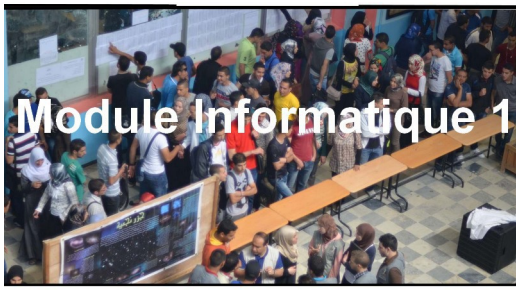
1.1

Informatique 01

Informatique 1

Formation des spécialistes en e-learning
Promotion 2015-2016



année universitaire 2015/2016
Bendiab Keltoum

Table des matières

Objectifs

La programmation est une activité fondamentale en informatique. La programmation peut être vue comme l'art de déterminer **un algorithme** (**une démarche**) pour résoudre un problème et d'exprimer cet algorithme au moyen d'un langage de programmation.

Ce module permet de savoir transcrire les différentes étapes de résolution d'un problème sous forme d'algorithme, de façon structurée et indépendante de toute contrainte matérielle ou logicielle.

Les **compétences** acquises sont :

- comprendre et analyser un algorithme préexistant ;
- analyser la situation : identifier les données d'entrée, de sortie, le traitement... ;
- mettre au point une solution algorithmique : comment écrire un algorithme en langage courant en respectant un code, identifier les boucles, les tests, des opérations d'écriture, d'affichage... ;
- valider la solution algorithmique par des jeux d'essais simples avec le langage pascale;

Introduction

le cours

Chapitre 5 : Enregistrements et fichiers

• L'objectif principal de ce chapitre est l'acquisition de la notion d'enregistrement et de fichier.
• Mettre au point une solution algorithmique en utilisant les enregistrements et les fichiers
• Valider la solution algorithmique par des jeux d'essais simples.

A. 1). Les enregistrements

1. 1). Problématique

Le type tableau nous a permis de définir une structure composée de plusieurs éléments, cette structure nous permettrait de réunir les éléments de même type. Mais si nous voulons regrouper les informations n'ayant pas nécessairement le même type au sein d'une même structure, par exemple: les informations concernant un étudiant.

On veut écrire un algorithme permettant de : Saisir les informations de vente de 100 produits et de déterminer le produit le plus bénéfique, chaque produit est caractériser par :

- Carte(**Type entier**)
- Nom(**Type caractères**)
- Notes [1..9](**Type réel**)
- moyenne (**Type réel**)
- Résultat (**Type booleén**)

La première question qui se pose est quelle est la structure de donnée nécessaire pour stocker les informations de chaque étudiant?

La seule solution possible est d'utiliser plusieurs tableaux pour stocker les informations de chaque produit.

```
Algorithme : etudiants
Variables
i :entier
carte[1..100] : tableau d'entier
nom[1..100] : tableau de chaine de caractaires
not1[1..100],...,note9[1..10] : tableau de réel
moyenne[1..100] : tableau de réel
résultats[1..100] : tableau de booléen
Début
/*Lecture des informations de 100 étudiants*/
pour i allant de 1 à 100 faire
Lire(carte[i])
Lire(nom[i])
Lire(note1[i],....,note9[i])
Lire(moyenne[i])
Lire(resultat[i])
Fin pour
Fin
```

Cette solution est non fiable pour résoudre ce type de problème, Une nouvelle structure appelée **enregistrement** ou **article** ou **record** est plus adaptée pour représenter ce type d'information. c'est une (**structure de donnée hétérogènes**) qui permet de regrouper à la fois les données numériques (notes,carte, moyenne, etc.) et les données alphanumériques (nom, adresses,etc.) dans la même variable.

2. 2). Définitions

Définition : Définition 1

Un **enregistrement** est un objet composé **statique** et **hétérogène** c'est-à-dire qui renferme plusieurs informations qui peuvent être de type différent. un enregistrement est défini par un ensemble de données ou élément encore appelé **champ**. Les champs sont les données élémentaires ou composées et peuvent être de type différent.

Définition : Définition 2

Une structure de données (au sens **enregistrement**) permet de désigner sous un seul nom un ensemble de valeurs pouvant être de type différent. un éléments d'un enregistrement est appelé **champs**).

3. 3). Déclaration de l'enregistrement

Syntaxe : 3.1) Déclaration de nouveau type d'enregistrement

Type
nom_de_type = **enregistrement**
champ 1 : Type 1
champ 2 : Type 2
.....
champ n : Type n
Fin nom_type

Syntaxes :
(notation inspirée du Pascal)

```
Type
   nom_type = enregistrement
                  | nom_champ1: type_champ1
                  |    ...
                  |  nom_champn: type_champn
              finenreg
```

Exemple:

```
Type
   tpersonne = enregistrement
                  | nom : chaîne
                  | prénom : chaine
                  | âge : entier
              finenreg
```

koko

Syntaxe : 3.2) Déclaration d'une variable enregistrement

ident_variable : de **nom_de_type**

Rappel : Manipulation d'un enregistrement

- En général on manipule un enregistrement champ par champ.
- On accède à un champ de l'enregistrement en indiquant le nom de l'enregistrement suivi du nom du champ. Les deux sont séparés par un point.
  - **Exemple:**
    Etudiant.matricule
    Etudiant.nom
- Si un champ de l'enregistrement est d'un type donné alors on peut réaliser sur ce champ toutes les opérations réalisables avec les objets de ce type.

Exemple : Exemple d'application de valeur

Algorithme de saisie des données concernant les personnes (nom, prénom, âge)pers1 et pers2, puis affichage de la différence d'âge entre ces deux personnes

```
Algorithme Exemple
Type
tpersonne : enregistrement
nom : chaîne
prénom : chaîne
```

```
âge : entier
Fintpersonne
Var
pers1, pers2 : tpersonne
Début
Ecrire( "Entrez le nom puis l'age de la personne 1")
lire(pers1.nom, pers1.age) // il est impossible d'écrire
lire(pers1)
ecrire("Entrez le nom puis l'âge de la personne 2")
lire(pers2.nom, pers2.age)
Ecrire(La différence d'âge entre ", pers1.nom, "et",
pers2.nom, " est de :")
Si pers1.age > pers2.age Alors
Ecrire(pers1.age – pers2.age, " ans ")
Sinon
Ecrire(pers2.age – pers1.age, " ans ")
FinSi
Fin
```

Exemple : Exemple 2 : Déclaration et manipulation des enregistrement

```
algorithme monAlgorithme
     // déclaration d'un enregistrement
     enregistrement Personne
          chaine nom;
          chaine prenom;
          entier age;
          réel taille;
     finenregistrement
     ...
     Personne p1, p2;
début
     // Initialisation d'un enregistrement
     p1.nom <- "Duchmol";
     p1.prenom <- "Robert";
     p1.age <- 24;
     p1.taille <- 1.80;
     ...
fin
```

enr22

Remarque : Remarques

- Les types des champs peuvent être prédéfinis ou définis par l'utilisateur
- Un champ a exactement les mêmes propriétés qu'une variable de même type
- Le champ d'une variable enregistrement peut être lui même un enregistrement

4. 4). Opérations sur les enregistrements

Fondamental : La structure avec...faire

Pour simplifier la licteur (écriture) et éviter l'utilisation répétée de nom de l'enregistrement. le nom de champs, nous pouvons utiliser l'instruction avec...faire

```
Avec nom_variable_ enregistrement faire
action chhamps1
action chhamps2
....
action chhampsn
Fin avec
```

Exemple : Exemple d'utilisation de avec...faire

utilisation de avec...faire pour lire la variable Etud1 de type d'enregistrement Étudiant

Avec per1 Faire

Lire(nom, prénom, age)

Fin avec

5. 5). Vecteur d'enregistrements

Il est bien entendu possible de regrouper sous un même nom, plusieurs structures de même type pour former un tableau d'enregistrements.

Exemple : Écrire un algorithme qui permet d'afficher les informations de 1500 étudiants (nom, prénom, notes de 9 modules), calcule la moyenne générale et affiche le résultat (admis ou ajournée) pour chaque étudiant

```
Algorithme : PV_etudiants
/*Déclaration de type d'enregistrement Etudiant*/
Type
Etudiant=enregistrement
Nom,prénom :chaine
Notes :tabeau[1 ..9] de réel
Moy-générale :réel
Resutat :(admis,ajourné)
Fin Etudiant
/*Déclaration de variables*/
Variables
i, j : entier
Etud[1..1500]: Tableau de type Etudiant
Début
/*Saisie des informations des étudiants*/
pour i allant de 1 à 1500 faire
Avec Etud[i] Faire
Lire(nom, prénom)
pour j allant de 1 à 9 faire
lire(Note[j])
Fin pour
Fin avec
```

```
Fin pour
/*calcule de la moyenne*/
pour i allant de 1 à 1500 faire
somme<--0
pour j allant de 1 à 9 faire
somme<--somme+Etud[i].note[j]
Fin pour
Etud[i].moyenne_gene<--somme/9
si (Etud[i].moyenne_gene>=10) alors
Etud[i].résultats<--"Admis(e)"
sinon
Etud[i].résultats<--"Ajourné(e)"
Fin si
Fin Pour
/*Affichage des résultats*/
pour i allant de 1 à 1500 faire
Avec Etud[i] Faire
Ecrire (nom)
Ecrire (prénom)
pour j allant de 1 à 9 faire
Ecrire (note[j])
Fin pour
Ecrire (moyen_gene)
Ecrire(resultats)
Fin avec
Fin pour
Fin
```

Exercice (Vecteur d'enregistrement)

Nous voulons constituer une base de données des enseignements. Chaque enseignement est caractérisé par:

- Le code.
- Le nom.
- Le prénom.
- Le grade.
- Le sexe.

Les enseignants seront stockés dans un vecteur

1. Donner une déclaration du type enseignant.
2. Ecrire une procédure **sui** permet de créer un vecteur de n enseignant.
3. Ecrire une procédure qui prend en paramètre le vecteur d'enseignement si cet enseignement existe ou pas.
4. Ecrire une procédure qui affiche la liste des enseignements féminins

```
{Déclaration de type d'enregistrement}
type enseignant=enregistrement
code : chaîne
nom : chaîne
prénom : chaîne
grade : chaîne
sexe : caractère
fin enregistrement
{Déclaration de vecteur d'enregistrement}
vecteur = tableau[1..100] d'enseignant
{procédure pour saisir le vecteur d'enregistrement}
procédure: créer_enseignant(v: vecteur, n: entier)
```

```
début
pour i=1 à n faire lire(V[i].code)
lire(V[i].nom)
lire(V[i].prénom)
lire(V[i].grade)
lire(V[i].sexe)
fin pour
fin
procédure: recherche(V: vecteur, n: entier, codeE: chaîne)
var i: entier
début
i←1
tant que i<= et codeE≠V[i].code faire
i←i+1
fin tant que
si codeE=V[i].code alors
écrire('Cet enseignant est dans la case N°: ',i)
sinon
écrire('Cet enseignant n'existe pas')
fin si
fin
{une procédure qui affiche la liste des enseignant
féminins}
procédure affiche_femini(V: vecteur, n:entier)
var i: entier
début
pour i=1 à n faire
si V[i].sexe = 'F' alors
écrire(V[i].code)
écrire(V[i].nom)
écrire(V[i].prénom)
écrire(V[i].grade)
fin si
fin pour
fin
```

B. 2). Les fichiers

2.1) Problématique

Jusqu'à présent, les informations utilisées dans nos programmes ne pouvaient provenir que de deux sources : soit elles étaient inclues dans l'algorithme lui-même, par le programmeur, soit elles étaient entrées en cours de route par l'utilisateur. Mais évidemment, cela ne suffit pas à combler les besoins réels d'un algorithme

Pour combler ce manque, la solution consiste alors à faire recours à d'autre structure qui s'appelle **fichier**. Ils servent à stocker des informations de manière permanente, entre deux exécutions d'un programme. les fichiers sont stockés sur des périphériques à mémoire de masse (disquette, disque dur, CD Rom...).

Définition : 2.2). Définition 1

Un **fichier** est un ensemble organisé de données ayant le même type, stocké sur une mémoire de masse (disque dur, CD-Rom, bande,...). Un fichier a généralement comme attributs : un nom ; un chemin d'accès ; une taille mesurée en octets ; une date de création et une de dernière modification. Un fichier sert à conserver des

données de manière permanente.

Deux grandes catégories de fichiers peuvent exister :

1. Fichier texte
2. Fichier binaire

1. Fichier texte

Ce type de fichier est couramment utilisé dès lors que l'on doit stocker des informations pouvant être assimilées à une base de données (données structurées).

2. Fichier binaire

Un fichier binaire est par contre contient des données non textuelles. C'est un fichier qui ne possède pas de structure de lignes (d'enregistrement). Il est constitué d'une suite de bits auxquels seuls des programmes adapté peuvent donner un sens.

	Fichiers Texte	Fichiers Binaires
On les utilise pour stocker...	De données structurées	tout, y compris des données structurées
Ils sont structurés sous forme de...	lignes (enregistrements)	Ils n'ont pas de structure apparente. Ce sont des octets écrits à la suite les uns des autres.
Les données y sont écrites...	exclusivement en tant que caractères	comme en mémoire vive
Les enregistrements sont eux-mêmes structurés...	au choix, avec un séparateur ou en champs de largeur fixe	en champs de largeur fixe, s'il s'agit d'un fichier codant des enregistrements
Lisibilité	Le fichier est lisible clairement avec n'importe quel éditeur de texte	Le fichier a l'apparence d'une suite d'octets illisibles
Lecture du fichier	On ne peut lire le fichier que ligne par ligne	On peut lire les octets de son choix (y compris la totalité du fichier d'un coup)

immmm

2.3) Organisation des fichiers

L'organisation d'un fichier désigne le mode d'implémentation des informations et des enregistrements dans ce fichier et fournit les propriétés d'accès.

- **Organisation séquentielle** : l'accès aux informations se fait en parcourant les enregistrements les uns après les autres.
- **Organisation relative** (dite aussi directe) : les enregistrements sont identifiés par un numéro d'ordre.

Fondamental : 2.4) Les modes d'accès aux fichiers

En informatique, nous distinguons deux types d'accès aux données d'un fichier :

- **Accès séquentiel** : pour accéder à l'information d'ordre n, on doit passer par les (n-1) informations précédentes.
- **Accès direct** : On accède directement à l'information désirée, en précisant le numéro d'emplacement (le numéro d'ordre) de cette information.

Remarque

Tout fichier peut être utilisé avec l'un des deux types d'accès. Donc le choix de type d'accès dans un fichier ne concerne pas le fichier lui-même mais concerne la manière dont il va être traité par la machine (le choix de type d'accès se fait seulement dans le programme).

C. 2.4) Les Fichiers à accès séquentiel

2.4.1) Présentation

Un fichier est dit à accès séquentiel (ou fichier séquentiel) si l'accès à son n^{ième} information nécessite le passage par les (n-1) informations précédentes

Syntaxe : 2.4.2) Déclaration

/*Déclaration de nouveau type de fichier*/

Type

nom_fichier =**Fichier** de **type**

/*Déclaration de la variable fichier*/

Variables

Nom_logique : de type nom_fichier

Remarque

- Comme on a déjà dit un fichier doit être enregistré sur un support externe, donc ce fichier doit avoir un nom et de préférence une **extension**. Ce nom est appelé le **nom externe** (ou le **nom physique**). Les fichiers de données (data) ont une extension .dat, .fch
- Le nom de l'objet déclaré dans le tableau des objets comme nom de fichier est le **nom interne** du fichier (ou aussi le **nom logique**). C'est le nom utilisé dans les instructions du programme.

Définition

3.2) Traitements sur les fichiers

Exemple : Ecrire un algorithme permettant de créer et remplir un fichier texte avec la chaîne "Bonjour" dix fois

Pour utiliser un fichier dans un algorithme il faut :

- Déclarer le ficher (créer la version mémoire centrale de fichier)
- Associer le nom logique au nom physique du fichier (Commande associer)
- Ouvrir le fichier (Commande ouvrir)
- Effectuer les traitement
- et enfin fermer le fichier (Fermer)

```
Solution
Algorithme : ficier_text
variables
i : entier
s : chaine de caractère
Fich : fichier de chiane de caratère
```

```
Debut
s<--"Bonjour"
Associer(Fich, 'C :\monfichier.txt')
ouvrir(fich, 'Ecriture')
{écriture d'une variable dans un fichier}
pour i=1 jusqu'à 10 faire
Ecrire(Fich, S)
Fin pour
Fermer (Fich)
{Lecture à partir de fichier}
ouvrir(Fich, 'lecture')
Tant que (non(eof(fich)) faire
Lire(Fich, S)
Ecrire(S)
Fin tant que
Fermer(Fich)
Fin
```

Bibliographie

[01] support du cours "informatique 1" Mezhoud Nassima, MCA, département électronique, université des frères Mentouri constantine

[01] Livre "ALGORITHMIQUE ET PROGRAMMATION POUR NON-MATHEUX", 2008 Christophe Darmangeat.

[02] Initiation `a l'algorithmique", Jaques Tisseau