MasterMind

Autheurs :
GARCIA CAMPOS, Pablo
BOUDIN, Lina

1. Cahier de charges : MasterMind

1.1 Contexte et définition du problème

Nous avons étudié les différentes couches réseau et travaillé sur les protocoles ARP, IP, TCP et UDP. Dans ce projet, le but est de programmer une application s'utilisant à distance grâce à un programme serveur (en remote) et un programme client (en local). L'application doit être algorithmiquement simple, ce qui nous permettra de nous centrer sur la partie réseau. Pour se faire, les deux programmes doivent contenir une partie de connexion des deux machines et une partie d'exécution du logiciel.

1.2 Objectif

Le logiciel choisi est le jeu MasterMind auquel il faut pouvoir jouer en distanciel. Le cœur du jeu passe en remote, dans un serveur. Le joueur aura un programme client pour pouvoir se connecter au serveur et échanger des informations (les tentatives du jeu ou le choix du niveau de difficulté). Un des objectifs fixés est la possibilité du serveur de connecter plusieurs joueurs qui pourront effectuer des parties simultanées sur le serveur sans que les informations transmises ne se mélangent et ceux sans maximum de joueurs.

1.3 Spécificités techniques de l'application

1.3.1 Le jeu : MasterMind

I Présentation : MasterMind est un jeu de réflexion et de déduction pour deux joueurs dont le but est de trouver une combinaison secrète de 4 couleurs entre certaines couleurs disponibles. L'ordre des couleurs de la combinaison secrète est important et il est possible qu'elle soit composée de la même couleurs placées plusieurs fois dans la séquence (exemple : jeune-vert-jaune-bleu). Originalement, un premier joueur créé une combinaison secrète (codeur), et le second essaie de la déchiffrer (déchiffreur). Dans cette version, l'ordinateur joue le rôle de codeur, et le joueur le rôle de déchiffreur. De plus, il faut que le joueur puisse choisir le difficulté avec lequel il souhaite jouer qui correspond au nombre de couleurs différentes possibles dans la combinaison : 5,6 ou 7 couleurs.

II Objectif : Le jeu consiste à deviner la combinaison secrète de 4 couleurs créée entre les couleurs disponibles et la deviner avec le moins de tours possibles. Les couleurs disponibles dépendent de la difficulté choisie.

III Dynamique du jeu : Au début de la partie, le joueur choisie la difficulté (nombre de couleurs) et l'ordinateur crée une combinaison secrète avec ces couleurs. À chaque tour, le

joueur donne une combinaison de 4 couleurs appelée tentative, après laquelle l'ordinateur va répondre :

- Combien de couleurs sont à la bonne place.
- Combien de couleurs sont bien présentes dans la combinaison secrète mais sont mal placées (les couleurs bien placées ne comptent pas à nouveau dans le nombre de mal placé en cas de répétition de couleurs).

La partie se termine quand le joueur trouve la combinaison secrète. Le nombre d'essais possibles est infini mais le calcul du score final dépend inversement du nombre de tours nécessités.

1.3.2 Application

L'application sera implémentée pour pouvoir jouer à distance. Pour cela, l'application compte deux programmes : un serveur TCP et un client TCP. Ce protocole permet d'établir une connexion entre le client et le serveur et s'assure de l’intégrité et du transfert de tous les paquets.

I Serveur : ce programme compte toutes les fonctions nécessaires pour établir une connexion TCP, toutes les fonctions du jeu, ainsi que toutes les fonctions nécessaires pour traiter l'information du jeu à partir du format reçu par le réseau et vice-versa. Enfin, il permet la connexion en parallèle d’autant de clients que nécessaires.

II Client : ce programme est un client TCP et permet au joueur d'interagir avec le jeu depuis son ordinateur. Ce programme est intermédiaire entre le joueur et le jeu, exécuté dans le serveur. Pour cela, il compte des fonctions de gestion d'information échangées entre l'interface joueur et du format de message envoyé par le réseau au serveur. Cette interface est une simple console qui lit les couleurs proposées par le joueur et imprime la réponse du serveur.

1.3 Budget du projet

Malheureusement ce projet est gratuit.

1.4 Délais de réalisation de l’application

Livraison le 01/12/2023.

2. Application en détail : protocoles, solutions choisies, …

2.1 Connexion serveur et client.

Quel protocole de transport choisir ?

Le protocole TCP est le plus adapté au jeu MasterMind car l’intégrité des données transférées est plus important dans ce type de logiciel que la rapidité et l’instantanéité de l’information fournit par UDP. En effet la transmission intact des strings des combinaisons et inversement le retour du résultat fournit par le serveur est essentiel pour le déroulement du jeu.

La connexion :

Le serveur a un port prédéfini, connu par le client. Ainsi, le client n'a besoin que de l'adresse IP du serveur, qui dans le cas local vaut 127.0.0.1, l'adresse du *loopback*.
Dans la figure 1, on résume avec des automates, l'ouverture et fermeture de la connexion. Le serveur écoute des potentielles connexion entrantes. Dès que le client démarre son programme, il se connecte au serveur pour établir la connexion avec l'échange des paquets SYN / SYNACK / ACK. De coté du serveur, le processus est dupliqué : un processus père (en noir) ferme la connexion avec le client et le processus fils (en vert) ferme la socket d'écoute. Ceci est nécessaire car les deux processus écoutent sur le même port, donc si on ne le fermait pas, il y aurait des interférences dans la lecture des données. Le processus père continue d'écouter dans l'attente de nouvelles connexions dans une boucle infinie, tandis que les fils gèrent la partie avec le client.
Sur cette figure le corps du jeu n'est pas détaillé, ceci est expliqué en détail dans la section 2.2.
Une fois la partie finie, chaque programme (client et serveur) ferme la connexion de son coté. Enfin, le serveur fils arrête le processus directement sans finir le code.

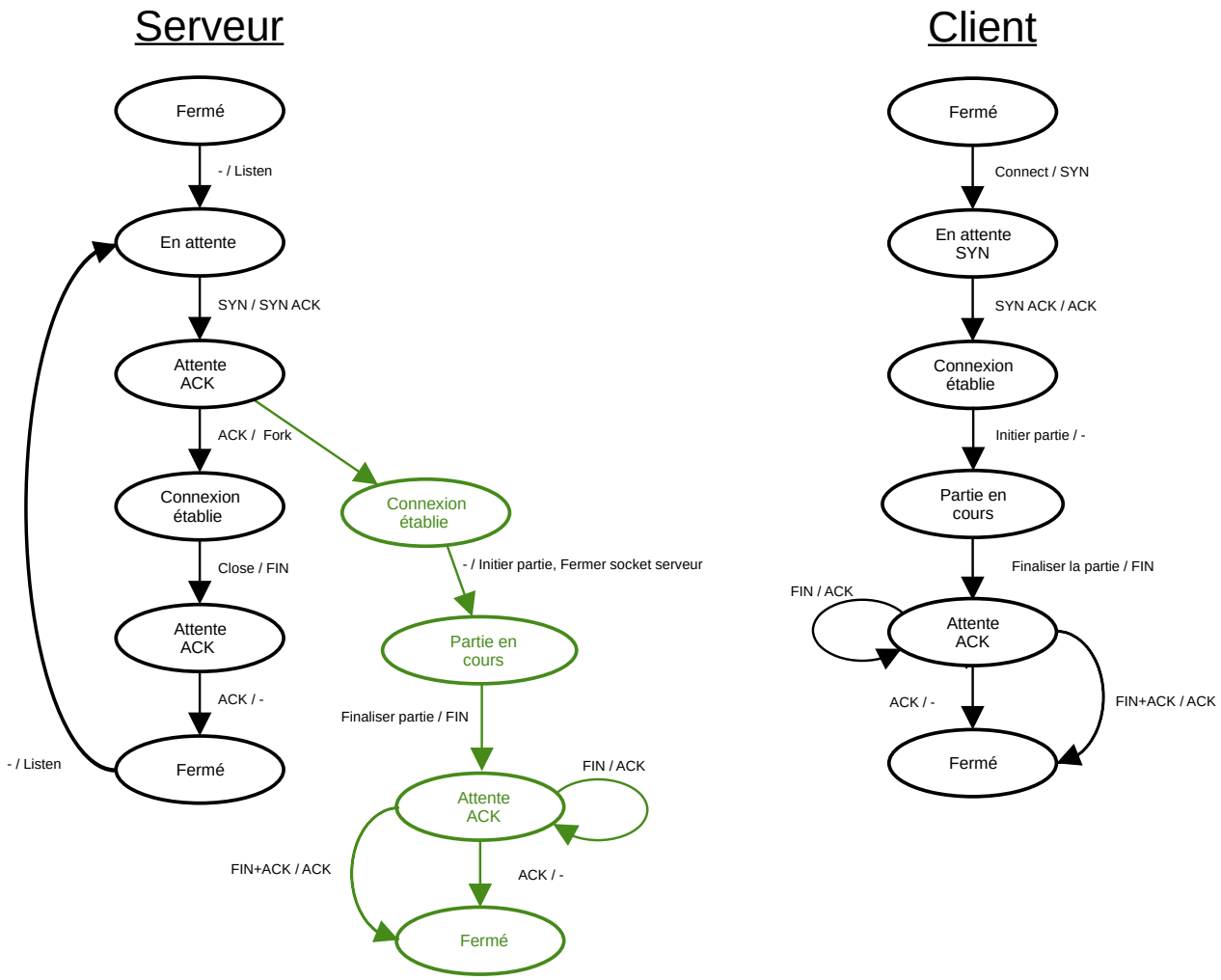


Figure 1: Automates du serveur et client pour l'ouverture, vie et fermeture des connexions. En particulier, le serveur duplique le processus une fois qu'il a accepté une connexion. Le processus père (noir) et le processus fils (vert) auront une suite différente comment observé dans la figure.

2.2 Jeu en détail

Au début de la partie, le client écrit les règles au joueur et puis introduit la difficulté souhaitée. Avec cela, le serveur crée la séquence secrète à deviner.
Le joueur, tour sur tour, donne des combinaisons de couleurs, auxquelles le serveur répond avec le nombre de couleurs bien et mal placés. Une fois la séquence secrète trouvée, le joueur reçoit un score en fonction du nombre de tours ayant été nécessaires.

Le fonctionnement du jeu est schématisé dans la figure 2. Les boxes rectangulaires représentent des fonctions, et les rhomboïdes des variables représentatives.
Les inputs du joueur sont toujours validés avant être traités (convertis en entiers). Ceci est réalisé par les deux fonctions de la fig2 : validationNiveau et validationSequence. Puis ces données sont transformées en séquence d'entiers, qui sera postérieurement misse en format de texte pour la transmettre par le réseau. Du coté serveur elle sera décodée et reconvertie en séquence d'entiers pour faciliter le traitement. Le retour est aussi implémenté en format de séquence, donc les mêmes fonctions sont réutilisées.
Le niveau de difficulté est une séquence d'un entier, tandis que la séquence de couleurs est une séquence de 4 entiers. Le retour du serveur est combien de couleurs sont bien et mal placés, et ceci est codé dans un séquence de deux entiers. Finalement, quand la séquence de couleurs est correct, le serveur augmente la taille de la séquence de retour à 4 : nombre de bien placés, nombre de mal places, nombre de tours et score. Le client sait que la partie est finie parce qu'il a reçu un séquence de taille 4.

Problème de la taille des informations échangées (format) :

Pour échanger des messages avec les fonctions h_reads et h_writes il est nécessaire de connaître la taille des messages envoyés et qui doivent être lu pour ne pas bloquer une des parties en attente constante d'informations. Pour cela, deux fonctions auxiliaires ont été créées, Ecrire et Lire, pour attribuer un nouveau format au messages précisant le nombre d'octets envoyés et qui doivent être lus par le serveur ou le client. La fonction Ecrire rajoute deux bytes en début de messages pour préciser la taille des messages envoyés, toujours en format texte. La fonction Lire permet à la machine réceptrice d'un message de lire tout d'abord seulement les 2 premiers bytes du message voulant être reçu et correspondant à la taille des données qui suivent. Ensuite grâce à cette informations la lecture par h_reads peut être effectué en connaissant la taille exacte des données.

Séquence : [1,2, 3] => texte : "1 2 3" => message envoyé : "051 2 3"

Lire(socketId, taille_str , 2) ;
Lire(socketId, text, atoi(taille_str)) ;

Vérifications supplémentaires : des fonctions pour vérifier les entrées saisies par le joueur avant de lancer la suite du programme ont aussi été créées. Ces fonctions vérifient notamment la composition des séquences fournis comme tentatives : est-ce que les couleurs sont bien celles possibles selon le niveau de difficulté choisi et est-ce qu'elles sont bien dans le bon format de saisie avec 4 couleurs séparées par des tirets.

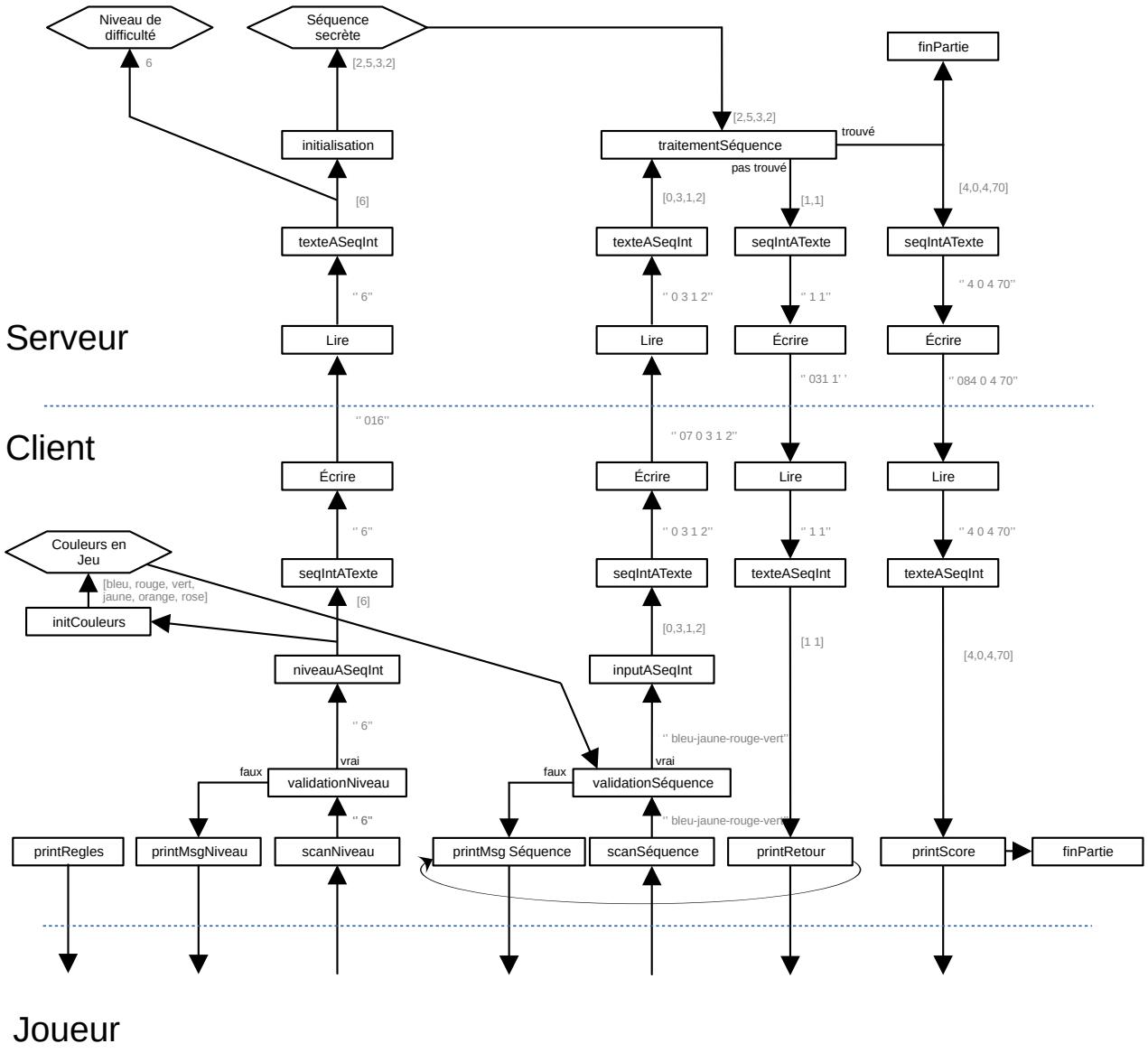


Figure 2: Schéma de la structure des logicielles. Au-dessus de la ligne bleue a tirée on a le serveur, dessous on la le client et tout en bas, le joueur. On a représenté en rectangle les différent fonctions et en rhomboïde des variables représentatives. Les flèches représentent la direction de flux de données. La boucle principale est schématisé par la flèche curve du client. Les annotations en gris à coté des flèches sont un exemple de valeurs possibles à chaque étape. En occurrence, ces donnés correspondent au jeu des figures 3 et 4.

2.3 Exemples

Voici un exemple de partie : coté client dans la figure 3 et coté serveur figure 4.
Après avoir lancé les exécutables serveurMasterMind et puis clientMasterMind,
le client présente les règles du jeu et puis demande la difficulté. A ce moment là le serveur créé la séquence secrète (voir figure 4).
A chaque tour, le client rappelle au joueur les couleurs en jeu, et demande une séquence. Celles-ci sont toujours vérifiés par le client avant d'envoyer au serveur (deuxième tentative du joueur). Une fois validées, on peut les voir du coté serveur.
Après 4 tentatives, le joueur a trouvé la séquence secrète, donc il a une ponctuation de 70 points.

Dans la figure 2, on peut observer certaines données de cette partie, en particulier comment les donnés de la difficulté, de la première séquence tentée et du retour final sont transformés lors de la traversée entre client et serveur.

```
Bienvenue dans Mastermind !
Le but de ce jeu est de deviner le plus rapidement possible une séquence de 4 couleurs dans le bon ordre.

Vous allez devoir tenter de deviner la combinaison secrète en soumettant une séquence de 4 couleurs qui vo
us semble correcte.
 Après chaque tentative il vous sera fait un retour sur le nombre de couleurs correctement placées dans vo
tre séquence ainsi que le nombre de couleurs bien présente dans la combinaison secrète mais mal placées.
 Attention il est possible que la même couleur soit présente plusieurs fois dans la combinaison secrète. L
ors de la rentrée d'une tentative il est nécessaire de faire attention à l'orthographe des couleurs et de
rentrer la séquence avec des tirets entre chaque couleurs (exemple: jaune-vert-rouge-vert)
Enfin, il est possible de choisir le nombre de couleurs différentes qui pourra être présente dans la combi
naison secrète et donc augmenter la difficulté du jeu !

Avec combien couleurs voulez-vous jouer? [5-7]: 6

Pour rappel les couleurs possibles sont:  bleu-rouge-vert-jaune-orange-magenta
Entrez la séquence de couleur proposée (séparé par des tirets):
bleu-jaune-rouge-vert

Vous avez trouvé 1 couleur bien placée et 1 couleur présente mais mal placée

Pour rappel les couleurs possibles sont:  bleu-rouge-vert-jaune-orange-magenta
Entrez la séquence de couleur proposée (séparé par des tirets):
magenta-range-jaune-vert

Pour rappel les couleurs possibles sont:  bleu-rouge-vert-jaune-orange-magenta
Entrez la séquence de couleur proposée (séparé par des tirets):
MAGENTA-ORANGE-JAUNE-VERT

Vous avez trouvé 2 couleurs bien placées et 1 couleur présente mais mal placée

Pour rappel les couleurs possibles sont:  bleu-rouge-vert-jaune-orange-magenta
Entrez la séquence de couleur proposée (séparé par des tirets):
o-m-j-v

Vous avez trouvé 3 couleurs bien placées

Pour rappel les couleurs possibles sont:  bleu-rouge-vert-jaune-orange-magenta
Entrez la séquence de couleur proposée (séparé par des tirets):
v-m-j-v

Vous avez trouvé 4 couleurs bien placées

Bravo, vous avez trouvé la séquence secrète en 4 tours !
Votre score est de 70 points !
```

Figure 3: Console du client. On observe une petite partie avec les différent possibilités d'input de séquence et une réitération de demande car on a écrit "range" au lieu d' "orange". Cette partie sert aussi d'exemple pour la transmission des données entre client et serveur représenté dans la figure 2 en notes gris clair.

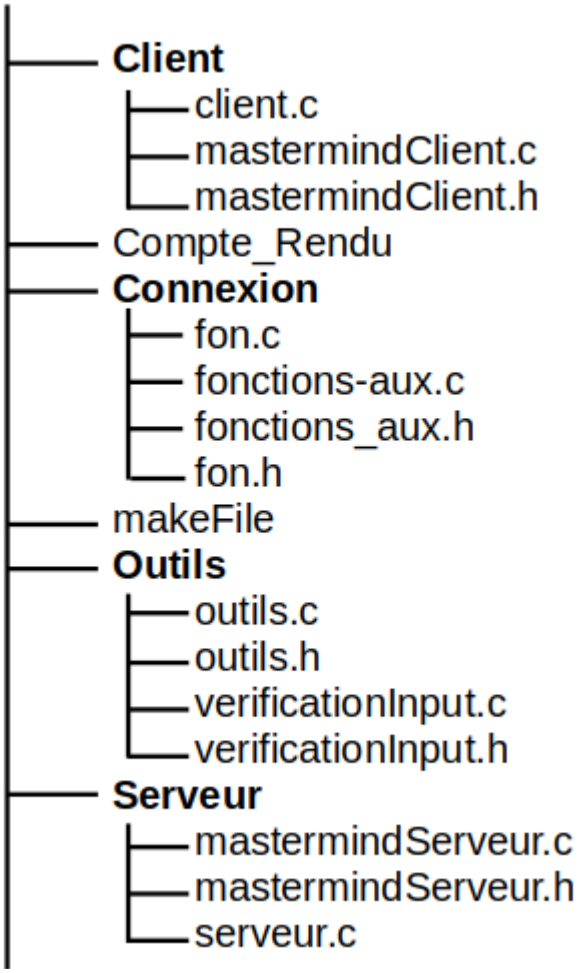
```
SERVEUR :

Sequence secrete: 2 5 3 2
Sequence du client : 0 3 1 2
Sequence du client : 5 4 3 2
Sequence du client : 4 5 3 2
Sequence du client : 2 5 3 2
```

Figure 4: Console du serveur lors d'une seule partie. Les séquences ne sont pas normalement affichées mais on a modifié cela pour montrer les séquences qui arrivent au serveur. Cette partie sert aussi d'exemple pour la transmission des données entre client et serveur représenté dans la figure 2 en notes gris clair.

2.4 Code source.

Il est possible de retrouver tout le code commenté du projet dans le dossier code et dans ses différentes sous-dossiers.

Les fichiers de code avec l'extension de nom de fichier .c sont accompagnés d'un fichier d'extension de nom .h (*headers*) contenant les entêtes utiles pour la compilation.



→ client.c : contient la fonction main du client appelant toutes les fonctions pour établir la connexion et jouer.

→ mastermindClient.c : contient les fonctions du jeu associé à la partie client dont fonctions des les messages de début et de fin de partie, la saisie de la séquence tentative,la conversion de type de la séquence, réception des messages, etc.

→ fonc.c : fonctions d'aide de gestion de la connexion fournit pour le TP

→ fonctions_aux.c : fonctions pour échanger des informations en connaissant la taille des messages voulant être transmis

→ makeFile **:** pour compiler les fichiers et créer les exécutables.

→ outils.c : fonctions de conversion de séquences de caractères en séquences d'entiers et inversement.

→ verificationInput.c : fonctions de vérification des entrées fournit par le client.

→ mastermidServeur.c : fonction du jeu associé à la partie serveur dont la création de la combinaison secrète, l'analyse des tentatives, etc.

→ serveur.c : contient la fonction main du client appelant les fonctions pour attendre une connexion en lancer le jeu