Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ **ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.01 Информатика и вычислительная техника**

О Т Ч Е Т

по лабораторной работе № 9

Название: Программирование с использованием Qt

Дисциплина: Объектно-ориентированное программирование

Студент ИУ6-22Б (Группа) _____ (Подпись, дата) С.В. Астахов (И.О. Фамилия)

Преподаватель _____ (Подпись, дата) _____ (И.О. Фамилия)

Москва, 2020

Задание

Лабораторная работа выполняется по методическим указаниям.

Исходный код

Части 1-4 см. метод указания

Часть 5:

(файл calc.cpp)

```
#include <QApplication>
#include "calcDialog.h"
int main(int argc, char *argv[])
{
 QApplication app(argc, argv);
 CalcDialog * dialog = new CalcDialog();

 dialog->show(); // отображаем окно
 return app.exec(); // запускаем цикл обработки сообщений
}
```

(файл calcDialog.h)

```
#ifndef _CALC_DIALOG_H_
#define _CALC_DIALOG_H_
#include <QDialog>
#include <QLineEdit>
#include <QSignalMapper>
/// Класс, реализующий калькулятор
class CalcDialog: public QDialog
{
 Q_OBJECT
public:
 CalcDialog( QWidget * parent = 0);
 virtual ~CalcDialog(){};
protected:
 QSignalMapper * m_pSignalMapper;
 QLineEdit * m_pLineEdit;
 double m_Val; ///< Значение, с которым будет выполнена операция
 int m_Op; ///< Код нажатой операции
 bool m_bPerf;///< Операция была выполнена. Надо очистить поле ввода
 void initNum();///< Инициализировать переменные, связанные с вычислениями
 double getNumEdit(); ///< Получить число из m_pLineEdit
 void setNumEdit( double ); ///< Отобразить число в m_pLineEdit
 /// Вычислить предыдущую операцию
 ///(в бинарных операциях был введен второй операнд)
```

void calcPrevOp(int curOp);

/// Проверить, была ли выполнена операция при нажатии на цифровую клавишу
/// Если операция выполнена, значит m_pLineEdit необходимо очистить
void checkOpPerf();
private slots:
/// Слот для обработки нажатий всех кнопок
void clicked(int id);
};
#endif

(файл calcDialog.cpp)

```
#include <QVector>
#include <QGridLayout>
#include <QPushButton>
#include <QHBoxLayout>
#include <QVBoxLayout>
#include <cmath>
#include "calcDialog.h"
// Идентификаторы кнопок
// Для цифровых кнопок идентификатор является соответствующая цифра
#define DIV 10
#define MUL 11
#define MINUS 12
#define PLUS 13
#define INVERSE 15
#define DOT 16
#define EQ 20
#define BKSP 30
#define CLR 31
#define CLR_ALL 32

#define LOGX 61
#define EXPY 62
#define SINX 63
#define COSX 64

// количество кнопок в группе, отображаемой в виде сетки
#define GRID_KEYS 16
/// Описатель кнопки
struct BtnDescr{
```

```
 QString text; ///< Отображаемый на кнопке текст
 int id; ///< Идентификатор кнопки
 BtnDescr() { id=0;}; ///< Конструктор по умолчанию
 ///< Конструктор для инициализации
 BtnDescr( const QString & str, int i)
 { text = str; id = i; };
};
/// Динамический массив-вектор элементов описателей кнопок
QVector<BtnDescr> _btnDescr;
/// Инициализация массива _btnDescr всеми отображаемыми кнопками
void InitBtnDescrArray()
{
	_btnDescr.push_back( BtnDescr("7", 7) );
	_btnDescr.push_back( BtnDescr("8", 8) );
	_btnDescr.push_back( BtnDescr("9", 9) );
	_btnDescr.push_back( BtnDescr("/", DIV) );
	_btnDescr.push_back( BtnDescr("4", 4) );
	_btnDescr.push_back( BtnDescr("5", 5) );
	_btnDescr.push_back( BtnDescr("6", 6) );
	_btnDescr.push_back( BtnDescr("*", MUL) );
	_btnDescr.push_back( BtnDescr("1", 1) );
	_btnDescr.push_back( BtnDescr("2", 2) );
	_btnDescr.push_back( BtnDescr("3", 3) );
	_btnDescr.push_back( BtnDescr("-", MINUS) );
	_btnDescr.push_back( BtnDescr("0", 0) );
	_btnDescr.push_back( BtnDescr("-/+", INVERSE) );
	_btnDescr.push_back( BtnDescr(".", DOT) );
	_btnDescr.push_back( BtnDescr("+", PLUS) );

	_btnDescr.push_back( BtnDescr("<-",BKSP) );
	_btnDescr.push_back( BtnDescr("CE",CLR) );
	_btnDescr.push_back( BtnDescr("C", CLR_ALL) );

	_btnDescr.push_back( BtnDescr("log[y]x", LOGX) );
	_btnDescr.push_back( BtnDescr("x^y", EXPY) );
	_btnDescr.push_back( BtnDescr("sin(x)", SINX) );
	_btnDescr.push_back( BtnDescr("cos(x)", SINX) );

	_btnDescr.push_back( BtnDescr("=", EQ) );
}
/// Конструктор класса калькулятора
CalcDialog::CalcDialog( QWidget * parent)
{
```

```
initNum(); // инициализируем счетные переменные
InitBtnDescrArray(); // инициализируем массив с описанием кнопок
// Создаем форму
m_pLineEdit = new QLineEdit(this);
// устанавливаем режим только чтения - разрешаем ввод только
// с нарисованных кнопок
m_pLineEdit->setReadOnly ( true );
m_pSignalMapper = new QSignalMapper(this);
// создаем схемы выравнивания
QVBoxLayout *extraLayout = new QVBoxLayout();
QGridLayout *gridLayout = new QGridLayout();
QHBoxLayout *bccKeysLayout = new QHBoxLayout();
QHBoxLayout *mainKeysLayout = new QHBoxLayout();
QVBoxLayout *dlgLayout = new QVBoxLayout();

// Заполняем форму кнопками из _btnDescr
for (int i = 0; i < _btnDescr.size(); i++) {
// Создаем кнопку с текстом из очередного описателя
        QPushButton *button = new QPushButton(_btnDescr[i].text);
// если кнопка в основном блоке цифровых или "=" -
// разрешаем изменение всех размеров
        if( i >= GRID_KEYS + 3 || i < GRID_KEYS)
                button->setSizePolicy ( QSizePolicy::Expanding,
                        QSizePolicy::Expanding);
// если кнопка не цифровая - увеличиваем шрифт надписи на 4 пункта
        if( _btnDescr[i].id >= 10 ){
                QFont fnt = button->font();
                fnt.setPointSize( fnt.pointSize () + 4 );
                button->setFont( fnt );
        }
// связываем сигнал нажатия кнопки с объектом m_pSignalMapper
        connect(button, SIGNAL(clicked()), m_pSignalMapper, SLOT(map()));
// обеспечиваем соответствие кнопки её идентификатору
        m_pSignalMapper->setMapping(button, _btnDescr[i].id);
if(i<GRID_KEYS) // Если кнопка из центрального блока - помещаем в сетку
        gridLayout->addWidget(button, i / 4, i % 4);
else if( i < GRID_KEYS + 3) // кнопка из верхнего блока - в bccKeysLayout
        bccKeysLayout->addWidget(button);
else if( i < GRID_KEYS + 3 + 4) // my buttons
        extraLayout->addWidget(button);
else
{
// кнопка "=" - помещаем в блок mainKeysLayout после gridLayout
mainKeysLayout->addLayout(extraLayout);
```

```
mainKeysLayout->addLayout(gridLayout);
mainKeysLayout->addWidget(button);
}
}
 // связываем сигнал из m_pSignalMapper о нажатии со слотом clicked
 // нашего класса
connect(m_pSignalMapper, SIGNAL(mapped(int)),
        this, SLOT(clicked(int)));
 // добавляем блоки кнопок в схему выравнивания всей формы
dlgLayout->addWidget(m_pLineEdit);
dlgLayout->addLayout(bccKeysLayout);
dlgLayout->addLayout(mainKeysLayout);
 // связываем схему выравнивания dlgLayout с формой
setLayout(dlgLayout);
// отображаем "0" в поле ввода чисел m_pLineEdit
setNumEdit( 0 );
};
// Обработка нажатия клавиш
void CalcDialog::clicked(int id)
{ // по идентификатору кнопки ищем действие для выполнения
        switch(id)
        {
case SINX:
{
  setNumEdit(sin(getNumEdit()));
  break;
};
case COSX:
{
  setNumEdit(cos(getNumEdit()));
  break;
};
case INVERSE: // унарная операция +/-
{
        setNumEdit( getNumEdit() * -1.0 ); break;
};
case DOT: // добавление десятичной точки
{
 // если на экране результат предыдущей операции - сбросить
 checkOpPerf();
 QString str = m_pLineEdit->text ();
 str.append( "." ); // добавляем точку к строке
 bool ok = false;
```

```
 // проверяем, является ли результат числом (исключаем 0.1. )
 str.toDouble(&ok);
 // если строка является числом - помещаем резульат в m_pLineEdit
 if( ok ) m_pLineEdit->setText ( str );
 break;
};
case DIV: // бинарные арифметические операции
case MUL:
case PLUS:
case MINUS:
case LOGX:
case EXPY:
case EQ:{
        calcPrevOp( id );
        break;
}
case CLR_ALL: initNum();// удалить всё
case CLR:{
 setNumEdit( 0 ); // записать в m_pLineEdit число 0
 break;
}
case BKSP:{ // удалить последний символ
 // если на экране резульатат предыдущей операции - сбросить
        checkOpPerf();
        QString str = m_pLineEdit->text ();
        if( str.length() ){
 // если строка в m_pLineEdit не нулевая - удалить символ
                str.remove( str.length()-1, 1 );
                m_pLineEdit->setText ( str );
        }
        break;
}
default:{ // обработка цифровых клавиш
 // если на экране результат предыдущей операции - сбросить
        checkOpPerf();
        QString sId;
 // сформировать строку по идентификатору нажатой клавиши
        sId.setNum( id );
        QString str = m_pLineEdit->text ();
        if( str == "0" )
 str = sId; // затираем незначащий нуль
else
 str.append( sId ); // добавить в m_pLineEdit нажатую цифру
```

```
m_pLineEdit->setText ( str );
}
};
};
// Получить число из m_pLineEdit
double CalcDialog::getNumEdit()
{
        double result;
        QString str = m_pLineEdit->text ();
 result = str.toDouble(); // преобразовать строку в число
 return result;
};
// записать число в m_pLineEdit
void CalcDialog::setNumEdit( double num )
{
        QString str;
 str.setNum ( num, 'g', 25 ); // преобразовать вещественное число в строку
 m_pLineEdit->setText ( str );
};
// Выполнить предыдущую бинарную операцию
void CalcDialog::calcPrevOp( int curOp )
{
 // получить число на экране
 // m_Val хранит число, введенное до нажатия кнопки операции
        double num = getNumEdit();
        switch( m_Op )
        {
                case DIV:{
                        if ( num != 0) m_Val /= num;
                        else m_Val = 0;
                        break;
                }
                case MUL:{
                        m_Val *= num;
                        break;
                }
                case PLUS:{
                        m_Val += num;
                        break;
                }
                case MINUS:{
                        m_Val -= num;
                        break;
```

```
                }
                case LOGX:{
                        m_Val = log(m_Val) / log(num);
                        break;
                }
                case EXPY:{
                        m_Val = pow(m_Val, num);
                        break;
                }
 case EQ: { // если была нажата кнопка "=" - не делать ничего
 m_Val = num;
 break; }
}
 m_Op = curOp; // запомнить результат текущей операции
 setNumEdit( m_Val ); // отобразить результат
 m_bPerf = true; // поставить флаг выполнения операции
};
void CalcDialog::checkOpPerf()
{
        if( m_bPerf ){
 // если что-то выполнялось - очистить m_pLineEdit
                m_pLineEdit->clear();
                m_bPerf = false;
        };
};
void CalcDialog::initNum()
{
        m_bPerf = false; m_Val = 0; m_Op = EQ;
};
```

Часть 6:

(файл front.cpp)

```
#include <QApplication>
#include "back.h"

int main(int argc, char *argv[])
{
   QApplication app(argc, argv);
   FormDialog *dialog = new FormDialog();
   dialog->show(); // отображаем окно
   return app.exec(); // запускаем цикл обработки сообщений
```

```
}
```

(файл back.h)

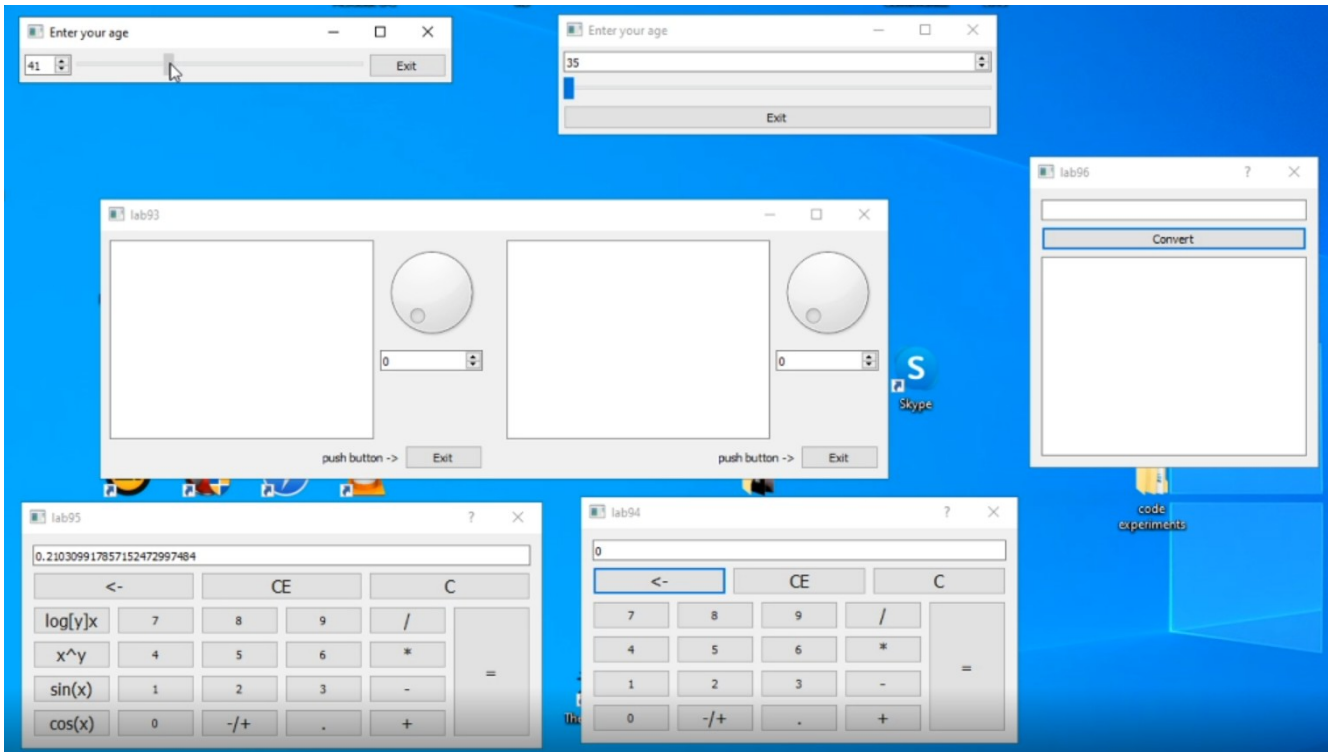
```
#ifndef BACK_H_
#define BACK_H_
#include <QDialog>
#include <QLineEdit>
#include <QSignalMapper>
#include <QTextEdit>
#include <QString>
/// Класс, реализующий редактор
class FormDialog: public QDialog
{
   Q_OBJECT
public:
   FormDialog( QWidget * parent = 0);
   virtual ~FormDialog(){};

protected:
   QLineEdit *lineEdit1;
   QTextEdit *field1;
   bool lower, isOut;


private slots:
   void swapper();
   void newQs();
};
#endif
```

(файл back.cpp)

```
#include <QPushButton>
#include <QVBoxLayout>
#include <QTextEdit>
#include <QLineEdit>
#include <iostream>
#include <QString>
#include "back.h"

//void FormDialog::newQs(bool& outId);
//void FormDialog::swapper(bool& caseId, bool& outId);
```

```
FormDialog::FormDialog(QWidget * parent){
	QVBoxLayout *mainLayout = new QVBoxLayout();
	lineEdit1 = new QLineEdit();
	QPushButton *button1 = new QPushButton("Convert");
	field1 = new QTextEdit();
	field1->setReadOnly(true);
	//QString str1;
	bool lower = true, isOut = false;


	connect(button1, SIGNAL(clicked()), this, SLOT(swapper()));
	connect(lineEdit1, SIGNAL(textEdited(QString)), this, SLOT(newQs()));
	mainLayout->addWidget(lineEdit1);
	mainLayout->addWidget(button1);
	mainLayout->addWidget(field1);
	setLayout(mainLayout);
};

void FormDialog::newQs(){
	//isOut = false;
	field1->setText("");
	field1->append("input: " + lineEdit1->text());
};

void FormDialog::swapper(){
	/*if(!(isOut)){
		field1->setText("");
		field1->append("input: " + lineEdit1->text());
	}*/
	//field1->append("_test_");
	if(lower){
		field1->append("lower case string: " + lineEdit1->text().toLower());

	}
	else{
		field1->append("UPPER CASE STRING: " + lineEdit1->text().toUpper());
	}
	lower = !(lower);
	//isOut = true;
};
```

Скриншоты


Enter your age
41
Exit
Enter your age
35
Exit
lab93
0
0
push button ->
Exit
push button ->
Exit
lab96
Convert
Skype
code
experiments
lab95
0.21030991785715247299484
<-
CE
C
log[y]x
x^y
sin(x)
cos(x)
7
8
9
/
4
5
6
*
1
2
3
-
0
-/+
.
+
=
lab94
0
<-
CE
C
7
8
9
/
4
5
6
*
1
2
3
-
0
-/+
.
+
=

Вывод

Qt предоставляет широкий набор средств для разработки кроссплатформенных графических интерфесов, которые могут создаваться как в специальном конструкторе(Qt Designer), так и с помощью создания объектов специальных классов(QWidget, QPushButton, QSpinBox и т.д.) в коде программы.