

---

# Rapport du projet

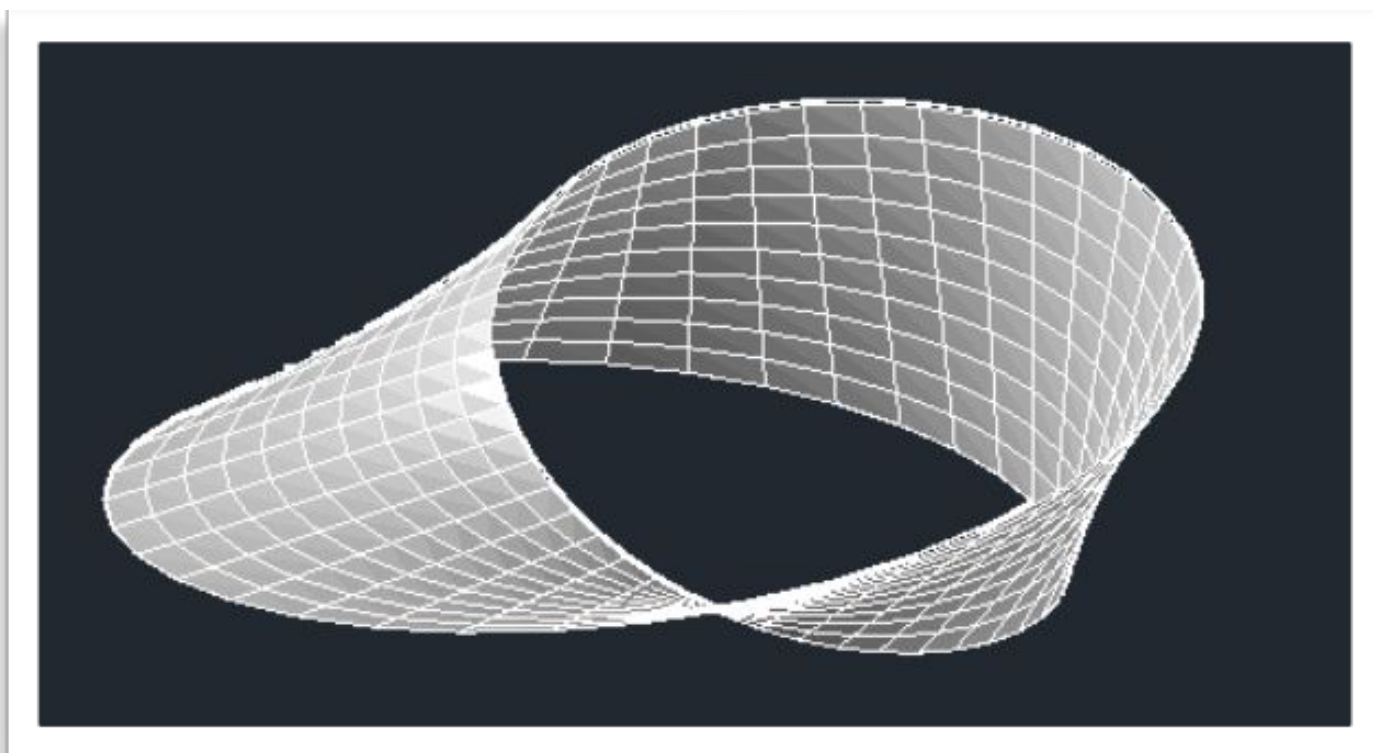
## Module Structure des Données et Algorithmes

Fangyan Lisa YE (groupe 111)

Mélodie RATAVO (groupe 112)

Imane HAÏF (groupe 104)

---



IUT DE PARIS



---

# Table des matières

Présentation de l'application.....	page 3
Graphe de dépendance des fichiers sources.....	page 4
Organisation et résultats des tests .....	page 5
Bilan de validation.....	page 8
Bilan du projet .....	page 9
Annexes .....	page 10

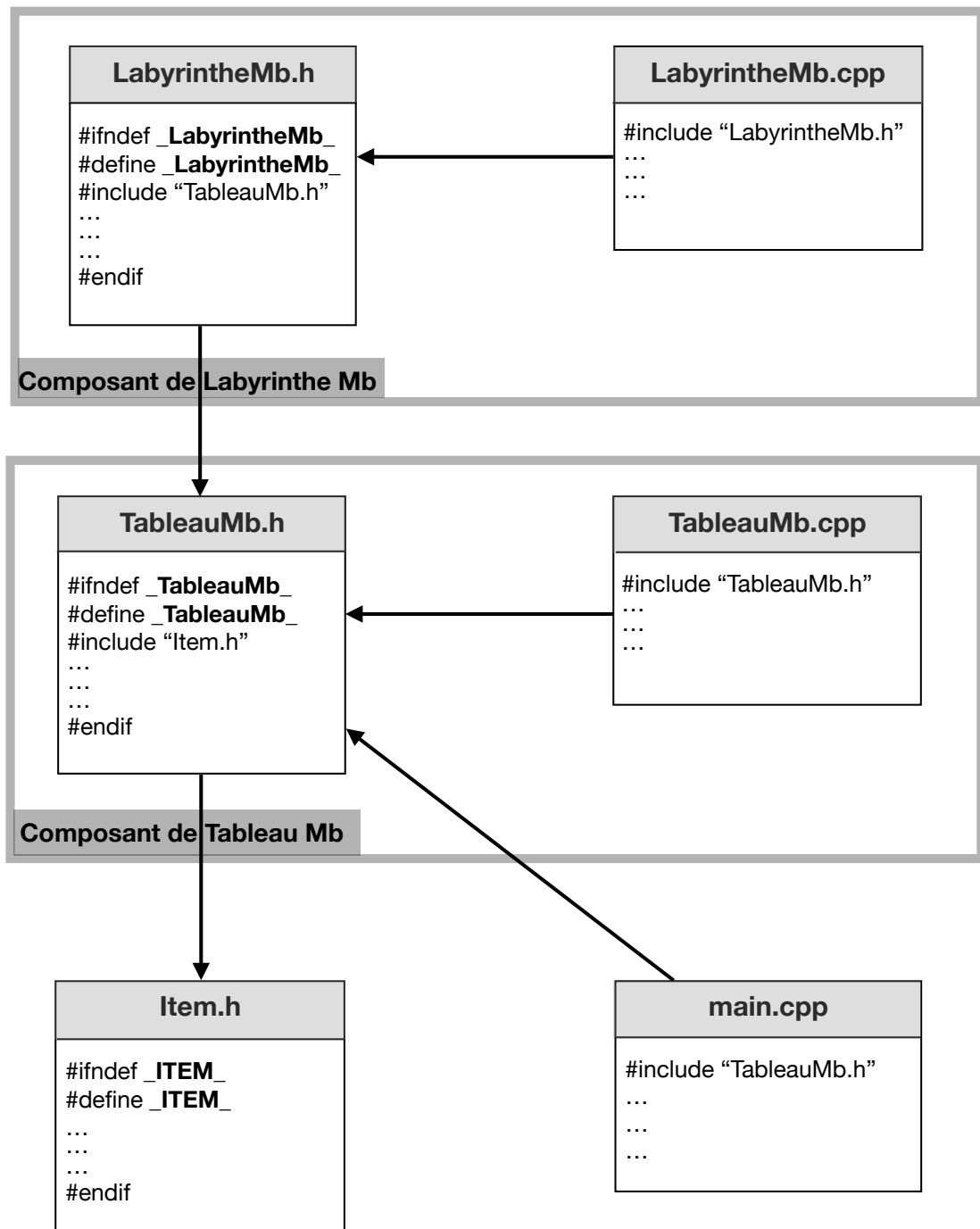
---

# Présentation de l'application

Cette application consiste à à faire en sorte qu'un dragon se déplaçant dans un labyrinthe pour trouver les Plans du Monde puisse ensuite en ressortir grâce au chemin aller vers les plans qu'il aura mémorisé. Le dragon peut seulement se déplacer grâce aux 8 déplacements autorisés lorsqu'il est sur une case : ouest, nord-ouest, nord, nord-est, est, sud-est, sud, sud-ouest. Il ne peut se déplacer que sur les "+" et doit éviter les murs "#" l'empêchant de passer. À chaque déplacement, les "+" seront remplacés par un chiffre. Le dragon se déplaçant sur un ruban de Möbius qui est un ruban infini, lorsqu'il sera sur les extrémités du ruban et qu'il avancera une fois de plus, il se retrouvera sur l'autre face du ruban.

Le sprint 1 consistait à créer des programmes permettant à la fois de lire le labyrinthe, de le mémoriser dans deux tableaux dynamiques en 2 dimensions (pour chaque face du labyrinthe) et de les détruire une fois exploités.

# Graphe de dépendance des fichiers sources

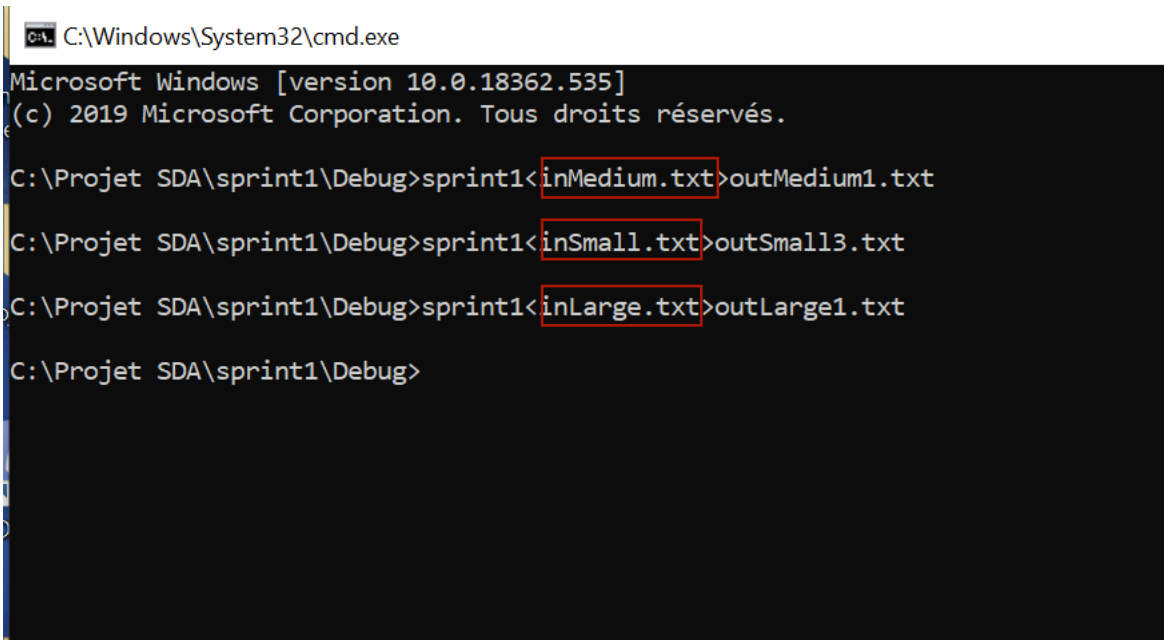


---

# Organisation et résultats des tests

Afin de savoir si notre programme fonctionnait, nous avons d'abord commencé par vérifier si le fichier qui devait être lu par l'algorithme était lisible grâce à un `if / else`. Puis, concernant les jeux de données test (JDT), nous avons comparé les résultats que l'on obtenait une fois l'algorithme exécuté avec les *out* donnés dans le cours projet. Si les résultats étaient identiques aux *out*, alors nous considérons que le sprint fonctionnait.

**Ci-dessous, les jeux de données test (JDT) utilisés pour le sprint 1 et utilisables pour les autres sprints. Il s'agit de labyrinthes de tailles différentes (petit, moyen et grand) servant à tester les sprints.**



```
C:\Windows\System32\cmd.exe
Microsoft Windows [version 10.0.18362.535]
(c) 2019 Microsoft Corporation. Tous droits réservés.

C:\Projet SDA\sprint1\Debug>sprint1<inMedium.txt>outMedium1.txt
C:\Projet SDA\sprint1\Debug>sprint1<inSmall.txt>outSmall3.txt
C:\Projet SDA\sprint1\Debug>sprint1<inLarge.txt>outLarge1.txt
C:\Projet SDA\sprint1\Debug>
```

Le *out* du sprint 1 avec pour entrée le petit labyrinthe (inSmall.txt) :

```
30 4
#####D#####
+#+#+#+#+#+#####+#+#+#+#+#+
#+#+#+#+#+#####+#+#+#+#+#+
#####
#####
+#+#####+#+#+#+#+#+#+#+#+#+
+++++P+++++#+#+#+#+#+#+#+#+#+
#####

Sortie de C:\Projet SDA\sprint1\Debug\sprint1.exe
Pour fermer automatiquement la console quand le d
ment la console à l'arrêt du débogage.
Appuyez sur une touche pour fermer cette fenêtre.
```

Comparaison du *in* (inSmall.txt) avec le *out* du sprint 1 :

The two files are identical

Editor ▾

↔

Save Diff

Share

Original Text	Changed Text
<pre>1 30 4 2 #####D##### 3 +#+#+#+#+#+#####+#+#+#+#+#+ 4 +#+#+#+#+#+#####+#+#+#+#+#+ 5 ##### 6 7 ##### 8 +#+#####+#+#+#+#+#+#+#+#+ 9 +++++P+++++#+#+#+#+#+#+#+#+ 10 ##### 11 12</pre>	<pre>1 30 4 2 #####D##### 3 +#+#+#+#+#+#####+#+#+#+#+#+ 4 +#+#+#+#+#+#####+#+#+#+#+#+ 5 ##### 6 7 ##### 8 +#+#####+#+#+#+#+#+#+#+#+ 9 +++++P+++++#+#+#+#+#+#+#+#+ 10 ##### 11 12</pre>

## Test du *in* inLarge.txt sur le sprint 1:

The two files are identical

Editor ▾

↔

Save Diff

Share

Original Text

Changed Text

61

62

63

64

60

61

62

Le débogueur a été très utile, notamment lorsqu'il s'agissait de corriger nos erreurs dans les algorithmes mêmes. Le débogueur nous a aussi beaucoup aidé à effectuer le graphe de dépendance entre les différents fichiers utilisés dans le sprint 1.

# Bilan de validation

La recette du projet étant un jour après la recette du dossier, nous ne sommes pas en mesure de donner le bilan de validation. Cependant, nous avons pu tester le jeu de données test (JDT) qu'un professeur nous a donné. Nous avons testé notre sprint le plus élevé, qui est le sprint 1, avec ce jeu de données test (JDT) et cela a fonctionné.

**Jeu de données test (JDT) du professeur valide :**

The two files are identical

Editor ▾



Save Diff

Share

Original Text

Changed Text

14	##### ##### #####	1	##### ##### #####
15	##### ##### #####	2	##### ##### #####
16	##### ##### #####	3	##### ##### #####
17	##### ##### #####	4	##### ##### #####
18	##### ##### #####	5	##### ##### #####
19	##### ##### #####	6	##### ##### #####
		7	##### ##### #####



---

# Bilan du projet

Ce projet nous a permis d'améliorer notre compréhension du langage C++. En effet, nous avons pu consolider les bases vues en cours et revues en TD et en TP.

Nous avons rencontré des problèmes quant à la gestion du temps car nous avions beaucoup de projet à rendre en même temps. De plus, du fait de la grève des transports qui a eu lieu pendant toute la période où nous devions effectuer le projet, nous avons eu énormément de mal à nous retrouver pour le faire. Nous avons donc utilisé l'outil google drive pour partager le projet entre les membres du groupe.

Il a aussi été difficile de séparer les fonctions et de trouver les dépendances entre ces dernières. Mais nous avons finalement réussi à le faire au mieux en s'entraidant.

Nous avons demandé de l'aide à nos camarades et à nos professeurs pour pouvoir comprendre et pour obtenir des précisions sur ce qu'il y avait à faire pour les sprints lorsque l'on rencontrait des difficultés.

Nous nous sommes aidé des corrections des TP de SDA afin d'avancer dans notre projet. Grâce à ce projet, nous avons pu retravailler le langage C++ pour les DST.

Pour le prochain projet, nous pourrions améliorer notre gestion du temps et notre organisation afin d'avancer au maximum sur le projet.

---

# Annexes

## Programme principal : main.cpp

```
main.cpp
1- /**
2  * @file main.cpp
3  * Projet SDA
4  * @author YEfangyuan Lisa (111), HAIF Imane (104), RATOVO Mélodie (112)
5  * @brief programme principal
6  */
7
8 #include<fstream>
9 #include<iostream>
10 #include<iomanip>
11 #include <cassert>
12 #include "TableauMb.h"
13 using namespace std;
14- /** @brief appelle toutes les fonctions pour
15  * afficher un labyrinthe composé de deux faces
16  */
17- int main() {
18     Lab lab; // initialiser le labyrinthe constitué par deux tableaux 2D
19     lab.tab1.nbC = lab.tab2.nbC = 0;
20     lab.tab1.nbl = lab.tab2.nbl = 0;
21     ifstream fichier;
22     fichier.open("C:\\inLarge.txt", ifstream::in); //Ouverture d'un fichier en lecture // ouverture du fichier pour le lire
23-     if (!fichier) {
24         cout << "ERREUR: Impossible d'ouvrir le fichier en lecture." << endl;
25     }
26-     else {
27         initialisation(lab, fichier); //création des tableaux 2D et enregistrement du contenu du fichier dans ces tableaux
28         afficherSP1(lab); // afficher le labyrinthe
29         destruction(lab); // détruire les tableaux
30     }
31     return 0;
32 }
```

## Composants utilisés par l'application : TableauMb.h

```
main.cpp  TableauMb.h :
1- /**
2-  * @file TableauMb.h
3-  * Projet SDA
4-  * @author Yefangyuan Lisa (111), HAIF Imane (104), RATOVO Mélodie (112)
5-  */
6- #ifndef _TableauMb_
7- #define _TableauMb_
8- #include "Item.h"
9- /** @brief Conteneur d'items alloués en mémoire dynamique
10-  * de capacité extensible suivant un pas d'extension
11-  */
12-
13- struct Tab2D {
14-     Item** tab; // adresse du tableau bidimensionnel
15-                // en mémoire dynamique
16-     int nbl;    // nombre de lignes de la matrice
17-     int nbC;    // nombre de colonnes de la matrice
18- };
19-
20- struct Lab {
21-     Tab2D tab1; //1er niveau tableau dynamique 2 dimension
22-     Tab2D tab2; //2er niveau tableau dynamique 2 dimension
23- };
24-
25- /** @brief initialise Lab
26-  * @param[in/out] lab, le labyrinthe
27-  */
28- void initialiser(Lab& lab);
29- /** @brief détruit Lab
30-  * @param[in] lab, le labyrinthe
31-  */
32- void detruire(Lab& lab);
33- /** @brief initialise Lab
34-  * @param[in] fichier, fichier entré pour la lecture
35-  * @param[in/out] lab, deux tableaux 2D
36-  */
37- void initialisation(Lab& lab, std::ifstream& fichier);
38- /** @brief détruit le labyrinthe
39-  * @param[in] lab, le labyrinthe
40-  */
41- void destruction(Lab& lab);
42- /** @brief affiche lab
43-  * @param[in] lab, le labyrinthe
44-  */
45- void afficherSP1(const Lab& lab);
46- #endif
47-
```

## LabyrintheMb.h

```
main.cpp  TableauMb.h  LabyrintheMb.h
1  #ifndef _LabyrintheMb_
2  #define _LabyrintheMb_
3  #include "TableauMb.h"
4  /** @brief Conteneur d'items alloués en mémoire dynamique
5   * de capacité extensible suivant un pas d'extension
6   */
7
8  /** @brief initialise tableau 2D
9   * param[in] t, tableau 2D
10  */
11 void initialiser(Tab2D& t, unsigned int c, unsigned int p);
12
13 /** @brief détruit tableau 2D
14  *param[in/out] t, tableau 2D
15  */
16 void detruire(Tab2D t);
17
18 /** @brief affiche t
19  *param[in/out] t, tableau 2D
20  */
21 void afficherSP1(const Tab2D& t);
22 #endif
```

## Item.h

```
main.cpp  TableauMb.h  LabyrintheMb.h  Item.h
1  #ifndef _ITEM_
2  #define _ITEM_
3  /**
4   * @file Item.h
5   * Projet SDA
6   * @author YEfanguyan Lisa (111), HAIF Imane (104), RATOV0 Mélodie (112)
7   * @brief Spécialisation du type Item
8   */
9  typedef char Item; // Item chaîne de caractère
10 #endif
```

## Fichiers corps : TableauMb.cpp

```
main.cpp | TableauMb.h | LabyrintheMb.h | Item.h | TableauMb.cpp |
1- /**
2-  * @file TableauMb.cpp
3-  * Projet SDA
4-  * @author Yefangyuan Lisa (111), HAIF Imane (104), RATOVO Mélodie (112)
5-  */
6-
7- #include<fstream>
8- #include<iostream>
9- #include<iomanip>
10- #include <cassert>
11- #include "TableauMb.h"
12- using namespace std;
13-
14- /** @brief initialise les deux tableaux 2D qui constituent le lab
15-  * @param[out] lab, le conteneur de Tableaux 2D
16-  */
17- void initialiser(Lab& lab) {
18-     int nbL, nbC;
19-     nbC = lab.tab1.nbC;
20-     nbL = lab.tab1.nbL;
21-     lab.tab1.tab = new Item * [nbL]; // création du premier tableau 2D
22-     for (int i = 0; i < nbL; ++i) {
23-         lab.tab1.tab[i] = new Item[nbC];
24-     }
25-     lab.tab2.tab = new Item * [nbL]; // création du deuxième tableau 2D
26-
27-     for (int i = 0; i < nbL; ++i) {
28-         lab.tab2.tab[i] = new Item[nbC];
29-     }
30- }
31- /** @brief détruit le labyrinthe qui contient les tableaux 2D
32-  * @param[in] lab, labyrinthe
33-  */
34- void detruire(Lab& lab) {
35-     for (int i = 0; i < lab.tab1.nbL; ++i) { // détruit le premier tableau 2D
36-         delete[] lab.tab1.tab[i];
37-         lab.tab1.tab[i] = NULL;
38-     }
39-     delete[] lab.tab1.tab;
40-     lab.tab1.tab = NULL;
41-     for (int i = 0; i < lab.tab1.nbL; ++i) { // détruit le deuxième tableau 2D
42-         delete[] lab.tab2.tab[i];
43-         lab.tab2.tab[i] = NULL;
44-     }
45-     delete[] lab.tab2.tab;
46-     lab.tab2.tab = NULL;
47- }
```

## LabyrintheMb.cpp

```
main.cpp  TableauMb.h  LabyrintheMb.h  Item.h  :  TableauMb.cpp  LabyrintheMb.cpp  :
1- /**
2-  * @file Item.h
3-  * Projet SDA
4-  * @author YEfangyuan Lisa (111), HAIF Imane (104), RATOVO Mélodie (112)
5-  */
6- #include<fstream>
7- #include<iostream>
8- #include<iomanip>
9- #include"LabyrintheMb.h"
10- using namespace std;
11- /** @brief initialise Lab
12-  * @param[in] fichier, fichier entré pour la lecture
13-  * @param[in/out] lab, deux tableaux 2D
14-  */
15- void initialisation(Lab& lab, std::ifstream& fichier) {
16-     int nbL;
17-     int nbC;
18-     //Tout est prêt pour la lecture.
19-     fichier >> lab.tab1.nbC; // initialiser le nombre de colonnes
20-     fichier >> lab.tab1.nbL; // initialiser le nombre de lignes
21-     lab.tab2.nbC = lab.tab1.nbC;
22-     lab.tab2.nbL = lab.tab1.nbL;
23-     nbL = lab.tab1.nbL;
24-     nbC = lab.tab1.nbC;
25-     initialiser(lab); // créer deux tableaux 2D
26-     for (int j = 0; j < nbL; j++) { //enregistre la première face du labyrinthe du fichier dans le tab1
27-         for (int i = 0; i < nbC; i++) {
28-             fichier >> lab.tab1.tab[j][i];
29-         }
30-     }
31-     for (int j = 0; j < nbL; j++) { //enregistre la deuxième face du labyrinthe du fichier dans le tab2
32-         for (int i = 0; i < nbC; i++) {
33-             fichier >> lab.tab2.tab[j][i];
34-         }
35-     }
36-     fichier.close(); // fermer fichier
37- }
38- /** @brief affiche lab
39-  * @param[in] lab, le labyrinthe
40-  */
41- void afficherSP1(const Lab& lab) {
42-     cout << lab.tab1.nbC << " " << lab.tab1.nbL << endl;
43-     for (int j = 0; j < lab.tab1.nbL; j++) { // afficher la première face du labyrinthe
44-         for (int i = 0; i < lab.tab1.nbC; i++) {
45-             cout << lab.tab1.tab[j][i];
46-         }
47-         cout << endl;
48-     }
49-     cout << endl;
50-     for (int j = 0; j < lab.tab1.nbL; j++) { // afficher la deuxième face du labyrinthe
51-         for (int i = 0; i < lab.tab1.nbC; i++) {
52-             cout << lab.tab2.tab[j][i];
53-         }
54-         cout << endl;
55-     }
56- }
57- /** @brief détruit le labyrinthe
58-  * @param[in] lab, le labyrinthe
59-  */
60- void destruction(Lab& lab) {
61-     void detruire(Lab & lab);
62- }
```

## Jeux de données test (JDT) personnels

Nous avons créé deux Jeux de données test (JDT) à partir des labyrinthes déjà donnés. Nous avons déplacé le départ D, remplacé des murs par des passages et inversement et même ajouté une colonne au labyrinthe. Ces tests ont parfaitement fonctionné.

The two files are identical

Editor ▾



Save Diff

Share

Original Text

Changed Text

```
1 30 4
2 #####D#####
3 +#+#+#+#+#####+
4 #+#+#+#+#+#####+
5 #####
6
7 #####
8 +#+#+#+#+#+#####+
9 ++++P++++#+#+#+#+
10 #####
11
12
```

```
1 30 4
2 #####D#####
3 +#+#+#+#+#####+
4 #+#+#+#+#+#####+
5 #####
6
7 #####
8 +#+#+#+#+#+#####+
9 ++++P++++#+#+#+#+
10 #####
11
12
```

The two files are identical

Editor ▾



Save Diff

Share

Original Text

Changed Text

```
1 31 4
2 #####D#####
3 +#+#+#+#+#####+
4 #+#+#+#+#+#####+
5 #####
6
7 #####
8 +#+#+#+#+#+#####+
9 +++++#+#+#+#+P++++
10 #####
11
12
```

```
1 31 4
2 #####D#####
3 +#+#+#+#+#####+
4 #+#+#+#+#+#####+
5 #####
6
7 #####
8 +#+#+#+#+#+#####+
9 +++++#+#+#+#+P++++
10 #####
11
12
```

