PX111-info: Jouons (2)- Le jeu du pendu

Grenoble INP – Esisar – Academic year 2025-26 • *Compiled on 30 novembre 2025*

Version du 30 novembre 2025

Objectif pédagogique

- Pratiquer l'algorithmique et la programmation C
- Gagner de l'autonomie dans la conception et les tests logiciels
- Pratiquer le vocabulaire de lecture de programmes
- Pratiquer la fiche de syntaxe C "niveau 1".
- Pratiquer *un peu* les fonctions de bibliothèque sur les chaînes.

Ni le code ni les réponses ne sont évaluées, l'objectif est de **pratiquer**.

Description :

Le pendu (cf Source wikipédia) est un jeu consistant à trouver un mot en devinant quelles sont les lettres qui le composent. Le jeu se joue traditionnellement à deux, avec un papier et un crayon, selon un déroulement bien particulier.

Nous allons ici jouer contre l'ordinateur, celui-ci:

- tire un mot au hasard dans le dictionnaire
- propose 2 modes à chaque tour:
  - le mode mot (m): la joueuse propose un mot, et l'ordinateur vérifie l'égalité des mots
  - le mode lettre (c): la joueuse propose un caractère, et l'ordinateur renvoie les emplacements de cette lettre dans le mot, s'ils existent.
- décompte ou non une tentative, et recommence, sauf si la joueuse a perdu (après 10 tentatives infructeuses).

Il vous est fourni (dans une archive sous Chamilo) un code à trous, sous la forme d'une archive comprenant les fichiers suivants: `dict.txt  main.c  pendu.c  pendu.h`

Pour compiler les deux fichiers en même temps, vous utiliserez la commande suivante : `clang main.c pendu.c -Wall -o pendu`.

Questions préliminaires de compréhension

Compréhension du jeu.

Décrire au papier et crayon deux scénarios de partie.

Utilisation de fonctions de chaînes (fichier teststring.c)

Réaliser dans le fichier `teststring.c` à part des tests de fonction de la bibliothèque `string`. On se rapportera à la documentation fournie dans votre système: par exemple, la commande: `man strlen` affiche dans le terminal quelque chose comme ceci:

```
strlen(3)                  Library Functions Manual                  strlen(3)

NAME
       strlen - calculate the length of a string
[cut]
```

En regardant à chaque fois le manuel associé, écrire des appels aux fonctions suivantes:

- `strlen`, `isalpha` et `toupper`.
- `strcmp` pour comparer deux chaînes.
- `strcpy` ou `strncpy` pour copier des chaînes.

Entrées/sorties et "buffer de lecture" (fichier à réaliser)

Le programme suivant essaie de lire un entier, puis un caractère:

```c
#include <stdio.h>
#include <stdlib.h>
int main(){
  int choice; char c;
  scanf("%d",&choice);
  scanf("%c",&c);
  printf("choice = --%d--, c=--%c--\n",choice,c);
  return 0;
}
```

- Que se passe-t-il si on entre un nombre suivi de <ENTREE> ? Vérifier avec une utilisation adéquate de `printf`.
- Ajouter `while(getchar()!='\n');` à un ou plusieurs endroits précis du code pour résoudre le problème. Remarquer le corps vide de la boucle tant que.

Fermer les 2 fichiers précédents: à partir d'ici, travailler sur les fichiers du jeu pendu.

Sur les fichiers fournis (pendu.c, pendu.h, main.c)

On pourra se rapporter aux transparents `Coursxx_ES.pdf` disponibles sur Chamilo éventuellement projetés par votre chargé.e de TP.

- Vérifier que les fichiers compilent (avec des warning), et réaliser une première exécution: `clang main.c pendu.c -Wall -o pendu`, puis `./pendu`
- Dans `pendu.c` et `main.c`, remarquer l'inclusion de `#include "pendu.h"`. **Ce fichier sert à déclarer des noms de fonctions sans implémentation** Techniquement, ici, vu que l'on n'utilise pas la *compilation séparée* (concept qui sera vu en 2A), ce fichier nous sert de *spécification* des fonctions à implémenter.

- Dans quel fichier la matrice `dictionnaire` est-elle déclarée ?
- Quelle fonction permet d'initialiser le dictionnaire à partir du fichier ?
- Dans `chargeMots`, expliquer pourquoi `fgets` a pour deuxième argument `MAXLEN+1`.
- Dans `chargeMots`, expliquer comment est réalisée la lecture des informations du fichier `dict.txt`. Dans quelle variable est finalement stocké le mot lu dans une ligne du fichier ?
- Expliquer à quoi va servir le tableau `statut`. Comment savoir que la joueuse a gagné dans le cas "mode mot" ? dans le cas "mode lettre" ?
- Expliquer pourquoi nous imprimons le mot à deviner (dans le `main`).

Étape 2 : complétion du jeu

Pour l'implémentation, il est recommandé d'utiliser au maximum les fonctions de chaînes.

- Bien comprendre le code fourni, et commencer par `normaliseMot` et `chargeMots`, vérifier que vous obtenez bien une représentation mémoire du contenu du fichier.
- Réaliser la boucle de jeu avec les deux modes de jeu, dans un premier temps ne faire que la structure et tester (demande du numéro de mode à l'utilisatrice, puis un caractère ou un mot).
- Réaliser le mode "mot entier", puis le mode "lettre", en réalisant des tests au fur et à mesure.

Si vous avez du temps

- Concevoir et programmer une extension du jeu pour 2 joueuses.
- Concevoir et programmer une façon de sauvegarder l'état de la partie courante et de la recharger.