Unité de formation : Langage C++ Correction TP13 Formateur : A.Serghini

Chainep.h

```
//la classe chaine est nommée par Chainep
//fichier header classe mère

#ifndef CHAINEP_H
#define CHAINEP_H
#include <iostream>
using namespace std;
class Chainep
{
   public:
       Chainep();
       Chainep(char*);
       Chainep(const Chainep &);
       virtual ~Chainep();
       int taille() const;
      virtual  void affiche() const;
       void ajout(char, int);
       void supp(int);
       Chainep&  inv();
       bool appart(char );
      Chainep& operator=(const Chainep &);
      bool  operator==(const Chainep &);
      Chainep& operator+(const Chainep &);
      char  operator[](int);
      friend ostream& operator<<(ostream &, Chainep& );
      friend istream& operator>>(istream &, Chainep& );
   protected:
        char* adr;
};

#endif // CHAINEP_H
```

//fichier cpp classe Mère

Chainep.cpp

```
#include "chainep.h"
#include <iostream>
using namespace std;
Chainep::Chainep()
{
   adr=new char[20];
  adr[0]= '\0' ;
}
Chainep::Chainep(char* ch)
{  int l=0, i;
    while(ch[l]!='\0')  l++;
   adr=new char[l+1];
   for(i=0;i<=l;i++)     adr[i]=ch[i];
}
Chainep::Chainep(const Chainep &v)
{  int  i;
   adr=new char[v.taille()+1];
   for(i=0;i<=v.taille();i++)     adr[i]=v.adr[i];
   cout<<"*****constructeur par recopie*********"<<endl;
}
Chainep::~Chainep()
{
    delete [] adr;
}
int Chainep::taille() const
{  int l=0;
    while(adr[l]!='\0')  l++;
     return l;
 }
void Chainep::affiche() const
{ int i;
   for(i=0;i<taille();i++)       cout<<adr[i];
   cout<<endl;
}
```

```
void Chainep::ajout(char c, int  p)
{ int i;
 if(p>=0&&p<=taille())
  {    for(i=taille();i>=p;i--)       adr[i+1]=adr[i];
       adr[p]=c;
  }
}
void Chainep::supp(int  p)
{ int i;
 if(p>=0&&p<taille())
      for(i=p;i<=taille();i++)         adr[i]=adr[i+1];
 }
Chainep&  Chainep::inv()
{ static Chainep res;
   int i;
   for(i=0;i<taille();i++)     res.adr[i]=adr[taille()-1-i];
   res.adr[taille()]='\0';
   return res;
}
bool Chainep::appart(char c)
{ int i;
  for(i=0;i<taille();i++)      if(adr[i]==c)    return true;
  return false;
 }

Chainep &Chainep::operator=(const Chainep &v)
{  int  i;
  delete [] adr;
   adr=new char[v.taille()+1];
  for(i=0;i<=v.taille();i++)     adr[i]=v.adr[i];
  cout<<"*****operateur d'affectation classe mere*********"<<endl;
  return *this;
}
bool Chainep::operator==(const Chainep &v)
{  int  i;
  if(v.taille()==taille())
         {
          for(i=0;i<taille();i++)     if(adr[i]!=v.adr[i])    return false ;
          return true;
```

```
            }
   else return false;
}
Chainep &Chainep::operator+(const Chainep &v)
{  int  i;
   static Chainep res;
  delete [] res.adr;
 res.adr=new char[v.taille()+taille()+1];
   for(i=0;i<taille();i++)     res.adr[i]=adr[i];
    for(i=taille();i<=taille()+v.taille();i++)     res.adr[i]=v.adr[i-taille()];
   // res.adr[v.taille()+taille()]='\0';
   return res;
}
 char Chainep::operator[](int i)
 {
     return adr[i];
 }
ostream &operator<<(ostream &ostr, Chainep& v)
{ int i;
  for(i=0;i<v.taille();i++)         ostr<<v.adr[i];
  ostr<<endl;
  return ostr;
}
istream &operator>>(istream &istr, Chainep& v)
{
       istr>>v.adr;
  return istr;
}

//fichier header classe fille
```

Chainep_T.h

```
#ifndef CHAINEP_T_H
#define CHAINEP_T_H
#include "chainep.h"
class Chainep_T : public Chainep
{
   public:
```

```
        Chainep_T();
        Chainep_T(char * );
        virtual ~Chainep_T();
        Chainep_T(const Chainep_T& );
        Chainep_T& operator=(const Chainep_T& );
        virtual void affiche() const;
        void calcul();
      private:
        bool Type;
        float Val;
};

#endif // CHAINEP_T_H
```

//fichier cpp classe fille

Chainep_T.cpp

```
#include "chainep_t.h"
#include "chainep.h"
#include <iostream>
using namespace std;
Chainep_T::Chainep_T()
{
  Type=false; Val=0;
}
Chainep_T::Chainep_T(char * ch):Chainep(ch)
{
  Type=false; Val=0;
}

Chainep_T::Chainep_T(const Chainep_T& v):Chainep(v)
{
   Type=v.Type;
  Val=v.Val;
  cout<<"*****constructeur par recopie de la classe fille*********"<<endl;
}
Chainep_T::~Chainep_T()
```

```
{

}
Chainep_T& Chainep_T::operator=(const Chainep_T& v)
{
  int  i;
  delete [] adr;
   adr=new char[v.taille()+1];
  for(i=0;i<=v.taille();i++)     adr[i]=v.adr[i];
  Type=v.Type;
  Val=v.Val;
  cout<<"*****operateur d'affectation classe fille*********"<<endl;
  return *this;
}
void   Chainep_T::affiche() const
{
   Chainep::affiche();
   cout<<"Type= "<<Type<<endl;
   cout<<"Val= "<<Val<<endl;
}
void   Chainep_T::calcul()
{float x;
x=atof(adr);
 if(x!=0)   {Type=true;    Val=x;}
  else       {Type=false; }
}
```

main.cpp

```
#include "chainep.h"
#include "chainep_t.h"
#include <iostream>

using namespace std;

int main()
{
   Chainep a("je suis la");
```

```
cout<<a.taille()<<endl;
a.affiche();
a.ajout('c',10);
cout<<a.taille()<<endl;
a.affiche();
a.supp(10);
 cout<<a.taille()<<endl;
a.affiche();
a.inv().affiche();
if(a.appart('p')==true)    cout<<"existe"<<endl;
else    cout<<"n'existe pas"<<endl;
Chainep b=a;
b=a.inv();
if(a==b)   cout<<"egalite"<<endl;
         else   cout<<" non egalite"<<endl;
 Chainep c;
 c=a+b;
cout<< c.taille()<<endl;
 c.affiche();
 cout<<c[5]<<endl;
 cout<<a<<b<<c;
 Chainep d;
 cout<<"entrer la chaine d"<<endl;
 cin>>d;
 cout<<d;
 Chainep_T h("dgerget"),g=h,k("123");
 g=h;
 h.affiche();
 h.calcul();
 h.affiche();
  k.affiche();
 k.calcul();
 k.affiche();

 d=k;  teste affectation classe mère classe fille
 d.affiche();
 k.affiche();
Chainep *dd;// teste affectation classe mère classe fille pour les pointeurs
dd=new Chainep("zzzzz");
```

```
    Chainep_T  *kk;
    kk=new Chainep_T("lllllll");
    dd->affiche();
    kk->affiche();
    dd=kk;
    dd->affiche();
     return 0;
}
```