OPERATOR OVERLOADING

In C++, you can use the 'operator' keyword to overload normal C operators to perform basic functions like adding and subtracting the way you would normally, by using the '+' and '-' symbols. Thus, instead of creating a function to do something like this:

```
Vector = myAddTwoVectors( Vector1, Vector2 );
```

you would overload the '+' operator so that the above statement would look like this:

```
Vector = Vector1 + Vector2;
```

The mechanics for operator overloading are shown in the following code segment:

```
// vector.cp
#include <iostream.h>

class Vector

   // friend needed to overload ios operators
   friend ostream &operator<<( ostream &os, const Vector &V );
   // note that implicit 'this' object not sent to friend
   // function outside scope of class

   public:
      // default constructor
      Vector() x=0; y=0;          // inline
      // shorthand assignment constructor
      Vector( const float a, const float b ) : x(a), y(b)
      // overload operators
      Vector operator+( const Vector &V );
      Vector &operator+=( Vector &V );
      Vector &operator=( Vector &V );
      Vector &operator++();         // prefix
      Vector operator++( int );   // postfix - requires dummy argument

   private:
      float x;
      float y;
;

Vector Vector::operator+( const Vector &V )

   return Vector( x + V.x, y + V.y );


Vector &Vector::operator+=( Vector &V )

   x = x + V.x;
   y = y + V.y;
   return *this;


Vector &Vector::operator=( Vector &V )

   // no different than default operator, but could be
   x = V.x;
   y = V.y;
   return *this;
```

```
Vector &Vector::operator++()

   ++x;
   ++y;
   return *this;


Vector Vector::operator++( int )

   Vector V;
   V = *this;
   x++;
   y++;
   return V;


ostream &operator<<( ostream &os, const Vector &V )

   return os << "Vector[ " << V.x << ", " << V.y << " ]";


main()

   Vector vectorA( 3, 4 );
   Vector vectorB( 2, 7 );

   // output line 1
   cout << vectorA << " + " << vectorA << " = ";
   vectorA += vectorA;
   cout << vectorA << endl;

   // output line 2
   cout << vectorA << " + " << vectorB << " = "
      << vectorA + vectorB << endl;

   // output line 3
   vectorA = vectorB;
   cout << vectorA << endl;

   // output line 4
   cout << ++vectorA << endl;

   return 0;

// end vector.cp
```