Compte rendu bataille navale

Ce programme est une bataille navale : c'est un jeu qui se joue à deux joueurs (dans notre cas l'utilisateur contre un ordinateur), chaque joueur place 5 pions sur une grille contenant 5 lignes et 5 colonnes. Pour des raisons de simplicité, un pion a une taille d'une case. L'objectif est de découvrir où l'adversaire a placé ses pions, le premier joueur à avoir découvert tous les pions de l'adversaire remporte la partie.
Nous allons développer ce programme en C++.

Le programme est découpé en 4 parties :

- La déclaration des fonctions (qui vont permettre les procédures d'affichage)
- La déclaration et l'initialisation des variables
- Le cœur du programme (avec toute la partie interaction avec l'utilisateur)
- Le message final

1. Déclaration des fonctions (procédures d'affichage)

On a besoin de créer 2 fonctions :

- Une pour l'affichage du placement des pions de l'utilisateur au début de la partie
- Une autre, quasiment identique à la première qui va afficher le déroulement de la partie de l'utilisateur et de l'ordinateur (c'est-à-dire l'affichage des pions découverts, les cases cachées et les cases vides).

Première fonction :

```
//Déclaration de la fonction qui va permettre la procédure d'affichage des pions placés par le joueur
void afficheTab(int tab[5][5]) {
    for (int ligne=0;ligne<5;ligne++) {
        if (ligne==0) {
            cout<<"  1 2 3 4 5"<<endl;
        }
        for (int col=0;col<5;col++) {
            if (col==0) {
                cout<<ligne+1<<" ";
            }
            cout<<(tab[ligne][col]==0?'~':'o')<<" ";
        }
        cout<<endl;
    }
}
```

On prend soin de bien afficher le numéro de la ligne et le numéro de la colonne (étapes encadrées en rouges)

Les cases vides sont représentées par « ~ » et les cases avec un pion sont représentées par « o » (étape encadrée en bleu)

Deuxième fonction :

```
//Déclaration de la fonction qui va permettre la procédure d'affichage des pions présents sur le plateau
void afficheTabCache(int tab[5][5]) {
    for (int ligne=0;ligne<5;ligne++) {
        if (ligne==0) {
            cout<<"  1 2 3 4 5"<<endl;
        }
        for (int col=0;col<5;col++) {
            if (col==0) {
                cout<<ligne+1<<" ";
            }
            cout<<(tab[ligne][col]==0?'?':'\0')<<(tab[ligne][col]==1?'?':'\0')<<(tab[ligne][col]==2?'o':'\0')<<(tab[ligne][col]==3?'x':'\0')<<" ";
        }
        cout<<endl;
    }
}
```

Cette fonction est quasiment copiée collée de la première, le seul changement est au niveau de l'affichage de l'état de la case : « ? » si la case est encore inconnue, « o » si la case contient un pion, « x » si la case est vide et découverte (étape entourée en bleu).

2. Déclaration et initialisation des variables

Ci-dessous, l'ensemble des variables et des tableaux que nous allons utiliser dans le programme :

```
//Déclaration des variables
int tabjoueur[5][5], tabordi[5][5], initialisationColonne=0, initialisationLigne=0, souhaitLigne=0, souhaitColonne=0, pion=0, nbPionTrouveJoueur=0,
    nbPionTrouveOrdi=0;
char rejouer;
```

tabJoueur : tableau du jeu de l'utilisateur
tabOrdi : tableau du jeu de l'ordinateur
initialisationColonne : variable qui va uniquement nous servir pour l'initialisation des colonnes des tableaux
initialisationLigne : variable qui va uniquement nous servir pour l'initialisation des lignes des tableaux
souhaitLigne : variable de saisie (position de la ligne où le joueur souhaite placer ou découvrir un pion)
souhaitColonne : : variable de saisie (position de la colonne où le joueur souhaite placer ou découvrir un pion)
pion : compteur pour la boucle « for » lors du positionnement des pions par les joueurs
nbPionTrouveJoueur : nombre de pions découverts par l'utilisateur
nbPionTrouveOrdi : nombre de pions découverts par l'ordinateur
rejoueur : variable de saisie pour demander à l'utilisateur s'il souhaite rejouer

On en profite également pour initialiser « srand », qui va nous servir pour tous les tirages aléatoires (positionnement des pions de l'ordinateur et choix de la case à découvrir par l'ordinateur) :

```
//Initialisation de srand
srand((unsigned int)time(NULL));
```

3. Cœur du programme (interaction avec l'utilisateur)

Il faut initialiser les tableaux et les variables **nbPionTrouveJoueur** et **nbPionTrouveOrdi** au début du programme au cas où l'utilisateur joue plusieurs parties, de manière à ce que ces variables soient remises à 0 à chaque début de partie :

```
//Tant que l'utilisateur souhaite rejouer
do {
    //Initialisation des tableaux (il faut obligatoirement les initialiser ici au cas ou l'utilisateur joue plusieurs parties)
    for (initialisationLigne=0 ; initialisationLigne<5 ; initialisationLigne++) {
        for (initialisationColonne=0 ; initialisationColonne<5 ; initialisationColonne++) {
            tabjoueur[initialisationLigne][initialisationColonne]=0;
            tabordi[initialisationLigne][initialisationColonne]=0;
        }
    }

    //Initialisation des variables de comptage de nombre de point gagnés (il faut obligatoirement les initialiser ici au cas ou l'utilisateur joue plusieurs
        parties)
    nbPionTrouveOrdi=0;
    nbPionTrouveJoueur=0;
```

Les joueurs positionnent leurs pions (à droite l'utilisateur et à gauche l'ordinateur) :

```
//Le joueur place ses 5 pions
for (pion=0 ; pion<=4 ; pion++) {
    //Tant que la case n'est pas valide
    do {
        //Tant que la ligne n'est pas valide
        do {
            //L'utilisateur choisit la ligne ou sera positionné son pion
            cout<<"A quelle ligne souhaitez vous placer votre pion ? : ";
            cin>>souhaitLigne;
            //On s'assure que le pion est dans les limites du tableau
            if (!(souhaitLigne>0 & souhaitLigne<6)) {
                cout<<"Erreur de saisie, vous êtes hors des limites du jeu. ";
            }
        }while(!(souhaitLigne>0 & souhaitLigne<6));
        //Tant que la colonne n'est pas valide
        do {
            //L'utilisateur choisit la colonne ou sera positionné son pion
            cout<<"Sur quelle colonne souhaitez vous placer votre pion ? : ";
            cin>>souhaitColonne;
            //On s'assure que le pion est dans les limites du tableau
            if (!(souhaitColonne>0 & souhaitColonne<6)) {
                cout<<"Erreur de saisie, vous êtes hors des limites du jeu. ";
            }
        }while(!(souhaitColonne>0 & souhaitColonne<6));
        //On vérifie que la case n'est pas déjà attribué à un pion
        if (tabjoueur[souhaitLigne-1][souhaitColonne-1]==1) {
            cout<<"Cette case est déjà prise par un pion. ";
        }
    }while(tabjoueur[souhaitLigne-1][souhaitColonne-1]==1);
    //On place le pion dans le tableau du joueur
    tabjoueur[souhaitLigne-1][souhaitColonne-1]=1;
}
```

```
//L'ordinateur place ses 5 pions
for (pion=0 ; pion<=4 ; pion++) {
    do {
        //Génération aléatoire de la ligne à laquelle sera placé le pion
        souhaitLigne=(rand()%5)+1;
        //Génération aléatoire de la colonne à laquelle sera placé le pion
        souhaitColonne=(rand()%5)+1;
    }while(tabordi[souhaitLigne-1][souhaitColonne-1]==1);
    //On place le pion dans le tableau de l'ordinateur
    tabordi[souhaitLigne-1][souhaitColonne-1]=1;
}
```

On note que le positionnement des pions est effectué manuellement par l'utilisateur alors qu'il s'effectue aléatoirement avec « srand » pour l'ordinateur.

On affiche le positionnement des pions de l'utilisateur :

```
//on affiche la grille du joueur
cout<<endl<<"Voici le positionnement de vos pions :"<<endl<<endl;
afficheTab(tabjoueur);
cout<<endl;
```

On entre maintenant au cœur du jeu. A partir de maintenant, on entre dans une boucle « do while » qu'on quittera lorsqu'un des joueurs aura découvert les 5 pions de son adversaire. L'utilisateur doit découvrir les pions placés par l'ordinateur et inversement.
Pour cela, l'utilisateur saisit la ligne où il souhaite effectuer son tir, puis la colonne. Quant à l'ordinateur, il choisit aléatoirement la ligne et la colonne où il tire. On prend soin de faire des contrôles de saisie afin de s'assurer que la saisie de l'utilisateur est cohérente par rapport au jeu.

```
//Tant que la ligne n'est pas valide
do {
    //L'utilisateur choisit la ligne ou il souhaite tirer
    cout<<"A vous de jouer !"<<endl<<"A quelle ligne souhaitez vous découvrir une case ? : ";
    cin>>souhaitLigne;
    //Controle de saisie : on vérifie que la saisie est dans les limites du jeu
    if (!(souhaitLigne>0 & souhaitLigne<6)) {
        cout<<"Erreur de saisie, vous êtes hors des limites du jeu. ";
    }
}while(!(souhaitLigne>0 & souhaitLigne<6));
//Tant que la colonne n'est pas valide
do {
    //L'utilisateur choisit la colonne ou il souhaite tirer
    cout<<"A quelle colonne souhaitez vous découvrir une case ? : ";
    cin>>souhaitColonne;
    //Controle de saisie :on vérifie que la saisie est dans les limites du jeu
    if (!(souhaitColonne>0 & souhaitColonne<6)) {
        cout<<"Erreur de saisie, vous êtes hors des limites du jeu. ";
    }
}while(!(souhaitColonne>0 & souhaitColonne<6));
```

Ci-dessus, l'utilisateur choisit la case ou il souhaite tirer.

```
//On avertit l'utilisateur que c'est au tour de l'ordinateur
cout<<"C'est au tour de l'ordinateur"<<endl<<"Tir en cours"<<endl;
```

Ci-dessus, l'ordinateur choisit la case ou il souhaite tirer.

Après le choix des joueurs, on effectue une pause de 2 secondes dans le programme pour « simuler le tir » :

```
//Pause de 2 secondes pour simuler le tir
cout<<"Tir en cours..."<<endl;
sleep(2);
```

Après cette pause, on s'assure évidemment d'afficher le résultat du tir de l'utilisateur dans un premier temps, puis celui de l'ordinateur. A droite, on a l'affichage du jeu de l'ordinateur et à gauche, celui de l'utilisateur.

```
//Affichage de l'action résultante du choix de l'utilisateur
switch (tabordi[souhaitLigne-1][souhaitColonne-1]) {
    case 0:
        cout<<"Raté !"<<endl<<endl;
        tabordi[souhaitLigne-1][souhaitColonne-1]=3;
        break;
    case 1:
        cout<<"Touché !"<<endl<<endl;
        tabordi[souhaitLigne-1][souhaitColonne-1]=2;
        nbPionTrouveJoueur++;
        break;
    default:
        cout<<"Tir à blanc !"<<endl<<endl;
}

//Affichage du jeu de l'ordinateur
afficheTabCache(tabordi);
cout<<endl;
```

```
//Affichage de l'action résultante du tirage aléatoire de l'ordinateur
switch (tabjoueur[souhaitLigne-1][souhaitColonne-1]) {
    case 0:
        cout<<"Raté !"<<endl<<endl;
        tabjoueur[souhaitLigne-1][souhaitColonne-1]=3;
        break;
    case 1:
        cout<<"Touché !"<<endl<<endl;
        tabjoueur[souhaitLigne-1][souhaitColonne-1]=2;
        nbPionTrouveOrdi++;
        break;
}

//Affichage du jeu de l'utilisateur
afficheTabCache(tabjoueur);
cout<<endl;
```

On note deux choses :

- Seul l'utilisateur peut faire un tir à blanc (l'ordinateur est programmé pour ne jamais en faire)
- On utilise notre fonction créée précédemment pour afficher les jeux des joueurs

4. Message final

On peut considérer cette partie comme « la fin du programme ». On affiche ici le résultat de la partie à l'utilisateur (s'il a gagné ou perdu la partie). On lui demande également s'il souhaite rejouer.

```
//Message de fin de partie et affichage du gagnant de la partie
cout<<"FIN DE LA PARTIE !!!"<<endl;
//Si l'utilisateur a gagné
if (nbPionTrouveJoueur==5) {
    cout<<"VOUS AVEZ GAGNE !!! "<<endl;
}
//Si l'ordinateur a gagné
if (nbPionTrouveOrdi==5) {
    cout<<"Vous avez perdu... "<<endl;
}
```

Ci-dessus, L'affichage du résultat de la partie

```
//On demande à l'utilisateur s'il souhaite rejouer
do {
    cout<<"Souhaitez vous rejouer ? ([O]ui/[N]on) : ";
    cin>>rejouer;
    rejouer=toupper(rejouer);
    //Controle de saisie : on s'assure que l'utilisateur a répondu "oui" ou "non"
    if (!(rejouer=='O' || rejouer=='N')) {
        cout<<"Erreur de saisie. ";
    }
}while(!(rejouer=='O' || rejouer=='N'));
```

Ci-dessus, on demande à l'utilisateur s'il souhaite rejouer, sans oublier le contrôle de saisie pour s'assurer de la cohérence de la saisie de l'utilisateur