UNIVERSITÉ PARIS 13

Institut Galilée
UNIVERSITÉ PARIS 13

PROJET PROGRAMMATION IMPERATIVE

Le Jeu De Dames

NOMS ET PRENOMS :

Ramdane SALHI
Tarek NAIT SAADA
Kamel NAIT SLIMANI

GROUPE : 7

Sommaire :

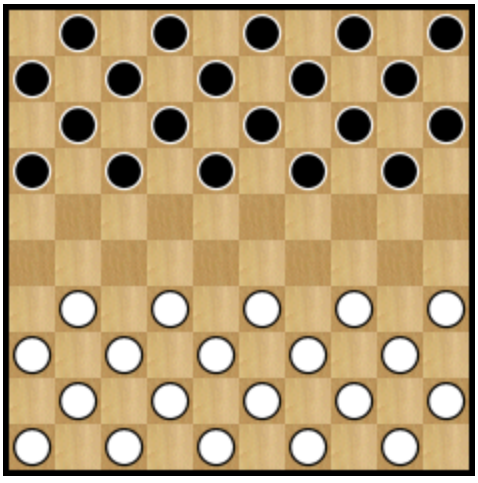
Introduction :

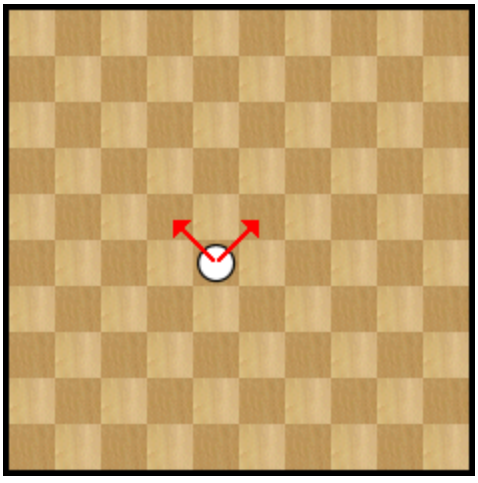
Dans ce projet de PROGRAMAMTION IMPERATIVE nous avons réalisé un jeu de dames programmé sur le langage C. L'intérêt de ce projet est de mettre en œuvre les connaissances acquises lors des cours et des TP de PROGRAMAMTION IMPERATIVE .

Tout d'abord on présentera le Jeu De Dames en gros la manière dont il se joue, ensuite on expliquera chaque partie du programme qu'on a écrit (DAMIER, AFFICHAGE, DEPLACEMENT …), enfin on citera les différentes difficultés rencontrées au cours de la réalisation de ce projet.

Présentation du Jeu :

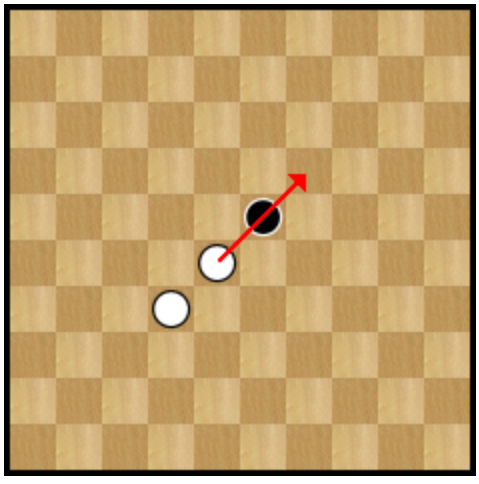
- Le jeu se joue sur un damier de 10 cases sur 10, orienté avec une case foncée en bas à gauche.
- Chaque joueur possède 20 pions, placés sur les cases foncées des 4 premières rangées.
- Les joueurs jouent chacun à leur tour. Les blancs commencent toujours.
- Le but du jeu est de capturer tous les pions adverses.
- Si un joueur ne peut plus bouger, même s'il lui reste des pions, il perd la partie.
- Chaque pion peut se déplacer d'une case vers l'avant en diagonale.
- Un pion arrivant sur la dernière rangée et s'y arrêtant est promu en « dame ». Il est alors doublé (on pose dessus un deuxième pion de sa couleur).
- La dame se déplace sur une même diagonale d'autant de cases qu'elle le désire, en avant et en arrière.





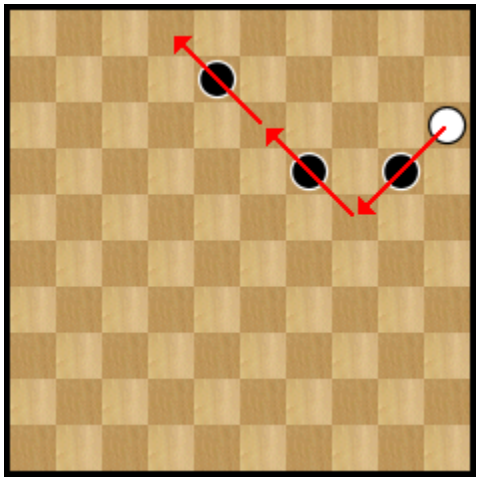
La prise par un pion :

- Un pion peut en prendre un autre en sautant par dessus le pion adverse pour se rendre sur la case - vide située derrière celui-ci. Le pion sauté est retiré du jeu.
- La prise peut également s'effectuer en arrière.
- La prise est obligatoire.
- Si, après avoir pris un premier pion, vous vous retrouvez de nouveau en position de prise, vous devez continuer, jusqu'à ce que cela ne soit plus possible.
- Les pions doivent être enlevés à la fin de la prise et non pas un par un au fur et à mesure.



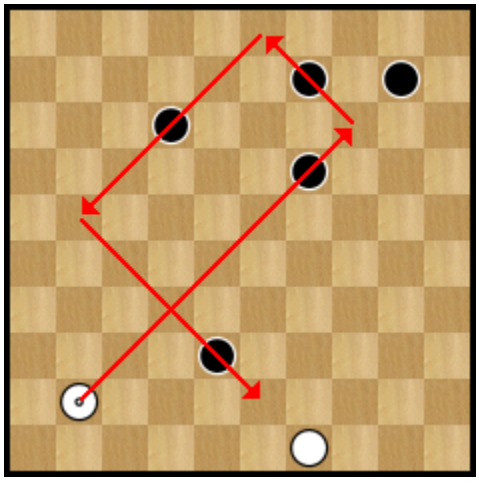
La prise majoritaire :

Lorsque plusieurs prises sont possibles, il faut toujours prendre du côté du plus grand nombre de pièces. Cela signifie que si on peut prendre une dame ou deux pions, il faut prendre les deux pions .Dans l'exemple ci-contre, le pion blanc peut en prendre 3, c'est donc ce coup qui doit être joué.



La prise par la dame :

Puisque la dame a une plus grande marge de manœuvre, elle a aussi de plus grandes possibilités pour les prises.
La dame doit prendre tout pion situé sur sa diagonale et doit changer de direction à chaque fois qu'une nouvelle prise est possible.
On ne peut passer qu'une seule fois sur un même pion.
En revanche, on peut passer deux fois sur la même case.
Dans cet exemple, la dame blanche peut prendre les 4 pions noirs
Enfin, la partie peut être déclarée nulle si aucun des deux joueurs ne peut prendre toutes les pièces adverses (par exemple 3 dames contre une).



3 - Programme et explication :

En premier lieu on a séparé le programme en plusieurs sous programmes (Fichiers, Fonctions, Structures …) . Nous avons donc séparé le code en plusieurs fichiers. Tous le code est séparé dans des fichiers (**piece.c, coordonnees.c, damier.c, capture.c, mouvement.c, liste_config.c, partie.c , forme_de_prise.c , deplacement.c** et bien sûr **main.c**) et chaque fichier avec son propre fichier header (.h) . Tout cela pour une meilleure organisation de notre programme et pour qu'il soit bien compréhensible .

3.1 Les structures :

Au début on a créé les différentes structures qu'on utilisera dans tout le programme et aussi on a ajouté des structures qu'on nous avait pas demandé de faire pour faciliter la programmation de notre programme. Ces structures sont stockées dans les fichiers **piece.h, coordonnees.h, damier.h, mouvement.h, capture.h, liste_config.h, partie.h, forme_de_prise.h, deplacement.h** et les plus importantes sont définies ainsi :

- La structure piece_t, qui symbolise une pièce, elle a deux éléments le joueur (1 ou 0) et le statut (promotion) .

```
enum statut_e{non_promue,promue};
typedef enum statut_e statut_t;

enum joueur_e{joueur0,joueur1,non_joueur};
typedef enum joueur_e joueur_t;

struct piece_s
{
        joueur_t joueur;
        statut_t statut;
};
typedef struct piece_s piece_t;
```

- La structure coordonnees_t qui symbolise une coordonnée dans un damier, elle contient deux entier x et y

```
struct coordonnees_s
{
        int x,y;
};
typedef struct coordonnees_s coordonnees_t;
```

- Les structures dans coordonnees.h sont des structures d'une listes doublement chainée de coordonnées (maillon, liste) .
- La structure cases_t qui représente chaque case du damier elle contient une piece et une couleur :

```
typedef struct cases_s
{
        piece_t piece;
        couleur_t couleur;
}cases_t;
```

- Les structures dans capture.h sont des structures d'une listes doublement chainée de pièces capturées (maillon, liste).

```
typedef struct piece_capture_s
{
        piece_t piece;
        coordonnees_t xy;
}piece_capture_t;
```

- La structure mouvement_t qui représente un mouvement (plus exactement dans maillon mouvement), elle contient une pièce, une liste de coordonnées et une liste de pièces capturées.

```
typedef struct mouvement_s
{
        piece_t piece;
        liste_coordonnees_t* position;
        liste_piece_capture_t* capture;

}mouvement_t;
```

- Les structures dans liste_config.h sont des structures d'une liste doublement chainée contenant les états successifs du damier (maillon, liste).
- La structure partie_t qui représente une partie de jeu elle contient un damier un liste de mouvement, une liste de configuration, et le trait :

```
typedef struct partie_s
{
        cases_t **damier;
        liste_mouvement_t* liste_coups;
        liste_configuration* liste_config;
        joueur_t trait;
}partie_t;
```

- La structure prise_possible_t, une prise possible (NORD, SUD, EST, WEST) , elle contient des coordonnées, le nombre de prise et 4 pointeur vers prise possible :

```
typedef struct prise_possible_s
{
        coordonnees_t xy;
        int nb_prise;
        struct prise_possible_s *E,*W,*N,*S;
}prise_possible_t;
```

- La structure tab_prise_possible_t elle contient un tableau de prises possibles sa taille et le nombre de prise max :

```
typedef struct tab_prise_possible_s
{
        prise_possible_t *tab;
        int taille;
        int prise_max;
}tab_prise_possible_t;
```

- On a fait aussi des structures pour la dame qui sont nécessaire lors de la capture quand c'est une dame.

typedef struct dame_prise_s{

coordonnees_t capture,position;

}dame_prise_t;

typedef struct dame_ensemble_s{

dame_prise_t** Dame;
int max;
int taille;

}dame_ensemble_t;

- Chaque fichier (.h) contient les déclarations des fonctions utilisées au cours de la programmation du jeu.

3.2 Le damier :

La réalisation du damier est faite dans le fichier damier.c . Le damier est un tableau à deux dimensions 10*10 de cases_t alloué dynamiquement. Pour son initialisation, on a deux indices i (9→0) et j (0→9) si i et j sont à la fois pairs ou à la fois impairs et si le i est inferieur a 4 on appelle la fonction creer_cases qui prend en argument la fonction piece_creer qui prend en argument joueur0 et non promu, et la couleur foncée (car elle peut se déplacer que sur la couleur foncée) sinon si le i est supérieur a 5 on appel la même fonction sauf que cette fois si avec joueur 1 sinon on appelle la même fonction avec non_joueur , enfin si par exemple i pair et j impair ou le contraire on appelle creer_case avec non_joueur et la couleur claire(On peut pas se déplacer sur une couleur claire).

Pour l'affichage trois boucles, une pour les indices et les deux autre pour afficher les pièces, On a rajouté dans printf des champs de couleur selon la case claire ou foncée pour une meilleure visualisation du damier.

Pour détruire le damier on a une boucle de 0 a 9 qui a chaque fois libère la case avec free, puis à la fin on libéré le damier.

3.3 Les listes :

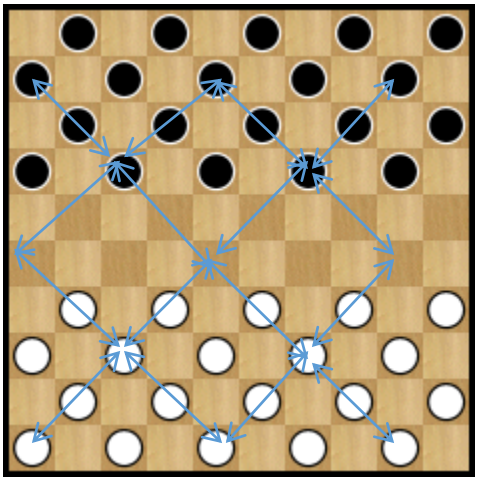
Dans notre programme on a créé 4 listes, chaque liste contient une série de maillon avec deux pointeurs suivant, précédent (Doublement chainée), le contenu des maillons diffère selon les liste :

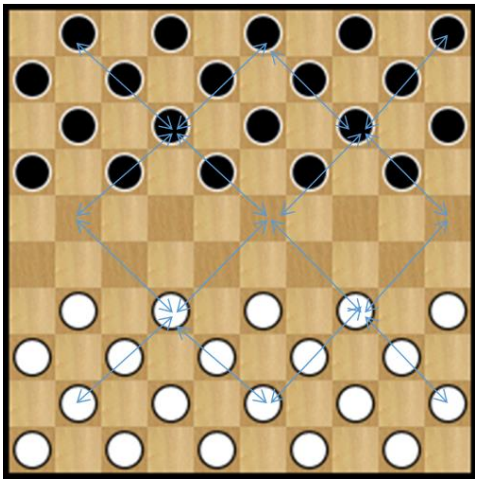
1- Liste Coordonnées chaque maillon contient des coordonnées (x,y)
2- Liste Capture chaque maillon contient des coordonnées (x,y) d'une pièce capturé
3- Liste Mouvement chaque maillon contient une pièce une liste de coordonnées et une liste de pièce capturer, c'est ce qui caractérise un mouvement.
4- Liste Configuration, chaque maillon contient un damier, on représente dans cette liste les états successifs du damier .

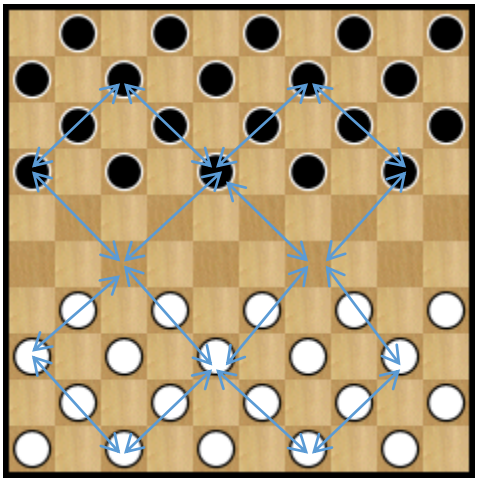
3.4 Le déplacement des pions :

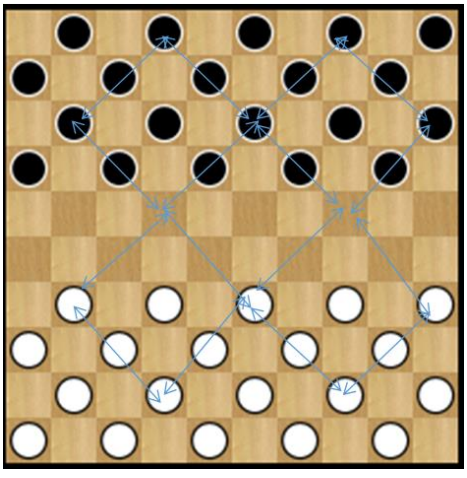
Apres avoir analysé le probleme de deplacement on a trouve que ca peut exister que quatre forme de prise pour les pions dont on a tous les ces derniers.

Alors ya la prise ou le i et le j les indices des coordoneées des pionts sont paires tout les deux et cette se trouve comme la forme A dans le programme ,puis la prisez ou le i est impair et le j est pair qui est la forme B dans le programme les deux forme derniere c'est presque la meme mais de diference d'une case .Puis on a la meme chose pour lee deux autres formes dont une paire et l'autre impaire comme les deux premeires.









Donc on a vu les différentes formes possibles pour la capture des pions et pour réaliser ces formes on a utilisé une structure **prise_possible_t** qui a les coordonnées du point sur le damier et quatre pointeurs vers la même structures pour pouvoir relier entre les déférents points de la forme et un entier sui désigne le nombre de pointeurs non nuls , puis on a stocké ses dernières structures dans un tableaux et tout ça pour chaque forme.

Puisque le stockage on le fait avec des calculs sur les coordonnées donc on ne peut pas avoir un tableau trié dans l'ordre croissant d'abord des indices des lignes puis les colonnes comme on veut alors il faut le trier ().

Après avoir fait ça on a besoin d'initialiser les pointeur des éléments des tableaux par rapport au damier et la forme de prise puis on supprime les pointeurs qui posent problème (non valide) et ça en vérifiant les différentes conditions de la capture et comment la prise peut se faire et on laissant juste les

prises maximales et on supprime les prise les autres prise mais la manipulation ça se passe par les pointeur et les configurations successives du damier .
Donc à la fin de toutes ces opérations on aura un tableau qui présente la prise dont le champ coordonnées des éléments du tableau désigne les positions intermédiaires du pion lorsqu'il se déplace et tout ça c'est juste pour les pions.

3.5 Le déplacement de la dame :

Déplacement dame :

Apres avoir tester si y'a pas un déplacement obligatoire avec un test sur les déférentes prise possible pour toutes les dames d'un joueur dont sur les quatre diagonales ou on appelle une fonction qui prend en arguments la partie des coordonnées un tableaux deux dimensions de **dame_prise_t** qui a son tour contient des couple position et capture , et deux indice de parcours du tableaux comme pointeurs et un max de la longueur maximum de prise dans le tableaux puis on supprime les ligne dont la taille est inférieure à max et on garde les ligne dont le j est supérieur ou égal à max on appelle récursivement sur toutes les cases vides de chaque diagonales avant un pion ou dame de la dame et avec ça on récupère les plus longues prises possibles dont le champs positions de la première colonne de du tableaux récupère les coordonnées de la pièce a déplacer .

Déplacement simple :

On demande à l'utilisateur de saisir les cordonnées de départ puis on appelle une fonction qui lui demande les coordonnées d'arrivée on test la diagonale ou elle se situe, plus la validité du déplacement et cette fonction elle se répète tant que les coordonnées d'arrivée sont pas valides.

3.6 La partie et le programme principal :

Tout d'abord on appelle la fonction de **do while** (non test fin de partie) dans puis le **do while** on crée une partie avec la fonction **partie_creer** ou on initialise la structure **partie_t,** on affiche le damier à l'aide de la fonction affiche plateau puis on appelle la fonction jouer qui prend en argument la partie qu'on vient de créer. Dans la fonction « **jouer** », on crée en premier deux tableaux, un tableau a deux dimensions de prise (la structure qu'on vient de voir tout au début du rapport) (***tab_prise_possible_t)***, puis un tableau unidimensionnel de prise sauf que cette fois-ci c'est des prises obligatoires, on appelle ensuite la fonction **prise_obligatoire** qui prend en argument la partie et la prise, ensuite on appelle la fonction **liste_mouvement_ajouter** qui prend en argument la **liste des coups jouer** dans la structure **partie_t** la fonction **mouvement_creer** ou on crée un nouveau maillon qu'on ajoute à la liste des coups jouer . On test ensuite si la prise est obligatoire avec la fonction **prise_obligatoire_test** si c'est « oui » on demande à l'utilisateur de saisir les coordonnées de la pièce à déplacer puis avec la variable prise_ob (obligatoire) on appelle la fonction **test_appartenire_prise_obligatoire_piece** qui prend en argument la partie la prise et la case tant que prise_ob est NULL. Ensuite on appelle la fonction **deplacement_prise_obligatoire_piece** (là ou on appelle la fonction **modif_damier** qui permet de modifier le damier) qui permet d'appliquer la prise obligatoire puis on appelle la fonction **liste_prise** qui prend en argument une partie, **liste_prise_piece** elle prend la liste des coordonnées dans le mouvement pour calculer le milieu de deux coordonnées et elle l'ajoute à la liste capture de ce mouvement sinon si la prise n'est pas obligatoire on appelle la fonction **deplacement_normal** (là ou on appelle la fonction **modif_damier** qui permet de modifier le damier) qui prend en argument la partie. **TOUT CELA DANS LE CAS OU LA PIECE N'EST PAS PROMUE DANS L'AUTRE CAS : C'est la même chose sauf que cette fois on appelle les fonctions de test de prise obligatoire et de prise maximale de la dame .**
A la fin du mouvement on ajoute le nouveau damier à la liste des configurations, puis on affecter la pièce déplacé dans la pièce du mouvement, Puis on teste la promotion si ce n'est pas une dame selon la dernière position de la pièce dans la liste des positions (liste coordonnées) de ce mouvement, puis on détruit la prise qu'on a allouée dynamiquement.

Après la fonction **jouer** on demande à l'utilisateur sois de valider son mouvement ou de l'annuler, si il valide on supprime les pièces capturés et on met des points, sinon appelle la fonction **annul_mouvement** qui permet d'annuler le mouvement.

3.7 Sauvegarde et replay :

L'utilisateur peut à chaque moment sauvegarder sa partie sois pour continuer de la jouer plus tard ou pour la revoir en mode Replay.
Dans le premier cas on enregistre sous forme de fichier binaire, on a enregistré sous forme binaire tout la liste des mouvements et sa taille (de même pour les listes qui la composent), le trait, et la liste des configurations plus sa taille. Pour les enregistrer on a utilisé la fonction fwrite en enregistrant le contenu des maillons ou des données mais pas les pointeurs et tout ça dans les fichiers **.PL**.

Et pour les récupérer on a utilisé la fonction fread et en suivant le même ordre de l'enregistrement pour que les informations se mélangent pas les unes dans les autres après avoir testé que c'est un fichier **.PL**.

Dans le deuxième cas on a utilisé les fonctions fprint et fscan pour enregistrer les damiers sous forme de caractères et tout ça pour qu'on puisse les récupérer et faire un replay pour que l'utilisateur puisse revoir sa partie.

Les problèmes rencontrés :

- La prise obligatoire des pions → Faite.
- La prise maximale des pions (les boules infinies), On trouve et on affiche les coordonnées la pièce avec laquelle il peut faire la prise max. → Faite.
- La prise obligatoire de la dame → Faite.
- La prise maximale de la dame (les boules infinies), On trouve et on affiche les coordonnées la dame avec laquelle il peut faire la prise max → Faite.
- Lors du changement des structures on a eu un problème de sauvegarde sous forme texte du coup on a tout sauvegardé sous forme binaire. → Fait.
- L'affichage en couleur. →Fait.
- Annuler un mouvement par rapport à tous les changements effectués sur les listes et l'algorithme de récupération des listes→Fait.

Conclusion

Ce jeu de dames nous a permis d'acquérir plus de connaissance en C, Le jeu de dames est un jeu qui as beaucoup de conditions, on a pu surpasser malgré tout la majorité des difficultés qu'on a eu. Ce projet a été très instructif sur le C il nous a permis de bien maitriser la programmation modulaire qui nous servira beaucoup au cours de notre cursus, la programmation structurée qui est vraiment efficace pour réduire le temps d'exécution et la complexité, et aussi d'approfondir les connaissances déjà acquises durant les cours.

Annexe A

Le Makefile

```
all: test.o partie.o coordonnees.o damier.o piece.o forme_de_prise.o deplacement.o jouer.o capture.o mouvement.o
liste_config.o jouer_partie.o prise_dame.o dep_n_dame.o sauvegarder.o
	gcc -g -Wall test.o coordonnees.o deplacement.o partie.o mouvement.o piece.o forme_de_prise.o jouer.o
capture.o liste_config.o jouer_partie.o prise_dame.o dep_n_dame.o sauvegarder.o damier.o -o "test12";rm *.o *~
test.o: test.c
	gcc -c -Wall test.c -o test.o
partie.o: partie.c partie.h
	gcc -c -Wall partie.c -o partie.o
coordonnees.o: coordonnees.c coordonnees.h
	gcc -c -Wall coordonnees.c -o coordonnees.o
damier.o:  damier.c  damier.h
	gcc -c -Wall  damier.c -o  damier.o
piece.o: piece.c piece.h
	gcc -c -Wall piece.c -o piece.o
forme_de_prise.o: forme_de_prise.c forme_de_prise.h
	gcc -c -Wall forme_de_prise.c -o forme_de_prise.o
deplacement.o: deplacement.h deplacement.c
	gcc -c -Wall deplacement.c -o deplacement.o
jouer.o: jouer.h jouer.c
	gcc -c -Wall jouer.c -o jouer.o
capture.o: capture.c capture.h
	gcc -c -Wall capture.c -o capture.o
mouvement.o: mouvement.h mouvement.c
	gcc -c -Wall mouvement.c -o mouvement.o
liste_config.o: liste_config.c liste_config.h
	gcc -c -Wall liste_config.c -o liste_config.o
jouer_partie.o: jouer_partie.c jouer_partie.h
	gcc -c -Wall jouer_partie.c -o jouer_partie.o
prise_dame.o: prise_dame.c prise_dame.h
	gcc -c -Wall prise_dame.c -o prise_dame.o
dep_n_dame.o: dep_n_dame.c dep_n_dame.h
	gcc -c -Wall dep_n_dame.c -o dep_n_dame.o
sauvegarder.o: sauvegarder.c sauvegarder.h
	gcc -c -Wall sauvegarder.c -o sauvegarder.o
```

Annexe B

Liste des fonctions

Nom :	Fichier :	Description :
piece_creer	Piece.h	Créer une pièce
piece_joueur	Piece.h	Identifier le joueur selon la pièce
piece_identifier	Piece.h.	Identifier le joueur selon le caractère
piece_caractere	Piece.h.	Identifier le caractère selon la pièce
piece_afficher	Piece.h	Afficher une pièce (p,P,d,D)
cmp_piece	Piece.h	Comparer entre deux pièces
coordonnees_creer	Coordonnees.h	Créer un maillon coordonnées
coordonnees_deteruire	Coordonnees.h	Détruire un maillon coordonnées
liste_coordonnees_initialiser	Coordonnees.h	Initialiser la liste des coordonnées
liste_coordonnees_vide	Coordonnees.h	Test si la liste est vide
liste_coordonnees_card	Coordonnees.h	Renvoi le cardinal de la liste des coordonnées
liste_coordonnees_afficher	Coordonnees.h	Afficher la liste des coordonnées
liste_coordonnees_ajouter	Coordonnees.h	Ajouter un maillon a la liste
liste_coordonnees_extraire_debut	Coordonnees.h	Extraire le maillon du début de la liste
liste_coordonnees_detruire	Coordonnees.h	Détruire la liste des coordonnées
cmp_coordonnees	Coordonnees.h	Comparer 2 coordonnées
coordonnees_valid	Coordonnees.h	Test si les coordonnées sont valides
piece_capture_creer	Capture.h	Créer un maillon pièce capturé
piece_capture_deteruire	Capture.h	Détruire un maillon pièce capturé
liste_piece_capture_initialiser	Capture.h	Initialiser la liste des pièces capturées
liste_piece_capture_vide	Capture.h	Test si la liste est vide
liste_piece_capture_card	Capture.h	Renvoi le cardinal de la liste des pièces capturées
liste_piece_capture_afficher	Capture.h	Afficher la liste des coordonnées
liste_piece_capture_ajouter	Capture.h	Ajouter un maillon a la liste
liste_piece_capture_extraire_debut	Capture.h	Extraire le maillon du début de la liste
liste_piece_capture_detruire	Capture.h	Détruire la liste
mouvement_creer	Mouvement.h	Créer un maillon mouvement
mouvement_deteruire	Mouvement.h	Détruire un maillon mouvement
liste_mouvement_initialiser	Mouvement.h	Initialiser la liste des mouvements
liste_mouvement_vide	Mouvement.h	Test si la liste est vide
liste_mouvement_card	Mouvement.h	Renvoi le cardinal de la liste des mouvements
liste_mouvement_afficher	Mouvement.h	Afficher la liste des mouvements
liste_mouvement_ajouter	Mouvement.h	Ajouter un maillon a la liste
liste_mouvement_extraire_debut	Mouvement.h	Extraire le maillon du début de la liste
liste_mouvement_detruire	Mouvement.h	Détruire la liste

maillon_configuration_creer	Liste_config.h	Créer un maillon configuration
maillon_configuration_detruire	Liste_config.h	Détruire un maillon configuration
creer_liste_configuration	Liste_config.h	Créer une liste de configuration
ajout_liste_configuration	Liste_config.h	Ajouter un maillon a liste de configuration
liste_configuration_detruire	Liste_config.h	Détruire maillon
afficher_fin	Partie.h	Afficher le joueur qui a gagné la partie
affiche_joueur	Partie.h	Afficher le joueur dans une partie
case_vide	Partie.h	Test si une case est vide

modifier_case	Partie.h	Modifier une case
changer_joueur	Partie.h	Switcher entre deux joueurs
modif_damier	Partie.h	Modifie le damier à partir de la liste des mouvements
annuler_mouvement	Partie.h	Annuler un mouvement
saisie_case	Partie.h	Demander au joueur de saisir une case
partie_creer	Partie.h	Créer une partie
partie_detruire	Partie.h	Détruire une partie
test_fin_parite	Partie.h	Tester si la parie est finie
creer_cases	Damier.h	Créer une case dans le damier
creer_damier	Damier.h	Créer un damier
initialier_damier	Damier.h	Initialiser un Damier
afficher_damier	Damier.h	Afficher le damier
detruire_damier	Damier.h	Détruire (Libérer l'espace) le damier
test_prise_valid	Deplacement.h	Test si la prise valide (pièce adversaire) à partir d'une coordonné
test_prise_N	Deplacement.h	Incrémente vers le nord et test avec la fonction test_prise_valid
test_prise_S	Deplacement.h	Incrémente vers le sud et test avec la fonction test_prise_valid
test_prise_E	Deplacement.h	Incrémente vers l'est et test avec la fonction test_prise_valid
test_prise_W	Deplacement.h	Incrémente vers l'ouest et test avec la fonction test_prise_valid
piece_prise_coordonnees	Deplacement.h	Calcule les coordonnées de la pièce à capturer
test_deplac_N	Deplacement.h	Test si elle peut de déplacer vers le nord
test_deplac_S	Deplacement.h	Test si elle peut de déplacer vers le sud
test_deplac_E	Deplacement.h	Test si elle peut de déplacer vers l'est
test_deplac_W	Deplacement.h	Test si elle peut de déplacer vers l'ouest
test_move	Deplacement.h	Fait le test complet avec les fonctions précédentes de test
tab_prise_creer	Forme_de_prise.h	Allouer de la mémoire pour le tableau de prise
tab_initialiser_xy_forme_a	Forme_de_prise.h	Initialiser les coordonnées selon la première forme de prise
tab_initialiser_xy_forme_b	Forme_de_prise.h	Initialiser les coordonnées selon la 2éme forme de prise
tab_initialiser_xy_forme_c	Forme_de_prise.h	Initialiser les coordonnées selon la 3éme forme de prise
tab_initialiser_xy_forme_d	Forme_de_prise.h	Initialiser les coordonnées selon la 4éme forme de prise
tab_forme_prise_afficher	Forme_de_prise.h	Afficher les coordonnées dans le tableau de forme de prise
tab_forme_prise_min	Forme_de_prise.h	Prend la plus petite coordonnées dans le tableau de prise et la renvoie
swap_prise_possible	Forme_de_prise.h	Swap entre deux cases dans le tableau de prise
tab_forme_prise_trie	Forme_de_prise.h	Trier le tableau de prise
tab_initialiser_pointeur_forme_ab	Forme_de_prise.h	Initialiser les pointeurs de la forme a et b
tab_initialiser_pointeur_forme_cd	Forme_de_prise.h	Initialiser les pointeurs de la forme c et d
supprimer_cases_non_valid	Forme_de_prise.h	Supprime les pointeurs au cas où la case est occupée par l'adversaire
prise_possible_nb	Forme_de_prise.h	Calcule le nombre de case auquel elle est pointée
max_4nb	Forme_de_prise.h	Calcule maximum entre 4 nombres
piece_prise_max	Forme_de_prise.h	Calcule le plus long chemin
supprimer_les_prise_min	Forme_de_prise.h	Détecte le chemin minimum des prises
supprimer_le_chemain_min	Forme_de_prise.h	Supprimer les chemins les plus courts
prise_max	Forme_de_prise.h	Elle supprime toutes les prises minimum

creer_forme_prise_possible	Forme_de_prise.h	Elle Crée les formes de prise possible
afficher_pionteur2	Forme_de_prise.h	Affiche les coordonnées de la case pointée
forme_prise_detruire	Forme_de_prise.h	Détruire les tableaux de forme de prise
prise_obligatoire_f	Forme_de_prise.h	Créer les formes de prise et choisi la plus grande
menu	Jouer.h	Affiche le menu
afficher_suite_mouvement	Jouer.h	Affiche la suite des damiers dans la liste de configurations
jouer_partie	Jouer.h	Jouer une partie
play	Jouer.h	Fonction principale du jeu ou il y'a la boucle infinie
partie_sauvegarder	Sauvegarde.h	Sauvegarde une partie complète
partie_charger	Sauvegarde.h	Charger une partie complète
afficher_ensemble	Prise_dame.h	
chercher_piece_cap_W	Prise_dame.h	Elle renvoi les coordnnesde la piece a capture et l'arrivee sur la diagonale haut gauche
chercher_piece_cap_N	Prise_dame.h	Elle renvoi les coordnnesde la piece a capture et l'arrivee sur la diagonale bas gauche
chercher_piece_cap_S	Prise_dame.h	Elle renvoi les coordnnesde la piece a capture et l'arrivee sur la diagonale bas gauche
chercher_piece_cap_E	Prise_dame.h	Elle renvoi les coordnnesde la piece a capture et l'arrivee sur la diagonale bas gauche
dame_ensemble_appartient	Prise_dame.h	Test si la coodonneees capture apartien a l'ensemble des coordonnees capturee
dame_vecteur_appartient	Prise_dame.h	Test si une ligne apartien a une matrice
ajouter_ensemble	Prise_dame.h	Il ajoute une structure a un tableau
dame_prise_max	Prise_dame.h	Si une prise maximum elle l'ajoute au tableaux
dame_ensemble_prise	Prise_dame.h	La modifaication du tabeaux selon la plus grande prise et on l'appelle pour chaque dame et c'est la fonction recurssife qui fais le plus grand traville du deplacement de la dame
initialiser_ensemble	Prise_dame.h	C'est l'initialitation de tous les couples du tableaux avec un calloc (0,0) pour les positions et (0,0) pour capture
detruire_ensemble	Prise_dame.h	Free l'ensemble
creer_ensemble_prise_dame	Prise_dame.h	C'est la creation de l'ensemble ou se trouve le tableau des prises
mouvement_possible	Dep_n_dame.h	Test si un mouvment est possible pour un deplacement simple
aprartenir_diag	Dep_n_dame.h	Identification de la diagonale ou se trouve la cooronnees d'arrivee
deplacement_simple_dame	Dep_n_dame.h	Efectue un deplacement simple sur la dame si c'est permis et modefie le damier