TP Introduction POO en C++

Sylvain Gault

18 mars 2024

1 Introduction

Ce TP est à réaliser par trois en C++ sous l'environnement de développement de votre choix. Sous Linux de préférence. Il sera à rendre le Vendredi 22/03/2024 à 13h37 UTC. Il sera à rendre de préférence sur Teams dans les fichiers patagés dans le répertoire *« TP intro objet »* ou par mail à l'adresse contact@sylvaingault.fr. Vous nommerez vos fichiers du noms des membres du groupe.

Vous rendrez un rapport en **PDF** contenant au moins les réponses aux questions ainsi que toute description nécessaire à la reproduction de vos résultats. **Numérotez les questions** auxquelles vous répondez dans votre rapport et incluez au moins une phrase pour chaque question. N'oubliez pas de lister **les noms** de tous les membres de votre groupe.

N'hésitez pas à googler la documentation en cas de doute.

2 Programmes inutiles

Exercice 1 Déclaration et usage

Le but de cet exercice est de tester les différents éléments de syntaxe de C++ concernant les classes.

1. Écrivez une classe `Point` qui a deux coordonnées flottantes `x` et `y`.
2. Écrivez une fonction `main` qui instancie un point.
3. Compilez et testez votre code.
4. Rajoutez une méthode `print` à votre classe qui affiche les coordonnées.
5. Assurez-vous de pouvoir appeler la méthode depuis la fonction `main`.
6. Assurez-vous que les attributs ne sont pas accessibles depuis le `main`.
7. Rajoutez un constructeur par copie.
8. Essayez de compiler votre code sans essayer d'utiliser le nouveau constructeur. Que se passe-t-il ?
9. Rajoutez un constructeur par défaut.
10. Vérifiez que vous pouvez utilisez vos deux nouveaux constructeurs.
11. Rajoutez un constructeur avec 2 arguments.
12. Vérifiez que vous pouvez l'utiliser.

13. Rajoutez une méthode `scale` qui prend un nombre flottant en argument et multiplie les deux coordonnées par cette valeur.
14. Testez votre code.

Exercice 2 Nombres complexe

Le but de cet exercice est d'implémenter une classe de nombres complexes dont on changera l'implémentation en cours de route.

1. Créez une classe `Complex` qui représente un nombre complexe avec deux attributs flottants : la partie réelle et la partie imaginaire.
2. Mettez un constructeur par défaut et un constructeur par copie.
3. Rajoutez deux accesseurs et deux mutateurs `get_re` pour récupérer la partie réelle, `get_im` pour récupérer la partie imaginaire. Et idem avec `set` pour définir ces valeurs.
4. Vérifiez que vous pouvez utiliser ces 4 méthodes ainsi que les différents constructeurs.
5. Rajoutez une méthode `get_mod` qui renvoie le module du nombre complexe. C'est à dire sa distance à l'origine. Demandez gentiment la formule à l'enseignant si besoin.
6. Rajoutez une méthode `get_arg` qui récupère l'argument du nombre complexe. C'est à dire l'angle.
7. Testez vos deux nouvelles méthodes sur différents nombres complexes.
8. Rajoutez une méthode `set_mod` pour définir le module du nombre complexe.
9. Rajoutez une méthode `set_arg` pour définir l'argument du nombre complexe.
10. Testez vos deux nouvelles méthodes.
11. Vous décidez de changer votre représentation interne des nombres complexes. Au lieu de stocker les parties réelle et imaginaire, vous décidez de stocker module et argument. Changez les attributs.
12. Vos méthodes permettant de récupérer et définir le module et l'argument vont maintenant être des accesseurs et mutateurs simples. Changez-les.
13. Vos accesseurs et mutateurs pour la partie réelle et imaginaire vont maintenant devoir faire des calculs plus complexes. Mettez des méthodes vides pour le moment.
14. Définissez `get_re` et `get_im`.
15. Définissez les méthodes `set_re` et `set_im`.
16. Vérifiez que votre code fonctionne toujours.

3 Jeu d'rôle

Exercice 3 Définition de classes

Le but de cet exercice est de définir des classes pour un petit jeu de rôle.

1. Définissez une classe `Weapon` qui aura un nom et des dégâts infligés.
2. Définissez une classe `RPCharacter` qui aura un nom, un level, des points d'expérience, des points de vie, un tableau de 10 `Weapon`, un entier indiquant combien d'armes le personnage possède, et un entier indiquant l'arme actuellement utilisée.

3. Rajoutez un constructeur par défaut à la classe `Weapon` qui sera l'arme inexistante.
4. Rajoutez un constructeur avec deux arguments à votre classe `Weapon`.
5. Rajouter des accesseurs pour votre classe `Weapon`. Vous ne mettrez pas de mutateur. Une arme ne change pas de nom ni de dégâts causés.
6. Rajoutez un constructeur à votre classe `RPCharacter` avec un seul argument, le nom du personnage. Vous mettrez des valeurs sensées pour les autres attributs. C'est à dire un petit level (moins de 3), aucune arme possédée, etc.
7. Rajoutez un attribut booléen `is_dead` indiquant si le personnage est vivant ou mort.
8. Définissez une méthode `apply_damage` qui prend en argument un entier et soustrait ces dommages aux points de vie. S'ils deviennent négatifs, mettez-les à 0 et définissez `is_dead` à `true`.
9. Définissez une méthode `get_weapon` qui renvoie l'arme actuellement utilisée par le personnage. S'il n'y en a pas, instanciez un nouveau `Weapon` pour attaquer à mains nues. Cette arme infligera des dégâts égaux à la moitié du level, mais au moins 1. (level 10 = dégâts 5, level 2 = dégâts 1, level 0 = dégâts 1.)
10. Définissez une méthode `attack` dans votre classe `RPCharacter` qui prend en argument une référence sur un autre personnage et l'attaque avec l'arme courante. Utilisez la méthode `get_weapon` pour récupérer l'arme du joueur attaquant et `apply_damage` sur l'autre personnage. Chaque coup porté augmente l'expérience d'autant de points que de dommages infligés.
11. Rajoutez une méthode `store_weapon` qui prend en argument un `Weapon` et le rajoute dans l'inventaire. Vous remplacerez simplement le dernier si l'inventaire est déjà plein.
12. Rajoutez une méthode `switch_weapon` qui prend en paramètre un entier entre 0 et 9 et qui change l'arme actuellement utilisée par le personnage. Si l'emplacement de l'inventaire demandé est vide, utilisez la dernière arme de l'inventaire.
13. Rajoutez une méthode `get_weapon` avec un argument entier. Celle-ci renverra l'arme de l'inventaire à la position demandé.
14. Modifiez la première méthode `get_weapon` pour utiliser la nouvelle.
15. Rajoutez une méthode `drink_potion` qui prend en argument un entier et qui remonte le nombre de points de vie d'autant.
16. Écrivez un petit scénario en dur dans votre `main` où deux joueurs trouvent des armes, s'affrontent.

Exercice 4 *(Bonus)* Rogue-like... en texte

Le but de cet exercice est de rendre votre jeu un peu plus interactif.

1. Définissez une méthode `print_inventory` qui affiche l'inventaire.
2. Utilisez `cin` (cherchez sur internet comment l'utiliser) pour demander son nom à l'utilisateur au début du programme.
3. Définissez un scénario interactif où l'utilisateur a le choix d'avancer (c'est à dire ne rien faire), changer d'arme, ramasser un objet s'il y en a un, attaquer s'il y a un ennemie.