# LIFAP1 – TD 11 : Exemple de Conception

*Objectifs :* Réviser toutes les notions abordées jusqu'à présent dans les travaux dirigés et travaux pratiques. Au travers d'un exemple de conception, analyser un problème et concevoir sa solution pas à pas.

Il s'agit dans cette séance de TD ainsi que dans la séance de TP qui suivra (TP 9) de réaliser un jeu de démineur. Dans le TD, les déclarations de types et les sous-programmes demandés devront être écrits en langage C/C++.

**Principe du jeu :**


L'objectif est de trouver les mines qui sont cachées aléatoirement par l'ordinateur dans les cases du tableau.

Si la case choisie contient une mine, la partie est perdue. Si la case choisie ne contient pas de mine alors apparaîtra un chiffre indiquant le nombre de mines qui se trouvent dans les 8 cases qui touchent directement la case sélectionnée.

Par exemple si le numéro découvert est un 2, cela indique qu'il y a 2 mines cachées parmi les 8 cases qui touchent directement celle choisie.

Attention : seul un affichage en mode texte sera demandé.

**Construction du jeu :**

1. Structures de données :
   a. Définir 3 constantes Taille_Grille_X, Taille_Grille_Y et Mines permettant de fixer les paramètres généraux du jeu. Dans l'exemple précédent, on aura comme valeurs respectives (9, 9 et 10).

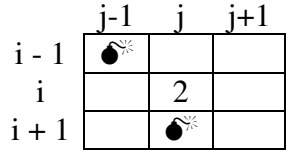
   b. Définir une structure *case_grille* comportant deux informations :
      - un entier représentant le contenu de cette case (-1 si mine, 0..8 pour le nombre de mines dans les cases adjacentes)
      - un booléen permettant de savoir si cette case a déjà été choisie par l'utilisateur. Cette valeur indiquera si la case doit être dévoilée à l'utilisateur lors de l'affichage.

   c. Définir la grille de jeu comme étant un tableau 2D de *case_grille*. Pour des besoins ultérieurs, nous allons volontairement surdimensionner la grille de jeu en ajoutant une ligne supplémentaire en haut et en bas et une colonne supplémentaire à droite et à gauche.

2. Initialisation de la grille de jeu : écrire un sous-programme permettant d'initialiser la grille de jeu. Pour chacune des cases de la grille du jeu, la valeur numérique sera initialisée à 0 et le booléen découvert à la valeur "false".

3. Positionnement aléatoire des bombes sur la grille de jeu : écrire un sous-programme permettant de positionner aléatoirement *Mines* bombes sur la grille de jeu. Attention, avant de placer une mine, on vérifiera que la case est "libre".

4. Remplissage complet de la grille de jeu. Dans ce sous-programme, nous allons pour chacune des cases (hors mine) compter le nombre de mines dans toutes les cases adjacentes. Pour éviter une gestion trop complexe des cases du pourtour, nous avons, dans la déclaration, pris soin de rajouter des cellules supplémentaires. Écrire le sous-programme permettant de remplir la grille avec ces valeurs.
   Exemple pour le calcul de la case i, j :




5. Affichage de la grille de jeu. La grille de jeu est stockée en interne sous forme d'un tableau de structures *case_grille* contenant une valeur numérique (nombre de mines ou -1) et un booléen indiquant si la case doit être affichée ou non. Pour rendre la grille plus lisible pour l'utilisateur, les cases non dévoilées seront affichées avec le symbole '-', les cases contenant une mine (-1) avec le caractère 'M' et les autres cases avec la valeur numérique correspondant au nombre de mines dans les cases adjacentes. Écrire un sous-programme permettant d'afficher le contenu de la grille de jeu.

Exemple pour une grille de taille 5*5 avec 4 mines.

| | | | | | | |
|---|---|---|---|---|---|---|
| 0/f | 0/f | 0/f | 0/f | 0/f | 0/f | 0/f |
| 0/f | 1/f | 1/f | 1/t | 1/f | -1/t | 0/f |
| 0/f | 1/f | -1/f | 1/f | 1/f | 1/f | 0/f |
| 0/f | 2/f | 3/t | 2/t | 1/t | 0/t | 0/f |
| 0/f | -1/f | 2/f | -1/f | 1/t | 0/t | 0/f |
| 0/f | 1/f | 2/t | 1/t | 1/f | 0/t | 0/f |
| 0/f | 0/f | 0/f | 0/f | 0/f | 0/f | 0/f |

*Grille stockée*

| | | | | |
|---|---|---|---|---|
| - | - | 1 | - | M |
| - | - | - | - | - |
| - | 3 | 2 | 1 | 0 |
| - | - | - | 1 | 0 |
| - | 2 | 1 | - | 0 |

*Grille affichée*

**Jeu**

1. Choix d'une case : écrire un sous-programme qui demande à l'utilisateur les coordonnées de la case à sonder.

2. Jeu : écrire le sous-programme permettant de recommencer le choix d'une case jusqu'à la fin de la partie.

3. Fin de partie : la partie s'arrête quand toutes les cases ont été découvertes sauf celles contenant des mines (partie gagnée) ou bien lorsqu'une mine est touchée (partie perdue). Écrire le programme principal permettant de jouer au démineur.