Compte Rendu TP2 CPOO

Théo CHAPON, Hassan EL OMARI ALAOUI

INSA de Rennes
4INFO, groupe 1.1

21 septembre 2014

TP1 : Conception de la classe chaine et surcharge

1 Etat du TP

Le projet est terminé, fonctionel et testé. Une incompréhension a été rencontrée lors de l'utilisation de l'attribut 'friend'. Mais celle-ci a été résolue après discution.

2 Objectif

Ce TP avait pour but d'effectuer de la surcharge d'opérateur. En plus de cela, nous avons pu travailler les pointeurs et la référence sur objet. Nous avons aussi pu tester les attributs 'friend' et 'inline'. Pour ce faire nous devions créer une classe chaine permettant de gérer une chaine de caractères ainsi que certaines opérations comme la concaténation.

3 Réalisation

Nous devions réaliser une classe permettant de gérer les chaines de caratères. Pour ce faire nous avons décidé d'utiliser un tableau de caratères comme attribut. Nous avons géré la longueur des chaines grâce au caractère de fin de chaine. Pour faciliter la gestion de la longueur, nous avons créé une méthode permettant de retourner la longueur d'une chaine.

Pour commencer, nous avons créé trois constructeurs permettant de créer une chaine par défault (contructeur vide), de créer une chaine de caractères à partir d'un tableau de caractères (pointeur sur char), de créer une chaine à partir d'une autre chaine (argument de type chaine passé par référence) et enfin un destructeur.

De plus, nous devions produire des méthodes permettant de gérer les chaines de caractères. Nous avions deux types de méthodes à traiter ; les méthodes de surcharge pour utiliser les opérateurs pour effectuer des comparaisons ($<$, $>$, $==$ etc.), pour faire une concaténation ($+$, $+=$) ou encore sélectionner un caractère indéxé de la chaine ([]). Nous avions aussi les méthodes normales comme les getters et les setters, les deux méthodes 'sous-chaine' dont la fonction était de récupérer une partie de la chaine à partir de deux indexes ou de deux caractères.

A cela nous avons aussi ajouté la surcharge au niveau de l'affichage d'une chaine (outstream : surcharge de l'opérateur «).

Listing 1 – Définition de la class chaine en C++

```
#ifndef CHAINE_H
#define CHAINE_H
#include "iostream"
using namespace std;
class Chaine {
  char* _chaine;
public:
  /**
   * \fn Chaine()
   * \brief Chaine constructor without parameters
   */
  Chaine();
  /**
   * \fn Chaine(const char* c)
   * \brief Chaine constructor with an array of character as parameter
   * \param[in] c array of character
   */
```

```
18    Chaine(const char* c);
19    /**
20     * \fn Chaine(const Chaine& c)
21     * \brief Chaine constructor with a parameter
22     * \param[in] c Chaine
23     */
24    Chaine(const Chaine& c);
25    /**
26     * \fn ~Chaine()
27     * \brief Chaine destructor
28     */
29    ~Chaine();
30    /**
31     * \fn char* getChaine() const
32     * \brief Getter of _chaine attribute
33     * \return char* _chaine
34     */
35    char* getChaine() const;
36    /**
37     * \fn int getLg() const
38     * \brief Return length of the string
39     * \return int
40     */
41    int getLg() const;
42    /**
43     * \fn Chaine& setChaine(const char* c)
44     * \brief Setter of _chaine attribute
45     * \param[in] c array of character
46     * \return Chaine&
47     */
48    Chaine& setChaine(const char* c);
49    /**
50     * \fn Chaine& setChaine(const Chaine& c)
51     * \brief Setter of _chaine attribute
52     * \param[in] c Chaine&
53     * \return Chaine&
54     */
55    Chaine& setChaine(const Chaine& c);
56    /**
57     * \fn ostream& print(ostream& os)
58     * \brief Display Chaine object
59     * \param[in,out] os output stream
60     * \return ostream& reference of output stream
61     */
62    ostream& print(ostream& os);
63    /**
64     * \fn Chaine sous_chaine(char deb, char fin) const
65     * \brief Return a substring of _chaine
66     * Return Chaine object created from substring of _chaine using deb
```

```
67    * as the first character and fin the last one
68    * \param[in] deb first character
69    * \param[in] fin last character
70    * \return Chaine
71    */
72   Chaine sous_chaine(char deb, char fin) const;
73   /**
74    * \fn Chaine sous_chaine(char deb, char fin) const
75    * \brief Return a substring of _chaine
76    * Return Chaine object created from substring of _chaine using ind1
77    * as the first index and ind2 the last one
78    * \param[in] ind1 first index
79    * \param[in] ind2 last index
80    * \return Chaine
81    */
82   Chaine sous_chaine(int ind1, int ind2) const;
83   /**
84    * \fn char& operator[](int i) const
85    * \brief Overloads subscripting operator
86    * Return the character that's in the index i given as parameter
87    * \param[in] i index
88    * \return char&
89    */
90   char& operator[](int i) const;
91   /**
92    * \fn friend bool operator==(const Chaine& c1, const Chaine& c2)
93    * \brief Overloads == operator
94    * Return true if c1 is equal to c2, false otherwise
95    * \param[in] c1 Chaine&
96    * \param[in] c2 Chaine&
97    * \return bool
98    */
99   friend bool operator==(const Chaine& c1, const Chaine& c2);
100  /**
101   * \fn friend bool operator>=(const Chaine& c1, const Chaine& c2)
102   * \brief Overloads >= operator
103   * Return true if c1 is superior or equal to c2, false otherwise
104   * \param[in] c1 Chaine&
105   * \param[in] c2 Chaine&
106   * \return bool
107   */
108  friend bool operator>=(const Chaine& c1, const Chaine& c2);
109  /**
110   * \fn friend bool operator<=(const Chaine& c1, const Chaine& c2)
111   * \brief Overloads <= operator
112   * Return true if c1 is less or equal to c2, false otherwise
113   * \param[in] c1 Chaine&
114   * \param[in] c2 Chaine&
115   * \return bool
```

```
116     */
117   friend bool operator<=(const Chaine& c1, const Chaine& c2);
118   /**
119    * \fn friend bool operator>(const Chaine& c1, const Chaine& c2)
120    * \brief Overloads > operator
121    * Return true if c1 is superior to c2, false otherwise
122    * \param[in] c1 Chaine&
123    * \param[in] c2 Chaine&
124    * \return bool
125    */
126   friend bool operator>(const Chaine& c1, const Chaine& c2);
127   /**
128    * \fn friend bool operator<(const Chaine& c1, const Chaine& c2)
129    * \brief Overloads < operator
130    * Return true if c1 is less to c2, false otherwise
131    * \param[in] c1 Chaine&
132    * \param[in] c2 Chaine&
133    * \return bool
134    */
135   friend bool operator<(const Chaine& c1, const Chaine& c2);
136   /**
137    * \fn friend bool operator+=(const Chaine& c1, const Chaine& c2)
138    * \brief Overloads += operator
139    * Return concatenation of c1 and c2
140    * \param[in,out] c1 Chaine&
141    * \param[in] c2 Chaine&
142    * \return Chaine&
143    */
144   friend Chaine& operator+=(Chaine& c1, const Chaine& c2);
145   /**
146    * \fn friend bool operator+(const Chaine& c1, const Chaine& c2)
147    * \brief Overloads + operator
148    * Return concatenation of c1 and c2
149    * \param[in,out] c1 Chaine&
150    * \param[in] c2 Chaine&
151    * \return Chaine*
152    */
153   friend Chaine* operator+(const Chaine& c1, const Chaine& c2);
154   /**
155    * \fn friend bool operator<<(const Chaine& c1, const Chaine& c2)
156    * \brief Overloads << operator
157    * Return output stream to display Chaine object
158    * \param[in,out] os ostream&
159    * \param[in] c Chaine&
160    * \return ostream&
161    */
162   friend ostream& operator<<(ostream& os, Chaine& c);
163 };
164 #endif
```

Listing 2 – Class chaine en C++

```
#include "stdafx.h"
#include "string.h"
#include "Chaine.h"

char* _copie(char* c) {
  char * chaine = (char*)malloc(sizeof(char)*(strlen(c) + 1));
  int i = 0;// c = 'a'
  do {
    chaine[i] = c[i];
  } while (c[i++] != '\0');
  return chaine;
}

Chaine::Chaine() : _chaine("")
{
  cout << "Constructeur1" << endl;
}

Chaine::Chaine(const char * c) {
  cout << "Constructeur2" << endl;
  _chaine = _copie((char*)c);
}

Chaine::Chaine(const Chaine& c) {
  cout << "Constructeur3" << endl;
  _chaine = c.getChaine();
}



Chaine::~Chaine() {
  cout << "Destructeur" << endl;
  delete _chaine;
}
char* Chaine::getChaine() const {
  return _chaine;
}

int Chaine::getLg() const {
  const char* s;
  for (s = _chaine; *s; ++s);
  return s - _chaine;
}
Chaine& Chaine::setChaine(const char* c) {
  _chaine = _copie((char*)c);
  return *this;
}
Chaine& Chaine::setChaine(const Chaine& c) {
```

```
49    _chaine = _copie(c.getChaine());
50    return *this;
51  }
52  Chaine Chaine::sous_chaine(char deb, char fin) const {
53    char* res;
54    bool flag = false;
55    int i,j;
56    res = (char*)malloc(2*sizeof(char));
57    for (i = 0, j = 0; _chaine[i] != '\0'; i++) {
58      if (_chaine[i] == deb) {
59        flag = true;
60      }
61      if (flag) {
62        res[j++] = _chaine[i];
63        res = (char*)realloc(res, (j+2)*sizeof(char));
64      }
65      if (_chaine[i] == fin) {
66        break;
67      }
68    }
69    res[j] = '\0';
70    return Chaine(res);
71  }
72  Chaine Chaine::sous_chaine(int ind1, int ind2) const {
73    char* res;
74    int lg = getLg();
75    if (ind1 > lg)
76      return Chaine();
77    int end = (ind2 > lg) ? lg : ind2;
78    res = (char*)malloc((end - ind1 + 1)*sizeof(char));
79    for (int i = ind1, j = 0; i < end; i++) {
80      res[j++] = _chaine[i];
81    }
82    res[end] = '\0';
83    return Chaine(res);
84  }
85  char& Chaine::operator[](int i) const {
86    return _chaine[i];
87  }
88
89  ostream& Chaine::print(ostream& os) {
90    return os << _chaine;
91  }
92
93
94  bool operator==(const Chaine& c1, const Chaine& c2) {
95    return (strcmp(c1._chaine, c2._chaine) == 0);
96  }
97  bool operator>=(const Chaine& c1, const Chaine& c2) {
```

```
98    return (strcmp(c1._chaine, c2._chaine) >= 0);
99  }
100
101 bool operator<=(const Chaine& c1, const Chaine& c2) {
102   return (strcmp(c1._chaine, c2._chaine) <= 0);
103 }
104
105 bool operator>(const Chaine& c1, const Chaine& c2) {
106   return (strcmp(c1._chaine, c2._chaine) > 0);
107 }
108
109
110 bool operator<(const Chaine& c1, const Chaine& c2) {
111   return (strcmp(c1._chaine, c2._chaine) < 0);
112 }
113
114
115 Chaine& operator+=(Chaine& c1, const Chaine& c2) {
116   int i = strlen(c1._chaine);
117   int j = 0;
118   do {
119     c1._chaine[i++] = c2._chaine[j];
120   } while (c2._chaine[j++] != '\0');
121   return c1;
122 }
123
124 Chaine* operator+(const Chaine& c1, const Chaine& c2) {
125   Chaine* res = new Chaine();
126   res->setChaine(c1);
127   *res += c2;
128   return res;
129 }
130
131 ostream& operator<<(ostream& os, Chaine& c) {
132   return c.print(os);
133 }
```