UNIVERSITÉ DE PAU ET DES PAYS DE L'ADOUR | COLLÈGE STEE SCIENCES ET TECHNOLOGIES POUR L'ÉNERGIE ET L'ENVIRONNEMENT

Programmation orientée objet - TP nº 4 - Héritage multiple et templates du langage C++

Samson Pierre <samson.pierre@univ-pau.fr>

29/01/2022

Ce TP a pour objectif de vous familiariser avec l'héritage multiple et templates du langage C++. Pour chaque exercice, sauf indication contraire, aucune classe ou fonction n'est autorisée et les mêmes options de compilation que celles du cours doivent être utilisées.

Exercice nº 1

Reprenez le code de l'exercice nº 6 du TP nº 2. Les bibliothèques `library_t` souhaitent dorénavant proposer d'autres documents que des livres. Ainsi, elles veulent proposer trois sortes de documents `doc_t` : des documents audio `audio_doc_t`, textuels `textual_doc_t` et vidéo `video_doc_t`. Parmi les documents audio, il y a les cassettes audio `audio_tape_t` et les CD audio `audio_cd_t`. Parmi les documents textuels, il y a les livres `book_t` et les journaux `newspaper_t`. Parmi les documents vidéo, il y a les cassettes vidéo `video_tape_t`, les DVD `dvd_t` et les Blu-ray `blu_ray_t`. Aussi, les bibliothèques veulent proposer un format de document hybride, qui est à la fois un document audio et un document textuel : les textes audio `audio_text_t`. Le texte d'une oeuvre existante est ainsi lu oralement par une personne et enregistré sur un support audio. Créez donc une hiérarchie de classes à partir de ces informations.

Une bibliothèque est caractérisée de la même façon que dans l'exercice nº 6 du TP nº 2, sauf pour les attributs `books` et `nbooks` qui sont dorénavant `docs` (un `vector` de pointeurs vers `doc_t`) et `ndocs` (un entier). Un document est caractérisé par son nom `name` (un pointeur vers `char`) et son année `year` (un entier). Ajoutez donc ces deux attributs à la classe `doc_t`. Un document audio est caractérisé par sa durée en secondes `duration` (un entier). Ajoutez donc cet attribut à la classe `audio_doc_t`. Un document vidéo est caractérisé par sa durée en secondes `duration` (un entier) et par le fait qu'il soit en couleur ou non `color_available` (un booléen). Ajoutez donc ces deux attributs à la classe `video_doc_t`. Un livre est caractérisé de la même façon que dans l'exercice nº 6 du TP nº 2, sauf pour les attributs `name` et `year` qui sont dorénavant déplacés vers la classe `doc_t`. Un journal est caractérisé par son éditeur `publisher` (un pointeur vers `char`) et son ISSN `issn` (un entier). Ajoutez donc ces deux attributs à la classe `newspaper_t`.

Dans la classe `doc_t`, définissez des accesseurs et mutateurs pour les attributs `name` et `year`. Dans la classe `audio_doc_t`, définissez un accesseur et un mutateur pour l'attribut `duration`. Dans la classe `video_doc_t`, définissez des accesseurs et mutateurs pour les attributs `duration` et `color_available`. Dans la classe `book_t`, retirez les mutateurs pour les attributs `name` et `year`. Dans la classe `newspaper_t`, définissez des mutateurs pour les attributs `publisher` et `issn`.

Dans chaque classe, définissez un constructeur par défaut. Ce constructeur initialise l'objet courant en affectant `NULL` aux attributs de type pointeur, `0` aux attributs de type entier et `false` aux attributs de type booléen. Ce constructeur est déjà défini dans la classe `book_t` grâce au code de l'exercice nº 6 du TP nº 2. Toutefois, modifiez son corps pour ne plus initialiser les attributs `name` et `year` qui seront automatiquement initialisés par le constructeur par défaut de la classe de base. Ce constructeur est déjà défini dans la classe `library_t` grâce au code de l'exercice nº 6 du TP nº 2. Toutefois, modifiez son corps pour ne plus initialiser l'attribut `nbooks` qui n'existe plus et pour initialiser le nouvel attribut `ndocs`.

Remplacez la méthode `add_book` de la classe `library_t` par la méthode `add_doc` qui prend en paramètres `doc` (un pointeur vers `doc_t`) et qui ne retourne rien. Cette méthode ajoute `doc` aux documents de la bibliothèque en redimensionnant le `vector`. Modifiez le corps de la méthode `print` de la classe `library_t` pour qu'elle n'utilise plus les attributs `books` et `nbooks` qui n'existent plus et pour qu'elle utilise les attributs `docs` et `ndocs`. Dans la fonction principale, remplacez l'appel à la méthode `add_book` qui n'existe plus par l'appel à la méthode `add_docs`.

Ajoutez aux classes `doc_t`, `audio_doc_t`, `textual_doc_t` et `video_doc_t` la méthode virtuelle pure `print` sans paramètre et qui ne retourne rien. Dans les classes dérivées de `audio_doc_t`, `textual_doc_t` et `video_doc_t`, redéfinissez la méthode `print` sans paramètre et qui ne retourne rien. Cette méthode affiche les informations sur le document (type de l'objet suivi de parenthèses qui contiennent la valeur de chaque attribut séparée par une virgule) sans afficher une nouvelle ligne. Cette méthode est déjà redéfinie dans la classe `book_t` grâce au code de l'exercice nº 6 du TP nº 2. Toutefois, modifiez son corps pour afficher les informations de la même façon que pour les autres méthodes `print` redéfinies. Une méthode `print` dans la classe `library_t` est déjà définie grâce au code de l'exercice nº 6 du TP nº 2. Toutefois, modifiez son corps pour afficher les informations sur la bibliothèque de la même façon que pour les autres méthodes `print` définies, sauf que cette méthode affiche une nouvelle ligne.

Utilisez le mot clé `class` pour la déclaration des toutes les classes afin que tous les membres aient une visibilité privée par défaut. Rendez publiques toutes les méthodes de toutes les classes. Rendez publics tous les héritages de toutes les classes dérivées. Retirez tous les destructeurs.

Dans la fonction principale, en plus des bibliothèques et livres, créez une cassette audio, un CD audio, un journal, une cassette vidéo, un DVD, un Blu-ray et un texte audio. La cassette audio a pour nom « Killers », pour date 1981 et pour durée 2298 secondes. Le CD audio a pour nom « The Book of Souls », pour date 2015 et pour durée 5531 secondes. Le journal a pour nom « The Guardian », pour date 1821, pour éditeur « Guardian Media Group » et pour ISSN 02613077. La cassette vidéo a pour nom « Aladdin », pour date 1992, pour durée 5400 secondes et est en couleur. Le DVD a pour nom « The Lion King », pour date 1994, pour durée 5280 secondes et est en couleur. Le Blu-ray a pour nom « Zootopia », pour date 2016, pour durée 6480 secondes et est en couleur. Le texte audio a pour nom « Harry Potter and the Philosopher's Stone », pour date 1997, pour durée 18000 secondes et est en couleur. Ajoutez ces sept documents à la bibliothèque « Novel Library ».

L'héritage multiple au niveau de la classe `audio_text_t` pose des problèmes d'ambiguïté lors de l'accès aux membres. Levez ces ambiguïtés à l'aide de l'héritage virtuel. Vous ne devez pas utiliser l'opérateur de résolution de portée dans cet exercice.

Voici le résultat attendu :

```
$ ./library.out
Library ("Sciences Library", ("monday", "tuesday", "wednesday", "thursday"), (Book ("The C++
Programming Language", ("Bjarne Stroustrup"), "Addison-Wesley", 2013, 9780321563842), Book ("C++:
The Complete Reference", ("Herbert Schildt"), "McGraw-Hill", 2003, 9780070532465)))
Library ("Novel Library", ("tuesday", "wednesday", "thursday", "friday"), (Book ("Harry Potter and
the Philosopher's Stone", ("J. K. Rowling"), "Bloomsbury", 1997, 9780747532699), Book ("Harry
Potter and the Chamber of Secrets", ("J. K. Rowling"), "Bloomsbury", 1998, 9780747538493), Audio
tape ("Killers", 1981, 2298 s), Audio CD ("The Book of Souls", 2015, 5531 s), Newspaper ("The
Guardian", 1821, "Guardian Media Group", 00726591), Video tape ("Aladdin", 1992, 5400 s, color
available: 1), DVD ("The Lion King", 1994, 5280 s, color available: 1), Blu-ray ("Zootopia", 2016,
6480 s, color available: 1), Audio text ("Harry Potter and the Philosopher's Stone", 1997, 18000
s)))
$
```

Voici les fonctions autorisées pour cet exercice :

```
int printf(const char *format, ...);
```

Voici les classes autorisées pour cet exercice :

```
std::vector
```

Exercice n° 2

Reprenez le code de l'exercice précédent. Retirez la classe `library_t`. Dans la classe `doc_t`, ajoutez une méthode `println` sans paramètre et qui ne retourne rien. Cette méthode appelle la méthode `print` sur l'objet courant puis affiche une nouvelle ligne. Retirez le mot clé `virtual` sur les héritages. Dans la fonction principale, supprimez les deux bibliothèques mais conservez les documents. Sur chaque document, appelez la méthode `println`.

Levez les ambiguïtés liées à l'héritage multiple à l'aide de l'opérateur de résolution de portée. Puisque la classe `doc_t` est héritée de deux façons différentes par la classe `audio_text_t`, en passant par `audio_doc_t` et en passant par `textual_doc_t`, vous devez choisir de passer par l'un des deux chemins pour accéder aux membres posant problème. Vous ne devez pas utiliser l'héritage virtuel dans cet exercice.

Voici le résultat attendu :

```
$ ./library.out
Book ("The C++ Programming Language", ("Bjarne Stroustrup"), "Addison-Wesley", 2013, 9780321563842)
Book ("C++: The Complete Reference", ("Herbert Schildt"), "McGraw-Hill", 2003, 9780070532465)
Book ("Harry Potter and the Philosopher's Stone", ("J. K. Rowling"), "Bloomsbury", 1997,
9780747532699)
Book ("Harry Potter and the Chamber of Secrets", ("J. K. Rowling"), "Bloomsbury", 1998,
9780747538493)
Audio tape ("Killers", 1981, 2298 s)
Audio CD ("The Book of Souls", 2015, 5531 s)
Newspaper ("The Guardian", 1821, "Guardian Media Group", 00726591)
Video tape ("Aladdin", 1992, 5400 s, color available: 1)
DVD ("The Lion King", 1994, 5280 s, color available: 1)
Blu-ray ("Zootopia", 2016, 6480 s, color available: 1)
Audio text ("Harry Potter and the Philosopher's Stone", 1997, 18000 s)
$
```

Voici les fonctions autorisées pour cet exercice :

```
int printf(const char *format, ...);
```

Voici les classes autorisées pour cet exercice :

```
std::vector
```

Exercice n° 3

Reprenez le code de l'exercice n° 14 du TP n° 2. Remarquez qu'un triangle est une forme mais aussi un instrument de musique à une note.

Un instrument de musique `musical_instrument_t` est caractérisé par la fréquence en hertz de sa note `frequency`. Ajoutez donc cet attribut à la classe `musical_instrument_t`. Dans la classe `musical_instrument_t`, définissez un constructeur à un paramètre `frequency`. Dans la classe `musical_instrument_t`, définissez un accesseur pour l'attribut `frequency`.

Faites hériter la classe dérivée `triangle_t` des classes de base `shape_t` et `musical_instrument_t`. Dans la classe `triangle_t`, remplacez le constructeur à deux paramètres par un constructeur à trois paramètres `width`, `height` et `frequency`. Ce constructeur appelle le constructeur de la classe `shape_t` et le constructeur de la classe de base `musical_instrument_t`. Modifiez le corps de la méthode `print` de la classe `triangle_t` pour qu'elle affiche également la fréquence.

Utilisez le mot clé `class` pour la déclaration des toutes les classes afin que tous les membres aient une visibilité privée par défaut. Rendez publiques toutes les méthodes de toutes les classes. Rendez publics tous les héritages de toutes les classes dérivées.

Dans la fonction principale, passez trois paramètres au lieu de deux lors de l'appel au constructeur du triangle. La fréquence de la note du triangle est de 1047 Hz. Remarquez que dans cet exercice, il n'y a pas de problème d'ambiguïté lié à l'héritage multiple. Lancez votre programme avec la commande `valgrind` afin de vérifier si la mémoire allouée dynamiquement a correctement été libérée.

Voici le résultat attendu :

```
$ ./shape.out
Rectangle (width: 10.00, height: 20.00, area: 200.00)
Triangle (width: 30.00, height: 30.00, area: 450.00, frequency: 1047.00 Hz)
Circle (width: 40.00, height: 40.00, area: 1256.00)
$
```

Voici les fonctions autorisées pour cet exercice :

```
int printf(const char *format, ...);
```

Exercice n° 4

Reprenez le code de l'exercice n° 7 du TP n° 3. Ajoutez une classe dérivée `amphibian_t` qui représente un véhicule amphibie qui hérite des classes de base `car_t` et `boat_t`. Dans cette classe, définissez un constructeur à six paramètres `fuel`, `fuel_cons`, `fuel_max`, `name`, `hand_brake_enabled` et `anchor_enabled`. Ce constructeur appelle les constructeurs des classes `vehicle_t`, `car_t` et `boat_t`. Dans cette classe, redéfinissez la méthode `print` pour qu'elle affiche les informations sur le véhicule amphibie. Dans la classe `car_t`, définissez un accesseur pour l'attribut `hand_brake_enabled`. Dans la classe `boat_t`, définissez un accesseur pour l'attribut `anchor_enabled`.

Utilisez le mot clé `class` pour la déclaration des toutes les classes afin que tous les membres aient une visibilité privée par défaut. Rendez publiques toutes les méthodes de toutes les classes. Rendez publics tous les héritages de toutes les classes dérivées.

Dans la fonction principale, créez un véhicule amphibie. Ce véhicule amphibie est nommé « LARC-V », son carburant restant est de 10,1 litres, sa consommation de carburant est de 210 litres pour 100 kilomètres, son carburant maximum est de 12,9 litres, son frein à main est inactif et son ancre est inactive. Une fois votre véhicule amphibie créé, après le déplacement de l'avion, appelez sur celui-ci les méthodes `print`, puis `move` en passant `1` en paramètre, puis `print`, puis `move` en passant `100` en paramètre, puis `print`. Vérifiez la valeur de retour de chaque appel de la méthode `move` afin d'afficher un message indiquant si c'est une réussite ou un échec. Si c'est une réussite, affichez ce message dans le flux de sortie standard, sinon affichez ce message dans le flux d'erreur standard. Après le ravitaillement en carburant de l'avion, ravitaillez le véhicule amphibie en carburant en appelant la méthode `refuel` puis appelez la méthode `print` sur le véhicule amphibie.

L'héritage multiple de la classe `amphibian_t` pose des problèmes d'ambiguïté lors de l'accès aux membres. D'abord, elle hérite deux fois de la classe `vehicle_t` mais aussi il y a un conflit de nom puisque la méthode `move` est disponible dans les deux classes de base. Le premier problème peut être résolu avec l'héritage virtuel (qui permet d'hériter que d'une fois de la même classe) mais le deuxième problème ne pourra être résolu qu'avec l'opérateur de résolution de portée. Levez ces ambiguïtés à l'aide de l'héritage virtuel (pour le premier problème) et à l'aide de l'opérateur de résolution de portée (pour le deuxième problème).

Voici le résultat attendu :

```
$ ./vehicles.out
Car (Mercedes-Benz A-Class, 1.70/2.10 l, 11.00 l/100 km, hand brake: 0)
Car moved (1 km).
Car (Mercedes-Benz A-Class, 1.59/2.10 l, 11.00 l/100 km, hand brake: 0)
Unable to move the car (100 km).
Car (Mercedes-Benz A-Class, 1.59/2.10 l, 11.00 l/100 km, hand brake: 0)
Car (BMW X4, 57.20/65.00 l, 11.30 l/100 km, hand brake: 0)
Car moved (1 km).
Car (BMW X4, 57.09/65.00 l, 11.30 l/100 km, hand brake: 0)
Car moved (100 km).
Car (BMW X4, 45.79/65.00 l, 11.30 l/100 km, hand brake: 0)
Car (BMW X4, 45.79/65.00 l, 11.30 l/100 km, hand brake: 1)
Unable to move the car (100 km).
Car (BMW X4, 45.79/65.00 l, 11.30 l/100 km, hand brake: 1)
Boat (Zodiac, 1.40/1.70 l, 4.70 l/100 km, anchor: 0)
Boat moved (1 km).
Boat (Zodiac, 1.35/1.70 l, 4.70 l/100 km, anchor: 0)
Unable to move the boat (100 km).
Boat (Zodiac, 1.35/1.70 l, 4.70 l/100 km, anchor: 0)
Plane (Concorde, 99786.00/119500.00 l, 544.00 l/100 km, landing gear: 0)
Plane moved (1 km).
Plane (Concorde, 99780.56/119500.00 l, 544.00 l/100 km, landing gear: 0)
Plane moved (100 km).
Plane (Concorde, 99236.56/119500.00 l, 544.00 l/100 km, landing gear: 0)
Amphibian (LARC-V, 10.10/12.90 l, 210.00 l/100 km, hand brake: 0, anchor: 0)
Amphibian moved (1 km).
Amphibian (LARC-V, 8.00/12.90 l, 210.00 l/100 km, hand brake: 0, anchor: 0)
Unable to move the amphibian (100 km).
Amphibian (LARC-V, 8.00/12.90 l, 210.00 l/100 km, hand brake: 0, anchor: 0)
Car (Mercedes-Benz A-Class, 2.10/2.10 l, 11.00 l/100 km, hand brake: 0)
Car (BMW X4, 65.00/65.00 l, 11.30 l/100 km, hand brake: 1)
Boat (Zodiac, 1.70/1.70 l, 4.70 l/100 km, anchor: 0)
Plane (Concorde, 119500.00/119500.00 l, 544.00 l/100 km, landing gear: 0)
Amphibian (LARC-V, 12.90/12.90 l, 210.00 l/100 km, hand brake: 0, anchor: 0)
vehicle_t::get_nkm(): 205.00
car_t::get_nkm(): 103.00
boat_t::get_nkm(): 1.00
plane_t::get_nkm(): 101.00
$
```

Voici les fonctions autorisées pour cet exercice :

```
int fprintf(FILE *stream, const char *format, ...);
int printf(const char *format, ...);
```

Exercice n° 5

Reprenez le code de l'exercice précédent. Levez les ambiguïtés liées à l'héritage multiple à l'aide de l'opérateur de résolution de portée. Puisque la classe `vehicle_t` est héritée de deux façons différentes par la classe `amphibian_t`, en passant par `car_t` et en passant par `boat_t`, vous devez choisir de passer par l'un des deux chemins pour accéder aux membres posant problème. Vous ne devez pas utiliser l'héritage virtuel dans cet exercice.

Voici les fonctions autorisées pour cet exercice :

```
int fprintf(FILE *stream, const char *format, ...);
int printf(const char *format, ...);
```

Exercice n° 6

Reprenez le code de l'exercice n° 8 du TP n° 3. Dans la classe `student_t`, remplacez les attributs `courses` et `ncourses` par `studied_courses` et `nstudied_courses`. En conséquence de ce changement, modifiez les méthodes de cette classe. Ajoutez des accesseurs pour les attributs de cette classe.

Ajoutez une classe `teacher_t` qui décrit un enseignant. Cette classe dérivée hérite de la classe de base `person_t`. Un enseignant est caractérisé par les cours qu'il enseigne `taught_courses` (un pointeur vers `course_t`) et le nombre de cours qu'il enseigne `ntaught_courses` (un entier). Ajoutez donc ces deux attributs à la classe `teacher_t`. Dans cette classe, ajoutez un constructeur à deux paramètres qui appelle le constructeur de la classe de base à deux paramètres et qui initialise l'attribut `taught_courses` à `NULL` et l'attribut `ntaught_courses` à 0. Dans cette classe, ajoutez un destructeur qui libère la mémoire allouée dynamiquement pour l'attribut `taught_courses` grâce à l'opérateur `delete`. Dans cette classe, ajoutez une méthode `add_course` à un paramètre `course` (un `course_t`) et qui ne retourne rien. Cette méthode ajoute un nouveau cours aux cours enseignés par l'enseignant. Cette méthode doit être implémentée à l'aide des opérateurs `new`

et `delete`. Une fois le cours ajouté, cette méthode doit incrémenter la valeur de l'attribut `ntaught_courses`. Dans cette classe, redéfinissez la méthode `print` qui affiche les informations sur l'enseignant. Cette méthode doit être implémentée en appelant les accesseurs des classes `person_t` et `course_t`.

Ajoutez une classe `phd_student_t` qui décrit un doctorant qui étudie et enseigne à la fois. Cette classe dérivée hérite des classes de base `student_t` et `teacher_t`. Dans cette classe, définissez un constructeur à deux paramètres `firstname` et `lastname`. Ce constructeur appelle les constructeurs des classes `student_t` et `teacher_t`. Dans cette classe, redéfinissez la méthode `print` pour qu'elle affiche les informations sur le doctorant. L'affichage doit comprendre les cours étudiés mais aussi les cours enseignés. Cette méthode doit être implémentée en appelant les accesseurs des classes `person_t`, `student_t`, `teacher_t` et `course_t`.

Utilisez le mot clé `class` pour la déclaration des toutes les classes afin que tous les membres aient une visibilité privée par défaut. Rendez publiques toutes les méthodes de toutes les classes. Rendez publics tous les héritages de toutes les classes dérivées. La méthode `print` définie dans la classe `person_t` et redéfinie dans ses classes dérivées doit afficher les informations sur l'objet courant de cette façon : type de l'objet suivi de parenthèses qui contiennent la valeur de chaque attribut séparée par une virgule.

Dans la fonction principale, remplacez les appels à la méthode `add_course` (qui n'existe plus) par des appels à la méthode `add_studied_course`. Créez un cours qui s'appelle « How to write a thesis ? ». Créez deux enseignants et doctorant. Le premier s'appelle Wilfrid Lefer et le deuxième Samson Pierre. Créez un doctorant qui s'appelle Nathan Pas. Appelez la méthode `add_taught_course` sur les enseignants autant de fois que nécessaire pour affecter à Wilfrid Lefer le cours « Programmation orientée objet » et à Samson Pierre les cours « Techniques de programmation » et « Programmation orientée objet ». Appelez la méthode `add_studied_course` sur le doctorant pour affecter à Nathan Pas le cours « How to write a thesis ? ». Appelez la méthode `add_taught_course` sur le doctorant autant de fois que nécessaire pour affecter à Nathan Pas les cours « Techniques de programmation » et « Programmation orientée objet ». Appelez la méthode `print` sur les enseignants et le doctorant. Lancez votre programme avec la commande `valgrind` afin de vérifier si la mémoire allouée dynamiquement a correctement été libérée.

Levez les ambiguïtés liées à l'héritage multiple à l'aide de l'héritage virtuel pour n'hériter qu'une seule fois de la classe `person_t`. Vous ne devez pas utiliser l'opérateur de résolution de portée dans cet exercice.

Voici le résultat attendu :

```
$ ./person.out
Student ("Jean", "Bon", ("Techniques de programmation"))
Student ("Paul", "Emploi", ("Techniques de programmation", "Programmation orientée objet"))
Student ("Anne", "Alyse", ("Techniques de programmation", "Programmation orientée objet",
"Technologie orientée objet"))
Teacher ("Wilfrid", "Lefer", ("Programmation orientée objet"))
Teacher ("Samson", "Pierre", ("Techniques de programmation", "Programmation orientée objet"))
Ph.D. ("Nathan", "Pas", ("How to write a thesis?"), ("Techniques de programmation", "Programmation
orientée objet"))
$
```

Voici les fonctions autorisées pour cet exercice :

```
int printf(const char *format, ...);
```

Exercice n° 7

Reprenez le code de l'exercice précédent. Levez les ambiguïtés liées à l'héritage multiple à l'aide de l'opérateur de résolution de portée. Puisque la classe `person_t` est héritée de deux façons différentes par la classe `phd_student_t`, en passant par `student_t` et en passant par `teacher_t`, vous devez choisir de passer par l'un des deux chemins pour accéder aux membres posant problème. Vous ne devez pas utiliser l'héritage virtuel dans cet exercice.

Voici les fonctions autorisées pour cet exercice :

```
int printf(const char *format, ...);
```

Exercice n° 8

Créez le programme `minmax.cpp` dans lequel vous définissez deux fonctions templates `min` et `max`. La fonction template `min` a deux paramètres `param1` (une valeur de type générique `T`) et `param2` (une valeur de type générique `T`) et retourne le minimum des deux paramètres (une valeur de type générique `T`). Pour l'implémentation de cette fonction, utilisez l'opérateur ternaire. La fonction template `max` a deux paramètres `param1` (une valeur de type générique `T`) et `param2` (une valeur de type générique `T`) et retourne le maximum des deux paramètres (une valeur de type générique `T`).

Dans la fonction principale, appelez la fonction template `min` en y passant en paramètres `10` (un entier) et `42` (un entier) et affichez sa valeur de retour. Appelez la fonction template `min` en y passant en paramètres `9.99` (un réel) et `3.14` (un réel) et affichez sa valeur de retour. Appelez la fonction template `max` en y passant en paramètres `10` (un entier) et `42` (un

entier) et affichez sa valeur de retour. Appelez la fonction template `max` en y passant en paramètres `9.99` (un réel) et `3.14` (un réel) et affichez sa valeur de retour. Puisque les types souhaités sont non ambigus, vous ne devez pas les indiquer lors de l'utilisation de ces fonctions templates dans cet exercice.

Voici le résultat attendu :

```
$ ./minmax.out
min(10, 42) = 10
min(9.990000, 3.140000) = 3.140000
max(10, 42) = 42
max(9.990000, 3.140000) = 9.990000
$
```

Voici les fonctions autorisées pour cet exercice :

```
int printf(const char *format, ...);
```

Exercice n° 9

Reprenez le code de l'exercice précédent. Remarquez que lorsque vous passez un premier paramètre à l'une de vos fonctions templates, le deuxième paramètre doit obligatoirement être du même type que le premier. Pour résoudre ce problème, utilisez un deuxième type générique `U` pour le deuxième paramètre de vos fonctions templates. Dans la fonction principale, appelez la fonction template `min` en y passant en paramètres `9.99` (un réel) et `10` (un entier) et affichez sa valeur de retour. Appelez la fonction template `max` en y passant en paramètres `9.99` (un réel) et `10` (un entier) et affichez sa valeur de retour. Puisque les types souhaités sont non ambigus, vous ne devez pas les indiquer lors de l'utilisation de ces fonctions templates dans cet exercice.

Voici le résultat attendu :

```
$ ./minmax.out
min(10, 42) = 10
min(9.990000, 3.140000) = 3.140000
min(9.990000, 10) = 9.990000
max(10, 42) = 42
max(9.990000, 3.140000) = 9.990000
max(9.990000, 10) = 10.000000
$
```

Voici les fonctions autorisées pour cet exercice :

```
int printf(const char *format, ...);
```

Exercice n° 10

Reprenez le code de l'exercice précédent. Bien que les types souhaités soient non ambigus, indiquez-les lors de l'utilisation de vos fonctions templates dans cet exercice.

Voici les fonctions autorisées pour cet exercice :

```
int printf(const char *format, ...);
```

Exercice n° 11

Reprenez le code de l'exercice précédent. Dans le template de vos fonctions templates, après les deux paramètres génériques `T` et `U`, ajoutez le paramètre `debug` (un booléen). Vos templates ont donc trois paramètres (deux génériques et un non générique). Dans le corps de vos fonctions faites un affichage de débogage si `debug` vaut `true`. Dans la fonction principale, appelez vos fonctions templates avec `debug` qui vaut `false` puis appelez-les avec `debug` qui vaut `true`.

Voici le résultat attendu :

```
$ ./minmax.out
min(10, 42, false) = 10
min(9.990000, 3.140000, false) = 3.140000
min(9.990000, 10, false) = 9.990000
max(10, 42, false) = 42
max(9.990000, 3.140000, false) = 9.990000
max(9.990000, 10, false) = 10.000000
debug: min function called
min(10, 42, true) = 10
debug: min function called
min(9.990000, 3.140000, true) = 3.140000
```

```
debug: min function called
min(9.990000, 10, true) = 9.990000
debug: max function called
max(10, 42, true) = 42
debug: max function called
max(9.990000, 3.140000, true) = 9.990000
debug: max function called
max(9.990000, 10, true) = 10.000000
$
```

Voici les fonctions autorisées pour cet exercice :

```
int printf(const char *format, ...);
```

Exercice n° 12

Reprenez le code de l'exercice n° 14 du TP n° 2. Modifiez toutes les classes pour qu'elles soient des classes templates à un type générique `T`. Dans ces classes, pour les attributs, paramètres de méthodes et valeurs de retour de méthodes qui étaient de type `float`, remplacez le type par le type générique `T`. Dans ces classes, supprimez la méthode `print`.

Dans la fonction principale, indiquez le type `int` lors de la création de vos trois formes puis créez trois autres formes similaires mais avec le type `float`. Pour vos six formes, affichez la valeur de retour de l'appel à la méthode `get_area`. Lancez votre programme avec la commande `valgrind` afin de vérifier si la mémoire allouée dynamiquement a correctement été libérée.

Voici le résultat attendu :

```
$ ./shape.out
ps1->get_area() = 200
ps2->get_area() = 450
ps3->get_area() = 1256
ps4->get_area() = 200.000000
ps5->get_area() = 450.000000
ps6->get_area() = 1256.000000
$
```

Voici les fonctions autorisées pour cet exercice :

```
int printf(const char *format, ...);
```