//File: Vector.h

//Programmer: Taylor Pedretti

//Class: CSE-330

//Project: Homework 1

#ifndef VECTOR\_H

#define VECTOR\_H

#include <algorithm>

#include <iostream>

#include <cassert>

template<typename T>

class Vector {

private:

int theSize;

int theCapacity;

T \* objects;

public:

static const int SPACE\_CAPACITY = 2;

Vector(int initsize = 0)

: theSize(initsize),

theCapacity(initsize + SPACE\_CAPACITY)

{

objects = new T[theCapacity];

}

Vector(const Vector &rhs)

: theSize(rhs.theSize),

theCapacity(rhs.theCapacity), objects(0)

{

objects = new T[theCapacity];

for (int k = 0; k < theSize; k++)

objects[k] = rhs.objects[k];

}

Vector(Vector &&rhs)

: theSize(rhs.theSize),

theCapacity(rhs.theCapacity),

objects{ rhs.objects }

{

rhs.objects = nullptr;

rhs.theSize = 0;

rhs.theCapacity = 0;

}

~Vector()

{

delete[] objects;

}

Vector & operator=(const Vector &rhs)

{

//Vector copy = this;

Vector copy(this);

std::swap(\*this, copy);

return this;

}

Vector & operator=(Vector &&rhs)

{

std::swap(theSize, rhs.theSize);

std::swap(theCapacity, rhs.theCapacity);

std::swap(objects, rhs.objects);

return \*this;

}

bool empty() const

{

return size() == 0;

}

int size() const

{

return theSize;

}

int capacity() const

{

return theCapacity;

}

T & operator[](int index)

{

assert(index >= 0 && index < theSize);

return objects[index];

}

const T & operator[](int index) const

{

assert(index >= 0 && index < theSize);

return objects[index];

}

void resize(int newSize)

{

if (newSize > theCapacity)

reserve(newSize \* 2);

theSize = newSize;

}

void reserve(int newCapacity)

{

if (newCapacity < theSize)

return;

T \*newArray = new T[newCapacity];

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

newArray[i] = std::move(objects[i]);

theCapacity = newCapacity;

std::swap(objects, newArray);

delete [] newArray;

}

void erase(int k)

{

T \*newArray = new T[theCapacity];

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

{

if((i > k))

{

newArray[i - 1] = objects[i];

}

else

{

newArray[i] = objects[i];

}

if (objects[i] == back())

break;

}

std::swap(objects, newArray);

pop\_back();

delete[] newArray;

}

void insert(int value, int location)

{

int k = (location > theSize) ? (theSize + 1) : location;

T \*newArray = new T[theCapacity + 1];

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

{

if (i == location)

{

newArray[i] = value;

}

else if ((i == theSize) && (k > theSize))

{

newArray[i] = objects[k];

newArray[i] = value;

}

else if (i > k)

{

newArray[i] = objects[k];

k++;

}

else

{

newArray[i] = objects[i];

}

}

resize(theSize + 1);

reserve(theCapacity + 1);

std::swap(objects, newArray);

delete[] newArray;

}

void push\_back(const T &x)

{

if (theSize == theCapacity)

reserve(2 \* theCapacity + 1);

objects[theSize++] = x;

}

void push\_back(T &x)

{

if (theSize == theCapacity)

reserve(2 \* theCapacity + 1);

objects[theSize++] = std::move(x);

}

void pop\_back()

{

assert(!empty());

--theSize;

}

const T & back() const

{

assert(!empty());

return objects[theSize - 1];

}

const T & front() const

{

assert(!empty());

return objects[0];

}

};

#endif // !VECTOR\_H

//File: VectorMain.cpp

//Programmer: Taylor Pedretti

//Class: CSE-330

//Project: Homework 1

#include <iostream>

#include "Vector.h"

using namespace std;

void print\_vector(Vector<int> v)

{

for (int i = 0; i < v.size(); i++)

cout << v[i] << " ";

cout << endl;

}

int main()

{

Vector<int> v1;

Vector<int> v2;

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

{

v1.push\_back(i);

//v2.push\_back(i);

}

cout << "Size of v1: " << v1.size() << endl;

cout << "Printing vector: ";

print\_vector(v1);

cout << endl;

cout << "Erasing index 10" << endl;

v1.erase(10);

cout << "Printing vector: ";

print\_vector(v1);

cout << endl;

cout << "Erasing index 3" << endl;

v1.erase(3);

cout << "Printing vector: ";

print\_vector(v1);

cout << endl;

cout << "Inserting value 777 into location 10 in the vector:" << endl;

v1.insert(777, 10);

cout << "Printing vector: ";

print\_vector(v1);

cout << endl;

cout << "Inserting value 777 into location 5 in the vector:" << endl;

v1.insert(777, 5);

cout << "Printing vector: ";

print\_vector(v1);

cout << endl;

//system("PAUSE");

return 0;

}

Homework 1 Typescript:

[005488635@csusb.edu@csevnc HW1]$ c++ main.cpp Vector.h

[005488635@csusb.edu@csevnc HW1]$ ./a.out

Size of v1: 10

Printing vector: 1 2 3 4 5 6 7 8 9 10

Erasing index 10

Printing vector: 1 2 3 4 5 6 7 8 9

Erasing index 3

Printing vector: 1 2 3 5 6 7 8 9

Inserting value 777 into location 10 in the vector:

Printing vector: 1 2 3 4 5 6 7 8 9 10 777

Inserting value 777 into location 5 in the vector:

Printing vector: 1 2 3 4 5 777 6 7 8 9 10 777

[005488635@csusb.edu@csevnc HW1]$