IFT1166 - Session Automne, Final

Mohamed Lokbani

IFT1166 - Final

Inscrivez tout de suite votre nom et code permanent.

Nom: _____| Prénom(s): _____|

Signature: _____| Code perm: _____|

Date : 13 décembre 2003

Durée: 2 heures 45 minutes (de 12h30 à 15h15) Local: N-515 du Pavillon Principal (P.P).

Ce document contient 11 pages.

Directives:

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

- Répondre directement sur le questionnaire.
- Les réponses **doivent être clairement présentées.**

1._____/20 (1.1 ; 1.2 ; 1.3 ; 1.4 ; 1.5)

2._____/17 (2.1)

3._____/15 (3.1)

4._____/20 (4.1 ; 4.2 ; 4.3)

5._____/28 (5.1 ; 5.2 ; 5.3 ; 5.4 ; 5.5 ; 5.6)

Total:_____/100

Question 1 (20 points)

1.1 Nommer deux types de fichiers qu'on peut utiliser lorsqu'on effectue des opérations d'entrées et de sorties.

1.2 Quelle est l'opération la plus importante lorsqu'on a fini de lire ou d'écrire dans un fichier? Qu'arrive-t-il si on omet cette opération?

1.3 Soit l'entrée suivante sur cin (ce que vous allez taper sur le clavier):

(espace)	E	x	a	m	e	n	(espace)	F	i	n	a	l	(espace)	I	F	T	1	1	6	6	(return)

Tout en expliquant votre réponse, quel sera l'affichage en sortie après l'exécution des instructions suivantes:

```
char Phrase[25];
cin >> Phrase;
cout << Phrase;
```

1.4 Tout en expliquant votre réponse, quel est l'effet de l'instruction suivante?

```
cout << setw(3) << setprecision(2) << 3.14159265358979 << endl;
```

1.5 Soient les déclarations suivantes:

```
class A {
private:
       int prive_a;
protected:
       int protege_a;
public:
       int public_a;

};

class B : private A {
       ..............
};

class C : protected A {
       ..............
};

class D : public A {
       ..............
};
```

Tout en expliquant votre réponse, préciser les droits d'accès aux classes dérivées B, C et D de chacune des données membres de la classe de base A.

Question 2 (17 points) Le fichier `NOTE.DAT` contient les notes d'un étudiant. Chaque ligne compte 5 informations : le nom et le prénom de l'étudiant, sa note au quiz 1 sur 20 (pondération 25%), sa note au quiz 2 sur 20 (pondération 25%) et sa note à l'examen final sur 20 (pondération 50%).

Écrire un programme qui demande à l'usager le nom du fichier de résultat et qui écrira dans celui-ci le nom de l'étudiant suivi de sa moyenne sur 20.

Voici un exemple du fichier NOTE.DAT:

Michel Albert 14.7 15.6 14.9

Prénom Max 20 caractères	espace	Nom Max 20 caractères	tabulation	float	tabulation	float	tabulation	float

La moyenne de Michel est calculée comme suit :

$$Moy = 14.7x0.25 + 15.6x0.25 + 14.9x0.5 = 15.0$$

Le fichier de résultat qui doit être fourni par l'usager aura la forme suivante :

Michel Albert 15.0

Prénom Max 20 caractères	espace	Nom Max 20 caractères	tabulation	float

Question 3 (15 points) Soit les fragments d'un programme :

```
enum COULEUR {PIQUE, COEUR, CARREAU, TREFLE, INCONNUE};

class UneCarte {
public:
    int m_valeur;
    COULEUR m_couleur;
    void initialise(int v = 0, COULEUR c = INCONNUE);
    bool peutAllerSur(UneCarte uneAutre);
};
void UneCarte::initialise(int v, COULEUR c){
    m_valeur = v;
    m_couleur = c;
}
```

La fonction peutAllerSur() doit renvoyer une valeur booléenne indiquant si la carte au titre de laquelle elle est invoquée peut être placée sur celle qui lui est transmise comme paramètre en respectant les règles suivantes :

- une carte ne peut être placée sur une autre que si elles sont de la même COULEUR ;
- les cartes doivent s'empiler dans l'ordre de valeur croissant (pour simplifier, les valets, dames et rois sont respectivement représentés par des cartes de valeur 11, 12 et 13).

Le programmeur chargé de définir cette fonction a produit le code suivant :

```
bool UneCarte::peutAllerSur(UneCarte uneAutre){

    if (uneAutre.m_couleur == m_couleur)
        return true;
    if (uneAutre.m_valeur == m_valeur - 1)
        return true;
    return false;
}
```

3.1 Pouvez-vous corriger la logique de cette fonction de façon à ce qu'elle produise le résultat qu'on attend d'elle ?

Question 4 (20 points) Soit la classe Test définie comme suit:

```
class Test {
public:
       double valeur;
       Test* ptr;
       Test* suivant();
       void pointeSur(T *arg);
};
```

4.1 Comment peut-on définir la fonction `pointeSur` pour qu'elle stocke l'adresse qui lui est transmise dans le membre `ptr` de l'instance au titre de laquelle elle est invoquée?

4.2 Comment peut-on définir la fonction membre `suivant` pour qu'elle renvoie l'adresse contenue dans le membre `ptr` de l'instance au titre de laquelle elle est invoquée?

4.3 Ecrire un fragment de code qui crée un tableau de 100 instances de `Test` et utilise la fonction `pointeSur` pour stocker dans le membre `ptr` de chacun des 99 premiers éléments du tableau l'adresse de l'élément suivant (Le membre `ptr` du 100e élément pourrait recevoir une valeur nulle).

Question 5 (28 points) Soit le fichier Exo5.cpp, qui est syntaxiquement correct:

```
#include <iostream>

using namespace std;

class Point {
    float x,y;

public :
    Point(float a=0, float b=0) {
        x=a;
        y=b;
        cout << "contructeur 1 : "<< x <<","<< y <<"\n";
    }
    Point(Point & p){
        x=p.x;
        y=p.y;
        cout << "contructeur 2 : "<< x <<","<< y <<"\n";
    }
    ~Point(){
        cout << "destructeur : "<< x <<","<< y <<"\n";
    }
    float getX() {
        return x;
    }
    float getY() {
        return y;
    }
    void setX(float z){
        x=z;
    }
    void setY(float z){
        y=z;
    }

    // Déplace le point à droite horizontalement d'une distance d
    void deplacerADroite(float d) throw (float) {
        if (d>=0) {
            x=x+d;
        }
        else {
            throw (d); // si d<0 il ne s'agit pas d'un déplacement à droite !
        }
    }
    void afficher(){
        cout << "(" <<x <<"," <<y <<")\n";
    }
    friend char * compareHauteur(Point,Point);
};
char * compareHauteur(Point a,Point b) {
    if (a.y<b.y) {
        return " le premier est plus bas \n";
    }
    else if (a.y>b.y){
        return " le second est plus bas \n";
    }
    else {
        return " les deux sont à la même hauteur \n";
    }
}
```

5.1 À quoi servent les fonctions `getX`, `getY`, `setX` et `setY`?

5.2 A quoi servent les fonction membres **`Point(Point & p)`** et **`~Point()`**? Sont-elles nécessaires dans le cas de la classe `Point`? Pourquoi?

5.3 Que signifie le mot clé **`friend`** devant le prototype de la fonction **`compareHauteur`**? À quoi cela sert-il?

5.4 On considère la classe Polygone suivante (un polygone est caractérisé par son nombre de sommets et par ses sommets, stockés dans un tableau). On utilise la classe Point définie précédemment et l'on suppose que les deux classes sont définies dans un même fichier.

```
class Polygone {
   int nb; // nombre de sommets
   Point* somt; //tableau des sommets
public :
   Polygone(int n=0){
     nb=n;
     somt=new Point[nb];
   }
   void afficher(){
     for (int i=0;i<nb;i++) {
         somt[i].afficher();
     }
   }
};
```

Dites ce qui s'affiche à la suite des deux instructions suivantes :

```
Polygone triangle(3);
triangle.afficher();
```

5.5 Ecrivez la surcharge de l'opérateur d'indexation [ ] qui retourne le ième Point du polygone. Cet opérateur lèvera une exception du type entier qui renverra l'indice demandé quand celui-ci n'est pas valide (i.e. est négatif ou supérieur au nombre de sommets).

5.6 Définir la méthode `input` membre de la classe Polygone qui permet de mettre à jour les abscisses (X) et les ordonnées (Y) des différents sommets de ce polygone.

BONNES FÊTES !