Projet L1 _ Promotion 2026 _ Groupe A

2021
2022

TAKUZU

Algorithmique et structure de données

Développé par :

Matthieu BRANDAO

Alexandre BAUDIN

eFrei

PARIS PANTHÉON-ASSAS UNIVERSITÉ

Présentation du projet :

Lors de notre second semestre nous avons étudié dans le module « Algorithmique et structure de données » le langage C. Afin de mettre en application nos connaissances du langage et des algorithmes, l'école a lancé le projet « Takuzu ».

Le Takuzu et un jeu de logique. Petit frère du très connu Sudoku, le principe est de poser des 1 et des 0 dans une grille carrée et selon des règles :

- Il ne doit y avoir plus de 3 fois le même chiffre à la suite dans une ligne ou une colonne.
- Dans chaque ligne où colonne il doit y avoir autant de 1 et de 0.
- Aucune colonne où ligne doit se retrouver en double dans la grille.

Le joueur commence avec une grille (de taille 4x4 ou 8x8). Quelques-uns des numéros de la grille sont déjà présents. Le but est de remplir complètement la grille en respectant les règles. Le joueur possède seulement 3 chances pour réussir. A chaque erreur une vie lui est retirée. Arrivé à 0 vie le joueur a perdu.

Les fonctionnalités proposées au joueur :

Arborescence du menu :

- Résoudre un 4x4
    - Choix du type de grille
    - Vérification par correspondance avec un modèle
        - Choisir un masque
        - Création intelligente du masque
    - Vérification intelligente
- Résoudre un 8x8
    - Choix du type de grille
    - Vérification par correspondance avec un modèle
        - Choisir un masque
        - Création intelligente du masque
    - Vérification intelligente

Choix du type de grille :

Le principe est de pouvoir choisir si l'on veut jouer sur une grille prédéfinie ou créée par l'ordinateur. Dans le cas où « grille prédéfinie » est sélectionné, le joueur continuera sur l'une des grilles au hasard présentent dans le code. Dans le cas où « création intelligente de la grille » est sélectionné, le programme crée la grille en fonction des règles du jeu, le joueur pourra alors jouer dessus.

Vérification par correspondance avec un modèle :

Le principe est de vérifier chacun des coups du joueur d'après un modèle prédéfini.
Lorsque le coup est joué une vérification est lancée. Le jeu compare le coup avec la grille-modèle. Si le coup correspond bien à celui attendu le joueur verra apparaitre « So good » et la grille sera dévoilée à la position jouée. Si au contraire le coup ne correspond pas à celui attendu, je joueur verra apparaitre « So bad », une vie lui sera retirée et aucune case ne sera dévoilée.

Vérification intelligente :

Le principe est de vérifier chacun des coups du joueur grâce aux règles du jeu.
Dans le code du jeu sont implémentées les règles. Lorsqu'un coup est joué, il est traité et passe par la vérification de chaque règle. Si le coup respecte toutes les règles le joueur verra apparaitre « So good » et la grille sera dévoilée à la position jouée. Au contraire si l'une des règles n'est pas respectée, je joueur verra apparaitre « So bad », une vie lui sera retirée et aucune case ne sera dévoilée.

Choisir un masque :

Le principe est de pouvoir sélectionner un masque d'après ceux déjà existants.
Après la saisie de « choisir un masque », l'ensemble des masques enregistrés apparaît. Ils sont numérotés. Il ne reste plus qu'à saisir le masque souhaité et le jeu se lancera en appliquant celui-ci.

Création intelligente du masque :

Le principe est de créer un masque en fonction du niveau entré.
Après la saisie de « création intelligente du masque », le niveau est demandé (plus il est élevé, plus la grille sera cachée) le jeu créera un masque aléatoire d'un maximum de la moitié des case cachées (le niveau maximum est sélectionné). Le jeu pourra alors se lancer en appliquant le nouveau masque.

Présentation technique :

Les fonctions importantes :

- Start_Game() : Démarre le jeu. Elle a comme paramètre la grille, le masque, et le type de vérification.

    Algorithme associé :

    - Initialisation des vies
    - Affichage de « Le jeu commence »
    - Si le mode de vérification est « par correspondance »
        - Faire
            - Afficher « A vous de jouer. » et « vies »
            - Récupération du coup joué
            - Si le coup est bon
                - Afficher « So good »
                - Dévoiler le masque à l'endroit du coup
            - Sinon (si le coup est faux)
                - Afficher « So bad »
                - Retirer une vie
            - Fin Si
        - Tant que la grille n'est pas complètement remplie ou que les vies ne sont pas à 0
    - Fin Si

    - Si le mode de vérification est "indices pour les coups invalides"
        - Faire
            - Afficher « A vous de jouer. » et « vies »
            - Récupération du coup joué

        - Si le coup est valide
            - Si le coup est correct
                - Afficher “coup valide et correct
                - Dévoiler le masque à l’endroit du coup
            - Si le coup est incorrect
                - Afficher “coup valide mais incorrect
        - Si le coup est invalide
            - Afficher le message d’erreur (l’indice)
            - Enlever une vie
        - Fin Si
    - Tant que la grille n’est pas complètement remplie ou que les vies ne sont pas à 0

- S’il reste des vies
    - Afficher « Vous avez gagné »
- Sinon (il ne reste plus de vies)
    - Afficher « Vous avez perdu »
- Fin Si

- Start_Automatic_Game() : Démarre le jeu. Elle a comme paramètre la grille et le masque
    - Algorithme associé :
        - Affichage de “l’ordinateur commence à jouer”
        - Affichage de la grille masquée
        - Récupération du nombre de cases masquées
        - Pour i allant de 0 au nombre de cases masquées :
            - Récupérer un choix valide grâce à la fonction Intelligent_choice()
            - La fonction Intelligent_choice() affiche la raison de son choix
            - Afficher le choix sous forme “position ligne \n , position colonne \n, choix”
            - Attendre 4 secondes
            - Dévoiler la case du masque à la position du choix
            - Afficher la grille
            - Attendre 4 secondes
        - Fin Pour

- create_matrix() : Création d’une matrice. Elle a comme paramètre la taille.
    Algorithme associé :
        - Initialisation de Dyn_tab
        - Création de l’espace mémoire pour la matrice 2D
        - Affectation de la taille à Dyn_tab
        - Retourne Dyn_Tab

- Fill_matrix() : Remplissage de la matrice avec une grille existante. Elle a comme paramètres la taille et la version de la grille.
    Algorithme associé :
        - Création de la matrice avec la taille

        - Affectation du type « grid » à la matrice
        - Si la taille de la matrice est 4
            - Pour i allant de 0 à la taille
                - Pour j allant de 0 à la taille
                    - Matrice[i][j] = Gates4[version][i][j]
                - Fin Pour
            - Fin Pour
        - Sinon (taille = 8)
            - Pour i allant de 0 à la taille
                - Pour j allant de 0 à la taille
                    - Matrice[i][j] = Gates8[version][i][j]
                - Fin Pour
            - Fin Pour
        - Fin Si
        - Retourne Matrice

- Fill_mask() : Remplissage d'un masque avec un masque existant. Elle a comme paramètres la taille et la version du masque.

    Algorithme associé :
        - Création de la matrice avec la taille
        - Affectation du type « mask » à la matrice
        - Si la taille de la matrice est 4
            - Pour i allant de 0 à la taille
                - Pour j allant de 0 à la taille
                    - Matrice[i][j] = Mask4[version][i][j]
                - Fin Pour
            - Fin Pour
        - Sinon (taille = 8)
            - Pour i allant de 0 à la taille
                - Pour j allant de 0 à la taille
                    - Matrice[i][j] = Mask8[version][i][j]
                - Fin Pour
            - Fin Pour
        - Fin Si
        - Retourne Mask

- move_like_model() : Vérifie si le coup est le même que sur le modèle. Elle a comme paramètres le coup(position, valeur) et la grille(modèle).

    Algorithme associé :
        - Si la grille à la position du coup est égale à la valeur du coup
            - Retourner vrai
        - Fin Si
        - Retourner faux

- print_matrix() : Affiche la matrice au complet. Elle a comme paramètre la matrice.

    Algorithme associé :
        - Affiche la ligne du dessus composé de lettres (ex. A B C D ..)
        - Pour i allant de 0 A la taille de la matrice

            - Afficher « i »
            - Pour j allant de 0 A la taille de la matrice
                - Afficher « j »
            - Fin Pour
            - Revenir à la ligne
        - Fin Pour

- print_matrix_mask() : Affiche la matrice selon le masque. Elle a comme paramètre la matrice.
    Algorithme associé :
        - Affiche la ligne du dessus composé de lettres (ex. A B C D ..)
        - Pour i allant de 0 A la taille de la matrice
            - Afficher « i »
            - Pour j allant de 0 A la taille de la matrice
                - Si le masque à la position i et j est égal à 1
                    - Afficher « j »
                - Sinon (est égal à 0)
                    - Afficher « »
                - Fin Si
            - Fin Pour
            - Revenir à la ligne
        - Fin Pour

- move_input() : Récupère la position du coup et la valeur du coup. Elle a en paramètre la matrice.
    Algorithme associé :
        - Initialisation de move
        - Move[1] est égal à la ligne entrée (lettre)
        - Move[2] est égal à la colonne entrée (chiffre)
        - Move[3] est égal à la ligne entrée (0 ou 1)
        - Retourne move

- check_mask_1() : Vérifie si le masque est complètement dévoilé. Elle a en paramètre le masque.
    Algorithme associé :
        - S=0
        - Pour i allant de 0 à la taille du masque:
            - Pour j allant de 0 à la taille du masque:
                - S prend S + la valeur de la case i,j du masque
            - Fin pour
        - Fin pour
        - Si S=taille du masque au carré
            - Retourne vrai
        - Sinon
            - Retourne faux

-Verif_move() : vérifie si le choix est valide, prend en paramètre la matrice, le masque et le choix
Son mode de fonctionnement un peu particulier peut être résumé ainsi :

- Si le choix n'est pas entouré de deux chiffres identiques à lui-même sur la ligne :
    - Si le choix " " " " " sur la colonne :
        - Si le choix n'est pas à coté de deux chiffres identiques sur la ligne
            - Si " " " " " sur la colonne :
                - Si le choix est le dernier chiffre à placer sur la ligne et qu'il est bon :
                    - Si " " " sur la colonne " " " :
                        - Et ainsi de suite jusqu'à exécuter tous les tests
                        - Si tous les tests sont réussis :
                            - Retourner 0
                        - S'il y a des règles que l'on a pas pu vérifier
                            - Retourner –1
                    - Sinon:
                        - Retourner 20
                - Sinon:
                    - Retourner 19
            - Sinon …
        …
    …
…

-Ainsi dès qu'une règle du Takuzu n'est pas respectée, on retourne un entier spécifique.

-Indice() : prend en paramètre la matrice, le masque et le choix, utilise la fonction verif_move() et en fonction de l'entier retourné par cette dernière, elle retourne un message dictant quelle règle qui n'a pas été respectée.

-Intelligent_choice():prend en entrée la grille et le masque.

- Son algorithme associé est :
- On récupère la liste des cases à découvrir, de taille N
- Pour i allant de 1 à N :
    - Le choix=1
    - Si le coup est valide :
        - On inverse le choix (choix=0)
        - On utilise la fonction indice pour afficher la règle que le choix inversé ne respecte pas
        - On ré-inverse le choix
        - On retourne le choix (et la position)
    - Si le coup est invalide :
        - On utilise la fonction indice pour afficher la règle que le choix ne respecte pas
        - On inverse le choix
        - On retourne le choix (et la position)
    - Si le coup est pseudo-valide (c.a.d qu'il ne viole aucune règle mais jouer l'inverse du choisi ne violerait aucune règle non plus) :
        - Si i=N-1 (c.a.d que c'est le dernier coup possible à jouer):

- On pioche la solution dans la grille de référence
- On n'affiche « pas d'indice disponible, l'ordinateur pioche la solution »
- On retourne la solution (et la position)

- Sinon:
  - On ne fait rien et on passe à l'itération suivante

Les structures de données :

Nous avons fait le choix de stocker nos masques et grilles sous forme de tableaux statiques. Cependant, dès le lancement du jeu, des fonctions (create_matrix() et fill_matrix()) traiteront ces tableaux statiques pour les transformer la grille et le masque de la partie en tableaux dynamiques. Tout au long du programme ne seront utilisé que ces tableaux dynamiques.

Les tableaux statiques ont l'avantage de pouvoir être stocké en dur dans le code. Plus simple à visualiser pour une programmation des fonctions de traitements.

Les tableaux dynamiques ont l'avantage d'être plus simple à manipuler.

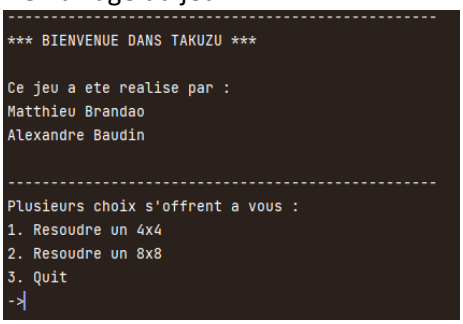
Difficultés techniques rencontrées :

- Lors de la saisie du coup, pour la colonne un entier est attendu or lorsqu'on entrait un caractère minuscule le programme tournait en boucle. Nous nous sommes donc aidés d'internet et avons trouvé la solution d'ajouter « Scanf(« %*c ») ; » après notre saisie.
- Sur l'un de nos ordinateurs, après une modification du code dans CLion des fichiers « CMake-Build », indésirables au commit, se logeait dans les fichiers à commit. Nous avons donc créé un fichier « .gitignore » nous permettant ignorer ces fichiers.

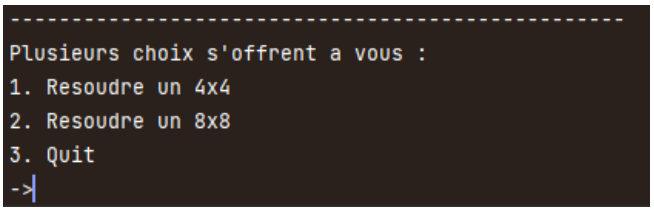
Présentation des résultats

Les menus

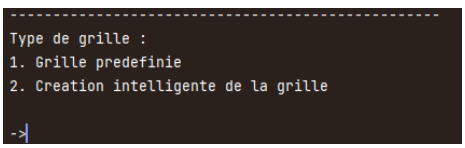
Démarrage du jeu :



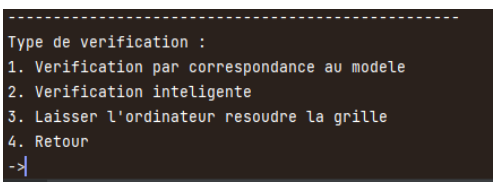
Lors d'un retour au menu :



2nd menu :



3e menu :



5e menu :

```
------------------------------------------------
Quel mode?
1. Choisir un masque
2. Creation intelligente du masque
3. Retour
->
```

Ces menus se présenteront toujours en premier et dans cet ordre

Dans le cas de « Choisir un masque » les masques prédéfinis s'afficheront :

```
Voici les masques existants :

1.
   A  B  C  D
  -- -- -- --
1| 1  1  1  1
2| 0  0  0  1
3| 0  0  0  1
4| 0  0  0  1

2.
   A  B  C  D
  -- -- -- --
1| 0  0  1  1
2| 1  1  1  1
3| 1  1  1  1
4| 1  1  1  1
Quelle est votre choix?
->
```

1er masque

2e masque

Lancement du jeu :

Début :

```
------------------------------------------------
Le jeu commence !
------------------------------------------------
   A  B  C  D
  -- -- -- --
1|       0  0
2| 0  0  1  1
3| 0  1  0  1
4| 1  0  1  0
```

Grille avec le masque appliqué dessus

Saisie du coup :

```
A vous de jouer.                    vie(s) : 3
ligne :
->1
colonne :
->A
choix de la valeur placee (0 ou 1) :
->1
```

Nombre de vies restantes

Ligne et colonne du coup

En cas d'erreur :

Si l'option « vérification d'après un modèle » sélectionnée précédemment :

```
------------------------------------------------
So bad. Le coup ne correspond pas. Une vie en moins!
------------------------------------------------
```

Vie retirée

Si l'option « vérification intelligente » sélectionnée précédemment :

```
------------------------------------------------
So bad,
sur une colonne il doit y avoir autant de 1 que de 0
------------------------------------------------
```

Vie retirée + correction expliquée

En cas de bon coup :
Si l'option « vérification d'après un modèle » sélectionnée précédemment :

```
------------------------------------------------
So good,
le coup est valide et correcte
------------------------------------------------
```

Confirmation du coup

Si l'option « vérification intelligente » sélectionnée précédemment :

```
------------------------------------------------
So good
------------------------------------------------
```

Confirmation du coup

Dans le cas de la sélection de « Laisser l'ordinateur résoudre la grille » :

```
------------------------------------------------
Resonnement : au dessus de deux 0 il doit y avoir un 1

Le coup joue est:
Ligne : 1
Colonne : A
Choix : 1

   A  B  C  D
  -- -- -- --
1| 1     0  0
2| 0  0  1  1
3| 0  1  0  1
4| 1  0  1  0

------------------------------------------------
```

Résonnement pour effectuer le coup

Coordonnées du coup joué

Coup appliqué sur la grille

Fin du jeu :

En cas de victoire :

```
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
Vous avez Gagne
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
```

En cas d'échec :

```
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
Vous avez Perdu
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
```

Quelques tests :
Jeu complètement automatique (4x4) :
1/2/3/2/[Niveau]

Jeu avec des masques et grilles prédéfinis à résoudre sois même (8x8) :
2/1/1/1/[Masque]/[Jouer]

Bilan

Notre première approche du projet était celle d'un défi à relever. En effet des mathématiques se cachent derrière le bon fonctionnement du jeu. De plus il faut être à l'aise avec le langage C afin de programmer sans difficultés. Il ne restait plus qu'à appliquer les bonnes formules avec le bon langage. Lors de ce projet nous avons donc pu tirer plusieurs enseignements.

Tout d'abord, nous avons pu exercer le traitement des tableaux dynamiques ainsi que des pointeurs. Un chapitre un peu complexe vu en cours qui ne nous a pas fait de mal de revoir. Nous avons pu aussi approfondir notre compréhension des liens entre les fichiers .c et .h.

Ensuite, nous avons pu travailler notre faculté à travailler en groupe. Nous avons organisé des réunions « avancement de projet » afin de se mettre d'accord sur les taches réaliser, celles restantes et leurs répartitions. Nous avons appris à utiliser Git et GitHub correctement pour partager notre code du jeu et gérer les versions du code.

Pour finir, nous avons dû apprendre à gérer notre temps. Les examens et les autres projets possédant des dates de rendu proche de celle du projet « Takuzu », nous avons alors eu besoin de le commencer tôt, de le gérer au cours du temps, et d'estimer le temps de programmation de chaque partie du code.

Le projet fini avec ses toutes dernières modifications le jour du rendu, nous vous déposons alors ce rapport espérant que le jeu vous plaira et que vous résoudrez toutes les grilles.