```
1 #include "IntList.h"
 2 #include <iostream>
 4 IntList::IntList() {
        this->count = 0;
        this ->first = 0;
 7 };
 8
 9 IntList::IntList(const IntList &other) {
        this->count = 0;
this->first = 0;
10
11
12
        IntListElem *el:
13
        el = other first;
14
15
        while (el) {
             this->insert(el->value, -1);
16
             el = el->next;
19 };
20
21 IntList &IntList::operator=(const IntList &other) {
22
        if (this != &other) {
   while (this->getCount()) {
23
24
                  this->remove(0);
25
26
27
             IntListElem *el;
28
             el = other.first;
29
             while (el) {
30
                  this->insert(el->value, -1);
31
                  el = el->next;
32
             }
33
        return *this;
34
35 };
36
   IntList::~IntList() {
        if (this->getCount() > 0 && this->first != 0) {
             IntListElem *cur = this->first;
IntListElem *next = cur->next;
39
40
41
             while (next) {
42
                  cur->value = 0;
43
                  delete cur;
44
                  cur = next:
45
                  next = cur->next;
46
             }
47
        }
48 };
50 int IntList::getCount() {
        return this->count;
52 };
53
54 bool IntList::isEmptv() {
        return this->getCount() > 0;
55
56 };
57
   void IntList::print() {
60
        if (this->getCount()) {
             IntListElem *el = this->first;
std::cout << "[" << el->value;
61
62
63
             el = el->next;
             while (el) {
64
                  std::cout << "," << el->value;
65
                  el = el->next;
66
67
             std::cout << "]" << std::endl;
68
        } else {
70
             std::cout << "empty list" << std::endl;</pre>
71
72 };
73
   void IntList::insert(int element, int position) {
   if (position > 0 && position < this->getCount()) {
        IntListElem *elmToMod = this->first;
        for (int i = 0; i < position; i++) {</pre>
74
75
76
77
78
                  elmToMod = elmToMod->next;
80
             elmToMod->value = element;
81
        } else if (position == this->getCount()) {
82
             IntListElem *elmToInsert = new IntListElem;
83
             elmToInsert->value = element;
84
             elmToInsert->next = 0;
             IntListElem *elmAfter = this->first;
85
86
             if (position == 0) {
                  this->first = elmToInsert;
87
88
             } else {
                  while (elmAfter->next) {
89
90
                       elmAfter = elmAfter->next;
91
92
                  elmAfter->next = elmToInsert;
```

File - /home/christian/studium/ipi-ws2014/Z08/IntList.cpp

```
93
 94
                     this->count++:
             this->countrr,
} else if (position == -1) {
    this->insert(element, this->getCount());
 95
 96
 97
 98 };
 99
100 void IntList::remove(int position) {
101
             IntListElem *elToDel, *elPrev;
             intlistitum ettobet, ettrev,
etToDel = findElement(position);
if (position == 0) {
   if (elToDel) {
      this->first = elToDel->next;
      this->count--;
}
102
103
104
105
106
107
108
             } else if (position > 0 && position < this->getCount()) {
109
110
                     findElement(position - 1)->next = elToDel->next;
111
                     this->count--;
112
             delete elToDel;
113
114
115 };
116
117 int IntList::getElement(int position) {
118     IntListElem *el = findElement(position);
119     return el ? el->value : 0;
120 };
121
121
122 IntList::IntListElem *IntList::findElement(int position) {
123         IntListElem *el = this->first;
124         for (int i = 0; i < position && el; i++) {
125             el = el->next;
126
126
127
              return el;
128 };
129
```