/\*--- vectorlab.cpp -------------------------------------------------------

A study of STL's vector container

Written by: Larry R. Nyhoff

Written for: Lab Manual for ADTs, Data Structures, and Problem

Solving with C++, 2E

Lab #7.1

Chris Dang Class: CSCI 1107

--------------------------------------------------------------------------\*/

#include <iostream>

#include <vector>

using namespace std;

template <typename T>

ostream & operator<<(ostream & out, const vector<T> & v)

/\*-------------------------------------------------------------------------

Overloaded output operator for vector<T>s.

Precondition: ostream out is open.

Postcondition: Elements of v have been output to out and out is returned.

--------------------------------------------------------------------------\*/

{

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

out << v.operator[](i) << " ";

return out;

}

int main()

{

// Declare 6 vectors v1, v2, v3, v4, v5, v6 to illustrate the

// various types of declarations (and constructors)

vector<int> v1;

vector<int> v2(2);

int numInts;

cout << "Enter capacity of v3: ";

cin >> numInts;

vector<int> v3(numInts);

vector<int> v4(3, 99);

// The preceding declaration should work, but it may not in some

// versions of some compilers. The following is a work-around:

// vector<int> v4(3);

// for (int i = 0; i < 3; i++) v4[i] = 99;

//--- End of work-around

int a[] = {1, 4, 9, 16, 25};

vector<int> v5(a, a + 5);

vector<int> v6;

//--- 1 --- Add:

// Statements to display the capacity and size of each vector<int>

// and whether it is empty

cout << "Capacity of vectors:\n" <<

"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n"

"v1: " << v1.capacity() << endl <<

"v2: " << v2.capacity() << endl <<

"v3: " << v3.capacity() << endl <<

"v4: " << v4.capacity() << endl <<

"v5: " << v5.capacity() << endl << endl ;

cout << "Size of vectors:\n" <<

"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n"

"v1: " << v1.size() << endl <<

"v2: " << v2.size() << endl <<

"v3: " << v3.size() << endl <<

"v4: " << v4.size() << endl <<

"v5: " << v5.size() << endl << endl ;

cout << "Vectors empty?:\n" <<

"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n"

"v1: " << v1.empty() << endl <<

"v2: " << v2.empty() << endl <<

"v3: " << v3.empty() << endl <<

"v4: " << v4.empty() << endl <<

"v5: " << v5.empty() << endl << endl ;

//--- 2 --- Add:

// Statements to display the maximum capacity of a vector<int>

cout << "Max capacity of v1 is: " << v1.max\_size() << endl << endl ;

//--- 3 --- Add:

// Statements to see the effect of the reserve() member function

cout << "Reserving space v4 to hold 7 elements.\n" ;

v4.reserve(7) ;

cout << "Capacity of v4 is: " << v4.capacity() <<

" Size of v4 is: " << v4.size() << endl << endl ;

//--- 4 --- Add:

// Output statements of the form cout << vector-variable << endl;

// to display the contents of each vector

cout << "contents of v1:\n" ;

//for (int i = 0; i < v1.capacity(); i++)

cout << v1 << endl ;

cout << "contents of v2:\n" ;

//for (int i = 0; i < v2.size(); i++)

cout << v2 << endl ;

cout << "contents of v3:\n" ;

//for (int i = 0; i < v2.size(); i++)

cout << v3 << endl ;

cout << "contents of v4:\n" ;

//for (int i = 0; i < v2.size(); i++)

cout << v4 << endl ;

cout << "contents of v5:\n" ;

//for (int i = 0; i < v2.size(); i++)

cout << v5 << endl ;

//--- 5 --- Add:

// Statements to append 11 to v2 and then output v2's size and contents

// append 22 to v2 and then output v2's size and contents

// append 33 to v2 and then output v2's size and contents

// remove the last element of v2 and then output v2's size

// and contents

cout << "\nAppending 11 to v2\n" ;

v2.push\_back(11) ;

cout << "After appending 11, v2's size is: " << v2.size() << endl <<

"v2's contents: "<< v2 << endl ;

cout << "\nAppending 22 to v2\n" ;

v2.push\_back(22) ;

cout << "After appending 22, v2's size is: " << v2.size() << endl <<

"v2's contents: "<< v2 << endl ;

cout << "\nAppending 33 to v2\n" ;

v2.push\_back(33) ;

cout << "After appending 33, v2's size is: " << v2.size() << endl <<

"v2's contents: "<<v2 << endl ;

cout << "\nDeleting last element of v2\n" ;

v2.pop\_back() ;

cout << "After deletion, v2's size is: " << v2.size() << endl <<

"v2's contents: "<< v2 << endl ;

//--- 6 --- Statements to investigate how capacities grow

// Add statements to append 111 to v1 and then output v1's capacity, size,

// and contents

cout << "\nAppending 111 to v1\n" ;

v1.push\_back(111) ;

cout << "After appending 111, v1's size is: " << v1.size() << endl

<< "v1's capacity is: " << v1.capacity() << endl <<

"v1's contents: " << v1 << endl ;

//--- 7 --- Statements to investigate how capacities grow

// Add statements to append 222, 333, 444, and 555 to v1 and output

// v1's capacity, size, and contents after each value is appended

for (int i = 1; i < 5; i++) {

cout << "\nAppending 111 to v1\n" ;

v1.push\_back(111 \* (i+1)) ;

cout << "After appending " << 111 \* (i+1) << " v1's size is: " << v1.size() << endl

<< "v1's capacity is: " << v1.capacity() << endl <<

"v1's contents: " << v1 << endl ;

}// end for

//--- 8 --- Statements to investigate how capacities grow

// Remove the comment delimiters from the following:

//

int oldCapacity = v1.capacity();

for (int i = v1.size() + 1; i <= 2500; i++)

{

v1.push\_back(999);

if (v1.capacity() == v1.size())

cout << "\n\*\*\* v1 is full with " << v1.size() << " elements\n";

if (v1.capacity() > oldCapacity)

{

cout << "Adding an element increases capacity from "

<< oldCapacity << " to " << v1.capacity() << endl;

oldCapacity = v1.capacity();

}

}

//--- 9 --- Statements to see if element type affects how capacities grow

// Add:

// A declaration of an empty vector<double> v0;

// A loop like the preceding but with v1 replaced by v0

//

// Then change double to char and run it again.

cout << "\n\nv0 starts here\n\n" ;

vector<char> v0 ;

int oldCapacity2 = v0.capacity();

for (int i = 1 ; i <= 2500; i++)

{

v0.push\_back(999);

if (v0.capacity() == v0.size())

cout << "\n\*\*\* v0 is full with " << v0.size() << " elements\n";

if (v0.capacity() > oldCapacity2)

{

cout << "Adding an element increases capacity from "

<< oldCapacity2 << " to " << v0.capacity() << endl;

oldCapacity2 = v0.capacity();

}

}

//--- 10 --- Statements to see how initial capacity affects

// how capacities grow

// Uncomment the following line:

// cout << "Initial capacity of v4 is " << v4.capacity() << endl;

// Add a loop like that in 9 but output changes in v4's capacity

cout << "\n\nv4 starts here\n\n" ;

int oldCapacity3 = v4.capacity();

for (int i = 1 ; i <= 2500; i++)

{

v4.push\_back(999);

if (v4.capacity() == v4.size())

cout << "\n\*\*\* v0 is full with " << v4.size() << " elements\n";

if (v4.capacity() > oldCapacity3)

{

cout << "Adding an element increases capacity from "

<< oldCapacity3 << " to " << v4.capacity() << endl;

oldCapacity3 = v4.capacity();

}

}

//--- 11 --- Statements to access the ends of a vector

// Uncomment the following line:

// cout << "Original contents of v5: " << v5 << endl;

// Add statements to:

// Output the first and last elements of v5

// Change the first element to 77 and the last element to 88

// Output the contents of v5

cout << "\n\nFirst value of v5 is: " << v5.front() << endl <<

"\n\n Last value of v5 is: " << v5.back() << endl << endl ;

//--- 12 --- Statements to demonstrate correct and incorrect

// use of the subscript operator

// Add statements that try using the subscript operator to:

// change the value in location 1 of v2 to 2222

// append the value 3333 to v2

// append a value to empty vector v6

v5.front() = 77 ;

v5.back() = 88 ;

cout << "\n\nFirst value of v5 is: " << v5.front() << endl <<

"\n\n Last value of v5 is: " << v5.back() << endl << endl ;

cout << "\nAssigning v2[1] to 2222.\n" ;

v2[1] = 2222 ;

cout << "Contents of v2: " << v2 << endl;

cout << "Size of v2: " << v2.size() << endl ;

cout << " Cap of v2: " << v2.capacity() << endl ;

/\*

cout << "\nAppending 3333 to v2.\n" ;

v2[v2.size()] = 3333 ;

cout << "Contents of v2: " << v2 ;

for (int i = 0; i <= v2.size(); i++)

cout << v2[i] << " " ;

cout << endl ;

\*/

//cout << "\nAppending 3333 to v6.\n" ;

//v6[0] = 3333 ;

//--- 13 --- Add statements that:

// assign v5 to v3

// check if they are equal

// check if v5 is less than v2

// swap contents of v5 and v2

// check if v5 is less than v2

cout << "\n\n\Copying contents of v5 to v3.\n" ;

v3 = v5 ;

cout << "Is v5 == v3? " << (v5 == v3) << endl ;

cout << "Is v5 < v2? " << (v5 < v2) << endl ;

cout << "Swapping contents of v5 and v2.\n" ;

v5.swap(v2) ;

cout << "Is v5 < v2? " << (v5 < v2) << endl ;

}







