BASES DE DONNÉES

4ème année de l'enseignement secondaire

Professeur
Mohamed TRABELSI

Sommaire

Chapitre 1

Notion de Base de données

Durée : 4 Heures
Type : Théorique

I. Introduction à la gestion des données

1. Notion de donnée et d'information

Activité 1 :

Un médecin désir informatiser la gestion de ses consultations. Suite à une interview avec lui, on dégage les opérations suivantes :

- La création d'une nouvelle fiche « **patient** ».
- L'édition d'une **ordonnance**.
- La prise des **rendez-vous**…

On lui demandant la description de la fiche **patient** il nous communique les données suivantes :

Fiche patient

- Nom :
- Prénom :
- Adresse :
- Téléphone :
- N° CIN :

Activité 2 :

Identifier quelques informations utilisées dans les domaines suivants :

- Une administration d'un lycée.
- Une banque.
- Un restaurant.

a. Définition 1

Une donnée est une description élémentaire d'une information.

b. Définition 2

Plusieurs données regroupées et se rapportant à un même contexte donnent naissance à **l'information**.

c. Eléments constituant une information :

L'information se réfère à un objet du monde réel. C'est **l'entité**.

Une entité est décrite par un ensemble de données. Ce sont les **attributs**.

Un attribut prend des **valeurs** différentes ayant un **type** de données bien déterminé (Texte, Numérique, date…)**.**

Les entités peuvent avoir des **liens** entre elles.

Exemple :

Patients
CIN
Nom
Prénom
Adresse

Ordonnances
Numéro_ord
Date
CIN#

Rendez-vous
Numéro_rdv
Date
heure
CIN#

2. La persistance

La persistance ⇔ mémorisation + disponibilité (des données).

Lorsqu'on mémorise les données, deux aspects sont pris en considération :

Le type du support de mémorisation : Disque dur (local, sur serveur, CD-ROM…)

Le format de mémorisation : Structure choisie pour stocker les données...

La persistance des données peut être assurée grâce plusieurs organisations :

- L'organisation papier :

  Fiches, registre, cahier…

  Contraintes :

  1. Délais de recherche.
  2. Problèmes de sécurité.
  3. Volume important.
  4. Classement et tri difficiles.

- L'organisation en fichiers :

  Un fichier (file) est un ensemble de données structurées stocké sur une mémoire de masse. Ces données se présentent sous forme d'enregistrements (Record).

  Contraintes :

  1. Nécessiter d'écrire des programmes
  2. Manque de sécurité
  3. Problèmes de redondance des données
  4. Accès exclusif (un utilisateur à la fois)

- L'organisation en feuille de calcul Tableur.
- L'organisation en bases de données.

II. Les bases de données : les notions de base

1. Définition

Une base de données est une collection de données structurées relatives à un ou plusieurs domaines du monde réel.

Exemple : BD d'une bibliothèque.

2. Avantages d'une BD

a. Centralisation :

Les données peuvent être utilisées par plusieurs programmes et plusieurs utilisateurs.

b. Indépendance entre données et programmes :

Dans une BD les données sont décrites indépendamment des programmes. Ce qui n'est pas le cas avec les fichiers.

c. Intégration des liaisons entre les données.

Pas besoin d'un programme pour retrouver les liens entre les données.

d. Intégrité des données

Ce sont des règles de sécurité assurant la cohérence des données :

- Unicité des enregistrements.
- Interdiction de la suppression des données utilisées par d'autres données.

e. Concurrence d'accès

Plusieurs utilisateurs peuvent Accéder simultanément à la BD.

3. Les modèles des bases de données

a. Le modèle hiérarchique

La BD se présente comme un arbre d'objets en relation.

Exemple relatif à l'activité 1 :

BD_cabinet

!__ Patients

!__ MOT07

!__ Nom

!__ Prénom

!__ FAH07…

!__ Ordonnances

!__ CN01…

b. Le modèle réseau

Dans cette organisation tous les types de liens sont possibles entre les objets.

c. Le modèle relationnel

Une BD relationnelle est composée de Tables. Une table est composée de colonnes (champs) et de lignes (enregistrements). Deux tables peuvent être liées entre elles grâce à des champs identiques des deux côtés.

Voir retenons (livre page 23).

Chapitre 2

Notion de Système de Gestion de Bases de données

Durée : 2 Heures
Type : Théorique

I. Introduction

1. Définition d'un SGBD

Un système de gestion de base de données (SGBD) est un logiciel qui permet de : décrire, modifier, interroger et administrer les données d'une base de données.

2. Structure d'un SGBD

Un SGBD est constitué de deux composantes principales :

- Le moteur
- L'interface

II. Les fonctions d'un SGBD

1. La définition des données

Le SGBD nous permet de créer et de décrire les objets de la base de données (tables, liens, utilisateurs…), grâce au Langage de Description des Données (LDD).

Exemple : La commande CREATE TABLE nom_table ( ).

2. La manipulation des données

La manipulation des données peut être :

- La recherche
- La lecture
- La suppression
- La modification
- L'ajout

Le SGBD nous offre un Langage de Manipulation des Données (LMD) à fin de pouvoir réaliser ces opérations.

Exemple : La commande INSERT INTO nom_table ( ).

3. L'intégrité des données

C'est l'ensemble des opérations de contrôle que le SGBD effectue pour préserver la cohérence des données.

Exemple : Vérification de la validité de la valeur d'un champ.

4. La gestion des accès concurrents

Le SGBD gère l'accès simultané des utilisateurs à la base de données.

5. La confidentialité

Tous les utilisateurs d'une base de données ne sont pas supposés pouvoir consulter toutes les informations. Des sous schémas de la base permettent de résoudre ce problème en plus des mots de passes et des droits d'accès.

6. La sécurité du fonctionnement

Faire une copie de sauvegarde de la base.

Remise en marche de la base en cas de panne.

III. Les principaux SGBD

- ORACLE
- Microsoft SQL SERVER
- MySQL
- Microsoft ACCESS

IV. Cycles de développement des bases de données

Niveau externe : Analyse de l'existant.

Niveau conception : Modélisation des entités du monde réel.

Niveau interne : Création de la base de données.

V. Intervenants du domaine BD

- Utilisateurs
- Concepteurs
- Administrateurs base de données (DBA)
- Développeurs d'application base de données

Chapitre 3

Structure d'une base de données relationnelle

Durée : 6 Heures
Type : Théorique

I. Notion de table

1. Définition :

Une table est un ensemble de données relatives à une même entité, structurée sous forme d'un tableau (liste). Une table peut être appelée aussi une "Relation".

Exemple : Cas d'un cabinet médical.

PATIENTS

CIN	Nom	Prénom	Téléphone	Date_n

ORDONNANCES

Numéro_ord	Date	CNAM	CIN

2. Remarques :

- Les données d'une table peuvent être stockées sur un ou plusieurs fichiers.
- Une table peut être considérée comme un ensemble mathématique. Ainsi, on pourra faire l'union ou l'intersection de deux tables.

II. Notion de colonne

1. Définition :

Une colonne (champ) représente une propriété élémentaire de l'entité décrite par cette table.

2. Caractéristiques d'un champ :

a. Nom

b. Type de données

c. Taille éventuelle

d. Obligatoire (oui / non) Not Null

e. Valeur par défaut

f. Valide si : On peut créer une règle indiquant les valeurs utilisées.

Exemple : Cas de la table **Patients**.

Nom de la colonne	Description	Type de données	Taille	Obligatoire	Valeur par défaut	Valeurs autorisées
CIN	Carte d'Identité Nationale	Chaîne	10	Oui		
Prénom		Chaîne	50	Non		
Nom		Chaîne	50	Non		
Téléphone		Chaîne	20	Non	"71"	
Date_n	Date de naissance	Date		Non		≤ Aujourd'hui ( )

III. Notion de ligne

1. Définition :

Une ligne (enregistrement) représente une occurrence du sujet représenté par la table.

ORDONNANCES

Numéro_ord	Date	CNAM	CIN
…			
…			
30327	22/09/2010	T3	02990399
…			
…			

IV. Notion de clé primaire

Voir activité page 50.

1. Définition :

La clé primaire d'une table est un champ ou un ensemble de champs permettant d'identifier de manière unique chaque enregistrement de la table.

2. Caractéristiques :

Unique et non nulle.

V. Liens entre les tables

1. Liens de type 1, n

Dans un contexte relationnel, les entités d'un système d'information admettent des relations entre elles. On peut formuler ces relations comme suit :

Cas des tables patients et ordonnances :

- Un patient peut avoir **1 ou plusieurs** ordonnances.
- Une ordonnance est délivrée à **un et un** seul patient.

Dans ce cas on parle de lien de type **1 à plusieurs**. (1, n)

Définition :

Un lien entre deux tables A et B (A est associée à une ou plusieurs occurrence de B) se traduit par l'ajout dans la table B d'un nouveau champ correspondant à la clé primaire de la table A. Ce champ est appelé clé étrangère.

Dans ce cas A est une table mère, B est une table fille.

2. Liens de type n, n

On peut aussi parler de liens **plusieurs à plusieurs**. (n, n)

Cas des tables Ordonnances et médicaments.

- Une ordonnance contient **un ou plusieurs** médicaments.
- Un médicament est inscrit dans **une ou plusieurs** ordonnances.

Définition :

Ce type de lien entraine la création d'une troisième table dite intermédiaire. Elle aura comme clé primaire les deux clés primaires de ses tables mères. D'autres champs pourront s'ajouter à cette clé en cas de besoin.

On obtient ainsi deux liens de type **1, n**.

VI. Notion de contrainte d'intégrité

Définition :

Une contrainte d'intégrité est une règle appliquée à un champ ou à une table et qui doit être toujours vérifiée.

a. **Les contraintes de domaines :** (valide si)

b. **Les contraintes d'intégrité de tables :** (clé primaire)

c. **Les contraintes d'intégrité référentielle :**

- Champ clé étrangère ne peut contenir qu'une valeur déjà existante dans la clé primaire correspondante.
- La suppression d'un enregistrement d'une table mère A utilisé par une table fille B est interdit.

VII. Représentation de la structure d'une base de données

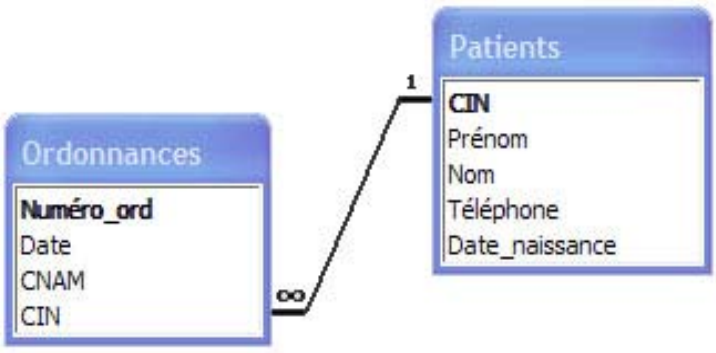
Le schéma base de données est une représentation des différentes structures de la base. Cette représentation peut être faite selon deux formalismes :

1. La représentation textuelle :

Patients (CIN, Prénom, Nom, Téléphone, Date_n)

Ordonnances (Numéro_ord, Date, CNAM, **CIN#**)

2. La représentation graphique :



VIII. Exemple de base de données

Voir livre pages 56 - 57.

Chapitre 4

Démarche de détermination de la structure d'une base de données

Durée : 6 Heures
Type : Théorique / Pratique

Introduction

Un projet BD commence par une phase d'analyse et de conception du système d'information (Domaine (s)) sur lequel porte la base. Cette phase conceptuelle (détermination de la structure d'une BD) est constituée de plusieurs étapes. En ce qui suit, une description de chacune de ces étapes.

0) Délimiter le domaine

A partir d'une description du monde réel on essaye de cerner le domaine qui constitue l'objet de la base.

1) Déterminer les colonnes (champs)

a. Règles :

- Atomicité : un champ contient une seule donnée.
- Colonne non calculée : Exemple : Qté, PU et Total
- Eviter les colonnes similaires.
- Eviter les colonnes manquantes.

Liste des colonnes

Nom de la colonne	Description	Type de données	Taille	Obligatoire	Valeur par défaut	Valeurs autorisées	Sujet

b. Affecter les colonnes aux tables

Cette étape est réalisée grâce à la propriété "sujet" du tableau 1.

2) Déterminer les tables

Chaque table correspond à un sujet du domaine étudié.

Liste des tables

Tables	Description	Sujets

3) Déterminer les clés primaires

Détecter les clés primaires de chaque table.

4) Déterminer les liens entre les tables

Liste des liens entre les tables.

Table mère	Table fille	Clé primaire	Clé étrangère

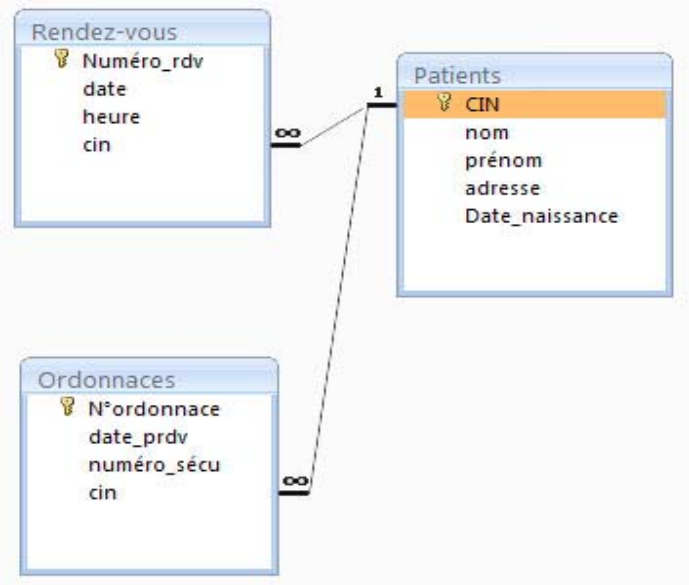
5) Représentation textuelle.

Exemple :

- Patients (CIN, Prénom, Nom, Téléphone, Date_n)
- Ordonnances (Numéro_ord, Date, CNAM, **CIN#**)
- Rendez-vous (Numéro_rdv, date, heure, CIN#)

6) Représentation graphique.

Exemple :



7) Analyser et affiner la structure de la base de données

Chapitre 5

Création et modification de la structure D'une base de données

Durée : 8 Heures
Type : Théorique / Pratique

Introduction

Il existe deux modes pour créer une BD :

- Mode assisté
- Mode commande

I. Création d'une base de données en mode assisté

Le mode assisté nous permet de créer les éléments de la base de données à travers des assistants graphiques.

1. Création de la base de données

0) Ouvrir MS Access :
1) Créer une nouvelle BD vide.
2) Nom de la base. (Bibliothèque)
3) Préciser l'emplacement sur le disque. (D:\)
4) Fichier / Propriétés.

2. Création d'une table

0) Créer la table en mode création.
1) Créer les colonnes.
2) Préciser leurs types.
3) Faire une description des champs (optionnelle).
4) Régler les propriétés de chaque colonne.
5) Enregistrer la table.

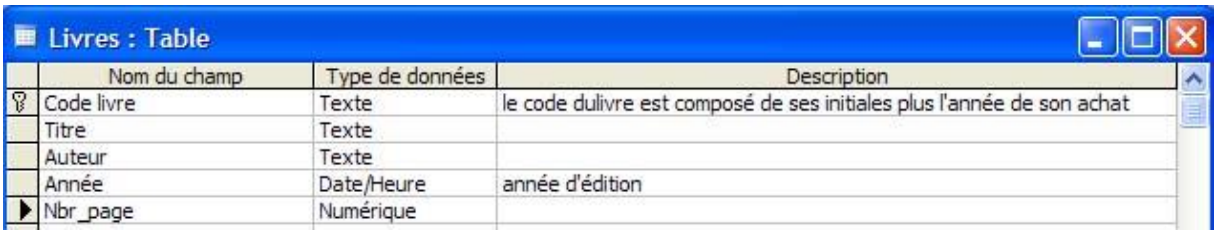
3. Indiquer la clé primaire d'une table

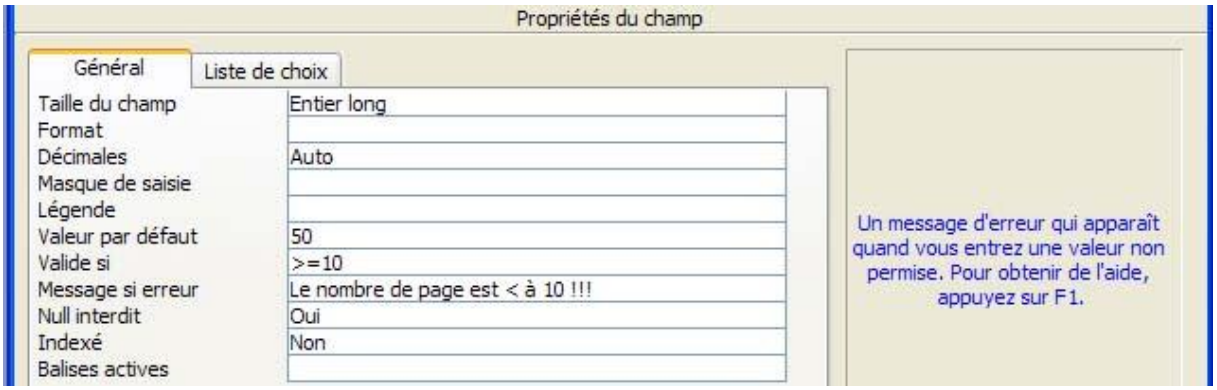
Créer la clé primaire :

1. **Sélectionner le ou les champs formant la clé primaire.**
2. **Cliquer sur le bouton clé primaire.** 

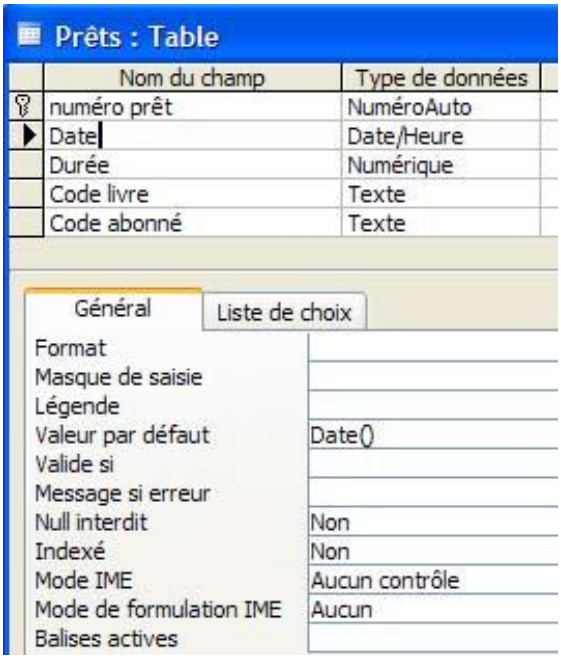
TAF :

- **Livres**





- **Prêts**



- **Abonnés**

4. Etablir un lien entre deux tables (clé étrangère)

0) Ouvrir la fenêtre des relations
1) Insérer les tables
2) Créer les liens (Cocher l'option : Appliquer l'intégrité référentielle)
3) Enregistrer le modèle.

5. Modification de la structure d'une base de données en mode assisté

a. Ajout d'une colonne

Ajouter la colonne **Editeur** dans la table **livre**.

b. Suppression d'une colonne

Supprimer la colonne **Durée** dans la table **Prêts**.

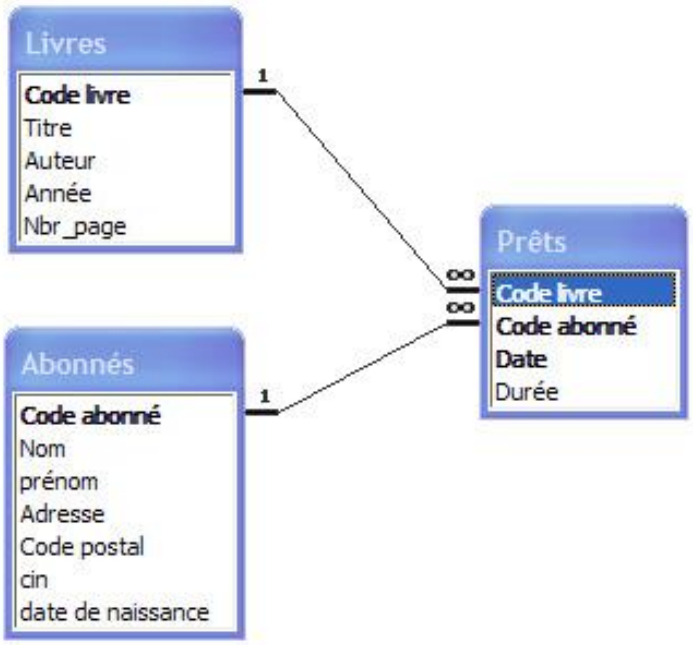
c. Modifier les propriétés d'une colonne

Changer la taille du **Nom abonné** à 100 caractères.

d. Modifier la clé primaire

Changer la clé primaire actuelle de la table Prêts par la clé composée suivante :

Code livre, **Code abonné**, **date**



e. Suppression d'une table

0) Créer la table **Employés** à l'aide de l'assistant.

1) Supprimer cette table.

6. Suppression d'une base de données

Supprimer tous les objets de la base.

II. Création d'une BD en mode commande

1. Le langage SQL (Structured Query Language) :

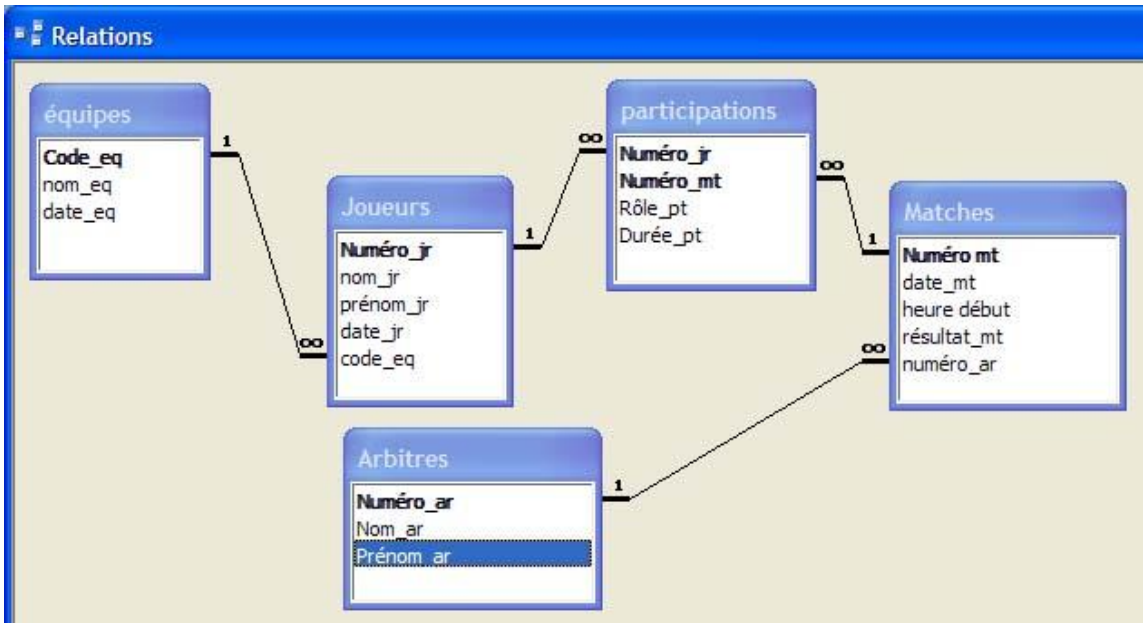
C'est un langage structuré de requêtes destiné à interroger ou à manipuler une base de données. On distingue trois familles de commande SQL :

- Langage de définition des données (LDD) : Permet de modifier la structure de la base de données.
- Langage de manipulation des données (LMD) : Permet de consulter / modifier le contenu de la base de données.
- Langage de contrôle des données (LCD) : Permet de gérer la sécurité de la BD et la cohérence des données.

2. Création de la base de données

0) Ouvrir MySQL. Administration / PhpMyAdmin
1) Créer une BD. Nom de la base : Matches

Représentation graphique de la base Matches.



3. Création d'une table

0) Exécuter une **requête**.
1) Ecrire le script SQL.
2) Cliquer sur **Exécuter**.

```
CREATE TABLE nom_table (
Nom_Colonne Type_colonne [[NOT] NULL] [DEFAULT valeur] [Contraintes sur colonne (1)],
        …,
[Contraintes sur la table (2)]);
```

(1) CONSTRAINT nom_contrainte
PRIMARY KEY
REFERENCES nom_table (nom_champ_référencé) [ON DELETE CASCADE]
CHECK (Condition de valeurs)

(2) Se sont des contraintes fonctionnant sur plusieurs champs à la fois
PRIMARY KEY (colonne1, colonne2,…)
FOREIGN KEY (colonne1, colonne2,…) REFERENCES nom_table (colonne1, colonne2,…)
[ON DELETE CASCADE]
CHECK (Condition)

T.A.F : Créer les tables et les liens de la base de données GMT (Gestion des matches de football).

Remarque : L'ordre de création des tables est important.

- **Equipes**

```
CREATE TABLE équipes (
code_eq varchar(10) CONSTRAINT pk_équipes PRIMARY KEY,
nom_eq varchar(50) NOT NULL,
date_eq date);
```

- **Joueurs**

```
CREATE TABLE joueurs (
numéro_jr int(5) CONSTRAINT pk_joueurs PRIMARY KEY,
nom_jr varchar(50) NOT NULL,
prénom_jr varchar(50) NOT NULL,
date_jr date,
code_eq varchar(10) CONSTRAINT Fk_joueurs_équipes REFERENCES
équipes(code_eq ) );
```

- **Arbitres**

```
CREATE TABLE arbitres (
numéro_ar int(5) CONSTRAINT pk_arbitres PRIMARY KEY,
nom_ar varchar(10),
prénom_ar varchar(10));
```

- **Matches**

```
CREATE TABLE matches (
numéro_mt int(5) CONSTRAINT pk_matches PRIMARY KEY,
date_mt date,
heure_début varchar(6),
résultat_mt varchar(10),
numéro_ar int(5) CONSTRAINT Fk_matches_arbitres REFERENCES
arbitres(numéro_ar) );
```

- **Participations**

```
CREATE TABLE participations (
numéro_mt int(5) CONSTRAINT Fk_participations_matches REFERENCES
matches(numéro_mt),
numéro_jr int(5) CONSTRAINT Fk_participations_joueurs REFERENCES
joueurs(numéro_jr),
role_pt varchar(50),
durée_pt int(3) DEFAULT 90 CHECK (durée_pt >= 90),
CONSTRAINT pk_participations PRIMARY KEY (numéro_mt, numéro_jr));
```

III. Modification de la structure d'une BD

1. Ajout d'une colonne :

Rajouter à la table **matches** la colonne Lieu.

```
ALTER TABLE matches
ADD COLUMN (Lieu varchar (100));
```

2. Suppression d'une colonne :

Supprimer la colonne Lieu de la table **matches**.

```
ALTER TABLE matches DROP COLUMN Lieu;
```

3. Ajout d'une contrainte :

Rendre la colonne role_pt de la table **participations** une clé primaire.

```
ALTER TABLE participations
ADD CONSTRAINT PRIMARY KEY(role_pt);
```

4. Suppression d'une contrainte :

Supprimer la contrainte clé primaire de la table **matches**.

```
ALTER TABLE matches DROP CONSTRAINT PRIMARY KEY;
```

5. Modification d'une propriété d'une colonne :

Modifier le type de la colonne Lieu de la table **matches** par le type Texte.

```
ALTER TABLE matches MODIFY Lieu Text;
```

6. Activer / désactiver une contrainte :

Désactiver la contrainte clé primaire de la table **matches**.

```
ALTER TABLE matches DISABLE CONSTRAINT PRIMARY KEY;
```

Réactiver cette même contrainte.

```
ALTER TABLE matches ENABLE CONSTRAINT PRIMARY KEY;
```

IV. Suppression d'une table

```
DROP TABLE nom_table;
```

Chapitre 6

Manipulation d'une base de données

Durée : 8 Heures
Type : Théorique / Pratique

I - Manipulation des données en mode assisté

1. Mise à jour des données

T.A.F :

- Ouvrir la BD Relation.mdb
- Compléter le schéma relationnel de la base.

a. Insertion d'une ligne

Ouvrir la table **Clients** et rajouter un client dont le Code est "FBF08".

Fermer la table Clients.

b. Modification d'une ligne

Modifier le numéro de téléphone du client "FBF08".

c. Suppression d'une ligne

Supprimer le client "FBF08" : clic droit / supprimer l'enregistrement.

Valider la suppression.

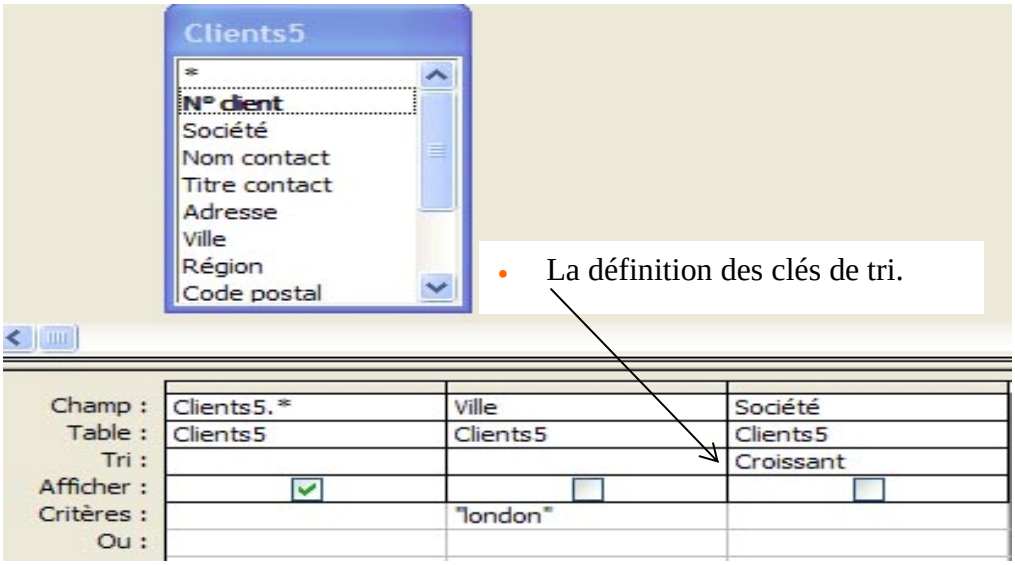
2. Recherche de données : Les requêtes

a. Création d'une requête

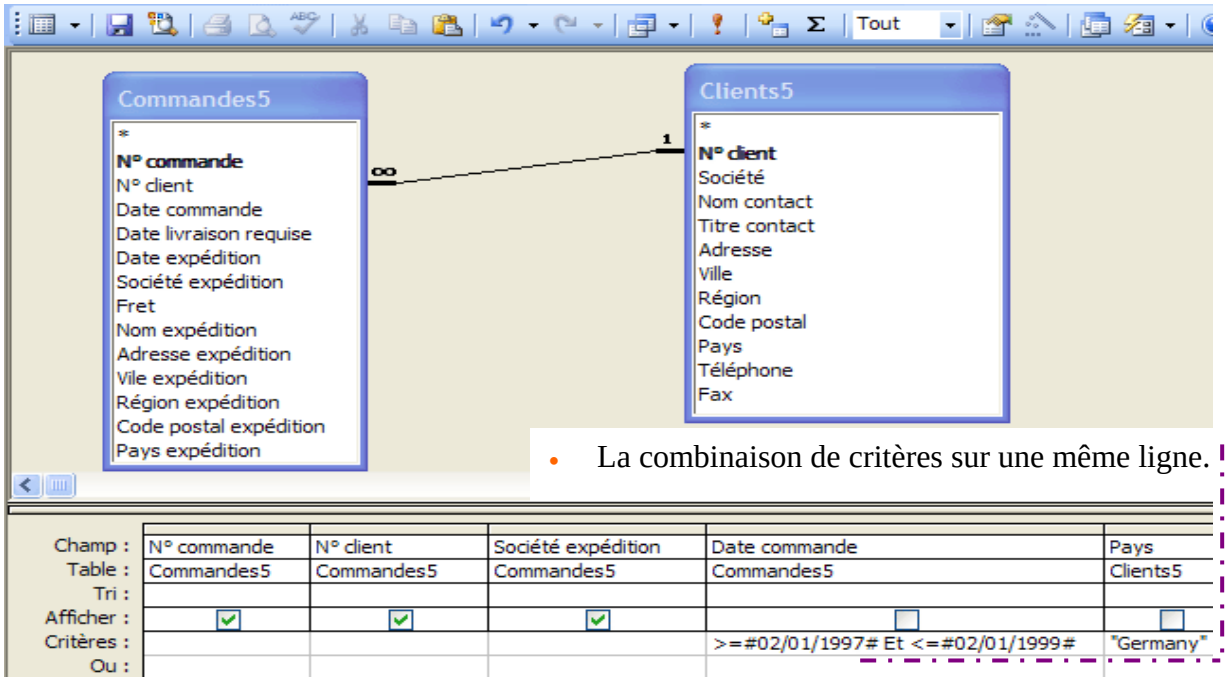
- Sélectionner l'onglet Requête dans la fenêtre BD Access.
- Créer une requête en mode création.
- Insérer les tables nécessaires pour cette requête : 
- Paramétrer la grille d'interrogation.
- Choix des colonnes à afficher dans une requête : Placer le champ dans une colonne de la grille d'interrogation. * signifie tous les champs.
- Visualisation du résultat d'une requête : 
- Enregistrement d'une requête. 
- Les critères de filtres : Texte, numérique, date.

  Exemples : ville = "London", pu > 20 ou date commande >= #02/01/97#

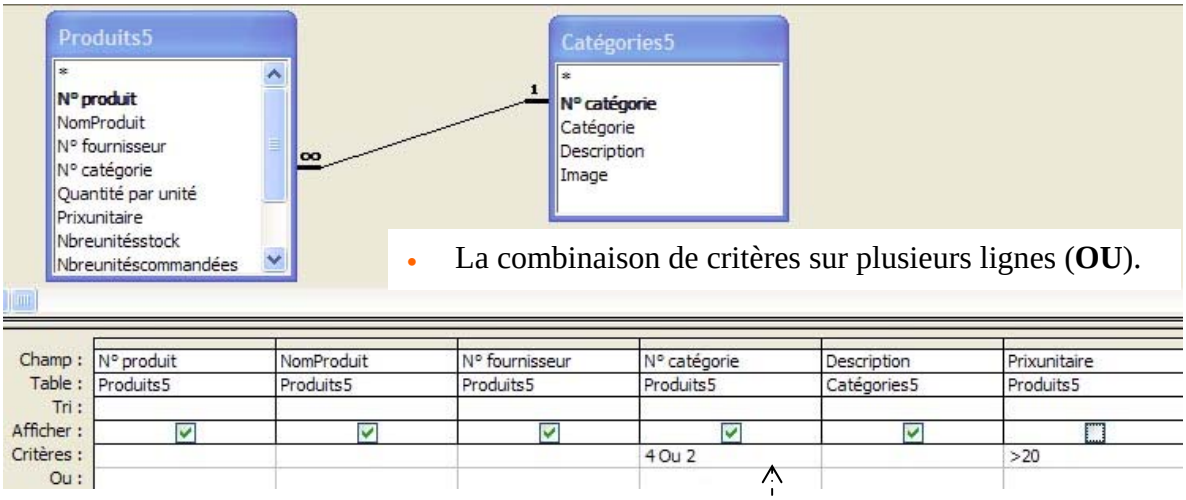
R1 : Donner toutes les informations des clients de la ville de "London" triées selon le nom de leur société dans l'ordre croissant.



R2 : Donner les n° commande, n° client et société expédition pour les commandes passées entre le 02/01/97 et le 02/01/99 des clients de l'Allemagne.



R3 : Donner les n° produits, nom produit, n° fournisseur, n° catégorie, Description catégorie pour les produits de catégorie 4 ou 2 et dont le Prix unitaire >20



- La combinaison de critères sur plusieurs lignes (**OU**).

Champ :	N° produit	NomProduit	N° fournisseur	N° catégorie	Description	Prixunitaire
Table :	Produits5	Produits5	Produits5	Produits5	Catégories5	Produits5
Tri :						
Afficher :	☑	☑	☑	☑	☑	☐
Critères :				4 Ou 2		>20
Ou :						

- Les requêtes paramétrées : Rendre le champ N° catégorie paramétrable.
  Dans critères il faut écrire : [Donner le numéro de catégorie :].

- Les formules

- Les champs calculés

- Import / export

II - Manipulation des données en mode commande

1. Mise à jour des données

a. Insertion de ligne

- Insérer dans la table **« produits »** l'article suivant (*tous les champs dans l'ordre*) :

N° produit	Nom produit	N° fournisseur	Catégorie	Quantité	Prix unitaire	Abandonné
78	Biscuits	12	5	10	1	non

INSERT INTO produits VALUES (78, Biscuits, 12, 5, 10, 1, non);

- Insérer dans la table **commandes** l'enregistrement suivant :

N° commande	Client	Date commande	Date livraison	Sté expédition	Fret
14002	FOLKO	17/04/1999			

INSERT INTO commandes (N° commande, date commande, client) VALUES (14002, 17/04/1999, FOLKO);

b. La modification d'une ligne

- Les prix unitaires de tous les **produits** de catégorie 4 ont augmentés de 2 €.

UPDATE produits
SET pu = pu + 2
WHERE catégorie = 4;

- Mettre à jour le client VICTE comme suit :

Nom contact = Marc Leclair, Titre contact = Directeur commercial. Effacer le contenu du champ pays.

UPDATE clients
SET nom_contact = ' Marc Leclair', titre_contact = 'Directeur commercial', pays = null
WHERE N° client = 'VICTE';

c. Suppression d'une ligne

- Supprimer de la table **«produits»** tous les produits de catégorie 9.

DELETE FROM produits
WHERE catégorie = 9;

- Supprimer tous les clients :

DELETE FROM clients.

2. Recherche de données : Les requêtes

a. Introduction

Une requête est une opération de recherche de données à partir d'une ou de plusieurs tables. Cette recherche peut concerner :

- Certaines colonnes d'une table : une projection.
- Certaines lignes d'une table : une sélection.
- Deux tables en relation : une jointure.

Une requête peut être réalisée en combinant ces trois actions.

b. Requête Projection

R1 : Afficher les trois premières colonnes de la table clients.

```
SELECT N_client 'Numéro client', Société 'Nom de la société', Nom_contact
'Nom du contact'
FROM clients ;
```

Les alias : Ce sont les noms de colonnes de la table résultat.

c. Requête Sélection

R2 : Donner toutes les informations des clients de la ville de "London" triées selon le nom société dans l'ordre croissant.

```
SELECT *
FROM clients
WHERE ville = 'london'
Order by société ASC;
```

- ORDER BY sert à trier la table résultat dans un ordre croissant ou décroissant (ASC, DESC).
- Les critères : Se sont des expressions logiques utilisant les opérateurs suivants :
  1) =, <, >, !=, <=, >=
  2) BETWEEN
  3) IN
  4) IS NULL, IS NOT NULL
  5) LIKE
  6) AND, OR, NOT

R3 : Afficher la liste des produits de catégorie 2, 4, 6 et 8 et dont le nom commence par 'C' ou ont comme deuxième caractère 'o'.

```
SELECT *
FROM produits
WHERE N_catégorie IN (2, 4, 6, 8)
AND Nom_produit LIKE 'C%' OR Nom_produit LIKE '_o%'
;
```

a. Requête jointure (Recherche à partir de plusieurs tables)

R4 : Donner les **numéros de commandes**, **numéros client** et **société d'expédition** pour les commandes passées entre le 02/01/97 et le 02/01/99 des clients de l'Allemagne.

```
SELECT N_commande,  C.N_client, Société_expédition
FROM clients L, Commandes C
WHERE Date_commande >=  '1/2/1997' AND Date_commande <=  '1/2/1999'
AND Pays ='Germany'
AND L.N_client = C.N_client ;
```

- C est l'alias de la table commandes.

R5 : Préparer la liste des commandes (n° commande, n° client) ayant bénéficiées d'une remise.

```
SELECT  DISTINCT commandes.n_commande, commandes.n_client,
FROM commandes, détails_commandes
WHERE détails_commandes.remise IS NOT NULL
AND commandes.N_commande = détails_commande.N_commande ;
```

- DISTINCT permet d'éliminer les redondances au niveau des enregistrements résultats d'une requête.

R6 : Donner les n° produits, nom produit, nom catégorie, pour les produits de catégorie 4 ou 2 et dont le Prix_unitaire est entre 20 et 50.

```
SELECT N_produit, nom_produit, catégorie
FROM produits, catégories
WHERE produits.N_catégorie = 2 OR produits.N_catégorie = 4
AND Prix_unitaire BETWEEN 20 AND 50
AND produits.N_catégorie = catégories.N_catégorie;
```

b. Requête de calculs

Le langage SQL prévoit des fonctions agrégats qui nous permettent de faire du calcul au niveau des requêtes.

- COUNT : Permet de compter le nombre de lignes résultats obtenues par la commande SELECT.

R7 : Afficher le nombre de société d'expédition aux quelles la société fait appel.

```
SELECT COUNT (*)
FROM expéditeurs ;
```

- SUM : Permet de faire la somme des valeurs d'une colonne dont le type de données est numérique.

R8 : Calculer la valeur de la quantité totale en stocke des produits de catégorie 2.

```
SELECT SUM (nbre_unités_stock)
FROM produits
WHERE catégorie = 2 ;
```

- MIN : Minimum
- MAX : Maximum
- AVG : Moyenne

R9 : Déterminer la valeur minimum, maximum et la moyenne des prix unitaires des produits non abandonnés.

```
SELECT MIN (Prix_unitaire) 'minimum PU',
MAX (Prix_unitaire) 'maximum PU', AVG (Prix_unitaire) 'La moyenne PU'
FROM produits
WHERE produit_abonné = 'non';
```

R10 : Donner pour chaque catégorie inférieure à 6, le nombre de produits dont le prix unitaire est inférieur ou égal à 20 €.

```
SELECT N_categorie, Count (*)
FROM produits
WHERE prix_unitaire <= 20
GROUP By N_categorie
 HAVING N_categorie < 6;
```

R11 : Afficher les détails commande avec le total facturé par ligne.

```
SELECT *, Prix_unitaire * Quantité - remise 'Total'
FROM details_commandes;
```

Chapitre 7

Sécurisation d'une base de données

Durée : Heures
Type : Théorique / Pratique

I – Mécanismes de mise en œuvre d'une stratégie de sécurité

- L'authentification
- Les droits et privilèges
- Les LOGs ou traces
- Tolérance aux pannes
- Sauvegarde et restauration
- Mécanismes transactionnels

II – Contrôle de données dans le langage SQL

LCD : Langage de Contrôle de données.

1. Création d'un utilisateur

Créer l'utilisateur Mourad_g et lui préparer le mot de passe :Kj3MG.

```
CREATE USER Mourad_g
SET PASSWORD Kj3MG ;
```

2. Les droits :

De quels droits dispose cet utilisateur ?

Réponse : Aucun droit.

a. Les types de droits :

Il existe deux types de droits :

- Les droits systèmes :

- Les droits sur objets :

b. Accorder des droits

Donner à l'utilisateur Mourad_g tous les droits système avec le profil administrateur.

```
GRANT ALL
TO Mourad_g
WITH ADMIN OPTION ;
```

Donner à l'utilisateur Mourad_g les droits suivants sur la table "Clients" : ALTER, DROP. Cet utilisateur devra pouvoir accorder lui-même ces droits à d'autres utilisateurs.

```
GRANT ALTER, DROP
ON Clients
TO Mourad_g
WITH GRANT OPTION ;
```

Donner à tous les utilisateurs tous les droits sur la table "Produits".

```
GRANT ALL
ON Produits
TO PUBLIC ;
```

c. Supprimer des droits

Supprimer tous les droits dont dispose Mourad_g sur la table "Clients".

```
REVOKE ALL
ON Clients
FROM Mourd_g ;
```

Chapitre III

Les sous programmes

Durée : 12 Heures
Type : Théorique et pratique

I - Partie rappel

I.1 Analyse modulaire :

Pour résoudre un problème complexe, on peut procéder à une décomposition de ce dernier en sous problèmes. Ces derniers sont à leur tour décomposés selon le besoin. La décomposition s'arrête aux sous problèmes relativement simples à résoudre. On associe à chaque sous problème un module (un sous programme) assurant sa résolution.

(Sources livre 2 TI – Chapitre 12)

I.2 Notions de sous-programme :

- Un sous-programme est un ensemble d'instructions, analogue à un programme.
- Sa peut être une procédure ou une fonction.
- Un sous-programme peut être exécuté plusieurs fois grâce à, des appels.
- Une procédure est un sous-programme qui produit zéro ou plusieurs résultats alors qu'une fonction est un sous-programme qui ne produit qu'un seul résultat de type simple.

I.3 Intérêts de l'analyse modulaire :

- Organisation du code source, il est plus efficace de séparer les différentes parties d'un programme.
- La disposition en modules nous permet aussi de savoir lequel des sous programmes est à corriger dans le cas où on a une erreur.
- Il est aussi plus facile de faire évoluer le programme et de passer d'une version à une autre.
- La réutilisation du code.

I.4 Les fonctions

a. Définition

Une fonction est un sous-programme qui retourne un résultat de **type simple** contenu dans son identificateur.

b. Appel d'une fonction

Une fonction se comporte comme une variable. L'appel de la fonction doit nécessairement apparaître dans une expression (d'affectation ou d'affichage,…).

En Algorithme

Variable ← **nom_fonction** (paramètres effectifs)
Ou
Ecrire (**nom_fonction** (paramètres effectifs))

En Pascal

Variable := **nom_fonction** (paramètres effectifs)
Ou
Write (**nom_fonction** (paramètres effectifs))

c. Définition d'une fonction :

En Algorithme

0) **Fonction** nom_fonction (liste des paramètres formels) : Type du résultat ;
1) Instruction 1
2) Instruction 2
3) .....
4) nom_fonction ← résultat ;
5) **Fin** nom_fonction

TDO Locaux

Objet	Type	Rôle

En Pascal

```
Function nom_fonction (liste des paramètres formels) : Type du résultat ;
        Var
            {Déclarations des variables locales} ;
        begin
            {Instructions de la fonction} ;
            Nom_fonction := résultat ;
end;
```

I.5 Les procédures

a. Définition

Une procédure est un sous-programme qui peut retourner zéro ou plusieurs résultats.

b. Appel d'une procédure

Voir livre page 110 (II.2)

c. Définition d'une procédure:

Voir livre page 111 (II.4)

I.6 Déclarations et accès aux objets :

a. Les objets locaux :

Un objet Local est un objet déclaré et connu seulement à l'intérieur d'un sous-programme.

b. Les objets globaux :

Un objet Global est déclaré dans la partie déclarative du programme principal. Il peut être utilisé par le programme principal ou par les différents autres sous-programmes.

c. Accès aux objets

La porté de l'objet défini les possibilités d'accès à ce dernier à partir de différents endroits du programme. Exemple, un objet déclaré dans un sous-programme n'est pas accessible à partir du programme principal. Par contre, un objet global est accessible à partir d'un sous-programme.

I.7 Les paramètres et leurs modes de passage :

a. Les paramètres formels et les paramètres effectifs :

- Les paramètres formels figurent dans l'entête de la définition d'un sous-programme (fonction ou procédure).
- Les paramètres effectifs figurent dans l'appel d'un sous-programme (fonction ou procédure).

Remarques :

- Les paramètres **effectifs** et les paramètres **formels** doivent s'accorder du point de vue nombre et ordre.
- Leurs types doivent être identiques ou compatibles, selon le mode de passage des paramètres.

b. Passage de paramètre par valeur et par variable :

La substitution des paramètres effectifs aux paramètres formels s'appelle passage de paramètre, il s'agit en fait de transfert de données entre le programme principal (P.P) et le sous programme ou l'inverse.

Nous utiliserons deux modes de passage de paramètres :

1. Le passage de paramètres par **valeur**
- Le passage de paramètres par **variable**

Si le paramètre formel n'est pas précédé par le mot **VAR** alors il s'agit d'un passage de paramètres effectifs par **valeur.**

Si le paramètre formel est précédé par le mot **VAR** alors il s'agit d'un passage de paramètres effectifs par **variable.**

Voir livre page 110 (II.3)

{exemple d'un programme construit en modules}	Commentaire

```
Program Combinaison ;
Uses wincrt;

Var
      f1, f2, f3 ,p , n : integer ;
      c : integer ;
```

Déclaration du programme principal

```
Procedure saisie ( var n, p :integer);
Begin
Repeat
      Writeln('donner un entire n ');
      readln(n) ;
      Writeln('donner un entire p');
      readln(p) ;
until (n>=p) and ( p>0);
end;
```

Entête

Le corps de la procédure

Sous programme Appelé

Paramètres formels

```
Function fact (x: integer) : integer ;
Var f, c :integer;
Begin
   f:=1;
 For c :=1  to  n  Do
     Begin
           f:= f * c;
       End;
Fact:=f;
End ;
```

Entête

Déclaration

Le corps de la fonction

Sous programme Appelé

```
Begin
Saisie (n, p);  {Appel}

f1:= fact (n);   {Appel}
f2 := fact (p);   {Appel}
f3 := fact (n-p);  {Appel}

c := f1 div (f2 *f3) ;

writeln ('la combinaison de p objets parmi n est  =  ', c);
End.
```

Paramètres effectifs

Programme principal

II - Applications

- **Application 1 :**

20	23	21	22	26	27	23

Ecart type

$$\text{Moyenne} = (\sum_{i=1}^{n} Mi) / n \qquad \text{écart type} = \sqrt{\sum_{i=1}^{n} (Mi - Moy)^2}$$

- **Application 2 :**

Énoncé :

On désire vérifier l'existence d'une chaîne de caractère **ch** dans un tableau **T** de n chaînes de caractères ($2 \leq n \leq 10$). Faire l'analyse de se problème, tout en prévoyant un module pour la saisie et un module pour la vérification de l'existence de **ch** dans **T**. Sachant que ce module renvoi l'indice de la case dans laquelle **ch** a été trouvé, sinon zéro pour dire que ch n'existe pas dans T afficher un message indiquant le résultat de la vérification.

Exemple :

ch = "BAC"

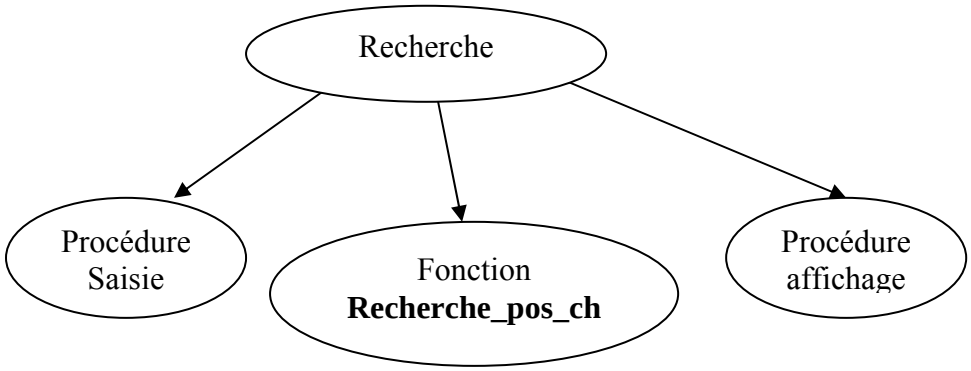
T =

INFO	BAC	SPORT

Affichage : ch existe à la case 2.

Solution :

a. Découpage modulaire



b. Analyse principale :

Résultat : affichage (p)

Traitements :

p ← Recheche_pos_ch (ch, T, n)

(ch, T, n) = saisie (ch, T, n)

Algorithme :

0) Début Recherche
1) saisie (ch, T, n)
2) p ← Recheche_pos_ch (ch, T, n)
3) affichage (p)
4) Fin Recherche

T.D.O Globaux

Objet	Type	Rôle
p, n	Octet	
ch	chaîne	
T	Tch	

Type
Tch = tableau de 10 chaînes

c. Analyse procédure affichage :

Procédure **affichage** (p : octet)

Résultat : Affichage
Traitements :

```
Si  p = 0  alors écrire ("La chaîne n'existe pas dans T")
           Sinon écrire ("La chaîne existe dans T à la case ", p)
Fin Si
```

Algorithme :

```
0) Procédure affichage ( p : octet)
1) Si  p = 0  alors écrire ("La chaîne ",ch," n'existe pas dans T")
               Sinon écrire ("La chaîne ",ch," existe dans T à la case ",p)
   Fin Si
2) Fin affichage
```

d. Analyse procédure affichage :

Fonction **Recheche_pos_ch** (ch : chaîne, T : Tch, n : octet) : octet

Résultat : Recheche_pos_ch
Traitements :

```
Recheche_pos_ch ← p
i ← 1
existe ← faux
Répéter
        Si T[i] = ch alors   p ← i
                             existe ← vrai
                     sinon   i ← i + 1
        Fin Si
Jusqu'à (existe) ou (i > n)
```

Algorithme :

```
0) Fonction Recheche_pos_ch ( ch : chaîne, T : Tch, n : octet) : octet
1) p ← 0
2) i ← 1
   existe ← faux
   Répéter
        Si T[i] = ch alors   p ← i
                             existe ← vrai
                     sinon   i ← i + 1
        Fin Si
   Jusqu'à (existe) ou (i > n)
3) Recheche_pos_ch ← p
4) Fin Recheche_pos_ch
```

T.D.O Locaux

Objet	Type	Rôle
i	Octet	compteur
existe	Booléen	drapeau
p	Octet	

e. Analyse de la procédure saisie :

Procédure **saisie** (var ch : chaîne, var T : Tch, var n : octet)

Résultat : ch, T et n saisis
Traitements :

```
Ch = donnée
Répéter
        n  = donnée (" Donner n entre 2 et 20 : " )
Jusqu'à  n dans [2..20]
Pour i de 1 à n faire
        Ecrire (" Donner la case ",i, " : "), Lire (T[i])
Fin Pour
```

Algorithme :

0) Procédure **saisie** (var ch : chaîne, var T : Tch, var n : octet)
1) Ecrire ("Donner une chaîne : "), Lire (ch)
2) Répéter
        Ecrire (" Donner n entre 2 et 20 : "), lire (n)
   Jusqu'à n dans [2..20]
3) Pour i de 1 à n faire
        Ecrire (" Donner la case ",i, " : "), Lire (T[i])
   Fin Pour
4) Fin **saisie**

T.D.O Locaux

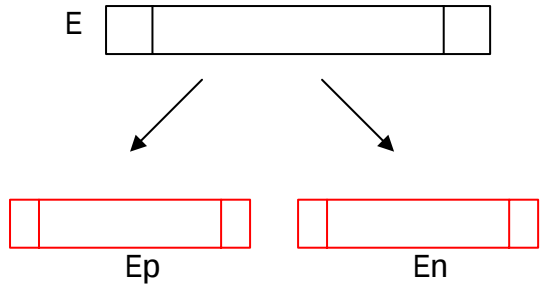
Objet	Type	Rôle
i	Octet	compteur

f. Traduction Pascal : (voir fichier : Pos_ch_t.pas)

- **Application 3 :**

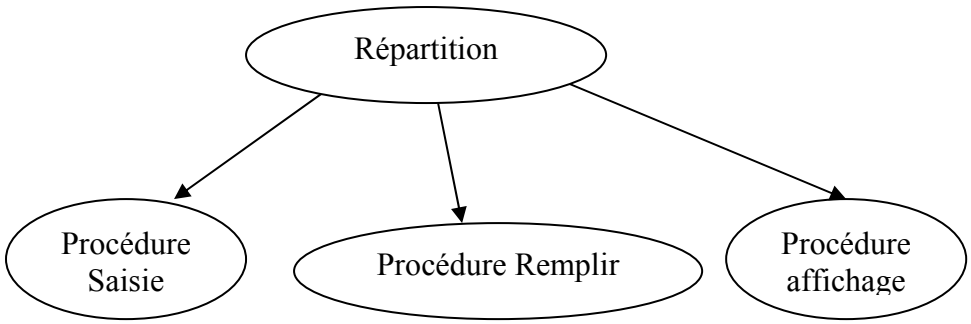
Énoncé :

Ecrire un programme nommé "rangement", qui permet de saisir un tableau E de n entier (1≤ n ≤ 50), et réaffecter les éléments positifs de E dans le tableau Ep (de taille n). (Zéro étant considéré comme positif) de même pour les éléments négatifs. Les éléments de E sont des entiers de 3 chiffres.



Solution :

a. Découpage modulaire



b. Analyse principale :

Résultat : affichage (Ep, En, n)

Traitements :

Remplir (Ep, En, E, n)

Saisie (E, n)

Algorithme :

0) Début répartition
1) Saisie (E, n)
2) Remplir (Ep, En, E, n)
3) affichage (Ep, En, n)
4) Fin répartition

T.D.O Globaux

Objet	Type	Rôle
n	Octet	
E	Elément	
Ep	Elément	
En	Elément	

Type
Elément = tableau de 50 entiers

c. Analyse procédure affichage :

Procédure **affichage** (Ep, En : élément, n : octet)

Résultat : Affichage
Traitements :
Prévoir deux boucle pour permettant l'affichage des 2 tableaux Ep et En.

Algorithme :

0) Procédure **affichage** (Ep, En : élément, n : octet)
1) Pour i de 1 à n faire
   Ecrire (Ep[i])
   Fin pour
2) Pour i de 1 à n faire
   Ecrire (En[i])
   Fin pour
3) Fin **affichage**

T.D.O Locaux

Objet	Type	Rôle
i	octet	Compteur

d. Analyse procédure remplir :

Procédure **remplir** (var Ep, En : élément, E : élément, n : octet)

Résultat : Ep, En remplis
Traitements :
Prévoir une boucle pour qui permet d'effectuer un parcours total dans E.
Prévoir 3 compteurs ,
i pour avancer dans E
j pour avancer dans Ep
k pour avancer dans En

Initialiser j, k à 1
Pour chaque case de E faire le test :
Si E[i] > 0 alors Ep[j] ←E[i]
j ← j + 1
Sinon En[k] ← E[i]
k ← k + 1
Fin si

Algorithme :

```
0) Procédure remplir (var Ep, En : élément, E : élément, n : octet)
1) K ← 1
   J ← 1
   Pour i de 1 à n faire
        Si E[i] > 0 alors Ep[j] ← E[i]
                          j ← j + 1
                     Sinon En[k] ← E[i]
                          k ← k + 1
        Fin si
   Fin pour
2) Fin remplir
```

T.D.O Locaux

Objet	Type	Rôle
i	octet	Compteur
j	octet	Compteur
k	octet	Compteur

e. Analyse procédure saisie :

Procédure **saisie** (var E : élément, var n : octet)
Résultat : E, n saisis

Traitements : saisie contrôlée de n [1..50]

Boucle pour jusqu'à n, prévoir une saisie contrôlée sur les éléments de E qui doivent être de 3 chiffres.

Algorithme

```
0) procédure saisie (var E : élément, var n :octet)
1) répéter
        lire (n)
   jusqu'à n dans [1..50]
2) pour i de 1 à n faire
     répéter
        lire (E[i])
     jusqu'à Abs (E[i]) dans [100..999]
   fin pour
3) fin saisie
```

f. Traduction Pascal : (voir fichier : distrib.pas)

- **Application 4**

Exercice 2 page 124

Énoncé :

Soient les suites U et W définies par :

$U_0=A$, $U_1=B$, $Un = (U_{n-1} + U_{n-2})/2$ A et B sont deux entiers données.

$W_0 =A$,

$W_n = U_n - W_{n-1}$

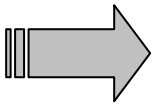
On veut écrire un programme qui permet de calculer la suite W pour un entier n donnée avec $n < 100$.

Solution :

Brouillon : analyse numérique

Pour $U_0 = 3$, $U_1 = -7$ et $n = 4$

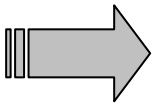
$U_4 = (U_3 + U_2) / 2$ $U_4 = (-4 - 2) / 2 = -3$
$U_3 = (U_2 + U_1) / 2$ $U_3 = (-2 - 7) / 2 = -4$
$U_2 = (U_1 + U_0) / 2$ $U_2 = (-7 + 3) / 2 = -2$



$U_4 = -3$

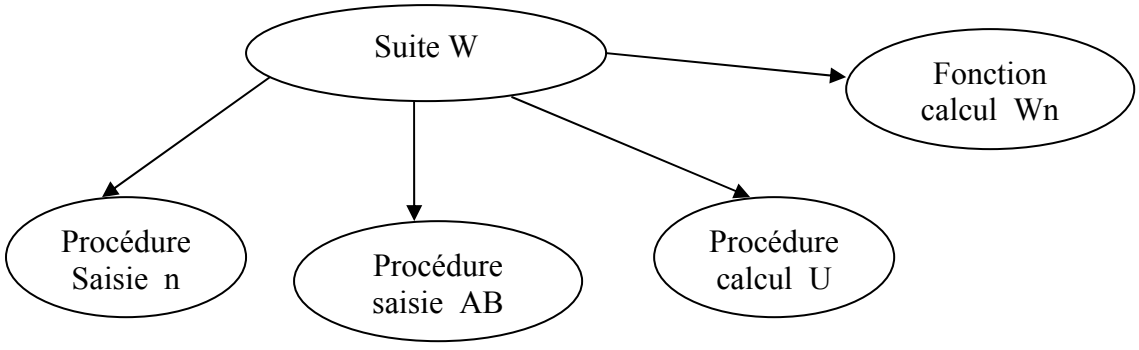
Pour $W_0 = 3$

$W_4 = U_4 - W_3$ $W_4 = -3 + 12 = 9$
$W_3 = U_3 - W_2$ $W_3 = -4 - 8 = -12$
$W_2 = U_2 - W_1$ $W_2 = -2 + 10 = 8$
$W_1 = U_1 - W_0$ $W_1 = -7 - 3 = -10$



$W_4 = 9$

a. Découpage modulaire



b. Analyse principale :

Résultat : écrire ("Wn est égal à : " , calcul_Wn (U,A,n))

Traitements :

calcul_U (U, A, B, n)

saisie_n (n)

saisie_AB (A, B)

Algorithme :

0) Début suiteW
1) saisie_n (n)
2) saisie_AB (A, B)
3) calcul_U (U, A, B, n)
4) écrire ("Wn est égal à : " , calcul_Wn (U,A,n))
5) Fin suiteW

T.D.O Globaux

Objet	Type	Rôle
n	entier	
A, B	entier	
U	Suite	Tableau contenant les termes de U

Type
Suite = tableau de 100 entiers

c. Analyse de la fonction calcul_wn :

Fonction **calcul_Wn** (U: suite, A:entier, n:entier) : entier

Résultat : calcul_Wn

Traitements :

calcul_Wn ← W[i]
Initialiser W[1] avec A
Créer une boucle permettant d'effectuer un parcours total sur W, on calculera la suite w selon la formule :

$W_n = U_n - W_{n-1}$

Algorithme :

0) Fonction **calcul_Wn** (U: suite, A:entier, n:entier) : entier
1) W [1] ← A
2) Pour i de 2 à n+1 faire
   W[i] ← U[i] – W[i-1]
   Fin pour
3) calcul_Wn ← W[i]
4) Fin **calcul_Wn**

T.D.O Locaux

Objet	Type	Rôle
i	entier	Compteur
w	Suite	

d. Analyse de la procédure calcul_U :

Procédure **calcul_U** (var U: suite, A,B : entier, n:entier)

Résultat : Le tableau U rempli

Traitements :

Initialiser U[1] avec A
Initialiser U[2] avec B

Créer une boucle permettant d'effectuer un parcours total sur U, et calculer la suite U selon la formule :

$Un = (U_{n-1} + U_{n-2})/2$

Algorithme :

0) Procédure **calcul_U** (var U: suite, A, B : entier, n:entier)
1) U[1] ← A
2) U[2] ← B

   Pour i de 2 à n+1 faire
   U[i] ← (U[i – 1] + U[i – 2]) div 2
   Fin pour

3) Fin **calcul_U**

T.D.O Locaux

Objet	Type	Rôle
i	entier	Compteur

e. Analyse de la procédure saisie_AB :

Procédure **saisie_AB** (var A, B : entier)

Résultat : A, B saisies
Traitements :

A = donnée
B = donnée

Algorithme :

0) Procédure **saisie_AB** (var A, B : entier)
1) Lire (A)
2) Lire (B)
3) Fin **saisie_AB**

f. Analyse de la procédure saisie_n :

Procédure **saisie_n** (var n : entier)

Résultat : n saisie

Traitements :

Répéter
n = donnée
Jusqu'à (n <100) et (n >1)

Algorithme :

0) Procédure **saisie_n** (var n : entier)
1) Répéter
n = donnée
Jusqu'à (n <100) et (n >1)
2) Fin **saisie_n**

g. Traduction Pascal : (voir fichier : suitew.pas)

Chapitre n°3 :

Les structures de contrôle conditionnelles

Objectifs du cours :

- Maîtriser la structure Si.
- Maîtriser la structure Selon.

Leçon 1 La structure de contrôle conditionnelle simple

I - La forme réduite

1. Syntaxe et vocabulaire : (voir livre page 72, 73)
2. Définition : (voir livre page 75)
3. Remarques

- L'initialisation est le fait d'affecter une valeur initiale à un objet.
- En Pascal, si le traitement après "Then" est composé d'une seule instruction, cette dernière peut être présentée sans les deux délimiteurs Begin et End

Activité 1 :

Écrire une analyse qui permet de saisir un caractère et une chaîne puis afficher un message pour dire si le caractère existe dans la chaîne ou non.

a. Analyse

	Nom **: recherche**	
S	**L.D.E**	**O.U**
4	Résultat = Ecrire (c, msg, ch)	c
3	msg = [msg ← " N'existe pas "]	ch
	Si POS (c, ch) ≠ 0 alors msg ← " Existe "	msg
	Fin si	
1	c = Donnée ("Donner un caractère")	
2	ch = Donnée ("Donner une chaîne")	
5	Fin **recherche**	

T.D.O

Objet	Type	Rôle
ch	Chaîne	
c	Caractère	Caractère recherché
msg	Chaîne	message

b. Algorithme

0) **Début** recherche
1) écrire ("Donner un caractère")
   Lire (c)
2) écrire ("Donner une chaîne")
   Lire (ch)
3) msg ← " N'existe pas "
   Si POS (c, ch) <> 0 alors msg ← " Existe "
   Fin si
4) Ecrire (c, msg, ch)
5) **Fin** recherche

c. Traduction en Pascal (voir fichier : if_pos.pas)

II - La forme alternative

1. Syntaxe et vocabulaire : (voir livre page 77, 78)
2. Définition : (voir livre page 80)

Activité 2 :

Saisir une chaîne de caractère et vérifier si elle est composée de plusieurs mots.

a. Analyse

	Nom **: phrase**	
S	L.D.E	O.U
	Résultat = Affichage	
2	Affichage = [ ]	
	Si POS (" ", ch) = 0 alors écrire (" Mot ")	ch
	Sinon écrire (" Phrase ")	
	Fin si	
1	ch = Donnée ("Donner une chaîne : ")	
3	Fin **phrase**	

T.D.O

Objet	Type	Rôle
ch	Chaîne	

b. Algorithme

0) **Début**
1) écrire ("Donner une chaîne")
   Lire (ch)
2) Si POS (" ", ch) = 0 alors écrire ("Votre chaîne est composée d'un seul mot ")
   Sinon écrire ("Votre chaîne est composée de plusieurs mots ")
   Fin si
3) **Fin**

c. Traduction en Pascal (voir fichier : phrase1.pas)

- Cas particulier :

ch contient "" message ⌦ "Votre chaîne est composée d'un seul mot" Faux !

ch contient " o " message ⌦ "Votre chaîne est composée de plusieurs mots" Faux !

ch contient " " message ⌦ "Votre chaîne est composée de plusieurs mots" Faux !

Version 2

0) **Début**
1) écrire ("Donner une chaîne"), Lire (ch)
2) Tant que (ch [1] = " ") et (long (ch) ≠ 0) faire
   Efface (ch, 1, 1)
   Fin Tant que
3) Tant que ch [Long (ch)] = " " et (long (ch) ≠ 0) faire
   Efface (ch, Long (ch), 1)
   Fin Tant que
4) Si Long (ch) = 0 alors écrire ("La chaîne est vide ")
   Sinon Si POS (" ", ch) = 0 alors écrire ("Votre chaîne est composée d'un seul mot")
   Sinon écrire ("Votre chaîne est composée de plusieurs mots ")
   Fin si
5) **Fin**

d. Traduction en Pascal (voir fichier : phrase2.pas)

Leçon 2 La structure de contrôle conditionnelle généralisée

I - Vocabulaire et syntaxe (voir livre page 86, 87)

II - Définition (voir livre page 90)

Activité 1 :

Écrire un programme qui affiche l'heure système, Avancer cet horaire d'une seconde et l'afficher au milieu de la fenêtre d'exécution.

Exemple : 15:34:26 ↳ 15:34:27

a. Analyse

	Nom : **heure_système**	
S	L.D.E	O.U
4	Résultat = Écrire (h, ":", m, ":", s)	h
3	h, m, s = [ ]	m
	Si s< 59 alors s ← s + 1	s
	Sinon Si m<59 alors	
	s ← 0	
	m ← m + 1	
	Sinon Si h<23 alors	
	s ← 0	
	m ← 0	
	h ← h + 1	
	Sinon	
	s ← 0	
	m ← 0	
	h ← 0	
	Fin si	
2	Écrire (h, ":", m, ":", s)	
1	Gettime (h, m, s, c)	c
5	Fin **heure_système**	

T.D.O

Objet	Type	Rôle
h	mot	Heure
m	mot	Minute
S	mot	Seconde
c	mot	Centième de seconde

b. Algorithme

0) **Début** heure_système
1) Gettime (h, m, s, c)
2) Écrire (h, ":", m, ":", s)
3) Si s< 59 alors s ← s + 1

```
Sinon Si m<59 alors
                s ← 0
                m ← m + 1
            Sinon Si h<23 alors
                        s ← 0
                        m ← 0
                        h ← h + 1
                    Sinon
                        s ← 0
                        m ← 0
                        h ← 0
```

Fin si

4) Écrire (h, ":", m, ":", s)
5) **Fin** heure_système

c. Traduction en Pascal (voir fichier : heure_sy.pas)

Activité 2 : Écrire une analyse, un algorithme et la traduction en pascal du programme intitulé équation1, qui détermine est affiche les solutions d'une équation du 1er ordre de la forme a x + b = 0. Avec a et b ∈ IR

a. Pré analyse :

Résultat : Afficher x

Traitement : 1er cas a ≠ 0 alors $x = -\frac{b}{a}$

2ème cas a = 0 alors 1) b = 0 alors écrire IR

2) b ≠ 0 alors écrire { }

Données : a et b entier.

b. Analyse :

Nom : équation1

S	L.D.E	O.U
	Résultat = R	
3	R = [ ] Si a ≠ 0 alors écrire (-b/a)	a
	Sinon si b = 0 alors écrire ("IR")	b
	Sinon écrire ("{ }")	
	Fin si	
2	a = Donnée ("Donner a : ")	
1	b = Donnée ("Donner b : ")	
4	**Fin** équation1	

Objet	Type	Rôle
a , b	Réel	

c. Algorithme

0) **Début** équation1
1) écrire ("Donner a : ") , lire (a)
2) écrire ("Donner b : ") , lire (b)
3) Si a < > 0 alors écrire (-b/a)
   Sinon Si b = 0 alors écrire ("IR")
   Sinon écrire ("{ }")
   Fin si
4) **Fin** équation1

d. Traduction Pascal

```
Program equation1;
uses wincrt;
var
    a,b : real;
begin
      write ('Donner a : ');
      readln (a);
      write ('Donner b : ');
      readln (b);

      if a<>0 then writeln(-b/a:5:2)
            else if b = 0 then writeln('x appartient à IR')
                          else writeln('la solution est l''ensemble vide');
End.
```

La structure de contrôle conditionnelle à choix

I - Introduction

Activité 1 :

Ecrire un programme qui saisit le numéro d'un mois et affiche la saison correspondante.

Exemple : si numéro du mois = 8, le programme affichera : saison été.

a. Pré analyse

Résultat :

- Afficher un message qui indique le nom de la saison du mois

Traitements :

- mois ∈ {12, 1, 2} ⇔ afficher "Saison Hiver"
- mois ∈ {3, 4, 5} ⇔ afficher "Saison Printemps"
- mois ∈ {6, 7, 8} ⇔ afficher "Saison Eté"
- mois ∈ {9, 10, 11} ⇔ afficher "Saison Automne"

Donnée :

- mois

b. Analyse

	Nom : Saison	
S	**L.D.E**	**O.U**
	Résultat = Affichage	
2	Affichage = [ ] Selon **mois** Faire	mois
	12, 1, 2 : écrire ("Saison Hiver")	
	3.. 5 : écrire ("Saison Printemps")	
	6 .. 8 : écrire ("Saison Eté")	
	9.. 11 : écrire ("Saison Automne")	
	Sinon	
	Ecrire ("Erreur !")	
	Fin Selon	
1	mois = Donnée ("saisir le n° d'un mois : ")	
3	**Fin** Saison	

T. D. O

Objets	Type	Rôle
mois	Entier	

c. Algorithme

0) Début Saison

1) Ecrire ("saisir votre Mois"), lire (**mois**)
2) Selon **mois** Faire

12, 1, 2 : écrire ("Saison Hiver")

3 .. 5 : écrire ("Saison Printemps")

6 .. 8 : écrire ("Saison Eté")

9 .. 11 : écrire ("Saison Automne")

Sinon

Ecrire ("Erreur !")

Fin Selon

3) **Fin** Saison

d. Traduction pascal (voir fichier : saison.pas)

II - Syntaxe et commentaires (voir livre page 96)

III - Définition (voir livre page 97)

Au niveau de l'analyse et de l'algorithme	Au niveau de turbo pascal

[ ]
Selon sélecteur **faire**
Valeur 1 : Action1
Valeur 2 : Action2-1
Action2-2
……..
Action2-n
Valeur 3, Valeur 4, Valeur 5 : Action3
Valeur 6 .. Valeur 7 : Action4
……
Valeur n : Action n
Sinon
Action R
Fin Selon
	{Initialisation};
	Case sélecteur **OF**
	Valeur 1 : Action1;
	Valeur 2 : **Begin**
	Action2-1 ;
	Action2-2 ;
	…..…….. ;
	Action2-n ;
	End ;
	Valeur 3, Valeur 4, Valeur 5 : Action3 ;
	Valeur 6 .. Valeur 7 : Action4 ;
	…… ;
	Valeur n : Action n ;
	Else
	Action R ;
	End ;