Programmation Orientée Objet

2022-2023

Devoir Surveillé 2 C

- Notes de cours et de TD/TP autorisées - Calculatrice autorisée - Durée : 1h30 - Ne pas répondre sur le sujet mais sur une copie à rendre avec nom prénom ou/et num étudiant - 2 pages - par défaut aucune justification n'est demandée -

Exercice 1. *Problème de compilation ?*

Ce programme compile et s'exécute correctement. Indiquer ce qu'affiche ce programme lorsqu'il est exécuté.

```
#include<iostream>
struct A{
    A(A& a){ std::cout<< "A(A&)_";}
    A(){ std::cout<< "A()_";}
   ~A(){std::cout<< "~A()_";}
};
void f(A b){}
int main()
{
    A a;
    A b=a;
    A* c;
    A& d=a;
    {
        A* n=new A{b};
        c=&d;
        A m;
    }
   A k;
}
```

Exercice 2. *Quad*

Voici du code C++ d'une classe de quadruplets. La classe mémorise quatres nombres.

```
#include<iostream>
class quad_t {
    const int a; // première composante du quadruplet
    const int b; // deuxième composante du quadruplet
    const int c; // troisième ...
    const int d; // quatrième ..
    /* a remplir */
};
```

Compléter la classe `quad_t` de façon à pouvoir utiliser la classe de la manière suivante :

```
const quad_t a{1,2,3,4}

a<<1;
   // renvoie un nouveau quadruplet (2,3,4,1) dont tout les nombres
   // on été décalé vers la gauche de 1
a<<2;
   // renvoie le quadruplet (3,4,1,2) (deux décalages vers la gauche)
a<<4;
   // renvoie le quadruplet (1,2,3,4)
```

Exercice 3. *Cmax*

Le but de cet exercice est de créer un patron de classe que l'on appellera `cmax_t`. Les objets de type `cmax_t` sont des conteneurs qui permettent de stocker des valeurs d'un type **type paramétré**. Les valeurs du type paramétré doivent pouvoir être comparées (avec l'opérateur `<`) et la classe `cmax_t` doit posséder une méthode `getMax` qui retire du conteneur la plus grande valeur stockée.

Il y a un exemple d'utilisation plus loin.

Penser à éviter les erreurs de gestions de mémoire (donc à bien implanter constructeur, affectation et destructeur) Interdiction d'utiliser les conteneurs de la bibliothèque standard (STL) comme `array`, `deque` etc...

L'évaluation de cet exercice ne portera pas sur les performances (la complexité en temps par exemple) de l'implantation.

```
cmax_t<int> C1{20};
   // creation d'une conteneur contenant au maximum 20 valeurs
   // on peut supposer que le type argument (ici int) possède un
   // constructeur sans paramètre et l'opérateur : <

C1.ajoute(4);
   // ajoute la valeur 4 dans le conteneur
   // doit signaler une erreur si plus de place (ici 20 places max).

int s=C1.getMax()
   // C1.getMax() retire la plus grande valeur stockée.
   // doit signaler une erreur si le conteneur est vide.

C1.nb()
   // renvoie le nombre de valeurs contenues actuellement
   // (au départ zero)

cmax_t<int> C2=C1;
   // construit C2, copie de C1

cmax_t<int> C3;
   // construit C3 un objet de type cmax_t pouvant contenir au
   // maximum 100 valeurs (par défaut)

C3=C2;
   // affecte à C3 une copie de C2

cmax_t<std::string> C4str{40};
   // C4str est  destiné à stocker des objets de type
   // string

C4str.ajoute("un");
```

Exercice 4. *Automaton*

On appelera Automaton un automate (déterministe) qui possède 3 états : $\{0, 1, 2\}$ sur un alphabet à 2 lettres $\{a, b\}$ Si l'automaton est dans l'état 0 et qu'on lui applique la transition a il passe dans l'état 2 : on peut résumer la règle ainsi $0 \xrightarrow{a} 2$. Voici l'ensemble des règles

- $0 \xrightarrow{a} 2$
- $0 \xrightarrow{b} 2$
- $1 \xrightarrow{a} 0$
- $1 \xrightarrow{b} 1$
- $2 \xrightarrow{a} 2$
- $2 \xrightarrow{b} 0$

On souhaite créer une classe `automaton_t` qui implante des automatons. Voici un exemple d'utilisation

```
1  automaton_t x{0};  // construit l'automate x dans l'état 0
2  automaton_t y{1}; // construit l'automate y dans l'état 1
3
4  y.trA() // applique la transition "a" à l'automate "y"
5          // "y" passe donc de l'état "1" à l'état 0 (selon la règle)
6
7  y.get() // renvoie 0 qui est l'état actuel de y
8
9  x.trB() // applique la transition "b" à l'automate "x"
10         // "x" passe donc de l'état 0 à l'état  2
```