//Devin Hardy

//CS372

//List Class

#include <iostream>

#include <iomanip>

#include <cstdlib>

using namespace std;

typedef int v\_t;

class List

{

private:

static const int CAP = 20;

v\_t Array[CAP];

int pos;

int used;

void toShift(int form, int to);

public:

//Constructor

List();

//Work Methods

bool empty();

void first();

void last();

void prev();

void next();

int getPos();

void setPos(int v);

void insertBefore(v\_t item);

void insertAfter(v\_t item);

v\_t getElement();

int size();

void replace(v\_t val);

void erase();

void clear();

//Overload

bool operator==(List L1);

bool operator!=(List L1);

List operator+(List L1);

void operator+=(List L1);

void operator=(List L1);

friend ostream& operator<<(ostream &out, List &L1);

};

List::List()

{

v\_t zero = 0;

pos = 0;

used = 0;

for(int i = 0; i < CAP; i++)

{

Array[i] = zero;

}

}

bool List::empty()

{

return used;

}

void List::first()

{

pos = 0;

}

void List::last()

{

pos = used - 1;

if(used == 0)

pos = 0;

}

void List::prev()

{

if(used == 0)

pos = 0;

else if(pos < 0)

pos = 0;

else pos = pos - 1;

}

void List::next()

{

if(used == 0)

pos = 0;

else if(pos > used)

pos = used - 1;

else

pos = pos + 1;

}

int List::getPos()

{

return pos;

}

void List::setPos(int v)

{

if(used == 0)

pos = 0;

else if(v > used)

pos = used - 1;

else

pos = v;

}

void List::insertBefore(v\_t item)

{

if(used == 0)

{

used++;

pos = 0;

Array[pos] = item;

}

else

{

if(used == CAP)

return;

else

{

used++;

for(int i = used-1; i > pos; i--)

{

Array[i] = Array[i-1];

}

Array[pos] = item;

}

}

}

void List::insertAfter(v\_t item)

{

if(used == 0)

{

used++;

pos = 0;

Array[pos] = item;

}

else

{

if(used == CAP)

return;

else

{

used++;

pos++;

Array[pos] = item;

}

}

}

v\_t List::getElement()

{

return(Array[pos]);

}

int List::size()

{

return (used);

}

void List::replace(v\_t val)

{

Array[pos] = val;

}

void List::erase()

{

// Erase / Shift / Done

if(used == 0)

return;

else

{

for(int i = pos; i < used; i++)

{

Array[i] = Array[i+1];

}

used--;

}

if(pos >= used)

pos = used - 1;

}

void List::clear()

{

used = 0;

}

//Overload

bool List::operator==(List L1)

{

int temp;

temp = L1.getPos();

L1.first();

for(int i = 0; i < used; i++)

{

if(Array[i] != L1.getElement())

return 0;

L1.next();

}

L1.setPos(temp);

return 1;

}

bool List::operator!=(List L1)

{

int temp;

temp = L1.getPos();

L1.first();

for(int i = 0; i < used; i++)

{

if(Array[i] == L1.getElement())

return 0;

L1.next();

}

L1.setPos(temp);

return 1;

}

List List::operator+(List L1)

{

int temp1, temp2;

int length;

List TempL;

temp1 = pos;

temp2 = L1.getPos();

length = L1.size();

L1.first();

pos = used - 1;

for(int i = 0; i < used ; i++)

{

TempL.insertAfter(Array[i]);

}

for(int i = 0; i < length ; i++)

{

TempL.insertAfter(L1.getElement());

L1.next();

}

pos = temp1;

L1.setPos(temp2);

return TempL;

}

void List::operator+=(List L1)

{

int temp;

int length;

temp = L1.getPos();

length = L1.size();

L1.first();

pos = used - 1;

for(int i = 0; i < length ; i++)

{

this -> insertAfter(L1.getElement());

L1.next();

}

L1.setPos(temp);

return;

}

void List::operator=(List L1)

{

int length;

L1.first();

length = L1.size();

for(int i = 0; i < length ; i++)

{

used++;

Array[i] = L1.getElement();

L1.next();

}

}

ostream& operator<<(ostream &out, List &L1)

{

int length;

length = L1.size();

L1.first();

for(int i = 0; i < length ; i++)

{

out << L1.getElement() << " ";

L1.next();

}

return out;

}

int main()

{

List a,b; int endit;

for (int i=1;i<=20;i++)

a.insertAfter(i\*2);

cout << "List a : " << endl;

cout << " " << a << endl;

cout << "Number of elements in a - " << a.size() << endl;

for (int i=1;i<=10;i++)

b.insertBefore(i\*3);

cout << "List b : " << endl;

cout << " " << b << endl;

cout << "Number of elements in b - " << b.size() << endl;

if ( a == b )

cout << "List a & b are equal" << endl;

else

cout << "List a & b are Not equal" << endl;

a.first();

b.first();

cout << "First elmenet in list a & b: " << a.getElement() << ", "

<< b.getElement() << endl;

a.last();

b.last();

cout << "Last elmenet in list a & b: " << a.getElement() << ", "

<< b.getElement() << endl;

cout << endl << endl << " Start of new stuff" << endl;

b.clear();

cout << "Empty List b: " << b << endl;

if ( a == b )

cout << "List a & b are equal" << endl;

else

cout << "List a & b are Not equal" << endl;

for (int i=1;i<=10;i++)

b.insertBefore(i\*5);

cout << "List b : " << endl << b << endl;

a.setPos(5);

b.first();

for ( int i=1; i<4; i++)

{

a.erase();

b.replace(i);

b.next();

}

cout << "Modified Object 'a' (shorter) " << endl;

cout << "List a: " << a << endl;

cout << "Modified Object 'b' " << endl;

cout << "List b: " << b << endl;

List c(b);

cout << "Copy Constructor c(b)" << endl;

cout << "List b : " << b << endl;

cout << "List c : " << c << endl;

if ( c == b )

cout << "List c & b are equal" << endl;

else

cout << "List c & b are Not equal" << endl;

List e;

e = c;

cout << "Object 'c' assigned to Object 'e':" << endl;

cout << "List c : " << c << endl;

cout << "List e : " << e << endl;

List d;

d=a;

d.first();

endit = d.size();

for ( int i = 1; i < endit; d.next(), i++)

{

d.insertBefore(d.getElement()\*(-2));

d.next();

}

cout << "Results after some Replaces on d " << endl;

cout << "List d : " << d << endl;

a.first();

endit = a.size();

for ( int i = 1; i < endit; a.next(), i++)

{

a.replace(a.getPos()+a.getElement());

a.next();

}

cout << "Results after some weird stuff on list a" << endl;

cout << "List a : " << a << endl;

List alist(b);

alist.clear();

for (int i=1;i<=10;i++)

alist.insertAfter(i);

alist.first();

cout << "New List alist with positions above: " << endl;

for (int i=1;i<=10;i++) {

cout << setw(5) << alist.getPos();

alist.next();

}

cout << endl;

alist.first();

for (int i=1;i<=10;i++) {

cout << setw(5) << alist.getElement();

alist.next();

}

cout << endl;

cout << endl << " check out boundary conditions" << endl;

List sq;

cout << "number of elements in empty sq list = " << sq.size() << endl;

sq.first();

sq.erase();

sq.setPos(5);

cout << "empty sq values " << sq << endl;

sq.insertBefore(777);

cout << "sq values " << sq << endl;

sq.next(); sq.next();

cout << "sq.getElement() = " << sq.getElement() << endl;

cout << "sq list = " << sq << endl;

return 0;

}

