



## **PROBLEME** ( 14 Pts )

On se propose d'écrire un programme qui permet de corriger les fautes d'orthographe d'un fichier texte «C:\bac2009\docum.txt » puis de l'afficher. Ce fichier renferme plusieurs lignes dont chacune renferme une phrase ; sachant que chaque phrase est une suite des mots et deux mots successifs sont séparés par un seul espace.

La correction se fait en se referant à un fichier dictionnaire suivant «C:\bac2009\dico.dic » qui renferme des mots de **25** caractères au maximum.

### **Le processus de correction se déroule comme suit :**

On consulte le fichier texte à corriger puis on vérifie l'existence de chacun de ses mots dans le dictionnaire. Si un mot n'existe pas alors un menu sera affiché présentant les choix suivants :

- 1- Ignorer l'erreur :** dans ce cas le mot sera conservé tel qu'il est
- 2- Ajouter au dictionnaire :** dans ce cas le mot sera ajouté à la fin du dictionnaire.
- 3- Suggestion :** le programme détermine et affiche une liste de **5 mots au maximum** qui ont un degré de ressemblance au mot à corriger  $>$  à 75%. si la liste de suggestion est vide alors le programme affichera le message « **aucune suggestion** » sinon l'utilisateur donne l'indice du mot à utiliser à partir de la liste des mots suggérés pour corriger le fichier texte.

**On définit le degré de ressemblance entre deux mots par :**

$$DR = (NCCBP^2 + NCCMP) / (L1 * L2) * 100$$

Avec :

NCCBP : nombre de caractères communs et bien placés

NCCMP : nombre de caractères communs et mal placés

L1, L2 : sont respectivement les longueurs du mot1 et du mot2

### **Exemples :**

<b>Exemple 1:</b> Mot1='Ecol' Mot2='Ecole' NCCBP=4 NCCMP=0 L1=4 L2=5 DR=(4 <sup>2</sup> +0)/(4*5)*100=16/20*100=80 %	<b>Exemple 2 :</b> Mot1='abcda' Mot2='acbdx' NCCBP=2 NCCMP=2 L1=6 L2=5 DR=(2 <sup>2</sup> +2)/(6*5)*100=6/30*100=20 %
---	--

**N.B. : On ne peut pas modifier une ligne dans un fichier texte ; on peut utiliser un fichier texte intermédiaire.**

### **Travail demandé :**

- 1- Donner les structures des données nécessaires pour résoudre ce problème.
- 2- Analyser le problème en le décomposant en modules.
- 3- Analyser les différents modules envisagés

# Correction devoir de synthèse N°3

## Problème

### I. Structure de données (1 point)

Pour résoudre ce problème on a besoin de :

- Un fichier texte pour traiter le fichier à corriger
- Un fichier typé pour le fichier dictionnaire

### II. Programme principal {2 points : analyse et TDO}

#### Analyse

	Nom : DevSynthes3	
S	LDE	OU
4	Résultat=proc affiche (FichText <sub>f</sub> )	FichText <sub>f</sub>
3	FichText <sub>f</sub> = proc corriger(fichtext <sub>i</sub> , Fdic)	Fichtext <sub>i</sub>
2	Fichtext <sub>i</sub> = associer (fichtext <sub>i</sub> , "C:\bac2009\docum.txt")	Fdic
1	FDic= associer(Fdic, " C:\bac2009\dico.dic")	
5	Fin DevSynthese1	
Objets	Type/Nature	Rôle
Fichtext <sub>(f,i)</sub>	Texte	Fichier texte à corriger (i et f représente l'état du fichier avant et après la correction.
Fdic	TFdic	Fichier contenant les mots du dictionnaire
affiche	procédure	Affiche le contenu du fichier s'il a subi une correction
Corriger	procédure	Corrige les fautes d'orthographe du fichier texte.

#### TDNT

Type
TFDic= fichier de chaine[26]

### III. modules {11 points}

#### 1. Analyse procédure affiche {1 points}

	Def Proc affiche (var f :texte )	
S	LDE	OU
3	Résultat= affichage	ligne
2	Affichage= [ Ouvrir (f)]	
1	Tant Que non finfichier(f) faire	
4	Lire_nl(f,ligne)	
	Ecrire_nl(ligne)	
	Fin tant que	
	Fermer(f)	
	Fin si	
	Fin affichage	
Objets	Type/Nature	Rôle
ligne	Chaîne	Contient une ligne du fichier texte

#### 2. Analyse de la Pprocédure corriger {1.5 point}

	Def proc corriger(var f :texte ;var fdic :)	
S	LDE	OU
1	Résultat= f	Corrigeligne ftemp
	F= [ associer(ftemp,c:\bac2009\temp.txt') ;	
	réécrire (ftemp)]	
	Tantque non finfichier(f) faire	
	Lire(f,ligne)	
	Ligne←Fn corrigeligne(ligne,fdic)	
	Ecrire(ftemp, ligne)	
	Fin tantque	
	Fermer (ftemp)	
	Efface(f)	
	Renommer (ftemp, ,"C:\bac2009\docum.txt")	
	Fin si	
2	Fin corriger	
objets	Type/Nature	Rôle
corrigeligne	fonction	Corrige une ligne
Ftemp	texte	Fichier temporaire

### 3. Analyse de la fonction corrigeligne (2,5 points)

	Def Fn corrigeligne(ch :chaine var fdic :TFDic ) :chaine	
S	LDE	OU
2	Résultat=CorrigeLigne←ChCor	Mot
1	<pre> ChCor=[chcor←" ;Insere(' ',ch,long(ch)+1)] P←pos(' ',ch) Tantque p&lt;&gt; faire     Mot← SousChaine(ch,1,p-1)     Si non Fn existe (mot, fdic) alors         Ecrire ('1- Ignorer')         Ecrire ('2- Ajouter au dictionnaire')         Ecrire ('3- Suggestion')         Répéter             Choix ← liretouche { ou choix=donnée             Jusqu'à choix dans ['1'..'3']         Selon choix faire             '2' : Proc ajouter (mot, fdic)             '3' : Proc suggerer (mot, listemots, n, fdic)                 P← i-long(mot)                 Si n = 0 alors                     Ecrire ('pas de sugestion')                 Sinon                     Pour j de 1 à n faire                         Ecrire( listemots[j])                     Fin pour                     Répéter                         Lire(j)                     Jusqu'à j dans [0..n]                     Si i&gt;0 alors                         Mot :=listemots[j]                     Fin si                 Fin si             Fin selon         Fin Si         chcor←chcor+mot         efface(ch,1,p)         p ←pos(' ',ch)     fin tantque </pre>	Choix J n listemots
3	Fin corrigeligne	
Objets	Type/Nature	Rôle
j	entier	compteur
Mot	chaine	Mot à corriger
Choix	caractère	Choix de l'utilisateur
n	entier	Nombre de mot à suggérer
listemots	Tab2	Liste des mots à suggérer
Type	Tab2= tableau de 5 chaine [26]	

### 4. Analyse procédure ajouter {1 point}

	Def proc ajouter (mot :chaine ; var fdic : tfdic)	
--	---	--

S	LDE	OU
2	Résultat=fermer(fdic)	
1	fdic=[ouvrir (fdic) ] pointer(fdic,tailleFichier(fdic)) ecrire(fdic,mot)	
3	Fin ajouter	

### 5. Analyse procédure suggerer {2 point}

	Def proc suggerer (mot :chaine ; var liste :tab2 ; var n : entier ;var fdic : tfdic)	
S	LDE	OU
1	Résultat=Liste,n Liste,n=[ouvrir (fdic) ;n←0] Tantque non FinFichier(fdic) et( n <5) faire Lire(fdic,m) Si Fn DegRes(mot,m) >75 alors n←n+1 Liste[n]← m Fin si Fin tantque	m DegRes
1	Fin suggerer	
Objets	Type/Nature	Rôle
m	chaine	Le mot lu du dictionnaire
DegRes	Fonction de type entier	Calcule le degré de ressemblance entre deux mots

### 6. Analyse Fonction Existe (1 point)

	Def Fn existe (mot :chaine var f : tfdic)	
S	LDE	OU
2	Résultat=Existe ← trouver	m
1	M=[ouvrir (fdic) ;trouver ←faux] Tantque non FinFichier(fdic) et( non trouver) faire Lire(fdic,m) Trouver←mot=m Fin tantque	trouver
3	Fin suggerer	
Objets	Type/Nature	Rôle
m	chaine	Le mot lu du dictionnaire
trouver	boolean	Le mot est trouvé ou non dans le dictionnaire

7.

**Analyse fonction DegRes {2 points}**

	Def fn DegRes(m1,m2 :chaine) :entier	
S	LDE	OU
4	Résultat= DegRes←( NCCBP <sup>2</sup> +NCCMP)/(L1*L2)*100	NBCCBP
1	L1,L2=[L1←long(m1) ;l2←long(m2)] Si l1>l2 alors m← m1 m1←m2 m2←m l←l1 l1←l2 l2← l Fin si	NCCMP L1 L2 L m
2	NCCBP=[NCCBP←0] Pour i de 1 à l1 faire Si majus(m1[i])=majus(m2[i]) alors NCCBP←NCCBP+1 M1[i]←' ' M2[i]←' ' Fin si Fin pour	
3	NCCMP=[ NCCMP←0 pour i de 1 à l2 faire m2[i]← Majus(m2[i]) fin pour]	
5	Pour i de 1 à l1 faire P←pos(majus(m1[i]),m2) Si (p<> 0) et m1[i]<>' ' alors NCCMP← NCCMP+1 M2[p]←' ' Fin si Fin Pour Fin DegRes	
Objets	Type/Nature	Rôle
NBCCBP	entier	Nombre de caractères communs et bien placés
NCCMP	entier	Nombre de caractères communs et mal placés
L1	entier	Longueur du mot le plus court
L2	entier	Longueur du mot le plus long
m	chaine	Variable intermédiaire
l	entier	Variable intermédiaire