

- Solution

In this lesson, we'll discuss the solution to the exercise.

WE'LL COVER THE FOLLOWING ^

- Solution
- Explanation

Solution

```
#include <iostream>

class Fraction{
public:
    Fraction(int num = 0, int denom = 0):numerator(num), denominator(denom){}

    friend std::istream& operator>> (std::istream& in, Fraction &frac);
    friend std::ostream& operator<< (std::ostream& out, const Fraction& frac);

private:
    int numerator;
    int denominator;
};

std::istream& operator>> (std::istream& in, Fraction& frac){

    in >> frac.numerator;
    in >> frac.denominator;

    return in;
}

std::ostream& operator<< (std::ostream& out, const Fraction& frac){

    out << frac.numerator << "/" << frac.denominator;
    return out;
}

int main(){

    std::cout << std::endl;

    Fraction frac(3, 4);
    Fraction frac2(7, 8);
```

```

std::cout << "frac(3, 4): " << frac << std::endl;
std::cout << "frac(7, 8): " << frac << std::endl;
std::cout << frac << " " << frac2 << std::endl;

std::cout << std::endl;

std::cout << "Enter two natural numbers for a Fraction: " << std::endl;
Fraction fracDef;
std::cin >> fracDef;
std::cout << "fracDef: " << fracDef << std::endl;

std::cout << std::endl;
}

```



Explanation

In the code above, we have created a `Fraction` class which contains two private variables, i.e., `numerator` and `denominator`. We have defined overloaded input and output operators for the class objects in `Fraction`. In `main`, we have created objects for the `Fraction` class and called them by using the `<<` output operator and used the `>>` input operator to take inputs.

In the next lesson, we'll learn some tools that can help us format our data.