1) Première partie : Génération automatique de logins.

On désire gérer un groupe d'utilisateurs, sauvegardés dans une base de données Access. *Dans la suite de questions ci-après, on fera abstraction des méthodes d'accès aux données et seule nous concernera la gestion "locale" des utilisateurs.*

Au démarrage, l'interface utilisateur est celle qui apparaît sur la figure n°1.
Son fonctionnement est le suivant :

- l'utilisateur saisit son nom et son prénom dans deux zones txtNom et txtPrenom. Il n'a le droit de taper que des caractères alphabétiques ou les touches Retour ou Entrée.
- il sélectionne ensuite la promotion à laquelle il appartient, dans une zone de liste cboPromo.
- Son login est alors généré automatiquement, sous la forme :
  - un chiffre pour la promotion (1 pour les premières années, 2 pour les deuxièmes années, 3 pour les étudiants de licence, etc...).
  - deux caractères pour les deux premières lettres du prénom
  - quatre caractères pour les quatre premiers caractères du nom de famille.

Le login n'est pas modifiable par l'utilisateur et sera affiché dans un composant de type label, plutôt qu'une zone de texte. Ce login une fois affiché, le focus se retrouvera directement dans la zone de texte suivante, c'est-à-dire la zone **txtMdp**.

Arès saisie du mot de passe, l'utilisateur peut enregistrer les informations qu'il a saisies en appuyant sur le bouton "Enregistrer".

Figure n°1 : saisie d'un nouvel utilisateur.

Travail à faire :

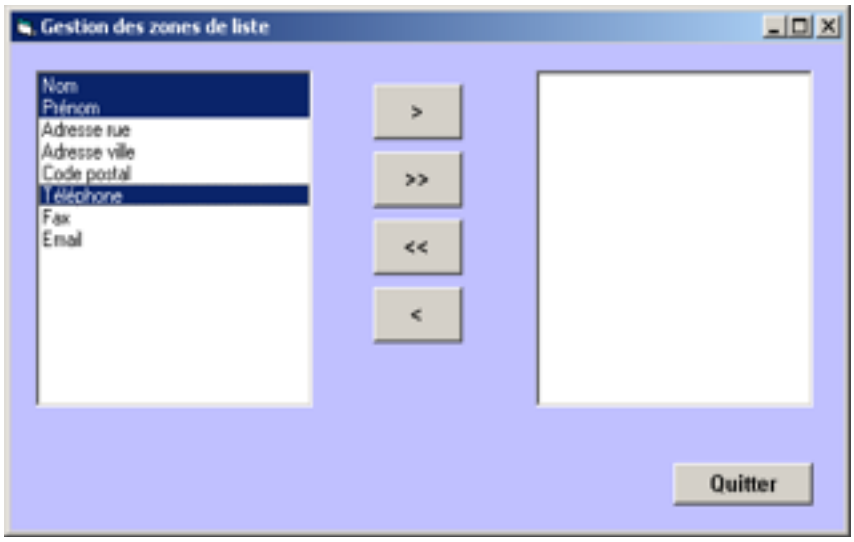
1. Proposer deux méthodes différentes pour le code de la procédure événementielle **Form_Load,** de manière à charger dans la zone de liste **cboPromo** un ensemble d'intitulés de promotions stockés pour le moment dans un tableau de n chaînes de caractères nommé **tabPromo**.

2. Ecrire le code de la procédure **txtNom_KeyPress** de manière à interdire à l'utilisateur les caractères autres que alphabétiques (ou les touches entrée et Retour).

3. Ecrire le code de la procédure **cboPromo_SelectedIndexChanged**, qui génère automatiquement le login de l'utilisateur après que celui-ci ait choisi sa promotion, et l'affiche dans un label intitulé **lblLogin**.

Deuxième partie : transfert d'éléments entre deux listes.

1. Concevoir une application permettant de transférer des éléments entre deux zones de liste, suivant le modèle ci-dessous :



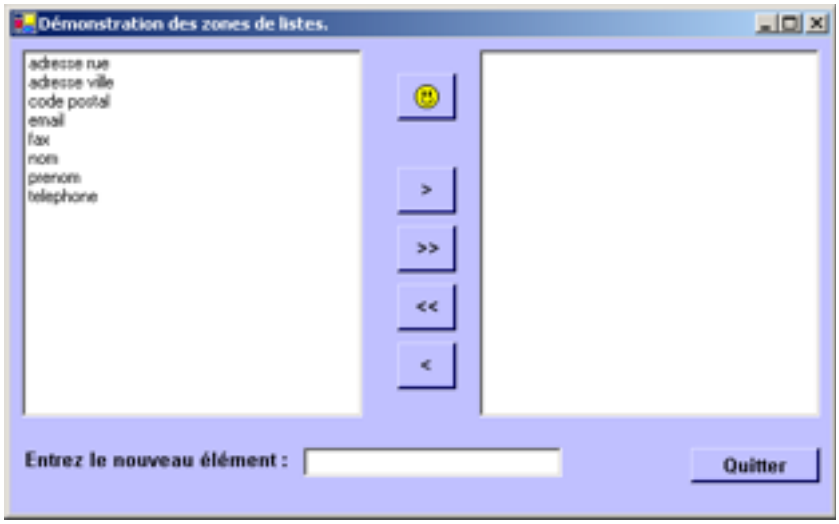
Remarques :
- Au départ, la zone de droite est vide.
- Les deux zones de liste doivent permettre une sélection multiple étendue.
- Un même élément ne doit pas apparaître plus d'une fois dans une liste.

Mise à jour de la liste gauche.

On désire pouvoir ajouter de nouvelles valeurs à la liste de gauche. Pour cela, l'utilisateur doit cliquer sur le bouton adéquat.

Déroulement événementiel :

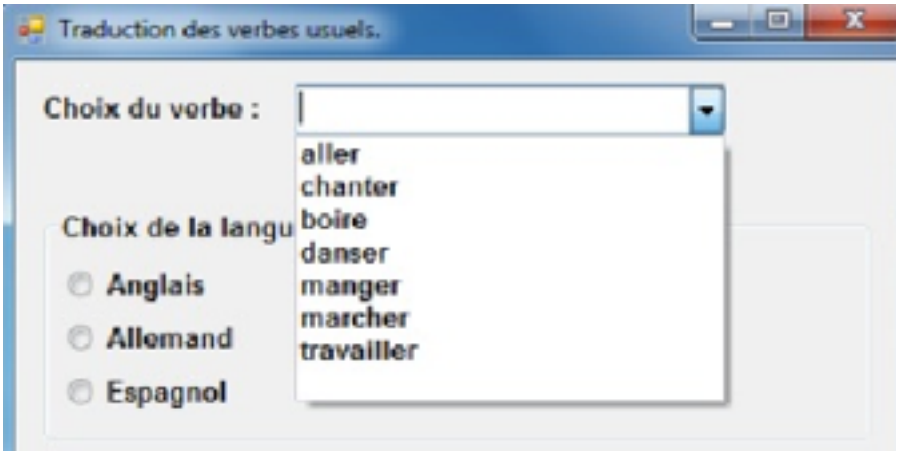
1. Une zone de saisie apparaît alors au bas de l'écran et le focus est donné à la zone de texte.



2. L'utilisateur entre un texte dans cette zone et lorsqu'il tape sur la touche Entrée (de code ASCII 13), un contrôle automatique est effectué.
3. Si le texte figure déjà dans la zone de liste de gauche, un message d'erreur l'indique à l'utilisateur, le contenu de la zone de saisie est effacé et le focus est replacé à cet endroit pour permettre une nouvelle saisie.
4. Si, par contre, il s'agit bien d'une nouvelle entrée, la nouvelle valeur est ajoutée à la liste de gauche et la zone de saisie du bas de l'écran disparaît.

Troisième partie : traduction de verbes.

L'application «TraducVerb» a pour objet l'apprentissage de verbes usuels dans la plupart des langues étudiées au lycée (anglais, allemand, espagnol, italien, ...).



Pour le moment, ces verbes, ainsi que leur traduction dans les différentes langues, sont stockés dans un tableau à deux dimensions «**tabVerbes**» (n lignes, m colonnes) dont vous trouverez un aperçu du contenu ci-dessous :

```
string [,] tabVerbes =  {
                    { "aller", "to go", "gehen", "ir", "andare" },
                    { "boire ", "to drink", "trinken", "beber", "bere" },
                    { "chanter", "to sing", "singen", "cantar", "cantare" },
                    { "danser", "to dance", "tanzen", "bailar", "ballare" },
                    { "fermer", "to close", "schliessen", "cerrar", "chiudere" },
                    { "manger", "to eat", "essen", "comer", "mangiare" },
                /.../
                    { "travailler", "to work", "arbeiten", "trabajar", "lavorare" },
                    };
```

Pour des raisons d'efficacité, le code de l'événement «CheckedChanged» de chacun des boutons radio a été associé (en mode graphique, via la fenêtre de propriétés) à l'événement **CheckedChanged** du bouton radio «**rdbAnglais**». Parallèlement, les propriétés Tag des différents boutons radio ont été initialisés avec les entiers de 0 et n-1 (n étant le nombre de boutons radio présents dans la groupbox «Choix de la langue»).

Travail à faire : Ecrire le code de la procédure événementielle :

```
private void rdbAnglais_CheckedChanged(object sender, EventArgs e)
{
  /.../
 }
```

de manière à afficher, dans le conteneur GrpTraduction du bas du formulaire, la traduction du verbe sélectionné au préalable dans la zone de liste cboVerbe, sous la forme décrite sur l'interface graphique suivante :