Chapter 14:

More About Classes

14.5

Operator Overloading

Operator Overloading

- Operators such as =, +, and others can be redefined when used with objects of a class
- The name of the function for the overloaded operator is `operator` followed by the operator symbol, e.g.,

  `operator+` to overload the + operator, and

  `operator=` to overload the = operator
- Prototype for the overloaded operator goes in the declaration of the class that is overloading it
- Overloaded operator function definition goes with other member functions

Notes on Overloaded Operators

- Can change meaning of an operator
- Cannot change the number of operands of the operator

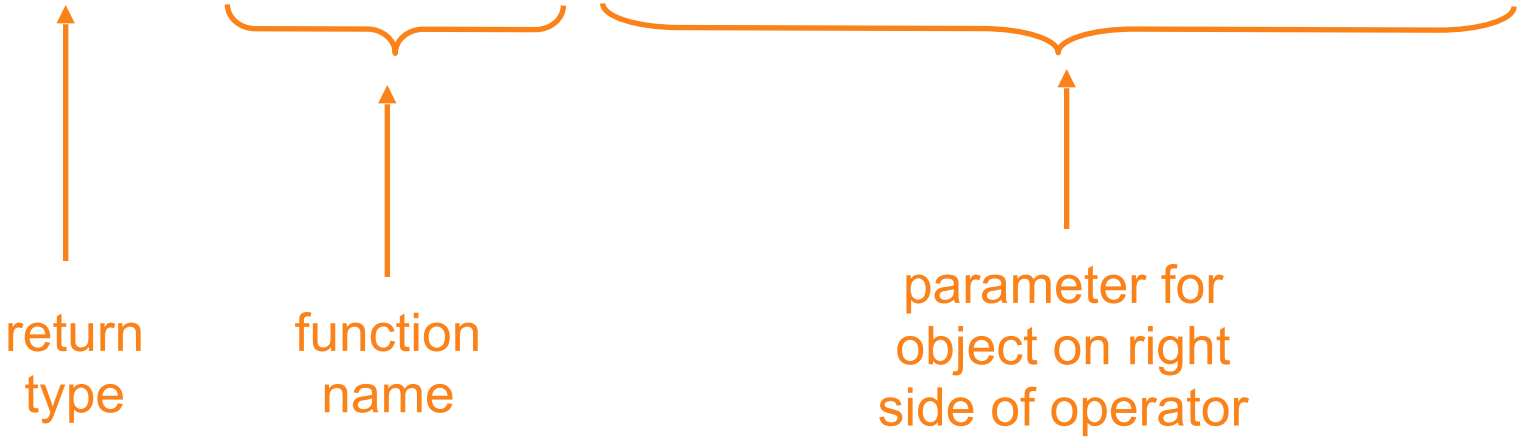
- Only certain operators can be overloaded.
- Cannot overload the following operators:

  scope operator ::
  sizeof
  member selector .
  member pointer selector *
  ternary operator ?:

Operator Overloading

- Prototype:

```
void operator=(const SomeClass &rval)
```



- Operator is called via object on left side

Invoking an Overloaded Operator

- Operator can be invoked as a member function:

```
object1.operator=(object2);
```

- It can also be used in more conventional manner:

```
object1 = object2;
```

Invoking an Overloaded Operator

- Review the attached example, over_asgn.cpp

Returning a Value

- Return type the same as the left operand supports notation like:

```
object1 = object2 = object3;
```

- Function declared as follows:

```
const SomeClass operator=(const someClass &rval)
```

- In function, include as last statement:

```
return *this;
```

Returning a Value (cont)

- Review the attached example over_asgn_asgn.cpp

Returning a Value (cont)

- Overloaded operator can return a value (example: operator.cpp)

```
class Point2d
{     int x, y;
      double square(double v)
      {     return v * v;
      }
      public:
      Point2d(int px, int py)
      {     x = px; y = py;
      }
      double operator-(const Point2d &right)
      {     return sqrt(square(x-right.x)
                        + square(y-right.y));
      }
};

int main()
{
      Point2d point1(2,2), point2(4,4);
      // compute and display distance between 2 points
      cout << point2 - point1 << endl;
      return 0;
}
```

Returning a Value (cont)

- Overloaded operator can return a value (over_plus.cpp)

```
class Point2d
{       int x, y;
        double square(double v)
        {       return v * v;
        }
        public:
        Point2d(int px, int py)
        {       x = px; y = py;
        }

       Point2d operator-(const Point2d &right)
        {       int nx = x+right.x;
                int ny = y+right.y;
                Point2d temp (nx, ny);
                return temp;
        }
};

int main()
{
        Point2d point1(2,2), point2(5,7), point3;
        // what about the following?
        point3 = point2 + point1;

        return 0;
}
```