HELHa
Haute École Louvain en Hainaut

Rapport de Projet

Bachelier en informatique de gestion
Bloc 1

Mastermind

ZARZYCKI ALEXIS

Cours de Mr. Alary et Mr. Moins

Année académique 2023 – 2024

ÉCONOMIQUE

Sommaire

Introduction

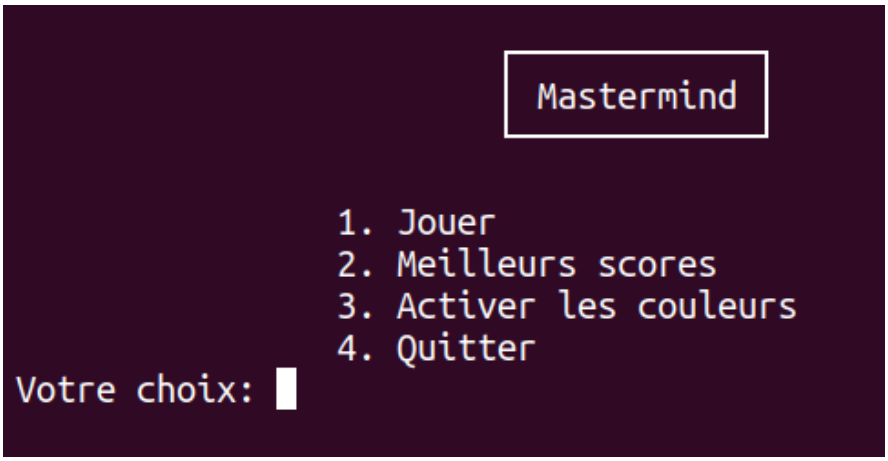
En cette année académique 2023-2024, nous avons dû pour le cours de projet, créer un jeu. Ce jeu est un Mastermind, pour celui qui ne connait pas le principe, ce jeu est à la base un jeu de société. De nos jours, il existe plusieurs sites en ligne qui permettent de jouer au même style de jeu. Vous connaissez sûrement Tusmo, ou encore Wordle. Revenons au principe du jeu, le but du jeu est de trouver un mot en faisant le moins d'essais possible, chaque lettre bien placée est représentée par un +, et une lettre mal placée sera notifiée par un -, la notification se fait dans la colonne à droite du mot, si la lettre n'est pas dans le mot rien n'est notifié. Prenons un exemple pour mieux illustrer, vous lancer une partie, le mot à deviner est "amis". Vous entrez le mot "arme", il y a une lettre bien placée, le "a". Il y a également la lettre "m" qui est présente, mais mal placée. Dans la colonne de droite, vous aurez l'affichage de +-. Une fois le mot deviné, vous avez l'opportunité d'entrer votre pseudo afin d'enregistrer votre score !

Les objectifs du projet

- Développer un projet fonctionnel sur base d’un squelette fourni
- Rédigé des tests fonctionnels
- Développer des tests unitaires
- Utiliser un gestionnaire de version (GIT)
- Mettre en avant les connaissances apprises lors des leçons
- Présenter et défendre un projet

Mode d'emploi de l'application

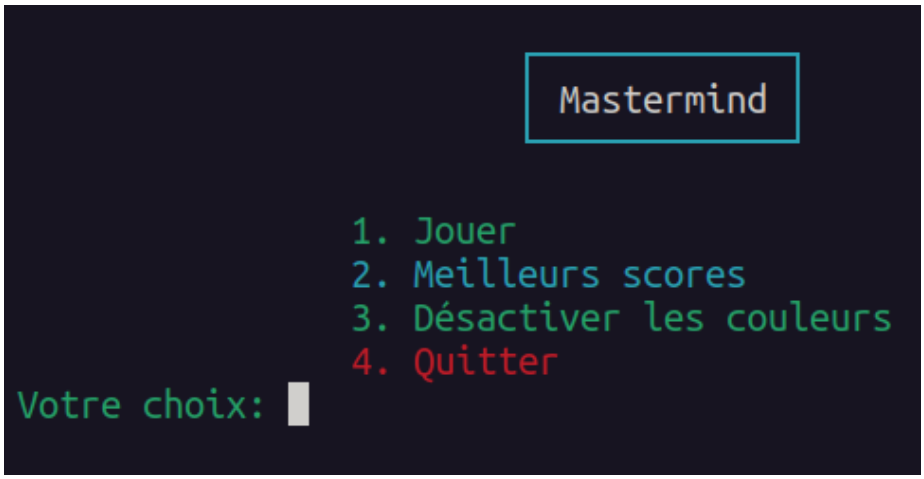
Lancement de l'application



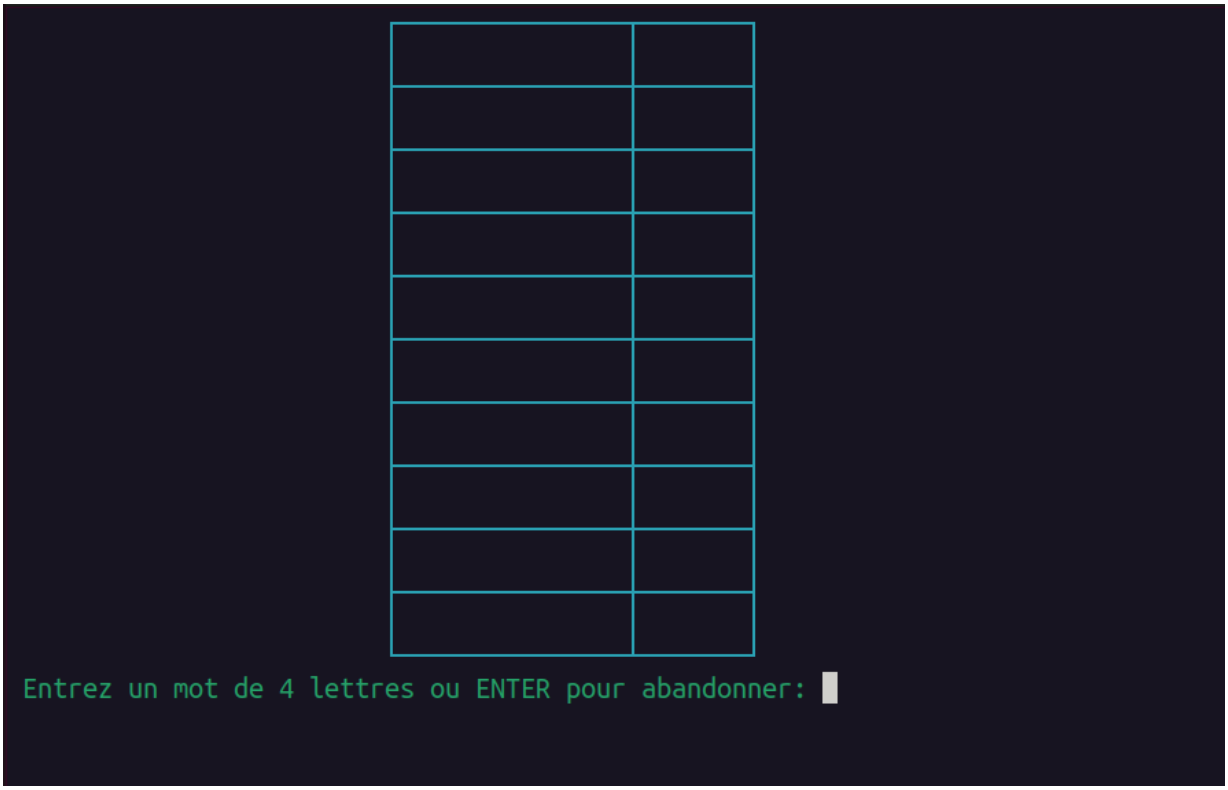
Dès le lancement de l'application, nous avons un menu qui s'affiche.

Celui-ci nous propose de jouer une partie, afficher les meilleurs scores, activer les couleurs ou quitter l'application. Si l'entrée est erronée, l'entrée est effacée et rien ne se passe.

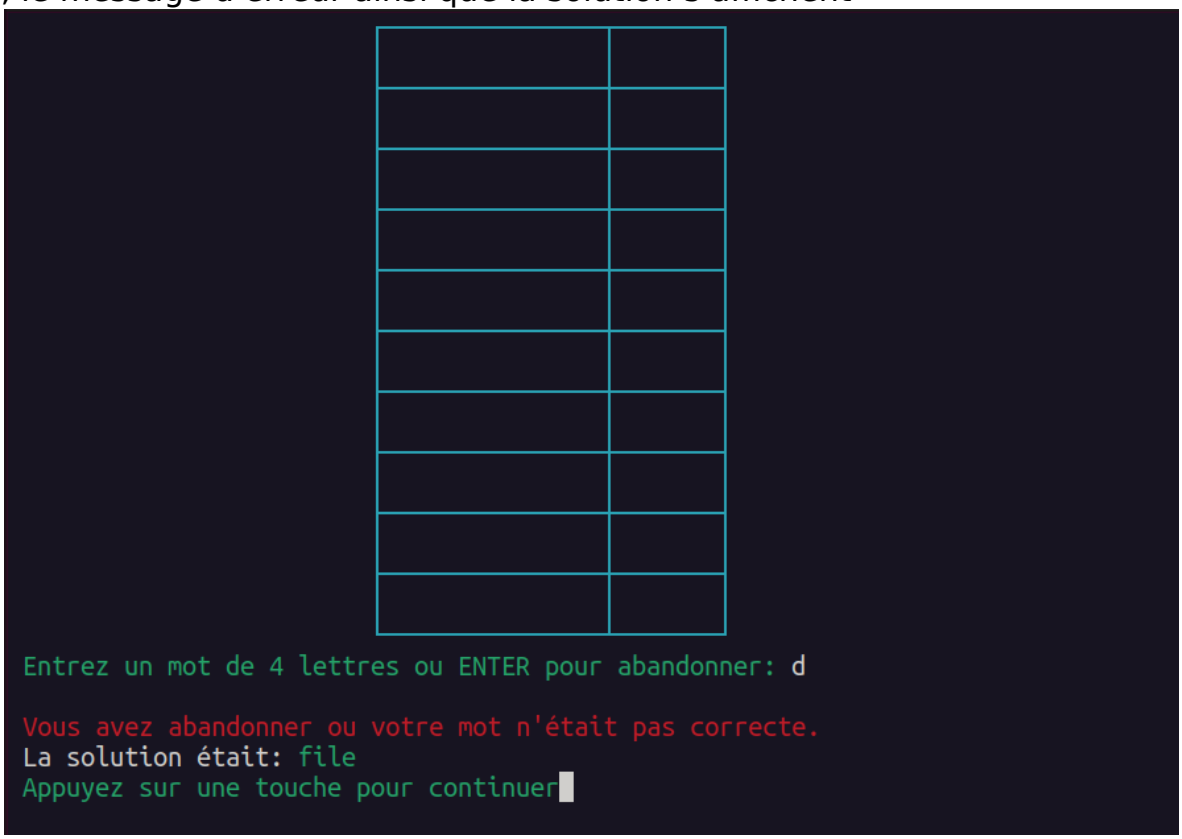
Si vous choisissez 3, les couleurs s'activent. Une fois les couleurs active, vous pouvez choisir 3 de nouveau afin de les désactiver.



Si vous entrez 1, le jeu se lance :



Une fois le jeu lancé, nous devons entrer un mot de 4 lettres. Si le mot ne fait pas 4 lettres, le message d'erreur ainsi que la solution s'affichent



Après avoir appuyé sur une touche pour continuer, l'écran des meilleurs scores s'affiche :

Après avoir appuyé sur ENTER, vous êtes de retour sur le menu principal.

Reprenons au lancement du jeu : une fois le jeu lancé, imaginons vous rentrer le mot « amis »

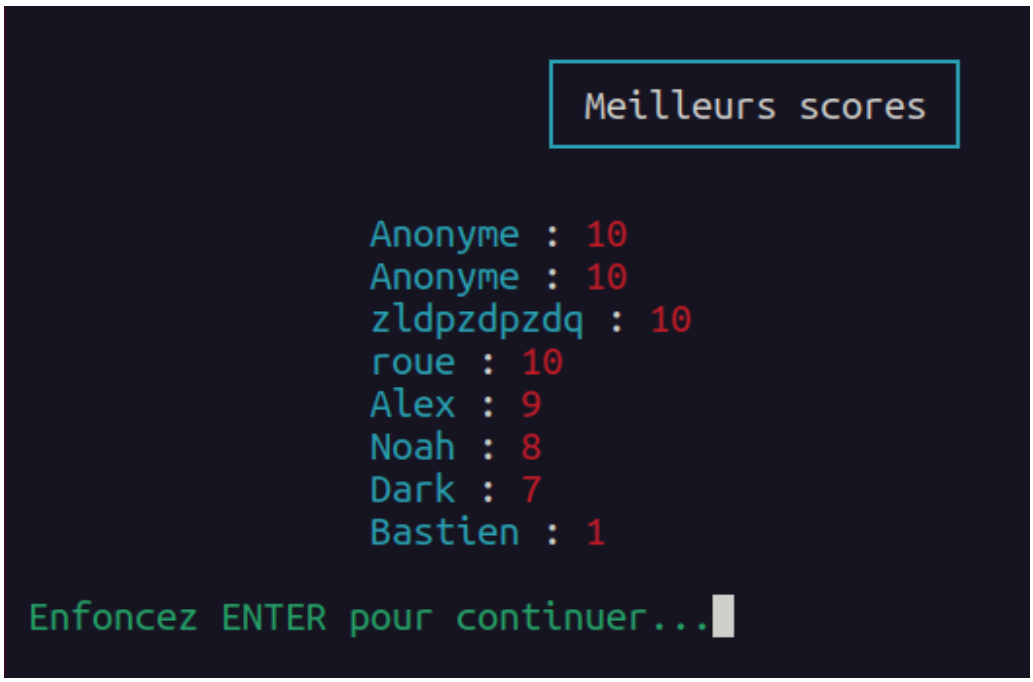
Dans cet exemple, notre solution est «faim». Nous avons donc une lettre bien placée, et deux lettres mal placées.

Une fois le mot adéquat trouvé, le jeu nous demande notre pseudo :

Si vous ne rentrez pas de pseudo, celui-ci sera « Anonyme »

Une fois le pseudo rentré, vous devez appuyer sur une touche pour continuer.

Ensuite, le tableau des meilleurs scores s'affichera :



Pour calculer le score, le jeu applique la formule suivante : 11 – nombre d'essais.
Pour notre partie exemple, j'ai mis 2 essais pour trouver le mot. J'ai donc eu 9 points.

Si vous appuyez sur 2, le tableau des meilleurs scores s'affichera :

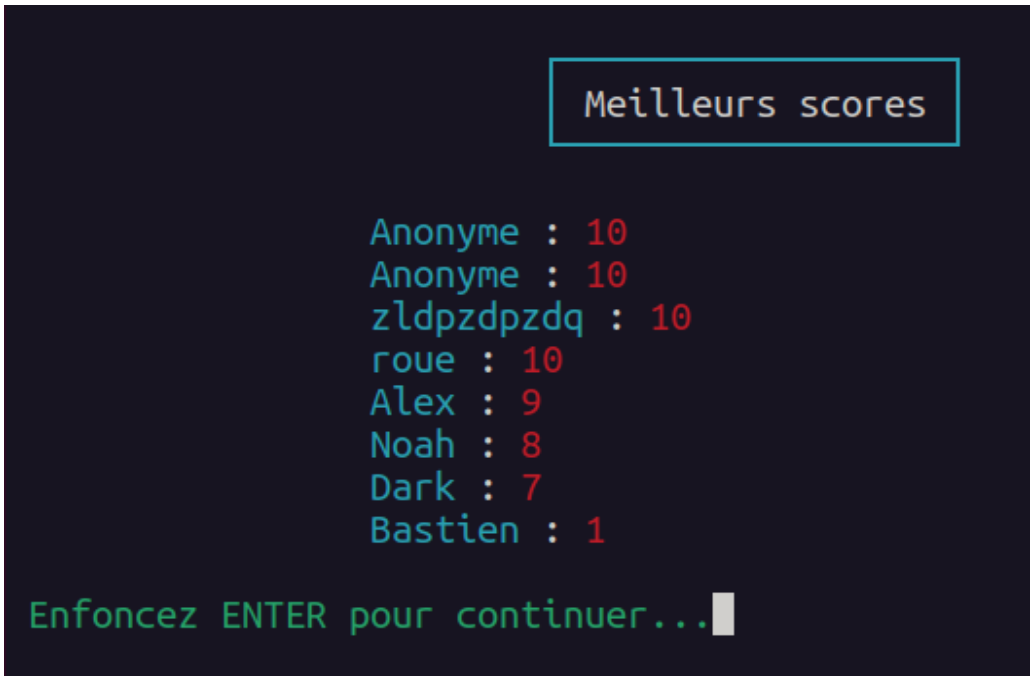
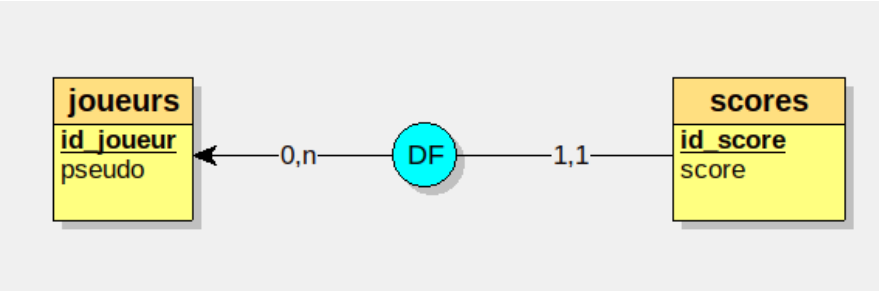


Tableau des mes fonctions

Nom et signature de la fonction	Nom des paramètres	Type des paramètres	Utilité	Type de retour	sources
afficherMenu()	Aucun	Aucun	Afficher le menu principal qui permet de lancer une partie, afficher les meilleurs et quitter l'application	void	partie.c
viderBaseDeDonnees()	BaseDeTest, *messageDeRetour	Bool, struct Dico_Message	Permets de vider les tables de la base de données sélectionnée grâce à l'argument BaseDeTest.	bool	score.c
creerBaseDeDonnees()	BaseDeTest, *messageDeRetour	Bool, struct Dico_Message	Permets de créer les tables de la base de données sélectionnée grâce à l'argument BaseDeTest	bool	score.c

Schéma de la base de données



Pour la base de données, j'ai créé un modèle E/A. Dans ce modèle, nous avons deux tables « joueurs » et « scores ». Chaque table possède une clé primaire « id_joueur » et « id_score ».
La table «joueurs» a pour but de stocker le pseudo du joueur.
La table « scores » a pour but de stocker le score d'un joueur grâce à son ID.

La relation entre la table « joueurs » et « scores » permet d'avoir une clé étrangère de «id_joueur» dans «scores».

Jeu de tests fonctionnels

Id	Use Cases	Nom du test	Description	Résultat attendu
1	Menu principal	Entrer autre chose qu'une proposition du menu	- Avoir lancé l'application - Appuyer sur une touche autre que « 1, 2, 3 , 4» - Appuyer sur votre touche «entrer»	Vous restez sur le menu
2	Menu principal	Entrer la proposition « 3 » activer / désactiver les couleurs	- Avoir lancé l'application - Appuyer sur votre touche « 3 » - Appuyer sur votre touche «entrer»	Les couleurs doivent s'afficher ou disparaître
3	Menu principal	Entrer la proposition « 4» pour quitter le jeu	- Avoir lancé l'application - Appuyer sur votre touche « 3 » - Appuyer sur votre touche «entrer»	Le jeu doit se fermer
4	Menu principal	Entrer la proposition « 2 » pour afficher les meilleurs scores	- Avoir lancé l'application - Appuyer sur votre touche « 2 » - Appuyer sur votre touche «entrer» - Appuyer sur votre touche «entrer»	Vous obtenez une page ayant pour titre « Meilleurs Scores ». Ensuite, appuyer sur « entrer » et vous retourner au menu principal.
5	Menu principal	Entrer la proposition « 1 » pour jouer	- Avoir lancé l'application - Appuyer sur votre touche « 1 » - Appuyer sur votre touche « entrer »	Le jeu se lance : le cadre s'affiche, et vous propose d'entrer un mot
6	Jouer	Entrer la proposition « * » pour afficher le mode débug	- Avoir lancé le jeu - Appuyer sur votre touche « * » - Appuyer sur «entrer»	La solution apparaît ou disparaît dans le contour du cadre supérieur

7	Jouer	Entrer un mot erroné (ne fais pas 4 caractères / contient un ou des chiffre(s) / n'est pas en minuscule..)	- Avoir lancé le jeu - Entrer un mot erroné - Appuyer sur « entrer » - Appuyer sur « entrer » - Appuyer sur « entrer »	Un message qui indique que vous avez abandonné ou que votre mot n'était pas correct s'affiche. La solution s'affiche. Appuyer sur « entrer », les meilleurs scores s'affichent. Appuyer sur « entrer » et vous retourner au menu principal.
8	Jouer	Entrer un mot non erroné (qui contient 4 caractères / ne contient pas de chiffre(s) / est en minuscule)	- Avoir lancé le jeu - Entrer un mot non erroné - Trouver la solution - Entrer votre pseudo - Appuyer sur «entrer» - Appuyer sur «entrer»	En fonction de la solution et du mot rentré, le jeu affiche le nombre de lettres bien placé et mal placé. Une fois la solution trouvée, entrer votre pseudo. Ensuite, appuyer sur « entrer », le menu des meilleurs scores s'affiche, appuyer sur « entrer », vous retourner au menu principal.
9	Jouer	Entrer un mot non erroné (qui contient 4 caractères / ne contient pas de chiffre(s) / est en minuscule)	- Avoir lancé le jeu - Entrer un mot non erroné - Trouver la solution - Appuyer sur «entrer» - Appuyer sur «entrer» - Appuyer sur «entrer»	En fonction de la solution et du mot rentré, le jeu affiche le nombre de lettres bien placé et mal placé. Une fois la solution trouvée, appuyer sur « entrer ». Votre pseudo devrait-être anonyme. Ensuite, appuyer sur « entrer », le menu des meilleurs scores s'affiche, appuyer sur « entrer », vous retourner au menu principal.
10	Jouer	Entrer un mot non erroné (qui contient 4 caractères / ne contient pas de chiffre(s) / est en minuscule)	- Avoir lancé le jeu - Entrer 10 mots non erronés - Ne pas trouver la solution - Appuyer sur «entrer» - Appuyer sur «entrer» - Appuyer sur «entrer»	Si vous ne trouvez pas la solution au bout de 10 essais, la solution s'affiche. Vous devez ensuite appuyer sur « entrer ». Ensuite, les meilleurs scores s'affichent. Appuyer sur « entrer », vous retourner au menu principal.

Description des logiques

1. Comparaisons des mots :

Pour ma fonction comparerMots, je vérifie si mon mot est correct. Ensuite, j'initialise mon nombre de lettres bien placé et mal placé à 0. Je crée deux copies temporaires des mots «solution» et « motPlace » qui permettent de modifier les mots sans modifier les originaux. Elle parcourt chaque lettre des mots, si une lettre est la même dans les deux mots, il incrémente d'un le nombre de lettres bien placées, elle remplace ces lettres par un espace afin de ne plus les comparer. Ensuite, elle parcourt à nouveau chaque lettre des mots, si une lettre est présente dans les deux mots, mais pas à la bonne place, elle incrémente d'un le nombre de lettres mal placées et remplace cette lettre par un espace.

2. Jouer une partie :

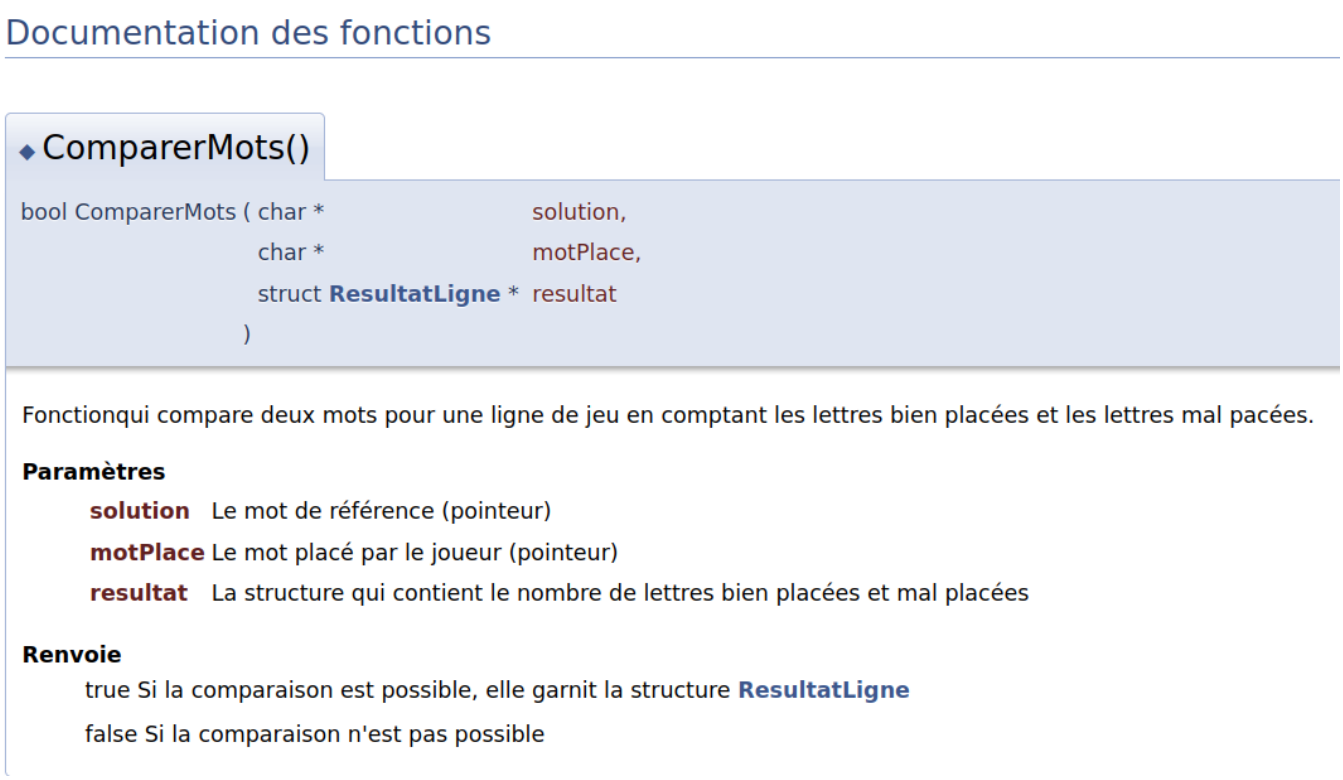
Pour ma fonction jouerPartie, je commence par déclarer un booléen «modeDebug », qui permettra de savoir si l'ont à activer où non le mode débug qui permet d'afficher la solution. Ensuite, elle entre dans une boucle qui continue tant que le joueur n'a pas trouvé la solution et que le nombre d'essais maximum n'a pas été atteint. Dans chaque tour de la boucle, elle affiche la partie en cours, puis demande au joueur d'entrer un mot.
Si le joueur entre « * », il active le mode débug, et le tour de la boucle se termine. Si le joueur entre autres choses, le mot est vérifié. Si le mot entré n'est pas correct, une erreur s'affiche, la solution est révélée, et la fonction se termine en renvoyant « false ». Si le mot entré est correct, il est comparé à la solution. Si le mot est identique, la partie en cours est affichée, le résultat est mis à true et la boucle se termine. Si le mot n'est pas identique à la solution, le nombre d'essais est incrémenté et la partie en cours est affichée. À la fin de chaque tour de la boucle, la réponse du joueur est libérée.

Éventuelles améliorations

Pour ce projet, on pourrait faire éventuellement une longueur de mot dynamique, et pas uniquement 4 caractères, cela rendrait le jeu plus amusant. Pour le reste, il me semble ne pas avoir d'autres améliorations, j'ai effectué divers dépassements afin d'embellir le jeu, et lors du développement j'ai fait très attention aux fuites de mémoire et aux respects des consignes.

Dépassements

En plus des dépassements proposés, j'ai également décidé de créer une « CDoc », une documentation semblable à la documentation JavaDoc disponible en Java. Pour ce faire, j'ai utilisé un outil gratuit disponible sous Linux nommé «doxygen». Cet outil permet de générer une documentation sous plusieurs formats différents à partir d'un commentaire formatée d'une façon spécifique. J'ai choisi de générer ma documentation en HTML comme la JavaDoc, grâce à la simplicité de parcourir celle-ci sur notre navigateur internet.
Pour tester celle-ci, vous pouvez ouvrir le fichier « index.html » qui est disponible dans le dossier HTML.
Voici un exemple de documentation pour la fonction « ComparerMots ».



Documentation des fonctions

◆ ComparerMots()

```
bool ComparerMots ( char *                  solution,
                    char *                  motPlace,
                    struct ResultatLigne *  resultat
                  )
```

Fonctionqui compare deux mots pour une ligne de jeu en comptant les lettres bien placées et les lettres mal pacées.

Paramètres

solution Le mot de référence (pointeur)

motPlace Le mot placé par le joueur (pointeur)

resultat La structure qui contient le nombre de lettres bien placées et mal placées

Renvoie

true Si la comparaison est possible, elle garnit la structure **ResultatLigne**

false Si la comparaison n'est pas possible

J'ai également utilisé un logiciel de détection de fuite de mémoire nommé Valgrind. Comme le montre la sortie de ma console ci-dessous, lorsque j'exécute mon programme afin de jouer une partie, à la fin de l'exécution de mon programme, j'obtiens un rapport de Valgrind. Ce rapport m'indique toutes les fuites de mémoires présentes. Nous pouvons voir que dans mon cas, les fuites de mémoires viennent de librairies externes, et ne sont pas causées par mon programme.

```
alexis@alexis-KLVC-WXX9:~/Bureau/Projet/mastermind$ valgrind --leak-check=full ./mastermind
==13731== Memcheck, a memory error detector
==13731== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==13731== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==13731== Command: ./mastermind
==13731==
==13731==
==13731== HEAP SUMMARY:
==13731==     in use at exit: 180,417 bytes in 255 blocks
==13731==   total heap usage: 4,274 allocs, 4,019 frees, 1,262,176 bytes allocated
==13731==
==13731== 9 bytes in 1 blocks are possibly lost in loss record 9 of 84
==13731==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==13731==    by 0x49CB58E: strdup (strdup.c:42)
==13731==    by 0x4904D54: ??? (in /usr/lib/x86_64-linux-gnu/libtinfo.so.6.3)
==13731==    by 0x4908098: _nc_tiparm (in /usr/lib/x86_64-linux-gnu/libtinfo.so.6.3)
==13731==    by 0x48D8238: newterm_sp (in /usr/lib/x86_64-linux-gnu/libncurses.so.6.3)
==13731==    by 0x48D8B4C: newterm (in /usr/lib/x86_64-linux-gnu/libncurses.so.6.3)
==13731==    by 0x48D8BDF: initscr (in /usr/lib/x86_64-linux-gnu/libncurses.so.6.3)
==13731==    by 0x10A6C6: InitialiserEcran (ecran.c:13)
==13731==    by 0x10D32B: main (mastermind.c:10)
==13731==
==13731== 24 bytes in 1 blocks are possibly lost in loss record 30 of 84
==13731==    at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==13731==    by 0x4A42F9E: tsearch (tsearch.c:337)
==13731==    by 0x4904D6E: ??? (in /usr/lib/x86_64-linux-gnu/libtinfo.so.6.3)
==13731==    by 0x4908098: _nc_tiparm (in /usr/lib/x86_64-linux-gnu/libtinfo.so.6.3)
==13731==    by 0x48D8238: newterm_sp (in /usr/lib/x86_64-linux-gnu/libncurses.so.6.3)
==13731==    by 0x48D8B4C: newterm (in /usr/lib/x86_64-linux-gnu/libncurses.so.6.3)
==13731==    by 0x48D8BDF: initscr (in /usr/lib/x86_64-linux-gnu/libncurses.so.6.3)
==13731==    by 0x10A6C6: InitialiserEcran (ecran.c:13)
==13731==    by 0x10D32B: main (mastermind.c:10)
==13731==
==13731== 168 bytes in 1 blocks are possibly lost in loss record 62 of 84
==13731==    at 0x484DA83: calloc (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==13731==    by 0x4904CCD: ??? (in /usr/lib/x86_64-linux-gnu/libtinfo.so.6.3)
==13731==    by 0x4908098: _nc_tiparm (in /usr/lib/x86_64-linux-gnu/libtinfo.so.6.3)
==13731==    by 0x48D8238: newterm_sp (in /usr/lib/x86_64-linux-gnu/libncurses.so.6.3)
==13731==    by 0x48D8B4C: newterm (in /usr/lib/x86_64-linux-gnu/libncurses.so.6.3)
==13731==    by 0x48D8BDF: initscr (in /usr/lib/x86_64-linux-gnu/libncurses.so.6.3)
==13731==    by 0x10A6C6: InitialiserEcran (ecran.c:13)
==13731==    by 0x10D32B: main (mastermind.c:10)
==13731==
==13731== LEAK SUMMARY:
==13731==    definitely lost: 0 bytes in 0 blocks
==13731==    indirectly lost: 0 bytes in 0 blocks
==13731==      possibly lost: 201 bytes in 3 blocks
==13731==    still reachable: 180,216 bytes in 252 blocks
==13731==         suppressed: 0 bytes in 0 blocks
==13731== Reachable blocks (those to which a pointer was found) are not shown.
==13731== To see them, rerun with: --leak-check=full --show-leak-kinds=all
==13731==
==13731== For lists of detected and suppressed errors, rerun with: -s
==13731== ERROR SUMMARY: 8 errors from 8 contexts (suppressed: 0 from 0)
alexis@alexis-KLVC-WXX9:~/Bureau/Projet/mastermind$
```

Conclusion

En conclusion, c'était un projet très amusant et pédagogique à réaliser, le fait de pouvoir travailler seul dessus permet de ne pas avoir de soucis de versions de code différentes d'un ordinateur à l'autre. L'utilisation d'une machine virtuelle globale pour tous est également une très bonne idée, cela permet d'avoir le projet compatible sur tous les ordinateurs. Le divers dépassement était très ludique à réaliser, ils permettaient d'améliorer le jeu. J'ai appris également à utiliser un logiciel gratuit et open source nommé « Valgrind » afin de détecter les fuites de mémoire, ce qui permet de réparer des fuites que je n'avais pas vues. J'ai également appris à configurer et utiliser doxygen afin de construire ma documentation.