Correction épreuve algo 2024 Session Principale

Exercice1 :

Exercice 1 : (3 points)

Soit l'algorithme suivant de la fonction **Inconnue** :

```
Fonction Inconnue(n, k : entier):.....
DEBUT
        Si k > n Div 2 Alors
                Retourner Vrai
        Sinon Si n Mod k = 0 Alors
                Retourner Faux
        Sinon Retourner Inconnue(n, k+1)
        FinSi
FIN
```

Travail demandé :

1) Donner le typè de retour de la fonction **Inconnue**.
2) Sachant que la valeur de **k** lors de l'appel de la fonction **Inconnue** est toujours égale à **2** :
   a. Donner une trace d'exécution manuelle de la fonction **Inconnue** pour les deux cas suivants :
      - Cas 1 : n = 4
      - Cas 2 : n = 11
   b. En déduire le rôle de la fonction **Inconnue**.
3) Ecrire un algorithme de la fonction **Inconnue(n)** en utilisant un traitement itératif donnant le même résultat.

1) Le type de retour de la fonction Inconnue est booléen

2)

a)

Pour n=4 :

Inconnue(4,2)→Retourner Faux Car (4 mod 2=0)

Pour n=11 :
Inconnue(11,2)→Inconnue(11,3)→Inconnue(11,4)→Inconnue(11,5)→ Inconnue(11,6)→Retourner Vrai Car (6>5)

b) Vérifier Si un nombre Supérieur à 1 est premier ou non.

3) Fonction Inconnue(n :entier) :Booléen

```
Début
    d←2
    Tantque(n mod d < >0 )et (d<=n div 2) Faire
            d←d+1
    Fin Tantque
    Retourner (d > n div 2)
Fin
```

T.D.O.L

Objet	Type/Nature
d	entier

Un autre méthode pour vérifier la primalité d'un nombre :

```
Fonction Inconnue(n :entier) :Booléen
Début
    S←0
    Pour i de 2 à n div 2 Faire
            Si( n mod i =0  ) Alors
                S←S+i
            Fin Si
    Fin Pour
    Retourner (S=0)
Fin
```

T.D.O.L

Objet	Type/Nature
S,i	entier

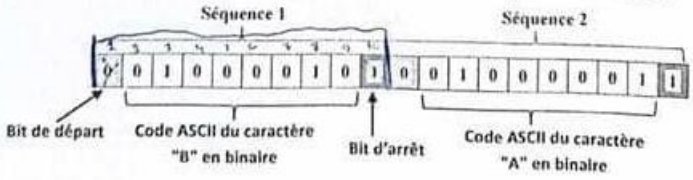
Exercice 2 :

Exercice 2 : (5 points)

La méthode de transmission en série asynchrone est une technique de transmission de données numériques qui permet d'envoyer les données d'un message caractère par caractère. L'envoi d'un caractère se fait selon le principe suivant :

- Convertir le code ASCII du caractère à transmettre en binaire sur **8 bits**.
- Ajouter à gauche des 8 bits obtenus un bit de départ dont la valeur est **0** et à droite un bit d'arrêt dont la valeur est **1** pour former une séquence contenant **10 bits**.

Exemple : pour la chaîne de caractères "BA" on obtient les deux séquences suivantes :



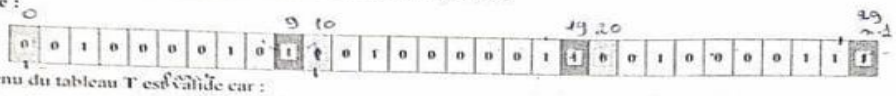
On se propose de vérifier la validité du contenu d'un tableau **T** contenant le codage en binaire, d'un message initial formé uniquement par des lettres majuscules.

Le contenu du tableau **T** est valide s'il vérifie les contraintes suivantes :

- La taille du tableau **T** est un multiple de **10**.
- Chaque case du tableau **T** doit contenir soit la valeur **0** soit la valeur **1**.
- Chaque séquence de **10** bits doit commencer par **0** (bit de départ) et doit se terminer par **1** (bit d'arrêt).
- Les **8** bits compris entre le bit de départ et le bit d'arrêt de chaque séquence doivent représenter l'équivalent binaire du code ASCII d'une lettre majuscule.

Exemple :

Pour T=



Le contenu du tableau **T** est valide car :

- La taille du tableau **T** est égale à 30 qui est un multiple de 10.
- Toutes les cases du tableau **T** ne contiennent que les valeurs **0** et **1**.
- Chaque séquence de **10** bits commence par **0** et se termine par **1**.
- Les **8** bits compris entre le bit de départ et le bit d'arrêt de chaque séquence représentent l'équivalent binaire du code ASCII d'une lettre majuscule.

  En effet :
  - l'équivalent en décimal des 8 bits **01000010** est **66** qui correspond au code ASCII de la lettre "B".
  - l'équivalent en décimal des 8 bits **01000001** est **65** qui correspond au code ASCII de la lettre "A".
  - l'équivalent en décimal des 8 bits **01000011** est **67** qui correspond au code ASCII de la lettre "C".

Travail demandé :

Ecrire un algorithme d'une fonction booléenne **Verif (T , N)** qui permet de vérifier, selon les contraintes décrites précédemment, si le contenu du tableau **T** de type **Tab** et de taille **N** est valide ou non.

Algorithme de la fonction Verif(T,N) :

```
Fonction Verif(T :TAB,N :entier) :Booléen
Début
Si(N mod 10 <>0 ) ou Non(Verif_Saisie(T,N) ) Alors
            Retourner Faux
Sinon
  Ch←Extraire_Tab_Ch(T,N)
  Répéter
     Ch1←sous_chaine(Ch,1,7) //Fin Exclu
     Test←(Ch[0]='0' et Ch[7]='9' et ConvB_10(Ch1,2) dans [65..90])
     Ch←effacer(Ch,0,8)  //Fin Exclu

  Jusqu'à (Ch='') ou ( Non(Test) )
Fin Si

Retourner(Test)

Fin
```

T.D.O.L

Objet	Type/Nature
Verif_Saisie Extraire_Tab_Ch ConvB_10	Fonction
Ch,Ch1 Test	Chaine de Caractères Booléen

Fonction Verif_Saisie(T :TAB,N :entier) :Booléen

Début

i←0,Test←Vrai

Répéter

Si(Non(T[i] dans [0..1])) Alors

Test←Faux

Fin Si

i←i+1

jusqu'à (Non(Test)) ou (i=N-1)

Retourner (Test)

Fin

T.D.O.L

Objet	Type/Nature
i	Entier
Test	Booléen

Fonction Extraire_Tab_Ch(T:TAB,N :entier) :Chaine

Début

Ch←"

Pour i de 0 à N-1 Faire

Ch←Ch+convch(T[i])

Fin Pour

Retourner(Ch)

Fin

T.D.O.L

Objet	Type/Nature
i	Entier
Ch	Chaine

Fonction ConvB_10(Ch :Chaine,B :entier) :Chaine

Début

S←0

P←1

Pour i de long(Ch)-1 à 0 (Pas=-1) Faire

Si(Ch[i] dans ['0'..'9'])Alors

S←S+(ord(ch[i])-48)*P

Sinon

S←S+(ord(ch[i])-55)*P

Fin Si

P←P*2

Fin Pour

Retourner (S)

Fin

T.D.O.L

Objet	Type/Nature
P,S,i	Entier

Exercice 3 :

Exercice 3 : (4,5 points)

On se propose de vérifier si un entier **M**, supérieur ou égal à **2**, est **n-rond**.

M est dit **n-rond** s'il existe un entier **n** tel que le plus grand facteur premier de **M**, noté **P**, vérifie la condition $P \leq \sqrt[n]{M}$ (avec $\sqrt[n]{M}$ est la racine $n^{ième}$ de **M**).

Afin de calculer une valeur approchée de la racine $n^{ième}$ de **M** ($\sqrt[n]{M}$), on utilise la suite x définie comme suit :

$$x \begin{cases} x_0 = \frac{M}{2} \\ x_k = \frac{1}{n}\left((n-1)\,x_{k-1} + \frac{M}{(x_{k-1})^{n-1}}\right) & pour\ k \geq 1 \end{cases}$$

Travail demandé :

1. Ecrire un algorithme d'une fonction **RacineN (M , n)** qui permet de retourner une valeur approchée de la racine $n^{ième}$ de **M**, en utilisant la suite x. Le calcul s'arrête lorsque $|x_k - x_{k-1}| \leq 10^{-4}$ et la valeur approchée de $\sqrt[n]{M}$ correspond alors au dernier terme calculé x_k.

2. Ecrire un algorithme d'une fonction **Facteur (M)** qui permet de retourner le plus grand facteur premier **P** de l'entier **M**.

Exemples :

- Pour **M=21** la fonction **Facteur** retourne 7 car sa décomposition en facteurs premiers donne $21 = 3 * 7$.
- Pour **M=432** la fonction **Facteur** retourne 3 car sa décomposition en facteurs premiers donne $432 = 2^4 * 3^3$.

3. En faisant appel aux deux fonctions **RacineN** et **Facteur**, écrire un algorithme d'une fonction **NRond (M)** qui permet de retourner le plus grand entier **n** qui vérifie $P \leq \sqrt[n]{M}$ dans le cas où **M** est **n-rond** et de retourner **-1** dans le cas contraire (avec **P** est le plus grand facteur premier de **M**).

Exemples :

- **NRond (432)** retourne **5** car pour **P=3**, qui est le plus grand facteur premier de **432**, l'entier **n=5** correspond au plus grand entier qui vérifie $3 \leq \sqrt[5]{432}$. En effet on a :

n	Racine $n^{ième}$	Constatation
2	$\sqrt{432} = 20{,}7846$	$3 < \sqrt{432}$
3	$\sqrt[3]{432} = 7{,}5595$	$3 < \sqrt[3]{432}$
4	$\sqrt[4]{432} = 4{,}5590$	$3 < \sqrt[4]{432}$
5	$\sqrt[5]{432} = 3{,}3658$	$3 < \sqrt[5]{432}$
6	$\sqrt[6]{432} = 2{,}7494$	$3 > \sqrt[6]{432}$

- **NRond (21)** retourne **-1** car pour **P=7**, qui est le plus grand facteur premier de **21**, il n'existe pas un entier **n** tel que $7 < \sqrt[n]{21}$. En effet $7 > \sqrt[3]{21} = 4{,}5825$.

1) Fonction RacineN(M,n :entier) :Réel

Début

X←M/2

Répéter

Xpred←X

X←1/n*((n-1)*Xpred+M/Puiss(Xpred,n-1))

Jusqu'à(Abs(Xpred-X)<=0.0001)

Ou <=10e-4

Retourner(X)

Fin

T.D.O.L

Objet	Type/Nature
Xpred,X	Réel
Puiss	Fonction

```
//Méthode récursive pour déterminer La Puissnce a^n
Fonction Puiss(a,n :entier) :Entier
Début
    Si (n=0) Alors
        Retourner(1)
    Sinon
        Retourner ( a*Puiss(a,n-1) )
    Fin Si
Fin

//Méthode itérative
Fonction Puiss(a,n :entier) :Entier
Début
    P←1
    Pour i de 1 à n Faire
        P←P*a
    Fin Pour
    Retourner(P)
Fin
```

2) **Fonction Facteur(M :entier) :Entier**

```
Début
    d←2
    Répéter
        Si(M mod d=0) Alors
            M←M div d
        Sinon
            d←d+1
        Fin Si
    Jusqu'à(M=1)
    Retourner(d)
Fin
```

T.D.O.L

Objet	Type/Nature
d	Entier

3) **Fonction NRond(M :entier) :Entier**

```
Début
    P←Facteur(M)
    N←1
    Tantque(P<=RacineN(M,N+1) ) Faire
            N←N+1
    Fin Tantque
    Si (N=1) Alors
        Retourner (-1)
    Sinon
        Retourner(N)
    Fin Si
Fin
```

T.D.O.L

Objet	Type/Nature
P,N	Entier

Exercice4 :

Exercice 4 : (7,5 points)

Dans le but d'attribuer des cadeaux à des invités présents dans une soirée, on se propose de choisir les personnes dont le nom est triangulaire et ayant le score le plus élevé. Ces personnes seront déclarées gagnantes.

Un nom est dit **triangulaire**, si son score est un nombre triangulaire.

Le score d'un nom est la somme des rangs dans l'alphabet de toutes les lettres qui le constituent.

Un nombre **S** est triangulaire s'il existe un entier ***n*** tel que :

$$S = \frac{n(n+1)}{2} \quad pour \quad n \geq 1$$

Exemple 1 : "Saber" est triangulaire car son score **S** est un nombre triangulaire.

En effet :

S = 45 est un nombre triangulaire car $45 = \frac{9*(9+1)}{2}$

Exemple 2 : "Ali" n'est pas triangulaire car son score **S** n'est pas un nombre triangulaire.

En effet :

S = 22 n'est pas un nombre triangulaire car il n'existe pas un entier **n** tel que $22 = \frac{n*(n+1)}{2}$

Sachant que les noms des invités sont enregistrés dans le fichier texte **"Invites.txt"**, on procède comme suit pour sélectionner les personnes gagnantes :

- Transférer, à partir du fichier **"Invites.txt"** les noms triangulaires vers un nouveau fichier d'enregistrements nommé **"Triangulaires.dat"** où chaque enregistrement est constitué de deux champs :
  - Nom : le nom de la personne.
  - Score : le score du nom de la personne.
- Afficher les noms des personnes gagnantes (ayant le score le plus élevé).

Travail demandé :

1. Ecrire un algorithme du programme principal en respectant le procédé décrit précédemment et en le décomposant en modules.
2. Ecrire un algorithme pour chaque module envisagé.

NB :

- Le fichier **"Invites.txt"** est déjà enregistré sous la racine du disque **C** et il contient au maximum **50** noms à raison d'un nom par ligne.
- Le fichier **"Triangulaires.dat"** sera créé sous la racine du disque **C**.

1) Algorithme du Programme Principal

Algorithme Probléme
Début
  Transfert_Noms(T,Finv,Ftr,N)
  Affiche_Gangants(T,N)
Fin

T.D.O.G

Objet	Type/Nature
T	TAB
Finv	Fiche
Ftr	Texte
N	Entier

Nouveaux Type

Inv=enregistrement
  nom :Chaine
  score :entier
Fin Inv
 	
TAB=tableau de 50 Inv
Fiche=Fichier de Inv

2) Les Sous-programmes

Procédure Transfert_Noms(T :TAB,@Finv :Fiche,@Ftr :Texte@N :entier)

Début

Ouvrir(‘’C:\Invites.txt”,Finv,”r”) //lecture

Ouvrir(‘’C:\Triangulaires.dat”,Ftr,”wb”) //écriture

N←0

Tantque(Non(Fin_Fichier(Finv)))Faire

Lire_ligne(Finv,E.nom)

E.score←Calcul_Score(E.nom)

Si(Triangulaire(E.score)) Alors

Ecrire(Ftr,E)

T[N]←E

N←N+1

Fin Si

Fin Tantque

Fermer(Finv)

Fermer(Ftr)

Fin

T.D.O.L

Objet	Type/Nature
E	**Inv**
Calcul_Score	**Fonction**
Triangulaire	

```
Fonction Calcul_Score(Ch :Chaine) :Entier
Début
   S←0
   Pour i de 0 à long(Ch)-1 Faire
        S←S+ord(Ch[i])-64
   Fin Pour
   Retourner(S)
Fin
```

T.D.O.L

Objet	Type/Nature
i,S	**Entier**

```
Fonction Triangulaire(Nb :entier) :Booléen
Début
   N←1
   Tantque( N*(N+1) div 2 < Nb ) Faire
        N←N+1
   Fin Tq
   Retourner( N*(N+1) div 2 =Nb )
Fin
```

T.D.O.L

Objet	Type/Nature
N	**Entier**

```
Fonction Max_Score(T :TAB,N :entier) :Entier
Début
   Max←T[0].score
   Pour i de 1 à N-1 Faire
        Si (T[i].score>Max) Alors
             Max←T[i].score
        Fin Si
   Fin Pour
```

T.D.O.L

Objet	Type/Nature
Max,i	**Entier**

```
 Retourner(Max)
Fin

Procédure Affiche_Gangants(T :TAB,N :entier)
Début
    Max←Max_Score(T,N)
    Pour i de 0 à N-1 Faire
        Si( T[i].score=Max ) Alors
            Ecrire(‘’le gangant est ‘’,T[i].nom)
        Fin Si
    Fin Pour                                    T.D.O.L
Fin
```

Objet	Type/Nature
i,Max	**Entier**
Max_Score	**Fonction**

Rappels:

5.3. Les fonctions sur les chaînes de caractères

Les fonctions sur le type chaîne de caractères				
Notation algorithmique	**Notation Python**	**Rôle**	**Exemples en Python - Résultat**	
Lo ← **long** (Ch)	Lo = **len** (Ch)	Retourne un entier représentant le nombre de caractères de la chaîne **Ch** (la longueur de Ch).	Lo = len ("Salut") Lo = len ("L'élève") Lo = len ("")	Lo == 5 Lo == 7 Lo == 0
Po ← **pos** (Ch1, Ch2)	Po = Ch2.**find** (Ch1)	Retourne un entier représentant la position de la 1ère occurrence de **Ch1** dans **Ch2**. Elle retourne **-1** si **Ch1** n'existe pas dans **Ch2**.	Ch1 = "Y" Ch2 = "BAYBAY" Po = Ch2.find (Ch1)	Po == 2
Ch2 ← **sous_chaîne** (Ch1, Début, Fin)	Ch2 = Ch1 [Début : Fin]	Retourne une copie de la chaîne **Ch1** à partir de l'indice **Début** à l'indice **Fin** (position **Fin** exclu).	Ch1 = "BACCALAUREAT" Ch2 = Ch1 [5 : 12]	Chr == "LAUREAT"
Ch2 ← **effacer** (Ch1, d, f)	Ch2= Ch1[: d]+Ch1 [f:]	Retourne une chaîne **Ch2** après avoir effacer, de la chaîne **Ch1**, les caractères de la position **d** à la position **f** (**f** exclu).	Ch1 = "INFORMATIQUE" Ch2 = Ch1[:6] + Ch1[11:]	Ch2 == "INFORME"
ChM ← **majus** (Ch)	ChM = Ch.**upper** ( )	Retourne la chaîne **ChM** représentant la conversion en Majuscule de la chaîne **Ch**.	Ch = "Bonjour" ChM = Ch.upper ( )	ChM =="BONJOUR"
Ch ← **convch** (**X**)	Ch = **str** (**X**)	Retourne la conversion du nombre **X** en une chaîne de caractères.	N = 358 Ch = str (N)	Ch == "358"
Test ← **estnum** (Ch)	**Pas de correspondance**. Toutefois, on pourra utiliser **isnumeric** ( ) malgré qu'elle ne répond pas aux exigences ou bien développer un module qui permet de réaliser cette tâche.	Retourne **VRAI** si la chaîne **Ch** est convertible en une valeur numérique et **FAUX** dans le cas contraire.	Ch = "489" Test = Ch . isnumeric ( ) Ch = "489.56" Test = Ch . isnumeric ( )	Test == True Test == False
N ← **valeur** (Ch)	N = **int** (Ch) *ou bien* N = **float** (Ch)	Retourne la conversion d'une chaîne **Ch** en une valeur numérique, si c'est possible.	Ch = "489" N = int (Ch) Ch = "489" N = float (Ch)	N == 489 N == 489.0

5.2. Les fonctions sur les caractères

Les fonctions sur le type caractère				
Notation algorithmique	**Notation Python**	**Rôle**	**Exemples en Python - Résultat**	
N ← **Ord** (Ca)	N = **ord** (Ca)	Retourne le code ASCII du caractère **Ca**.	N = ord ("0") N = ord ("A") N = ord ("a")	N == 48 N == 65 N == 97
Ca ← **Chr** (**X**)	Ca = **chr** (**X**)	Retourne le Caractère dont le code ASCII est **X**.	Ca = chr (50) Ca = chr (90)	Ca == "2" Ca == "Z"

5. Les fonctions prédéfinies

5.1. Les fonctions arithmétiques

Les fonctions sur les types numériques				
Notation algorithmique	**Notation Python**	**Rôle**	**Exemples en Python - Résultat**	
N ← **abs** (X)	N = **abs** (X)	Retourne la valeur absolue de **X**.	N = abs (-20) N = abs (.5.8)	N == 20 N == 5.8
N ← **ent** (X)	N = **int** (X)	Retourne un **Entier** représentant la partie entière de **X**.	N = int (5.2) N = int (-5.8)	N == 5 N == -5
N ← **arrondi** (X)	N = **round** (X)	Retourne **l'Entier** le plus proche de **X**. **N.B.** : En Python, si la partie fractionnaire est égale à **5**, l'entier **Pair** le plus proche est retourné.	N = round (2.2) N = round (2.8) N = round (2.5) N = round (3.5)	N == 2 N == 3 N == 2 N == 4
N ← **racinecarré** (X)	**from math import sqrt** N = **sqrt** (X)	Retourne un **Réel** représentant la racine carrée de **X**. Si **X** < 0, elle provoque une erreur.	N = sqrt (9) N = sqrt (25.0) N = sqrt (-5)	N == 3.0 N == 5.0 Erreur
N ← **aléa** (Vi , Vf)	**from random import randint** N = **randint** (Vi , Vf)	Retourne un entier d'une façon aléatoire et automatique de l'intervalle [**Vi** , **Vf**].	N = randint (2 , 5) N pourra avoir 2 ou 3 ou 4 ou 5	

Appartenance	**Notation**	
	Algorithmique	**Python**
Ensemble	x ∈ {val1, val2, …, valn}	**x in {val1, val2, …}**
Intervalle	x ∈ [val1..valn] *ou bien* **val1 ≤ x ≤ valn**	Pour les entiers : **x in range(val1, valn+1)** *ou bien* **val1 <= x <= valn** Pour les caractères : **ord(x) in range(ord(val1),ord(valn)+1)** *ou bien* **val1 <= x <= valn**

Opérateurs logiques (booléens) & Priorités & Tables de vérité						
Opération	**Opérateurs & Priorités & Tables de vérité**					
	Algo.	**Python**	**Priorité**	**Table de vérité**		
Négation	**NON**	**not**	**1**	**A**	**not (A)**	
				True	**False**	
				False	**True**	
Conjonction	**ET**	**and**	**2**	**A**	**B**	**A and B**
				True	True	**True**
				True	False	**False**
				False	True	**False**
				False	False	**False**
Disjonction	**OU**	**or**	**3**	**A**	**B**	**A or B**
				True	True	**True**
				True	False	**True**
				False	True	**True**
				False	False	**False**

6. Les types de données structurées et leurs déclarations

6.1. Les tableaux à une seule dimension

6.1.1. Déclaration en algorithmique

❖ 1ère méthode

Tableau de Déclaration des Objets (T.D.O)	
Objet	**Type/Nature**
Nom_Tableau	**Tableau** de N **Type_élément**

❖ 2ème méthode

Tableau de Déclaration des Nouveaux Types (T.D.N.T)
Nom_Type_Tableau = **Tableau** de N **Type_élément**

Tableau de Déclaration des Objets (T.D.O)	
Objet	**Type/Nature**
Nom_Tableau	Nom_Type_Tableau

6.1.2. Déclaration en Python en utilisant la bibliothèque Numpy

Déclaration dans le cas général
import numpy as np **Nom_Tableau = np.array ([Type_élément ( )] * N [,dtype = object])**

Exemples de déclarations en Python	
Déclaration	**Explication**
import numpy as np T = np.array ([int ( )] * 20)	Pour déclarer un tableau de 20 entiers avec importation de la bibliothèque Numpy.
import numpy as np T = np.array ([float ()] * 100)	Pour déclarer un tableau de 100 réels avec importation de la bibliothèque Numpy.
import numpy as np T = np.array ([str ( )] * 50 , dtype = object) *ou bien* T = np.array ([str] * 50)	Pour déclarer un tableau de 50 chaînes de caractères avec importation de la bibliothèque Numpy.
import numpy as np T = np.array ([bool ( )] * 10)	Pour déclarer un tableau de 10 booléens avec importation de la bibliothèque Numpy.

6.3. Les enregistrements

6.3.1. Déclaration en algorithmique

❖ 1ère méthode

Tableau de Déclaration des Objets (T.D.O)	
Objet	**Type/Nature**
Nom_Enregistrement	**Enregistrement** Champ1 : Type1 Champ2, Champ3 : Type2 ..... ChampN : TypeM **Fin**

❖ 2ème méthode

Tableau de Déclaration des Nouveaux Types (T.D.N.T)
Nom_Type_Enregistrement = **Enregistrement** Champ1 : Type1 Champ2, Champ3 : Type2 ..... ChampN : TypeM **Fin**

Tableau de Déclaration des Objets (T.D.O)	
Objet	**Type/Nature**
Nom_Enregistrement	Nom_Type_Enregistrement

6.4. Les fichiers

6.4.1. Déclaration en algorithmique (1ère méthode)

	Tableau de Déclaration des Objets (T.D.O)	
	Objet	**Type/Nature**
Fichier texte	Nom_Fichier	Fichier texte
Fichier de données (typé)	Nom_Fichier	Fichier de Type _élément

6.4.2. Déclaration en algorithmique (2ème méthode)

	Tableau de Déclaration des Nouveaux Types (T.D.N.T)
Fichier texte	Nom_Type = Fichier texte
Fichier de données (typé)	Nom_Type = Fichier de Type _élément

	Tableau de Déclaration des Objets (T.D.O)	
	Objet	**Type/Nature**
Fichier texte	Nom_Fichier	Nom_Type
Fichier de données (typé)	Nom_Fichier	Nom_Type

N.B. :

- La position initiale du pointeur dans un fichier texte ou de données est **Zéro**.
- **Type_Element** peut être : Entier, Réel, Caractère, chaîne de caractères, Booléen, Enregistrement.

10. Les fonctions et les procédures sur les fichiers

10.1. Les fonctions et les procédures sur les fichiers de données (typés)

Notation algorithmique	Rôle
Ouvrir ("Chemin\Nom_Physique" , *Nom_Logique* , "Mode" **)**	Ouverture d'un fichier. **Mode d'ouverture :** • **"rb"** : Lecture • **"wb"** : Écriture (Création) • **"ab"** : Écriture à la fin du fichier
Notation en Python	
Nom_Logique **= open** ("Chemin\Nom_Physique" , "Mode") *ou bien* *Nom_Logique* **= open** ("Chemin/Nom_Physique" , "Mode")	

Notation algorithmique	Notation en Python	Rôle
Lire (*Nom_Logique* , Objet)	**import pickle as pic** Objet **= pic.load** (*Nom_Logique*)	Lecture d'un enregistrement d'un fichier.
Ecrire (*Nom_Logique* , Objet)	**import pickle as pic** **pic.dump** (Objet , *Nom_Logique*)	Écriture dans un fichier.
Fin_Fichier (*Nom_Logique*)	Pas de correspondance	Retourner **VRAI** si le pointeur est à la fin du fichier sinon elle retourne **FAUX**.
Fermer (*Nom_Logique*)	*Nom_Logique*.**close** ( )	Fermeture du fichier.

N.B. :

- La position initiale du pointeur dans un fichier de données est **Zéro**.
- Les traitements sont réalisés dans la mémoire centrale.