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 **09.03.03 Прикладная информатика**

О Т Ч Е Т

по лабораторной работе № 9

Название: Программирование с использованием библиотеки Qt

Дисциплина: Объектно-ориентированное программирование

Студент ИУ6-25 Б — (Группа) — (Подпись, дата) — Дуленков И.Н. (И.О. Фамилия)

Преподаватель — (Подпись, дата) — Веселовский С.А. (И.О. Фамилия)

Москва, 2023

Часть 1. Создание простого приложения

Задание: составить приложение, в котором предлагается ввести возраст с использованием одного из трех вариантов ввода:

1) непосредственного ввода числа,
2) посредством стрелок (элемент типа QSpinBox), последовательно увеличивающих или уменьшающих значение,
3) с помощью специального ползунка (слайдера – элемент типа QSlider).

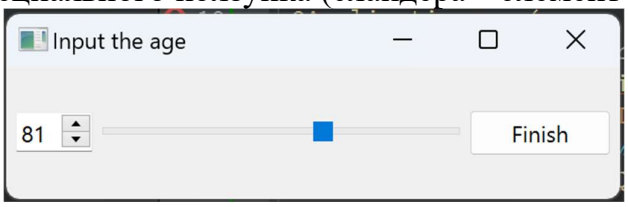

Рисунок 1 – работающая версия программы

Вывод: мы научились создавать простое приложение с различными способами ввода значения.

Часть 2. Создание простого приложения в QtDesigner

Задание: написать программу, позволяющую вводить значение двумя способами и просматривать их.

//Отображаем форму так, как сделано в QtDesigner DialogEx2 * dialog1 = new DialogEx2(); dialog1->show();

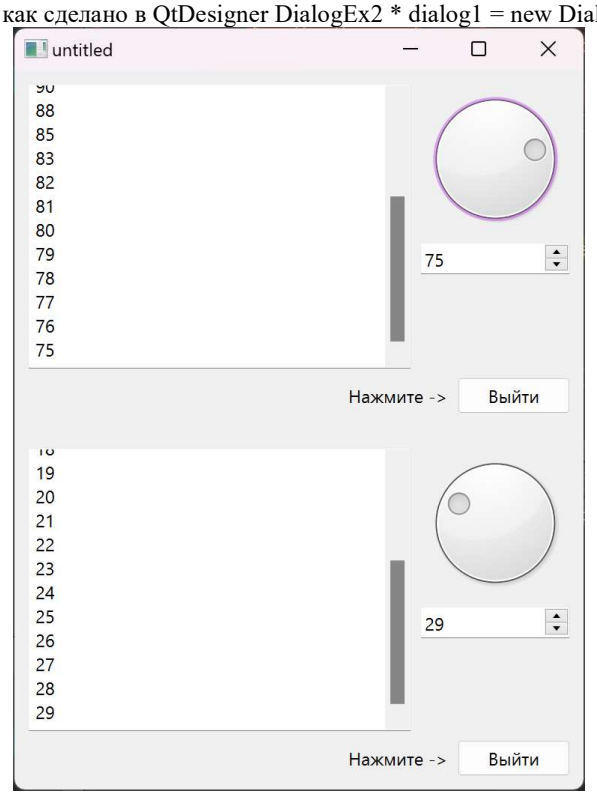

Рисунок 2 – работающая версия программы

Вывод: мы научились работать с формой в Qt Creator, связывать сигналы со слотами, а также создавать несколько подобных форм в одном окне при помощи виджета QSplitter.

Часть 3. Разработка калькулятора

Задание: создать приложение “калькулятор”. Добавьте кнопки, выполняющие: бинарные операции x^y , $\log_y x$ (по аналогии с операциями +,-,/,*), а

также унарные sin(x) и cos(x) (по аналогии с операцией -/+) и разместите этот ряд кнопок вертикально, слева от цифровых кнопок с использованием нового объекта выравнивания (Layout).

Код модуля caldialog.h:

```
#ifndef CALCDIALOG_H
#define CALCDIALOG_H

#include <QDialog>
#include <QLineEdit>
#include <QSignalMapper>
/// Класс, реализующий калькулятор
class CalcDialog: public QDialog
{
   Q_OBJECT
public:
   CalcDialog(QWidget * = nullptr);
   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
   void setNumEdit(QString str);
   /// 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 // CALCDIALOG_H
```

Код модуля caldialog.cpp:

```
#include <QVector>
#include <QGridLayout>
```

```
#include <QPushButton>
#include <QHBoxLayout>
#include <QVBoxLayout>
#include <QDebug>
#include <QMessageBox>

#include "calcDialog.h"

#include <cmath>

// 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 SIN 33
#define COS 34
#define POW 35
#define LOG 36
// количество кнопок в группе, отображаемой в виде сетки
#define GRID_KEYS 16

using namespace std;

/// 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()
{
    // Виджеты располагаются в ряд
    _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));

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

    // new functions
    _btnDescr.push_back(BtnDescr("^", POW));
    _btnDescr.push_back(BtnDescr("log", LOG));
    _btnDescr.push_back(BtnDescr("sin", SIN));
    _btnDescr.push_back(BtnDescr("cos", COS));

    _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
        QGridLayout* gridLayout = new QGridLayout();
        QHBoxLayout* bccKeysLayout = new QHBoxLayout();
        QHBoxLayout* mainKeysLayout = new QHBoxLayout();
        QVBoxLayout* dlgLayout = new QVBoxLayout();
        QVBoxLayout* additionalLayout = 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) // i >= GRID_KEYS + 3 -
пропускаем upper block
              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 + 7) // кнопка из доп. блока - в additionalLayout
              additionalLayout->addWidget(button);
           else
```

```
            { // кнопка "=" - помещаем в блок mainKeysLayout после gridLayout
                mainKeysLayout->addLayout(additionalLayout);
                mainKeysLayout->addLayout(gridLayout);
                mainKeysLayout->addWidget(button);
            }
        }

        qDebug() << "The form is filled with buttons";

        // связываем сигнал из m_pSignalMapper о нажатии со слотом clicked
нашего класса
        connect(m_pSignalMapper, SIGNAL(mappedInt(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
        // qDebug() << id;
        switch (id) {
        // Унарные операции
        case INVERSE: // унарная операция +/-
        {
            setNumEdit(getNumEdit() * -1.0);
            break;
        };
        case SIN: // унарная операция sin
        {
            setNumEdit(sin(getNumEdit()));
            break;
        };
        case COS: // унарная операция
        {
            setNumEdit(cos(getNumEdit()));
            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 LOG:
case POW:
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', 6); // преобразовать вещественное число в строку
    m_pLineEdit->setText(str);
};

// записать число в m_pLineEdit
void CalcDialog::setNumEdit(QString str)
{
    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
    // для sin/cos - вводим число, затем функцию,
    //для log вводим x (m_Val), log, затем y (num) (b=m_Val, a=num)
    double num = getNumEdit();
    bool op_correct = true;
    switch (m_Op)
```

```
        {
        case DIV: {
            if (num != 0) m_Val /= num;
            else {
                QMessageBox::warning(this, "Предупреждение", "Вы не соблюли
ОДЗ");
                op_correct = false;
            }
            break;
        }
        case MUL: {
            m_Val *= num;
            break;
        }
        case PLUS: {
            m_Val += num;
            break;
        }
        case MINUS: {
            m_Val -= num;
            break;
        }
        case POW: {
            m_Val = pow(m_Val, num);
            break;
        }
        case LOG: {
            if (m_Val > 0 and num > 0 and num != 1) {
                m_Val = log(m_Val) / log(num);
            }
            else {
                QMessageBox::warning(this, "Предупреждение", "Вы не соблюли ОДЗ
для log");
                op_correct = false;
            }
            break;
        }
        case EQ: { // если была нажата кнопка "=" - не делать ничего
            m_Val = num;
            break;
        }
        }

        m_Op = curOp; // запомнить результат текущей операции
        if (op_correct) {
            setNumEdit(m_Val); // отобразить результат
```

```
    }
    else {
        setNumEdit("");
    }
    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;
};
```

Код модуля main.cpp:

```
#include "calcdialog.h"

#include <QApplication>

#include <QLocale>
#include <QTranslator>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);

    QTranslator translator;
    const QStringList uiLanguages = QLocale::system().uiLanguages();
    for (const QString &locale : uiLanguages) {
        const QString baseName = "part3_" + QLocale(locale).name();
        if (translator.load(":/i18n/" + baseName)) {
            a.installTranslator(&translator);
            break;
        }
    }
    CalcDialog * dialog = new CalcDialog();
    dialog->show();
    return a.exec();
}
```

Рисунок 3 – работающая версия программы

Вывод: мы научились создавать калькулятор в Qt Creator.

Часть 4. Простейшие элементы ввода-вывода

Задание: написать программу, которая выводит сначала просто введенную строку, потом, ее же, только делая все буквы маленькими, а потом ее же, только делая все буквы заглавными.

Код модуля strdialog.h:

```
#ifndef STRDIALOG_H
#define STRDIALOG_H

#include <QDialog>
#include <QLineEdit>
#include <QPushButton>
#include <QTextEdit>

class strDialog : public QDialog

{
   Q_OBJECT

public:
   strDialog(QWidget *parent = nullptr);
   virtual~strDialog(){};

protected:
   QLineEdit* Edit;
   QPushButton* Button;
   QPushButton* Exit;
   QTextEdit* Text;
   QString getEdit();
private slots:
   /// Слот для обработки нажатий всех кнопок
   void clicked();
};

#endif // STRDIALOG_H
```

Код модуля strdialog.cpp:

```
#include "strdialog.h"
#include <QHBoxLayout>
#include <QVBoxLayout>
#include <QDebug>

strDialog::strDialog(QWidget *parent)
{
   this->setWindowTitle("Преобразование строки");
   Edit= new QLineEdit;
   Button=new QPushButton;
   Exit=new QPushButton;
   Text= new QTextEdit;
   Button->setText("Преобразовать");
   Exit->setText("Выйти");
   Text->setReadOnly(true);
   QVBoxLayout *layout = new QVBoxLayout();
   layout->addWidget(Edit);
   layout->addWidget(Button);
   layout->addWidget(Text);
   layout->addWidget(Exit);
   setLayout(layout);
   connect(Button, SIGNAL (clicked()), this, SLOT (clicked()));
   connect(Exit, SIGNAL(clicked()), this, SLOT(close()));
};
void strDialog::clicked(){
   Text->clear();
   QString out = getEdit();
   Text->append("input: "+out);
   Text->append("all lower: " + out.toLower());
   Text->append("ALL UPPER: "+ out.toUpper());
};
QString strDialog::getEdit(){
   return Edit->text();
};
```

Код модуля main.cpp:

```
#include "strdialog.h"

#include <QApplication>
#include <QLocale>
#include <QTranslator>
```

```
int main(int argc, char *argv[])
{
    QApplication a(argc, argv);

    QTranslator translator;
    const QStringList uiLanguages = QLocale::system().uiLanguages();
    for (const QString &locale : uiLanguages) {
        const QString baseName = "untitled_" + QLocale(locale).name();
        if (translator.load(":/i18n/" + baseName)) {
            a.installTranslator(&translator);
            break;
        }
    }
    strDialog * dialog = new strDialog();
    dialog->show();
    return a.exec();
}
```

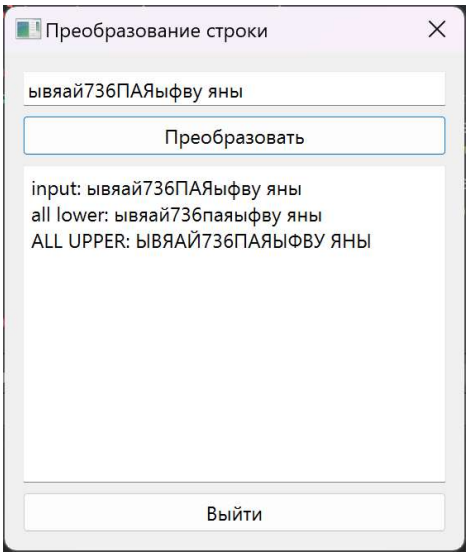

Рисунок 4 – работающая версия программы

Вывод: мы научились работать со строками в форме при помощи среды Qt Creator.