

## SAÉ S1.02 : comparaison d'approches algorithmiques

### Version 1 : le Sokoban autonome

#### Présentation générale

Vous êtes chargés de programmer un **jeu de Sokoban autonome** : en lui fournissant une partie (fichier `.sok`) et une suite de déplacements (fichier `.dep`), votre programme devra "dérouler" la partie correspondante. Votre programme devra donc :

1. demander à l'utilisateur le nom d'une partie (fichier `.sok`),
2. demander à l'utilisateur le nom d'une suite de déplacements (fichier `.dep`),
3. "faire jouer" la partie : chaque déplacement doit être visible à l'écran et une pause entre les déplacements doit être prévue, suffisamment longue pour pouvoir suivre facilement le déroulement,
4. afficher le bilan.

#### Détail des étapes

##### Étapes 1 et 2

Vous trouverez sur Moodle les procédures permettant la lecture des deux types de fichiers. Vous trouverez également quelques exemples de fichiers `.sok` et `.dep` qui vous permettront de tester votre programme.

##### Étape 3

Cette étape consiste à traiter un à un les symboles de la liste des déplacements et à actualiser l'affichage en conséquence. Vous programmerez une pause d'un quart de seconde (soit 250 000 microsecondes) après chaque déplacement pour que le déroulé à l'écran puisse être suivi confortablement.

##### Étape 4

Au final, votre programme devra indiquer si la suite de déplacements constitue une solution ou non pour cette partie et, si c'est le cas, afficher le nombre de déplacements de la suite (voir exemples page suivante).



Exemple 1 :

**La suite de déplacements <fichier.dep> est bien une solution pour la partie <fichier.sok>.**

**Elle contient 999 déplacements.**

Exemple 2 :

**La suite de déplacements <fichier.dep> N'EST PAS une solution pour la partie <fichier.sok>.**

## Détails techniques

### Programmer une pause

La procédure `usleep(...)` permet d'interrompre l'exécution d'un programme durant un nombre donné de microsecondes. Par exemple, l'instruction `usleep(500000)` permet de programmer une pause d'une demi-seconde.

### Manipulation de fichiers

Les procédures de lecture de fichier sont fournies sur Moodle ; elles sont à ajouter au code source de votre programme :

- `void chargerPartie(t_plateau plateau, char fichier[])`  
lit les données du fichier dont le nom est passé en paramètre pour initialiser le tableau passé en paramètre.
- `void chargerDeplacements(t_tabDeplacement t,  
char fichier[],  
int * nb)`  
lit les données du fichier dont le nom est passé en paramètre pour initialiser le tableau passé en paramètre. Fournit en sortie le nombre d'éléments lus et transférés du fichier vers le tableau.

### Mise en majuscule/minuscule

La fonction `toupper()` retourne la majuscule correspondant au caractère passé en paramètre. La fonction `tolower()` fait l'inverse.

Exemples :    `toupper('g')` retournera `'G'`  
                  `toupper('H')` retournera `'H'`  
                  `tolower('H')` retournera `'h'`  
                  `tolower('h')` retournera `'h'`

Pensez à inclure la bibliothèque `<ctype.h>`.