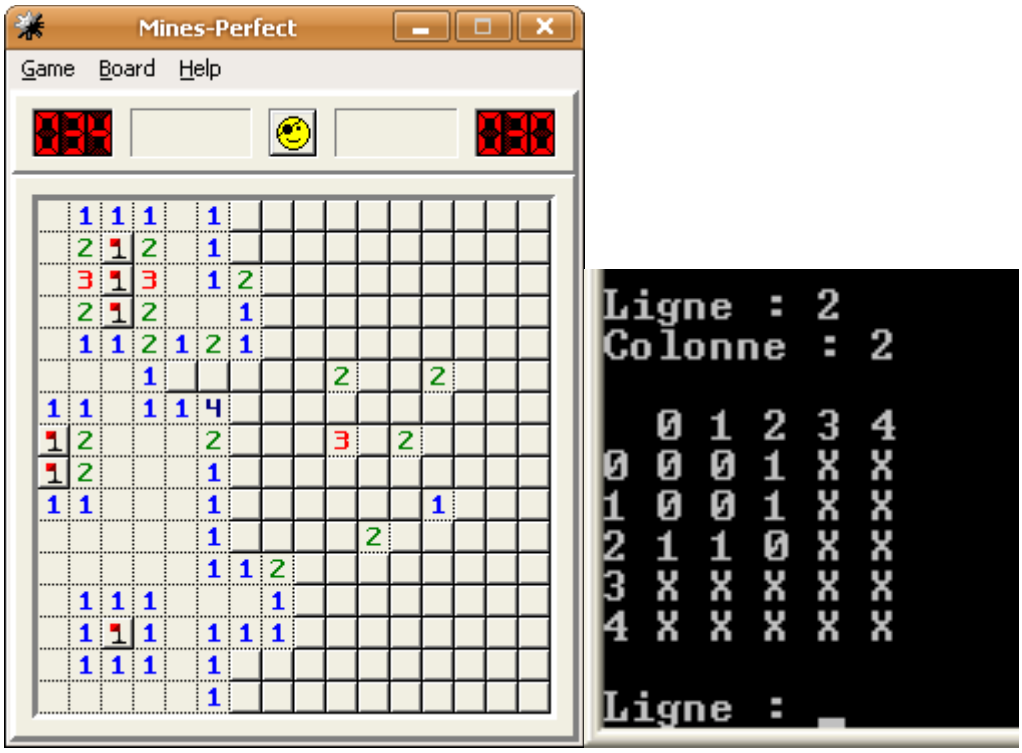
Projet Langage C : semaine A12 – A13

Travail à réaliser (deadline : 16.12.15 à 16h35)

Le but de ce projet est de réaliser le jeu du **Démineur**.



Règles du jeu :

- Le champ de mines est représenté par une grille
- Chaque case de la grille peut soit cacher une mine, soit être vide.
- Le but du jeu est de découvrir toutes les cases libres sans faire exploser les mines (c'est-à-dire sans sélectionner les cases qui les dissimulent).
- Lorsque le joueur sélectionne une case libre et que toutes les cases adjacentes le sont également, une case vide est affichée.
- Si au moins l'une des cases avoisinantes contient une mine, un chiffre apparaît, indiquant le nombre de cases adjacentes contenant une mine.
- Le jeu est chronométré, ce qui permet de conserver les meilleurs scores.

TO DO (acquis)

- Afficher la grille (taille de la grille de base : 10X10, nombre de mines : 10)
- Placer les mines aléatoirement.
  - Eviter le chevauchement des mines
- Donner à l'utilisateur la possibilité de :
  - dévoiler le contenu d'une case.
  - marquer une case comme bombe.
  - effacer la marque « bombe » de la case.
- Chaque case de la grille doit être une **structure** *Case* avec les champs suivants :

  - BOOLEAN isClicked (où BOOLEAN est une variable de type **enum**) : elle dit si la case a été cliquée et donc son contenu doit être affiché.
  - isMarkedAs {noBomb, bomb …} : elle dit si la case a été marquée comme bombe ou autre.
  - Valeur de la case (nombre de bombes adjacentes à la case sélectionnée).
  - … (ajouter des champs si nécessaire).
- Contrôle de saisie de manière à ce que le joueur ne puisse pas saisir des coordonnées invalides.
- Contrôler si le joueur a gagné/perdu.
  - Le joueur gagne s'il découvre toutes les cases « noBomb » ou s'il marque comme « bomb » toutes (et uniquement) les cases qui cachent une bombe.
  - Le joueur perd s'il découvre une case avec une bombe.
- Garder dans un fichier texte les scores des 10 dernières parties :
  - Nom, prénom joueur, temps employé pour gagner.
- Expliquer le code avec des commentaires.
- Utiliser la programmation modulaire.

TO DO (Très Bien Acquis)

- Ajouter différents niveaux de difficulté (en changeant la taille de la grille, le nombre de mines).
- Possibilité de marquer une case comme « dangereuse » (modifier la structure Case pour inclure cette possibilité).
- Si le joueur clique sur une case vide, libérer automatiquement toutes les cases vides à côté.
- Sauvegarder dans le même fichier texte aussi les 10 « All Time Best Scores » classés en ordre décroissent.

A rendre (à la fin de chaque séance):

- Code source (fichiers .c et .h)
- Rapport de Service (RdS) :
  - Liste (**schématique**) des activités de la journée
    - fonctions développées
    - difficultés rencontrées
    - solutions adoptées
  - Activités à compléter dans les prochaines séances
- Le **dernier** RdS sera le rapport final sur le projet

L'évaluation du projet se fera sur le code, la modularité, les fonctionnalités disponibles, les commentaires, l'indentation du code et la qualité & ponctualité des RdS rendus.

Suggestions

- Pour la gestion du temps dans le code vous pouvez vous inspirer de :
  - http://en.wikipedia.org/wiki/C_date_and_time_functions ou
  - http://cboard.cprogramming.com/c-programming/7015-time-h-counter.html (dernier post)

- Faites des **backups réguliers de votre programme !!!**
- Pour la création de la grille vous pouvez vous inspirer de votre travail sur la bataille navale
- system("PAUSE") [ou getch () dans **conio.h**];
    - Met la console en pause jusqu'à ce que l'on appuie sur une touche du clavier
- system("cls") [ou clrscr() dans **conio.h**];
    - Efface le contenu de la console