Cours « Algorithmique »

2ème année Licence

Chapitre 3 (Suite):

Notions d’ « Algorithmique »

Instructions conditionnelles :

L'instruction conditionnelle détermine si le bloc d'instructions (Début … Fin) qui la suit sera exécuté ou non selon la valeur de sa condition.

Une condition est une expression booléenne.

- **Syntaxe :**

Si (condition) **alors**
Bloc d'instructions 1;
sinon
Bloc d'instructions 2 ;
fin si

Si la condition est vérifiée alors le bloc d'instructions 1 est exécuté, et si la condition n'est pas vérifiée c'est le bloc d'instructions 2 qui sera exécuté.

- **Conditionnelle imbriquée :**

il est possible d'imbriquer des blocs de programme les uns dans les autres,

Exemple: affichage des mentions suivant leurs codes.

- 'A' : ''Très Bien''.
- 'B' : ''Bien''.
- 'C' : ''Moyen''.
- 'D' : ''Insuffisant''.

La première solution utilise des instructions conditionnelles simples :

```
algorithme mentions;
var x : char ;
début
        lire(x) ;
        si(x = ‘A’) alors écrire(‘‘Très Bien’’) ;
        si(x = ‘B’) alors écrire(‘‘Bien’’) ;
        si(x = ’C’) alors écrire(‘‘Moyen’’) ;
        si(x = ’D’) alors écrire(‘‘Insuffisant’’) ;
        si((x ≠ ‘A’) et (x ≠ ‘B’) et (x ≠ ’C’) et (x ≠ ’D’)) alors
        écrire(‘‘valeur saisie incorrecte’’) ;
fin .
```

une autre solution :

```
algorithme mentions;
var x : char ;
début
        lire(x) ;
        si(x = ‘A’) alors écrire(‘‘Très Bien’’) ;
        sinon
                si(x = ‘B’) alors écrire(‘‘Bien’’) ;
                sinon
                        si(x = ’C’) alors écrire(‘‘Moyen’’) ;
                        sinon
                                si(x = ’D’) alors écrire(‘‘Insuffisant’’) ;

                                sinon
                                        écrire(‘‘valeur saisie incorrecte’’) ;
                                fin si
                        fin si
                fin si
        fin si
fin
```

Structure à choix multiples (selon) :

La structure de choix multiple permet d'effectuer des actions différentes suivant les valeurs que peut prendre une même variable.

Selon (variable ou expression)
    Cas Valeur1 : action1 ;
    Cas Valeur2 : action2 ;
    Cas Valeur3 :action3 ;
    Cas Sinon action par défaut ;
Fin Selon

Exemple :

```
algorithme mentions ;
var x : char ;
début
        lire(x) ;
        selon (x)
                cas ‘A’ : écrire(‘‘Très Bien’’) ;
                cas ‘B’ : écrire(‘‘Bien’’) ;
                cas ‘C’ : écrire(‘‘Moyen’’) ;
                cas ‘D’ alors écrire(‘‘Insuffisant’’) ;
                cas sinon écrire(‘‘valeur saisie incorrecte’’) ;
        fin selon
Fin.
```

LES BOUCLES

Structures itératives ou répétitives

Définition : les boucles permettent d'exécuter plusieurs fois consécutives un même bloc d'instructions.

La répétition s'effectue tant que la valeur de l'expression booléenne de la boucle (condition de la boucle) est vraie.

La boucle « Tant que … Faire » :

On utilise la boucle tant que quand l'algorithme doit répéter le même traitement tant qu'une condition est vérifiée.

Tant que (condition) **faire**

Bloc d'instructions ;

Fin tq

Phases d'exécution :

Evaluation de la condition:

Evaluation = vrai : Exécuter les actions puis reprise de l'étape précédente.

Evaluation = faux : Arrêt de l'itération et le programme poursuit son exécution après fin tq.

Exemple

```
Algorithme  Somme
var s, i, n : entier;
Début
écrire(‘‘Donner la valeur de n’’) ;
lire(n) ;
        s ← 0 ;
        i ← 1 ;
        tant que (i ≤ n) faire
                s ←s + i ;
                i ←i + 1 ;
        fin tq
écrire("La somme est : ‘’) ;
Écrire (s);
Fin.
```

La variable s est utilisée pour additionner les différentes valeurs. Elle est initialisée à 0.

La variable i est appelée variable d’itération. Elle passe en revue les nombres de 1 à n

Cette instruction permet de faire évoluer i à la valeur suivante.

La boucle « Pour … Faire » :

Cette boucle permet de répéter un ensemble d'actions un nombre connu de fois.

Pour (variable **de** v_initiale **à** v_finale **pas** v_incrément) **faire**

Bloc d'instructions ;

Finpour

// v_initiale = valeur initiale

// v_finale = valeur finale

// v_incrément= valeur d'incrémentation

(variable ← variable + v_incrément)

La valeur de la variable va augmenter (de v_incrément) à partir de la valeur initiale jusqu'à la valeur finale

Exemple 1

```
Algorithme  Somme_entiers;
var s, i, n : entier;
Début
        écrire(‘‘Entrer la valeur de n’’) ;
        lire(n) ;
        S ← 0 ;
        pour (i de 1 à n pas 1) faire
                S ←s + i ;
        finpour
        écrire(''La somme est : '‘);
        écrire (s) ;
Fin .
```

Exemple 2

```
Algorithme  Somme_entiers_pairs;
var s, i, n : entier;
Début
        écrire(‘‘Entrer la valeur de n’’) ;
        lire(n) ;
        S ← 0 ;
        pour (i de 0 à n pas 2) faire
                S ←s + i ;
        finpour
        écrire(''La somme est : '‘);
        écrire (s) ;
Fin
```

Exemple 3

```
Algorithme  Somme_entiers_impairs;
var s, i, n : entier;
Début
        écrire(‘‘Entrer la valeur de n’’) ;
        lire(n) ;
        S ← 0 ;
        pour (i de 1 à n pas 2) faire
                S ←s + i ;
        finpour
        écrire(''La somme est : '‘);
        écrire (s) ;
Fin
```

Procédures et Fonctions :

Une procédure ou une fonction est un algorithme indépendant, l'appel de la fonction déclenche l'exécution de son bloc d'instructions.

Une fonction se termine en retournant ou non une valeur.

Dans le cas où aucune valeur n'est retournée on parle de procédure.

L'intérêt de l'utilisation des procédures et fonctions peut être résumé dans les points suivants

Intérêt des procédures et des fonctions :

- Le code des algorithmes devient plus simple, plus clair et plus court.

- Une seule modification dans la procédure ou la fonction sera automatiquement répercutée sur tous les algorithmes qui l'utilisent.

- L'utilisation dans plusieurs algorithmes différents permet de réutiliser son travail et de gagner du temps.

Structure d'une fonction :

```
Fonction nomDeLaFonction (paramètres) : typeDeRetour;
   //partie déclaration ;
début
    Instructions ;
Fin
```

Structure d'une procédure :

```
Procédure nomDeLaProcédure (liste des paramètres);
     //partie déclaration ;
début
     Instructions ;
Fin
```

Les fichiers :

Définition d'un fichier : Un fichier est un regroupement logique de données mémorisées sur un support permanent afin de permettre une réutilisation ultérieure des informations qu'il contient

Structuration des données dans un fichier

Fichiers NON structurés : Ce sont les fichiers constitués d'un texte dont la structure n'est pas déterminée (documents texte, codes sources, etc.).

Fichiers structurés : Les fichiers structurés sont composés d'un ensemble de données élémentaires identifiables de manière précise au sein du fichier.

Structure par un balisage : Chaque donnée élémentaire est encadrée par un balisage. Exemples : XML, HTML, etc..

Structure par enregistrements : Un enregistrement est un bloc de données élémentaires qui décrit une entité.
Par exemple, au sein d'un fichier « Clients », un enregistrement correspond aux données relatives à un client.

Organisation du placement des données

L'organisation des données dans un fichier détermine comment seront placés chacun des enregistrements.

Organisation séquentielle :

Les enregistrements sont mémorisés consécutivement dans l'ordre de leur entrée et peuvent seulement être lus dans cet ordre.

Organisation calculée

Les enregistrements sont placés à une position précise du fichier. Cette position est selon un algorithme de placement appliqué à une clef (on parle de placement aléatoire).

Organisation indexée

Dans ce cas au moins deux fichiers sont nécessaires (Fichier de données et Fichier de la table d'index)

Modes d'accès aux enregistrements d'un fichier

Accès séquentiel : consiste à parcourir, dans l'ordre dans lequel ils sont stockés, les enregistrements d'un fichier. L'accès séquentiel est effectué dans un seul « sens » : on parcourt les enregistrements du début à la fin, sans retour en arrière.

Accès direct : permet de retrouver directement un enregistrement ou un bloc d'enregistrements dans le fichier, l'accès se fait :

- Soit grâce à un numéro d'ordre de placement ;
- Soit grâce à une clef.

Modes d'accès aux enregistrements d'un fichier

Accès séquentiel : consiste à parcourir, dans l'ordre dans lequel ils sont stockés, les enregistrements d'un fichier. L'accès séquentiel est effectué dans un seul « sens » : on parcourt les enregistrements du début à la fin, sans retour en arrière.

Accès direct : permet de retrouver directement un enregistrement ou un bloc d'enregistrements dans le fichier, l'accès se fait :

- Soit grâce à un numéro d'ordre de placement ;
- Soit grâce à une clef.

Modes d'ouverture d'un fichier

Le mode d'ouverture d'un fichier détermine les actions autorisées sur le fichier

Mode d'ouverture	Actions autorisée
Lecture	En mode lecture, seules seront autorisés l'accès aux données, sans modification possible du contenu
Ecriture	En mode écriture, le fichier est vidé de son contenu, et ne sera possible que l'écriture de nouveaux enregistrements
Ajout	En mode ajout, les données présentes dans le fichier seront préservées, et il sera possible d'ajouter de nouveaux enregistrements
Lecture/ Ecriture	En mode lecture/écriture, les données présentes dans le fichier seront préservées, et il sera possible de modifier le contenu de certains enregistrements.