Partie PC

10 questions, 15 points. Si vous bloquez sur une question, vous pouvez me demander la réponse. Dans ce cas, la question sera comptée 0 mais vous ne serez pas bloqué. Pensez-y !

Je fais du C++ donc je sais coder un p'tit jeu !

1995, votre BTS en poche (*allez, un peu d'imagination !*), vous êtes recruté par "Ensemble Studios", une toute nouvelle entreprise à fort potentiel de croissance (une startup si vous préférez). Votre mission va être de coder la base de leur nouveau jeu de guerre "Age of Empire".

- "Pfffff ! Un jeu de guerre ! Quelle mauvaise idée ! En plus, avec un nom pareil, jamais ils le vendront leur jeu..." vous dîtes-vous.

Malgré votre réticence, vous acceptez la mission. Il est vrai qu'en ce temps-là, le boulot manque... Voici ce qu'il vous est demandé:

Le jeu mettra en scène des armées prêtes à tout pour conquérir le monde. Une armée sera composée de guerriers. Vous devrez d'abord définir ce qu'est un guerrier. Pour cela vous créerez une structure "guerrier" qui contiendra les champs suivants :

- "indice" : **Un entier** permettant d'identifier un guerrier.
- "ptsVie" : Représente les points de vie du guerrier sous la forme d'**un entier**.
- "ptsAtt" : Représente les points d'attaque du guerrier sous la forme d'**un entier**.
- "ptsDef" : Représente les points de défense du guerrier sous la forme d'**un entier**.

1. Ecrire la structure "guerrier" correspondant à l'énoncé ci-dessus.

Nous voulons maintenant représenter une armée. Une armée aura les champs suivants :

- "nomArmee" : Le nom de l'armée sous la forme d'**un tableau de 50 caractères**.
- "nbGuerriers" : Représente le nombre de guerriers composant l'armée sous la forme d'**un entier**.
- "guerriers" : La liste des guerriers composant l'armée. Ce champ sera de la forme **pointeur vers guerrier**.

2. Ecrire la structure "armee" correspondant à l'énoncé ci-dessus.

Maintenant que vous avez les objets principaux de votre jeu, vous allez coder la partie fonctionnelle.

Dans un jeu de rôle réel, les différentes valeurs attribuées aux personnages (points de vie, de force ou de défense) le sont grâce au lancer de dés.

Notre première fonction ("lancerDe") aura pour but de simuler un ou plusieurs lancers de dé. Dans tout le TP, nous ne parlerons que de dés à 6 faces. **Cette fonction prendra en paramètre un entier** représentant **le nombre de fois que l'on veut lancer un dé. Elle retournera un entier** représentant le total de tous les lancers effectués. Son prototype est le suivant :

"int lancerDe(int nbLancers);"

Pour générer un nombre aléatoirement (et donc faire comme si vous lanciez un dé), vous utiliserez la fonction rand(). Cette fonction rand() retourne un entier de façon aléatoire. Comme nous ne voulons que des nombres compris entre 1 et 6 (comme pour un dé), nous allons devoir appliquer un modulo 6 au retour de la fonction rand().

Exemple :

```
int nombre;
nombre = (rand()%6);
```

Grâce à l'instruction ci-dessus, la variable "nombre" aura une valeur comprise entre 0 et 5 (cette valeur est générée au hasard par votre programme). Il suffit alors d'ajouter 1 pour avoir une valeur comprise entre 1 et 6. Nous obtenons finalement :

```
nombre = (rand()%6) + 1;
```

3. Ecrire la fonction "lancerDe" qui simule un ou plusieurs lancers de dé (en fonction du paramètre "nbLancers") et qui retourne le résultat. Si le paramètre "nbLancers" est inférieur ou égal à 0 alors la fonction retournera 0.

Bien, vous avez maintenant tous vos objets (guerrier, armée et dés) pour jouer à votre jeu !

La première chose que ~~nous allons~~ vous allez faire est de coder une fonction permettant de créer un guerrier. Vous appellerez cette fonction "creerGuerrier". Cette fonction prendra un paramètre "i" de type entier et retournera un guerrier.

Cette fonction devra :

i. Créer une variable "g" de type guerrier (le but de la fonction va être maintenant de remplir tous les champs de cette variable "g").
ii. Remplir le champ "indice" : Ce champ prendra la valeur du paramètre "i".
iii. Remplir le champ "ptsVie" : Pour cela, vous simulerez 10 lancers de dé (en faisant une fois appel à la fonction lancerDe).
iv. Remplir le champ "ptsAtt" : Pour cela, vous simulerez 5 lancers de dé (en faisant une fois appel à la fonction lancerDe).
v. Remplir le champ "ptsDef" : Pour cela, vous simulerez 5 lancers de dé (en faisant une fois appel à la fonction lancerDe).
vi. Retourner "g".

4. Ecrire la fonction "creerGuerrier" correspondant à l'énoncé ci-dessus et ayant le prototype suivant: "guerrier creerGuerrier(int i);".

Maintenant que vous savez créer des guerriers, vous êtes capable de créer une armée ! Vous l'aurez compris, la prochaine fonction que vous devrez coder est la fonction "creerArmee".

Cette fonction devra :

i. Créer une variable "a" de type "armee" (le but de la fonction va être maintenant de remplir tous les champs de cette variable "a").
ii. Demander à l'utilisateur de saisir un nom pour cette armée. Vous remplirez le champ "nomArmee" de "a" avec ce qu'il aura saisi.
iii. Remplir le champ "nbGuerriers" de "a" de façon aléatoire. Vous devrez appeler la fonction lancerDe() afin de simuler **30 lancers** de dé (un seul appel à la fonction lancerDe).
iv. Une fois que le champ "nbGuerrier" est fixé, vous allez remplir le champ "guerriers" de "a". Ce champ "guerriers" est de type pointeur vers guerrier. Il peut donc être considéré comme un tableau de guerrier. Grâce à l'allocation dynamique (new), vous ferez en sorte que ce champ "guerriers" soit un tableau de "nbGuerrier" cases. Chaque case étant un "guerrier".

v. Ensuite, vous remplirez ce tableau "guerriers" en appelant, pour chaque case du tableau, la fonction "creerGuerrier". En effet, chaque élément du tableau "guerriers" prendra la valeur de retour de la fonction "creerGuerrier".

vi. Enfin, vous retournerez "a".

5. Ecrire la fonction "creerArmee" correspondant à l'énoncé ci-dessus et ayant le prototype suivant:

"armee creerArmee();"

Afin d'avoir un aperçu rapide de l'état de votre armée, vous allez coder les fonctions suivantes :

i. La fonction "nbGuerriersEnBonneSante" retournera le nombre de guerriers ayant leurs points de vie (champ "ptsVie") supérieurs ou égaux à 40. Cette fonction aura le prototype suivant

"int nbGuerriersEnBonneSante(armee a);"

ii. La fonction "nbGuerriersEnDanger" retournera le nombre de guerriers ayant leurs points de vie (champ "ptsVie") compris entre 15 (inclus) et 1 (inclus). Cette fonction aura le prototype suivant

"int nbGuerriersEnDanger(armee a);"

iii. La fonction "nbGuerriersMorts" retournera le nombre de guerriers ayant leurs points de vie (champ "ptsVie") inférieurs ou égaux à 0. Cette fonction aura le prototype suivant

"int nbGuerriersMorts(armee a);"

iv. La fonction "nbGuerriersSanteCorrecte" retournera le nombre de guerriers n'étant ni en bonne santé, ni en danger, ni morts. Pour calculer ce nombre, **vous utiliserez les 3 fonctions codées précédemment** ("nbGuerriersEnBonneSante", "nbGuerriersEnDanger" et "nbGuerriersMorts") **ainsi que le champ "nbGuerriers"**. Cette fonction aura le prototype suivant : "int nbGuerriersSanteCorrecte(armee* a);". **Attention ! comme vous pouvez le constater, cette fonction prend un pointeur vers une armée en paramètre !**

6. Ecrire les 4 fonctions "nbGuerriersEnBonneSante", "nbGuerriersEnDanger", "nbGuerriersMorts" et "nbGuerriersSanteCorrecte" correspondant à l'énoncé ci-dessus. **Vous devez absolument respecter les prototypes et l'énoncé**!

Il vous faut maintenant afficher ces statistiques. Vous allez coder la fonction "afficherStatsArmee" qui affichera, pour une armée passée en paramètre, le nombre de guerriers en bonne santé, le nombre de guerriers en correcte santé, le nombre de guerriers en danger et le nombre de morts. Cette fonction fera évidemment appel aux 4 fonctions "nbGuerriersEnBonneSante", "nbGuerriersSanteCorrecte", "nbGuerriersEnDanger" et "nbGuerriersMorts". L'affichage doit être de la forme :

Statistiques de l'armee <nomArmee> :
Nombre de guerriers en bonne sante : 19/114
Nombre de guerriers en sante correcte : 95/114
Nombre de guerriers en danger : 0/114
Nombre de guerriers morts : 0/114

(114 représentant le nombre total de guerriers au sein de l'armée).

7. Coder la fonction "afficherStatsArmee" ayant pour prototype :

"void afficherStatsArmee(armee a);"

Un jour ou l'autre, une armée du jeu sera vaincue et il faudra le vérifier. Pour cela, vous allez coder la fonction "estVaincue". Cette fonction aura le prototype suivant :
"int estVaincue(armee a)". Une armée est considérée comme vaincue lorsque tous ses guerriers ont leurs points de vie (champ "ptsVie") inférieurs ou égaux à 0. Dans ce cas là, votre fonction retournera -1. Dans tous les autres cas, elle retournera l'indice du premier guerrier à avoir ses points de vie strictement supérieurs à 0.
Par exemple, si les 3 premiers guerriers de votre armée (guerriers[0], guerriers[1] et guerriers[2]) ont des points de vies <=0 mais que le 4è guerrier de votre armée (guerriers[3]) a ses points de vie >0 **alors la fonction retournera 3**.

8. Coder la fonction "estVaincue" en respectant le prototype donné et l'énoncé ci-dessus.

Enfin, s'il n'y a pas de combats, il ne risque pas d'y avoir d'armée vaincue... Or, qu'est-ce qu'un combat entre deux armées ? Ce n'est ni plus ni moins qu'un enchainement de combats entre deux guerriers de chacune des deux armées. Vous suivez ? Nous allons donc commencer par coder la fonction "combatGuerriers" puis nous terminerons par la fonction "combatArmees". Oh yeah !

La fonction "combatGuerriers" opposera deux guerriers. Elle aura donc en paramètres deux pointeurs vers des guerriers. Par contre, elle ne retournera rien. Voici son prototype :
"void combatGuerrier(guerrier* g1, guerrier* g2)". Voici maintenant comment va se dérouler un combat entre 2 guerriers:

1. Pour chaque guerrier :
   - vous simulerez 2 lancers de dé (un seul appel à la fonction "lancerDe")
   - puis vous récupérerez ce résultat et l'ajouterez aux points d'attaque du guerrier.
2. Le guerrier ayant obtenu le plus grand score à l'issue de cette opération sera le guerrier attaquant (et donc l'autre guerrier sera le défenseur). En cas d'égalité, on recommence à l'étape 1.
3. On va ensuite calculer les points de dégâts que l'attaquant pourrait infliger au défenseur. Pour calculer ces points de dégâts, vous simulerez 6 lancers de dé (un seul appel à la fonction "lancerDe") puis vous retrancherez à ce résultat les points de défense du défenseur.
   - Si les points de dégâts sont >0 alors vous enlèverez ces points aux points de vie du défenseur.
   - Sinon (si les points de dégâts <=0) nous considèrerons que l'attaque a échoué et donc aucun point de vie ne sera enlevé au défenseur.
4. Et ainsi de suite (on répète tout à partir de l'étape 1), tant qu'il n'y a pas un des deux guerriers mort (ses points de vie <=0).

Donc, lorsqu'un des deux guerriers est mort (points de vie <=0), la fonction "combatGuerriers" se termine.

9. Coder la fonction "combatGuerriers" en respectant le prototype donné et l'énoncé ci-dessus.

Ouffff c'était dur hein ?! Mais ce n'est pas fini ! On termine réellement avec la fonction "combatArmees" qui est encore plus compliquée :(
Voici son prototype : "armee* combatArmees(armee* a1, armee* a2)".
Et voici comment un combat entre deux armées va se dérouler :

1. Pour chaque armée, vous appellerez la fonction estVaincue pour récupérer l'indice du premier guerrier vivant de votre armée. C'est ce guerrier qui va prendre part au combat. Tant pis pour lui !
2. Une fois que vous avez sélectionné un guerrier dans chaque armée, vous appellerez la fonction "combatGuerriers" en lui passant en paramètre ces deux guerriers sélectionnés.

3. Et ainsi de suite (on répète tout à partir de l'étape 1), tant qu'il n'y a pas une des deux armée vaincue (fonction "estVaincue" retourne -1).
4. Lorsqu'une des deux armées est vaincue, la fonction "combatArmees" retournera l'adresse de l'armée vainqueur.

10. Coder la fonction "combatArmees " en respectant le prototype donné et l'énoncé ci-dessus.

Vous pouvez aller annoncer à votre chef de projet que vous en avez terminé. Afin de lui prouver que tout marche, vous allez quand même lui faire une démonstration : Vous allez donc créer coder la fonction "main" qui :

- Créera 2 variables de type "armee",
- initialisera chacune des deux variables par l'appel à la fonction "creerArmee",
- appellera, pour chacune des 2 variables, la fonction "afficherStatsArmee",
- appellera la fonction "combatArmees",
- affichera le message à l'écran :

Suite au combat entre l'armee <nomArmee1> et l'armee <nomArmee2>, le vainqueur est : <nomArmee>.

- appellera une nouvelle fois, pour chacune des 2 variables, la fonction "afficherStatsArmee".

- "Wahouah !!!! Il est trop cool ton jeu fiston !" s'exclame votre chef de projet en voyant cela.
- "Pffff... Y a pas de quoi s'extasier. Un enfant aurait pu le faire. Surtout qu'il marchera jamais ton jeu..."

Quelques mois plus tard...

"Age of Empires" connait dès sa sortie un important succès commercial en restant plusieurs mois en première place des ventes des jeux de stratégie aux Etats-Unis, au Royaume-Uni, en France et en Allemagne. Le jeu s'était déjà vendu à plus 850000 exemplaires le 5 février 1998 et dépasse les trois millions de titres vendus moins de trois ans après sa sortie. Le jeu fut également récompensé à de nombreuses reprises incluant le titre de « jeu de l'année » décerné par le site Gamecenter en 1997 et le titre de « jeu de stratégie de l'année » attribuée par l'Academy of Interactive Arts and Sciences en 1998...

Fin