Structure de données

TP2 : Tableaux et sous-programmes

Exercice 1 : Lecture d'un tableau

Ecrire un programme qui initialise un tableau d'entiers en lisant des valeurs au clavier. On considèrera trois modes de lecture :

- lecture de la taille effective du tableau, puis des valeurs ;
- lecture des valeurs les unes après les autres et arrêt sur une valeur particulière (une valeur négative par exemple) ;
- lecture de couples (indice, valeur) avec arrêt lorsque l'indice donné est -1.

Quels sont les avantages et les inconvénients de ces différentes approches?

Remarque : on veillera à ce que la capacité du tableau ne soit pas dépassée.

Exercice 2 : Tri par sélection

Soit $A[1..N]$ un vecteur de N entiers relatifs quelconques, l'objectif est de trier le vecteur A en utilisant le tri par sélection. Le vecteur A est trié si $A[1] \leq A[2] \leq \cdots \leq A[N]$.

Le tri par sélection est un tri en $(N-1)$ étapes. L'étape i consiste à ranger à sa place le i^{e} plus petit élément du vecteur.

Exemple : Voici les différentes valeurs du vecteur 8 2 9 5 1 7 après chaque étape (la partie encadrée correspond à la partie du vecteur déjà traitée et donc triée) :

vecteur initial	:	8 2 9 5 1 7
après l'étape 1	:	[1] 2 9 5 8 7
après l'étape 2	:	[1 2] 9 5 8 7
après l'étape 3	:	[1 2 5] 9 8 7
après l'étape 4	:	[1 2 5 7] 8 9
après l'étape 5	:	[1 2 5 7 8 9]

Écrire un programme qui lit une série de N ($N \geq 0$, lu au clavier) entiers relatifs, les range dans un vecteur, les classe par ordre croissant en utilisant le tri par sélection et, enfin, affiche la série ordonnée.

Utiliser les sous-programmes pour structurer votre programme.

Exercice 3 : Recherche d'occurrences

Dans cet exercice, nous considérons un tableau d'entiers et nous nous intéressons aux occurrences de ses éléments.

3.1. Indiquer si un entier donné est présent dans le tableau ou non.

3.2. Indiquer l'indice, dans le tableau, de la première occurrence d'un entier donné.

3.3. Indiquer l'indice, dans le tableau, de la dernière occurrence d'un entier donné.

3.4. Indiquer l'indice, dans le tableau, de la n^{e} occurrence d'un entier donné.

Exercice 4 : Loi de Benford

Prenons un ensemble de données numériques quelconques et classons-les en fonction de leur premier chiffre significatif. Qu'obtenons-nous ? Intuitivement, nous nous attendons à trouver 9 classes contentant une quantité équivalente d'échantillons, c'est-à-dire environ 11% de nombres commençant par 1, puis 11 % par le chiffre 2...

Pourtant, on s'aperçoit généralement qu'une telle distribution intuitive n'est pas du tout respectée ! Le chiffre 1 est bien plus représenté, le chiffre 2 un peu moins... le chiffre 9 n'apparaissant que dans guère plus de 4 % des cas.

C'est en constatant l'usure excessive du premier volume des tables de logarithmes que *Newcomb* découvrit en 1881 une telle irrégularité et proposa une loi de distribution (reprise plus tard par *Benford*). Cette loi stipule que le chiffre $i \neq 0$ apparaît avec une probabilité de l'ordre de $\log_{10}(1 + 1/i)$.

Cette loi n'a toujours pas pu être démontrée ni son cadre d'application clairement défini. Elle trouve cependant quelques applications (détection de fraudes par exemple).

L'objectif de cet exercice est de vérifier, sur quelques exemples, la loi de *Benford*.

4.1. Ecrire un programme permettant de mesurer la fréquence du premier chiffre significatif des valeurs d'un échantillon (d'entiers). Les résultats seront affichés sous forme de pourcentage.

4.2. Appliquer le programme précédemment défini sur un ensemble de mesures, par exemple sur la taille des fichiers contenus dans vos répertoires[1].

[1] Pour cela, vous pouvez utiliser quelques commandes Unix (telles que `du`) pour fournir les données à votre programme ou enrichir ce dernier pour qu'il détermine lui-même la taille de chaque fichier.

Exercice 5 : Pile d'entiers

Une pile est une structure de données contenant des données de même type. Le principe d'une pile et que le dernier élément ajouté est le premier à en être retiré. Si on considère une pile d'assiettes, on pose une nouvelle assiette au sommet de la pile. Si on prend une assiette, c'est naturellement celle du sommet de la pile. On dit que la pile est une structure de données de type LIFO (Last In, First Out : dernier entré, premier sorti). Les opérations sur une pile sont les suivantes :

- initialiser une pile (une variable de type Pile). Une pile ne peut être utilisée que si elle a été initialisée ;
- empiler un nouvel élément dans une pile ;
- dépiler une pile (supprimer le dernier élément ajouté dans la pile) ;
- accéder à l'élément au sommet de pile ;
- savoir si une pile est vide ou non.

5.1. *Spécification des opérations de la pile*. Définir l'interface des sous-programmes correspondants aux opérations définies ci-dessus.

5.2. *Utilisation de la pile.* La pile étant définie, on peut écrire une application l'utilisant même si l'implantation des opérations de la pile n'est pas encore définie.

Par exemple, on peut utiliser une pile d'entiers pour afficher un entier à l'écran en utilisant simplement l'affichage d'un caractère. Ecrire un tel sous-programme. On appliquera le schéma de *Horner*.

5.3. *Implantation des opérations de la pile*. On choisit de représenter la pile par un enregistrement composé d'un tableau d'entiers (les éléments mis dans la pile) et un entier (le nombre d'éléments dans la pile). Le i^e élément ajouté dans la pile est à la i^e position du tableau. Ecrire les implantations des sous-programmes spécifiés à la question 5.1.