Programmation en C et structures de données

guillaume.revy@univ-perp.fr

CC2, lundi 16/12/2024 de 14h à 18h

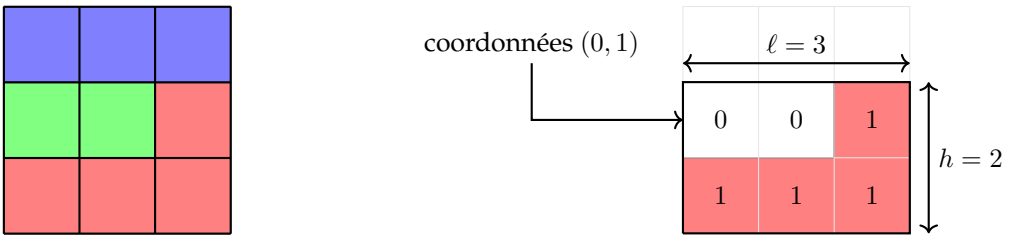
Ce programme est à réaliser seul, sous environnement GNU/Linux. Il doit être laissé dans le répertoire ProgC-CC2-2024/ à la racine du compte examen, sous forme d'un unique fichier `<nom_prenom.c>`. Seuls les documents disponibles dans ce réportoire sont autorisés. Tout autre document (cours, exercices, ...), ou autre matériel (téléphone portable, ...) est interdit.

Conseils :

- Il est à noter que la qualité primera sur la quantité : un programme court, clair, bien conçu et qui compile sera mieux noté qu'un gros programme mal conçu, incompréhensible, voire qui ne compile pas. Utilisez des noms de variables explicites, indentez votre programme, et n'hésitez pas à le commenter.
- Lisez tout le sujet avant de commencer.
- Veuillez respecter les noms de fonctions et structures proposées dans le sujet.
- Le sujet est assez court : une attention particulière devra être apportée à l'allocation dynamique et la libération de la mémoire ... Pensez à utiliser `valgrind` !

Un programme qui ne compile pas sera considéré comme faux, et il entrainera la note de 0.

C'est Noël. Vous avez commandé un puzzle, mais vous n'arrivez pas à le résoudre. Le puzzle est similaire à celui de taille 3×3 en Figure 1a.



(a) puzzle résolu (b) pièce rouge

FIGURE 1 – Exemple de puzzle.

L'objectif du TP est d'extraire la liste de pièces d'un fichier, de résoudre le puzzle, et de générer l'image correspondant. Saurez-vos relevez le défi ?

1. Modélisation des pièces et de la grille

Dans ce TP, un puzzle est vu comme une grille de dimensions $n \times m$ (n lignes, m colonnes), composée d'un tableau de k pièces que nous allons chacune poser à une position différente sur la grille. Une pièce peut être modélisée par :

- un petit tableau 2D englobant complètement cette pièce,
- les dimensions (ℓ, h) de ce tableau, le code couleur de la pièce,
- et les coordonnées (x, y), indiquant la position sur la grille de la case en haut à gauche de ce tableau,

Par exemple, la pièce rouge de la Figure 1 peut sera modélisée par le tableau de taille 3×2 dont le contenu sera $\{\{0, 0, 1\}, \{1, 1, 1\}\}$, comme illustré sur la Figure 1b, et positionné en coordonnées $(0, 1)$.

► **1.** Définir les structures `piece` et `grille`.

► **2.** Écrire les fonctions `creer_piece` et `detruire_piece`.

► **3.** Écrire les fonctions `creer_grille` et `detruire_grille`.

Une pièce non encore posée sur la grille sera de coordonnées $(-1, -1)$.

2. Chargement et affichage texte du puzzle

Les pièces du puzzle sont stockées dans un fichier. Plus particlièrement, ce fichier contient les dimensions $n \times m$ de la grille, le nombre k de pièces, et une ligne par pièce. Pour chaque pièce, une ligne contient :

— les dimensions (ℓ, h),
— le contenu du tableau 2D représentant la pièce, stocké ligne par ligne, colonne par colonne (c'est-à-dire, une suite de $\ell \times h$ cacratères 0 ou 1),
— et un code couleur représenté par un entier.

Par exemple, la ligne correspondant à la pièce rouge (de code couleur 4) de la Figure 1b est la suivante.

```
3 2 0 0 1 1 1 1 4
```

Nous considérons que le fichier est bien formé.

► **1.** Écrire la fonction `charger_puzzle` qui, étant un nom de fichier, charge le contenu de puzzle.

► **2.** Écrire la fonction `afficher_pieces` qui affiche en mode texte les pièces non encore posées.

► **3.** Écrire enfin la fonction `afficher` qui affiche en mode texte l'état courant du puzzle, c'est-à-dire, la grille avec les pièces déjà positionnées.

Par exemple, pour le puzzle de la Figure 1a, sur lequel seules les pièces rouge et bleue auraient été positionnées, l'affichage pourra être le suivant.

```
+------+
|222222|
|222222|
|....44|
|....44|
|444444|
|444444|
+------+
```

3. Résolution du puzzle

Nous allons maintenant écrire un programme qui permet de résoudre le puzzle. Ce programme doit permettre à l'utilisateur de saisir le numéro d'une pièce (indice dans le tableau de k pièces) et les coordonnées de la case où la poser, ou bien de retrier une pièce du puzzle. Attention, une pièce peut être positionnée de plusieurs manières, comme illustré ci-dessous pour la pièce rouge de la Figure 1b.

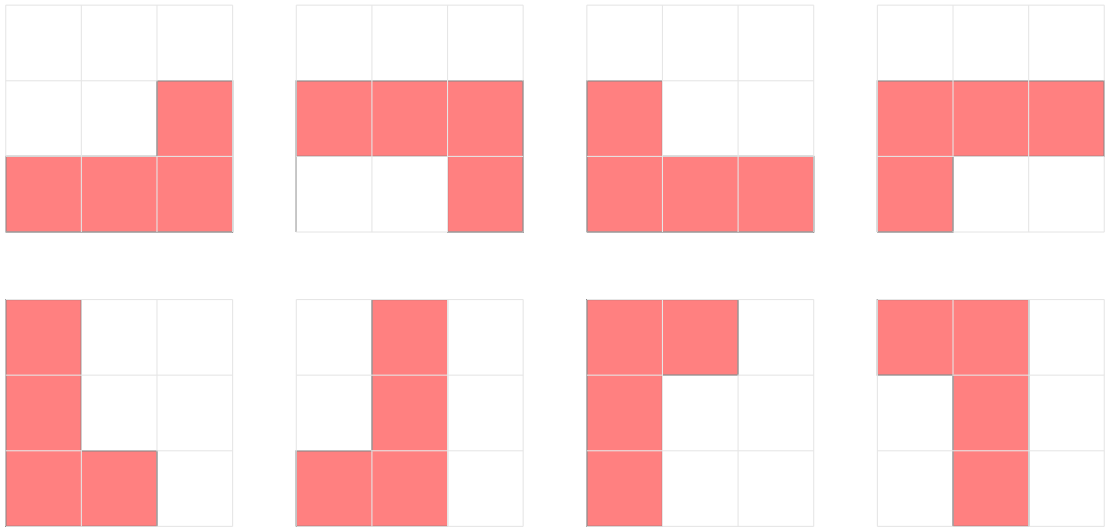


FIGURE 2 – Les 8 positions de la pièce rouge.

Nous souhaitons conserver une trace de tous les mouvements, c'est-à-dire, les pièces posées et retirées de la grille. Pour implanter cela, et à la demande générale, nous allons utiliser une liste chaînée contenant le mouvement (pose ou retrait), le numéro de la pièce, et les coordonnées.

► **1.** Définir la structure `mouvement`, permettant de manipuler une liste chaînée de mouvements.

► **2.** Écrire la fonction `ajouter_mouvement`, qui ajoute un mouvement en queue de la liste chaînée.

► **3.** Pour aider au déboguage, écrire une fonction `afficher_mouvements`.

► **4.** Écrire enfin la fonction `detruire_mouvements`.

Nous pouvons maintenant résoudre le puzzle.

► **5.** Écrire la fonction `est_recouvert`, qui test si la case de coordonnées (x, y) de la grille est recouverte par une pièce, et retourne le numéro de cette pièce, et -1 sinon.

► **6.** Écrire la fonction `test_position`, qui teste si une pièce peut être posée sur la grille aux coordonnées (x, y) passées en paramètre.

► **7.** Écrire les fonctions `symetrie`, qui applique une symétrie sur une pièce suivant un axe, et `rotation`, qui applique une rotation d'un quart de tour sur une pièce.

► **8.** Écrire la fonction `resoudre`, qui permet de résoudre une grille de puzzle.

► **9.** Illustrer ceci dans un programme principal.

4. Génération d'une image du puzzle

Une fois résolu, nous allons générer une image correpondant au puzzle. Pour cela, nous allons utiliser le même format que pour le CC1 (format PPM). Le format d'un tel fichier est le suivant :

— un nombre magique (P3, pour PPM ASCII),
— la largeur et la hauteur de l'image,
— la valeur maximale utilisée pour coder les couleurs (≤ 65536, représentable sur 16 bits),
— les données de l'image : un pixel est représenté par trois valeurs successives représentant les composantes rouge, verte et bleue du pixel (RGB).

Nous considérons qu'une case de la grille est représentée par un carré de taille $c \times c$ pixels, avec $c = 20$, par exemple.

► **1.** Écrire une fonction `creer_image`, qui génére l'image d'un puzzle résolu et qui la stocke dans un fichier dont le nom est passé en paramètre.

► **2.** Illustrer ceci dans un programme principal.

5. Exemple de fichier

```
7 7 11
3 2 0 1 1 1 1 0 1
3 3 0 1 0 1 1 1 1 0 0 2
3 2 1 1 1 1 0 1 3
2 3 0 1 0 1 1 1 4
2 3 1 0 1 1 1 0 5
3 2 1 1 1 1 1 0 6
2 3 0 1 1 1 1 0 7
3 3 0 0 1 1 1 1 1 0 0 8
2 3 1 0 1 0 1 1 9
3 3 0 0 1 0 1 1 1 1 0 10
2 2 1 1 1 1 11
```