ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ МОСКОВСКОЙ ОБЛАСТИ

«Университет «Дубна»

ИНСТИТУТ СИСТЕМНОГО АНАЛИЗА И УПРАВЛЕНИЯ

Кафедра распределенных информационно-вычислительных систем

**КУРСОВАЯ РАБОТА**

по дисциплине

**«Объектно-ориентированное программирование»**

Реализация программы для моделирования клеточных автоматов с произвольными правилами.

**ТЕМА**: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(наименование темы)

1255

**Выполнил**: студент группы \_\_\_\_\_\_\_\_\_

Фролов Евгений Витальевич

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

(Ф.И.О.)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(подпись студента)

**Руководитель**:

асс Тестов М.С.

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

(ученая степень, ученое звание, занимаемая должность, ФИО)

Дата защиты: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись руководителя)

Дубна, 2022

**Содержание**

[Введение 3](#_Toc105378206)

[Постановка задачи 4](#_Toc105378207)

[Теоретическая часть 5](#_Toc105378208)

[Практическая часть 9](#_Toc105378209)

[Список литературы 18](#_Toc105378210)

# Введение

Клеточный автомат — дискретная модель, пространство из прилегающих друг к другу клеток, образующих решетку (далее сетка), каждая из которых может принимать одно из конечного количества состояний, изменяющихся в каждом поколении по заданным правилам.

В настоящее время клеточные автоматы используются в криптографии, биологии, для моделирования физических процессов, и поведения людей, а также во многих других областях прикладной и научной деятельности.

# Постановка задачи

**Цель работы**

Целью выполнения работы является создание программы для моделирования клеточных автоматов с заданными правилами, предназначенное для помощи в решении аналитических задач в сфере физики, биологии, социологии, криптографии и других областях.

**Исходные данные**

Задаваемые пользователем правила, начальное состояние клеток в сетке, задаваемое пользователем с использованием клавиш и жестов мыши.

**Априорные модельные представления**

Модель взаимодействия с пользователем — графический интерфейс. Интерфейс предполагает возможность изменения данных симуляции: правил, сетки, цветовой палитры используемой для визуализации сетки, рисование на сетке, изменения размеров сетки и ее заполнение заданными величинами.

Модель клеточного автомата – ограниченная сетка, элементами которой являются квадраты, каждая клетка которой может принимать конечное количество состояний — от 2 до 99 включительно, с возможностью настройки ее размера и состояния клеток в текущем поколении, правил, а также параметров ее отображения (цветовая палитра, положение на экране, размер и т.п.).

Язык описания правил, с помощью которых можно будет описать правила большого множества правил клеточных автоматов.

**Результат работы**

Программа визуализации клеточных автоматов с интерфейсом для изменения параметров модели.

**Критерии оценивания**

Возможность описания правил многих клеточных автоматов таких как “Игра жизнь” и подобные ей, клеточные автоматы серии “Поколения”, “*Wireworld*”, “Муравей Лэнгтона” и подобные, и наблюдать за визуализацией их сетки, изменять состояние клеток в ней в текущем поколении, а также параметров визуализации, таких как палитра, масштаб и положение отображаемой области на экране.

# Теоретическая часть

Клеточный автомат — дискретная модель, пространство из прилегающих друг к другу клеток, образующих сетку, размер которой может быть конечный или бесконечный. Каждая клетка может принимать одно из конечного множества состояний. Для каждой клетки определено множество клеток, называемых окрестностью. Модель также включает в себя правила перехода клеток из одного состояния в другое на основе текущего состояния клетки и/или ее окрестности (далее правила). Сетка изменяется с каждым поколением, используя правила для определения нового состояния клетки в следующем поколении для каждой из клеток сетки определяется следующее поколение клеточного автомата.

Каждый из параметров: сетка, правила, окрестность; могут быть представлены по-разному, что приводит к большому разнообразию клеточных автоматов. сетка может быть составлена из разных компонентов – квадратов или шестиугольников, окрестность клетки может быть определена по-разному, две самые распространенные разновидности окрестностей — это окрестность Мура и окрестность фон Неймана, каждая из которых так же подразделяется на порядки (см. рис. 1).

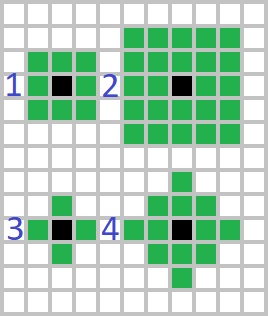


Рис. 1. Окрестности: 1 – Мура 1-го порядка, 2 – Мура 2-го порядка, 3 – фон Неймана 1-го порядка, 4 – фон Неймана 2-го порядка.

**Язык описания правил**

Необходимо разработать язык описания правил, с помощью которого можно будет описать правила клеточных автоматов: “Игра жизнь” и подобных ей, серии “Поколения”, “*Wireworld*”, “Муравей Лэнгтона” и подобных.

Правила должны представлять собой текстовый файл, который пользователь может создать сам и позже открыть в программе, расширение файла будет иметь вид “*.car\**”, где символом \* обозначено количество состояний которое может принимать клетка и может принимать значения от 2 до 99 включительно.

Структура файла представляет собой набор строк (далее инструкций) каждая из которых должна иметь начальное состояние клетки, итоговое состояние клетки и условия перехода, все инструкции применяются последовательно к каждой клетке, итоговым результатом шага является итоговое значение последней инструкции условия которой были выполнены.

Для того что бы возможно было описать правила всех задуманных клеточных автоматов, необходимо два вида инструкций.

Инструкции условия которых представляют собой функцию сравнения вида *a(n)<b*, где *a(n)* – количество клеток в окрестности рассматриваемой клетки в состоянии *n*, *b* – целое число, а между *a(n)* и *b* пользователь может задавать любой знак из набора >,<,≤,≥,=,≠.

Инструкции-шаблоны, где условием является некая заданная окрестность, и условие считается выполненным если состояния клеток в окрестности рассматриваемой клетки соответственно равны состояниям клеток в заданной окрестности. Заданная окрестность может быть неполной, то есть некоторые клетки могут быть пропущены, в таком случае состояние соответствующей клетки в окрестности рассматриваемой клетки может быть любым.

В итоге всех вышеперечисленных критериев была разработана следующая структура языка для описания правил:

Инструкция сравнения вида “*f*[*c*](*r(o)*):*a*“, где *c* – состояние клетки, *a* – итоговое состояние клетки в следующем поколении; *r(o)* – логическая функция от окрестности рассматриваемой клетки, вида “[c1]>N1,…,[cn]>Nn”, где [с] – функция возвращающая количество клеток в окрестности в состоянии c и между ними может стоять любой знак сравнения , а N произвольные числа, результатом которой является логическая величина; а *f* – литерал указывающий что следует использовать соседство фон Неймана 1-го порядка при подсчете количества клеток с указанным состоянием в окрестности, если он пропущен то будет использовано соседство мура 1-го порядка

Инструкция-шаблон, вида “[*c00*,*c01*,*c02*,*c03*,*c10*,*c11*,*c12*,*c20*,*c21*,*c22*]:*a*”, где *сij* – это состояние клеток в окрестности клетки и ее самой (см. рис. 2), некоторые или все значения могут отсутствовать, тогда будет приниматься любое состояние клетки по этому индексу, *a* – итоговое состояние клетки в следующем поколении.

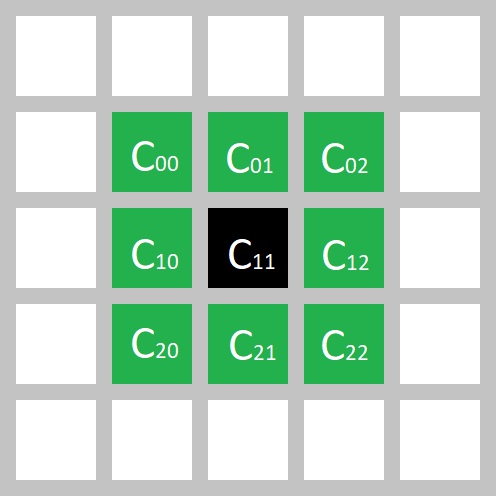


Рис. 2. Порядок расположения клеток в инструкции шаблоне

Инструкции разделяются символом “*;*” и/или переносом строки.

Инструкции имеют возрастающий приоритет, при выполнении условий нескольких инструкций, клетка примет итоговое значение инструкции, которая встречается в файле позже.

В качестве примера формализуем правила двух клеточных автоматов на описанном выше языке, рассмотрим правила “Игры жизнь” и “*Wireworld*”.

В “Игре жизнь” клетки могут принимать 2 состояния – живые и мертвые, а правила звучат так – если мертвая клетка имеет по соседству ровно три живые клетки, она оживает, если живая клетка имеет по соседству 2 или 3 живых клетки, то она продолжает жить, а иначе умирает. В качестве окрестности используется соседство Мура первого порядка. На вышеописанном языке эти правила будут выглядеть так: *“[0]([1]==3):1;[1]([1]!=2,[1]!=3):0;”*.

В “*Wireworld*” клетки могут принимать 4 состояния ­– пустая, голова сигнала, хвост сигнала, проводник. Правила “*Wireworld*” звучат так:

1. Пустая клетка всегда остается пустой
2. Голова сигнала превращается в хвост сигнала в следующем поколении.
3. Хвост сигнала превращается в проводник.
4. Проводник превращается в голову сигнала при условии, что на соседних клетках есть ровно 1 или 2 головы сигнала, иначе остается проводником.

На описанном языке эти правила будут выглядеть так: “[,,,,1,,,,]:2;[,,,,2,,,,]:3;[3]([1]>=1,[1]<=2):1;”, так как нам не нужно явно указывать что пустая клетка всегда остается пустой.

**Базовая архитектура приложения**

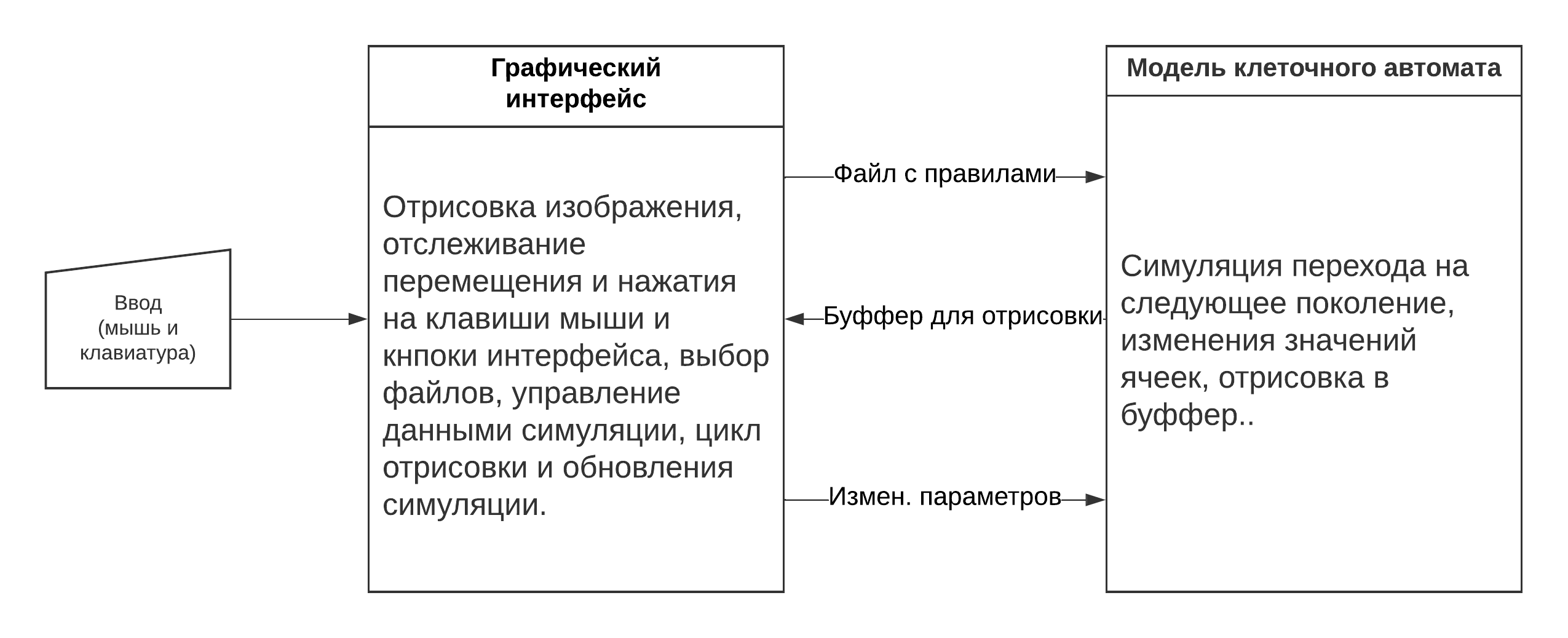


Рис. 3. Структура приложения

Структурно приложение будет представлять собой два модуля. Модуль отвечающий за моделирование клеточных автоматов и их визуализацию состоящий из подмодулей. Модуль графического интерфейса, посредством которого конечный пользователь будет взаимодействовать с моделью.

# Практическая часть

**Структура проекта и инструменты**

Для написания программы использовалась интегрированная среда разработки *Microsoft Visual Studio*, в связи с наличием удобных инструментов менеджмента решений и проектов, конструктора графических интерфейсов.

Было создано три проекта в одном решении (см. рис. 4).

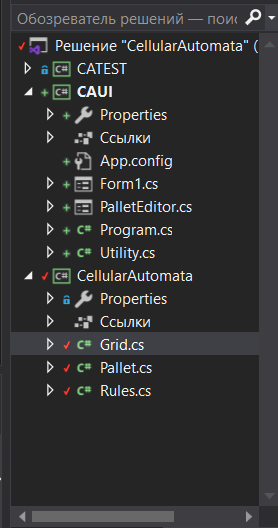


Рис. 4. Структура решения в *Visual Studio*

Первый – *CellularAutomata* – содержащий в себе классы *Grid* – сетка, основной класс моделирующий все процессы клеточного автомата, *Rules* – класс ответственный за загрузку правил клеточного автомата и исполнение инструкций, *Pallet* – палитра, используемая для визуализации сетки.

Второй – *CATEST* – используется исключительно для теста модуля *CellularAutomata* и рассматриваться не будет.

Третий – *CAUI* – графический интерфейс, используемый для визуализации сетки клеточного автомата, задания и редактирования его параметров.

**Реализация модели клеточного автомата**

Класс *Grid* представляет собой сетку, содержит в себе массив состояний клеток, методы для перехода на следующее поколение, объекты классов *Rules* и *Pallet*, методы взаимодействия с сеткой – заполнение одним состоянием и заполнение сетки случайными состояниями, параметры и методы необходимые для визуализации сетки средствами *.NET Framework* (см. рис. 5). Все методы являются защищенными для параллельного доступа, путем использования мьютексов, так как предполагается что визуализация сетки в графическом интерфейсе и цикл перехода на новые поколения будет осуществляться в многопоточном режиме.

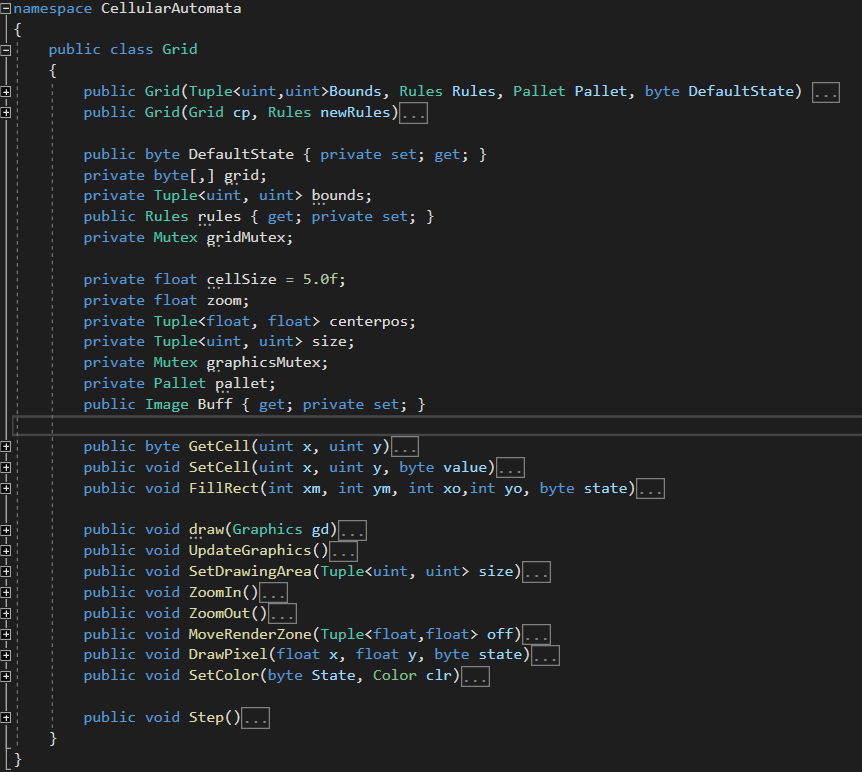


Рис. 5. Класс *Grid*

Основные методы класса *Grid*: метод перехода на следующее поколение – Step (см. рис. 6) – создает копию сетки и в цикле для каждой клетки используя правила определяет состояние клетки в следующем поколении, после чего заменяет оригинал на копию; методы визуализации сетки используются для отрисовки в буффер текущего состояния сетки используя палитру, изменения размера буффера, координат центра отображаемой области и масштаба, на основе которых используя простейшие преобразования координат вычисляется местоположение и видимость клеток сетки на отображаемой поверхности.

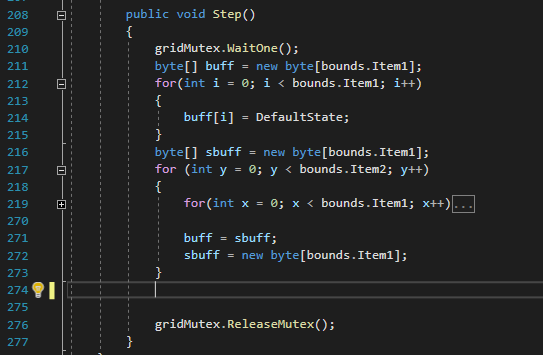


Рис. 6. Метод *Step* класса *Grid*

Класс *Rules* представляет из себя набор правил, описанных ранее в теоретической части, преобразованный в удобный для использования программе вид – коллекцию объектов, каждый из которых является реализацией отдельной инструкции, типа *IRule* (см. рис. 7,8). Для преобразования правил из формата текстового файла в коллекцию объектов типа *IRule* используются регулярные выражения (см. рис. 9).

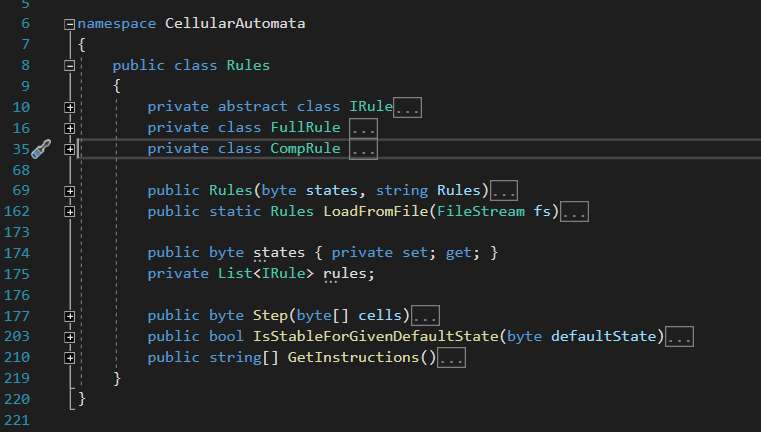


Рис. 7. Класс *Rules*

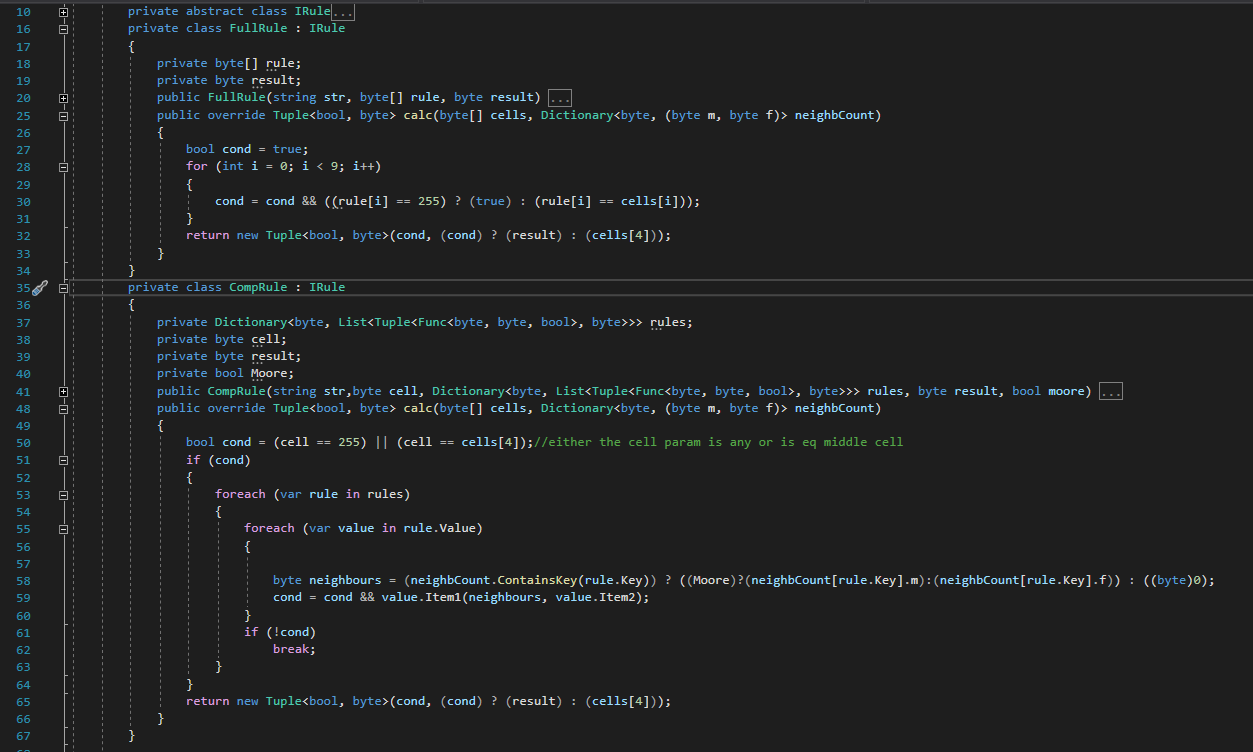


Рис. 8. Классы *FullRule* и *CompRule*, используемые для хранения инструкций

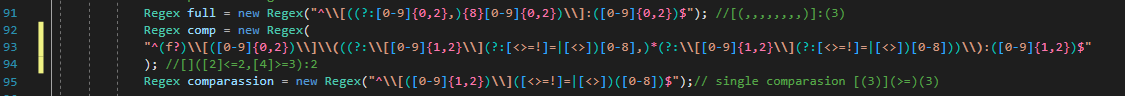


Рис. 9. Использование регулярных выражений для парсинга инструкций.

Класс *Pallet* – обертка над коллекцией цветов представляющий доступ к объектам кистей цветов из коллекции для последующего использования при отрисовке, а также редактирования этих цветов.

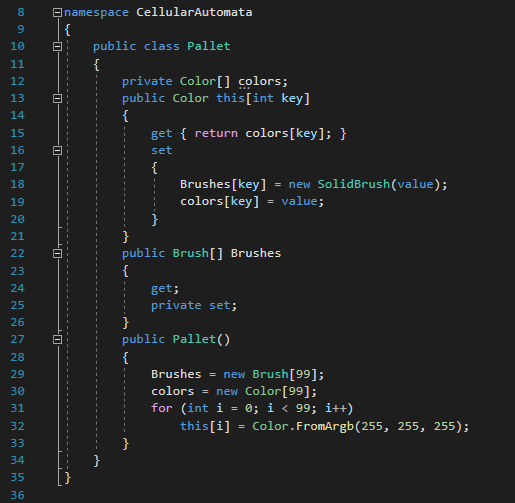


Рис. 10. Класс *Pallet*

**Реализация графического интерфейса**

Основное окно графического интерфейса состоит из двух частей – основная панель и панель инструментов (см. рис. 11).

Основная панель – панель на которой отображается сетка, также используется для перемещения отображаемой области, изменения масштаба и рисования на отображаемой части сетки используя клавиши и движения мыши.

Панель инструментов используется для изменения размеров сетки, ее заполнения выбранным состоянием или случайными состояниями, выбора правил и редактирования палитры, состоит из следующих элементов: кнопки взаимодействия с сеткой, правилами, палитрой, очистки сетки, выбора “цвета” кистей используемой при “рисовании” на основной панели, кнопки паузы/продолжения симуляции и регулятора скорости (поколений в секунду).



Рис. 11. Основное окно приложения: 1 – основная панель, 2 – панель инструментов

**Оценка итогового результата**

Для выполнения всех критериев итогом работы должна быть программа, в которой мы сможем наблюдать визуализацию моделей клеточных автоматов “Игра жизнь” и подобных ей, клеточных автоматы серии “Поколения”, “*Wireworld*”, “Муравей Лэнгтона”.

Реализация “Игры жизнь” на языке описания правил (далее ЯОП) будет состоять из трех инструкций (см. рис. 12), разработанный язык так же позволяет описать правила множества других подобных “Игре жизнь” правил таких как правила “Семена” ­– клетка становится живой если в ее окрестности есть ровно 2 живые клетки и умирает всегда, (см. рис. 13) или “*HighLife*” – клетка становится живой если в ее окрестности есть 3 или 6 живых клеток и умирает если в ее окрестности менее 1-ой или более 3-х живых клеток(см. рис. 13).

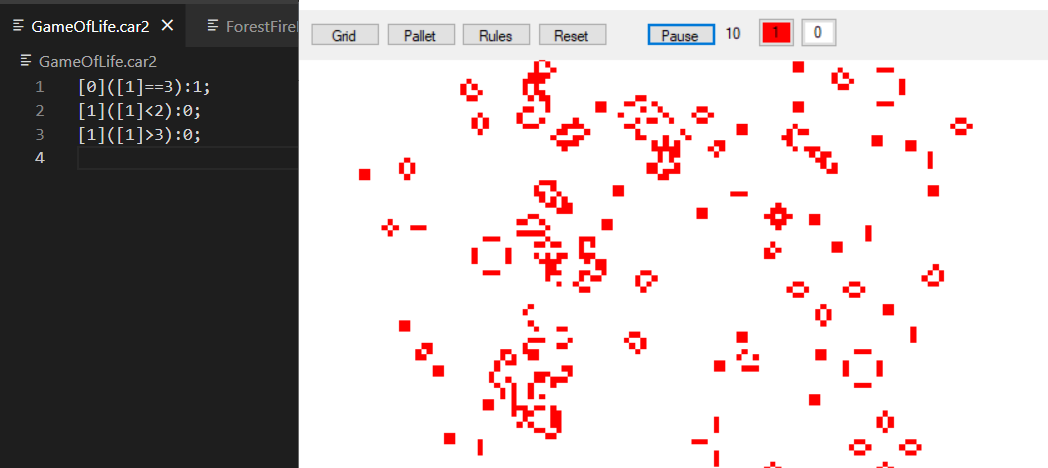


Рис. 12. “Игра Жизнь”

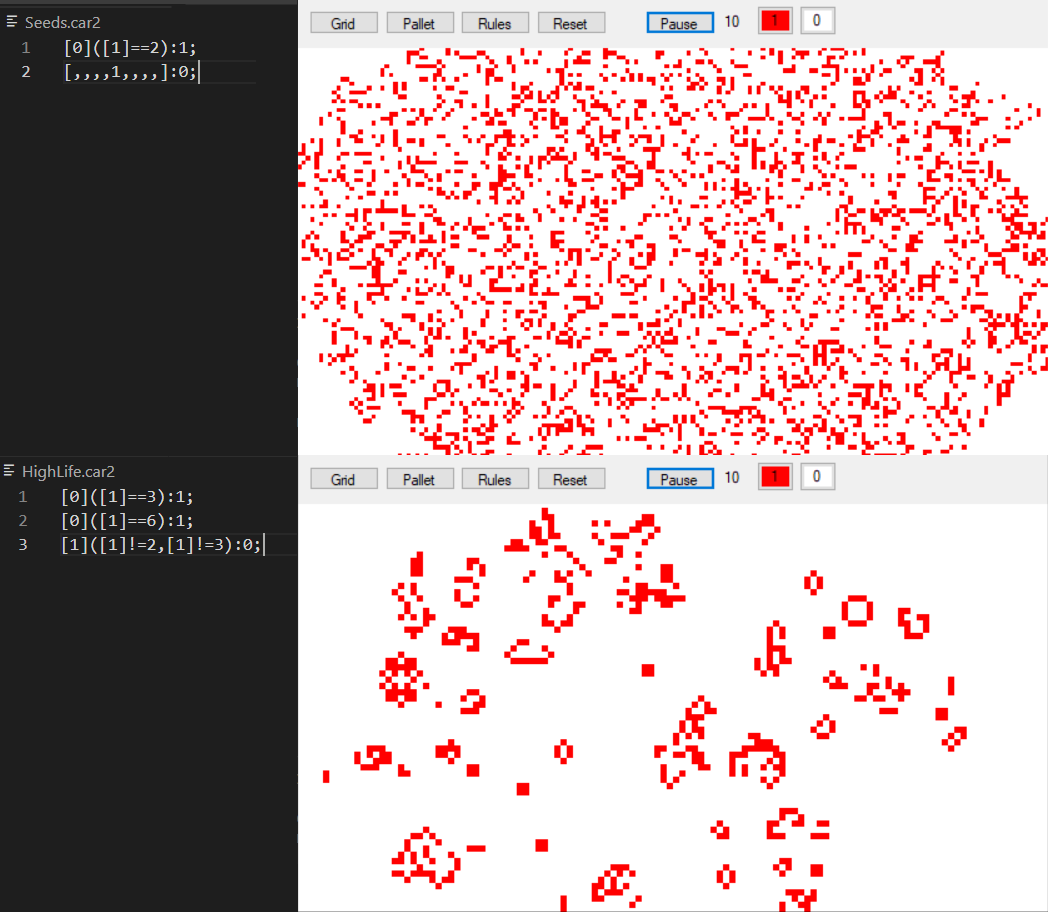


Рис. 13. “Семена” и “*HighLife*”

Возможности ЯОП также позволяют описать многие правила клеточных автоматов из серии “Поколения” ­– правила аналогичны игре жизнь, однако “клетки” не умирают а стареют в течении определенного количества поколений и только после этого умирают; На ЯОП можно описать правила этих автоматов количество поколений за которое “стареет” клетка не превышает 98 (см. рис. 14), однако с увеличением количества поколений старения клетки увеличивается и количество инструкций необходимых для описания правил на ЯОП, что создает достаточно высокую нагрузку на устройство, на котором выполняется программа в связи с реализацией.

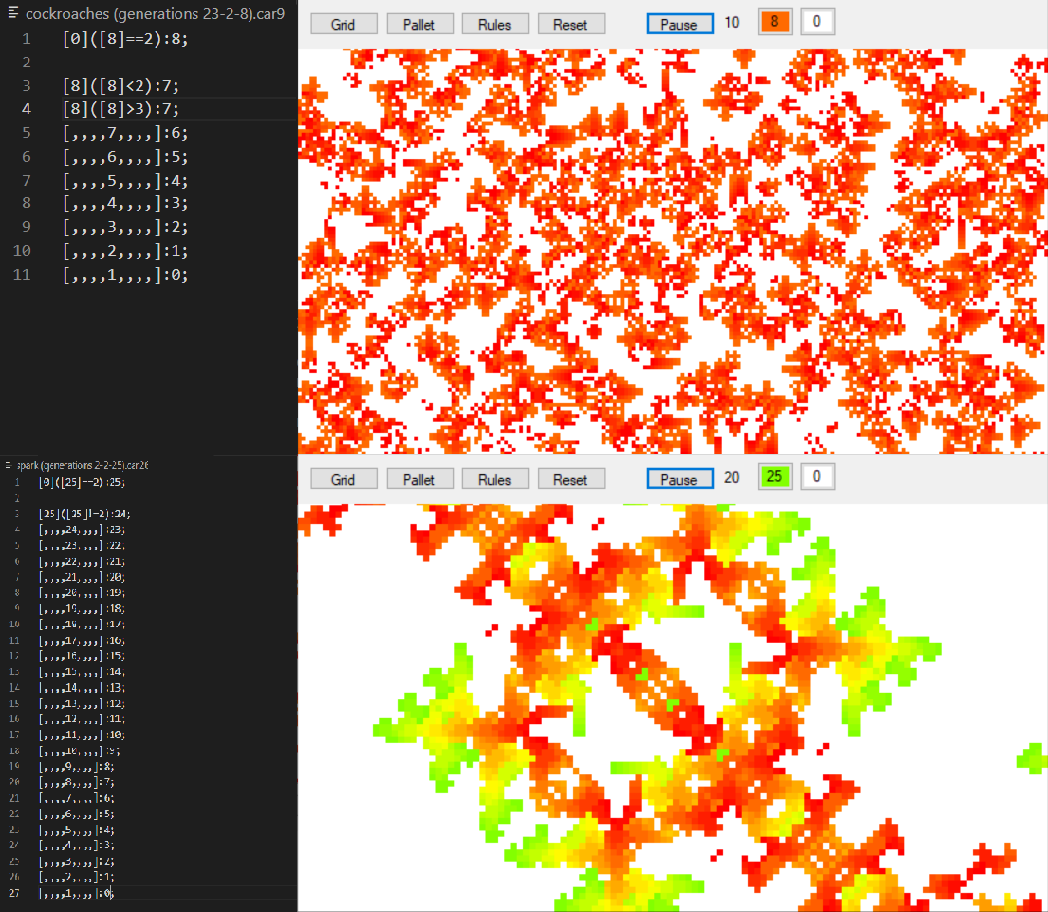


Рис. 14. Клеточные автоматы серии “Поколения”, правила 23/2/8 и 2/2/25

*Wireworld* – клеточный автомат клетки в котором имеют 4 состояния, чаще всего используемый для симуляции электронных логических элементов, по аналогии с “Игрой жизнь” может быть описан на ЯОП используя только инструкции сравнения (см. рис. 15).

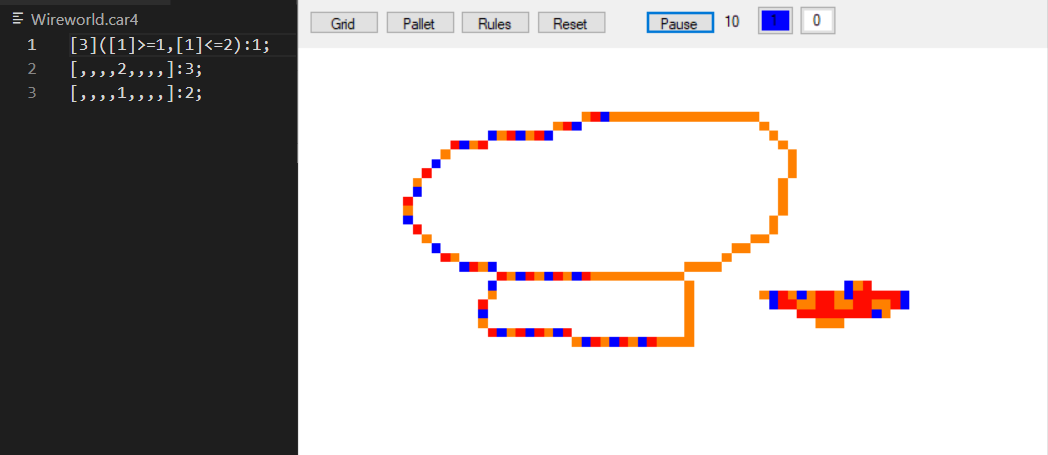


Рис. 15. “Wireworld”

Муравей Лэнгтона из всех примеров клеточных автоматов является не тривиальным по отношению к другим так как появляется сущность “муравей”, который имеет координаты и направление взгляда, который перемещается по сетке, и меняет свое направление в зависимости от типа клетки на которой он стоит, однако для того что бы описать правила данного клеточного автомата на ЯОП мы можем добавить дополнительные 8 состояний каждое из которых будет представлять из себя клетку одного из двух состояний на которой находится муравей, смотрящий в одну из четырех сторон (см. рис. 16).

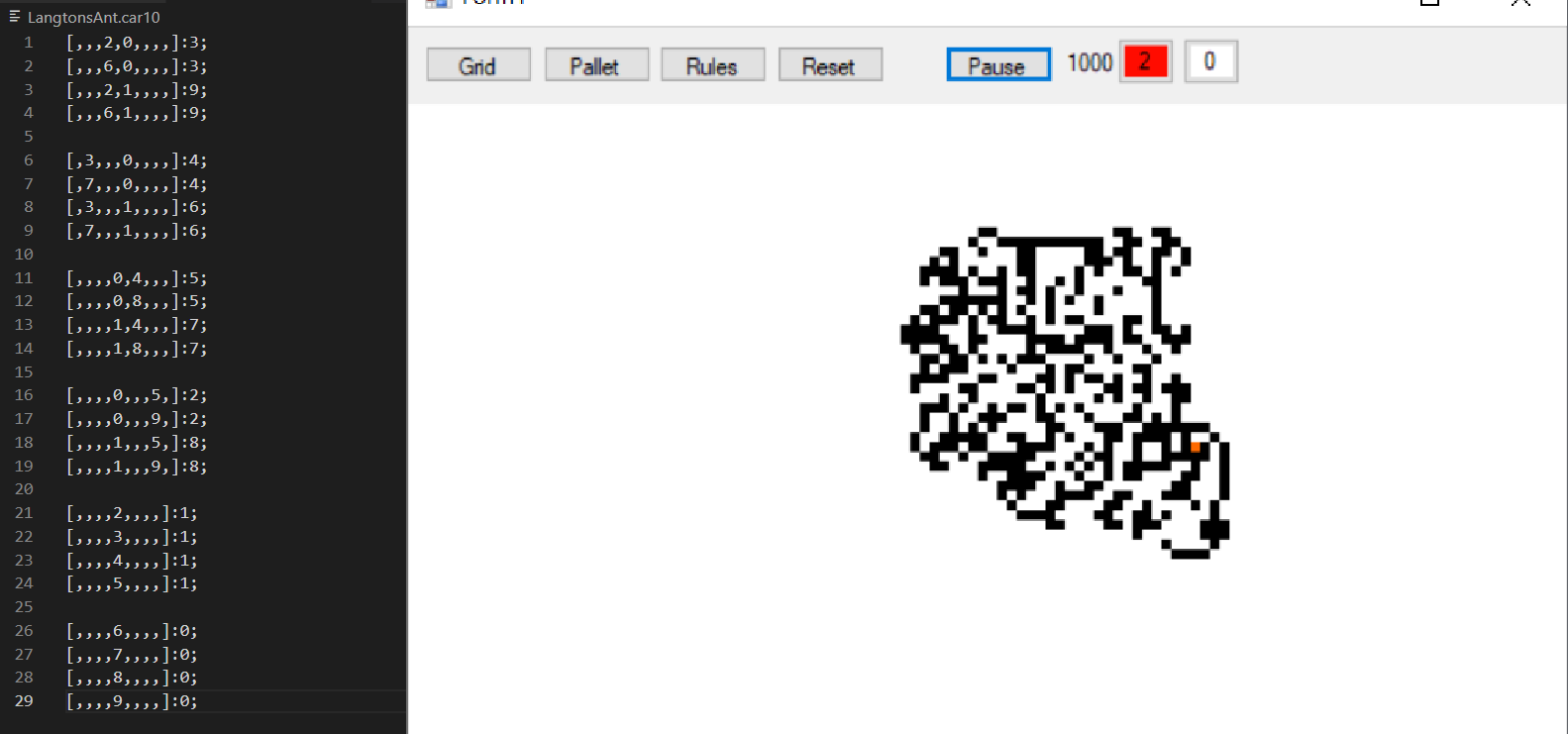


Рис. 16. “Муравей Лэнгтона”

# Заключение

В итоге работы была разработана прикладная программа для симуляции работы клеточных автоматов, предназначенная для помощи в решении задач в сфере биологии, физики, криптографии, социологии и других сферах научной и прикладной деятельности.

# Список литературы

1. Wolfram MathWorld – CellularAutomata [Электронный ресурс] – Электрон. текст. – 2021 – Режим доступа: <https://mathworld.wolfram.com/topics/CellularAutomata.html>, свободный (дата обращения: 13.01.2022)
2. Тоффоли Т., Марголус Н. Машины клеточных автоматов. – М.: Мир, 1991 – С. 22–80.
3. R. Miles C# Yellow Book [Электронный ресурс] – Электрон. Текст. – 2019 – Режим доступа: <https://static1.squarespace.com/static/5019271be4b0807297e8f404/t/5df9306ec60881645ea57ced/1576611956760/CSharp+Book+2019+Refresh.pdf>, – свободный (дата обращения: 13.01.2022)

# Приложение

Студент: Левчук Роман Эдуардович

Группа: 1255

Тема: Реализация программы для моделирования клеточных автома-тов с произвольными правилами. (C#)

|  |  |  |  |
| --- | --- | --- | --- |
| № | Этап выполнения | Предполагаемая дата выполнения | Реальная дата выполнения |
| 1 | Выбор, согласование и утверждение темы курсовой работы | 23.11.2021 | 23.11.2021 |
| 2 | Составление и сдача преподавателю плана-графика выполнения курсовой работы | 25.03.2022 | 25.03.2022 |
| 3 | Сбор информации по теме курсовой работы (если необходимо) |  |  |
| 4 | Разработка начального варианта интерфейса проекта | 20.02.2022 | 13.01.2022 |
| 5 | Согласование его с преподавателем | 8.04.2022 | 9.04.2022 |
| 6 | Доработка и согласование с преподавателем окончательного интерфейса проекта | 26.05.2022 | 30.05.2022 |
| 7 | Написание кода проекта (*возможно разбиение на два этапа*) | 29.04.2022 | 6.05.2022 |
| 8 | Создание и утверждение текстового отчета по курсовой работе | 5.05.2022 | 26.05.2022 |
| 9 | Защита курсовой работы | 26.05.2022 | 1.06.2022 |