Correction TD N° 1 Poesie L3 C++ 2011, FLIN 606

Vers.h

```
#ifndef vers_h
#define vers_h
class Vers
{
private:
  string suiteMots;
  string rime;
public:
  Vers();
  Vers(string s);
  Vers(string s, string r);
  virtual ~Vers();
  virtual string getSuiteMots()const;
  virtual void setSuiteMots(string sm);
  virtual string getRime()const;
  virtual void setRime(string r);
  virtual void saisie(istream& is);
  virtual void affiche(ostream& os)const;
};

ostream& operator<<(ostream&, const Vers&);
istream& operator>>(istream&, Vers&);
#endif
```

Vers.cc

```
#include<iostream>
#include<string>
using namespace std;
#include"Vers.h"


Vers::Vers(){}
Vers::Vers(string sm){suiteMots=sm;}
Vers::Vers(string sm, string r){suiteMots=sm;rime=r;}
Vers::~Vers(){cout <<"destructeur vers" <<endl;}

string Vers::getSuiteMots()const
{return suiteMots;}
void Vers::setSuiteMots(string sm)
{suiteMots=sm;}
string Vers::getRime()const
{return rime;}
void Vers::setRime(string r)
{rime=r;}

void Vers::saisie(istream& is)
{cout <<"vers puis rime" <<endl;is>>suiteMots>>rime;}

void Vers::affiche(ostream& os)const
{os<<"<<"<<suiteMots<<">>";}

ostream& operator<<(ostream& flotSortie, const Vers& v)
{v.affiche(flotSortie); return flotSortie;}

istream& operator>>(istream& flotEntree, Vers& v)
{v.saisie(flotEntree); return flotEntree;}
```

Strophe.h

```
#ifndef Strophe_h
#define Strophe_h
class Strophe
{
private:
  Vers ** suiteVers;
  int nbVers;
  virtual int* calculeSchemaCode(string schemaRimique);
public:
  Strophe();
  Strophe(const Strophe&);
  virtual ~Strophe();
  virtual void saisie(istream& is);
  virtual Vers* vers(int i)const;
  virtual void affiche(ostream& os)const;
  virtual Strophe& operator=(const Strophe&);
  virtual Vers*& operator[](int);
  virtual bool verifieSchema(string schemaRimique);
};

ostream& operator<<(ostream& flotSortie, const Strophe& s);
istream& operator>>(istream& flotEntree, Strophe& s);
#endif
```

Strophe.cc

```
#include<iostream>
#include<string>
using namespace std;
#include"Vers.h"
#include"Strophe.h"

Strophe::Strophe(){suiteVers=NULL; nbVers=0;}
Strophe::Strophe(const Strophe& autreStrophe)
{
        /*
        nbVers=autreStrophe.nbVers;
        suiteVers=new Vers*[nbVers];
        for (int i=0; i<nbVers; i++)
                suiteVers[i]=autreStrophe.suiteVers[i];
        */
        *this=autreStrophe;
}
Strophe::~Strophe(){if (suiteVers) delete[] suiteVers;}

Vers* Strophe::vers(int i)const
{if (i>=0 && i<nbVers) return suiteVers[i]; else return NULL;}

void Strophe::saisie(istream& is)
{
 if (suiteVers) delete[] suiteVers;
 cout << "Entrer le nombre de vers : " << endl;
 is>>nbVers; suiteVers = new Vers*[nbVers];
 for (int i=0; i<nbVers; i++)
   {
    Vers *v=new Vers(); v->saisie(is); suiteVers[i]=v;
   }
}
void Strophe::affiche(ostream& os)const
{
 for (int i=0; i<nbVers; i++)
   {
    suiteVers[i]->affiche(os); os << endl;
   }
}

Strophe& Strophe::operator=(const Strophe& s)
{
cout << "passage par affectation" << endl;
        if (this != &s) // pour le cas x=x
```

```
        {
                if (suiteVers) delete[] suiteVers; // effacer l'ancienne strophe
                nbVers=s.nbVers; // copier le champ entier
                suiteVers=new Vers*[nbVers]; // copie profonde du tableau de vers
                for (int i=0; i<nbVers; i++) // les vers restent partagŽs
                        suiteVers[i]=s.suiteVers[i];
        }
        return *this; // pour autoriser x=y=z
}

Vers*& Strophe::operator[](int i)
{return suiteVers[i];}

ostream& operator<<(ostream& flotSortie, const Strophe& s)
{s.affiche(flotSortie); return flotSortie;}

istream& operator>>(istream& flotEntree, Strophe& s)
{s.saisie(flotEntree); return flotEntree;}

int* Strophe::calculeSchemaCode(string schemaRimique)
{
 int *structAux=new int[nbVers];
 for (int i=0; i<nbVers; i++)
  structAux[i]=-1;
 for (int i=0; i<nbVers; i++)
  {
   if (structAux[i]==-1)
                for (int j=i+1; j<nbVers; j++)
                        if (schemaRimique[j]==schemaRimique[i]) structAux[j]=i;
  }
        //pour tester uniquement
        cout << "structure auxiliaire" << endl;
 for (int i=0; i<nbVers; i++)
  {
   cout << structAux[i] << " ";
  }
 cout << endl;
        return structAux;
}

bool Strophe::verifieSchema(string schemaRimique)
{
        if (nbVers != schemaRimique.length())
                return false;
 int *structAux=calculeSchemaCode(schemaRimique);
 for (int i=0; i<nbVers; i++)
 {
```

```
        if (structAux[i]!=-1 && !(suiteVers[i]->getRime()==(suiteVers[structAux[i]])-
>getRime()))
                return false;
        }
        return true;
}
```

Poesie.h

```
class Poesie
{
private:
        string genre;
        Strophe **suiteStrophes;
        int nbStrophes;
public:
        Poesie();
        Poesie(string g);
        Poesie(const Poesie&);
        virtual ~Poesie();
        virtual string getGenre()const;
        virtual void setGenre(string g);
        virtual void saisie(istream&);
        virtual void affiche(ostream&)const;
        virtual Poesie& operator=(const Poesie&);
};

ostream& operator<<(ostream&,const Poesie&);
istream& operator>>(istream&,Poesie&);
```

Poesie.cc

```
#include<iostream>
#include<string>
using namespace std;
#include"Vers.h"
#include"Strophe.h"
#include"Poesie.h"

Poesie::Poesie()
{
        nbStrophes=0; suiteStrophes=NULL;
}

Poesie::Poesie(string g)
{
        genre=g; nbStrophes=0; suiteStrophes=NULL;
}

Poesie::Poesie(const Poesie& p)
{
        *this=p;
}

Poesie::~Poesie()
{
if (suiteStrophes) delete[] suiteStrophes;
}

string Poesie::getGenre()const
{
        return genre;
}

void Poesie::setGenre(string g)
{
        genre=g;
}

void Poesie::saisie(istream& is)
{
 if (suiteStrophes) delete[] suiteStrophes;
  cout << "Entrer le nombre de strophes : " << endl;
  is>>nbStrophes; suiteStrophes = new Strophe*[nbStrophes];
  for (int i=0; i<nbStrophes; i++)
    {
      Strophe *s=new Strophe(); s->saisie(is); suiteStrophes[i]=s;
```

```
   }

}

void Poesie::affiche(ostream& os)const
{
 for (int i=0; i<nbStrophes; i++)
   {
     suiteStrophes[i]->affiche(os); os << endl;
   }

}

Poesie& Poesie::operator=(const Poesie& p)
{
if (this != &p) // pour le cas x=x
        {
                if (suiteStrophes) delete[] suiteStrophes; // effacer l'ancienne strophe
                nbStrophes=p.nbStrophes; // copier le champ entier
                suiteStrophes=new Strophe*[nbStrophes]; // copie profonde du tableau de
ptr de strophes
                for (int i=0; i<nbStrophes; i++) // les strophes restent partagŽes
                        suiteStrophes[i]=p.suiteStrophes[i];
        }
        return *this; // pour autoriser x=y=z

}

ostream& operator<<(ostream& flotSortie,const Poesie& p)
{
        p.affiche(flotSortie); return flotSortie;
}

istream& operator>>(istream& flotEntree,Poesie& p)
{
        p.saisie(flotEntree); return flotEntree;
}
```

mainPoesie.cc

```
#include<iostream>
#include<string>
using namespace std;
#include"Vers.h"
#include"Strophe.h"
#include"Poesie.h"

//Exemple cours sur les modes de passage de param□tres
void f(int fi, int* fpi, int &fri)
{
        fi++; // incremente fi, copie locale du premier parametre reel
        (*fpi)++; // incremente le contenu de la zone pointee par le deuxi□me param□  tre rŽel
        fri++; // incremente le troisi□me parametre reel
}

main()
{
/*
//Exemple cours sur les modes de passage de param□tres
        int i = 4;
        int *pi = new int;
        *pi = 4;
        int j=4;
        cout << "i=" << i << "  *pi=" << *pi << "  j=" << j << endl; // i=4  *pi=4  j=4
        f(i, pi, j);
        cout << "i=" << i << "  *pi=" << *pi << "  j=" << j << endl; // i=4  *pi=5  j=5

//Exemple cours sur les vers et les strophes
 Vers *pv1=new Vers();
 cout << typeid(*pv1).name() << endl;

 //pv1->saisie(cin);
 cin >> *pv1;
// pv1->affiche(cout);
 cout << *pv1;


 Vers *pv2=new Vers();
 pv2->saisie(cin);
 pv2->affiche(cout);  cout << endl;

  Vers *pv3=new Vers("vienne la nuit, sonne l'heure","eur");
  Vers *pv4=new Vers("les jours s'en vont, je demeure","eur");
  pv3->affiche(cout);  cout << endl;
  pv4->affiche(cout);  cout << endl;
```

```
  delete pv3;
*/
/*
  Strophe *s = new Strophe();
  //s->saisie(cin);
  cout << "saisie de s" << endl;
  cin >> *s;
  //s->affiche(cout);
  cout << *s;

  Strophe *s2 = new Strophe(*s);
  //*s2 = *s; // pas s2=s qui ne copierait que les pointeurs
  cout << "s2 apres copie de s dans s2" << endl;
  s2->affiche(cout);
  cout << "saisie de s2" << endl;
  s2->saisie(cin);
  cout << "s2 apres copie puis saisie" << endl;
  s2->affiche(cout);
  cout << "s apres copie dans s2" << endl;
  s->affiche(cout);

  int i;
  cout << "quel numero de vers dans s ?" << endl; cin >> i;
  if (s->vers(i))
                {s->vers(i)->affiche(cout); cout << endl;}
  else cout << "pas de vers a cet indice" << endl;


        Strophe S;
        S.saisie(cin); // hypothse : saisie de 8 vers
        cout << *S[0] << endl; // affiche le premier vers
        S[1]=new Vers("comme je descendais des fleuves impassibles","ible");
        cout << "nouvelle strophe" << endl << S << endl;

        Strophe *pS=new Strophe();
        pS->saisie(cin); // hypothse : saisie de 8 vers
        cout << *((*pS)[0]) << endl; // affiche le premier vers
        (*pS)[1]=new Vers("comme je descendais des fleuves impassibles","ible");
        cout << "nouvelle strophe" << endl << *pS << endl;


        Poesie *pp = new Poesie("ballade");
        cin >> *pp;
        cout << "poesie saisie" << endl << *pp;
        */
```

```
        Strophe *ppS=new Strophe();
        ppS->saisie(cin);
        cout << "Entrer le schema rimique a verifier" << endl;
        string schema;
        cin >> schema;
        cout << ppS->verifieSchema(schema) <<endl;

}
```