Nom : Houssine
Prénom :Youssef
n° Étudiant :21601530

Rapport Projet C

I/ Introduction.

Le projet de cette année est un jeu de Craps revisité , avec des règles différentes du vrai jeu mais avec le même système que le vrai.
Nous avons eu le sujet aux alentours de la mi Avril ce qui nous a laissé quelques temps pour le faire malgré les examens de fin de semestre.
Dans ce rapport je vais essayer de décrire la méthode que j'ai utilisé, et d'expliquer les fonctions principales de mon jeu. En même temps je vais essayer de parler des difficultés rencontrés lors de la réalisation de celui-ci.

II/ Comment j'ai fait ?

Avant même d'avoir écrit quoi que ce soit, j'ai pris le temps de comprendre le sujet. J'ai donc commencé par faire un plan de travail sur un document texte afin d'avoir un fil conducteur et ne pas me perdre.
Ma méthode était donc la suivante :

1. Compréhension du sujet.
2. Faire une To Do list afin de ne pas oublier d'éléments
3. Commencer le code.

III/ Étapes du code et description des fonctions de celui-ci.

Il faut tout d'abord savoir que j'ai utillisé Visual Studio afin de faire ce projet, tout simplement car il s'agit du compilateur que l'on utilise en tp. Mais a la fin je me suis assuré que mon code est compilable avec Dev c++.

Concernant le code : Tout d'abord j'ai commencé par créer les 3 fichiers , craps,c craps,h et main,c
Craps,c contiendra mon code
craps,h contiendra les prototypes.
Main,c et bien c'est tout simplement mon main, c'est ici que tout se relie.

Ensuite j'ai donc pris ma to do list , et j'ai commencé à réellement écrire du code.
J'ai tout d'abord crée ma fonction nombreJoueurs() ; qui est la fonction qui va me permettre de demander combien de personnes vont jouer et me renvoyer un int. Vu que c'est un jeu console je vais beaucoup utiliser le printf ainsi que le scanf afin de demander au joueur d'entrer des informations.

Une fois que j'ai obtenu mon nombre de joueurs , je vais créer ma liste de joueurs avec ma structure joueur. Qui est définie dans le craps,h et qui contient les informations que je veux savoir sur les joueurs.
J'ai ainsi crée la fonction *initialisation(int nb) ; qui prend en paramètres le nombre de joueurs désirés et qui renvoi un pointeur sur structure. Au sein de cette fonction je crée un tableau de structure et je lui alloue (nombreJoueurs) fois la taille d'un joueur. Tout au long du projet joueur sans « s » est ma structure joueur et joueurs avec « s » est mon tableau de joueurs.
Je demande donc logiquement pour chaque joueur son nom et son budget , le budget étant la somme maximal qu'il a sur lui et qu'il pourra jouer.

Un schéma que je réutilise très souvent dans mon projet est le boucle for qui boucle de i=0 à i = nbJoueurs. Cette boucle me permet de parcourir tout les joueurs et de tous les « tester » en fonction de ce que je veux leur faire faire.

Ensuite j'ai séparé les types de mises en 3 fonctions.

1. `*mises(int nb,struct joueur *joueurs)`
2. `*mise_suspens(int suspens, int nb , struct joueur *joueurs)`
3. `mise2(int suspens, int nb, struct joueur *joueurs)`

Description : **mises :** C'est la fonction de ma mise de base, c'est elle qui à chaque tour va demander le type de mise que le joueur veut faire , et la somme initiale qu'il va miser. Le type de mise est géré par un int , et à chaque valeurs correspond un type de mise. 0 rien 1 pass 2 don't pass. Ce qui me permettra ensuite pour chaque joueur de tester ce qu'il mise avec une condition. Il faut aussi savoir que mises s'applique uniquement aux joueurs dont le pStatus(statut de joueur) est de 1. Autrement le joueur ne joue pas car sois il est en banqueroute , sois il ne mise rien sur ce tour.
Donc en fonction des choix je modifie les valeurs changées du joueur concerné , directement dans sa structure.

mise_suspens : Mise suspens est la fonction qui permet aux joueurs misant pass ou don't pass de doubler leur mise au tour de suspens 1, ou de la diviser par 2 sur les autres suspensions.
Cette fonction prend un int en plus pour savoir ou est-ce que nous en sommes dans la partie, afin que cette fonction ne demande pas à chaque étape si les joueurs veulent modifier leur mise.

Mise2 : Est la fonction qui demande aux joueurs si ils veulent augmenter leur mise si ils misent sur pass ou diminuer ou retirer si ils misent don't pass.
Ces 3 fonctions suivent le même schéma, je demande ce que le joueur veut faire avec un scanf et je fais ensuite des conditions pour effectuer ce que je veux aux joueurs répondant aux conditions .

Fonction **lanceDes() ;**

Maintenant je vais expliquer brièvement ma fonction lanceDes() : C'est une fonction permettant de lancer les dés et de renvoyer la somme des deux dés. Je me contente de faire un rand() % 7 et de le refaire si jamais je suis tombé sur un multiple de 7. J'avais un doute sur l'équiprobabilité de cette fonction j'ai donc demandé à mon prof de probabilités qui m'a confirmé que c'était équiprobable. Ensuite je renvoi tout simplement la valeur obtenue.

Cette fonction m'a posé quelques problèmes car je comprenais pas comment fonctionnait le random et que au début mon random renvoyait souvent la même chose. De plus dans le cours il n'y avait pas de slide parlant du random en c. J'ai réussi à régler le problème grâce à srand(time(NULL)).

Fonction principale : `*point(int nb , struct joueur *joueurs).`

Cette fonction est tout bonnement celle qui fait tourner le jeu. Celle qui gère ma phase 1 et ma phase 2 de la partie. Elle est longue mais très simple au final , elle ne m'a rien demandé de particulier à savoir. Il s'agit d'une succession de if de for et un while. Et comme partout ailleurs je parcours chaque joueur pour voir ce qu'il mise et ce qu'il gagne ou perd en fonction de ce que les dés diront. Ensuit si j'entre en phase 2 j'entre dans la boucle while , et tant que je ne refais pas le point précédemment déclaré et bien je continue à tirer les dés. Ensuite en fonction de l'issue finale je donne ou retire les mises de mes joueurs.

Par le suite je vais présenter mes petites fonctions annexes :

Display : permet d'afficher les informations de mes joueurs

Score : Permet de trouver le max , et de le renvoyer. Le score fonctionne de la manière suivant dans mon code : Si un joueur gagne c'est à dire qu'il mise correctement , il prend +1 si il perd il prend -1. j'ai procédé comme cela car si on se base sur les cagnottes, c'est inégal. Il suffit qu'un joueur mise 1 000 000 pour qu'il soit systématiquement premier, et ce n'est en aucun cas démonstratif de son niveau de jeu. Donc j'ai opté pour un système de highscore qui valorise les meilleurs joueurs , ceux qui gagnent le plus de fois , et non ceux qui gagnent le plus.

Testj : Cette fonction permet de tester si il reste au moins un joueur en jeu afin que le jeu s'arrête quand il n'y en a plus.

Et pour finir :

play() : C'est la fonction qui s'exécute dans le main, elle regroupe toutes les autres et les assemble correctement afin que tout puisse fonctionner dans l'ordre. Et elle permet aussi d'arrêter les jeu en fin de partie.

IV/ Difficultés rencontrées en général.

1. Les règles.
   La première grosse difficulté qui m'est apparue est en fait la compréhension des règles, de base le craps est un jeu que je ne connais pas, de plus je ne suis que très peu familier avec les casinos donc je n'ai aucune idée de comme cela fonctionne. De plus d'après moi il a manqué de précisions dans le sujet. Mais d'un autre côté cela a laissé libre cour à notre esprit d'initiative afin de prendre les bonnes décisions.

2. La suppression de joueurs : j'ai voulu essayer de supprimer les joueurs quittant la partie de mon tableau de structures. Mais je n'ai malheureusement pas réussi. J'y ai sûrement passé plusieurs heures , a faire plusieurs tentatives , mais rien ne fonctionne j'avais toujours une erreur windows qui ressortait. Je pense que cela venait du fait que je n'arrivais pas à sélectionner correctement le joueur à supprimer et donc le realloc ne rangeait pas correctement mes structures. J'ai quand même laissé mon bout de code en commentaire, sait on jamais.

V/Conclusion :

Pour conclure, ce projet m'a plu. Tout simplement car je suis partisan des projets pour apprendre car le fait de galérer sur certains points nous permet de mieux apprendre. Et c'est aussi enrichissant sur les notions d'organisation et de gestion.