Projet: Bataille Navale
Algorithmique et structures de données
Licence 2 Informatique

Julien Bernard

1 Spécifications

Le but de ce projet est d'implémenter un programme capable de jouer sans assistance à une version améliorée de la bataille navale.

1.1 Règles du jeu

La partie oppose deux joueurs et se déroule sur deux plans d'eau de taille 10×10. Les colonnes sont nommées de *A* à *J* tandis que les lignes sont numérotées de *0* à *9*. La coordonnée d'une case est donc identifiée par une lettre suivie d'un chiffre, par exemple *D7*. Chaque joueur possède un plan d'eau.

En début de partie, chaque joueur dépose cinq mines sur le plan d'eau adverse. Ces mines ne changent pas de position au cours de la partie mais sont inconnues du joueur. Puis, chaque joueur place cinq navires dont le nom et la taille sont décrits dans la table 1. Les navires ne doivent pas se chevaucher. Si le joueur place un de ses navires sur une mine, celui-ci est détruit immédiatement. Un navire détruit reste sur le plan d'eau.

Puis, la partie commence. À chaque tour de jeu, le joueur peut effectuer une des actions suivantes :

— Attaquer. Le joueur désigne alors la case où il souhaite tirer un missile. En réponse, le joueur sait s'il a touché un navire adverse ou pas.

— Sonder. Le joueur lance une sonde sur une case et, si un navire se trouve sur cette case ou une des 8 cases autour, il reçoit l'information de la case. Si plusieurs navires se trouvent dans le périmètre, alors un seul des navires est repéré.

— Déplacer. Le joueur peut déplacer un de ses navires d'une seule case dans la direction dans laquelle se trouve le navire (haut-bas si le navire

Nom	Taille
Porte-avions	5
Croiseur	4
Contre-torpilleur	3
Sous-marin	3
Torpilleur	2

Table 1 – Nom et taille des navires

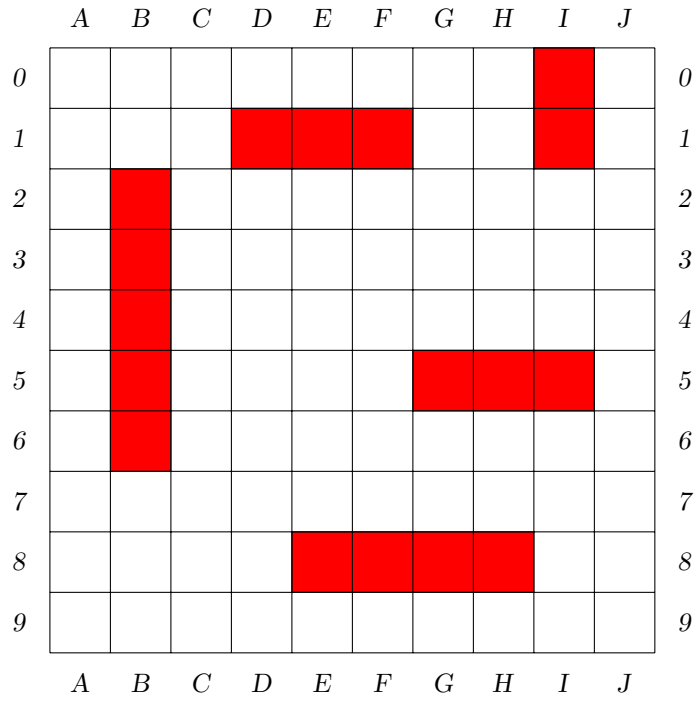

FIGURE 1 – Un plan d'eau avec les cinq navires

est vertical ou gauche-droite si le navire est horizontal). On ne peut pas déplacer un navire touché par un missile. Si un navire touche une mine après s'être déplacé, il est détruit immédiatement.

Un joueur détecte les attaques de l'adversaire mais ne détecte ni les sondages, ni les déplacements de l'adversaire. Le gagnant est celui qui a détruit tous les navires de son adversaire.

1.2 Protocole

Votre programme échange des données avec le serveur via son entrée et sa sortie standard, sous forme textuelle, ligne par ligne.

1. Au début du jeu :
   (a) Le joueur envoie les coordonnées des cinq mines, une sur chaque ligne
   (b) Le joueur envoie successivement la position de ses cinq navires sous la forme de deux coordonnées collées indiquant les extrêmités du navire, une position sur chaque ligne (par exemple, sur la figure 1, le porte-avions a pour position *B2B6* ou *B6B2*). Le serveur répond soit `OK` si le navire est conforme et n'a pas été placé sur une mine, soit `KO` si le navire est non-conforme (et la partie est perdue pour le joueur), soit `BOOM` si le navire a été placé sur une mine.
   (c) Le serveur envoie ensuite une suite de `KABOOM` (les navires adverses font un bruit différent) avec les coordonnées des mines qui ont explosé

parce que l'adversaire a posé son navire sur une mine. Ensuite, il annonce le début de la partie avec un `START`.

2. Puis à chaque tour de jeu :
   (a) Le joueur envoie une action parmi `SHOOT`, `POLL` ou `MOVE`.
   — Si le joueur envoie `SHOOT`, il indique alors la coordonnée de son tir. Le serveur lui répond soit `HIT` si le tir a touché un navire, soit `MISS` si le tir a fini dans l'eau. Si le joueur tire sur une de ses mines, celle-ci explose et `MISS` est renvoyé.
   — Si le joueur envoie `POLL`, il indique alors la coordonnée de son sondage. Le serveur lui répond soit `EMPTY` si aucun navire ne se trouve dans la zone de sondage, soit `SHIP` s'il y a un navire, suivi des coordonnées d'un des navires présent dans la zone sur une nouvelle ligne.
   — Si le joueur envoie `MOVE`, il indique alors les coordonnées d'une extrêmité d'un de ses navires. Le serveur répond soit `OK` si le déplacement a réussi, soit `BOOM` si le navire a touché une mine lors de son déplacement, soit `KO` si les coordonnées ne correspondait pas à une extrêmité d'un navire ou si le navire ne pouvait pas se déplacer ou que le déplacement était impossible. Un déplacement est prioritaire sur un tir, c'est-à-dire qu'on déplace d'abord le navire avant de regarder si un tir adverse touche ou pas.
   (b) Le serveur envoie ensuite le résultat de l'action de l'adversaire. Il envoie soit `NOTHING` si rien de visible ne s'est passé, soit `KABOOM` si un de ses navires a touché une mine, suivi des coordonnées de la mine qui a explosé, soit `MISSED` s'il a tiré sans toucher, suivi des coordonnées du tir manqué, soit `ATTACK` s'il a tiré et touché, suivi des coordonnées du navire touché.

Le squelette suivant est un exemple d'interaction avec le serveur.

```
#include <assert.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define BUFSIZE 256

int main() {
  setbuf(stdout, NULL);

  char buffer[BUFSIZE];

  // mines

  printf("C2\n");
  printf("H2\n");
  printf("C7\n");
  printf("H7\n");
  printf("E5\n");
```

```
  // ships

  printf("B2B6\n");
  fgets(buffer, BUFSIZE, stdin);
  assert(strcmp(buffer, "OK\n") == 0);

  printf("E8H8\n");
  fgets(buffer, BUFSIZE, stdin);
  assert(strcmp(buffer, "OK\n") == 0);

  printf("D1F1\n");
  fgets(buffer, BUFSIZE, stdin);
  assert(strcmp(buffer, "OK\n") == 0);

  printf("G5I5\n");
  fgets(buffer, BUFSIZE, stdin);
  assert(strcmp(buffer, "OK\n") == 0);

  printf("I0I1\n");
  fgets(buffer, BUFSIZE, stdin);
  assert(strcmp(buffer, "OK\n") == 0);

  fgets(buffer, BUFSIZE, stdin);
  assert(strcmp(buffer, "START\n") == 0);

  for (;;) {
    printf("SHOOT\n"); // or POOL or MOVE
    printf("D2\n");

    fgets(buffer, BUFSIZE, stdin);
    assert(strcmp(buffer, "MISS\n") == 0);

    fgets(buffer, BUFSIZE, stdin);
    // NOTHING or MISSED or KABOOM or ATTACK
  }

  return EXIT_SUCCESS;
}
```

2 Implémentation

Vous êtes totalement libre pour l'implémentation du programme (choix des structures, choix des fonctions, etc). Cependant, voici quelques conseils qui peuvent vous être utiles.

N'oubliez pas que vous avez deux plans d'eau à gérer, le vôtre (avec vos navires) et celui de votre adversaire (avec les mines que vous avez déposées et les navires adverses). Il vous faudra retenir des informations sur ces deux plans d'eau si vous voulez avoir une stratégie efficace.

Si vous avez tiré sur une case et qu'il n'y avait aucun navire, rien ne garantit

qu'un navire ne s'y trouvera pas par la suite puisque les navires peuvent se déplacer n'importe quand. C'est un élément important à prendre en compte.

Considérations générales et évaluation

Vous serez évalués sur le choix des structures et des algorithmes utilisés, en particulier leur complexité. Il vous est demandé de commenter votre code *abondamment* de manière à expliquer vos choix.

Un tournoi aura lieu permettant de tester vos algorithmes les uns contre les autres. Les trois premiers du tournoi recevront un point bonus sur leur projet. L'arbitre utilisé pendant le tournoi sera mis à disposition avant le tournoi par les encadrants. La date et les modalités exactes du tournoi seront précisées ultérieurement.

Le projet est à faire en binôme. Le résultat, sous forme d'une archive `tar.gz` contenant l'ensemble de vos sources et nommée à l'aide des noms des deux membres du binôme, est à rendre sur MOODLE avant la date indiquée (aucun retard autorisé).

La note du projet tiendra compte des points suivants (liste non-exhaustive) :

— la complexité optimale des algorithmes proposés ;
— la séparation du projet en plusieurs unités de compilation ;
— la présence d'un Makefile fonctionnel ;
— les commentaires dans le code source ;
— l'absence de fuites mémoire.