МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ
УНИВЕРСИТЕТ им. В. Г. ШУХОВА»**
(БГТУ им. В.Г. Шухова)

ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И УПРАВЛЯЮЩИХ СИСТЕМ

Лабораторная работа №7

по дисциплине: ООП
тема: «Исключительные ситуации в C++»

Выполнил: ст. группы ПВ-223
Пахомов Владислав Андреевич

Проверили:
пр. Черников Сергей Викторович

Белгород 2024 г.

Лабораторная работа №7
«Исключительные ситуации в C++»
Вариант 10

Цель работы: получение теоретических знаний об исключительных ситуациях в C++. Получение практических навыков при работе с исключениями в C++.

Разработать программу в соответствии с заданным вариантом задания.

Разработать класс “Калькулятор”. Предусмотреть исключительные ситуации, деления на ноль, переполнение и др.

main.cpp

```
#include "../libs/alg/calculator/calculator.h"

#include <iostream>

int main() {
    std::cout << (Calculator::get() << (Calculator::get() << 4 << Calculator::CalculatorOperation::DIVIDE << 2 <<
    ↪  Calculator::CalculatorOperationEnd::END)
                                    << Calculator::CalculatorOperation::PLUS << 7 <<
                                    ↪  Calculator::CalculatorOperationEnd::END) << std::endl;
    std::cout << (Calculator::get() << 3 << Calculator::CalculatorOperation::MULTIPLY << 1.3 <<
    ↪  Calculator::CalculatorOperationEnd::END) << std::endl;
    std::cout << (Calculator::get() << 1 << Calculator::CalculatorOperation::PLUS << 0.9 <<
    ↪  Calculator::CalculatorOperationEnd::END) << std::endl;
    std::cout << (Calculator::get() << 4 << Calculator::CalculatorOperation::MINUS << 2 <<
    ↪  Calculator::CalculatorOperationEnd::END) << std::endl;
}
```

calculator.h

```
#ifndef CALCULATOR_H
#define CALCULATOR_H

#include <string>

class Calculator {
public:
    enum CalculatorOperation {
        PLUS,
        MINUS,
        DIVIDE,
        MULTIPLY
    };

    enum CalculatorOperationEnd {
        END
    };

    bool op_left_assigned = false;
    double op_left;
    bool op_assigned = false;
```

```
    CalculatorOperation op;
    bool op_right_assigned = false;
    double op_right;

    static Calculator get();
    Calculator& operator<<(double val);
    Calculator& operator<<(CalculatorOperation op);
    double operator<<(CalculatorOperationEnd op);
private:
    Calculator() {

    };
};

#endif // CALCULATOR_H
```

calculator.cpp

```
#include <stdexcept>
#include <limits>

#include "calculator.h"

Calculator Calculator::get() {
    return Calculator();
}

Calculator& Calculator::operator<<(double val) {
    if (!this->op_left_assigned && !this->op_right_assigned && !this->op_assigned) {
        this->op_left = val;
        this->op_left_assigned = true;
    } else if (!this->op_right_assigned && this->op_left_assigned && this->op_assigned) {
        this->op_right = val;
        this->op_right_assigned = true;
    } else {
        throw std::invalid_argument("Invalid operator");
    }
}

Calculator& Calculator::operator<<(CalculatorOperation op) {
    if (this->op_left_assigned && !this->op_assigned && !this->op_right_assigned) {
        this->op = op;
        this->op_assigned = true;
    } else {
        throw std::invalid_argument("Invalid operator");
    }
}

double Calculator::operator<<(CalculatorOperationEnd op) {
    if (op == CalculatorOperationEnd::END) {
        if (!(this->op_assigned && this->op_left_assigned && this->op_right_assigned))
            throw std::invalid_argument("Expression incomplete");
```

```
        double res;

        switch (this->op) {
        case CalculatorOperation::PLUS:
            res = this->op_left + this->op_right;
            break;
        case CalculatorOperation::MINUS:
            res = this->op_left - this->op_right;
            break;
        case CalculatorOperation::MULTIPLY:
            res = this->op_left * this->op_right;
            break;
        case CalculatorOperation::DIVIDE:
            if (this->op_right == 0) throw std::domain_error("Can't divide by zero");

            res = this->op_left / this->op_right;
            break;
        default:
            throw std::invalid_argument("Unsupported operation");
        }

        if (res == std::numeric_limits<double>::infinity() || res == -std::numeric_limits<double>::infinity())
            throw std::overflow_error("Number is too big");

        return res;
    } else {
        throw std::invalid_argument("Unsupported operation");
    }
}
```

Ссылка на репозиторий

Вывод: в ходе лабораторной работы получили теоретические знания об исключительных ситуациях в C++. Получение практических навыков при работе с исключениями в C++.