RAPPORT JEU SOKOBAN

Minagnikpo Koccou-Gnankou Adoualou

Janvier 2023

Table des matières

1 Explication des programmes

1.1 Structure de la grille

Ce code définit une structure contenant un tableau de caractères de taille MAX-SIZE * MAX-SIZE nommée Grille. La structure Grille a cinq champs :
- grille : un tableau à deux dimensions qui contient les données de la grille
- nbLignes : un entier qui contient le nombre de lignes de la grille
- nbColonnes : un entier qui contient le nombre de colonnes de la grille
- posX : un entier qui contient la position x de la grille
- posY : un entier qui contient la position y de la grille

1.2 la fonction rechercherCoordonneesP

Ce code permet de rechercher les coordonnées d'un caractère 'P' dans une grille et de les stocker dans les variables posX et posY de la grille. Il parcourt chaque ligne et chaque colonne de la grille et, lorsqu'il trouve un caractère 'P', il stocke ses coordonnées dans les variables posX et posY.

1.3 la fonction recupererTaille

Ce code lit un fichier et détermine le nombre de lignes et de colonnes dans le fichier. Il stocke ces informations dans une structure de données appelée "Grille". Il ouvre le fichier en lecture et lit chaque ligne jusqu'à ce qu'il atteigne la fin du fichier. Pour chaque ligne, il compte le nombre de caractères et stocke le nombre maximum de caractères dans la variable "nbColonnes". Il compte également le nombre de lignes et stocke ce nombre dans la variable "nbLignes". Enfin, il stocke le nombre de lignes et de colonnes dans la structure de données "Grille".

1.4 la fonction chargerLabyrinthe

Ce code lit un fichier texte et remplit une grille avec le contenu du fichier. Il commence par récupérer la taille de la grille à partir du fichier, puis ouvre le fichier et lit chaque ligne. Chaque ligne est ensuite parcourue et les caractères sont ajoutés à la grille. Une fois que toutes les lignes ont été lues, le fichier est fermé.

1.5 la fonction afficherLabyrinthe

Ce code affiche un labyrinthe. La fonction prend en paramètre un pointeur vers une structure appelée Grille qui contient des informations sur le labyrinthe, notamment le nombre de lignes et de colonnes. Le code commence par une boucle for qui itère sur les lignes de la grille. Pour chaque ligne, une autre boucle for itère sur les colonnes. Si la ligne est la première ligne, le code affiche le caractère à l'emplacement donné sans espace. Sinon, le code affiche le caractère avec un espace devant. Une fois les caractères de la ligne affichés, le code passe à la ligne suivante et répète le processus jusqu'à ce que toutes les lignes aient été affichées.

1.6 la fonction deplacerPersonnage

Ce code permet de déplacer un personnage sur une grille. Il prend en paramètre un pointeur vers une grille et une direction (haut, bas, gauche ou droite).

Le code commence par calculer la nouvelle position du personnage en fonction de la direction donnée. Ensuite, il vérifie si la nouvelle position est dans les limites de la grille.

Si ce n'est pas le cas, un message est affiché et la fonction se termine. Sinon, il vérifie si la nouvelle position contient un mur. Si c'est le cas, un message est affiché et la fonction se termine.

Ensuite, le code vérifie si la nouvelle position contient une caisse. Si c'est le cas, le code calcule la position de la caisse et vérifie si elle est dans les limites de la grille, si elle contient un mur ou une autre caisse.

Si une des conditions est remplie, un message est affiché et la fonction se termine. Sinon, le code déplace la caisse si elle est sur une case d'arrivée ou la laisse à sa position si ce n'est pas le cas.

Enfin, le code déplace le personnage et met à jour la position du personnage dans la grille.

1.7 la fonction verifierGagne

Cette fonction vérifie si le jeu est gagné. Elle prend une grille en paramètre et vérifie chaque élément de la grille pour voir si tous les caisses 'C' déplacées par le personnage sont arrivées a destination 'I' et renvoie 1 pour signifier que le jeu est gagné et 0 sinon.

1.8 la fonction afficherScore

Cette fonction calcule le score du joueur pour un niveau donné. La fonction commence par initialiser le score à 0. Elle calcule ensuite le nombre de secondes écoulées depuis le début de la partie jusqu'à sa fin. Elle calcule également le nombre de caisses présentes sur la grille. Ensuite, elle calcule le score en fonction du nombre de secondes, du nombre de mouvements et du nombre de caisses bien placées. Enfin, elle affiche le score du joueur.

1.9 Fonctionnement du main()

Pour terminer nous allons vous présenter comment fonctionne le main() : Ce code permet de lancer un jeu de labyrinthe. Il commence par déclarer une grille, un numéro de niveau, un nom de fichier, un début et une fin de temps, ainsi qu'un nombre de mouvements et de caisses bien placées. Ensuite, une boucle dowhile est exécutée pour chaque niveau, qui commence par charger le labyrinthe à partir du fichier spécifié, puis rechercher les coordonnées du joueur. Ensuite, le jeu commence et le temps de début et de fin sont enregistrés. Le nombre de mouvements est ensuite incrémenté et le jeu est vérifié pour voir si le joueur a gagné. Si c'est le cas, le nombre de caisses bien placées est incrémenté. Enfin, le score est affiché et le joueur est invité à choisir un niveau.

2 Fonctionnement du jeu

Notre code permet de jouer au jeu de **Sokoban**, où l'objectif est de déplacer un personnage représenté par un 'P' à travers un labyrinthe et de placer toutes les caisses représentées par des 'C' sur les cases d'arrivées représentées par des 'I',quand cela se produit la position est changé par un 'O'.Pour écrire le code, nous avons commencé par inclure les bibliothèques nécessaires au fonctionnement du programme, comme **stdlib.h**,**stdio.h** et **time.h**.Nous avons définit une structure de grille qui stocke les informations sur le labyrinthe et des fonctions pour le déroulement du jeu, comme le **chargement du labyrinthe** à partir d'un fichier, **l'affichage du labyrinthe**, **le déplacement du personnage** et **la vérification de la fin du jeu**. Nous avons également une fonction qui affiche le score à la fin du jeu. Pour lancer le jeu, nous avons écrit une fonction principale qui initialise les variables nécessaires, puis démarre une boucle qui charge le labyrinthe à partir du fichier contenant les différents labyrinthe, affiche le labyrinthe et demande à l'utilisateur de choisir une direction **('h' pour haut, 'b' pour bas, 'g' pour gauche ,'d' pour droite)** pour déplacer le personnage et pour quitter le jeu appuyer sur la touche 'q'. Si le déplacement du personnage n'est pas possible, un message s'affiche et l'utilisateur peut essayer une autre direction. Si le jeu est gagné (toutes les cases 'C' ont été placées sur les cases d'arrivée 'I'), un message "vous avez gagné" s'affiche et l'utilisateur doit appuyer sur 'q' pour quitter le niveau, à la suite de cela grâce à la bibliothèque time.h et la fonction **afficherScore** son temps et son score seront affichés.Après cela, un message demande à l'utilisateur à quel niveau il souhaite jouer, (sachant que le niveau va de 1 à 10 et que la difficulté avance de manière croissante). Pour quitter complètement le jeu l'utilisateur poura entrer la commande 'e'.

3 Makefile

-Pour compiler : **Make** -Pour exécuter : **./main** -Pour effacer les exécutables et les fichiers.o **make clean**