Operator Overloading

C++ Has Many Operators

- Arithmetic: + - * / %
- Equality: == !=
- Relational: < <= > >=
- Increment and decrement: ++ --
- Input and output: >> <<
- Assignment: =
- Subscript: [ ]
- Access (pointer): * ->
- And other less-known ones

Operators Are Functions

- They have special names: the reserved word *operator* followed by the operator's symbol.
- Less-than (<) is the same as operator<
- Normal functions are directly called:
    - MyFunction (name1, name2);
- Operators are indirectly called when they're encountered:

```
if (name1 < name2)
        becomes internally
if (operator<(name1, name2))
```

Operators and Arguments

- Most operators are binary operators; they have two arguments, the lefthand side and the righthand side.
- Some operators are unary operators; they have only one argument. They aren't covered in this presentation.
- Arguments can be:
    - built-in types such as *int* or *float*
    - User-defined types—structs and classes
- Operators for built-in types are built into the compiler
- Operators for user-defined types must be overloaded—specially defined for each type.
- Most often, it's the relational operators that need overloading
- Only a few operators, usually < or ==, need overloading

Two Types of Operator Functions

Member

- Declared inside classes:

```
class       MyClass
{
      bool              operator<(const type& rhs)
}
```

- The lefthand side argument is the class object.

- **Non-member**
- Declared outside classes:

```
bool       operator<(const type& lhs, const type& rhs)
```

- Have the lefthand side and the righthand side style.

Member or Non-member?

- Some operators must be only one of these two possibilities
- Some operators can be either, but there's a recommended choice.

Must Be Member Functions

- Assignment =
- Subscript [ ]
- Member access ->

Must Be Non-member Functions

- Input and output >> <<

Should Be Member Functions

- Access (pointer) *
- Compound assignment += -= *= /= %=
- Increment and decrement ++ --

Should Be Non-member Functions

- Equality == !=
- Relational < <= > >=
- Arithmetic + - * / %

Common Example: operator<

- Comparing a Person structure that has last name and first name members

```
struct     Person
{
   string   lastName_;
   string   firstName_;
}
```

- If the last names are different, they're the only members that must be compared
- If the last names are equal, the first names must be compared

Code For Non-member (Recommended) Function

```
if (lhs.lastName_ < rhs.lastName_)
        return(true);
    else if (lhs.lastName_ > rhs.lastName_)
        return(false);
    else
        if (lhs.firstName_ < rhs.firstName_)
            return(true);
        else
            return(false);
```

Code For Member (Permitted) Function

```
if (lastName_ < rhs.lastName_)
        return(true);
    else if (lastName_ > rhs.lastName_)
        return(false);
    else
        if (firstName_ < rhs.firstName_)
            return(true);
        else
            return(false);
```

- The lefthand side is the object that the operator is a member function of