Mini projet : la bataille navale

Christophe DENIS

1 Consignes

Un rapport de quelque pages vous est demandé. Ce rapport doit comporter les éléments suivants :

- une introduction qui énonce clairement le sujet, le plan du document, *etc* ;
- une partie qui décrit les algorithmes que vous avez utilisés pour faire le mini-projet. Ces algorithmes seront écrits de préférence en langage naturel ; en particulier, vous pourrez y souligner vos contributions personnelles par rapport au sujet initial ;
- un mode d'emploi du programme ;
- une conclusion dans laquelle vous citerez toutes les améliorations que vous pourriez apporter à votre programme.

Les apports personnels contribuent de façon très significatives à une bonne note du projet sachant que le respect du cahier des charges (c'est-à-dire ce qui vous est demandé dans la section "Présentation du projet" de chaque sujet), le code C fonctionnel correspondant et un rapport qui suit les régles énoncées précédemment vous assure au moins la moyenne.
Les règles suivantes sont à respecter lors de l'écriture du programme :

- mettre des commentaires à bon escient dans le programme (ne pas commenter chaque ligne du programme mais préciser par exemple le rôle des variables) ;
- choisir des noms de variables explicites ;
- indenter le programme.

2 Présentation du mini-projet

L'objectif du mini-projet est de développer un jeu de bataille navale. Le joueur saisit tout d'abord la taille *taille_plateau* de du plateau de jeu. Le programme ensuite place aléatoirement six navires de taille variant de 2 à 6 cases sur ce plateau de jeu.

Il est demandé ensuite au joueur une case sur laquelle il veut lancer un torpille. Le programme ensuite affiche :

- touché si sur cette case est placé un navire;
- a l'eau si sur cette case n'est pas placé un navire;
- deja joué si cette case a déjà été jouée.
- une grille de jeu de taille $taille\_plateau \times taille\_plateau$ pour laquelle une case est représentée par le signe

- x, si la case a eté jouée et si un navire est placé sur cette case;
- o, si la case a eté jouée et si aucun navire est pas placé sur cette case;
- ., si la case n'a pas eté jouée.

Lorsque toutes les cases d'un navire ont été touchées un message indique que le navire en précisant sa taille a été coulée.
Ce processus est répété tant qu'il reste des navires non coulés.

3 Modélisation du problème

3.1 Structures de données

Une case du jeu est modélisée par la structure suivante :

```
typedef struct une_case {
  int x; /* position de la case en x*/
  int y; /* position de la case en y*/
} Case;
```

Un navire est modélisé par la structure suivante :

```
typedef struct navire {
  int sens; /* 0 haut 1 droite 2 bas 3 gauche */
  Case premiere_case;
  int taille;
} Navire;
```

Le plateau du jeu *plateau* de taille $taille\_plateau \times taille\_plateau$ sur lequel le programme place les navires est allouée dynamiquement.
La grille de jeu *grille* de taille $taille\_plateau \times taille\_plateau$ est allouée dynamiquement.

3.2 Fonctions à utiliser

La fonction **int nb aleatoire(int max)** qui renvoie un nombre, tiré au hasard, compris entre 1 et max.

```
#include<stdlib.h>
#include<time.h>

/* Initialiser le generateur au debut du main par  l'instruction
suivante*/
 void init_nb_aleatoire() {
    srandom(time(NULL));
}

/* Renvoie un nombre, tiré au hasard, compris entre 1 et max*/
 int nb_aleatoire(int max) {
    return (random()%max);
```

Voici la liste des prototypes des fonctions à utiliser lors du mini-projet. Vous avez la liberté d'en écrire de nouvelles selon vos besoins.

- **Navire creer_navire(int taille,int taille_plateau)** : cette fonction permet de créer un navire d'une taille donnée dont la case de d´epart et le sens sont fixés aléatoirement.
- **int est_valide(int ** plateau,int taille_plateau, struct navire * nav)** : cette fonction retourne 1 s'il est bien situé dans les limites du plateau, et quil ne se chevauche pas avec un autre navire, sinon elle retourne 0.
  - plateau est une matrice représentant le plateau de jeu, dans laquelle la cases occupéees par des navires contiennent un 1 et les autres un 0
- **void initialisation_plateau(int ** plateau,int taille_plateau)** : cette procédure initialise aléatoirement six navires de taille 2 à 6 dans le plateau.
- **void proposition_joueur(int ** plateau, int ** prop, int * NbTouche,int * NbJoue,int * NbToucheNav, int taille_plateau)** : cette fonction demande à l'utilisateur de saisir une case (x,y) à jouer et selon la valeur contenue plateau[x][y] enregistre dans prop[x][y] la valeur :
  - 0 si la case ne contient pas de navire
  - -1 si la case a déjà été joué
  - 1 si la case contient un navire
  - **NbJoue** est le compteur du nombre de coups
  - **NbTouche** est le compteur de cases touchées
  - **NbToucheNav** est un tableau qui contient le nombre de cases touchées pour chaque navire. NbToucheNav[i] indique le nombre de cases touchées pour le navire de taille i.
- **void affichage_plateau(int ** plateau,int taille_plateau)** :

Il est possible utiliser un fichier d'entête (extension *.h) contenant toutes les définitions de vos fonctions (voir avec votre chargé de TD).

4 Améliorations possibles

- Ecriture d'une fonction qui affiche différemment les cases coulées et les cases touchées.
- Sauvegarde et chargement de parties en cours.