Exercices sur les fichiers

Auteur: E. Thirion - Dernière mise à jour: 22/2/2019

Les exercices suivants sont en majorité des projets à complèter. L'interface graphique de ces projets est déjà réalisée ce qui vous permettra un gain de temps important. Ces projets sont disponibles par téléchargement. Pour savoir comment y accéder cliquez ici.

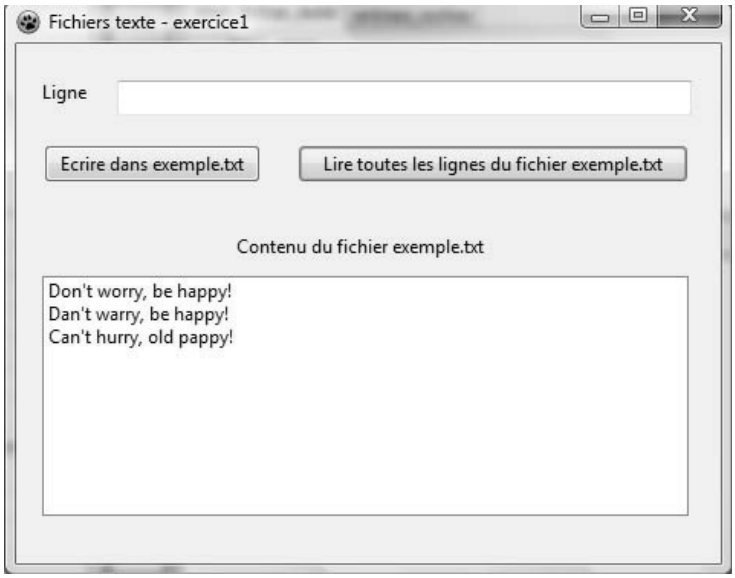
D'autre part, ce document fait partie d'un ensemble de cours du même auteur (programmation procédurale et objet, programmation web, bases de données) auxquels vous pouvez accédez en cliquant ici.

Fichiers Texte

Exercice 1

Objectif: réaliser un programme permettant de créer un fichier texte de une ligne, puis de lire son contenu.

Projet: Exo-Fichiers/Pascal/FichTextExo1Pascal/projet_exo1_fichier_texte.lpi

Le formulaire	Nom des contrôles
	TxtLigne ZlFichier

Variables globales

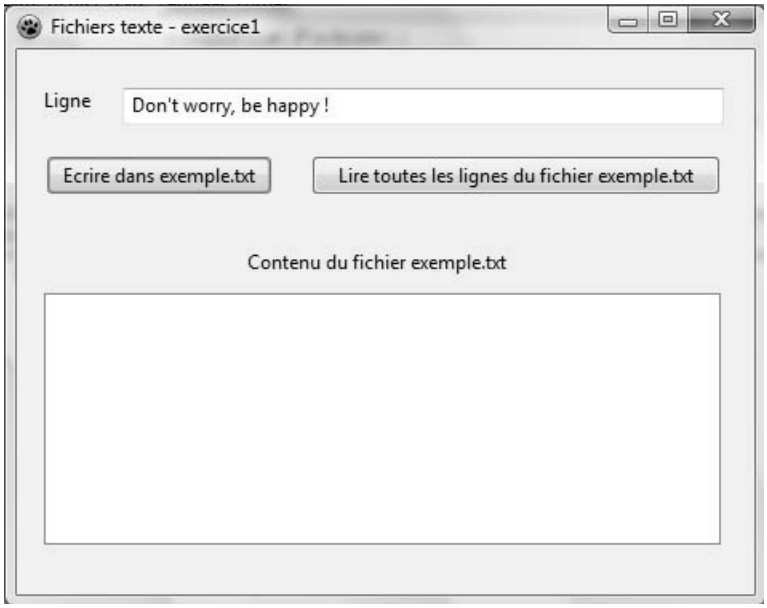
Les variables globales suivantes sont déclarées:

```
var
  Form1: TForm1;
  Ligne: String;
  Fichier : TextFile;
```

Pour toutes les questions de cet exercices, vous devez utiliser uniquement les variables **Ligne** et **Fichier**. Ne déclarez aucune autre variable !

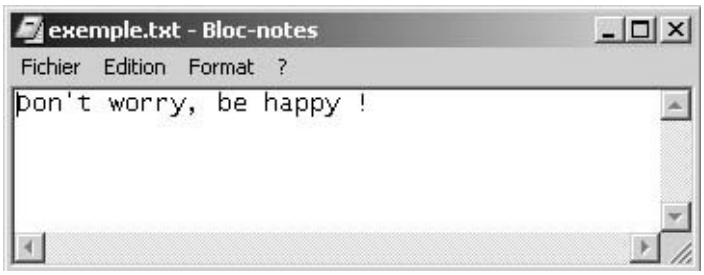
Question1 : Création de fichier et écriture

Ecrire le code associé au bouton intitulé **"Ecrire dans Exemple.txt"**: lorsque l'utilisateur appuie sur ce bouton, le programme créé un fichier nommé **exemple.txt** dans le répertoire courant de l'application et y écrit la ligne contenue dans la zone de texte:



Tests

Vérifiez que le répertoire courant de l'application contient bien un fichier nommé **exemple.txt**. Ouvrez ensuite ce fichier avec un éditeur de texte (par exemple avec le Bloc Note) et vérifiez son contenu:

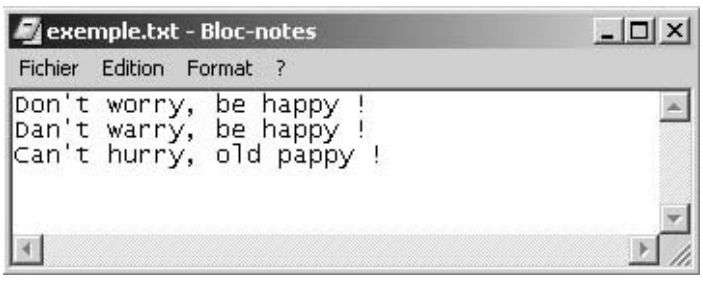


Question2 : Lecture et affichage

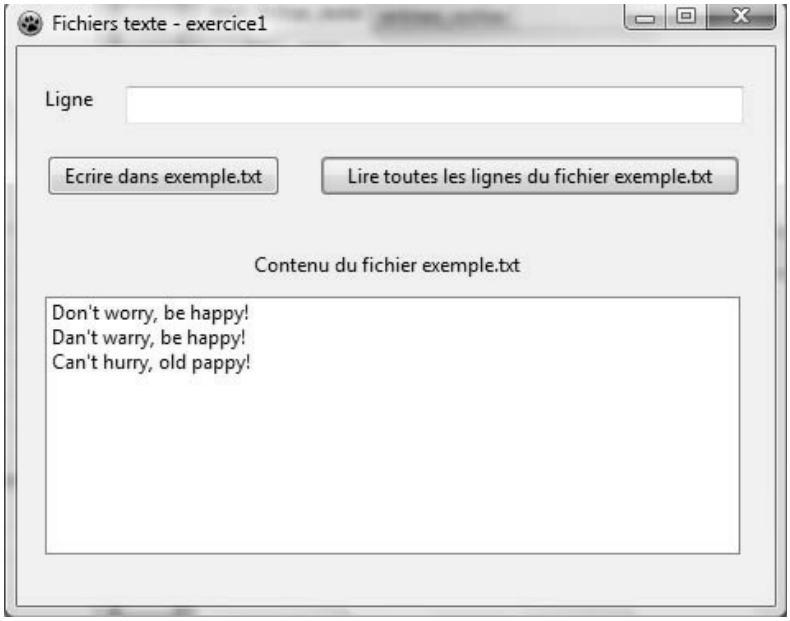
Ecrire le code associé au bouton **"Lire toutes les lignes du fichier exemple.txt"**: lorsque l'utilisateur appuie sur ce bouton, le programme doit lire chaque ligne du fichier **exemple.txt** et l'afficher dans la zone de liste.

Tests

Pour tester votre programme, modifiez le contenu du fichier **exemple.txt** en y rajoutant des lignes:



Puis vérifiez que votre programme les affiche correctement:



Question 3 : Protection contre les erreurs

Il n'est pas facile de protéger un programme contre les erreurs d'accès aux fichiers en utilisant directement les procédures de Pascal. Pour cette question, je vous demanderais donc d'utiliser les sous-programmes suivants de la librairie **entrees_sorties.pas** :

- **function** OuvrirEnEcriture (**var** f: **TextFile** ; n: **String**): **boolean**;

  Assigne la variable fichier **f** au nom de fichier **n** et ouvre **f** en écriture Si cela provoque une erreur, la fonction retourne **false**. Si tout se passe bien, elle retourne la valeur **true**.

- **procedure** EcrireUneLigne (**var** f: **TextFile**; l: **String**);
  Ecrit la ligne **l** dans le fichier **f**. Si cette opération provoque une erreur le message suivant est affiché

  Une erreur s'est produite lors de l'appel de la fonction EcrireUneLigne

- **function** OuvrirEnLecture (**var** f: **TextFile** ; n :**String**): **boolean**;

  Assigne la variable fichier **f** au nom de fichier **n** et ouvre **f** en lecture Si cela provoque une erreur, la fonction retourne **false**. Si tout se passe bien, elle retourne la valeur **true**.

- **function** LireUneLigne (**var** f: **TextFile**) : **String**;

  Retourne la ligne courante du fichier **f**. Si la lecture propoque une erreur, le message suivant est affiché:

  Une erreur s'est produite lors de l'appel de la fonction LireUneLigne

Question 3-A : Protection contre les erreurs d'écriture

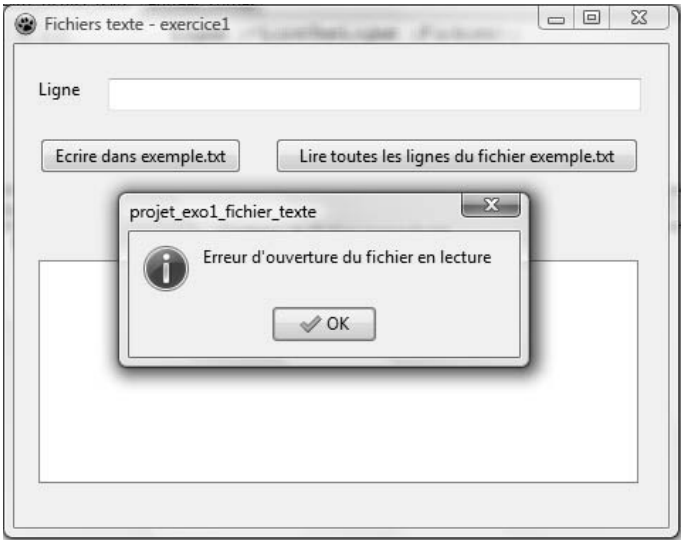
Réécrivez le code de la procédure évènementielle associée au bouton **Ecrire dans exemple.txt** en utilisant les sous-programmes de **entrees_sorties.pas**, de manière à afficher un message d'erreur s'il se produit une erreur lors de cette opération:

Test

Mettez le fichier **exemple.txt** en lecture seule et vérifiez que votre programme affiche un message d'erreur lorsque vous essayez d'écrire dans ce fichier.

Question 3-B : Protection contre les erreurs de lecture

Réécrivez le code de la procédure évènementielle associée au bouton **Lire toutes les lignes ..** en utilisant les sous-programmes de **entrees_sorties.pa**s, de manière à afficher un message d'erreur s'il se produit une erreur lors de cette opération:



Test

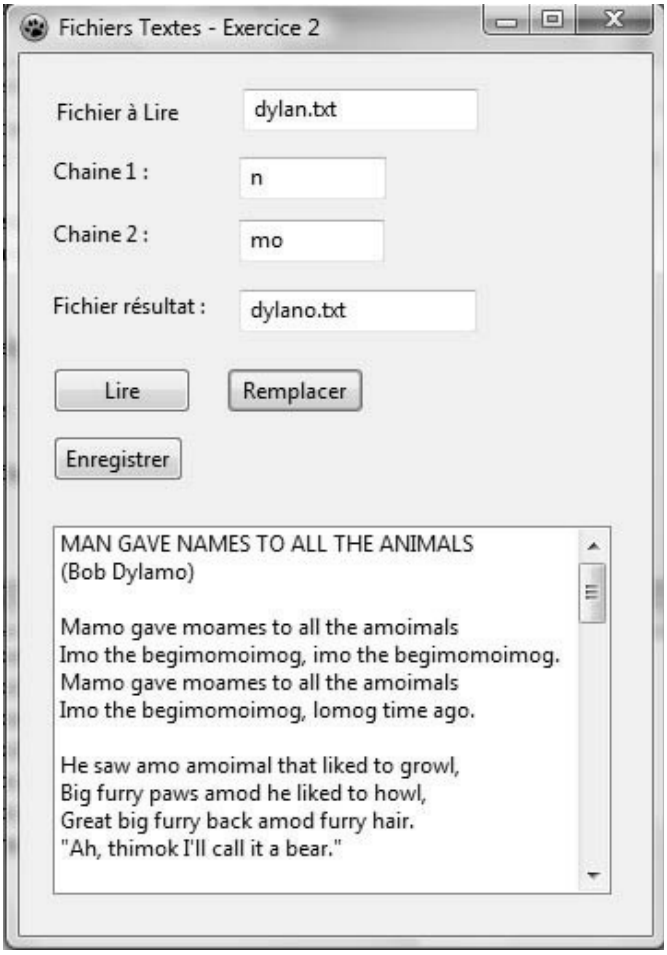
Modifiez le nom du fichier (ou supprimez le) et vérifiez que votre programme affiche un message d'erreur lorsque vous essayez de lire le fichier **exemple.txt**.

Exercice 2

Objectif

Dans cet exercice, il s'agit de réaliser un programme qui modifie un fichier texte en remplacant toutes les occurrences d'une certaine chaine de caractères (chaine1) par une autre (chaine2). Le résultat de la substitution peut être enregistré dans un fichier dont le nom est donné par l'utilisateur.

Projet: Exo-Fichiers/Pascal/FichTextExo2Pascal/projet_fichier_texte_exo2.lpi

Le formulaire	Nom des contrôles
	ZtFichL TxtChaine1 TxtChaine2 TxtFichRes LstFichier

Variables globales imposées

Lors de la lecture d'un fichier, chaque ligne est transférée dans un tableau de chaine de caractères nommé **Buffer** et le nombre de lignes lues est mémorisé dans un entier nommé **Nligne**.

Voici la déclaration de ces deux variables globales:

```
const
  TAILLE_BUFFER = 1000;
var
  Form1: TForm1;
  NLigne : integer;
  Buffer: array [1..TAILLE_BUFFER] of String;
```

Question1 : Lecture du fichier

Vous constaterez que la procédure du bouton **Lire** contient le code suivant:

```
procedure TForm1.BtLireClick(Sender: TObject);
var nf : String;
begin

  Lire (nf, ZtFichL);
  if LireFichier(nf) then
    AfficherBuffer(LstFichier)
  else
   ShowMessage('Erreur de lecture du fichier '+nf);

end;
```

La fonction **LireFichier** et la procédure **AfficherBuffer** sont incomplètes.

La fonction **LireFichier** doit faire la chose suivante:

- Transférer le contenu du fichier texte à lire dans le tableau **Buffer**.
- Si la lecture provoque une erreur, la fonction retourne la valeur **False**, sinon elle retourne la valeur **True**.

La procédure **AfficherBuffer(**zl:**TListBox)** affiche le contenu du tableau **Buffer** dans la zone de liste **zl**.

Complèter la fonction **LireFichier** et la procédure **AfficherBuffer** de manière à ce qu'un clic sur le bouton **Lire** provoque le transfert du fichier dans le tableau **Buffer** puis son affichage dans la zone de liste.

Pour faire des essais vous pouvez utiliser le fichier **dylan.txt** (chanson de Bob Dylan) qui se trouve dans le répertoire du projet.

Question2: Remplacement de chaine de caractères

La procédure évenementielle associée au bouton **Remplacer** contient le code suivant:

```
procedure TForm1.BtRemplacerClick(Sender: TObject);
var chaine1, chaine2 : String;
begin
  Lire (chaine1, TxtChaine1);
  Lire (chaine2, TxtChaine2);
  Remplacer (chaine1, chaine2);
  AfficherBuffer (LstFichier);
end;
```

La procédure **Remplacer (c1:String;c2:String)**, que vous devez écrire, doit remplacer la chaine **c1**, par la chaine **c2** dans tout le tableau **Buffer**. Par exemple, **Remplacer('a', 'e')** va remplacer tous les **'a'** par des **'e'** dans tous le tableau **Buffer**.

On vous demande donc de complèter la procédure **Remplacer** de manière à ce qu'un clic sur le bouton **Remplacer** provoque le remplacement de chaque occurrence de la chaine 1 par la chaine 2 dans le tableau **Buffer**, puis son affichage dans la zone de liste.

Pour cela vous pouvez utiliser la fonction **AnsiReplaceStr(**c,c1,c2**)** de Pascal. Elle remplace toutes les occurrences de la chaine **c1** par la chaine **c2** dans la chaine **c**.

Par exemple, si **c** contient **'Baratin'**, après **AnsiReplaceStr(**c, 'a', 'ata'**)** , **c** contiendra **'Bataratatin'**.

Question3: Enregistrement du fichier

Complètez la procédure associée au bouton **Enregistrer** de manière à ce qu'un clic sur ce bouton provoque l'enregistrement du tableau **Buffer** dans le fichier résultat.

Fichiers Structurés

Partie commune aux exercices 1 et 2

Le thème général des exercices 1 et 2 est la gestion d'horaires de vol.

Un horaire de vol est défini par une ville de départ, une ville d'arrivée, une heure de départ et une heure d'arrivée.

On représentera donc un horaire par un type structuré et les horaires seront mémorisés dans un fichier structuré.

Voici la déclaration du type **Horaire**:

```
Horaire =
   record
     VilleDepart: String[20];
     VilleArrivee: String[20];
     HeureDepart: integer;
     HeureArrivee: integer;
   end;
```

Dans les deux exercices on traitera des fichiers structurés composés d'enregistrements de type **Horaire**. Un fichier **f** de ce type est donc déclaré de la manière suivante:

```
Var f : file of Horaire;
```

Exercice 1

Ouvrir le projet: Exo-Fichiers/Pascal/FichStructExo1Pascal/ProjetFStruct1.lpi

Le formulaire de ce projet doit permettre d'ajouter des horaires de vol dans le fichier **Horaire.dat** (bouton **Enregistrer**) ou bien d'afficher les vol contenus dans ce fichier (bouton **Lire**).

Le formulaire	Nom des contrôles
	ZtVD ZtVA ZtHD ZtHA ZtNE

Variables globales imposées

Les deux variables globales suivantes sont déclarées dans le projet:

```
f : file of Horaire;
NumEnr : integer;
```

La variable **NumEnr** contiendra le numéro de l'enregistrement courant, c'est à dire de celui qui vient d'être lu du fichier ou écrit dans le fichier par le programme.

Question 1 : Ouverture et fermeture du fichier

Dès que l'on lance le programme, le fichier **Horaire.dat** doit être ouvert et assigné à la variable **f**. Dès qu'on le quitte, il doit être fermé.

L'instruction d'ouverture doit donc figurer dans **TForm1.FormCreate** et l'instruction de fermeture dans **TForm1.FormClose**.

Vérifiez qu'après avoir lancé le programme, le fichier Horaire.dat a bien été créé.

Question 2 : Protection contre les erreurs d'ouverture

Pour protéger votre programme contre les erreurs d'ouverture, vous pouvez utiliser la fonction suivante:

```
function Ouvrir (var f:file of Horaire; n : String): boolean;
```

Cette fonction permet d'ouvrir un fichier de nom **n** et de l'assigner à la variable **f.** S'il y a une erreur

d'ouverture, elle retourne **False**. Si tout se passe bien, elle retourne **True**.

Réécrivez le code d'ouverture du fichier de manière à afficher un message d'erreur si le fichier **Horaire.dat** n'a pas pu être ouvert correctement.

Pour effectuer des tests, vous pouvez pas exemple provoquer une erreur d'ouverture en faisant précéder le nom du fichier par un chemin inexistant ou bien le mettre en lecture seule.

Question 3 : Enregistrement des horaires

L'enregistrement des horaires se fait de la manière suivante: l'utilisateur lance le programme, saisi un horaire de vol, clique sur **Enregistrer**, saisi l'horaire de vol suivant, clique sur **Enregistrer** et ainsi de suite ...

A chaque enregistrement, le numéro de l'enregistrement courant (contenu dans **NumEnr**) est incrémenté et affiché.

Ecrivez le code permettant cela dans la procédure évènementielle associée au bouton **Enregistrer**.

Si vous avez des erreurs d'exécution, vous pouvez utiliser la procédure suivante pour écrire un enregistrement:

```
procedure EcrireUnEnregistrement (var f: file of Horaire ; h : horaire);
```

Cette procédure a le même effet que **write**, mais elle vous affichera un message d'erreur en cas de problème d'écriture.

Question 4 : Affichage des horaires

Pour afficher le contenu du fichier **Horaire.dat**, l'utilisateur lance le programme puis clique sur le bouton **Lire**. Il voit alors apparaître le premier enregistrement, ou autrement dit le premier horaire. S'il clique de nouveau sur le même bouton, il verra apparaitre le deuxième enregistrement, etc...

A chaque lecture d'enregistrement, le numéro de l'enregistrement courant (contenu dans **NumEnr**) est incrémenté et affiché.

Ecrivez le code permettant ceci dans la procédure évènementielle associée au bouton **Lire**.

Attention: il n'est pas possible de lire au-delà de la fin du fichier. Lorsque la fin du fichier est atteinte, avertissez l'utilisateur en affichant le message 'Fin de fichier atteinte'.

Si vous avez des erreurs d'exécution, vous pouvez utiliser la procédure suivante pour lire un enregistrement:

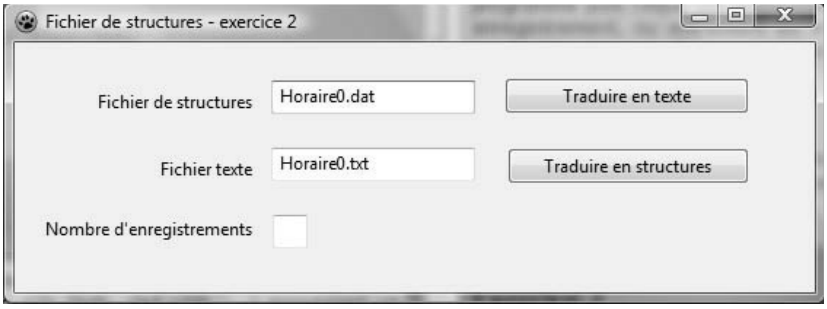
```
function LireUnEnregistrement (var f: file of Horaire): horaire;
```

Cette fonction retourne l'enregistrement lu et affiche un message d'erreur en cas de problème.

Exercice 2

Projet: Exo-Fichiers/Pascal/FichStructExo2Pascal/ProjetFichStructExo2.lpi

Objectif: Ce projet doit permettre de traduire un fichier structuré constitués d'enregistrements de type **Horaire** en un fichier texte et inversement.

Le formulaire	Nom des contrôles
	ZtStruct ZtText ZtNE

Format du fichier texte

Dans le fichier texte, chaque horaire est représenté par un groupe de quatre lignes:

- la première ligne commence par **VILLE-DEPART:** suivi par le nom de la ville de départ.
- la deuxième par **HEURE-DEPART:** suivi par l'heure de départ.
- **la troisème par VILLE-ARRIVEE:** suivi par le nom de la ville d'arrivée.
- la quatrième par **HEURE-ARRIVEE:** suivi par l'heure d'arrivée.

Exemple

Le fichier **Horaire0.dat** que vous trouverez dans le répertoire de l'application contient les trois horaires suivants:

1. Strasbourg-Berlin, 8h-10h
2. Berlin-Amsterdam, 12h-14h
3. Amsterdam-Oslo, 17h-18h

La traduction de ce fichier en un fichier texte, devrait vous donner le résultat suivant:

```
VILLE-DEPART:Strasbourg
HEURE-DEPART: 8
VILLE-ARRIVEE:Berlin
HEURE-ARRIVEE: 10
VILLE-DEPART:Berlin
HEURE-DEPART: 12
VILLE-ARRIVEE:Amsterdam
HEURE-ARRIVEE: 14
VILLE-DEPART:Amsterdam
HEURE-DEPART: 17
VILLE-ARRIVEE:Oslo
HEURE-ARRIVEE: 18
```

Question1 : traduction du fichier structuré en fichier texte

Un clic sur le bouton intitulé **"Traduire en Texte"** doit traduire le fichier structuré (dont le nom est contenu dans la zone de texte **ZtStruct**) en un fichier texte (dont le nom est contenu dans la zone de texte **ZtText**) respectant le format défini au paragraphe précédent.

Le nombre d'enregistrements lus est affiché dans la zone de texte **ZtNE**.

Pour réaliser cette traduction utilisez la procédure **EcrireUneLigne** de **ETBib** (voir Exercie 1 sur les fichiers texte Pascal) ainsi que la fonction **LireUnEnregistrement** (voir exercice précédent).

Attention: pensez à protéger votre programme contre les erreurs d'ouverture de fichier.

Test

Pour tester votre programme utilisez **Horaire0.dat** comme fichier structurés et **Horaire0.txt** comme fichier texte, puis vérifiez le contenu de **Horaire0.txt** avec un éditeur de texte quleconque.

Question2 : traduction du fichier texte en fichier structuré

Un clic sur le bouton intitulé **"Traduire en structures"** doit traduire fichier texte (dont le nom est contenu dans la zone de texte **ZtText**) en un fichier structuré (dont le nom est contenu dans la zone de texte **ZtStruct**).

Le nombre d'enregistrements écrits est affiché dans la zone de texte **ZtNE**.

Pour réaliser cette traduction utilisez:

- la procédure **LireUneLigne**
- la fonction **EcrireUnEnregistrement**
- les quatres fonctions **VilleDepart**, **VilleArrivee**, **HeureDepart**, **HeureArrivee** figurant dans le fichier source du projet. Ces fonctions vous permettent respectivement d'extraire la ville de départ, la ville d'arrivée, l'heure de départ et l'heure d'arrivée à partir d'une ligne **L** du fichier texte. Elles ne fonctionnent correctement que si la ligne **L** commence respectivement par **VILLE-DEPART:**, **VILLE-ARRIVEE:**, **HEURE-DEPART:**, **HEURE-ARRIVEE:**.

Attention: pensez à protéger votre programme contre les erreurs d'ouverture de fichier.

Test

Traduisez le fichier **Horaire0.txt** obtenu à la question précédente en **Horaire1.dat**. Pour vérifiez que le contenu de ce fichier est correcte, traduisez le en un fichier texte **Horaire1.txt**, puis comparez avec **Horaire0.txt.**