CPS

Héloïse Fernandes de Almeida

April 2016

1 Sujet

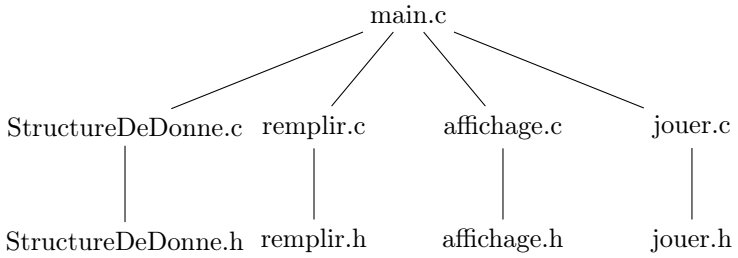
Le but de ce projet est de réaliser une "Bataille Navale".

Fonctionnement du jeu :
1 - Le joueur1 ou 2 appel l'exécutable et passe en paramètre la taille de la grille
2 - On demande au joueur1 de choisir un fichier à charger
3 - Le joueur2 propose des coordonnées et le programme lui répond si les bateaux sont coulé ou non
4 - La partie continue ainsi jusqu'à ce que l'ensemble des bateaux soient coulés. On affiche à la fin le nombre de coup joués

Pour réaliser ce jeu nous avons créer deux versions :
- Les bateaux sont représentés par une structure appelée maillon
- Les bateaux sont représentés par un entier

2 Choix de conception

2.1 Organisation des fichiers



StructureDeDonne.h/.c : ces fichiers contiennent toute les structures de données(structures et constances) nécessaire au bon fonctionnement du programme ainsi que les fonction pour les manipuler. Dans la version 2 ces fichiers contiennent les différentes fonctions set_field et get_field.

affichage.h/.c : Ces fichiers contiennent les fonctions d'affichage.

remplir.h/.c : Ces fichiers permettent de remplir la grille du joueur 1 ou d'initialiser une grille à BLANC.

jouer.h/.c : Ces fichiers implémentent la fonction principale du jeu, joue, ainsi que des fonctions qui mettent à jour la grille du jouer 2 si un bateau a été coulé ou non.

2.2 Structure de données

Nous avons réaliser un ensemble de constante pour ne pas mettre les nombre en dur dans le code et ainsi pouvoir changer ces constante à n'importe quel moment.

```
//Nombre de bateau
#define NB_PORTE_AVION 1
#define NB_CROISEUR 2
#define NB_CONTRE_TORPILLEURS 3
#define NB_TORPILLEURS 4

//Taille des bateaux
#define T_PORTE_AVION 6
#define T_CROISEUR 4
#define T_CONTRE_TORPILLEURS 3
#define T_TORPILLEURS 2

//Differente valeaur de la grille
#define COLORIER 'N'
#define BLANC 'B'
#define RATEE 'X'
#define COULE 'C'
#define TOUCHER 'T'
```

Grille : Nous avons choisis un tableau de caractère à deux dimension car :
- l'affichage est plus simple
- l'accès au différent élément est lui aussi plus simple au niveaux du codage

```
typedef char** grille;
```

Maillon version 1.Dans la deuxième version la structure maillons contient deux champs. L'un représenté la valeur du maillon(un int_32) et l'autre est un pointeur sur le maillon suivant.

```
typedef struct maillon{
int i_deb;
int j_deb;
int i_fin;
int j_fin;
short int coule;
struct maillon * suivant;
}maillon;
```

Liste de navire : Cette structure est composé de deux pointeur l'un sur le début de la liste et l'autre sur la fin de la liste. Le pointeur de fin de liste est là pour facilité l'insertion.Ainsi nous n'avons pas à parcourir l'ensemble de la liste pour pouvoir insérer un maillons.

```
typedef struct liste_navires{
maillon * debut;
maillon * fin;
}liste_navires;
```

2.3 Format du fichier txt contenant la grille du joueur1

Format des fichiers texte :

`10(0;0)(0;1)(0;3)(1;3)(2;3)(0;6)(0;7)(0;8)(0;9)(2;0)(3;0)(4;0)...(9;9)f`

Chaque couple (x;y) correspond à une case colorée dans la grille, la lettre f permet de définir la fin du fichier et le premier nombre correspond à la taille de la grille.

Chaque grille doit respecter les règles suivantes : - les bateaux ne sont pas collés - le nombre de bateaux et leur tailles sont correctes - les bateaux ne se chevauchent pas - les bateaux on des coordonnées qui appartiennent à la grille

Les exemples :

Fichier	Problème
grille1.txt	aucun problème
test_colle.txt	bateaux collés
test_manquant.txt	bateau manquant
test_trop.txt	bateau en trop
test_coordonee.txt	coorodonée en dehors de la grille

3 Choix de programmation

3.1 Fonctions

3.1.1 StructuredeDonnes

Fonction qui initialise la liste des navires :

```
liste_navires * liste_vide();
```

Fonction qui créer un nouveau maillon :

```
maillon * nouveau(int ideb,int ifin, int jdeb, int jfin);
```

Fonction qui insère un maillon dans la liste de navire :

```
void insertion (maillon * m, liste_navires * l);
```

Fonction qui insère l'ensemble des bateaux dans la liste :

```
liste_navires * cree_liste_navires(grille g, int n);
```

Version 2 :
Fonction d'accès vers les différentes valeurs d'un maillon sous forme d'entier :

```
void set_ideb(int *m, int v);
void set_ifin(int *m, int v);
void set_jdeb(int *m, int v);
void set_jfin(int *m, int v);
void set_coule(int *m, int v);
int get_ideb(int *m);
int get_ifin(int *m);
int get_jdeb(int *m);
int get_jfin(int *m);
int get_coule(int *m);
```

3.1.2 affiche

Fonction qui affiche la grille du joueur 1 :

```
void affiche_jeu (grille g, int taille);
```

Fonction qui affiche la grille du joueur 2 :

```
void affiche_etat_coules(grille g, int taille);
```

3.1.3 joue

Fonction qui met à jour la grille du joueur 2 et le bateau en paramètre si celui si est coulé :

```
int navire_coule(maillon *m, int ic, int jc, grille gc);
```

Fonction qui verifie si un bateau a été touché

```
int un_navire_coule(liste_navires l, int ic, int jc, grille gc);
```

Fonction qui permet de donner des coorodonées tant que l'ensemble des bateaux n'a pas été coulé

```
void joue(grille g, grille gc, int n , liste_navires l , int i, int j);
```

Fonction qui détermine la fin du jeu :

```
int jeu_fini(liste_navires l);
```

3.1.4 remplir

Fonction qui permet de choisir un fichier

```
FILE* ouvrir_fichier();
```

Fonction qui permet de vérifier que la grille est complète

```
int complet(grille g,int n);
```

Fonction qui rempli la grille du joueur2

```
void remplir_grille(grille g,int n);
```

Fonction qui initialise la grille à blanc

```
void initialiser_grille(grille g, int n);
```

3.1.5 Fonctionnement de complet

```
int porte_avion = NB_PORTE_AVION;
int croiseur =  NB_CROISEUR;
int contre_torpilleur = NB_CONTRE_TORPILLEURS;
int torpilleur = NB_TORPILLEURS;
//tableau qui fait état des case parcourus ou non(tableau visité)
int tab_est_passe[n][n];
```

Le format de fichier pour charger la grille de jeu est un peu complexe et les joueurs peuvent faire des erreurs. La fonction complet vérifie qu'il n'y a pas d'erreur.

Pour cela on utilise deux tableaux pour l'algorithme l'un correspond à la grille rempli et l'autre correspond au case visitées durant le parcours(au début toute les cases sont initialisés à 0).

Ensuite on parcours la grille ligne par ligne.Lorsque l'on rencontre un case marquée du caractère 'N' dans la grille et de l'entier 0 dans le tableau visite et cherche tout d'abord à déterminer si le bateau est à l'horizontale ou la verticale. Après on parcours le tableau à l'horizontale ou la verticale pour obtenir la taille du bateau.En parallèle les cases 'N' parcourus de la grille sont mis à 1 dans le tableau visité. Pendant le parcours en plus de mettre à jour le tableau visité on vérifie pour chaque case 'N' visité que la case sous elle pour les bateaux horizontaux ou à droite d'elle pour les bateaux verticaux ne sont pas marquée d'un 'N'. Cela permet d'éviter que les bateaux soient collés.

Enfin, quand on a déterminer la taille du tableau on décrément le nombre de bateau correspondant à cette taille(porte_avion, croiseur, contre_torpilleur, torpilleur) tout en vérifiant que le nombre ne soit pas inférieur à 0.

Pour finir avant de terminer la fonction on vérifie que la somme du nombre de bateau encore présent est de 0 (porte_avion+croiseur+contre_torpilleur+torpilleur==0) et si oui la fonction se termine. Dans le cas d'erreur la fonction affiche l'erreur repéré et arrete le programme.

4 Exécution

4.1 Compilation

Pour v1 :

```
=> cd ./Projet/v1/src/
=> make
```

Pour v2 :

```
=> cd ./Projet/v2/src/
=> make
```

4.2 Lancement du programme

Pour v1 :

```
=> cd ./Projet/v1/bin/BatailleNavale 10
=> Quel est votre nom de fichier?  ./exemples/grille1.txt
```

Pour v2 :

```
=> cd ./Projet/v1/bin/BatailleNavale 10
=> Quel est votre nom de fichier?  ./exemples/grille1.txt
```

4.3 Jeu

4.3.1 Touché

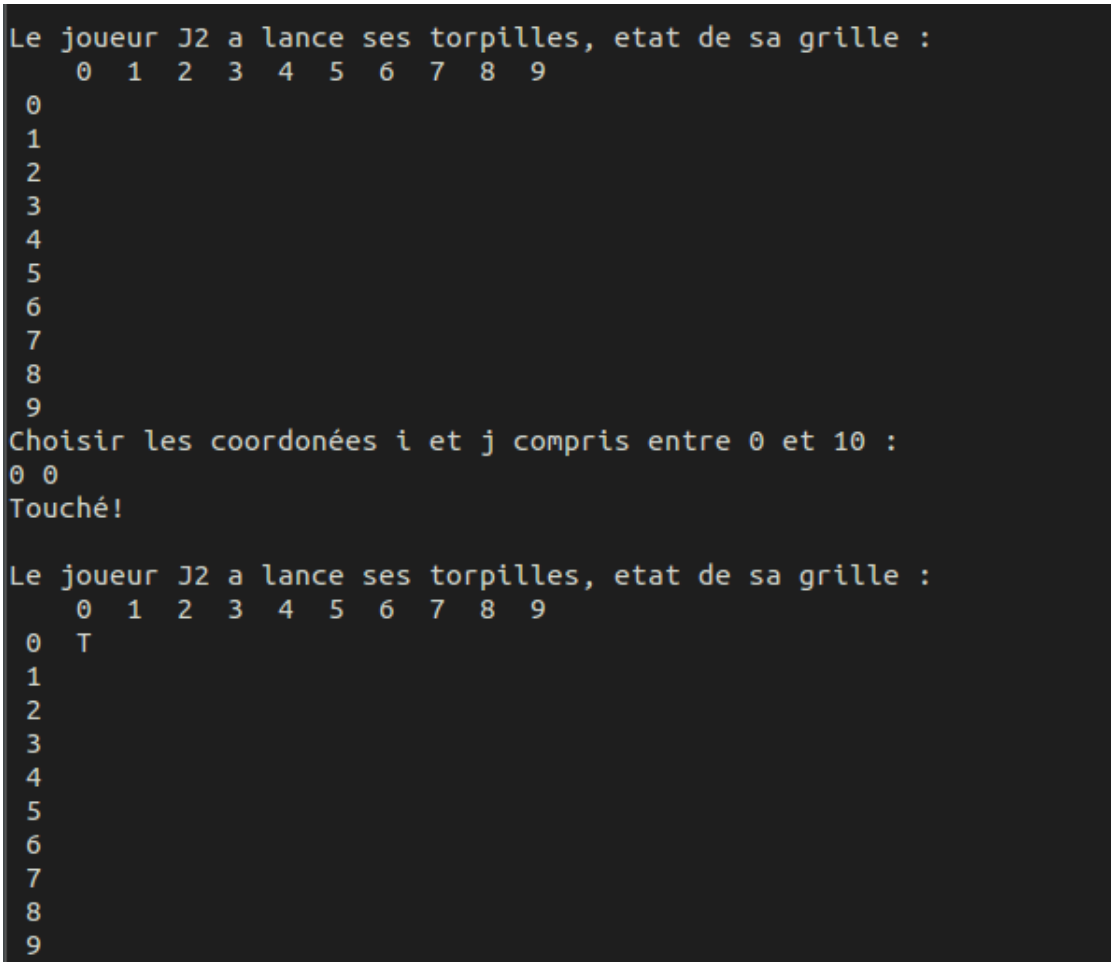


Figure 1: Touché!

4.3.2 Coulé

```
Choisir les coordonées i et j compris entre 0 et 10 :
0 1
Touché!
Coulé!

Le joueur J2 a lance ses torpilles, etat de sa grille :
    0  1  2  3  4  5  6  7  8  9
 0  C  C
 1
 2
 3
 4
 5
 6
 7
 8
 9
```

Figure 2: Coulé!

4.3.3 Raté

```
Choisir les coordonées i et j compris entre 0 et 10 :
1 0
Raté!

Le joueur J2 a lance ses torpilles, etat de sa grille :
    0  1  2  3  4  5  6  7  8  9
 0  C  C
 1  X
 2
 3
 4
 5
 6
 7
 8
 9
```

Figure 3: Raté!

4.3.4 Victoire

```
=> Vous avez gagné !!
=> La partie à durée 33 coups
```