Operators

Nguyen Van Vu
nvu@fit.hcmus.edu.vn

Topics

- Operators
- Friend function

Operators

- We can do the following for built-in types

```
void main()
{
        int   a, b;
        int   c = a + b;
}
```

- We define classes, we also want to do the same for two objects, like below

```
void main()
{
        MyString   str1, str2;
        MyString   str3 = str1 + str2;
}
```

Defining an operator in C++

- C++ provides a special function “operator” whose name is an operator (a math symbol)

- Declaration syntax:

  [return-type] operator <math-symbol>(params)

- Example:

  MyString operator +(Mystring s1, MyString s2);

- After implementing, we can use

  MyString str3 = str1 + str2;

Two types of operator

- Independent operator

  Fraction ***operator +****(Fraction p1, Fraction p2);*

  - Does not belong to any class
  - Number of arguments = operator n-nary.

- Class operator

  Fraction ***Fraction::operator +****(Fraction p);*

  - A method of class
  - Number of arguments = operator n-nary - 1

- **They act the same!!**

Operators that can be redefined

N-nary	Group	Operator
Unary	Inc / Dec	++, --
	Math sign	+, -
	Bit	!, ~
	Pointer	*, &
	Type-cast	int, float, double, …
Binary	Arithmetic	+, -, *, /, %
	Comparison	>, <, ==, >=, <=, !=
	Logic	&&, ||, &, |
	Input / Output	<<, >>
	Assignment	=, +=, -=, *=, /=, %=
	Array indexing	[ ]

Limitations for operators

- We cannot create a new operator (we redefine instead)
- We cannot redefine operators for build-in types
- We cannot change operator n-nary
- We cannot change operator precedence order

Operator function

- Dr. Guru:
  - Rule of re-defining operator:
    - Name: **operator <math symbol>**
    - Arguments: **n-nary**
    - Return type: **operator result**

- Practice:
  - Operator > (Fraction)
  - Operator [ ] (Array)

Special operators

- Assignments (=, +=, -=, *=, /=, …):
  - Provide operator **+=** for **Fraction**.
  - n-nary?
  - Return result?

```
Fraction&  Fraction::operator +=( const Fraction &p );
```

Special operators

- Increasing / Decreasing (++, --):
  - Provide operator **++** for **Fraction**
  - n-nary?
  - Return result?
  - Prefix vs. posfix?

```
Fraction&  Fraction::operator ++( );          // Prefix.
Fraction  Fraction::operator ++( int x );     // Posfix, fake argument.
```

Friend function

- Operator +
  - Provide operator **+** for **Fraction**
  - Use independent operator

    Fraction ***operator +*** (const Fraction &p1, const Fraction &p2);
  - How to access ***private members***?

- Operator <<
  - Provide operator **<<** for **Fraction**

    ```
    Fraction  p( 1, 3 );
    cout  <<  p;
    ```
  - Which class operator **<<** belongs to?

Friend function

- Friend is a function that can access class **private** members
  - Declaration: ***friend*** **<method>**, inside class
  - Implementation: do not use keyword ***friend***, outside class

```
class Fraction
{
        friend ostream& operator <<(ostream &os, const Fraction &p);
};

ostream& operator <<(ostream &os, const Fraction &p)
{
        os << p.m_num << "/" << p.m_denom << endl;
        return os;
}
```

Concept summary

- Operator function
- Friend function

Practice

- Let's define and implement a Fraction class which represents a fraction number with the following operators
  - Arithmetic: +, *
  - Comparison: >, <, ==, >=, <=, !=
  - Assignment: =, +=, *=
  - Increasing / Decreasing: ++, -- (add/subtract 1 unit)
  - Type-cast: (float), (int)
  - Input/Output: >>, <<