Nous avons codé le jeu classique du démineur : il y a un certain nombre de bombes cachée dans une grille, que l’on affiche sur le terminal et il faut réussir à les trouver sans les faire exploser. Il y a une légère différence cependant : il ne faut pas ici marquer les bombes avec un drapeau mais réussir à afficher toutes les cases libres.

Le joueur interagit grâce au terminal, on lui demande de rentrer une valeur de ligne et de colonne correspondant à la case qu’il souhaite déminer, le résultat lui est ensuite affiché également sur le terminal. Ce processus est réitéré jusqu’à ce que le joueur ait réussi à afficher toutes les cases libres, ou bien qu’il soit tombé sur une bombe.

Explication de l’algorithme :

Nous avons évidemment découpé l’algorithme en un maximum de fonction, pour mener à bien le projet et le garder compréhensible.

Il y a donc plusieurs grandes tâches à accomplir :

* Créer la grille, et l’initialiser.
* Placer
* Afficher les grilles
* Jouer

Pour la création de la grille, nous avons opté pour un tableau en deux dimensions, à savoir un tableau de tableau. Chaque case est une structure avec deux entiers. L’un des entiers « val » correspond au nombre de bombes voisines pour une case vide ou à la présence d’une bombe si la valeur est 9. Le second entier « bool » permet de savoir s’il faut afficher la case ou non.

On initialise au départ la grille avec des valeurs à zéro, puis on place le nombre choisi de bombes, sans les mettre deux fois au même endroit. Lorsque l’on place les bombes on s’assure également que la valeur des cases voisines s’incrémentent.

Pour afficher la grille, on regarde la valeur « bool » qui nous dit si la case doit apparaitre ou non, puis on fait en sorte de l’afficher sous forme de grille dans le terminal.

Une dimension importante du programme est la propagation : quand le joueur choisit une case vide, il faut afficher les cases vides de proche en proche jusqu’à afficher l’ensembles des cases voisines de bombes les plus proches. Cela nécessite donc un algorithme de propagation à part entière, nous avons choisi d’en écrire un en récursif, quand le joueur choisit une case de valeur nulle, on retourne l’algorithme de propagation des cases voisines, si on tombe sur une case de valeur non nulle on affiche sa valeur.

Il y a ensuite une partie du code qui consiste à pouvoir jouer, on demande directement au joueur de nous donner les coordonnées de la case qu’il souhaite dévoiler et on découvre la grille à chaque tour après propagation.

Pour faire tourner le jeu, on code également une fin en cas de victoire ou de défaite, et dans une boucle, on répète les actions de choix du joueur, de propagation et d’affichage.

Notre code est sur une seule feuille de code, où nous avons rassemblé toutes les fonctions nécessaires à notre main.