Exercice 11.1

Créer sur disquette puis afficher à l'écran le fichier INFORM.TXT dont les informations sont structurées de la manière suivante:

Numéro de matricule (entier)

Nom (chaîne de caractères)

Prénom (chaîne de caractères)

Le nombre d'enregistrements à créer est à entrer au clavier par l'utilisateur.

Exercice 11.2

Ecrire un programme qui crée sur disquette un fichier INFBIS.TXT qui est la copie exacte (enregistrement par enregistrement) du fichier INFORM.TXT.

Exercice 11.3

Ajouter un nouvel enregistrement (entré au clavier) à la fin de INFORM.TXT et sauver le nouveau fichier sous le nom INFBIS.TXT.

Exercice 11.4

Insérer un nouvel enregistrement dans INFORM.TXT en supposant que le fichier est trié relativement à la rubrique NOM et sauver le nouveau fichier sous le nom INFBIS.TXT.

Exercice 11.5

Supprimer dans INFORM.TXT tous les enregistrements:

a) dont le numéro de matricule se termine par 8

b) dont le prénom est "Paul" (utiliser **strcmp**)

c) dont le nom est un palindrome. Définir une fonction d'aide PALI qui fournit le résultat 1 si la chaîne transmise comme paramètre est un palindrome, sinon la valeur zéro.

Sauver le nouveau fichier à chaque fois sous le nom INFBIS.TXT.

Exercice 11.6

Créer sur disquette puis afficher à l'écran le fichier FAMILLE.TXT dont les informations sont structurées de la manière suivante:

Nom de famille

Prénom du père

Prénom de la mère

Nombre d'enfants

Prénoms des enfants

Le nombre d'enregistrements à créer est entré au clavier.

Attention: Le nombre de rubriques des enregistrements varie avec le nombre d'enfants !

Exercice 11.7

Ecrire un programme qui crée sur disquette le fichier MOTS.TXT contenant une série de 50 mots au maximum (longueur maximale d'un mot: 50 caractères). La saisie des mots se terminera à l'introduction du symbole `'*'` qui ne sera pas écrit dans le fichier.

Exercice 11.8

Ecrire un programme qui affiche le nombre de mots, le nombre de palindromes ainsi que la longueur moyenne des mots contenus dans le fichier MOTS.TXT. Utiliser les deux fonctions d'aide PALI et LONG_CH définies au chapitre 10.

Exercice 11.9

Ecrire un programme qui charge les mots du fichier MOTS.TXT dans la mémoire centrale, les trie d'après la méthode par propagation (méthode de la bulle - voir exercice 7.15) et les écrit dans un deuxième fichier MOTS_TRI.TXT sur la disquette. Les mots seront mémorisés à l'aide d'un tableau de pointeurs sur **char** et la mémoire nécessaire sera réservée de façon dynamique.

Exercice 11.10

A l'aide d'un éditeur de textes, créer un fichier NOMBRES.TXT qui contient une liste de nombres entiers. Dans le fichier, chaque nombre doit être suivi par un retour à la ligne. Ecrire un programme qui affiche les nombres du fichier, leur somme et leur moyenne.

Exercice 11.11

Ecrire un programme qui remplace, dans un fichier contenant un texte, les retours à la ligne par des espaces. Si plusieurs retours à la ligne se suivent, seulement le premier sera remplacé. Les noms des fichiers source et destination sont entrés au clavier.

Exercice 11.12

Ecrire un programme qui détermine dans un fichier un texte dont le nom est entré au clavier, le nombre de phrases terminées par un point, un point d'interrogation ou un point d'exclamation.

Utiliser une fonction d'aide FIN_PHRASE qui décide si un caractère transmis comme paramètre est un des séparateurs mentionnés ci-dessus. FIN_PHRASE retourne la valeur (logique) 1 si le caractère est égal à `'.'`, `'!'` ou `'?'` et 0 dans le cas contraire.

Exercice 11.13

Ecrire un programme qui détermine dans un fichier un texte dont le nom est entré au clavier:

- le nombre de caractères qu'il contient,

- le nombre de chacune des lettres de l'alphabet

(sans distinguer les majuscules et les minuscules),

- le nombre de mots,

- le nombre de paragraphes (c.-à-d.: des retours à la ligne),

Les retours à la ligne ne devront pas être comptabilisés dans les caractères. On admettra que deux mots sont toujours séparés par un ou plusieurs des caractères suivants:

- fin de ligne

- espace

- ponctuation: `. : , ; ? !`

- parenthèses: `( )`

- guillemets: `"`

- apostrophe: `'`

Utiliser une fonction d'aide SEPA qui décide si un caractère transmis comme paramètre est l'un des séparateurs mentionnés ci-dessus. SEPA restituera la valeur (logique) 1 si le caractère est un séparateur et 0 dans le cas contraire. SEPA utilise un tableau qui contient les séparateurs à détecter.

Exemple:

```
Nom du fichier texte : A:LITTERA.TXT
Votre fichier contient:
   12 paragraphes
   571 mots
   4186 caractères
dont
   279 fois la lettre a
   56 fois la lettre b
   . . .
   3 fois la lettre z
et 470 autres caractères
```

Exercice 11.14

Ecrire un programme qui affiche le contenu d'un fichier texte sur un écran de 25 lignes et 80 colonnes en attendant la confirmation de l'utilisateur (par 'Enter') après chaque page d'écran. Utiliser la fonction **getchar**.

Exercice 11.15

Ecrire un programme qui vérifie la validité d'une série de numéros de CCP mémorisés dans un fichier. Un numéro de CCP est composé de trois parties: un numéro de compte, un séparateur '-' et un numéro de contrôle. Un numéro de CCP est correct:

- si le reste de la division entière de la valeur devant le séparateur '-' par 97 est différent de zéro et égal à la valeur de contrôle.

- si le reste de la division par 97 est zéro et la valeur de contrôle est 97.

Exemple:

Nombre de CCP 15742-28 :	15742 modulo 97 = 28	correct
Nombre de CCP 72270-5 :	72270 modulo 97 = 5	correct
Nombre de CCP 22610-10 :	22610 modulo 97 = 9	incorrect
Nombre de CCP 50537-0 :	50537 modulo 97 = 0 nombre incorrect, car la valeur de contrôle devrait être 97.	
Nombre de CCP 50537-97 :	50537 modulo 97 = 0	correct

Utiliser une fonction CCP_TEST qui obtient comme paramètres les deux parties numériques d'un nombre de CCP et qui affiche alors un message indiquant si le numéro de CCP est valide ou non.

Pour tester le programme, créer à l'aide d'un éditeur de texte un fichier CCP.TXT qui contient les numéros ci-dessus, suivis par des retours à la ligne.

Exercice 11.16

Deux fichiers FA et FB dont les noms sont à entrer au clavier contiennent des nombres entiers triés dans l'ordre croissant. Ecrire un programme qui copie le contenu de FA et FB respectivement dans les tableaux TABA et TABB dans la mémoire centrale. Les tableaux TABA et TABB sont fusionnés dans un troisième tableau trié en ordre croissant TABC. Après la fusion, la tableau TABC est sauvé dans un fichier FC dont le nom est à entrer au clavier.

La mémoire pour TABA, TABB et TFUS dont les nombres d'éléments sont inconnus, est réservée dynamiquement après que les longueurs des fichiers FA et FB ont été détectées.