// Demonstrate the use of the Standard Template Library "vector" class.
// and associated iterators, and templated subroutines
// George F. Riley, ECE3090 Georgia Tech, Fall 2009

#include <iostream>
#include <vector>
#include <algorithm>
#include <iterator>

using namespace std;

typedef vector<char> CharVec_t; // Define a vector of character

class Car {
public:
    Car(const char* n, int c); // Constructor with name and cost
    bool operator<(const Car& rhs); // Define a less than operator
public:
    int cost;
    CharVec_t name; // Use char vector to store name (variable length)
};

// Car constructor
// This demonstrates the use of the vector constructor that takes
// two iterators and pushes all of the specified items on the vector.
// Also notice that "char*" is not a subclass of iterator, but this
// works anyway. Think about why this works
Car::Car(const char* n, int c) {
    cost = c;
    while(*n) name.push_back(*n++); // Add the name characters to name vector
}

// Car comparator
bool Car::operator<(const Car& rhs) {
    return cost < rhs.cost;
}

typedef vector<Car> CarVec_t; // Define a vector of cars

// Define output operators for CharVec_t and Car
ostream& operator<<(ostream& os, CharVec_t& cv) {
    for (CharVec_t::size_type i = 0; i < cv.size(); ++i)
        os << cv[i];
    return os;
}

ostream& operator<<(ostream& os, Car& car) {
    os << "Name " << car.name << " cost " << car.cost;
    return os;
}

// Define a subroutine to print an arbitrary vector

Program vector-iterators.cc
template <class ForwardIterator>
void PrintVec(ForwardIterator b, ForwardIterator e, bool addEndl = true)
{
    while(b != e)
    {
        cout << (*b++);
        if (addEndl) cout << endl;
    }
}

template <class ForwardIterator>
void Sort(ForwardIterator b, ForwardIterator e)
{ // This sort is inefficient, and used for illustrative purposes only
    while(b != e)
    {
        ForwardIterator i = b;
        while(i != e)
        {
            if (*i < *b)
            { // Need to swap. This iter_swap is defined in "algorithm"
                iter_swap(i, b); // Swap the two value.
            }
            ++i;
        }
        ++b;
    }
}

int main()
{
    CarVec_t cars; // Maintain a vector of cars
    cars.push_back(Car("Ferrari", 150000));
cars.push_back(Car("Toyota", 18000));
cars.push_back(Car("Yugo", 10000));
cars.push_back(Car("Volkswagen", 15000));
cars.push_back(Car("Ford", 20000));
cars.push_back(Car("Chrysler", 30000));
cars.push_back(Car("Mercedes", 60000));

    // Print each car using the indexing operator and integer index
    cout << "Printing indexing operator" << endl;
    for (CarVec_t::size_type i = 0; i < cars.size(); ++i)
    {
        cout << cars[i] << endl;
    }

    // Print each car using iterators
    cout << "Print using Iterators" << endl;
    CarVec_t::iterator it = cars.begin(); // Points to first element
    while(it != cars.end())
    { // Loop until end reached
        cout << (*it++) << endl;
    }

    cout << "Printing using the PrintVec subroutine" << endl;

Program vector-iterators.cc (continued)
Program vector-iterators.cc (continued)