Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Ferhat Abbes – Sétif-1
Faculté des Sciences
Département d'Informatique

**1 ère année LMD**

## Support de TD
## Algorithmique et Structures de Données 1 (ASD1)

# Rappel de cours et exercices corrigés

Dr. Samir FENANIR

*Année Universitaire 2022/2023*

# Table des matières

## Avant-propos

Ce polycopié constitue un support de TD (Travaux dirigés) sur l'Algorithmique et Structures de Données1 (ASD1) pour des étudiants de la première année LMD (socle commun Mathématiques et Informatique). Il s'adresse aux étudiants n'ayant aucune connaissance en programmation et à ceux qui souhaitent en savoir plus sur l'art de la programmation. Il présente les notions fondamentales de l'algorithmique est de la programmation en langage C++. Il a pour objectif d'exposer en douceur la manière d'analyser et de résoudre un problème informatique. Il comprend de nombreux exemples et exercices. Il est à noter que La solution proposée à chaque exercice n'est pas unique et dans plusieurs cas elle n'est optimale car on a toujours privilégié l'apport pédagogique et la simplicité. Ce polycopié est structuré en six chapitres comme suit :

Le premier chapitre commence par une introduction générale, définissant les concepts fondamentaux de l'informatique, y compris les ordinateurs, leurs composants, ainsi que les systèmes informatiques, qui englobent le matériel et les logiciels.

Le second chapitre introduit les notions de base de l'écriture d'un algorithme simple, en présentant la structure général d'un algorithme et les différentes parties qui le composent : variables, constantes, expressions et instructions de base.

Le troisième chapitre présente les structures de contrôle qui permettent de modifier l'ordre d'exécution des instructions en fonction des valeurs relevées et des conditions prédéfinies en expliquant comment utiliser les tests et les boucles pour évaluer des conditions, puis réagir en conséquence.

Le dernier chapitre aborde l'utilisation des structures de données, notamment les tableaux, les chaines de caractères et les enregistrements.

# Chapitre 1 : Introduction Générale

## 1. Introduction

L'informatique intervient aujourd'hui dans tous les domaines de la vie, y compris celui de l'éducation, le commerce, la médecine, la télécommunication, etc. Aussi, avec l'accessibilité de l'Internet, les informations sont disponibles partout et pour tous. Ces informations sont traitées et présentées grâce à un ordinateur. Cependant, l'ordinateur seul ne suffit pas, il ne peut en effet qu'exécuter les ordres qui lui sont fournis par l'intermédiaire d'un programme.

## 2. C'est quoi l'informatique

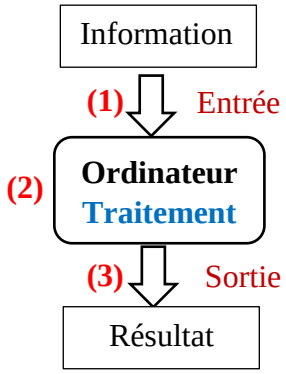
Le terme *Informatique* est composé de 02 parties: *Infor* : **Information** et *Matique* : **Automatique.** C'est-à-dire: Traitement Automatique de l'Information.

**Traitement** : C'est une action d'examiner et de régler une question ou un problème. ***Exemple:*** Un médecin traite un malade.

**Information** : C'est une connaissance qui peut être: un nombre, un texte, une image, un son, une vidéo,…

**Automatique**: Le traitement de l'information s'effectue par une machine, cette machine est appelée: **ordinateur.**

## 3. C'est quoi un ordinateur




L'ordinateur est une machine électronique capable de:

(1) Recevoir des informations,

(2) Traiter ces informations,

(3) Fournir le résultat en sortie.

**Exemple:** Résoudre l'équation: $Ax^2+Bx+C=0$


## 4. Système informatique

Le système informatique se compose de 02 parties: **Matériel** et **Logiciel**.

### 4.1. Matériel /(Hardware) / (عتاد)

Comporte tous les composants matériels de l'ordinateur.

- **Périphériques d'entrée :** Sont des composants rattachés à l'unité centrale servent à entrer les informations à l'ordinateur, parmi lesquelles : Clavier, Souris, Microphone, Scanner, ….

- **Périphériques de sortie :** Sont des composants rattachés à l'unité centrale servent à sortir le résultat à l'utilisateur, parmi lesquelles : Ecran, Imprimante, ….
- **Composants de Traitement :** Ils se trouvent dans l'unité centrale, ils permettent de traiter les informations et de communiquer avec les périphérique d'entrée et de sortie. Parmi ces composants: Microprocesseur, Mémoire centrale, Unités d'entrée et sortie
- **Supports de Stockage :** Ils permettent de conserver des informations d'une façon permanente. Parmi ces supports: Disquette, Disque dur, CD, DVD, Flash disque, Carte mémoire, …

### 4.1.1. Capacité de stockage

Quantité d'informations qui peuvent être stockées, mesurée par l'**Octet :**

**1 Octet (Byte) = 8 Bits.**

### 4.1.2. Bit (BINARY DIGIT)

C'est un chiffre binaire peut valoir **0** (Le courant ne passe pas) ou **1** (Le courant passe). A l'aide d'un **octet** on peut coder un caractère.

**Exemple :** Le caractère **A** codifié s'écrit **01000001**.

### 4.1.3. Multiples de l'octet

- 1 Kilo octets (Ko) = $2^{10}$ octets= 1024 octets
- 1 Méga octets (Mo) = $2^{20}$ octets =1024 Ko
- 1 Giga octets (Go) = $2^{30}$ octets = 1024 Mo
- 1 Téra octets (To) = $2^{40}$ octets = 1024 Go
- 1 Péta octets (Po) = $2^{50}$ octets = 1024 To

## 4.2. Logiciel / (Software) / (برمجة)

Comporte tous les programmes de base et les programmes des utilisateurs. Un logiciel (programme) est un ensemble d'instructions que l'ordinateur doit les exécuter pour résoudre un problème donné. Il y a 02 types de programmes:

- **Programmes de base**
- **Programmes des utilisateurs (Applications)**

### 4.2.1. Programmes de base

Ensemble des programmes intermédiaires entre l'utilisateur et le matériel de l'ordinateur, sont à la base de toute exploitation. Ces programmes sont appelés: **Système d'exploitation ( نظام التشغيل), Exemples: DOS, WINDOWS, UNIX, ….**

### 4.2.2. Programmes d'application / البرامج التطبيقية

Sont tous les programmes destinés aux applications des utilisateurs, tels que:

- Bureautique (Traitement de texte, tableurs,….)
- Technique (Langages de programmation, Antivirus,…)

- Commercial (Gestion du stock, Gestion des assurés,…)
- Éducatif (encyclopédie, dictionnaire,…)
- Jeux

## Série d'exercices 1

*Répondez aux questions suivantes :*

1. Définissez le terme informatique.
2. Qu'est-ce qu'un système informatique?
3. Qu'est-ce qu'un ordinateur?
4. Quels sont les composants d'un ordinateur ?
5. Qu'est-ce qu'un algorithme?
6. Qu'est-ce qu'un langage de programmation?
7. Quels sont les 3 grandes étapes pour la création d'un programme?
8. Quelle est la différence entre un programmeur et un utilisateur
9. Quelle est la différence entre le langage machine et le langage évolué?
10. Quelle est la différence entre un compilateur et interpréteur?

## Solutions

1. Le terme informatique est composé de deux mots: information et automatique; c'est la science de traitement de l'information à l'aide d'une machine appelée "ordinateur".

2. Qu'est-ce qu'un système informatique? c'est l'ensemble des moyens ***matériels*** et ***logiciels*** nécessaire pour satisfaire les besoins informatiques des utilisateurs.

3. Qu'est-ce qu'un ordinateur? : est une machine électronique permettant de traiter et manipuler des données sous format binaire (0,1).

4. Les composants d'un ordinateur :
   - Périphériques d'entrée : servent à obtenir des informations (ou données) pour l'ordinateur tel que clavier, souris, scanner micro, webcam, etc.
   - Périphériques de sortie : servent à faire sortir des informations de l'ordinateur tel que écran, imprimante, haut-parleur, etc.
   - Composants de traitement : traitent les informations, ex : Processeur, RAM, Cartes d'interface, …
   - Supports de stockage : pour stocker les informations, ex : Disque dur, flash disque, DVD, ….

5. Un algorithme est une procédure étape par étape pour résoudre un problème donné. Nous pouvons trouver des algorithmes partout dans notre vie, pas seulement en informatique. Par exemple, le processus de préparation d'une recette de cuisine peut être exprimé sous forme d'algorithme. Certaines étapes, dans un certain ordre, doivent être suivies afin d'atteindre notre objectif.
6. Un langage de programmation est un ensemble de mots et de symboles qui servent à exprimer des opérations, des instructions et des données comprises par un ordinateur, par exemple: Pascal, C, C++, Delphi, Java, …etc.
7. Il y a trois étapes pour la création d'un programme : 1) Ecriture du programme en utilisant un éditeur spécifique ; 2) Compilation : permet de corriger les erreurs syntaxique et de convertir le programme en langage machine ; 3) Exécution : sortie des résultats.
8. Le programmeur est celui qui écrit un programme, et l'utilisateur est celui qui utilise ce programme. Par exemple. Considérons un vrai programme informatique. Prenons un jeu vidéo, dans lequel une personne (le programmeur) écrit le programme dans un langage informatique, quelqu'un autre (l'utilisateur) l'utilise ou joue avec en utilisant un ordinateur.
9. La différence entre un langage machine et un langage évolué :

Langage machine:

- est une suite de 0 et de 1 (langage bas niveau);
- Un programme écrit en langage machine est directement exécutable par l'ordinateur; c.à.d. compréhensible par la machine.
- Ecrire un programme en langage machine est une tâche délicate, qui peut être réalisée seulement par des spécialistes.

Langage évolué:

- Est un langage compréhensible par les programmeurs (être humain); aussi nommé langage de haut niveau
- Pour exécuter un programme écrit en langage évolué, il doit être traduit en langage machine; cette opération est la compilation

10. La différence entre un compilateur et interpréteur :

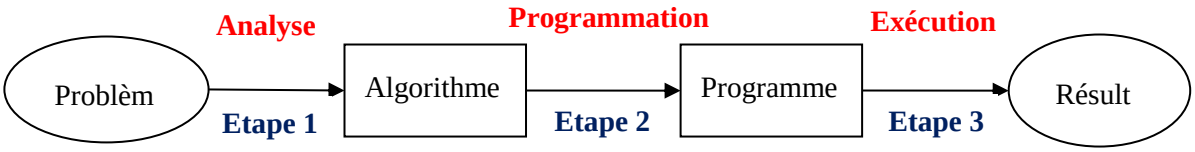
- Un compilateur traduit des instructions écrites dans un langage de haut niveau en un programme en langage machine distinct. Nous pouvons ensuite exécuter le programme en langage machine à tout moment.

- Un interpréteur traduit et exécute simultanément les instructions écrites dans un langage de haut niveau. Au fur et à mesure que l'interpréteur lit chaque instruction individuelle dans le programme de langage de haut niveau, il la traduit en un code de langage machine, puis l'exécute directement.

# Chapitre 2 : Notions Algorithmiques

## 1. Résolution d'un problème en informatique

Plusieurs étapes sont nécessaires pour résoudre un problème en informatique (figure 2) :




**Etape 1 (Analyse) :** Elle consiste à trouver le moyen de passer des données aux résultats en répondant aux 3 questions suivantes :

- Quelles sont les entrées (données)?
- Quelles sont les opérations de traitement?
- Quelles sont les sorties (résultats)?

Le résultat de cette étape est l'algorithme qui est un ensemble d'instructions écrites en pseudo code et décrivant la méthode de résolution du problème.

**Etape 2 (Programmation) :** elle consiste à traduire l'algorithme dans un langage de programmation spécifique. Ainsi, étant donné que le langage de programmation est destiné à l'ordinateur, il doit respecter une syntaxe stricte. Le résultat de la phase de programmation est un programme; c'est l'ensemble d'instructions qui seront exécutées par l'ordinateur.

**Etape 3 (Exécution) :** elle permet à l'utilisateur de voir les résultats après avoir saisi les données nécessaires à l'ordinateur. Si les résultats obtenus ne sont pas ceux attendus, on dira qu'il existe des erreurs d'exécution. Dans ce cas, il faut revoir si l'algorithme a été bien traduit, ou encore est-ce qu'il y a eu une bonne analyse.

## 2. Structure générale d'un algorithme

Un algorithme est composé de 3 parties principales :

**Algorithme** nom_algorithme **Entête**

---

**Constante** Liste des constantes **Déclaration des constantes**

**Variable** Liste des variables **Déclaration des variables**

---

**Début**

Instruction 1;

Instruction 2; **Corps de l'algorithme**

……

Instruction n;

**Fin**

- **Entête :** sert de donner un nom à l'algorithme elle est précédé par le mot **Algorithme**.
- **Déclaration :** sert à déclarer des variables et des constantes.
- **Corps de l'algorithme :** comprend les instructions de l'algorithme, elle est délimitée par les mots : **Début** et **Fin**.

## Série d'exercices N°2

***Exercice 1***

- Donner les différentes conditions pour qu'un terme puisse être un identificateur.

- Est-ce qu'on peut utiliser les termes suivants comme identificateurs?

| | | | | | | |
|---|---|---|---|---|---|---|
| Numéro | Nom-Client | $X^2$ | rouge | début | 1surface | Nom Pren |
| N1 | C++ | faux | variable | A..Z | Algerien | A_1 |

***Exercice 2***

On veut informatiser la gestion des comptes d'une banque. Pour cela, pour chaque client, on a besoins des informations suivantes:

- Numéro du compte;
- Nom et Prénom du client ;
- Age du client ;
- Avoir du compte (somme déposée en DA);
- Etat civil du client (C : Célibataire, M : Marié, D : Divorcé, V : Veuf);
- Actif ou non Actif;

Donner un identificateur pour chacune de ces informations avec son type.

***Exercice 3***

a) Donner les expressions algorithmiques des expressions mathématiques suivantes, avec leurs évaluations.

$B^2-4AC$ $2\pi R^2$ $\frac{n(n-1)x^2}{2}$ $\frac{5\frac{3x^2+5x+2}{7w-\frac{1}{z}}-z}{4\frac{3+x}{7}}$

b) Sachant que a = 4, b = 5, c = -1 et d = 0, évaluer les expressions logiques suivantes :
- (a < b) **ET** (c ≥ d)
- **NON** (a < b) **OU** ( c ≠ b )
- **NON (**(a < b) **OU** ( c ≠ b ))
- **NON (**(a≠ b) **ET** (a*c < d))

c) Sachant que : A= VRAI , B=FAUX, C= VRAI ; évaluer les expressions logiques suivantes:
- (A **OU** B) **ET** (A **OU** C)
- (NON A ET B) OU (A ET NON B)
- (A **ET** B) **ET** (B **ET** C) **OU** (C **ET** A)
- (A **ET** B) **OU** (B **ET** C) **ET** (C **ET** A)
- (A **OU** (A **ET** B)) **ET** (A **OU** (B **ET** C))

***Exercice 4***

Formuler les phrases suivantes en expressions logiques:

1. Un nombre entier A est impair;
2. Un nombre entier A est multiple d'un autre B;
3. Un nombre réel X appartient à l'intervalle ] 0, 1];
4. La moyenne est sup ou égal à 10 ou le crédit total est sup ou égal à 30
5. Le reste de la division de A sur B est divisible par 4;
6. Le milieu de l'intervalle [x1, x2] est plus grand que 0;
7. A est supérieur à 5 et le quotient entier de A sur B est multiple de 3;
8. La moyenne des trois notes N1, N2 et N3 est supérieur à 10, et au moins l'une de ces notes est supérieur ou égale à 14;
9. Le théorème de Pythagore pour un triangle (ABC) rectangle en A.
10. Un caractère alphabétique

***Exercice 5***

1) Faites l'analyse de chacun des problèmes suivants en dégageant les données, les résultats et les formules de calcul :

1. La somme et la moyenne de trois nombres réels ;
2. Le périmètre d'un cercle ;
3. Permutation des valeurs de deux variables.
4. Le nombre d'heures, de minutes et de secondes à partir d'un nombre donné en secondes.
5. Convertir une somme d'argent de l'euro en dinar
6. Le prix TTC (Toute Taxe Comprise) d'un produit.
7. Coller deux mots pour en générer un troisième.
8. La racine d'une équation de deuxième degré ;

9. Afficher si la différence entre deux nombres a et b (a-b) est positive, nulle, ou négative.
10. Afficher si un étudiant est admis ou ajourné selon sa moyenne.

2) Ecrire les algorithmes correspondant aux 5 premiers problèmes précédents

***Exercice 6***

Donnez la trace d'exécution des algorithmes suivants :

```
Algorithme Algo_01;
   Variables A, B : entier ;
Début
      A← - Carre(2) ;
      B← A * A ;
      Ecrire (A, B, A-B, A*B) ;
Fin
```

```
Algorithme Algo_02 ;
Variables A, B, C : entier ;
          D : booléen ;
Début
    A ← 5 ;
    B ← 6 ;
    C ← A + B*2 + 3 ;
    D ← (C mod A) < (C div B) ;
    Ecrire (A,B,C,D) ;
Fin
```

## Solution

***Exercice 1:***

- Un identificateur doit respecter les trois conditions suivantes:

  - Il est représenté par une suite quelconque de caractères alphanumériques (alphabétiques: majuscules –A.. Z– ou minuscules –a .. z– et numériques : –0.. 9–); ne doit pas contenir de caractère spécial excepté le _
  - Commençant obligatoirement par une lettre (a .. z) ou (A .. Z);
  - Doit être différent des mots réservés;

*De préférence,* le nom est choisi en rapport avec le contenu de l'objet.

- Est-ce qu'on peut utiliser les termes suivants comme identificateurs?

| Mot | Peut ou non être identificateur | Cause | Correction |
|---|---|---|---|
| Rouge | Oui | | |
| N1 | | | |
| Algerien | | | |
| A_1 | | | |
| Numéro | Non | Caractère spécial (é) | Numero |
| Nom-Client | | Caractère spécial (-) | Nom_Client |
| $X^2$ | | Caractère spécial ($^2$) | X2 |
| debut | | Mot réservé | D |

| | | | |
|---|---|---|---|
| 1surface | | Commence par un chiffre | surface1 |
| Nom Pren | | Caractère spécial (espace) | Nom_Pren |
| C++ | | Caractère spécial (+) | C2Plus |
| faux | | Mot réservé | F |
| variable | | Mot réservé | V |
| A..Z | | Caractère spécial (..) | A_Z |

***Exercice 2***

Identificateurs pour les informations et leurs types:

| ***Information*** | ***Identificateur*** | ***Type*** |
|---|---|---|
| Numéro du compte | Num_Cpt | Chaîne de caractère |
| Nom et prénom du client | Nom_Pren | Chaîne de caractère |
| Age du client | Age | entier |
| Avoir du compte | Avoir | Réel |
| Etat civil du client | Etat_Civ | caractère |
| Actif ou non actif | Actif | Booléen |

***Exercice 3***

a) Donner les expressions algorithmiques des expressions mathématiques suivantes, avec leurs évaluations.

$B^2 - 4AC$ B*B-4*A*C évaluation : (B*B)-(4*A*C) *** prioritaire à -**

$2\pi R^2$ 2*Pi*R*R Pi est une constante = 3 ,14

$\frac{n(n-1)x^2}{2}$ (n*(n-1)*x*x) / 2 évaluation : **- , * , *, * , /**

$$\frac{5\,\frac{3x^2+5x+2}{7w-\frac{1}{z}}-z}{4\,\frac{3+x}{7}}$$

**(5 * (3*x*x+5*x+2) / (7*w-1/z)-z) / (4 * (3+x) / 7)**

Evaluation : **(***++) ; (*/-) ; (* / -) ; (+) ; (*/)**

b) Sachant que a = 4, b = 5, c = -1 et d = 0, évaluer les expressions logiques suivantes :

- (a < b) **ET** (c ≥ d) → (4 < 5) **ET** (-1 ≥ 0) → VRAI **ET** FAUX → **FAUX**
- **NON** (a < b) **OU** ( c ≠ b ) → **NON** (4 < 5) **OU** ( -1 ≠ 5 ) → **NON** (VRAI) **OU** ( VRAI ) FAUX **OU** VRAI → **VRAI**
- **NON (**(a < b) **OU** ( c ≠ b )) → **NON (**(4 < 5) **OU** ( -1 ≠ 5 )) → **NON (**VRAI **OU** VRAI) → **NON** VRAI → **FAUX**
- **NON (**(a≠ b) **ET** (a*c < d)) → **NON (**(4 ≠ 5) **ET** (4*-1 < 0)) → **NON (**VRAI **ET** VRAI) → **NON (**VRAI**)** → **FAUX**

***Exercice 4***

Formuler les phrases suivantes en expressions logiques:

1. Un nombre entier A est impair; **(A mod 2 = 1)**
2. Un nombre entier A est multiple d'un autre B; **(A mod B = 0)**

3. Un nombre réel X appartient à l'intervalle ] 0, 1]; **(X > 0) et (X <=1)**
4. La moyenne est sup ou égal à 10 ou le crédit total est sup ou égal à 30 **(Moy>=10) ou (Credit>=30)**
5. Le reste de la division de A sur B est divisible par 4; **(A mod B) mod 4 = 0**
6. Le milieu de l'intervalle [x1, x2] est plus grand que 0; **(x1 + x2)/2 > 0**
7. A est supérieur à 5 et le quotient entier de A sur B est multiple de 3; **(A > 5) et ((A div B) mod 3 =0)**
8. La moyenne des trois notes N1, N2 et N3 est supérieur à 10, et au moins l'une de ces notes est supérieur ou égale à 14; **(((N1+ N2+ N3)/3) > 10) et ((N1>=14) ou (N2 >=14) ou (N3 >=14))**
9. Le théorème de Pythagore pour un triangle (ABC) rectangle en A. **(AB*AB+AC*AC=BC*BC)**
10. Un caractère alphabétique (C **>= 'a') et (C <='z') ou** (C **>= 'A') et (C <='Z')**

***Exercice 5***

1) Faites l'analyse des problèmes suivants en dégageant les données, les résultats et les formules de calcul :

| Problème | Données (Entrée) | Résultats (sorties) | Formules (Traitement) |
|---|---|---|---|
| 1 | Les trois réels R1, R2, R3 | La somme S (réel)<br>La moyenne M (réel) | S = R1 + R2 + R3<br>M = S / 3 |
| 2 | Le rayon R (réel) | Le périmètre P (réel) | P = 2 * Pi * R |
| 3 | Deux valeurs A, B | valeurs A, B<br>(Echangés) | C ← A  A ← B  B ← C |
| 4 | S (seconde) | H (heure)<br>M(minute) S (seconde) | H= S div 3600<br>M = S mod 3600 |
| 5 | E (Euro) | D (Dinar) | D=E * Taux  (Taux de conversion) |
| 6 | Le prix P (réel)<br>Taux des taxes TVA (réel) | Le prix TTC :<br>P_TTC(réel) | P_TTC = P + (P * TVA) |
| 7 | Mot1, Mot2 (chaine) | Mot3 (chaine) | Mot3=Mot1+Mot2 |
| 8 | Les trois réels A, B, C | Les solutions<br>X1,X2(réels) | Delta = B * B – 4 * A * C<br>X1 = (-B - racine (Delta))/(2*A)<br>X2 = (-B + racine (Delta))/(2*A) |
| 9 | Deux nombres A, B | D est<br>Positif, Négatif ou Nulle | D = A-B<br>Si D>0 Alors Positif<br>Si D<0 Alors Négatif<br>Si D=0 Alors Nulle |
| 10 | Moyenne (Réel) | Admis ou Ajourné | Si Moyenne>=10 Alors Admis<br>Sinon Ajourné |

**2)** Ecrire les algorithmes correspondant aux 5 premiers problèmes précédents :

```
Algorithme Pb1 ;
Variables R1, R2, R3, S, M : réel ;
Debut
    Lire (R1, R2, R3) ;
    S ← R1+R2+R3 ;
    M ← S / 3 ;
    Ecrire (S, M) ;
Fin
```

```
Algorithme Pb2 ;
Constant Pi = 3.14 ;
Variables R, P : réel ;
Debut
    Lire (R) ;
    P ← 2*Pi*R ;
    Ecrire (P) ;
Fin
```

```
Algorithme Pb3 ;
Variables A, B, C : réel ;
Debut
    Lire (A, B) ;
    C ← A ;
    A ← B;
    B ← C;
    Ecrire (A, B) ;
Fin
```

```
Algorithme Pb4 ;
Variables S, M, H : entier ;
Debut
    Lire (S) ;
    H ← S div 3600 ;
    S ← S mod 3600 ;
    M ← S div 60 ;
    S ← S mod 60;
    Ecrire (H, M, S) ;
Fin
```

```
Algorithme Pb5 ;
Constant Taux = 155
Variables D, E : réel ;
Debut
    Ecrire (”Donner une valeur en Euro) ;
    Lire (E) ;
    D ← E * Taux ;
    Ecrire (E,” Euro = ”, D, ” Dinar”) ;
Fin
```

***Exercice 6***

La trace d’exécution des algorithmes :

**a :**

| Etape | A | B | Ecran |
|---|---|---|---|
| 1 | -4 | | |
| 2 | -4 | 16 | |
| 3 | -4 | 16 | -4 16 -20 -64 |

**b :**

| Etape | A | B | C | D | Ecran |
|---|---|---|---|---|---|
| 1 | 5 | | | | |
| 2 | 5 | 6 | | | |
| 3 | 5 | 6 | 20 | | |
| 4 | 5 | 6 | 20 | Vrai | |
| 5 | 5 | 6 | 20 | Vrai | 5 6 20 Vrai |

# Chapitre 3 : Structures de contrôle

## 1. Les Tests

Il y a 2 formes pour la structure de test :

**(1)**

**Si** Condition **Alors**
Instructions 1
**Sinon**
Instructions 2
**FinSi**

**(2)**

**Si** Condition **Alors**
Instructions
**FinSi**

La **condition** est une expression logique (Ex: note>=10) qui peut être vraie ou fausse

Si la condition est **vraie**, les instructions 1 qui seront exécutées

Si la condition est **fausse**, les instructions 2 qui seront exécutées

### 1.2. Conditions composées

Une condition composée est une condition formée de plusieurs conditions simples reliées par les opérateurs logiques: **Et**, **Ou**, **Non**.

**Exemples :**

- x est compris entre 2 et 6 : Elle est vraie si les 2 conditions sont vraies.
- n divisible par 3 ou par 2 : Elle est fausse si les 2 conditions sont fausses.
- Non Condition : C'est la négation de la condition.

### 1.3. Tests Imbriqués

Les tests peuvent avoir un degré quelconque d'imbrications, par exemple :

```
Si  Condition1 alors
    Si  Condition2  alors
        Instructions A
    Sinon
        Instructions B
    Finsi
 Sinon
    Si  Condition3  alors
        Instructions C
    Finsi
 Finsi
```

**Pour exécuter instructions A:**
**Condition1 = Vrai**
**Condition2 = Vrai**

**Pour exécuter instructions B:**
**Condition1 = Vrai**
**Condition2 = Faux**

**Pour exécuter instructions C:**
**Condition1 = Faux**
**Condition3 = Vrai**

### 1.4. La Structure *Selon*

La structure *Selon* permet de choisir le traitement à effectuer en fonction de la valeur ou de l'intervalle de valeur.

```
Selon sélecteur faire
   Valeur1 : Traitement1 ;
   Valeur2 : Traitement2 ;
   ……..
   ValeurN : Traitement N ;
Sinon
   Traitement N+1 ;
FinSelon
```

- Le sélecteur est une expression de type (entier, caractère ou booléen).
- La partie sinon est facultative.
- On peut utiliser un ensemble de valeurs  sous forme: (Vi, Vj ou Vi.. Vf).

## 2. Les boucles

Les boucles servent à répéter l'exécution d'un bloc d'instructions un certain nombre de fois. Il y a 03 types: **Boucle Tant que**, **Boucle Pour** et **Boucle répéter.**

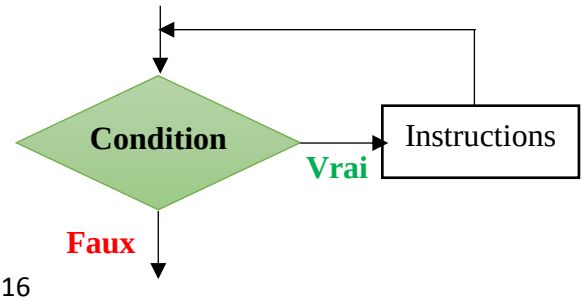
### 2.1. La boucle *Tantque*

On répète des instructions tant qu'une condition est vraie, On sort de la boucle dès que la condition est fausse.

```
Tantque Condition faire

    Instructions ;

FinTantque
```

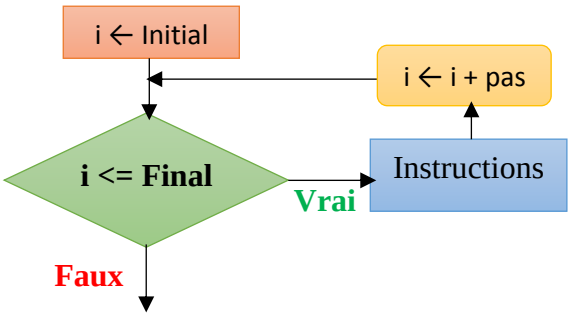


- ✓ Un tour de boucle est appelé: **Itération** (Répétition)
- ✓ Les instructions de la boucle doit changer la valeur de condition de vrai à faux (après un nombre d'itérations), sinon il y a une **Boucle infinie**.

### 2.2. La boucle *Pour*

On répète des instructions pour un compteur allant de la valeur ***Initiale*** à la valeur ***Finale***.

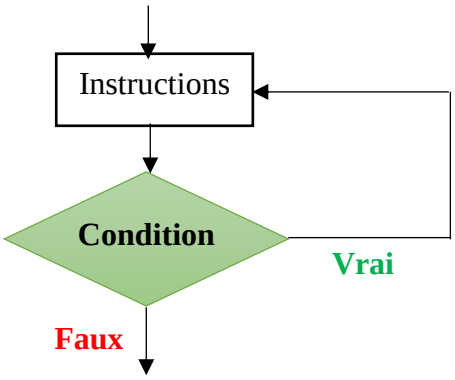
```
Pour i allant de Initial à Final pas n faire
    Instructions ;
FinPour
```




- ❖ **i** est un **compteur** allant de la valeur **Initiale** à la valeur **Finale**.
- ❖ **Pas** est un entier qui peut être positif ou négatif. **Pas** peut ne pas être mentionné, car par défaut sa valeur est égal à 1.
- ❖ La sortie de la boucle s'effectue lorsque le compteur atteint la valeur finale.

### 2.3. La boucle *Répéter*

```
Répéter
    Instructions ;
Tantque Condition
```




- ❖ La condition est testée à la fin de la boucle.
- ❖ Les instructions entre ***Faire*** et ***TantQue*** sont exécutées au moins une fois.
- ❖ La sortie de la boucle s'effectue lorsque la condition sera **Fausse.**

## Série d'exercices N°3

***Exercice 1***

a) Écrire un algorithme qui lit un nombre, et l'informe ensuite si ce nombre est pair ou impair.
b) Écrire un algorithme qui lit 2 nombres a et b, et l'informe ensuite si a est un multiple de b.
c) Écrire un algorithme qui lit un nombre entier et affiche ensuite s'il est composé de 3 chiffres.
d) Écrire un algorithme qui demande trois noms à l'utilisateur et l'informe ensuite s'ils sont rangés ou non dans l'ordre alphabétique.
e) Ecrire un algorithme qui lit un caractère et détermine ensuite s'il est un chiffre, une lettre de l'alphabet ou un autre type de caractères.
f) Ecrire un algorithme qui lit trois entiers qui représentent les longueurs des côtés d'un triangle, et permet de dire s'il s'agit d'un triangle équilatéral, isocèle ou scalène.

***Exercice 2***

Le passage d'un étudiant inscrit en première année (L1) à la deuxième année (L2) se fait en respectant les règles suivantes :

- si la moyenne des deux semestres S1 et S2 est supérieur ou égale à 10 alors l'étudiant est déclaré comme admis.
- sinon, si l'étudiant a validé un minimum de 30 crédits avec au moins 10 crédits dans un semestre et 20 crédits dans l'autre, alors l'étudiant est déclaré comme admis en dette.
- sinon, l'étudiant est déclaré comme ajourné.

Écrire un algorithme qui demande à l'utilisateur sa moyenne en S1 et en S2 et, dans le cas échéant, ses crédits en S1 et en S2 puis lui affiche s'il est admis, admis en dette ou bien ajourné

***Exercice 3***

Écrire un algorithme qui lit le nom, le prénom, les notes de quatre modules avec des coefficients (4, 4, 2, 1) d'un étudiant, et puis calcule sa moyenne et affiche son nom, son prénom, sa moyenne ainsi que sa mention sachant que :

| | ***moy***>=16 | 16>***moy***>=14 | 14>***moy***>=12 | 12>***moy***>=10 | ***moye***<10 |
|---|---|---|---|---|---|
| ***Mention*** | Très bien | Bien | Assez Bien | Passable | Ajourné |

***Exercice 4***

a) Ecrire un algorithme qui demande l'âge d'un enfant à l'utilisateur. Ensuite, il l'informe de sa catégorie :

- "Poussin" de 6 à 7 ans
- "Pupille" de 8 à 9 ans
- "Minime" de 10 à 11 ans
- "Cadet" après 12 ans

b) Écrire un algorithme qui lit 3 nombres entiers pour le jour, le mois et l'année d'une date et vérifie si c'est une date valide, puis l'affiche dans le format suivant : 15 septembre 2004. Supposant que chaque mois compte 30 jours.

***Exercice 5***

Écrire des algorithmes qui permettent de :

a) afficher les nombres impairs inférieurs à N par ordre croissant puis par ordre décroissant.
b) calculer et afficher le factoriel de N
c) calculer et afficher la puissance nème d'un nombre x : $x^n$=x*x*x*x*…. (n fois)
d) calculer et afficher la somme : 1+3+5+…+N.

***Exercice 6***

Écrire un algorithme qui permet à l'utilisateur d'entrer 100 nombres, puis calcule et affiche la valeur moyenne des nombres positifs.

***Exercice 7***

**1)** Écrire un algorithme qui calcul la valeur approchée de : $\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \cdots + \frac{1}{n}$

**2)** Modifier l'algorithme en s'arrêtant lorsque le terme 1/x est plus petit que **ε** (epsilon)

***Exercice 8***

a) Écrire un algorithme qui permet de lire deux nombres entiers strictement positifs A et B, puis calculer et afficher leur plus grand diviseur commun (PGCD).
b) Donner la représentation graphique (organigramme) de l'algorithme.

***Exercice 9***

Écrire les algorithmes qui permettent de :

1) Demander à l'utilisateur un nombre de départ, ensuite afficher les dix nombres suivants. Par exemple, si l'utilisateur entre le nombre 17, le programme affichera les nombres de 18 à 27.
2) Saisir une suite de caractères (se terminant par le caractère '.'), compter et afficher le nombre de lettres, chiffres et d'espaces.

***Exercice 10***

Écrire un algorithme qui affiche toutes les valeurs entières possibles de *x* et *y* dans la plage −20 à +20 qui valident la formule suivante : $3x^2 - 6y^2 = 6$

***Exercice 11***

a) Ecrire un algorithme qui permet d'afficher l'ensemble des nombres premiers compris entre 1 et 100.
b) Donner la représentation graphique (organigramme) de l'algorithme.

***Exercice 12***

Écrire un algorithme qui permet d'afficher les nombres parfaits inférieurs à 25000. Un nombre est dit parfait si la somme de ses diviseurs est égale à lui-même. (Exemple : 6 est un nombre parfait).

## Solutions

**Exercice 1**

**Algorithme** exo1_a ;

**Variable** A : entier ;

**Début**

Ecrire (" donnez un nombre : ") ;

Lire (A) ;

**Si** (A mod 2 = 0) alors

Ecrire ("le nombre est pair ") ;

**Sinon**

Ecrire ("le nombre est impair ") ;

**FinSi**

**Fin**

**Algorithme** exo1_b ;

**Variable** A, B : entier ;

**Début**

Ecrire ("donnez deux nombres : " ) ;

Lire (A,B) ;

**Si** (A mod B= 0) alors

Ecrire (A, " est un multiple de ", B) ;

**Sinon**

Ecrire (A, " n'est pas un multiple de ", B) ;

**FinSi**

**Fin**

**Algorithme** exo1_c ;

**Variable** A : entier ;

**Début**

Ecrire (" donnez un nombre : ") ;

```
   Lire (A) ;
   Si (A >=100)  et (A<1000) alors
      Ecrire (" le nombre est composé de trois chiffres ") ;
   Sinon
      Ecrire (" le nombre n'est pas composé de trois chiffres ") ;
   FinSi
Fin

Algorithme exo1_d ;
Variable A, B, C : chaine de caractères ;
Début
   Ecrire (" donnez les trois noms : ") ;
   Lire (A, B, C) ;
   Si ((A < B) et (B<C)) alors
      Ecrire (" les trois noms sont rangés par ordre alphabétique ");
   Sinon
      Ecrire (" les trois noms ne sont pas rangés par ordre alphabétique ");
   Finsi
Fin

Algorithme exo1_e ;
Variable C : caractères ;
Début
   Ecrire (" donnez un caractère: ") ;
   Lire (C) ;
   Si (C >='0' ) et (C<='9') alors
      Ecrire (" le caractère est un chiffre ");
   Sinon
      Si (C >='a' ) et (C<='Z') Alors
         Ecrire (" le caractère est une lettre de l'alphabet ");
```

```
    Sinon
        Ecrire (" Autre caractère ");
    Finsi
  Finsi
Fin
```

```
Algorithme exo1_f ;
Variable X, Y, Z : Réel ;
Début
  Ecrire (" donnez les longueurs des 3 côtés d'un triangle: ") ;
  Lire (X, Y, Z) ;
  Si (X = Y) et (Y = Z) alors
    Ecrire (" le triangle est équilatéral ");
  Sinon
    Si (X = Y ) ou (X = Z) ou (Y = Z) Alors
      Ecrire (" le triangle est isocèle ");
    Sinon
      Ecrire (" le triangle est scalène ");
    Finsi
  Finsi
Fin
```

**Exercice 2**

```
Algorithme exo2 ;
Variable MS1, MS2: Réel ;
         CS1, CS2 : entier ;
Début
  Ecrire (" donnez les moyennes des 2 semestres: ") ;
  Lire (MS1, MS2) ;
  Ecrire (" donnez les crédits des 2 semestres: ") ;
```

Lire (CS1, CS2) ;

**Si** ((MS1+MS2) / 2 >= 10) **alors**

Ecrire (" l'étudiant est admis sans dettes ");

**Sinon**

**Si** (CS1+ CS2 >= 30 ) et ((CS1 >= 10) et (CS2 >= 20))

ou ((CS1 >= 20) et (CS2 >= 10)) **Alors**

Ecrire (" l'étudiant est admis avec dettes ");

**Sinon**

Ecrire (" l'étudiant est ajourné ");

**Finsi**

**Finsi**

**Fin**

**<u>Exercice 3</u>**

**Algorithme** exo3 ;

**Constant** C1=4, C2=4, C3=2, C4=1 ;

**Variable** Nom, Pren, Mention : chaine de caractère ;

Note1, Note2, Note3, Note4, Moy : réel ;

**Début**

Ecrire ("donnez le nom et le prénom " ) ;

Lire (Nom, Pren) ;

Ecrire ("donnez les notes des quatre modules ") ;

Lire (Note1, Note2, Note3, Note4) ;

Moy← (Note1*C1 + Note2*C2 + Note3*C3 + Note4*C4) / (C1+C2+C3+C4) ;

Ecrire (" La moyenne de ", Nom, " ", Pren , " est : ", Moy) ;

**Si** (Moy>= 16) **alors**

Mention ← "Très bien" ;

**Sinon**

**Si** (Moy>= 14) **alors**

Mention ← "bien" ;

```
        Sinon
            Si  (Moy>= 12) alors
                Mention ← ”assez bien” ;
            Sinon
                Si (Moy>= 10) alors
                    Mention ← ”passable” ;
                Sinon
                    Mention ← ”ajourné” ;
                Finsi
            Finsi
        Finsi
    Finsi
    Ecrire (”la mention est : ”, Mention) ;
Fin
```

**Exercice 4**

```
Algorithme exo4_a;
Variable Age : entier ;
Début
    Ecrire (”donnez l’age : ”) ;
    Lire (A) ;
    Selon A faire
        6 .. 7 : Ecrire (”poussin”) ;
        8 .. 9 : Ecrire (”Pupille”) ;
       10 .. 11 : Ecrire (”Minime”) ;
    Sinon
      Ecrire (”Cadet”) ;
    Finselon
Fin
```

```
Algorithme exo4_b;
Variable J, M, A : entier ;
```

**Début**

Ecrire (”donnez le jour, le mois et l’année : ”) ;

Lire (J, M, A) ;

**Si** (J >= 1) et (J <= 30) et (M >= 1) et (M <= 12) et (A >= 1) **Alors**

**Selon** M **faire**

1 : Ecrire (J, ” Janvier ”, A) ;

2 : Ecrire (J, ” Février ”, A) ;

3 : Ecrire (J, ” Mars ”, A) ;

4 : Ecrire (J, ” Avril ”, A) ;

5 : Ecrire (J, ” May ”, A) ;

6 : Ecrire (J, ” Juin ”, A) ;

7 : Ecrire (J, ” Juillet ”, A) ;

8 : Ecrire (J, ” Aout ”, A) ;

9 : Ecrire (J, ” Septembre ”, A) ;

10 : Ecrire (J, ” Octobre ”, A) ;

11 : Ecrire (J, ” Novembre ”, A) ;

12 : Ecrire (J, ” Décembre ”, A) ;

**Sinon**

Ecrire (‘erreur’) ;

**Finselon**

**Sinon**

Ecrire (”La date est invalide ”) ;

**Finsi**

**Fin**

## Exercice 5

### a- Avec la boucle Tantque

```
Algorithme Impaire_a ;
Var i , N:entier ;
Début
    Ecrire(”Donner N”) ;
    Lire(N) ;
    i ← 0 ;
    TantQue i<=N Faire
      Si  i mod 2 ≠ 0 alors
          Ecrire(i) ;
      Finsi
      i ←i+1 ;
    FinTanque
Fin
```

```
Algorithme Impaire_b ;
Var i , N: entier ;
Début
    Ecrire(”Donner N”) ;
    Lire(N) ;
    i ← N ;
    TantQue i>=0 Faire
      Si  i mod 2 ≠ 0  alors
          Ecrire(i) ;
      Finsi
      i ←i-1 ;
    FinTanque
Fin
```

```
Algorithme Factoriel ;
Var i , N, F :entier ;
Début
    Ecrire(”Donner N”) ;
    Lire(N) ;
    i ← 1 ;
    F ←1 ;
    TantQue i<=N Faire
      F ←F*i ;
      i ←i+1 ;
    FinTanque
    Ecrire (N,’ != ‘, F) ;
Fin
```

```
Algorithme Puissance ;
Var i , N, P, X :entier ;
Début
    Ecrire(”Donner N”) ;
    Lire(N) ; Lire(X) ;
    i ← 1 ;
    P ←1 ;
    TantQue i<=N Faire
      P ←P*X ;
      i ←i+1 ;
    FinTanque
    Ecrire (P) ;
Fin
```

```
Algorithme Somme ;
Var i , N, S :entier ;
Début
    Ecrire(”Donner N”) ;
    Lire(N) ;
    i ← 1 ;
    S ←0 ;
    TQ i<=N Faire
      S ←S+i ;
      i ←i+2 ;
    FinTQ
    Ecrire (S) ;
Fin
```

## b- Avec la boucle Pour

```
Algorithme Factoriel ;
Var i , N, F :entier ;
Début
    Ecrire(”Donner N”) ;
    Lire(N) ;
    F ←1 ;
    Pour i allant de 1 à N Faire
        F ←F*i ;
    FinPour
    Ecrire (N,’! = ‘, F) ;
Fin
```

```
Algorithme Puissance ;
Var i , N, P, X :entier ;
Début
    Ecrire(”Donner N”) ;
    Lire(N) ; Lire(X) ;
    P ←1 ;
    Pour i allant de 1 à N Faire
        P ←P*X ;
    FinPour
    Ecrire (P) ;
Fin
```

```
Algorithme Somme ;
Var i , N, S :entier ;
Début
    Ecrire(”Donner N”) ;
    Lire(N) ;
    S ←0 ;
    Pour i allant de 1 à N Pas 2 Faire
        S ←S+i ;
    FinPour
    Ecrire (S) ;
Fin
```

## Exercice 6

```
Algorithme  Ex02;
Var N, S , Nb, M: entier ;
Début
    S ← 0 ; Nb ← 0 ;
    Pour i ← 1 à 100 Faire
        Lire (N) ;
        Si  N>0 alors
            S ← S + N ;
            Nb← Nb +1;
        Finsi
    FinPour
    Si Nb≠0 Alors
        M← S/Nb ;
        Ecrire (M)
    FinSI
Fin
```

## Exercice 7 : Le choix entre TantQue et pour

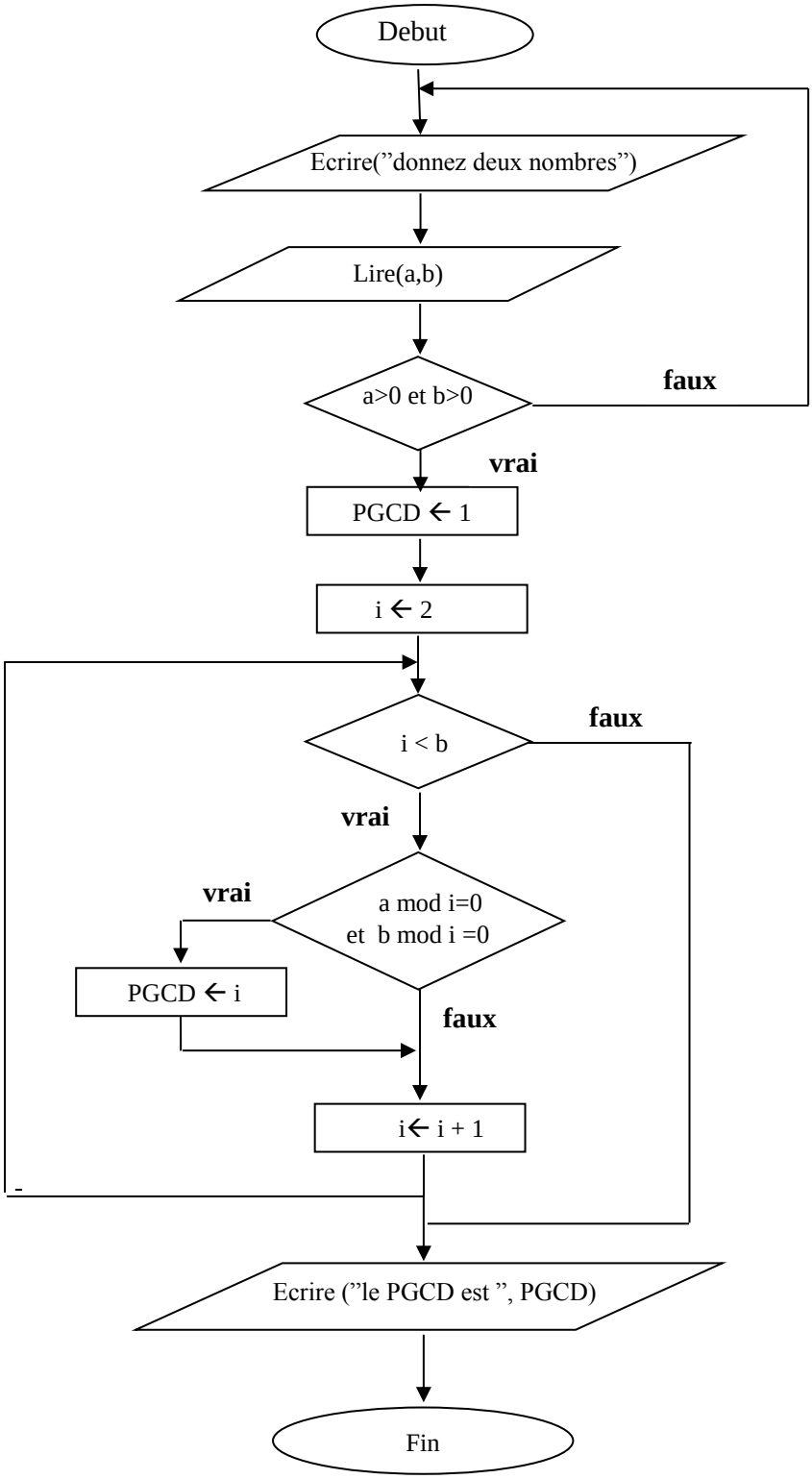
```
Algorithme PI_a
Var S, i, n : entier ;
        Pi: Réel ;
Debut
  Pi← 0 ;
  S ← 1 ;
  Lire (n) ;
Pour i allant de 1 à n Pas 2 Faire
     Pi ← Pi +S*1/i ;
     S ← -S ;
 FinPour
    Ecrire(″ π = ″, Pi*4) ;
Fin
```

```
Algorithme PI_b
Var S, i : entier ;
E, Pi: Réel ;
Debut
  Pi← 0;  S ← 1 ;
  i ←  1; Lire (E) ;
Tantque 1/i >= E Faire
  Pi ← Pi +S*1/i ;
```

## Exercice 8 : Pour et Répéter

```
Algorithme exo4 ;
Variables  a, b, i, PGCD : entier ;
Début
   Faire
        Ecrire (”donnez deux nombres”) ;
        Lire (a, b) ;
   TantQue ((a<=0) ou (b<=0))
   PGCD ← 1 ;
   Pour  i  allant de 2 à b  faire
      si ((a mod i = 0) et (b mod i = 0)) alors
         PGCD ← i ;
      finsi
   FinPour
   Ecrire (”le PGCD est ”, PGCD) ;
Fin
```

**Remarque : Il y a des langages qui utilisent la syntaxe : Répéter … Jusqu'à**



## Exercice 9 : La boucle répéter

```
Algorithme EXO9_1 ;
Var i , N :entier ;
Début
    Ecrire(”Donner N”) ;
    Lire(N) ;
    i ← 1 ;
    Faire
        Ecrire(N+i) ;
        i ←i+1 ;
    TantQue (i ≤ 10)
Fin
```

```
Algorithme EXO9_2 ;
Var c : Caractère ;
      NbLet, NbChi, NbEsp : entier ;
Début
    Nblet←0 ; NbChi ←0 ; NbEsp←0 ;
    Faire
        Lire(c) ;
        Selon c faire
            ‘a’..’z’ : Nblet ← Nblet+1 ;
            ‘A’..’Z’ : Nblet ← Nblet+1 ;
            ‘0’..’9’ : NbChi ← NbChi+1 ;
                ‘ ‘ : NbEsp ← NbEsp+1 ;
        FinSelon
    TantQue (c ≠ ’.’)
    Ecrire (Nblet, NbChi, NbEsp) ;
Fin
```

## Exercice 10 : Les boucles imbriquées

```
Algorithme EX10 ;
Var x, y : entier ;
Début
  Pour x ← -20 à 20 faire
      Pour y ← -20 à 20 faire
          Si (3*x*x - 6*y*y =6) alors
              Ecrire (x, y) ;
          FinSi
      FinPour
  FinPour
Fin
```

## Exercice 11 et 12 : Les boucles imbriquées

```
Algorithme Premier ;
Variables i, j, d : entier ;
Début
    Pour i allant de 1 à 100 faire
       d← 0 ; //Nbre de diviseurs
       j← 1 ;
        TantQue j<=i Faire

                Si ( i mod j = 0) alors
                d←d + 1 ;
                 Finsi
               j←j + 1 ;
        FinTanque

       Si (d = 2) alors
          Ecrire (i) ;
      Finsi
   FinPour
Fin
```

```
Algorithme Parfait ;
Variables  i, j, d : entier ;
Début
  Pour i allant de 1 à 25000  faire
     d← 0 ;   //Somme des diviseurs
     Pour j allant de 1 à (i-1) faire
         Si ( i mod j = 0) alors
            d← d + j ;
         Finsi
     FinPour
     Si (d = i) alors
        Ecrire (i) ;
     Finsi
  FinPour
Fin
```

# Chapitre 4 : Tableaux et Enregistrements

## 1- Introduction

Dans les chapitres précédents, nous avons vu les variables simples, alors que dans ce présent chapitre nous allons étudier les variables structurées qui permettent de représenter plusieurs données en MC, ce type de variables est appelé : structure de données. On distingue plusieurs types de structures de données, parmi lesquels : Tableaux et enregistrements.

Supposons qu'on veut déclarer les informations de 100 étudiants, chaque étudiant est définie par : N°Inscription, Nom, Prénom, Date de naissance et Moyenne. Nous avons donc besoin de 100 variables pour déclarer les N°Inscriptions, ainsi que pour déclarer les autres variables, cela deviendrait fastidieux.

Pour résoudre ce problème, nous utilisons un tableau qui permet de regrouper plusieurs données de même type, et un enregistrement qui regroupe plusieurs données de type différents.

## 2. Tableaux

Un **tableau** est une structure de donnée qui permet de stocker un certain nombre d'éléments repérés par des indices. Il existe des tableaux à une dimension et des tableaux à plusieurs dimensions.

### 2.1. Tableau à une dimension (*vecteur*)

Un tableau à une dimension est une suite d'éléments portant le même nom de variable et le même type, chaque élément est repéré par un nombre entier appelé **indice.** Pour désigner un élément du tableau, on fait figurer le **nom** du tableau, suivi de l'indice de l'élément entre deux crochets.

**Exemple :** Voici un tableau avec un nom **Notes** composé de 10 éléments de type réel.

| **indice** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
|---|---|---|---|---|---|---|---|---|---|---|
| **Notes** | 5 | 4.5 | 12 | 14.5 | 8 | 10 | 9 | 17 | 11.5 | 7 |

Le nom du tableau : **Notes**

La taille du tableau : **10**

**Notes [ 5 ] = 8** (le cinquième élément a une valeur de 8)

#### 2.1.1. Déclaration des tableaux à une dimension

La déclaration d'un tableau s'effectue en précisant le **nom** (identificateur), la **taille** (nombre d'éléments) et le **type** du tableau.

| **Dans un algorithme** | **En C++** |
|---|---|
| ***Variables*** nom [taille] : type | Type nom[taille]; |

**Exemple**

**En algorithmique : Variable** Notes [100] : réel ; **En C++ : float** Notes [10];

### 2.2. Tableaux à deux dimensions (Matrices)

Un tableau à deux dimensions est constitué de lignes et de colonnes, où ses éléments sont repérés par **2 indices.**

**Exemple :**

**MATRICE**

| | 1 | 2 | 3 |
|---|---|---|---|
| 1 | 5 | 1 | 9 |
| 2 | 1 | 8 | 0 |
| 3 | 9 | 0 | 7 |
| 4 | 4 | 3 | 8 |

Le nom : MATRICE

La taille : 4*3

(4 lignes et 3 colonnes)

**MATRICE[4,2] = 3**

4 est la valeur de l'élément situé à l'intersection de la ligne 4 et la colonne 2

#### 2.2.2. Déclaration d'une matrice

| **Dans un algorithme** | **En C++** |
|---|---|
| ***Variables*** nom [N, M] : **type** | **Type** nom[N, M]; |

**Exemple**

**En algorithmique : Variable** MAT [10, 20] : **entier ;**

**En C++ :** **int** MAT [10, 20] ;

## Série d'exercices N°4

### Exercice 1

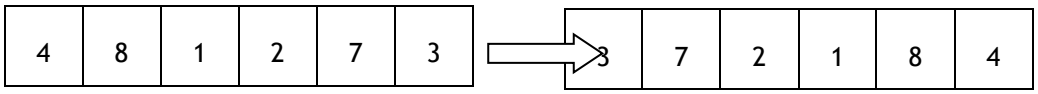
Écrire un algorithme qui permet de:

1. déclarer un tableau de 30 notes, dont on fait ensuite saisir les valeurs par l'utilisateur (la note doit être entre 0 et 20).
2. Calculer et afficher la moyenne
3. Comparer chaque note à la moyenne et afficher "égal", "inférieur" ou "supérieur à la moyenne"
4. Compter et afficher combien il y a de notes supérieures à la moyenne
5. Chercher la note la plus grande. Afficher cette note et sa position dans le tableau
6. Ajouter 1 à toutes les notes qui sont inférieur à 10.

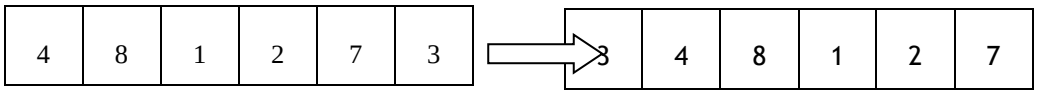
### Exercice 2

Écrire un algorithme constituant un tableau de N éléments,

1) inverser les valeurs de ce tableau et afficher le résultat à l'écran en utilisant 2 méthodes (avec 1 seul tableau ; avec 2 tableaux)




2) Faire une rotation du tableau.




**Exercice 3**

Écrire un algorithme qui permet de :

1. saisir les valeurs d'une matrice de 4 lignes et 5 colonnes
2. afficher les valeurs de cette matrice, la moyenne des éléments de chaque ligne, de chaque colonne et la moyenne globale.

**Exercice 4**

Écrire un algorithme qui permet de :

**1)** lire une matrice carrée *A(n,n)* , tel que *n* est une constante.
**2)** calculer le nombre d'éléments positifs au-dessus de la première diagonale
**3)** inverser la matrice par rapport à la première diagonale.

***Exemple***

| | | |
|---|---|---|
| **35** | 10 | 24 |
| 22 | **11** | 25 |
| 12 | 9 | **55** |

devient

| | | |
|---|---|---|
| **35** | 22 | 12 |
| 10 | **11** | 9 |
| 24 | 25 | **55** |

**4)** calculer le nombre des éléments négatifs au-dessous de la deuxième diagonale.
**5)** inverser la matrice par rapport à la deuxième diagonale

***Exemple***

| | | |
|---|---|---|
| 35 | 10 | **24** |
| 22 | **11** | 25 |
| **12** | 9 | 55 |

devient

| | | |
|---|---|---|
| 55 | 25 | **24** |
| 9 | **11** | 10 |
| **12** | 22 | 35 |

**Exercice 5**

Ecrire un algorithme qui lit une ligne de texte (ne dépassant pas 150 caractères) la mémorise dans une variable TXT et l'affiche ensuite:

1) Longueur L de la chaîne.
2) Nombre de voyelle
3) Nombre de lettres minuscules, majuscules et les autres caractères
4) Nombre de mots (les mots sont séparés par des espaces).
5) Remplacer la lettre 'A' par 'a'

**Exercice 6**

Supposons que 2 tableaux de taille 100 contiennent les informations de connexion (les noms d'utilisateur et les mots de passe) de 100 employés d'une entreprise.

Écrire un algorithme qui permet de :

1) saisir les informations de tous les employés.
2) inviter l'utilisateur à entrer un nom d'utilisateur et son mot de passe, puis affiche le message "Connexion OK !" lorsque la combinaison du nom d'utilisateur et du mot de passe est valide ; le message « Échec de la connexion ! » doit être affiché dans le cas contraire.

Supposons que les noms d'utilisateur sont uniques mais que les mots de passe ne le sont pas. De plus, les noms d'utilisateur et les mots de passe ne sont pas sensibles à la casse.

| | 1 | 2 | 3 | | | | 99 | 100 |
|---|---|---|---|---|---|---|---|---|
| Username | U1 | U2 | U3 | ……. | ……. | ……. | U99 | U100 |
| Password | P1 | P2 | P3 | ……. | ……. | ……. | P99 | P100 |

3) En utilisant le tri par sélection, trier les noms des utilisateurs par ordre croissant
4) En utilisant la méthode de recherche dichotomique, chercher un nom d'utilisateur lu au clavier et afficher son mot de passe.

**Exercice 7**

Écrire le programme C++ qui permet de déterminer la plus grande valeur, la plus petite valeur et la moyenne dans le tableau ci-dessous.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 5 | 4 | 15 | 7 | 9 | 1 | 18 | 2 | 8 | 9 |

Max = 18 ; Min = 1 ; Moy = 7.8

**Exercice 8**

Écrire un programme C++, qui permet de lire deux vecteurs (tableaux à une dimension) de même taille et de calculer leur somme dans un nouveau vecteur, et leur produit scalaire, puis afficher le vecteur résultat et le produit scalaire. **Exemple :**

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 4 | 6 | 3 | 8 | **+** | 1 | 5 | 2 | -1 | **=** | 5 | 11 | 5 | 7 |

Le produit scalaire = 4*1+6*5+3*2+8*(-1) = **32**.

## Exercice 9

Soit un vecteur T contenant N nombres entiers. Écrire les programmes permettant de :

1) mettre les valeurs négatives au début et les valeurs positives à la fin du tableau.

2) Même question en utilisant deux tableaux.

## Exercice 10

Écrire un programme C++ qui permet de déclarer et d'initialiser la matrice carrée 4*4 suivante :

| | | | |
|---|---|---|---|
| 2 | 1 | 4 | 9 |
| 5 | 6 | 3 | 1 |
| 7 | 4 | 1 | 0 |
| 8 | 2 | 9 | 3 |

La somme = **65**

La moyenne = **4,06**

Calculer la somme et la moyenne des éléments de la matrice et le nombre d'occurrence de la valeur **1**

## Exercice 11

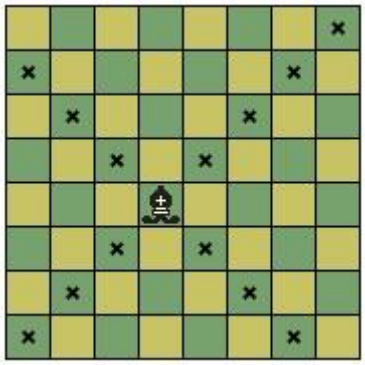
Écrire un programme qui permet de construire et d'afficher

une matrice carrée A(NxN) qui représente le triangle de pascal:

| | | | | |
|---|---|---|---|---|
| **1** | **0** | **0** | **0** | **0** |
| **1** | **1** | **0** | **0** | **0** |
| **1** | **2** | **1** | **0** | **0** |
| **1** | **3** | **3** | **1** | **0** |
| **1** | **4** | **6** | **4** | **1** |

## Exercice 12

Ecrire un programme C++ qui lit une position (i,j) d'une matrice de taille 8×8, rempli par des zéros. Et marque toutes les cases des deux diagonales qui passent par cette position en y mettant des uns.



## Exercice 13

Écrivez un programme qui détermine si un mot est un palindrome (Un palindrome est un mot dont la succession des lettres est la même quand on le lit de gauche à droite ou de droite à gauche, exemple : « LAVAL », « RADAR »)

**Exercice 14**

Un des plus anciens systèmes de cryptographie consiste à décaler les lettres d'un message pour le rendre illisible. Ainsi, les *a* deviennent des *b*, les *b* des *c*, …, les *z* des *a*.

Écrivez un programme qui :

1. demande une phrase à l'utilisateur et qui la crypte selon ce principe.

   Exemple : Phrase : « bonjour les amis » ----------> Phrase cryptée : « cpokpvs mft bnjt »

2. fait le contraire pour décrypter la phrase précédente.

   Exemple : Phrase cryptée : « cpokpvs mft bnjt » ----------> Phrase décryptée: « bonjour les amis »

**Exercice 15**

Écrivez un programme C++ qui demande à l'utilisateur d'entrer le nom et l'âge de 50 personnes. En utilisant l'algorithme de tri à Bulle, afficher ces informations,

1) triées par âge, par ordre croissant.
2) triées par Nom, par ordre croissant.

**Exercice16**

***Etudiant*** est une structure composée de trois champs : Nom, Prénom et Adresse. L'adresse est une autre structure composée de trois champs : Rue, Ville et le Code postal.
Écrire un algorithme qui permet de saisir et d'afficher le nom, prénom et la ville d'un étudiant

**Exercice 17**

On désire gérer les colis d'une entreprise de transport. Pour cela, chaque colis est décrit par son poids, sa date d'envoi, sa destination, son état (envoyé ou non envoyé). On regroupe tous les colis dans un tableau de N éléments. On veut remplir le tableau des colis puis effectuer un ensemble de traitements à ce tableau.

**Questions**

1. Spécifier les types nécessaires pour l'écriture de l'algorithme de gestion des colis.
2. Remplir le tableau.
3. Calculer et afficher le poids total des colis envoyés.
4. Afficher les colis non envoyés.
5. Afficher le colis non envoyé le plus lourd.
6. Lire une date donnée puis chercher et afficher l'indice du colis ayant cette date d'envoi.
7. Modifier la destination des colis (l'ancienne et la nouvelle destination sont lues au clavier)

**Exercice 18**

La plateforme d'enseignement à distance Moodle est un moyen de former des étudiants à distance, c'est-à-dire sans avoir besoin de se rendre dans une université pour assister à des cours. Pour cela, chaque étudiant est défini par un Nom d'utilisateur (Username), un nom et prénom, un mot de passe, une date de naissance et un groupe. Ecrire un algorithme qui permet de :

1) Définir une structure ***étudiant***

2) Lire ensuite une liste d'étudiants

3) Modifier le mot de passe d'un étudiant donné par l'utilisateur

4) Ajouter un nouvel étudiant à la fin de la liste.

5) Insérer un nouvel étudiant dans une position entrée par l'utilisateur.

6) Supprimer un étudiant dans une position entrée par l'utilisateur.

**Exercice 19**

Soit la structure **personne** composée de champs Nom, Age et date d'embauche, tel que la date d'embauche est une autre structure composée de : Jour, Mois et Année.

Écrire un programme qui permet de remplir et d'afficher les différents champs avec un dialogue se présentant sous la forme suivante:

Nom : Ahmed

Age : 40

Date d'embauche (jj mm aaaa) : 10 12 2005

**Exercice 20**

Soit une structure représentant un point d'un plan définie par un nom et les coordonnées x et y

1. Définir le Type Point et déclarer un tableau (nommé courbe) de N points (N est une constante).
2. Lire les valeurs des différents « points » du tableau courbe.
3. Effectuer un déplacement de la courbe défini par ses deux arguments dx et dy .
4. Affiche les valeurs des différents « points » du tableau courbe, sous la forme : point D de coordonnées 10 2

**Exercice 21**

Un étudiant est représenté par un N°Inscription, un nom et prénom, un numéro de groupe, et les notes de 5 modules.

1. Déclarer un tableau de N étudiants (nommé classe), N est une constante.
2. Remplir la fiche de tous les étudiants de la classe.

3. Rechercher l'étudiant ayant un numéro lu au clavier et calculer sa moyenne
4. Ajouter un nouvel étudiant à la fin du tableau
5. Insérer un nouvel étudiant dans une position entrée par l'utilisateur.
6. Supprimer un étudiant dans une position entrée par l'utilisateur.
7. Afficher la nouvelle liste des étudiants

## Solutions

### Exercice 01

```
Algorithme Ex01_01 ;
Var  Notes[30] , i : entier ;
Début
   Pour (i allant de 1 à 30) faire
      Lire (Notes [i])
   FinPour
Fin.
```

```
Var  S, M : réel ;
 S ← 0;
 Pour (i allant de 1 à 30) faire
     S ← S + Notes [i]);
 FinPour
 M  ← S / 30
 Ecrire (M)
```

```
   Pour (i allant de 1 à 30) faire
      Si Notes [i]>M Alors
         Ecrire("Supérieur")
      Sinon
         Si Notes [i]>M Alors
            Ecrire("Inférieur")
         Sinon
            Ecrire("égale")
         FinSi
      FinSi
   FinPour
```

```
   Var nb : entier ;
     Nb ← 0
   Pour (i allant de 1 à 30) faire
      Si Notes [i]>M Alors
         Nb ← Nb + 1
      FinSi
   FinPour
   Ecrire(Nb)
```

```
Var  Max : réel ;  Pos : entier ;
   Max ← Notes [1] ; Pos ← 1 ;
   Pour (i allant de 2 à 30) faire
      Si (Notes [i] < Max) Alors
         Max ← Notes [i] ;
         Pos ← i ;
      FinSi;
   FinPour
   Ecrire(Max, Pos) ;
```

```
   Pour (i allant de 1 à 30) faire
      Notes [i] ← Notes [i] + 1 ;
   FinPour
```

Remarque : l'indice du tableau peut commencer par 0 (C++, VBA, ..) ou par 1 (pascal, .. )

## Exercice 02

```
Algorithme  Inverse1 ;
Const  N = 10 ;
Var  i , A, Tab[N] : entier ;
Début
   Pour (i allant de 1 à N) faire
      Lire( Tab[i] ) ;
   FinPour
   Pour (i allant de 1 à N div 2) faire
      A ← Tab[i] ;
      Tab[i]  ← Tab[N-i+1] ;
      Tab[N-i+1]  ← A ;
   FinPour
   Pour (i allant de 1 à N) faire
      ecrire( Tab[i] ) ;
   FinPour
Fin.
```

```
Algorithme  Inverse2 ;
Const  N = 10 ;
Var  i , Tab1[N] , Tab2[N]: entier ;
Début
   Pour (i allant de 1 à N) faire
      Lire( Tab1[i] ) ;
   FinPour
   Pour (i allant de 1 à N) faire
      Tab2[i] ← Tab1[N-i+1];
   FinPour
   Pour (i allant de 1 à N) faire
      ecrire( Tab2[i] ) ;
   FinPour

Fin.
```

## Exercice 03

```
Algorithme  Matrice ;
Const  N = 4 ; M=5 ;
Var  i , j, SL, SC, S, Moy, Mat[N, M] : entier ;
Début
   Pour (i allant de 1 à N) faire
      Pour (j allant de 1 à M) faire
         Lire(Mat[i, j] ) ;
      FinPour
   FinPour

   Pour (i allant de 1 à N) faire
      Pour (j allant de 1 à M) faire
         Ecrire(Mat[i, j] ) ;
      FinPour
   FinPour

  Pour (i allant de 1 à N) faire
     SL ←0 ;
     Pour (j allant de 1 à M) faire
        SL ← SL+Mat[i, j] ;
     FinPour
     Ecrire(SL/M) ;
   FinPour
```

```
  Pour (j allant de 1 à M) faire
     SC ←0 ;
     Pour (i allant de 1 à N) faire
        SC ← SC+Mat[i, j] ;
     FinPour
     Ecrire(SC/N) ;
   FinPour

   S ←0 ;
   Pour (i allant de 1 à N) faire
      Pour (j allant de 1 à M) faire
         S ← S+Mat[i, j] ;
      FinPour
   FinPour
   Moy ← S/(N*M) ;
   Ecrire(Moy) ;

Fin.
```

## Exercice 04

```
Algorithme  EX04 ;
Const N = 10 ;
Var  i , j, , Nb_Pos, Nb_Neg,  T
     Mat[N, N] : entier ;
Début
   Pour (i allant de 1 à N) faire
      Pour (j allant de 1 à N) faire
          Lire(Mat[i, j] ) ;
      FinPour
   FinPour
   Nb_Pos ←0 ;
   Pour (i allant de 1 à N-1) faire
      Pour (j allant de i+1 à N) faire
         Si  (Mat[i, j] > 0) Alors
             Nb_ Pos ← Nb_ Pos + 1 ;
         FinSin
      FinPour
   FinPour
  Ecrire(Nb_Pos) ;
```

```
   Nb_Neg ←0 ;
   Pour (i allant de 2 à N) faire
      Pour (j allant de N-i+2 à N) faire
         Si  (Mat[i, j] < 0) Alors
             Nb_Neg ← Nb_Neg + 1 ;
         FinSin
      FinPour
   FinPour
   Ecrire(Nb_Neg) ;
   Pour i allant de 1 à N-1 faire
     Pour j allant de 1 à N-i faire
       T ← Mat [i, j] ;
       Mat [i, j]← Mat [N-j+1, N-i+1];
       Mat [N-j+1, N-i+1]← T ;
      FinPour
   FinPour
Fin.
```

```
Pour (i allant de 1 à N-1) faire
   Pour (j allant de i+1 à N) faire
        T ← Mat [i, j] ;
        Mat [i, j]← Mat [j, i];
        Mat [j, i]← T ;
       FinSin
   FinPour
FinPour
```

**Exercice 05**

```
Algorithme  EX05 ;
Const  N = 150;
Var TXT[N] :Caractère ;
    i, Nba, Nbv, L, NbMaj, NbMin,
    NbAutres, NbMot: entier ;
Debut
   Ecrire("Donner une chaine") ;  Lire(TXT) ;
   L ← longueur(TXT) ;

   Nbv←0 ;
   Pour i allant de 1 à L  faire
      Selon  TXT [i]   faire
         'a','e','i','o','u','y' : Nbv ← Nbv+1 ;
      FinSelon
  FinPour
  Ecrire(Nbv) ;

 NbMaj ← 0 ; NbMin ← 0 ; NbAutres ← 0 ;
 Pour i allant de 1 à L  faire
      Selon  TXT [i]   faire
         'A'..'Z' : NbMaj ← NbMaj + 1 ;
         'a'..'z' : NbMin ← NbMin + 1 ;
      Sinon
          NbAutres ← NbAutres +1 ;
      FinSelon
  FinPour
  Ecrire(NbMaj, NbMin, NbAutres)
```

```
NbMot ← 0 ; i ← 1 ;
Tantque  i ≤ L  Faire
   Si TXT [i]=' '  Alors
      NbMot ←NbMot +1 ;
   Finsi ;
   i←i+1 ;
Fintque;
 Ecrire(" Nombre de mots : ", NbMot) ;

   Pour i allant de 1 à L  faire
      Si TXT [i]='A'  Alors
         TXT [i]='a' ;
      Finsi
   FinPour
```

## Exercice 06

```
Algorithme  EX06;
Const  N = 100;
Var Username[N], Password[N], A :Chaine ;
    i, j, m, prem, dern, milieu, indice: entier ;
   Connexion, trouve : Booléen ;
Debut
    Pour i allant de 1 à N  faire
       Lire(Username [i], Password[i])
   FinPour
   Lire(User, Pass)
   Connexion ← Faux ;
  TantQue (i <= N)  et (Connexion = Faux) faire
       Si (Username [i] = User) et (Password[i]) = Pass) Alors
          Connexion ← Vrai ;
       Sinon
           i ← i + 1;
       Finsi
   FinPour
   Si connexion = Vrai Alors
      Ecrire(" Connexion Ok") ;
   Sinon
      Ecrire(" Échec de la connexion ")
   Finsi
  Pour i allant de 1 à N-1 faire
        m← i ;
        Pour j allant de i+1 à N faire
          Si Username [j]< Username [m] alors
              m← j ;
          Finsi
        Finpour
        A← Username [i] ;
        Username [i]← Username [m] ;
        Username [m]← A ;
        A← Password [i] ;
        Password [i]← Password [m] ;
        Password [m]← A ;
   Finpour
```

```
prem←1;  dern←N; Lire (User);
milieu← N div 2 ; trouve←faux;
TQ (non trouve) et (prem<=dern) faire
    si  Username [milieu]= User alors
        trouve← vrai; indice←milieu;
    sinon
        si (Username [milieu]> User) alors
            dern← milieu-1;
        sinon
            prem←milieu+1;
        finsi
        milieu← (prem+dern) div 2;
    finsi
finTQ
Si  trouve  alors
     Ecrire(Password [indice])
Sinon
    Ecrire("Nom d'utilisateur invalide " )
finsi
```

## Exercice 7

```
#include <iostream>
using namespace std;
int main()
{
      int i, Max, Min, S, Tab[10]={5, 4, 15, 7, 9, 1, 18, 2, 8, 9};
      Max=Tab[0]; Min=Tab[0]; S=Tab[0];
      for (i=1; i<10; i++)
      {
          S=S+Tab[i];
             if (Tab[i]>Max)
            Max=Tab[i];
          if (Tab[i]<Min)
            Min=Tab[i];
      }
      cout<<"Max = "<<Max<<"\n";
      cout<<"Min = "<<Min<<"\n";
      cout<<"Moy = "<<S/10.0<<"\n";

      system("pause");
      return 0;
}
```

## Exercice 8

```
#include <iostream>
using namespace std;
const int N = 5;  //ou bien: #define N 5
int main ()
{
      int vect1[N], vect2[N], vect3[N];
      //Lecture des deux vecteurs
      cout<<"Vecteur 1:\n";
      for (int i=0; i<N; i++)   {
            cout<<"Entrez l'element "<<i<<" : ";
            cin>>vect1[i];      }
      cout<<"Vecteur 2:\n";
      for (int i=0; i<N; i++)
      {
            cout<<"Entrez l'element "<<i<<" : ";
            cin>>vect2[i];      }
    //Somme des deux vecteurs
    for (int i=0; i<N; i++)
        vect3[i]=vect1[i]+ vect2[i];
    //Produit scalaire
    int P=0;
      for (int i=0; i<N; i++)
        P=P+vect1[i]* vect2[i];
    //Affichage de résultats
    cout<<"Vecteur 3:\n";
    for (int i=0; i<N; i++)
            cout<<vect3[i]<<"  ";
    cout<<"\nLe produit scalaire = "<<P;
}
```

## Exercice 9

```
#include <iostream>
using namespace std;
int main()
{
        const int Max=100;
        int i,j,N,X, T[Max] ;
    //Lecture de la taille du tableau
    do
    {
          cout<<"Entrez la taille: " ;
          cin>>N;
    }
      while ((N<=0)||(N>100)) ;
   //Lecture des elements de T
    for (i=0; i<N; i++)
          cin>>T[i] ;
    //passer les valeurs negatives au
debut
    j=0 ;
    while ((j<N) && (T[j]<0))
           j++ ;
//deplacer les valeurs negatives au
debut //(trouver la premiere valeur
positive I )
    i=j ;
    while(j<N)
    {
          if (T[j]<0)
       {
           X=T[i] ; T[i]=T[j];
           T[j]=X; i++;
        }

       j++;
    }
//Affichage du nouveau tableau T
    for(i=0; i<N; i++) cout<<T[i]<<" " ;
}
```

```
#include <iostream>
using namespace std;
int main()
{
       const int Max=100;
       int i,j,N,k, T[Max], V[Max] ;

    //Lecture de la taille exacte du
tableau
    do
    {
      cout<<"Donner la taille du
      tableau (N<100): " ;
      cin>>N;    }
      while ((N<=0)||(N>100)) ;
   //Lecture des elements de T
    for (i=0; i<N; i++)
          cin>>T[i] ;
    //passer les valeurs negatives au
debut et les valeurs positif à la fin
    j=0 ; k=N-1;
      for (i=0; i<N; i++)
        if (T[i]<0) {
            V[j]=T[i];
            j++;
        }
        else
        {
            V[k]=T[i];
            k--;
        }
//Affichage du nouveau tableau T
    for(i=0; i<N; i++) cout<<V[i]<<" "
;

    return 0;
}
```

## Exercice 10

```
#include <iostream>
using namespace std;
int main()
{
      int i, j, nb,  Mat[4][4]={{2, 1, 4, 9}, {5, 6, 3, 1},
                                {7, 4, 1, 0}, {8, 2, 9, 3}};
      float M, S=0;
      for (i=0; i<4; i++)
        for (j=0; j<4; j++)
           S=S+Mat[i][j];
      M=S/16;
      nb=0;
      for (i=0; i<4; i++)
        for (j=0; j<4; j++)
           if (Mat[i][j]==1) nb++;
      cout <<"La somme = "<<S<<endl;
      cout <<"La Moyenne = "<<M<<endl;
      cout <<"Nombre d'occurence de 1 = "<<nb<<endl;
}
```

## Exercice 11

```
#include <iostream>
using namespace std;
int main()
{
      const int N=10 ;
    int i, j, A[N][N] ;

    for (i =0; i<N; i++)
    {
      for (j =0; j<N; j++)
      {
           if (j>i)
              A[i][j]=0;
        else
           if ((i==j)||(j==0))
             A[i][j]=1;
           else
             A[i][j] = A[i-1][j-1] + A[i-1][j] ;
      }
    }
    //Affichage
    for (i =0; i<N; i++)
    {
       for (j =0; j<N; j++)
            cout<<A[i][j]<<"\t" ;
       cout<<"\n";
      }
}
```

## Exercice 12

L'objectif de cet exercice est d'apprendre à établir une relation arithmétique entre l'indice de ligne et de colonne, dans ce cas, la somme peut être ramenée à un calcul de somme sur un tableau à 1 dimension.

```
#include <iostream>
using namespace std;
int main()
{
    const int N=8 ;
    int i, j, L, C,  MAT[N][N] ;

    for (i =0; i<N; i++)
      for (j =0; j<N; j++)
          MAT[i][j]=0;

    cout<<"Entrez une position: ";
    cin>>L>>C;
    // Marquage se la première diagonale
       if (L-C>0)
             for (j=0; j<N-(L-C); j++) MAT[j+L-C][j]=1;
       else
         if (L-C<0)
             for (i=0; i<N-(C-L); i++)  MAT[i][i+C-L]=1;
         else
             for (i=0; i<N; i++)  MAT[i][i]=1;

        // Marquage de la deuxième diagonale
        if (L>N-C-1)
             for (i=C+L-N+1; i<N; i++) MAT[i][C+L-i]=1;
        else
          if (L<N-C-1)
             for (i=0; i<=C+L; i++)  MAT[i][C+L-i]=1;
          else
              for (i=0; i<N; i++)  MAT[i][N-i-1]=1;

         //Affichage de la Matrice
         for (i =0; i<N; i++)
         {
             cout<<endl;
             for (j =0; j<N; j++)
            cout<<MAT[i][j]<<" ";
         }
}
```

**Exercice 13**

```
#include <iostream>
#include <string.h>
using namespace std;
#define N 50
int main()
{
      int i, j;
      char ch[N];
      bool Pal;
    cout<<"Entrez une chaine : ";
    gets(ch);
    i=0;
    j=strlen(ch)-1;
    Pal=true;
    while ((i<j) && Pal)
    {
      if (ch[i]!=ch[j])
         Pal=false;
      else
      {
       i++;
       j--;
         }
    }
    if (Pal)
      cout<<"La chaine est un palindrom";
    else
      cout<<"La chaine n'est pas un palindrom";

}
```

**Exercice 14**

```
#include <iostream>
#include <string.h>
using namespace std;
#define N 250
int main()
{
      int i, j, T, Pos;
      bool trouve;
      char PH[N], Alpha[27]="abcdefghijklmnopqrstuvwxyz";
      //La dernière case est réservée au caractère de fin de chaine  '\0'

      cout<< "Entrez la phrase a coder (en Miniscule): ";
    gets(PH);
    T=strlen(PH);
    for (i=0; i<T; i++)
    {
        if (PH[i]=='z')
                PH[i]='a';
           else
           {
                 j=0; trouve=false;
                 do
                 {
                   if (Alpha[j]==PH[i])
                      trouve=true;
                   j++;
                 }
                 while ((j<26)&&(!trouve));
                 if (trouve)
                    PH[i]=Alpha[j];
           }

      }
      cout<<"Phrase crypte: "<<PH;

      return 0;
}
```

## Exercice 15

```cpp
#include <iostream>
using namespace std;
int main()
{
        const int N=10;
        int i,j, A, Age[N] ;
        string B, Nom[N];
        bool Trie;

    //Lecture du nom et l'age

    for (i=0; i<N; i++)
    {
       cout<<"Nom "<<i<<" : " ;
        cin>>Nom[i];
       cout<<"Age "<<i<<" : ";
      cin>>Age[i];
    }
      //Tri par age (tri à bulle)
      Trie=false;
      while (not Trie)
      {
            Trie=true;
            for (i=0; i<N-1; i++)
            {
                  if (Age[i]>Age[i+1])
                  {
                    A = Age[i];
                    Age[i]=Age[i+1];
                    Age[i+1]=A;

                    B = Nom[i];
                    Nom[i]=Nom[i+1];
                    Nom[i+1]=B;

                    Trie=false;
                }
            }
      }

    for (i=0; i<N; i++)
    {
      cout<<Nom[i]<<"  "<<Age[i]<<endl;
    }
}
```

```cpp
#include <iostream>
using namespace std;
int main()
{
        const int N=10;
        int i,j, A, Age[N] ;
        string B, Nom[N];
        bool Trie;

    //Lecture du nom et l'age

    for (i=0; i<N; i++)
    {
       cout<<"Nom "<<i<<" : " ;
      cin>>Nom[i];
       cout<<"Age "<<i<<" : ";
      cin>>Age[i];
    }
   // Tri par age (tri à bulle)
      Trie=false;
      while (not Trie)
      {
            Trie=true;
            for (i=0; i<N-1; i++)
            {
                  if (Nom[i]>Nom[i+1])
                  {
                    A = Age[i];
                    Age[i]=Age[i+1];
                    Age[i+1]=A;

                    B = Nom[i];
                    Nom[i]=Nom[i+1];
                    Nom[i+1]=B;

                    Trie=false;
                }
            }
      }

    for (i=0; i<N; i++)
    {
      cout<<Nom[i]<<"  "<<Age[i]<<endl;
    }
}
```

## Exercice 16

```
Algorithme Ex01 ;
Const N=100 ;
Structure  Adresse
    Rue : Chaine ;
    Ville : Chaine ;
    C_post : entier ;
FinStructure ;
Structure  Etudiant
```

```
Debut
  Ecrire("Donner le nom : ") ;
  Lire(Etd.Nom) ;
  Ecrire("Donner le prénom : ") ;
  Lire(Etd.Prenom) ;
  Ecrire("Donner la ville : ") ;
  Lire(Etd.Adr.Ville) ;
```

## Exercice 17

**1)**

```
Algorithme  Ex02 ;
Const N=100 ;
Structure  date
    Annee : entier ;
    Mois : entier ;
    Jour : entier ;
FinStructure ;
Structure  colis
    Poids : réel ;
    Date_env : date ;
    Destin : chaine ;
    Etat : booleen ;
FinStructure ;
Var  TAB[N] : colis ;
   i, S,J,M,A : entier ;
   PMAX : réel ;
   trouve :Booléen ;
```

**2)**

```
Debut
  //Initialisation du tableau
  Pour (i allant de 1 à N) faire
      Lire(TAB [i].Poids) ;
      Lire(TAB [i].Date_env.Annee) ;
      Lire(TAB [i]. Date_env.Mois) ;
      Lire(TAB [i]. Date_env.Jour) ;
      Lire(TAB [i].Destin) ;
      Lire(TAB [i].Etat) ;
  FinPour ;
```

**3)**

```
S ← 0 ;
Pour (i allant de 1 à N) faire
   S ← S + TAB [i].Poids ;
FinPour
Ecrire("Le nombre total : ", S) ;
```

**4)**

```
//Les colis non envoyés
Pour (i allant de 1 à N) faire
   Si (TAB[i].Etat = faux) Alors
      Ecrire(i) ;
   FinSi
FinPour
```

**5)**

```
// Le colis non envoyé le plus lourd
PMAX ← 0 ;
Pour (i allant de 1 à N) faire
   Si (TAB [i]. Etat=faux) et (TAB [i].Poids >PMAX) Alors
      PMAX ← TAB [i].Poids ;
   FinSi;
```

**6)**

```
Ecrire("Entrez la date d'envoi : ") ;
lire(J, M, A) ;
i ← 1 ;  trouve ← faux ;
Tantque (i <=N) et (Non trouve) faire
   Si  (TAB[i].Date_env.Annee = A) et
      (TAB[i].Date_env.Mois = M)  et
      (TAB[i].Date_env.Jour = J)   Alors
         trouve ← vrais ;
   Sinon
         i ←i +1 ;
   FinSi;
FinTantQue ;
```

**7)**

```
Ecrire("Entrez l'ancienne et la nouvelle
        destination ") ;
lire(Dest_A , Dest_N ) ;
 Pour (i allant de 1 à N) faire
    Si (TAB[i].Destin = Dest_A ) Alors
       TAB[i].Destin ← Dest_N ;
    FinSi
 FinPour ;

Fin.
```

## Exercice 18

**1)**

```
Algorithme  Ex02 ;
Const Max=100 ;
Structure  Etudiant
    User : chaine ;
    Passw: chaine ;
    Nompre : chaine ;
    Dnais : chaine ;
    Groupe : entier ;
FinStructure ;
Var  Et[Max] : Etudiant;
```

**2)**

```
Debut
  //Initialisation du tableau
  Faire
     Ecrire (" nb d'étudiant :") ;
     Lire(N)
  TQ (N<1) ;
  Pour (i allant de 1 à N) faire
     Lire(Et[i].User) ;
     Lire(Et[i].Passw) ;
```

**3)**

```
Lire (NP) ; trouve ← faux ;
Tantque (i <=N) et (Non trouve) faire
   Si  (Et[i].Nompre = NP)   Alors
         Trouve ← vrais ;
   Sinon
         i ←i +1 ;
   FinSi;
FinTantQue ;
```

**4)**

```
Si N < Max Alors
   Lire(Et[N+1].User) ;
   Lire(Et[N+1].Passw) ;
   Lire(Et[N+1].Nompre ) ;
   Lire(Et[N+1].Dnais ) ;
   Lire(Et[N+1].Groupe ) ;
   N ← N + 1;
Sinon
   Ecrire ("Ajout impossible" );
FinSi
```

**5)**

```
Si N < Max Alors
     Lire (Pos)      // Pos <= N
     pour i ← N à Pos pas -1 faire
          Et[i+1] ←E[i]
     FinPour
     Lire(Et[Pos].User) ;
     Lire(Et[Pos].Passw) ;
     Lire(Et[Pos].Nompre ) ;
     Lire(Et[Pos].Dnais ) ;
     Lire(Et[Pos].Groupe ) ;
     N ← N + 1;
FinSi
```

**6)**

```
Si N>=1 Alors
    Lire (Pos)     // Pos <= N
    pour i ← Pos à N-1 faire
       Et[i] ←Et[i + 1]
    FinPour
    N ← N - 1;
FinSi
```

## Exercice 19

```
#include <iostream>
using namespace std;

int main()
{
    struct Date  {
       int Jour;
       string Mois;
       int Annee;
    };
    struct Personne  {
       string Nom;
       int Age;
       Date D_Embauche; };

    Personne P1;
    //Saisie
    cout<<"Nom: "; cin>>P1.Nom;
    cout<<"Age: "; cin>>P1.Age;
```

## Exercice 20

```
#include <iostream>
using namespace std ;
const int N = 4 ; // nombre de points d'une courbe
struct point
{ char c ;
  int x, y ;
} ;
int main()
{
  point courbe [N] ;
  int dx, dy;

   //Lecture des points

    for (int i=0 ; i<N ; i++)
    { cout << "Entrez le nom (1 caractere) et les coordonnees du point "
           << i+1 << "\n" ;
      cin >> courbe[i].c >> courbe[i].x >> courbe[i].y ;
    }


  //Déplacement des points

   cout<< "Donner les valeurs de deplacement: ";
   cin>> dx >> dy;
   for (int i=0 ; i<N ; i++)
   {
     courbe[i].x+=dx;
     courbe[i].y+=dy;
   }

   //Aaffichage des points

   for (int i=0 ; i<N ; i++)
      cout << "point " << courbe[i].c << " de coordonnees "
           << courbe[i].x << " " << courbe[i].x << "\n" ;

   return 0;
}
```

## Exercice 21

```
#include <iostream>
using namespace std;
#define Max 100
#define M 3
int main()
{
       struct Etudiant  {
       int NInsc;
       string Nom_Pren;
       int groupe;
       float Notes[M];
    };
    Etudiant classe[Max];
       int num, indice, i, j, N, pos;
       bool trouve;
       float moy;

       do {
          cout<<"Entrez le nombre d etudiants ";
          cin>>N;
       } while (N<1);

       //Initialisation
       for (i=0; i<N; i++)
       {
              cout<<"Etudiant "<<i+1<<" : \n" ;
              cout<<"Num Inscription: ";
              cin>>classe[i].NInsc;
              cout<<"Nom et Prenom: ";
              cin>>classe[i].Nom_Pren;
              cout<<"Groupe: ";
              cin>>classe[i].groupe;
              for (j=0; j<M; j++)
              {
                     cout<<"Note"<<j+1<<" : ";
                     cin>>classe[i].Notes[j];
              }
       }
    //Recherche d'un étudiant  et calcul de sa la moyenne
       cout<<"Donner le numero d'inscription: ";
       cin>>num;
       trouve=false; i=0;
       while ((!trouve) && (i<N)) {

              if (classe[i].NInsc==num)
              {
                     trouve= true;
                     indice=i;
              }
              else
                  i++;
       }
```

```
if (!trouve)
            cout<<"L'étudiant n'existe pas";
      else
      {
            moy=0;
            for(i=0; i<M; i++)
                  moy+=classe[indice].Notes[i];
            cout<<"La moyenne = "<<moy/M;
      }

      // Ajout un nouvel étudiant à la fin
      if (N<Max) {

       cin>>classe[N+1].NInsc;
       cin>>classe[N+1].Nom_Pren;
       cin>>classe[N+1].groupe;
       for (j=0; j<M; j++)
            {
                  cout<<"Note"<<j+1<<" : ";
                  cin>>classe[N+1].Notes[j];
            }
       N=N+1;
      }
    else
      cout<<"Ajout impossible";

    // Insertion d'un nouvel étudiant à une position donnée
    if (N>=Max)
        cout<<"Ajout impossible";
      else
      {
       do
         {
               cout<<"donnez la position";
               cin>>pos;      //  Pos <= N
         }
         while ((pos>N)|| (pos<1));
       for (i=N; i>1; i++)
         classe[i+1] = classe[i];

       cin>>classe[pos].NInsc;
       cin>>classe[pos].Nom_Pren;
       cin>>classe[pos].groupe;
       for (j=0; j<M; j++)
         {
            cout<<"Note"<<j+1<<" : ";
            cin>>classe[pos].Notes[j];
         }
       N=N+1;
   }
```

```
// Suppression d'un étudiant à une position donnée
   if (N<1)
          cout<<"Il n y a plus d etudiant";
       else
       {
        do
          {
                cout<<"donnez la position";
                cin>>pos;       //  Pos <= N
          }
        while ((pos>N)|| (pos<1));

          for (i=pos; i<N; i++)
              classe[i]=classe[i + 1];

        N=N-1;
   }

 // affichage
       for (int i=0; i<N; i++)
       {
              cout<<"Etudiant "<<i+1<<" : \n" ;
              cout<<"Num Inscription: "<<classe[i].NInsc<<"\n";
              cout<<"Nom et Prenom: "<<classe[i].Nom_Pren<<"\n";
              cout<<"Groupe: "<<classe[i].groupe<<"\n";
              for (int j=0; j<M; j++) {
                    cout<<"Note"<<j+1<<" : "<<classe[i].Notes[j]<<"\n";
              }
       }

       return 0;

}
```

## Références


- Claude Delannoy. Apprendre à programmer en Turbo C. Chihab- EYROLLES, (1994).
- Damien Berthet et Vincent Labatut. Algorithmique & programmation en langage C-vol.1 : Supports de cours. Licence. Algorithmique et Programmation, Istanbul, Turquie. (2014), pp.232. vol.3 : Corrigés de travaux pratiques. Licence. Algorithmique et
- D. MOKHTARI," Le PascalFacile", Edition HAMDI, Algérie(1999).
- E. MORVANT,"Algorithmique et programmation Pascal", Saint-Louis(2009).
- L. AMIROUCHE et S.KERIREM," Introduction à l'Algorithmique" (2009).
- S. GRAINE,"Algorithmique et structures de donnée", les éditions l'Abeille(2001).
- Thomas H. Cormen, Algorithmes Notions de base Collection : Sciences Sup, Dunod, (2013).
- Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest Algorithmique - 3ème édition - Cours avec 957 exercices et 158 problèmes Broché, Dunod, (2010).
- Rémy Malgouyres, Rita Zrour et Fabien Feschet. Initiation à l'algorithmique et à la
- programmation en C : cours avec 129 exercices corrigés. 2ième Edition. Dunod, Paris, (2011). ISBN : 978-2-10-055703-5.