/\* Assignment: General List Program

Author: Ryan Wood

Created On: 1/28/2018

Requirements: Defined in Document "General List Program"

provided by Dr. Rimes

\*/

/\*List definition file\*/

#ifndef LIST\_H

#define LIST\_H

#include <iostream>

#include <fstream>

namespace rimes

{

typedef int et;

static const size\_t CAPACITY = 20;

class List

{

public:

/\* Function: List()

Purpose: default constructor

Return: an empty List

\*/

List();

/\* Function: List()

Purpose: copy constructor

Paramters: the List to copy

Return: a new List with the same values as

the one given

\*/

List(const List &lstA);

/\* Function: emtpy()

Purpose: determines whether the list has no values

Return: whether elements are used

\*/

bool empty() const;

/\* Function: first()

Purpose: sets the position to the first element of the array, 0

\*/

void first();

/\* Function: last()

Purpose: sets the position of the array to last element

in the array, used-1

\*/

void last();

/\* Function: prev()

Purpose: sets the position back by one index

\*/

void prev();

/\* Function: next()

Purpose: sets the postition forward by one index

\*/

void next();

/\* Function: getPosition() const

Purpose: retrieves the current position index

Return: the index

\*/

et getPos() const;

/\* Funcrtion: setPos(et)

Purpose: sets the current position to the given

index if it is valid

Parameters: an index less than the size of the List

\*/

void setPos(et);

/\* Function: insertBefore(et)

Purpose: inserts the given value before the current

position in the list

Parameters: the value

\*/

void insertBefore(et);

/\* Function: insertAfter(et)

Purpose: inserts the given value after the

current position in the List

Parameters: the value

\*/

void insertAfter(et);

/\* Function: getElement() const

Purpose: retrieves the value at the current position

in the list

Return: the value at the current index

\*/

et getElement() const;

/\* Function: getElement(et)

Purpose: retrieves the value at the given index to the List

if it is valid for the list

Parameters: the index

Return: the value

\*/

et getElement(et pos) const;

/\* Function: size()

Purpose: gets the size of the list, the number of

elements that have been used

Return: the number of elements used

\*/

size\_t size() const;

/\* Function: replace(et)

Purpose: replaces the value at the current index

to the list with the value given

Parameters: the value

\*/

void replace(et);

/\* Function: erase()

Purpose: erases the element at the current index to

the list, shifting all other elements down

to take the empty place

\*/

void erase();

/\* Function: clear()

Purpose: deletes all elements in the list, setting

pos and used both to 0

\*/

void clear();

private:

/\* Function: copy(List&)

Purpose: copies the given list to the current list,

element by element, copying the values of

the array and the value of pos and used

Parameters: a valid list

\*/

void copy(const List &lstA);

et arry[CAPACITY];

et used;

et pos;

};//List class

};//namespace

/\* Function: operator<<(ostream&, List&)

Purpose: overloads the stream insertion operator using the given

output to the console

Parameters: the console output stream, the List to print

Return: a reference to the output stream

\*/

std::ostream& operator <<(std::ostream &out, const rimes::List &lst);

/\* Function: operator <<(ofstream&, List&)

Purpose: overloads the given file output stream and prints the

given list to the file

Parameters: a reference to the file output stream,

a reference to the LIst to print

Return: a reference to the file output stream

\*/

std::ofstream& operator <<(std::ofstream &out, const rimes::List &lst);

/\* Function: operator==(List&, List&)

Purpose: determines whether the given lists are equal

Parameters: the first list, the second list

Return: whether the lists have the same values

\*/

bool operator ==(const rimes::List &lstA, const rimes::List &lstB);

/\* Function: operator !=(List&, List&)

Purpose: determines whether the given lists are not equal

Parameters: the first list, the second list

Return: whether the two lists do not have the same values

\*/

bool operator !=(const rimes::List &lstA, const rimes::List &lstB);

/\* Function: operator +(List&, List&)

Purpose: adds the values of the second list into the remaining values

of the first list, creating a new list with the results

Parameters: the first list, the second List

Return: a list with values of both lists

\*/

rimes::List operator +(const rimes::List &lstA, const rimes::List &lstB);

#endif

/\* Assignment: General List Program

Author: Ryan Wood

Created On: 1/28/2018

Requirements: Defined in Document "General List Program"

provided by Dr. Rimes

\*/

/\*List Implementation file\*/

#include <iostream>

#include "List.h"

using namespace std;

using namespace rimes;

List::List()

{

used = 0;

pos = 0;

}

List::List(const List &lstA)

{

copy(lstA);

}

bool List::empty() const

{

return(used == 0);

}

void List::first()

{

pos = 0;

}

void List::last()

{

pos = used-1;

}

void List::prev()

{

if(pos > 0)

pos--;

}

void List::next()

{

if(pos < CAPACITY-1 && pos < used-1)

pos++;

}

et List::getPos() const

{

return pos;

}

void List::setPos(et val)

{

if(val < used && val > -1)

pos = val;

}

void List::insertBefore(et val)

{

et \*current;

et \*tmp;

//case used = 0

if(used < CAPACITY)

{

if(used == 0)

arry[pos] = val;

else

{

current = &arry[pos];

tmp = &arry[used-1];

while(tmp >= current)

{

\*(tmp+1) = \*tmp;

tmp--;

}

\*current = val;

}

used++;

}

}

void List::insertAfter(et val)

{

et \*current;

et \*tmp;

if(used < CAPACITY)

{

//case used = 0;

if(used == 0)

arry[pos] = val;

else

{

current = &arry[pos];

tmp = &arry[used-1];

//from the last item to the current item, loop

//backward through array, moving elements forward by one

while(tmp > current)

{

\*(tmp+1) = \*tmp;

tmp--;

}

current++;

\*current = val;

pos++;

}

used++;

}

}

et List::getElement(et ePos) const

{

et val = 0;

if(pos >= 0 && pos < used)

{

val = arry[ePos];

}

return val;

}

et List::getElement() const

{

et val = 0;

if(used > 0)

{

val = arry[pos];

}

return val;

}

size\_t List::size() const

{

return used;

}

void List::replace(et val)

{

arry[pos] = val;

}

void List::erase()

{

int count = 0;

if(used > 0)

{

//delete current element

arry[pos] = 0;

//left justify the rest of the array

count = pos;

while(count < used)

{

arry[count] = arry[count+1];

count++;

}

used--;

pos--;

}

}

void List::clear()

{

int count = 0;

for(count = 0; count < used; count++)

{

arry[count] = 0;

}

used = 0;

pos = 1;

}

void List::copy(const List &lstA)

{

int count = 0;

used = 0;

pos = 0;

if(!lstA.empty())

{

for(count = 0; count < lstA.size(); count++)

{

arry[count] = lstA.getElement(count);

}

}

pos = lstA.getPos();

used = lstA.size();

}

std::ostream& operator <<(std::ostream &out, const List &lst)

{

int count = 0;

for(count = 0; count < lst.size(); count++)

{

if(count == 0)

out << lst.getElement(count);

else out << ", " << lst.getElement(count);

}

return out;

}

std::ofstream& operator <<(std::ofstream &out, const rimes::List &lst)

{

int count = 0;

for(count = 0; count < lst.size(); count++)

{

if(count == 0)

out << lst.getElement(count);

else out << ", " << lst.getElement(count);

}

return out;

}

bool operator ==(const List &lstA, const List &lstB)

{

int count = 0;

if(lstA.size() != lstB.size())

return false;

for(count = 0; count < lstA.size(); count++)

{

if(lstA.getElement(count) != lstB.getElement(count))

return false;

}

return true;

}

bool operator !=(const List &lstA, const List &lstB)

{

return(lstA != lstB);

}

List operator +(const List &lstA, const List &lstB)

{

int count = 0;

List lstC;

for(count = 0; count < lstA.size(); count++)

{

lstC.insertAfter(lstA.getElement(count));

}

for(count = 0; count < lstB.size(); count++)

{

lstC.insertAfter(lstA.getElement(count));

}

return lstC;

}

/\* Assignment: General List Program

Author: Ryan Wood

Created On: 1/28/2018

Requirements: Defined in Document "General List Program"

provided by Dr. Rimes

\*/

/\*List Main file provided by Dr. Rimes. Modified to output to file\*/

#include <iostream>

#include <iomanip>

#include <fstream>

#include "List.h"

using namespace std;

using namespace rimes;

int main()

{

ofstream outFile("TestList.out");

List a,b; int endit;

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

a.insertAfter(i\*2);

outFile << "List a : " << endl;

outFile << " " << a << endl;

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

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

b.insertBefore(i\*2);

outFile << "List b : " << endl;

outFile << " " << b << endl;

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

if ( a == b )

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

else

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

a.first();

b.first();

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

<< b.getElement() << endl;

a.last();

b.last();

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

<< b.getElement() << endl;

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

b.erase();

if ( a == b )

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

else

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

a.setPos(10);

for ( int i=1; i<8; i++)

{

a.erase();

b.replace(i);

b.next();

}

outFile << "Modified Object 'a' " << endl;

outFile << "List a: " << a << endl;

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

outFile << "List b: " << b << endl;

List c(b);

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

outFile << "List b : " << b << endl;

outFile << "List c : " << c << endl;

if ( c == b )

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

else

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

List e;

e = c;

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

outFile << "List c : " << c << endl;

outFile << "List e : " << e << endl;

List d;

d=a;

d.first();

endit = d.size()/2;

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

{

d.next();

d.erase();

}

outFile << "Results after some erases: Object d " << endl;

outFile << "List d : " << d << endl;

d.first();

endit = d.size();

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

{

d.insertBefore(d.getElement()\*5);

d.next();

}

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

outFile << "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();

}

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

outFile << "List a : " << a << endl;

List alist(b);

alist.clear();

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

alist.insertAfter(i);

alist.first();

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

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

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

alist.next();

}

outFile << endl;

alist.first();

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

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

alist.next();

}

outFile << endl;

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

List sq;

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

sq.first();

sq.erase();

sq.setPos(5);

outFile << "empty sq values " << sq << endl;

sq.insertBefore(777);

outFile << "sq values " << sq << endl;

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

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

outFile << "sq list = " << sq << endl;

outFile.close();

return 0;

}

List a :

2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40

Number of elements in a - 20

List b :

40, 38, 36, 34, 32, 30, 28, 26, 24, 22, 20, 18, 16, 14, 12, 10, 8, 6, 4, 2

Number of elements in b - 20

List a & b are Not equal

First elmenet in list a & b: 2, 40

Last elmenet in list a & b: 40, 2

Start of new stuff

List a & b are Not equal

Modified Object 'a'

List a: 2, 4, 6, 8, 24, 26, 28, 30, 32, 34, 36, 38, 40

Modified Object 'b'

List b: 40, 38, 36, 34, 32, 30, 28, 26, 24, 22, 20, 18, 16, 14, 12, 10, 8, 6, 7

Copy Constructor c(b)

List b : 40, 38, 36, 34, 32, 30, 28, 26, 24, 22, 20, 18, 16, 14, 12, 10, 8, 6, 7

List c : 40, 38, 36, 34, 32, 30, 28, 26, 24, 22, 20, 18, 16, 14, 12, 10, 8, 6, 7

List c & b are equal

Object 'c' assigned to Object 'e':

List c : 40, 38, 36, 34, 32, 30, 28, 26, 24, 22, 20, 18, 16, 14, 12, 10, 8, 6, 7

List e : 40, 38, 36, 34, 32, 30, 28, 26, 24, 22, 20, 18, 16, 14, 12, 10, 8, 6, 7

Results after some erases: Object d

List d : 2, 30, 32, 34, 36, 38, 40

Results after some Replaces on d

List d : 10, 2, 150, 30, 160, 32, 170, 34, 180, 36, 190, 38, 40

Results after some weird stuff on list a

List a : 2, 4, 8, 8, 28, 26, 34, 30, 40, 34, 46, 38, 112

New List alist with positions above:

0 1 2 3 4 5 6 7 8 9

0 1 2 3 4 5 6 7 8 9

check out boundary conditions

number of elements in empty sq list = 0

empty sq values

sq values 777

sq.getElement() = 777

sq list = 777