C++面向对象程序设计 作业报告6

报告人：邓雍　学号：2022141220184

目标实现：实现一个链表，左闭右闭。支持双端添加，双端删除，随机查找，在末尾添加另一个链表。全程用指针实现

实现细节：指针的使用使动态结构成为可能，也节约了大量空间。由于监测点要求左闭又闭，故head，tail初始设置为nullptr更合理。但注意对nullprt的很多操作是不合法的，所以需要添加很多状态判断来检测。实际测试点中还要求对输出和[]重载，存在const变量的调用问题，故某些函数需要加上const。

代码实现如下

```cpp
LinkedList::Node::Node(double v){this->value=v;next=previous=nullptr;}

double LinkedList::Node::getValue(){return this->value;}

void LinkedList::Node::setValue(double v){this->value=v;}

LinkedList::LinkedList(){N=0;tail=head=nullptr;}

LinkedList::LinkedList(const LinkedList & A)
{
    this->N=0;this->head=this->tail=nullptr;
    if(A.N)
    {
        Node *p=A.head;
        while(p!=A.tail)
        {
            this->push_back(p->getValue());
            p=p->next;
        }
        this->push_back(A.tail->getValue());
    }
}

LinkedList::LinkedList(std::initializer_list<double> A)
{
    this->N=0;this->head=this->tail=nullptr;
    for(auto X: A) this->push_back(X);
}

LinkedList::~LinkedList()
{
    if(this->head!=nullptr)
    {
        Node *p=this->head;
        while(p!=nullptr&&p!=tail)
        {
            Node *next=p->next;
            if(p!=nullptr)delete p;
            p=next;
        }
        if(tail!=nullptr) delete tail;
    }
}
```

实现了基本所有的构造与析构函数

```cpp
void LinkedList::push_back(double v)
{
    if(this->empty())
    {
        head=tail=new Node(v);
    }
    else
    {
        Node *X=new Node(v);
        X->previous=tail;X->next=nullptr;
        tail->next=X;
        tail=X;
    }
    N++;
}

void LinkedList::push_front(double v)
{
    if(this->empty())
    {
        head=tail=new Node(v);
    }
    else
    {
        Node *X=new Node(v);
        X->next=head;X->previous=nullptr;
        head->previous=X;
        head=X;
    }
    N++;
}
```

双端插入操作大致相仿。注意判断空集时的情况。

```cpp
double LinkedList::pop_back()
{
    if(this->empty())throw std::logic_error("Empty List!");
    double v=this->back();
    tail=tail->previous;
    if(N>1) delete(tail->next),tail->next=nullptr;
    else head=nullptr;
    N--;
    return v;
}

double LinkedList::pop_front()
{
    if(this->empty())throw std::logic_error("Empty List!");
    double v=this->front();
    head=head->next;
    if(N>1) delete(head->previous),head->previous=nullptr;
    else tail=nullptr;
    N--;
    return v;
}

double LinkedList::back()
{
    if(this->empty())throw std::logic_error("Empty List!");
    return tail->getValue();
}

double LinkedList::front()
{
    if(this->empty())throw std::logic_error("Empty List!");
    return head->getValue();
}

bool LinkedList::empty() const{return (!N);}
```

删除操作基本是插入炒作的逆向，把插入反过来进行即可。查询值直接查询head与tail即可。

```cpp
void LinkedList::clear()
{
    Node *p=this->head;
    while(p!=tail)
    {
        Node *next=p->next;
        delete p;
        p=next;
    }
    delete tail;
    head=tail=nullptr;N=0;
}

void LinkedList::show()
{
    if(this->empty()){cout<<endl;return ;}
    Node *p=this->head;
    while(p!=tail)
    {
        cout<<p->getValue()<<" ";
        p=p->next;
    }
    cout<<tail->getValue()<<endl;
}

int LinkedList::getSize(){return this->N;}

void LinkedList::extend(const LinkedList &A)
{
    if(this->empty())
    {
        this->N=0;
        this->head=this->tail=nullptr;
    }
    if(A.empty()) return ;
    Node *p=A.head;
    while(p!=A.tail)
    {
        Node *next=p->next;
        this->push_back(p->getValue());
        p=next;
    }
    this->push_back(A.tail->getValue());
}
```

clear，show等操作基本都是以访问操作再进行不同处理，访问操作都大体相同。

```cpp
double& LinkedList::operator[](const int &x)
{
    if(this->empty()) throw std::logic_error("Empty List!");
    if(x<0||x>=N) throw std::logic_error("Out of range!");
    int now=0;
    Node *p=head;
    while(p!=tail)
    {
        if(x==now) return p->value;
        ++now;
        p=p->next;
    }
    if(x==now) return p->value;
    throw std::logic_error("Out of range!");
}

ostream &operator<<(ostream &out, const LinkedList::Node &node)
{
    out<<node.value;
    return out;
}
```

重载两个题目所需的运算符，查询第x个元素与重载输出。

```
root@457ba46ceb43:/ws/Assignment6# make && ./main
make: 'main' is up to date.
RUNNING TESTS ...
[==========] Running 10 tests from 1 test suite.
[----------] Global test environment set-up.
[----------] 10 tests from assignment6Test
[ RUN      ] assignment6Test.NodeTest
0 10
[       OK ] assignment6Test.NodeTest (0 ms)
[ RUN      ] assignment6Test.LinkedListConstructors

3.14 20 13 -5 0 -3.2
[       OK ] assignment6Test.LinkedListConstructors (0 ms)
[ RUN      ] assignment6Test.LinkedListCopyConstructor
1 2 3 4 5 6 7 8 9 10
0 1 2 3 4 5 6 7 8 9
[       OK ] assignment6Test.LinkedListCopyConstructor (0 ms)
[ RUN      ] assignment6Test.LinkedListPush
0 1 2 3 4
0 1 2 3 4
-7 -6 -5 -4 -3 -2 -1 0 1 2 3
[       OK ] assignment6Test.LinkedListPush (0 ms)
[ RUN      ] assignment6Test.LinkedListPop
2 3 4 5 6
4 5 6 7 8
[       OK ] assignment6Test.LinkedListPop (0 ms)
[ RUN      ] assignment6Test.LinkedListSides
[       OK ] assignment6Test.LinkedListSides (0 ms)
[ RUN      ] assignment6Test.LinkedListExtend


1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9

1 2 3 4 5 6 7 8 9 10 11 12 13 14
10 11 12 13 14
[       OK ] assignment6Test.LinkedListExtend (0 ms)
[ RUN      ] assignment6Test.LinkedListBracket
0 1 4 9 16 25 36 49 64
[       OK ] assignment6Test.LinkedListBracket (0 ms)
[ RUN      ] assignment6Test.LinkedListMainNodes
[       OK ] assignment6Test.LinkedListMainNodes (0 ms)
[ RUN      ] assignment6Test.LinkedListOthers
[       OK ] assignment6Test.LinkedListOthers (0 ms)
[----------] 10 tests from assignment6Test (0 ms total)

[----------] Global test environment tear-down
[==========] 10 tests from 1 test suite ran. (0 ms total)
[  PASSED  ] 10 tests.
<<<SUCCESS>>>
root@457ba46ceb43:/ws/Assignment6#
```

附上代码通过截图