PL1 - Travaux pratiques - Séance 3

Tableaux

Exercice 1 - Somme des éléments d'un tableau [obligatoire]

On considère l'algorithme suivant (proche de l'exemple E3.1 du Polycopié d'Algorithmique) qui permet d'initialiser un tableau par lecture au clavier puis de calculer la somme de ses éléments.

lexique
N : constante entière 10
T : tableau sur [0..N-1] d'entier
Somme : entier { pour calculer la somme }
debut
pour i allant de 0 a N-1 : Ecrire("Saisissez un entier") ; Lire(T_i)
Somme $\leftarrow$ 0
pour i allant de 0 a N-1 : Somme $\leftarrow$ Somme + T_i
Ecrire(Somme)
fin

Q1. Traduire cet algorithme en C dans un fichier de nom ***somme_tab.c***.
Compilez (`gcc -Wall somme_tab.c -o somme_tab`) et exécutez votre programme.

Q3. Créez un fichier de nom ***donnees.txt*** contenant 10 entiers (un par ligne) Exécutez maintenant votre programme en re-dirigeant les entrées depuis ce fichier : `./somme_tab < donnees.txt`

Exercice 2 - Recherche séquentielle [obligatoire]

On considère un tableau T de 10 entiers. On souhaite savoir si ce tableau contient un entier x donné. On utilise pour cela l'algorithme suivant :

lexique
N : constante entière 10
T : tableau sur [0..N-1] d'entier
x : entier
j : entier sur 0..N
debut
{Lecture du tableau}
pour i allant de 0 a N-1 : Ecrire("Saisissez un entier") ; Lire(T_i)
Ecrire("Entier à chercher ?") ; Lire(x)
{Recherche de l'entier x dans T}
j $\leftarrow$ 0
tantque j $<$ N etpuis $T_j \neq$ x : j $\leftarrow$ j+1
si j $<$ N alors Ecrire("trouvé" j) sinon Ecrire("non trouvé")
fin

Q1. Traduire cet algorithme en C dans un fichier de nom ***recherche.c***.

Q2. Compilez (`gcc -Wall recherche.c -o recherche`) et exécutez votre programme.

Exercice 3 - Représentation d'une séquence dans un tableau [optionnel]

On considère un fichier **data.txt** contenant une séquence de p entiers strictement positifs avec $0 < p < N$, où N est une constante donnée. On souhaite écrire un programme qui affiche cette séquence dans l'ordre inverse.

Q1. On propose de représenter cette séquence dans un tableau T avec *longueur explicite* :

N : constante entiere > 0
SeqLE : type $<$ T : tableau sur 0..N-1 d'entier, L : entier sur 0..N $>$
S : SeqLE { S est une séquence avec longueur explicite}

1. Ecrivez un programme C basé sur ce lexique qui lit une séquence de p entiers, la mémorise dans la variable S et affiche la séquence inverse à l'écran. Compilez votre programme en un exécutable de nom **inverse**.
2. Compilez et testez votre programme. Vous pouvez pour cela utiliser la redirections des entrées-sorties : `./inverse < data.txt > data2.txt`

Q2. Même question en représentant cette séquence dans un tableau T avec *marque de fin* :

N : constante entiere > 0
M : constante -1 {M est la marque de fin}
SeqMF : type tableau sur 0..N-1 d'entier
S : SeqMF { S est une séquence avec marque de fin}

Indication : Avec cette reprèsentation de la séquence, l'accès au dernier élément n'est pas immédiat (la longueur de la séquence n'est pas connue). Il sear donc nécessaire d'effectuer un premier parcours de la sequence afin d'obtenir l'indice de son dernier élément (celui qui précède la marque de fin).

Exercice 4 - Chaînes de caractères [conseillé]

On propose d'écrire un programme qui lit une chaîne de caractères au clavier et la ré-écrit à l'écran après avoir transformé les lettres majuscules en lettres minuscules (les autres caractères étant inchangés).

Exemple : `"M2P_CCI_c'est_TROP_Cool_!"` sera transformé en `"m2p_cci_c'est_trop_ cool_!"`

On pourra utiliser les éléments suivants :

— représentation d'une chaîne de catactères en C :
```
char s[50] ; // s est une chaîne d'au plus 49 caractères terminée par '\0'
```
— longueur d'une chaîne de caractères : la fonction `strlen(s)` renvoie le nombre de caractères de `s`, sans compter la marque de fin. Il faut inclure la librairie `<string.h>` en début de fichier pour pouvoir utiliser cette fonction.
— lecture-écriture d'une chaîne de caractères :
```
char s[50] ;
scanf("%s", s) ; // lecture de la chaîne s
printf("%s", s) ; // écriture de la chaîne s
```
— les lettre majuscules ont un code Ascii dans l'intervalle $[65, 90]$
— en ajoutant 32 au code Ascii d'une majuscule on obtient le code Ascii de la minuscule correspondante.