Algorithmique 2

Les Enregistrements

Tarek Boutefara

2026-02-07

Table of Contents

1. Introduction . . . 1
2. Définition . . . 2
3. Différences Clés entre Tableau et Enregistrement . . . 3
4. Déclaration d'un Enregistrement . . . 4
5. Manipulation des Enregistrements (Accès aux Champs) . . . 5
6. Combinaison des Structures : Le Tableau d'Enregistrements . . . 7
7. Exemples d'applications . . . 9
    7.1. Système de Gestion de Scolarité . . . 9
    7.2. Gestion d'un Stock de Pharmacie . . . 10
    7.3. Système de Réservation de Vols . . . 11
8. Le Lien entre Chaînes de Caractères et Enregistrements : Le "Parsing" . . . 13
    8.1. Des données brutes vers les données structurées . . . 13
    8.2. Méthodologie de conversion (Algorithme de Parsing) . . . 13
9. Conclusion . . . 14

Chapitre 1. Introduction

Jusqu'à présent, l'apprentissage s'est focalisé sur la manipulation de données de même nature à travers les types simples (Entier, Réel, Chaîne) et les types structurés homogènes (Tableaux). Cependant, la réalité des systèmes d'information impose souvent de manipuler des entités complexes composées d'informations de types différents.

Pour représenter un Étudiant, par exemple, il est nécessaire de lier son Nom (chaîne), son Âge (entier) et sa Moyenne (réel). Utiliser trois tableaux distincts pour gérer ces informations serait non seulement fastidieux, mais aussi source d'erreurs lors des tris ou des suppressions.

L'Enregistrement apporte une solution élégante à cette problématique. Il permet de définir un nouveau type de donnée, dit hétérogène, capable de regrouper sous un seul nom plusieurs variables (appelées Champs) ayant des types différents. Cette structure est le pilier de la modélisation de données : elle permet de passer de la simple gestion de listes à la création de véritables "fiches" d'information, offrant ainsi une organisation du code plus proche de la logique du monde réel.

Chapitre 2. Définition

Nous avons vu que le tableau est une structure **homogène** (tous les éléments sont du même type). Cependant, de nombreux problèmes nécessitent de regrouper des informations de **types différents**.

Par exemple, pour décrire un "Étudiant", nous avons besoin de son Nom (Chaîne), de son Prénom (Chaîne), de son Âge (Entier) et de sa Moyenne (Réel). Il n'est pas possible de stocker ces informations dans un seul tableau.

L'**enregistrement** (parfois appelé `Structure` dans certains langages) est une structure de données conçue pour répondre à ce besoin. Un enregistrement est un ensemble de **champs** (ou **membres**) qui permet de regrouper sous un **unique nom** des données hétérogènes (de types différents) mais logiquement liées.

Chapitre 3. Différences Clés entre Tableau et Enregistrement

Critère	Tableau (Vecteur/Matrice)	Enregistrement (Structure)
Homogénéité	**Homogène**	**Hétérogène**
Contenu	Tous les éléments doivent être du **même type**.	Chaque champ peut avoir son **propre type**.
Accès	Par **indice** numérique (ex: `T[5]`).	Par **nom de champ** (ex: `E.Nom`).
Traitement	Conçu pour le **traitement répétitif** (parcours avec des boucles).	Conçu pour le **traitement logique** (manipulation de champs spécifiques).

D'un point de vue conceptuel, le tableau est idéal pour une **collection** d'items identiques (ex: une liste de notes), tandis que l'enregistrement est idéal pour **décrire** un objet complexe (ex: une personne, un produit).

Chapitre 4. Déclaration d'un Enregistrement

La déclaration d'un enregistrement se fait presque toujours via un nouveau **type** personnalisé dans la section `Type`. On utilise des mots-clés (comme `Structure` ou `Enregistrement`) pour définir la liste des champs qui composent cet enregistrement. Chaque cahmps est définit comme une simple variable (nom suivi par ":" suivi par le type).

- `Type`
  - `FicheEtudiant = Enregistrement`
    - `Nom : Chaîne de caractères`
    - `Prenom : Chaîne de caractères`
    - `Age : Entier`
    - `Moyenne : Réel`
  - `FinEnregistrement`
- `Variables`
  - `E1 : FicheEtudiant`
  - `E2 : FicheEtudiant`

Chapitre 5. Manipulation des Enregistrements (Accès aux Champs)

On ne manipule pas l'enregistrement dans son ensemble. Pour accéder à une information spécifique à l'intérieur de la variable, on utilise l'**opérateur "point" (.)**, suivi du nom du champ désiré.

`VariableEnregistrement.NomDuChamp`

Cette notation (ex: `E1.Age`) se comporte exactement comme une variable simple du type du champ (ici, un `Entier`). On peut donc l'utiliser dans des affectations, des lectures ou des écritures.

Exemple :

Voici un algorithme qui remplit (lit) les informations d'un étudiant, puis les affiche (écrit).

Lecture des champs d'un enregistrement

```
Algorithme GestionFicheEtudiant

Type
    FicheEtudiant = Enregistrement
        Nom : Chaîne de caractères
        Prenom : Chaîne de caractères
        Age : Entier
    FinEnregistrement

Variables
    UnEtudiant : FicheEtudiant

Début

    Ecrire("--- Saisie de la fiche étudiant ---")

    Ecrire("Veuillez saisir le Nom : ")
    Lire(UnEtudiant.Nom)

    Ecrire("Veuillez saisir le Prénom : ")
    Lire(UnEtudiant.Prenom)

    Ecrire("Veuillez saisir l'Âge : ")
    Lire(UnEtudiant.Age)

    Ecrire("--- Fin de la saisie ---")

    Ecrire("--- Affichage de la fiche ---")

    Ecrire("Nom complet : ", UnEtudiant.Prenom, " ", UnEtudiant.Nom)
    Ecrire("Âge : ", UnEtudiant.Age, " ans.")
```

```
Fin.
```

Chapitre 6. Combinaison des Structures : Le Tableau d'Enregistrements

La véritable efficacité en gestion de données est atteinte lorsque nous combinons les structures. L'exemple le plus courant et le plus puissant est le **Tableau d'Enregistrements**.

Reprenons notre `FicheEtudiant` (un enregistrement) et notre `Tableau` (une collection). En déclarant un `Tableau [1..30] de FicheEtudiant`, nous ne créons pas un tableau de Noms et un tableau d'Âges ; nous créons une structure unique qui réserve 30 "emplacements", où **chaque emplacement contient une `FicheEtudiant` complète** (avec son Nom, son Prénom, son Âge, etc.).

Nous obtenons le meilleur des deux mondes :

1. La capacité du **Tableau** à gérer une collection avec une boucle (parcourir les étudiants de 1 à 30).
2. La capacité de l'**Enregistrement** à garder les données d'un étudiant logiquement groupées.

Pour manipuler cette structure, nous avons besoin des deux syntaxes : l'indice `[i]` pour sélectionner **quel étudiant** dans le tableau, et le point `.Champ` pour sélectionner **quelle information** de cet étudiant. Par exemple : `Classe[i].Nom` signifie "prendre le Nom de la FicheEtudiant qui se trouve à l'indice 'i' du tableau Classe".

Exemple :

Voici un algorithme complet qui gère une petite classe (3 étudiants) en utilisant un tableau d'enregistrements. Il lit les informations de la classe, puis affiche la liste des étudiants avec leur moyenne.

Exemple de combinaison des tableaux et des enregistrements

```
Algorithme GestionClasse

Constantes
    NB_ETUDIANTS = 3

Type
    FicheEtudiant = Enregistrement
        Nom : Chaîne de caractères
        Moyenne : Réel
    FinEnregistrement

   TypeClasse = Tableau [1..NB_ETUDIANTS] de FicheEtudiant

Variables
    MaClasse : TypeClasse
    i : Entier

Début
```

```
Ecrire("--- Saisie des informations de la classe ---")

   Pour i AllantDe 1 A NB_ETUDIANTS Faire
        Ecrire("Saisie pour l'étudiant N°", i)
        Ecrire(" Nom : ")
        Lire(MaClasse[i].Nom)
        Ecrire(" Moyenne : ")
        Lire(MaClasse[i].Moyenne)
    FinPour

   Ecrire("--- Affichage de la liste de la classe ---")

    Pour i AllantDe 1 A NB_ETUDIANTS Faire
       Ecrire("Étudiant : ", MaClasse[i].Nom, " - Moyenne : ", MaClasse[i].Moyenne)
    FinPour

Fin.
```

Chapitre 7. Exemples d'applications

7.1. Système de Gestion de Scolarité

7.1.1. Description de la réalité

Dans une université, nous ne gérons pas des "noms" ou des "notes" de manière isolée. Nous gérons des individus. Un étudiant possède une identité propre (nom, prénom, matricule), une date de naissance, et une performance académique (moyenne).

De plus, ces étudiants sont regroupés dans des sections ou des groupes (ex: Groupe 01 de la Licence Informatique). Un groupe possède des caractéristiques : un nom, une spécialité, et la liste des étudiants qui le composent.

7.1.2. Analyse et Modélisation

Pour représenter cette réalité, nous devons procéder par emboîtement (structures imbriquées) :

- La Date : Elle est composée de trois entiers. C'est une structure réutilisable.
- L'Etudiant : Il regroupe des informations disparates (chaînes, entiers, réels) et utilise la structure Date.
- Le Groupe : C'est une structure qui contient un Tableau d'enregistrements Etudiant.

7.1.3. Définition et Déclaration Algorithmique

Définition des Types

```
Type
    { Structure de base pour les dates }
    Date = Enregistrement
        Jour : Entier
        Mois : Entier
        Annee : Entier
    FinEnregistrement

    { Structure représentant un étudiant }
    Etudiant = Enregistrement
        Matricule : Chaîne
        Nom : Chaîne
        Prenom : Chaîne
        DateNaissance : Date  { Utilisation d'un enregistrement dans un autre }
        Moyenne : Réel
    FinEnregistrement

    { Structure représentant un groupe complet }
    Groupe = Enregistrement
        NomGroupe : Chaîne  { ex: "G01" }
```

```
        Specialite : Chaîne { ex: "Informatique" }
        ListeEtudiants : Tableau [1..40] de Etudiant { Tableau d'enregistrements }
        Effectif : Entier   { Nombre réel d'étudiants inscrits }
    FinEnregistrement
```

Déclaration des Variables

```
Variables
    MonGroupe : T_Groupe
    MeilleurEtudiant : T_Etudiant
```

Exemple de Manipulation (Accès aux champs)

L'accès suivant :

```
MonGroupe.ListeEtudiants[3].DateNaissance.Mois
```

Est exécuté comme suit :

- MonGroupe : On cible le groupe.
- .ListeEtudiants[3] : On cible le 3ème étudiant du tableau.
- .DateNaissance : On entre dans sa fiche de naissance.
- .Mois : On accède enfin à la donnée numérique souhaitée.

7.2. Gestion d'un Stock de Pharmacie

7.2.1. Description de la réalité

Une pharmacie doit gérer un inventaire complexe de médicaments. Chaque produit ne se résume pas à son nom. Il possède un code-barres unique, appartient à un laboratoire précis et a un prix de vente. Deux aspects sont critiques : la sécurité (date de péremption et nécessité ou non d'une ordonnance) et la logistique (quantité disponible en rayon).

7.2.2. Analyse et Modélisation

Pour modéliser un stock, nous décomposons les informations ainsi :

- Composant réutilisable : La Date (déjà définie précédemment).
- L'unité : Le Medicament qui regroupe les caractéristiques techniques et commerciales.
- Le système global : La Pharmacie qui contient l'ensemble des références.

7.2.3. Définition et Déclaration Algorithmique

```
Type
```

```
    Date = Enregistrement
        Jour, Mois, Annee : Entier
    FinEnregistrement

    Medicament = Enregistrement
        CodeBarre : Chaîne      { Ex: "6131234567890" }
        Designation : Chaîne    { Ex: "Paracétamol 500mg" }
        Laboratoire : Chaîne    { Ex: "Saidal" }
        Prix : Réel
        DateExpiration : Date
        QuantiteStock : Entier
        BesoinOrdonnance : Booléen
    FinEnregistrement

    StockPharmacie = Enregistrement
        NomOfficine : Chaîne
        Inventaire : Tableau [1..2000] de Medicament
        NbReferences : Entier  { Nombre de types de médicaments différents }
    FinEnregistrement

Variables
    MaPharmacie : T_StockPharmacie
```

Pour vérifier si le premier médicament du stock est périmé :

```
Si (MaPharmacie.Inventaire[1].DateExpiration.Annee < 2026) Alors ...
```

7.3. Système de Réservation de Vols

7.3.1. Description de la réalité

Un système de compagnie aérienne gère des Vols. Chaque vol est défini par un numéro unique, une provenance et une destination. Un vol est aussi lié à un calendrier précis (date et heure). Enfin, un vol n'est pas vide : il transporte des Passagers, chacun ayant un nom, un numéro de passeport et un numéro de siège assigné.

7.3.2. Analyse et Modélisation

Ici, la structure est encore plus imbriquée :

- Le temps : Une structure Horaire (Heure, Minute).
- L'individu : Un enregistrement Passager.
- L'objet principal : Le Vol, qui contient à la fois des informations simples (numéro, villes), des structures temporelles (date, heure) et une collection d'individus (tableau de passagers).

7.3.3. Définition et Déclaration Algorithmique

```
Type
    Horaire = Enregistrement
        Heure, Minute : Entier
    FinEnregistrement

    Passager = Enregistrement
        NomComplet : Chaîne
        NumPassport : Chaîne
        NumSiege : Entier
    FinEnregistrement

    Vol = Enregistrement
        NumVol : Chaîne          { Ex: "AH1020" }
        VilleDepart : Chaîne     { Ex: "Constantine" }
        VilleArrivee : Chaîne    { Ex: "Marseille" }
        DateVol : Date
        HeureDepart : Horaire
        ListePassagers : Tableau [1..250] de Passager
        NbPassagersInscrits : Entier
    FinEnregistrement

Variables
    VolAlgerParis : Vol
```

Chapitre 8. Le Lien entre Chaînes de Caractères et Enregistrements : Le "Parsing"

Dans un système informatique réel, les données n'arrivent que rarement sous une forme directement structurée. Elles sont souvent reçues, lues ou scannées sous la forme d'une chaîne de caractères brute (appelée flux de données).

L'opération consistant à analyser cette chaîne pour en extraire les informations et les ranger dans les champs d'un enregistrement s'appelle le Parsing (ou analyse syntaxique).

8.1. Des données brutes vers les données structurées

Une donnée brute est souvent une ligne de texte où les informations sont séparées par un caractère spécial (le délimiteur), comme un point-virgule (;), une virgule (,) ou un espace.

- Exemple : Une douchette de pharmacie scanne un produit et génère la chaîne suivante :

```
S_Brute <- "ASPIRINE;613002;120.50"
```

À ce stade, l'ordinateur ne peut pas effectuer de calcul sur le prix ni de recherche sur le nom, car tout est mélangé dans une seule variable de type Chaîne. Le passage vers un Enregistrement est alors indispensable.

8.2. Méthodologie de conversion (Algorithme de Parsing)

Le processus de parsing repose sur l'utilisation combinée des fonctions de manipulation de chaînes et des procédures de conversion.

- Repérage : Utilisation de Position pour trouver les délimiteurs (ex: le ;).
- Extraction : Utilisation de SousChaine pour isoler chaque segment.
- Conversion : Utilisation de Val pour transformer les segments numériques en entiers ou réels.
- Affectation : Stockage des résultats dans les champs correspondants de l'enregistrement.

Chapitre 9. Conclusion

En conclusion, la structure Enregistrement marque une étape décisive dans l'évolution de la pensée algorithmique : le passage de la manipulation de variables éparpillées à la modélisation d'objets cohérents.

L'intérêt des enregistrements dépasse la simple commodité de regroupement. Ils favorisent une programmation plus sûre et plus lisible grâce à :

- La clarté syntaxique : L'accès aux données par l'opérateur point (Etudiant.Moyenne) rend le code explicite.
- La modularité : Un enregistrement peut être passé en paramètre à une fonction comme une seule entité, simplifiant ainsi les interfaces entre les différentes parties d'un programme.
- L'extensibilité : Il est aisé d'ajouter un nouveau champ à une structure sans modifier l'ensemble des algorithmes qui l'utilisent.

Cette capacité à structurer l'information est le prérequis indispensable à l'étude des structures de données avancées, comme les fichiers ou les listes chaînées, et constitue une première introduction conceptuelle à ce que sera, plus tard, la programmation orientée objet.