Programmation orientée objet

LIV 2024-2025

Travaux pratiques 2

Site du cours : `https://defelice.up8.site/poo.html`
Les exercices marqués de (@) sont à faire dans un second temps.
Un fichier écrit en langage C++ se termine conventionnellement par .cpp.
Une commande de compilation est `g++ fichier_source1.cpp fichier_source2.cpp fichier_source3.cpp`.
Voici des options de cette commande.

- `-o nom_sortie` pour donner un nom au fichier de sortie (par défaut `a.out`).
- `-Wall` pour demander au compilateur d'afficher plus d'avertissements (warnings)
- `-Wextra` pour demander au compilateur d'afficher plus d'avertissements (warnings)
- `-std=c++20` pour compiler selon la norme c++20
- `-g -fsanitize=address` pour que l'exécutable puisse detecter certaines erreurs de mémoire (executable plus lent, -g pour avoir plus de précision).
- `-g` pour ajouter des informations de débugage, utile avec debug (executable plus lent).

Exemple : `g++ -Wall -Wextra fichier1.cpp -o monprogramme`

Exercice 1. *Echange*

On souhaite construire une fonction `echange` qui échange les contenus de deux variables `int`.

1. Construire la fonction `echange`. Exemple d'utilisation :

```
int a=3;
int b=2;
echange(a,b); // et pas echange(&a,&b);
// ici a==2 et b==3.
```

2. Peut-on écrire `echange(3,2)` ? Pourquoi ?

Exercice 2. *Personne*

On souhaite construire une classe `Personne` qui utilise les méthodes.

- `Personne(string nom,string prenom,int anneeNaissance)` (construit un objet `Personne` avec le nom et le prénom de naissance de la personne).
- `int getAge()` Qui renvoie l'age de la personne en 2025 (on suppose que la personne est née un 1 janvier)
- `string getProfession(void)` et aussi `void setProfession(string)`.
- `string affiche()` : renvoie une chaine qui présente la personne (le nom, prénom, l'age, la profession si existante) (ou alors vous implanter `void affiche()` qui imprime cette chaîne à l'écran.)

1. Réfléchir à la nature privé/publique des attributs de la classe et implanter la en C++.
2. Cela a t-il un sens d'ajouter à la classe une méthode `getNom()` (qui renvoie le nom d'une personne)
3. Cela a t-il un sens d'ajouter à la classe une méthode `setAnnee(...)` (qui modifie l'année de naissance d'une personne) ?

Exercice 3. *Attribut const*

Voici une partie de la classe AtrConst contenant un seul attribut constant

```
class AtrConst{
   const int atr;
   ....
};
```

Completez-la avec un constructeur et une méthode get de façon à pouvoir écrire

```
#include<cassert>
...
int main(){
   ..
   AtrConst c1{9};
   assert(c1.get()==9);
   ...
}
```

Exercice 4. *Couple*

Écrire la classe `Couple` qui représente un couple de valeurs entières. Exemple d'utilisation :

```
class Couple{....};
int main(void){
   Couple obj{4,2}; // on initialise le couple avec deux valeurs 4 et 2
   obj.p();// renvoie 4,  p pour première composante
   obj.p()=3; // oui oui c'est bien <obj.p()=3;>
   obj.p();// renvoie 3
   obj.d(); // renvoie 2, d pour second
}
```

Exercice 5. *Vecteur*

Écrire la classe `Vecteur` qui représente des vecteurs du plan.

- Utiliser le mot clé `class` au lieu de `struct` pour définir l'objet.
- Faire en sorte que le constructeur de la classe appelé sans paramètre construise le point à l'origine (0,0) et que avec un seul paramètre a construire le point (a,0) (vous ne devez écrire qu'un seul constructeur).
- Implanter l'opérateur + qui renvoie la somme de deux vecteurs.

Exemple d'utilisation

```
class Vecteur{....};
int main(void){
   Vecteur p1;
   Vecteur p6{1.3};
   Vecteur p2{5.7,6.8};
   Vecteur p3{0.1,9.4};
   Vecteur p4=p2+p3 // contient le vecteur (5.8; 16.2)
   Vecteur p5=p2+p1 // contient le vecteur (5.7;6.8)
   Vecteur p7=p3+p6 // contient le vecteur (1.4;9.4)
   p4.affiche(); // affiche :  (5.8;16.2);
}
```