Operators
Lecture 10

Robb T. Koether

Hampden-Sydney College

Fri, Feb 6, 2009
Definition (Operator)

An operator is a function that can be represented by a symbol, such as + or *.

- Different operators may have different numbers of arguments
  - Unary operators have 1 argument.
  - Binary operators have 2 arguments.
  - Ternary operators have 3 arguments.
Operators as Functions

- Unary operators may be **prefix** or **postfix**.
- Most operators are binary.
- Most unary operators are prefix.
Most operators can be overloaded.

- **Unary:** +, −, *, &
- **Arithmetic:** +, −, *, /, %
- **Equality:** ==, !=
- **Order:** <, >, <=, >=
- **Logical:** &&, |, ||, !
- **Bitwise:** &, |, ~, ^
- **Shift:** <<, >>
- **Assignment:** =
- **Compound assignment:** +=, -=, *=, /=, %=, &=, |=, ^=, <<=, >>=
- **Increment and decrement:** ++, −
- **Allocation:** new, delete
- **Miscellaneous:** ,, −>*, −>, (), [ ]
A few operators cannot be overloaded.

- Member access: .
- Member access: .*
- Scope: ::
- Selection: ?:
- Size of: sizeof
The name of an operator function consists of the keyword `operator` followed by the symbol for the operator.

The expression

\[ a + b \]

is interpreted as

\[ \text{operator+}(a, b) \]
Example (OperatorFunctions.cpp)

- Download and run OperatorFunctions.cpp.
Binary Operators as Member Functions

Operator as Member Function

Type ClassName::operator+(Parameters);

- An operator may be defined as a member function of a class.
- That may or may not be a good idea.
- A binary operator is invoked by the left operand of the expression.
- Thus, the expression \( a + b \) is interpreted as \( a\cdot\text{operator+}(b) \).
Binary Operators as Member Functions: Considerations

- **Advantage**
  - The operator has access to the private members of the left operand.

- **Disadvantages**
  - If $a$ and $b$ are objects of different classes, then $a + b$ and $b + a$ will invoke different functions.
  - The left operand may be a member of a class that we do not have access to.
Implementing Binary Operators with Facilitators

**Definition (Facilitator)**

A facilitator is a member function that is invoked by a non-member operator.

- The facilitator performs the work of the operator.
- The operator simply
  - Invokes the facilitator.
  - Returns the appropriate object, typically the same one that is returned by the facilitator.
A binary operator has two parameters.
The corresponding facilitator has one parameter, namely, the right operand.
If we write the facilitator as a member function, then we write the operator as a non-member function.
The operator is invoked by the operands as an *ordered pair*.
We may then use *either* operand to invoke the facilitator.
The Prototypes

- The prototype of the facilitator is
  \[
  \text{Type1 Type2::FacilitatorName}(\text{Type3 obj3});
  \]

- The prototype of the operator is
  \[
  \text{Type1 operator+}(\text{Type2 obj2, Type3 obj3});
  \]
  or
  \[
  \text{Type1 operator+}(\text{Type3 obj3, Type2 obj2});
  \]
Binary Operators with Facilitators: Considerations

- **Advantages**
  - The left operand need not be an object of the same class as the facilitator.
  - The expressions $a + b$ and $b + a$ can be handled by the same facilitator, even if $a$ and $b$ are objects of different types.

- **Disadvantage**
  - Requires an additional function call.
Using Operators with Mixed Types

Example (Operators with Mixed Types)

```cpp
Complex operator+(const Complex& c, int n) {
    return c.add(Complex(n));
}
Complex operator+(int n, const Complex& c) {
    return c.add(Complex(n));
}
```