



Ce qu'il faut savoir...

I – Fichiers et enregistrements :

Dans ce chapitre, ce qu'il faut savoir :

- 0- Déclarer des types enregistrements, type fichiers...
- 1- Créer et remplir un fichier de données
- 2- Créer et remplir un fichier texte.
- 3- Ajouter des données à un fichier.
- 4- Modifier des données dans un fichier.
- 5- Supprimer un ou plusieurs données

- 6- Trier un fichier.
- 7- Copier des données d'un fichier de données à un fichier texte et inversement.
- 8- Chercher des données selon un critère.
- 9- Afficher les données d'un fichier de données ou fichier texte.

Exemple :

Votre professeur en matière algorithmique veut informatiser la gestion de ces élèves ; en sauvegardant dans un fichier ses fiches de renseignements. Chaque fiche comporte les renseignements suivants :

- Un identifiant
- Nom et prénom
- Date de naissance
- Sexe (M pour masculin et F pour féminin)

Le professeur devra être capable de réaliser les traitements suivants :

- (1) La saisie et la sauvegarde des fiches de n élèves dans un fichier f (n compris entre 5 et 40).
- (2) La première note de la classe.
- (3) Ajouter un élève.
- (4) Supprimer un élève selon son identifiant.
- (5) Modifier un élève selon son identifiant
- (6) La liste triée des élèves par ordre alphabétique.
- (7) Copier ce fichier dans un fichier TEXTE.

Solution :

0- Structures de données adéquates :

Type
tab =tableau de 5 réels
fiche = enregistrement
id : chaîne[10]
np :chaîne[30]
date_naiss :chaîne[10]
sexe : char
moy :réel
fin fiche
f_fiches = fichier de fiche

Résumé Algorithmique

1. Créer le fichier et enregistrer les élèves :	
Analyse	Algorithme
<p>Résultat =f Traitement : f= Assigner(f,"C:\travail\employe.dat") f= Récréer(f)</p> <p>f= pour k de 1 à n faire Avec fic faire id = données("id : ") np = données("np : ") date_naiss = données("date : ") sexe = données("sexe : ") moy= donnée(" moy : ") fin Avec Ecrire(f, fic) fin pour</p>	<p>0. DEFPROC CREATION (VAR F :femp,n:entier) 1. Assigner (Fe, "C:\travail\employe.dat") 2. Recréer(F) 3. pour k de 1 à n faire Avec fic faire ecrire("id : ") ;lire(id) ecrire("date : ") ;lire(date_naiss) ecrire("np : ") ;lire(np) ecrire("sexe : ") ;lire(sexe) ecrire("moy : ") ;lire(moy) fin Avec Ecrire(f, fic) fin pour Fin CREATION</p>
2. Chercher la 1 ^{ère} note de la classe :	
Analyse	Algorithme
<p>Résultat =Minimum Minimum ← min Traitement : f = ouvrir(f) min =[lire(f, fic), min← fic.moy] Tantque(non(fin_fichier(f)))faire lire(f, fic) avec fic faire si (moy < min) alors min←moy finsi fin avec fin tantque</p>	<p>0. DEFFN MINIMUM (VAR F :femp) 1. Ouvrir(F) 2. Lire(f, fic) 3. Min← fic.moy 4. Tantque (non(fin_fichier(f))) faire lire(f, fic) avec fic faire si (moy < min) alors min←moy finsi fin avec fin tantque 5. Minimum ← min Fin MINIMUM</p>
3. Ajouter un élève à la fin du fichier :	
Analyse	Algorithme
<p>Résultat =f Traitement : f = ouvrir(f) fic = Avec fic faire id = données("id : ") np = données("np : ") date_naiss = données("date : ") sexe = données("sexe : ") nb = données("nb : ") moy=donnée(" moy") fin Avec f = [pointer(f, taille_fichier(f))] Ecrire(f, fic)</p>	<p>0. DEFPROC AJOUT(VAR F :femp) 1. Ouvrir(F) 2. Avec fic faire ecrire("id : ") ;lire(id) ecrire("date : ") ;lire(date_naiss) ecrire("np : ") ;lire(np) ecrire("sexe : ") ;lire(sexe) ecrire("moy : ") ;lire(moy) fin Avec 3. pointer(f, taille_fichier(f)) 4. Ecrire(f, fic) 5. Fin AJOUT</p>

Résumé Algorithmique

4. Supprimer un élève :	
Analyse	Algorithme
<p>Résultat =f</p> <p>Traitement</p> <p>f = ouvrir(f)</p> <p>ftemp = Assigner(ftemp,"c:\ftemp.dat ")</p> <p>ftemp = récréer(ftemp)</p> <p>id_sup = donnée(" Idenifiant élève a supprimer")</p> <p>ftemp = tantque (non (fin_fichier(f)))faire lire(f, fic)</p> <p> Avec fic faire</p> <p> si (id <> id_sup) alors</p> <p> ecrire(ftemp, fic)</p> <p> fin si</p> <p> fin tantque</p> <p>f = [Récréer(f), Ouvrir(ftemp)]</p> <p>tantque (non (fin_fichier(ftemp)))faire lire(ftemp, fic)</p> <p> ecrire(f, fic)</p> <p> fin tantque</p>	<p>0. DEFPROC SUPPRIMER(VAR F :femp)</p> <p>1. Ouvrir(F)</p> <p>2. Assigner(ftemp,"c:\ftemp.dat ")</p> <p>3. Recréer(ftemp)</p> <p>4. Ecrire("Identifiant élève à supprimer"), lire(id_sup)</p> <p>5. tantque (non (fin_fichier(f)))faire lire(f, fic)</p> <p> Avec fic faire</p> <p> si (id <> id_sup) alors</p> <p> ecrire(ftemp, fic)</p> <p> fin si</p> <p> fin tantque</p> <p>6. Récréer(f)</p> <p>7. Ouvrir(ftemp)</p> <p>8. tantque (non (fin_fichier(ftemp)))faire lire(ftemp, fic)</p> <p> ecrire(f, fic)</p> <p> fin tantque</p> <p>9. Fin SUPPRIMER</p>

5. Modifier un élève :	
Analyse	Algorithme
<p>Résultat =f</p> <p>Traitement :</p> <p>f = ouvrir(f)</p> <p>id_mod = donnée(" Idenifiant élève a modifier")</p> <p>f = tantque (non (fin_fichier(f)))faire lire(f, fic)</p> <p> Avec fic faire</p> <p> si (id = id_mod) alors</p> <p> id = données("id : ")</p> <p> np = données("np : ")</p> <p> date_naiss = données("date : ")</p> <p> sexe = données("sexe : ")</p> <p> moy=donnée("moy :")</p> <p> fin Avec</p> <p> Pointer(f, position_fichier(f)-1)</p> <p> Ecrire(f, fic)</p> <p> fin si</p> <p>fin tantque</p>	<p>0. DEFPROC MODIFIER(VAR F :femp)</p> <p>1. Ouvrir(F)</p> <p>2. Ecrire("Identifiant élève à modifier"), lire(id_mod)</p> <p>3. tantque (non (fin_fichier(f)))faire lire(f, fic)</p> <p> Avec fic faire</p> <p> si (id = id_mod) alors</p> <p> ecrire("id : ") ;lire(id)</p> <p> ecrire("date : ") ;lire(date_naiss)</p> <p> ecrire("np : ") ;lire(np)</p> <p> ecrire("sexe : ") ;lire(sexe)</p> <p> ecrire("moy : ") ;lire(moy)</p> <p> fin Avec</p> <p> Pointer(f, position_fichier(f)-1)</p> <p> Ecrire(f, fic)</p> <p> fin si</p> <p>fin tantque</p> <p>4. Fin MODIFIER</p>

Résumé Algorithmique

5. Trier la liste des élèves selon leur nom :	
Analyse	Algorithme
Résultat =f Traitement f = ouvrir(f) T,n = [n<0]tantque (non (fin_fichier(f)))faire lire(f, fic) n ← n+1 T[n] ← fic fin tantque T= Proc Tri_insertion(T,n) f =[récréer(f)] pour i de 1 à n faire ecrire (f,T[i]) fin pour	0. DEFPROC TRI (VAR F :femp) 1. Ouvrir(F) 2. [n<0]tantque (non (fin_fichier(f)))faire lire(f, fic) n ← n+1 T[n] ← fic fin tantque 3. Proc Tri_insertion(T,n) 4. Recréer(f) 5. pour i de 1 à n faire ecrire (f,T[i]) fin pour 6. Fin TRI

Trie d'un tableau d'enregistrement :	
Analyse	Algorithme
Résultat =T T= Pour i de 2 à n faire V ← T[i] j ← i Tantque(T[j-1].np>V.np) et (j>1) faire T[j] ← T[j-1] j ← j-1 FinTantque T[j] ← V Finpour Fin TRI_INSERTION	0. DEFPROC TRI_INSERTION (VAR T :TAB;N:entier) 1. Pour i de 2 à n faire V ← T[i] j ← i Tantque(T[j-1].np>V.np) et (j>1) faire T[j] ← T[j-1] j ← j-1 FinTantque T[j] ← V Fin pour 2. Fin TRI_insertion

5. copier un fichier de donnée dans un fichier texte :	
Analyse	Algorithme
Résultat =f,ft Traitement : f = ouvrir(f) ft = Assigner(ft,"c:\ftexte.txt ") ft = récréer(ft) ft = tantque (non (fin_fichier(f)))faire lire(f, fic) Avec fic faire Convch(moy, chmoy) fin Avec ligne ← id + " " + np+ " " + date_naiss ligne ← ligne + " " + sexe+ " " + chnb ligne ← ligne + " " + chmoy Ecrire_nl (ft,ligne) fin tantque	0. DEFPROC copier (VAR F :femp; var ft:text) 1. Ouvrir(F) 2. Assigner (ft,"c:\ftexte.txt ") 3. Recréer (ft) 4. tantque (non (fin_fichier(f)))faire lire(f, fic) Avec fic faire Convch(moy, chmoy) fin Avec ligne ← id + " " + np+ " " + date_naiss ligne ← ligne + " " + chmoy Ecrire_nl (ft,ligne) fin tantque 5. fin copier

Résumé Algorithmique

11 – Récurrence & Récursivité :

Dans ce chapitre, ce qu'il faut savoir :

- 1- Remplir un tableau (méthode récurrente ou récursive)
- 2- Remplir une matrice (méthode récurrente ou récursive).
- 3- Opérations sur les tableaux et les matrices (exp : minimum, maximum, somme, décalage, nombre d'occurrence, recherche, décalage à droite,)
- 4- Saisie et affichage récursif d'un fichier.
- 5- Traitements récurrents et récursifs sur les chaînes (exp : déterminer si deux chaînes

- sont anagrammes, tautogrammes.. ; compter le nombres d'occurrence d'une chaîne dans une autre, supprimer les espaces de début d'une chaîne...)
- 6- Traitements récurrents et récursifs sur les suites (exp : suite de Fibonacci, suite d'Ackerman, suite de MacCarty..).
 - 7- Transformer les traitements récurrents en traitements récursifs (exp : calcul puissance, suites,somme,...).

1. Remplissage d'un tableau :	
Version itérative	Version recursive
0. DEFPROC Remplir (VAR T :Tab;n:entier) 1. Pour i de 1 de à n faire lire(T[i]) fin pour 2. Fin Remplir	0. DEFPROC Remplir (VAR T :Tab;n,i:entier) 1. Si (i=n) alors lire(T[i]) sinon lire(T[i]) Proc remplir (t,n,i+1) finsi 2. Fin remplir

Condition d'arrêt

Appel récursif

- N.B :* Si vous voulez un remplissage à l'envers il suffit d'inverser les deux instructions dans la clause sinon :

```

sinon
  Proc remplir(t,n,i+1)
  T[i]=donnée(T[i ] )
finsi
  
```

* Si vous voulez un affichage d'un tableau il suffit de changer le nom de la procedure (exp : affiche()) et de remplacer l'instruction

T[i]=donnée("T["i,""] : ") par Ecrire(T[i])...

2. Remplissage d'une matrice :	
Version itérative	Version recursive
0. DEFPROC Remplir (VAR M :mat;nl,nc:entier) 1. Pour i de 1 de à nl faire Pour j de 1 à nc faire lire(M[i,j]) fin pour 2. Fin Remplir	0. DEFPROC Remplir (VAR T :Tab;n,i,j:entier) 1. Si (i=nl)et(j=nc) alors lire(M[i,j]) sinon si(j<=nc) lire(M[i,j]) Proc remplir (m,nl,nc,i,j+1) Sinon Proc remplir (m,nl,nc,i+1,1) finsi 2. Fin remplir

Condition d'arrêt

Appel récursif

Appel récursif

Résumé Algorithmique

Traitements récurrents et récursifs sur les chaînes :

N.B :

- lorsque vous voulez faire un traitement sur les chaînes, veuillez toujours à exploiter les procédures et les fonctions prédéfinies sur les chaînes (long(), pos(), effacer(),...)
- Contrairement à la manipulation des matrices et tableaux où on n'est obligé à ajouter les compteurs dans l'entête de la procédure ou la fonction, dans les chaînes on les ajoute pas mais on utilisera soit la fonction copy() ou la procédure effacer().

(exp1 : Eliminer les espaces au début de la chaîne...)

Version itérative	Version recursive
<pre> 0. DEFFN eliminer(ch:chaîne):chaîne 1. Tantque pos(" ",ch)<>0 faire Effece(ch,1,1) fin tantque 2. eliminer←ch 3. Fin eliminer </pre> <p style="text-align: center;"><i>Condition d'arrêt</i></p>	<pre> 0. DEFFN eliminer(ch:chaîne):chaîne 1. Si pos(" ",ch)=0 alors eliminer←ch sinon Effece(ch,1,1) Eliminer←eliminer(ch) finsi 2. Fin eliminer </pre> <p style="text-align: center;"><i>Appel récursif</i></p>

(exp2 : Déterminer si deux chaînes sont anagrammes...)
Deux chaînes ch1, ch2 sont dites anagrammes, si les lettres qui composent la 1^{ère} chaîne existent tous dans la deuxième chaîne. (exp : ch1=chien & ch2=niche).

Version itérative	Version recursive
<pre> 0. DEFFN anagramme(ch1,ch2:chaîne):chaîne 1. [ok←vrai]Tantque ch1<>"" faire Si pos(ch1[1],ch2)=0 alors Ok← faux sinon Effece(ch1,1,1) fin tantque 2. anagramme←ok 3. Fin anagramme </pre> <p style="text-align: center;"><i>Condition d'arrêt</i></p>	<pre> 0. DEFPROC anagramme(ch1,ch2:chaîne):chaîne 1. Si ch1=""alors anagramme←vrai sinon si (pos(ch1[1],ch2)=0)alors anagramme← faux sinon efface(ch1,1,1) anagramme←anagramme(ch1,ch2) finsi 2. Fin anagramme </pre> <p style="text-align: center;"><i>Appel récursif</i></p>

N.B :* Si vous utiliser la boucle tantque il faut inverser la condition(ch1<>""→ ch1="")

* Si vous utiliser la boucle Répéter ... jusqu'à la condition reste la même(ch1=""→ ch1="")

Résumé Algorithmique

Traitements récurrents et récursifs sur les suites :

(*exp1 : fonction puissance*)

$$P = \underbrace{x * x * \dots * x}_{n \text{ fois}}$$

Version itérative	Version recursive
0. DEFFN puissance (x,n:entier):entier 1. [p←1]pour i de 1 à n faire P←p* x fin pour 2. puissance←p 3. Fin puissance <p style="text-align: center;"><i>Condition d'arrêt</i></p>	0. DEFPROC puissance (x,n,i:entier):entier 1. Si j=n alors puissance←x sinon puissance ← x * puissance (x,n,i+1) fin si 2. Fin anagram <p style="text-align: center;"><i>Appel récursif</i></p>

Solution récursive :

(*exp2 : calcul de puissance*)

$$P = \underbrace{a * a * \dots * a}_{n \text{ fois}} * \underbrace{b * b * \dots * b}_{m \text{ fois}}$$

Version itérative	Version recursive
4. DEFFN puissance (x,n:entier):entier 5. [p←1]pour i de 1 à n+m faire si (i≤n) alors P←p* a Sinon p ← p* b fin pour 6. puissance←p 7. Fin puissance <p style="text-align: center;"><i>Condition d'arrêt</i></p>	3. DEFPROC puissance (x,n,m,i:entier):entier 4. Si j=n+m alors puissance←b sinon si (i≤n) alors puissance ← a * puissance (x,n,m,i+1) sinon puissance ← b * puissance (x,n,m,i+1) fin si 5. Fin anagram <p style="text-align: center;"><i>Appel récursif</i></p>

Résumé Algorithmique

III- Conversion entre les base :

Dans ce chapitre, ce qu'il faut savoir :

(Base)10 $\xrightarrow{\text{Division successive}}$ Autre base

Autre base $\xrightarrow{\text{Multiplication successive}}$ (Base)10

- Exemple ((Base)10 à la (base)2)

N.B :

- lorsque vous voulez changer la conversion de la base 10 à une autre base différente de 2, il suffit de remplacer 2 par le numéro de base (exp :8,9,10...16)
- Concernant la conversion vers la base 16, il faut tenir compte du reste qui dépasse 9, pour cela il suffit d'écrire :

```
0- DEFFN conv_dec_b(dec,b :entier) :chaine
1- [ch ←'']
   Répéter
     R ← dec mod 16
     si ( R dans [0..9]) alors
       convch(R,ch_R)

   sinon
     ch ← chr(R+55)
   fin si
   ch ← ch_R + ch
   dec ← dec div 2
   jusqu'à (dec=0)
2- conv_dec_b ←ch
3- fin conv_dec_b
```

- Exemple ((Base)2 à la (base)10)

N.B :

- lorsque vous voulez changer la conversion à partir d'une autre base à la base 10,il suffit de remplacer la fonction puissance selon le numéro de base (exp :8,9,10...16)
- Concernant la conversion vers la base 16, il faut tenir compte des nombres qui dépasse le chiffre 9 , pour cela il faut 'écrire :

```
0- DEFFN conv_b_dec(ch,b :entier) :chaine
1- [dec ←0]
   Pour i de 1 à long(ch) faire
     Si(ch[i] dans ['0'..'9'])alors
       Valeur(ch[i],v,e)
     Sinon
       V ← ord(ch[i])-55
     Fin si
     Dec ← dec +v*FN puiss(b,long(ch)-i)
   Fin pour
2- conv_b_dec ←dec
3- fin conv_b_dec
```