DEUST IOSI

FORMATION CONTINUE

PROGRAMMATION STRUCTUREE

PARETIAS Philippe
Septembre 2005

paretias@univ-valenciennes.fr

SOMMAIRE

Chap 1 - GENERALITES

A - PRELIMINAIRES

1 - Notions générales

L'ordinateur ne peut exécuter que des parties mécanisables de l'activité intellectuelle de l'homme. La contribution humaine, la plus noble, est le **raisonnement**. Il appartient donc à l'informaticien d'exprimer son raisonnement d'une façon telle que l'ordinateur puisse le comprendre. Le produit final est ce que nous appelons **un programme**.

Il est le résultat d'une double activité :

- passer de l'énoncé d'un problème à la description d'un modèle de comportement : **l'algorithme**,
- traduire cet algorithme dans un langage approprié, compréhensible par l'ordinateur : le **programme**.

On peut donc représenter comme suit, l'activité de programmation :

Raisonnement *Codification*

Enoncé du problème ---------->Algorithme ---------- >Programme

2 - Exemple

Soit à définir l'ensemble des opérations élémentaires à réaliser entre le moment où le réveil sonne et le moment où l'on sort pour aller à l'université.

1 - Je déjeune
2 - Je mets mon manteau
3 - Le réveil sonne
4 - Je me douche
5 - Je me lève
6 - Je sors
7 - Je mets mes chaussures
8 - Je prépare le café
9 - Je me réveille
10 - Je ferme la porte
11 - J'enlève mon pyjama
12 - Je m'habille
13 - J'ouvre la porte

Ces opérations étant dans le désordre, retrouvez l'ordre des opérations.

3 - Définition

Un algorithme fait passer d'un état initial à un état final de façon déterministe. Il doit respecter les règles suivantes :

- il est défini sans ambiguïté,
- il se termine après un nombre fini d'opérations appelées **actions**,
- il doit être effectif : toutes les opérations doivent pouvoir être effectuées par un homme utilisant des moyens manuels,
- il manipule des objets qui doivent être définis de manière très précise.

B - NOTIONS D'OBJETS ELEMENTAIRES

Le traitement d'un objet concerne la valeur de cet objet. Si cette valeur ne peut être modifiée, nous parlerons de **constantes**, sinon nous parlerons de **variables**.

Un objet est parfaitement défini si nous connaissons ses trois caractéristiques :

- son identificateur : c'est une suite quelconque de caractères alphanumériques, sans espace, commençant obligatoirement par une lettre,

- sa valeur : constante ou variable,

- son type :

* ***entier ou réel*** : ensemble des valeurs de ***Z*** ou ***R***
ensemble des opérateurs :
+ : addition
- : soustraction
* : multiplication
/ ou DIV : division
^ : élévation à la puissance

* ***booléen*** : ensemble des valeurs logiques (VRAI, FAUX)
ensemble des opérateurs :
+ : NON
$\vee$: OU
$\wedge$: ET

* ***caractère*** : ensemble des valeurs suivantes
26 lettres de l'alphabet (majuscules et minuscules)
10 chiffres
codes opérations (+, -, *, ...)
codes de ponctuation (;, ., ...)
autres codes ($, £, §, ...)
Un caractère est lié à un code numérique (code ASCII) qui le représente et qui permet d'établir une relation d'ordre.

L'association d'une information à son type est généralement appelée **déclaration**.

La notation des déclarations dépend de la façon utilisée pour représenter les algorithmes :
pseudo code ou **arbre programmatique**

	Déclarations
Pseudo code	→ **pour les constantes** **Const** nom info **de type** désignation du type = valeur → **pour les variables** **Soit** nom info **de type** désignation du type
Arbre programmatique	→ **pour les constantes** nom info : désignation du type = valeur → **pour les variables** nom info : désignation du type

C - NOTIONS D'OBJETS STRUCTURES

1 - Le type chaîne de caractères

C'est un ensemble fini d'éléments de type caractère.

2 - La structure cartésienne

C'est une chaîne de caractères décomposable en objets élémentaires

D - EXERCICES

1-1 – Donner le type des variables suivantes :
REVENU, NBENFANT, SITUATION-FAMILLE, ADRESSE_ETU

1-2 – Validité des variables suivantes :
TOTAL, NB_DE_LIVRES, CUMUL 1

1-3 – Evaluation des priorités :
A * B + C / D – E ^ N
A + B * C ^ D - E

1-4 – Etablir la structure de ETAT CIVIL

Chap 2 - INSTRUCTIONS ELEMENTAIRES

A - SAISIE DE L'INFORMATION

	Notation
Pseudo code	ENTRER (nom variable)
Arbre programmatique	

Une donnée est entrée dans l'ordinateur à partir d'un organe d'entrée (le clavier en général). L'opération consiste à modifier la valeur de la variable citée. La nouvelle valeur étant celle présentée sur l'organe d'entrée.

Exemples :

1. ENTRER (REVENU)
2. ENTRER (NOM)

AVANT	INSTRUCTION	APRES
?	ENTRER (NOM)	

	ENTRER (NOM)	

B - L'AFFECTATION

	Notation : si X est une variable
Pseudo code	X ← valeur ou variable
Arbre programmatique	

Cette action consiste à affecter (c'est-à-dire donner) une valeur à une information. Cette valeur peut être une constante, le contenu d'un autre objet (variable ou constante) ou le résultat d'un calcul.

Exemples :

1. NBENFANTS ← 2
2. TVA ← 19,60
3. CHAR ← "X"
4. NOM ← "TITI"
5. SURNOM ← NOM

AVANT	INSTRUCTION	APRES
?	NOMBRE ← 5	

	NOMBRE ← -8	

C - LA RESTITUTION DE L'INFORMATION

	Notation : si X est une variable
Pseudo code	AFFICHER (nom variable)
Arbre programmatique	

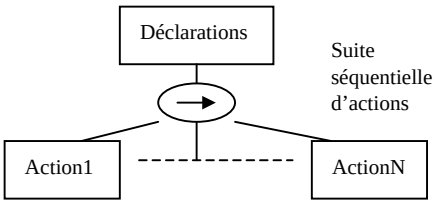
Cette action présente sur l'organe de sortie (l'écran en général) la valeur de la variable citée.

Exemples :

1. AFFICHER (IMPOTS)
2. AFFICHER ("Texte à l'écran")

D - PREMIER ALGORITHME

1 - Structure d'un algorithme

Pseudo code	Arbre programmatique
Déclarations DEBUT Actions FIN	

2 - Exemple

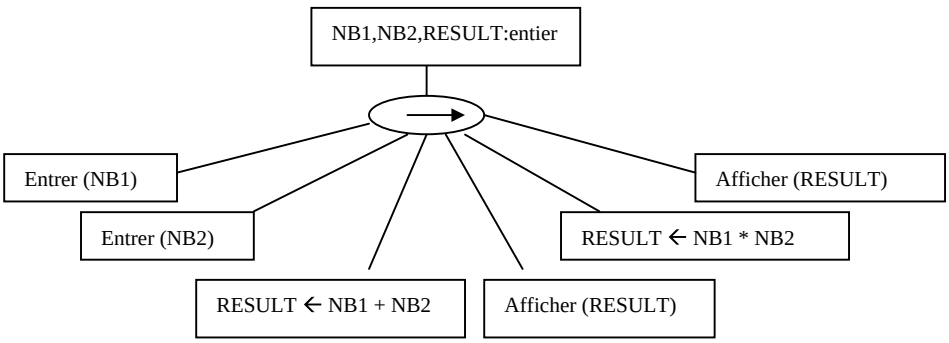
A partir de la saisie de deux nombres, écrire un algorithme permettant de faire la somme et le produit de ces deux valeurs.

- En pseudo code

```
Soit NB1, NB2, RESULT de type ENTIER
DEBUT
    ENTRER (NB1)
    ENTRER (NB2)
    RESULT <-- NB1 + NB2
    AFFICHER (RESULT)
    RESULT <-- NB1 * NB2
    AFFICHER (RESULT)
FIN
```

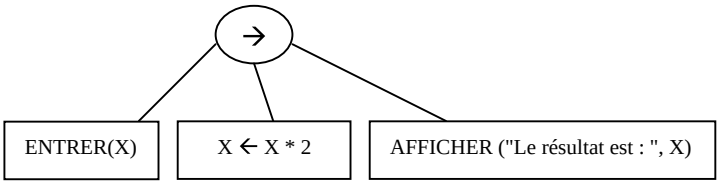
Remarque : Pour rendre convivial un algorithme, il est possible d'ajouter :
- l'affichage de chaînes de caractères pour accompagner la saisie ou les résultats,
- des commentaires pour une plus grande lisibilité :
notation → **/*** texte du commentaire ***/**

- En arbre programmatique



E - LA SEQUENCE D'ACTIONS

C'est une **suite** séquentielle d'actions simples.

Pseudo code	Arbre programmatique
ENTRER (X) X <-- X * 2 AFFICHER ("Le résultat est : ", X)	

F - LA TRACE

Afin de vérifier le résultat d'un algorithme il est nécessaire de faire une **trace**. C'est un tableau dans lequel on retrouve :

- l'ensemble des instructions exécutées,
- la liste de toutes les variables et constantes,
- le clavier,
- l'écran.

Instructions	Clavier	Liste des variables et constantes			Ecran

Une trace est une exécution manuelle de l'algorithme pour certaines valeurs

G - EXERCICES

2-1 – A partir de la saisie de deux nombres entiers, écrire un algorithme en pseudo code permettant de faire les quatres opérations de base (addition, soustraction, multiplication et division) et d'afficher les résultats obtenus.

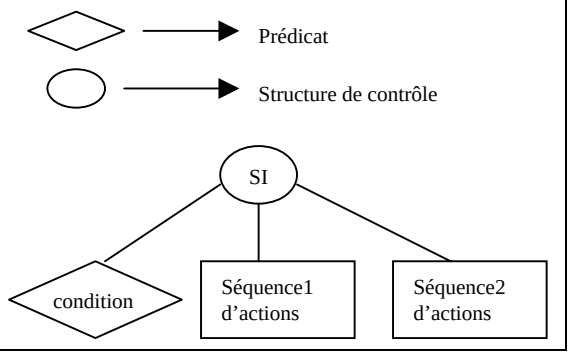
2-2 – Retranscrire cet algorithme sous la forme d'un arbre programmatique.

2-3 – Vérifier votre algorithme à l'aide d'une trace.

Chap 3 - LES STRUCTURES CONDITIONNELLES

A - LA STRUCTURE ALTERNATIVE

Pseudo code	Arbre programmatique

SI <condition> Alors
  Séquence1 d'actions
SINON
  Séquence2 d'actions
FINSI
	

La condition sera exprimée en utilisant un des comparateurs suivant :

=	égal
<>	différent
<	inférieur
<=	inférieur ou égal
>	supérieur
>=	supérieur ou égal

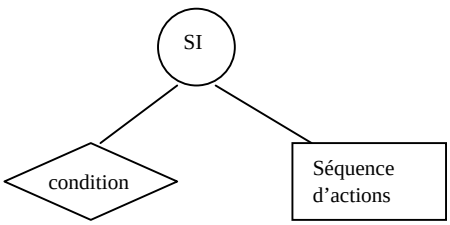
Lorsque l'évaluation de la condition produit la valeur
VRAI : la séquence1 d'actions est exécutée,
FAUX : la séquence2 d'actions est exécutée

Exemple

Ecrire un algorithme sous la forme d'un arbre programmatique, permettant d'afficher la différence positive entre deux nombres entiers A et B saisis au clavier

- A = 3 et B = 5 → l'algorithme fera B - A
- A = 6 et B = 2 → l'algorithme fera A - B

B - LA STRUCTURE CONDITIONNELLE

Pseudo code	Arbre programmatique
SI <condition> Alors Séquence d'actions FINSI	

Cette structure a la particularité de ne pas avoir de traitement à effectuer lorsque l'évaluation de la condition produit la valeur **FAUX**. La séquence d'actions ne sera exécutée uniquement lorsque l'évaluation de la condition produit la valeur **VRAI**.

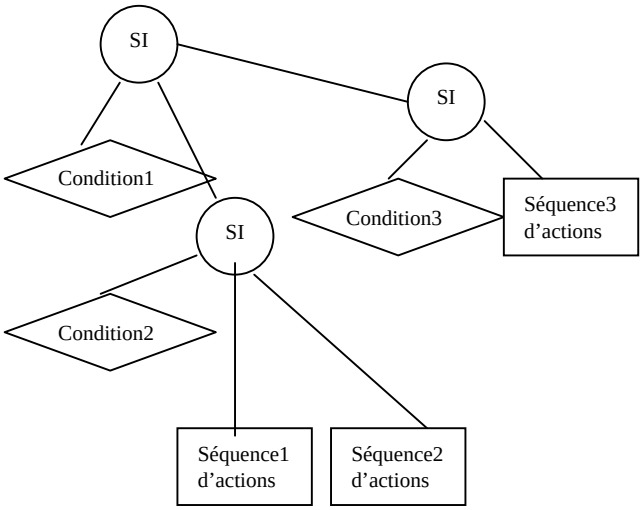
Exemple

A partir de la saisie d'une année de naissance, écrire un algorithme en pseudo code, permettant de calculer l'âge. Un message particulier ("Trop jeune") s'affiche pour toute personne de moins de 18 ans.

C – L'IMBRICATION DE SI

Il se peut, dans certains cas, que l'expression de la condition d'un SI ne suffise pas pour exprimer tous les cas de figures. Nous pouvons alors imbriqués les SI et également mélanger les structures alternatives et les structures conditionnelles.

Exemple

Pseudo code	Arbre programmatique
Si <condition1> Alors Si <condition2> Alors Séquence1 d'actions Sinon Séquence2 d'actions Fin Si Sinon Si <condition3> Alors Séquence3 d'actions Fin Si Fin Si	

Remarque : On peut également réunir les conditions entre elles dans un SI avec les opérateurs logiques (ET, OU).

Exemple

Modifier l'algorithme permettant d'afficher la différence positive entre deux nombres en traitant l'égalité comme un cas particulier.

D - EXERCICES

3-1 Ecrire un algorithme en arbre programmatique, permettant de comparer deux caractères saisis au clavier et afficher le premier dans l'ordre alphabétique.

Exemples

L'utilisateur saisi le caractère "T", puis le caractère "D". Le programme affichera "D".
L'utilisateur saisi le caractère "A", puis le caractère "B". Le programme affichera "A".
L'utilisateur saisi le caractère "M", puis le caractère "M". Le programme affichera "Lettres identiques".

3-2 Un robot conduit une voiture. Il peut exécuter trois actions "s'arrêter", "ralentir", "passer" en fonction de la couleur des feux (vert, orange ou rouge) qui sera une variable saisie. Ecrire l'algorithme en pseudo code qui permet d'afficher au robot ce qu'il doit faire.

Exemples

L'utilisateur saisi la couleur Rouge, le message "s'arrêter" s'affiche.
L'utilisateur saisi la couleur Vert, le message "passer" s'affiche.
L'utilisateur saisi la couleur Orange, le message "ralentir" s'affiche.
L'utilisateur saisi la couleur Bleu, le message "couleur inconnue" s'affiche.

3-3 Ecrire un algorithme en arbre programmatique permettant de déterminer la quantité de nombre égaux dans un triplet.

Exemples

Pour (1, 3, 5), le programme affichera 0
Pour (2, 3, 2), le programme affichera 2
Pour (3, 3, 3), le programme affichera 3

3-4 A partir de la saisie de deux nombres réels A et B, écrire un algorithme en pseudo code permettant de résoudre l'équation :

$$ax + b=0$$

Méthode

Si a = 0 et b = 0 alors l'ensemble des solutions est R
Si a = 0 et b <> 0 alors l'ensemble des solutions est l'ensemble vide
Si a <> 0 alors l'ensemble des solutions est (-b/a)

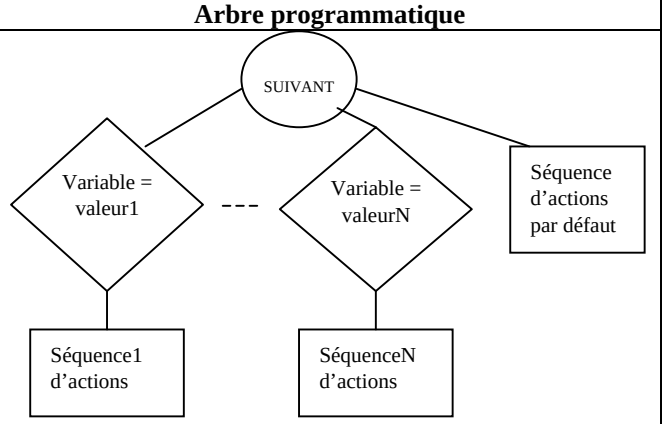
3-5 Soit l'algorithme suivant:

```
Soit X, Y, Z de type entier                    /* X, Y, Z > 0 */
DEBUT
    Entrer (X)
    Entrer (Y)
    Entrer (Z)
    X <-- 2 * X
    Y <-- 3 * Y
    Z <-- 2 * X + 3 * Y
    SI  X <= Z  Alors
    |       Y <-- X
    SINON
    |       Y <-- Z
    FSI
    SI  X <> Y  Alors
    |       Afficher (X)
    SINON
    |       Afficher (Y)
    FSI
FIN
```

Travail à faire

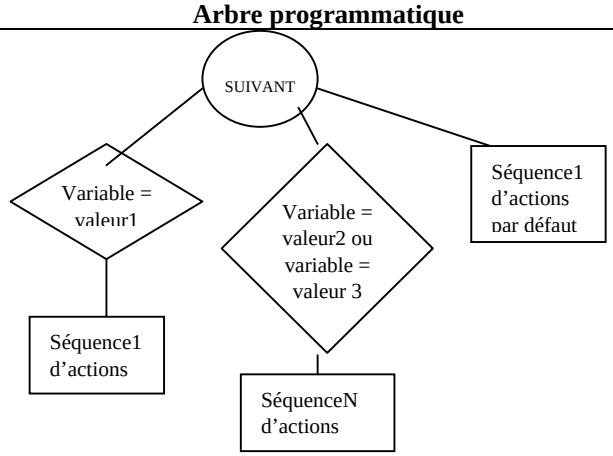
1 - Exécutez cet algorithme pour X = 7, Y = 8, Z = 5.
2 - Quelles sont les instructions inutiles ? Justifiez votre réponse.
3 - Quelles sont les instructions qui ne seront jamais exécutées ? Justifiez votre réponse.
4 - Simplifiez cet algorithme en tenant compte de ces remarques.

E - LA STRUCTURE DE CHOIX

Pseudo code	Arbre programmatique
SUIVANT <variable ou expression> Faire | <valeur1> : Séquence1 d'actions | <valeur2> : Séquence2 d'actions | <valeur3> : Séquence3 d'actions | .... | <valeurN> : SéquenceN d'actions | SINON | Séquence d'actions par défaut FINSUIVANT	

La variable (ou l'évaluation de l'expression) est comparée aux différentes constantes (valeur1 à valeurN) et les séquences d'actions entreprises dépendent de cette valeur. Nous disposons d'une séquence d'actions par défaut (facultative) pour le cas où la variable n'est pas égale à aucune des constantes énumérées.

Plusieurs valeurs peuvent entraîner un même traitement. Dans ce cas, il est possible d'énumérer ces valeurs en les séparant par des virgules :

Pseudo code	Arbre programmatique
SUIVANT <variable ou expression> Faire <valeur1> : Séquence1 d'actions <valeur2>, <valeur3> : Séquence2 d'actions SINON Séquence d'actions par défaut FINSUIVANT	

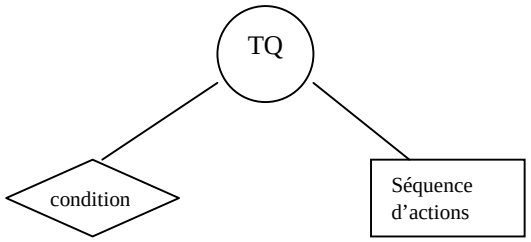
Remarque : La structure de choix permet une présentation plus claire d'un ensemble d'alternatives imbriquées.

Exemple

A partir de la saisie d'un nombre compris entre 0 et 9, écrire un algorithme en pseudo code permettant d'afficher si ce nombre est nul, pair ou impair. Si l'utilisateur entre un nombre inférieur à 0 ou supérieur à 9, l'algorithme affichera le message "valeur non traitée".

Chap 4 - LA STRUCTURE ITERATIVE

A - FORMAT GENERAL

Pseudo code	Arbre programmatique
TANT QUE <condition> Faire | <séquence d'actions> FIN TANT QUE	

Cette structure permet la répétition d'une (ou de plusieurs) action(s) tant qu'une condition est satisfaite.

La condition est testée avant la première exécution de la séquence d'actions définie dans la structure itérative.

si la condition est vérifiée alors
| la séquence d'actions est réalisée
| arrivé à la fin tant que l'algorithme "remonte" au début du tant que
| nouveau test de la condition
sinon
| l'algorithme va directement à la fin du tant que et continue en séquence
finsi

L'utilisation d'une structure itérative est nécessaire lorsque l'action peut-être exécutée de 0 à N fois.

B - EXEMPLE D'APPLICATION

1. Ecrire l'algorithme (arbre programmatique) permettant la saisie de la réponse à la question : "Aimez-vous l'informatique (O/N) ?". Un message sera affiché en cas de mauvaise réponse, et dans ce cas la question sera renouvelée jusqu'à ce que la réponse soit correcte.
2. Faire la trace correspondante à cet algorithme.

METHODE

Nous utiliserons un objet du type caractère recevant la réponse.

Nous distinguons le premier affichage de la question, des suivants. La réponse peut être correcte dès le départ, auquel cas nous n'afficherons pas de message d'erreur. Dans le cas contraire nous répétons l'affichage du message d'erreur et la saisie d'une nouvelle réponse autant de fois que nécessaire.

La condition d'arrêt du traitement est la suivante : réponse = "O" ou réponse = "N".

EXERCICES

Soit A, B de type entier (B > 0)

```
DEBUT
    A ← 1
    ENTRER (B)
    TQ  A <> B  Faire
        A ← A + 1
        B ← B – 1
    FTQ
FIN
```

1. Faire la trace pour B = 3
2. Faire la trace pour B = 4

Ecrire un algorithme en pseudo code permettant d'afficher les entiers non nuls inférieur ou égal à un nombre donné N entier positif. Le contrôle de la saisie est à envisager.

Exemple

Pour N = 5, le programme affichera : 1
2
3
4
5

Ecrire un algorithme (arbre programmatique) permettant de calculer et d'afficher la somme des N premiers nombres entiers positifs. Le contrôle de la saisie est à envisager.

Exemples

Pour N = 5, le programme affichera :	15	(1+2+3+4+5)
Pour N = 1, le programme affichera :	1	
Pour N = 0, le programme affichera :	0	

Ecrire un algorithme (pseudo code) permettant de calculer et d'afficher la factorielle d'un nombre entier positif. Le contrôle de la saisie est à envisager.

Exemples

Pour N = 5, le programme affichera :	120	(1*2*3*4*5)
Pour N = 4, le programme affichera :	24	(1*2*3*4)
Pour N = 1, le programme affichera :	1	
Pour N = 0, le programme affichera :	1	

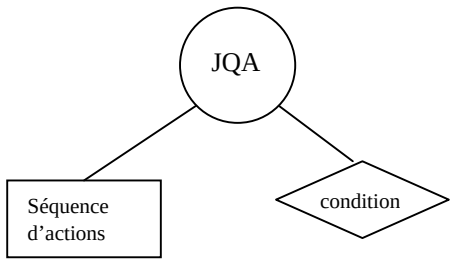
Ecrire un algorithme (arbre programmatique) permettant de rechercher et d'afficher la puissance de 2 immédiatement supérieure ou égale à un nombre donné N non nul et positif.

Exemples

Pour N = 7, le programme affichera :	8	(2x2x2 → 2^3)
Pour N = 15, le programme affichera :	16	(2x2x2x2 → 2^4)
Pour N = 8, le programme affichera :	8	(2x2x2)
Pour N = 1, le programme affichera :	1	(2^0)

Chap 5 - STRUCTURE REPETITIVE

A - FORMAT GENERAL

Pseudo code	Arbre programmatique
REPETER | <séquence d'actions> JUSQUA <condition>	

Cette structure permet la répétition d'une (ou de plusieurs) action(s) jusqu'à ce qu'une condition soit vérifiée.

Elle ressemble à la structure itérative, à cette différence près que la condition exprimée permet l'arrêt du traitement

La condition est testée après l'exécution de la séquence d'actions définie dans la structure répétitive.

```
si la condition n'est vérifiée alors
|     la séquence d'actions est réalisée à nouveau
|     puis nouveau test de la condition
sinon
|     l'algorithme continue en séquence
finsi
```

L'utilisation d'une structure répétitive est nécessaire lorsque l'action peut-être exécutée au moins 1 fois.

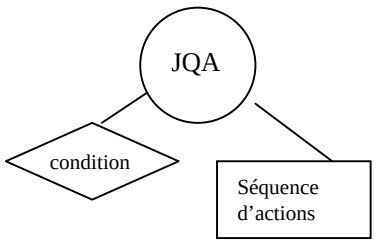
B - EXEMPLE D'APPLICATION

Ecrire avec une structure répétitive, l'algorithme(pseudo code) permettant l'affichage du menu décrit ci-dessous, ainsi que le traitement correspondant.

```
MENU
1 - Somme de deux nombres
2 - Produit de deux nombres
3 - Fin
      Votre choix
```

Chap 6 - LA STRUCTURE POUR

A - FORMAT GENERAL

Pseudo code	Arbre programmatique
POUR <identificateur> = valeur_initiale à valeur_finale FAIRE <séquence d'actions> FIN POUR	

Cette structure permet de répéter une (ou plusieurs) action(s) un nombre connu de fois.
L'identificateur est de type entier. La valeur_initiale et la valeur_finale sont également des valeurs entières ou des variables de type entier

L'utilisation d'une structure Pour est utilisée lorsque l'on connait le nombre de répétitions.

Remarque : Si la valeur initiale est égale à la valeur finale, la séquence d'actions est exécutée **une seule fois.**

B - EXEMPLE D'APPLICATION

Ecrire avec une structure Pour, un algorithme qui affiche les 10 premières valeurs positives non nulles. Faire une trace.