// Demonstrate the use of the Standard Template Library "vector" class.
// and associated iterators, and templated subroutines
// George F. Riley, ECE3090 Georgia Tech, Fall 2005

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

using namespace std;

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

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

// Wheels 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
Wheels::Wheels(const char * n, int c) : cost(c), name(n, n + strlen(n))
{
    // while(*n) name.push_back(*n++); // Add the name characters to name vector
}

// Wheels comparator
bool Wheels::operator<(const Wheels& rhs)
{
    // Less than is defined as less cost
    return cost < rhs.cost;
}

typedef vector<Wheels> WheelsVec_t; // Define a vector of cars

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

ostream& operator<<(ostream& os, Wheels& 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)
{
  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()
{
  WheelsVec_t cars; // Maintain a vector of cars
  cars.push_back(Wheels("Ferrari", 150000));
  cars.push_back(Wheels("Toyota", 18000));
  cars.push_back(Wheels("Yugo", 10000));
  cars.push_back(Wheels("Volkswagen", 15000));
  cars.push_back(Wheels("Ford", 20000));
  cars.push_back(Wheels("Chrysler", 30000));
  cars.push_back(Wheels("Mercedes", 60000));

  // Print each car
  for (WheelsVec_t::size_type i = 0; i < cars.size(); ++i)
  {
    cout << cars[i] << endl;
  }

  cout << "Printing using the PrintVec subroutine" << endl;
  // Use the PrintVec templated subroutine
  PrintVec(cars.begin(), cars.end());

  // Sort the values
  Sort(cars.begin(), cars.end());
  cout << "After sorting" << endl;
  PrintVec(cars.begin(), cars.end());
}