Trimestre Hiver, 2007

Mohamed Lokbani

IFT1169 – Examen Intra –

Inscrivez tout de suite votre nom et votre code permanent.

Nom: _____| Prénom(s): _____|

Signature: _____| Code perm: _____|

Date : jeudi 22 février 2007

Durée : 2 heures (de 18h30 à 20h30)

Local : Z-317 ; Pavillon Claire-McNicoll (ancienne aile Z).

Directives:

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

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

1._____/25 (1.1, 1.2, 1.3, 1.4, 1.5)

2._____/15 (2.1)

3._____/20 (3.1)

4._____/20 (4.1, 4.2, 4.3, 4.4)

5._____/20 (5.1, 5.2, 5.3)

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)

1.1 Citez les 3 paradigmes de la programmation orientée objet.

1.2 Soit le fragment de code suivant :

```
class A {
public:
    double uneautre(int,double);
};
class B : public A {} ;
```

a) « **Surdéfinir** » la fonction « double uneautre(int,double); » consiste à faire quoi au juste en pratique?

b) « **Redéfinir** » la fonction « double uneautre(int,double); » consiste à faire quoi au juste en pratique?

1.3 Un programmeur donné a écrit une série de lignes de code représentant un ensemble de classes et de méthodes. Ce programmeur a décidé de les rendre disponibles mais sans fournir pour cela le code source. Le programmeur doit fournir quoi au juste pour que cet ensemble de classes et de méthodes soit quand même utilisable?
Expliquez brièvement votre réponse :

1.4 Soit le fragment de code suivant :

```
class A {
private:
    int a;
protected:
    int b;
public:
    int c;
};
class B: protected class A{
public:
    void affiche();
};
```

a) Est-ce que la méthode affiche de la classe « B » a accès au membre « a » de la classe « A »? (dire pourquoi)

b) Est-ce que la méthode affiche de la classe « B » a accès au membre « b » de la classe « A »? (dire pourquoi)

c) Est-ce que la méthode affiche de la classe « B » a accès au membre « c » de la classe « A »? (dire pourquoi)

1.5 La liaison dynamique est essentielle pour assurer :

l'encapsulation	Vrai	Faux
le polymorphisme	Vrai	Faux
l'héritage	Vrai	Faux
la marginalisation	Vrai	Faux

Expliquez brièvement votre réponse :

Exercice 2 (20 points)

```
 1    #include <iostream>
 2    using namespace std;
 3
 4    class Test{
 5    protected :
 6        int x;
 7    public :
 8        Test (int i=0) : x(i) {}
 9        ~Test(){}
10        virtual void affiche (){cout << x << endl;};
11    };
12    int main (){
13        Test x;
14        x.affiche();
15        return 0;
16    }
17
```

exo2.cpp

Lors de la compilation du programme ci-dessus, le compilateur génère l'avertissement suivant :

```
>g++  -Wall -pedantic -Os -c exo2.cpp -o exo2.o
   exo2.cpp:4: warning: `class Test' has virtual functions but
   non-virtual destructor
>Exit code: 0
```

Expliquez la signification d'un tel avertissement. Dites par la suite comment il faudra le corriger dans le programme, pour éviter qu'il soit signalé par le compilateur.

Exercice 3 (20 points) Pour le fragment de code suivant, dites si les lignes d'instructions « 9 », « 16 », « 20 » et « 24 » sont syntaxiquement correctes ou non.

```
 1      namespace A {
 2          int i;
 3          namespace B {
 4              namespace C {
 5                  int i;
 6              }
 7              using namespace A::B::C;
 8              void f1() {
 9                  i = 5;   // ←
10              }
11          }
12          namespace D {
13              using namespace B;
14              using namespace C;
15              void f2() {
16                  i = 5; // ←
17              }
18          }
19          void f3() {
20              i = 5; // ←
21          }
22      }
23      void f4() {
24          i = 5; // ←
25      }
```

Ligne 9 | Correcte ☐ Incorrecte ☐

Explications :

Ligne 16 | Correcte ☐ Incorrecte ☐

Explications :

Ligne 20 | Correcte ☐ Incorrecte ☐

Explications :

Ligne 24 | Correcte ☐ Incorrecte ☐

Explications :

Exercice 4 (20 points) Soit le programme suivant qui compile correctement :

```
 1    #include <iostream>
 2    #include <exception>
 3    #include <cstdlib>
 4
 5    using namespace std;
 6
 7    double nombre = 0;
 8
 9    void convert(char* arg){
10
11        nombre = strtod(arg,0);
12        if (nombre == 0.0)
13            throw nombre;
14    }
15
16
17    int main(int argc, char* argv[]) {
18
19        if (argc > 1) {
20            try {
21                convert (argv[1]);
22                cout << " Bon ";
23            }
24            catch(double){
25                cout << " Mauvais ";
26                nombre = NAN; // NAN: Not-A-Number (valeur non numérique)
27            }
28        } else {
29            cout << " Mauvais ";
30        }
31        cout << nombre << endl;
32        return 0;
33    }
```

exo4.cpp

La fonction « double `strtod(const char* unechaine,char** ptr_fin)` » (`STRing TO Double`) convertit la chaîne de caractères « `unechaine` » en une valeur du type « `double` ».

- La chaîne de caractères « `unechaine` » doit correspondre au format suivant :
  « `[espace] [signe] [ddd] [.] [ddd] [e|E[signe]ddd]` ».
  Par exemple, les chaînes suivantes peuvent-être converties par la fonction « `strtod` » : `[+1234.2007e-1] [75362.74E9] [+469731.157].`

- L'argument « `ptr_fin` » s'il ne vaut pas « `NULL` », « `strtod` » lui donne comme valeur un pointeur sur le premier caractère qui a arrêté l'examen de la chaîne. Ce pointeur est pratique pour gérer les erreurs. Pour simplifier cet exercice, nous l'avons mis à zéro. Dans ce cas il ne sera pas utilisé. De ce fait, il ne faut pas s'attarder sur lui!

- La fonction renvoie la valeur convertie du type « `double` ». En cas de débordement, la valeur renvoyée est « `HUGE_VAL` » symbolisée dans cet exemple par « `0.0` ».

Tout en justifiant votre réponse, dites ce qui sera affiché en sortie si le programme est exécuté avec les commandes suivantes :

4.1 `exo4.exe`
Explications :

4.2 `exo4.exe 3`
Explications :

4.3 `exo4.exe G`
Explications :

4.4 `exo4.exe 3 G`
Explications :

Exercice 5 (20 points) Soit la fonction « `main` » suivante qui compile correctement :

```
 1    int main() {
 2        int x=2;
 3        Chiffre deux(x), moins_deux(-x);
 4        cout << "Le chiffre associé à deux est: " << deux <<endl;
 5        cout << "Le chiffre associé à moins_deux est: " << moins_deux << endl;
 6
 7        if (-deux == moins_deux)
 8            cout<< -deux << " est égal à " << moins_deux << endl;
 9
10        if (deux != moins_deux)
11            cout<< deux << " est différent de " << moins_deux << endl;
12
13        return 0;
14    }
15
```

Exo5.cpp

L'affichage en sortie est comme suit :

```
>exo5
Le chiffre associé à deux est: 2
Le chiffre associé à moins_deux est: -2
-2 est égal à -2
2 est différent de -2
>Exit code: 0
```

5.1 Citez le (ou les) constructeur(s) qu'il faudra définir dans la classe « Chiffre » pour assurer le bon fonctionnement de la fonction « main » ci-dessus.

5.2 Citez les opérateurs qu'il faudra surcharger dans la classe « Chiffre » pour assurer le bon fonctionnement de la fonction « main » ci-dessus.

5.3 Écrire la définition complète (code complet des méthodes, fonctions etc.) de la classe « Chiffre » pour assurer le bon fonctionnement de la fonction « main » ci-dessus.