POLYTECH SORBONNE

Sorbonne Université
2025-2026

Programmation en langage C – EI3

Travaux Pratiques

Objectif(s)

- ★ Tableaux statiques
- ★ Manipulation de tableaux
- ★ Boucles sur des tableaux

Exercice 1 – Tri par sélection

Dans cet exercice, on cherche à écrire un programme `selection.c` qui lit un tableau de réels dans un fichier texte, puis le trie et l'affiche.

1. Écrire un programme qui lit un fichier texte `"valeurs.txt"` contenant :
   - le nombre de valeurs du fichier (noté n)
   - toutes les valeurs réelles

   et les stocke dans un tableau T. On sait qu'il n'y a pas plus de 100 valeurs dans le fichier.
   À la fin du programme, on n'oubliera pas de fermer le fichier.
2. En commentaires, écrire l'algorithme (sous forme de pseudo-code) de *tri par sélection* qui se base sur le principe suivant : on sélectionne tout d'abord l'élément le plus petit du tableau (on trouve son indice). Une fois cet indice p trouvé, on échange les élements $T[0]$ et $T[p]$. Puis on recommence ces opérations pour le reste du tableau (c'est-à-dire les éléments compris entre les indices 1 et n). On recherche alors le plus petit élément de cette nouvelle suite de nombres et on échange avec $T[1]$. Et ainsi de suite jusqu'au moment où on a placé tous les élements du tableau.
3. En déduire le programme qui trie le tableau T. Le tester avec le fichier trouvé dans le repertoire .
4. Terminer le programme en écrivant le tableau de valeurs triées dans le fichier `"valeurs-triees.txt"`.

Exercice 2 – Tri à bulle

Dans cet exercice, on reprend le programme de l'exercice de *tri par sélection*, mais en utilisant un autre algorithme de tri.
On utilise le *tri à bulles* qui consiste à comparer successivement tous les éléments adjacents d'un tableau et de les échanger si le premier élément est supérieur au second.
Les détails sont donnés dans l'algorithme suivant :

Algorithme 1 : Tri à bulles

Données : T, le tableau à trier, n le nombre d'éléments de L
Sorties : T la liste triée par ordre croissant

```
1 pour i de 0 à n − 1 faire
2 |   pour j de 1 à n − i − 1 faire
3 |   |   si T[j − 1] > T[j] alors
4 |   |   |   échanger T[j − 1] et T[j]
5 |   |   fin
6 |   fin
7 fin
```

1. Écrire le programme C correspondant dans un fichier `bulles.c` (lecture des valeurs dans le fichier, tri, et sauvegarde des valeurs triées)

2. On peut constater que lorsqu'aucune permutation n'est effectuée lors d'un passage, le tableau est trié, alors les passages ultérieurs sont donc inutiles. Utiliser cette propriété pour optimiser l'algorithme précédent et écrire une nouvelle version du programme.

Exercice 3 – Sudoku

Tout le monde connaît le Sudoku : il s'agit d'une grille 9×9 de chiffres allant de 1 à 9.
Le but du jeu est de remplir cette grille avec une série de chiffres tous différents, qui ne se trouvent jamais plus d'une fois sur une même ligne, dans une même colonne ou dans une même sous-grille (9 carrés de 3×3).

Le but de cet exercice est de vérifier qu'une grille est correctement remplie (qu'elle vérifie donc les conditions précédemment énoncées).

Il n'est pas demander dans cet exercice d'écrire une/des fonction(s), mais cela est possible si vous pensez y arriver.

1. Écrire un programme `sudoku.c` qui remplit un tableau 9×9 d'entiers avec les chiffres lues dans le fichier "`grille.txt`" du répertoire .
   On pensera à gérer correctement les erreurs de fichier (fichier inexistant, etc.).

2. Ce programme doit ensuite afficher la grille sous la forme suivante (avec les caractères – et `|`) :

```
-----------------------------------
| 3 | 8 | 4 | 1 | 2 | 3 | 7 | 9 | 5 |
-----------------------------------
| 2 | 5 | 7 | 9 | 8 | 3 | 1 | 6 | 4 |
-----------------------------------
| 9 | 1 | 6 | 7 | 5 | 4 | 3 | 2 | 8 |
-----------------------------------
| 4 | 7 | 8 | 6 | 3 | 2 | 9 | 5 | 1 |
-----------------------------------
| 1 | 3 | 5 | 8 | 4 | 9 | 2 | 7 | 6 |
-----------------------------------
| 6 | 2 | 9 | 5 | 1 | 7 | 4 | 8 | 3 |
-----------------------------------
| 8 | 4 | 3 | 2 | 9 | 5 | 6 | 1 | 7 |
-----------------------------------
| 7 | 9 | 1 | 3 | 6 | 8 | 5 | 4 | 2 |
-----------------------------------
| 5 | 6 | 2 | 4 | 7 | 1 | 8 | 3 | 9 |
-----------------------------------
```

3. Compléter le programme pour qu'il vérifie si la grille de sudoku est correcte pour toutes les lignes
   En cas de grille non correcte, vous pouvez aussi afficher un message d'erreur (du genre "*condition sur ligne 2 non remplie.*).

4. Compléter avec la vérification sur les colonnes.

5. Enfin, effectuer la vérification sur les sous-tableaux.