Nom : LATAPIE 11,5

Prénom : Florian

C++
Polytech Sophia
21 octobre 2022

On ne mentionne pas les directives `#include` *ou* `using namespace std`. *Sauf mention contraire, ne pas justifier les réponses. Respecter les emplacements prévus pour les réponses. Si on demande l'affichage produit ou une correction de bogue, répondre directement sur le code.*
Non respect d'une indication : -2

Exercice 1 (pemdas et alii) *Barrer ce qui est incorrect ou écrire ce qui est affiché.*

```
cout << 6 / 2*(1+2);   1
cout << 2 - 3 - 4;     -5
cout << 1 << "plus" << 2;   1plus2
cout << 1<2;
```

3

Exercice 2 (mvector) *Soit le programme suivant.*

```
1   class MVector { vector<double> v;
2       friend ostream &operator<<(ostream &os, const MVector &mv) {
3           for (int i=0; i<mv.taille(); i++) os << mv.v[i] << " ";
4           return os;
5       }
6   public:
7       MVector(double x, double y) { v[0]=x; v[1]=y; }
8       MVector(): v(3,3.3) { v[1]=5; }
9       int taille() const { return v.size(); }
10      class Bad_Dim { };
11      double &max() { int m=0;
12          for (int i=1; i<taille(); i++) if (v[m]<v[i]) m=i;
13          return v[m];
14      }
15  };
16
17  int main() { MVector mv;
18      mv.max() = 0; cout << mv;
19  }
```

a. Le constructeur à 2 arguments est-il correct ? non *Si non, corriger.*
v.push_back(x); v.push_back(y)

b. La fonction-membre `taille` *est déclarée* `const`. *Pourquoi ?*
car l'appeler ne modifie pas l'objet courant

3

c. L'opérateur `<<` *est déclaré avant le mot-clé* `public`. *Est-ce important ? Pourquoi ?*
non car le mot clé "friend" est placé avant

d. L'argument os *de l'opérateur << est passé par référence. Pourquoi ?*

cela permet de concaténer sur la même ligne plusieurs << car on utilise bien le même objet

e. L'argument mv *de l'opérateur << est passé par référence. Pourquoi ?*

si mv est très grand le passer par copie est long et inutile

f. Faut-il ajouter des parenthèses à l'expression `mv.v[i]` *(ligne 3) ? Pourquoi ?*

grâce à friend on peut accéder aux attributs privés de mv, l'opérateur [] est déjà défini dans vector, il n'y a pas besoin de ()

g. Ecrire ce qui est affiché.

3.3 8 0

h. Ecrire un constructeur de conversion qui convertit un flottant x *en un* `MVector` *de taille 1 dont le coefficient vaut* x.

```
MVector(const float& x) : v(1,x){}
```

i. Ecrire un opérateur de conversion qui convertit un `MVector` *de taille 1 en un flottant. L'exception* `Bad_Dim` *sera émise si la taille ne vaut pas 1.*

```
operator float(const MVector& mv){if(taille!=1){throw Bad_Dim();} return v[0];}
```

j. On suppose que l'on répondu aux deux questions précédentes et que l'on a écrit l'opérateur +, l'addition de deux `MVector`*. L'expression* `MVector(2.2)+3.3` *est-elle correcte ? Proposer éventuellement une correction.*

4,5

sachant qu'il y a un constructeur de conversion pour les 2 types, c'est ambigu, il faut en marquer un "explicit"

Exercice 3 (make heritage) *On considère le programme suivant.*

```
1   class A { public:
2       A() { cout << "A()"; }
3       A(int n) { cout << "A(int)"; }
4   };
5
6   class B: public A { int n; public:
7       B(): A(33) { cout << "B()"; }
8       B(int n) { cout << "B(int)"; }
9   };
10
11  int main() { B b1; B b2(12); }
```

a. Ecrire ce qui est affiché.

A(int); B(); B(int);

b. *Peut-on dire quelque-chose lorsque l'on compare l'encombrement mémoire d'un objet de type A et d'un objet de type B ?*

B hérite de A => Encombrement de la classe B ≥ encombrement de la classe A

c. *On suppose que l'on a respectivement déclaré et implémenté les classes* `A` *et* `B` *dans les fichiers* `a.h`, `a.cpp`, `b.h`, `b.cpp`. *La fonction* `main` *est dans le fichier* `main_ab.cpp`. *Proposer un* `Makefile` *qui permet de produire fichier exécutable* `main_ab` *en compilant séparément les fichiers sources.*

```
a.o : a.cpp a.h
    g++ -c a.cpp
b.o : b.cpp b.h a.h
    g++ -c b.cpp
main : a.o b.o main_ab.cpp -o main
    g++ -c main_ab.cpp
all : main
clean : rm *.o
```

1,0

Exercice 4 (amitié et héritage)

```
1   class A { int n; public:
2       A(): n(0) { }
3       friend void fonc(A theA) { cout << theA.n; }
4   };
5
6   class B: public A { };
7
8   int main()
9   {
10      B y;
11      fonc(y);
12  }
```

1

Le programme précédent est-il correct ? ... main ..... *Si oui, écrire ce qui est affiché.*

=> fonction n'(int x) n'existe pas

Exercice 5 (default values) *On propose la déclaration suivante.*

```
1   int fonc(int p=0, int q=0);
```

Combien de fonctions sont déclarées ? 1