LO43-2009A-MSO1-01.

utbm
université de technologie
Belfort-Montbéliard

LO43 : Programmation Orientée Objet - Médian Automne 2009

Durée : 2h. Documents non autorisés, Traducteurs autorisés, Les réponses se feront sur le sujet

NOM : Signature :

PRENOM :

1 Questions de cours :

Question 1 :

Qu'est ce que le polymorphisme ?

Réponse :

Question 2 :

Comment défini-t-on une méthode virtuelle pure ?
Qu'est ce que cela implique pour la classe ?

Réponse :

2 Exercice : Pre et Post

2.1 Profil

Considérons la classe vecteur vue en TD. Ecrivez le profil des opérateurs de post et de pré-incrémentation. Vous justifierez tous vos choix (type de retour, modes de passage,paramètres,...). *Pour rappel voici une partie du header de la classe vecteur.*

```
class Vecteur
{ private :
int taille;
float* v;

public :
vecteur(void);
vecteur(const vecteur&);

 vecteur(void);

vecteur& operator=(const vecteur&);
?
};
```

Réponse :

2.2 Contenu

Ecrivez l'implémentation de ces opérateurs.

Réponse :

3 Exercice : les opérateurs, ces faux amis ! !

Qu'affiche exactement le programme main.cpp (cf.annexe) ?

Réponse :

```
/*
 *  A.h
 *  medianL043A09
*/

class A {
protected:
    int b;
public :
    A();
    A(const A &);
    A(const int &);
    A& operator=(const A&);
    ~A();
    virtual A& operator+(const A&);
    virtual A operator-(const A&);
    A& operator*(const A&);
    int value();
};
```

```
/*
 *  A.cpp
*/

#include "A.h"
A::A():b(){};
A::A(const A & a):b(a.b){};
A::A(const int & a):b(a){};
A::~A(){};
A& A::operator=(const A& a){b=a.b; return *this;}
A& A::operator+(const A& a){A c(*this); b=(--c.b)-a.b; return *this;}
A A::operator-(const A& a){A c(a);  b+=(c.b--); c=*this+a;this->b--; return c;}
A& A::operator*(const A& a){A c=(*this)-a; b=(c.b++)+a.b; return *this;}
int A::value(){return b;}
```

```
/ *  B.h */
 #include "A.h"
class B : public A{
public:
    B();
    B(const B&);
    B(int i);
    ~B();
    B& operator=(const B&);
    B& operator+(const B&);
    B operator-(const B&);
};
```

```
/*   B.cpp */
#include "B.h"
B::B(){};
B::B(const B&){};
B::B(int i){b=i++;}
B::~B(){};
B& B::operator=(const B& a){A::operator=(a); return *this;};
B& B::operator+(const B& a){B c(*this); b+=(c.b)+a.b; return *this;}
B B::operator-(const B& a){B c(a); c=*this+a; this->b++; return c;}
```

```
/*  main.cpp */
#include <iostream>
#include "B.h"
int main (int argc, char * const argv[]) {
    A a,b(2),* pta = & a;
    B d(4),e(2);
    std::cout << e.value() <<" et "<< d.value() << " et " << b.value() << " et
" << a.value() << " et " << pta->value() << std::endl;

    A c=(a+b)-(a-b);
    std::cout << e.value() <<" et "<< d.value() << " et " << c.value() << " et
" << b.value() << " et " << a.value() << " et " << pta->value()<< std::endl;

    b=a+c;
    std::cout << e.value() <<" et "<< d.value() << " et " << c.value() << " et
" << b.value() << " et " << a.value() << " et " << pta->value()<< std::endl;

    c= a*(c-b);
    std::cout << e.value() <<" et "<< d.value() << " et " << c.value() << " et
" << b.value() << " et " << a.value() << " et " << pta->value()<< std::endl;

    b=*pta+c;
    std::cout << e.value() <<" et "<< d.value() << " et " << c.value() << " et
" << b.value() << " et " << a.value() << " et " << pta->value()<< std::endl;

    pta = &e;
    a =(A)  (d - *((B*) pta));
    std::cout << e.value() <<" et "<< d.value() << " et " << c.value() << " et
" << b.value() << " et " << a.value() << " et " << pta->value()<< std::endl;

    e = (d - d);
    std::cout << e.value() <<" et "<< d.value() << " et " << c.value() << " et
" << b.value() << " et " << a.value() << " et " << pta->value()<< std::endl;

    b = ((A) d - *((B*) pta));
    std::cout << e.value() <<" et "<< d.value() << " et " << c.value() << " et
" << b.value() << " et " << a.value() << " et " << pta->value()<< std::endl;

    e = (d - *((B*) pta));
    std::cout << e.value() <<" et "<< d.value() << " et " << c.value() << " et
" << b.value() << " et " << a.value() << " et " << pta->value()<< std::endl;

    return 0;}
```