HANYANG UNIVERSITY

창의적 소프트웨어 프로그래밍
실습 교재_06

Overview

- 목표
  - Operator Overloading
  - Friend Class and Function

Operator Overloading

- Argument에 의해 분리
  - int operator+ (int n, int m){ … }
  - int operator+ (double n, double m){ .. }
- const 에 의해 분리
  - int operator+ (int n, int m){ … }
  - const int operator+ const (int n, int m){ .. }

Friend Class and Function

- one-way!
- why do we need a "friend"?

Example

```
#include <stdio.h>
#include <string>

class Complex;

class Tester
{
public:
   double testfunc(Complex& c);
};
```

Example

```
struct Complex
{
public:
   Complex() : real(0.0), imag(0.0) {}
   Complex(double v) : real(v), imag(0.0) {}
   Complex(double r, double i) : real(r), imag(i) {}
   Complex(const Complex& c) : real(c.real), imag(c.imag) {}

   Complex& operator = (const Complex& c){
      real = c.real, imag = c.imag;
      return *this;
   }

   Complex operator+ () const { return *this; }
   Complex operator- () const { return Complex(-real, -imag); }
```

Example

```
double& operator[] (int i) {
    printf("no const\n");
    return i == 0 ? real : imag;
}

const double& operator[] (int i) const {
    printf("const\n");
    return i == 0 ? real : imag;
}

void show(std::string sz_prefiex){
    printf("[%s] real: %lf, imag: %lf\n", sz_prefiex.c_str(), real, imag);
}
```

Example

```
private:
   double real, imag;

   friend Complex operator+ (const Complex& lhs, const Complex& rhs);
   friend bool operator< (const Complex& lhs, const Complex& rhs);

   friend double Tester::testfunc(Complex &c);
};
```

Example

```
double Tester::testfunc(Complex& c){
    printf("[Tester] %lf, %lf", c.real, c.imag);
    return c.real;
}


Complex operator+ (const Complex& lhs, const Complex& rhs){
    return Complex(lhs.real + rhs.real, lhs.imag + rhs.imag);
}


bool operator< (const Complex& lhs, const Complex& rhs){
    return lhs.real < rhs.real && lhs.imag < rhs.imag;
}
```

Example

```
int Test(){
   Complex a(11.0, 2.0), b(2.0, 5.0), c;
   const Complex cc(33.0, 11.0);
   Tester t;

   // overloading test
   a[n];
   cc[1];

   c = a + b;
   c.show("c = a + b");
   (a+b).show("(a+b)");

   // friend test
   t.testfunc(a);

   return 0;
}
```

Example

```
int main(){
    Test();


    return 0;
}
```

Appendix #1. Member vs Global Overloading

/* Member Function */

```
Complex& operator+ (const Complex& c) const
{
        return Complex(real + c.real, imag + c.imag);
}
```

/* Global Function */

```
Complex operator+ (const Complex& lhs, const Complex& rhs)
{
   return Complex(lhs.real + rhs.real, lhs.imag + rhs.imag);
}
```

Appendix #2. Return Value vs Reference

- 사전에 정의된 별도 의미는 없다
- Value와 Reference의 특성을 따른다!
  - 오버로딩함수 스택 내에서 생성한 변수를 Reference 반환해서는 안된다
    - 스택이 종료되면 변수의 메모리 반환 가능
    - Value로 반환해야 한다
  - Reference 형태 인자를 받은 경우에는 해당 인자를 Reference로 반환