Operator Overloading

Overloading

- Refers to multiple meanings of the same name or symbol
  - For example: An overloaded function is a function with multiple definitions
- Operator overloading refers to multiple definitions of operators such as: +, - , *, /, + +, =, <<, >>, and more.

For Example

- The operators +, -, *, and /, as we have seen, are all already overloaded
  - Indeed, 3/4 = 0 is integer division
  - But, 3.0/4.0 = 0.75 for floating point division
- Subtle but true and we, the programmers, have to be very careful about this difference.

Other Overloaded Operators

- >> is overloaded so that class object cin can be used to read data
- << is also overloaded so that the class object cout can be used to write data
  - Note that these operators were originally defined as bitwise shift operators
  - Now that they are overloaded, the same symbols can be used for output of many different types of output: screen, file, and so on.

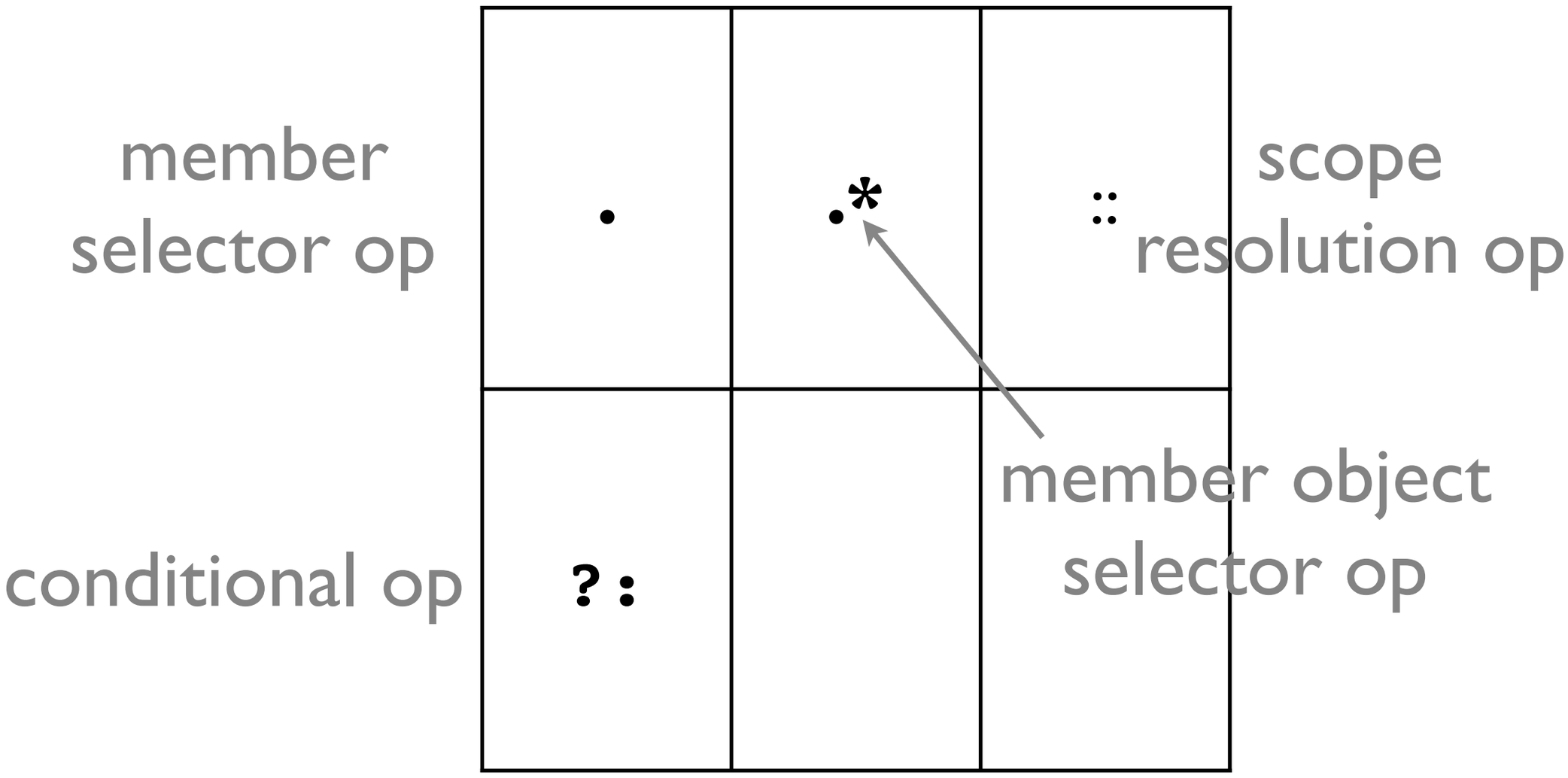
The following operators can be overloaded

+	-	*	/	%	^	&
|	~	!	=	<	>	+=
-=	*=	/=	%=	^=	&=	|=
<<	>>	<<=	>>=	==	!=	<=
>=	&&	||	++	--	,	->*
->	( )	[ ]	new	new[ ]	delete	delete[ ]

The function call op

The subscript op

The following operators cannot be overloaded



An Example

```
class Over {
public:
   Over operator+ (Over &c2)
   //......
};
```

following the usual syntax for invoking a method, *operator+* can be invoked as:

```
Over  c2, c3, c4;
c4 = c3.operator+(c2);
```

where c2, c3, c4 are objects of the class Over

An Example continued

c4 = c3.operator+(c2); // can be invoked as c4 = c3 + c2;

How about a one parameter (unary) operator example:

```
class Over{
public:
   Over& operator!( );
   //....
};
```

Notice that the unary operator *!* has no parameters, unlike the binary operator + which uses one parameter.

How About Multiplication

```
class Over {
public:
   Over operator* (Over &c2)
   //......
};
```

following the usual syntax for invoking a method, *operator** can be invoked as:

```
Over  c2, c3, c4;
c4 = c3.operator*(c2);
```

Lets look at the program