

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ НЕФТИ И ГАЗА
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)
ИМЕНИ И.М. ГУБКИНА»

ФАКУЛЬТЕТ КОМПЛЕКСНОЙ БЕЗОПАСНОСТИ ТЭК
КАФЕДРА БЕЗОПАСНОСТИ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Лабораторная работа №17
по дисциплине «Специализированные языки и технологии
программирования»
на тему «Комплексный проект с применением знаний курса и выявлением
точек роста»

Выполнил студент:
группы КА-22-06
Воронин Алексей Дмитриевич

Преподаватель:
Греков Владимир Сергеевич

Москва, 2025

Оглавление	
Цели работы.....	3
Этап 1: Подготовка.....	4
Этап 2: Разработка.....	4
Тестирование приложения	9
ЗАКЛЮЧЕНИЕ	11
Контрольные вопросы	12

Цели работы

- Применить на практике ключевые темы курса в рамках выбранного проекта.
- Выявить и реализовать улучшения (точки роста) для существующего приложения.
- Исследовать одну новую технологию или подход и продемонстрировать возможности её интеграции.

Этап 1: Подготовка

В качестве основы послужит лабораторная работа 13, в которой было необходимо реализовать калькулятор.

Цели модернизации калькулятора сложения включают:

1. Добавить новые операции: вычитание, умножение, возведение в степень и т.д.(1 час)
2. Добавить возможность решения примеры с несколькими действиями.(1 час)
3. Добавить новые системы счисления: двоичная восьмиричная, шестандцатиричная.(1 час)
4. Добавить в историю отображения о системах счисления, в которых решался пример.(15 минут)
5. Добавить переводчик систем счисления.(2 часа)
6. Ограничить ввод пользователя, чтобы можно было вводить только цифры, знаки и некоторые буквы (30 минут)
7. Тестирование и доработка недостатков(1 час)

Этап 2: Разработка

MainPage.qml:

```
import QtQuick 2.0
import Sailfish.Silica 1.0

ApplicationWindow {
    id: app
    initialPage: mainPage
    allowedOrientations: Orientation.All

    ListModel {
        id: historyModel
    }

    Page {
        id: mainPage

        Column {
            anchors.centerIn: parent
            spacing: Theme.paddingLarge
            width: parent.width - 2 * Theme.paddingLarge

            ComboBox {

```

```

        id: modeSelector
        width: parent.width
        label: "Система счисления"
        menu: ContextMenu {
            MenuItem { text: "Десятичная" }           // index 0
            MenuItem { text: "Двоичная" }             // index 1
            MenuItem { text: "Восьмеричная" }         // index 2
            MenuItem { text: "Шестнадцатеричная" } // index 3
        }
        currentIndex: 0
    }

    TextField {
        id: expressionField
        width: parent.width
        height: 120
        placeholderText: "Введите пример"
        horizontalAlignment: Text.AlignHCenter

        onTextChanged: {
            var cleaned = text
            switch (modeSelector.currentIndex) {
                case 1: // двоичная
                    cleaned = cleaned.replace(/[^01+\-*^() . ]/gi, "")
// без / и корней
                    break
                case 2: // восьмеричная
                    cleaned = cleaned.replace(/[^0-7+\-*^() . ]/gi,
 "") // без / и корней
                    break
                case 3: // шестнадцатеричная
                    cleaned = cleaned.replace(/[^0-9A-Fa-f+\-*^() . ]/gi, "") // без / и корней
                    break
                default: // десятичная
                    cleaned = cleaned.replace(/[^0-9+\-*/*() . \sqrt]/gi, "")
            }
            if (cleaned !== text) {
                text = cleaned
            }
        }
    }

    Label {
        text: "Ответ"
        font.pixelSize: 30
        horizontalAlignment: Text.AlignHCenter
        anchors.horizontalCenter: parent.horizontalCenter
    }

    TextField {
        id: resultField
        width: parent.width
        height: 120
        placeholderText: "0"
        readOnly: true
        horizontalAlignment: Text.AlignHCenter
    }

```

```

        Button {
            text: "Подтвердить"
            anchors.horizontalCenter: parent.horizontalCenter
            height: 70
            onClicked: {
                try {
                    var expr = expressionField.text

                    var base = 10
                    if (modeSelector.currentIndex === 1) base = 2
                    if (modeSelector.currentIndex === 2) base = 8
                    if (modeSelector.currentIndex === 3) base = 16

                    if (base !== 10) {
                        expr = expr.replace(/\b[0-9A-Fa-f]+\b/g,
function(match) {
                        return parseInt(match, base)
                    })
                }

                expr = expr.replace(/sqrt\(([^\)]+)\)/gi,
"Math.sqrt($1)")

                expr = expr.replace(/\sqrt(\d+(\.\d+)?)/g,
"Math.sqrt($1)")

                expr = expr.replace(/(\d+(\.\d+)?)\^(\d+(\.\d+)?)/g,
"Math.pow($1,$3)")

                var res = eval(expr)

                if (isNaN(res)) {
                    resultField.text = "Ошибка"
                } else {
                    if (base !== 10) {
                        resultField.text =
Math.floor(res).toString(base).toUpperCase()
                        historyModel.append({
                            example: expressionField.text + " = " +
Math.floor(res).toString(base).toUpperCase() +
                                " (" + base + " + " + base + ")"
                        })
                    } else {
                        resultField.text = res.toString()
                        historyModel.append({
                            example: expressionField.text + " = " +
res
                        })
                    }
                }
            } catch (e) {
                resultField.text = "Ошибка"
            }
        }
    }

    Button {

```

```

        text: "История"
        anchors.horizontalCenter: parent.horizontalCenter
        height: 70
        onClicked: pageStack.push(historyPage)
    }
    Button {
        text: "Переводчик"
        anchors.horizontalCenter: parent.horizontalCenter
        height: 70
        onClicked:
pageStack.push(Qt.resolvedUrl("ConverterPage.qml"))
    }

}
}

Page {
    id: historyPage

    SilicaListView {
        anchors.fill: parent
        model: historyModel
        header: PageHeader { title: "История" }

        delegate: ListItem {
            Label {
                text: example
                anchors.verticalCenter: parent.verticalCenter
            }
        }
    }
}
}

```

ConverterPage.qml:

```

import QtQuick 2.0
import Sailfish.Silica 1.0

Page {
    id: converterPage

    Column {
        anchors.centerIn: parent
        spacing: Theme.paddingLarge
        width: parent.width - 2 * Theme.paddingLarge

        ComboBox {
            id: fromSelector
            width: parent.width
            label: "Выбор системы ввода"
            menu: ContextMenu {
                MenuItem { text: "Десятичная" }
                MenuItem { text: "Двоичная" }
                MenuItem { text: "Восьмеричная" }
                MenuItem { text: "Шестнадцатеричная" }
            }
            currentIndex: 0
        }
    }
}

```

```

    }

ComboBox {
    id: toSelector
    width: parent.width
    label: "Выбор системы вывода"
    menu: ContextMenu {
        MenuItem { text: "Десятичная" }
        MenuItem { text: "Двоичная" }
        MenuItem { text: "Восьмеричная" }
        MenuItem { text: "Шестнадцатеричная" }
    }
    currentIndex: 1
}

Row {
    spacing: Theme.paddingLarge
    width: parent.width

    TextField {
        id: numberField
        width: parent.width / 2 - Theme.paddingLarge / 2
        height: 100
        placeholderText: "Введите число"
        horizontalAlignment: Text.AlignHCenter
        onTextChanged: {
            var allowed
            switch (fromSelector.currentIndex) {
                case 0: allowed = /^[^0-9]/gi; break
                case 1: allowed = /^[^01]/gi; break
                case 2: allowed = /^[^0-7]/gi; break
                case 3: allowed = /^[^0-9A-Fa-f]/gi; break
            }
            var cleaned = text.replace(allowed, "")
            if (cleaned !== text) text = cleaned
        }
    }

    TextField {
        id: resultField
        width: parent.width / 2 - Theme.paddingLarge / 2
        height: 100
        placeholderText: "Результат"
        readOnly: true
        horizontalAlignment: Text.AlignHCenter
    }
}

Button {
    text: "Перевести"
    height: 70
    anchors.horizontalCenter: parent.horizontalCenter
    onClicked: {
        var fromBase = [10, 2, 8, 16][fromSelector.currentIndex]
        var toBase = [10, 2, 8, 16][toSelector.currentIndex]

        if (numberField.text.length === 0) {
            resultField.text = "Ошибка"
            return
        }
    }
}

```

```
        var decimalValue = parseInt(numberField.text, fromBase)
        if (isNaN(decimalValue)) {
            resultField.text = "Ошибка"
        } else {
            resultField.text =
decimalValue.toString(toBase).toUpperCase()
        }
    }
}
```

Тестирование приложения

Соберем и запустим приложение.

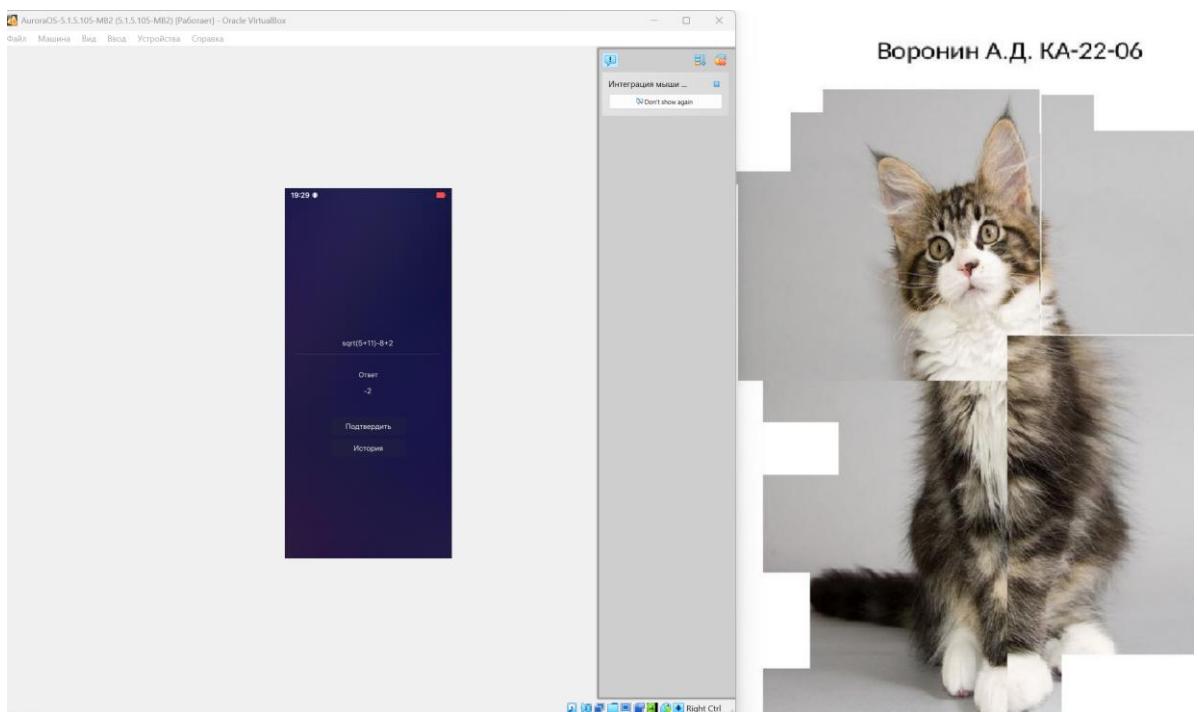


Рисунок 1 – Пример ввода

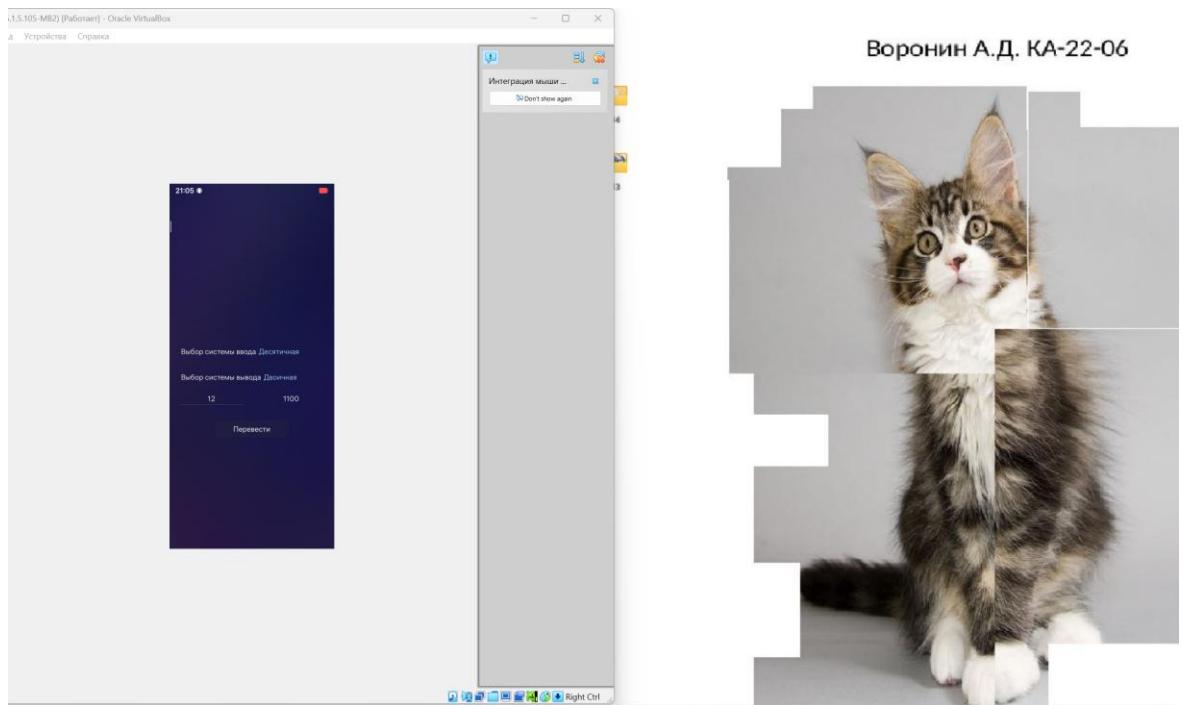


Рисунок 2 – Перевод систем счисления

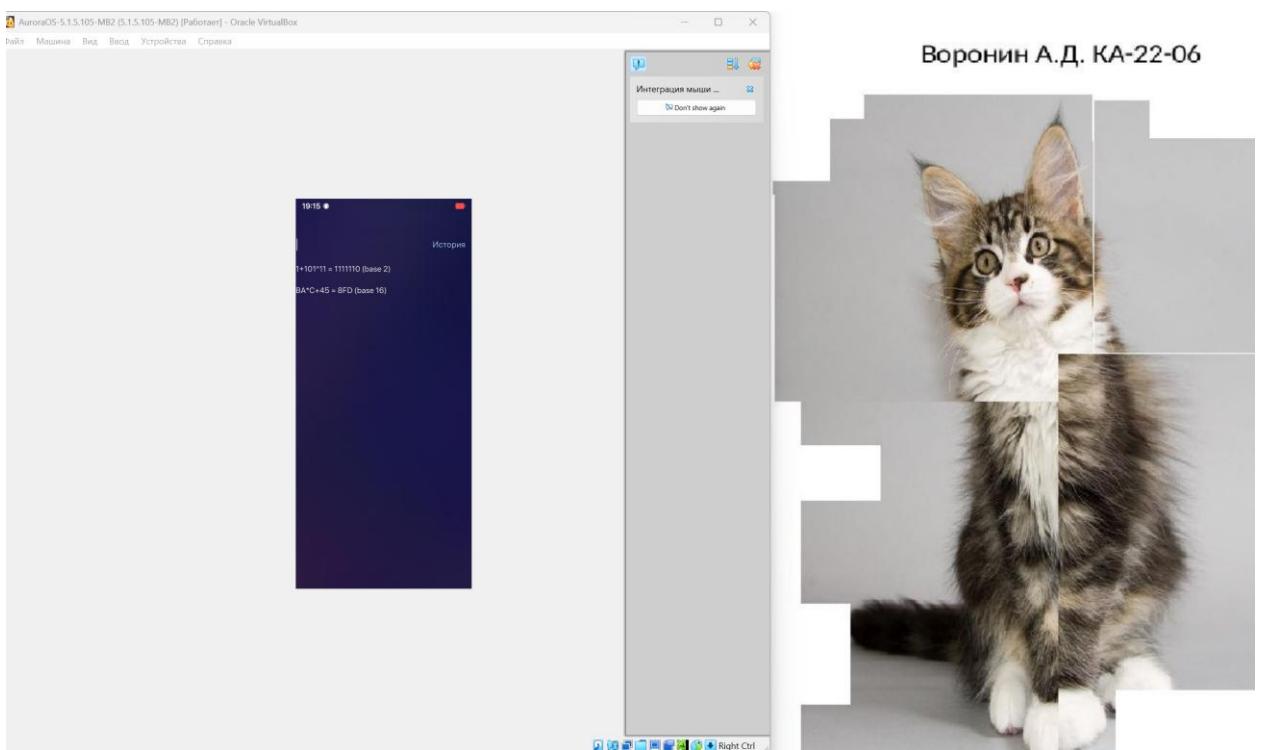


Рисунок 3 – История ввода примеров

ЗАКЛЮЧЕНИЕ

В заключение, подводя итог проделанной работе, все поставленные в начале цели и задания данной лабораторной работы были достигнуты в полном объеме.

Контрольные вопросы

1. Какие улучшения вы внесли и почему они важны?

1. Возможность вводить примеры с несколькими действиями. Данное введение является основным в приложении, что значимо расширяет его функционал.

2. Возможность решать примеры для нескольких систем счисления, что также является базовым улучшением функционала.

3. Ограничение ввода – избавляет от нежелательного ввода со стороны пользователя.

4. Перевод в разные системы счисления. Сделано для удобства пользователя, чтобы можно было сразу перевести в нужную систему счисления.

2. Какие точки роста вы выявили?

1. Поддержка новых математических функций: \sin , \cos и т.д.

2. Поддержка нецелых и отрицательных чисел в недесятичных системах счисления.

3. Доработка общего интерфейса приложения.

3. Какую новую технологию вы исследовали, и какие преимущества она дала?

В проекте были исследованы новые возможности Sailfish Silica:

Ограничение ввода в зависимости от системы счисления

Для каждой системы счисления реализовано ограничение допустимых символов. Это предотвращает ошибки ещё на этапе ввода.

Для вычислений калькулятора применялся метод eval, что позволило гибко обрабатывать сложные арифметические выражения, включая возведение в степень и квадратные корни, без необходимости ручного разложения формул.

Использование ComboBox и готовых компонентов Sailfish Silica. Для выбора системы счисления, режима калькулятора и других настроек использованы выпадающие списки (ComboBox) и стандартные UI-компоненты Sailfish Silica. Это позволило быстро создать интуитивно понятный интерфейс.

4. Как проводилось тестирование и оптимизация проекта?

Функциональное тестирование

- Проверялось корректное выполнение арифметических операций
- В режиме перевода систем счисления тестировался ввод чисел в разных системах счисления и правильность преобразования.
- Проверялось ограничение ввода для различных систем счисления.
- Проводилось тестирование интерфейса: правильное отображение полей, корректная работа кнопок .

Негативное тестирование

- Проверялись ошибки: пустой ввод, ввод недопустимых символов. В таких случаях приложение должно было выводить сообщение "Ошибка" вместо падения.

Оптимизация проекта:

- Дублирующийся код был вынесен в универсальные функции.
- Добавлено ограничение ввода в зависимости от выбранной системы счисления — это предотвращает ошибки ещё на этапе ввода.

5. Какие методы профилирования вы использовали и какие результаты получили?

В код добавлялись временные функции вывода в консоль для проверки скорости вычислений и корректности работы функций. Это помогало выявить, где выражения обрабатываются дольше всего. Так как особо сложных операций не выполняется то вывод был мгновенным.

Визуальное профилирование показало, что интерфейс работает стабильно, нет "фризов" при переключении страниц и прорисовке элементов.