|  |
| --- |
| Guillaume de La Grandiere, Luca Randazzo |
| Structures des Données et Algorithmes fondamentaux |
| Dossier de développement : créer un jeu de Boggle en C++ |

Table des matières

[Présentation 2](#_Toc61040545)

[Introduction 2](#_Toc61040546)

[Objectif du projet 2](#_Toc61040547)

[Organisation du développement 3](#_Toc61040548)

[Sprints 3](#_Toc61040549)

[Tests des sprints 3](#_Toc61040550)

[Graphe de dépendance 4](#_Toc61040551)

[Bilan du projet 5](#_Toc61040552)

[Difficultés rencontrées 5](#_Toc61040553)

[Conclusion 5](#_Toc61040554)

[Annexe 6](#_Toc61040555)

[Fichiers d’en-tête 6](#_Toc61040556)

[Mot.h 6](#_Toc61040557)

[Plateau.h 7](#_Toc61040558)

[ConteneurMot.h 9](#_Toc61040559)

[ListeConteneurMot.h 12](#_Toc61040560)

[Fichiers sources 14](#_Toc61040561)

[Mot.cpp 14](#_Toc61040562)

[Plateau.cpp 15](#_Toc61040563)

[ConteneurMot.cpp 18](#_Toc61040564)

[ListeConteneurMot.cpp 23](#_Toc61040565)

[Main.cpp 26](#_Toc61040566)

# Présentation

## Introduction

Le jeu de société est un divertissement très populaire de nos jours. Cependant avec la récente crise sanitaire il n’est plus possible de se réunir entre amis pour jouer. Il est donc nécessaire de pouvoir créer des versions informatique de ses jeux pour permette aux gens de continuer à jouer. De plus un programme pourra effectuer des taches fastidieuses plus rapidement qu’un humain ce qui rendait les parties plus fluides et limiterais es fautes d’origine humaine.

## Objectif du projet

Le but de ce projet est de créer un ensemble de programmes qui puissent héberger une partie de « Boggle ». Ils doivent donc être capables de lire des mots et de les stocker, et de comparer des listes de mots entre elles. De plus ils doivent pouvoir lire un plateau pour vérifier quels mots sont représentables dessus. Enfin ils doivent pourvoir vérifier l’existence d’un mot et attribuer des points au joueurs en fonction du nombre de mots trouvés.

# Organisation du développement

## Sprints

Le projet est organisé en sprints, permettant une évolution des programme par palliés. Chaque sprint correspond à un unique programme. Ils sont catégorisés comme suit :

* Sprint 1 : le programme doit prendre une liste de mots écrits au clavier, compter et afficher le nombre de points que vaut la liste.
* Sprint 2 : le programme doit prendre une liste de mots au clavier, le trier et enlever les mots en doubles pour enfin l’afficher à l’écran.
* Sprint 3 : le programme doit prendre deux listes de mots au clavier et les comparer pour afficher les mots de la seconde liste absents de la première liste.
* Sprint 4 : le programme doit prendre deux listes de mots au clavier et les comparer pour afficher les mots de la seconde liste présents dans la première liste.
* Sprint 5 : le programme doit prendre une liste de listes de mots et les comparer entre elles afin d’afficher tous les mots présents dans au moins deux listes.
* Sprint 6 : le programme doit lire un plateau de jeu de « Boggle » et afficher tous les mots représentables à partir de celui-ci.

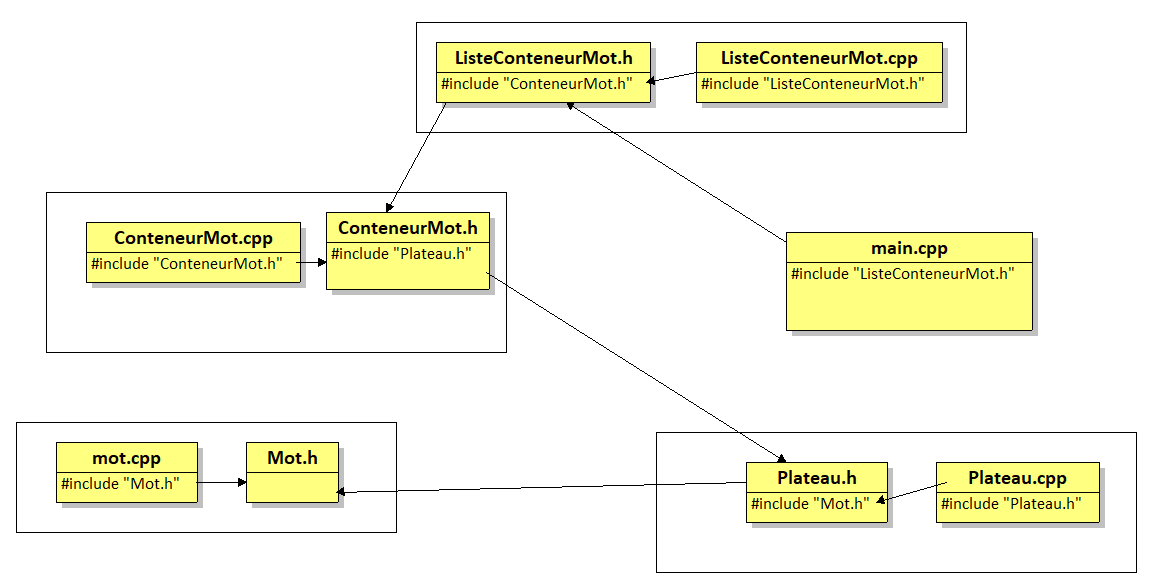
## Tests des sprints

Pour valider les tests, un fichier contenant des entrées doit être lu et exécuté par le programme et donner en sortie un fichier identique à celui fourni. Le fichier « out » peut être obtenu en passant le fichier « in » en entrée lors d’une exécution du programme dans l’invite de commande. Le fichier « out » ainsi obtenue peut être alors comparé au fichier « out » fourni pour vérifier la validité du code. Nous avons utilisé le site [diffchecker.com](https://www.diffchecker.com/) pour effectuer la vérification.

Nous avons effectué des vérifications tout au long du développement pour valider chaque sprint. Pour cela nous avons utilisé les fichiers « in » et « out » fournis. Ces tests se sont tous avérés positifs. Nous avons aussi menés nos propre test dans un objectif de débogage. Enfin nous avons effectué les tests complémentaires fournis qui se sont avérés fonctionnels.

## Graphe de dépendance

Voici le graphe de dépendance de notre projet. Vous pourrez retrouver le code source des projets en annexe.



# Bilan du projet

## Difficultés rencontrées

Les plus grosses difficultés ont été de débugger certaines parties du programme qui étaient dues à des tentatives d’optimisations d’espaces mal gérées (unsigned int négatifs entre autres).

Les autres problèmes mineurs ressentis étaient surtout dû à l’adaptation des systèmes vus en cours sur notre projet ainsi que la traduction du pseudo code su sprint 6 en C++.

Les autres difficultés ont pu être évitées grâce a un code structuré et lisible, qui était donc adapté au travail en binôme et qui a permis des relectures faciles.

## Conclusion

Ce projet se trouvait dans la continuité du projet de C, ce qui nous a permis de mieux nous organiser en apprenant de nos erreurs du projet précédent.

Nous avons été plus efficaces et malgré le manque de présentiel nous avons bien communiqué et la mise en place d’un répertoire Github pour partager le code a été efficace.

Les énoncés étant assez vagues, cela permet de s’éloigner de conditions de travail plus scolaire et force a fouiller dans les fichiers « in » et « out » ce qui rends le travail plus proche de ce que pourrait être un travail plus professionnel.

Nous nous sommes bien répartis les tâches et le travail a été effectué correctement par les deux membres du groupe ce qui montre une évolution par rapport au premier projet ou la répartition avait pue être très inégale entre les deux binômes.

# Annexe

## Fichiers d’en-tête

### Mot.h

#ifndef Mot\_h

#define Mot\_h

typedef const char\* Mot;*// Définition du type Mot en tant que chaine de caractères*

*/\**

*[brief] : Fonction vérifiant si le premier Mot passé en paramètre est situé plus loin*

*dans l'ordre alphabétique que le second Mot passé en paramètre*

*Mot [in] : premier mot à passer en paramètre*

*Mot [in] : second mot à passer en paramètre*

*return : booléen, retourne vrai si c'est le cas et faux dans le cas contraire*

*\*/*

bool estSuperieurOrdreAlphabetique(const Mot&, const Mot&);

#endif

### Plateau.h

#ifndef Plateau\_h

#define Plateau\_h

#include "Mot.h"

struct Case {*// Type correspondant à une case du plateau*

    char lettre;*// Lettre de la case*

    bool visite;*// booléen pour vérifier si une case a été visitée dans le cadre de la fonction de recherche*

};

struct Plateau {*// Type correspondant au plateau entier*

    Case coords[4][4];*// Tableau de Cases à deux dimensions (4\*4)*

};

struct Coords {*// Type correspondant aux coordonnées dans le tableau*

    int x;*// Abscisses*

    int y;*// Ordonnées*

};

*/\**

*[brief] : lire au clavier un plateau pour le stocker dans une variable de type plateau*

*Plateau [in-out] : Plateau passé par référence*

*\*/*

void lireClavierPlateau(Plateau&);

*/\**

*[brief] : pour afficher le plateau à l'écran*

*Plateau [in] : Plateau pass� par r�f�rence (avec un const pour emp�cher la modification)*

*\*/*

void afficherPlateau(const Plateau&);

*/\**

*[brief] : vérifier si un mot est présent dans le tableau à l'aide de la fonction récursive sousRecherche*

*Mot [in] : Mot passé par référence*

*Plateau [in-out] : Plateau passé par référence (avec modification possible pour changer le statut des cases,*

*et passer de visitée à non visitée)*

*return : retourne vrai si le mot est pr�sent, faux dans le cas contraire*

*\*/*

bool recherche(const Mot&, Plateau&);

*/\**

*[brief] : fonction permettant de vérifier la présence d'un mot dans un plateau en exploitant le principe de*

*récursivité*

*Mot [in] : mot que l'on souhaite vérifier*

*unsigned int [in] : position dans le mot*

*Coords [in] : Coordoonées passées par référence*

*Plateau [in-out] : Plateau passé par référence (avec modification possible pour changer le statut des cases,*

*et passer de visitée à non visitée)*

*return : retourne vrai si le mot est pr�sent, faux dans le cas contraire*

*\*/*

bool sousRecherche(const Mot&, unsigned int, const Coords&, Plateau&);

#endif

### ConteneurMot.h

#ifndef ConteneurMot\_h

#define ConteneurMot\_h

#include "Plateau.h"

#include <iostream>

struct ConteneurMot {*// Type correspondant � une liste de Mots*

    Mot\* tab;*// Tableau dynamique de Mots*

    unsigned int nbMots = 0;*// Nombre de Mots contenus dans le tableau*

};

*/\**

*[brief] : pour initialiser la liste de mots et pour permettre le stockage de mots à l'intérieur*

*ConteneurMot [in-out] : liste de mots passée par référence*

*\*/*

void intialiserConteneurMot(ConteneurMot&);

*/\**

*[brief] : pour ajouter un Mot à la liste de mots passée en paramètre*

*ConteneurMot [in-out] :  liste de mots passée par référence*

*Mot [in] : Mot à ajouter à la liste de mots*

*\*/*

void ajouterMot(ConteneurMot&, const Mot&);

*/\**

*[brief] : Pour afficher le contenu d'une liste de mots*

*ConteneurMot [in] : liste de mots passée par référence (avec un const pour empécher la modification)*

*\*/*

void afficherConteneurMot(const ConteneurMot&);

*/\**

*[brief] : pour lire au clavier une liste de mots et l'ajouter à une liste de mots*

*ConteneurMot [in-out] : liste de mots passée par référence*

*\*/*

void lireClavierConteneurMot(ConteneurMot&);

*/\**

*[brief] : pour trier une liste de mots par ordre alphabétique*

*ConteneurMot [in-out] : liste de mots passée par référence*

*\*/*

void trierConteneurMot(ConteneurMot&);

*/\**

*[brief] : pour vérifier si une liste de mots est trié*

*ConteneurMot [in] : liste de mots passée par référence (avec un const pour empécher la modification)*

*return : booléen, vrai si le conteneur est trié et faux dans le cas contraire*

*\*/*

bool estTrie(const ConteneurMot&);

*/\**

*[brief] : pour vérifier si un certain mot est présent dans une liste de mots*

*Mot [in] : le mot que l'on veut vérifier*

*ConteneurMot [in] : liste de mots passée par référence (avec un const pour empécher la modification)*

*return : booléen, retourne vrai si le mot est présent et faux dans le cas contraire*

*\*/*

bool motPresentdansConteneurMot(const Mot&, const ConteneurMot&);

*/\**

*[brief] : compter le nombre de points que représente une liste de mots*

*ConteneurMot [in] : liste de mots passée par référence (avec un const pour empécher la modification)*

*\*/*

void compterPointsConteneurMot(const ConteneurMot&);

*/\**

*[brief] : retourne les mots présents dans la première liste de mots passée en paramètre mais*

*absent de la seconde liste passé en paramètre*

*ConteneurMot [in] : première liste de mots passée par référence (avec un const pour empécher la modification)*

*ConteneurMot [in] : seconde liste de mots passée par référence pour empécher la modification)*

*return : retourne un ConteneurMot contenant les mots trouvés*

*\*/*

ConteneurMot motsAbsentsDansSecondConteneurMot(const ConteneurMot&, const ConteneurMot&);

*/\**

*[brief] : retourne les mots présents dans les deux listes de mots passées en paramètre via une*

*recherche dichotomique*

*ConteneurMot [in-out] : première liste de mots passée par référence (avec triage ultérieur possible si la liste n'est pas triée)*

*ConteneurMot [in] : seconde liste de mots passée par référence (avec un const pour empécher la modification)*

*return : retourne un ConteneurMot contenant les mots trouvés*

*\*/*

ConteneurMot rechercheDichotomique(ConteneurMot&, const ConteneurMot&);

*/\**

*[brief] : retourne les mots présents dans la liste de mots passée en paramètre*

*et sur le plateau passé en paramètre*

*ConteneurMot [in] : première liste de mots passée par référence (avec un const pour empécher la modification)*

*Plateau [in-out] : le plateau (sans const pour permettre le basculement du statut des cases, entre visitées ou non)*

*return : retourne un ConteneurMot contenant les mots trouvés*

*\*/*

ConteneurMot motsPresentsDansPlateau(const ConteneurMot&, Plateau&);

#endif

### ListeConteneurMot.h

#ifndef ListeConteneurMot\_h

#define ListeConteneurMot\_h

#include "ConteneurMot.h"

#include <iostream>

struct ListeConteneurMot {*// type correspondant à une liste de liste de mots*

    ConteneurMot\* tab;*// Tableau dynamique de ConteneurMots*

    unsigned int nbConteneurs = 0;*// Nombre de conteneurs dans le tableau*

};

*/\**

*[brief] : pour initialiser le conteneur de mots et pour permettre le stockage de mots à l'intérieur*

*ListeConteneurMot [in-out] : liste de liste de mots passée par référence*

*\*/*

void initialiserListeConteneurMot(ListeConteneurMot&);

*/\**

*[brief] : pour ajouter une liste de mots à la liste de liste de mots passée en paramètre*

*ListeConteneurMot [in-out] :  liste de liste de mots passée par référence*

*ConteneurMot [in] : liste de mots à ajouter à la liste de liste de mots*

*\*/*

void ajouterConteneurMot(ListeConteneurMot&, ConteneurMot&);

*/\**

*[brief] : Pour afficher le contenu d'une liste de liste de mots*

*ListeConteneurMot [in] : liste de liste de mots passée par référence (avec un const pour empécher la modification)*

*\*/*

void afficherListeConteneurMot(const ListeConteneurMot&);

*/\**

*[brief] : pour lire au clavier une liste de liste de mots et l'ajouter à une liste de liste de mots*

*ListeConteneurMot [in-out] : liste de liste de mots passée par référence*

*\*/*

void lireClavierListeConteneurMot(ListeConteneurMot&);

*/\**

*[brief] : pour trier une liste de liste de mots par ordre alphabétique*

*ListeConteneurMot [in-out] : liste de liste de mots passée par référence*

*\*/*

void trierListeConteneurMots(ListeConteneurMot&);

*/\**

*[brief] : retourne les mots présents dans au moins deux listes de mots située dans une liste de liste de mots*

*ListeConteneurMot [in-out] : liste de liste de mots passée par référence (avec modification possible pour permettre*

*le tri pour la fonction de recherche dichotomique*

*return : retourne un ConteneurMot contenant les mots trouvés*

*\*/*

ConteneurMot MotsPresentsDansMinDeuxConteneurs(ListeConteneurMot&);

#endif

## Fichiers sources

### Mot.cpp

#include "Mot.h"

#include <iostream>

bool estSuperieurOrdreAlphabetique(const Mot& m1, const Mot& m2) {

    unsigned int taille;

    if (strlen(m1) > strlen(m2))

        taille = int(strlen(m2));

    else

        taille = int(strlen(m1));

    for (unsigned int i = 0; i < taille; i++) {

        if (m1[i] > m2[i])

            return true;

        else if (m1[i] < m2[i])

            return false;

    }

    if (strlen(m1) > strlen(m2))

        return true;

    else

        return false;

}

### Plateau.cpp

#include <iostream>

#include "Plateau.h"

void lireClavierPlateau(Plateau& p) {

    for (unsigned int i = 0; i < 4; i++) {

        char\* buffer = new char[4];

        std::cin >> buffer;

        for (unsigned int j = 0; j < 4; j++) {

            p.coords[i][j].lettre = buffer[j];

        }

    }

}

void afficherPlateau(const Plateau& p) {

    for (unsigned int i = 0; i < 4; i++) {

        for (unsigned int j = 0; j < 4; j++) {

            std::cout << " ";

            std::cout << p.coords[i][j].lettre;

        }

        std::cout << std::endl;

    }

}

bool recherche(const Mot& m, Plateau& p){

    for (unsigned int i = 0; i < 4; i++) {

        for (unsigned int j = 0; j < 4; j++) {

            p.coords[i][j].visite = false;

        }

    }

    for (unsigned int i = 0; i < 4; i++) {

        for (unsigned int j = 0; j < 4; j++) {

            Coords c;

            c.x = i;

            c.y = j;

            if (sousRecherche(m, 0, c, p))

                return true;

        }

    }

    return false;

}

bool sousRecherche(const Mot& m, unsigned int pos, const Coords& c, Plateau& p) {

    if (pos >= strlen(m))

        return true;

    if (c.x > 3 || c.y > 3 || c.x < 0 || c.y < 0)

        return false;

    if (p.coords[c.x][c.y].lettre != m[pos])

        return false;

    if (p.coords[c.x][c.y].visite)

        return false;

    p.coords[c.x][c.y].visite = true;

    for (unsigned int i = 0 ; i < 8 ; i++) {

        Coords c2 = c;

        switch (i) {

        case 0: c2.x++; break;

        case 1: c2.y++; break;

        case 2: c2.x--; break;

        case 3: c2.y--; break;

        case 4: c2.x++; c2.y++; break;

        case 5: c2.x++; c2.y--; break;

        case 6: c2.x--; c2.y++; break;

        case 7: c2.x--; c2.y--; break;

        }

        if (sousRecherche(m, pos + 1, c2, p))

            return true;

    }

    p.coords[c.x][c.y].visite = false;

    return false;

}

### ConteneurMot.cpp

#include "ConteneurMot.h"

void intialiserConteneurMot(ConteneurMot& c){

    c.tab = new Mot[1];

}

void ajouterMot(ConteneurMot& c, const Mot& m){

    if(c.nbMots > 0){

        Mot\* new\_tab = new Mot[c.nbMots+1];

        for(unsigned int i = 0 ; i < c.nbMots ; i++){

            if (strcmp(c.tab[i], m) == 0)

                return;

            new\_tab[i] = c.tab[i];

        }

        delete[]c.tab;

        c.tab = new\_tab;

    }

    c.tab[c.nbMots] = m;

    c.nbMots++;

}

void afficherConteneurMot(const ConteneurMot& c){

    for(unsigned int i = 0 ; i < c.nbMots ; i++){

        std::cout << c.tab[i] << std::endl;

    }

    std::cout << '\*' << std::endl;

}

void lireClavierConteneurMot(ConteneurMot& c) {

    while (true) {

        char\* buffer = new char[30];

        std::cin >> buffer;

        if (\*buffer == '\*')

            break;

        Mot m = buffer;

        ajouterMot(c, m);

    }

}

void trierConteneurMot(ConteneurMot& c) {

    while (!estTrie(c)) {

        for (unsigned int i = 1; i < c.nbMots; i++) {

            if (estSuperieurOrdreAlphabetique(c.tab[i-1], c.tab[i])) {

                Mot temp = c.tab[i];

                c.tab[i] = c.tab[i-1];

                c.tab[i-1] = temp;

            }

        }

    }

}

bool estTrie(const ConteneurMot& c) {

    for (unsigned int i = 1; i < c.nbMots; i++) {

        if (estSuperieurOrdreAlphabetique(c.tab[i-1], c.tab[i]))

            return false;

    }

    return true;

}

bool motPresentdansConteneurMot(const Mot& mot, const ConteneurMot& liste) {

    for (unsigned int j = 0; j < liste.nbMots; j++) {

        if (strcmp(mot, liste.tab[j]) == 0) {

            return true;

        }

    }

    return false;

}

ConteneurMot motsAbsentsDansSecondConteneurMot(const ConteneurMot& liste1, const ConteneurMot& liste2) {

    ConteneurMot resultat;

    intialiserConteneurMot(resultat);

    for (unsigned int i = 0; i < liste2.nbMots; i++) {

        if (!motPresentdansConteneurMot(liste2.tab[i],liste1))

            ajouterMot(resultat, liste2.tab[i]);

    }

    return resultat;

}

ConteneurMot rechercheDichotomique(ConteneurMot& liste1, const ConteneurMot& liste2) {

    ConteneurMot resultat;

    int max,min,mid;

    if (!estTrie(liste1))

        trierConteneurMot(liste1);

    intialiserConteneurMot(resultat);

    for (unsigned int i = 0; i < liste2.nbMots ; i++) {

        max = liste1.nbMots-1;

        min = 0;

        while (min <= max) {

            mid = (min + max) / 2;

            if (strcmp(liste2.tab[i], liste1.tab[mid]) == 0) {

                ajouterMot(resultat, liste2.tab[i]);

                break;

            }

            if (strcmp(liste2.tab[i], liste1.tab[mid]) < 0) {

                max = mid-1;

            }

            else {

                min = mid+1;

            }

        }

    }

    trierConteneurMot(resultat);

    return resultat;

}

void compterPointsConteneurMot(const ConteneurMot& c) {

    unsigned int points = 0;

    for (unsigned int i = 0; i < c.nbMots; i++) {

        unsigned int taille = strlen(c.tab[i]);

        if (taille <= 2)

            continue;

        if (taille > 2 && taille < 5)

            points++;

        else {

            switch (taille) {

            case 5: points += 2; break;

            case 6: points += 3; break;

            case 7: points += 5; break;

            default: points += 11; break;

            }

        }

    }

    std::cout << points;

}

ConteneurMot motsPresentsDansPlateau(const ConteneurMot& c, Plateau& p) {

    ConteneurMot resultat;

    intialiserConteneurMot(resultat);

    for (unsigned int i = 0; i < c.nbMots; i++) {

        if (recherche(c.tab[i], p))

            ajouterMot(resultat, c.tab[i]);

    }

    trierConteneurMot(resultat);

    return resultat;

}

### ListeConteneurMot.cpp

#include "ListeConteneurMot.h"

void initialiserListeConteneurMot(ListeConteneurMot& l) {

    l.tab = new ConteneurMot[1];

}

void ajouterConteneurMot(ListeConteneurMot& l,ConteneurMot& c){

    if(l.nbConteneurs > 0){

        ConteneurMot\* new\_tab = new ConteneurMot[l.nbConteneurs+1];

        for(unsigned int i = 0 ; i < l.nbConteneurs ; i++){

            new\_tab[i] = l.tab[i];

        }

        delete[]l.tab;

        l.tab = new\_tab;

    }

    l.tab[l.nbConteneurs] = c;

    l.nbConteneurs++;

}

void afficherListeConteneurMot(const ListeConteneurMot& l){

    for(unsigned int i = 0 ; i < l.nbConteneurs ; i++){

        afficherConteneurMot(l.tab[i]);

    }

    std::cout << "\*" << std::endl;

}

void lireClavierListeConteneurMot(ListeConteneurMot& l) {

    while (true) {

        ConteneurMot temp;

        intialiserConteneurMot(temp);

        char\* buffer = new char[30];

        std::cin >> buffer;

        if (\*buffer == '\*')

            break;

        else {

            Mot m = buffer;

            ajouterMot(temp, m);

        }

        lireClavierConteneurMot(temp);

        ajouterConteneurMot(l, temp);

    }

}

void trierListeConteneurMots(ListeConteneurMot& l) {

    for (unsigned int i = 0; i < l.nbConteneurs; i++) {

        trierConteneurMot(l.tab[i]);

    }

}

ConteneurMot MotsPresentsDansMinDeuxConteneurs(ListeConteneurMot& l) {

    ConteneurMot resultat, temp;

    intialiserConteneurMot(resultat);

    for (unsigned int i = 0; i < l.nbConteneurs; i++) {

        for (unsigned int j = 0; j < l.nbConteneurs; j++) {

            if (j == i)

                continue;

            temp = rechercheDichotomique(l.tab[j],l.tab[i]);

            for (unsigned int k = 0; k < temp.nbMots ; k++) {

                ajouterMot(resultat, temp.tab[k]);

            }

        }

    }

    trierConteneurMot(resultat);

    return resultat;

}

### Main.cpp

#include <iostream>

#include "ListeConteneurMot.h"

*/\**

*[brief] : Fonction correspondant � l'ex�cution du sprint 1*

*\*/*

void exo1();

*/\**

*[brief] : Fonction correspondant � l'ex�cution du sprint 2*

*\*/*

void exo2();

*/\**

*[brief] : Fonction correspondant � l'ex�cution du sprint 3*

*\*/*

void exo3();

*/\**

*[brief] : Fonction correspondant � l'ex�cution du sprint 4*

*\*/*

void exo4();

*/\**

*[brief] : Fonction correspondant � l'ex�cution du sprint 5*

*\*/*

void exo5();

*/\**

*[brief] : Fonction correspondant � l'ex�cution du sprint 6*

*\*/*

void exo6();

int main() {

    unsigned int choix;

    std::cin >> choix;

    switch(choix){

        case 1: exo1(); break;

        case 2: exo2(); break;

        case 3: exo3(); break;

        case 4: exo4(); break;

        case 5: exo5(); break;

        case 6: exo6(); break;

        default: exit(1);

    }

}

void exo1() {

    ConteneurMot c;

    intialiserConteneurMot(c);

    lireClavierConteneurMot(c);

    compterPointsConteneurMot(c);

}

void exo2() {

    ConteneurMot c;

    intialiserConteneurMot(c);

    lireClavierConteneurMot(c);

    trierConteneurMot(c);

    afficherConteneurMot(c);

}

void exo3() {

    ConteneurMot liste1;

    ConteneurMot liste2;

    intialiserConteneurMot(liste1);

    intialiserConteneurMot(liste2);

    lireClavierConteneurMot(liste1);

    lireClavierConteneurMot(liste2);

    trierConteneurMot(liste1);

    trierConteneurMot(liste2);

    afficherConteneurMot(motsAbsentsDansSecondConteneurMot(liste1, liste2));

}

void exo4() {

    ConteneurMot liste1;

    ConteneurMot liste2;

    intialiserConteneurMot(liste1);

    intialiserConteneurMot(liste2);

    lireClavierConteneurMot(liste1);

    lireClavierConteneurMot(liste2);

    trierConteneurMot(liste1);

    trierConteneurMot(liste2);

    afficherConteneurMot(rechercheDichotomique(liste1, liste2));

}

void exo5() {

    ListeConteneurMot liste;

    initialiserListeConteneurMot(liste);

    lireClavierListeConteneurMot(liste);

    afficherConteneurMot(MotsPresentsDansMinDeuxConteneurs(liste));

}

void exo6() {

    Plateau p;

    lireClavierPlateau(p);

    ConteneurMot c;

    intialiserConteneurMot(c);

    lireClavierConteneurMot(c);

    afficherConteneurMot(motsPresentsDansPlateau(c, p));

}