```
1  // Demonstrate the use of the Standard Template Library "vector" class.
2  // George F. Riley, ECE4493/8893 Georgia Tech, Fall 2012
3
4  // A vector is a variable length array.  It starts out as "zero" length
5  // and grows or shrinks as needed.  Further, the vector is a array
6  // of any arbitrary type, using the C++ "templates" feature.
7
8  #include <iostream>
9  #include <vector>
10 class A {
11 public:
12   A();         // Default constructor
13   A(int);      // Non-Default Constructor
14   A(const A&); // A copy constructor is used by the compile whenever
15   A& operator=(const A&); // Assignment operator
16   ~A();        // Destructor
17 public:
18   int x;       // Single data member
19 };
20
21
22 typedef std::vector<A>  AVec_t; // Define a type that is vector of A's
23 typedef std::vector<A*> APVec_t;// Define a type that is vector of A pointers
24
25 A::A()
26 {
27   std::cout << "Hello from A::A() Default constructor" << std::endl;
28 }
29
30 A::A(int i)
31     : x(i)
32 {
33   std::cout << "Hello from A::A(int) constructor" << std::endl;
34 }
35
36 A::A(const A& a)
37     : x(a.x)
38 {
39   std::cout << "Hello from A Copy constructor" << std::endl;
40 }
41
42 A& A::operator=(const A& rhs)
43 {
44   std::cout << "Hello from A Assignment operator" << std::endl;
45   x = rhs.x;
46 }
47
48 A::~A()
49 {
50   std::cout << "Hello from A Destructor" << std::endl;
51 }
```

Program vector.cc

```
52  int main()
53  {
54    std::cout << "Creating A Vector"; getchar();
55    AVec_t av0;
56
57    std::cout << "Adding an three elements to av0"; getchar();
58    av0.push_back(A(2));    // Elements are appended using "push_back"
59    std::cout << "After first push_back"; getchar();
60    av0.push_back(A(10));   // Elements are appended using "push_back"
61    std::cout << "After second push_back"; getchar();
62    av0.push_back(A(100));  // Elements are appended using "push_back"
63    // Number of elements in a vector can be queried with "size()"
64    std::cout << "After third push_back, size av0 is "<<av0.size()<<std::endl;
65
66    // Now reserve space for  up to 10 elements, allowing for more
67    // efficient push_back.
68    std::cout << "Reserving 10 elements"; getchar();
69    av0.reserve(10);
70    // Push a few more to show better efficiency
71    std::cout << "Pushing three more elements"; getchar();
72    av0.push_back(A(101));
73    av0.push_back(A(102));
74    av0.push_back(A(103));
75
76    // Individual elements can be accessed with the [] operator
77    std::cout << "Accessing elements with the [] operator"; getchar();
78    std::cout << "av0[0].x is " << av0[0].x << std::endl;
79    std::cout << "av0[1].x is " << av0[1].x << std::endl;
80    std::cout << "av0[2].x is " << av0[2].x << std::endl;
81
82    // Front and back of list have special accessors
83    std::cout << "Accessing elements with the front and back"; getchar();
84    std::cout << "av0.front().x is " << av0.front().x << std::endl;
85    std::cout << "av0.back().x is "  << av0.back().x << std::endl;
86
87    // Vectors can be copied with copy constructor or assignment operator
88    std::cout << "Making a copy of av0"; getchar();
89    AVec_t av1(av0);
90    std::cout << "Size of av1 is " << av1.size() << std::endl;
91    std::cout << "av1[0].x is "    << av1[0].x << std::endl;
92
93    // Vectors can be shrunk with "pop_back".  Notice that pop_back
94    // does NOT return the element being popped
95    std::cout << "Shrinking av0 with pop_back"; getchar();
96    av0.pop_back(); // Remove last element
97    std::cout << "Size of av0 is " << av0.size() << std::endl;
98    av0.pop_back(); // Remove another element
99    std::cout << "Size of av0 is " << av0.size() << std::endl;
100
101   // Vectors can be initialized to "n" copies of a specified object
102   std::cout << "Constructing  AVec_t with 10 elements"; getchar();
103   AVec_t av2(10, A(1)); // Makes 10 elements of A(1)
104   std::cout << "Size of av2 is " << av2.size() << std::endl;
105   std::cout << "av2[0].x is " << av2[0].x << std::endl;
106
107   // All elements of a vector can be removed with "clear()"
```

Program vector.cc (continued)

```
108     std::cout << "Clearing av2"; getchar();
109     av2.clear();
110     std::cout << "Size of av2 is " << av2.size() << std::endl;
111
112     // Push another element to demonstrate that "clear" did not
113     // free the memory.
114     std::cout << "push another on av2"; getchar();
115     av2.push_back(A(100));
116
117     std::cout << "push another on av2"; getchar();
118     av2.push_back(A(200));
119
120     // Create and populate a vector of A poiners
121     std::cout << "Creating A Pointer Vector"; getchar();
122     APVec_t apv0;
123
124     std::cout << "Adding an three elements to apv0"; getchar();
125     apv0.push_back(new A(2));
126     apv0.push_back(new A(10));
127     apv0.push_back(new A(100));
128     // Number of elements in a vector can be queried with "size()"
129     std::cout << "Size of apv0 is " << apv0.size() << std::endl;
130
131     // Clear the apv0 vector.  Note: ~A() NOT called.  Why not?
132     std::cout << "Clearing apv0"; getchar();
133     apv0.clear();
134     std::cout << "Size of apv0 is " << apv0.size() << std::endl;
135
136     std::cout << "Main program exiting"; getchar();
137     return 0;
138 }
139
140 // Output from this program is:
141 //
142 // Creating A Vector
143 // Adding an three elements to av0
144 // Hello from A::A(int) constructor
145 // Hello from A Copy constructor
146 // Hello from A Destructor
147 // After first push_back
148 // Hello from A::A(int) constructor
149 // Hello from A Copy constructor
150 // Hello from A Copy constructor
151 // Hello from A Destructor
152 // Hello from A Destructor
153 // After second push_back
154 // Hello from A::A(int) constructor
155 // Hello from A Copy constructor
156 // Hello from A Copy constructor
157 // Hello from A Copy constructor
158 // Hello from A Destructor
159 // Hello from A Destructor
160 // Hello from A Destructor
161 // After third push_back, size av0 is 3
162 // Reserving 10 elements
163 // Hello from A Copy constructor
```

Program vector.cc (continued)

```
// Hello from A Copy constructor
// Hello from A Copy constructor
// Hello from A Destructor
// Hello from A Destructor
// Hello from A Destructor
// Pushing three more elements
// Hello from A::A(int) constructor
// Hello from A Copy constructor
// Hello from A Destructor
// Hello from A::A(int) constructor
// Hello from A Copy constructor
// Hello from A Destructor
// Hello from A::A(int) constructor
// Hello from A Copy constructor
// Hello from A Destructor
// Accessing elements with the [] operator
// av0[0].x is 2
// av0[1].x is 10
// av0[2].x is 100
// Accessing elements with the front and back
// av0.front().x is 2
// av0.back().x is 103
// Making a copy of av0
// Hello from A Copy constructor
// Hello from A Copy constructor
// Hello from A Copy constructor
// Hello from A Copy constructor
// Hello from A Copy constructor
// Hello from A Copy constructor
// Size of av1 is 6
// av1[0].x is 2
// Shrinking av0 with pop_back
// Hello from A Destructor
// Size of av0 is 5
// Hello from A Destructor
// Size of av0 is 4
// Constructing  AVec_t with 10 elements
// Hello from A::A(int) constructor
// Hello from A Copy constructor
// Hello from A Copy constructor
// Hello from A Copy constructor
// Hello from A Copy constructor
// Hello from A Copy constructor
// Hello from A Copy constructor
// Hello from A Copy constructor
// Hello from A Copy constructor
// Hello from A Copy constructor
// Hello from A Destructor
// Size of av2 is 10
// av2[0].x is 1
// Clearing av2
// Hello from A Destructor
// Hello from A Destructor
// Hello from A Destructor
// Hello from A Destructor
```

Program vector.cc (continued)

```
220 // Hello from A Destructor
221 // Hello from A Destructor
222 // Hello from A Destructor
223 // Hello from A Destructor
224 // Hello from A Destructor
225 // Hello from A Destructor
226 // Size of av2 is 0
227 // push another on av2
228 // Hello from A::A(int) constructor
229 // Hello from A Copy constructor
230 // Hello from A Destructor
231 // push another on av2
232 // Hello from A::A(int) constructor
233 // Hello from A Copy constructor
234 // Hello from A Destructor
235 // Creating A Pointer Vector
236 // Adding an three elements to apv0
237 // Hello from A::A(int) constructor
238 // Hello from A::A(int) constructor
239 // Hello from A::A(int) constructor
240 // Size of apv0 is 3
241 // Clearing apv0
242 // Size of apv0 is 0
243 // Main program exiting
244 // Hello from A Destructor
245 // Hello from A Destructor
246 // Hello from A Destructor
247 // Hello from A Destructor
248 // Hello from A Destructor
249 // Hello from A Destructor
250 // Hello from A Destructor
251 // Hello from A Destructor
252 // Hello from A Destructor
253 // Hello from A Destructor
254 // Hello from A Destructor
255 // Hello from A Destructor
```

Program vector.cc (continued)