Module/Elément : POO-C++ Nom de l'enseignant : Youssef Baddi

TP 3

Exercise 1

On souhaite réaliser une classe vecteur3d permettant de manipuler des vecteurs é trois composantes. On prévoit que sa déclaration se présente ainsi :

```
class vecteur3d
{
  float x, y, z ; // pour les 3 composantes (cartÃ©siennes)
  .....
};
```

On souhaite pouvoir déclarer un vecteur, soit en fournissant explicitement ses trois composantes, soit en en fournissant aucune, auquel cas le vecteur créé possédera trois composantes nulles. écrire le ou les constructeurs correspondants :

- en utilisant des fonctions membre surdéfinies ;
- en utilisant une seule fonction membre ;
- en utilisant une seule fonction en ligne.

Exercise 2

Soit une classe vecteur3d définie comme suit :

```
class vecteur3d
{ floatx,y,z;
public :
vecteur3d (float c1=0.0, float c2=0.0, float c3=0.0)
{ x = c1 ; y = c2 ; z = c3 ;
} .....
};
```

Introduire une fonction membre nommée coincide permettant de savoir si deux vecteurs ont les mémes composantes :

- en utilisant une transmission par valeur ;
- en utilisant une transmission par adresse ;
- en utilisant une transmission par référence.

Si v1 et v2 désignent 2 vecteurs de type vecteur3d, comment s'écrit le test de coéncidence de ces 2 vecteurs, dans chacun des 3 cas considérés ?

Exercise 3

Comment concevoir le type classe chose de faéon que ce petit programme :

```
main()
{ chose x ;
cout << "bonjour\n" ; }
```

fournisse les résultats suivants :

```
crÃ©ation objet de type chose
bonjour
destruction objet de type chose
```

Que fournira alors l'exécution de ce programme (utilisant le méme type chose) :

```
main()
{ chose * adc = new chose }
```

Exercise 4

Quels seront les résultats fournis par l'exécution du programme suivant (ici, la déclaration de la classe demo, sa définition et le programme d'utilisation ont été regroupés en un seul fichier) :

```
#include <iostream>
using namespace std ;
class demo
{ intx,y;
public :
demo (int abs=1, int ord=0)
{x=abs; y=ord; cout << "constructeur I : " << x << " " << y << "\n" ;
}
demo (demo &) ;
~demo () ;
};

demo::demo (demo & d)
{ cout << "constructeur II (recopie) : " << d.x << " " << d.y << "\n"
   ;
x=d.x; y=d.y; }
demo::~demo ()
{ cout<<"destruction :"<<x<<""<<y<<"\n";
}

main () {
void fct (demo, demo *) ;
cout << "debut main\n" ;
demo a ;
demo b = 2 ;
demo c = a ;
demo * adr = new demo (3,3) ;

// proto fonction independante fct
fct (a, adr) ;
demo d = demo (4,4) ;
c = demo (5,5) ;
cout << "fin main\n" ;
}
void fct (demo d, demo * add) {
cout << "entreee fct\n" ;
delete add ;
cout << "sortie fct\n" ;
}
```

Exercise 5

Créer une classe point ne contenant qu'un constructeur sans arguments, un destructeur et un membre donnée privé représentant un numéro de point (le premier créé portera le numéro 1, le suivant le numéro 2...). Le constructeur affichera le numéro du point créé et le destructeur affichera le numéro du point détruit. écrire un petit programme d'utilisation créant dynamiquement un tableau de 4 points et le détruisant.

Exercise 6

1. Réaliser une classe nommée set_int permettant de manipuler des ensembles de nom- bres entiers. On devra pouvoir réaliser sur un tel ensemble les opérations classiques suivantes : lui ajouter un nouvel élément, connaétre son cardinal (nombre d'éléments), savoir si un entier donné lui appartient.

Ici, on conservera les différents éléments de l'ensemble dans un tableau alloué dynamiquement par le constructeur. Un argument (auquel on pourra prévoir une valeur par défaut) lui précisera le nombre maximal d'éléments de l'ensemble.

2. écrire, en outre, un programme (main) utilisant la classe set_int pour déterminer le nombre d'entiers différents contenus dans 20 entiers lus en données.

3. Que faudrait-il faire pour qu'un objet du type set_int puisse étre transmis par valeur, soit comme argument d'appel, soit comme valeur de retour d'une fonction ?

Exercise 7

Soit la classe point suivante :

```
1 class point
2 { int x,y;
3 public :
4 point (int abs=0, int ord=0)
5 { x = abs ; y = ord ;
6 } };
```

écrire une fonction indépendante affiche, amie de la classe point, permettant d'afficher les coordonnées d'un point. On fournira séparément un fichier source contenant la nouvelle déclaration (définition) de point et un fichier source contenant la définition de la fonction affiche. écrire un petit programme (main) qui crée un point de classe automatique et un point de classe dynamique et qui en affiche les coordonnées.

Exercise 8

Créer deux classes (dont les membres donnée sont privés) :

- l'une, nommée vect, permettant de représenter des vecteurs é 3 composantes de type double ; elle comportera un constructeur et une fonction membre d'affichage ;
- l'autre, nommée matrice, permettant de représenter des matrices carrées de dimen- sion 3 x 3 ; elle comportera un constructeur avec un argument (adresse d'un tableau de 3 x 3 valeurs) qui initialisera la matrice avec les valeurs correspondantes.

Réaliser une fonction indépendante prod permettant de fournir le vecteur correspondant au produit d'une matrice par un vecteur. écrire un petit programme de test. On fournira séparé- ment les deux déclarations de chacune des classes, leurs deux définitions, la définition de prod et le programme de test.

Exercise 9

Soit une classe vecteur3d définie comme suit :

```
class vecteur3d
{ float x,y,z;
public :
vecteur3d (float c1=0.0, float c2=0.0, float c3=0.0)
{ x = c1 ; y = c2 ; z = c3 ;
} };
```

Définir les opérateurs == et != de maniére qu'ils permettent de tester la coïncidence ou la non-coïncidence de deux points : a. en utilisant des fonctions membre ;
b. en utilisant des fonctions amies.

Exercise 10

Soit la classe vecteur3d ainsi définie :

```
class vecteur3d
{ float x,y,z;
public :
vecteur3d (float c1=0.0, float c2=0.0, float c3=0.0)
{ x = c1 ; y = c2 ; z = c3 ;
} };
```

Définir l'opérateur binaire + pour qu'il fournisse la somme de deux vecteurs, et l'opérateur binaire * pour qu'il fournisse le produit scalaire de deux vecteurs. On choisira ici des fonctions amies.

Exercise 11

Soit la classe point suivante :

```
class point
{ int x,y;
public :
point (int abs=0, int ord=0)
{ x = abs ; y = ord ;
}
// .....
};
```

a. La munir d'un opérateur de cast permettant de convertir un point en un entier (corres- pondant à son abscisse).

b. Soient alors ces déclarations :

```
point p ;
int n ;
void fct (int) ;
```

Que font ces instructions : n = p ; // instruction 1 fct (p) ; // instruction 2

Exercise 12

Quels résultats fournira le programme suivant :

```
#include <iostream>

using namespace std;

class point
{ int x,y;
public :
point (int abs, int ord)
{ x = abs ; y = ord ; }
operator int() // "cast" point --> int
{cout<<"***appelint()pourlepoint"<<x<<""<<y<<"\n";
return x ; }
};
main()
{ point a(1,5), b(2,8) ;
int n1, n2, n3 ;
n1 = a + 3 ; // instruction 1
cout << "n1 = " << n1 << "\n" ;
n2 = a + b ; // instruction 2

 cout << "n2 = " << n2 << "\n" ;
double z1, z2 ;
z1 = a + 3 ; // instruction 3
cout << "z1 = "<< z1 << "\n" ;
z2 = a + b ; // instruction 4
cout << "z2 = " << z2 << "\n" ;
}
```

Exercise 13

Soient les deux classes suivantes :

```
class B
{ //...
public :
  B () ;// constructeur sans argument
    B (int) ; // constructeur Ã  un argument
// ...
};
class A
{ //...
friend operator + (A, A) ;
public :
A () ;  // constructeur sans argument
A (int) ;    // constructeur Ã  un argument entier
A (B) ;  // constructeur Ã  un argument entier de type B
// ... };
```

a. Dans un programme contenant les déclarations :

```
A a1, a2, a3 ;
B b1, b2, b3 ;
```

les instructions suivantes seront-elles correctes et, si oui, que feront-elles ?

```
a1 = b1 ; // instruction 1
b1 = a1 ; // instruction 2
a3=a1+a2; //instruction3
a3=b1+b2; //instruction4
b3=a1+a2; //instruction5
```

b. Comment obtenir le méme résultat sans définir, dans A, le constructeur A(B) ?

Exercise 14

On dispose d'un fichier nommé point.h contenant la déclaration suivante de la classe point :

```
class point
{ float x,y;
public :
void initialise (float abs=0.0, float ord=0.0)
{ x = abs ; y = ord ;
}
void affiche ()
{ cout << "Point de coordonnÃ©es : " << x << " " << y << "\n" ;
}
float abs () { return x ; } float ord () { return y ; }
};
```

— Créer une classe pointb, dérivée de point comportant simplement une nouvelle fonc- tion membre nommée rho, fournissant la valeur du rayon vecteur (premiére coordonnée polaire) d'un point.
— Méme question, en supposant que les membres x et y ont été déclarés protégés (protected) dans point, et non plus privés.
— Introduire un constructeur dans la classe pointb.
— Quelles sont les fonctions membre utilisables pour un objet de type pointb ?

Exercise 15

Quels seront les résultats fournis par ce programme :

```
#include <iostream>
using namespace std ;
class A
{ int n;
float x ;
public :
A (int p = 2) {n=p;x=1;
cout << "** construction objet A : " << n << " " << x << "\n" ; }
};
class B
{ int n;
float y ;
public :
B (float v = 0.0) {n=1; y=v;
cout << "** construction objet B : " << n << " " << y << "\n" ; }
};
class C : public B, public A {
int n;
int p ;
```

```
public :
C (int n1=1, int n2=2, int n3=3, float v=0.0) : A (n1), B(v) { n = n3
   ; p = n1+n2 ;
cout << "** construction objet C : " << n << " " << p <<"\n" ; }
};
main()
{ C c1 ;
C c2 (10, 11, 12, 5.0) ;
}
```

Exercise 16

Méme question que précédemment, en remplaçant simplement l'en-téte du constructeur de C par :

```
C (int n1=1, int n2=2, int n3=3, float v=0.0) : B(v)
```

Exercise 17

Écrire un programme utilisant une classe rectangle dont le constructeur prend deux paramètres, largeur et hauteur et qui offre les fonctions suivantes avec cpp

- calcul du périmètre
- calcul de la surface
- affichage

ainsi que les accesseurs et mutateurs triviaux (lecture et modification de la largeur et de la hauteur).

Exercise 18

Une pile est un ensemble dynamique d'éléments où le retrait se fait d'une façon particulière. En effet, lorsque l'on désire enlever un élément de l'ensemble, ce sera toujours le dernier inséré qui sera retiré. Un objet pile doit répondre aux fonctions suivantes :

- Initialiser une pile
- Empiler un élément sur la pile (push)
- Dépiler un élément de la pile (pop)

pour cela nous allons supposer que les éléments à empiler sont de type int.

Le programme main comprend la définition d'une classe pile et un programme de test qui crée deux piles p1 et p2,empile dessus des valeurs entières et les dépiler pour vérifier les opérations push et pop.

Exercise 19

L'objectif de cet exercice est de gérer les notes des étudiants d'une école à l'aide d'une classe C++ Etudiant définie par :

Les attributs suivants :

- matricule : l'identifiant de l'étudiant (auto incrémenté)
- nom : nom d'un étudiant
- nbrNotes : le nombre de notes de l'étudiant
- *tabNotes : tableau contenant les notes d'un étudiant (allocation dynamique).

Les méthodes suivantes :

- Un constructeur d'initialisation
- Un constructeur avec arguments
- Un destructeur Etudiant ()
- Un constructeur de recopie Etudiant (const Etudiant &)
- Les getters et setters
- void saisie () : permettant la saisie des note d'un étudiant
- void affichage () : permettant l'affichage des informations d'un étudiant
- float moyenne () : retourne comme résultat la moyenne des notes de l'étudiant.
- bool admis () : retourne comme résultat la valeur true, si un étudiant est admis et la valeur false, sinon. Un étudiant est considéré comme étant admis lorsque la moyenne de ses notes est supérieure ou égale à 10.
- bool comparer() : qui compare la moyenne des deux étudiants, retourne comme résultat la valeur true, si deux étudiants ont la même moyenne et la valeur false, sinon.

Bon courage ! ! !