Technology Tarena 达内科技

标准C++语言

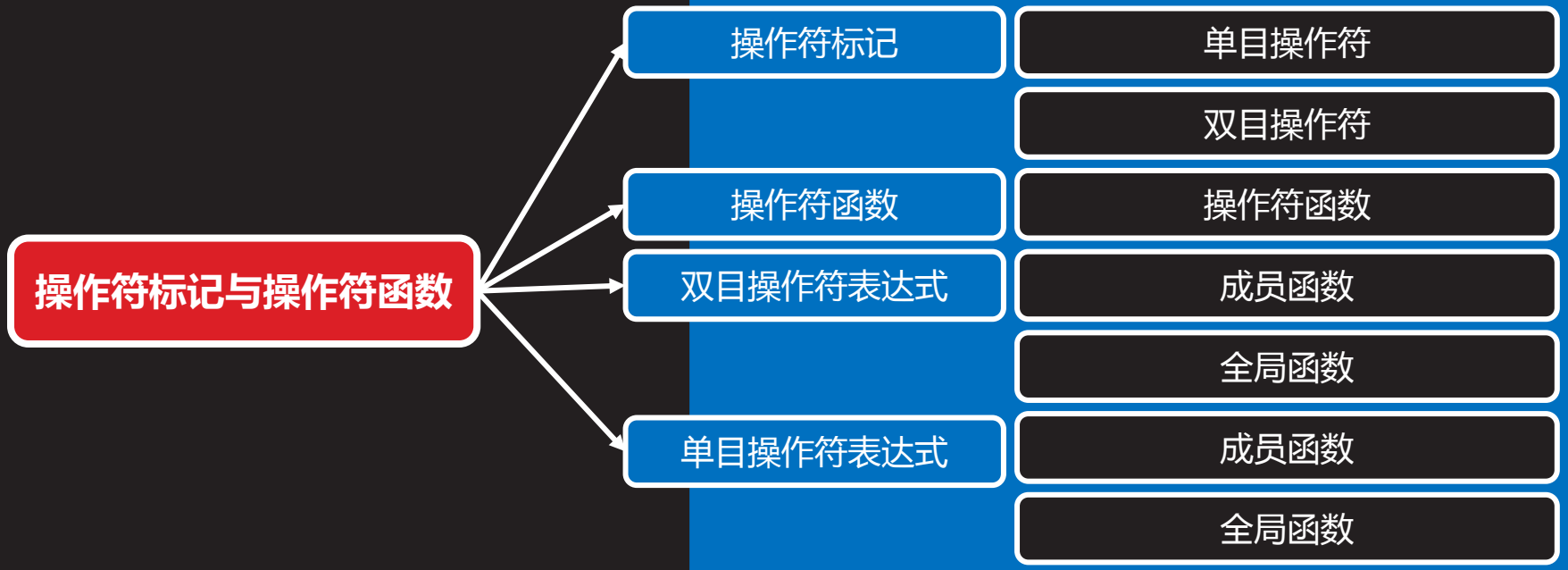
PART 1

DAY05

内容

上午	09:00 ~ 09:30	作业讲解和回顾
	09:30 ~ 10:20	操作符标记与操作符函数
	10:30 ~ 11:20	典型双目操作符的重载
	11:30 ~ 12:20	
下午	14:00 ~ 14:50	典型单目操作符的重载
	15:00 ~ 15:50	
	16:00 ~ 16:50	输入输出操作符的重载
	17:00 ~ 17:30	总结和答疑

操作符标记与操作符函数



Technology Tarena 达内科技

操作符标记

单目操作符

- 只有一个操作数的操作符
  - 相反数：-
  - 自增：++
  - 自减：--
  - 取地址：&
  - 解引用：*
  - 间接成员访问：->
  - 逻辑非：!
  - 位反：~
  - 函数：()
  - 类型转换

知识讲解

双目操作符

- 有左右两个操作数的操作符
  - 算术运算：*、/、%、+、-
  - 关系运算：>、>=、<、<=、==、!=
  - 逻辑运算：&&、||
  - 位运算：&、|、^、<< 、>>
  - 赋值与复合赋值：=、+=、-=、*=、/=、%=、&=、|=、^=、<<= 、>>=
  - 下标：[]

知识讲解

Technology Tarena 达内科技

操作符函数

操作符函数

- 在特定条件下，编译器有能力把一个由操作数和操作符共同组成的表达式，解释为对一个全局或成员函数的调用，该全局或成员函数被称为操作符函数
- 通过定义操作符函数，可以实现针对自定义类型的运算法则，并使之与内置类型一样参与各种表达式

知识讲解

Technology Tarena 达内科技

双目操作符表达式

成员函数

- 形如L#R的双目操作符表达式，将被编译器解释为
  - L.operator# (R)
    左操作数是调用对象，右操作数是参数对象
  - a - b + c
    a.operator- (b).operator+ (c)
  - a - (b + c)
    a.operator- (b.operator+ (c))
  - a - b * c
    a.opertor- (b.operator* (c))

知识讲解

全局函数

- 形如L#R的双目操作符表达式，将被编译器解释为
  - ::operator# (L, R)
    左操作数是第一参数，右操作数是第二参数
  - a - b + c
    ::operator+ (::operator- (a, b), c)
  - a - (b + c)
    ::operator- (a, ::operator+ (b, c))
  - a - b * c
    ::operator- (a, ::operator* (b, c))

知识讲解

单目操作符表达式

成员函数

- 形如#O或O#的单目操作符表达式，将被编译器解释为
  - O.operator# ()
    唯一的操作数是调用对象
  - -a
    a.operator- ()
  - ++a
    a.operator++ ()
  - a++
    a.operator++ (0)

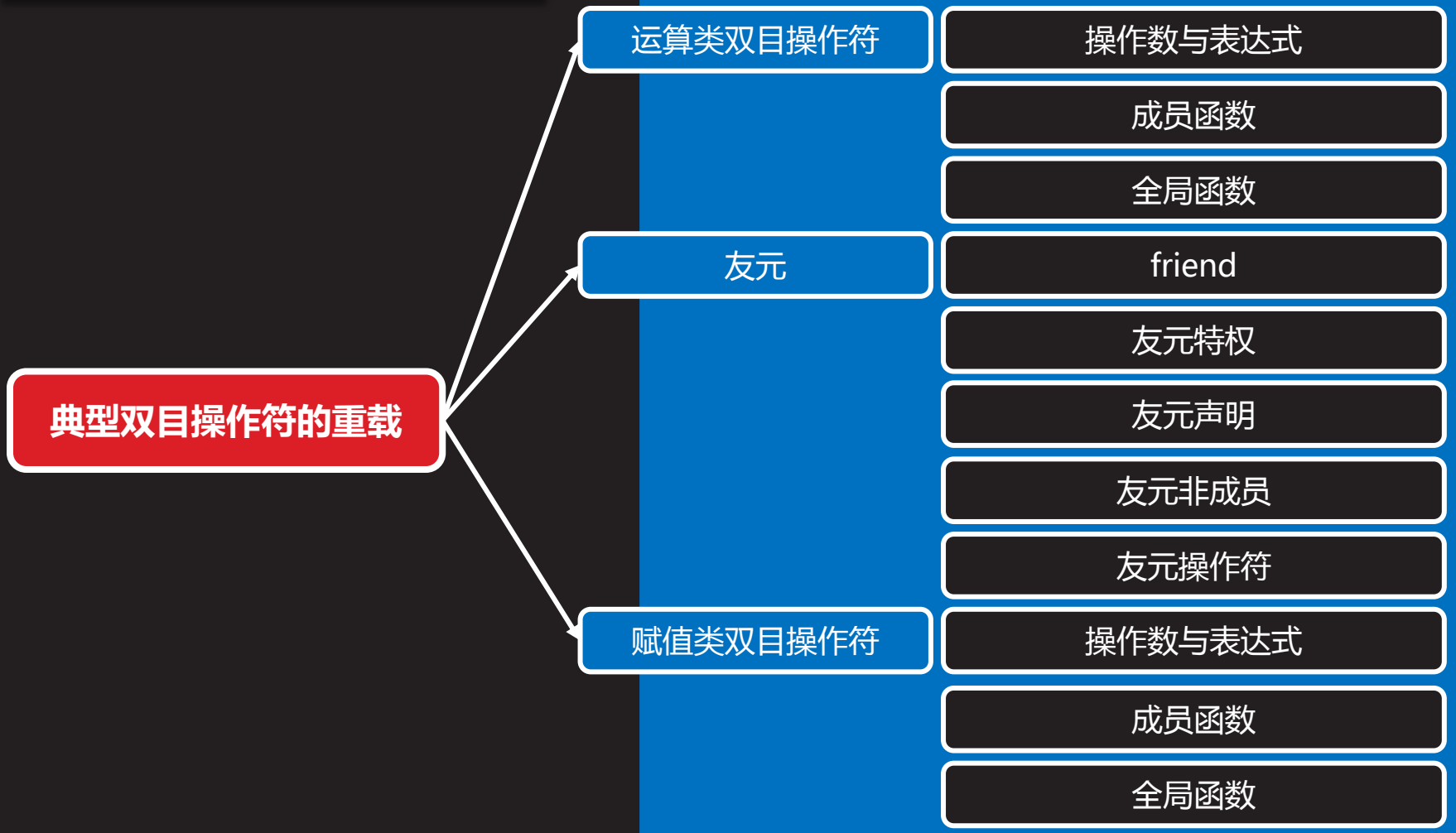
知识讲解

全局函数

- 形如#O或O#的单目操作符表达式，将被编译器解释为
  - ::operator# (O)
    唯一的操作数是参数对象
  - -a
    ::operator- (a)
  - ++a
    operator++ (a)
  - a++
    operator++ (a, 0)

知识讲解

典型双目操作符的重载


典型双目操作符的重载
运算类双目操作符
操作数与表达式
成员函数
全局函数
友元
friend
友元特权
友元声明
友元非成员
友元操作符
赋值类双目操作符
操作数与表达式
成员函数
全局函数

运算类双目操作符

操作数与表达式

- 左右操作数均可为左值或右值
  - Complex lv (1, 2);
    Complex const rv (3, 4);
    lv + rv;
    rv + lv;
- 表达式的值必须是右值
  - Complex a (1, 2), b (3, 4), c (5, 6);
    (a + b) = c; // 错误

知识讲解

成员函数

- 常函数以支持右值型左操作数，常参数以支持右值型右操作数，常返回值以支持右值型表达式的值
  - class Complex {
    public:
      Complex const operator+ (
        Complex const& rhs) const {
        return Complex (m_r + rhs.m_r, m_i + rhs.m_i);
      }
    };

知识讲解

全局函数

- 常第一参数以支持右值型左操作数，常第二参数以支持右值型右操作数，常返回值以支持右值型表达式的值
  - Complex const operator+ (Complex const& lhs,
    Complex const& rhs) {
    return Complex (lhs.m_r + rhs.m_r,
      lhs.m_i + rhs.m_i);
  }
- 为了在一个全局操作符函数中直接访问其操作数类型的私有及保护成员，同时又不破坏其操作数类型的封装性，可以将该操作符函数声明为其操作数类型的友元
  - friend Complex const operator+ (Complex const&,
    Complex const&);

Technology Tarena 达内科技

友元

friend

- 可以通过friend关键字，把一个全局函数、另一个类的成员函数或者另一个类整体，声明为某个类的友元
  - class A {
    ```
    friend void foo (void);
    friend void B::bar (void) const;
    friend class C;
    };
    ```
  - 其中A称为全局函数foo、成员函数B::bar和类C的授权类
  - 全局函数foo、成员函数B::bar和类C称为A的友元函数和友元类

知识讲解

友元特权

- 友元拥有访问授权类任何非公有成员的特权

知识讲解

访问控制限定符	访问控制属性	内部	子类	外部	友元
public	公有成员	OK	OK	OK	OK
protected	保护成员	OK	OK	NO	OK
private	私有成员	OK	NO	NO	OK

友元声明

- 友元声明可以出现在授权类的公有、私有或者保护等任何区域，且不受访问控制限定符的约束
  - class A {
    friend void foo (A& a);
    public:
    friend void bar (A const* a);
    protected:
    friend void hum (void);
    private:
    int m_pri; };
  - void foo (A& a) { a.m_pri++; }
    void bar (A const* a) { cout << a->m_pri << endl; }
    void hum (void) { A a; a.m_pri = 0; }

知识讲解

友元非成员

- 友元不是成员，即便将其定义在授权类的内部，其作用域也不属于授权类，当然也不拥有授权类的this指针
  - class Integer {
    public:
    void set (int n) { m_n = n; } // 成员
    friend void set (Integer& i, int n) { i.m_n = n; } // 友元
    int get (void) const { return m_n; } // 成员
    friend int get (Integer const& i) { return i.m_n; } // 友元
    private:
    int m_n;
    };
- 因为不隶属于同一个作用域，友元和成员也不可能构成重载

友元操作符

- 双目操作符函数常被声明为其左操作数的成员，同时也是其右操作数的友元，这样它就可以毫无障碍地访问左右两个操作数的非公有成员，而不对封装构成太大影响
  - class Mail {
    friend void Mbox::operator<< (Mail const&) const;
    private:
    string m_title, m_content; };
  - class Mbox {
    void operator<< (Mail const& mail) const {
    … mail.m_title … mail.m_content … } };
  - Mbox mbox ("minwei@tarena.com.cn");
    Mail mail ("通知", "今天18:00召开教学例会");
    mbox << mail;

知识讲解

支持+/-操作符的复数类

【参见：TTS COOKBOOK 】

课堂练习

- 支持+/-操作符的复数类

赋值类双目操作符

操作数与表达式

- 右操作数可为左值或右值，但左操作数必须是左值
  - Complex lv (1, 2);
    Complex const rv (3, 4);
    lv += rv;
    rv += lv; // 错误
- 表达式的值为左值，且为左操作数本身(而非副本)
  - Complex a (1, 2), b (3, 4), c (5, 6);
    (a += b) = c; // c->a

知识讲解

成员函数

- 非常函数以支持左值型左操作数，常参数以支持右值型右操作数，非常返回值以支持左值型表达式的值，返回自引用即左操作数本身
  - class Complex {
    public:
      Complex& operator+= (Complex const& rhs) {
        m_r += rhs.m_r;
        m_i += rhs.m_i;
        return *this;
      }
    };

知识讲解

全局函数

- 非常第一参数以支持左值型左操作数，常第二参数以支持右值型右操作数，非常返回值以支持左值型表达式的值，返回引用型第一参数即左操作数本身
  - Complex& operator+= (Complex& lhs,
    Complex const& rhs) {
    lhs.m_r += rhs.m_r;
    lhs.m_i += rhs.m_i;
    return lhs;
    }

知识讲解

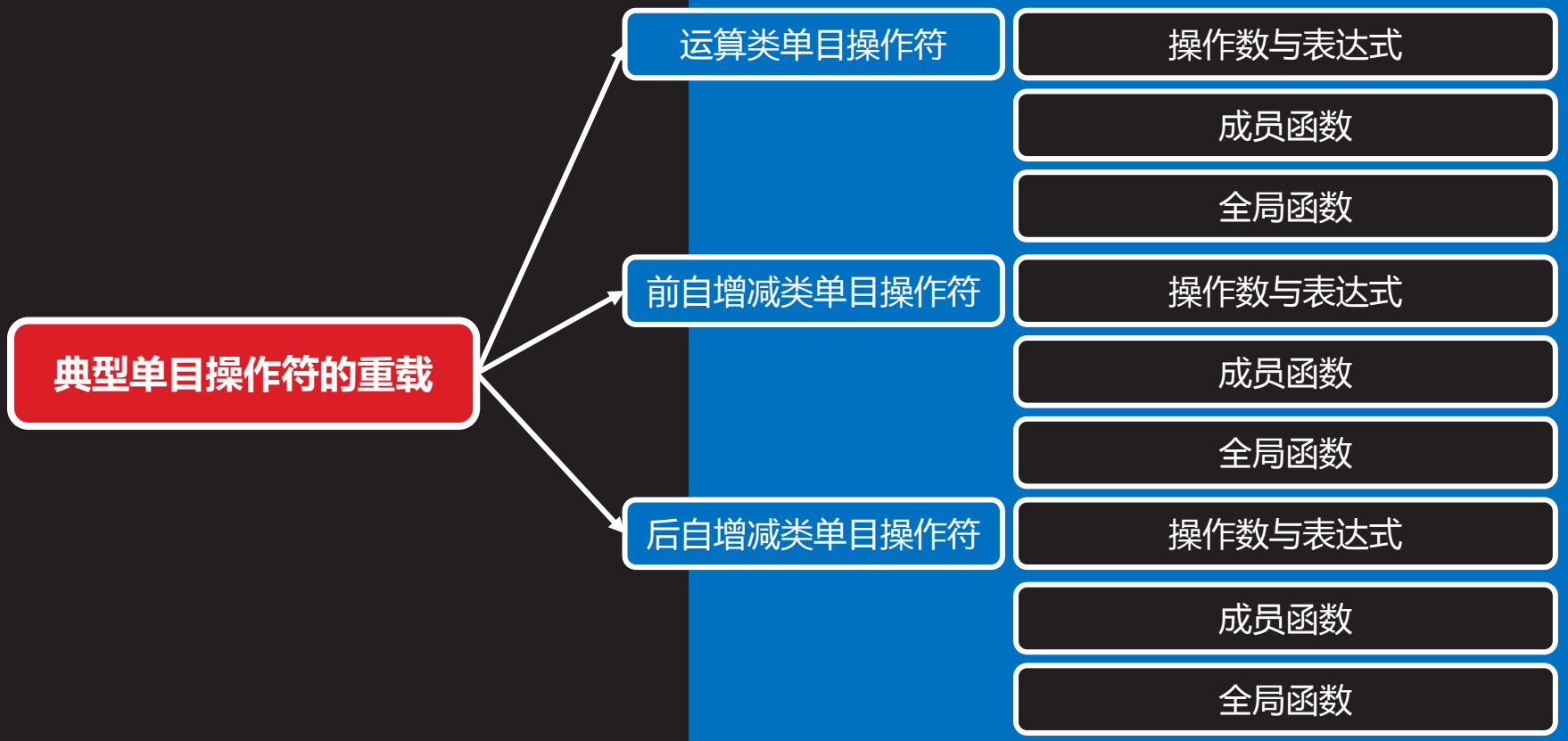
支持+=/-=操作符的复数类

【参见：TTS COOKBOOK 】

课堂练习

- 支持+=/-=操作符的复数类

典型单目操作符的重载


典型单目操作符的重载
运算类单目操作符
操作数与表达式
成员函数
全局函数
前自增减类单目操作符
操作数与表达式
成员函数
全局函数
后自增减类单目操作符
操作数与表达式
成员函数
全局函数

Technology Tarena 达内科技

运算类单目操作符

操作数与表达式

- 操作数为左值或右值
  - Complex lv (1, 2);
    Complex const rv (3, 4);
    -lv;
    ~rv;
- 表达式的值必须是右值
  - Complex a (1, 2), b (3, 4);
    -a = b; // 错误

知识讲解

成员函数

- 常函数以支持右值型操作数，常返回值以支持右值型表达式的值
  - class Complex {
    public:
        Complex const operator- (void) const {
            return Complex (-m_r, -m_i);
        }
    };

知识讲解

全局函数

- 常参数以支持右值型操作数，常返回值以支持右值型表达式的值
  - Complex const operator- (Complex const& opd) {
    return Complex (-opd.m_r, -opd.m_i);
    }

知识讲解

Technology Tarena 达内科技

支持-/~操作符的复数类

【参见：TTS COOKBOOK 】

课堂练习

- 支持-/~操作符的复数类

前自增减类单目操作符

操作数与表达式

- 操作数为左值
  - Complex lv (1, 2);
    Complex const rv (3, 4);
    ++lv;
    --rv; // 错误
- 表达式的值为左值，且为操作数本身(而非副本)
  - Complex a (1, 2), b (3, 4);
    ++++a; // ++(++a)
    ++a = b; // b->a

知识讲解

成员函数

- 非常函数以支持左值型操作数，非常返回值以支持左值型表达式的值，返回自引用即操作数本身
  - class Complex {
    public:
      Complex& operator++ (void) {
        ++m_r;
        ++m_i;
        return *this;
      }
    };

知识讲解

全局函数

- 非常参数以支持左值型操作数，非常返回值以支持左值型表达式的值，返回引用型参数即操作数本身
  - Complex& operator++ (Complex& opd) {
    ++opd.m_r;
    ++opd.m_i;
    return opd;
    }

知识讲解

支持前++/--操作符的复数类

【参见：TTS COOKBOOK 】

课堂练习

- 支持前++/--操作符的复数类

Technology Tarena 达内科技

后自增减类单目操作符

操作数与表达式

- 操作数为左值
  - Complex lv (1, 2);
    Complex const rv (3, 4);
    lv++;
    rv--; // 错误
- 表达式的值为右值，且为操作数在运算前的副本
  - Complex a (1, 2), b (3, 4);
    a++++; // 错误
    a++ = b; // 错误

知识讲解

成员函数

- 非常函数以支持左值型操作数，常返回值以支持右值型表达式的值，整型哑元参数以区别于前自增减
  - class Complex {
    public:
      Complex const operator++ (int) {
        Complex old = *this;
        ++m_r; ++m_i; return old;
      }
    };
- 对于后自增减表达式，编译器在调用操作符函数时会多传一个整型参数，通过重载解析匹配到后缀操作符函数
  - a++; // a.operator++ (0);
  - ++a; // a.operator++ ();

知识讲解

全局函数

- 非常参数以支持左值型操作数，常返回值以支持右值型表达式的值，整型哑元参数以区别于前自增减
  - Complex const operator++ (Complex& opd, int) {
    Complex old = opd;
    ++opd.m_r;
    ++opd.m_i;
    return old;
    }
- 对于后自增减表达式，编译器在调用操作符函数时会多传一个整型参数，通过重载解析匹配到后缀操作符函数
  - a++; // ::operator++ (a, 0);
  - ++a; // ::operator++ (a);

知识讲解

支持后++/--操作符的复数类

【参见：TTS COOKBOOK 】

课堂练习

- 支持后++/--操作符的复数类

输入输出操作符的重载

输出操作符

操作数与表达式

- 左操作数为左值形式的输出流(ostream)对象，右操作数为左值或右值
  - Complex lv (1, 2);
    Complex const rv (3, 4);
    cout << lv;
    cout << rv;
- 表达式的值为左值，且为左操作数本身(而非副本)
  - Complex complex (1, 2);
    cout << complex << endl;
  - cout.operator<< (comlex).operator<< (endl);
    ::operator<< (::operator<< (cout, complex), endl);

全局函数

- 左操作数的类型为ostream，若以成员函数形式重载该操作符，就应将其定义为ostream类的成员，该类为标准库提供，无法添加新的成员，因此只能以全局函数形式重载该操作符
- 非常第一参数以支持左值型左操作数，常第二参数以支持右值型右操作数，非常返回值以支持左值型表达式的值，返回引用型第一参数即左操作数本身
  - ostream& operator<< (ostream& lhs,
    Complex const& rhs) {
    return lhs << rhs.m_r << '+' << rhs.m_i << 'i';
    }

知识讲解

支持<<操作符的复数类

【参见：TTS COOKBOOK 】

课堂练习

- 支持<<操作符的复数类

Technology Tarena 达内科技

输入操作符

操作数与表达式

- 左操作数为左值形式的输入流(istream)对象，右操作数为左值
  - Complex lv (1, 2);
    Complex const rv (3, 4);
    cin >> lv;
    cin >> rv; // 错误
- 表达式的值为左值，且为左操作数本身(而非副本)
  - Complex c1, c2;
    cin >> c1 >> c2;
  - cin.operator>> (c1).operator>> (c2);
    ::operator>> (::operator>> (cin, c1), c2);

知识讲解

全局函数

- 左操作数的类型为istream，若以成员函数形式重载该操作符，就应将其定义为istream类的成员，该类为标准库提供，无法添加新的成员，因此只能以全局函数形式重载该操作符
- 非常第一参数以支持左值型左操作数，非常第二参数以支持左值型右操作数，非常返回值以支持左值型表达式的值，返回引用型第一参数即左操作数本身
  - istream& operator>> (istream& lhs, Complex& rhs) {
    return lhs >> rhs.m_r >> rhs.m_i;
    }

知识讲解

支持>>操作符的复数类

【参见：TTS COOKBOOK】

课堂练习

- 支持>>操作符的复数类

Technology Tarena 达内科技

总结和答疑