```
1  // Demonstrate "Smart Pointers".
2  // ECE 3090 - Fall 2009
3  // George F. Riley, Georgia Tech, Fall 2006
4
5  // This example demonstrates the use of "smart pointers", that
6  // keep up with how many references there are to each allocated
7  // memory block, and only free the space when the reference count
8  // goes to zero.
9
10 #include <iostream>
11 #include <string>
12
13 using namespace std;
14
15 class SPointer
16 {
17 public:
18   SPointer(char*);
19   SPointer(const SPointer&);               // Need a copy constructor
20   ~SPointer();                             // Need a destructor
21   SPointer& operator=(const SPointer& rhs); // Need an assignment operator
22
23   char  Get(int i);          // Get the "i'th" character from the array
24   void  Set(int i, char c);  // Set the "i'th" character to the char 'c'
25   void  Set(const char* s);  // Set the string to s
26   void  Print(const string&);// Print the string
27 private:
28   char* pointer;   // This is the "shared" pointer
29   int*  refCount;  // This keeps up with how many references there are
30   int   lth;       // Length of shared memory
31 public:
32   static int allocCount;   // Debug..track count of alloc/deletes
33   static int deleteCount;
34 };
35
36 // Constructor
37 SPointer::SPointer(char* s)
38     : lth(strlen(s) + 1)
39 {
40   pointer = new char[lth]; // Allocate the dynamic memory
41   strcpy(pointer, s);
42   refCount = new int(1);   // Create the reference count variable, set to 1
43   allocCount++;            // For debug, count the allocations
44
45 }
46
47 // Copy constructor
48 SPointer::SPointer(const SPointer& c)
49     : pointer(c.pointer), refCount(c.refCount), lth(c.lth)
50 {
51   (*refCount)++;            // Increment the reference count
52 }
53
54 // Destructor
55 SPointer::~SPointer()
56 {
```

Program smartpointers.cc

```
57    (*refCount)--;            // Decrement the reference count
58    if (*refCount == 0)
59      { // This is the last reference, delete
60        delete [] pointer;
61        delete refCount;
62        deleteCount++;       // For debug, count the deletions
63      }
64  }
65
66  // Assignment operator
67  SPointer& SPointer::operator=(const SPointer& rhs)
68  {
69    if (pointer != rhs.pointer)
70      { // not self assignment
71        (*refCount)--;
72        if (*refCount == 0)
73          { // Last reference to my string, delete
74            delete [] pointer;
75            delete refCount;
76            deleteCount++;       // For debug, count the deletions
77          }
78        pointer = rhs.pointer;
79        refCount = rhs.refCount;
80        lth = rhs.lth;
81        (*refCount)++; // Count this reference
82      }
83    return *this;
84  }
85
86  // Get and Set functions
87  char SPointer::Get(int i)
88  {
89    if (i < lth) return pointer[i];
90    return '0'; // Out of range, just return 0
91  }
92
93  void  SPointer::Set(int i, char c)// Set the "i'th" character to the char 'c'
94  { // Set the new value.  However, we must make a copy of the data
95    // This is called "Copy on Write" semantics
96    if (i >= lth) return;                    // Out of range
97
98    if (*refCount > 1)
99      { // If this is not the only reference, we need to realloc and copy
100       (*refCount)--;                        // Decrement reference count
101       char* newPointer = new char[lth];     // Get a new memory array
102       refCount = new int(1);                // Get a new reference count = 1
103       memcpy(newPointer, pointer, lth);     // Copy the data
104       pointer = newPointer;                 // And set the new pointer
105       allocCount++;                         // Count the allocation
106     }
107   pointer[i] = c;
108 }
109
110
111 void  SPointer::Set(const char* s)         // Set the string to s
112 { // Length of s must be less than or equal to lth
```

Program smartpointers.cc (continued)

```
113     if (*refCount > 1)
114       { // If this is not the only reference, we need to realloc and copy
115         (*refCount)--;                        // Decrement reference count
116         char* newPointer = new char[lth];     // Get a new memory array
117         refCount = new int(1);                // Get a new reference count int
118         pointer = newPointer;                 // And set the new pointer
119         allocCount++;                         // Count the allocation
120       }
121     strncpy(pointer, s, lth);                 // Copy, but no more then "length"
122   }
123
124   void SPointer::Print(const string& prompt)
125   {
126     cout << prompt << " pointer is " << (void*)pointer
127          << " string is \"" << string(pointer)
128          << "\", refCount " << *refCount
129          << endl;
130   }
131
132   int SPointer::allocCount = 0;
133   int SPointer::deleteCount = 0;
134
135   void Sub()
136   { // Make a subroutine that tests SPointers.
137     // We use a subroutine so that all SPointers will go out of scope
138     // on exit, so we can verify the allocCount and deleteCount are the
139     // same.
140
141     SPointer sp1("This is a test");
142     SPointer sp2(sp1); // Copy constructor, sp2 shares the pointer with sp1
143
144     SPointer sp3("ShortString");
145     sp3 = sp1; // Assignment operator, sp3 also shares the pointer with sp1
146     sp1.Print("sp1");
147     sp2.Print("sp2");
148     sp3.Print("sp3");
149     cout << endl;
150
151     // Now change sp1, and notice that sp2/sp3 don't change
152     sp1.Set("Another test");
153     sp1.Print("sp1");
154     sp2.Print("sp2");
155     sp3.Print("sp3");
156     cout << endl;
157
158     // Change sp2, see that sp1 and sp3 are unchanged
159     sp2.Set(0, 'K');
160     sp1.Print("sp1");
161     sp2.Print("sp2");
162     sp3.Print("sp3");
163     cout << endl;
164   }
165
166   // Test program
167   int main()
168   {
```

Program smartpointers.cc (continued)

```
169     Sub();
170     cout << "AllocCount is  " << SPointer::allocCount << endl;
171     cout << "DeleteCount is " << SPointer::deleteCount << endl;
172   }
173
174   // The output from this program is:
175   //
176   // sp1 pointer is 0xe2400 string is "This is a test", refCount 3
177   // sp2 pointer is 0xe2400 string is "This is a test", refCount 3
178   // sp3 pointer is 0xe2400 string is "This is a test", refCount 3
179
180   // sp1 pointer is 0xe2720 string is "Another test", refCount 1
181   // sp2 pointer is 0xe2400 string is "This is a test", refCount 2
182   // sp3 pointer is 0xe2400 string is "This is a test", refCount 2
183
184   // sp1 pointer is 0xe2720 string is "Another test", refCount 1
185   // sp2 pointer is 0xe2770 string is "Khis is a test", refCount 1
186   // sp3 pointer is 0xe2400 string is "This is a test", refCount 1
187
188   // AllocCount is  4
189   // DeleteCount is 4
190
191
192
193
194
195
196
197
198
199
200
```

Program smartpointers.cc (continued)