Raport du projet de C : Nice Craps

Nom : Tognetti
Prenom : Yohann
N° etu : 21704017

Tout d'abord, pour commencer le projet avec une approche méthodique, j'ai commencé à créer les structures et des fonctions qui vont avec celle-ci pour gérer chaque paramètre de mes structures. Cela va me permettre par la suite de pouvoir réaliser le projet en ayant déjà toutes les fonctions qui me semblé nécessaire pour la réalisation du jeu Nice Craps. Par la suite il me suffisait, selon le lancer d'appliquer une fonction ou une autre selon le résultat du lancer de dés. Mais durant le projet je me suis aperçu qu'il me manquait beaucoup de fonction ou variable que je n'avais pas créée au début, j'ai donc dû continuer en les rajoutant tout au long du développement.

Repartition du code:

Au niveau de la répartition du code dans les fichiers, je me suis contenté de mettre l'affichage des menus principaux dans le "main.c" avec la fonction main qui gère le bon déroulement de la partie et appelle des 3 différents menus principaux durant la partie. Puis les fonctions qui traitent ces menus et qui permettent de gérer les différentes structures dans le fichier "craps.c"". Ensuite dans le fichier "craps.h" se trouve la définition des différentes structures utiliser et la different fonction qui sont dans "craps.c" liée d'une explication de l'utilité de la fonction en question.

La structure "player":

Pour ce projet, j'ai donc commencé par créer la structure "player" qui contient toutes les informations d'un joueur comme son pseudonyme "name" qui est alloué dynamiquement, son argent "money", ces mises sur PassLine et sur Don't PassLine nommé respectivement "Pass" et "dontPass", et deux booléens qui ont pour but de stocker si le résultat a déjà été mis a jour ou non (doubler ou diviser par deux) durant la phase actuelle, ces booléens sont utilisés uniquement à partir du moment où il y a une "suspension".Cette structure est utilisée sous forme de pointeur et chaque joueur est créé par la fonction "playerInit()" qui alloue dynamiquement l'espace de cette structure a l'aide d'un malloc.

Ces fonction:

Une fois cette structure initialisée, elle est construite a l'aide du joueur des informations données par le joueur. Celles-ci sont appris par la structure à l'aide de deux fonctions:
"playerSetName" pour mettre son pseudonyme.
"playerSetMoney" pour mettre la somme d'argent sélectionné.
Puis, pour le parier en phase 1 un joueur peut mettre la somme voulue sur PassLine et ou Don't PassLine. Pour ce faire, j'ai utilisé deux fonctions qui ont toutes deux le même fonctionnement:"setPariePass" et "setParieDontPass" qui retire l'argent pour miser sur PassLine ou Don't PassLine, je suis parti du principe qu'une même personne peut miser sur les deux en même temps du tout que le joueur possède assez d'argent pour cela.
Ensuite la fonction "tirage" permet de tirer les dés et donc de passé a la phase 2 (sauf si le tour se termine). Au début de la deuxième phase le joueur peut augmenter ça mise sur PassLine à l'aide de la fonction "augmenterPass" et diminuer ou annuler la mise sur Don't PassLine avec "diminuerDontPass".
Puis, si on arrive à la phase 3, cela veut dire que le point est suspendu pour la première fois, le joueur peut donc doubler ça mise que ce soit sur PassLine ou Don't PassLine avec la même fonction pour Pass que pour la phase 2 et la fonction "AugmenterDontPass" pour Don't PassLine mais cette fois la valeur est mise automatiquement à deux fois la valeur misé sur PassLine ou Don't PassLine selon ce que le joueur veut doubler.
Pour les phases suivantes, les seules modifications acceptées sont de diviser par deux les mises, en

utilisant le même principe que pour la première suspension, en mettant comme valeurs en entrer la moitié de la mise en négative et cela permet d'enlever la moitié.

La structure Liste:

Maintenant que j'ai expliqué comment fonctionner mon code pour un joueur, le principe est le même avec plusieurs joueurs, je vais donc expliquer comment j'ai stocké les différents joueurs.
C'est alors que j'ai commencé une seconde structure qui était un tableau dynamique que je réallouais selon le nombre de joueurs, mais je me suis très vite aperçu qu'il allait avoir beaucoup trop de reallocation de la mémoire.
J'ai donc décidé de changer de structure pour contrôler tous les joueurs. Je suis donc parti sur une liste chainée qui me semble beaucoup plus efficace. Pour ce faire, j'ai réutilisé la liste chainée que nous avons créée lors du Tp n°4 ou j'ai modifié la donnée pour quel contient en des pointeurs sur les joueurs. La liste est constituée de deux structures, la "Liste" et "element".
La liste contient tout d'abord un pointeur vers le premier élément que j'ai nommé "premier", puis toutes les valeurs utiles pour le bon déroulement de jeu: Le nombre de jouer "nbPlayer", la valeur du point "Point" (pour la deuxième phase), le numéro de la phase actuel "Phase" pour avoir un repère sur la partie et un pointeur vers le tableau des 10 meilleurs gains "bGain".
Tandis que, la structure élément est constituée d'uniquement 2 éléments, d'un joueur "P" et de l'adresse de l'élément suivant".

Ces fonction:

La première fonction de celle-ci est "afficheListe" et permet d'afficher tous les joueurs qu'elle possède en la parcourant et affiche l'indice de chacun qui permet de sélectionner un joueur ou un autre. Au niveau du fonctionnement, quand un joueur crée un profil avec son pseudonyme et ça somme d'argent initiale, celui-ci est ajouter à la fin de liste avec la fonction "pushListe". Cette fonction lui alloue une structure element de manière dynamique et initialise son pointeur suivant à NULL car il est le dernier element tandis que le pointeur de l'element qui était le dernier avant l'ajout de celui-ci se fait redéfinir son pointeur vers cette element.
Pour la suppression d'un joueur, la fonction "supElt" qui prend en paramètre la liste et l'indice de l'element à supprimer (les joueurs et l'indice sont affiché à l'écran). Cette fonction change le pointeur de l'element précédent et point vers le suivant soit l'element à supprimer n'est plus dans la liste chainée il ne reste plus qu'à libérer la mémoire avec la fonction "freeElt" qui libère chaque espace prit par le joueur.
Pour gérer les joueurs individuellement la fonction "selectPlayer" permet à l'aide de l'indice de renvoyer l'adresse du joueur sélectionné, cela permet donc d'appliquer les fonctions vues précédemment sur les mises du joueur.
La fonction "resetModifVariable" permet de remettre à 1 les booléens que possèdent chaque joueur en jeu entre deux tirages de dés.
Ensuite, la dernière fonction de cette structure est "afficheTirage_ActualiseGain" qui permet le passage d'une phase à une autre. La première action de cette fonction est le tirage des dés à l'aide de la fonction "tirage". Une fois le tirage effectué il a 2 possibilités que gère cette même fonction:

Pour la Phase 1:

-2 : Les joueurs qui ont misé sur PassLine perdent leurs mises et ce qui ont jouer sur Don't PassLine récupère le double de leurs mises.

-7 : Les joueurs qui ont misé sur PassLine récupère le double de leurs mises et eux sur Don't PassLine perde leurs mises.

-12 : Les joueurs qui ont misé sur PassLine perdent leurs mises et ce qui ont jouer sur Don't PassLine recupere leurs mises à l'identique, leur coup est nul.

-le reste: Le point est fait, la partie passe à la phase suivante de la variable "Point" de la liste retient la valeur du tirage.

Pour la phase 2 jusqu'a la fin:

-Le point: Les joueurs qui ont misé sur PassLine perdent leurs mises et ce qui ont jouer sur Don't PassLine récupère le double de leurs mises.

-7 : Les joueur qui on misé sur Don't PassLine recupere le double de leurs mise et eux sur PassLine perde leurs mise.

-Le reste: le résultat est suspendu, il va y avoir un autre lancer et certaines mises pourront bouger.

La structure des meilleur gain:

La dernière structure implementer au projet a été bestGain qui a deux attribue, un nom "name" et le gain". Durant tout le programme cette structure est utilisée comme un pointeur qui est alloué dynamiquement et peut stocker les10 plus haut gain associé au nom du contributeur.
Elle possède une fonction "actualiseBestGain" qui permet d'ajouter si nécessaire un gain au tableau si le gain fait partie des 10 plus haut et est appelé à chaque fin de tour par "afficheTirage_ActualiseGain".
De plus elle est sauvegardée dans le fichier "bestGain.txt" qui est ouvert en lecture au démarrage du jeu pour récupérer les sauvegardes et le fichier est recrié à la fin de chaque tour.

Les menu:

Les menus sont un element essentiel pour la jouabilité, il permet la navigation parmi les different paramètres énoncés précédemment. Pour l'interface, j'ai utilisé principalement 3 menus qui permet chacun de faire des choix:

-Le menu principale:

```
<<<Nice Craps>>>>
Selectionner en entrant le numero correspondant
1 : Ajouter un joueur
2 : Afficher les joueur
3 : Supprimer un joueur
4 : Sauvegarder un joueur
5 : Charger un joueur
6 : Commencer le tour
7 : Tableau des scores
8 : Quiter le jeu
Votre choix :
```

-Le menu des phase:

```
Phase: 1
Selectionner en entrant le numero correspondant
1 : miser ou modifier les mise
2 : Lancer les des
Votre choix :
```

-Le menu des parie

```
Selectionner le joueur en entrant le numero correspondant
1 : nom: player1 ; argent: 100
2 : nom: toto ; argent: 64
3 : nom: titi ; argent: 312
4 : pour retourner au menu precedent
Votre choix :
```

Chacun de ces 3 menus a une fonction qui traite le choix effectuer. Dans certains de ces traitements il y a à des sous-menus qui sont inclus comme pour les actions sur les mises ou pour les sauvegardes qui permettent au joueur de choisir exactement l'action qu'il a l'intention d'effectuer dans le menu sélectionné.

La sauvegarde des joueur:

Celle-ci fonctionne a l'aide de 4 fonctions:

-la fonction "savePlayer" permet de sauvegarder le nom et la somme d'argent d'un joueur sélectionner au préalable quelle recopie a la fin du fichier "playerSave.txt"

-la fonction "afficheSavePlayer" affiche la liste des joueurs sauvegarder et leur numéro correspondant qui permet et l'utilisateur de sélectionner parmi la liste.

-la fonction "chargePlayer" permet de récupérer les données du joueur sélectionné à de l'ajouter à la liste des joueurs en jeu la sauvegarde est par la suite supprimée.

-la fonction "suppSave" permet de supprimer sauvegarder sélectionner au préalable par le joueur. Pour ce faire je crée un fichier tampon dans lequel je recopie tout de fichier "playerSave.txt" sauf le joueur que je veux supprimer de la sauvegarde puis je renomme ce fichier en "playerSave.txt" et remplace l'ancien. Cette fonction est appelée par "chargePlayer" et supprime donc le joueur charger cela permet au joueur de soit sauvegarder de nouveau après ça parti ou de définitivement supprimer ce joueur.

Conclusion.

Ce projet m'a permis de me familiariser avec le langage de programmation C, d'apprendre plus concrètement le fonctionnement de la mémoire d'un programme et de mettre plus de commentaire car à des durées d'une semaine d'intervalle je devais essayer de comprendre le code de la semaine précédent. Je trouve ce sujet de projet très pédagogique car il nous force à utiliser des pointeurs et de gérer la mémoire qui n'est pas présente dans les langages que nous avons vue jusque-là (python, javascripte,php et scheme). Néanmoins, j'aurais préféré une liberté plus importante au niveau du nombre de fichiers pour rajouter un fichier spécialement pour les menus (menu.c et menu.h) ou j'aurais pu placer le traitement dans tous les menues.