ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА  
(САМАРСКИЙ УНИВЕРСИТЕТ)»

Факультет информатики   
Кафедра программных систем

**ОТЧЕТ**

к лабораторному практикуму

по дисциплине «Технологии программирования»

по теме «Система моделирования работы

машины Тьюринга»

Студент А.С. Постников

Студент А.А. Терёхин

Студент А.А. Шапошников

Руководитель Л.С. Зеленко

Самара 2017

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА  
(САМАРСКИЙ УНИВЕРСИТЕТ)»

Факультет информатики

Кафедра программных систем

**ЗАДАНИЕ**

на лабораторный практикум по дисциплине

«Технологии программирования»

студентов группы № 6401-090301D  
А.С. Постникову  
А.А. Терёхину  
А.А. Шапошникову

1. **Тема проекта:** «Система моделирования работы машины Тьюринга
2. **Исходные данные к проекту:** см. приложение к заданию
3. **Перечень вопросов, подлежащих разработке:**
   1. Произвести анализ предметной области: изучить основные принципы конечных автоматов, изучить особенности машины Тьюринга
   2. Выполнить обзор существующих систем-аналогов
   3. Разработать информационно-логический проект
   4. Разработать и реализовать программное и информационное обеспечение, провести тестирование и отладку
   5. Оформить документацию на проект
   6. Подготовить презентацию по разработанной системе
4. **Перечень графических разработок:**
   1. Структурная схема системы
   2. Диаграммы объектов, классов, компонентов
   3. Схемы основных алгоритмов
5. **Календарный план выполнения работ**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № п/п | Содержание работы по этапам | Объем этапа в % к общему объему проекта | Срок  окончания | Фактическое выполнение |
| 1 | Оформление технического задания и его утверждение | 5 | 21.09.2017 |  |
| 2 | Описание и анализ предметной области (1 раздел) | 10 | 29.10.2017 |  |
| 3 | Проектирование системы (2 раздел) | 30 | 10.11.2017 |  |
| 3.1 | Разработка структурной схемы системы | 5 | 13.10.2017 |  |
| 3.2 | Разработка функциональной спецификации системы | 10 | 27.10.2017 |  |
| 3.3 | Разработка прототипов экранных форм | 10 | 27.10.2017 |  |
| 3.4 | Разработка основных алгоритмов | 5 | 10.11.2017 |  |
| 4 | Реализация проекта, разработка контрольных примеров. Предъявление реализации руководителю (3 раздел) | 45 | 14.12.2017 |  |
| 5 | Корректировка проекта и оформление документации проекта. Защита проекта с представлением презентации | 10 | 22.12.2017 |  |

Задание принял  
к исполнению А.С. Постников

А.А. Терёхин

А.А. Шапошников

**ПРИЛОЖЕНИЕ  
к заданию на лабораторный практикум**студентов группы № 6401-090301D  
А.С. Постникова  
А.А. Терёхина  
А.А. Шапошникова

Тема проекта: **«Система моделирования работы машины Тьюринга»**

**Исходные данные к проекту:**

1. **Характеристика объекта автоматизации:**
2. объект автоматизации: машина Тьюринга (МТ);
3. виды автоматизируемой деятельности:
   * процесс работы МТ с заданным алгоритмом;
   * процесс визуализации работы МТ;
   * процесс составления алгоритма;
4. основание системы счисления – 10;
5. количество операндов – 2;
6. количество способов задания операндов – 2;
7. минимальная длина ленты – 500;
8. максимальная длина ленты – 500;
9. минимальное количество состояний МТ – 1;
10. максимальное количество состояний МТ – 100;
11. минимальная длина алфавита – 3;
12. максимальная длина алфавита – 100;
13. количество стандартных алгоритмов – 2;
14. количество команд перемещения, выполняемых МТ – 3;
15. количество режимов работы МТ – 3;
16. количество способов визуализации – 2;
17. количество скоростных режимов – 5;
18. минимальная величина задержки – 15мс;
19. максимальная величина задержки – 500мс.
20. **Требования к информационному обеспечению:**
21. информационное обеспечение разрабатывается на основе следующего источника:
    * Описание структуры машины Тьюринга [Электронный ресурс]. URL: ru.wikipedia.org/wiki/Машина\_Тьюринга (дата обращения: 10.09.2017);
22. предусмотреть проверку на наличие бесконечных циклов;
23. трасса хранится в текстовом файле формата \*.tmt
24. алгоритмы для МТ хранятся в файлах формата .tma, структура файла определяется в процессе проектирования.
25. **Требования к техническому обеспечению:**
26. тип ЭВМ - IBM PC совместимый;
27. монитор с разрешающей способностью не ниже 800 х 600;
28. манипулятор – мышь;
29. технические характеристики определяются в процессе выполнения проекта.
30. **Требования к программному обеспечению:**
31. тип операционной системы ‑ Windows 7/8 и выше;
32. язык программирования – С#;
33. среда программирования – Visual Studio 2017;
34. среда проектирования – StarUML 2.8.0.
35. **Общие требования к проектируемой системе:**

***5.1 Функции, реализуемые системой:***

* + визуализация процесса работы МТ;
  + синтаксический анализ алгоритма;
  + семантический анализ алгоритма;
  + выдача справочной информации о системе;
  + выбор способа задания операндов;
  + редактирование символа на ленте;
  + выбор режима работы МТ;
  + выбор скорости визуализации работы МТ
  + создание алфавита;
  + редактирование алфавита;
  + выбор стандартного алгоритма;
  + ручное составление алгоритма;
  + запись алгоритма в файл;
  + загрузка алгоритма из файла;
  + формирование трассы выполняемого алгоритма;
  + сохранение трассы в файл.

***5.2 Технические требования к системе:***

1. режим работы ‑ диалоговый;
2. система должна удовлетворять санитарным правилам и нормам  
    СанПин 2.2.2./2.4.2198-07;
3. условия работы средств вычислительной техники (содержание вредных веществ, пыли и подвижность воздуха) должны соответствовать ГОСТ 12.1.005, 12.01.007;
4. температура окружающего воздуха – 15-35°С;
5. влажность воздуха – 45-75%.

Руководитель   
проекта Л.С. Зеленко

Задание принял  
к исполнению А.С. Постников

А.А. Терёхин

А.А. Шапошников

**РЕФЕРАТ**

Пояснительная записка 85 с, 10+8+21+13=52 рисунка, 0+2+5 = 7 таблиц, 23 источника, 2 приложения.

Графическая часть: 14 слайдов презентации PowerPoint.

**МАШИНА ТЬЮРИНГА, АЛГОРИТМ, КОНЕЧНЫЙ АВТОМАТ, ПРОЦЕДУРНЫЙ ЯЗЫК.**

Во время лабораторного практикума разработаны алгоритмы и соответствующая им программа, позволяющая моделировать работу машины Тьюринга. Существует возможность загрузить алгоритм из стандартных, открыть из файла или создать собственный. Реализовано три режима работы: пошаговый, с задержкой и мгновенный. Операнды можно задавать на форме или на ленте. Система позволяет отображать работу в двух режимах: с двигающейся лентой и с подсвечиванием. Также предусмотрена возможность ведения трассы. В программе имеется возможность сохранения алгоритма и трассы выполнения в файл.

Программа написана на языке С# в среде Visual Studio 2017, использует базу данных SQLite и функционирует под управлением операционной системы Windows 7/8/10.

**СОДЕРЖАНИЕ**

[Введение 1](#_Toc501368260)0

[1 Описание и анализ предметной области 1](#_Toc501368261)2

[1.1 Принцип работы машины Тьюринга 1](#_Toc501368262)2

[1.2 Алгоритм сложения](#_Toc501368263) 14

[1.3 Варианты реализаций машины Тьюринга 15](#_Toc501368264)

[1.3.1 Машина Тьюринга, работающая на полубесконечной ленте 15](#_Toc501368266)

[1.3.2 Двумерные машины Тьюринга 16](#_Toc501368267)

[1.3.3 Вероятностная машина Тьюринга 18](#_Toc501368268)

[1.3.4 Универсальная машина Тьюринга 18](#_Toc501368268)

[1.4 Обзор систем-аналогов 19](#_Toc501368269)

[1.4.1 Машина Поста 19](#_Toc501368268)

[1.4.2 «Algo2000» 20](#_Toc501368268)

[1.5 Диаграмма объектов предметной области 21](#_Toc501368269)

[1.6 Постановка задачи 22](#_Toc501368270)

[2 Проектирование системы 25](#_Toc501368271)

[2.1 Структурная схема системы 25](#_Toc501368272)

[2.2 Спецификация системы 26](#_Toc501368273)

[2.2.1 Функциональная спецификация 26](#_Toc501368274)

[2.2.2 Спецификация качества 31](#_Toc501368275)

[2.2.3 Перечень исключительных ситуаций 31](#_Toc501368276)

[2.3 Разработка прототипа интерфейса пользователя системы 31](#_Toc501368277)

[2.4 Разработка и описание алгоритмов обработки данных 34](#_Toc501368280)

[2.5 Выбор и обоснование комплекса программных средств 37](#_Toc501368281)

[2.5.1 Выбор языка программирования и среды разработки 37](#_Toc501368282)

[2.5.2 Выбор операционной системы 37](#_Toc501368283)

[2.5.3 Выбор среды программирования 38](#_Toc501368284)

[3 Реализация системы 39](#_Toc501368286)

[3.1 Разработка и описание интерфейса пользователя 39](#_Toc501368287)

[3.1.1 Разработка и описание пользовательского меню 39](#_Toc501368288)

[3.1.2 Работа с алгоритмом 41](#_Toc501368289)

[3.1.3 Настройка параметров 42](#_Toc501368290)

[3.1.4 Моделирование работы МТ 44](#_Toc501368291)

[3.2 Реализация классов 46](#_Toc501368292)

[3.3 Выбор и обоснование комплекса технических средств 49](#_Toc501368295)

[3.3.1 Расчет объема занимаемой памяти 49](#_Toc501368296)

[3.3.2 Минимальные требования, предъявляемые к системе 50](#_Toc501368297)

[Заключение 51](#_Toc501368298)

[Список использованных источников 52](#_Toc501368299)

[Приложение А Руководство пользователя 55](#_Toc501368300)

[А.1 Введение 55](#_Toc501368301)

[А.2 Условия работы системы 55](#_Toc501368302)

[А.3 Подготовка к работе 55](#_Toc501368303)

[А.4 Описание операций 56](#_Toc501368304)

[А.4.1 Пользовательское меню 56](#_Toc501368305)

[А.4.2 Создание нового алгоритма 56](#_Toc501368305)

[А.4.3 Настройка параметров моделирования 58](#_Toc501368305)

[А.4.4 Моделирование работы алгоритма 60](#_Toc501368305)

[Приложение Б Листинг модулей программы 63](#_Toc501368308)

# ВВЕДЕНИЕ

В 1935 г. возникло такое положение: свойства, обнаруженные у некоторого точно определенного класса вычислимых теоретико-числовых функций, изучавшихся Чёрчем и Клини в 1932-1935 годах, упорно подсказывали мысль, что этот класс, может быть, охватывает все функции, которые в соответствии с нашим интуитивным представлением можно рассматривать как вычислимые. При этих обстоятельствах Чёрч выдвинул тезис (опубликован в 1936 г.), что все функции, которые интуитивно мы можем рассматривать как вычислимые, или, говоря его словами, как «эффективно вычислимые», являются общерекурсивными.

Несколько позже, но независимо появилась статья Тьюринга (1936), в которой был введен еще один точно определенный класс интуитивно вычислимых функций, «функции, вычислимые по Тьюрингу». Вскоре Тьюрингом в 1937 году было показано, что его вычислимые функции – это то же самое, что λ-определимые функции, и, следовательно, то же самое, что и общерекурсивные функции [1].

Логичным являлся вопрос: можно ли для любого математического утверждения указать конечную последовательность инструкций, которые могли бы выполняться механически одна за другой человеком или вычислительным устройством, и в итоге выяснить, истинно это утверждение или ложно? В качестве математической модели для описания таких алгоритмов было предложено абстрактное вычислительное устройство, которое впоследствии было названо машина Тьюринга (МТ). МТ – дискретное вычислительное устройство, изменяющее свои характеристики в определенные моменты времени. Хотя и доказано, что поставленная проблема в общем случае неразрешима, применение МТ вышло далеко за пределы первоначальной постановки задачи. Хотя МТ не стала реально действующим устройством, она до настоящего времени постоянно используется в качестве основной модели для выяснения сущности таких понятий, как «вычислительный процесс», «алгоритм», а также для выяснения связи между алгоритмом и вычислительными машинами [2].

Так как процесс составления алгоритма вручную занимает много времени, предпочтительно автоматизировать этот процесс. В связи с этим во время лабораторного практикума авторам необходимо разработать систему, моделирующую работу МТ, которая будет составлять алгоритмы, проверять их работоспособность и визуализировать свою работу.

При разработке будут использованы методология структурного подхода к проектированию, построенная на декомпозиции (разбиении) системы на автоматизируемые функции, и методология ООАП (объектно-ориентированный анализ и проектирование), в основе которой лежит объектная декомпозиция, а так же технология RAD (от англ. RapidApplicationDevelopment – быстрая разработка приложений), которая хороша в первую очередь для относительно небольших проектов, разрабатываемых для конкретного заказчика, так называемое заказное ПО [3].

Основные принципы технологии RAD:

− разработка приложений итерациями;

− необязательность полного завершения работ на каждом этапе ЖЦ

− обязательное вовлечение пользователей на этапе разработки;

− тестирование и развитие проекта одновременно с разработкой;

− грамотное руководство разработкой, четкое планирование и контроль выполнения работ.

1. **ОПИСАНИЕ И АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ**

Предметная область – область деятельности в реальном мире, которая содержит элементы системы, процессы их взаимодействия между собой, информационные потоки, различные функциональные подсистемы, а также внешние по отношению к исследуемой системе процессы, события и явления [4].

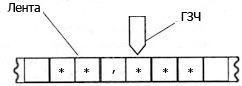
* 1. **Принципы работы машины Тьюринга**

Машина Тьюринга – абстрактный исполнитель (абстрактная вычислительная машина). Была предложена Аланом Тьюрингом в 1936 году для формализации понятия алгоритма.

Машина Тьюринга является расширением конечного автомата и, согласно тезису Чёрча-Тьюринга, способна имитировать всех исполнителей (с помощью задания правил перехода), каким-либо образом реализующих процесс пошагового вычисления, в котором каждый шаг вычисления достаточно элементарен.

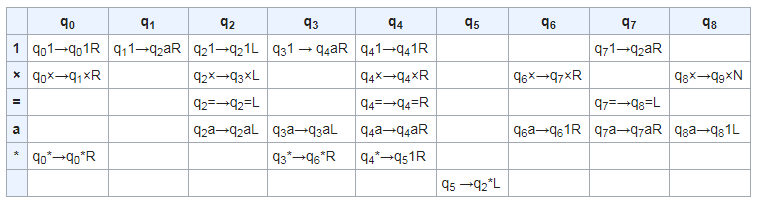
То есть всякий интуитивный алгоритм может быть реализован с помощью некоторой машины Тьюринга.

В состав машины Тьюринга входит неограниченная в обе стороны лента (возможны машины Тьюринга, которые имеют несколько бесконечных лент), разделённая на ячейки, и управляющее устройство (также называется головкой записи-чтения (ГЗЧ)), способное находиться в одном из множества состояний. Число возможных состояний управляющего устройства конечно и точно задано (на месте, вправо, влево). На рисунке 1.1 изображена структура МТ.

  
Рисунок 1.1 – Структура МТ

Управляющее устройство может перемещаться влево и вправо по ленте, читать и записывать в ячейки символы некоторого конечного алфавита. Конечный алфавит – конечное множество атомарных символов какого-либо формального языка [5]. Примером конечного алфавита являются ноты, цифры. Выделяется особый пустой символ, заполняющий все клетки ленты, кроме тех из них (конечного числа), на которых записаны входные данные.

Управляющее устройство работает согласно правилам перехода, которые представляют алгоритм, реализуемый данной машиной Тьюринга. Каждое правило перехода предписывает машине, в зависимости от текущего состояния и наблюдаемого в текущей клетке символа, записать в эту клетку новый символ, перейти в новое состояние и переместиться на одну клетку влево или вправо. Некоторые состояния машины Тьюринга могут быть помечены как терминальные, и переход в любое из них означает конец работы, остановку алгоритма (рисунок 1.2):

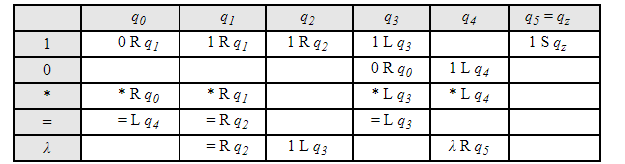
Рисунок 1.2 – Пример реализации правил перехода

Алфавитом являются строки (1,x,=,a,\*, ), а состояниями являются столбцы (q0 – q8). На пересечении строки и столбца располагается команда. В качестве примера разберём МТ, которая в данный момент находится на символе «1» и в состоянии q3. Для данного примера команда «q4 a R» осуществляет переход в состояние q4, заменяет текущий символ на «a», а ГЧЗ сдвигает вправо.

Машина Тьюринга называется детерминированной, если каждой комбинации состояния и ленточного символа в таблице соответствует не более одного правила. Если существует пара «ленточный символ - состояние», для которой существует 2 и более команд, такая машина Тьюринга называется недетерминированной.

* 1. **Алгоритм сложения**

Рассмотрим алгоритм сложения двух чисел, заданных на ленте в унарной системе счисления для машины Тьюринга. Алгоритм принимает 2 числа, разделённых символом разделителя ('\*'), и записывает результат их сложения на ленте сразу за этими числами отделяя его символом '='. Начальное состояние машины Тьюринга – q0, а головка обозревает первый символ первого числа. Конечное состояние – qz, при этом головка возвращается к исходной позиции. Так, чтобы вычислить 3+2, следует записать в качестве исходных данных 11\*111 результатом должно быть 11111. Программа перед своим завершением должна восстановить исходные данные на ленте (рисунок 1.3):

  
Рисунок 1.3 – Программа сложения двух чисел в унарной системе для машины Тьюринга

Программа находит очередной символ '1' слагаемых, заменяет его на символ '0' (своеобразная пометка, что этот символ уже обработан) и записывает очередной символ '1' в результат. Затем головка возвращается к началу в поисках символа '0'. Если есть ещё символы '1' до символа '=', то повторяем предыдущий шаг, в противном случае – возвращаем головку в исходное состояние, восстанавливая на ленте входные данные (заменяем   
'0' на '1')

Трасса вычислений выглядит следующим образом:

**1**1\*111; 1**1**\*111; 11**\***111; 11**1**111; 111**1**11; 111**1**11; 1111**1**1; 11111**1;** 11111**\_**

* 1. **Варианты реализаций машины Тьюринга**

Модель машины Тьюринга допускает расширения. Можно рассматривать машины Тьюринга с произвольным числом лент и многомерными лентами с различными ограничениями. Однако все эти машины являются полными по Тьюрингу и моделируются обычной машиной Тьюринга.

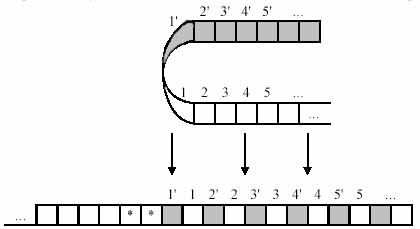
* + 1. ***Машина Тьюринга, работающая на полубесконечной ленте***

В качестве примера такого сведения рассмотрим следующую теорему: для любой машины Тьюринга существует эквивалентная машина Тьюринга, работающая на полубесконечной ленте (то есть на ленте, бесконечной в одну сторону).

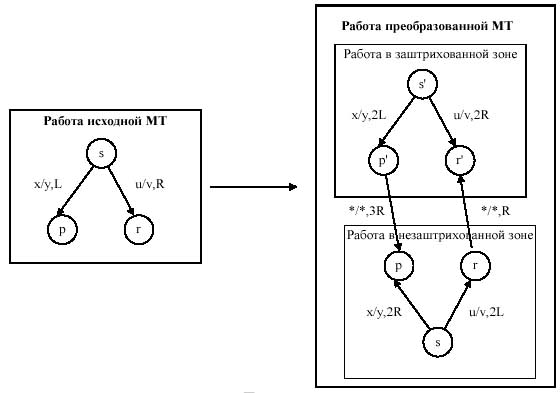
Рассмотрим доказательство, приведённое Ю. Г. Карповым в книге «Теория автоматов» [6]. Доказательство этой теоремы конструктивное, то есть мы дадим алгоритм, по которому для любой машины Тьюринга может быть построена эквивалентная машина Тьюринга с объявленным свойством. Во-первых, произвольно занумеруем ячейки рабочей ленты МТ, то есть определим новое расположение информации на ленте (рисунок 1.4):

https://upload.wikimedia.org/wikipedia/ru/5/5e/Mt1.jpg  
Рисунок 1.4 – Нумерованные ячейки рабочей ленты МТ

Затем перенумеруем ячейки, причём будем считать, что символ «\*» не содержится в словаре МТ (рисунок 1.5):

  
Рисунок 1.5 – Ячейки рабочей ленты МТ после преобразования

Наконец, изменим машину Тьюринга, удвоив число её состояний, и изменим сдвиг головки считывания-записи так, чтобы в одной группе состояний работа машины была бы эквивалентна её работе в заштрихованной зоне, а в другой группе состояний машина работала бы так, как исходная машина работает в незаштрихованной зоне. Если при работе МТ встретится символ ‘\*’, значит головка считывания-записи достигла границы зоны (рисунок 1.6).

  
Рисунок 1.6 – Алгоритм работы преобразованной МТ

Начальное состояние новой машины Тьюринга устанавливается в одной или другой зоне в зависимости от того, в какой части исходной ленты располагалась головка считывания-записи в исходной конфигурации. Очевидно, что слева от ограничивающих маркеров «\*» лента в эквивалентной машине Тьюринга не используется [7].

* + 1. ***Двумерные машины Тьюринга***

Муравей Лэнгтона - двумерный клеточный автомат с очень простыми правилами, изобретенный Крисом Лэнгтоном. Муравья можно также считать двумерной машиной Тьюринга с 2 символами и 4 состояниями.

*Правила*

Рассмотрим бесконечную плоскость, разбитую на клетки, покрашенные некоторым образом в чёрный и белый цвет. Пусть в одной из клеток находится «муравей», который на каждом шаге может двигаться в одном из четырёх направлений в клетку, соседнюю по стороне. Муравей движется согласно следующим правилам:

На чёрном квадрате – повернуть на 90° влево, изменить цвет квадрата на белый, сделать шаг вперед на следующую клетку.

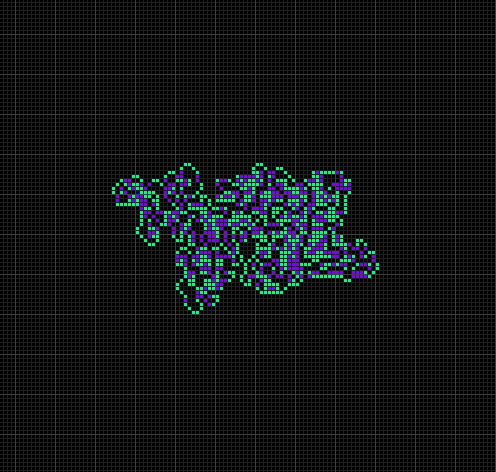
На белом квадрате – повернуть на 90° вправо, изменить цвет квадрата на чёрный, сделать шаг вперед на следующую клетку.

Эти простые правила вызывают довольно сложное поведение: после некоторого периода довольно случайного движения муравей, видимо, начинает непременно строить дорогу из 104 шагов, повторяющуюся бесконечно, независимо от изначальной раскраски поля. Это наводит на мысль, что «магистральное» поведение является стабильным [аттрактором](https://ru.wikipedia.org/wiki/%D0%90%D1%82%D1%82%D1%80%D0%B0%D0%BA%D1%82%D0%BE%D1%80) муравья Лэнгтона.

Муравей Лэнгтона также может быть описан как [клеточный автомат](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B5%D1%82%D0%BE%D1%87%D0%BD%D1%8B%D0%B9_%D0%B0%D0%B2%D1%82%D0%BE%D0%BC%D0%B0%D1%82), в котором почти всё поле покрашено в чёрно-белый цвет, а клетка с «муравьём» имеет один из восьми различных цветов, кодирующих соответственно все возможные комбинации чёрного/белого цвета клетки и направления движения муравья. На рисунке 1.7 изображён хаотичный рост муравья Лэнгтона.

Существует простое расширение муравья Лэнгтона, в котором используется более двух цветов клеток. Цвета изменяются циклически. Для таких муравьев существует также простая форма названия: для каждого следующего цвета используется буква L или R (Л и П), в зависимости от того, поворачивает ли муравей направо или налево. Таким образом, муравей Лэнгтона – это муравей RL.

Некоторые из этих обобщенных муравьев Лэнгтона рисуют узоры, которые становятся все более [симметричными](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BC%D0%BC%D0%B5%D1%82%D1%80%D0%B8%D1%8F). Один из простых примеров – муравей RLLR. Одно достаточное условие этого заключается в том, что имя муравья, рассматриваемое как циклический список, состоит из последовательных пар повторяющихся букв LL или RR (цикличность списка означает, что последняя буква может спариваться с первой) [8].

  
Рисунок 1.7 – Муравей Лэнгтона

* + 1. ***Вероятностная машина Тьюринга***

Вероятностная машина Тьюринга представляет собой [детерминированную машину Тьюринга](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%B0_%D0%A2%D1%8C%D1%8E%D1%80%D0%B8%D0%BD%D0%B3%D0%B0), имеющую дополнительно аппаратный источник случайных битов, любое число которых, например, она может «заказать» и «загрузить» на отдельную ленту и потом использовать в вычислениях обычным для [МТ](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%B0_%D0%A2%D1%8C%D1%8E%D1%80%D0%B8%D0%BD%D0%B3%D0%B0) образом [9].

***1.3.4 Универсальная машина Тьюринга***

Универсальной машиной Тьюринга называют [машину Тьюринга](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%B0_%D0%A2%D1%8C%D1%8E%D1%80%D0%B8%D0%BD%D0%B3%D0%B0), которая может заменить собой любую машину Тьюринга. Получив на вход программу и входные данные, она вычисляет ответ, который вычислила бы по входным данным машина Тьюринга, чья программа была дана на вход [10].

* 1. **Обзор систем–аналогов**

В настоящее время существует достаточно много реализаций МТ. Рассмотрим возможности некоторых из них

* + 1. ***Машина Поста***

Тренажёр «Машина Поста» – это учебная модель универсального исполнителя (абстрактной вычислительной машины), основанного на работах [Э.Л. Поста](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%81%D1%82,_%D0%AD%D0%BC%D0%B8%D0%BB%D1%8C_%D0%9B%D0%B5%D0%BE%D0%BD) по уточнению понятия алгоритма [11]. Согласно тезису Поста, любой алгоритм может быть записан в виде программы для машины Поста. В качестве примера рассмотрим программу К. Полякова (рисунок 1.8).

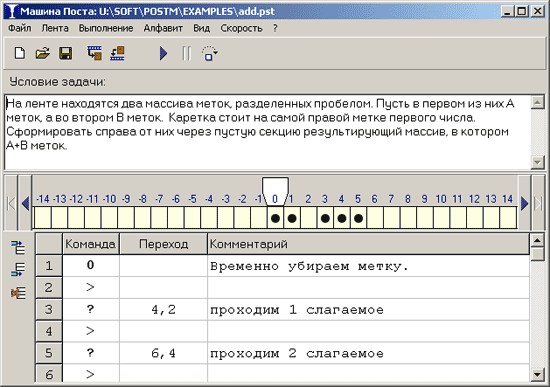


Рисунок 1.8 – Экранная форма тренажёра «Машина Поста»

Машина Поста состоит из каретки (считывающей и записывающей головки) и бесконечной ленты, разбитой на ячейки. Каждая ячейка ленты может быть либо пустой («0»), или содержать метку («1»).

Программа состоит из пронумерованных строк. В каждой строке записывается одна из следующих команд:

> N переместить каретку вправо на 1 ячейку и перейти к строке с номером N;

< N переместить каретку влево на 1 ячейку и перейти к строке с номером N;

0 N записать в текущую ячейку «0» (стереть метку) и перейти к строке с номером N;

1 N записать в текущую ячейку «1» (поставить метку) и перейти к строке с номером N;

? N, M если текущая ячейка содержит «0» (не отмечена), то перейти к строке с номером N, иначе перейти к строке M;

. остановить программу.

Номер строки перехода в командах >, <, 0 и 1 можно не указывать, при этом происходит переход к следующей строке.

Для завершения работы программы достаточно сделать переход на строку 0, например, так:

? 25, 0 остановить программу, если текущая ячейка содержит «1», иначе перейти к строке 25.

* + 1. ***«Algo2000»***

«Algo2000» – интерпретатор машины Поста и машины Тьюринга [12]. Проверка правильности составления алгоритма вычисления функции с помощью компьютера полностью исключает возможность ошибки при работе по программе на ленте. При этом пользователь имеет возможность регулировать скорость выполнения программы и видеть каждый шаг машины при обработке числа, что позволяет своевременно вносить изменения в программу при обнаружении ошибки. Программа имеет удобный интерфейс, предусмотрена возможность сохранения составленных программ, поддерживаются различные внешние алфавиты машины Тьюринга, а также имеется справочник пользователя (рисунок 1.9).

Программа имеет умеренные требования: компьютер IBM PC AT 486 и выше, наличие операционной системы Windows'95/98/NT.

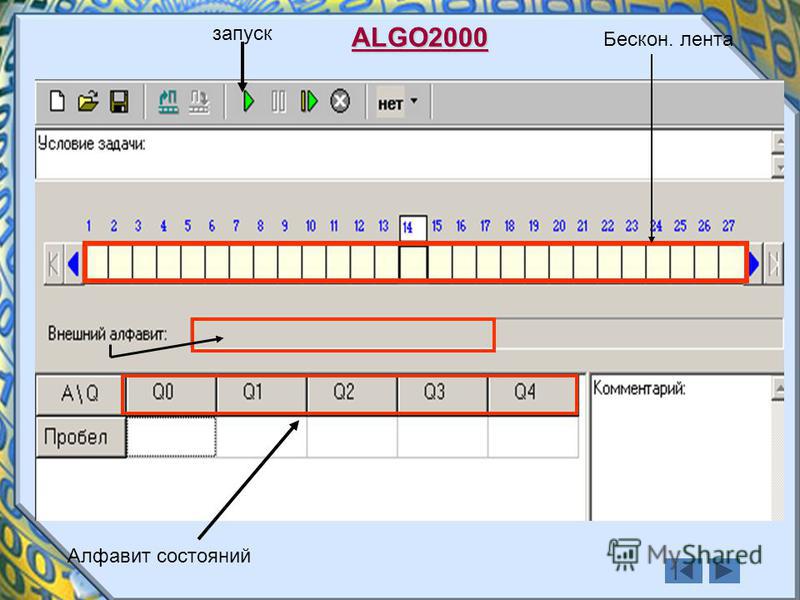


Рисунок 1.9 – Экранная форма программы «Algo2000»

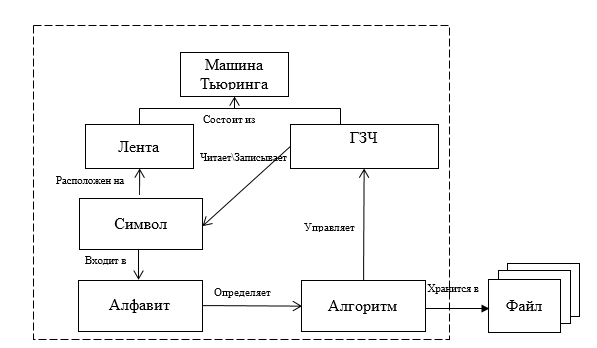
* 1. **Диаграмма объектов предметной области**

Объектно-ориентированная декомпозиция представляет собой разбиение системы на сущности, являющиеся какими-либо объектами, действующими в той ситуации, которую как раз и моделирует система.

Объект – элемент системы, объединяющий в себе данные и операции над ними, обладающий свойствами наследования, инкапсуляции и полиморфизма [13].

Объектная декомпозиция может применяться многократно, или быть многоуровневой. Это значит, что каждый объект может рассматриваться как система, которая состоит из элементов, взаимодействующих друг с другом. При многоуровневой декомпозиции на каждом уровне мы получаем объекты с более простым поведением, что позволяет разрабатывать системы повышенной сложности по частям.

На рисунке 1.10 приведена диаграмма объектов данной предметной области.

Рисунок 1.10 – Диаграмма объектов предметной области

Как видно из диаграммы, машина Тьюринга представляет собой объединение ленты и ГЗЧ, которая выполняет определённый алгоритм, который определяется алфавитом, символы которого располагаются на ленте. Кроме этого, машину Тьюринга и алгоритм можно сохранить в файл или загрузить из него.

* 1. **Постановка задачи**

Перед авторами поставлена задача – разработать систему моделирования работы машины Тьюринга.

Перед началом моделирования пользователь может либо создать собственный (новый) алгоритм, либо загрузить стандартный (в программе должна быть реализация двух алгоритмов: сложения и НОД), либо из файла.

Алгоритм должен быть представлен в табличном виде (см. рисунок 1.3), число строк в таблице фиксировано (ограничено алфавитом), количество столбцов (состояний) пользователь может менять (в пределах от 1 до 100). Пользователь должен иметь возможность добавлять/удалять любые состояния. Чтобы добавить новое состояние, пользователь должен выбрать какое количество столбцов ему необходимо добавить. Эти столбцы добавятся в конец. Для удаления состояния следует выделить соответствующий столбец. Команды управления МТ должны располагаться в ячейках таблицы. Семантический и синтаксический анализы команд на данном этапе не проводятся. Пользователь должен иметь возможность сохранить алгоритм в файл.

В системе должен быть предусмотрен алфавит «По умолчанию» (E\*10=), но при необходимости пользователь должен иметь возможность его изменить (максимальное количество символов в алфавите – 100). Перед запуском МТ пользователю необходимо задать значения операндов либо в цифровом виде (в десятичной системе счисления), либо в символьном (непосредственно на ленте), а также выбрать один из режимов работы (пошаговый, автоматический, мгновенный). При автоматическом способе пользователь должен задать скорость выполнения с помощью ползунка в пределах от 15 мс до 500 мс. В процессе выполнения должна подсвечиваться текущая ячейка ленты и команда управления МТ.

В программе должна быть возможность получения трассы с возможностью записи её в текстовый файл.

После запуска МТ будут произведены синтаксический и семантический анализы введённых команд, при ошибке пользователь должен быть уведомлён об этих ошибках. Если ошибок нет, то МТ начнёт свою работу в соответствии с выбранным режимом работы.

В системе также должна быть обеспечена возможность получения справочной информации как о самой системе, так и о представляемых ею возможностях.

Таким образом, система должна выполнять следующие функции:

* + визуализация процесса работы МТ;
  + синтаксический анализ алгоритма;
  + семантический анализ алгоритма;
  + выдача справочной информации о системе;
  + выбор способа задания операндов;
  + редактирование символа на ленте;
  + выбор режима работы МТ;
  + выбор скорости визуализации работы МТ
  + создание алфавита;
  + редактирование алфавита;
  + выбор стандартного алгоритма;
  + ручное составление алгоритма;
  + запись алгоритма в файл;
  + загрузка алгоритма из файла;
  + сохранение трассы выполняемого алгоритма;
  + сохранение трассы в файл.

1. ПРОЕКТИРОВАНИЕ СИСТЕМЫ
   1. Структурная схема системы

Под системой понимается единство целей, ресурсов и строения (структуры), свойства которого определяются отношение входящих в рассматриваемую совокупность объектов (элементов).

*Структурная схема* – это совокупность элементарных звеньев объекта и связей между ними, один из видов графической модели, которая разрабатывается на начальных стадиях проектирования. Она определяет основные функциональные части, назначение и взаимосвязи между ними [14].

Структурная схема системы приведена на рисунке 2.1.

  
Рисунок 2.1 – Структурная схема системы

В состав системы входят следующие подсистемы:

1. *файловая подсистема*, позволяющая сохранять и загружать алгоритмы;
2. *подсистема составления алгоритма*, которая отвечает за составление алгоритма;
3. *подсистема настройки параметров,* позволяющая задавать алфавит и количество состояний*;*
4. *подсистема моделирования работы МТ*, которая проводит синтаксический и семантические анализы, а также выполняет команды;
5. *подсистема настройки параметров моделирования,* которая позволяет настраивать режим работы и способ задания операндов;
6. *справочная подсистема*, которая выдает сведения о системе (руководство пользователя) и об ее разработчиках;
7. *подсистема визуализации*, которая отображает процесс выполнения алгоритма.
   1. Спецификация системы

Спецификация требований к ПО(SRS) – процесс формализованного описания функциональных и нефункциональных требований, требований к характеристикам качества в соответствии со стандартом качества ISO/IEC 9126-94, которые будут отрабатываться на этапах ЖЦ ПО [15].

* + 1. Функциональная спецификация

Функциональная спецификация включает в себя перечень всех функций системы с привязкой их к конкретной подсистеме и к информационной среде (входные и выходные данные), перечень исключительных ситуаций и реакцию системы на их возникновение [15].

Функциональная спецификация системы приведена в таблице 2.1.

Таблица 2.1 – Перечень функций, выполняемых системой

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Название  подсистемы | Название функции | Информационная среда | | | |
| Входные данные | | Выходные данные | |
| Назначение (наименование) | Тип, ограничения | Назначение (наименование) | Тип, ограничения |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 1 Справочная | 1.1 Выдать сведения о разработчиках | Сведения о разработчиках системы (ФИО, номер группы) | Текст (МЕМО) | Визуальное отображение информации | – |
| 1.2 Выдать сведения о системе | Файл справки | Текстовый (\*.chm) |
| Код ошибки | Целое |
| 2 Составления алгоритма | 2.1 Ввести команду перехода | Количество состояний | Целое,1...100 | Команда | Строка |
| Алфавит | Массив символов |
| Команды управления МТ | R, L, S |

|  |  |  |  |
| --- | --- | --- | --- |
| Продолжение таблицы 2.1 | | | |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 3 Настройки параметров алгоритма | 3.1 Задать алфавит | Допустимый диапазон значений | Символьный, русские, латинские буквы, спец. символы | Алфавит | Массив символов |
| 3.2 Задать длину алфавита | Допустимый диапазон значений | Целое, 3..100 | Длина алфавита | Целое |
| 3.3 Восстановить исходный алфавит | – | – | Алфавит | Массив символов |
| 3.4 Задать число состояний | Допустимый диапазон значений | Целое, 1..100 | Число состояний | Целое |
| 4 Визуализации | 4.1 Выбрать режим работы | Список типов | * пошаговый * с задержкой * мгновенный | Режим работы | Перечислимый |
| 4.2 Выбрать режим визуализации | Список типов | * с окрашиванием * с двигающейся лентой | Режим визуализации | Перечислимый |

|  |  |  |  |
| --- | --- | --- | --- |
| Продолжение таблицы 2.1 | | | |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 4 Визуализации | 4.3 Выбрать скорость визуализации | Список вариантов расположения | * очень быстрая * быстрая * средняя * медленная * очень медленная | Скорость визуализации | Перечислимый |
| 5 Моделирования работы МТ | 5.1 Выполнить синтаксический анализ | Команда | Строка | Сообщение о наличии/отсутствии ошибок | Строка |
| 5.2 Выполнить семантический анализ | Алгоритм | Объект «Алгоритм» | Сообщение о наличии/отсутствии ошибок | Строка |
| 5.3 Составить трассу | Лента | Массив символов | Трасса | Массив |
| 6 Файловая | 6.1 Сохранить файл с алгоритмом | Алгоритм | Объект «Алгоритм» | Файл | Тип файла, структура определяется в ходе проектирования |
| Имя файла | Строка, \*.tma |

|  |  |  |  |
| --- | --- | --- | --- |
| Продолжение таблицы 2.1 | | | |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 6 Файловая | 6.2 Загрузить файл с алгоритмом | Имя файла | Строка, \*.tma | Алгоритм | Объект «Алгоритм» |
| Код ошибки | Целое |
| 6.3 Сохранить файл с трассой | Трасса | Объект «Трасса» | Файл | Тип файла, структура определяется в ходе проектирования |
| Имя файла | Строка, \*.tmt |

* + 1. Спецификация качества

Каждая система должна выполнять определенные функции, обладать рядом свойств, позволяющим успешно ее использовать в течение длительного периода, т.е. обладать определенным качеством.

Качество – это совокупность черт и характеристик, которые влияют на способность системы удовлетворять заданные потребности пользователей. В настоящее время критериями качества системы принято считать:

* формулируются требования по быстродействию;
* требования к разработке интерфейса и т.п.
* контроль ввода всех параметров в системе должен осуществляться с привязкой к диапазонам допустимых значений.
* должна быть реализована обработка исключительных ситуаций.
  + 1. Перечень исключительных ситуаций

*Исключительная ситуация* – это ситуация, при которой система не может выполнить возложенных на нее функций или которая может привести к денормализации работы системы [15].

В таблице 2.2 приведен перечень исключительных ситуаций для разрабатываемой системы и описаны реакции системы на их возникновение.

* 1. Разработка прототипа интерфейса пользователя системы

*Пользовательский интерфейс* – совокупность элементов, позволяющих пользователю управлять работой программы или вычислительной системы и получать требуемые результаты. Разработка пользовательского интерфейса включает следующие основные этапы:

* постановка задачи – определение типа интерфейса и общих требований к нему;
* анализ требований и определение спецификаций – определение сценариев использования и пользовательской модели интерфейса;
* проектирование – проектирование диалогов и их реализация в виде процессов ввода-вывода;
* реализация – программирование и тестирование интерфейсных процессов.

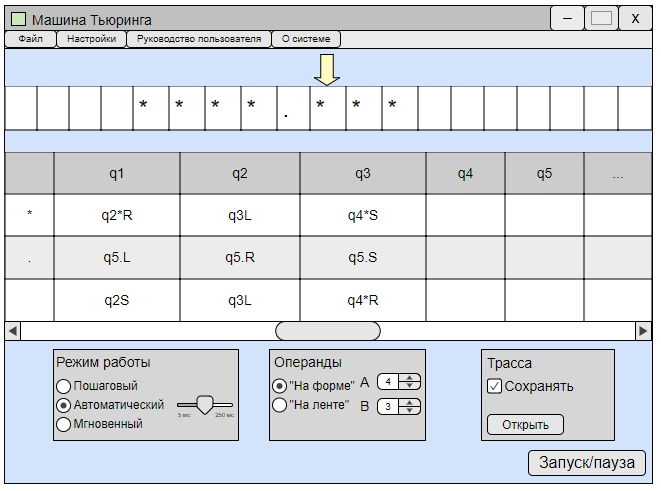
Таблица 2.2 – Перечень исключительных ситуаций

|  |  |  |
| --- | --- | --- |
| Название подсистемы | Название исключительной ситуации | Реакция системы |
| 1 Справочная | 1.1 Невозможно открыть файл справки | Выдача сообщения «Файл справки поврежден» |
| 1.2 Невозможно найти файл справки | Выдача сообщения «Отсутствует файл справки» |
| 2 Файловая | 2.1 Невозможно открыть файл алгоритма | Выдача сообщения «Файл алгоритма поврежден» |
| 3 Моделирования | 3.1 Найдены семантические ошибки | Выдача сообщения «Семантическая ошибка: <указатель на ошибку>» |
| 3.2 Найдены синтаксические ошибки | Выдача сообщения «Синтаксическая ошибка: <указатель на ошибку>» |

При проектировании пользовательских интерфейсов необходимо учитывать психофизические особенности человека, связанные с восприятием, запоминанием и обработкой информации [16].

Прототип интерфейса пользователя представлен на рисунках 2.2-2.5.

На рисунке 2.2 представлен прототип главной экранной формы, на которой должны располагаться: лента, таблица переходов, параметры режима работы, режим визуализации, операнды и их режим задачи, сохранение и открытие трассы.

Рисунок 2.2 – Прототип главной экранной формы

Операнды можно задать как на форме, так и непосредственно на ленте, для этого следует выбрать соответствующий переключатель в панели «Операнды».

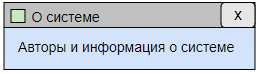
Если есть необходимость вести файл трассы, то нужно выбрать соответствующую кнопку «Сохранять» на панели «Трасса». После выполнения алгоритма трассу можно открыть соответствующей кнопкой.

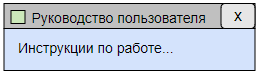
Прототип экранной формы настроек представлен на рисунке 2.3, где можно выбрать такие параметры как: длина алфавита, длина ленты, количество состояний, а также должна быть реализована возможность создать свой алфавит.

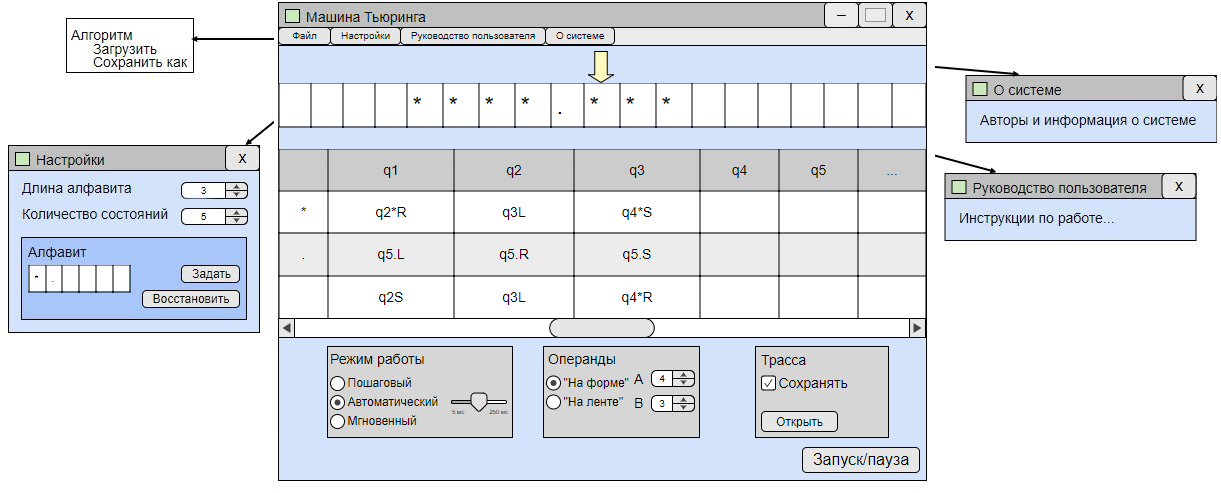
На рисунке 2.4 и рисунке 2.5 представлены формы «О системе» и «Руководство пользователя» соответственно.

На рисунке 2.6 представлена навигационная модель приложения.

  
Рисунок 2.3 – Прототип экранной формы «Настройки»

  
Рисунок 2.4 – Прототип экранной формы «О системе»

  
Рисунок 2.5 – Прототип экранной формы «Руководство пользователя»

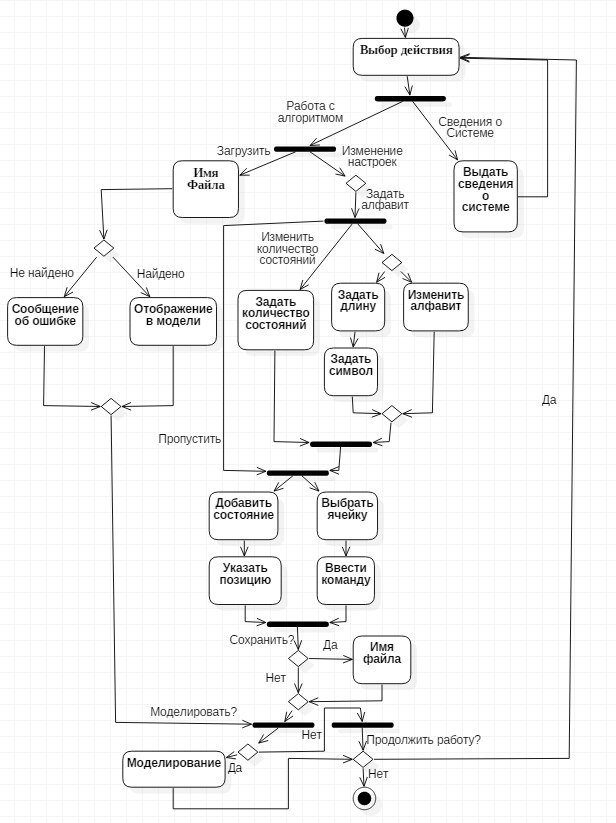
  
Рисунок 2.6 – Навигационная модель приложения

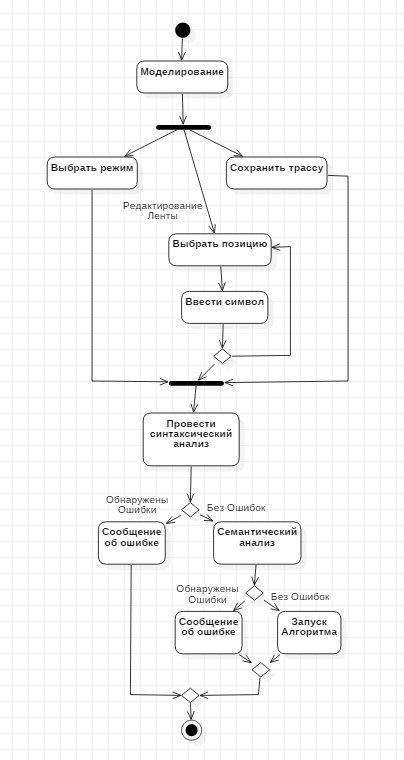
* 1. Разработка и описание алгоритмов обработки данных

Понятие алгоритма обработки данных используется для описания метода решения задачи, который в дальнейшем возможно реализовать в выбранной среде программирования [17].

Алгоритм – набор инструкций, описывающих порядок действий исполнителя для достижения некоторого результата [18].

Диаграмма деятельности при работе с программой приведена на рисунках 2.7 и 2.8.

  
Рисунок 2.7 – Диаграмма деятельности «Работа с программой»

  
Рисунок 2.8 – Диаграмма деятельности «Моделирование работы МТ»

При выборе пункта «Файл» появляется возможность сохранить или загрузить алгоритм из файла.

При выборе пункта «Работа с алгоритмом» первоначально необходимо задать алфавит. Для этого следует указать длину алфавита и допустимые символы. Кроме задания алфавита можно изменить количество состояний (по умолчанию 3) путем добавления по одному, или указать количество, а также задать команды. Для задания команды нужно выбрать ячейку, в которой она находится и изменить её.

Пункт «Параметры моделирования» отвечает за возможность выбора режима работы программы, а также позволяет сохранять трассу программы в файл.

Вся информация о системе может быть получена при выборе соответствующих пунктов.

* 1. Выбор и обоснование комплекса программных средств
     1. Выбор языка программирования и среды разработки

Выбранным языком программирования является C#. Это язык программирования для разработки приложений, предназначенных для выполнения в среде .NET Framework. Язык C# прост, типобезопасен и объектно-ориентирован. Благодаря множеству нововведений C# обеспечивает возможность быстрой разработки приложений.

В среде .NET можно разрабатывать практически любое приложение для запуска в Windows. Например, с применением C# и .NET Framework можно создавать динамические веб-страницы, приложения Windows Presentation Foundation, веб-службы XML, компоненты для распределенных приложений, компоненты для доступа к базам данных, классические настольные приложения Windows и даже клиентские приложения нового интеллектуального типа, обладающие возможностями для работы в оперативном и автономном режимах [19].

* + 1. Выбор операционной системы

В качестве операционной системы была выбрана система MS Windows 10. Windows 10 – операционная система семейства Windows NT, производства корпорации Microsoft. Она поддерживает MS Visual Studio 2017 и предоставляет разработчикам все необходимые средства для создания качественного приложения.

* + 1. Выбор среды программирования

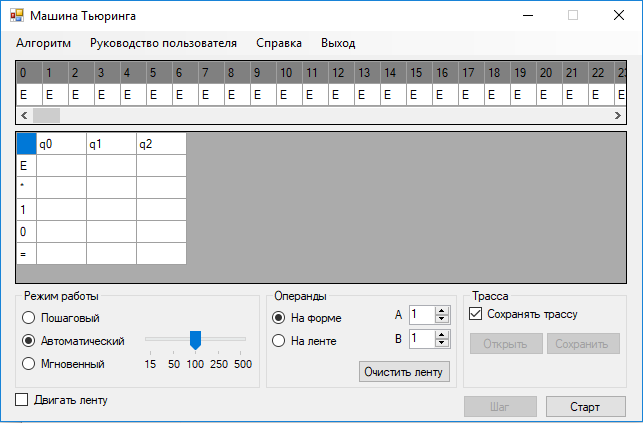
В качестве среды разработки была выбрана MS Visual Studio 2017. Microsoft Visual Studio предоставляет несколько способов разработки приложений Windows, работающих локально на компьютерах пользователей. Преимущество использования Visual Studio заключается в том, что эта среда предоставляет средства, благодаря которым процесс разработки приложений становится намного быстрее, проще и надежнее [20].

В Visual Studio 2017 есть все необходимое для создания отличных приложений для устройств, настольных приложений, веб-приложений и облачных решений. Обладает преимуществами простой навигации по коду, быстрой сборки и развертывания в кратчайшие сроки. Visual Studio повышает продуктивность и позволяет удобно работать самостоятельно и в составе команды [21].

1. реализация СИСТЕМЫ
   1. Разработка и описание интерфейса пользователя

Интерфейс пользователя – совокупность средств и методов, при помощи которых пользователь взаимодействует с различными, чаще всего сложными, машинами, устройствами и аппаратурой [22].

После запуска системы откроется главная форма (рисунок 3.1).

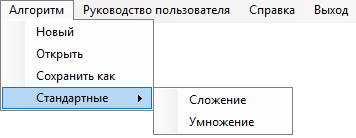
  
Рисунок 3.1 – Главное окно программы

* + 1. Разработка и описание пользовательского меню

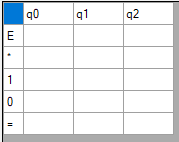
Меню – элемент интерфейса пользователя, позволяющий выбрать одну из нескольких перечисленных опций программы [23].

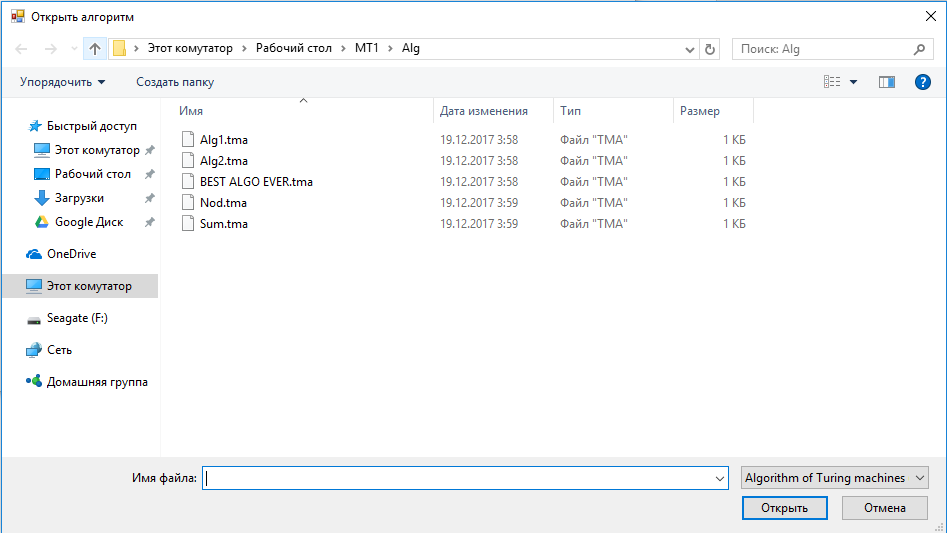
На рисунке 3.2 приведена структура меню. В меню верхнего уровня включены следующие пункты: «Алгоритм», «Руководство пользователя», «Справка», «Выход». По нажатию на пункт «Алгоритм» можно выбрать одну из трёх функций: новый алгоритм (рисунок 3.3), открыть алгоритм (рисунок 3.4), сохранить как алгоритм (рисунок 3.5) и одного раздела – стандартные. При наведении на этот раздел появится возможность выбрать алгоритм сложения (рисунок 3.6) или умножения (рисунок 3.7).

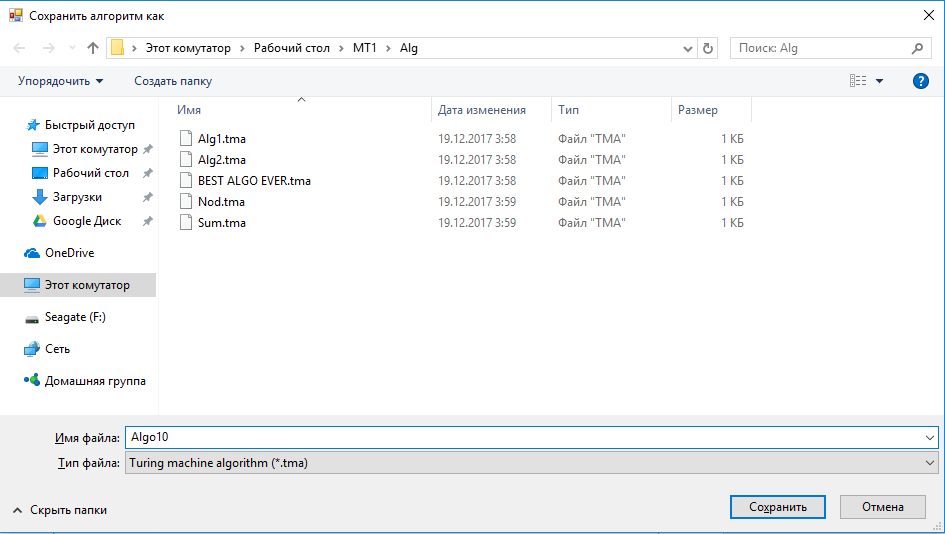
Рисунок 3.2 – Структура меню

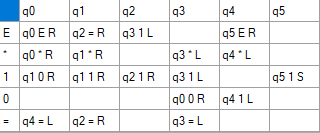


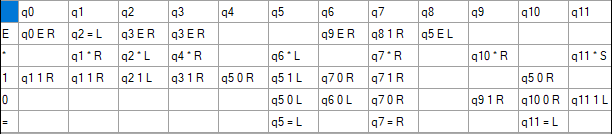
По нажатию на пункт меню «Руководство пользователя» откроется файл формата .chm для получения подробной информации о программе (рисунок 3.8). При выборе пункта «Справка» можно посмотреть сведения о разработчиках. Пункт «Выход» закрывает программу.

  
Рисунок 3.3 – Новый алгоритм

  
Рисунок 3.4 – Окно выбора алгоритма

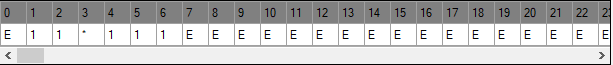
  
Рисунок 3.5 – Окно сохранения алгоритма

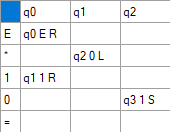
  
Рисунок 3.6 – Стандартный алгоритм сложения

  
Рисунок 3.7 – Стандартный алгоритм умножения

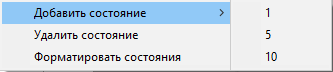
* + 1. ***Работа с алгоритмом***

Основными элементами формы являются «Лента» (рисунок 3.8) и «Правила перехода» (рисунок 3.9) или «Алгоритм».

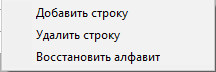
  
Рисунок 3.8 – Лента МТ

  
Рисунок 3.9 – Правила перехода МТ

При нажатии правой кнопкой мыши на состояние появится контекстное меню состояния (рисунок 3.10) с трёмя пунктами: «Добавить состояние», при выборе его подпунктов в конец добавятся нужное количество состояний, «Удалить состояние», при нажатии на который текущее состояние удалится и «Форматировать состояния», который переименует все состояния в следующий вид: q0, q1, q2, …, qn.

  
Рисунок 3.10 – Контекстное меню состояния

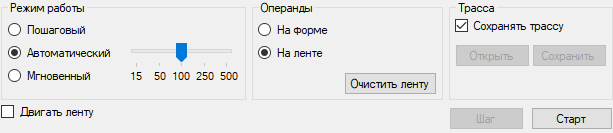
При нажатии правой кнопкой мыши на алфавит появится контекстное меню алфавита (рисунок 3.11) с трёмя пунктами: «Добавить строку», который добавляет целую строку, «Удалить строку», при выборе которого текущая строка удаляется и «Восстановить алфавит», при нажатии на который алфавит заменится на стандартный.

  
Рисунок 3.11 – Контекстное меню алфавита

В ячейки на пересечении алфавита и состояний можно записывать правила (рисунок 3.9)

* + 1. ***Настройка параметров***

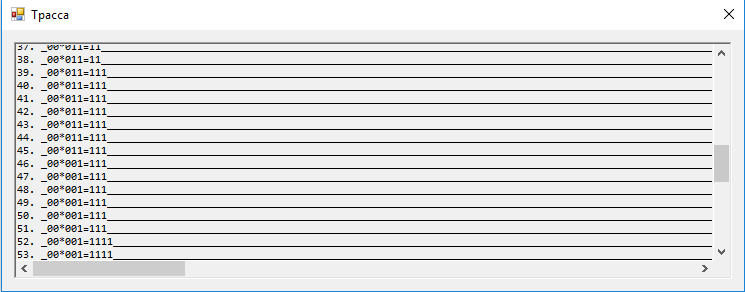
В программе предусмотрена настройка параметров системы (рисунок 3.12)

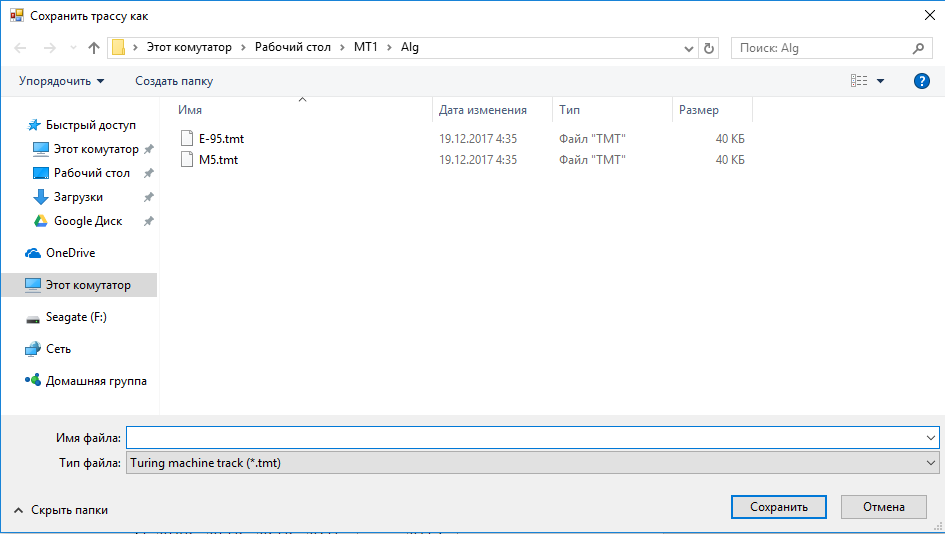
  
Рисунок 3.12 - Панели настройки параметров системы

В панели «Режим работы» есть возможность выбора одного из трёх режимов работы: «Пошаговый», при котором МТ совершает 1 шаг после нажатия кнопки «Шаг», «Автоматический», при котором МТ работает с определённой задержкой (15мс, 50мс, 100мс, 250мс, 500мс) и «Мгновенный», при выборе которого визуализация не предусмотрена, а скорость работы очень высока (более 9000 итераций в секунду).

На панели «Операнды» можно выбрать каким способом будет произведено задание операндов: «На форме» или «На ленте». Также присутствует кнопка, которая очищает ленту, т.е. заменяет всё её символы на пустые («E» по умолчанию).

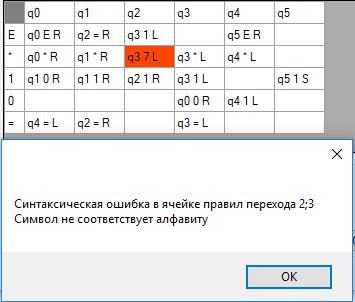
В панели «Трасса» существует переключатель, позволяющий сохранять трассу и две кнопки: «Открыть», которая открывает трассу в текстовом представлении (рисунок 3.13) и «Сохранить», которая сохраняет трассу в текстовый файл (рисунок 3.14).

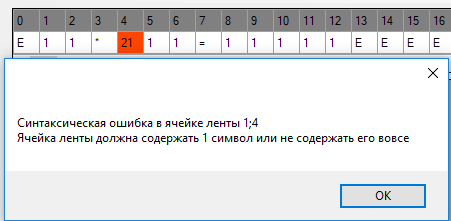
  
Рисунок 3.13 – Окно открытия трассы

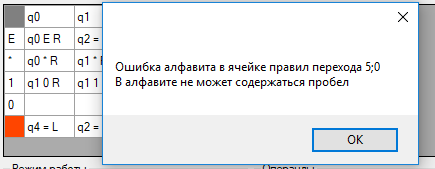
  
Рисунок 3.14 – Окно сохранения трассы

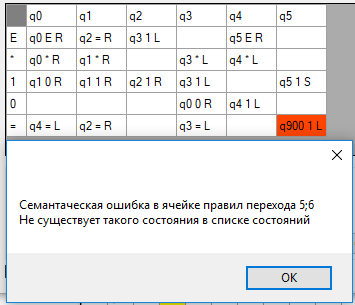
* + 1. ***Моделирование работы МТ***

После настройки всех параметров необходимо нажать кнопку «Старт», после чего произойдёт синтаксический и семантический анализы введённого алгоритма. Примеры ошибок приведены на рисунках 3.15-3.18

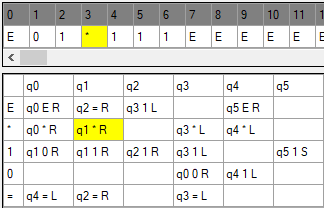
  
Рисунок 3.15 – Синтаксическая ошибка алгоритма

  
Рисунок 3.16 – Синтаксическая ошибка ленты

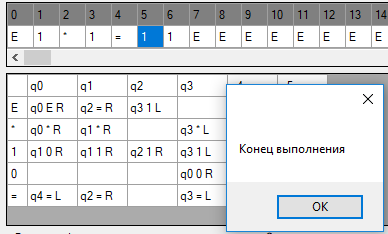
  
Рисунок 3.17 – Синтаксическая ошибка алфавита

  
Рисунок 3.18 – Семантическая ошибка алгоритма

После прохождения всех проверок в зависимости от выбранного режима и настроек МТ начнёт свою работу. На рисунке 3.19 приведен её пример.

  
Рисунок 3.19 – Визуализация работы МТ

Когда МТ перейдёт в финальное состояние «S», появится уведомление о завершении моделирования (рисунок 3.20).

  
Рисунок 3.20 – Окончание моделирования МТ

* 1. **Реализация классов**

Основные параметры и функции классов описаны в таблицах 3.1-3.5. Диаграмма сущностных классов представлена на рисунке 3.21.

Таблица 3.1 – Реализация класса «ViewTrack»

|  |  |
| --- | --- |
| Функции | private string toN – приведение к нужному числу цифр |
| public void delTrack – удаление трассы |
| public void addTrack – добавление в трассу |

Таблица 3.2 – Реализация класса «dgv»

|  |  |
| --- | --- |
| Функции | public static DataGridView StatesStart – иницаиализация работы переходов |
| public static DataGridView RibbonStart – инициаоизация работы ленты |
| public static DataGridView StatesFormat – приведенение к стандартному формату после изменений |
| public static DataGridView RibbonFormat – приведение к стандартному формату после изменений |
| public static DataGridView toWhite – обеление всех закрашенных ячеек в разные цвета |
| public static DataGridView RibbonClear – очищение ленты |
| public static DataGridView RibbonReadOnly – лента только для чтения |

Таблица 3.3 – Реализация класса «File»

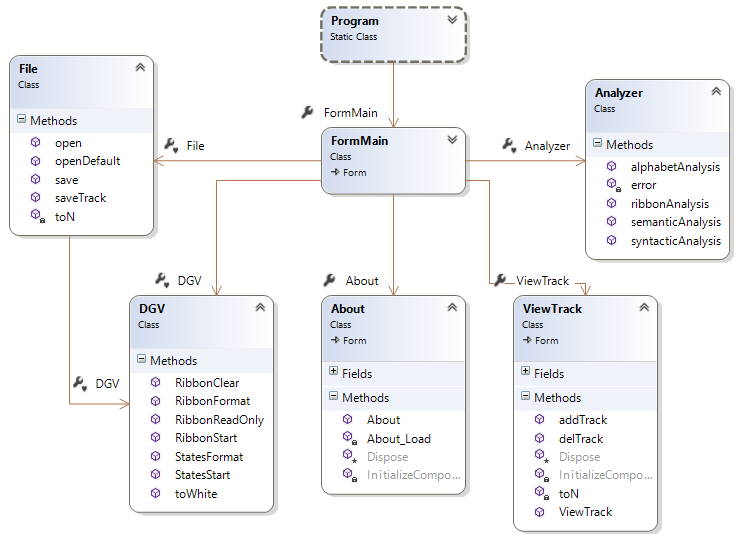
|  |  |
| --- | --- |
| Функции | private static string toN – приведение к нужному числу цифр |
| public static DataGridView openDefault – открытие стандартного алгоритма |
| public static DataGridView open – открытие алгоритма |
| public static void save – сохранение алгоритма |
| public static void saveTrack – сохранение трассы |

Таблица 3.4 – Реализация класса «Analyzer»

|  |  |
| --- | --- |
| Функции | private static void error – выведение ошибок |
| public static bool syntacticAnalysis – синтаксический анализ |
| public static bool semanticAnalysis – семантический анализ |
| public static bool ribbonAnalysis – анализ ленты |
| public static bool alphabetAnalysis – анализ алфавита |

Таблица 3.5 – Реализация класса «Main»

|  |  |
| --- | --- |
| Параметры | private string alphabet – текущий алфавит |
| private string defaultAlphabet – стандартный алфавит |
| private int operatingMode – режим работы |
| private int speed – скорость |
| private bool operandsModeForm – способ задания операндов |
| private int operand – операнд А |
| private int operand – операнд Б |
| private bool saveTrack – сохранять ли трассу |
| Private List<string> fullTrack – трасса |
| private int currentMouseOverRow – положение мыши в dgv |
| private int currentMouseOverColumn – положение мыши в dgv |
| private int addCount – сколько состояний добавить |
| private string[] states – список состояний |
| private int currentRibonIndex – индекс текущего символа на ленте |
| private string currentRibonSymbol – текущий символ на ленте |
| private int currentStateIndex – индекс текущего состояния |
| private string currentState – текущее состояние |
| private int currentAlphabetIndex – текущий индекс алфавита |
| private int colorIndexRibon – ячейка для перекрашивания трассы |
| private int colorIndexStates1 – для перекрашивания ленты |
| private int colorIndexStates2 – для перекрашивания ленты |
| private int shift – сдвиг ленты |
| Функции | private void blockOnStart() – блокирование элементов формы |
| private void unblockOnFinish() – разблокирование элементов формы |
| private void start – дейтвия на старте |
| private bool step – 1 итерация МТ |

  
Рисунок 3.21 – Диаграмма классов

* 1. Диаграмма модулей
  2. Выбор и обоснование комплекса технических средств

Выбор комплекса технических средств в значительной степени определяет функциональные возможности и надежность создаваемой программы. Выбор комплекса технических средств производится на основе перечня решаемых задач и их информационных характеристик, спецификой обрабатываемой информации, техническими возможностями оборудования и т.д.

* + 1. Расчёт объёма занимаемой памяти

*Расчет объема внешней памяти*

Для того чтобы сделать и обосновать выбор комплекса технических средств, необходимо провести расчет объема занимаемой памяти (жесткого диска и ОЗУ).

Для расчета необходимого объема свободной внешней памяти, необходимой для функционирования системы, воспользуемся следующей формулой:

VЖД= VОС + VПР+VД,

где VОС – объем памяти, занимаемый операционной системой (операционная система Windows 10, VОС = 20 Гб);

VПР – объем памяти, занимаемый непосредственно файлами приложения (VПР = 43 Кб);

VД – объекм памяти, занимаемый стандартными алгоритмами и файлом справки (VД = ОБЪЁМ ФАЙЛА СПРАВКИ +1КБ)

Таким образом, суммарный объем внешней памяти составит:

VЖД = 20 Гб + 43 Кб +… ≈ 20 Гб.

*Расчет объема ОЗУ*

Для расчета необходимого объема ОЗУ воспользуемся следующей формулой:

VОЗУ = VОС + VПР,

где VОС – ОЗУ, занимаемое операционной системой (1,6 Гб);

VПР – ОЗУ, которое займет само приложение (не превысит 400 Мб);

Суммарные объемы ОЗУ составит:

VОЗУ = 1,6 Гб + 400 Мб ≈ 2 Гб.

* + 1. Минимальные требования, предъявляемые к системе

Для корректного функционирования системы необходимо:

1) монитор с разрешением 1024x768;

2) манипулятор – мышь, клавиатура;

3) объем оперативной памяти – не менее 2 Гб;

4) объем свободного дискового пространства – не менее 1 Мб.

**ЗАКЛЮЧЕНИЕ**

В рамках лабораторного практикума была разработана система моделирования работы машины Тьюринга, которая позволяет пользователю создавать алгоритмы и проверять их работоспособность на разных алфавитах в разных режимах.

В первом разделе была изучена структура МТ, были найдены аналоги ее программной реализации, разобраны некоторые алгоритмы и сформулирована постановка задачи.

На этапе проектирования системы были разработаны структурная схема и спецификация системы, продуман и проиллюстрирован прототип интерфейса пользователя, выбран и обоснован комплекс технических и программных средств.

На этапе реализации приведено описание интерфейса системы, описана диаграмма классов, рассчитан объем занимаемой памяти и сформулированы требования, предъявляемые к системе.

Данная программа будет полезна студентам и людям, которые интересуются конечными автоматами.

**СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ**

1. Теория алгоритмов [Электронный ресурс] // StudFiles: файловый архив студентов. 2014-2017. URL: [https://studfiles.net/preview/ 5903677/](https://studfiles.net/preview/%205903677/) (дата обращения:20.10.2017).
2. О.Н. Граничин, С.Л. Молодцов. Создание гибридных сверхбыстрых компьютеров и системное программирование. СПб.:СПБГУ, 2006. С. 16.
3. Методология RAD [Электронный ресурс] // CITForum: библиотека свободно доступных материалов по информационным технологиям. 1996-2017. URL: <http://citforum.ru/database/case/glava1_3_2.shtml> (дата обращения:20.10.2017).
4. А.А. Шевченко. Концепции моделирования: технологии и решения. // Прикладная информатика. №6. 2006. URL: https://cyberleninka.ru/article/n/kontseptsii-modelirovaniya-tehnologii-i-resheniya/ (дата обращения:20.10.2017).
5. Алфавит (формальный язык) [Электронный ресурс] // Википедия: электрон. энциклопедия. 2001-2017. URL: [https://ru.wikipedia.org/  
   wiki/Алфавит\_(формальный\_язык)](https://ru.wikipedia.org/wiki/Алфавит_(формальный_язык)) (дата обращения:20.10.2017).
6. Ю. Г. Карпов. Теория автоматов. СПб.: Питер, 2003. С. 195.
7. Машина Тьюринга [Электронный ресурс] // Википедия: электрон. энциклопедия. 2001-2017. URL: [https://ru.wikipedia.org/  
   wiki/Машина\_Тьюринга](https://ru.wikipedia.org/wiki/Машина_Тьюринга) (дата обращения:20.10.2017).
8. Муравей Лэнгтона [Электронный ресурс] // Википедия: электрон. энциклопедия. 2001-2017. URL: [https://ru.wikipedia.org/wiki/  
   Муравей\_Лэнгтона](https://ru.wikipedia.org/wiki/Муравей_Лэнгтона) (дата обращения:20.10.2017).
9. Вероятностная Машина Тьюринга [Электронный ресурс] // Википедия: электрон. энциклопедия. 2001-2017. URL: <https://ru.wikipedia.org/wiki/Вероятностная_машина_Тьюринга> (дата обращения:20.10.2017).
10. Универсальная Машина Тьюринга [Электронный ресурс] // Википедия: электрон. энциклопедия. 2001-2017. URL: <https://ru.wikipedia.org/wiki/Универсальная_машина_Тьюринга> (дата обращения: 20.10.2017).
11. Машина Поста [Электронный ресурс] // Методические материалы и программное обеспечение. 2008-2017. URL: http://kpolyakov.spb.ru/prog/post (дата обращения:21.10.2017).
12. Эмулятор Машины Поста (ALGO 2000) [Электронный ресурс] // Сеть блогов учителей ГБОУ СОШ № 585 Кировского района Санкт-Петербурга. 2012-2017. URL: http://teach.sc585.spb.ru/inf/2011/11/21/эмулятор-машины-поста-algo-2000/ (дата обращения:21.10.2017).
13. Основные принципы ООП: инкапсуляция, наследование, полиморфизм [Электронный ресурс] //Wiki, фэндом жизни сообщества. 2003-2020. URL: http://gos-it.wikia.com/wiki/ (дата обращения:21.10.2017).
14. Структурная схема [Электронный ресурс] // Википедия: электрон. энциклопедия. 2001-2017. URL: <https://ru.wikipedia.org/wiki/Структурная_схема> (дата обращения: 04.11.2017).
15. Принципы разработки учебных программ [Электронный ресурс] // StudFiles: файловый архив студентов. 2014-2017. URL: https://studfiles.net/preview/2114024 (дата обращения: 20.10.2017).
16. Психофизические особенности человека [Электронный ресурс] // TehProg электронное пособие. 2009-2010 URL: http://tehprog.ru/index.php\_page=lecture43.html 2009-2010
17. Ресурсная эффективность алгоритмов [Электронный ресурс] // Intuit НОУ «ИНТУИТ». 2003-2017. URL: http://www.intuit.ru/studies/courses/648/504/lecture/11461%2525253Fpage%2525253D3 (дата обращения: 20.10.2017).
18. Алгоритм [Электронный ресурс] // Википедия: электрон. энциклопедия. 2001-2017. URL: <https://ru.wikipedia.org/wiki/Алгоритм> (дата обращения: 20.10.2017).
19. Среда .NET [Электронный ресурс] // StudFiles: файловый архив студентов. 2014-2017. URL: https://studfiles.net/preview/5946738/ (дата обращения: 04.11.2017).
20. Общие сведения о приложениях для Windows [Электронный ресурс] // Microsoft: [сайт]. URL: https://msdn.microsoft.com/ru-ru/library/5b13a7k4(v=vs.100).aspx (дата обращения: 28.11.2017).
21. Вас приветствует Visual Studio 2013 [Электронный ресурс] // Microsoft: [сайт]. URL: https://msdn.microsoft.com/ru-ru/library/dd831853(v=vs.120).aspx (дата обращения:  
    28.11.2017).
22. Понятие пользовательского интерфейса [Электронный ресурс] // Студопедия: [сайт]. URL: http://studopedia.ru/4\_153814\_udalennie-banki-dannih.html (дата обращения: 20.10.2016 г.).
23. Меню (информатика) [Электронный ресурс]. – https://ru.wikipedia.  
    org/wiki/Меню\_(информатика) (дата обращения: 14.12.2017).

**Приложение А  
Руководство пользователя**

**А.1 Введение**

Настоящее руководство предназначено для использования в процессе работы с машиной Тьюринга (МТ).

Система моделирования машины Тьюринга предназначена для составления и моделирования работы алгоритмов машины Тьюринга.

Пользователи должны обладать базовыми навыками работы с персональным компьютером, уверенным знанием принципов работы с программным интерфейсом, реализованным в соответствии со стандартами человеко-машинного интерфейса для операционной системы семейства Windows.

**А.2 Условия применения**

Для функционирования системы необходимо:

1. IBM PC – совместимый компьютер;
2. не менее 2 Гб оперативной памяти;
3. объем свободного дискового пространства – не менее 1 Мб.;
4. дисплей с разрешением не менее 1024x768 пикселей;
5. клавиатура или иное устройство ввода;
6. мышь или иное манипулирующее устройство.

**А.3 Подготовка к работе**

На персональном компьютере должен быть установлен следующий перечень программного обеспечения:

1. -.Net Framework версии 4.5;
2. - Операционная система Windows 7 и новее.

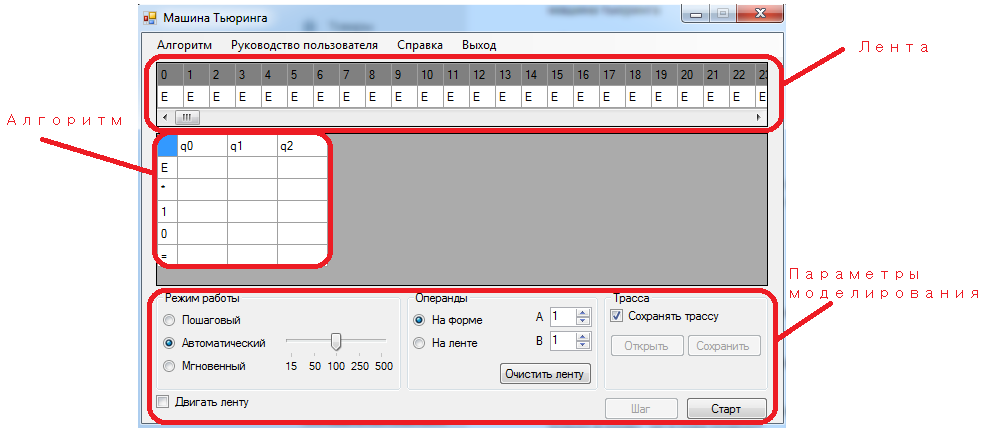
**А.4 Описание операций**

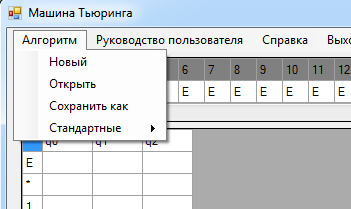
***А.4.1 Пользовательское меню***

После запуска системы на мониторе отображается главная форма (рисунок А.1), на которой отображаются алгоритм и панель инструментов работы с ним, лента, параметры моделирования.

Для перехода к работе с алгоритмами пользователю необходимо выбрать соответствующий пункт в верхней области (рисунок А.2).

Меню для работы с алгоритмами состоит из кнопок для создания нового алгоритма (Новый), загрузки существующего алгоритма (Открыть), сохранения текущего алгоритма в файл (Сохранить как), загрузки одного из стандартных алгоритмов: сложения и умножения.

  
Рисунок А.1 – Главная форма системы

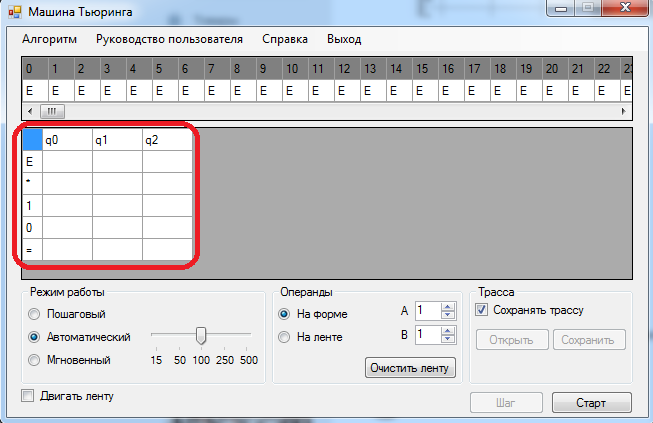
  
Рисунок А.2 – Панель навигации по шаблонам отчетов

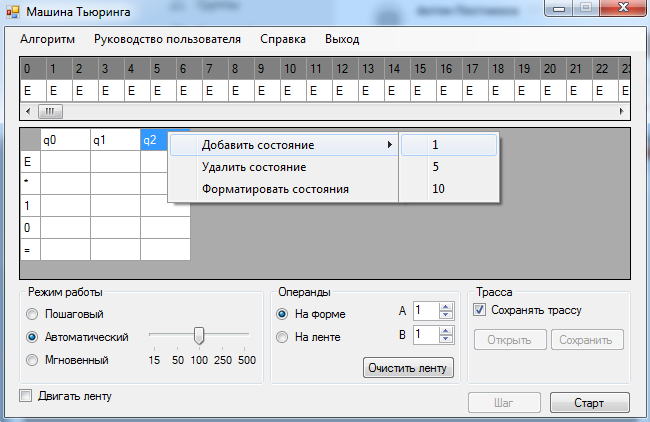
***A.4.2 Создание нового алгоритма***

При создании нового алгоритма, очищается таблица переходов и загружаются стандартные алфавит (строки) и состояния (столбцы) (рисунок А.3).

Для изменения алфавита или состояния необходимо нажать на ячейку крайнего левого столбца для изменения алфавита, и верхней крайней строчки для изменения состояний. Добавление состояний производится нажатием правой кнопки мыши по таблице и выбора одного из предложенных вариантов (рисунок А.4).

Для добавления или изменения команды необходимо кликнуть на ячейку левой кнопкой мыши по ячейке на пересечении состояния и алфавита, а затем написать команду в формате «qN Al C», где qN-номер состояния,   
Al-символ из алфавита, который будет поставлен в ячейку на ленте,   
C – команда движения (может быть R(вправо), L(влево), S(на месте)).

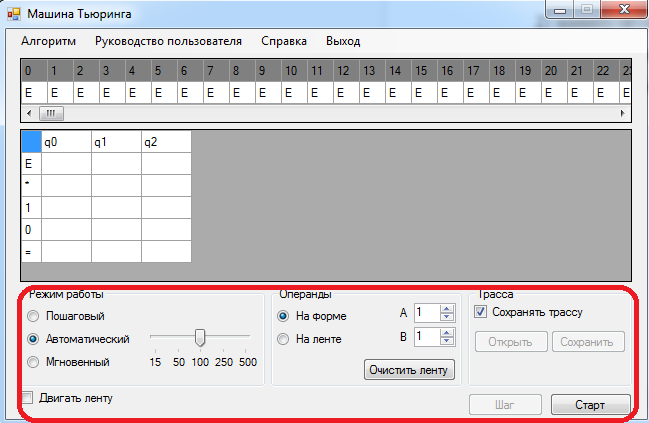
  
Рисунок А.3 – Структура таблицы переходов

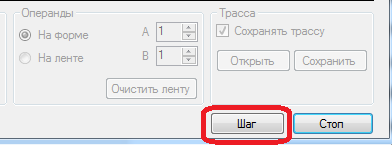
  
Рисунок А.4 – Добавление состояний

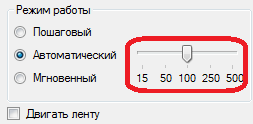
***A.4.3 Настройка параметров моделирования***

В нижней части формы располагается область параметров моделирования (рисунок А.5), в которой содержатся кнопки, отвечающие за изменение скорости моделирования, режима работы, кнопки задачи операндов, кнопки для сохранения и просмотра трассы.

Доступны три режима работы машины Тьюринга (моделирование работы запускается только после нажатия кнопки «Старт»). При выборе «пошагового» режима пользователь сам контролирует скорость работы МТ, следующая команда выполняется только после нажатия на кнопку шаг (рисунок А.6). Работа в автоматическом режиме подразумевает выполнение команд с интервалом, величина которого равна выбранному на ползунке скорости значению (рисунок А.7). При мгновенном режиме работы, алгоритм выполняется без задержек и сразу выдается готовый результат. Кнопка двигать ленту отвечает за отображение чтения записи на ленте, если кнопка активна, то отображение ленты будет следовать за головкой чтения/записи.

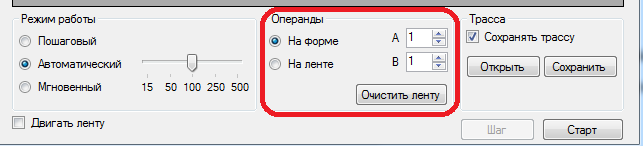
  
Рисунок А.5 – Область параметров моделирования

  
Рисунок А.6 – кнопка шаг

  
Рисунок А.7 – ползунок выбора скорости

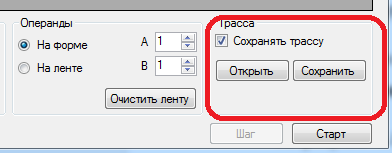
Операнды алгоритма можно задать двумя способами:

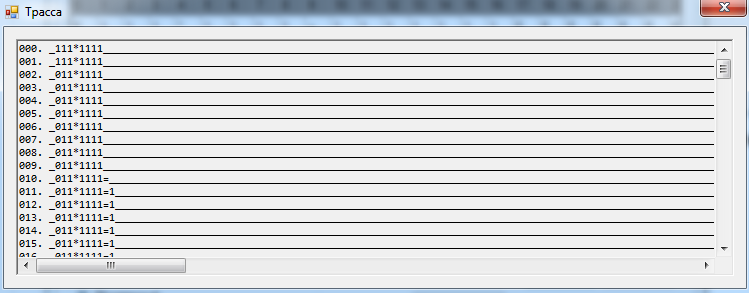
1. на форме (они задаются в целочисленной форме (рисунок А.8));
2. на ленте, здесь значения операндов задаются посимвольно в ячейках ленты (символы на ленте должны соответствовать алфавиту алгоритма, иначе возникнет сообщение об ошибке).

  
Рисунок А.8 – операнды

Так же существует кнопка очистки ленты, которая заменяет все символы на ленте на символы пустоты (по умолчанию E).

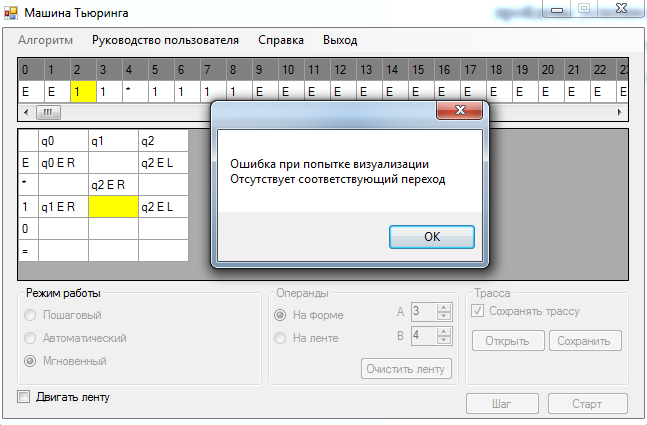
При необходимости можно сформировать трассу выполнения алгоритма, которая является «историей» выполнения алгоритма, просмотреть её или сохранить в текстовый файл с помощью соответствующих кнопок (рисунок А.9). Пример трассы приведен на рисунке А.10.

  
Рисунок А.9 – Работа с трассой

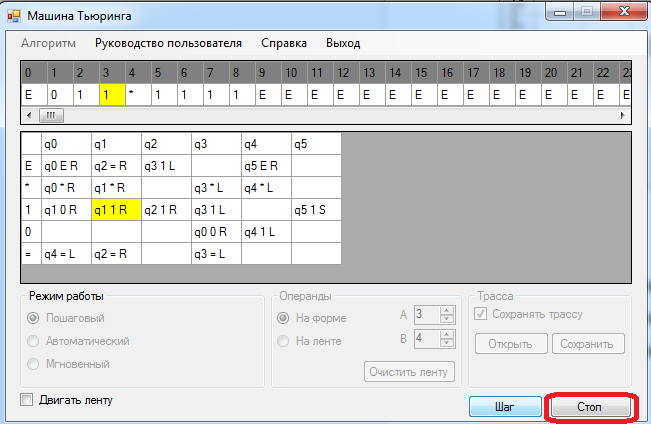
  
Рисунок А.10 – Пример трассы

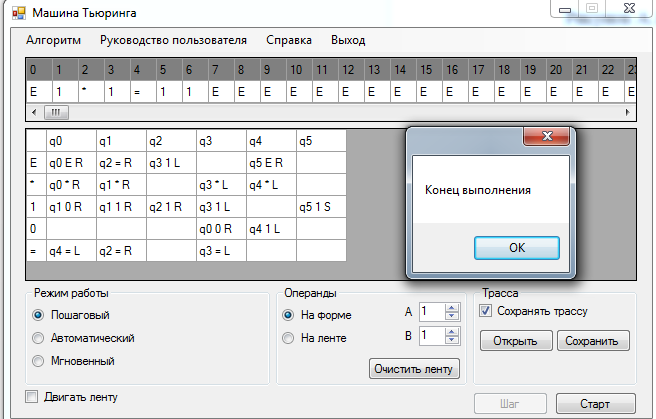
***A.4.4 Моделирование работы алгоритма***

После нажатия кнопки «Старт» система начнет проверять алгоритм и ленту на наличие синтаксических и семантических ошибок, если проверки пройдены успешно, начнется моделирование работы алгоритма, в противном случае пользователь получит сообщение об ошибке, и выделятся ячейки, которые являются причиной ошибки (рисунок А.11).

  
Рисунок А.11 – Сообщение об ошибке

При моделировании работы в режимах «Пошаговый» и «Автоматический» текущая ячейка на ленте и текущая команда перехода подсвечиваются желтым цветом (рисунок А.12). Так же в этих двух режимах имеется возможность остановить выполнение алгоритма не дожидаясь его окончания при нажатии кнопки «Стоп». При успешном выполнении алгоритма выдается сообщение об успешном выполнении (рисунок А.13).

  
Рисунок А.12 – Визуализация работы алгоритма

  
Рисунок А.13 – Сообщение об окончании выполнения

**ПРИЛОЖЕНИЕ Б  
Листинг модулей программы**

**Main.cs**

using System;

using System.IO;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Threading;

namespace MT

{

public partial class FormMain : Form

{

public FormMain()

{

InitializeComponent();

//Инициализируем переходы и ленту

dgvStates = DGV.StatesStart(dgvStates, alphabet, 4);

dgvStates = DGV.StatesFormat(dgvStates);

dgvRibbon = DGV.RibbonStart(dgvRibbon, alphabet, 500, 2);

dgvRibbon = DGV.RibbonFormat(dgvRibbon);

}

//Текущее состояние алфавита

private string alphabet = "E\*10=";

private string defaultAlphabet = "E\*10=";

#region Жмяки по кнопкам

private void новыйToolStripMenuItem\_Click(object sender, EventArgs e)

{

//очищаем dgv

dgvStates.Rows.Clear();

//Алфавит берём дефолтный

alphabet = defaultAlphabet;

//Запускаем нашу dgv

dgvStates = DGV.StatesStart(dgvStates, alphabet, 4);

dgvStates = DGV.StatesFormat(dgvStates);

}

private void загрузитьАлгоритмToolStripMenuItem\_Click(object sender, EventArgs e)

{

dgvStates = File.open(dgvStates, defaultAlphabet);

}

private void сохранитьАлгоритмКакToolStripMenuItem\_Click(object sender, EventArgs e)

{

File.save(dgvStates);

}

private void сложениеToolStripMenuItem\_Click(object sender, EventArgs e)

{

dgvStates = File.openDefault(dgvStates,"default/sum.tma",defaultAlphabet);

}

private void нОДToolStripMenuItem\_Click(object sender, EventArgs e)

{

dgvStates = File.openDefault(dgvStates, "default/multiplication.tma", defaultAlphabet);

}

private void руководствоПользователяToolStripMenuItem\_Click(object sender, EventArgs e)

{

try

{

System.Diagnostics.Process.Start(Application.StartupPath + @"\guide.chm");

}

catch

{

MessageBox.Show("Файл справки не найден");

}

}

private void справкаToolStripMenuItem\_Click(object sender, EventArgs e)

{

About about = new About();

about.StartPosition = FormStartPosition.CenterParent;

about.ShowDialog();

}

private void выходToolStripMenuItem\_Click(object sender, EventArgs e)

{

this.Close();

}

#endregion

#region Режим работы

//Режим работы. 1-Пошаговый, 2-Автоматический, 3-Мгновенный

private int operatingMode = 2;

//Узнаём режим работы

private void radioButtonStep\_CheckedChanged(object sender, EventArgs e)

{

operatingMode = 1;

trackBarSpeed.Visible = false;

labelSpeed.Visible = false;

checkBoxMove.Checked = true;

checkBoxMove.Visible = true;

}

private void radioButtonAuto\_CheckedChanged(object sender, EventArgs e)

{

operatingMode = 2;

trackBarSpeed.Visible = true;

labelSpeed.Visible = true;

checkBoxMove.Checked = true;

checkBoxMove.Visible = true;

}

private void radioButtonNow\_CheckedChanged(object sender, EventArgs e)

{

operatingMode = 3;

trackBarSpeed.Visible = false;

labelSpeed.Visible = false;

checkBoxMove.Checked = false;

checkBoxMove.Visible = false;

}

//Скорость при 2-Автоматическом режиме работы

private int speed = 250;

//Узнаём скорость

private void trackBarSpeed\_Scroll(object sender, EventArgs e)

{

switch (trackBarSpeed.Value)

{

case 1:

speed = 15;

return;

case 2:

speed = 50;

return;

case 3:

speed = 100;

return;

case 4:

speed = 250;

return;

case 5:

speed = 500;

return;

}

}

#endregion

#region Операнды

//Режим задания операндов true-На форме, false-На ленте

private bool operandsModeForm = true;

//Узнаем режим задания операндов

private void radioButtonForm\_CheckedChanged(object sender, EventArgs e)

{

operandsModeForm = true;

numericUpDownA.Visible = true;

numericUpDownB.Visible = true;

labelA.Visible = true;

labelB.Visible = true;

dgvRibbon = DGV.RibbonClear(dgvRibbon, alphabet);

dgvRibbon = DGV.RibbonReadOnly(dgvRibbon, true);

}

private void radioButtonRibbon\_CheckedChanged(object sender, EventArgs e)

{

operandsModeForm = false;

numericUpDownA.Visible = false;

numericUpDownB.Visible = false;

labelA.Visible = false;

labelB.Visible = false;

dgvRibbon = DGV.RibbonClear(dgvRibbon, alphabet);

dgvRibbon = DGV.RibbonReadOnly(dgvRibbon, false);

}

//Операнды A,B

private int operandA = 1;

private int operandB = 1;

//Узнаём операнды

private void numericUpDownA\_ValueChanged(object sender, EventArgs e)

{

operandA = Convert.ToInt32(numericUpDownA.Value);

}

private void numericUpDownB\_ValueChanged(object sender, EventArgs e)

{

operandB = Convert.ToInt32(numericUpDownB.Value);

}

private void buttonClearRibbon\_Click(object sender, EventArgs e)

{

string oldAlphabet = alphabet;

alphabet = "";

for (int i = 1; i < dgvStates.Rows.Count; i++)

alphabet += dgvStates.Rows[i].Cells[0].Value.ToString();

if (Analyzer.alphabetAnalysis(dgvStates, alphabet))

{

dgvRibbon = DGV.RibbonClear(dgvRibbon, alphabet);

dgvStates = DGV.toWhite(dgvStates, 0);

dgvStates.Rows[0].Cells[0].Style.BackColor = Color.Gray;

dgvRibbon = DGV.toWhite(dgvRibbon, 1);

}

else

alphabet = oldAlphabet;

}

#endregion

#region Трасса

//Сохранение трассы

private bool saveTrack = true;

//Трасса

List<string> fullTrack = new List<string>();

//Узнаём, сохранять ли трассу

private void checkBoxSaveTrack\_CheckedChanged(object sender, EventArgs e)

{

saveTrack = checkBoxSaveTrack.Checked;

}

//Открыть трассу

private void buttonOpenTrack\_Click(object sender, EventArgs e)

{

ViewTrack viewTrack = new ViewTrack();

viewTrack.StartPosition = FormStartPosition.CenterParent;

viewTrack.delTrack();

viewTrack.addTrack(fullTrack);

viewTrack.ShowDialog();

}

//Сохранение трассы

private void buttonSaveTrack\_Click(object sender, EventArgs e)

{

File.saveTrack(fullTrack);

}

#endregion

#region dgv

//Для определения в какую ячейку ткнули

int currentMouseOverRow;

int currentMouseOverColumn;

//Для контекстного меню и обработки нажатий по красным клеткам с ошибками

private void dgvStates\_MouseClick(object sender, MouseEventArgs e)

{

currentMouseOverRow = dgvStates.HitTest(e.X, e.Y).RowIndex;

currentMouseOverColumn = dgvStates.HitTest(e.X, e.Y).ColumnIndex;

if (e.Button == MouseButtons.Right)

{

ContextMenu m = new ContextMenu();

if ((currentMouseOverRow == 0) && (currentMouseOverColumn == 0))

{

MenuItem addTo = new MenuItem("Добавить состояние");

MenuItem addAlp = new MenuItem("Добавить строку");

MenuItem formatStates = new MenuItem("Форматировать состояния");

MenuItem formatAlph = new MenuItem("Восстановить алфавит");

addTo.MenuItems.Add(new MenuItem("1"));

addTo.MenuItems.Add(new MenuItem("5"));

addTo.MenuItems.Add(new MenuItem("10"));

m.MenuItems.AddRange(new MenuItem[] { addTo, addAlp, formatStates, formatAlph });

m.MenuItems[0].MenuItems[0].Click += new System.EventHandler(this.add1);

m.MenuItems[0].MenuItems[1].Click += new System.EventHandler(this.add5);

m.MenuItems[0].MenuItems[2].Click += new System.EventHandler(this.add10);

m.MenuItems[1].Click += new System.EventHandler(this.m\_addAlph);

m.MenuItems[2].Click += new System.EventHandler(this.m\_formatStates);

m.MenuItems[3].Click += new System.EventHandler(this.m\_formatAlph);

}

else

if (currentMouseOverColumn == 0)

{

MenuItem addAlp = new MenuItem("Добавить строку");

MenuItem delAlp = new MenuItem("Удалить строку");

MenuItem formatAlph = new MenuItem("Восстановить алфавит");

m.MenuItems.AddRange(new MenuItem[] { addAlp, delAlp, formatAlph });

m.MenuItems[0].Click += new System.EventHandler(this.m\_addAlph);

m.MenuItems[1].Click += new System.EventHandler(this.m\_deleteAlph);

m.MenuItems[2].Click += new System.EventHandler(this.m\_formatAlph);

}

else

if (currentMouseOverRow == 0)

{

MenuItem addTo = new MenuItem("Добавить состояние");

addTo.MenuItems.Add(new MenuItem("1"));

addTo.MenuItems.Add(new MenuItem("5"));

addTo.MenuItems.Add(new MenuItem("10"));

MenuItem delCur = new MenuItem("Удалить состояние");

MenuItem format = new MenuItem("Форматировать состояния");

m.MenuItems.AddRange(new MenuItem[] { addTo, delCur, format });

m.MenuItems[0].MenuItems[0].Click += new System.EventHandler(this.add1);

m.MenuItems[0].MenuItems[1].Click += new System.EventHandler(this.add5);

m.MenuItems[0].MenuItems[2].Click += new System.EventHandler(this.add10);

m.MenuItems[1].Click += new System.EventHandler(this.m\_deleteState);

m.MenuItems[2].Click += new System.EventHandler(this.m\_formatStates);

}

if (currentMouseOverRow == 0 || currentMouseOverColumn == 0)

m.Show(dgvStates, new Point(e.X, e.Y));

}

//При любом нажатии на мышку осветляем эту ячейку, но не трогаем 0ые

if (!(currentMouseOverRow == 0 && currentMouseOverColumn == 0) && (currentMouseOverRow >= 0 && currentMouseOverColumn >= 0))

if (dgvStates.Rows[currentMouseOverRow].Cells[currentMouseOverColumn].Style.BackColor != Color.Yellow)

dgvStates.Rows[currentMouseOverRow].Cells[currentMouseOverColumn].Style.BackColor = Color.White;

}

//Очищение цвета ячейки Ribbon

private void dgvRibbon\_MouseClick(object sender, MouseEventArgs e)

{

currentMouseOverRow = dgvRibbon.HitTest(e.X, e.Y).RowIndex;

currentMouseOverColumn = dgvRibbon.HitTest(e.X, e.Y).ColumnIndex;

if (currentMouseOverRow > 0)

if (dgvRibbon.Rows[currentMouseOverRow].Cells[currentMouseOverColumn].Style.BackColor != Color.Yellow)

dgvRibbon.Rows[currentMouseOverRow].Cells[currentMouseOverColumn].Style.BackColor = Color.White;

}

//Обработка нажатия удаления и добавления

int addCount;

private void add1(object sender, System.EventArgs e)

{

addCount = 1;

m\_addStates(sender, e);

}

private void add5(object sender, System.EventArgs e)

{

addCount = 5;

m\_addStates(sender, e);

}

private void add10(object sender, System.EventArgs e)

{

addCount = 10;

m\_addStates(sender, e);

}

private void m\_addStates(object sender, System.EventArgs e)

{

if (addCount + dgvStates.ColumnCount + 1 >= 100)

if (MessageBox.Show("Количество состояний не может быть более 100\nЖелаете добавить до 100?", "", MessageBoxButtons.YesNo) == System.Windows.Forms.DialogResult.Yes)

addCount = 100 - dgvStates.ColumnCount + 1;

else

addCount = 0;

for (int i = 0; i < addCount; i++)

{

dgvStates.ColumnCount += 1;

dgvStates.Rows[0].Cells[dgvStates.Columns.Count - 1].Value = "";

dgvStates.Columns[dgvStates.Columns.Count - 1].Width = 50;

}

}

private void m\_deleteState(object sender, System.EventArgs e)

{

if (dgvStates.ColumnCount <3)

{

MessageBox.Show("Количество столбцов не может быть менее 1");

}

else

{

dgvStates.Columns.RemoveAt(currentMouseOverColumn);

}

}

private void m\_deleteAlph(object sender, System.EventArgs e)

{

if (dgvStates.RowCount > 3)

dgvStates.Rows.RemoveAt(currentMouseOverRow);

else

MessageBox.Show("Количество строк не может быть менее 2 ");

}

private void m\_addAlph(object sender, System.EventArgs e)

{

if (dgvStates.RowCount > 100)

{

MessageBox.Show("Количество строк не может быть более 100");

}

else

{

dgvStates.RowCount++;

dgvStates.Rows[dgvStates.Rows.Count - 1].Cells[0].Value = "";

}

}

private void m\_formatStates(object sender, System.EventArgs e)

{

for (int i = 1; i < dgvStates.ColumnCount; i++)

dgvStates.Rows[0].Cells[i].Value = "q" + (i - 1);

}

private void m\_formatAlph(object sender, System.EventArgs e)

{

for (int i = 1; i < dgvStates.RowCount; i++)

dgvStates.Rows[i].Cells[0].Value = defaultAlphabet[i - 1];

}

#endregion

#region Работа МТ

//Метод, в котором блокируются кнопки и всё-всё-всё, которые не следует нажимать пользователю во время работы

private void blockOnStart()

{

groupBoxOperands.Enabled = false;

groupBoxTrack.Enabled = false;

// groupBoxWork.Enabled = false;

radioButtonAuto.Enabled = false;

radioButtonNow.Enabled = false;

radioButtonStep.Enabled = false;

trackBarSpeed.Enabled = false;

алгоритмToolStripMenuItem.Enabled = false;

buttonStep.Enabled = false;

for (int i = 0; i < dgvRibbon.ColumnCount; i++)

dgvRibbon.Rows[1].Cells[i].ReadOnly = true;

for (int i = 0; i < dgvStates.ColumnCount; i++)

for (int j = 0; j < dgvStates.RowCount; j++)

dgvStates.Rows[j].Cells[i].ReadOnly = true;

}

//Угадайте

private void unblockOnFinish()

{

groupBoxOperands.Enabled = true;

groupBoxTrack.Enabled = true;

// groupBoxWork.Enabled = true;

radioButtonAuto.Enabled = true;

radioButtonNow.Enabled = true;

radioButtonStep.Enabled = true;

trackBarSpeed.Enabled = true;

алгоритмToolStripMenuItem.Enabled = true;

buttonStep.Enabled = false;

buttonStart.Text = "Старт";

buttonStep.Text = "Шаг";

buttonStart.Enabled = true;

if (saveTrack)

{

buttonOpenTrack.Enabled = true;

buttonSaveTrack.Enabled = true;

}

for (int i = 0; i < dgvRibbon.ColumnCount; i++)

dgvRibbon.Rows[1].Cells[i].ReadOnly = radioButtonForm.Checked;

for (int i = 0; i < dgvStates.ColumnCount; i++)

for (int j = 0; j < dgvStates.RowCount; j++)

dgvStates.Rows[j].Cells[i].ReadOnly = false;

}

//Список всех состояний

string[] states;

//Индекс символа на ленте

int currentRibonIndex;

//Сам символ на ленте

string currentRibonSymbol;

//Индекс текущего состояния в правилаъ

int currentStateIndex;

//Само текущее состояние

string currentState;

//Индс текущего символа в правилах

int currentAlphabetIndex;

//Для окрашивания

int colorIndexRibon;

int colorIndexStates1;

int colorIndexStates2;

//Сдвиг

int shift;

//Кнопка старта

private void buttonStart\_Click(object sender, EventArgs e)

{

if (buttonStart.Text == "Старт")

{

//Шаманство с алфавитом

string oldAlphabet = alphabet;

alphabet = "";

for (int i = 1; i < dgvStates.Rows.Count; i++)

alphabet += dgvStates.Rows[i].Cells[0].Value.ToString();

//Белим

dgvStates = DGV.toWhite(dgvStates, 0);

dgvStates.Rows[0].Cells[0].Style.BackColor = Color.Gray;

dgvRibbon = DGV.toWhite(dgvRibbon, 1);

dgvStates.ClearSelection();

shift = 1;

if (operandsModeForm)

{

if (operandA + operandB + 1 + shift > dgvRibbon.ColumnCount)

dgvRibbon.ColumnCount += (operandA + operandB + 1 + 1) - dgvRibbon.ColumnCount;

for (int i=0; i<shift;i++)

dgvRibbon.Rows[1].Cells[i].Value = alphabet[0];

for (int i = shift; i < operandA + shift; i++)

dgvRibbon.Rows[1].Cells[i].Value = alphabet[2];

dgvRibbon.Rows[1].Cells[operandA + shift].Value = alphabet[1];

for (int i = operandA + 1 + shift; i < operandA + operandB + 1 + shift; i++)

dgvRibbon.Rows[1].Cells[i].Value = alphabet[2];

for (int i = operandA + operandB + 1 + shift; i < dgvRibbon.ColumnCount; i++)

dgvRibbon.Rows[1].Cells[i].Value = alphabet[0];

//dgvRibbonFormat();

dgvRibbon = DGV.RibbonFormat(dgvRibbon);

}

if (Analyzer.alphabetAnalysis(dgvStates,alphabet) && Analyzer.syntacticAnalysis(dgvStates,alphabet,states) && Analyzer.semanticAnalysis(dgvStates,dgvRibbon,alphabet,states) && Analyzer.ribbonAnalysis(dgvRibbon,alphabet))

{

blockOnStart();

colorIndexRibon = 0;

colorIndexStates1 = 0;

colorIndexStates2 = 0;

start();

}

else

{

alphabet = oldAlphabet;

}

}

else

if (buttonStart.Text == "Пауза")

{

trackBarSpeed.Enabled = true;

timerAuto.Enabled = false;

buttonStart.Text = "Продолжить";

}

else

if (buttonStart.Text == "Продолжить")

{

trackBarSpeed.Enabled = false;

timerAuto.Interval = speed;

timerAuto.Enabled = true;

buttonStart.Text = "Пауза";

}

else

if (buttonStart.Text == "Стоп")

step(true);

}

//Метод старта, в котором обработки нажатий кнопок и режимы работы

private void start()

{

dgvRibbon.FirstDisplayedScrollingColumnIndex = 0;

states = new string[dgvStates.ColumnCount];

//Определяем

for (int i = 1; i < states.Length; i++)

states[i] = dgvStates.Rows[0].Cells[i].Value.ToString();

currentRibonIndex = shift - 1;

currentRibonSymbol = dgvRibbon.Rows[1].Cells[currentRibonIndex].Value.ToString();

currentState = states[1];

currentStateIndex = Array.IndexOf(states, currentState);

currentAlphabetIndex = alphabet.IndexOf(currentRibonSymbol) + 1;

fullTrack.Clear();

switch (operatingMode)

{

case (1):

{

buttonStart.Text = "Стоп";

buttonStep.Enabled = true;

return;

}

case (2):

{

buttonStep.Enabled = true;

buttonStep.Text = "Стоп";

timerAuto.Interval = speed;

timerAuto.Enabled = true;

buttonStart.Text = "Пауза";

return;

}

case (3):

{

buttonStart.Enabled = false;

while (!step(false)) ;

return;

}

}

}

//Пошаговая/стоп

private void buttonStep\_Click(object sender, EventArgs e)

{

if (buttonStep.Text == "Шаг")

step(false);

else

if (buttonStep.Text == "Стоп")

{

buttonStep.Text = "Шаг";

timerAuto.Enabled = false;

step(true);

}

}

private bool step(bool stop)

{

bool last = false;

string[] words = {};

try

{

if (saveTrack)

{

string track = "";

for (int i = 0; i < dgvRibbon.ColumnCount; i++)

track += dgvRibbon.Rows[1].Cells[i].Value.ToString();

track = track.Replace(alphabet[0], '\_');

fullTrack.Add(track);

}

words = dgvStates.Rows[currentAlphabetIndex].Cells[currentStateIndex].Value.ToString().Split(' ');

currentState = words[0];

currentStateIndex = Array.IndexOf(states, currentState);

dgvRibbon.Rows[1].Cells[currentRibonIndex].Value = words[1];

if (words[2] == "R")

{

currentRibonIndex++;

currentRibonSymbol = dgvRibbon.Rows[1].Cells[currentRibonIndex].Value.ToString();

currentAlphabetIndex = alphabet.IndexOf(currentRibonSymbol) + 1;

}

if (words[2] == "L")

{

currentRibonIndex--;

currentRibonSymbol = dgvRibbon.Rows[1].Cells[currentRibonIndex].Value.ToString();

currentAlphabetIndex = alphabet.IndexOf(currentRibonSymbol) + 1;

}

if (words[2] == "S")

{

last = true;

}

currentAlphabetIndex = alphabet.IndexOf(currentRibonSymbol) + 1;

dgvRibbon.Rows[1].Cells[colorIndexRibon].Style.BackColor = Color.White;

dgvStates.Rows[colorIndexStates1].Cells[colorIndexStates2].Style.BackColor = Color.White;

colorIndexRibon = currentRibonIndex;

colorIndexStates1 = currentAlphabetIndex;

colorIndexStates2 = currentStateIndex;

if (checkBoxMove.Checked)

{

if (colorIndexRibon - 11 + 1 > 0)

dgvRibbon.FirstDisplayedScrollingColumnIndex = colorIndexRibon - 11 + 1;

else

dgvRibbon.FirstDisplayedScrollingColumnIndex = 0;

}

dgvRibbon.Rows[1].Cells[colorIndexRibon].Style.BackColor = Color.Yellow;

dgvStates.Rows[colorIndexStates1].Cells[colorIndexStates2].Style.BackColor = Color.Yellow;

if (last || stop)

{

timerAuto.Enabled = false;

unblockOnFinish();

dgvRibbon.Rows[1].Cells[colorIndexRibon].Style.BackColor = Color.White;

dgvStates.Rows[colorIndexStates1].Cells[colorIndexStates2].Style.BackColor = Color.White;

}

if (last)

MessageBox.Show("Конец выполнения");

if (stop)

MessageBox.Show("Принудительное завершение");

}

catch

{

stop = true;

last = true;

timerAuto.Enabled = false;

Color error = Color.OrangeRed;

if (words.Length < 3)

MessageBox.Show("Ошибка при попытке визуализации\nОтсутствует соответствующий переход");

else

if (currentRibonIndex >= dgvRibbon.ColumnCount)

MessageBox.Show("Ошибка при попытке визуализации\nАлгоритм уходит за правый край ленты");

else

if (currentRibonIndex < 0)

{

MessageBox.Show("Ошибка при попытке визуализации\nАлгоритм уходит за левый край ленты");

error = Color.Gray;

}

else

MessageBox.Show("Ошибка при попытке визуализации\nНепредвиденная ошибка");

unblockOnFinish();

dgvRibbon.Rows[1].Cells[colorIndexRibon].Style.BackColor = Color.Red;

dgvStates.Rows[colorIndexStates1].Cells[colorIndexStates2].Style.BackColor = error;

}

return last || stop;

}

//Таймер

private void timerAuto\_Tick(object sender, EventArgs e)

{

step(false);

}

#endregion

}

}

**File.cs**

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace MT

{

class File

{

//Для сохранения трассы подбиваем до нужного количества символов

private static string toN(int i, int N)

{

string s = Convert.ToString(i);

while (s.Length != N) s = "0" + s;

return s;

}

public static DataGridView openDefault(DataGridView dgv1, string path, string defaultAlphabet)

{

try

{

StreamReader sr = new StreamReader(@path);

try

{

//длина и ширина

int a = 0;

int b = 0;

//Считываем длину и ширину

a = Convert.ToInt32(sr.ReadLine());

b = Convert.ToInt32(sr.ReadLine());

//Cоздаём новые размеры

dgv1.Rows.Clear();

dgv1.Rows.Add(a);

dgv1.ColumnCount = b;

string newAlphabet = "";

for (int i = 0; i < a; i++)

{

string[] mas = sr.ReadLine().Split(';');

for (int j = 0; j < b; j++)

{

//переписываем в dgv

dgv1.Rows[i].Cells[j].Value = mas[j];

//Если не нулевая ячейка, то считываем алфавит

if (j == 0 && i != 0)

newAlphabet += mas[j];

}

}

dgv1 = DGV.StatesFormat(dgv1);

return dgv1;

}

catch

{

MessageBox.Show("Проблемы считывания стандартного алгоритма");

dgv1 = DGV.StatesStart(dgv1, defaultAlphabet, 4);

return dgv1;

}

finally

{

sr.Close();

}

}

catch

{

MessageBox.Show("Стандартный алгоритм не найден");

dgv1 = DGV.StatesStart(dgv1, defaultAlphabet, 4);

return dgv1;

}

}

public static DataGridView open(DataGridView dgv1, string defaultAlphabet)

{

//Файловый диалог

OpenFileDialog ofd = new OpenFileDialog();

ofd.Filter = "Algorithm of Turing machines|\*.tma";

ofd.Title = "Открыть алгоритм";

//Открываем и ждём нажатия кнопки

if (ofd.ShowDialog() == DialogResult.OK)

{

//Открываем поток с нашим файлом

StreamReader sr = new StreamReader(ofd.FileName);

//ошибка считывания либо синтаксиса

bool ex = false;

//длина и ширина

int a = 0;

int b = 0;

try

{

//Считываем длину и ширину(пытаемся)

a = Convert.ToInt32(sr.ReadLine());

b = Convert.ToInt32(sr.ReadLine());

//Создаём новые размеры

dgv1.Rows.Clear();

dgv1.Rows.Add(a);

dgv1.ColumnCount = b;

//заводим переменную для нового алфавита, который будет считан

string newAlphabet = "";

for (int i = 0; i < a; i++)

{

//Считываем в массив строк

string[] mas = sr.ReadLine().Split(';');

//Если количество ; и столбцов не совпало хотя бы раз, то ex и уходим

if (mas.Length != b + 1)

{

ex = true;

MessageBox.Show("Неверный формат файла");

break;

}

for (int j = 0; j < b; j++)

{

//переписываем в dgv

dgv1.Rows[i].Cells[j].Value = mas[j];

//Если не нулевая ячейка, то считываем алфавит

if (j == 0 && i != 0)

newAlphabet += mas[j];

if (ex) break;

}

if (ex) break;

}

if(ex)

{

dgv1 = DGV.StatesStart(dgv1, defaultAlphabet, 4);

dgv1 = DGV.StatesFormat(dgv1);

}

else

{

dgv1 = DGV.StatesFormat(dgv1);

return dgv1;

}

}

//Если вылетели по ошибке, то загружаем новый алгоритм

catch

{

MessageBox.Show("Неверный формат файла");

ex = true;

dgv1 = DGV.StatesStart(dgv1, defaultAlphabet, 4);

dgv1 = DGV.StatesFormat(dgv1);

return dgv1;

}

//В любом случае нужно закрыть поток

finally

{ sr.Close(); }

}

return dgv1;

}

public static void save(DataGridView dgv1)

{

SaveFileDialog sfd = new SaveFileDialog();

sfd.Filter = "Turing machine algorithm|\*.tma";

sfd.Title = "Сохранить алгоритм как";

//Ждём кнопку ОК

if (sfd.ShowDialog() == DialogResult.OK)

{

StreamWriter sw = new StreamWriter(sfd.FileName);

sw.WriteLine(dgv1.RowCount);

sw.WriteLine(dgv1.ColumnCount);

for (int i = 0; i < dgv1.RowCount; i++)

{

for (int j = 0; j < dgv1.ColumnCount; j++)

sw.Write(dgv1.Rows[i].Cells[j].Value + ";");

if (i != dgv1.RowCount - 1)

sw.WriteLine("");

}

sw.Close();

}

}

public static void saveTrack(List<string> fullTrack)

{

SaveFileDialog sfd = new SaveFileDialog();

sfd.Filter = "Turing machine track|\*.tmt";

sfd.Title = "Сохранить трассу как";

//Ждём кнопку ОК

if (sfd.ShowDialog() == DialogResult.OK)

{

StreamWriter sw = new StreamWriter(sfd.FileName);

for (int i = 0; i < fullTrack.Count; i++)

sw.WriteLine(toN(i + 1, Convert.ToString(fullTrack.Count).Length) + ". " + fullTrack[i]);

sw.Close();

}

}

}

}

**ViewTrack.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace MT

{

public partial class ViewTrack : Form

{

public ViewTrack()

{

InitializeComponent();

}

private string toN(int i,int N)

{

string s = Convert.ToString(i);

while (s.Length != N) s = "0" + s;

return s;

}

public void delTrack()

{

richTextBoxTrack.Clear();

}

public void addTrack(List<string> fullTrack)

{

for(int i=0;i<fullTrack.Count;i++)

richTextBoxTrack.Text += toN(i,Convert.ToString(fullTrack.Count).Length) + ". " + fullTrack[i] + "\n";

}

}

}

**dgv.cs**

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Reflection;

namespace MT

{

class DGV

{

//При старте

public static DataGridView StatesStart(DataGridView dgv, string alphabet, int a)

{

dgv.GetType().GetProperty("DoubleBuffered", BindingFlags.NonPublic | BindingFlags.Instance).SetValue(dgv, true, null);

dgv.Rows.Clear();

//Разрешаем добавлять строки

dgv.AllowUserToAddRows = true;

//Количество строк = 4(3+1 для определения алфавита)

dgv.ColumnCount = a;

dgv.Rows.Clear();

//Задаю количество строк = длине алфавита+1, т.к. нулевая ячейка

dgv.Rows.Add(alphabet.Length + 1);

//Заполняю столбец алфавитом

for (int i = 0; i < alphabet.Length; i++)

dgv.Rows[i + 1].Cells[0].Value = alphabet[i];

//Рисую состояния

for (int i = 1; i < dgv.ColumnCount; i++)

dgv.Rows[0].Cells[i].Value = "q" + (i - 1);

//Инициализация каждой ячейки

for (int i = 1; i < dgv.Rows.Count; i++)

for (int j = 1; j < dgv.Columns.Count; j++)

dgv.Rows[i].Cells[j].Value = "";

//Запрещаем добавлять строки

dgv.AllowUserToAddRows = false;

return dgv;

}

public static DataGridView RibbonStart(DataGridView dgv, string alphabet, int a, int b)

{

dgv.GetType().GetProperty("DoubleBuffered", BindingFlags.NonPublic | BindingFlags.Instance).SetValue(dgv, true, null);

dgv.AllowUserToAddRows = true;

dgv.ColumnCount += 1;

dgv.Rows.Add(b);

for (int i = 0; i < a - 1; i++)

{

dgv.ColumnCount += 1;

dgv.Rows[1].Cells[i].Value = alphabet[0];

dgv.Rows[1].Cells[i].ReadOnly = true;

dgv.Columns[i].FillWeight = 1;

}

// dgvRibbon.Rows.Add(b);

dgv.AllowUserToAddRows = false;

return dgv;

}

//0 строка и 0 столбец - ReadOnly, обрабока 0;0, определяем FillWeight и ширину ячеек ну и всякую фигню.

public static DataGridView StatesFormat(DataGridView dgv)

{

for (int i = 0; i < dgv.ColumnCount; i++)

{

dgv.Columns[i].FillWeight = 1;

dgv.Columns[i].Width = 50;

}

dgv.Columns[0].Width = 20;

dgv.Rows[0].Cells[0].Style.BackColor = System.Drawing.Color.DarkGray;

dgv.Rows[0].Cells[0].Value = " ";

return dgv;

}

public static DataGridView RibbonFormat(DataGridView dgv)

{

for (int i = 0; i < dgv.ColumnCount; i++)

{

dgv.Rows[0].Cells[i].Value = i;

dgv.Rows[0].Cells[i].ReadOnly = true;

dgv.Rows[0].Cells[i].Style.BackColor = Color.Gray;

dgv.Rows[0].Cells[i].Style.SelectionBackColor = Color.Gray;

dgv.Rows[0].Cells[i].Style.SelectionForeColor = Color.Black;

dgv.Rows[1].Cells[i].ReadOnly = true;

dgv.Columns[i].FillWeight = 1;

if (dgv.ColumnCount <= 100)

dgv.Columns[i].Width = 20;

else

dgv.Columns[i].Width = 26;

}

return dgv;

}

public static DataGridView toWhite(DataGridView dgv, int index)

{

for (int i = index; i < dgv.Rows.Count; i++)

for (int j = 0; j < dgv.Columns.Count; j++)

dgv.Rows[i].Cells[j].Style.BackColor = Color.White;

return dgv;

}

public static DataGridView RibbonClear(DataGridView dgv,string alphabet)

{

for (int i = 0; i < dgv.ColumnCount; i++)

dgv.Rows[1].Cells[i].Value = alphabet[0];

return dgv;

}

public static DataGridView RibbonReadOnly(DataGridView dgv, bool value)

{

for (int i = 0; i < dgv.ColumnCount; i++)

dgv.Rows[1].Cells[i].ReadOnly = value;

return dgv;

}

}

}

**Analyzer.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace MT

{

class Analyzer

{

private static void error(DataGridView dgv, int a, int b, char error, char where)

{

//с какой dgv работаем

//позиции a,b

//e=0 вернуть обратно цвет ячейки, e=1 - синтаксическая ошибка, e=2 - семантическая, e=3 - синтаксическая на ленте

//where = ' ' - пустой параметр, where='0,1,2' - позиция ошибки(либо её характер)

dgv.Rows[a].Cells[b].Style.BackColor = System.Drawing.Color.OrangeRed;

//СНАЧАЛА СВИТЧИМ ERROR, ПОТОМ WHERE

switch (error)

{

case ('0'):

{

break;

}

case ('1'):

{

switch (where)

{

case ('0'):

{

MessageBox.Show("Синтаксическая ошибка в ячейке правил перехода " + a + ";" + b + "\nПроверьте название состояния");

return;

}

case ('1'):

{

MessageBox.Show("Синтаксическая ошибка в ячейке правил перехода " + a + ";" + b + "\nСимвол не соответствует алфавиту");

return;

}

case ('2'):

{

MessageBox.Show("Синтаксическая ошибка в ячейке правил перехода " + a + ";" + b + "\nНеверная команда перемещения");

return;

}

case ('3'):

{

MessageBox.Show("Синтаксическая ошибка в ячейке правил перехода " + a + ";" + b + "\nНеверное число параметров в ячейке");

return;

}

case (' '):

{

MessageBox.Show("Синтаксическая ошибка в ячейке правил перехода " + a + ";" + b + "\nНепредвиденная ошибка");

return;

}

}

break;

}

case ('2'):

{

switch (where)

{

case ('0'):

{

MessageBox.Show("Семантаческая ошибка в ячейке правил перехода " + a + ";" + b + "\nПроверьте верность состояния");

break;

}

case ('1'):

{

MessageBox.Show("Семантаческая ошибка в ячейке правил перехода " + a + ";" + b + "\nНе существует такого состояния в списке состояний");

break;

}

case ('2'):

{

MessageBox.Show("Семантаческая ошибка в ячейке правил перехода " + a + ";" + b + "\nНет обработки начального состояния");

break;

}

}

break;

}

case ('3'):

{

switch (where)

{

case ('0'):

{

MessageBox.Show("Синтаксическая ошибка в ячейке ленты " + a + ";" + b + "\nЯчейка ленты должна содержать 1 символ или не содержать его вовсе");

return;

}

case ('1'):

{

MessageBox.Show("Синтаксическая ошибка в ячейке ленты " + a + ";" + b + "\nЯчейка ленты должна содержать символы алфавита");

return;

}

}

break;

}

case ('4'):

{

switch (where)

{

case ('0'):

{

MessageBox.Show("Ошибка алфавита в ячейке правил перехода " + a + ";" + b + "\nВ ячейке должен содержаться 1 символ алфавита");

return;

}

case ('1'):

{

MessageBox.Show("Ошибка алфавита в ячейке правил перехода " + a + ";" + b + "\nВ алфавите не могут содержаться одинаковые символы");

return;

}

case ('2'):

{

MessageBox.Show("Ошибка алфавита в ячейке правил перехода " + a + ";" + b + "\nВ алфавите не может содержаться пробел");

return;

}

}

break;

}

}

}

public static bool syntacticAnalysis(DataGridView dgv, string alphabet, string[] states)

{

//Для состояний

states = new string[dgv.ColumnCount];

//Определяем

for (int k = 1; k < states.Length; k++)

states[k] = dgv.Rows[0].Cells[k].Value.ToString();

for (int i = 1; i < dgv.RowCount; i++)

for (int j = 1; j < dgv.ColumnCount; j++)

//Если строка не пустая

try

{

if (dgv.Rows[i].Cells[j].Value.ToString().Length != 0)

try

{

string[] words = dgv.Rows[i].Cells[j].Value.ToString().Split(' ');

//Если три слова

if (words.Length != 3)

{

error(dgv, i, j, '1', '3');

return false;

}

//Если только одна буква во втором слове и она содержится в алфавите

if (words[1].Length != 1 || !alphabet.Contains(words[1]))

{

error(dgv, i, j, '1', '1');

return false;

}

//Если только одна буква в последнем слове и если эта буква R,L,S

if (words[2].Length != 1 || !"RLS".Contains(words[2]))

{

error(dgv, i, j, '1', '1');

return false;

}

}

catch

{

error(dgv, i, j, '1', ' ');

return false;

}

}

catch

{

//При написании чего-ибо в ячейку, а потом удаление онного, в ячейке не "", а null. Исправляем.

dgv.Rows[i].Cells[j].Value = "";

}

return true;

}

public static bool semanticAnalysis(DataGridView dgvStates, DataGridView dgvRibbon, string alphabet, string[] states)

{

//Для состояний

states = new string[dgvStates.ColumnCount];

//Определяем

for (int k = 1; k < states.Length; k++)

states[k] = dgvStates.Rows[0].Cells[k].Value.ToString();

for (int i = 1; i < dgvStates.RowCount; i++)

for (int j = 1; j < dgvStates.ColumnCount; j++)

//Если строка не пустая

try

{

if (dgvStates.Rows[i].Cells[j].Value.ToString().Length != 0)

try

{

if (Array.IndexOf(states, dgvStates.Rows[i].Cells[j].Value.ToString().Substring(0, dgvStates.Rows[i].Cells[j].Value.ToString().IndexOf(' '))) == -1)

{

error(dgvStates, i, j, '2', '1');

return false;

}

}

catch

{

error(dgvStates, i, j, '2', '0');

return false;

}

}

catch

{

//При написании чего-ибо в ячейку, а потом удаление онного, в ячейке не "", а null. Исправляем.

dgvStates.Rows[i].Cells[j].Value = "";

}

if (dgvStates.Rows[alphabet.IndexOf(dgvRibbon.Rows[1].Cells[0].Value.ToString()) + 1].Cells[1].Value.ToString() == "")

{

error(dgvStates, alphabet.IndexOf(dgvRibbon.Rows[1].Cells[0].Value.ToString()) + 1, 1, '2', '2');

return false;

}

return true;

}

public static bool ribbonAnalysis(DataGridView dgv, string alphabet)

{

for (int i = 0; i < dgv.ColumnCount; i++)

{

try

{

if (dgv.Rows[1].Cells[i].Value.ToString().Length > 1)

{

error(dgv, 1, i, '3', '0');

return false;

}

if (!alphabet.Contains(dgv.Rows[1].Cells[i].Value.ToString()))

{

error(dgv, 1, i, '3', '1');

return false;

}

}

catch

{

//При написании чего-ибо в ячейку, а потом удаление онного, в ячейке не "", а null. Исправляем.

dgv.Rows[1].Cells[i].Value = alphabet[0];

}

}

return true;

}

public static bool alphabetAnalysis(DataGridView dgv, string alphabet)

{

string S = "";

for (int i = 1; i < dgv.RowCount; i++)

{

S += dgv.Rows[i].Cells[0].Value;

if (dgv.Rows[i].Cells[0].Value.ToString().Length != 1)

{

error(dgv, i, 0, '4', '0');

return false;

}

}

for (int i = 0; i < alphabet.Length; i++)

for (int j = 0; j < alphabet.Length - i; j++)

if (i != j)

if (alphabet[i] == alphabet[j])

{

error(dgv, j, i, '4', '1');

return false;

}

for (int i = 0; i < alphabet.Length; i++)

if (alphabet[i] == ' ')

{

error(dgv, i + 1, 0, '4', '2');

return false;

}

return true;

}

}

}

**About.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace MT

{

public partial class About : Form

{

public About()

{

InitializeComponent();

}

private void About\_Load(object sender, EventArgs e)

{

labelDev.Text = "Разработчики:" + "\n"+"студенты группы 6401-090301D"+"\n"+"Постников А.С."+"\n"+"Терёхин А.А."+"\n"+"Шапошников А.А.";

foreach (Label label in this.Controls.OfType<Label>())

label.Left= (this.Width - label.Width) / 2-7;

labelDev.Left = 10;

}

}

}