

NOM Bloquet

Prénom Romain

Promo 2018

Date 12/05/2015



BLOQUET Romain
PL2 - 2014

MATIÈRE C++

Exercice 1.

①

a = 7

a = 10

x = 2

y = 1 3

#include <iostream>

using namespace std;

②

void exercice_references () {

int a = 7;

int b = 27;

int x = 2;

int y = 3;

int& ref = a;

cout << "a = " << ref << endl;

ref = ref + y;

cout << "a = " << ref << endl;

~~ref = x~~ Non

cout << "x = " << ref << endl;

y = y + (5 * ref);

cout << "y = " << y << endl;

}

325/4

Exercice 2.

①

musicien.cpp

#include "Musicien.h"
#include <iostream>
#include <string.h>

Musicien::Musicien (const string a_nom, Instrument a_instrument,
Date a_naissance): a_nom(a_nom), a_instrument
(a_instrument), a_naissance(a_naissance)
{ }.

0.75/1

②

Musicien::Musicien (const Musicien& m) {
~~strcpy (a_nom, m.a_nom);~~
~~a_instrument = m.a_instrument;~~
~~a_naissance = m.a_naissance;~~
~~return;~~
}.

A expliquer!!

0.5/1

③

a). Dans le .h: Void swap (Musicien& M1, Musicien& M2);

Dans le .cpp:

Void Musicien::Swap (Musicien& M1, Musicien& M2) {
Instrument tmp;
tmp = M1.Instrument;
M1.Instrument = M2.Instrument;
M2.Instrument = tmp;
return;
}

1/1

b). Dans le .h: friend Void swap (Musicien& M1, Musicien& M2);

Dans le .cpp: Void swap (Musicien& M1, Musicien& M2) {
Instrument tmp;
tmp = M1.Instrument;

M1. Instrument = M2. Instrument;

M2. Instrument = Imp;

return;

}

Exercice 3:

50/8

①. arbre.h.

#ifndef Arbre_H

#define Arbre_H

#include <iostream>

using namespace std;

class Arbre {

private: // attributs

double _taille;

int _age;

public: // méthodes

Arbre (double _taille = 175, int _age = 20);

// Constructeur avec valeur par défaut.

};

a). Il n'est pas nécessaire de définir un Constructeur de Copie, car en C++ un constructeur de Copie élémentaire est généré*. N'ayant pas de pointeurs et de références en attributs il n'est donc pas nécessaire d'en définir un autre. Dans ce cas il serait équivalent à :

Arbre (const Arbre & a) {

_taille = a._taille;

_age = a._age;

return;

* Copiant les attribut membres et membres.

b). En C++, une surcharge d'opérateur d'affectation est gérée automatiquement, affectant les attributs membre par membre. N'ayant pas de pointeurs et de références en attributs dans notre cas il n'est donc pas nécessaire de définir une surcharge d'opérateur d'affectation.

c). Il n'est pas nécessaire de définir un destructeur dans notre cas car la mémoire n'est pas allouée dynamiquement. Un destructeur est défini automatiquement et libère la mémoire à la fin de l'exécution du programme.

d). - Arbre (double_taille = 175, int_age = 20);

Accesseur en écriture :

```
void SetTaille (double taille) { _taille = taille; }  
void Set age (int age) { _age = age; }
```

Accesseur en lecture :

```
void printTaille () const { cout << "Taille = " << _taille << endl; }  
void print age () const { cout << "age = " << _age << endl; }
```

```
friend ostream & operator << (ostream & os, Arbre & b) {  
    os << b._taille << " " << b._age ;  
    return os ;  
}
```

②

ArbreTest.cpp

```
#include "arbre.h"  
#include "ostream"  
using namespace std;
```

NOM Blouquet
Prénom Romain
Promo 2018
Date 13/5/15

MATIÈRE C++

```
main() {
```

```
    Arbre a1();
```

```
    Arbre *a2(180, 21);
```

```
    a2 → new Setage(23);
```

```
    Cont << "premier arbre : " << a1 << "deuxième arbre : " << a2 << endl;
```

```
    return 0;
```

```
}
```

```
#include "musicien.h"
```

```
#include <iostream>
```

```
#include <string>
```

Exercice 4 :

Musicien :: Musicien (char * S1, int naissance, int mort, char * S2) {

```
    free a_nom;
```

```
    free a_instrument;
```

```
    a_nom = new char [(strlen S1) + 1];
```

```
    strcpy(a_nom, S1);
```

```
    a_naissance = naissance;
```

```
    a_mort = mort;
```

```
    a_instrument = new char [(strlen S2) + 1];
```

```
    strcpy(a_instrument, S2);
```

```
    return;
```

```
}
```

- ②. Oui il est nécessaire de définir un constructeur de copie car le constructeur de copie généré automatiquement ne permet pas de copier en profondeur et dans notre cas comme on a des

pointeurs en membre il le faut.

// Constructeur de Copie.

Musicien::Musicien (Musicien & m) {
~~if (this == &m) return *this;~~
~~free a_nom;~~

~~free a_instrument;~~

a_nom = new char [strlen(m.a_nom) + 1];

a_nom = m.a_nom;

a_naissance = m.a_naissance;

a_mort = m.a_mort;

a_instrument = new char [strlen(m.a_instrument) + 1];

a_instrument = m.a_instrument;

return *this;

}

③ Il n'est pas nécessaire de définir une surcharge de l'opérateur d'affectation car celui qui est géré automatiquement fait le travail nécessaire. ~~Non~~ pointeurs!

④ Comme il y a des pointeurs en attributs, le destructeur géré automatiquement ne sera pas suffisant dans notre cas. Il nous faudra donc le définir.

// destructeur.

Musicien::~Musicien() {

~~free a_nom;~~

~~free a_instrument;~~

return;

025/1

Exercice 5:

Une classe abstraite est une classe qui comporte une fonction virtuelle pure. On l'utilise en mettant Virtual avant la fonction et const = 0 à la fin.

exemple: Virtual Void getAin () const = 0;

Elle n'a pas de définition.

Exercice 6:

Le polymorphisme permet de choisir dynamiquement une méthode.

Elle permet à plusieurs objets d'utiliser la même interface.
(Interface commune).

Incomplet!

