```
1  // Example illustrating Subclassing with virtual functions
2  // ECE3090
3  // George F. Riley, Georgia Tech, Spring 2009
4
5  #include <iostream>
6
7  using namespace std;  // We will discuss namespaces later
8
9  class TwoInt { // Simple class with two integer variables
10 public:
11   TwoInt(int a0, int b0);
12   // Note the use of the "virtual" keyword below
13   virtual void Print() const;   // Print the variables
14   int  First() const;   // Return the first int
15   int  Second() const;  // Return the second int
16   virtual int  Sum() const;     // Get the sum
17   virtual double Average() const ;
18   // Hello is NOT virtual
19   void Hello() const;
20   // Note the use of "private" here
21 private:
22   int a;
23   int b;
24 };
25
26 // Define a class FourInt that INHERITS from TwoInt
27 // This means the FourInt class has EVERYTHING that TwoInt has,
28 // plus any additional things we might add
29
30 class FourInt : public TwoInt {
31   // Inherits from TwoInt the members and functins
32 public:
33   FourInt(int a0, int b0, int c0, int d0);
34   virtual void Print() const; // Print the variables
35   int  Third() const;
36   int  Fourth() const;
37   virtual int  Sum() const;     // Get the sum
38   virtual double Average() const ;
39   // Hello is NOT virtual
40   void Hello() const;
41 private:
42   int c;
43   int d;
44 };
45
46 // Methods for TwoInt
47 TwoInt::TwoInt(int a0, int b0)
48   : a(a0), b(b0) // Initialize with correct constructors
49 { // Nothing else needed
50 }
51
52 void TwoInt::Print() const   // Print the variables
53 {
54   cout << "a " << a << ", b " << b << endl;
55 }
56
```

Program virtual-functions.cc

```
57  int  TwoInt::First() const   // Return the first int
58  {
59    return a;
60  }
61
62  int  TwoInt::Second() const  // Return the second int
63  {
64    return b;
65  }
66
67  int  TwoInt::Sum() const  // Return the sum
68  {
69    cout << "Hello from TwoInt::Sum()" << endl;
70    return a+b;
71  }
72
73  double  TwoInt::Average() const  // Return the average
74  {
75    cout << "Hello from TwoInt::Average()" << endl;
76    return Sum() / 2.0;
77  }
78
79  void TwoInt::Hello() const
80  {
81    cout << "Hello from TwoInt::Hello" << endl;
82  }
83
84  // Methods for FourInt
85  // Define the FourInt constructor, with four integers
86  FourInt::FourInt(int a0, int b0, int c0, int d0)
87    : TwoInt(a0, b0), c(c0), d(d0)
88  { // Nothing else needed
89  }
90
91  void FourInt::Print() const // Print the variables
92  {
93    TwoInt::Print();  // Let TwoInt "Print Itself"
94    cout << "c " << c << ", d " << d << endl;
95  }
96
97  int  FourInt::Third() const     // Get the third int
98  {
99    return c;
100 }
101
102 int  FourInt::Fourth() const    // Get the fourth int
103 {
104   return d;
105 }
106
107 int  FourInt::Sum() const  // Return the sum
108 {
109   cout << "Hello from FourInt::Sum()" << endl;
110   return TwoInt::Sum() + c + d;
111 }
112
```

Program virtual-functions.cc (continued)

```
113 double FourInt::Average() const // Compute average
114 {
115   return Sum() / 4.0;
116 }
117
118 void FourInt::Hello() const
119 {
120   cout << "Hello from FourInt::Hello" << endl;
121 }
122
123 void Sub1Ref(const TwoInt& ti)
124 { // Sub1Ref expects a "TwoInt" reference as a paramenter.
125   // We can pass any object of type TwoInt or any subclass of TwoInt
126   // We can call any TwoInt function,
127   cout << "Hello from Sub1Ref()" << endl;
128   ti.Print();
129   cout << "Sub1Ref() calling sum" << endl;
130   int s = ti.Sum();
131   cout << "Sub1Ref() calling average" << endl;
132   double avg = ti.Average();
133   cout << "Sum is " << s << " average " << avg << endl;
134   ti.Hello(); // What gets called here?
135 }
136
137 void Sub1Ptr(TwoInt* ti)
138 { // Sub1Ptr expects a "TwoInt" pointer as a paramenter.
139   // We can pass any object of type TwoInt or any subclass of TwoInt
140   // We can call any TwoInt function,
141   cout << "Hello from Sub1Ptr()" << endl;
142   ti->Print(); // Note different syntax from Sub1Ref() above
143   cout << "Sub1Ptr() calling sum" << endl;
144   int s = ti->Sum();
145   cout << "Sub1Ptr() calling average" << endl;
146   double avg = ti->Average();
147   cout << "Sum is " << s << " average " << avg << endl;
148   ti->Hello(); // What gets called here?
149 }
150
151 void Sub1Value(TwoInt ti)
152 { // Sub1Value expects a TwoInt BY VALUE.  Although this appears similar
153   // to the two examples above, it is quite a bit different.  This will
154   // become apparent when we discuss virtual functions.
155   cout << "Hello from Sub1Value()" << endl;
156   ti.Print();
157   cout << "Sub1Value() calling sum" << endl;
158   int s = ti.Sum();
159   cout << "Sub1Value() calling average" << endl;
160   double avg = ti.Average();
161   cout << "Sum is " << s << " average " << avg << endl;
162   ti.Hello(); // What gets called here?
163 };
164
165 int main()
166 {
167   TwoInt ti1(1, 2);
168   TwoInt ti2(2, 4);
```

Program virtual-functions.cc (continued)

```
169     FourInt fi1(10, 11, 12, 13);
170     FourInt fi2(fi1);
171     // We can call the Sub1 variants passing "TwoInt" objects as parameters
172     // either by reference, by pointer, or by value
173     cout << "Calling Sub1Ref() passing TwoInt" << endl;
174     Sub1Ref(ti1);
175     cout << "Calling Sub1Ptr() passing TwoInt" << endl;
176     Sub1Ptr(&ti2);
177     cout << "Calling Sub1Value() passing TwoInt" << endl;
178     Sub1Value(ti1);
179
180     // Note we can pass any subclass of TwoInt to the Sub1's
181     cout << "Calling Sub1Ref() passing FourInt" << endl;
182     Sub1Ref(fi1);
183     cout << "Calling Sub1Ptr() passing FourInt" << endl;
184     Sub1Ptr(&fi2);
185     // What happens when we pass a FourInt to Sub1Value?
186     cout << "Calling Sub1Value() passing FourInt" << endl;
187     Sub1Value(fi1);
188
189     cout << "Calling Hello on TwoInt and FourInt" << endl;
190     ti1.Hello();
191     fi1.Hello();
192   }
193
194
195
196
```

Program virtual-functions.cc (continued)