Compte Rendu TP2 CPOO

Maxime Havez, Gareth Thiveux

INSA de Rennes
4INFO, groupe 2.2

3 octobre 2010

Listing 1 – Fichier en-tête de la classe Chaine

```
1  /**
2
3  *
4
5  * \file chaine.h
6
7  * \brief Header file which describes the "chaine" class
8
9  * \author Maxime HAVEZ
10
11 * \author Gareth THIVEUX
12
13 * \date 30/09/10
14
15 *
16
17 */
18
19 #include <iostream>
20
21 #include <fstream>
22
23
24
25 #ifndef CHAINE_H
26
27 #define CHAINE_H
28
29
30
```

```
31 #define TAILLE_MAX 26
32
33
34
35 class Chaine {
36
37
38
39 private:
40
41   //Class Attributes
42
43   int _taille;
44
45   char* _tabchar;
46
47
48
49 public:
50
51
52
53   /**
54
55   * \fn inline Chaine Chaine::tabchar() const {return _tabchar;}
56
57   * \brief gives an access to the current string
58
59   *
60
61   * \return current string
62
63   */
64
65   inline Chaine Chaine::tabchar() const {return _tabchar;}
66
67
68
69   /**
70
71   * \fn inline Chaine Chaine::taille() const {return _taille;}
72
73   * \brief gives an access to the length of the current  string
74
75   *
76
77   * \return integer refering to the length of the current string
78
79   */
```

```
80
81  inline int Chaine::taille() const {return _taille;}
82
83
84
85  /**
86
87  * \fn Chaine();
88
89  * \brief String constructor with no paremeter
90
91  *
92
93  */
94
95  Chaine();
96
97
98
99  /**
100
101 * \fn Chaine(const char * c);
102
103 * \brief String constructor with one paremeter (char* type)
104
105 *
106
107 * \param[in] c Pointer on a constant char
108
109 */
110
111 Chaine(const char * c);
112
113
114
115 /**
116
117 * \fn Chaine(const Chaine & c);
118
119 * \brief String constructor with one paremeter (string type)
120
121 *
122
123 * \param[in] c Reference on a constant string
124
125 */
126
127 Chaine(const Chaine & c);
128
```

```
129
130
131   /**
132
133   * \fn chaine sous_chaine(char deb, char fin);
134
135   * \brief extracts the substring of the current string beginning by the "deb"
          char and finishing by the "fin" char
136
137   *
138
139   * \param[in] deb first char of the substring
140
141   * \param[in] fin last char of the substring
142
143   * \return the substring in between "deb" char and "fin" char
144
145   */
146
147   Chaine sous_chaine(char deb, char fin);
148
149
150
151   /**
152
153   * \fn chaine sous_chaine(int ind1, int ind2);
154
155   * \brief extracts the substring of the current string beginning at position
          "ind1" and finishing at position "ind2"
156
157   *
158
159   * \param[in] ind1 integer refering to the position of first char of the
          substring
160
161   * \param[in] ind2 integer refering to the position of last char of the substring
162
163   * \return the substring in between position ind1 and position ind2
164
165   */
166
167   Chaine sous_chaine(int ind1, int ind2);
168
169
170
171   /**
172
173   * \fn bool operator= (const Chaine& c) const;
174
```

```
175   * \brief affects the given string to the current one
176   *
177   * \param[in] c reference on a constant string
178   * \return a reference on the current string
179   */
180   Chaine& Chaine::operator= (const Chaine& c);
181
182
183   /**
184   * \fn bool operator== (const Chaine& c) const;
185   * \brief tests the equality of two strings
186   *
187   * \param[in] c reference on a constant string
188   * \return true if the two strings are equals, if not, false
189   */
190   bool operator== (const Chaine& c) const;
191
192
193   /**
194   * \fn bool operator> (const Chaine& c) const;
195   * \brief tests if the current string is superior to the given one
196   *
197   * \param[in] c reference on a constant string
198   * \return true if the current string is superior to the given one, if not, false
199   */
200   bool operator> (const Chaine& c) const;
201
202
203
```

```
224
225   /**
226
227   * \fn bool operator< (const Chaine& c) const;
228
229   * \brief tests if the current string is inferior to the given one
230
231   *
232
233   * \param[in] c reference on a constant string
234
235   * \return true if the current string is inferior to the given one, if not, false
236
237   */
238
239   bool operator< (const Chaine& c) const;
240
241
242
243   /**
244
245   * \fn bool operator>= (const Chaine& c) const;
246
247   * \brief tests if the current string is superior or equal to the given one
248
249   *
250
251   * \param[in] c reference on a constant string
252
253   * \return true if the current string is superior or equal to the given one, if
          not, false
254
255   */
256
257   bool operator>= (const Chaine& c) const;
258
259
260
261   /**
262
263   * \fn bool operator<= (const Chaine& c) const;
264
265   * \brief tests if the current string is inferior or equal to the given one
266
267   *
268
269   * \param[in] c reference on a constant string
270
271   * \return true if the current string is inferior or equal to the given one, if
```

```
    not, false
272
273 */
274
275 bool operator<= (const Chaine& c) const;
276
277
278
279 /**
280
281 * \fn Chaine operator+ (const Chaine& c) const;
282
283 * \brief concatenates the given string to the current one
284
285 *
286
287 * \param[in] c reference on a constant string
288
289 * \return the concatenated string
290
291 */
292
293 Chaine operator+ (const Chaine& c) const;
294
295
296
297 /**
298
299 * \fn Chaine charInd (int i);
300
301 * \brief gives the char of the current string situated at position i
302
303 *
304
305 * \param[in] i integer refering to a position in the current string
306
307 * \return the char situated at position i
308
309 */
310
311 char charInd (int i) const;
312
313
314
315 /**
316
317 * \fn chaine operator+= (const Chaine& c) const;
318
319 * \brief concatenates the given string to the current one
```

```
320
321   *
322
323   * \param[in] c reference on a constant string
324
325   * \return a reference on the current concatenated string
326
327   */
328
329   Chaine& operator+= (const Chaine& c);
330
331
332
333   /**
334
335   * \fn ~Chaine();
336
337   * \brief String destructor with one paremeter (char* type)
338
339   *
340
341   * \param[in] c Pointer on a constant char
342
343   */
344
345   ~Chaine();
346
347
348
349   /**
350
351   * \fn ~Chaine(const char * c));
352
353   * \brief String destructor  with one paremeter (char* type)
354
355   *
356
357   * \param[in] c Pointer on a constant char
358
359   */
360
361   ~Chaine(const char * c){
362
363
364
365   /**
366
367   * \fn ~Chaine(const Chaine & c){;
368
```

```
369   * \brief String destructor with one paremeter (string type)
370
371   *
372
373   * \param[in] c Reference on a constant string
374
375   */
376
377     ~Chaine(const Chaine & c){
378
379 };
380
381
382
383 #endif
```

Listing 2 – Classe Chaine

```
1  /**
2
3  *
4
5  * \file chaine.cpp
6
7  * \brief file of "chaine" class (function description)
8
9  * \author Maxime HAVEZ
10
11 * \author Gareth THIVEUX
12
13 * \date 30/09/10
14
15 *
16
17 */
18
19
20
21 #include "chaine.h"
22
23 #include <cstdlib>
24
25
26
27
28
29
30
31 // String constuctors
32
```

```
33 Chaine::Chaine(){
34
35   _taille = TAILLE_MAX;
36
37   _tabchar[_taille];
38
39   std::cout << "CONSTRUCTEUR : Chaine par dfaut !\n";
40
41 }
42
43
44
45 Chaine::Chaine(const char * c){
46
47   int i=0;
48
49   while(c[i]!='\0'){
50
51     _tabchar[i]=c[i];
52
53     i++;
54
55   }
56
57   _taille = i;
58
59   std::cout << "CONSTRUCTEUR : Chaine constitue d'un tableau de char !\n";
60
61 }
62
63
64
65 Chaine::Chaine(const Chaine & c){
66
67   int i=0;
68
69   for(i=0; i<c._taille; i++){
70
71     _tabchar[i]=c._tabchar[i];
72
73   }
74
75   std::cout << "CONSTRUCTEUR : Chaine constitue d'une chaine !\n";
76
77 }
78
79
80
81 Chaine Chaine::sous_chaine(char deb, char fin){
```

```
82
83    Chaine resul;
84
85    int i=0;
86
87    int j=1;
88
89    while(_tabchar[i]!=deb){
90
91      if(i>=_taille){
92
93        std::cerr << deb << "n'a pas t trouv dans la chane" << std::endl;
94
95        return(0);
96
97      }
98
99      i++;
100
101   }
102
103   resul._tabchar[0]=deb;
104
105   while(_tabchar[i]!=fin && i<_taille){
106
107     if(i>_taille){
108
109       std::cerr << fin << "n'a pas t trouv dans la chane" << std::endl;
110
111       return(0);
112
113     }
114
115     resul._tabchar[j]=_tabchar[i];
116
117     i++; j++;
118
119   }
120
121   resul._tabchar[j] = fin;
122
123   resul._taille = j+1;
124
125   return resul;
126
127 }
128
129
130
```

```
131 Chaine Chaine::sous_chaine(int ind1, int ind2){
132
133   Chaine resul;
134
135   if(((ind1 || ind2) > _taille) || (ind1 > ind2)){
136
137     std::cerr << ind1 << "ou" << ind2 << "n'a pas une valeur correcte" <<
          std::endl;
138
139     return(0);
140
141   }
142
143   int i=0;
144
145   int j=0;
146
147   for(i=ind1; i<=ind2; i++){
148
149     resul._tabchar[j]=_tabchar[i];
150
151     j++;
152
153   }
154
155   resul._taille = j;
156
157   return resul;
158
159 }
160
161
162
163 bool Chaine::operator== (const Chaine& c) const{
164
165   bool res=true;
166
167   if (_taille != c._taille) {res=false;}
168
169   else {
170
171     int i;
172
173     for(i=0;i<_taille;i++){
174
175       if (_tabchar[i] != c._tabchar[i]){res=false;}
176
177     }
178
```

```
179   }
180
181   return res;
182
183 }
184
185
186
187 Chaine& Chaine::operator= (const Chaine& c){
188
189   int i=0;
190
191   Chaine* tmp;
192
193   if(_taille>=c._taille){
194
195     for(i=0;i<c._taille;i++){
196
197       _tabchar[i] = c._tabchar[i];
198
199     }
200
201   }else{
202
203     tmp = new Chaine(c);
204
205     delete (this);
206
207   }
208
209   return *tmp;
210
211 }
212
213
214
215 bool Chaine::operator> (const Chaine& c) const{return _taille>c._taille;}
216
217
218
219 bool Chaine::operator< (const Chaine& c) const{return _taille<c._taille;}
220
221
222
223 bool Chaine::operator>= (const Chaine& c) const{return _taille>=c._taille;}
224
225
226
227 bool Chaine::operator<= (const Chaine& c) const{return _taille<=c._taille;}
```

```
228
229
230
231 Chaine Chaine::operator+ (const Chaine& c) const{
232
233   Chaine concat;
234
235   int i=0;
236
237   int j=0;
238
239   concat._taille = _taille + c._taille;
240
241   for(i=0;i<_taille;i++){
242
243     concat._tabchar[j]=_tabchar[i];
244
245     j++;
246
247   }
248
249   for(i=0;i<c._taille;i++){
250
251     concat._tabchar[j]=c._tabchar[i];
252
253     j++;
254
255   }
256
257   if(j != concat._taille){
258
259     std::cerr << "La chaine n'a pas une taille cohrente ! " << concat._taille <<
        "different de" << j <<std::endl;
260
261   }
262
263   return concat;
264
265 }
266
267
268
269 char Chaine::charInd (int i) const {
270
271   if(i>=_taille){
272
273       std::cerr << "Indice suprieur  la taille de la chane !" << std::endl;
274
275       return(0);
```

```
276
277     }
278
279   return _tabchar[i];
280
281 }
282
283
284
285
286
287 Chaine& Chaine::operator+= (const Chaine& c){
288
289   return *this = *this + c;
290
291 }
292
293
294
295 Chaine::~Chaine() {
296
297     delete _tabchar;
298
299 }
300
301
302
303 Chaine::~Chaine(const Chaine & c){
304
305         delete _tabchar;
306
307 }
308
309
310
311 Chaine::~Chaine(const char * c){
312
313         delete _tabchar;
314
315 }
```

Listing 3 – Main

```
1 /**
2
3 *
4
5 * \file main.cpp
6
7 * \brief Main class
```

```
8
9  * \author Maxime HAVEZ
10
11 * \author Gareth THIVEUX
12
13 * \date 30/09/10
14
15 *
16
17 */
18
19
20
21
22
23 #include "chaine.h"
24
25 #include <iostream>
26
27 #include <cstdlib>
28
29
30
31 using namespace std;
32
33
34
35 ostream& operator<< (ostream& os, const Chaine& c) {
36
37   int i=0;
38
39   for (i=0;i<c.taille();i++){
40
41     os <<c.charInd(i);
42
43   }
44
45   return os;
46
47 }
48
49
50
51 int main() {
52
53
54
55 Chaine * test = new Chaine("fonctionne");
56
```

```
57 cout << *test ;
58
59 Chaine * ch1 = new Chaine("c");
60
61 Chaine * ch2 = new Chaine("o");
62
63
64
65 std::cout << (*ch2 == *ch1) << std::endl;
66
67
68
69 //*ch1 += "on me voit !";
70
71
72
73 //*ch1 += * ch2;
74
75
76
77 //Chaine * ch3;
78
79
80
81 //std::cin >> *ch3;
82
83
84
85 }
86
87
88
89 /*
90
91 Rsultats du main :
92
93 CONSTRUCTEUR : Chaine constituee d'un tableau de char !
94
95 fonctionne
96
97
98
99 Le message "L'application TP2.exe a cess de fonctionner" apparait. Impossible
      d'aller plus loin dans l'execution de notre main ...
100
101 Notre constructeur  partir d'un tableau de char semble fonctionner.
102
103 Nous n'avons autrement pas russi  surcharger l'oprateur >>
104
```

```
105 */
```