EIDGENÖSSISCHE TECHNISCHE HOCHSCHULE – LAUSANNE
POLITECNICO FEDERALE – LOSANNA
SWISS FEDERAL INSTITUTE OF TECHNOLOGY – LAUSANNE

**Faculté Informatique et Communication**
Cours d'Informatique à la section SV
Sam J.

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

# PROGRAMMATION II

# Corrigé Examen

## Exercice 1 : Conception OO et programmation (61 points)

Il existe plusieurs façons de coder les détails des classes impliquées.
Notez que seuls les prototypes étaient demandés dans la partie 1.
Un corrigé possible est fourni dans les fichier sources `termites.cc`

## Exercice 2 : Questions sur les concepts (39 points)

1. (a)
```
A-> 1
B-> 2
~A()
```
justification: le constructeur par défaut de `B` est appelé à la ligne 30. Ce dernier appelle le constructeur par défaut de `A` (implicitement). c'est ce qui cause les deux premiers affichages. La ligne 31 appelle le destructeur sur la variable `ptr_a` qui est de type `A*` mais contient un pointeur sur un `B`. Comme le destructeur de `A` n'est pas virtuel, `ptr_a` est vu comme un `A*` et non un `B*` et du coup il n'y a que le destructeur de `A` qui est appelé d'où le dernier affichage.

   (b) puisque seul le destructeur de `A` est appelé, l'attribut `b_` de `B`, dynamiquement alloué n'est jamais désalloué.

   (c) Pour réparer la fuite de mémoire, il suffit de déclarer le destructeur de `A` comme virtuel. L'affichage devient:
```
A-> 1
 B-> 2
~B()
~A()
```

2. (a) Le programme affiche :
```
B
```
justification: la méthode `output` est virtuelle et elle s'applique en ligne 20 a un pointeur. Le polymorphisme est donc possible et `ptr_a` est vu comme un `B*` même s'il est de type `A*`. Du coup, c'est la méthode `output` de `B` qui est appelée.

   (b) si on supprime le `const`, la méthode déclarée à la ligne 12 n'est plus une redéfinition (`override`) de celle déclarée en ligne 5. On n'est donc plus dans le cadre d'une situation polymorphique lors de l'appel de la ligne 20. `ptr_a` n'est plus vu que comme un `A*`, et le programme affiche
```
A
```

   (c) on est à nouveau dans un cadre polymorphique, mais on appelle explicitement sur un objet de type `B` la méthode `output` de sa super-classe. Le programme affiche alors aussi:
```
A
```

3. La virtualité pure permet de prévoir la présence d'une méthode dans une hiéraérarchie de classe sans avoir à lui donner un corps (parcequ 'il n'est pas possible de lui en donner un à un niveau d'abstraction donné). Par exemple une méthode de calcul de surface impossible à coder dans une classe `FigureGeometrique` mais qu'il devient possible de coder concrètement dans une sous-classe `Carre`. Par opposition à la virtualité pure, définir une méthode `surface()` avec un corps vide dans `FigureGeometrique` n'est pas une bonne solution car cela n'impose pas sa redéfinition correcte dans la classe `Carre`.
Bonus: en C++, le destructeur peut-être déclaré virtuel pur dans une classe dans le but de la rendre abstraite (non instantiable).

4. (a) la zone mémoire associée à la variable `a` et allouée statiquement est désallouée deux fois : une fois lors de la destruction de `unA` (appelé à la fin du `main`) et en sortant de la portée de `main` (désallocation automatique des variables allouées statiquement).

   (b) le destructeur de `A` n'a pas de raison de faire un `delete`, la classe `A` n'est responsable d'aucune allocation dynamique (ce n'est pas parcequ"un attribut est un pointeur que le destructeur doit faire un `delete`). Il faut donc supprimer l'instruction `delete`.

5. Il faut utiliser un template :

```
template <typename T>
size_t count(std::vector<T> v, T val)
{
   size_t count(0);
   for (const auto& entry : v){
      if( entry == val) ++count;
   }
   return count;
}
```

   Cette fonction pourra être instanciée avec tout type `T` disposant de l'opérateur de comparaison `==`

6. (a) l'itération sur ensemble de valeur fait un parcours en lecture du coup les entrées de `collection` ne sont pas mises à zéro quand il le faut (ce sont des copies qui sont mises à zéro).

   (b) il faut ajouter `&` après `auto`.



## Exercice 3 : Déroulement de programme [15 points]

Le programme affiche:

```
1 : -4
2 : 8
3 : 8
4 : 8
5 : 8
```

de brèves explications sont attendues sur les points suivants:

1. appels des constructeurs : les 2 mêmes pour tous (par défaut et avec int)
2. invocation des construteurs de copie
3. polymorphisme dans `operator+=` car référence (`other`) ou pointeur (`this`, implicite) et méthodes virtuelles (`f` et `g`)
4. les 4 derniers cas sont pareils (en raison des explications ci-dessus)
5. méthodes `f` et `g` appelées; (`A::g`, `B::f`, `B::g` et `C::f`)
6. `operator+=` retourne l'objet, i.e. le premier opérande, qui est affiché grace à la surchage de `operator<<` (qui affiche simplement `a`).

EXAMEN COURS PROGRAMMATION 2 – SV
J. Sam
24 Mai 2016