SÉANCE 5

Objectif

Le but de cette sixième séance est de continuer le travail sur les fonctions et le passage de paramètres avec l'utilisation des pointeurs, de manipuler des chaînes de caractères et d'introduire la gestion dynamique des tableaux. **Nous vous rappelons que du travail personnel, en dehors des séances, est indispensable.**

Consigne pour cette séance : pour ceux qui n'ont pas terminé la séance 4, terminez-la et n'oubliez pas d'appeler l'enseignant·e en cas de difficulté ; passez ensuite à l'exercice 1 ci-dessous.

Exercices

Exercice 1 (Fonctions de manipulation de chaînes de caractères)

L'objectif de cet exercice consiste à écrire des fonctions de manipulation de chaînes de caractères **sans** utiliser les fonctions de la bibliothèque standard (donc sans inclure `string.h`).

Testez les unes après les autres les fonctions en utilisant la fonction principale fournie dans le fichier `tp5ex1main.c` et en enlevant les commentaires au fur et à mesure de votre progression.

Dans la suite, une chaîne de caractères est manipulée au travers de l'adresse de son premier caractère (adresse de début de la chaîne).

1. Écrivez la fonction d'en-tête :
   `int Lecture(char *ch, int NbCarMax)`
   qui lit une ligne de texte tapée au clavier et la stocke, en ajoutant le caractère `'\0'` mais sans stocker le caractère `'\n'`, dans la chaîne désignée par le paramètre `ch`. Le paramètre `NbCarMax` contient le nombre maximal de caractères lus (pour éviter les débordements en mémoire). La fonction doit retourner le nombre de caractères lus (sans compter le caractère `'\n'`).
   Vous pouvez vous inspirer de ce que vous avez écrit pour l'exercice 3 de la séance 3.
2. Écrivez la fonction d'en-tête :
   `int Longueur(char *ch)`
   qui retourne la longueur (nombre de caractères sans compter le caractère `'\0'`) de la chaîne dont l'adresse du premier caractère est passée en paramètre.
3. Écrivez la fonction d'en-tête :
   `void Copie(char *Destination, char *Source)`
   qui copie la chaîne de caractères désignée par le paramètre `Source` dans la chaîne désignée par le paramètre `Destination`. Attention, la copie doit être une « vraie » chaîne de caractères, c'est-à-dire se terminer par `'\0'`.
4. Écrivez la fonction d'en-tête :
   `int Differentes(char *ch1, char *ch2)`
   qui retourne 1 si les chaînes désignées par `ch1` et `ch2` sont différentes (contiennent au moins un caractère différent), 0 sinon.
5. Écrivez la fonction d'en-tête :
   `void Ajout(char *Destination, char *Source)`
   qui ajoute la chaîne de caractères désignée par `Source` à la fin de la chaîne de caractères désignée par `Destination`.

☞ Exercice 2 (Gestion dynamique de tableaux)

L'objectif de cet exercice est d'utiliser les deux fonctions principales de gestion dynamique de la mémoire pour allouer et libérer un tableau.

1. Écrivez la fonction d'en-tête :
   `double *CreerTableau(int NbElts)`
   qui alloue dynamiquement un tableau de `NbElts` éléments de type `double` et retourne l'adresse de ce tableau ou `NULL` en cas de problème lors de l'allocation.
2. Écrivez la fonction d'en-tête :
   `void SaisirTableau(double *Tab, int NbElts)`
   qui saisit `NbElts` éléments de type double tapés au clavier et les range dans le tableau `Tab`. On suppose que l'appelant a réservé l'espace mémoire nécessaire.
3. Écrivez la fonction d'en-tête :
   `void AfficherTableau(double *Tab, int NbElts)`
   qui affiche à l'écran le contenu du tableau passé en paramètre.
4. Écrivez la fonction d'en-tête :
   `void LibererTableau(double *Tab)`
   qui libère l'espace mémoire occupé par le tableau passé en paramètre.
5. Écrivez la fonction principale d'un programme permettant de tester les fonctions précédentes en demandant à l'utilisateur de donner le nombre de valeurs qu'il souhaite saisir puis les valeurs du tableau.