



C++ 面向对象程序设计

Assignment 6

声明LinkedList为Node的友元类,以便LinkedList中的函数访问以及修改Node中的私有成员的值。对链表进行操作时,注意特判链表是否为空。

Listing 1: linkedlist.h

```
1  #ifndef LINKEDLIST_H
2  #define LINKEDLIST_H
3
4  #include <iostream>
5
6  class LinkedList {
7  public:
8      class Node {
9          friend class LinkedList;
10         public:
11             Node();
12             Node(double);
13
14             Node *next;
15             Node *previous;
16
17             double getValue() const;
18             void setValue(double);
19             friend std::ostream &operator<<(std::ostream &, const Node &);
20
21         private:
22             double value;
23     };
24
25     LinkedList();
26     LinkedList(const std::initializer_list<double>&);
27     LinkedList(const LinkedList &);
28
29     ~LinkedList();
30
31     void push_back(double);
32     void push_front(double);
33     double pop_back();
34     double pop_front();
35     double back() const;
```

```

36     double front() const;
37
38     bool empty() const;
39     int getSize() const;
40
41     void clear();
42     void show() const;
43     void extend(const LinkedList &);
44
45     double &operator[](size_t);
46     const double &operator[](size_t) const;
47
48 private:
49     int N{0};
50
51 public:
52     Node *head;
53     Node *tail;
54 };
55
56 #endif //LINKEDLIST_H

```

Listing 2: linkedlist.cpp

```

1  #include "linkedlist.h"
2  #include <stdexcept>
3
4  LinkedList::Node::Node(): value(0), next(nullptr), previous(nullptr) { }
5
6  LinkedList::Node::Node(double val): value(val), next(nullptr), previous(nullptr) { }
7
8  double LinkedList::Node::getValue() const {
9      return value;
10 }
11
12 void LinkedList::Node::setValue(double val) {
13     value = val;
14 }
15
16 std::ostream &operator<<(std::ostream &os, const LinkedList::Node &node) {
17     os << node.value;
18     return os;
19 }
20
21 LinkedList::LinkedList(): head(nullptr), tail(nullptr), N(0) { }
22
23 LinkedList::LinkedList(const std::initializer_list<double> &list): head(nullptr), tail(nullptr), N(0) {

```

```

24     for (auto &val: list)
25         push_back(val);
26 }
27
28 LinkedList::LinkedList(const LinkedList &list): head(nullptr), tail(nullptr), N(0) {
29     Node *cur = list.head;
30     while (cur != nullptr) {
31         push_back(cur->value);
32         cur = cur->next;
33     }
34 }
35
36 LinkedList::~LinkedList() {
37     clear();
38 }
39
40 void LinkedList::push_back(double val) {
41     Node *newNode = new Node(val);
42     if (N == 0) {
43         head = newNode;
44         tail = newNode;
45     } else {
46         tail->next = newNode;
47         newNode->previous = tail;
48         tail = newNode;
49     }
50     ++N;
51 }
52
53 void LinkedList::push_front(double val) {
54     Node *newNode = new Node(val);
55     if (N == 0) {
56         head = newNode;
57         tail = newNode;
58     } else {
59         head->previous = newNode;
60         newNode->next = head;
61         head = newNode;
62     }
63     ++N;
64 }
65
66 double LinkedList::pop_back() {
67     if (N == 0) {
68         throw std::logic_error("pop_back(): empty list");
69     }

```

```

70     double val = tail->value;
71     Node *newTail = tail->previous;
72     delete tail;
73     tail = newTail;
74     if (tail == nullptr) {
75         head = nullptr;
76     } else {
77         tail->next = nullptr;
78     }
79     --N;
80     return val;
81 }
82
83 double LinkedList::pop_front() {
84     if (N == 0) {
85         throw std::logic_error("pop_front(): empty list");
86     }
87     double val = head->value;
88     Node *newHead = head->next;
89     delete head;
90     head = newHead;
91     if (head == nullptr) {
92         tail = nullptr;
93     } else {
94         head->previous = nullptr;
95     }
96     --N;
97     return val;
98 }
99
100 double LinkedList::back() const {
101     if (N == 0) {
102         throw std::logic_error("back(): empty list");
103     }
104     return tail->value;
105 }
106
107 double LinkedList::front() const {
108     if (N == 0) {
109         throw std::logic_error("front(): empty list");
110     }
111     return head->value;
112 }
113
114 bool LinkedList::empty() const {
115     return N == 0;

```

```

116 }
117
118 int LinkedList::getSize() const {
119     return N;
120 }
121
122 void LinkedList::clear() {
123     while (head != nullptr) {
124         Node *next = head->next;
125         delete head;
126         head = next;
127     }
128     head = tail = nullptr;
129     N = 0;
130 }
131
132 void LinkedList::show() const {
133     Node *cur = head;
134     while (cur != nullptr) {
135         std::cout << cur->value << ", ";
136         cur = cur->next;
137     }
138     std::cout << std::endl;
139 }
140
141 void LinkedList::extend(const LinkedList &list) {
142     Node *cur = list.head;
143     while (cur != nullptr) {
144         push_back(cur->value);
145         cur = cur->next;
146     }
147 }
148
149 double &LinkedList::operator[](size_t n) {
150     if (n >= N) {
151         throw std::logic_error("index out of range");
152     }
153     Node *cur = head;
154     for (int i = 0; i < n; i++) {
155         cur = cur->next;
156     }
157     return cur->value;
158 }
159
160 const double &LinkedList::operator[](size_t n) const {
161     if (n >= N) {

```

```

162     throw std::logic_error("index out of range");
163 }
164 Node *cur = head;
165 for (int i = 0; i < n; i++) {
166     cur = cur->next;
167 }
168 return cur->value;
169 }

```

运行 所有 CTest x

✓ 测试 已通过: 10 共 10 个测试 - 300 毫秒

✓ 测试结果	300 毫秒
✓ assignment6Test.NodeTest	30 毫秒
✓ assignment6Test.LinkedListConstru	30 毫秒
✓ assignment6Test.LinkedListCopyCt	30 毫秒
✓ assignment6Test.LinkedListPush	30 毫秒
✓ assignment6Test.LinkedListPop	30 毫秒
✓ assignment6Test.LinkedListSides	30 毫秒
✓ assignment6Test.LinkedListExtend	30 毫秒
✓ assignment6Test.LinkedListBrackel	30 毫秒
✓ assignment6Test.LinkedListMainNc	30 毫秒
✓ assignment6Test.LinkedListOthers	30 毫秒

```

10: Test command: /tmp/assignment6/cmake-build-debug/assignment6 "--gtest_filter=
10: Test timeout computed to be: 10000000
10: RUNNING TESTS ...
10: Note: Google Test filter = assignment6Test.LinkedListOthers
10: [=====] Running 1 test from 1 test suite.
10: [-----] Global test environment set-up.
10: [-----] 1 test from assignment6Test
10: [ RUN      ] assignment6Test.LinkedListOthers
10: [         OK ] assignment6Test.LinkedListOthers (0 ms)
10: [-----] 1 test from assignment6Test (0 ms total)
10:
10: [-----] Global test environment tear-down
10: [=====] 1 test from 1 test suite ran. (0 ms total)
10: [ PASSED   ] 1 test.
10: <<<SUCCESS>>>

100% tests passed, 0 tests failed out of 10

```