Projet de programmation impérative

Le but de ce projet auquel nous consacrerons trois séances de TP, est de programmer un jeu de bataille navale contre l'ordinateur.

1 Principe du jeu

Le jour dispose d'une grille de 10 x 10 cases. Les colonnes de la grille sont repérées par des lettres de 'A' à 'J' et les lignes par des nombres de 1 à 10. Chaque joueur place des bateaux dans la grille :

- Un porte-avion d'une longueur de quatre cases,
- Deux croiseurs d'une longueur de trois cases,
- Trois destroyers d'une longueur de deux cases,
- Quatre vedettes d'une longueur d'une case.

Les bateaux sont placés de manière à ne pas se toucher.

Chaque joueur, à tour de rôle, donne les coordonnées d'une case. Son adversaire dit «touché» si cette case appartient à un bateau, ou «coulé» si toute les cases de son bateau ont été touchées. Si la case ne correspond à aucun bateau le joueur dit «dans l'eau» ou «plouf».

Remarque : il est plus commode que chaque joueur dispose de deux grilles : une pour placer ses bateaux et une pour gérer ses tirs dans la grille adverses.

2 Travail à réaliser

Programmer le jeu d'un joueur contre l'ordinateur prévoyant les points suivants :

- l'ordinateur tire au hasard (stratégie 0) ou applique une stratégie (stratégies 1 et 2),
- un historique permet de revenir sur les coups précédents,
- on peut sauver et restaurer une partie dans un fichier.

2.1 Détails du programme

Affichage des grilles du joueur en mode texte. Le quadrillage est rendu par les caractères « | » et « - ». Les cases ne contenant rien sont remplies avec des espaces. Les cases des bateaux

contiennent les lettres : « P »=porte-avion, « C »=croiseur, « D »=destroyer, « V »=vedette.

A chaque tour de jeu le contenu des cases est modifié si besoin est. Les symboles pour les tirs sont : « X » touché, « o » dans l'eau.

Initialisation : la phase d'initialisation comprend :

- le placement automatique des bateaux de l'ordinateur,
- le placement des bateaux du joueur en manuel ou automatique au hasard,
- la vérification du placement par rapport aux règles du jeu,

Combat : la phase de combat gère :

- les tours de jeu,
- les tirs du joueur et de l'ordinateur,
- la vérification des coups,
- la détermination du résultat,
- le changement de joueur,
- le ré-affichage des deux grilles du joueur.

On prévoira un menu pour la saisie dans la phase d'initialisation, et la phase de combat avec une possibilité d'abandon, de sauvegarde, d'annulation des *n* derniers coups, de triche par *cheat* code permettant de connaître la position d'un ou plusieurs bateaux de l'ordinateur.

2.2 Stratégies avancées de l'ordinateur

2.2.1 Stratégie 1

Dans cette stratégie, l'ordinateur dès qu'il a touché un bateau adverse, cherche à suivre le bateau afin de le couler. Il y a une phase de détermination de l'orientation.

2.2.2 Stratégie 2

Vous êtes libre dans ce niveau de chercher une stratégie plus évoluée, comme par exemple déterminer les zones de la grille ou il y a eu peu de tirs pour les privilégier dans les coups suivants. Ces stratégies devront êtres documentées.

2.3 Règles de codage

Le programme devra être modulaire. Vous devez identifier les différentes parties du programme (modules, sous-module, etc.) et leur relations. Vous écrirez avant tout codage, les spécifications informelles (langage naturel, schémas) de chaque module et identifierez leurs relations avec les autres modules. Chaque partie (module ou constituant de module) correspondra à au moins un fichier de code C et à un header (.h). Les fichiers commenceront par une entête en commentaire indiquant le nom et le rôle du module, le programmeur, la date de création. Le code devra être commenté. Chaque sous programme (fonctions et procédures) commencera par des lignes de commentaires précisant :

- pour chaque paramètre s'il est en entrée, en sortie ou en entrée/sortie ainsi que leur domaine et le mode de passage (copie, référence).

- S'il y a un effet de bord comme l'accès ou la modification d'une variable extérieure (grilles par exemple),
- L'objet du sous programme et les spécifications si elles sont demandée.

Il est demandé de faire des tests dits unitaires pour chaque sous programme, et des tests dits d'intégration pour chaque partie de code appelant des sous parties.

Le programme complet doit être compilé à l'aide d'un Makefile.

2.4 Organisation des séances de TP

Semaine 1 : Structure de données, affichage grilles, modules, initialisation.

Semaine 2 : menus, phase de combat, jeu de l'ordinateur au hasard.

Semaine 3 : stratégies, triche, sauvegarde.

2.5 A rendre

L'ensemble du code C (.c, .h), les spécifications informelles, le *makefile.*

3 Notation

Le notation se fait en deux étapes :

3.1 Catégories de programmes

Le programme que vous allez rendre sera classé dans l'une des catégories ci-dessous et sera noté par rapport à la note maximale associée. Vous devez à la validation du projet indiquer dans quelle catégorie vous vous placez.

Attention : Pour obtenir les points relatifs à une version, il faut que les versions inférieures aient été traitées dans leur intégralité.

Version basique : 12 points maximum

- L'ordinateur tire au hasard (stratégie 0),
- Le placement des bateaux du joueur est manuel, celui des bateaux de l'ordinateur est au hasard,
- Pas d'historique ni de sauvegarde,
- La fonction *retrouveBateau* est spécifiée et prouvée à l'aide de la technique vue en cours (pfp). Le prédicar de sortie de la fonction *appartienBateau(I,J,Type,Bateau)* est complété.
- Pas de makefile.

Version intermédiaire : 16 points maximum

- Le placement des bateaux du joueur est au choix manuel ou automatique, celui des bateaux de l'ordinateur est au hasard,
- L'ordinateur applique la stratégie 1,
- Il y a une sauvegarde et une restauration.

Version « top black belt» : 20 points maximum

- Les critères de la version intermédiaire,
- Historique avec annulation des *n* derniers coups et mode triche : *cheat codes*
- Makefile.

Version « topissime kungfu master» : pour la gloire

- Une super stratégie de type 2, une interface graphique, etc.

3.2 Détermination de la note finale

Les trois principaux points qui guideront la notation dans chaque catégorie sont :

Point 1 : 60% de la note finale

- compilation, exécution : le programme compile et s'exécute sans *bugs*.

Point 2 : 25 % de la note finale

1. qualité du code :
   - pertinence des type utilisés,
   - efficacité des algorithmes,
   - simplicité, efficacité du code C,
   - lisibilité du code : indentation, choix du nom des identificateurs,
2. respect des règles de codage,
3. pertinence des commentaires

Point 3 : 15% de la note finale

1. modularité : pertinence du découpage en modules et sous programmes,
2. pertinence des .h,
3. compilation séparée, *makefile* : automatisation de la compilation.

4 Annexe

Indication sur une structure de données possible :

Chaque joueur aura deux tableaux NxN de caractères où N est défini comme 10. Les caractères sont ' ' (espace), 'P', 'C', 'D', 'V' et 'X' pour la grille des bateaux et ' ', 'P', 'C', 'D', 'V', 'o' pour la grille de tir.

Pour la gestion des bateaux on utilise une structure **s_bateau** :

```
char typeBateau ∈ {'P', 'C', 'D', 'V' }
int NbCases          /* nombre de cases du bateau */
int absOrig          /* absice  de la première case du bateau */
int ordOrig          /* ordonnée de la première case du bateau */
char Orientation ∈ {'N', 'S', 'W', 'E'}
```

et une structure **s_listeBateau** :

```
int NbAFlot                      /* nombre de bateaux encore à flot */
s_bateau tabBateaux[NbBateaux]   /* tableau des bateaux */
```

Tout est en double pour chaque joueur : une structure pour les bateaux du joueur, une pour les bateaux adverses.

bool appartientBateau(int i,int j, char Type, s_bateau bateau)
teste si la case passée en paramètre appartient au bateau passée en paramètre.

PE={0<=I,J ,AbsOrig, OrdOrig<N ^ Type ∈ {'P','C','D','V'}}
b=appartienBateau(I,J,Type,Bateau)
PS={b=true→....}

int retrouveBateau(int i,int j, char typeBateau, s_listeBateaux listeBateaux)
parcourt la liste des bateaux pour trouver le bateau touché.

PE = {0<=I,J<N ^ TypeBateau ∈ {'P','C','V','D'} }
indiceBateau=retrouveBateau(I,J,TypeBateau,ListeBateaux)
PS={1<=indiceBateau<=NbBateaux
^ appartientBateau(i,j, typeBateau ,listeBateaux.tabBateaux[indiceBateau])=true}

MAJlisteBateaux(s_listeBateaux listeBateau, int absTir,int ordTir, char typeBateau)
met à jour la structure des bateaux

Séquence des opérations pour le joueur A :

A tire :

Actions de A :

si dans l'eau maj grille de tir de A
si touché maj grille de tir A et maj liste bateau de A pour B

Actions de B :

si dans l'eau maj grille des bateaux (si on veut garder la trace des tirs adverses)
si touché maj grille bateau et récupération du type et des coordonnées
maj structure bateau

teste si fin du jeu.

La séquence des opérations pour le joueur B est symétrique.