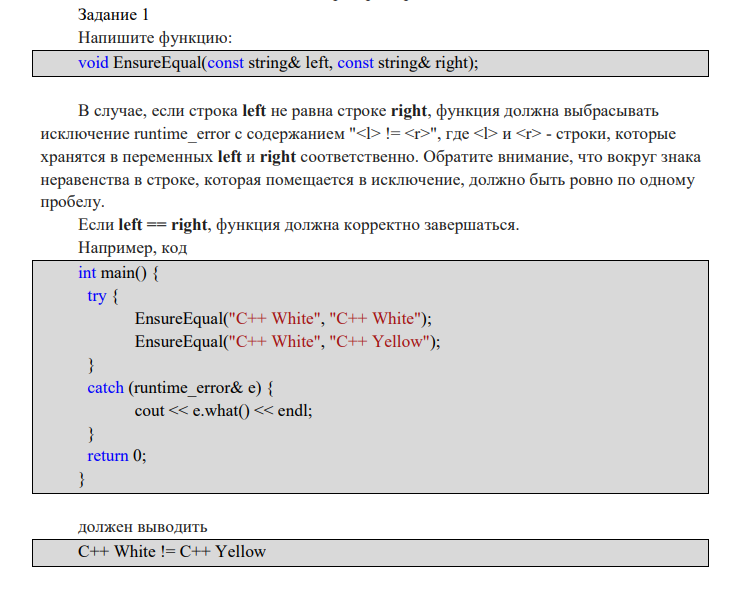
Лабораторная работа №8

Авдеев Владислав ИКТ-31



Код:

#include <iostream>

#include <vector>

#include <string>

#include <sstream>

using namespace std;

void EnsureEqual(const string&,const string&);

int main(){

    try {

        EnsureEqual("C++ White", "C++ White");

        EnsureEqual("C++ White", "C++ Yellow");

    }

    catch (runtime\_error& e) {

        cout << e.what() << endl;

    }

    return 0;

}

void EnsureEqual(const string& left,const string& right){

    std:: stringstream ss;

    if (left!=right){

        ss<<left<<" != "<<right;

        throw runtime\_error(ss.str());

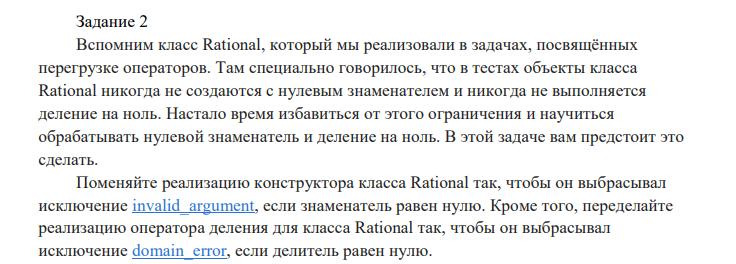
    }

    else{

        cout<<"Vse rabotaet"<<endl; // Чтобы хоть как-то отобразить корректность работы кода без ошибок

    }

}



Код:

#include <iostream>

#include <vector>

#include <string>

#include <stdexcept>

using namespace std;

// В общем я так и не нашёл реализацию класса Rational, поэтому напишу свою

class Rational{

    public:

  explicit  Rational(const int& num=0,const int& denom=1){

      if(denom==0){

        throw std:: invalid\_argument("Denominator must not be 0.");

      }

    Numerator=num;

    Denominator=denom;

  };

    friend Rational operator/(const Rational& lhs, const Rational& rhs);

    private:

    int Numerator;

    int Denominator;

};

    Rational operator/(const Rational& lhs, const Rational& rhs){

        if((lhs.Denominator\*rhs.Numerator)==0){

            throw std::domain\_error("Nelzya delit na nol");

        }

        return Rational(lhs.Numerator\*rhs.Denominator,lhs.Denominator\*rhs.Numerator);

    }

int main(){

    try {

        Rational r(1, 0);

        cout << "Doesn't throw in case of zero denominator" << endl;

        return 1;

    }

    catch (invalid\_argument& e) {

        cout<<e.what()<<endl;

    }

    try {

        auto x = Rational(1, 2) / Rational(0, 1);

        cout << "Doesn't throw in case of division by zero" << endl;

        return 2;

    }

    catch (domain\_error& e) {

        cout<<e.what()<<endl;

    }

    cout << "OK" << endl;

    return 0;

}