УДК 004.422

Рабданова В.В., кандидат экономических наук, доцент
доцент кафедры «Программная инженерия и искусственный интеллект»
Восточно-Сибирский государственный университет технологий и управления
Россия, г. Улан-Удэ
Дугаров В.А.,
студент
4 курс, факультет Компьютерных наук и технологий
Восточно-Сибирский государственный университет технологий и управления
Россия, г. Улан-Удэ

РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ «СИМУЛЯТОР ЛОГИЧЕСКИХ СХЕМ»

Аннотация: *В данной статье рассматривается разработка мобильного приложения «Симулятор логических схем». Основными этапами в ходе исследования являются: определение логической схемы; определение интерфейса мобильного приложения; реализация работы симулятора логических схем на высокоуровневом языке программирования* Kotlin.

Ключевые слова: *мобильное приложение, симулятор, логическая схема, математическая логика, теория алгоритмов.*

Annotation: *This article discusses the development of a mobile application "Logic Circuit Simulator". The main stages in the course of the study are: definition of a logical circuit; definition of the interface of a mobile application; implementation of the operation of a logic circuit simulator in the high-level Kotlin programming language.*

Key words: *mobile application, simulator, logic circuit, mathematical logic, theory of algorithms.*

В ходе обучения дисциплине «Математическая логика и теория алгоритмов» возникла потребность в разработке мобильного приложения для построения логических схем. Данное приложение –симулятор логических схем - было разработано для облегчения изучения основ алгебры логики. Симулятор обладает средствами визуализации и интуитивно понятным интерфейсом для пользователя. Именно поэтому этот логический симулятор был выбран для подготовки серии практических работ по дисциплине.

Логической схемой называется совокупность логических электронных элементов, соединенных между собой таким образом, чтобы выполнялся заданный закон функционирования схемы, иначе говоря, — выполнялась заданная логическая функция [1]. Она широко используется в электронике, телекоммуникациях и других сферах для построения вычислительных устройств и систем.

Известно, что всякую булеву функцию от n аргументов можно выразить через элементы: конъюнкцию, дизъюнкцию и отрицание, которые составляют так называемый основной базис с технологической точки зрения [2, с.155]. Эти элементы формируют основу для создания сложных цифровых устройств.

Программа реализована на языке Kotlin под платформу Android. Для реализации интерфейса был использован Jetpack Compose. В нем используется декларативный способ задания интерфейса, то есть построение интерфейса происходит непосредственно в коде программы.

Одной из ключевых особенностей программы является возможность виртуального моделирования. Пользователи могут создавать логические схемы в виртуальной среде. В данной работе рассмотрено, как это взаимодействие происходит внутри программы.

Отрисовка логических элементов будет проходить в компоненте Canvas. Также для реализации соединения элементов было решено добавить на входы и выход элемента кнопки, которые будут отвечать за соединение элементов.

Для хранения информации о типах элементов было создано перечисление «COMPONENT_TYPE_CLASS»

Данное перечисление включает такие компоненты как AND, OR, INV, NAND и NOR. Код данного перечисления изображен на рисунке 1.

```
enum class COMPONENT_TYPE_CLASS {
    AND,
    OR,
    INV,
    NAND,
    NOR
}
```

Рисунок 1. Перечисление типов элементов

Для удобства создания элементов и линий между ними были реализованы следующие интерфейсы и классы.

Интерфейс IComponent содержит в себе информацию о логическом элементе, координаты элемента на форме, информацию о кнопках элемента, и тип элемента. Код интерфейса изображен на рисунке 2.

```
interface IComponent {
    val layout_x: MutableState<Float>;
    val layout_y: MutableState<Float>;
    val enter1_button: ButtonPoint
    val enter2_button: ButtonPoint
    val exit_button: ButtonPoint
    val component_type: COMPONENT_TYPE_CLASS
}
```

Рисунок 2. Интерфейс IComponent

Реализацией данного интерфейса является класс MyComponent. Код данного класса изображен на рисунке 3.

```
data class MyComponent(
    override val layout_x: MutableState<Float>,
    override val layout_y: MutableState<Float>,
    override val enter1_button: ButtonPoint,
    override val enter2_button: ButtonPoint,
    override val exit_button: ButtonPoint,
    override val component_type: COMPONENT_TYPE_CLASS

) : IComponent
```

Рисунок 3. Реализация интерфейса IComponent

Таким же образом были реализованы другие элементы. На форме были размещены кнопки для добавления элементов. Для хранения элементов был использован список componentList, а для хранения линий между элементами - список lineList. При добавлении элемента в componentList добавляется новый компонент, и переменная elementCount увеличивается на 1. При изменении переменной elementCount вызывается функция DrawElement(), которая создает элемент на форме.

Для реализации соединения двух элементов линией были использованы две переменные activePoint1 и activePoint2 (рисунок 4). В них записывается нажатый вход/выход и если были нажаты две кнопки, то в LineList добавляются эти кнопки, и переменная lineCount увеличивается на 1 и между ними рисуется линия. Реализация отрисовки линии изображена на рисунках 5, 6.

```
activePoint1: MutableState<ButtonPoint?>,
activePoint2: MutableState<ButtonPoint?>
```

Рисунок 4. Переменные для отрисовки линии

```
onClick = {
    if (activePoint1.value == null) {
        activePoint1.value = component.enter1_button
    } else if (activePoint2.value == null) {
        activePoint2.value = component.enter1_button
    }
    if (activePoint1.value != null && activePoint2.value != null) {
        lineList.add(LineBetweenComponent(activePoint1.value!!, activePoint2.value!!))
        lineCount.value++
    }
},
```

Рисунок 5. Код добавление элементов в lineList

```
repeat(lineCount.value) { index ->
    drawLine(
        color = Color.Black,
        start = Offset(
            x: lineList[index].start_button.layout_x.value + 7.dp.toPx(),
            y: lineList[index].start_button.layout_y.value + 7.dp.toPx(),
        ),
        end = Offset(
            x: lineList[index].end_button.layout_x.value + 7.dp.toPx(),
            y: lineList[index].end_button.layout_y.value + 7.dp.toPx(),
        ),
        strokeWidth = 2.dp.toPx(),
        cap = StrokeCap.Round
    )
    activePoint1.value = null
    activePoint2.value = null
}
```

Рисунок 6. Код отрисовки линий между элементами

Элементы можно перемещать, используя drag-and-drop. Данная функция была реализована с помощью pointerInput и detectDragGestures. Код представлен на рисунке 7.

```
.pointerInput(Unit) { this: PointerInputScope
    detectDragGestures { change, dragAmount ->
        change.consume()
        component.layout_x.value += dragAmount.x
        component.layout_y.value += dragAmount.y
        component.enter1_button.layout_x.value += dragAmount.x
        component.enter1_button.layout_y.value += dragAmount.y
        if (component.component_type.name != "INV") {
            component.enter2_button.layout_x.value += dragAmount.x
            component.enter2_button.layout_y.value += dragAmount.y
        }
        component.exit_button.layout_x.value += dragAmount.x
        component.exit_button.layout_y.value += dragAmount.y
    }
}
```

Рисунок 7. Код реализации drag-and-drop

Пример построенной схемы в приложении изображен на рисунке 8.

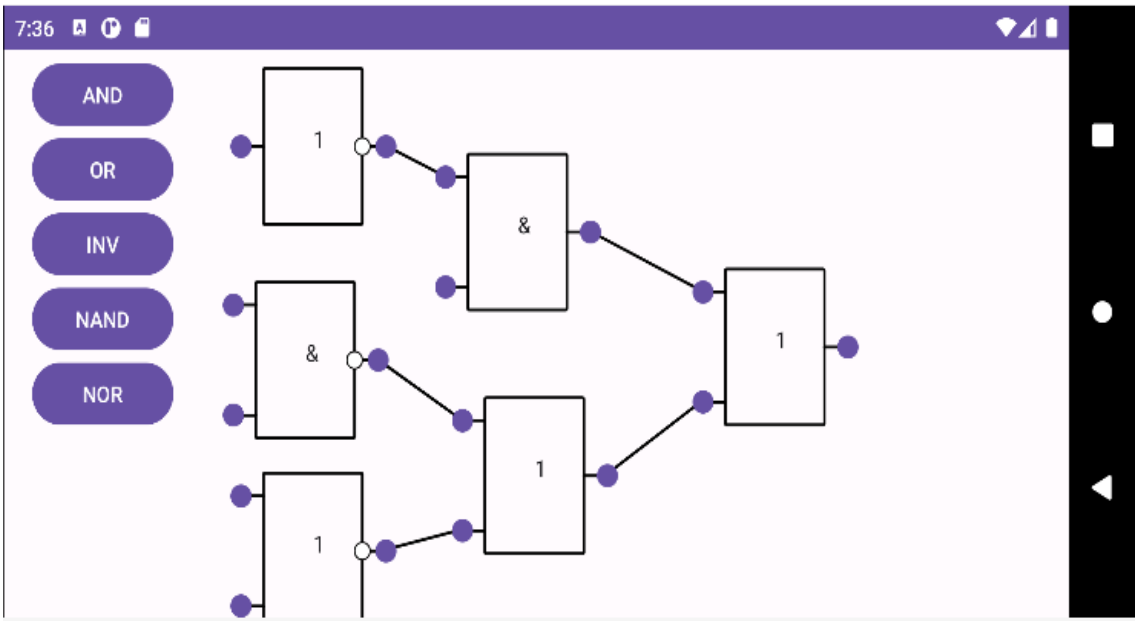

Рисунок 8. Пример работы приложения

Разработанное приложение может быть использовано для обучения студентов дисциплинам: «Математическая логика и теория алгоритмов», «Программирование для мобильных устройств». Мобильное приложение «Симулятор логических схем» становится инструментом, способствующим

более широкому пониманию теории алгоритмов и объектно-ориентированного программирования в целом.

Использованные источники:

1. Самостоятельное изучение схемотехники. Основные понятия. Часть 1. URL: (дата обращения: 19.12.2023)

2. Оленев А.А., Киричек К.А., Потехина Е.В. Математическая логика: построение логических схем из логических элементов в Maple // Вест. КРАУНЦ. Физ.-мат. науки. 2021. №3. URL: https://cyberleninka.ru/article/n/matematicheskaya-logika-postroenie-logicheskih-shem-iz-logicheskih-elementov-v-maple (дата обращения: 19.12.2023.

3. «Android Studio» // Википедия, свободная энциклопедия [онлайн]. 19.12.2023. URL: http://ru.wikipedia.org/wiki/Android Studio