Programmation pour le calcul scientifique
Année : 2017-2018
Formation : L3 Ingénieurie Mathématique

TP5 : Makefile et classes

1 Makefile basique

Téléchargez les fichiers joints au TP dans un même répertoire, et dupliquez-les sous des noms différents afin de créer deux codes indépendants (bien qu'identiques) contenant chacun deux fichiers .cpp et un fichier .h. Ecrivez un makefile basique qui permet de compiler soit chaque programme séparément, soit les deux programmes d'un seul coup avec la commande `make all`.

2 Makefile avancé

Modifier le makefile de l'exercice précédent pour y intégrer les éléments vus en cours :

1. variables pour définir le compilateur et les options de compilation
2. variables réservées pour désigner la cible et les dépendances
3. variables pour traiter de manière générique les fichiers finissant par les mêmes extensions

3 Classes

1. Dans un nouveau fichier, définissez la classe `point`, qui contient comme données privées deux réels `x` et `y`.
2. Définissez une variable de la classe `point` (= une instance de la classe `point`). Essayer de modifier directement les valeurs de ces données. Que se passe-t-il ?
3. Définir les fonctions suivantes comme membres publics de la classe `point` :

   ```
   double get_x() {return x;}
    double& get_y() {return y;}
   ```

   Testez-les en utilisant tout à tour les commandes :

   ```
   A.get_y() = 1.;
   A.get_x() = 1.;
   ```

   Que constatez-vous ? Comment l'expliquez-vous ?
4. Modifiez l'une des deux fonctions afin de pouvoir modifier les valeurs des deux données de la classe `point`.
5. Définissez un `double` et affectez-lui `A.x` : que se passe-t-il ? Déclarez (temporairement : juste pour cette question) publiques les données de la classe et re-testez la question précédente.
6. Déclarez maintenant ces fonctions comme privées et essayez de recompiler : que se passe-t-il ?
7. Ecrire une fonction membre publique `affiche`, qui ne renvoie rien mais affiche les données d'une instance de la classe `point`. Testez-la pour la valider.
8. Ecrire une fonction membre publique `translation`, prend en argument deux réels v_x et v_y et renvoie un `point` qui est le résultat de la translation d'un `point` par le vecteur (v_x, v_y). Testez-la pour la valider.

9. Ecrivez une fonction publique `adresse`, qui renvoie un pointeur vers l'adresse d'une instance de la classe, en utilisant le pointeur this. Testez-la en créant un pointeur avec cette fonction et en affichant la valeur de l'adresse du `point` considéré.
10. Comment modifier les valeurs des données correspondant à cette adresse en utilisant le pointeur ainsi créé ?
11. Ecrire une fonction amie `addition` qui prend en arguments deux instances de la classe `point` et retourne comme résultat une instance qui est la somme des deux précédentes. Testez-la pour la valider.
12. Structurez votre code en trois fichiers : un .cpp contenant le programme principal, un .cpp contenant les fonctions et un .h contenant les informations nécessaires.
13. Modifiez le makefile de l'exercice précédent afin qu'il génère aussi l'exécutable du code que vous venez d'écrire.