30/11/14

GUEDOUAR Wajdi
OKROU Poda
THIEULART Godefroy

Dépôt git : https://github.com/WajdiG/projet-outils

PROJET BLACKJACK

L2 SPI Semestre 3

1 _ Introduction :

L'objectif de ce projet est de concevoir un jeu de BlackJack en collaboration. Nous sommes un groupe au nombre de trois : Guedouar Wajdi, OKROU Poda et THIEULART Godefroy. Dans le cadre de ce projet, nous devions réaliser le code permettant de jouer au BlackJack, commenter ce code en utilisant Doxygen, rédiger un makefile permettant de compiler le code, effectuer des débogages et effectuer des tests.

Pour rappel, le BlackJack est un jeu de cartes dans lequel le joueur affronte la banque et doit avoir une main d'une valeur supérieure à celle de la banque sans toutefois dépasser 21 points. Ce jeu peut se jouer à plusieurs mais nous n'avons pas ajouter cette fonctionnalité dans le code.

2 _ Organisation :

Rédaction du code : GUEDOUAR & THIEULART
Génération du Doxyfile : OKROU
Rédaction du makefile : GUEDOUAR
Débogage : THIEULART
Test : THIEULART
Compte-Rendu : GUEDOUAR

3 _ Objectif principal du jeu

Lorsque le jeu commence, 2 cartes sont tirées pour la banque, dont une cachée (qui restera invisible jusqu'à la fin de la partie) et 2 cartes pour le joueur. Une fois les cartes distribuées, le joueur peut alors choisir de tirer une carte supplémentaire ou de passer son tour. L'ordinateur (la banque) peut lui aussi tirer une carte ou s'abstenir, il demandera une carte seulement si son score est inférieur à 17 ou si le joueur à cesser de tirer une carte et que son score est supérieur à celui de la banque. La partie se poursuit jusqu'à ce que l'un des deux atteigne ou dépasse 21, ou que les deux décident de ne plus tirer de carte.

4 _ Fonctions et structures de données

a) variables globales

Une variable « nb_as_joueur » qui comptabilisera le nombre d'as dans la main du joueur et une variable « nb_as_banque » qui comptabilisera le nombre d'as dans la main de la banque.

Un tableau « cartes » de type short contenant 52 valeurs représentant les 52 cartes du jeu, si la carte n'a pas encore été tirée, la valeur correspondante dans le tableau vaut 0, si elle a été tirée par le joueur elle vaut 2, si elle a été tirée par la banque et qu'elle est visible elle vaut 1, si elle est cachée elle vaut 3. banque. Nous définissons donc les constantes suivantes : LIBRE vaut 0, BANQUE vaut 1, JOUEUR vaut 2 et BANQUE_CACHEE vaut 4.

Afin de mieux nous y retrouver dans le tableau « cartes », nous définissons les constantes suivantes qui permettront de distinguer les couleurs dans l'ordre coeurs,carreaux, piques, trèfles : DEB_COEURS vaut 0, FIN_COEURS vaut 12, DEB_CARREAUX vaut 13, FIN_CARREAUX vaut 25, DEB_PIQUES vaut 26, FIN_PIQUES vaut 38, DEB_TREFLES vaut 39, FIN_TREFLES vaut 51.

b) fonctions

La fonction « tirer_carte » renvoie un entier de type short prenant en paramètre un entier de type short représentant le joueur qui tire la carte. Cette fonction permet de tirer aléatoirement une carte représentée par un entier allant de 0 à 51 et associe le joueur passé en paramètre à cette carte dans le tableau « cartes ».

La fonction « afficher_carte » est de type void et prend en paramètre un entier de type short, cette entier correspond à une valeur du tableau « cartes ». Cette fonction permet d'afficher à l'écran la valeur de la carte, sa couleur et son type (cœurs, carreaux...), pour cela on se servira des

constantes définies précédemment.

La fonction « afficher_mains » est de type void et prend en paramètre un entier de type short représentant le joueur dont on souhaite afficher la main. Cette fonction parcourt le tableau « cartes » et récupère toutes les cartes correspondantes au joueur passé en paramètre. Puis on utilise la fonction « afficher_carte ».

NB : cette fonction n'affichera jamais la carte cachée de la banque car le code correspondant dans le tableau « cartes » n'est pas le même que celui de la banque.

La fonction « afficher_mains_cachee » est de type void et ne prend aucun paramètre. Elle affichera la main de la banque, carte cachée incluse.

La fonction « donner_valeur_carte » est une fonction de type short et prend deux paramètres entiers : short joueur représentant le joueur possédant la carte et short carte représentant la valeur de la carte. Cette fonction devra retourner la valeur de la carte et devra incrémenter nb_as_joueur ou nb_as_banque en fonction du joueur passé en paramètres si la carte passée en paramètre est un as.

La fonction « evaluer_score » est une fonction de type short qui prend trois paramètres : short joueur qui est un entier représentant le joueur concerné, short carte_recue la carte venant d'être tirée par le joueur passé en paramètres et short *score pointeur sur l'entier score représentant le score du joueur passé en paramètres. Cette fonction a pour but de mettre à jour le score du joueur passé en paramètre en fonction de la carte passée en paramètre.

5 _ Test et cas d'erreur

Nous avons tout d'abord veillé à bien initialiser les variables de nos fonctions car nous avons plusieurs fois été confronté à des cas où, n'ayant pas initialisé ces variables, les valeurs étaient totalement erronées.

Nous avons également borné les choix du joueur afin qu'il ne puisse pas entrer de valeurs autres que celles voulues.

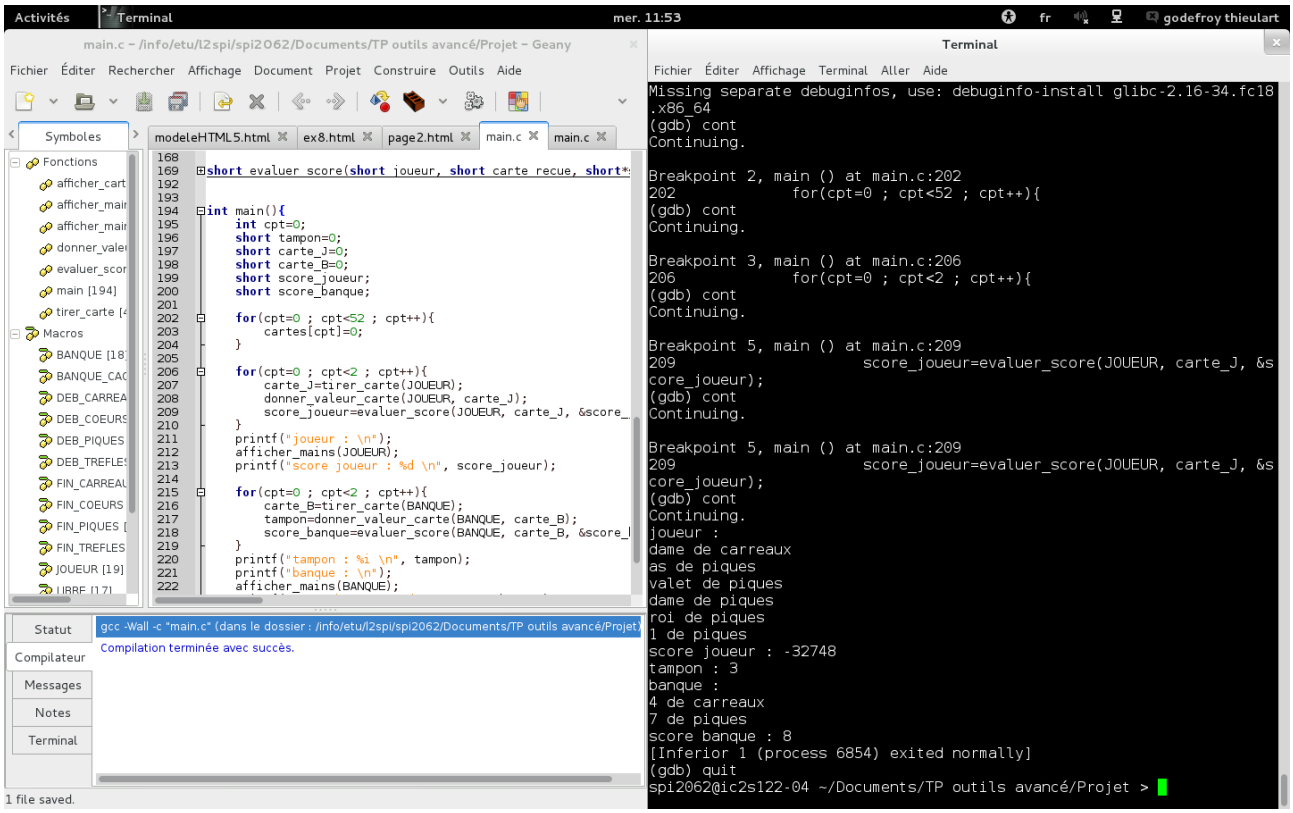
Nous avons également réalisé un fichier à part qui contient le code ainsi que tout les tests nécessaires, nous avons choisi de créer un nouveau fichier, un nouveau makefile et un nouvel exécutable afin de ne pas encombrer le code original et de permettre une meilleure lisibilité de celui-ci.

6_ Débogage

Les débogages nous ont été relativement peu utiles, toutefois nous avons dû y faire appel lors de nos premiers tests. En effet, il s'est avéré lors de la première exécution du programme que les scores étaient totalement illogiques avec la partie en cours.

30/11/14

Voici donc un exemple de débogage, dans ce cas, nous cherchions à déterminer l'origine des valeurs erronées des scores de la banque et du joueur.



7 _ Conclusion :

Le programme dans sa version finale est donc totalement fonctionnel, il respecte les consignes de l'énoncé et a été terminé dans les délais. Il peut toutefois être encore optimisé en ajoutant un ou plusieurs joueurs supplémentaires. Nous pourrions également travailler sur l'interface graphique afin de rendre le jeu plus ergonomique et esthétique.