Trimestre Automne, 2019

Mohamed Lokbani

IFT1169 – Examen Final –

Inscrivez tout de suite : votre nom et le code permanent.

Nom : _____| Prénom(s) : _____|

Signature : _____| Login : _____|

Date : mardi 10 décembre 2019

Durée : 3 heures (de 16h30 à 19h30)

Local : Z-305, Pavillon Claire McNicoll

Directives :

- Toute documentation est permise.
- Appareils électroniques **non** permis.

- Répondre directement sur le questionnaire.
- Les réponses **doivent être brèves, précises, claires et nettement présentées.**

1._____/20 (1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9. 1.10)

2._____/20 (2.1, 2.2, 2.3, 2.4)

3._____/20 (3.1)

4._____/20 (4.1)

5._____/20 (5.1)

Total : _____/100

Directives officielles

* Interdiction de toute communication verbale pendant l'examen.

* Interdiction de quitter la salle pendant la première heure.

* L'étudiant qui doit s'absenter après la première heure remettra sa carte d'étudiant au surveillant, l'absence ne devant pas dépasser 5 minutes. Un seul étudiant à la fois peut quitter la salle.

* Toute infraction relative à une fraude, un plagiat ou un copiage est signalée par le surveillant au directeur de département ou au professeur qui suspend l'évaluation.

F.A.S

Exercice 1 (20 points) répondez par « vrai » ou « faux » en y incluant une très courte explication.

1.1 [VRAI | FAUX] Les exceptions sont un des paradigmes de la programmation orientée objet.

1.2 [VRAI | FAUX] La liaison dynamique est essentielle pour réaliser l'héritage.

1.3 [VRAI | FAUX] L'opérateur de résolution de portée permet de sélectionner un membre via un objet.

1.4 [VRAI | FAUX] Même si le langage offre pour une classe canonique toutes les fonctionnalités par défaut, nous sommes dans l'obligation de définir dans tous les cas un destructeur.

1.5 [VRAI | FAUX] Une méthode "static" dans une classe, ne peut pas être déclarée virtuelle.

1.6 [VRAI | FAUX] Une redéfinition c'est quand deux méthodes d'une même classe portent le même nom, c'est juste le nombre d'arguments qui est différent.

1.7 [VRAI | FAUX] Les espaces de noms sont utilisés pour encapsuler les données.

1.8 [VRAI | FAUX] Il n'est pas possible de créer soi-même des instances de la classe « type_info ».

1.9 [VRAI | FAUX] Par héritage la classe dérivée récupère aussi les attributs privés même si elle n'a pas accès.

1.10 [VRAI | FAUX] La classe B hérite de la classe A. La classe A a un seul constructeur et ce dernier a des arguments. La classe B n'a pas besoin de définir un constructeur spécifique, celui offert par défaut peut faire l'affaire.

Exercice 2 (20 points)

2.1 Comment compiler avec « g++ » un programme qui utilise les threads enseignés en cours?

2.2 Dans un programme « multithreadés », si un thread crée une variable locale, tous les threads ont-ils accès à cette variable?

2.3 Un serveur crée un thread pour chaque client. Pas plus de N threads (et donc clients) ne peuvent être actives immédiatement. Comment dire au thread « main » qu'un thread enfant s'est terminé et qu'il peut servir maintenant un autre client?

2.4 Tout en ignorant le retour des différentes fonctions (pthread), le fragment de code suivant pose problème. Expliquer cela brièvement et proposer 2 manières pour fixer ce problème.

```
01  #include <iostream>
02  #include <thread>
03  #include <mutex>
04
05  std::mutex a;
06  std::mutex b;
07
08  void f(){
09          a.lock();
10          b.lock();
11          std::cout<<"Un thread C++11\n";
12          b.unlock();
13          a.unlock();
14  }
15  void g(int x){
16          b.lock();
17          a.lock();
18          std::cout<<"la valeur de x est: "<< x << std::endl;
19          a.unlock();
20          b.unlock();
21  }
22  int main(){
23          std::thread t(f);
24          std::thread v(g,10);
25          t.join();
26          v.join();
27
28          return 0;
29  }
```

Exercice 3 (20 points) ce programme **compile** et **s'exécute correctement** :

```
01  #include <iostream>
02  #include <string>
03
04  using namespace std;
05
06  class Alpha {
07          string message;
08  public:
09          Alpha (const string m):message(m){}
10          friend ostream& operator<<(ostream& out, const Alpha& e1){
11                  out << e1.message << endl;
12                  return out;
13          }
14  };
15
16  class Beta {
17          int valeur;
18  public:
19          Beta(int v): valeur(v) { }
20          int getValeur ( ) { return valeur; }
21  };
22
23  class Charlie { };
24
25  int F (int a, int b) {
26          if (a==0) throw Alpha("Inferieur");
27          if (a==3) return 10*b;
28          if (a>b) throw Charlie();
29          if (a<b) throw Beta(a-b);
30          return a+b;
31  }
32
33  void une_fonction(int i, int j){
34          try{
35                  F(i,j);
36          }
37          catch (Alpha a) { cout << a;}
38          catch (Beta b) { cout << b.getValeur( ) << endl;}
39          catch ( ... ) { cout << "Erreur" << endl;}
40          cout << '\t';
41
42  }
43  int main ( ) {
44          une_fonction(0,3);
45          une_fonction(6,2);
46          une_fonction(2,3);
47          une_fonction(3,2);
48
49          cout << "Fini!" << endl;
50          return 0;
51  }
```

Tout en développant votre réponse, que va-t-il afficher en sortie?

Exercice 4 (20 points) ce programme **compile** et **s'exécute correctement** :

```
1   #include<iostream>
2   using namespace std;
3
4   class test {
5           int* bob;
6
7   public:
8           test(int b): bob(new int(b)) {
9                   cout << "C\n";
10          }
11          ~test() {
12                  cout << "D\n";
13                  delete bob;
14          }
15          test(const test& t) {
16                  cout << "P\n";
17                  bob = new int(*(t.bob));
18          }
19          test(test&& t) {
20                  cout << "M\n";
21                  bob = t.bob;
22                  t.bob = nullptr;
23          }
24          test& operator=(const test& t) {
25                  cout << "A\n";
26                  *bob = *(t.bob);
27                  return *this;
28          }
29          test& operator=(test&& t) {
30                  cout << "T\n";
31                  if(this != &t) {
32                          bob = t.bob;
33                          t.bob = nullptr;
34                  }
35                  return *this;
36          }
37          friend test operator+(test lhs, const test& rhs) {
38                  cout << "Do +\n";
39                  lhs += rhs;
40                  return lhs;
41          }
42          test& operator+=(const test& rhs) {
43                  cout << "Dx +=\n";
44                  *bob += *(rhs.bob);
45                  return *this;
46          }
47  };
48
49  int main() {
50          test a(5), b(3);
51          cout << "ops:\n";
52          a+=b;
53          test c(a+b);
54          cout << "a:\n";
55          test d(a);
56
57          return 0;
58  }
```

Tout en développant votre réponse, que va-t-il afficher en sortie?

Exercice 5 (20 points) ce programme **compile** et **s'exécute correctement** :

```
 1  #include <iostream>
 2  using namespace std;
 3
 4  class Voiture {
 5          int vitesse;
 6  public:
 7          Voiture( ) { vitesse=0; }
 8          int getVitesse( ) { return vitesse; }
 9          void acceleration1(int delta) { vitesse+=delta; }
10          virtual void acceleration2(int delta) { vitesse+=delta; }
11  };
12
13  class VoitureCourse: public Voiture {
14  public:
15          VoitureCourse( ): Voiture( ) { }
16          void acceleration1(int delta) {
17                  Voiture::acceleration1(2*delta);
18          }
19          virtual void acceleration2(int delta) {
20                  Voiture::acceleration2(2*delta);
21          }
22  };
23  void stop1(Voiture& x) { x.acceleration1(-x.getVitesse( )); }
24
25  void stop2(Voiture& x) { x.acceleration2(-x.getVitesse( )); }
26
27  int main( ) {
28          Voiture c1, c2;
29          c1.acceleration1(65);
30          cout << c1.getVitesse( ) << endl;
31          // Affichage -1- avec une courte explication
32
33
34
35
36
37
38
39
40
41
42          stop1(c1);
43          cout << c1.getVitesse( ) << endl;
44          // Affichage -2- avec une courte explication
45
46
47
48
49
50
51
52
53
54
55          c2.acceleration2(65);
56          cout << c2.getVitesse( ) << endl;
57          // Affichage -3- avec une courte explication
58
59
60
61
62
63
64
65
66
```

```
 67          stop2(c2);
 68          cout << c2.getVitesse( ) << endl;
 69          // Affichage -4- avec une courte explication
 70
 71
 72
 73
 74
 75
 76
 77
 78
 79
 80          VoitureCourse r1, r2;
 81          r1.acceleration1(100);
 82          cout << r1.getVitesse( ) << endl;
 83          // Affichage -5- avec une courte explication
 84
 85
 86
 87
 88
 89
 90
100
101
102
103          stop1(r1);
104          cout << r1.getVitesse( ) << endl;
105          // Affichage -6- avec une courte explication
106
107
108
109
110
111
112
113
114
115
116          r2.acceleration2(100);
117          cout << r2.getVitesse( ) << endl;
118          // Affichage -7- avec une courte explication
119
120
121
122
123
124
125
126
127
128
129          stop2(r2);
130          cout << r2.getVitesse( ) << endl;
131          // Affichage -8- avec une courte explication
132
133
134
135
136
137
138
139
140
141
142
143
144
```

```
145         Voiture *p1=new VoitureCourse, *p2=new VoitureCourse;
146         p1->acceleration1(100);
147         cout << p1->getVitesse( ) << endl;
148         // Affichage -9- avec une courte explication
149
150
160
161
162
163
164
165
166
167
168         stop1(*p1);
169         cout << p1->getVitesse( ) << endl;
170         // Affichage -10- avec une courte explication
171
172
173
174
175
176
177
178
179
180
181         p2->acceleration2(100);
182         cout << p2->getVitesse( ) << endl;
183         // Affichage -11- avec une courte explication
184
185
186
187
188
189
190
191
192
193
194         stop2(*p2);
195         cout << p2->getVitesse( ) << endl;
196         // Affichage -12- avec une courte explication
197
198
199
200
201
202
203
204
205
206
207         return 0;
208 }
```

Joyeuses Fêtes et Bonne Année!