Rapport sur le Projet FreeCell en C

Réaliser par : Yassine Ben Akki encadrer par : Mr. Abdelwahab Naji

Le projet FreeCell en C vise à implémenter le jeu de cartes FreeCell en utilisant des structures de données et des fonctionnalités du langage C. Le code est organisé en plusieurs fichiers pour faciliter la gestion et la compréhension du programme.

I. Structures de Données

Le projet utilise les structures suivantes :

Structure de donnée (Card) : Cette structure représente une carte dans le jeu, avec des membres pour le rang (rank), le type (suit), et des pointeurs vers les cartes précédentes et suivantes dans une pile.

Zones de Jeu (Zone1, Zone2, Zone3) : Ces tableaux de pointeurs vers le type Card représentent les différentes zones de jeu où les cartes sont empilées. Ils sont utilisés pour stocker les cartes selon les règles du jeu de Solitaire.

Tableaux de Comptage (zone1, zone2, zone3) : Ces tableaux comptent le nombre de cartes présentes dans chaque zone de jeu.

II. Initialisation du Jeu

Les fonctions utilisées :

1.initializeVariables : Initialise les variables de jeu, y compris les zones de jeu et les tableaux de comptage.

2.createCard : Crée une nouvelle carte avec un rang et une couleur spécifiée.
3.shuffleDeck : Mélange le jeu de cartes.

4.generateFullDeck : Génère un tableau de pointeur de Card qui contient toutes les 52 cartes dispersées d'une manière aléatoire.

5.Empiler et Depiler : Ajoutent ou enlèvent une carte d'une pile.

6.Addcardstozone1 : c'est la fonction principale dans cette partie il utilise tous ces fonctions, il permet de déplacer toutes les 52 cartes retourner par generateFullDeck dans Zone1 suivant les règles de jeu, Zone1 est un tableau de pointeurs sur le type

Card qui contient 8 cases chaque case se comporte comme une pile pour stocker les cartes en utilisant les fonctions Empiler et Depiler.

III. Affichage du Jeu

Les fonctions utilisées :

1.Print_Card : affichage d'une carte par son rang et son type.

2. print_cards_zone1 : affichage des cartes du Zone1 qui est composé de 8 colonnes, chaque colonne a la forme d'une pile.

3. print_zone2 : affichage des cartes du Zone2 qui est composé de 4 colonnes, chaque colonne a la possibilité de contenir une seule carte.

4. print_Zone3 : affichage des cartes du Zone3 qui est composé de 4 colonnes, chaque colonne a la forme d'une pile il peut contenir 13 cartes de même type mais dans l'affichage, on affiche seulement la dernière carte empiler.

Dans la fonction **Print_Card** j'ai appelé une autre fonction **SetCardColorsBySuit** qui change le fond de chaque carte en blanc et change la couleur d'écriture en noir ou blanc selon la 'suit' de chaque carte

La mise à jour de l'état de console est effectuée à l'aide de « **system("cls") »** qui supprime l'état de console pour laisser l'espace a le nouvel état.

IV. Mouvements de Cartes :

Les fonctions utilisées :

1.ismove_valid : Vérifie si le mouvement d'une carte source vers une carte destination est valide. Les règles du jeu, telles que les cartes devant être en ordre décroissant de rang et de couleurs opposées, sont respectées.

2. move_Card_inside_Zone1 : déplace une carte à l'intérieure de la zone1 si les conditions de déplacement sont vérifiées.

3. move_cards_zone1to2 : déplace une carte de la zone1 vers la zone2 en respectant les règles de jeu.

4. move_cards_zone2to1 : déplace une carte de la zone2 vers la zone1 en respectant les règles de jeu.

5. is_move_valid_1or2to3 : vérifié si le déplacement d'une carte de la zone1 ou 2 vers la zone3 est valable, en respectant les règles de jeu.

6. move_zone1to3 et move_zone2to3 : déplace une carte de la zone1 ou 2 vers la zone3 qui est notre objectif dès le début, en respectant les règles de jeu.

V. Réinitialisation du Jeu

La fonction **Addcardstozone1** permet de réinitialiser le jeu , il permet de vider les zones 1,2 et 3 et de remplier la zone1 par les 52 cartes à nouveau.

VI. Conditions de Victoire :

La fonction **win_check** vérifie si les 4 pile dans la zone3 sont pleines par les 52 cartes en respectant les règles et si les zones 2 et 3 sont vides.

VII. Interface Utilisateur

Le programme propose une interface utilisateur simple en ligne de commande. L'utilisateur peut effectuer divers mouvements en choisissant parmi les options présentées à l'écran.

Ce jeu est 'user-friendly' grâce à l'utilisation de la fonction **getValidChoice** , qui assure une interaction conviviale en garantissant que l'utilisateur ne peut entrer que des choix valides, contribuant ainsi à une expérience de jeu plus fluide et intuitive.

VIII. Conclusion

En conclusion, ce projet en langage C illustre une utilisation efficace des structures de données pour créer le jeu FreeCell. L'approche modulaire avec des fonctions facilite la maintenance du code et permet d'ajouter de nouvelles fonctionnalités plus facilement. En respectant les règles du FreeCell et en offrant une interface utilisateur conviviale grâce à des fonctions comme ‘getValidChoice’, le jeu devient fluide et agréable à jouer. De plus, les fonctions de mouvement et de vérification s'assurent que les règles du jeu sont suivies et évitent les mouvements invalides. En résumé, ce projet réussit à équilibrer la complexité du code, la facilité d'utilisation et le respect des règles du jeu, offrant ainsi une version numérique attrayante du célèbre FreeCell.