Projet de Programmation Informatique en C
Jeu du Pendu

FOUCRAS Baptiste
TOURÉ Makalé
2022-2023

ENSIM
École d'ingénieurs
Le Mans Université

Sommaire

Introduction

Dans le cadre du mini-projet de programmation informatique, nous avons choisi de coder le jeu du pendu. Il s'agit d'un jeu dont le but est de trouver un mot avant que le pendu ne se forme, sous peine de le perdre.

Chaque erreur fait apparaître la partie suivante du pendu, permettant au total 11 erreurs.

Lancement du programme

Si vous possédez l'utilitaire GNU Make[1], il suffit d'aller au dossier racine du projet.
Depuis ce dossier, utilisez la commande "make run".

Si vous ne possédez pas GNU Make[1], il suffit d'effectuer les instructions ci-dessous:

1. gcc src/*.c src/alphabet/alphabet.c src/dictionary/dictionary.c src/embeddedString/embeddedString.c src/utils/print/print.c -o hangman.exe -Wall -Wextra
2. ./src/hangman.exe

La commande "make run" effectue tout simplement les deux lignes au dessus

Tapez "make" ou "make help" pour avoir une explication sur chaque règle présente dans le fichier "Makefile". Sera notamment présente une règle pour effectuer tous les tests unitaires.

Notez que tout programme exécuté autre part que depuis le dossier racine du projet finira inévitablement par une erreur, les fichiers ne pouvant être ouverts.

ALGORITHME des fonctions principales

1) Fonction Hangman(const Dictionary *tabMots) : entier

début
choixLettre, *hasardMotStr : chaîne de caractère
tentatives, nbLettersFinded, typeChar : entier;
tentative ← 11
hasardMot : EmbeddedString (structure créée) ;
*alphabet : Alphabet (structure créée) ;

Si la taille logique de l'objet tabMots est égale à zéro:

Affichez un message indiquant qu'il n'y a pas de mots disponibles et qu'il faut en saisir.

Appelez la fonction putWordsToFile pour remplir tabMots à partir d'un fichier de mots.

Fin Si

Appelez la fonction getRdmStr pour obtenir un mot au hasard à partir de l'objet tabMots qu'on placera dans hasardMotStr.

Si aucun mot n'est obtenu:

Retournez -1.

Fin Si

Allouez de la mémoire pour l'objet hasardMot en fonction de la longueur du mot obtenu au hasard (hasardMotStr) et en ajoutant 1 pour le caractère de fin de chaîne.

Si l'allocation de mémoire échoue

Retournez -1.

Fin Si

Appeler la fonction "transformInEmbeddedStr" en lui passant "hasardMot" et "tabMots[rdm]", et vérifier que le résultat est non-nul

Si le résultat est nul

Libérer la mémoire de "hasardMot" et "tabMots"

Retourner -1

Fin Si

Appeler la fonction "createAlphabet" en lui passant l'adresse de "alphabet", et vérifier que le résultat est non-nul

Si le résultat est nul

Libérer la mémoire de "hasardMot", "alphabet" et "hasardMotStr"

Retourner -1

Fin Si

Nettoyer l'interface utilisateur grâce à la fonction clearConsole().

Tant qu'il reste des tentatives et que le mot caché (hasardMot) n'est pas trouvé.

Affichez l'interface utilisateur en appelant la fonction printUserInterface avec comme arguments (11-tentatives, &hasardMot, alphabet)

Positionner le curseur à l'endroit souhaité avec la fonction consoleGotoCoords

Afficher le nombre de tentatives restantes

Demandez à l'utilisateur de proposer une lettre en utilisant la fonction scanf. Utilisez la fonction emptyStream pour vider toute entrée supplémentaire de l'utilisateur.

Utilisez la fonction isProposedLetterValid pour vérifier si la lettre proposée a déjà été soumise ou si elle est valide.

Tant que la lettre n'est pas valide :

Demandez à l'utilisateur de rentrer une nouvelle lettre jusqu'à ce qu'une lettre valide soit entrée.

Fin Tant que

Si la lettre proposée est valide et que typeChar est égal à 2 (on cherche à deviner le mot):

Demandez à l'utilisateur d'entrer le mot caché.

Utiliser la fonction toLowerCase s'il le rentre en majuscule

Selon (mixedStrcmp(hasardMot, suggestedStr))

S'il vaut -2 :

Afficher « null pointer »

Libérer alphabet, hasartMot et hasardMotStr

Jeu du pendu

Retourner -1

S'il vaut 0 :

Afficher « Incroyable ! le mot était bien hasardMotStr en 11-tentatives »

Libérer alphabet, hasartMot et hasardMotStr

Retourner -1

Default

Afficher « Dommage, ce n'était pas le bon mot ! Vous perdez une tentative !")

Décrémenter tentatives de 1

Fin Si

Sinon

Mettez à jour l'alphabet en appelant la fonction updateAlphabet avec l'alphabet et la lettre proposée.

Appelez la fonction updateFindEmbeddedStr avec le mot caché et la lettre proposée dans nbLettersFinded.

Selon (nbLettersFinded – 2)

Si "nbLettersFinded" - 2 vaut -3, afficher "null pointer for hasardMot", libérer la mémoire de "tabMots", "alphabet", et "hasardMot", puis retourner -1

Si "nbLettersFinded" - 2 vaut -2, afficher un message indiquant que la lettre proposée n'est pas présente dans le mot à deviner, décrémenter "tentatives" de 1

Si "nbLettersFinded" - 2 vaut -1, afficher « Bravo, vous avez trouvé %d lettres.\n", nbLettersFinded »

Si "nbLettersFinded" - 2 vaut 0, afficher « GG, vous avez trouvé %d lettres.\n", nbLettersFinded »

Default, afficher « Incroyable, vous êtes parvenu(e) à trouver %d lettres en une fois !!!\n", nbLettersFinded »

Fin Selon

Fin Sinon

Fin Tant que

Affichez l'interface utilisateur en appelant la fonction printUserInterface avec les arguments (11 - tentatives, &hasardMot, alphabet).

S'il reste des tentatives :

Affichez un message indiquant que le mot a été trouvé

Libérer alphabet, hasardMotStr, hasardMot

Retournez 1

Fin Si

Sinon

Affichez un message indiquant que l'utilisateur a perdu et le mot caché

Libérer alphabet, hasardMotStr, hasardMot

Retournez 0.

Fin Sinon

Fin

2) Fonction menu(vide) : vide

//nous avons créé un menu qui va appeler notre fonction hangman
début
choix : chaîne de caractère
*wordsAvailable : pointeur sur Dictionary, *wordsAvailable← NULL
initConsole(); // pour rendre les terminaux windows compatible avec les ansi escape statement
Nettoyer l'interface utilisateur
Faire:

Jeu du pendu

Affichez les options de menu en appelant la fonction printMenuOptions.

Demandez à l'utilisateur de saisir une option en utilisant la fonction fgetc, le mettre dans « choix »

Vider le buffer avec la fonction emptyStream.

Selon l'option choisie par l'utilisateur :

Si l'utilisateur a choisi 'q' ou 'Q', quittez le menu.

Si l'utilisateur a choisi 'a'

Effacez l'interface utilisateur

Chargez les mots disponibles en appelant la fonction loadWords

Si les mots n'ont pu être chargé

Afficher un message d'erreur

Fin si

Affichez-les mots disponibles à l'écran

Demandez à l'utilisateur d'appuyer sur ENTREE pour retourner au menu.

Si l'utilisateur a choisi 'b'

Effacez l'interface utilisateur

Chargez les mots disponibles en appelant la fonction loadWords

Si aucun mot n'a pu être chargé, retourner un message d'erreur

Démarrez un jeu de pendu en appelant la fonction Hangman.

Si l'utilisateur a choisi 'c'

Dictionary *newerTab

Chargez les mots disponibles en appelant la fonction loadWords

Si aucun mot n'a pu être chargé, retourner un message d'erreur

Appeler la fonction putWordsToFile et mettre le résultat dans newerTab

Si newerTab est nulle, afficher un message d'erreur quant à l'allocation de mémoire

Sinon si la taille logique de newerTab < la taille logique de wordsAvailable, afficher un message d'erreur quant à la sauvegarde des fichiers

Libérer l'espace occupé

Default

Afficher « le choix n'a pas été reconnu »

Jusqu'à ce que l'utilisateur quitte le menu en choisissant l'option 'q' ou 'Q'.

Libérer l'espace occupé

Fin

Pour toutes les fonctions utilisées, leur documentation sera disponible en suivant le chemin suivant à partir du dossier doc > program_documentation > html > index.html

AFFICHAGE

Dans cette section, il sera expliqué les principaux choix pour l'affichage, ainsi que des screens permettant d'illustrer chaque partie de l'application de terminal.

En figure 1 se trouve le menu du jeu, lorsque vous exécutez le programme. C'est le point d'entrée de l'application, et propose 3 choix principaux:
"afficher la liste de mots possibles lors du pendu" permet d'afficher les mots que vous pouvez avoir à deviner lors d'une partie, son affichage se trouve en figure 2.

La seconde option permet de lancer une partie de pendu, dont l'affichage est trouvable en figure 3.

La dernière option permet d'ajouter depuis le programme des mots que vous pourrez ensuite deviner lors des parties. La figure 4 reprend son ajustement prévu dans un terminal.

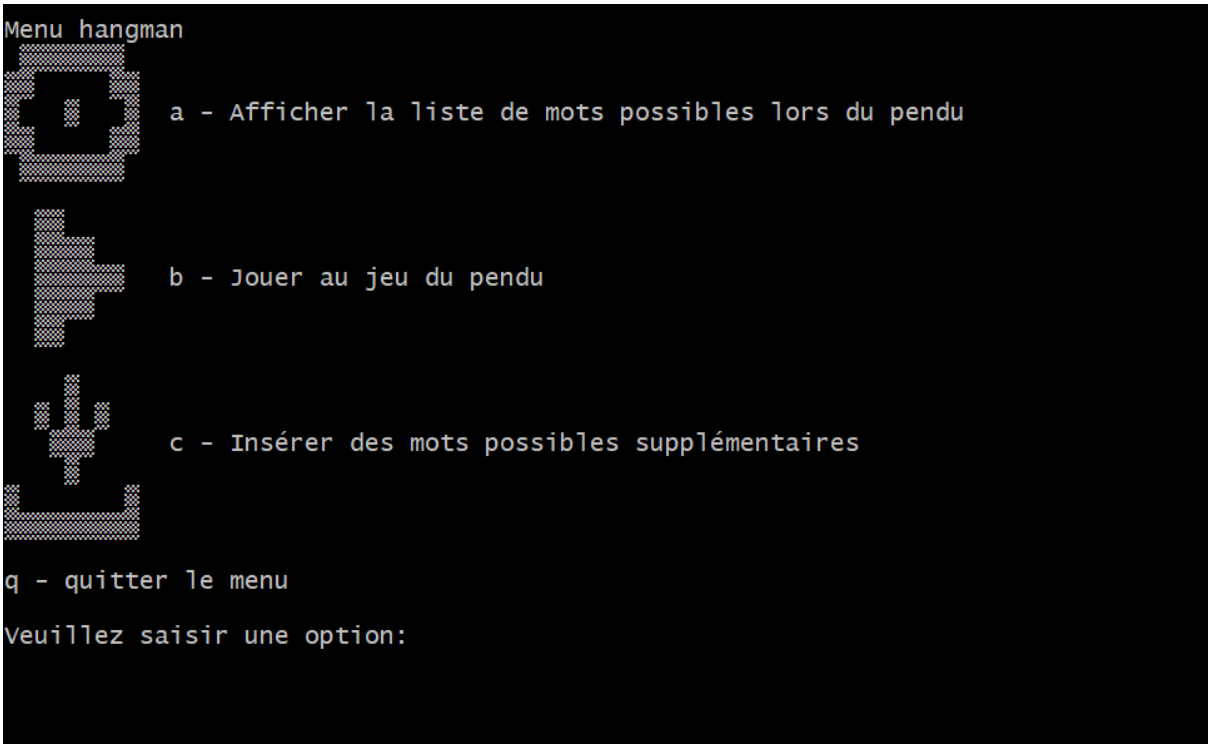

figure 1: affichage du menu du jeu

Tout l'affichage se base sur les "ANSI espace sequence"[2]. En effet, l'avantage de ce code est qu'il est normalement supporté par tous les terminaux modernes, bien qu'il faille pour ceux Windows activer des paramètres (voir la fonction "initConsole" dans src/utils/print/print.h pour la fonction activant ces paramètres).

Les pixels arts sont produits à partir de fichiers de données (présents dans src/ressources/). Chaque pixel est représenté par une minuscule (les espaces et caractères non minuscules sont ignorés), et sa valeur indique si le pixel doit être plein ou vide. Cette méthode a pour avantage de facilement pouvoir modifier chaque image, par rapport à un codage brut dans le code.

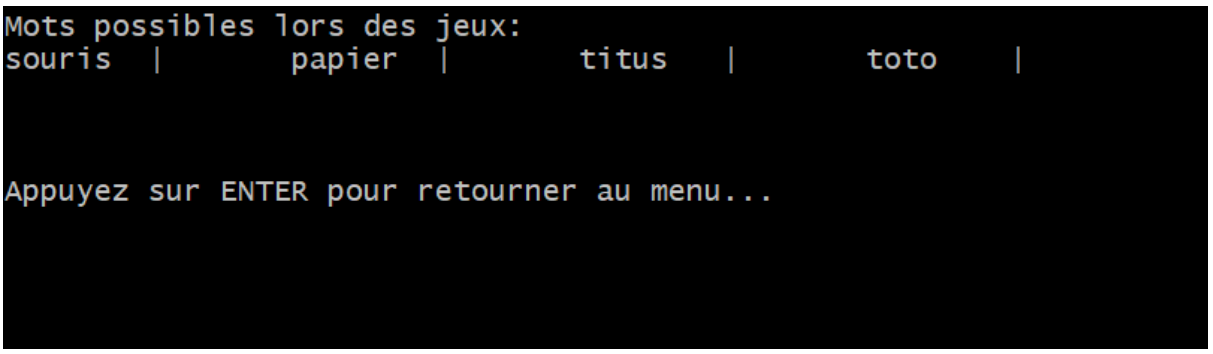

figure 2: affichage des mots possibles (choix 'a' du menu)

Lorsque le choix 'a' du menu en figure 1 est choisi, le contenu de "src/ressources/dictionary.don" est récupéré. Chaque mot composant ce fichier est affiché en ligne sur cet écran.

figure 3: affichage lors d'une partie du jeu du pendu (choix 'b' du menu)

En figure 3 se trouve l'affichage du jeu du pendu en lui-même.
Chaque partie de l'écran est placé à un endroit précis du shell à l'aide des "ANSI escape code"[2], et ici de la fonction "consoleGoToCoords(int x, int y)" (dans src/utils/print/print.h). Cela permet d'afficher à des endroits non possible sans les éléments de l'interface.
L'affichage du pendu est plus précisément expliqué dans la partie des problèmes rencontrés.

figure 4: affichage lors de la saisie de nouveaux mots (choix 'c' du menu)

Prendre le choix 'c' lors du menu permet de saisir de nouveaux mots.
Toujours à l'aide de "consoleGoToCoords", les mots déjà présents, récupéré du fichier "src/ressources/dictionary.don" sont affichés.

PROBLÈMES RENCONTRÉES ET SOLUTIONS

Un des soucis principaux lors de la rédaction du code fut de produire un code réutilisable, extensible et robuste. Afin d'atteindre ce but, de nombreuses structures et fonctions ont été créées. Cela permet d'obtenir un code réutilisable, plus simple à corriger car reprend le principe de petits modules. Par exemple, la structure "embeddedChar" permet, outre la chaine de caractère qui reste connue, de savoir quelles lettres de cette dernière ont été trouvées jusque là.

L'un des autres problèmes rencontrés fut l'affichage du pendu. Nous avons décidé de le faire grâce à un fichier comme présenté en figure 5, semblable à celui des images du menu en figure 1, et au principe des images "Portable Pixmap"[3].

```
a a a a a a a a a a a a a a a a a a a a a a a a a a
a a a c d d d d d d d d d d d d d d d d d d a a a
a a a c a a a a e a a a a a a a a a f a a a a a a a
a a a c a a a e a a a a a a a a a a f a a a a a a a
a a a c a a e a a a a a a a a a a g a a a a a a a
a a a c a e a a a a a a a a a a g a g a a a a a a
a a a c e a a a a a a a a a a a g a g a a a a a a
a a a c a a a a a a a a a a a a g a a a a a a a a
a a a c a a a a a a a a a a a a h a a a a a a a a
a a a c a a a a a a a a a a a a h a a a a a a a a
a a a c a a a a a a a a a a a j h i a a a a a a a
a a a c a a a a a a a a a a j a h a i a a a a a a
a a a c a a a a a a a a a a a h a a a a a a a a a
a a a c a a a a a a a a a a a h a a a a a a a a a
a a a c a a a a a a a a a a l h k a a a a a a a a
a a a c a a a a a a a a a a l a a a k a a a a a a
a a a c a a a a a a a a a a l a a a k a a a a a a
a a a c a a a a a a a a a a a a a a a a a a a a a
a a a c a a a a a a a a a a a a a a a a a a a a a
a a a c a a a a a a a a a a a a a a a a a a a a a
a b b b b b b b b b b b b b b a a a a a a a a a a
```

figure 5: contenu du fichier src/ressources/hangman.don

Cela nous permet de pouvoir le modifier plus facilement que si nous l'avions programmé statiquement dans le code, tout comme les images du menu. Il a été préféré l'utilisation des lettres afin de pouvoir utiliser la fonction fgetc qui nous permet de savoir plus facilement quelle partie afficher. Le caractère est affiché pour les valeurs différentes de 'a' si jamais la différence entre la lettre et 'a' est inférieure ou égale au nombre d'erreurs jusque là.

Lexicographie

1. https://www.gnu.org/software/make/
2. https://en.wikipedia.org/wiki/ANSI_escape_code
3. https://fr.wikipedia.org/wiki/Portable_pixmap