Licence 1

Informatique, Développement d'Application

Cours : Introduction à l'Algorithme

Séquence 1 : Les Structures de contrôles

UVS
UNIVERSITÉ VIRTUELLE DU SÉNÉGAL

Les Structures de contrôles

Lors de la rédaction d'un algorithme, il faut être très explicite et indiquer clairement les conditions et l'ordre dans lesquels les actions envisagées doivent être effectuées.
Les directives d'un algorithme qui sont utilisées à cette fin sont appelées *structures de contrôle*. Elles permettent de décrire la structure logique d'un algorithme et facilitent sa traduction en programme.
Une structure logique dite de test (On parle également de structure alternative) permet de donner des séries d'instructions à effectuer selon que la situation se présente d'une manière ou d'une autre.

I. Structures logiques

Une structure logique dite de test (On parle également de structure alternative) permet de donner des séries d'instructions à effectuer selon que la situation se présente d'une manière ou d'une autre.

Nous avons trois schéma possibles : conditionnel, alternatif et le Choix multiple.

1. Schéma conditionnel

Si (condition)

Alors instruction (s)
Finsi

Si la condition est vraie, alors la suite d'instructions est exécutée.

Exemple

```
Algorithme pair
variable
      n: Entier
Début
      Ecrire("saisir un entier");
      lire(n);
      if(n mod
      2=0) Alors
        Ecrire(n,"est pair")
      Fsi
      if(n mod 2<>0)
      Alors
        Ecrire(n,"est impair")
      Fsi
Fin
```

2. schéma alternatif

Si (condition)

Introduction à l'algorithmique

```
        Alors    instruction (s)
        Sinon
                Autres Instruction(s)
 Finsi
```

Si la condition est vraie, alors la suite d'instructions est exécutée. Par contre si la condition renvoie la valeur fausse, alors la suite d'instructions(autres instruction) est exécutée.

```
Algorithme pair
variable
        n: Entier
Début
        Ecrire("saisir un entier");
        lire(n);
        if(n mod
        2=0) Alors
          Ecrire(n,"est pair")
        Sinon
          Ecrire(n,"est impair")
        Fsi
Fin
```

3. Notion de condition

Un condition est une expression dont la valeur est VRAI ou FAUX. Cela peut donc être :

- **une variable de type booléen**
- **une expression**

Une condition est une comparaison, composée de trois éléments :

- une valeur
- un opérateur de comparaison
- une autre valeur

Les valeurs peuvent être a priori de n'importe quel type (numériques, caractères…). Mais si l'on veut que la comparaison ait un sens, il faut que les deux valeurs de la comparaison soient du même type !

4. **Choix multiple**

```
 Suivant        choix   faire

 < val1 >   :        a1
        .                .       .                .
        .                .
 < valn >       :        an
```

Introduction à l'algorithmique

Sinon : s
FinCas

Le terme **choix** désigne une variable ou une expression de même type que les constantes des ensembles <vali>, i=1, …, n.
ai et s sont des suites d'actions.

II. Structures répétitives (Boucles)

Elles permettent d'effectuer plusieurs fois consécutives la même opération. Nous avons trois types de boucles possibles: **La boucle Pour**, **la boucle Tant que** et **la Boucle Répéter**.

1. La boucle Pour

Si le nombre d'itérations est connu à l'avance, on utilise le schéma suivant :

Syntaxe

```
Pour <compteur> := <valeur initiale> à <valeur finale> faire
        action (à répéter)
```

Un pas (généralement égal à 1) permet de faire passer automatiquement le compteur d'une valeur initiale à une valeur finale, et à chaque étape (ou itération ou tour de boucle), on exécute l'action spécifiée.
Ce pas, appelé aussi incrément, peut être négatif ; cela sous entend que la valeur initiale est supérieure à celle finale.

Exemple :
Voici un algorithme qui affiche les entiers compris entre 1 et 100.

```
variable i : entier
debut
        pour i :=1 à 100 faire
        ecrire(i)
fin
```

2. La boucle TANT QUE

Syntaxe

```
tant que <condition> faire
        <instruction(s)>
ftanque
```

Introduction à l'algorithmique

Elle permet de contrôler la répétition et de garantir son achèvement.

Par exemple, pour afficher dix fois le message 'Bonjour à tous les nouveaux programmeurs !', on peut utiliser la suite d'instructions:

```
Nbfois := 0 ;
Tant que Nbfois < 10 faire
Debut
        Ecrire ('Bonjour à tous les nouveaux programmeurs !')
        Nbfois := Nbfois + 1 ;
Fin
```

Remarques :

1. Dans une boucle **tant que**, la condition contrôlant la répétition est testée avant chaque itération i.e. l'exécution du corps de la boucle.
2. Lorsque le corps de la boucle contient plus d'une instruction (comme dans l'exemple ci-dessus) il faut l'encadrer par les termes *debut* et *fin.*

3. La boucle REPETER

Syntaxe

```
    Répéter
            <instruction(s)>
    jusqu'à <condition>
```

Elle est presque identique à la boucle **tant que** à la différence près que la condition de sortie est testée après exécution des instructions.

Dans une boucle **répéter**, on est sûr que le corps de la boucle sera éxécuté au moins une fois. Donc, on choisira cette structure de contrôle si l'on veut que le corps de la boucle soit exécuté au moins une fois.

Ici, même si le corps de la boucle contient plus d'une instruction, il n'est pas nécessaire de l'encadrer par *debut* et *fin.*

Par exemple, si l'on devait rédiger l'algorithme de l'exercice précédent en utilisant l'instruction repeter :

```
Variable rep :caractère ;
debut
        repeter
                        ecrire('Aimez vous l''algorithmique (o/n) ? ') ;
                        lire(rep) ;
        jusqu'à (rep='o') ou (rep='n');
fin.
```