RAPPORT - SAÉ S1.02 : LE QUART DE SINGE

Clothilde Proux 111

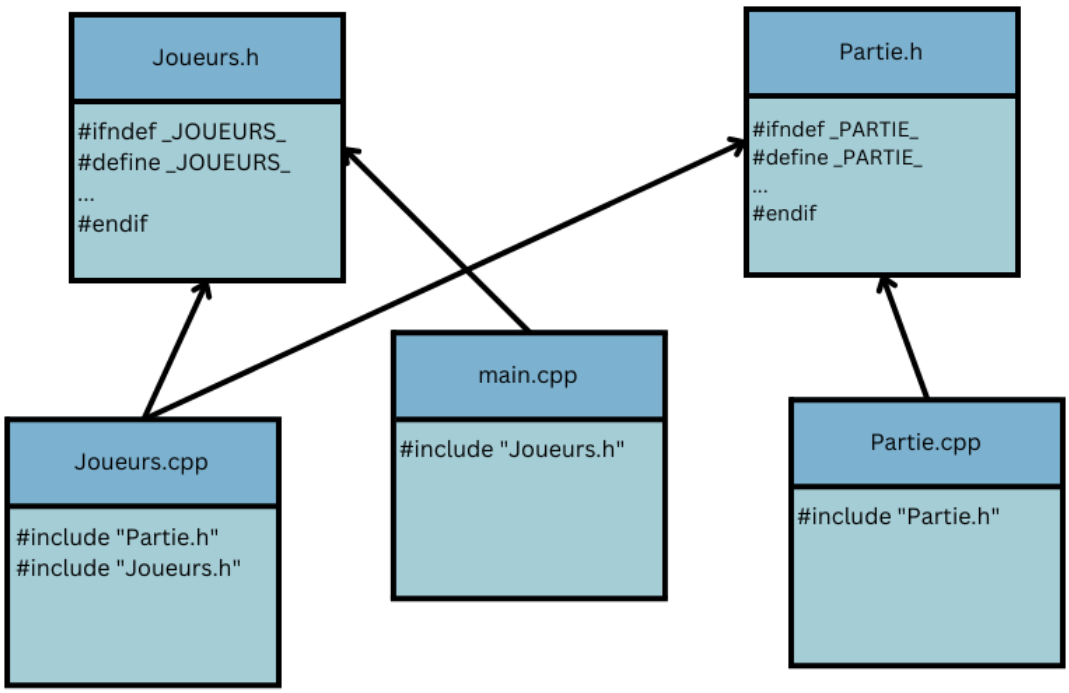
Table des matières

Introduction

Ce projet est un jeu codé en C++ reprenant les règles du « Quart de singe ». Le but du jeu est de rajouter des lettres sans finir un mot existant. Cependant, les coups jouables ne sont pas que des lettres. Un joueur peut se faire interroger sur le mot auquel il pensait en mettant un « ? ». Le joueur doit donc mettre des lettres cohérentes entre elles pour pouvoir être sûr d'avoir un mot en tête. Le joueur peut aussi abandonner la mache, avec « ! ». Dans chacun des cas perdants, le joueur récupère un « quart de singe », soit une pénalité. Au bout de quarte quart de singe, la partie est finie.

Ce projet a été effectué dans le cadre de la formation BUT en première année à l'IUT Paris – Rives de Seine.

Graphe de dépendance des fichiers sources



Tests unitaires

Test 1 :

Le premier test est une partie entre six joueurs humains. C'est le fichier d'entrée (in) qui a été donné. Il est conforme au fichier de sortie (out) fournis avec.

1H, () > A
2H, (A) > D
3H, (AD) > V
4H, (ADV) > E
5H, (ADVE) > R
6H, (ADVER) > S
1H, (ADVERS) > I
2H, (ADVERSI) > T
3H, (ADVERSIT) > E
le mot ADVERSITE existe, le joueur 3H prend un quart de singe
1H : 0; 2H : 0; 3H : 0.25; 4H : 0; 5H : 0; 6H : 0
3H, () > A
4H, (A) > G
5H, (AG) > H
6H, (AGH) > Q
1H, (AGHQ) > ?
6H, saisir le mot > AGI
le mot AGI ne commence pas par les lettres attendues, le joueur 6H prend un quart de singe
1H : 0; 2H : 0; 3H : 0.25; 4H : 0; 5H : 0; 6H : 0.25
6H, () > A
1H, (A) > T
2H, (AT) > C
3H, (ATC) > H
4H, (ATCH) > O
5H, (ATCHO) > U
6H, (ATCHOU) > ?
5H, saisir le mot > ATCHOUM
le mot ATCHOUM existe, le joueur 6H prend un quart de singe
1H : 0; 2H : 0; 3H : 0.25; 4H : 0; 5H : 0; 6H : 0.5
6H, () > A
1H, (A) > T
2H, (AT) > O
3H, (ATO) > C
4H, (ATOC) > !
le joueur 4H abandonne la manche et prend un quart de singe
1H : 0; 2H : 0; 3H : 0.25; 4H : 0.25; 5H : 0; 6H : 0.5
4H, () > A
5H, (A) > D
6H, (AD) > S
1H, (ADS) > O
2H, (ADSO) > R
3H, (ADSOR) > B
4H, (ADSORB) > I
5H, (ADSORBI) > O
6H, (ADSORBIO) > N
1H, (ADSORBION) > ?
6H, saisir le mot > ADSORBIONS
le mot ADSORBIONS existe, le joueur 1H prend un quart de singe
1H : 0.25; 2H : 0; 3H : 0.25; 4H : 0.25; 5H : 0; 6H : 0.5
1H, () > A
2H, (A) > L
3H, (AL) > V
4H, (ALV) > E
5H, (ALVE) > O
6H, (ALVEO) > L
1H, (ALVEOL) > I
2H, (ALVEOLI) > T
3H, (ALVEOLIT) > D
4H, (ALVEOLITD) > ?

3H, saisir le mot > ALVEOLITE
le mot ALVEOLITE ne commence pas par les lettres attendues, le joueur 3H prend un quart de singe
1H : 0.25; 2H : 0; 3H : 0.5; 4H : 0.25; 5H : 0; 6H : 0.5
3H, () > A
4H, (A) > T
5H, (AT) > Y
6H, (ATY) > P
1H, (ATYP) > I
2H, (ATYPI) > Q
3H, (ATYPIQ) > U
4H, (ATYPIQU) > !
le joueur 4H abandonne la manche et prend un quart de singe
1H : 0.25; 2H : 0; 3H : 0.5; 4H : 0.5; 5H : 0; 6H : 0.5
4H, () > A
5H, (A) > F
6H, (AF) > A
1H, (AFA) > P
2H, (AFAP) > ?
1H, saisir le mot > AFAT
le mot AFAT ne commence pas par les lettres attendues, le joueur 1H prend un quart de singe
1H : 0.5; 2H : 0; 3H : 0.5; 4H : 0.5; 5H : 0; 6H : 0.5
1H, () > A
2H, (A) > C
3H, (AC) > C
4H, (ACC) > L
5H, (ACCL) > I
6H, (ACCLI) > M
1H, (ACCLIM) > A
2H, (ACCLIMA) > T
3H, (ACCLIMAT) > O
4H, (ACCLIMATO) > N
5H, (ACCLIMATON) > !
le joueur 5H abandonne la manche et prend un quart de singe
1H : 0.5; 2H : 0; 3H : 0.5; 4H : 0.5; 5H : 0.25; 6H : 0.5
5H, () > A
6H, (A) > V
1H, (AV) > O
2H, (AVO) > C
3H, (AVOC) > A
4H, (AVOCA) > S
5H, (AVOCAS) > S
6H, (AVOCASS) > I
1H, (AVOCASSI) > E
2H, (AVOCASSIE) > X
3H, (AVOCASSIEX) > ?
2H, saisir le mot > AVOCASSIEZ
le mot AVOCASSIEZ ne commence pas par les lettres attendues, le joueur 2H prend un quart de singe
1H : 0.5; 2H : 0.25; 3H : 0.5; 4H : 0.5; 5H : 0.25; 6H : 0.5
2H, () > A
3H, (A) > T
4H, (AT) > A
5H, (ATA) > M
6H, (ATAM) > A
1H, (ATAMA) > ?
6H, saisir le mot > ATAMAN
le mot ATAMAN existe, le joueur 1H prend un quart de singe
1H : 0.75; 2H : 0.25; 3H : 0.5; 4H : 0.5; 5H : 0.25; 6H : 0.5
1H, () > A
2H, (A) > C
3H, (AC) > U
4H, (ACU) > T
5H, (ACUT) > A
6H, (ACUTA) > N
1H, (ACUTAN) > G
2H, (ACUTANG) > L
3H, (ACUTANGL) > !
le joueur 3H abandonne la manche et prend un quart de singe
1H : 0.75; 2H : 0.25; 3H : 0.75; 4H : 0.5; 5H : 0.25; 6H : 0.5
3H, () > A
4H, (A) > P
5H, (AP) > N

6H, (APN) > E
1H, (APNE) > I
2H, (APNEI) > Q
3H, (APNEIQ) > U
4H, (APNEIQU) > E
le mot APNEIQUE existe, le joueur 4H prend un quart de singe
1H : 0.75; 2H : 0.25; 3H : 0.75; 4H : 0.75; 5H : 0.25; 6H : 0.5
4H, () > A
5H, (A) > I
6H, (AI) > O
1H, (AIO) > L
2H, (AIOL) > ?
1H, saisir le mot > AIOLI
le mot AIOLI existe, le joueur 2H prend un quart de singe
1H : 0.75; 2H : 0.5; 3H : 0.75; 4H : 0.75; 5H : 0.25; 6H : 0.5
2H, () > A
3H, (A) > B
4H, (AB) > E
5H, (ABE) > L
6H, (ABEL) > I
1H, (ABELI) > E
2H, (ABELIE) > C
3H, (ABELIEC) > ?
2H, saisir le mot > ABELIEN
le mot ABELIEN ne commence pas par les lettres attendues, le joueur 2H prend un quart de singe
1H : 0.75; 2H : 0.75; 3H : 0.75; 4H : 0.75; 5H : 0.25; 6H : 0.5
2H, () > A
3H, (A) > A
4H, (AA) > K
5H, (AAK) > ?
4H, saisir le mot > AAS
le mot AAS ne commence pas par les lettres attendues, le joueur 4H prend un quart de singe
1H : 0.75; 2H : 0.75; 3H : 0.75; 4H : 1; 5H : 0.25; 6H : 0.5
La partie est finie

Test 2 :

La deuxième partie se dispute entre deux robots et deux humains. Ce test montre le niveau de difficulté des robots. Ils sont conçus pour ne jamais perdre (ils le peuvent évidemment).

1R, () > L
2H, (L) > i
3R, (LI) > C
4H, (LIC) > h
1R, (LICH) > I
2H, (LICHI) > ?
1R, saisir le mot > LICHIEZ
le mot LICHIEZ existe, le joueur 2H prend un quart de singe
1R : 0; 2H : 0.25; 3R : 0; 4H : 0
2H, () > m
3R, (M) > O
4H, (MO) > r
1R, (MOR) > A
2H, (MORA) > j
3R, (MORAJ) > ?
2H, saisir le mot > morajer
le mot MORAJER n'existe pas, le joueur 2H prend un quart de singe
1R : 0; 2H : 0.5; 3R : 0; 4H : 0
2H, () > h
3R, (H) > U
4H, (HU) > !
le joueur 4H abandonne la manche et prend un quart de singe
1R : 0; 2H : 0.5; 3R : 0; 4H : 0.25
4H, () > y
1R, (Y) > A
2H, (YA) > h
3R, (YAH) > ?
2H, saisir le mot > yahourt
le mot YAHOURT n'existe pas, le joueur 2H prend un quart de singe
1R : 0; 2H : 0.75; 3R : 0; 4H : 0.25

2H, () > t
3R, (T) > H
4H, (TH) > y
1R, (THY) > A
2H, (THYA) > r
3R, (THYAR) > ?
2H, saisir le mot > tbyar
le mot TBYAR ne commence pas par les lettres attendues, le joueur 2H prend un quart de singe
1R : 0; 2H : 1; 3R : 0; 4H : 0.25
La partie est finie

Test 3 :

La troisième est une partie exclusivement entre robots. Les mots se répètent quelque peu à cause du pseudo-aléatoire de la fonction rand() qui rapidement tourne sur les mêmes lettres de départ. Mis-à-part cela, tout fonctionne correctement.

1R, () > L
2R, (L) > O
3R, (LO) > C
1R, (LOC) > A
2R, (LOCA) > U
3R, (LOCAU) > ?
2R, saisir le mot > LOCAUX
le mot LOCAUX existe, le joueur 3R prend un quart de singe
1R : 0; 2R : 0; 3R : 0.25
3R, () > R
1R, (R) > H
2R, (RH) > U
3R, (RHU) > B
1R, (RHUB) > A
2R, (RHUBA) > R
3R, (RHUBAR) > B
1R, (RHUBARB) > ?
3R, saisir le mot > RHUBARBE
le mot RHUBARBE existe, le joueur 1R prend un quart de singe
1R : 0.25; 2R : 0; 3R : 0.25
1R, () > F
2R, (F) > E
3R, (FE) > A
1R, (FEA) > U
2R, (FEAU) > ?
1R, saisir le mot > FEAUX
le mot FEAUX existe, le joueur 2R prend un quart de singe
1R : 0.25; 2R : 0.25; 3R : 0.25
2R, () > K
3R, (K) > H
1R, (KH) > A
2R, (KHA) > G
3R, (KHAG) > N
1R, (KHAGN) > ?
3R, saisir le mot > KHAGNE
le mot KHAGNE existe, le joueur 1R prend un quart de singe
1R : 0.5; 2R : 0.25; 3R : 0.25
1R, () > Q
2R, (Q) > A
3R, (QA) > N
1R, (QAN) > U
2R, (QANU) > ?
1R, saisir le mot > QANUN
le mot QANUN existe, le joueur 2R prend un quart de singe
1R : 0.5; 2R : 0.5; 3R : 0.25
2R, () > Y
3R, (Y) > A
1R, (YA) > C
2R, (YAC) > H
3R, (YACH) > ?
2R, saisir le mot > YACHT
le mot YACHT existe, le joueur 3R prend un quart de singe

1R : 0.5; 2R : 0.5; 3R : 0.5
3R, () > U
1R, (U) > R
2R, (UR) > D
3R, (URD) > ?
2R, saisir le mot > URDU
le mot URDU existe, le joueur 3R prend un quart de singe
1R : 0.5; 2R : 0.5; 3R : 0.75
3R, () > Q
1R, (Q) > A
2R, (QA) > N
3R, (QAN) > U
1R, (QANU) > ?
3R, saisir le mot > QANUN
le mot QANUN existe, le joueur 1R prend un quart de singe
1R : 0.75; 2R : 0.5; 3R : 0.75
1R, () > F
2R, (F) > E
3R, (FE) > A
1R, (FEA) > U
2R, (FEAU) > ?
1R, saisir le mot > FEAUX
le mot FEAUX existe, le joueur 2R prend un quart de singe
1R : 0.75; 2R : 0.75; 3R : 0.75
2R, () > J
3R, (J) > A
1R, (JA) > C
2R, (JAC) > E
3R, (JACE) > N
1R, (JACEN) > ?
3R, saisir le mot > JACENT
le mot JACENT existe, le joueur 1R prend un quart de singe
1R : 1; 2R : 0.75; 3R : 0.75
La partie est finie

Test 4 :

Partie sans but précis entre deux humains et un robot. La sortie correspond aux attentes.

1H, () > JA
2H, (J) > *
veuillez entrer une lettre non-accentuée
2H, (J) > O
3R, (JO) > I
1H, (JOI) > R
2H, (JOIR) > ?
1H, saisir le mot > Jouirne
le mot JOUIRNE ne commence pas par les lettres attendues, le joueur 1H prend un quart de singe
1H : 0.25; 2H : 0; 3R : 0
1H, () > R
2H, (R) > I
3R, (RI) > D
1H, (RID) > I
2H, (RIDI) > C
3R, (RIDIC) > U
1H, (RIDICU) > L
2H, (RIDICUL) > !
le joueur 2H abandonne la manche et prend un quart de singe
1H : 0.25; 2H : 0.25; 3R : 0
2H, () > D
3R, (D) > I
1H, (DI) > C
2H, (DIC) > T
3R, (DICT) > O
1H, (DICTO) > I
2H, (DICTOI) > N
3R, (DICTOIN) > ?
2H, saisir le mot > !
le mot ! ne commence pas par les lettres attendues, le joueur 2H prend un quart de singe
1H : 0.25; 2H : 0.5; 3R : 0

2H, () > H
3R, (H) > U
1H, (HU) > I
le mot HUI existe, le joueur 1H prend un quart de singe
1H : 0.5; 2H : 0.5; 3R : 0
1H, () > vous avez dû
2H, (V) > remarquer que
3R, (VR) > A
1H, (VRA) > je ne suis pas très forte
2H, (VRAJ) > contre ces robots…
3R, (VRAJC) > ?
2H, saisir le mot > Ou alors, ce sont eux qui sont trop forts pour moi ?
le mot OU ne commence pas par les lettres attendues, le joueur 2H prend un quart de singe
1H : 0.5; 2H : 0.75; 3R : 0
2H, () > !
le joueur 2H abandonne la manche et prend un quart de singe
1H : 0.5; 2H : 1; 3R : 0
La partie est finie

Bilan du projet

Les difficultés rencontrées :

J'ai eu du mal à faire un exécutable mais j'ai réussi à trouver dans CLion où est-ce qu'il est créé. De plus, au tout départ, je ne comprenais pas le fichier donné par notre professeur, les notions de « argc » et « argv » essentiellement. J'ai dû demander à une personne tierce de me l'expliquer. Aussi, j'ai peiné à accéder au dictionnaire qui n'était pas trouvé par mon programme.

Ce qui a été réussi :

J'ai réussi à réaliser un jeu qui fonctionne autant pour des joueurs humains que pour des robots.

Ce qui peut être amélioré :

- Intégrer des conteneurs dynamiques à la place de la structure simple.
- Complexifier le jeu du robot pour avoir des parties plus différentes lors des parties sans humain. Souvent les mêmes mots sortent car seule la première lettre et en aléatoire.

Code source

Ordre des fonctions :

- Partie.h
- Partie.cpp

- Joueurs.h
- Joueurs.cpp
- main.cpp

Partie.h

```
#ifndef PROJETQUARTDESINGE_PARTIE_H
#define PROJETQUARTDESINGE_PARTIE_H


/**
 * @file Partie.h
 * SAÉ S1.02 : Le quart de singe
 * @authors PROUX Clothilde
 * @version 1 - 30/12/2022
 * @brief Jouer une partie de Quart de Singe.
 * Comparaison d'approches algorithmiques -- BUT Paris - Rives de Seine
*/


#include <iostream>
#include <fstream> // pour ifstream
#include <iomanip> // pour setw
#include <cstdlib> // pour rand

using namespace std;


/**
 * @brief Ouvre le dictionnaire.
 * @return dico -> le-dit dictionnaire
 */
ifstream ouvertureDico();


/**
 * @brief Teste si le mot entré appartient au dictionnaire.
 * @param[in] motCherche -> mot entré par le joueur
 * @return 0 -> le mot n'appartient pas ; 1 -> il appartient ; 2 -> le dictionnaire ne s'est
pas ouvert
 */
bool appartientDico(const char motCherche[]);


/**
 * @brief Donne le prochain coup d'un joueur robot.
 * @param[in] lettresJouees -> la liste des lettres données par les joueurs précédents
 * @return coupJoue -> le caractère joué par le robot
 */
char jeuRobot(const char *lettresJouees);


#endif //PROJETQUARTDESINGE_PARTIE_H
```

Partie.cpp

```
/**
 * @file main.cpp
 * SAÉ S1.02 : Le quart de singe
 * @authors PROUX Clothilde
 * @version finale - 04/01/2023
 * @brief Jouer une partie de Quart de Singe.
 * Comparaison d'approches algorithmiques -- BUT Paris - Rives de Seine
*/

#include <iostream>
#include <iomanip> // pour setw
```

```cpp
#include <cstdlib> // pour rand
#include "Partie.h" // Fonctions de la partie


using namespace std;
const int MAX_MOT = 26;



/**
 * @brief Ouvre le dictionnaire.
 * @return dico -> le-dit dictionnaire
 */
ifstream ouvertureDico() {
    ifstream dico("ods4.txt",ios::in | ios::binary);

    if (!dico) {
        cerr << "Le dictionnaire n'a pu etre ouvert" << endl;
        exit(2);
    }
    return dico;
}


/**
 * @brief Teste si le mot entré appartient au dictionnaire.
 * @param[in] motCherche -> mot entré par le joueur
 * @return 0 -> le mot n'appartient pas ; 1 -> il appartient ; 2 -> le dictionnaire ne s'est
pas ouvert
 */
bool appartientDico(const char motCherche[]) {
    ifstream dico = ouvertureDico();
    bool trouve = false;
    char motLu[MAX_MOT];

    dico >> setw(MAX_MOT) >> motLu;
    while (dico) {
        if (strcmp(motCherche, motLu) == 0) { // Le mot cherché appartient au dictionnaire
            trouve = true;
        }
        else if (strcmp(motCherche, motLu) < 0) { // Le mot lu est classé dans l'ordre
alphabétique après le mot cherché
            break; // Le mot n'appartient pas au dictionnaire. On sort de la boucle
        }
        dico >> setw(MAX_MOT) >> motLu;
    }
    dico.close();
    return trouve;
}


/**
 * @brief Donne le prochain coup d'un joueur robot.
 * @param[in] lettresJouees -> la liste des lettres données par les joueurs précédents
 * @return coupJoue -> le caractère joué par le robot
 */
char jeuRobot(const char *lettresJouees) {
    ifstream dico = ouvertureDico();
    char coupJoue;
    char motLu[MAX_MOT];
    char motTrouve[MAX_MOT] = "";
    char debutMotInterdit[MAX_MOT] = "";

    dico >> setw(MAX_MOT) >> motLu;
    if (strlen(lettresJouees) == 0) { // Le robot joue en premier
        coupJoue = (char) (rand() % 26 + 65); // Il sort une lettre aléatoire
    }
    else {
        while (dico) {
            if (strncmp(lettresJouees, motLu, strlen(lettresJouees)) == 0
                // Si les premières lettres de mot lu correspondent aux lettres jouées
                && (strncmp(motLu, debutMotInterdit, strlen(debutMotInterdit)) != 0 ||
strlen(debutMotInterdit) == 0)) {
                // Et que ça ne commence pas par les lettres interdites ou qu'il n'y en a pas
                if (strlen(motLu) == strlen(lettresJouees) + 1) {
                    // Pour ne pas donner une lettre qui terminerait un mot
                    strcpy(debutMotInterdit, motLu);
```

```
                }
                else if (strlen(motLu) > strlen(lettresJouees) + 1
                        && (strlen(motTrouve) == 0 || strlen(motTrouve) > strlen(motLu))) {
                    // Pour trouver le mot le plus court
                    strcpy(motTrouve,motLu);
                }
            }
            else if (strncmp(lettresJouees, motLu, strlen(lettresJouees)) < 0) {
                // Le mot lu est classé dans l'ordre alphabétique après les lettres jouées
                break;
            }
            dico >> setw(MAX_MOT) >> motLu;
        }

        if (strlen(motTrouve) == 0) // Aucun mot n'a été trouvé
            coupJoue = '?';
        else
            coupJoue = motTrouve[strlen(lettresJouees)];
        dico.close();
    }
    return coupJoue;
}
```

Joueurs.h

```
#ifndef PROJETQUARTDESINGE_JOUEURS_H
#define PROJETQUARTDESINGE_JOUEURS_H


/**
 * @file Joueurs.h
 * SAÉ S1.02 : Le quart de singe
 * @authors PROUX Clothilde
 * @version 1 - 30/12/2022
 * @brief Toutes les fonctions des joueurs.
 * Comparaison d'approches algorithmiques -- BUT Paris - Rives de Seine
*/


/**
 * @brief Structure de données de type Joueurs
 */
struct Joueurs {
    int nbJoueurs;
    char *joueurs;
    unsigned int joueurActuel;
    unsigned int joueurPrecedent;
    unsigned int joueurPerdant;
    float *scoresJoueurs;
};


/**
 * @brief Le joueur abandonne la manche.
 * @param joueursManche -> la structure contenant les joueurs, leur nombre et leur score
 */
void abandonManche(Joueurs &joueursManche);


/**
 * @brief Le joueur abandonne la manche.
 * @param[in,out] joueursManche -> la structure contenant les joueurs, leur nombre et leur
score
 */
void reponseJoueurPrecedent(Joueurs &joueursManche, const char *lettresJouees);


/**
 * @brief Vérifie si la lettre jouée par le joueur appartient au dictionnaire.
 * @param[in,out] joueursManche -> la structure contenant les joueurs, leur nombre et leur
score
 * @param[in] lettresJouees -> la liste des lettres déjà jouées
 * @param[in] coupJoue -> le coup joué par le joueur actuel
```

```
 */
bool lettreJoueeHumain(Joueurs &joueursManche, char *lettresJouees, char coupJoue);


/**
 * @brief Renseigne la nature du joueur actuel.
 * @param[in,out] joueursManche -> la structure contenant les joueurs, leur nombre et leur
score
 * @param[in] lettresJouees -> la liste des lettres données par les joueurs précédents
 * @return coupJoue -> le coup joué par le joueur actuel
 */
char jeuJoueurActuel(Joueurs &joueursManche, const char *lettresJouees);


/**
 * @brief Le joueur à qui on a demandé le mot auquel il pensait gagne ou non la manche.
 * @param[in,out] joueursManche -> la structure contenant les joueurs, leur nombre et leur
score
 * @param[in] lettresJouees -> la liste des lettres déjà jouées par les joueurs précédents
 */
void reponseHumainPrecedent(Joueurs &joueursManche, const char *lettresJouees);


/**
 * @brief Affiche le mot auquel "pensait" le robot
 * @param[in,out] joueursManche -> la structure contenant les joueurs, leur nombre et leur
score
 * @param[in] lettresJouees -> la liste des lettres déjà jouées
 */
void reponseRobotPrecedent(Joueurs &joueursManche, const char *lettresJouees);


/**
 * @brief Vérifie si la lettre entrée est correcte.
 * @param[in,out] joueursManche -> la structure contenant les joueurs, leur nombre et leur
score
 * @param[in] lettresJouees -> la liste des lettres déjà jouées
 * @param[in] coupJoue -> le coup joué par le joueur actuel
 * @return lettreBonne -> le booléen qui vérifie si la lettre est bonne
 */
bool lettreJoueeJoueurActuel(Joueurs &joueursManche, char *lettresJouees, char coupJoue);


/**
 * @brief Joue une manche.
 * @param[in,out] joueursManche -> la structure contenant les joueurs, leur nombre et leur
score
 */
void manche(Joueurs &joueursManche);


/**
 * @brief Joue une partie entière.
 * @param[in,out] joueursManche -> la structure contenant les joueurs, leur nombre et leur
score
 */
void partie(Joueurs &joueursManche);


#endif //PROJETQUARTDESINGE_JOUEURS_H
```

Joueurs.cpp

```
/**
 * @file Joueurs.cpp
 * SAÉ S1.02 : Le quart de singe
 * @authors PROUX Clothilde
 * @version finale - 05/01/2023
 * @brief Jouer une partie de Quart de Singe.
 * Comparaison d'approches algorithmiques -- BUT Paris - Rives de Seine
*/

#include <iostream>
```

```cpp
#include <iomanip> // pour setw
#include <cstdlib> // pour rand
#include "Joueurs.h" // Structure joueurs
#include "Partie.h" // Fonctions de la partie


using namespace std;
const int MAX_MOT = 26;


/**
 * @brief Le joueur abandonne la manche.
 * @param[in,out] joueursManche -> la structure contenant les joueurs, leur nombre et leur
score
 */
void abandonManche(Joueurs &joueursManche) {
    joueursManche.scoresJoueurs[joueursManche.joueurActuel - 1] += 0.25;
    joueursManche.joueurPerdant = joueursManche.joueurActuel;
    cout << "le joueur " <<  joueursManche.joueurActuel
         << (char) toupper(joueursManche.joueurs[joueursManche.joueurActuel - 1])
         << " abandonne la manche et prend un quart de singe" << endl;
}


/**
 * @brief Le joueur demande au précédent le mot auquel il pensait.
 * @param[in,out] joueursManche -> la structure contenant les joueurs, leur nombre et leur
score
 * @param[in] lettresJouees -> la liste des lettres déjà jouées par les joueurs précédents
 */
void reponseJoueurPrecedent(Joueurs &joueursManche, const char *lettresJouees) {
    joueursManche.joueurPrecedent = joueursManche.joueurActuel - 1;
    if (joueursManche.joueurPrecedent == 0) joueursManche.joueurPrecedent =
joueursManche.nbJoueurs;
    // Le premier joueur demande au dernier
    cout << joueursManche.joueurPrecedent
         << (char) toupper(joueursManche.joueurs[joueursManche.joueurPrecedent - 1]) << ",
saisir le mot > ";

    if (toupper(joueursManche.joueurs[joueursManche.joueurPrecedent - 1]) == 'H') // Jouer
Humain
        reponseHumainPrecedent(joueursManche, lettresJouees);
    else // Joueur Robot
        reponseRobotPrecedent(joueursManche, lettresJouees);
    joueursManche.scoresJoueurs[joueursManche.joueurPerdant - 1] += 0.25;
}


/**
 * @brief Vérifie si la lettre jouée par le joueur appartient au dictionnaire.
 * @param[in,out] joueursManche -> la structure contenant les joueurs, leur nombre et leur
score
 * @param[in] lettresJouees -> la liste des lettres déjà jouées
 * @param[in] coupJoue -> le coup joué par le joueur actuel
 */
bool lettreJoueeHumain(Joueurs &joueursManche, char *lettresJouees, const char coupJoue) {
    bool lettreBonne = true;

    if (toupper(coupJoue) >= 65 && toupper(coupJoue) <= 90) { // La lettre est non-accentuée
        if (strlen(lettresJouees) < MAX_MOT - 1)
            lettresJouees[strlen(lettresJouees)] = (char) toupper(coupJoue);
        if (strlen(lettresJouees) > 2) { // Mot testé à partir de trois lettres
            if (appartientDico(lettresJouees)) {
                cout << "le mot " << lettresJouees << " existe, le joueur " <<
joueursManche.joueurActuel
                     << (char) toupper(joueursManche.joueurs[joueursManche.joueurActuel-1])
                     << " prend un quart de singe" << endl;
                joueursManche.scoresJoueurs[joueursManche.joueurActuel - 1] += 0.25;
                joueursManche.joueurPerdant = joueursManche.joueurActuel;
                lettreBonne = false;
            }
        }
    }
    else {
        cout << "veuillez entrer une lettre non-accentuée" << endl;
        joueursManche.joueurActuel--;
    }
```

```
    return lettreBonne;
}


/**
 * @brief Renseigne la nature du joueur actuel.
 * @param[in,out] joueursManche -> la structure contenant les joueurs, leur nombre et leur
score
 * @param[in] lettresJouees -> la liste des lettres données par les joueurs précédents
 * @return coupJoue -> le coup joué par le joueur actuel
 */
char jeuJoueurActuel(Joueurs &joueursManche, const char *lettresJouees) {
    char coupJoue;

    if (joueursManche.joueurActuel > joueursManche.nbJoueurs) joueursManche.joueurActuel = 1;
    // le joueur après le dernier est le premier
    cout << joueursManche.joueurActuel << (char)
toupper(joueursManche.joueurs[joueursManche.joueurActuel - 1])
         << ", ("<< lettresJouees << ") > ";
    if (toupper(joueursManche.joueurs[joueursManche.joueurActuel - 1]) == 'H') { // Joueur
humain
        cin >> coupJoue;
        cin.ignore(INT_MAX, '\n');
    }
    else if (toupper(joueursManche.joueurs[joueursManche.joueurActuel - 1]) == 'R') { //
Joueur robot
        coupJoue = jeuRobot(lettresJouees);
        cout << coupJoue << endl;
    }
    else { // Joueur alien
        cerr << "joueur invalide. veuillez réitérer une partie" << endl;
        exit(1);
    }
    return coupJoue;
}


/**
 * @brief Le joueur à qui on a demandé le mot auquel il pensait gagne ou non la manche.
 * @param[in,out] joueursManche -> la structure contenant les joueurs, leur nombre et leur
score
 * @param[in] lettresJouees -> la liste des lettres déjà jouées par les joueurs précédents
 */
void reponseHumainPrecedent(Joueurs &joueursManche, const char *lettresJouees) {
    char motSaisi[MAX_MOT];
    bool gagne;

    cin >> setw(MAX_MOT) >> motSaisi;
    cin.ignore(INT_MAX, '\n');
    for (int i = 0; i < strlen(motSaisi); i++) { // Mettre en majuscule le mot entré
        motSaisi[i] = (char) toupper(motSaisi[i]);
    }
    if (strncmp(motSaisi, lettresJouees, strlen(lettresJouees)) != 0) {
        // Si ça ne commence pas par les lettres déjà jouées
        cout << "le mot " << motSaisi << " ne commence pas par les lettres attendues, le
joueur "
             << joueursManche.joueurPrecedent << (char)
toupper(joueursManche.joueurs[joueursManche.joueurPrecedent - 1])
             << " prend un quart de singe" << endl;
        gagne = false;
    }
    else {
        // Teste si le mot est dans le dictionnaire
        if (appartientDico(motSaisi)) { // Le mot appartient au dictionnaire
            cout << "le mot " << motSaisi << " existe, le joueur " <<
joueursManche.joueurActuel
                 << (char) toupper(joueursManche.joueurs[joueursManche.joueurActuel-1]) << "
prend un quart de singe" << endl;
            gagne = true;
        }
        else { // Le mot n'appartient pas
            cout << "le mot " << motSaisi << " n'existe pas, le joueur " <<
joueursManche.joueurPrecedent
                 << (char) toupper(joueursManche.joueurs[joueursManche.joueurPrecedent - 1])
                 << " prend un quart de singe" << endl;
            gagne = false;
        }
```

```
    }
    if (gagne)
        joueursManche.joueurPerdant = joueursManche.joueurActuel;
    else
        joueursManche.joueurPerdant = joueursManche.joueurPrecedent;
}


/**
 * @brief Affiche le mot auquel "pensait" le robot
 * @param[in,out] joueursManche -> la structure contenant les joueurs, leur nombre et leur
score
 * @param[in] lettresJouees -> la liste des lettres déjà jouées
 */
void reponseRobotPrecedent(Joueurs &joueursManche, const char *lettresJouees) {
    ifstream dico = ouvertureDico();

    char motLu[MAX_MOT];
    char motTrouve[MAX_MOT] = "";
    dico >> setw(MAX_MOT) >> motLu;
    while (dico) {
        if (strncmp(lettresJouees, motLu, strlen(lettresJouees)) == 0) {
            // Recherche un mot qui commence par les lettres jouées
            strcpy(motTrouve,motLu);
            cout << motTrouve << endl;
            cout << "le mot " << motTrouve << " existe, le joueur " <<
joueursManche.joueurActuel
                 << (char) toupper(joueursManche.joueurs[joueursManche.joueurActuel-1])
                 << " prend un quart de singe" << endl;
            break;
        }
        dico >> setw(MAX_MOT) >> motLu;
    }
    dico.close();
    joueursManche.joueurPerdant = joueursManche.joueurActuel;
}


/**
 * @brief Vérifie si la lettre entrée est correcte.
 * @param[in,out] joueursManche -> la structure contenant les joueurs, leur nombre et leur
score
 * @param[in] lettresJouees -> la liste des lettres déjà jouées
 * @param[in] coupJoue -> le coup joué par le joueur actuel
 * @return lettreBonne -> le booléen qui vérifie si la lettre est bonne
 */
bool lettreJoueeJoueurActuel(Joueurs &joueursManche, char *lettresJouees, const char coupJoue)
{
    bool lettreBonne = true;

    if (toupper(joueursManche.joueurs[joueursManche.joueurActuel - 1]) == 'H')  // Par un
humain
        lettreBonne = lettreJoueeHumain(joueursManche, lettresJouees, coupJoue);
    else { // Par un robot
        lettresJouees[strlen(lettresJouees)] = coupJoue;
    }
    return lettreBonne;
}


/**
 * @brief Joue une manche.
 * @param[in,out] joueursManche -> la structure contenant les joueurs, leur nombre et leur
score
 */
void manche(Joueurs &joueursManche) {
    char coupJoue;
    char lettresJouees[MAX_MOT] = "";
    bool mancheFinie = false;

    while (!mancheFinie) {
        coupJoue = jeuJoueurActuel(joueursManche, lettresJouees);

        // Coups possibles
        if (coupJoue == '!') { // Abandon de la manche
            abandonManche(joueursManche);
            mancheFinie = true;
```

```
        }
        else if (coupJoue == '?') { // Demande le mot du joueur précédent
            reponseJoueurPrecedent(joueursManche, lettresJouees);
            mancheFinie = true;
        }
        else { // Lettre jouée → Ajout d'une lettre à celles déjà jouées
            if (!lettreJoueeJoueurActuel(joueursManche, lettresJouees, coupJoue))
                mancheFinie = true;
        }
        joueursManche.joueurActuel++;
    }
}


/**
 * @brief Joue une partie entière.
 * @param[in,out] joueursManche -> la structure contenant les joueurs, leur nombre et leur
score
 */
void partie(Joueurs &joueursManche) {
    joueursManche.scoresJoueurs = new float[joueursManche.nbJoueurs];
    joueursManche.joueurActuel = 1;
    for (int j = 0; j < joueursManche.nbJoueurs; ++j) joueursManche.scoresJoueurs[j] = 0.;
    // Initialise les scores à zéro

    do {
        manche(joueursManche);
        joueursManche.joueurActuel = joueursManche.joueurPerdant;
        for (int i = 1; i <= joueursManche.nbJoueurs; ++i) {
            cout << i << (char) toupper(joueursManche.joueurs[i - 1]) << " : " <<
joueursManche.scoresJoueurs[i - 1];
            if (i != joueursManche.nbJoueurs)
                cout << "; ";
            else
                cout << endl;
        }
    } while (joueursManche.scoresJoueurs[joueursManche.joueurPerdant - 1] != 1.);
    cout << "La partie est finie" << endl;
}
```

main.cpp

```
/**
 * @file main.cpp
 * SAÉ S1.02 : Le quart de singe
 * @authors PROUX Clothilde
 * @version finale - 04/01/2023
 * @brief Jouer une partie de Quart de Singe.
 * Comparaison d'approches algorithmiques -- BUT Paris - Rives de Seine
*/

#include <iostream>
#include "Joueurs.h" // Structure joueurs


using namespace std;


/**
 * @brief Programme principal.
 * @param[in] argc -> nombre de paramètres sur la ligne de commande (+ 1)
 * @param[in] argv -> paramètres de la ligne de commande (argv[0] : la cde, argv[1] : le 1er
paramètre, etc)
 * @return 0 -> le programme s'est exécuté sans problème
 */
int main(int argc, const char* argv[]) {
    Joueurs joueursManche{};

    // Paramètre sur la ligne de commande
    if (argc >= 2) {
        if (strlen(argv[1]) > 1) {
            joueursManche.nbJoueurs = (int) strlen(argv[1]);
            joueursManche.joueurs = new char[joueursManche.nbJoueurs];
```

```
            strcpy(joueursManche.joueurs, argv[1]);
            partie(joueursManche);
        }
        else
            cout << "Il faut au minimum deux joueurs" << endl;
    }
    else
        cout << "Il n'y a pas de parametre" << endl;
    return 0;
}
```