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 №10

по дисциплине: ООП
тема: «Закрепление навыков программирования в объектно-ориентированном стиле. Визуальные компоненты. Знакомство с QT.»

Выполнил: ст. группы ПВ-223
Пахомов Владислав Андреевич

Проверили:
пр. Черников Сергей Викторович

Белгород 2024 г.

Лабораторная работа №10

«Закрепление навыков программирования в объектно-ориентированном стиле. Визуальные компоненты. Знакомство с QT.»

Вариант 10

Цель работы: приобретение практических навыков создания приложений на языке C++.

Создать репозиторий под контролем git на открытой площадке gitlab.com Задать имя репозитория <год>_<группа>_<имя студента в транслите>_<номер варианта по журналу>. Выполнить проектирование задачи в соответствии с вариантом (табл. 1). Для реализации поставленной задачи необходимо с проектировать, реализовать и использовать шаблон «умные указатели». Соответственно это учесть при проектировании программного обеспечения. Выполнить реализацию в соответствии с вариантом задачи (табл. 1), использую среду разработки QT. Реализация должна быть кроссплатформенной и выполнена на основе графических окон. *main.cpp*

```
#include "mainwindow.h"
#include "smartptr.h"

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    smart_ptr<QMainWindow> w(new QMainWindow());
    Ui_MainWindow b;
    b.setupUi(w.operator->());

    w->show();

    return a.exec();
}
```

mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QVariant>
#include <QAction>
#include <QApplication>
#include <QButtonGroup>
#include <QHeaderView>
#include <QMainWindow>
#include <QMenuBar>
#include <QPushButton>
#include <QStatusBar>
#include <QTextEdit>
#include <QToolBar>
#include <QWidget>
#include <QGridLayout>

#include <queue>
```

```
#include "../libs/alg/mathparser/mathparser.h"
#include "smartptr.h"

QT_BEGIN_NAMESPACE
class Ui_MainWindow {
    const static int screenHeight = 400;
    const static int screenWidth = 300;

    smart_ptr<QWidget> centralWidget = new QWidget();
    smart_ptr<QGridLayout> mainLayout = new QGridLayout(centralWidget);

    smart_ptr<QTextEdit> field = new QTextEdit;

    smart_ptr<QPushButton> plusButton = new QPushButton("+");
    smart_ptr<QPushButton> multButton = new QPushButton("*");
    smart_ptr<QPushButton> divButton = new QPushButton("/");
    smart_ptr<QPushButton> powButton = new QPushButton("^");

    smart_ptr<QPushButton> tildaButton = new QPushButton("~");
    smart_ptr<QPushButton> lParButton = new QPushButton("(");
    smart_ptr<QPushButton> rParButton = new QPushButton(")");

    smart_ptr<QPushButton> oneButton = new QPushButton("1");
    smart_ptr<QPushButton> twoButton = new QPushButton("2");
    smart_ptr<QPushButton> threeButton = new QPushButton("3");
    smart_ptr<QPushButton> sinButton = new QPushButton("sin");


    smart_ptr<QPushButton> fourButton = new QPushButton("4");
    smart_ptr<QPushButton> fiveButton = new QPushButton("5");
    smart_ptr<QPushButton> sixButton = new QPushButton("6");
    smart_ptr<QPushButton> cosButton = new QPushButton("cos");

    smart_ptr<QPushButton> sevenButton = new QPushButton("7");
    smart_ptr<QPushButton> eightButton = new QPushButton("8");
    smart_ptr<QPushButton> nineButton = new QPushButton("9");
    smart_ptr<QPushButton> minusButton = new QPushButton("-");

    smart_ptr<QPushButton> zeroButton = new QPushButton("0");
    smart_ptr<QPushButton> dotButton = new QPushButton(".");
    smart_ptr<QPushButton> eqButton = new QPushButton("=");
    smart_ptr<QPushButton> eraseButton = new QPushButton("<-");

    std::deque<MathParser::Token> tokens;
public:
    void setupUi(QMainWindow *mainWindow) {
        mainWindow->setWindowTitle("Инженерный калькулятор");

        centralWidget->setMinimumHeight(screenHeight);
        centralWidget->setFixedHeight(screenHeight);
        centralWidget->setMaximumHeight(screenHeight);

        centralWidget->setMinimumWidth(screenWidth);
        centralWidget->setFixedWidth(screenWidth);
```

```
    centralWidget->setMaximumWidth(screenWidth);

    mainWindow->setMinimumHeight(screenHeight);
    mainWindow->setFixedHeight(screenHeight);
    mainWindow->setMaximumHeight(screenHeight);

    mainWindow->setMinimumWidth(screenWidth);
    mainWindow->setFixedWidth(screenWidth);
    mainWindow->setMaximumWidth(screenWidth);

    field->setMaximumHeight(50);
    field->setReadOnly(true);

    mainLayout->addWidget(field, 0, 0, 1, 4);

    mainWindow->setCentralWidget(centralWidget);

    mainLayout->addWidget(plusButton, 1, 0);
    mainLayout->addWidget(multButton, 1, 1);
    mainLayout->addWidget(divButton, 1, 2);
    mainLayout->addWidget(powButton, 1, 3);

    mainLayout->addWidget(lParButton, 2, 0);
    mainLayout->addWidget(rParButton, 2, 1);
    mainLayout->addWidget(tildaButton, 2, 2);
    mainLayout->addWidget(eraseButton, 2, 3);

    mainLayout->addWidget(oneButton, 3, 0);
    mainLayout->addWidget(twoButton, 3, 1);
    mainLayout->addWidget(threeButton, 3, 2);
    mainLayout->addWidget(sinButton, 3, 3);

    mainLayout->addWidget(fourButton, 4, 0);
    mainLayout->addWidget(fiveButton, 4, 1);
    mainLayout->addWidget(sixButton, 4, 2);
    mainLayout->addWidget(cosButton, 4, 3);

    mainLayout->addWidget(sevenButton, 5, 0);
    mainLayout->addWidget(eightButton, 5, 1);
    mainLayout->addWidget(nineButton, 5, 2);
    mainLayout->addWidget(minusButton, 5, 3);

    mainLayout->addWidget(zeroButton, 6, 0);
    mainLayout->addWidget(dotButton, 6, 1);
    mainLayout->addWidget(eqButton, 6, 2, 1, 2);

    std::vector<QPushButton*> numButtons = {oneButton, twoButton, threeButton, fourButton, fiveButton, sixButton,
    ↪  sevenButton, eightButton, nineButton, zeroButton, dotButton};

    for (auto numButton : numButtons) {
        QObject::connect(numButton, &QPushButton::clicked, [this, numButton]() {
            if (!this->tokens.empty() && this->tokens.back().isNumber()) {
                auto token = this->tokens.back();
                this->tokens.pop_back();
```

```
                this->tokens.push_back(MathParser::Token(token.val + numButton->text().toStdString()));
            } else {
                this->tokens.push_back(MathParser::Token(numButton->text().toStdString()));
            }

            this->onTokensUpdated();
        });
    }

    std::vector<QPushButton*> opButtons = {plusButton, minusButton, multButton, divButton, powButton, sinButton,
    ↪  cosButton, lParButton, rParButton, tildaButton};

    for (auto opButton : opButtons) {
        QObject::connect(opButton, &QPushButton::clicked, [this, opButton]() {
            auto searchToken = std::find_if(MathParser::tokens.begin(), MathParser::tokens.end(),
                                            [opButton](MathParser::Token v){return v.val ==
                                            ↪  opButton->text().toStdString();});
            this->tokens.push_back(*searchToken);

            this->onTokensUpdated();
        });
    }

    QObject::connect(eraseButton, &QPushButton::clicked, [this]() {
        if (!this->tokens.empty() && this->tokens.back().isNumber() && this->tokens.back().val.size() > 1) {
            auto token = this->tokens.back();
            this->tokens.pop_back();

            this->tokens.push_back(MathParser::Token(token.val.substr(0, token.val.size() - 1)));
        } else if (tokens.empty()) {
            field->setText("");
        } else {
            this->tokens.pop_back();
        }

        this->onTokensUpdated();
    });

    QObject::connect(eqButton, &QPushButton::clicked, [this]() {
        try {
            std::queue<MathParser::Token> exprQueue(this->tokens);
            exprQueue = MathParser::infixToPostfix(exprQueue);
            auto res = MathParser::evaluate(exprQueue);
            field->setText(QString::fromStdString(res.val));
        } catch(...) {
            field->setText("Syntax error");
        }

        this->tokens.clear();
    });
}

void onTokensUpdated() {
```

```
        std::queue<MathParser::Token> exprQueue(this->tokens);
        std::string v = printSequence(exprQueue);
        field->setText(QString::fromStdString(v));
    }
};
namespace Ui{
class MainWindow:public Ui_MainWindow{};
}//namespaceUi
QT_END_NAMESPACE


#endif // MAINWINDOW_H
```

Ссылка на репозиторий

Вывод: в ходе лабораторной работы приобрели практические навыки создания приложений на языке C++.