Examen partiel 443 – Programmation orientée objet en C++

Université Paris Saclay – Master E3A

8 mars 2024

Les documents ne sont pas autorisés. Le barème donné est indicatif. Toute question peut être traitée en admettant les résultats précédents.

Exercice 1 — Questions de compréhension

1. Écrire la forme canonique de Coplien (FCC) de la classe A. Donnez des exemples des appels (à partir de la fonction `main` d'un programme) qui utilisent tout ce que vous avez implémenté pour la FCC.
2. Expliquez en détail ce qui se passe en mémoire et quelles fonctionnalités de la classe A sont invoquées lors de l'exécution de :

   ```
   A** tab = new A*[5];
   ```

3. Expliquez ce qui se passe dans les instructions suivantes :

   ```
   int *ptr;
   int &ref = *ptr;
   ```

4. Marcel a écrit le code suivant :

   ```
   #include <iostream>
   using namespace std;
   int main()
   {
       int array[2];
       array[0] = 1;
       array[1] = 2;
       array[3] = 3;
       array[4] = 4;
       cout << array[3] << endl;
       cout << array[4] << endl;
       return 0;
   }
   ```

   Marcel compile son code et exécute ce programme, qui affiche 3 et 4. Est-ce qu'il y a des erreurs dans le code ? Si oui, expliquez en détail ce qui se passe.

Exercice 2 — Utilisation de base

1. Quels sont les opérateurs **booléens** que vous connaissez en C++ ? Donnez des exemples très simples d'utilisation.
2. Vous disposez de trois variables booléennes :

   ```
   bool a, b, c;
   ```

   En utilisant uniquement des opérateurs booléens, construire une expression booléenne à partir de `a, b, c` qui donne `true` si au moins deux des trois variable est `true`, et qui donne `false` sinon.

Exercice 3 — Utilisation de base

1. Expliquer ce qui se passe dans le programme suivant :

   ```
   #include <stdlib.h>
   #include <stdio.h>
   #include <iostream>
   ```

```
using namespace std;

class Test {
public:
  void* operator new[](long unsigned int size) { return NULL;}
};

int main()
{
  Test *obj = new Test;
  Test *arr = new Test[10];
  cout << obj << endl;
  cout << arr << endl;
  return 0;
}
```

Qu'est-ce qu'on verra dans la console lors d'une exécution de ce programme ?

Exercice 4 — Utilisation de base

1. Expliquer ce qui se passe dans le programme suivant :

```
#include<iostream>
using namespace std;
class A
{
    int i;
public:
    A(int ii = 0) : i(ii) {}
    void show() {  cout << i << endl;  }
};

class B
{
    int x;
public:
    B(int xx) : x(xx) {}
    operator A() const {  return A(); }
};

void g(A a)
{
    a.show();
}

int main()
{
    B b(10);
    g(b);
    return 0;
}
```

Qu'est-ce qu'on verra dans la console lors d'une exécution de ce programme ?

Exercice 5 — Algorithmique (8 points)

Un **ensemble** ("set" en anglais) est une collection non-ordonnée d'objets. Par exemple, l'ensemble d'entiers {1,2,3} est égal a l'ensemble {3,1,2}. De même, les éléments dupliqués n'apportent rien de plus au contenu de l'ensemble : l'ensemble d'entiers {3,1,2} est égal à l'ensemble {3,1,2,3,2}. L'ensemble vide ne contient pas d'élément et son symbole est ∅ ou { }.
L'objectif de l'exercice est d'implémenter une classe `IntegerSet` qui gère un ensemble d'entiers avec des valeurs comprises entre 0 et N-1. La classe contient deux membres privés, un tableau

```
bool vals[N];
```

où `vals[i]` est `true` si la valeur `i` est présente dans l'ensemble au moins une fois, sinon `vals[i]` est `false`. Le deuxième membre privé

```
int d;
```

représente le nombre de valeurs distinctes de l'ensemble.

1. Quel est le contenu des membres privés correspondants aux ensemble suivants pour `N=5` : l'ensemble vide, l'ensemble `A = {0}` et l'ensemble `B = {1,3,4,3,1,3}` ?
   Pour la suite de l'exercice, la constante `N` est déclarée dans l'en-tête de la classe `IntegerSet` par la directive macroprocesseur suivante :

   ```
   #define N 10000
   ```

2. Ecrire la déclaration de la classe, et implémenter le constructeur sans paramètres de la classe qui initialise l'objet courant à l'ensemble vide.
3. Implémenter le constructeur

   ```
   IntegerSet(int* v, int n)
   ```

   qui prend en paramètre un vecteur d'entiers et la taille de ce vecteur, et qui initialise correctement les membres privés de l'objet. Grâce a ce constructeur, on pourra initialiser dans la fonction `main()` un objet de type `IntegerSet` de la manière suivante :

   ```
   int valeurs[]={3,1,2,3,2};
   IntegerSet obj1(valeurs,5);
   ```

   Quelle est la condition pour que les entiers contenus dans le vecteur soient valides ? Vérifiez cette condition dans le constructeur.
4. Déclarer et implémenter le constructeur de copie.
5. Déclarer et implémenter l'opérateur d'affectation et le destructeur.
6. Implémenter une méthode `bool isEmpty()` qui retourne `true` seulement si l'objet courant est un ensemble vide.
7. Surcharger l'opérateur `<<` qui vous permettra d'insérer dans le flot de sortie des objets comme `obj1` créé précédemment :

   ```
   cout << "L'ensemble␣est:␣" << obj1 << endl;
   ```

   Suite a l'instruction précédente, on devrait voir apparaître dans la console :

   ```
   L'ensemble␣est:␣{1␣2␣3}
   ```

8. Surcharger l'opérateur `+` qui vous permettra d'avoir en résultat l'union de deux ensembles. Quel est l'union des ensembles `B = {1,3,4,3,1,3}` et `C = {3,1,2,3,2}` ? Écrire le code C++ qui réalise l'union entre `B` et `C` avec l'opérateur `+` implémenté précédemment.
9. Surcharger l'opérateur `-` qui vous permettra d'avoir en résultat la différence S_- entre deux ensembles S_1 et S_2 définie comme étant l'ensemble d'éléments de S_1 qui ne figurent pas en S_2 : $S_- = \{p \in S_1 \ \ et \ \ p \notin S_2\}$. Avec les ensembles `B` et `C` de la question précédente, quel est le résultat de `B-C` ?
10. Surcharger l'opérateur de comparaison `==` en vous appuyant sur les méthodes précédentes pour implémenter la propriété suivante : l'ensemble S_1 est identique à S_2 si et seulement si $S_1 - S_2$ est vide et $S_2 - S_1$ est vide.
11. (**difficile**) Les binômes sont toutes les paires **distinctes** d'éléments **distincts** de deux ensembles. Par exemple, pour `B` et `C` les binômes sont : $\{\{1,2\},\{1,3\},\{1,4\},\{2,3\},\{2,4\},\{3,4\}\}$. Déclarer et implémenter une **méthode** qui prend en paramètre un `IntegerSet` et qui retourne un tableau dynamique avec tous les binômes construits avec l'objet courant et l'objet paramètre.
12. Est-ce qu'il est possible, en appelant la méthode précédente, de connaître le nombre de binômes dans le tableau résultat ? Quelle amélioration proposez vous pour la méthode précédente ?