

**UQÀM** Université du Québec à Montréal

**INF-3105**, Structures de données et algorithmes  
— 2009

**Examen intra**

— 2009

CONSIGNES

- **Les règlements de l'UQAM concernant le plagiat seront strictement appliqués.**
- Aucune sortie n'est permise durant l'examen.
- Il est important de bien expliquer vos choix s'il y a lieu.
- Aucun document n'est permis.
- La durée de l'examen est de 3 heures.
- Vous pouvez utiliser les versos comme brouillon ou comme espace supplémentaire.
- **Il est interdit de dégrafer le questionnaire.**
- Les téléphones cellulaires, calculatrices, ordinateurs, palm, baladeurs, iPods, etc. sont interdits.

IDENTIFICATION

NOM : \_\_\_\_\_

PRÉNOM : \_\_\_\_\_

CODE PERMANENT : \_\_\_\_\_

SIGNATURE : \_\_\_\_\_

GROUPE : \_\_\_\_\_

PROFESSEUR : \_\_\_\_\_

#1 \_\_\_\_\_ / 0

#2 \_\_\_\_\_ / 0

#3 \_\_\_\_\_ / 0

#4 \_\_\_\_\_ / 0

#5 \_\_\_\_\_ / 0

#6 \_\_\_\_\_ / 0

#7 \_\_\_\_\_ / 0

\_\_\_\_\_  
TOTAL

\_\_\_\_\_ / 0

commentaire :

---

**Numéro 1. (0 pts)**

**Objectif(s) :**

- Application des connaissances.
- Constructeur, Destructeur.

**Question :** Que va afficher le programme suivant :

```
#include <iostream>

using namespace std;

class G {
public :
    G(){ cout << "G" << endl; };
    virtual ~G(){ cout << "-G" << endl; };
};

class H : public G {
public :
    H(): G() { cout << "H" << endl; };
    virtual ~H(){ cout << "-H" << endl; };
};

class K {
public :
    K(){ cout << "K" << endl; };
    virtual ~K(){ cout << "-K" << endl; };
};

class F : public H, public K {
public :
    F(): K(), H() { cout << "F" << endl; };
    virtual ~F(){ cout << "-F" << endl; };
};

void t() {
    F f;
}

int main() {
    t();
    return 0;
}
```

---

**Numéro 2. (0 pts)****Objectif(s) :**

- Application des connaissances.
- Surcharge d'opérateur.

**Question :** Soit la déclaration suivante :

```
class NombreComplexe
private :
    double reelle;
    double imaginaire;
public :
    NombreComplexe( double a_reelle = 0.0, double a_imaginaire = 0.0 ) :
        reelle( a_reelle ), imaginaire( a_imaginaire ) ;
    virtual ~NombreComplexe();
;
```

Écrivez le code qui permet de surcharger l'opérateur de multiplication (\*) afin de multiplier deux nombres complexes sachant que  $re = x1.re * x2.re - x1.im * x2.im$  et  $im = x1.re * x2.im + x1.im * x2.re$ .

---

Numéro 3. (0 pts)

Objectif(s) :

- Synthèse de la matière.
- Fonction amie.

**Question :** Expliquez le rôle du mot **friend** lorsqu'il est appliqué à une fonction.

---

#### Numéro 4. (0 pts)

Objectif(s) :

- Application des connaissances.
- Utilisation des pointeurs.

**Question :** Dites ce que va afficher le code suivant :

```
#include <iostream>

using namespace std;

class C {
public :
    C(int a_a = 0) : a(a_a) {}
    C(const C & a_c) : a( a_c.a ) {}
    virtual ~C(){}
    C * f(){
        C * r = this;
        r++;
        return r;
    }
    friend ostream & operator<<( ostream &, C & );
private :
    int a;
};

ostream & operator<<( ostream & a_cout, C & a_c ){
    a_cout << a_c.a;
    return a_cout;
}

int main() {
    C t[10];
    int i = 0;
    for( i = 0; i < 10; i++ ){
        t[i] = C( 10 - i );
    }

    C * k = t + 3;
    cout << *k;
    *( k + 1 ) = *( k - 1 );
    cout << k[1];
    k = k->f();
    cout << k[0];

    return 0;
}
```

---

**Numéro 5. (0 pts)**

**Objectif(s) :**

- **Synthèse de la matière.**
- **Classe Abstraite.**

**a) (0 pts)** Expliquez ce qu'est une méthode abstraite pure.

**b) (0 pts)** Expliquez ce qu'est une classe abstraite.

**c) (0 pts)** Quel est l'utilité d'avoir des classes abstraite pure ?

---

**Numéro 6. (0 pts)****Objectif(s) :**

- **Application des connaissances.**
- **Programmation C++.**

**Question :** Le code suivant contient deux (2) classes. Complétez le en ajoutant le code pour les constructeurs par défaut et le code pour les constructeurs de copies.

Note: L'entête des constructeurs par défaut est déjà incluse.

```
class Forme2D {
public:
    Forme2D (double x = 0, double y = 0);
    virtual double surface () = 0;
protected:
    double centre_x;
    double centre_y;
};

class Cercle : virtual public Forme2D {
public:
    Cercle (double x = 0, double y = 0, double r = 0);
    double surface ();
protected:
    double rayon;
};
```

---

**Numéro 7. (0 pts)****Objectif(s) :**

- Application des connaissances.
- Programmation C++.

**Question :** Soit les classes **Forme2D** et **Cercle** déclarées au numéro précédant, et la classe **Forme3D** suivante:

```
class Forme3D : virtual public Forme2D {
public:
    Forme3D (double x = 0, double y = 0, double z = 0);
    virtual double volume () = 0;
protected:
    double centre_z;
};
```

Écrivez le code pour les deux classes concrètes (non abstraites) **Cylindre** et **Sphere**. Ces classes doivent contenir un constructeur par défaut. Les classes **Cylindre** et **Sphere** doivent être dérivées des classes **Cercle** et **Forme3D**.

	Cylindre	Sphere
surface	$2\pi r h + 2\pi r^2$	$4\pi r^2$
volume	$\pi r^2 h$	$\frac{4\pi r^3}{3}$

où  $h$  est la hauteur du cylindre et  $\pi \approx 3.1415926$ .