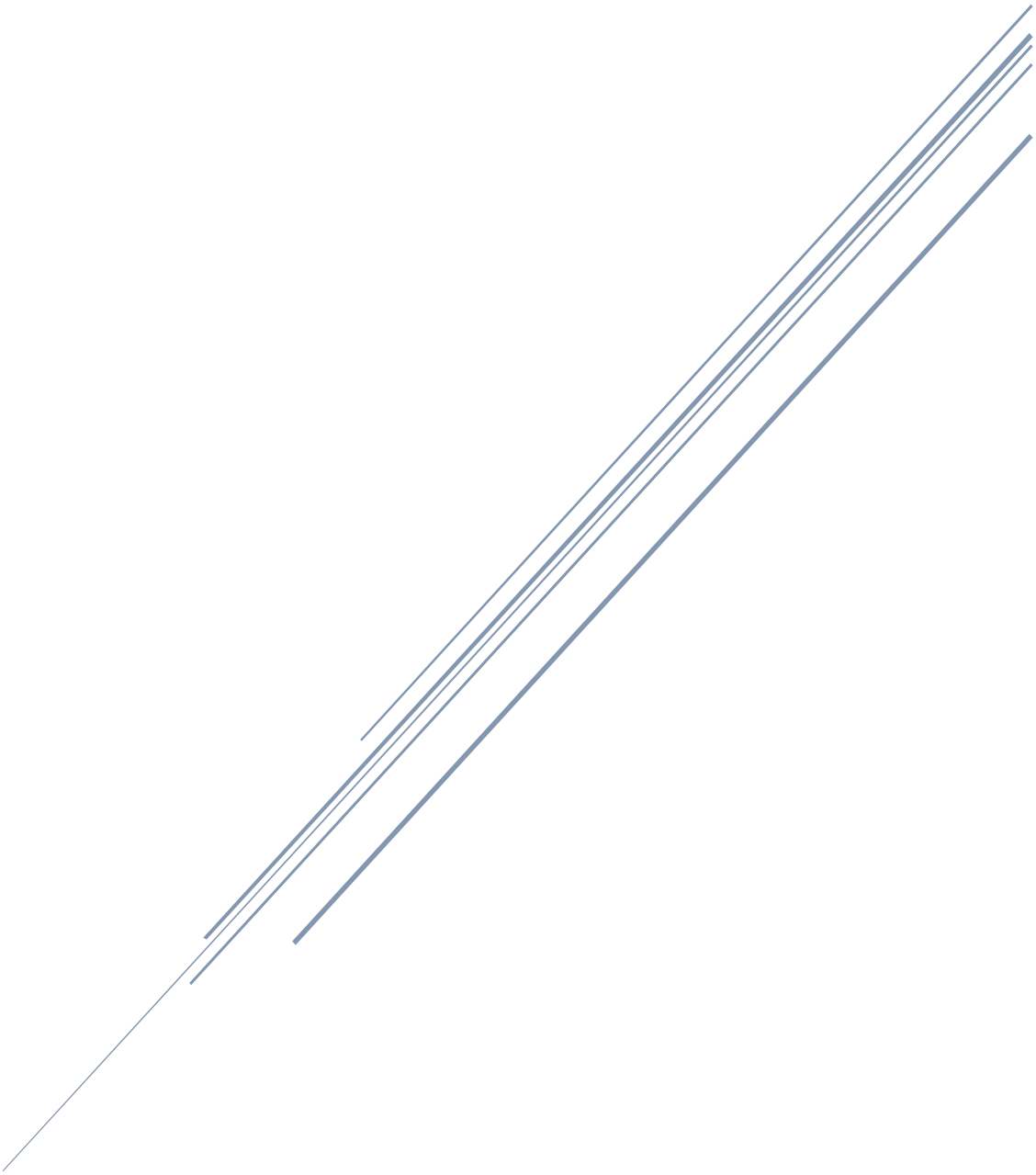
PROJET RUZZLE

Par Biardeau Tristan, Bussereau Keryann et Tabot Alexandre



Université du Maine
Outil de Développement

Bilan du Travail réalisé :

Pour le projet d'Algorithmique et outil de développement, il nous a été donné de réaliser par groupe de trois personnes un solveur d'un jeu de type ruzzle.

Pour nous organiser, nous avons, après l'analyse conceptuelle, ouvert un projet sur le site Trello, afin de nous répartir les différentes tâches.

Nous avons donc séparé les différentes fonctions essentielles du programme entre les trois membres du trinôme. Ce partage s'est trouvé plutôt équitable entre chacun de nous.

Cela permettait **de tester nos portions de codes séparément des autres portions**, et être sûr qu'elle fonctionnait avant de tout tester d'un coup.

Nous avons donc mis chacun nos fonctions dans le Trello, puis Keryann s'est chargé de tout déposer sur le github une fois que les fonctions principales avaient été créées.

Tristan explique son travail

Mes réalisations :

Fonctions réalisées :

- points.c
- matrice.c

Doxygen

De mon côté, je me suis chargé de la génération aléatoire de la matrice, et du comptage de points. J'ai également aidé mes camarades lorsqu'ils ont eu besoin d'aide pour déboguer certaines parties de leurs programmes. En effet, nous avions sur notre Trello une partie « Aide » permettant d'y déposer nos fonctions que l'on n'arrivait pas à faire fonctionner.

Pour finir, je me suis également chargé de la documentation Doxygen, utilisant le format Javadoc.

Fonction présentes dans le fichier source : points.c

La fonction compte_points reçoit en paramètre un mot dont elle renvoie un nombre de points en fonction des lettres présentes dans ce mot.

Fonctions présente dans le fichier source : matrice.c

La fonction afficherMatrice parcourt toute la matrice reçue en paramètre et en affiche chaque caractère.

La fonction rand_a_b renvoie un nombre aléatoire compris entre la valeur minimum et la valeur maximum reçues en paramétre.

La fonction getLettreRandom renvoie une lettre, selon ça fréquence d'apparition dans la langue française en fonction du nombre reçu en paramètre.

La fonction creaMatrice qui fait appel aux trois fonctions précédentes permet de créer une matrice avec des lettres aléatoires et d'afficher la matrice ensuite.

En ce qui concerne Doxygen je me suis chargé de faire toute la documentation des fonctions en m'aidant des commentaires laissés par mes camarades sur leurs fonctions. Ce fut assez fastidieux étant donné le nombre de fonctions.

J'ai trouvé ce projet enrichissant dans l'apprentissage d'une façon de travailler en groupe autour de l'informatique et de la programmation ainsi que dans la découverte des outils de programmation.

Keryann explique son travail

Mes réalisations :

Fonctions réalisées :

- search.c
- main.c

Github

Mise en commun des différents travaux

Pour ma part, je me suis chargé de faire la fonction de recherche des mots dans la matrice (search.c) ainsi que la fonction principale (main.c). Je me suis également chargé de faire le makefile et de rassembler toutes les fonctions, j'ai été confronté à beaucoup de problème pour la compilation. Je me suis aussi chargé de ranger le git et de rassembler tout ce que nous avons écrit pour le rapport final ainsi que de créer l'archive du rendu final.

Fonctions présente dans le fichier source : search.c

Dans la fonction init_tab on réinitialise un tableau avec des marqueurs de fin de mots pour chaque caractère afin de pouvoir réutiliser plusieurs fois ce tableau de caractères.

Dans la fonction concat on concatène les mots trouvés et le nombre de points qui lui est associé, pour cela on utilise les fonctions compte_points et ecrireFichier.

Dans la fonction chercheMatrice on se place sur chacune des cases de la matrice au fur à mesure et à chaque fois on vérifie au-dessus, à gauche, à droite, en bas et dans les différentes diagonales, pour cela on fait appel aux deux fonctions précédentes ainsi qu'à la fonction chercheDico.

Fonctions présente dans le fichier source : main.c

Dans la fonction main on fait seulement appel à la fonction creaMatrice puis à la fonction chercheMatrice.

Pour ce qui est de github je me suis chargé du rangement des dossiers en essayant de suivre au plus près les consignes, j'ai également essayé d'aider mes collègues à utiliser github.

En ce qui concerne la mise en commun des différents travaux j'ai récupéré les travaux de chacun d'entre nous pour ensuite les mettre en commun de façon à uniformiser notre travail.

J'ai trouvé ce projet fort intéressant puisqu'il nous a permis d'apprendre à travailler ensemble dans notre domaine qui sera notre futur puisqu'il s'agit de l'informatique. Cela nous a également permis de découvrir en appliquant les outils de programmation. Je n'ai pour ma part pas eu besoin de gdb puisque je n'ai pas eu de problème à l'exécution les seuls problèmes rencontrés étaient à la compilation.

Alexandre explique son travail

Mes réalisations :

Fonction réalisées :

- ruzzleEcrireFichier.c
- ruzzleDicoParCara.c
- ruzzleDicoParMot.c

Cunit

Gdb

J'ai réalisé les fonctions de manipulation de fichier pour chercher dans le fichier texte faisant office de dictionnaire si la combinaison de lettres évaluée durant la lecture de la matrice et d'écrire les mots valides trouvés.

J'ai réalisé ruzzleEcricreFichier pour remplir l'objectif d'écrire dans un fichier

A l'intérieur de celle-ci j'ai mis le commentaire définissant son rôle, comme pour toutes les autres fonctions que j'ai créées :

// Renvoie : la fonction écrit le mot passé en paramètre dans le fichier dont le nom est passé en paramètre

J'ai directement utilisé le paramètre « a » de la fonction fprintf car j'avais déjà réalisé des fichiers texte de sauvegarde. Ma 1e fonction de comparaison de mots dans le dictionnaire recherché dans tout le dictionnaire si le mot entier passé en paramètre existe dans le dictionnaire. Cette fonction utilisait les fonctions de manipulation et de comparaison de chaînes de caractères. Après discussion avec le groupe et notamment avec Keryann, qui était en charge de la conception de la fonction de recherche de mots dans la matrice grille, il est apparu opportun pour limiter les boucles de recherche et de consultation du dictionnaire d'ajouter de retour supplémentaire à la fonction. Cette nouvelle fonction devait définir si le tableau de caractère passé en paramètre formait le début de mots, un mot entier, ou un mot entier et le début d'autres mots dans le dictionnaire ou pas. Au vu des contraintes de recherche, la taille maximale d'un mot était de 4 caractères. Lors de la création de cette fonction, j'ai rencontré des difficultés au niveau de la comparaison des caractères de fin de lignes. Je ne suis pas parvenu à débloquer cette fonction, même avec l'utilisation du débogueur en ligne de commande *gdb*. Pour contourner cette difficulté, j'ai repris ma première fonction de recherche opérationnelle, je l'ai transformé pour qu'elle retourne les 4 valeurs de retour attendues par la fonction de recherche de mots dans la grille de Keryann. Cette fonction s'appelait au départ ruzzleDicoParMot et deviendra par commodité de nomenclature ; chercheDico. Son commentaire dès son début résume cela.

Les premiers tests unitaires ont été faits « à la main ». Dans le main d'une fonction test, j'ai appelé les fonctions testées avec les mots passés en paramètre suivant : abri, la, ab, aaaa. Je me suis également inspiré des matrices générées par les fonctions de Tristan. Ensuite, j'ai expérimenté l'utilisation des assertions par CUnit, qui s'est révélé être extrêmement long à comprendre pour moi est donc à nécessiter beaucoup de temps. Ce fut également le cas pour l'utilisation de la plate-forme

de partage de fichiers GitHub. J'ai passé plusieurs dizaines d'heures à résoudre mes problèmes de synchronisation avec les différents ordinateurs qui m'ont servi pour le projet Ruzzle.

Ce projet fut très intéressant pour ma part, mais je déplore le manque de temps pour l'appropriation des outils de collaboration et de débogage. Nous avions plusieurs projets et connaissances à pratiquer l'entité en même temps. En ce qui concerne notre organisation en équipe, nous sommes parvenus assez tôt à nous mettre d'accord sur le découpage en fonction du programme. Nous avons aussi fixé de manière claire les paramètres qu'attendent chaque fonction et les valeurs que chacune retourne. Je pense que notre répartition des tâches et fonctions à réaliser, a été juste, et cela s'est bien déroulé.