```
1  // Demonstrate use of Templates for simple summation
2  // ECE4893/8893  - Fall 2011
3  // George F. Riley, Georgia Tech, Fall 2011
4
5  #include <iostream>
6
7  using namespace std;
8
9  // Define class A with various constructors and addition/multiplaction operator
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 compiler whenever
15                // a "copy" of an object is needed.
16   const A operator+(const A&) const;
17   const A operator*(const A&) const;
18   A& operator+=(const A&);
19   A& operator*=(const A&);
20
21 public:
22   int x;       // Single data member
23 };
24
25 // Implementation of class A
26 A::A()
27 {
28 }
29
30 A::A(int i)
31     : x(i)
32 {
33 }
34
35 A::A(const A& a)
36     : x(a.x)
37 {
38 }
39
40 // Implement the operators
41 const A A::operator+(const A& rhs) const
42 {
43   return A(x + rhs.x);
44 }
45
46 const A A::operator*(const A& rhs) const
47 {
48   return A(x * rhs.x);
49 }
50
51 A& A::operator+=(const A& rhs)
52 {
53   x += rhs.x;
54   return *this;
55 }
56
```

Program templateintroduction.cc

```
57  A& A::operator*=(const A& rhs)
58  {
59    x *= rhs.x;
60    return *this;
61  }
62
63  // Now define two functions to sum and product arrays of ints and doubles
64
65  // The summation functions
66  int SumInt(const int* p, unsigned length)
67  {
68    int result = 0;
69    for (int i = 0; i < length; ++i)
70      {
71        result += p[i];
72      }
73    return result;
74  }
75
76  int SumDouble(const double* p, unsigned length)
77  {
78    double result = 0;
79    for (int i = 0; i < length; ++i)
80      {
81        result += p[i];
82      }
83    return result;
84  }
85
86  // The product functions
87  int ProdInt(const int* p, unsigned length)
88  {
89    int result = 1;
90    for (int i = 0; i < length; ++i)
91      {
92        result *= p[i];
93      }
94    return result;
95  }
96
97  int ProdDouble(const double* p, unsigned length)
98  {
99    double result = 1;
100   for (int i = 0; i < length; ++i)
101     {
102       result *= p[i];
103     }
104   return result;
105 }
106
107 // Now, what if we want a summation/product of class A variables
108 A SumA(const A* p, unsigned length)
109 {
110   A result = 0;
111   for (int i = 0; i < length; ++i)
112     {
```

Program templateintroduction.cc (continued)

```
113         result += p[i];
114       }
115     return result;
116   }
117
118   A ProdA(const A* p, unsigned length)
119   {
120     A result = 1;
121     for (int i = 0; i < length; ++i)
122       {
123         result *= p[i];
124       }
125     return result;
126   }
127
128   // Also an output operator for A
129   ostream& operator<<(ostream& ofs, const A& rhs)
130   {
131     ofs << rhs.x;
132     return ofs; // This is important
133   }
134
135
136   // The above is silly; we wrote exactly the same code 6 times, the only
137   // difference is the type of the argument and return value.
138   // A C++ "Template" allows us to fix this problem.
139
140   template <typename T>
141   T SumTemplate(const T* p, unsigned length)
142   {
143     T result = 0;
144     for (int i = 0; i < length; ++i)
145       {
146         result += p[i];
147       }
148     return result;
149   }
150
151   template <typename T>
152   T ProdTemplate(const T* p, unsigned length)
153   {
154     T result = 1;
155     for (int i = 0; i < length; ++i)
156       {
157         result *= p[i];
158       }
159     return result;
160   }
161
162
163
164   int main()
165   {
166     int    i[10] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
167     double d[10] = { 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0};
168     A      a[10] = { A(1), A(2), A(3), A(4), A(5), A(6), A(7), A(8), A(9), A(10)};
```

Program templateintroduction.cc (continued)

```
169
170     int    sumi = SumInt(   i, sizeof(i) / sizeof(int));
171     double sumd = SumDouble(d, sizeof(d) / sizeof(double));
172     A      sumA = SumA(     a, sizeof(a) / sizeof(A));
173
174     int    prodi = ProdInt(   i, sizeof(i) / sizeof(int));
175     double prodd = ProdDouble(d, sizeof(d) / sizeof(double));
176     A      prodA = ProdA(     a, sizeof(a) / sizeof(A));
177
178     // Now illustrate the use of the templated versions
179     // This version uses "implicit instantiation
180     int    sumit = SumTemplate(i, sizeof(i) / sizeof(int));
181     double sumdt = SumTemplate(d, sizeof(d) / sizeof(double));
182     A      sumAt = SumTemplate(a, sizeof(a) / sizeof(A));
183
184     int    prodit = ProdTemplate(i, sizeof(i) / sizeof(int));
185     double proddt = ProdTemplate(d, sizeof(d) / sizeof(double));
186     A      prodAt = ProdTemplate(a, sizeof(a) / sizeof(A));
187
188     // Print out the results
189     cout << "sumi " << sumi << " sumit " << sumit << endl;
190     cout << "sumd " << sumi << " sumdt " << sumdt << endl;
191     cout << "sumA " << sumA << " sumAt " << sumAt << endl;
192
193     cout << "prodi " << prodi << " prodit " << prodit << endl;
194     cout << "prodd " << prodd << " proddt " << proddt << endl;
195     cout << "proda " << prodA << " prodAt " << prodAt << endl;
196
197 }
198
199
```

Program templateintroduction.cc (continued)