

```

1 // Operator overloading
2 // George F. Riley, Georgia Tech, Spring 2009
3
4 #include <stdio.h>
5 #include <iostream>
6
7 using namespace std;
8
9 // Define class A with a default constructor, non-default constructor,
10 // and a "Copy Constructor"
11 class A {
12 public:
13     A();          // Default constructor
14     A(int);       // Non-Default Constructor
15     A(const A&); // A copy constructor is used by the compiler whenever
16                 // a "copy" of an object is needed.
17 public:
18     int x;       // Single data member
19 };
20
21 A::A()
22 {
23     cout << "Hello from A::A() Default constructor" << endl;
24 }
25
26 A::A(int i)
27     : x(i)
28 {
29     cout << "Hello from A::A(int) constructor" << endl;
30 }
31
32 A::A(const A& a)
33     : x(a.x)
34 {
35     cout << "Hello from A::A(const A&) constructor" << endl;
36 }
37
38 A operator+(const A& lhs, const A& rhs)
39 {
40     cout << "Returning from A::operator+" << endl;
41     return A(lhs.x + rhs.x);
42 }
43
44
45 // Define a class B similar to A.
46 // But B uses "member function" operator overloading
47 class B {
48 public:
49     B();          // Default Constructor
50     B(int);       // Non-Default Constructor
51     B(const B&); // Copy constructor
52     ~B();        // Destructor
53     B operator+(const B& rhs); // Define member function addition operator
54 public:
55     int x;       // Single data member
56 };

```

Program operators.cc

```

57
58 B::B()
59     : x(0)
60 {
61     cout << "Hello from B::B() default constructor" << endl;
62 }
63
64 B::B(int i)
65     : x(i)
66 {
67     cout << "Hello from B::B(int) constructor" << endl;
68 }
69
70 B::B(const B& b)
71     : x(b.x)
72 {
73     cout << "Hello from B::B(const B&) constructor" << endl;
74 }
75
76 // Implement the member function addition operator
77 B B::operator+(const B& rhs)
78 {
79     cout << "Returning from B::operator+" << endl;
80     B r(x + rhs.x); // Note LHS is "x", RHS is "rhs.h"
81     return r;
82 }
83
84 B::~~B()
85 {
86     cout << "Hello from B::~~B() destructor" << endl;
87 }
88
89
90 // Define class C similar to A and B
91 // addition operator.
92 class C {
93 public:
94     C();           // Default constructor
95     C(int);       // Non-Default Constructor
96     C(const C&);  // Copy constructor
97 public:
98     int x;       // Single data member
99 };
100
101 C::C()
102     : x(0)
103 {
104     cout << "Hello from C::C() default constructor" << endl;
105 }
106
107 C::C(int c)
108     : x(c)
109 {
110     cout << "Hello from C::C(int) constructor" << endl;
111 }
112

```

Program operators.cc (continued)

```

113 C::C(const C& c)
114     : x(c.x)
115 {
116     cout << "Hello from C::C(const B&) constructor" << endl;
117 }
118
119 // Non-member addition operator for C
120 C operator+(const C& lhs, const C& rhs)
121 {
122     cout << "Returning from C::operator+, non-member function" << endl;
123     return C(lhs.x + rhs.x);
124 }
125
126 // We can also make an addition operator to add an A and B, returning A
127 A operator+(const A& lhs, const B& rhs)
128 {
129     return A(lhs.x + rhs.x);
130 }
131
132 int main()
133 {
134     cout << "Creating A objects"; getchar();
135     A a0(1);
136     A a1(10);
137     A a2;
138     cout << "Adding a0 + a1"; getchar();
139     a2 = a0 + a1;
140
141     cout << "Creating B objects"; getchar();
142     B b0(2);
143     B b1(20);
144     B b2;
145     B b3;
146     cout << "Adding b0 + b1"; getchar();
147     b2 = b0 + b1;
148     cout << "Done adding" << endl;
149     cout << "Adding b0 and 5" << endl;
150     b2 = b0 + 5; // Why does this compile?
151     cout << "Resulting b2 is " << b2.x << endl;
152     // try this
153     b2 = B(5) + b0;
154
155     // Add an A and B object
156     cout << "Adding a0 + b0"; getchar();
157     a2 = a0 + b0;
158     cout << "Resulting a2 is " << a2.x << endl;
159
160     cout << "Creating C objects"; getchar();
161     C c0(3);
162     C c1(30);
163     C c2;
164
165     cout << "Adding c0 + c1"; getchar();
166     c2 = c0 + c1;
167     cout << "Resulting c2 is " << c2.x << endl;
168     return 0;

```

Program operators.cc (continued)

```

169 }

```

Program operators.cc (continued)