main.c

```
/**
* @file main.c
* @author Durand Thomas
* @brief main qui permet de lancer le jeux de la vie grace au
chemin d'un fichier d'initialisation de plateau
* @version 0.1
* @date 2019-10-16
*
* @copyright Copyright (c) 2019
*
*/
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>


#include "plateau.h"
#include "jeux.h"
/**
* @brief main qui lance le jeux de la vie en fonction d'un fichier
de config
*
* @param argc nombre d'arguments
* @param argv 1: chemin du fichier de configuration
* @return int
*/
int main(int argc, char *argv[]){

if (argc == 2){
char* path = argv[1];
int X = calculeX(path);
int Y = calculeY(path);
int** plateau = init(X,Y,path);
while (1){
affichage(plateau,X,Y);
calculerNouvelleGrille(plateau,X,Y);
sleep(1);
}
}else{
printf("veillez rentre un fichier d'initialisation\n");
}
}
```

plateau.c

```
/**
* @file plateau.c
* @author Durand Thomas
* @brief ensemble des fonctions permettant de gerer le plateau
(initialisation et affichage)
* @version 0.1
* @date 2019-10-16
*
* @copyright Copyright (c) 2019
*
*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/**
* @brief ouvre un fichier
*
* @param path chemin du fichier
* @return FILE*
*/
FILE* lireFichier(char* path){
FILE* fichier = NULL;
fichier = fopen(path, "r+");
if (fichier == NULL){
printf("Impossible d'ouvrir le fichier\n");
exit(EXIT_SUCCESS);
}
return fichier;
}


/**
* @brief calcul axe X
*
* @param path chemin du fichier
* @return int
*/
int calculeX(char* path){
FILE* fichier = lireFichier(path);
char ligne[200];
fgets(ligne,200,fichier);
```

```
int i = 0;
int N;
while (ligne[i] ≠ '\n'){
if(ligne[i] ≠ ' '){
N ++;
}
i++;
}
fclose(fichier);
return N;
}
/**
* @brief calcul axe Y
*
* @param path chemin du fichier
* @return int
*/
int calculeY(char* path){
FILE* fichier = lireFichier(path);
int N=1;
int c;
for (c = getc(fichier); c ≠ EOF; c = getc(fichier))
if (c == '\n') // Increment count if this character is newline
N++;
fclose(fichier);
return N;
}

/**
* @brief initialise la matrice de jeu
*
* @param X taille axe X
* @param Y taille axe Y
* @param path chemin du fichier
* @return int**
*/
int** init(int X,int Y,char* path){
int** plateau = malloc(Y*sizeof(int*));
for (int i = 0; i ⩽ Y; i++){
plateau[i] = malloc(X*sizeof(int));
}
FILE* fichier = lireFichier(path);
char ligne[200];
int placei = 1;
```

```
int placej = 1;
int j;
for (int i = 1; i ⩽ Y; i++){
fgets(ligne,200,fichier);
j=0;
while (ligne[j] ≠ '\n'){
if (ligne[j] == 48){
plateau[placei][placej] = 0;
placej++;
}else if (ligne[j] == 49){
plateau[placei][placej] = 1;
placej++;
}
j++;
}
placej = 1;
placei++;
}
return plateau;
}

/**
* @brief affiche la matrice dan sle terminal
*
* @param plateau matrice de jeu
* @param X taille axe X
* @param Y taille axe Y
*/
void affichage(int** plateau,int X,int Y){
system("clear");
for (int i = 1; i ⩽ X; i++){
printf("+---");
}
printf("+\n");
for (int i = 1; i ⩽ Y; i++){
for (int j = 1; j ⩽ X; j++){
if (plateau[i][j] == 0){
printf("| ");
}else{
printf("| %d ",plateau[i][j]);
}
}
printf("|\n");
for (int i = 1; i ⩽ X; i++){
```

```
printf("+---");
}
printf("+\n");
}
}
```

jeux.c

```
/**
* @file jeux.c
* @author Durand Thomas
* @brief ensemble des fonctions permetant le calcul d'une
iteration de la grille
* @version 0.1
* @date 2019-10-16
*
* @copyright Copyright (c) 2019
*
*/
#include <stdio.h>
#include <stdlib.h>

/**
* @brief change les coordonnes passe en parametre pour permettre
d'avoir une grille torique
*
* @param x coordonne a change
* @param N taille de la matrice
* @return int
*/
int bonneCoord(int x,int N){
if (x < 1){
return N;
}else if (x > N){
return 1;
}else{
return x;
}
}

/**
* @brief compte le nombre de cellule voisine a une cellule
*
* @param tab matrice de jeu
```

```
* @param x coordonne x
* @param y coordonne y
* @param X taille X de la matrice
* @param Y taille Y de la matrice
* @return int
*/
int compteCellulesVoisines(int** tab,int x, int y, int X,int Y){
int rez = 0;
for (int i = -1; i < 2; i++){
for (int j = -1; j < 2; j++){
if (i =/= 0 || j =/= 0){
//inversion X Y
int xtemp = bonneCoord(x + i,Y);
int ytemp = bonneCoord(y + j,X);
if (tab[xtemp][ytemp] == 1){
rez += 1;
}
}
}
}
return rez;
}

/**
* @brief calcule la nouvelle valeur d'une cellule apres un tour de
jeu
*
* @param tab matrice de jeu
* @param x coordonne x
* @param y coordonne y
* @param X taille X de la matrice
* @param Y taille Y de la matrice
*
* @return int
*/
int calculerNouvelleValeur(int** tab,int x ,int y,int X,int Y){

int voisin = compteCellulesVoisines(tab,x,y,X,Y);

if (tab[x][y] == 1){
if (voisin == 2 || voisin ==3){
return 1;
}else{
return 0;
```

```
}
}else{
if (voisin == 3){
return 1;
}else{
return 0;
}
}
}

/**
* @brief calcule la nouvelle grille apres un tour de jeu
*
* @param tab matrice de jeu
* @param X taille X de la matrice
* @param Y taille Y de la matrice
*/
void calculerNouvelleGrille(int** tab,int X,int Y){
int** nvltab = malloc(Y*sizeof(int*));
for (int i = 0; i <= Y; i++){
nvltab[i] = malloc(X*sizeof(int));
}

for (int i = 1; i <= Y; i++){
for (int j = 1; j <= X; j++){
nvltab[i][j] = calculerNouvelleValeur(tab,i,j,X,Y);
}
}

for (int i = 1; i <= Y; i++){
for (int j = 1; j <= X; j++){
tab[i][j] = nvltab[i][j];
}
}
free(nvltab);
}
```