

# C++: Practical session 9

## 1 Rational class

Design and write a `Rational` class that implements exact operations on the rationals. It should deal correctly with addition, subtraction, multiplication, and division. The operate-and-assign operators should also be implemented.

Inside the class, you should use integer arithmetic only for the numerator and denominator, except when returning the (approximate) floating-point `value()`.

For example, the following code might be a valid use of the class:

```
Rational a(10,3);
std::cout << "a = " << a.num() << "/" << a.denom() << " ~ " <<
    a.value() << std::endl;
a /= 5;
Rational b(3,4);
a += b*3;
std::cout << "a = " << a.num() << "/" << a.denom() << " ~ " <<
    a.value() << std::endl;
```

and output

```
a = 10/3 ~ 3.3333333
a = 35/12 ~ 2.9166666
```

### 1.1 Extensions

1. Implement an output operator. This function should have the signature

```
std::ostream& operator<<(std::ostream& os, const Rational& r);
```

2. Implement an input operator that allows a user to input a rational number in the form 10/3, i.e. you can write:

```
Rational r;
std::cin >> r;
```

and if the user entered “10/3” at the terminal, `r` would take the value  $\frac{10}{3}$ .

3. Ensure that all fractions are cancelled to their simplest form after every operation.
4. Detect and flag any integer overflow.

## 2 Matrix class

You should now be able to design and write your own simple class to store and calculate with square matrices. The matrices should have constant size, defined at compile-time.

### 3 Extensions

1. Instead of allowing only double precision entries, template your class to allow any type.
2. In the above, what happens for an integer matrix, especially for inversion? How might you correct for this?