OVERLOADED OPERATORS AND friend FUNCTIONS

Reusing operators

Delroy A. Brinkerhoff

OPERATORS

- An operator is a function with a special calling syntax
- “Regular” functions:
  - y = sqrt(x);
  - p = pow(b, e)
- Operators
  - z = x + y
  - -n
  - new person(“Dilbert”);

OPERATOR OVERLOADING

- Is a form of function overloading (i.e., they are functions named operator☺ where ☺ is an overloadable operator)
- Does not change the meaning of any operator for an intrinsic data type
- Cannot alter the precedence or associativity of an operator
- Cannot change the number of arguments
- Cannot create a new operator (e.g., **)
- Overloaded operators should be used intuitively (e.g., in a way similar to the original meaning)

friend FUNCTIONS

- friend functions are not members of a class, but are still allowed access to private class features
- A function may be a friend of more than one class (called a bridge function)
- A function must be declared as a friend in a class
  - Can be inline
  - Can be defined outside of a class
- friend functions are most often used with overloaded operators

OPERANDS

- Operands are the data that operators operate on (i.e., function arguments)
- Operators are characterized by the number of operands that they require
- Unary
  - 1 operand: *x, &a, -n
- Binary
  - 2 operands: x*y, x+y, cout << x

	Explicit Arguments	
	Unary	Binary
Member	0	1
friend	1	2

Function calls:

```
implicit.function(explicit);
function(explicit, explicit);
```