SÉANCE 7

Objectif

Le but de cette septième séance est de manipuler les structures et de poursuivre l'utilisation des fonctions et des chaînes de caractères. Cette séance est aussi l'occasion de gérer le tampon d'entrée.

Outils

Voici deux fonctions de la bibliothèque standard de manipulation des chaînes de caractères qui vous seront utiles lors de cette séance. Elles nécessitent l'inclusion de `string.h`.

- `size_t strlen(const char *Ch)`
  La fonction `strlen` retourne le nombre de caractères de la chaîne représentée par `Ch`, caractère `'\0'` non compris.
- `int strcmp(const char *Ch1, const char *Ch2)`
  La fonction `strcmp` retourne :
  - 0 si les deux chaînes sont identiques ;
  - un entier positif si la chaîne représentée par `Ch1` est supérieure à la chaîne représentée par `Ch2` selon l'ordre lexicographique ;
  - un entier négatif si la chaîne représentée par `Ch1` est inférieure à la chaîne représentée par `Ch2` selon l'ordre lexicographique.

Exercices

Exercice 1 (Manipulations élémentaires d'une variable structurée)

1. Écrivez la définition de la structure `sContact` constituée de trois champs : `Nom`, `Prenom` et `Mail`. Ces champs sont des tableaux de caractères permettant de stocker, sous la forme de chaînes de caractères (donc terminées par le caractère `'\0'`), le nom, le prénom et l'adresse mail d'une personne. On suppose qu'un nom comporte au plus 30 caractères, qu'un prénom comporte au plus 20 caractères. Une adresse mail est constituée d'au plus 254 caractères.
2. Écrivez la fonction d'en-tête :
   `void AfficherContact1(struct sContact Contact)`
   qui affiche à l'écran les valeurs des trois champs de la variable structurée `Contact`, chaque champ étant affiché sur une ligne différente.
3. Écrivez la fonction principale permettant de tester la fonction précédente. Pour cela, vous pouvez tout simplement déclarer une variable structurée et en initialiser les champs au moment de la déclaration avec les valeurs de votre choix.
4. Écrivez la fonction d'en-tête :
   `void AfficherContact2(const struct sContact *pContact)`
   qui a le même rôle que la fonction `AfficherContact1`. Mais cette seconde version permet d'éviter de passer toute la variable structurée en paramètre et de ne passer que son adresse. Le qualifieur `const` permet de préciser aux utilisateurs de la fonction `AfficherContact2` que cette dernière ne modifiera pas la variable structurée pointée (cf. les supports et les explications données en cours pour l'utilisation de `const`).
5. Modifiez la fonction principale pour tester la fonction précédente.

6. Écrivez la fonction d'en-tête :
   `void VidageTamponEntree(void)`
   qui lit les caractères présents dans le tampon d'entrée jusqu'au caractère `'\n'` inclus (cf. les supports et les explications données en cours pour la gestion du tampon d'entrée).
7. Après avoir relu la description de la fonction `fgets` sur les supports de cours, écrivez la fonction d'en-tête :
   `void LireLigne(char Ch[], int LongueurMax)`
   qui lit une ligne de texte tapée au clavier et la stocke sous forme d'une chaîne de caractères dans le tableau représenté par le paramètre `Ch`. La fonction `LireLigne` doit :
   - faire appel à la fonction `fgets` ;
   - supprimer (en le remplaçant par `'\0'`) le caractère `'\n'` que `fgets` stocke à la fin de la chaîne si le nombre maximal de caractères n'a pas été atteint ;
   - vider le tampon d'entrée si le nombre maximal de caractères a été atteint, afin de préparer d'éventuelles lectures à venir.
8. Écrivez la fonction d'en-tête :
   `void LireContact(struct sContact *pContact)`
   qui lit au clavier le nom, le prénom et l'adresse mail d'une personne et les stocke dans la variable structurée d'adresse `pContact`. On suppose que l'utilisateur tapera chacun des trois champs sur une ligne différente. Cette fonction fera appel à la fonction `LireLigne`.
9. Complétez la fonction principale pour tester la fonction précédente.

☞ Exercice 2 (Gestion d'un répertoire de contacts)

Un répertoire de contacts sera représenté par un tableau de contacts déclaré dans la fonction principale. On suppose que le répertoire contient au plus 64 contacts.

1. Écrivez la fonction d'en-tête :
   `void AfficherRepertoire(struct sContact Repertoire[], int NbContacts)`
   qui affiche à l'écran le contenu du répertoire `Repertoire` contenant `NbContacts` personnes. Les informations de chaque personne seront affichées en faisant appel à la fonction `AfficherContact2` de l'exercice 1.
2. Écrivez la fonction principale permettant de tester la fonction précédente. Pour cela, vous pouvez tout simplement initialiser le tableau `Repertoire` au moment de sa déclaration avec les informations relatives à trois ou quatre personnes.
3. Écrivez la fonction d'en-tête :
   `void AjouterContact(struct sContact Repertoire[], int *pNbContacts)`
   qui :
   - demande à l'utilisateur de taper les informations concernant une personne ;
   - lit ces informations grâce à la fonction `LireContact` de l'exercice 1 ;
   - ajoute ce contact au répertoire.
4. Modifiez la fonction principale pour tester la fonction précédente.
5. Écrivez la fonction d'en-tête :
   `int Rechercher(char NomRecherche[],struct sContact Repertoire[], int NbContacts)`
   qui recherche dans le répertoire `Repertoire` contenant `NbContacts` la première personne dont le nom est identique à celui contenu dans la chaîne de caractères `NomRecherche`. Cette fonction doit retourner :
   - l'indice de l'élément trouvé dans le tableau `Repertoire` si le nom existe dans le répertoire ;
   - `-1` sinon.

   Cette fonction utilisera la fonction `strcmp` de la bibliothèque standard.

6. Écrivez la fonction d'en-tête :
   `void RechercherContact(struct sContact Repertoire[], int NbContacts)`
   qui :
   - demande à l'utilisateur de taper le nom de la personne recherchée ;
   - lit ce nom avec la fonction `LireLigne` de l'exercice 1 ;
   - recherche ce nom dans le répertoire avec la fonction `Rechercher` ;
   - affiche les informations correspondantes avec la fonction `AfficherContact2` de l'exercice 1 si la personne existe ou un message adéquat sinon.
7. Modifiez la fonction principale pour tester la fonction précédente.

Pour aller plus loin

☞ Exercice 3 (Ajout d'un menu)

L'objectif de cet exercice est de mettre en place un menu permettant d'utiliser plus confortablement les fonctionnalités développées dans l'exercice 2.

1. Ajoutez au programme de l'exercice 2 la fonction d'en-tête :
   `int LireChoixAux(int ChoixMax)`
   qui :
   - demande à l'utilisateur de taper son choix sous la forme d'un entier compris entre `0` et `ChoixMax` ;
   - lit cet entier :
   - le retourne à l'appelant.
2. Écrivez la fonction d'en-tête :
   `int LireChoix(int ChoixMax)`
   qui :
   - lit le choix de l'utilisateur grâce à la fonction `LireChoixAux` ;
   - tant que l'entier tapé par l'utilisateur n'est pas dans l'intervalle attendu le redemande à l'utilisateur ;
   - vide le tampon d'entrée en vue d'éventuelles entrées ultérieures ;
   - retourne l'entier lu.
3. Écrivez la fonction d'en-tête :
   `void AfficherMenu(void)`
   qui affiche à l'écran le menu sous la forme suivante :

```
--------- MENU ---------
1 - Afficher le repertoire
2 - Ajouter un contact
3 - Rechercher un contact par son nom

0 - Quitter le programme
```

4. Après avoir lu, pages 15 et 16 du support complémentaire, la description de l'instruction `switch` permettant d'effectuer des choix multiples, modifiez la fonction principale afin que, tant que l'utilisateur ne souhaite pas quitter le programme, elle affiche le menu, lise le choix de l'utilisateur et effectue l'action correspondante (c'est ici que l'instruction `switch` peut être utile).