Trimestre Hiver, 2014

Mohamed Lokbani

IFT1169 – Examen Intra –

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

Nom : _____| Prénom(s) : _____|

Signature : _____| Code perm : _____|

Date : mardi 25 février 2014

Durée : 2 heures (de 18h30 à 20h30)

Local : Z-350 ; Pavillon Claire McNicoll

Directives :

- Toute documentation est permise.
- Calculatrice **non** permise.

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

1._____/20 (1.1. à 1.6)

2._____/20 (2.1 à 2.4)

3._____/30 (3.1)

4._____/30 (4.1 à 4.7)

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) Encerclez la bonne réponse et donnez une courte explication.

1.1 [VRAI | FAUX] La séance du cours IFT1169 a lieu le vendredi de 08h à 10h.

1.2 [VRAI | FAUX] On peut surcharger le destructeur.

1.3 [VRAI | FAUX] Le pointeur « `this` » n'est pas disponible pour une méthode statique.

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

1.5 [VRAI | FAUX] Le fichier d'en-tête « `string.h` » est nécessaire pour pouvoir manipuler une fonction à ellipse.

1.6 [VRAI | FAUX] Les fonctions en C++, autre que la fonction « `main` » sont exécutées avant l'exécution de la fonction « `main` ».

1.7 [VRAI | FAUX] Dans un « `makefile` », une ligne de commande doit obligatoirement débuter par une tabulation.

1.8 [VRAI | FAUX] Lorsque « `make` » exécute une commande, il affiche les commandes exécutées.

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

1.10 [VRAI | FAUX] Lors de la compilation, il suffit de spécifier les fichiers sources. Les fichiers d'en-tête seront automatiquement inclus.

Exercice 2 (20 points) Le programme suivant compile et s'exécute correctement.

```
01  #include <iostream>
02  using namespace std;
03
04  struct Erreur {
05          Erreur(const char * message_) : message(message_) {}
06          const char * message;
07  };
08
09  class Chaud {
10  public:
11          Chaud(int i_ = 0) : i(i_){
12                  if(i == 99)
13                          throw Erreur("Chaud n'aime pas le chiffre 99");
14
15                  cout << "Chaud " << i << " est construite" << endl;
16          }
17          ~Chaud(){
18                  cout << "Chaud " << i << " est detruite" << endl;
19          }
20  private:
21          int i;
22  };
23
24  bool throw_froid_constructeur = true;
25
26  class Froid {
27  public:
28          Froid() : membre_chaud(1), p1(nullptr), p2(nullptr) {
29                  try {
30                          p1 = new Chaud(76);
31                          p2 = new Chaud(99);
32                  }
33                  catch(...) {
34                          cout << "exception a ete levee dans le
35                                                  constructeur de Froid" << endl;
36                          cout << "p1= " << p1 << endl;
37                          delete p1;
38                          cout << "p2= " << p2 << endl;
39                          delete p2;
40                          throw;
41                  }
42
43                  if(throw_froid_constructeur) {
44                          cout << "Le constructeur de Froid va lever une
45                                                          exception" << endl;
46                          cout << "p1= " << p1 << endl;
47                          delete p1;
48                          cout << "p2= " << p2 << endl;
49                          delete p2;
50
51                          throw Erreur("Construction de Froid a plante!");
52                  }
53                  cout << "Froid est construite" << endl;
54          }
55          ~Froid(){
56                  delete p1;
57                  delete p2;
58                  cout << "Froid est detruite" << endl;
59          }
60  private:
61          Chaud membre_chaud;
62          Chaud* p1;
63          Chaud* p2;
64  };
65
66
67
68
```

```
69  int main(){
70          try {
71                  cout << "On declare un objet Froid" << endl;
72                  Froid g;
73                  cout << "Nous allons quitter le bloc try du main"
74                                                                  << endl;
75          }
76          catch(Erreur& erreur){
77                  cout << "capturer l'erreur: " << erreur.message
78                                                                  << endl;
79          }
80          cout << "Nous allons quitter le main()" << endl;
81
82          return 0;
83  }
```

2.1 Tout en développant brièvement votre réponse, que va afficher en sortie le précédent programme?

2.2 Tout en développant brièvement votre réponse, que va afficher en sortie le précédent programme si on remplace les lignes « 30 » et « 31 » par ce qui suit :

```
30                      p1 = new Chaud(76);
31                      p2 = new Chaud(6);
```

2.3 Tout en développant brièvement votre réponse, que va afficher en sortie le précédent programme si on remplace les lignes « 30 » et « 31 » par ce qui suit :

```
30                          p1 = new Chaud(99);
31                          p2 = new Chaud(6);
```

2.4 Tout en développant brièvement votre réponse, que va afficher en sortie le précédent programme si on remplace les lignes « 30 » et « 31 » par ce qui suit :

```
30                          p1 = new Chaud(99);
31                          p2 = new Chaud(99);
```

Exercice 3 (30 points) Le programme suivant compile et s'exécute correctement.

```
1  #include <iostream>
2  using namespace std;
3
4  class A {
5  public:
6        A() {cout << "Cons. A" << endl;}
7        A(A& a) {cout << "Cons. Recopie A" << endl;}
8        virtual ~A() {cout << "Destr. A" << endl;}
9        virtual void f() {cout << "f de A" << endl;}
10 };
11 class B: public A {
12 public:
13       B() {cout << "Cons. B" << endl;}
14       ~B() {cout << "Destr. B" << endl;}
15       void f() {cout << "f de B" << endl;}
16 };
17 class C: public B {
18 public:
19       C() {cout << "Cons. C" << endl;}
20       ~C() {cout << "Destr. C" << endl;}
21       void   f() {cout << "f de C" <<      endl;}
22 };
23 class X {
24 public:
25       X() {cout << "Cons. X" << endl;}
26       X(X& x) {cout << "Cons. Recopie X" << endl;}
27       X& operator=(const X& x) {cout << "Opt.= X" << endl; return
28 (*this);}
29       ~X() {cout << "Destr.  X" << endl; }
30       void f() {cout << "f de X" << endl;}
31 };
32 X foo(A& a, X x) {
33       cout << "*** On est dans foo" << endl;
34       A aa;
35       aa = a;
36       X xx(x);
37       return xx;
38 }
39 int main() {
40       cout << "*** On entre dans main" << endl;
41       A* a_ptr_to_a = new A;
42       B* b_ptr_to_b = new B;
43       C* c_ptr_to_c = new C;
44       X x1;
45       X* x_ptr_to_x = &x1;
46
47       a_ptr_to_a -> f();
48       b_ptr_to_b->f();
49       c_ptr_to_c -> f();
50       x_ptr_to_x->f();
51
52       A* a_ptr_to_b = b_ptr_to_b;
53       A * a_ptr_to_c = c_ptr_to_c;
54       B * b_ptr_to_c = c_ptr_to_c;
55
56       a_ptr_to_b -> f();
57       a_ptr_to_c -> f();
58       b_ptr_to_c -> f();
59
60       X x2;
61       x2 = foo(*a_ptr_to_a,x1);
62
63       delete a_ptr_to_a;
64       delete b_ptr_to_b;
65       delete b_ptr_to_c;
66
67       return 0;
68 }
```

Tout en expliquant brièvement votre démarche, donnez les affichages en sortie obtenus après l'exécution du précédent programme.

Réponse

Ligne ~~46~~ (40) `*** On entre dans main↵`

Un affichage en sortie avec « `cout` » de la chaîne « `*** On entre dans main` » puis retour à la ligne.

Exercice 4 (30 points) Soit le fragment de code suivant qui est syntaxiquement correct :

```
 1  #include <iostream>
 2  #include <string>
 3  #include <string.h>
 4
 5  using namespace std;
 6
 7  const char *lemois_str[]={ "Janvier", "Fevrier", "Mars", "Avril",
    "Mai",   "Juin",   "Juillet",   "Aout",   "Septembre",   "Octobre",
    "Novembre", "Decembre" };
10
11  class Mois {
12        int mois;
13  public:
14        Mois(int m);
15        Mois operator++();
16        Mois  operator++(int);
17
18        friend ostream& operator<<(ostream&, Mois);
19        friend istream& operator>>(istream&, Mois&);
20
21  };
22
23
24  /* ici vient votre code */
25
26  int main(){
27
28        Mois m1(1);
29        Mois m2(2);
30
31        for (int i = 0; i < 13; ++i) {
32              cout << m1++ << "  " << ++m2 << '\n';
33        }
34        cout << "Donner un mois, ou appuyez sur Ctrl-C pour sortir du
35  programme:\n";
36        for (;;){
              cin >> m1;
38            if (!cin)
39                    break;
40            cout << m1 << '\n';
41        }
42
43        return 0;
44  }
```

4.1 Écrivez la fonction « `int cherche_mois(string s)` ». Cette fonction accepte un « string » qui correspond au mois de l'année et retourne l'index de ce mois dans la table « `lemois_str` » définie à la ligne 7.

4.2 Écrivez le code du constructeur de la classe « `Mois(int m);` ».

4.3 Écrivez le code ~~de~~ de l'opérateur « `Mois operator++();` ».

4.4 Écrivez le code ~~de~~ de l'opérateur « `Mois  operator++(int);` ».

4.5 Écrivez le code ~~de~~ de l'opérateur « `friend ostream& operator<<(ostream&, Mois);` ».

4.6 Écrivez le code ~~de~~ de l'opérateur « `friend istream& operator>>(istream&, Mois&);` ».

4.7 Donnez l'affichage complet obtenu en sortie.

À la question «`"Donner un mois, ou appuyez sur Ctrl-C pour sortir du programme:\n";` », dans le premier cas de figure vous allez répondre « Mars ». Dans le second cas de figure, vous allez répondre « Rien ».