Selvaratnam Akash groupe 107

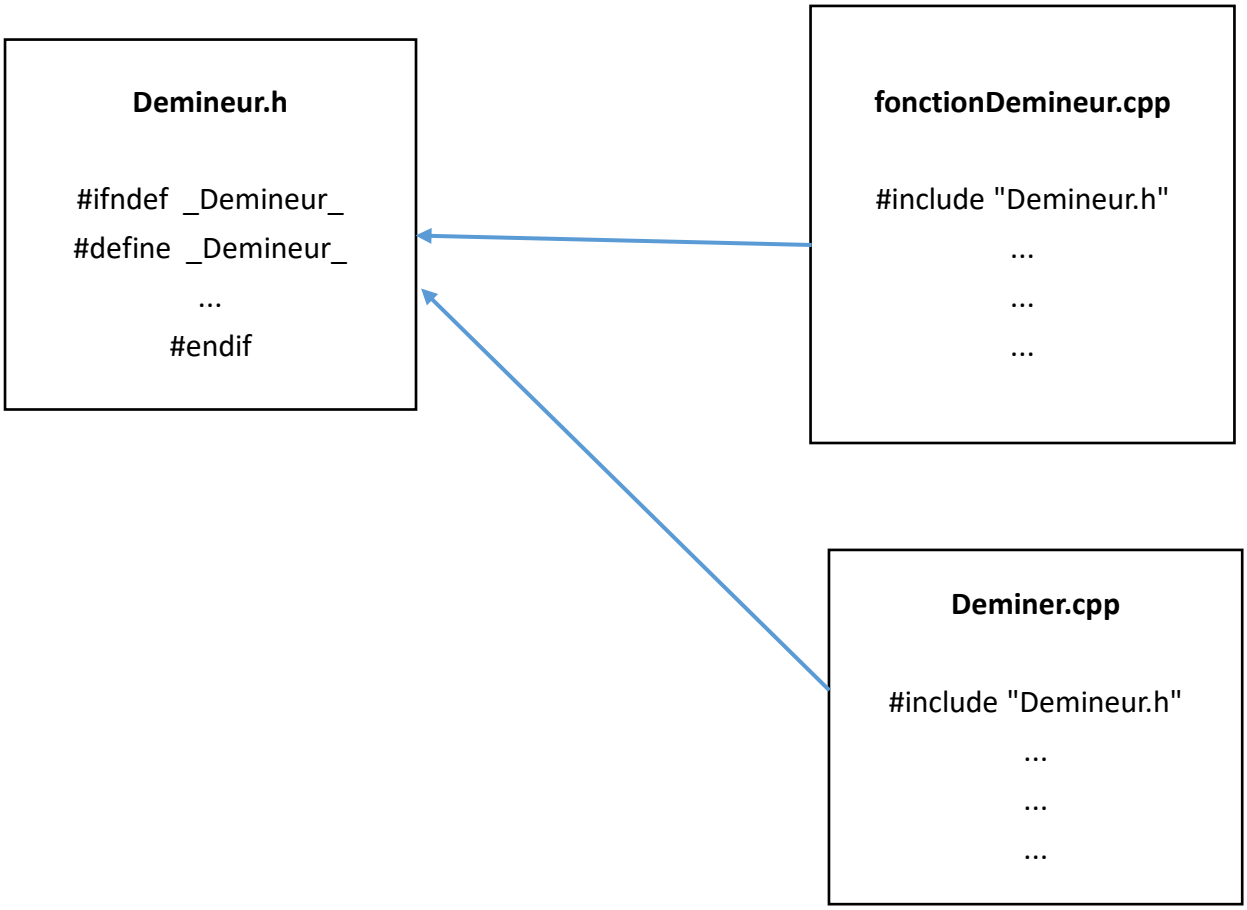
Casenaz Lucas groupe 112

Minesweeper

Présentation du projet

Le jeu du démineur, consiste à devoir trouver toutes les mines, sans les faire exploser. Ce jeu n'est pas un jeu d'hasard, il se joue avec une réflexion particulière. Sur l'image ci-dessous, ce trouve un jeu de démineur résolu par mes soins. Chaque drapeau représente une mine. Les chiffres représentent le nombre de mines ce trouvant aux alentours, par exemple, lorsqu'on voit un chiffre un, ça veut dire qu'il n'y a qu'une seule mine dans les cases autour de lui (représente en rouge ci-dessous). Le but du jeu est de ne pas tomber sur une mine, auquel cas le jeu sera perdu et ressemblera à la seconde image. Toutes les mines sont dévoilées, et il est impossible de reprendre la partie. Pour cela, nous devons demander au joueur la taille de sa grille en choisissant un nombre de colonnes et de lignes et il doit également donner le nombre de mines dont il a envie dans sa grille puis le programme génère les positions aléatoires des mines. Après avoir obtenu ces informations, nous devons créer une grille en fonction des données fourni par le joueur, pour que le joueur puisse jouer il doit donner le nombre de fois dont il a envie de jouer accompagné des positions des cases dont il a envie de jouer, il peut également masquer les cases de la grille. Nous devons informer le joueur si la partie est gagné, elle est gagnée si toutes les cases ne contenant pas de mines a été dissimulés et qu'aucune case ne contenant une mine a été dissimulé alors le programme dit que la partie est gagné (game won) sinon il nous affiche que la partie n'est pas gagné (game not won) soit la case dissimuler est une mine ou toutes les cases ne contenant pas de mines n'ont pas été tous découvertes. On doit également informer le joueur si la partie est perdue (game lost), une partie est perdue si une case qui contient une mine a été dissimulé ou une case ne contenant pas de mine a été masqué sinon il affiche la partie n'est pas perdue (game not lost) et si on reçoit une grille en entré il faut que le programme soit capable de ressortir un coup possible dans cette grille, un coup est disponible si elle est cachée par un point.

Graphes de dépendances des fichiers sources



Les jeux d'essai

Dans la première commande nous devons entrer dans le compilateur, le nombre de lignes, le nombre de colonnes pour notre grille de Démineur et également insérer les nombres de mines qu'on veut dans cette grille puis le programme ressort les nombres de lignes, de colonnes et de mines entré par le joueur suivi des positions des mines généré aléatoirement par le programme.

Fichier in de la commande 1 : 1 6 3 4

Fichier out de la commande 1 : 6 3 4 2 9 16 4

Dans la deuxième commande nous devons entrer dans le compilateur, le nombre de lignes, le nombre de colonnes, le nombre de mines suivi des positions aléatoirement générer par la première commande, le nombre de coups et les coups joué par le joueur. Le programme ressort le nombre de ligne, de colonnes et nous affiche juste en dessous la grille.

Fichier in de la commande 2 : 2 4 6 5 1 5 12 7 19 3 D15 M5 D0

Fichier out de la commande 2 :

4 6

| 2 || . || . || . || . || x |

| . || . || . || . || . || . |

| . || . || . || 2 || . || . |

| . || . || . || . || . || . |

Fichier in de la commande 2 : 2 4 6 5 1 5 12 7 19 4 D15 M5 D0 D19

Fichier out de la commande 2 :

4 6

| 2 || . || . || . || . || x |

| . || . || . || . || . || . |

| . || . || . || 2 || . || . |

| . || m || . || . || . || . |

Dans la troisième commande nous devons entrer dans le compilateur, le nombre de lignes, le nombre de colonnes, le nombre de mines suivi des positions aléatoirement générer par la première commande, le nombre de coups et les coups joué par le joueur. Le programme

ressort si la partie est gagné donc game won, une partie est gagné toutes les cases sauf ceux qui contiennent des mines ont été démasqué sinon il affiche game not won.

Fichier in de la commande 3 : 3 4 6 5 1 5 12 7 19 3 D15 M5 D0

Fichier out de la commande 3 : game not won

Fichier in de la commande 3 : 3 4 6 5 1 5 12 7 19 4 D15 M5 D0 D19

Fichier out de la commande 3 : game not won

Fichier in de la commande 3 : 3 3 3 3 7 8 9 7 D0 D1 D2 D3 D4 D5 D6

Fichier out de la commande 3 : game won

Dans la quatrième commande nous devons entrer dans le compilateur, le nombre de lignes, le nombre de colonnes, le nombre de mines suivi des positions aléatoirement générer par la première commande, le nombre de coups et les coups joué par le joueur. Le programme ressort si la partie est perdue game lost, une partie est perdue si aucune case contenant une mine à été dévoilé et aucune case ne contenant pas de mines a été masqué sinon il affiche game not lost.

Fichier in de la commande 4 : 4 4 6 5 1 5 12 7 19 3 D15 M5 D0

Fichier out de la commande 4 : game not lost

Fichier in de la commande 4 : 4 4 6 5 1 5 12 7 19 4 D15 M5 D0 D19

Fichier out de la commande 4 : game lost

Fichier in de la commande 4 : 4 3 3 3 5 2 4 3 D7 D2 M6

Fichier out de la commande 4 : game lost

Bilan du projet

Nous avons effectué le projet du démineur avec un réel plaisir car nous avons appris de nombreuses choses comme utiliser les pointeurs avec une référence, ce qui était une réelle difficulté lors du projet du Classement WTA donc on est vraiment satisfait de maîtriser les pointeurs grâce à ce projet, on a également mieux compris comment utiliser les boucles while, if ou for et on a compris comment réaliser différente unité logicielle. Au cour de ce projet, on a aussi rencontré des difficulté surtout sur la commande 2, nous avons pas réussi à placer les chiffres qui représentent le nombres de mines se trouvant aux alentours, nous avons essayé plusieurs solutions mais aucune qui nous a afficher un résultat convenable nous avons donc placer le chiffre 2 sur toutes les case ne qui ne contient pas de mines, c'est-à-dire que si le joueur démasque une case différente d'une case mine alors la case afficher est 2 et le reste des cases non joué affiche un point, mais ceci ne pas le seul problème de notre commande, nous avons également un problème sur l'affichage de toutes les cases de la grille car quand une case contenant une mine à été découverte, toute la grille doit s'afficher mais dans notre cas, même si le joueur a démasqué une case contenant une mine alors seulement les cases choisi par le joueur sont découverte, nous avons essayé une solution en effectuant une boucle if dans un while inferieur au nombre de mines, cette boucle if a pour condition que si la case démasqué par le joueur sur la grille doit avoir une valeur égale à m et la variable j, cette variable parcours le tableau, doit avoir une valeur différentes des mines alors le programme affiche toutes la grille mis a part la case 0 et 1 et notre dernière difficulté est sur l'affichage de grille, nous avons pas réussi à afficher le signe (_) en dessous de chaque cases de la grilles. Aux niveau des réussites du projet, nous avons réussi a réaliser un problème en générant aléatoirement les positions des mines grâce à la bibliothèque <random>, mis à part le problème des chiffres indiquant le nombre de mines se trouvant aux alentours, nous avons tout de même réussi à générer une grille, à faire jouer le joueur en démasquant ou masquant une case et a placer les bons symboles sur la grille c'est-à-dire (le lettre x pour masquer une case, le chiffre 2 pour démasquer une case, la lettre m pour démasquer une mine). Nous avons réussi à effectuer la commande 3 en affichant (game won) si une partie est gagné, elle est gagnée si toutes les cases ne contenant pas de mines a été dissimulés et qu'aucune case ne contenant une mine a été dissimulé alors le programme dit que la partie est gagné sinon il nous affiche que la partie n'est pas gagné (game not won) soit la case dissimuler est une mine ou toutes les cases ne contenant pas de mines n'ont pas été tous découvertes. Nous avons aussi réussi à effectuer la commande 4 en affichant (game lost) si une partie est perdue, une partie est perdue si une case qui contient une mine a été dissimulé ou une case ne contenant pas de mine a été masqué sinon il affiche la partie n'est pas perdue (game not lost). Ce qui aurait pu être améliorer dans notre projet est surtout aux niveaux des boucles (if, while, for), je pense qu'il est possible de réaliser ce projet en effectuant moins de boucle, cela aurait été beaucoup plus soigneux donc mieux présenter.

Demineur.h

```
#include <iostream>
using namespace std;

#ifndef _Demineur_
#define _Demineur_

typedef struct {//structure de grille
	int lignes;//nombre de ligne
	int colonnes;//nombre de colonne
	int mines;//nombre de mines
	int max;//nombre de case dans une grille
	int j[255];//les positions des mines
}Grille;

typedef struct {//structure du tableau du Demineur
	char tab[255];//le tableau
}tableau;

typedef struct {//Structure de Personne
	int nbcoup;//le nombre de fois dont le joueur à envie de jouer
	char coup;//le type de coup
	int numcase;//la position du coup
}Personne;

typedef struct loose {//Structure de loose
	int g;//Partie Perdu
	int l;//Partie non perdu
};

typedef struct win {//structure de win
	int ga;//Partie gagné
	int ng;//Partie non gagné
};

void probleme(Grille& g, int i);

void grille(Grille& g, tableau& t, Personne& p, int i);

void winner(Grille& g, tableau& t, Personne& p, win& w, int i, int v);

void looser(Grille& g, tableau& t, Personne& p, loose& l, int i, int v);




#endif // !_Demineur_
```

fonctionDemineur.cpp

```
#include <iostream>
#include <random>
using namespace std;


#include "Demineur.h"
/**
*@brief Creation d'un problememe
*param[in, out] le nombre de ligne, de colonne et de mines
*param[out] les positions des mines générer aléatoirement
*/
void probleme(Grille& g, int i) {
	int v = 0;
	srand(time(NULL));
	cin >> g.lignes >> g.colonnes >> g.mines;//Entree des ligne, colonne et mines
par le joueur
	g.max = g.lignes * g.colonnes;// calcul de la capacité total de la grille
	cout << g.lignes << " " << g.colonnes << " " << g.mines;//afficher les lignes,
les colonnes et le nombre de mines entre par le joueur
	for (i; i < g.mines; i++) {//boucle for pour générer les positions des mines
		g.j[i] = rand() % g.max;//Générer les positions des mines aléatoirement
entre 0 et la capacité total
		while (v < g.mines && g.j[i] == g.j[v]) {//boucle while pour vérifier si
les positions des mines génerer ne sont pas identitique
			g.j[i] = rand()% g.max;//si c'est le cas alors génerer de nouveau
une position de mines
			v++;//incrémentation de v
		}
		if (v < g.mines) {//Si v est inférieur aux nombres de mines alors
			v = 0;//v est a 0
		}
		cout << " " << g.j[i];//afficher les positions des mines
	}
}
/**
@**Création de la grille et les coups joué par le joueur
*param[in] le nombre de mines, les positions des mines, le nombre de coup joué et les
coup joué
*param[in, out] le nombre de ligne, le nombre de colonne
*param[out] le tableau
*/
void grille(Grille& g, tableau& t, Personne& p, int i) {
	int k = 0;
	int l = 0;
	int s = 0;
	cin >> g.lignes >> g.colonnes >> g.mines;
	g.max = g.lignes * g.colonnes;
	while (k < g.mines) {//boucle while pour entrer les mines generer dans la
fonction probleme
		cin >> g.j[k];
		k++;
	}
	cin >> p.nbcoup;//entrer le nombre de fois dont on a envie de jouer
	cout << g.lignes << " " << g.colonnes << "\n";//afficher le nombre de lignes et
de colonnes

	for (int j = 0; j < g.max; j++) {//boucle for pour la création de la grille
```

```
		if (j < p.nbcoup) {//boucle if pour entre quelle case on a envie de masquer ou demasquer
			cin >> p.coup >> p.numcase;
		}
		while (i < g.mines) {
			if (p.coup == 'D' && p.numcase == g.j[i]) {//Si le coup joué est un coup démasqué et le numéro de case est égal a un numéro de mine alors la case dévoilé est m
				t.tab[p.numcase] = 'm';
				break;
			}
			if (p.coup == 'D' && p.numcase != g.j[i]) {//Si le coup joué est un coup démasqué et le numéro de case est differnet a un numéro de mine alors la case dévoilé est 2
				t.tab[p.numcase] = '2';
			}
			if (p.coup == 'M') {//Si le coup joué est un un coup masqué alors on affiche x a la case démandé
				t.tab[p.numcase] = 'x';
			}
			i++;
		}
		if (i == g.mines) {//boucle if permettant de remettre la variable i à 0
			i = 0;
		}
		while (l < g.max) {
			if (t.tab[l] != 'm' && t.tab[l] != '2' && t.tab[l] != 'x') {//Si les cases devoilé sont différent de m, de 2 ou de x alor afficher un point sur la case
				t.tab[l] = '.';
			}
			l++;
		}
		if (l == g.max) {//boucle if permettant de remettre la variable l à 0
			l = 0;
		}
	}
	for (int k = 0; k < g.max; k++) {//boucle for pour l'affichage de la grille
		if (k > 0 && k % g.colonnes == 0) {//si k est superieur et k modulo le nombre de colonnes est egale a 0  alors sauter une ligne
			cout << "\n";
		}
		cout << "|" << " " << t.tab[k] << " " << "|";// afficher le resultat
	}
}


/**
*@brief Informer le joueur si la partie est gagné ou elle est en cour
*param [in] le nombre de ligne, de colonne, de mines, les positions des mines, le nombre de coup joué et les coup joué
*/
void winner(Grille& g, tableau& t, Personne& p, win& w, int i, int v) {
	int k = 0;
	int l = 0;
	w.ga = 5;
	w.ng = 5;
	cin >> g.lignes >> g.colonnes >> g.mines;
	g.max = g.lignes * g.colonnes;
	while (k < g.mines) {//boucle while pour entrer les mines generer dans la fonction probleme
		cin >> g.j[k];
		k++;
```

```
	}
	cin >> p.nbcoup;//entrer le nombre de fois dont on a envie de jouer
	for (int j = 0; j < g.max; j++) {//Création de la grille
		if (j < p.nbcoup) {//Coup joué par le joueur
			cin >> p.coup >> p.numcase;
		}
		while (i < g.mines) {
			if (p.coup == 'D' && p.numcase == g.j[i]) {//Si le coup joué est
un coup démasqué et le numéro de case est égal a un numéro de mine alors la case
dévoilé est m
				t.tab[p.numcase] = 'm';
				break;
			}
			if (p.coup == 'D' && p.numcase != g.j[i]) {//Si le coup joué est
un coup démasqué et le numéro de case est differnet a un numéro de mine alors la case
dévoilé est 2
				t.tab[p.numcase] = '2';
			}
			if (p.coup == 'M') {//Si le coup joué est un un coup masqué alors
on affiche x a la case démandé
				t.tab[p.numcase] = 'x';
			}
			i++;
		}
		if (i == g.mines) {//boucle if permettant de remettre la variable i à 0
			i = 0;
		}
		while (l < g.max) {//Si les cases devoilé sont différent de m, de 2 ou de
x alor afficher un point sur la case
			if (t.tab[l] != 'm' && t.tab[l] != '2' && t.tab[l] != 'x') {
				t.tab[l] = '.';
			}
			l++;
		}
		if (l == g.max) {//boucle if permettant de remettre la variable l à 0
			l = 0;
		}
		while (v < g.mines) {
			if (j != g.j[v] && t.tab[j] != '.' || j == g.j[v] && t.tab[j] !=
'm') {//si la grille d'indice j different des indices des mines est different a une
valeur point ou la grille j d'indice les cases des mines a une valeur differente de m
alors c'est égale à 0
				w.ga = 0;
			}
			if (j != g.j[v] && t.tab[j] == '.' || j == g.j[v] && t.tab[j] ==
'm') {//si la grille d'indice j different des indices des mines est égale a un point
ou la grille j d'indice les cases des mines a une valeur égale a m alors c'est égale à
1
				w.ng = 1;
			}
				v++;
			}
		if (j == g.max -1 && w.ga == 0) {//quand j a atteint la valeur max - 1 de
la grille est que w.ga est égal a 0 alors il affiche game won
			cout << "game won";
		}
		if (j == g.max -1 && w.ng == 1 && w.ga != 0) {//quand j a atteint la
valeur max - 1 de la grille est que w.ng est égal a 0 et w.ga different de 0 alors il
affiche game not won
			cout << "game not won";
		}
```

```
	}
}
/**
*@brief Informer le joueur si la partie est perdu ou elle est n'est pas perdue
*param [in] le nombre de ligne, de colonne, de mines, les positions des mines, le
nombre de coup joué et les coup joué
*/
void looser(Grille& g, tableau& t, Personne& p, loose& lo, int i, int v) {
	int k = 0;
	int l = 0;
	int tmp = 0;
	lo.g = 5;
	lo.l = 5;
	cin >> g.lignes >> g.colonnes >> g.mines;
	g.max = g.lignes * g.colonnes;
	while (k < g.mines) {//boucle while pour entrer les mines generer dans la
fonction probleme
		cin >> g.j[k];
		k++;
	}
	cin >> p.nbcoup;//entrer le nombre de fois dont on a envie de jouer
	for (int j = 0; j < g.max; j++) {//Création de la grille
		if (j < p.nbcoup) {
			cin >> p.coup >> p.numcase;//Coup joué par le joueur
		}
		while (i < g.mines) {
			if (p.coup == 'D' && p.numcase == g.j[i]) {//Si le coup joué est
un coup démasqué et le numéro de case est égal a un numéro de mine alors la case
dévoilé est m
				t.tab[p.numcase] = 'm';
				break;
			}
			if (p.coup == 'D' && p.numcase != g.j[i]) {//Si le coup joué est
un coup démasqué et le numéro de case est differnet a un numéro de mine alors la case
dévoilé est 2
				t.tab[p.numcase] = '2';
			}
			if (p.coup == 'M') {// Si le coup joué est un un coup masqué alors
on affiche x a la case démandé
				t.tab[p.numcase] = 'x';
			}
			i++;
		}
		if (i == g.mines) {//boucle if permettant de remettre la variable i à 0
			i = 0;
		}
		while (l < g.max) {//Si les cases devoilé sont différent de m, de 2 ou de
x alor afficher un point sur la case
			if (t.tab[l] != 'm' && t.tab[l] != '2' && t.tab[l] != 'x') {
				t.tab[l] = '.';
			}
			l++;
		}
		if (l == g.max) {//boucle if permettant de remettre la variable l à 0
			l = 0;
		}
		while (v < g.mines) {
			if (p.coup == 'D' && p.numcase == g.j[v] || p.coup == 'M' &&
p.numcase != g.j[v]) {//Si le coup joué est Demasquer et le numero case est égal a une
position de mines ou une case est masquer a une position differente des positions de
mine alors lo.g est égale a 0
```

```
                        lo.g = 0;
                    }
                    if (p.coup == 'D' && p.numcase != g.j[v] || p.coup == 'M' &&
p.numcase == g.j[v]) {//Si le coup joué est Demasquer et le numero case est différent
à une position de mines ou une case est masquer est égal a une positions de mine alors
lo.l est egale a 1
                        lo.l = 1;
                        tmp++;
                    }
                    v++;
            }
            if (tmp == 1) {
                    lo.g = 2;
            }
            if (tmp <= 5) {
                    tmp = 0;
            }
            if (v == g.mines) {
                    v = 0;
            }
            if (j == g.max-1 && lo.g == 0) { //si j atteint le nombre de case maximum
et lo.g est egale a 0 alors il affiche game lost
                    cout << "game lost";
            }
            if (j == g.max-1 && lo.l == 1 && lo.g != 0) {//si j atteint le nombre de
case maximum et lo.l est egale a 1 et lo.g est diffirent de 0 alors il affiche game
not lost
                    cout << "game not lost";
            }
      }
}
```

Deminer.cpp

```
#include <iostream>
using namespace std;


#include "Demineur.h"


int main(void) {
	int commande;//initialisation de la variable commande
	int i = 0;//la variable i prend la valeur de 0
	int v = 0;//La variable v prend la valeur de 0
	cin >> commande;// Entré de lo commande
	Grille g;
	tableau t;
	Personne p;
	loose l;
	win w;

	if (commande == 1) {//Si la commande est egal a 1 alors executer la fonction
probleme
		probleme(g, i);
	}
	else if (commande == 2) {//Si la commande est egal a 2 alors executer la
fonction grille
		grille(g, t, p, i);
	}
	else if (commande == 3) {//Si la commande est egal a 3 alors executer la
fonction winner
		winner(g, t, p, w, i, v);
	}
	else if (commande == 4) {//Si la commande est egal a 4 alors executer la
fonction looser
		looser(g, t, p, l, i, v);
	}
	return 0;

}
```