1、运算符重载的概念与限制

运算符重载使系统内置的运算符可以用于 class 类型的对象，但不能创建新的运算符。
不能改变运算符的优先级和结合性，也不能改变运算符的操作数个数。
不是所有的运算符都能重载。

运算符重载的本质是写一个函数来说明某个运算符如何处理某个 class 类型的对象。
运算符重载函数的函数名必须是 operator@，其中@代表要重载的运算符。
大多数运算符可以重载为全局函数或 class 类型的成员函数。
如果重载为全局函数且需要访问 class 类型的私有成员，那么应该声明为 class 类型的友元。
赋值（=）、下标（[]）、函数调用（（））和成员访问（->）只能重载为成员函数。

2、运算符重载函数的参数与返回类型

运算符重载为全局函数时，形式参数的个数等于运算符的操作数个数。
运算符重载为成员函数时，如果算上隐含的 this 指针，则形式参数的个数也等于运算符的操作数个数。对于二元运算符，this 指针将指向左操作数（即该成员函数的当前对象）。

大多数由运算符构建的表达式都需要一个运算结果，因此运算符重载函数通常应有返回值。
如果该表达式的运算结果应为当前对象本身，则该运算符重载函数应返回当前对象的引用。

3、具有赋值性质的运算符重载函数

如果类的创建者没有定义赋值运算符重载函数，那么系统为其生成一个缺省的赋值运算符重载函数，在对应的数据成员间赋值。
【每个类必有的四个成员函数：构造函数、拷贝构造函数、析构函数、赋值运算符重载函数】
当类的成员中包含指针且不同对象的该指针不应指向相同目标时，类的创建者应该自定义赋值运算符重载函数（也应该自定义拷贝构造函数）。
在赋值运算符重载函数的函数体中，往往首先检查是否为当前对象给自己赋值（对更广泛的成员函数而言，如果当前对象和参数对象是同一个，要注意避免对二者的操作互相干扰）。

C++规定“++”和“--”作为前缀时是一元操作符，作为后缀时是二元操作符。
“++”和“--”作为后缀时，第二个操作数是整型数，编译器自动将其赋值为 0。

赋值运算符、复合赋值运算符（如+=）、自增自减运算符（前缀）最好都重载为成员函数，且最好返回当前对象的引用。

4、输入输出运算符重载函数

输入输出运算符必须被重载成全局函数，因为左操作数是输入/输出流对象（即 cin/cout），无法调用自定义的 class 类的成员函数。
输入/输出运算符重载函数的第一个参数是 istream/ostream 类的非常量引用。
通常应将输入输出运算符重载函数声明为 class 类的友元，以便输入/输出其私有成员。
注意，输入运算符重载函数的第二个参数不能加 const 限定。

5、自定义类型转换运算符

从内置类型转换为 class 类型，可利用 class 类型的构造函数。
如有 Rational 类的对象 r，执行 r=2 会先以 2 为参数调用 Rational 的构造函数，创建 Rational 类的临时对象，再用该临时对象给 r 赋值。
同理，如果将“+”重载为 Rational 类的友元函数，那么 2+r 会先以 2 为参数调用 Rational 的构造函数，创建 Rational 类的临时对象，再将该临时对象与 r 相加（如果将“+”重载为 Rational 类的成员函数，则无法实现这个操作）。

从 class 类型转换为其它类型（以下称为目标类型），需定义类型转换运算符重载函数。
其原型为 operator 目标类型名 ( ) const，没有参数，也没有返回类型。
但该函数有返回语句，返回一个目标类型的对象。