapha DIOP	***Janvier 2011 – Chapitre VII***

Nous avons vu qu'un tableau à une dimension correspond à une liste ordonnée de valeurs, repérées chacune par un indice. Mais dans la vie courante, nous avons surtout tendance à appeler tableau, un ensemble de données susceptible d'être présentées sous forme de lignes et de colonnes.

Noms	Maths	Anglais	SGBD
Alphonse	10	13	10
Ahmadou	11	18	15
Seynabou	18	13	15
Anna	20	17	19
Mado	09	06	14

Comme dans le cas des tableaux à une dimension, chaque note est repérée par un indice, mais dans le cas de figure chaque note est repérée par deux indices qui en précisent la position : l'indice de ligne et l'indice de colonne.

Une note sera alors repérée dans un tableau par le nom et la valeur des indices de ligne et de colonnes. Ainsi si le tableau précédent porte le nom de Notes, Note(2,4) représentera la note d'Ahmadou en SGBD

Par convention et pour le restant du chapitre, nous considérons le 1er indice comme indice de ligne, et le second comme indice de colonne.

1. Comment utiliser un tableau à deux dimensions

1.1. Déclaration :

Il faut d'abord le déclarer, c'est à dire lui attribuer de la place en mémoire en lui donnant un nom, préciser son type, et donner la valeur maximale des lignes et des colonnes

Ex : tableau notes (20, 10) : numérique

Pour repérer un élément dans un tableau : chaque élément du tableau sera repéré par deux indices dont l'un sera compris entre 1 et 20 et l'autre entre 1 et 10

1.2. Affectation des valeurs à un tableau :

Considérons le programme suivant :

Tableau x(2,3) : numérique

X (1,1)◄—— 5

X (1,2)◄—— 12

X (1,3)◄—— 2

X (2,1)◄——8

X (2,2)◄——9

X (2,3)◄—— 5

Son exécution place les valeurs 5,12, 2, 8,9, 5 dans le tableau nommé x et nous aurons :

X

5	12	2
8	9	5

Considérons maintenant le programme suivant

Tableau x (2, 3) : numérique

Début

X(1,1) ◄——1

X(1,2) ⟵ 1

X(1,3) ⟵ 1

X(2,1) ⟵ 1

X(2,2) ⟵ 1

X(2,3) ⟵ 1

Fin

Son exécution place la valeur 1 dans chaque case du tableau X. Ce programme peut être simplifié si l'on considère qu'il nous suffit de répéter une instruction de la forme x (i, j) où i prend les valeurs 1, 2 et j prend les valeurs 1, 2, 3. On aura

Tableau x (2,3) : numérique

Répéter pour i = 1 à 2

Répéter pour j = 1 à 3

X (i,j) ⟵ 1

1.3. Exemple de lecture des éléments d'un tableau

Soit un programme qui permet de remplir en lecture un tableau de 2 lignes et 3 colonnes

Tableau x(2,3) : numérique

Début

Lire x(1,1)

Lire x(1,2)

Lire x (1, 3)

Lire x (2,1)

Lire x(2,2)

Lire x (2,3)

fin

On peut simplifier l'écriture du programme en ceci

Tableau x (2,3) : numérique

Variable i, j : numériques

Debut

Répéter pour i=1 à 2

Répéter pour j = 1 à 3

lire x (i, j)

Fin

1.4. Exemple d'écriture des éléments d'un tableau

Exemple : soir un programme appelé à remplir un tableau de deux lignes et trois colonnes avec une suite de nombres de raison 1, puis d'afficher les éléments du tableau.

Tableau x (2,3) : numérique

Variable i, j, val : numérique

Val ← 1

Répéter pour i = 1à 2

Répéter pour j = 1 à 3

X (i, j) ← val

Val ← val +1

Répéter pour i = 1 à 2

Répéter pour j = 1 à 3

Écrire x (i, j)

Exercice : quels résultats fournira ce programme

tableau t(4, 2) : numérique

variables k,m : numériques

Répéter pour k = 1 à 4

Répéter pour m=1 à 2

t (k, m) ← k + m

Répéter pour k =1 à 4

Répéter pour m = 1 à 2

Ecrire t (k, m)

2. Quelques algorithmes classiques appliqués aux tableaux à 2 dimensions

2.1. Calcul de la somme des éléments d'un tableau à 2 dimensions

Nous savons que t (i, j) permet de repérer un élément du tableau. Donc pour calculer la somme des éléments d'un tableau de 20 lignes et de 50 colonnes nous aurons :

Variable i, j, som : numériques

Tableau t (20, 50) : numérique

Som ⟵ 0

Répéter pour i=1 à 20

Répéter pour j = 1à 50

Som ⟵ som +t (i, j)

2.2. Calcul de la moyenne des éléments d'un tableau à 2 dimensions

Faire l'algorithme qui permet de faire la moyenne des éléments d'un tableau de 2 lignes et 3 colonnes

Exercices

Faire l'algorithme qui permet de charger un tableau de 2 lignes et 3 colonnes avec des entiers positifs, puis de déterminer le nombre le plus élevé

Démarche : charger d'abord le tableau, puis déterminer le nombre le plus élevé

Cours : Algorithme	**Introduction sur les fichiers**
Moustapha DIOP	***Janvier 2011 – Chapitre VIII***

Un fichier est constitué d'enregistrements ou articles. Un enregistrement est constitué de zones ou champs. Chaque champ est défini par sa nature (alphabétique, Alphanumérique, numérique …).

L'organisation du fichier détermine le mode d'accès aux données.

Matricule	Prénom	Nom	Adresse	Catégorie

Ex d'enregistrement

6. Organisation séquentielle

On dit que l'accès est séquentiel si l'accès à un enregistrement donné (de rang n) est lié à l'accès à l'enregistrement précédent (de rang n-1).

Les enregistrements sont affichés suivant l'ordre de création. Cette organisation est très souple, mais peut être trop lente pour certains traitements. Pour retrouver par exemple un enregistrement, il faut parcourir tous les enregistrements qui l'ont précédés. Cette consultation peut être très longue si le fichier est important et si l'information à rechercher est loin sur le support.

Un fichier en Organisation séquentielle peut être créé sur n'importe quel support séquentiel ou adressable, mais la seule organisation possible pour les supports séquentiels.

L'ajout d'un nouvel enregistrement se fait à la fin du fichier.

7. Organisation séquentielle indexée

Ce type d'organisation rappelle celle d'un dictionnaire qui est divisé en pages. L'accès à une page se fait à travers une clé qui correspond aux lettres qui commencent la page. Pour un fichier qui a une organisation analogue, on crée une table de repérage ou table d'index qui découpe le

fichier en un certain nombre de pages. La table d'index sera composée d'un certain nombre de matricules dans le cas de l'article cité plus haut, chaque matricule étant suivi de l'adresse.

Au moment de l'affichage des articles, ceux-ci ne sont pas affichés par ordre de création, mais par ordre croissant sur la clé. L'ajout d'un nouvel enregistrement se fait à la fin du fichier et son repérage se fait dans la table d'index d'où son repérage se fait grâce à la clé.

8. Organisation aléatoire

Cette organisation permet un accès plus directe à un article par détermination d'une page et par recherche séquentielle sur ce bloc et de ne pas utiliser de table d'accès en obtenant l'adresse de la page grâce à un calcul sur la clé de l'article.

Nous effectuons tous les jours des opérations sur les fichiers. Ces opérations sont les suivantes :*

- Création
- Ajout d'enregistrement
- Modifier des enregistrements
- Supprimer des enregistrements

9. Classification des fichiers

On peut distinguer schématiquement deux catégories de fichiers : les ***fichiers permanents*** et les ***fichiers mouvements***.

9.1. Les fichiers permanents

Ils correspondent à la description d'entités stables dans le temps. Les informations qu'ils contiennent sont réutilisables à chaque traitement car les valeurs ne varient pas systématiquement d'une période à l'autre

Ex : Fichier régulier des clients, fichier des salariés etc….

9.2. Les fichiers mouvements

Ils servent à la description d'Evènements. Ils contiennent des informations nouvelles et périodiques qui ne seront pas réutilisables car les évènements ne se reproduisent pas d'une manière strictement identique.

Ex : l'ensemble des versements des élèves, l'ensemble des bons de sorties etc…

10. Opérations sur les fichiers

Plusieurs opérations sont possibles avec les fichiers. Ces opérations nous permettent de manipuler les fichiers

10.1. La consultation

Elle consiste à extraire du fichier différents articles pour les lire, sans les modifier

10.2. La mise à jour consiste :

- A ajouter de nouveaux articles
- Supprimer des articles
- Modifier des articles

10.3. La fusion

Elle consiste à créer un fichier unique à partir de plusieurs fichiers

10.4. Eclatement de fichier

A partir d'un fichier d'origine, créer plusieurs fichiers

10.5. Tri

A l'intérieur d'un fichier, ranger les articles selon l'ordre qu'un aura choisi

Cours : Algorithme	Autre représentation d'Algorithme: Les Tables de Décision
Moustapha DIOP	*Janvier 2011 – Chapitre I*

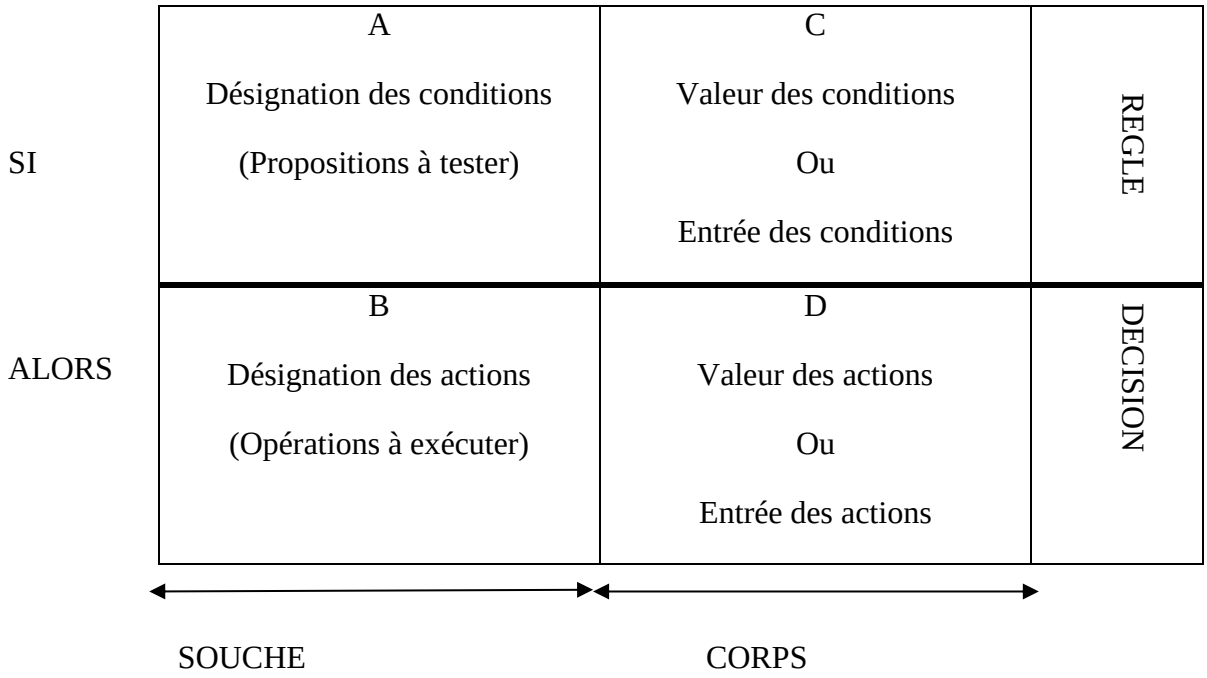
Lorsque l'on veut décrire un traitement itératif, l'ordinogramme apparaît comme l'un des meilleurs procédés de représentation. Par contre, si dans le traitement il est nécessaire de décrire de nombreux choix logiques, la multiplication des alternatives confère rapidement à l'organigramme un aspect embrouillé qui le rend peu agréable à utiliser. Si les conditions sont nombreuses, il est parfois préférable d'utiliser un autre mode de représentation: **Les tables de décision**.

1. Structure Générale

Une table de décision peut être divisée en 2 parties principales, chacune divisée à son tour en 2 parties:

a. La souche comprend deux parties:
   - Les *conditions* qui interviennent dans la table (Partie A) → énoncé des propositions
   - Les *actions* (ou tâches) qui doivent être exécutées pour chaque ensemble des conditions possibles (Partie B)

b. Le Corps comprend également deux parties:
   - La valeur des conditions d'entrée (entrée des conditions): un groupe de valeurs de conditions d'entrée à remplir simultanément, occupe une même colonne et constitue une règle
   - La valeur des actions (Entrée des actions): Les actions qui correspondent à une règle constituent les décisions.



A chaque règle, correspond une ou une série de décisions. On peut traduire une colonne de table par une expression du genre:

Si … (conditions), Alors Exécuter … (Décisions)

Les actions sont exécutées dans l'ordre où elles sont écrites sur la souche, de haut en bas.

Il existe deux types de table de décision:

- Les tables à entrées étendues
- Les tables à entrées limitées

Quelque soit le type de tables, la détermination du nombre de colonnes dépend du nombre de conditions. S'il ya **n** conditions, le nombre de colonnes est égal à 2^n. Donc pour une table à 3 conditions, le nombre de colonne est égal à 2^3 soit 2x2x2=8 colonnes

2. Les différents types de Tables de décision

2.1. Les tables à entrées étendues

Pour les tables à entrées étendues, les valeurs des conditions figurent dans les colonnes; de même, la nature des actions à exécuter est préciser dans la colonne

Exercice d'Application

Pour le paiement des factures, l'entreprise ROBEIN prévoit deux cas:

1° Le cas du Client entrepreneur. Il ya eu de distinguer selon que la facture est supérieure ou égale à 10000F. Si elle est supérieure à 10 000F, on distinguera l'entrepreneur fidèle ou

l'entrepreneur occasionnel. Le 1er pourra payer par traite à 60 jours; le second ne disposera qu'un délai de paiement de 0 jours.

2ème Le cas du client non entrepreneur. Si la facture est supérieure ou égale à 10 000F et si le client est fidèle, il bénéficie d'une remise de 10% et pourra payer par traite à 60 jours.

Dans les autres cas (client occasionnel ou facture inférieure 10 000F), le paiement est exigible sous huitaine, mais on accordera une remise de 5%.

Etablir la table de décision des conditions de règlement.

Exemple: Un établissement scolaire, étudie les conditions d'orientation des élèves de 3ème dans les différentes séries: S, G, L

- *Pour passer en série S, il faut remplir les conditions suivantes: être âgé de 16 ans et avoir une moyenne annuelle supérieure à 13*
- *Pour passer en série G et L pas de restriction sur l'âge.*
- *Pour la série G avoir une moyenne inférieure ou égale à 13 et supérieure ou égale à 11,5*
- *Pour la série L avoir une moyenne >=10 et inférieure à 11,5*

Décrivez par une table de décision, le texte ci-dessus

Vous voyez à travers cet exemple, que plusieurs conditions pour posées pour une seule décision. Le tableau que vous construirez doit représenter tous les cas de figure qui se présenteront afin de pouvoir respecter la décision du conseil.

Le premier problème à résoudre c'est d'étudier les conditions

Le second problème est d'étudier les actions à entreprendre

La détermination du nombre de colonnes découlera naturellement du nombre de conditions.

A méditer

2.2. Les tables à entrées limitées

Les tables à entrées étendues sont commodes pour la présentation de la première analyse d'un traitement. Toutefois, pour effectuer une analyse plus détaillée et pour utiliser les règles de l'algèbre logique, on utilise des tables à entrées limitées

Dans la forme à entrées limitées, la condition ou l'action est exprimée entièrement dans la souche. Dans chaque colonne :

- Pour les conditions on indique :

O si la condition est vérifiée

N si la condition n'est pas vérifiée,

Rien si la condition est sans influence

- Pour les actions, on indique :

  Une croix (x), si l'action doit être exécutée

  Rien si l'action ne doit pas être exécutée

Exercice d'application: ***Un laboratoire de produits de beauté procède de la manière suivante pour la facturation à ses clients:***

- ***Si la commande atteint 50F, un cadeau est offert***
- ***Si la commande est supérieure ou égale 100F et inférieure à 150F, remise de 5%***
- ***Si la commande est supérieure ou égale à 150F, remise 10%***
- ***Si la commande est inférieure à 100F, ajouter 45F de frais d'envoi***
- ***Au-dessus de 100F pas de frais d'envoi.***

Présenter la table de décision

Exercice :

Un grand magasin accorde à son personnel ayant une ancienneté supérieure à un an, des réductions sur les prix pratiqués.

- Le personnel du service ventes a droit à une réduction de 5**%** sur les achats effectués dans le magasin.
- Le personnel n'appartenant pas au service vente et les chefs de rayon du service ventes ont droit à une réduction de 3**%**.
- Les articles en promotion ne donnent droit à aucune réduction.
  Établir la table de décision

TD n°1

Les étudiants de LPIG1 ont composé dans les matières suivantes : Algo, Maths, Compta et Economie. Sachant que les coefficients sont respectivement 7, 5, 4, 3 écrire le programme qui permet d'afficher le nom et la moyenne de chaque étudiant, puis de charger la moyenne de chaque étudiant dans un tableau dont vous définirez la dimension.

Vous afficherez la moyenne la plus élevée et la plus faible

TD n°2

Les étudiants de l'état disposent d'une cantine scolaire, où ils peuvent récupérer les produits dont ils ont besoin. Les quantités dépendent de l'étudiant, mais le maximum autorisé est 3.

Ecrire le programme qui permet pour chaque étudiant

- de lire les informations sur tous les produits pris
- de calculer la valeur totale des produits
- puis d'afficher le nom et la valeur totale de ses produits

Enfin calculer la valeur totale des produits consommés par tous les étudiantsc