МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»

Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой ЭВМ
_____Г.И. Радченко
«___»_____ 2021 г.

РАЗРАБОТКА ЭМУЛЯТОРОВ МАШИНЫ ТЬЮРИНГА И МАШИНЫ ПОСТА

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

Руководитель работы,
к.п.н., доцент каф. ЭВМ
_____Ю.Г. Плаксина
«___»_____2021 г.

Автор работы,
студент группы КЭ-405
_____А.А. Артёмов
«___»_____2021 г.

Нормоконтролёр,
ст. преп. каф. ЭВМ
_____ С.В. Сяськов
«___»_____2021 г.

Челябинск-2021

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

УТВЕРЖДАЮ
Заведующий кафедрой ЭВМ
_____ Г.И. Радченко
«___»_____2021 г.

ЗАДАНИЕ

на выпускную квалификационную работу бакалавра

студенту группы КЭ-405
Артёмову Антону Александровичу
обучающемуся по направлению
09.03.01. «Информатика и вычислительная техника»

Тема работы: «Разработка эмуляторов машины Тьюринга и машины Поста» утверждена приказом ректора Южно-Уральского государственного университета от 26 апреля 2021 г. №714-13/12 (приложение №73)

1. **Срок сдачи студентом законченной работы:** 1 июня 2021 г.

2. **Исходные данные к работе:** статьи, книги, техническое задание.
   - Ершов, С.С. Элементы теории алгоритмов / С.С Ершов. – Челябинск: Издательский центр ЮУрГУ, 2009. — 64 с.
   - Пильщиков, В.Н. Машина Тьюринга и алгоритмы Маркова. Решение задач / В.Н. Пильщиков, В.Г. Абрамов, А.А. Вылиток, И.В. Горячая. – М.: Наука, 2006. – 567 с.
   - Успенский, В. А. Машина Поста / В. А. Успенский. — М.: Наука, 1988. — 96 с.

3. **Перечень подлежащих разработке вопросов:**
   - рассмотрение существующих программных решений, реализующих эмуляторы машины Тьюринга и Поста;
   - рассмотрение оптимальных средств разработки;
   - проектирование и разработка собственного программного комплекса эмуляторов машины Тьюринга и Поста;
   - оценка работоспособности разработанного программного комплекса.

4. **Дата выдачи задания:** 1 декабря 2020 г.

Руководитель работы _____/*Ю.Г. Плаксина*/

Студент _____/*А.А. Артёмов* /

КАЛЕНДАРНЫЙ ПЛАН

Этап	Срок сдачи	Подпись руководителя
Введение и обзор литературы	01.03.2021	
Разработка модели, проектирование	01.04.2021	
Реализация системы	01.05.2021	
Тестирование, отладка, эксперименты	15.05.2021	
Компоновка текста работы и сдача на нормоконтроль	20.05.2021	
Подготовка презентации и доклада	24.05.2021	

Руководитель работы _____/*Ю.Г. Плаксина*/

Студент _____/*А.А. Артёмов* /

Аннотация

Артёмов А.А. Разработка эмуляторов машины Тьюринга и машины Поста, машины Поста. – Челябинск, ЮУрГУ, ЭВМ; 2021, 52 с.,17 ил., библиогр. список – 13 наим.

В рамках выпускной квалификационной работы был проведён анализ существующих аналогов, а также схожих по функционалу систем. Результатом анализа были выявлены недостатки этих систем, с целью их ликвидации в выпускной квалификационной работе, а также достоинства этих систем, с целью включения их в выпускную квалификационную работу. Рассмотрены основные технологии, применяющиеся в разработке схожих систем, и были выбраны наиболее подходящие для данного проекта.

После анализа были произведены проектирование, разработка и тестирование эмуляторов машины Тьюринга и машины Поста.

Результатом работы являются эмуляторы, реализующие заявленный функционал.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность

Машина Тьюринга и машина Поста являются одними из первых аппаратов в исследованиях по теории алгоритмов. Известен тезис Тьюринга: «Всякий алгоритм может быть задан посредством тьюринговой функциональной схемы и реализован в соответствующей машине Тьюринга». Многие пытаются опровергнуть эту гипотезу – ищут пример алгоритма, который не может быть реализован с помощью тьюринговой функциональной схемы. Однако на данный момент опровергнуть гипотезу никому не удалось. Машина Поста схожа с машиной Тьюринга в этом отношении.

Идея о создании эмуляторов для данных абстрактных машин не нова, и это вполне закономерно, так как данные абстрактные машины являются, возможно, самыми наглядными пособиями для изучения теории алгоритмов. И, действительно, эмуляторов для данных абстрактных машин существует несчетное количество.

Данные абстрактные машины являются одним из элементов технологии обучения студентов дисциплине «Математическая логика и теория алгоритмов» на кафедре ЭВМ. Знание принципов работы данных машин необходимо для глубокого понимания материала, преподаваемого в курсе «Математическая логика и теория алгоритмов». Для лучшего понимания принципов работы данных аналитических машин нужны эмуляторы. Эмуляторы, полностью удовлетворяющие требованиям кафедры.

Именно для решения данной проблемы отсутствия системы с необходимыми требованиями и обусловлена актуальность выпускной квалификационной работы.

Цели работы

Выпускная квалификационная работа посвящена разработке графических приложений, реализующих эмуляторы машин Поста и Тьюринга. В дальнейшем данные приложения планируется использовать в образовательных целях, для наглядной демонстрации работы абстрактных машин Поста и Тьюринга.

Задачи работы:

— изучить работу заданных абстрактных алгоритмов;

— найти существующие на данный момент аналоги и схожие по функционалу системы;

— провести анализ с целью выявления преимуществ и недостатков этих систем, сделать выводы;

— составить техническое задание и эскизный проект системы;

— выполнить программную реализацию проекта;

— определить методы тестирования и провести тестирование разработанного приложения на корректность выполнения задач, удобство интерфейса.

1.АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. ОБЗОР АНАЛОГОВ

Сейчас существует множество эмуляторов абстрактных машин Поста, Тьюринга и алгоритмов Маркова. Все они имеют как достоинства, так и недостатки, например, неудобный интерфейс, отсутствие проверок внешнего алфавита и т.д. Разберем более подробно различные примеры, выявив их достоинства и недостатки. Достоинства по возможности реализуем в проекте, недостатки же нужно будет исключить.

1. Тренажер-эмулятор машины Тьюринга разработанный преподавателем К. Ю. Поляковым[6]. Тренажер-эмулятор машины Тьюринга позволяет настраивать внешний алфавит и вводить символы алфавита в ячейки ленты, расположенные в верхней части окна. В таблице, в нижней левой части окна вводится программа. В нижней правой части окна находится окно для комментариев.

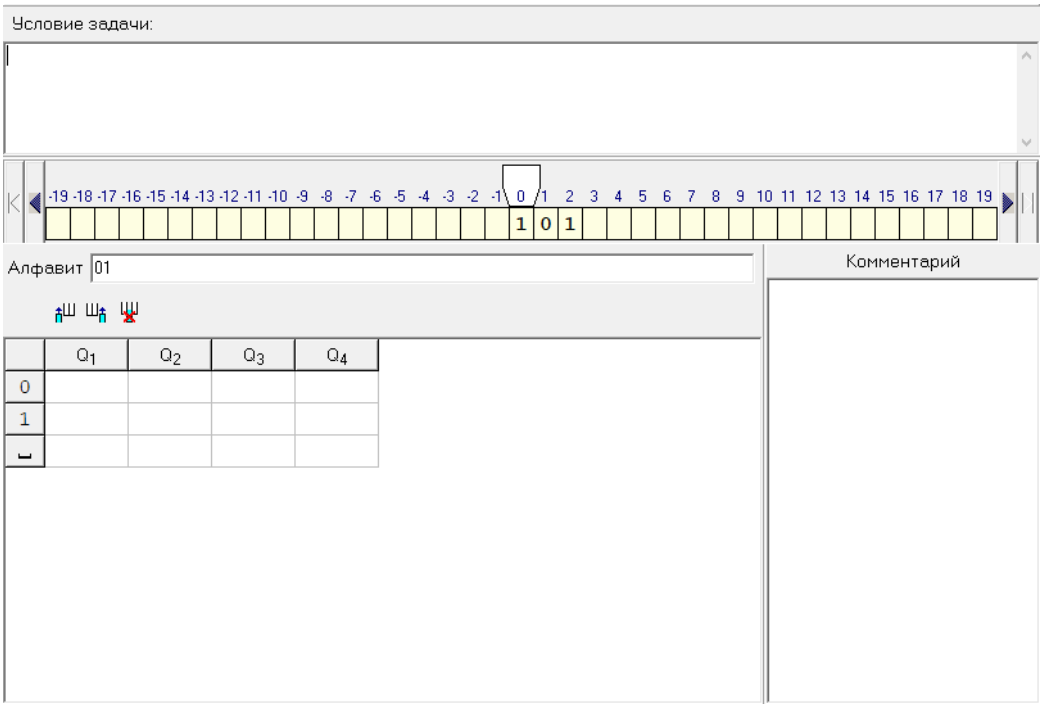

Рисунок 1.1 — интерфейс тренажера «Машина Тьюринга»

Достоинства:

— удобный, интуитивно понятный интерфейс;

— наличие большого количества полей для комментариев.

Недостатки:

— отсутствует проверка символов внешнего алфавита. Программа никак не проверяет символы, введенные пользователем.

— не классическая запись команд программы в таблице. Не является как таковым недостатком.

2. Тренажер-эмулятор машины Поста за авторством К. Ю. Полякова[5]. Имеет ленту с ячейками для меток (наличие метки в ячейке — «1», отсутствие — «0») в верхней части окна под полем для формулировки задачи. В нижней части интерфейса имеет таблицу для записи команд, переходов между строками, а также столбец комментариев.

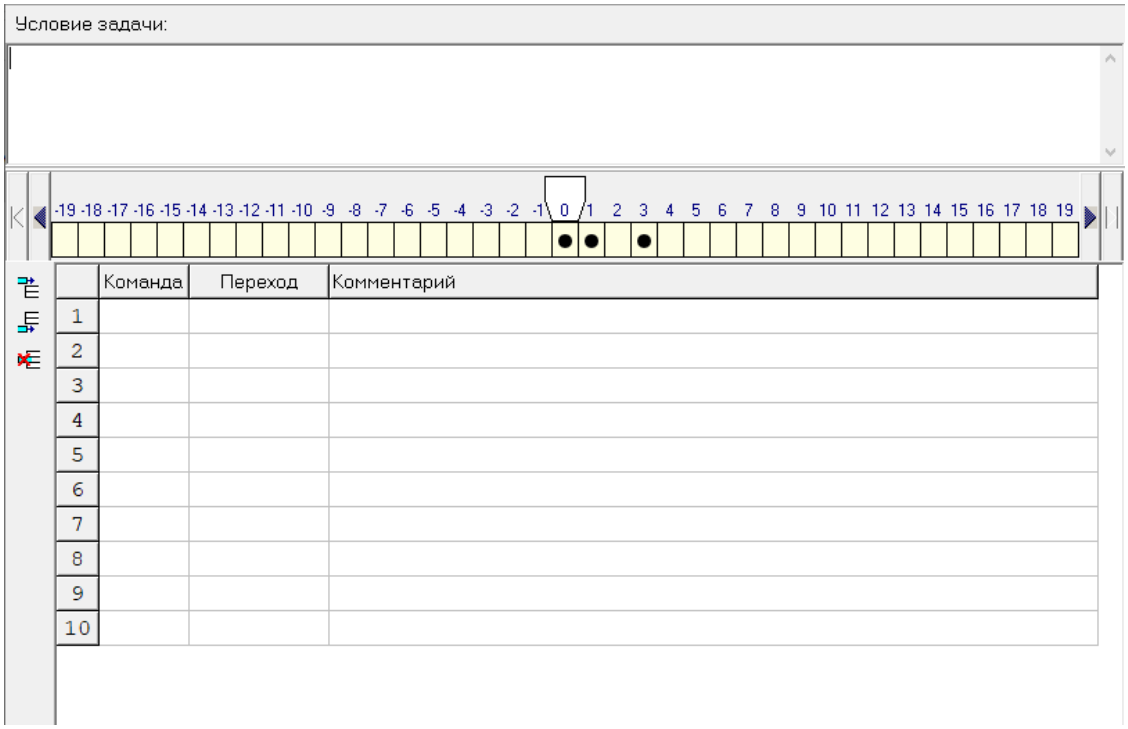

Рисунок 1.2 — интерфейс тренажера «Машина Поста»

Достоинства:

— удобный, интуитивно понятный интерфейс;

— наличие большого количества полей для комментариев;

— привычный формат записи программы.

Недостатки:

— отсутствие режима троичной машины Поста. Не является как таковым недостатком.

Данный эмулятор является примером наиболее подходящей под требования системы.

3. Тренажер машины Тьюринга, разработчик Грачев А[7].

Внешний вид представлен на рисунке 1.

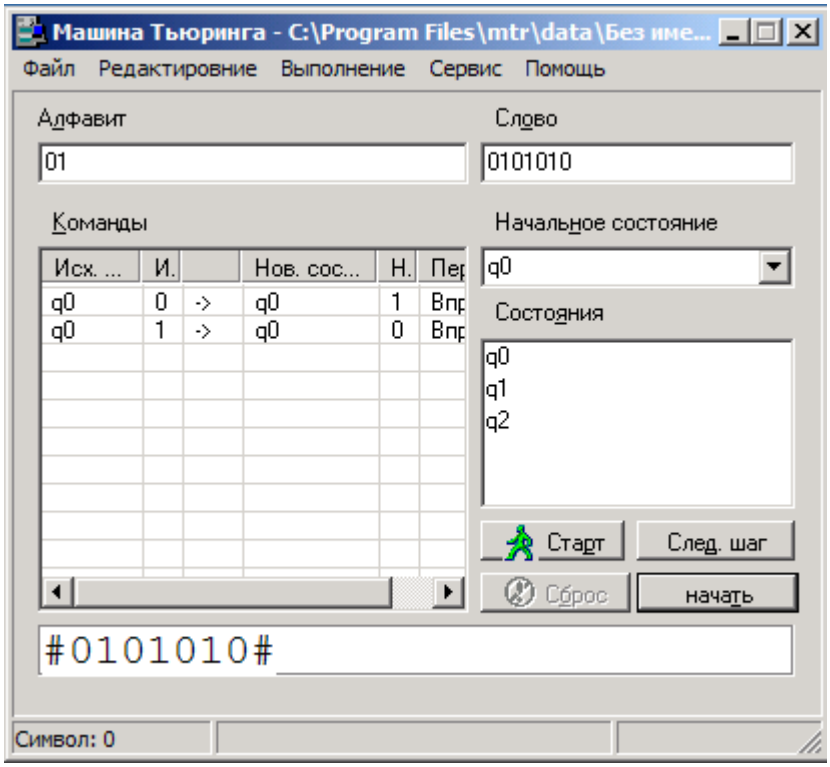

Рисунок 1.3 — внешний вид программы-тренажера машины Тьюринга

Имеет ленту без ячеек внизу, в форме обычной строки, поле для внешнего алфавита, поле для состояний, таблицу для команд. Введение команд и состояний осуществляется с помощью клика на соответствующем поле правой кнопкой мыши с последующим введением данных в появившееся окно. После введения команд, состояний и алфавита необходимо воспользоваться кнопками «старт», «начать» и т.д.

Достоинства:

— проверка внешнего алфавита;

— каноничный формат записи программы.

Недостатки:

— не самый удобный маленький интерфейс. Не является как таковым недостатком;

— отсутствует автоматический режим выполнения программы, есть только пошаговый;

— лента выполнена в виде простой строки.

Данный эмулятор является примером подходящей под требования системы, однако имеет ряд недостатков:

— интерфейс слишком мал и неудобен, нет возможности увеличить его;

— отсутствие автоматического режима работы.

4. Для машины Поста существует эмулятор, написанный для Android, и, распространяемый через Google Play[8].

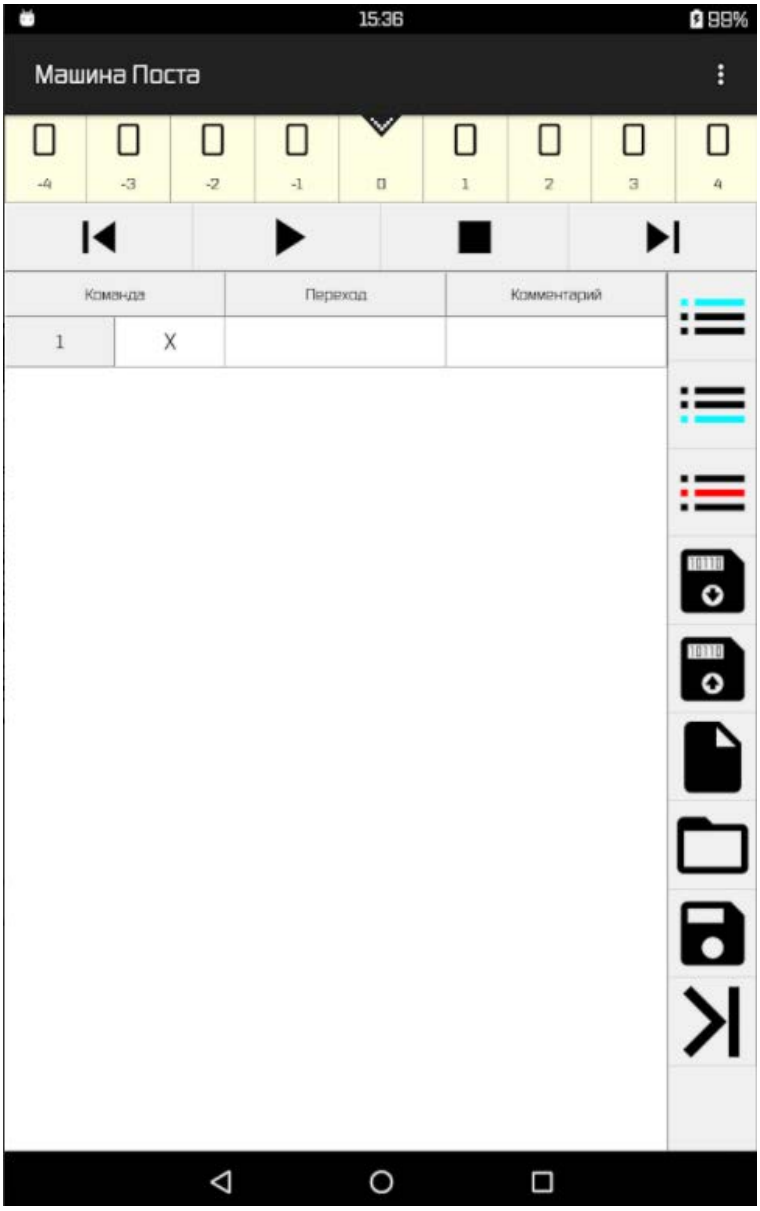

Рисунок 1.4 — внешний вид программы-тренажера машины Поста для Android

Интерфейс данного эмулятора схож с интерфейсом эмулятора от К.Ю. Полякова за исключением полей для комментариев. Кнопки «сохранить» и «загрузить» находятся сбоку.

Достоинства:

— удобный, интуитивно понятный интерфейс;

— наличие большого количества полей для комментариев;

— привычный формат записи программы.

Недостатки:

— отсутствие режима троичной машины Тьюринга;

— эмулятор создан для Android, а не для Windows.

Хороший эмулятор, не соответствующий необходимым требованиям. Сравнительный анализ приведен в таблице 1.

Таблица 1 — сравнительный анализ существующих аналогов

Эмуляторы машины Поста				
	Наличие троичной машины Поста	Возможность работы на Windows	Классический формат записи программы	Удобный интерфейс
Эмулятор для Android	нет	нет	да	да
Эмулятор К.Ю. Полякова	нет	да	да	да
Эмуляторы машины Тьюринга				
	Наличие автоматического режима работы	Проверка внешнего алфавита	Классический формат записи программы	Удобный интерфейс
Эмулятор К.Ю. Полякова	да	нет	нет	да

Продолжение таблицы

Эмуляторы машины Тьюринга				
	Наличие автоматического режима работы	Проверка внешнего алфавита	Классический формат записи программы	Удобный интерфейс
Эмулятор А. Грачева	нет	да	да	нет

Вывод:

— разрабатываемый эмулятор машины Поста должен иметь достоинства аналогичных систем, при этом имея режим работы троичной машины Поста;

— разрабатываемый эмулятор машины Тьюринга должен иметь достоинства как эмулятора А. Грачева, так и эмулятора К.Ю. Полякова.

1.3. АНАЛИЗ ОСНОВНЫХ ТЕХНОЛОГИЧЕСКИХ РЕШЕНИЙ

1.3.1 Выбор языка программирования и фреймворков

Для разработки графического приложения приложения для Windows можно использовать следующие фреймворки и интерфейсы программирования приложений: WinForms, WPF, Qt. Проанализируем недостатки перечисленных решений.

Недостатки WPF[9]:

— зависимость от драйверов и дополнительного ПО (DirectX);

— малое (по сравнению с WinForms) количество готовых элементов управления;

— необходимость изучать и использовать для создания приложения язык XAML.

Недостатки Qt[11]:

— высокий порог вхождения;

— большой вес готовых приложений.

Для создания графического приложения выбран WinForms, так он является популярным интерфейсом программирования приложений, созданным специально для написания приложения для Windows. Windows Forms — это технология интеллектуальных клиентов (интеллектуальный клиент — это приложение с полнофункциональным графическим интерфейсом, простое в развертывании и обновлении) для .NET Framework. Она представляет собой набор управляемых библиотек, упрощающих выполнение стандартных задач, таких как чтение из файловой системы и запись в нее.

Преимущества WinForms[10]:

— большое количество элементов управления;

— широкие возможности;

— низкий порог вхождения;

— специально для Windows.

В качестве языка программирования для разработки был выбран C#.

C# — современный объектно-ориентированный и типобезопасный язык программирования. C# позволяет разработчикам создавать множество типов безопасных и надежных приложений, работающих в экосистеме .NET[4].

Преимущества C#:

— большое количество библиотек;

— возможность использовать инструментарий .NET;

— бесплатность ряда инструментариев.

1.3.2 Выбор среды разработки

IDE (или интегрированная среда разработки) — это программа, предназначенная для разработки программного обеспечения. Как следует из названия, IDE объединяет несколько инструментов, специально предназначенных для разработки. Эти инструменты обычно включают редактор, предназначенный для работы с кодом (например, подсветка синтаксиса и автодополнение); инструменты сборки, выполнения и отладки; и определённую форму системы управления версиями.

Большинство IDE поддерживают множество языков программирования и имеют много функций, из-за чего могут быть большими, занимать много времени для загрузки и установки и требуют глубоких знаний для правильного использования.

С другой стороны, есть редакторы кода, которые представляют собой текстовый редактор с подсветкой синтаксиса и возможностями форматирования кода. Большинство хороших редакторов кода могут выполнять код и использовать отладчик, а лучшие даже могут взаимодействовать с системами управления версиями. По сравнению с IDE, хороший редактор кода, как правило, легковесней и быстрее, но зачастую ценой меньшей функциональности.

В качестве среды разработки была выбрана Microsoft Visual Studio 2019.

2.ОСНОВНЫЕ ТРЕБОВАНИЯ

2.1. ОСНОВНЫЕ ТРЕБОВАНИЯ К ФУНКЦИОНАЛУ СИСТЕМЫ

Для эмулятора машины Тьюринга:

— возможность выбора внешнего языка;

— проверка внешнего языка;

— запись программы в форме «$q_1 \mathrm{OC}_1 \rightarrow q_2 \mathrm{OC}_2 \Delta$, где Δ — право, лево, стереть.»

— система должна сохранять и загружать задачи из файлов;

— наличие графического интерфейса.

Для эмулятора машины Поста:

— классический формат записи команд программы;

— наличие функции троичной машины Поста;

— система должна сохранять и загружать задачи из файлов;

— наличие графического интерфейса.

2.2. ОСНОВНЫЕ НЕФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ

Приложение должно придерживаться определенного стиля:

— хорошо читаемый шрифт;

— отсутствие сильно выделяющихся цветов;

— система должна адаптироваться к различным экранам;

— система должна работать на следующих ОС: Windows;

— система должна работать на компьютерах со слабыми техническими характеристиками.

3. ПРОЕКТИРОВАНИЕ

3.1. ОСНОВНЫЕ СВЕДЕНИЯ

Алгоритмическая система Поста (Машина Поста; Post)[1] реализуется с использованием бесконечной в обе стороны ленты Л с клетками-ячейками, в которых пусто (λ) или находится метка (V, 1), и универсальной (считывающей и записывающей) головкой Г. Будем условно считать, лента неподвижна, а головка может смещаться влево или вправо на одну позицию (клетку, ячейку) единовременно (рис. 3.1).

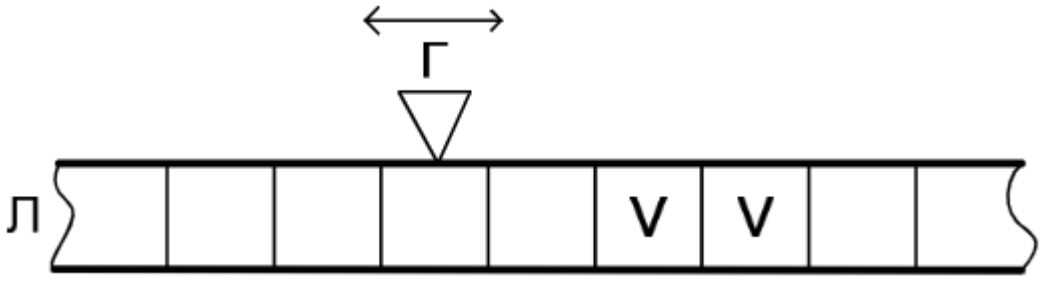

Рисунок 3.1 — конструкция машины Поста

Машина Поста использует, таким образом, двоичный алфавит Ап = {0, 1}, или Ап = {λ, 1}. В системе команд машины Поста всего 6 команд:

— запись единицы (:= 1);

— запись нуля (:= 0);

— сдвиг головки влево на одну позицию (L1, ←);

— сдвиг головки вправо на одну позицию (R1, →);

— условный переход (УП) по нулю (если под головкой, действительно, нуль, т.е. пустая клетка, переход осуществляется по адресу A1, в противном случае — по адресу A2); эта команда, таким образом, двухадресная (например, N: ? A1, A2; здесь N — номер команды в программе – 1, 2, ...); безусловный переход (БП), очевидно, реализуется как условный с одинаковыми адресами перехода (N: ? A, A);

— останов (Ост.); эта команда – безадресная (N: Ост.).

Отсюда видно, машина Поста имеет простейшую систему образующих (:= 0, := 1; =0).

Завершение программы в машине Поста может быть троякое:

— результативный останов (и хорошо, если на ленте останутся также исходные данные);

— сбойный останов (по ошибке);

— зацикливание (бесконечный цикл), это следствие ошибочности алгоритма.

Расширение машины Поста связано с появлением множества отличий-дополнений в машине Тьюринга[1], являющейся представителем действительно канонической алгоритмической системы. Внешний алфавит машины Тьюринга (алфавит символов) содержит всего I+1 символ:

$$A = \{a0, a1, ..., aI\}.$$

Здесь a0 — пустой символ (пробел).

В машине Тьюринга имеется еще один алфавит — внутренний (алфавит состояний). Он содержит J+1 элемент:

$$Q = \{q0, q1, ... , qj, qz\}.$$

Здесь q1 — начальное, а qZ — конечное состояния.

В машине Тьюринга (МТ) под головкой на ленте размещается, например, символ ai1 в состоянии МТ qj1. Тогда рассматривается пара (ai1, qj1) и реализуется команда

$$qj1\ ai1 \rightarrow qj2\ ai2\ D$$

D — это действие, перемещение головки влево (L), вправо (R) или перемещения нет (E), т.е. D ∈ {L, R, E}. Позднее выяснится, что от случая D = E можно освободиться. В машине Поста реализуется программа (схема алгоритма и текст программы), в машине же Тьюринга можно увидеть черты автоматного подхода (граф состояний и переходов ГСП и таблица состояний и переходов ТСП). ТСП содержит I+1 строку и J столбцов (рис. 3.2).

ТСП	...	q_{j1}	...
...	...	...	...
a_{i1}	...	$q_{j2}a_{i2}D$	...
...	...	...	...

Рисунок 3.2 — таблица состояний и переходов

Машинное слово (α) МТ содержит, в общем случае 2 части: α1 и α2.

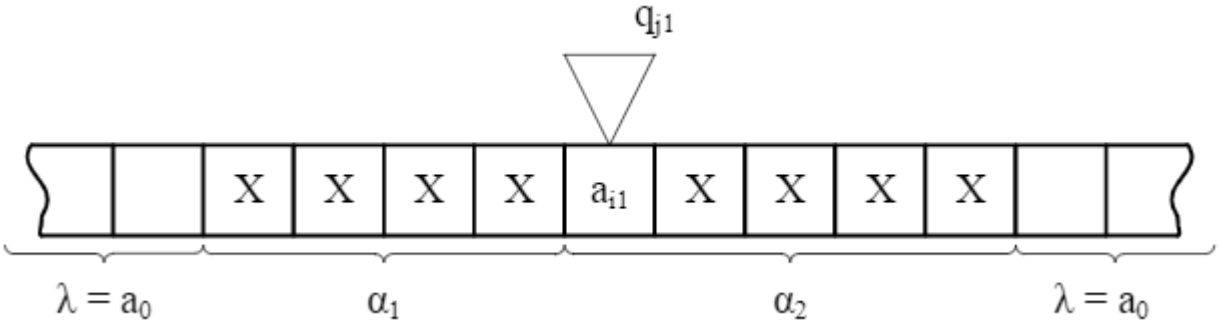

Рисунок 3.3 — конфигурация машины Тьюринга

3.2. ПРОЕКТИРОВАНИЕ СТРУКТУРЫ ГРАФИЧЕСКОГО ИНТЕРФЕЙСА

Интерфейс графического приложения состоит из 1 основного окна. На этом окне пользователем производится выбор между машиной Поста и машиной Тьюринга. В зависимости от выбора пользователя виджеты на окне изменятся (одни исчезнут, другие появятся. Также на главном окне помимо кнопок «Пост» и «Тьюринг» и виджетов, соответствующих текущему режиму работы (в форме Поста — поле для команд, в форме Тьюринга — таблица состояний и т.д.) должны находиться кнопки «Сохранить» и «Загрузить» для сохранения текущей конфигурации машины в файл собственного разрешения и для загрузки ранее сохраненной конфигурации машины соответственно. При нажатии на кнопки «Сохранить» и «Загрузить» должно открываться окно для поиска места сохранения/загрузки файла.

Макет графического интерфейса сайта представлен на рисунках 3.4 – 3.5.

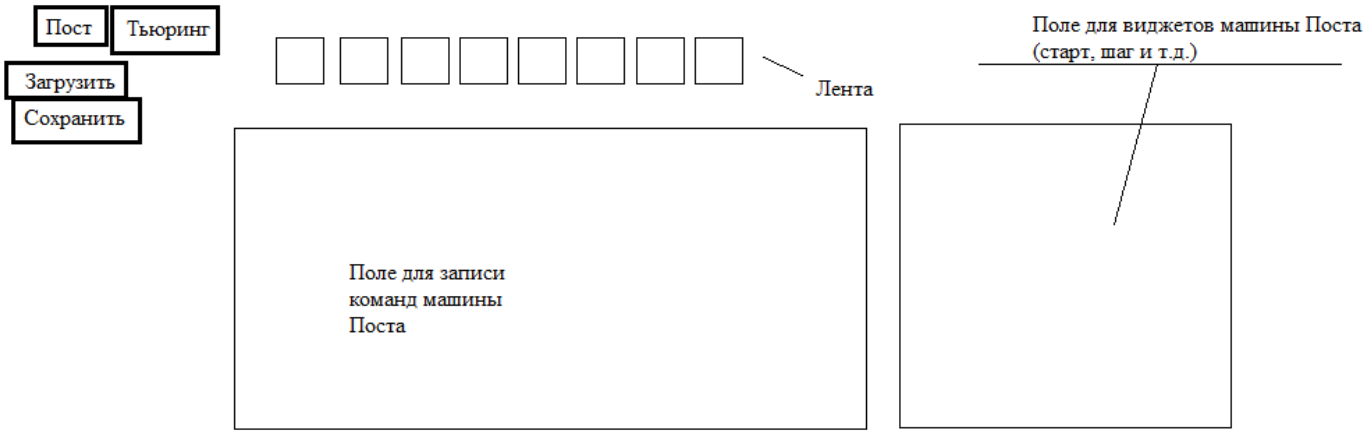

Рисунок 3.4 — планируемый интерфейс машины Поста

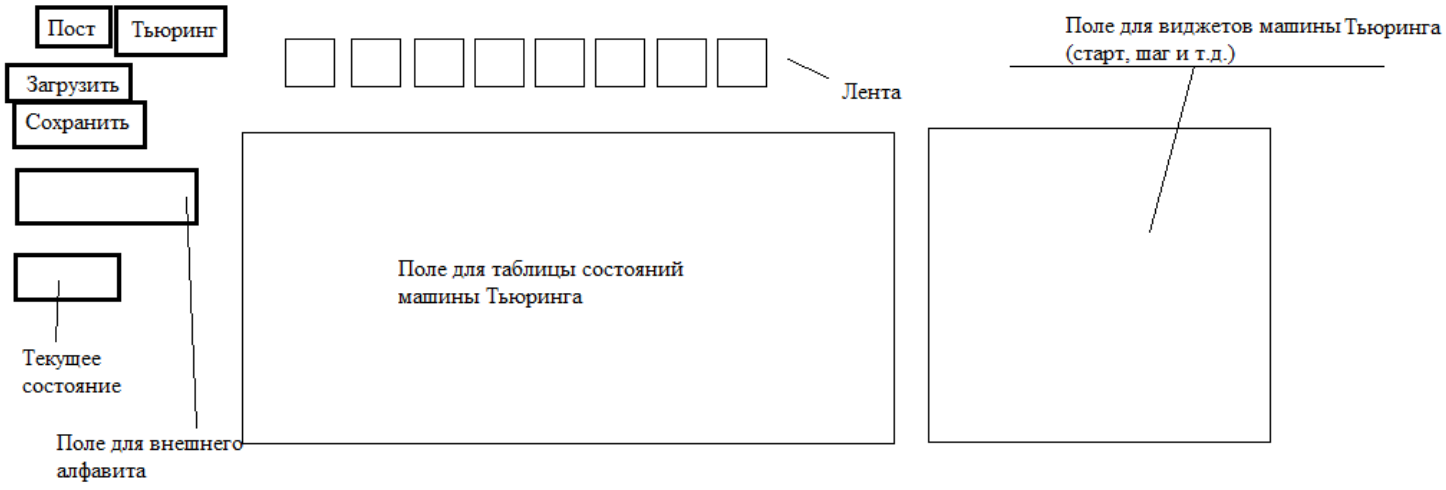

Рисунок 3.5 — планируемый интерфейс машины Тьюринга

4.РЕАЛИЗАЦИЯ

4.1 РЕАЛИЗАЦИЯ ГРАФИЧЕСКОГО ИНТЕРФЕЙСА

В главном окне приложения расположены две кнопки, отвечающие за режим работы приложения. Кнопка «Пост» отвечает за режим работы приложения в форме машины Поста. Кнопка «Тьюринг» отвечает за режим работы приложения в форме машины Тьюринга. В режиме работы машины Поста в главном окне приложения расположены виджеты соответствующие режиму работы машины Поста, а именно:

— кнопки «Загрузить» и «Сохранить», отвечающие за функции загрузки конфигурации машины из файла и сохранения конфигурации машины в файл соответственно (при нажатии открывается окно обзора файлов);

— чекбокс «Троичная», отвечающий за режим работы эмулятора в форме троичной машины Поста при постановке в него галочки (для возвращения в исходное состояние галочка снимается);

— пронумерованные чекбоксы образующие ленту машины Поста (галочка в чекбоксе обозначает «1», квадратик, при режиме работы троичной машины Поста, обозначает «x», пустой чекбокс обозначает «0»);

— бегунок со стрелками для перематывания ленты;

— большое текстовое поле — список команд, в котором находятся команды программы;

— кнопка «Шаг» для выполнения одного шага работы эмулятора;

— чекбокс «До останова» для автоматического режима работы эмулятора (для запуска — ставится галочка, для отключения — убирается);

— малое текстовое поле, в которое вводятся команды, которые затем можно добавить в большое текстовое поле с командами программы с помощью кнопок «Заменить», «Добавить»;

— кнопка «Заменить» для замены выбранной команды в списке команд на другую из малого текстового поля;

— кнопки «Вверх» и «Вниз» (кнопки с соответствующими стрелками) для выбора команд в списке команд;

— кнопка «Добавить» для добавления введенной команды в список команд;

— кнопка «Удалить» для удаления выбранной команды;

Подсветка цветом циан обозначает, что на данной ячейке стоит считывающая головка.

Исходный код виджетов на листинге 1 приложения.

Реализация интерфейса машины Поста представлена на рисунках 4.1 — 4.5.

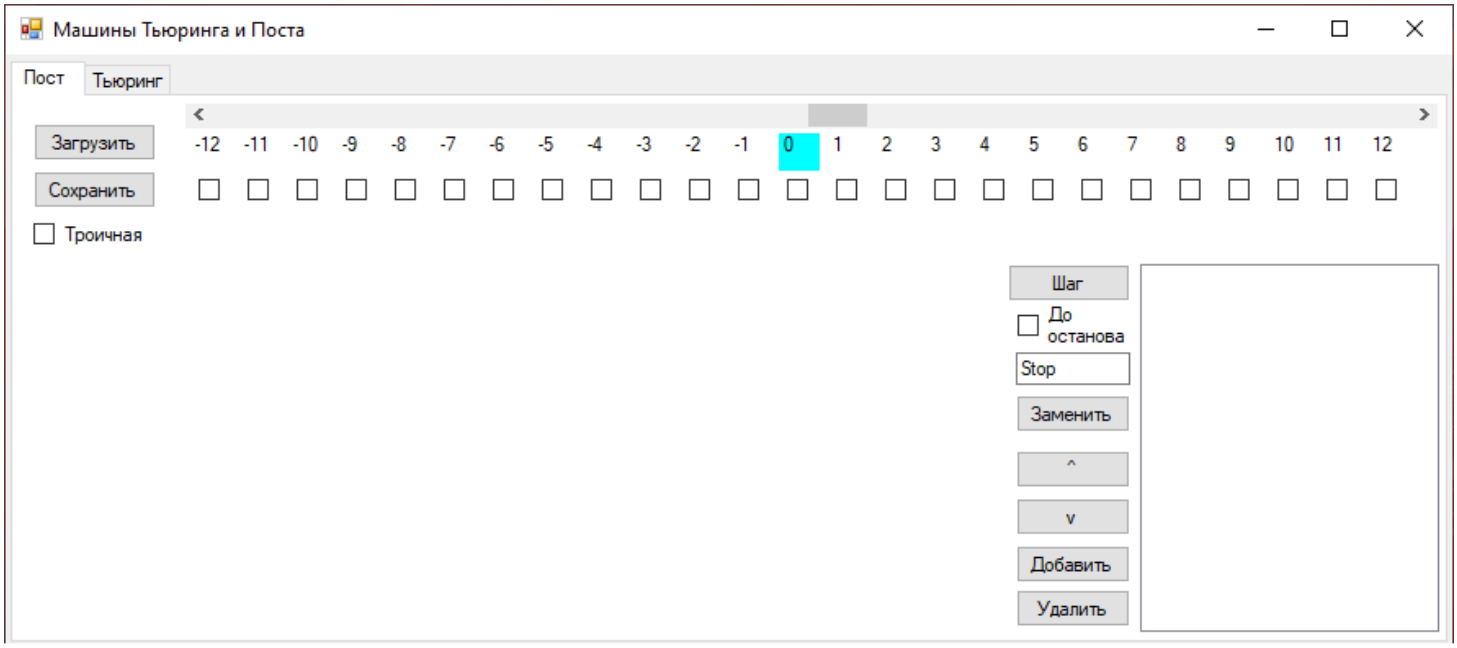

Рисунок 4.1 – реализация интерфейса машины Поста

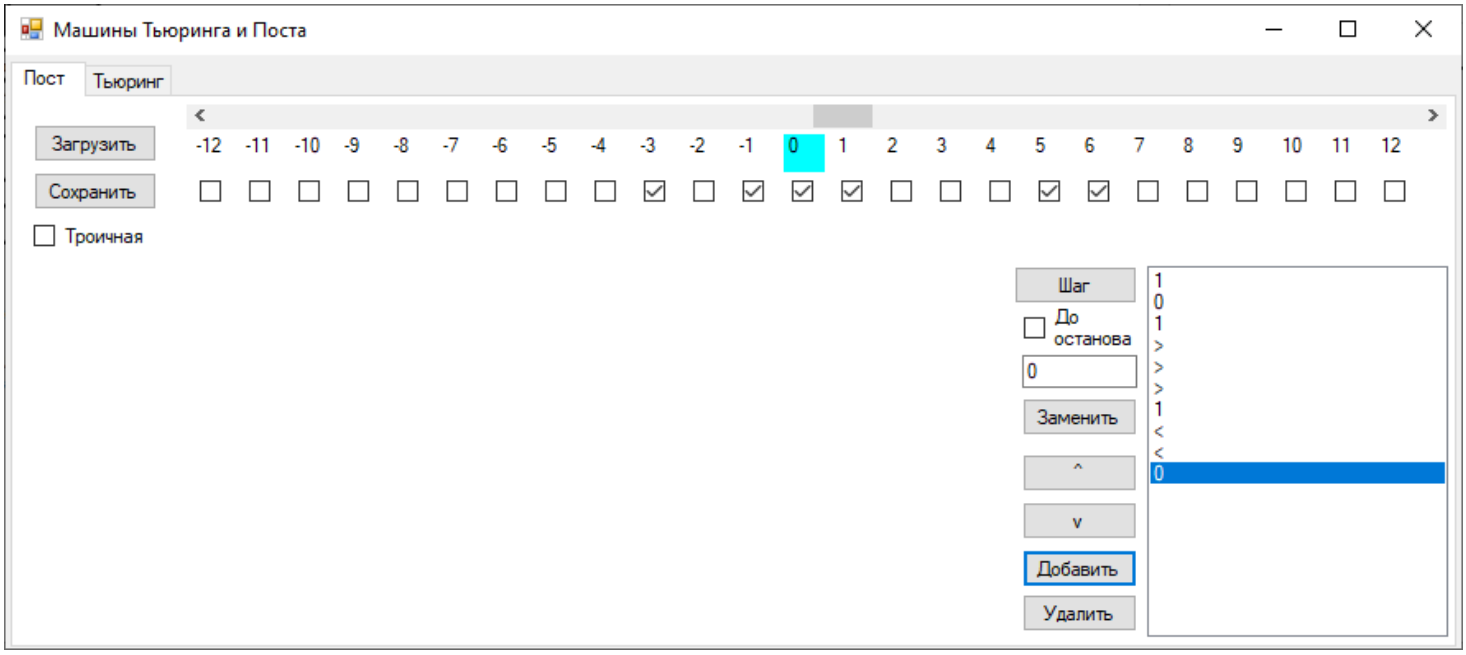

Рисунок 4.2 – машина Поста с введенными командами

Рисунок 4.3 – троичная машина Поста

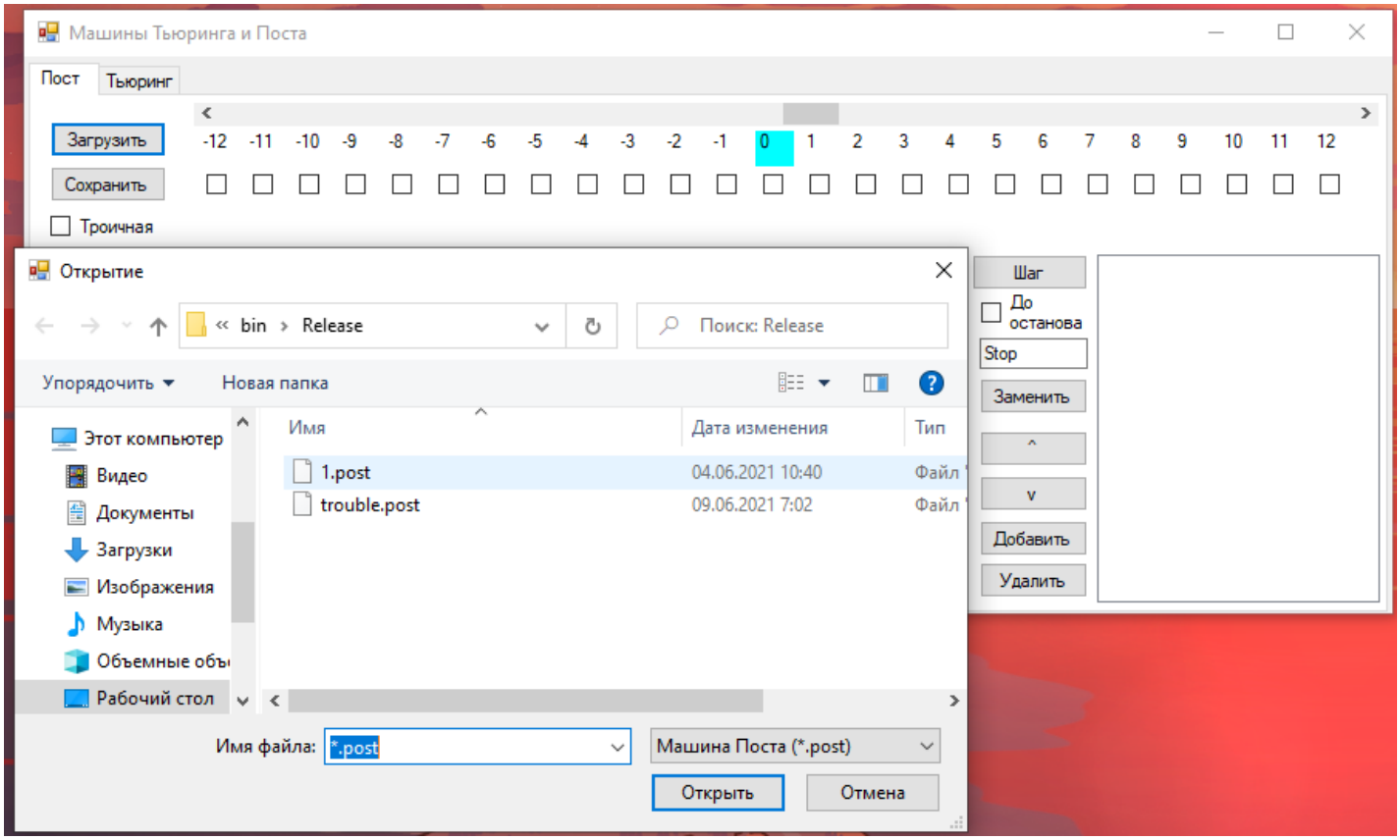

Рисунок 4.4 – загрузка конфигурации

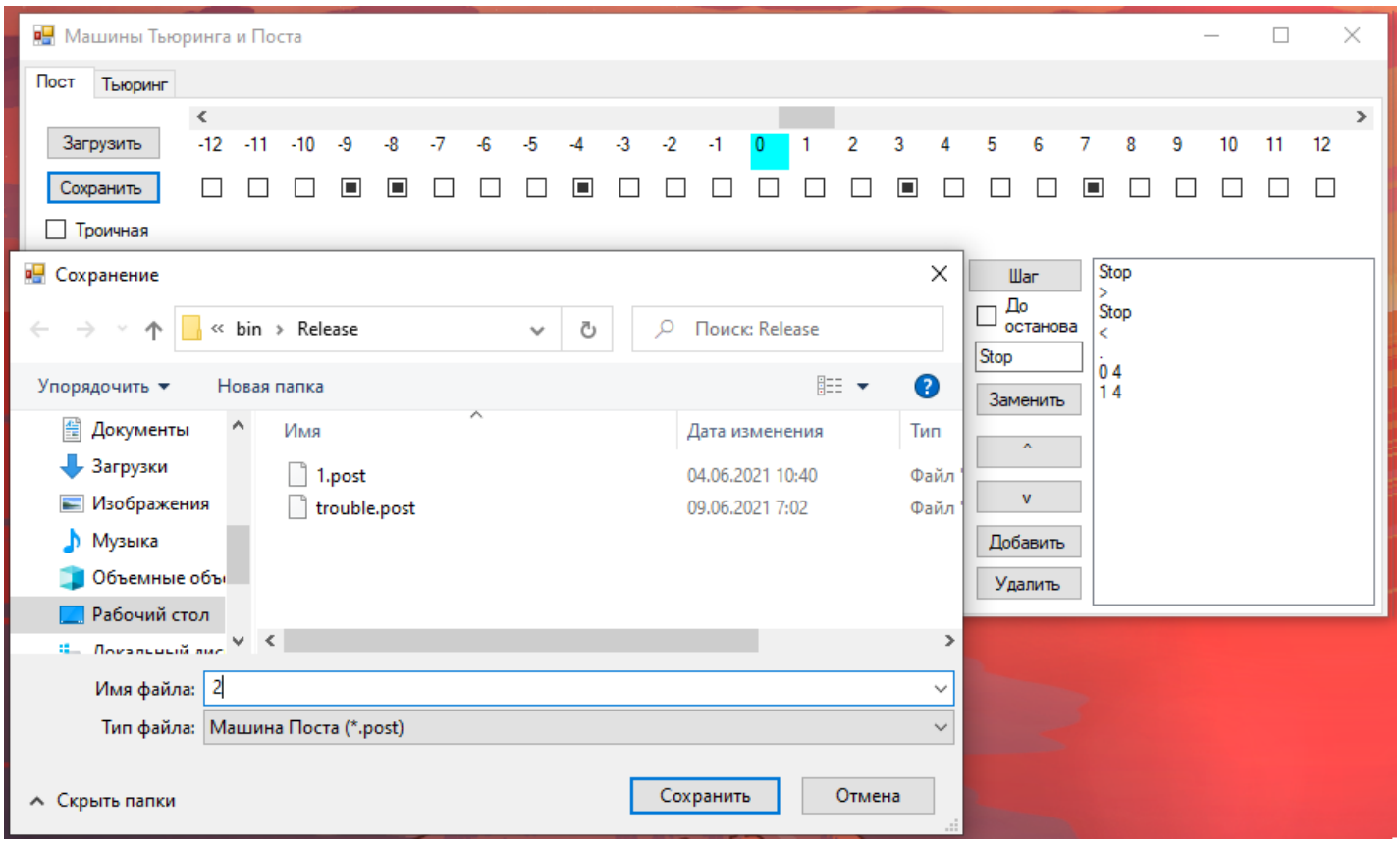

Рисунок 4.4 – сохранение конфигурации

В режиме работы машины Тьюринга в главном окне приложения расположены виджеты соответствующие режиму работы машины Тьюринга, а именно:

— кнопки «Загрузить» и «Сохранить», отвечающие за функции загрузки конфигурации машины из файла и сохранения конфигурации машины в файл соответственно (при нажатии открывается окно обзора файлов);

— текстовое поле «Алфавит», заполняется автоматически символами, взятыми из списка правил;

— кнопка со стрелочками «Число состояний» для указания количества состояний машины;

— пронумерованные текстовые поля, образующие ленту машины Тьюринга (с самого начала заполнены символами «_» обозначающими пробел);

— бегунок со стрелками для перематывания ленты;

— комбобокс «Текущее состояние» для выставления текущего состояния машины в данный момент;

— большое серое поле — таблица состояний и переходов, в которое добавляются строки и столбцы таблицы состояний и переходов машины;

— большое текстовое поле «Список правил», в котором находятся правила программы;

— кнопка «Шаг» для выполнения одного шага работы эмулятора;

— чекбокс «До останова» для автоматического режима работы эмулятора (для запуска — ставится галочка, для отключения — убирается);

— малое текстовое поле, в которое вводятся правила, которые затем можно добавить в «Список правил» и таблицу состояний и переходов с помощью кнопок «Заменить», «Добавить»;

— кнопка «Заменить» для замены выбранного правила в «Списке правил» на другое из «Списка правил»;

— кнопки «Влево» (кнопка с соответствующей стрелочкой) для перенесения правила из «Списка правил» в таблицу состояний и переходов;

— кнопка «Добавить» для добавления введенного правила в «Список правил» и таблицу;

— кнопка «Удалить» для удаления выбранного правила.

Подсветка цветом циан обозначает, что на данной ячейке стоит считывающая головка.

Исходный код виджетов на листинге 1 приложения.

Реализация интерфейса машины Поста представлена на рисунках 4.6 — 4.7.

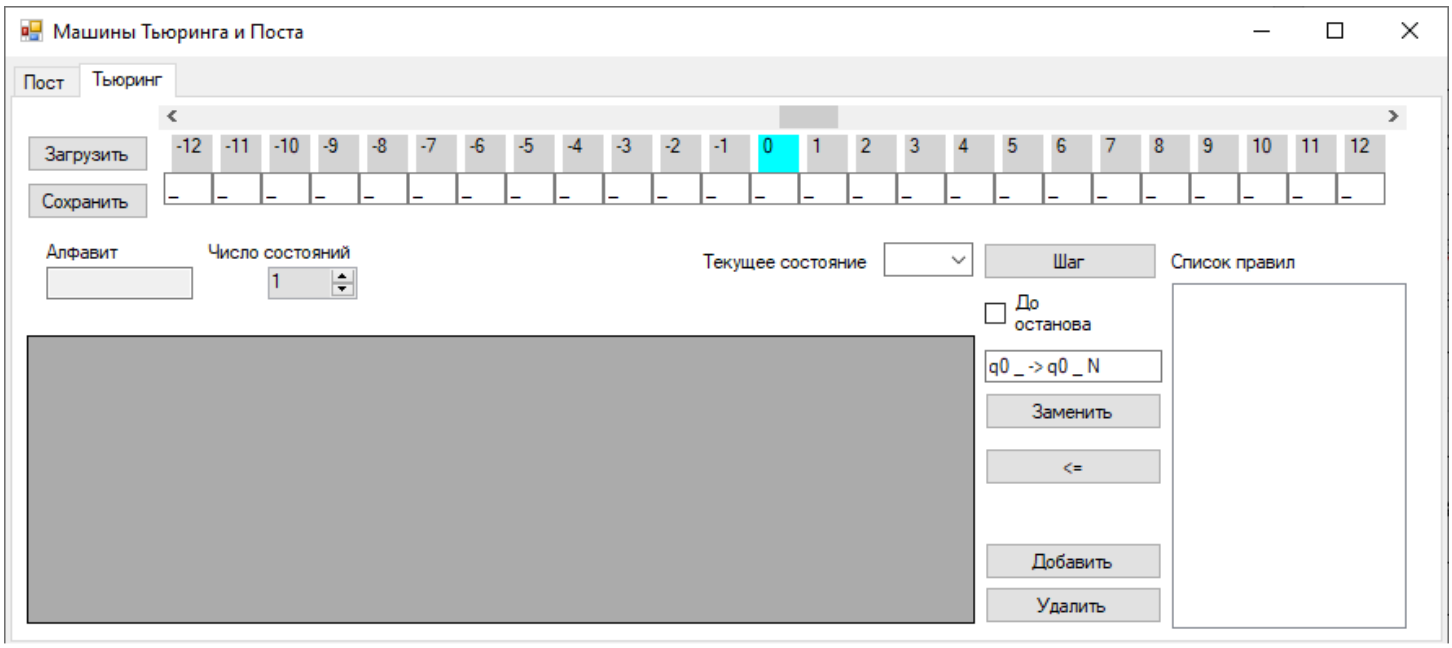

Рисунок 4.6 – реализация интерфейса машины Тьюринга

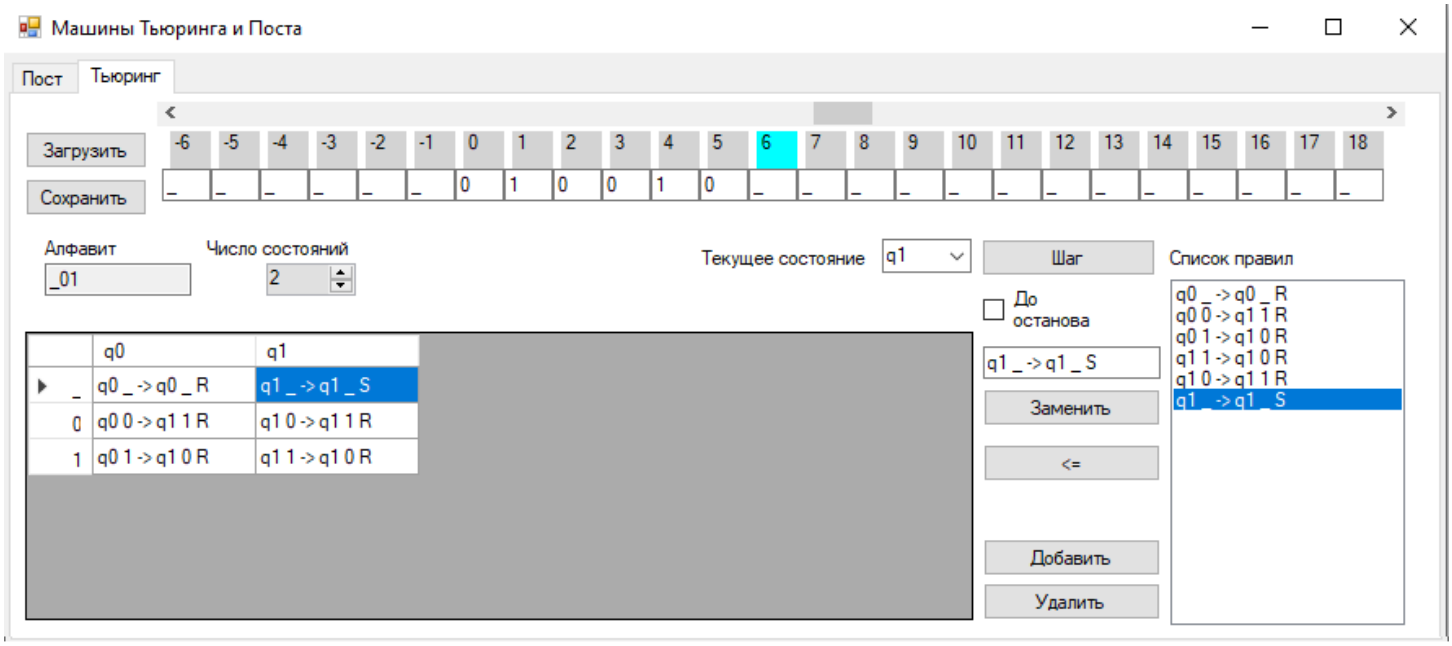

Рисунок 4.6 – машины Тьюринга с введенными правилами перехода

4.2. РЕАЛИЗАЦИЯ ЗАГРУЗКИ И СОХРАНЕНИЯ КОНФИГУРАЦИЙ

Конфигурации машин Тьюринга и Поста сохраняются в текстовые файлы с расширением «turing» и «post» соответственно.

На файл «post» сохраняются список команд, затем, через одну пустую строку, состояние всех чекбоксов ленты (двести штук).

```
Stop
>
Stop
<
.
0 4
1 4

-100    Unchecked
-99     Unchecked
-98     Unchecked
-97     Unchecked
-96     Unchecked
-95     Unchecked
-94     Unchecked
-93     Unchecked
```

Рисунок 4.7 – содержимое файла «.post»

На файл «turing» сохраняются внешний алфавит конфигурации, количество состояний, номера и содержимое всех ячеек ленты, чье содержимое не является символом «_», затем, через пустую строку список правил перехода машины.

```
_01
2
0       0
1       1
2       0
3       0
4       1
5       0
-1      _
6       _
-3      _
-2      _

q0 _ -> q0 _ R
q0 0 -> q1 1 R
q0 1 -> q1 0 R
q1 1 -> q1 0 R
q1 0 -> q1 1 R
q1 _ -> q1 _ S
```

Рисунок 4.7 – содержимое файла «.turing»

5. ТЕСТИРОВАНИЕ СИСТЕМЫ

Для тестирования веб-сайта были выбраны следующие виды тестирования:

— приемочное пользовательское тестирование;

— функциональное тестирование;

— интеграционное тестирование.

5.1. МЕТОДОЛОГИЯ ТЕСТИРОВАНИЯ

Юзабилити тестирование

Юзабилити-тестирование — это метод оценки интерфейса со стороны удобства и эффективности его использования. Чтобы получить его, надо привлечь представителей целевой аудитории программного продукта[12].

Функциональное тестирование

Функциональное тестирование — это тестирование ПО в целях проверки реализуемости функциональных требований, то есть способности ПО в определённых условиях решать задачи, нужные пользователям. Функциональные требования определяют, что именно делает ПО, какие задачи оно решает[13].

Интеграционное тестирование

Интеграционное тестирование представляет собой полную проверку готового программного продукта с целью выявления ошибок, возникающих в процессе интеграции программных компонентов.

5.2. ПРОВЕДЕНИЕ ПРОЦЕДУРЫ ТЕСТИРОВАНИЯ

Юзабилити тестирование

Юзабилити тестирование осуществлялось рядом независимых пользователей, с различным уровнем компьютерной грамотности.

Интерфейс приложения простой, и в ходе тестирования у респондентов не возникло сложностей при работе с сайтом.

Интеграционное тестирование

Для проведения интеграционного тестирования приложение было запущено эмулированных операционных системах:

— Windows 7;

— Windows 10.

На различных операционных системах эмулятор работал в соответствии с заявленным функционалом (рис. 5.1).

Функциональное тестирование

Основной функцией графического приложения является эмуляция работы машин Поста и Тьюринга. С помощью эмулятора было запущено более 20 конфигураций машин Поста и Тьюринга. В ходе тестирования эмулятор работал корректно, в соответствии с заявленным функционалом. Было выявлено отсутствие функциональных ошибок.

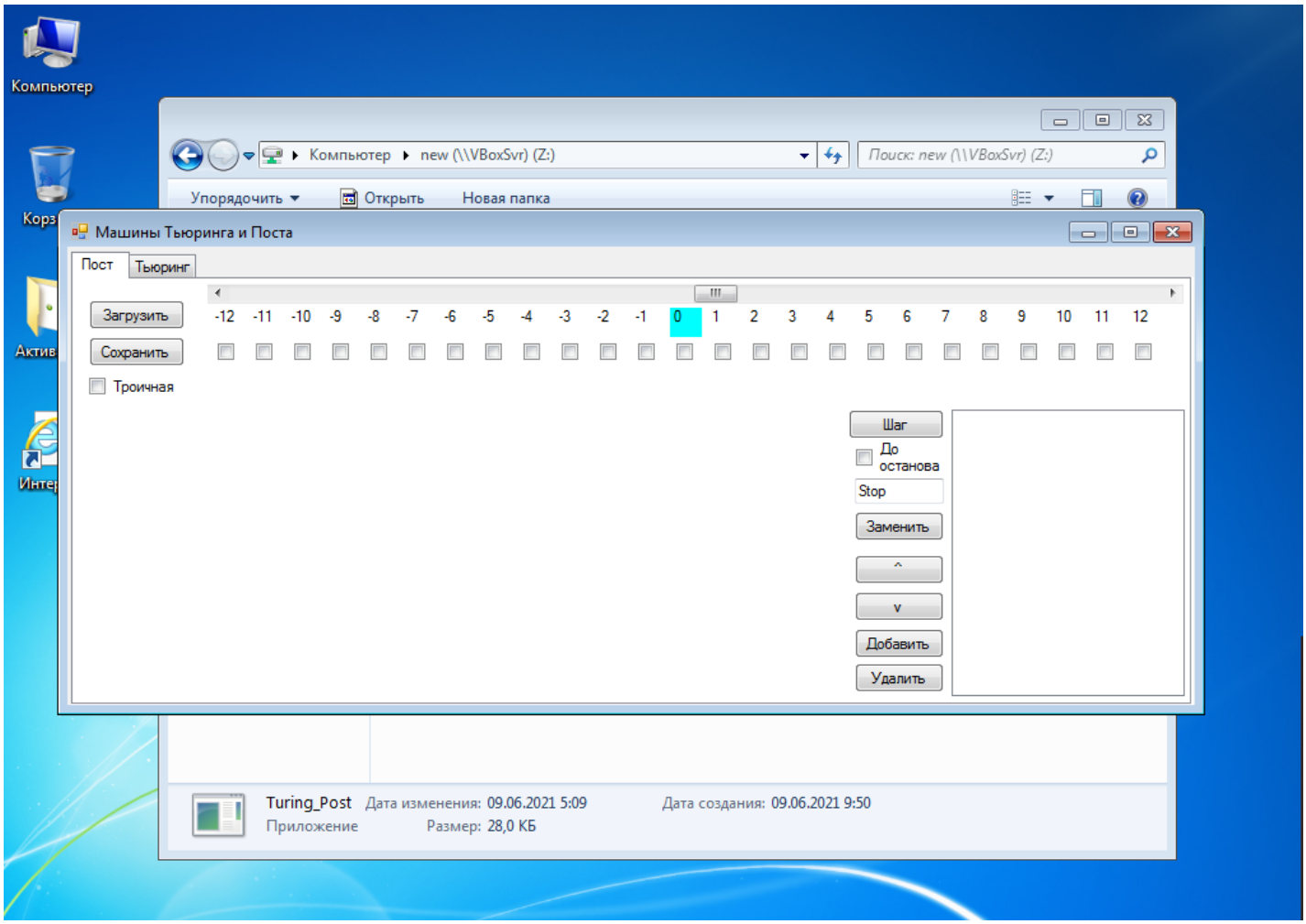

Рисунок 5.1 – эмулятор, запущенный на Windows 7

ЗАКЛЮЧЕНИЕ

В рамках выпускной квалификационной работы было спроектировано и реализовано графическое приложение «Turing_Post», которое является эмулятором машины Поста и машины Тьюринга.

Для достижения поставленной цели были решены следующие задачи:

— выполнен анализ предметной области;

— выполнено проектирование графического приложения;

— реализация графического приложения с учётом необходимых функциональных требований;

— проведено тестирование веб-приложения с использованием юзабилити-тестирования, интеграционного тестирования, функционального тестирования.

В итоге можно отметить, что приложение работает корректно.

Реализованное графическое приложение «Turing_Post» может быть полезно определенной категории пользователей, преимущественно студентам, изучающим дисциплину «Математическая логика и теория алгоритмов», и преподавателям, преподающим дисциплину «Математическая логика и теория алгоритмов».

В дальнейшем планируется сделать приложение кроссплатформенным, переделав приложение в фреймворке Qt. Также планируется сделать построение графов для машины Тьюринга для ещё более наглядной визуализации работы алгоритма.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Ершов, С.С. Элементы теории алгоритмов / С.С Ершов. – Челябинск: Издательский центр ЮУрГУ, 2009. — 64 с.
2. Пильщиков, В.Н. Машина Тьюринга и алгоритмы Маркова. Решение задач / В.Н. Пильщиков, В.Г. Абрамов, А.А. Вылиток, И.В. Горячая. – М.: Наука, 2006. – 567 с.
3. Успенский, В. А. Машина Поста / В. А. Успенский. — М.: Наука, 1988. — 96 с.
4. Обзор языка C# [Электронный ресурс]. – Режим доступа: https://docs.microsoft.com/ru-ru/dotnet/csharp/tour-of-csharp (Дата обращения: 15.05.2021).
5. Машина поста [Электронный ресурс]. – Режим доступа: https://kpolyakov.spb.ru/prog/post.htm (Дата обращения: 15.05.2021).
6. Машина Тьюринга [Электронный ресурс]. – Режим доступа: https://kpolyakov.spb.ru/prog/turing.htm (Дата обращения: 15.05.2021).
7. Тренажер машины Тьюринга [Электронный ресурс]. – Режим доступа: https://chursinvb.ucoz.ru/load/5-1-0-20 (Дата обращения: 15.05.2021).
8. Машина Поста [Электронный ресурс]. – Режим доступа: https://play.google.com/store/apps/details?id=ru.anbroid.postmachine&hl=ru&gl=US (Дата обращения: 15.05.2021).
9. WPF vs. WinForms [Электронный ресурс]. – Режим доступа: https://www.wpf-tutorial.com/ru/2/несколько-слов-о-wpf/wpf-vs-winforms/ (Дата обращения 15.05.2021).
10. Руководство по классическим приложениям (Windows Forms .NET) [Электронный ресурс]. – Режим доступа: https://docs.microsoft.com/ru-ru/dotnet/desktop/winforms/overview/?view=netdesktop-5.0 (Дата обращения 15.05.2021).
11. Qt [Электронный ресурс]. – Режим доступа: https://ru.bmstu.wiki/Qt_(программного_обеспечения)#:~:text=Qt%20(произносится%20%5Bˈkjuːt%5D%20(кьют)%20как,используемый%20для%20создания%20графических%20интерфейсов (Дата обращения 15.05.2021).

12. Юзабилити тестирование [Электронный ресурс]. – Режим доступа: https://habr.com/ru/company/luxoft/blog/508146/(Дата обращения: 15.05.2021).
13. Функциональное тестирование [Электронный ресурс]. – Режим доступа https://habr.com/ru/post/549054/ (Дата обращения: 15.05.2021).

ПРИЛОЖЕНИЕ

Листинг 1 — исходный код виджетов:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Turing_Post
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        List<Label> VisibleBandCaptionPost = new List<Label>();
        List<CheckBox> VisibleBandPost = new List<CheckBox>(); //Часть ленты для показа
        Dictionary<int, CheckState> BandPost = new Dictionary<int, CheckState>(); //вся лента

        List<TextBox> VisibleBandTuring = new List<TextBox>();
        List<Label> VisibleBandCaptionTuring = new List<Label>();
        Dictionary<int, string> BandTuring = new Dictionary<int, string>();

        Color CarretColor = Color.Cyan;

        private void Form1_Load(object sender, EventArgs e)
        {
            //Видимая часть ленты Поста
            int left = 3;
```

```
int width = 30;
while (left < panelPost.Width - width)
{
    Label l = new Label();
    l.Top = 0;
    l.Left = left;
    l.Width = width - 5;
    l.Parent = panelPost;
    l.Text = "-";
    VisibleBandCaptionPost.Add(l);

    CheckBox c = new CheckBox();
    c.Top = l.Height;
    c.Left = left+5;
    c.Width = width;
    c.Parent = panelPost;
    c.Text = "";
    c.CheckStateChanged += PostCheckChanged;
    VisibleBandPost.Add(c);
    left += width;
}
ShowPost();

//Видимая часть ленты Тьюринга
left = 3;
width = 30;
while (left < panelTuring.Width - width)
{
    Label l = new Label();
    l.Top = 0;
    l.Left = left+5;
    l.Width = width-5;
    l.Parent = panelTuring;
    l.Text = "-";
    VisibleBandCaptionTuring.Add(l);

    TextBox t = new TextBox();
```

```
            t.Top = l.Height;
            t.Left = left;
            t.Width = width;
            t.Parent = panelTuring;
            t.Leave += LeaveTuring;
            t.Enter += LeaveTuring;
            VisibleBandTuring.Add(t);
            left += width;
        }

        ShowTuring();
    }



    private void buttonLoadPost_Click(object sender, EventArgs e)
    {
        if (openFileDialogPost.ShowDialog() != DialogResult.OK) return;
        LoadPost(openFileDialogPost.FileName);
        ShowPost();
    }


    void LoadPost(string FileName)
    {
        string[] data = File.ReadAllLines(FileName);
        //Машина поста описывается двумя наборами строк.
        //Набор программ и набор данных.
        //Разделены в файле пустой строкой

        bool command = true;
        listBoxPost.Items.Clear(); //Очистить программу
        BandPost.Clear(); //и данные
        for (int k = -100; k <= 100; k++) //Первые 2*100
            if (checkBoxTriple.Checked)
            BandPost[k] = CheckState.Indeterminate;
        else
```

```
                BandPost[k] = CheckState.Unchecked;

            foreach (string s in data)
            {
                if (s == "") { command = false; continue; }
                if (command) { listBoxPost.Items.Add(s); continue; }
                if (!command)
                {
                    string[] value = s.Split('\t'); //Разделитель табуляция
                    int index = int.Parse(value[0]);
                    CheckState state = CheckState.Indeterminate;
                    if (value[1] == "Checked") state = CheckState.Checked;
                    if     (value[1]     ==     "Unchecked")     state     =
CheckState.Unchecked;
                    BandPost[index] = state;
                }
            }
        }

        //Сохранить машину поста
        private void buttonSavePost_Click(object sender, EventArgs e)
        {
            if (saveFileDialogPost.ShowDialog() != DialogResult.OK) return;
            SavePost(saveFileDialogPost.FileName);
        }

        void SavePost(string FileName)
        {
            List<string> data = new List<string>();
            foreach (string s in listBoxPost.Items)
                if (s == "") data.Add("Nop"); else
                data.Add(s);
            data.Add("");
            foreach (int index in BandPost.Keys)
                  data.Add(index.ToString()         +         "\t"         +
BandPost[index].ToString());
            File.WriteAllLines(FileName, data.ToArray());
```

```
        }

        //Показать состояние машины Поста (данные)
        bool locked = false;
        void ShowPost()
        {
            locked = true;
            int offset = hScrollBarPost.Value; //Смещение
            offset = -offset;
            int middle = VisibleBandPost.Count / 2; //Серединка (там [текущий]
элемент данных)
            for (int k=0; k<VisibleBandPost.Count; k++)
            {
                int p = k - middle - offset;
                VisibleBandCaptionPost[k].Text = p.ToString();
                if (BandPost.ContainsKey(p))
                    VisibleBandPost[k].CheckState = BandPost[p];
                else
                    if (checkBoxTriple.Checked)
                    VisibleBandPost[k].CheckState                          =
CheckState.Indeterminate;
                else
                    VisibleBandPost[k].CheckState = CheckState.Unchecked;
                VisibleBandCaptionPost[middle].BackColor = Color.LightGray;
            }

            VisibleBandCaptionPost[middle].BackColor = CarretColor;

            locked = false;
        }

        //Редактирование одной ячейки данных
        //При изменении значения (Пост, Чекбокс)
        void PostCheckChanged(object sender, EventArgs e)
        {
            if (locked) return;
            CheckBox Sender = sender as CheckBox;
```

```
            int index = VisibleBandPost.IndexOf(Sender);
            int p = int.Parse(VisibleBandCaptionPost[index].Text);
            BandPost[p] = Sender.CheckState;
            ShowPost();
        }


        //Заменить (пост)
        private void buttonEditPost_Click(object sender, EventArgs e)
        {
            if (listBoxPost.SelectedIndex < 0) return;
            //Проверить синтаксическую правильноссть команды машины Поста
            string s = textBoxPostCommand.Text;
            //Длина
            if (s.Length==0)
            {
                MessageBox.Show("Пустая команда");
            }
            else
            switch (s[0])
            {
                case '>':
                case '<':
                case '0':
                case '1':
                case '?':
                case '.':
                case 'S':
                case 'N':
                case 'X':
                    break;
                default:
                    MessageBox.Show("Команда может начинаться только с
символов ><01?.SNX");
                    break;
            }
```

```
            listBoxPost.Items[listBoxPost.SelectedIndex]                    =
textBoxPostCommand.Text;
        }

        private    void    listBoxPost_SelectedIndexChanged(object    sender,
EventArgs e)
        {
            if (listBoxPost.SelectedIndex < 0) return;
            textBoxPostCommand.Text                                         =
(string)listBoxPost.Items[listBoxPost.SelectedIndex];
        }

        private void buttonUp_Click(object sender, EventArgs e)
        {
            if (listBoxPost.SelectedIndex <= 0) return;
            var temp = listBoxPost.Items[listBoxPost.SelectedIndex];
            listBoxPost.Items[listBoxPost.SelectedIndex]                    =
listBoxPost.Items[listBoxPost.SelectedIndex - 1];
            listBoxPost.Items[listBoxPost.SelectedIndex - 1] = temp;
            listBoxPost.SelectedIndex--;
        }

        private void buttonDown_Click(object sender, EventArgs e)
        {
            if   (listBoxPost.SelectedIndex   >=   listBoxPost.Items.Count-1)
return;
            var temp = listBoxPost.Items[listBoxPost.SelectedIndex];
            listBoxPost.Items[listBoxPost.SelectedIndex]                    =
listBoxPost.Items[listBoxPost.SelectedIndex + 1];
            listBoxPost.Items[listBoxPost.SelectedIndex + 1] = temp;
            listBoxPost.SelectedIndex++;
        }

        private void buttonInsert_Click(object sender, EventArgs e)
        {
            string text = textBoxPostCommand.Text;
            listBoxPost.Items.Add("");
```

```
            listBoxPost.SelectedIndex = listBoxPost.Items.Count - 1;
            textBoxPostCommand.Text = text;
            buttonEditPost_Click(sender, e);
        }

        private void buttonDelete_Click(object sender, EventArgs e)
        {
            if (listBoxPost.SelectedIndex < 0) return;
            listBoxPost.Items.RemoveAt(listBoxPost.SelectedIndex);
        }



        //Выполнить один шаг
        private void buttonStepPost_Click(object sender, EventArgs e)
        {
            if (listBoxPost.SelectedIndex < 0) return;
            //Прочитать команду
            string                          command                          =
listBoxPost.Items[listBoxPost.SelectedIndex].ToString();
            command = command.Replace(" ", ""); //Без пробелов
            int index = hScrollBarPost.Value;
            //Заполнить значением
            if (!BandPost.ContainsKey(index))
                if (checkBoxTriple.Checked)
                    BandPost[index] = CheckState.Indeterminate;
            else
                    BandPost[index] = CheckState.Unchecked;
            //Команда может быть :
            switch (command[0])
            {
                case '>'://> [N]
                    Next(1, command);
                    break;
                case '<'://< [N]
                    Next(-1, command);
                    break;
                case '0': //0 [N]
```

```
            BandPost[index] = CheckState.Unchecked;
            Next(0, command);
            break;
        case '1'://1 [N]
            BandPost[index] = CheckState.Checked;
            Next(0, command);
            break;
        case '?':
            Split(command);
            if (checkBoxTriple.Checked)
            {
                //? N,M,L
                //undef => N
                if (BandPost[index] == CheckState.Indeterminate)
                    listBoxPost.SelectedIndex = N;
                //0 => M
                if (BandPost[index] == CheckState.Unchecked)
                    listBoxPost.SelectedIndex = M;
                //1 => L
                if (BandPost[index] == CheckState.Checked)
                    listBoxPost.SelectedIndex = L;
            }
            else
            {
                //? N,M
                //0->N
                if (BandPost[index] == CheckState.Unchecked)
                    listBoxPost.SelectedIndex = N;
                //1->M
                if (BandPost[index] == CheckState.Checked)
                    listBoxPost.SelectedIndex = M;
            }
            break;
        case '.': case 'S'://. или Stop
            ///Ничего не делать (вообще)
            checkBoxAnimatorPost.Checked = false;
            MessageBox.Show("Машина остановлена");
```

```
                    break;
                case 'N': case ' '://Nop
                    listBoxPost.SelectedIndex++; //ничего не делать и перейти
к следующей команде
                    break;
                case 'X'://X [N]
                    BandPost[index] = CheckState.Indeterminate;
                    Next(0, command);
                    break;

            }
            ShowPost();
        }

        int N, M, L;

        private void checkBoxTriple_CheckedChanged(object sender, EventArgs
e)
        {
            foreach (CheckBox c in panelPost.Controls)
                c.ThreeState = checkBoxTriple.Checked;
            ShowPost();
        }

        private void hScrollBarPost_Scroll(object sender, ScrollEventArgs e)
        {
            ShowPost();
        }


        void Split(string command)
        {
            command = command.Substring(1, command.Length - 1);
            try
            {
                string[] NML = command.Split(',');
```

```
            N = listBoxPost.SelectedIndex + 1;
            M = N;
            L = N;

            switch (NML.Length)
            {
                case 0:
                    break;
                case 1:
                    if (NML[0] != "")
                        N = int.Parse(NML[0]);
                    break;
                case 2:
                    N = int.Parse(NML[0]);
                    M = int.Parse(NML[1]);
                    break;
                default:
                    N = int.Parse(NML[0]);
                    M = int.Parse(NML[1]);
                    L = int.Parse(NML[2]);
                    break;
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show("При обработке команды " + command + "
получено " + ex.Message,
                "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }


    void Next(int dataoffset, string command)
    {
        hScrollBarPost.Value+=dataoffset;
        Split(command);
        if (N>=0 && N< listBoxPost.Items.Count)
```

```
      listBoxPost.SelectedIndex = N;
}

//********************  Тьюринг ******************************
private void buttonLoadTuring_Click(object sender, EventArgs e)
{
    if (openFileDialogTuring.ShowDialog() != DialogResult.OK) return;
    LoadTuring(openFileDialogTuring.FileName);
    buttonListToTable_Click(sender, e);
    ShowTuring();
}

void LoadTuring(string FileName)
{
    //Машина Тьюринга описывается:
    //Алфавитом
    //Количество состояний
    //Списком значений ленты
    //Набором правил перехода

    string[] Data = File.ReadAllLines(FileName);
    textBoxAlphabet.Text = Data[0];
    numericUpDownQ.Value = int.Parse(Data[1]);

    int k = 2;
    while (Data[k] != "")
    {
        string[] band = Data[k].Split('\t');
        int index = int.Parse(band[0]);
        BandTuring[index] = band[1];
        k++;
    }
    k++;

    listBoxTuring.Items.Clear();
    for (; k < Data.Length; k++)
        listBoxTuring.Items.Add(Data[k]);
```

```
        }

        private void buttonSaveTuring_Click(object sender, EventArgs e)
        {
            if (saveFileDialogTuring.ShowDialog() != DialogResult.OK) return;
            SaveTuring(saveFileDialogTuring.FileName);
        }

        void SaveTuring(string FileName)
        {
            List<string> Data = new List<string>();
            //Алфавит
            Data.Add(textBoxAlphabet.Text);
            //Состояния
            Data.Add(numericUpDownQ.Value.ToString());
            //Значения на ленте
            foreach (int key in BandTuring.Keys)
            {
                string band = key.ToString() + "\t" + BandTuring[key];
                Data.Add(band);
            }

            //Таблица состояний
            Data.Add("");
            foreach (string s in listBoxTuring.Items)
                Data.Add(s);
            File.WriteAllLines(FileName, Data.ToArray());
        }

        private   void   listBoxTuring_SelectedIndexChanged(object   sender,
EventArgs e)
        {
            if (listBoxTuring.SelectedIndex < 0) return;
            textBoxTuringCommand.Text                                        =
(string)listBoxTuring.Items[listBoxTuring.SelectedIndex];
        }
```

```
        private void buttonEditTuring_Click(object sender, EventArgs e)
        {
            if (listBoxTuring.SelectedIndex < 0) return;
            //Проверить синтаксическую правильность команды машины Тьюринга

            //qi Si -> qj Sj M

            string[] ss = textBoxTuringCommand.Text.Split(' ');
            if (ss.Length != 6)
            {
                MessageBox.Show("Команда - это 6 полей, разделенных пробелом","Ошибка",MessageBoxButtons.OK,MessageBoxIcon.Error);
                return;
            };

            if (ss[0][0] != 'q')
            {
                MessageBox.Show("Первый операнд - номер состояния. Например, q0", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }

            if (ss[2] != "->" && ss[2] != "=>")
            {
                MessageBox.Show("Должна быть стрелка", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }


            if (ss[3][0] != 'q')
            {
                MessageBox.Show("Четвертый операнд - номер состояния. Например, q0", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }
```

```
            if ("NLRS".IndexOf(ss[5]) < 0)
            {
                MessageBox.Show("Последний операнд - L,R,N или S", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }

            listBoxTuring.Items[listBoxTuring.SelectedIndex]                =
textBoxTuringCommand.Text;
            buttonListToTable_Click(sender, e); //Обновить табличку
        }

        private void buttonListToTable_Click(object sender, EventArgs e)
        {
            //Каждая команда в формате
            //Qi Ai -> Qj Aj R|L|N
            //Где
            //Qi - входное состояние
            //Ai - входной символ
            //Qj - выходное состояние
            //Aj - выходной символ
            //R|L|N - смещение головки

            //Определить наличие в алфавите всех использованных в программе
символов
            int qq = Q;
            Q = 0;
            string Alphabet = "";
            //Определить количество состояний
            int column, row;
            foreach (string s in listBoxTuring.Items)
            {
                string[] command = s.Split(' ');
                column = int.Parse(command[0].Substring(1, command[0].Length
- 1));
                row = Alphabet.IndexOf(command[1]);
                if (column > Q) Q = column;
```

```
                if (row < 0) Alphabet += command[1];

                column = int.Parse(command[3].Substring(1, command[3].Length
- 1));
                row = Alphabet.IndexOf(command[4]);
                if (column > Q) Q = column;
                if (row < 0) Alphabet += command[4];
            }
            //Применить изменения
            textBoxAlphabet.Text = Alphabet;
            numericUpDownQ.Value = Q+1;

            numericUpDownQ_ValueChanged(sender, e);
            //Очистить таблицу
            for (row = 0; row < dgv.RowCount; row++)
                for (column = 0; column < dgv.ColumnCount; column++)
                    dgv.Rows[row].Cells[column].Value = null;
            //Заполнить таблицу
            foreach (string s in listBoxTuring.Items)
            {
                string[] command = s.Split(' ');
                column = int.Parse(command[0].Substring(1, command[0].Length
- 1));
                row = textBoxAlphabet.Text.IndexOf(command[1]);

                if (dgv.Rows[row].Cells[column].Value == null)
                    dgv.Rows[row].Cells[column].Value = s;
                if (dgv.Rows[row].Cells[column].Value.ToString() != s)
                    MessageBox.Show("Конфликт правил "+s+" и "+
dgv.Rows[row].Cells[column].Value.ToString(),
                        "Ошибка",MessageBoxButtons.OK,MessageBoxIcon.Error);
            }

            //Заполнить список состояний
            comboBoxQ.Items.Clear();
            for (int q = 0; q < numericUpDownQ.Value; q++)
                comboBoxQ.Items.Add("q" + q.ToString());
```

```
    Q = qq;
    comboBoxQ.SelectedIndex = Q;
}

private void buttonAddTuring_Click(object sender, EventArgs e)
{
    string text = textBoxTuringCommand.Text;
    listBoxTuring.Items.Add("");
    listBoxTuring.SelectedIndex = listBoxTuring.Items.Count - 1;
    textBoxTuringCommand.Text = text;
    buttonEditTuring_Click(sender, e);
}

private void buttonRemoveTuring_Click(object sender, EventArgs e)
{
    if (listBoxTuring.SelectedIndex < 0) return;
    listBoxTuring.Items.RemoveAt(listBoxTuring.SelectedIndex);
}

private void textBoxAlphabet_Leave(object sender, EventArgs e)
{
    numericUpDownQ_ValueChanged(sender, e);
}

int Q = 0; //Состояние машины Тьюринга
int P = 0; //Выбранная клетка машины Тьюринга

private void buttonStepTuring_Click(object sender, EventArgs e)
{
    P = hScrollBarTuring.Value;
    //Найти в таблице команду для текущего состояния и символа
    int p;
    if (!BandTuring.ContainsKey(P)) BandTuring[P] = "_";
    p = textBoxAlphabet.Text.IndexOf(BandTuring[P]); //Символ
    int q = Q; //состояние
    if (p<0 || dgv.Rows[p].Cells[q].Value == null)
    {
```

```
                checkBoxAnimatorTuring.Checked = false;
                MessageBox.Show("Действие машины не определено");
                return;
            }
            string command = dgv.Rows[p].Cells[q].Value.ToString();
            string[] band = command.Split(' ');
            //Первые 2 определяют текущее состояние и положение, уже
использованы
            //Вторые 2 определяют
            //новое состояние
            Q = int.Parse(band[3].Substring(1));
            //новое значение в ячейке
            BandTuring[P] = band[4].Substring(0, 1);
            //последняя определяет изменение номера ячейки
            switch (band[5])
            {
                case "L":
                    P--;
                    break;
                case "R":
                    P++;
                    break;
                case "N":
                    break;
                case "S":
                    checkBoxAnimatorTuring.Checked = false;
                    MessageBox.Show("Машина остановлена");
                    return;
            }
            hScrollBarTuring.Value = P;
            hScrollBarTuring_Scroll(sender, null);
            comboBoxQ.SelectedIndex = Q;
        }

        private void hScrollBarTuring_Scroll(object sender, ScrollEventArgs
e)
        {
```

```
            ShowTuring();
        }

        private void numericUpDownQ_ValueChanged(object sender, EventArgs e)
        {
            //Обработать количество состояний
            while (dgv.ColumnCount < numericUpDownQ.Value)
                dgv.Columns.Add("", "q" + (dgv.ColumnCount).ToString());
            while (dgv.ColumnCount > numericUpDownQ.Value)
                dgv.Columns.RemoveAt(dgv.ColumnCount - 1);

            //Обработать алфавит
            string Alphabet = textBoxAlphabet.Text;
            //Заменить " " на "_"
            Alphabet = Alphabet.Replace(" ", "_");
            //Проверить наличие символа "_"
            //Если нет, то добавить
            if (Alphabet.IndexOf("_") < 0) Alphabet += "_";

            textBoxAlphabet.Text = Alphabet;

            while (dgv.RowCount<Alphabet.Length) dgv.Rows.Add();

            while          (dgv.RowCount          >          Alphabet.Length)
dgv.Rows.RemoveAt(dgv.RowCount - 1);

            for (int row = 0; row<dgv.RowCount; row++)
                dgv.Rows[row].HeaderCell.Value = Alphabet.Substring(row, 1);

        }

        private void comboBoxQ_SelectedIndexChanged(object sender, EventArgs
e)
        {
            Q = comboBoxQ.SelectedIndex;
        }

        private  void  checkBoxAnimatorTuring_CheckedChanged(object  sender,
EventArgs e)
        {
            timerTuring.Enabled = checkBoxAnimatorTuring.Checked;
        }

        private   void   checkBoxAnimatorPost_CheckedChanged(object   sender,
EventArgs e)
```

```
        {
            timerPost.Enabled = checkBoxAnimatorPost.Checked;
        }

        void ShowTuring()
        {
            locked = true;
            int offset = hScrollBarTuring.Value; //Смещение
            offset = -offset;
            int middle = VisibleBandTuring.Count / 2; //Серединка (там [0]
элемент данных)
            for (int k = 0; k < VisibleBandTuring.Count; k++)
            {
                int p = k - middle - offset;
                VisibleBandCaptionTuring[k].Text = p.ToString();
                if (BandTuring.ContainsKey(p))
                    VisibleBandTuring[k].Text = BandTuring[p];
                else
                    VisibleBandTuring[k].Text = "_";
                if (p == P)
                    VisibleBandCaptionTuring[k].BackColor = CarretColor;
                else
                    VisibleBandCaptionTuring[k].BackColor = Color.LightGray;
            }
            locked = false;
        }

        //Обработчик завершения редактирования тьюринга
        void LeaveTuring(object sender, EventArgs e)
        {
            TextBox Sender = sender as TextBox;
            int index = VisibleBandTuring.IndexOf(Sender);
            int p = int.Parse(VisibleBandCaptionTuring[index].Text);
            if (Sender.Text.Length==0)
            {
                BandTuring[p] = "_";
            }
            else
                BandTuring[p] = Sender.Text.Substring(0,1);
            P = p; //Редактированное == текущему
            ShowTuring();
        }
    }
}
```