|  |  |  |
| --- | --- | --- |
| ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ Федеральное государственное автономное образовательное  учреждение высшего профессионального образования Национальный исследовательский университет  "Высшая школа экономики"  Пермский филиал  Факультет бизнес-информатики | | |
| Кафедра информационных технологий в бизнесе | | |
| УДК 004.02 | | |
| **Проект**  «Формализация алгоритмов: машины Тьюринга и нормальные алгорифмы Маркова» | | |
|  |  | Работу выполнили студент ы группы БИ-15-1 1 курса факультета бизнес-информатики  Шерстобитова П.Д.  Самсонова А.О  Хафизов А.С.  Корзухин.А.А.  Принял:  Доцент кафедры ИТБ ,к.т.н  Лядова.Л.Н  “\_8\_\_\_\_”апреля 20\_16\_ г. |
| Пермь 20*16* | | |

Оглавление

[Глава1. Постановка задачи 4](#_Toc448001368)

[Глава2. Журнал тестов 6](#_Toc448001369)

[Версия 1.0 6](#_Toc448001370)

[Тесты 6](#_Toc448001371)

[Ошибки 8](#_Toc448001372)

[Версия 1.1 8](#_Toc448001373)

[Исправления 8](#_Toc448001374)

[Тесты 8](#_Toc448001375)

[Ошибки 9](#_Toc448001376)

[Версия 1.2 9](#_Toc448001377)

[Исправления 9](#_Toc448001378)

[Тесты 9](#_Toc448001379)

[Ошибки 9](#_Toc448001380)

[Версия 1.3 9](#_Toc448001381)

[Исправления 9](#_Toc448001382)

[Тесты 9](#_Toc448001383)

[Ошибки 9](#_Toc448001384)

[Версия 1.4 10](#_Toc448001385)

[Исправления 10](#_Toc448001386)

[Глава3. Описание алгоритма 11](#_Toc448001387)

[Текстовое описание алгоритма 11](#_Toc448001388)

[**Графическое описание алгоритма** 18](#_Toc448001389)

[Глава4. Описание порядка использования программы 25](#_Toc448001390)

[Руководство пользователя 25](#_Toc448001391)

[Глава5. Исходный код программы 28](#_Toc448001392)

[Заключение 45](#_Toc448001393)

[Библиографический список 46](#_Toc448001394)

[Приложения 47](#_Toc448001395)

# Постановка задачи

Тренажёр «Нормальные алгорифмы Маркова» — это учебная модель универсального исполнителя, предложенного в 1940-х годах [А.А. Марковым](http://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%80%D0%BA%D0%BE%D0%B2,_%D0%90%D0%BD%D0%B4%D1%80%D0%B5%D0%B9_%D0%90%D0%BD%D0%B4%D1%80%D0%B5%D0%B5%D0%B2%D0%B8%D1%87_%28%D0%BC%D0%BB%D0%B0%D0%B4%D1%88%D0%B8%D0%B9%29) для уточнения понятия алгоритма. Марков предположил, что любой алгоритм может быть записан в виде нормального «алгорифма». Позднее было доказано, что нормальные алгорифмы Маркова эквивалентны по своим возможностям другим универсальным исполнителям: [машине Тьюринга](http://kpolyakov.spb.ru/prog/turing.htm) и [машине Поста](http://kpolyakov.spb.ru/prog/post.htm).

Нормальный алгорифм задает метод преобразования строк с помощью системы подстановок. Каждая подстановка состоит из слова-образца и слова-замены, разделенных цепочкой символов «**->**». На каждом шаге замены подстановки просматриваются по порядку сверху вниз, и выполняется первая из них, которая подошла: первое найденное слово-образец рабочей строки заменяется на слово-замену.

Задача: создание интерпретатора для решения различных прикладных задач на абстрактной вычислительной машине.

Таблица 1.1 (Исходные данные)

|  |  |  |
| --- | --- | --- |
| **Исходные данные** | **Классы** | **Ожидаемый результат** |
| Пустой алфавит | Ввести строку самому  Выбрать из предложенных  Выгрузить из файла | Сгенерированный алфавит |
| Пустая рабочая строка | Элементы не из заданного алфавита | Ошибка |
| Элементы из алфавита | Рабочая строка |
| Ввод данных в таблицу постановок | Неподходящая строка  Неподходящее число | Ошибка |
| Строка, соответствующая заявленному алфавиту | Продолжение работы программы, сформированная часть правила |
| Файл с заготовкой программы | Бинарный файл | Продолжение работы с программой |

При создании интерфейса интерпретатора основывались в основном на готовых программах. В первую очередь хочется сказать про интерпретатор нормальных алгорифмах Маркова, который можно увидеть, перейдя по данной ссылке:<http://kpolyakov.spb.ru/prog/turing.htm>. В дальнейшем она будет упоминаться, как «НАМ2». В НАМ2 пользователь не имеет возможности задать алфавит, он вводит данные сразу на ленту, и, если они не указаны в правилах, то с ними не происходит действия, а в нашей программе существует несколько способов задания алфавита: при помощи заготовок, с клавиатуры и из файла. Также в НАМ2 пустой символ обозначается пустой строкой, что не совсем верно с точки зрения логики, также при вводе правила о замене пустоты на элемент, данный элемент ставится перед словом, что совершенно непонятно, ведь пустота существует везде. В нашей программы для обозначения пустоты используется специальный символ Лямда, что обеспечивает понятность при реализации программ. Также в нашей программе существует два типа направлений: стрелочка и стрелочка с точкой, обозначающая окончание программы, а в НАМ2 точка ставится после слова-замены, что на наш взгляд является недопустимым. Также в НАМ2 при вводе задании на входе пустой строки, программа не сообщает об этом, а завершает программу в штатном режиме, по нашим же алгоритмах, пользователю выводит сообщение об ошибке.

# Журнал тестов

## Версия 1.0

### Тесты

1. Создание алфавита: работа с бинарным алфавитом (нажатие на кнопку «Бинарный алфавит» в меню выбора алфавита)

* Результат – в рабочую строку вводятся только цифры 0 и 1 (другие символы не появляются на экране), длина строки ограничена 40 символами (при попытке ввести большее количество символов ничего не происходит), в поля «Экземпляр» и «Замена» также записываются только 0 и 1 (при попытке записать другие символы возникает сообщение «Данные введены неверно» и записанные символы стираются)

1. Создание алфавита: работа с английским алфавитом (нажатие на кнопку «Английский алфавит» в меню выбора алфавита)

* Результат – в рабочую строку вводятся только строчные и прописные английские буквы (другие символы не появляются на экране), длина строки ограничена 40 символами (при попытке ввести большее количество символов ничего не происходит), в поля «Экземпляр» и «Замена» также записываются только строчные и прописные английские буквы (при попытке записать другие символы возникает сообщение «Данные введены неверно»и записанные символы стираются)

1. Создание алфавита: работа с русским алфавитом (нажатие на кнопку «Русский алфавит» в меню выбора алфавита)

* Результат – в рабочую строку вводятся только строчные и прописные русские буквы (другие символы не появляются на экране), длина строки ограничена 40 символами (при попытке ввести большее количество символов ничего не происходит), в поля «Экземпляр» и «Замена» также записываются только строчные и прописные русские буквы (при попытке записать другие символы возникает сообщение «Данные введены неверно» и записанные символы стираются)

1. Создание алфавита: работа со своим алфавитом (ввод своего алфавита «01230123» и нажатие кнопки «Создание»)

* Результат – из алфавита удаляются повторяющиеся символы (остаются только символы 0, 1, 2 и 3), в рабочую строку вводятся только символы 0, 1, 2 и 3 (другие символы не появляются на экране), длина строки ограничена 40 символами (при попытке ввести большее количество символов ничего не происходит), в поля «Экземпляр» и «Замена» также записываются только символы 0, 1, 2 и 3 (при попытке записать другие символы возникает сообщение «Данные введены неверно» и записанные символы стираются)

1. Создание алфавита: создание алфавита из файла (нажатие на кнопку «Из файла» и выбор файла с алфавитом «Alphabet.txt»)

* Результат –в рабочую строку вводятся только символы A, B, Cи D (другие символы не появляются на экране), длина строки ограничена 40 символами (при попытке ввести большее количество символов ничего не происходит), в поля «Экземпляр» и «Замена» также записываются только символы A, B, Cи D (при попытке записать другие символы возникает сообщение «Данные введены неверно» и записанные символы стираются)

1. Ввод собственного рабочего алгоритма

* Примечание: были введены следующие данные:

Алфавит: 1abc\*

Рабочая строка: 111\*11111

Команды:

\*1 -> \*a

a1 ->aa

ba -> cab

bc ->cb

ac -> ca

b ->λ[пустота]

1\* -> \*b

a ->λ [пустота]

\* ->λ [пустота]

c -> 1

В результате выполнения данной программы должно произойти умножение унарных чисел 111 и 11111 (ответ – 111111111111111)

* Результат – алгоритм выполняется, получен правильный ответ 111111111111111

1. Работа с верхним меню: открытие файла с программой (нажатие на кнопку «Открыть…» и выбор в диалоговом окне файла с программой «0 влево 1 вправо.bin»)

* Примечание: предварительное состояние интерпретатора – рабочая строка, заполненная символами A, B, Cи D и программа из 4 команд
* Результат – рабочая строка обновлена на строку из файла, вспомогательные сообщения «Строка на предыдущем шаге» и «Исходная строка» не удалены, сама программа не переписана полностью (первая команда переписана на ту, что из файла, остальные 3 команды остались от предыдущей программы), при выполнении возникает ошибка «Пожалуйста, заполните все ячейки с командами или удалите ненужные данные», в строке «Ваш алфавит» отображается алфавит ABCDвместо 01 (ожидаемый результат – полное обновление команд и алфавита и стирание вспомогательных сообщений)
* Примечание: при повторном открытии файла после всех написанных команд появлялась еще одна пустая строка. При попытке сохранить программу с помощью кнопки «Сохранить…» возникает ошибка «Необрабатываемое исключение в приложении. Индекс находился вне границ массива»

1. Работа с верхним меню: открытие файла с программой (нажатие на кнопку «Открыть…» и выбор в диалоговом окне файла с программой «Из двоичной в унарную.bin»

* Примечание: предварительное состояние программы – пустой интерфейс (никаких данных введено не было)
* Результат – программа переписывает условие, рабочую строку, алфавит и строки из файла, но кнопки «Выполнить» и «Пошаговое выполнение» заблокированы и программу невозможно выполнить
* Примечание: после открытия других файлов с программами кнопки также оказывались заблокированными

### Ошибки

* Открытие программы из файла не ведет к удалению всех предыдущих данных;
* Программы из файлов невозможно выполнить (кнопки заблокированы).

## Версия 1.1

### Исправления

* Все рабочие поля программы обновляются после открытия программы из файла;
* Кнопки для выполнения программы больше не блокируются при наличии всех введенных данных;
* Рабочая строка ограничена 100 символами.

### Тесты

1. Проверка зацикливания на самостоятельно введенной программе

* Примечание: программа состоит из рабочей строки, состоящей из случайно введенных символов A, B, C, и Dи следующих команд:

A ->B

B ->C

C ->D

D ->A

В результате программа должна зациклиться (циклично заменять первый символ на один из 4 вариантов)

* Результат – при выполнении программы возникает сообщение об ошибке «Ваш алгоритм зациклился» (механизм распознавания зацикливания работает)

1. Отладка всех компонентов программы (на примере файла с программой «Скобки.bin»)

* Результат –кнопки «Добавить/удалить строку» выполняют свои действия, при нажатии кнопки «Алфавит» и выборе нового алфавита программа стирается, при нажатии кнопок «Выполнение» или «Пошаговое выполнение» возникает сообщение «Пожалуйста, заполните все ячейки с командами или удалите ненужные данные»

### Ошибки

* Программы из файла не выполняются.

## Версия 1.2

### Исправления

* Программы из файла выполняются.

### Тесты

1. Повторное выполнение программ из файлов (были протестированы все существующие файлы с алгоритмами для этого интерпретатора)

* Результат – алгоритмы в файлах «0 влево 1 вправо.bin» и «из двоичной в унарную.bin» при первом выполнении работают в полном объеме, но при повторном открытии этих файлов пошаговое выполнение не работает (возникает сообщение «Пожалуйста, заполните все ячейки с командами или удалите ненужные данные»)

### Ошибки

* Некоторые программы из файлов не работают при второй и последующих попытках их выполнения.

## Версия 1.3

### Исправления

* Все программы работают и на второй, и на последующий разы.

### Тесты

1. Ввод бесконечного алгоритма

* Примечание: были введены следующие данные:

Алфавит – 01 (бинарный);

Рабочая строка – 0;

Команда – 0 -> 01

Ожидаемый результат работы – бесконечно увеличивающаяся строка (необходимо соответствующее предупреждение)

* Результат – программа после выполнения не реагирует на нажатия кнопок («зависает»), из-за чего ее приходится принудительно закрывать.

### Ошибки

* Не работает остановка алгоритма в случае бесконечного алгоритма

## Версия 1.4

### Исправления

* При вводе бесконечного алгоритма интерпретатор автоматически останавливает работу

*В данной версии не было выявлено новых ошибок, были повторно проведены все тесты из предыдущих версий интерпретатора, которые оказались неудачными. Весь заявленный функционал интерпретатора алгоритма Маркова работает.*

# Описание алгоритма

## Текстовое описание алгоритма

**Сохранение файла**

Начало

Начало

Инициализировать строковый массив

Конец

Начало

Создать путь для записи потока

Конец

Начало

Инициализировать сериализующий класс

Конец

Начало

Записать файл, используя сериализацию

Конец

Начало

Закрыть поток записи

Конец

Конец

**Открытие файла**

Начало

Начало

Создать потом для открытия файла

Конец

Начало

Инициализировать сериализующий класс

Конец

Начало

Восстановить информацию из файла в двумерный массив

Конец

Начало

Возвратить массив

Конец

Конец

**Создание бинарного алфавита**

Начало

Начало

Создать массив символом размерности 2

Конец

Начало

Заполнить элементы массива числами "0" и "1"

Конец

Конец

**Создание латинского алфавита**

Начало

Начало

Создать символьный массив размерности 52

Конец

Начало

Счетчик j = 0

Инициализация счетчика i

От i = "A" до "Z" с шагом +1

Начало

Присвоить j = i

Конец

Конец

Начало

От i = "a" до "z" с шагом +1

Начало

Присвоить j = i

Конец

Конец

Конец

**Создание русского алфавита**

Начало

Начало

Создать символьный массив размерности 66

Конец

Начало

Счетчик j = 0

От i = "А" до "Я" с шагом +1

Присвоить j = i

Конец

Начало

От i = "а" до "я" с шагом +1

Присвоить j = i

Конец

Конец

**Создание своего алфавита**

Начало

Начало

Считать введенную строку с алфавитом в переменную S

Конец

Начало

Удалить пробелы из строки S

Конец

Начало

Создать символьный массив и записать в его элементы посимвольно строку S

Конец

Конец

**Открытие алфавита**

Начало

Начало

Открыть диалоговое окно

Конец

Начало

Создать потом для считывания алфавита из файла

Конец

Начало

Переписать информацию из файла в строку S

Конец

Начало

Удалить пробелы из строки S

Конец

Начало

Создать символьный массив bufMas равный длине строки S

Конец

Начало

Счетчик i = 0

От i = 0 до S - 1 с шагом +1

Начало

Элементу bufMas номера i присваиваем значение символа номера i строки S

Конец

Конец

Начало

Закрыть поток

Конец

Конец

**Удаление строки**

Начало

Начало

Если количество строк больше 1, то

Начало

Уменьшить количество строк на 1

Конец

Начало

Создать двумерный массив символов с количеством строк предыдущего - 1, столбцов 3

Конец

Начало

Счетчик i = 0

От i = 0 до количества строк нового массива с шагом +1

Начало

Счетчик j = 0

От j = 0 до количества столбцов с шагом +1

Начало

Переписать элеметы [i,j] из старого массива в новый

Конец

Конец

Конец

Конец

Конец

**Добавление строки**

Начало

Начало

Увеличить количество строк на 1

Конец

Начало

Создать двумерный массив символов с количеством строк предыдущего +1, столбцов 4

Конец

Начало

Счетчик i = 0

От i = 0 до количества строк нового массива с шагом +1

Начало

Счетчик j = 0

От j = 0 до количества строк столбцов с шагом +1

Начало

Переписать элемент [i,j] из старого массива в новый

Конец

Конец

Конец

Конец

**Проверка на принадлежность к алфавиту**

Начало

Начало

Присвоить логической переменной flag = false

Конец

Начало

Счетчик i = 0

Инициализировать S, как длину поступившей строки

От i = 0 до S с шагом +1

Начало

Присвоить flag = false

Конец

Начало

Счетчик j = 0

От j = 0 до длины алфавита

Начало

Если S[i] = элементу алфавита j

Присвоить Flag=true

Конец

Конец

Конец

Начало

Если Flag=false, то

Начало

Возвратить значение Flag

Конец

Конец

Начало

Вернуть возвращение Flag

Конец

Конец

## Графическое описание алгоритма

Инициализируем строковый массив

Создание пути для записи (потока)

Инициализация сериализующего класса

Запись массива в файл при помощи сериализации

Закрытие потока записи

Начало

Конец

рис 3. 1 Структурограмма (сохранение файла)

Начало

Создание потока для открытия файла

Инициализация сериализующего класса

Восстановление (запись) информации из файла в двумерный массив

Закрытие потока

Возвращаем массив

Конец

рис 3. 2 Структурограмма (открытие файла)

Начало

Создание массива символов размера 2

Заполнение элементов массива символами «0» и «1»

Конец

рис 3. 3 Структурограмма (создание бинарного алфавита)

Начало

Создание символьного массива длинны 52

Счетчик j =0

Конец

Счетчик i от 'A' до ‘Z’, шаг 1

Элементу масс. J присвоить i

Счетчик i от 'a' до ‘z’, шаг 1

Элементу масс. J присвоить i

рис 3. 4 Структурограмма (создание латинского алфавита)

Начало

Создание символьного массива длинны 66

Счетчик j =0

Конец

Счетчик i от 'A' до ‘Я’, шаг 1

Элементу масс. J присвоить i

Счетчик i от 'a' до ‘я’, шаг 1

Элементу масс. J присвоить i

рис 3. 5 Структурограмма (создание русского алфавита)

Начало

Считывание введенной строки с алфавитом в переменную S

Удаляем пробелы из строки S

Создаем символьный массив и записываем в его элементы посимвольно строку S

Конец

рис 3. 6 Структурограмма (создание своего алфавита)

Начало

Открытие диалогового окна

Создание потока для считывания алфавита из файла

Переписываем информацию из файла в строку s

Удаляем пробелы из строки s

Счетчик Iот 0 до длинны S-1, шаг 1

Создаем символьный массивbufMas длинны строки s

Элементу bufMasномера Iприсваиваем значение символа номера Iстроки s

Закрываем поток

Конец

рис 3. 7 Структурограмма (открытие алфавита)

Начало

Уменьшаем кол-во строк на 1

Кол-во строк больше1?

Создаем дв. массив символов с количеством строк предыдущего -1, столбцов 3

Счет. I от 0 до кол-ва строк нов. Массива, шаг 1

Счет. J от 0 до кол-ва столбцов, шаг 1

Переписываем элемент [I,J] из старого массива в новый

Конец

+

-

рис 3. 8 Структурограмма (удаление строки)

Начало

Увеличиваем кол-во строк на 1

Создаем дв. массив символов с количеством строк предыдущего +1, столбцов 4

Счет. Iот 0 до кол-ва строк нов. Массива, шаг 1

Счет. Jот 0 до кол-ва столбцов, шаг 1

Переписываем элемент [I,J] из старого массива в новый

Конец

рис 3. 9 Структурограмма (добавление строки)

Начало

Переменной логического типа flagприсваиваем false

Счетчик Iот 0 до длинны поступившей строки s, шаг 1

Flag присваиваем false

Счетчик jот 0 до длины алфавита

S[i] равноэл. алф. J?

Flag=true

+

-

Flag=false?

Возвращаем значение flag

+

-

Возвращаем значение flag

Конец

рис 3. 10 Структурограмма (проверка на принадлежность к алфавиту)

# Описание порядка использования программы

## Руководство пользователя

Рис. 4.1 (Интерфейс интерпретатора)

В верхней части программы находится поле редактора, в которое можно ввести условие задачи в свободной форме.

Система постановок, задающая нормальный алгорифм Маркова, набирается в таблице в нижней части окна программы.

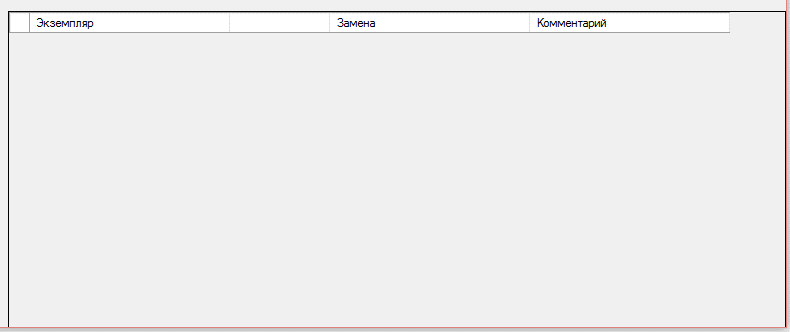
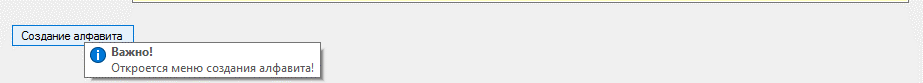
 Но сначала пользователю необходимо выбрать алфавит.После нажатия кнопки, откроется диалоговое окно, где пользователь

Рисунок 10 Таблица для постановок

Рис. 4.2 (Таблицы для подстановок)

Рис. 4.3 (Диалоговое окно)

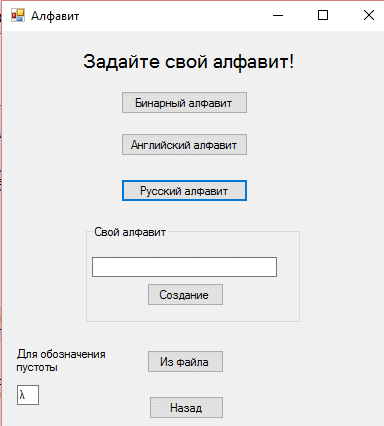
сможет выбрать алфавит из предложенных, придумать алфавит сам или выгрузить из уже заранее подготовленного файла.

Рис. 4.4 (Выбор алфавита)

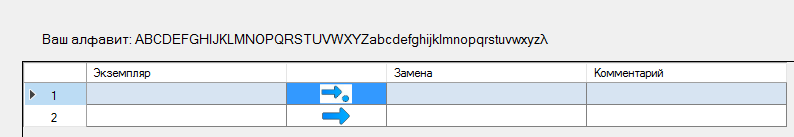
После выбора алфавита в программе откроется возможность редактировать таблицу, в столбце экземпляр вводится то, что нужно заменить, в строке замены: то, на что хочет заменить, а в строке между ними посредством клика можно выбрать условие выполнения данного шага алгоритма (стрелочка с точкой означает завершения работы программы после данного шага), в строке «комментарий» можно указать описание данной замены.

Рис. 4.5 (Вывод вашего алфавита (в данном случае английского))

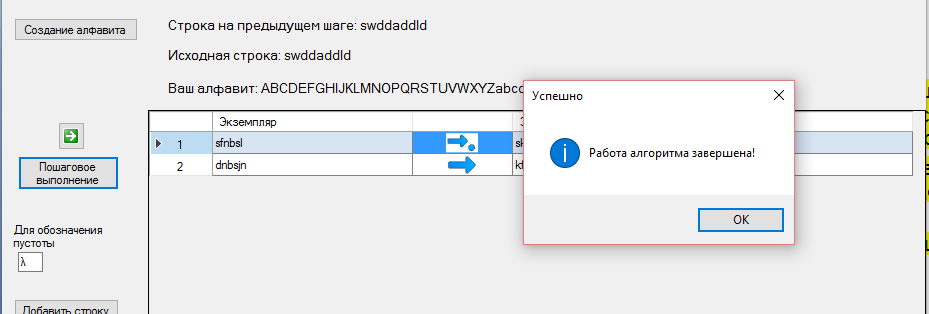
Также пользователь задает рабочую строку, в которую можно вводить только те, элементы, которые есть в выбранном алфавите, иначе программа не позволит пользователю ввести их.

Рис. 4.7 (Пошаговое выполнение)

Рис. 4.6 (Рабочее поле)

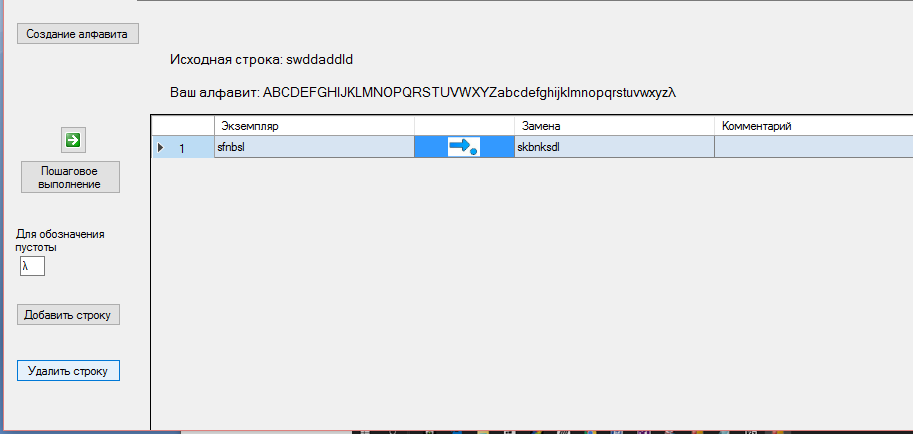
****Пользователь может или запустить программу, нажав на зеленую стрелку, или выполнить ее пошагово как показано на рисунке

Рис. 4.9Удаление строк

Рис. 4.8Добавление строк

Пользователь может как добавить, так и удалять строки в таблице, чтобы задавать больше условий или ограничить пользователя в них, внимание, при запуске программы должны быть заполнены все добавленный строки, иначе программа предложит вам это сделать.

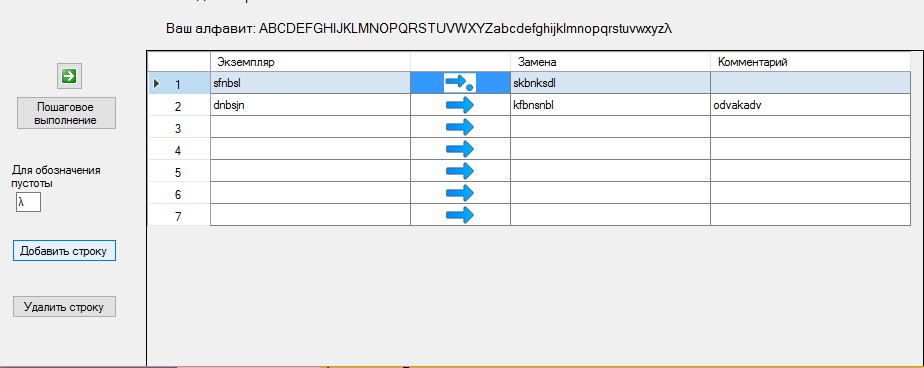
Задачи можно сохранять в файлах и загружать из файлов. Сохраняется условие задачи, исходное слово и система подстановок.

Рис. 4.10Панель для загрузки и выгрузки файлов

# Исходный код программы

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 НАМ

{

public partial class Main : Form

{

public CreateAlp ca = new CreateAlp();

public string[,] masTable = new string[0,4];

public Main()

{

InitializeComponent();

}

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

{

DataGridViewImageColumn img = new DataGridViewImageColumn();

img.Name = "img";

img.HeaderText = "";

img.ValuesAreIcons = true;

EventTable.Columns.Insert(1, img);

WorkStr.Enabled = false;

buttonAddStr.Enabled = false;

buttonDelStr.Enabled = false;

buttonStart.Enabled = false;

buttonStep.Enabled = false;

textBoxEmpty.Text = "λ";

toolStripButtonSave.Enabled = false;

}

#region Interface

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

{

ca.ShowDialog();

if ((ca.mas != null) && (ca.flag))

{

labelAlp.Text = "Ваш алфавит: ";

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

labelAlp.Text += ca.mas[i];

WorkStr.Enabled = true;

WorkStr.Clear();

EventTable.RowCount = 0;

masTable = new string[0, 4];

AddStr1(EventTable);

buttonAddStr.Enabled = true;

buttonDelStr.Enabled = true;

buttonStart.Enabled = true;

buttonStep.Enabled = true;

labelStepBefore.ResetText();

labelStrFirst.ResetText();

textBoxTask.Clear();

}

}

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

{

AddStr1(EventTable);

}

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

{

if (EventTable.RowCount > 1)

{

EventTable.RowCount--;

string[,] bufTable = new string[masTable.GetLength(0) - 1, 3];

for (int i = 0; i < bufTable.GetLength(0); i++)

for (int j = 0; j < bufTable.GetLength(1); j++)

bufTable[i, j] = masTable[i, j];

masTable = bufTable;

}

}

private void AddStr1(DataGridView table)

{

table.RowCount++;

table.Rows[table.RowCount - 1].HeaderCell.Value = (table.RowCount).ToString();

int number\_of\_rows = table.RowCount;

Icon image = Properties.Resources.direction;

table.Rows[number\_of\_rows - 1].Cells["img"].Value = image;

table.Rows[number\_of\_rows - 1].Cells["img"].Tag = "direction";

string[,] bufTable = new string[masTable.GetLength(0) + 1, 4];

for (int i = 0; i < masTable.GetLength(0); i++)

for (int j = 0; j < masTable.GetLength(1); j++)

bufTable[i, j] = masTable[i, j];

masTable = bufTable;

masTable[number\_of\_rows - 1, 1] = "direction";

}

private void EventTable\_CellClick(object sender, DataGridViewCellEventArgs e)

{

if (e.ColumnIndex == 1)

{

if ((string)EventTable.Rows[e.RowIndex].Cells["img"].Tag == "direction")

{

EventTable.Rows[e.RowIndex].Cells["img"].Value = Properties.Resources.arrow\_dot;

EventTable.Rows[e.RowIndex].Cells["img"].Tag = "direction\_dot";

masTable[e.RowIndex, e.ColumnIndex] = "direction\_dot";

}

else

{

EventTable.Rows[e.RowIndex].Cells["img"].Value = Properties.Resources.direction;

EventTable.Rows[e.RowIndex].Cells["img"].Tag = "direction";

masTable[e.RowIndex, e.ColumnIndex] = "direction";

}

}

}

#endregion

#region Stroka

public string mainStr = "";

private bool \_2=true;

private void WorkStr\_TextChanged(object sender, EventArgs e)

{

if (\_2 == true)

{

TextBox btn = (TextBox)sender;

string d = btn.Text;

if (BelongToAlp(d,ca.mas))

mainStr = d;

else

{

\_2 = false;

if (WorkStr.Text != "")

WorkStr.Text = mainStr;

else

mainStr = "";

}

}

else

{

\_2 = true;

}

}

private bool BelongToAlp(string s, char[] ar)

{

bool flag = false;

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

{

flag = false;

for (int j=0; j<ar.Length;j++)

{

if (s[i] == ar[j])

flag = true;

}

if (!flag)

return flag;

}

return flag;

}

#endregion

#region CodeTable

int posTableStep=0;

bool NEVER = true;

private void EventTable\_CellEndEdit(object sender, DataGridViewCellEventArgs e)

{

if (e.ColumnIndex != 3)

{

try

{

if (e.ColumnIndex == 2)

{

if (BelongToAlp((string)EventTable.Rows[e.RowIndex].Cells[e.ColumnIndex].Value, ca.mas))

{

masTable[e.RowIndex, e.ColumnIndex] = (string)EventTable.Rows[e.RowIndex].Cells[e.ColumnIndex].Value;

}

else

{

if ((string)EventTable.Rows[e.RowIndex].Cells[e.ColumnIndex].Value == "λ")

{

masTable[e.RowIndex, e.ColumnIndex] = "";

}

else

{

if ((string)EventTable.Rows[e.RowIndex].Cells[e.ColumnIndex].Value == "")

{

EventTable.Rows[e.RowIndex].Cells[e.ColumnIndex].Value = "λ";

}

else

{

MessageBox.Show("Данные введены неверно!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

EventTable.Rows[e.RowIndex].Cells[e.ColumnIndex].Value = null;

}

}

}

}

else

{

if (BelongToAlp((string)EventTable.Rows[e.RowIndex].Cells[e.ColumnIndex].Value, ca.mas))

{

masTable[e.RowIndex, e.ColumnIndex] = (string)EventTable.Rows[e.RowIndex].Cells[e.ColumnIndex].Value;

}

else

{

MessageBox.Show("Данные введены неверно!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

EventTable.Rows[e.RowIndex].Cells[e.ColumnIndex].Value = "";

}

}

}

catch

{

MessageBox.Show("Данные не введены!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

else

{

masTable[e.RowIndex, e.ColumnIndex] = (string)EventTable.Rows[e.RowIndex].Cells[e.ColumnIndex].Value;

}

}

private void buttonStart\_Click(object sender, EventArgs e) //в поле экземпляр запретить вводить пустоту

{

labelStrFirst.Text = "Исходная строка: " + mainStr;

bool flag = true;

for (int i = 0; i < masTable.GetLength(0); i++)

{

for (int j = 0; j < masTable.GetLength(1) - 1; j++)

{

if (masTable[i, j] == null)

flag = false;

if (masTable[i, j] == "")

EventTable.Rows[i].Cells[j].Value = "λ";

}

}

if (flag)

{

try

{

int posTable = 0;

if (mainStr != "")

{

int circle = 0;

int OldPosTable;

do

{

circle++;

OldPosTable = posTable;

int index = mainStr.IndexOf(masTable[posTable, 0]);

if (index != -1)

{

mainStr = mainStr.Remove(index, masTable[posTable,0].Length);

mainStr = mainStr.Insert(index, masTable[posTable, 2]);

posTable = 0;

}

else

{

posTable++;

}

} while ( (posTable < masTable.GetLength(0)) && (masTable[OldPosTable, 1] != "direction\_dot") && (circle<10000) );

if (circle >= 10000)

MessageBox.Show("Ваш алгоритм зациклился или произошло переполнение!", "Внимание", MessageBoxButtons.OK, MessageBoxIcon.Warning);

else

{

WorkStr.Text = mainStr;

MessageBox.Show("Работа алгоритма завершена!", "Успешно", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

else

{

MessageBox.Show("Сначала введите слово в рабочую строку!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

catch

{

MessageBox.Show("Пожалуйста, заполните все ячейки с командами или удалите не нужные!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

else

{

MessageBox.Show("Пожалуйста, заполните все ячейки с командами или удалите не нужные!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

if (NEVER)

{

labelStrFirst.Text = "Исходная строка: " + mainStr;

NEVER = false;

}

string lsb = labelStepBefore.Text;

labelStepBefore.Text = "Строка на предыдущем шаге: " + mainStr;

bool flag = true;

int oldPosTableStep;

for (int i = 0; i < masTable.GetLength(0); i++)

{

for (int j = 0; j < masTable.GetLength(1) - 1; j++)

{

if (masTable[i, j] == null)

flag = false;

if (masTable[i, j] == "")

EventTable.Rows[i].Cells[j].Value = "λ";

}

}

if (flag)

{

try

{

if (mainStr != "")

{

bool exit = false;

do

{

int index = mainStr.IndexOf(masTable[posTableStep, 0]);

oldPosTableStep = posTableStep;

if (index != -1)

{

mainStr = mainStr.Remove(index, masTable[posTableStep, 0].Length);

mainStr = mainStr.Insert(index, masTable[posTableStep, 2]);

EventTable.Rows[posTableStep].Selected = true;

posTableStep = 0;

exit = true;

}

else

{

posTableStep++;

}

} while (!exit && posTableStep < masTable.GetLength(0));

WorkStr.Text = mainStr;

if ((masTable[oldPosTableStep, 1] == "direction\_dot") || (posTableStep > masTable.GetLength(0)-1))

{

MessageBox.Show("Работа алгоритма завершена!", "Успешно", MessageBoxButtons.OK, MessageBoxIcon.Information);

posTableStep = 0;

labelStrFirst.Text = "Исходная строка: " + mainStr;

labelStepBefore.Text = "";

NEVER = true;

}

}

else

{

if (lsb == "")

{

MessageBox.Show("Сначала введите слово в рабочую строку!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

labelStepBefore.Text = "";

}

else

{

string[] tr = lsb.Split(':');

tr[tr.Length - 1] = tr[tr.Length - 1].Remove(0, 1);

bool end = false;

for (int i1 = 0; i1 < masTable.GetLength(0); i1++)

if ((masTable[i1, 0] == tr[tr.Length - 1]) && (masTable[i1, 2] == ""))

end = true;

if (end)

{

MessageBox.Show("Работа алгоритма завершена!", "Успешно", MessageBoxButtons.OK, MessageBoxIcon.Information);

posTableStep = 0;

labelStrFirst.Text = "Исходная строка: " + mainStr;

labelStepBefore.Text = "";

NEVER = true;

}

else

{

MessageBox.Show("Сначала введите слово в рабочую строку!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

labelStepBefore.Text = "";

}

}

}

}

catch

{

MessageBox.Show("Пожалуйста, заполните все ячейки с командами или удалите не нужные!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

labelStepBefore.Text = "";

}

}

else

{

MessageBox.Show("Пожалуйста, заполните все ячейки с командами или удалите не нужные!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

labelStepBefore.Text = "";

}

}

private bool CellBelongToAlp(string s, char[] ar)

{

bool flag = false;

char[] bufAr = new char[ar.Length + 1];

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

{

bufAr[i] = ar[i];

}

bufAr[bufAr.Length - 1] = 'λ';

ar = bufAr;

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

{

flag = false;

for (int j = 0; j < ar.Length; j++)

{

if (s[i] == ar[j])

flag = true;

}

if (!flag)

return flag;

}

return flag;

}

#endregion

#region file

string path;

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

{

OpenFileDialog sfd = new OpenFileDialog();

sfd.Filter = "bin files (\*.bin)|\*.bin";

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

{

WriteRead wr = new WriteRead();

path = sfd.FileName;

masTable = new string[0, 4];

string[,] bufMasTable = wr.OpenFile(path);

labelAlp.Text = "Ваш алфавит: ";

EventTable.RowCount = 0;

EventTable.ColumnCount = bufMasTable.GetLength(1);

for (int i = 0; i < bufMasTable.GetLength(0)-1; i++)

{

AddStr1(EventTable);

for (int j = 0; j < bufMasTable.GetLength(1); j++)

{

masTable[i, j] = bufMasTable[i, j];

if (j != 1)

{

EventTable.Rows[i].Cells[j].Value = masTable[i, j];

}

else

{

if (masTable[i, j] == "direction\_dot")

{

EventTable.Rows[i].Cells[j].Value = Properties.Resources.arrow\_dot;

}

}

}

}

toolStripButtonSave.Enabled = true;

ca.mas = new char[bufMasTable[bufMasTable.GetLength(0) - 1, 2].Length];

for (int i = 0; i < ca.mas.GetLength(0); i++)

{

ca.mas[i] = bufMasTable[bufMasTable.GetLength(0) - 1, 2][i];

labelAlp.Text += ca.mas[i];

}

textBoxTask.Text = bufMasTable[bufMasTable.GetLength(0) - 1, 0];

mainStr= bufMasTable[bufMasTable.GetLength(0) - 1, 1];

WorkStr.Text = mainStr;

WorkStr.Enabled = true;

buttonStart.Enabled = true;

buttonStep.Enabled = true;

buttonAddStr.Enabled = true;

buttonDelStr.Enabled = true;

labelStepBefore.Text = "";

labelStrFirst.Text = "Исходная строка:";

}

}

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

{

if (path != null)

{

WriteRead wr = new WriteRead();

string[,] bufMasTable = new string[masTable.GetLength(0) + 1, masTable.GetLength(1)];

for (int i = 0; i < masTable.GetLength(0); i++)

for (int j = 0; j < masTable.GetLength(1); j++)

{

bufMasTable[i, j] = masTable[i, j];

}

bufMasTable[masTable.GetLength(0), 0] = textBoxTask.Text; //условие

bufMasTable[masTable.GetLength(0), 1] = mainStr; //рабочая строка

for (int i = 0; i < ca.mas.Length; i++) //алфавит

bufMasTable[masTable.GetLength(0), 2] += ca.mas[i];

wr.CreateFile(bufMasTable, path);

}

else

MessageBox.Show("Сначала откройте файл!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

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

{

SaveFileDialog sfd = new SaveFileDialog();

sfd.Filter = "bin files (\*.bin)|\*.bin";

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

{

WriteRead wr = new WriteRead();

path = sfd.FileName;

string[,] bufMasTable = new string[masTable.GetLength(0)+1, masTable.GetLength(1)];

for (int i=0; i<masTable.GetLength(0);i++)

for(int j=0; j<masTable.GetLength(1);j++)

{

bufMasTable[i, j] = masTable[i, j];

}

bufMasTable[bufMasTable.GetLength(0)-1, 0] = textBoxTask.Text; //условие

bufMasTable[bufMasTable.GetLength(0)-1, 1] = mainStr; //рабочая строка

for (int i = 0; i < ca.mas.Length; i++) //алфавит

bufMasTable[masTable.GetLength(0), 2] += ca.mas[i];

wr.CreateFile(bufMasTable, path);

}

}

#endregion

}

public partial class CreateAlp : Form

{

public char[] mas;

public bool flag = true;

public CreateAlp()

{

InitializeComponent();

}

private void alph1\_Click(object sender, EventArgs e) //бинарный алфавит

{

mas = new char[2];

mas[0] = '0';

mas[1] = '1';

flag = true;

Hide();

}

private void alph2\_Click(object sender, EventArgs e) //Английский алфавит

{

mas = new char[53];

int j = 0;

for (char i = 'A'; i <= 'Z'; i++, j++)

{

mas[j] = i;

}

for (char i = 'a'; i <= 'z'; i++, j++)

{

mas[j] = i;

}

mas[52]= 'λ';

flag = true;

Hide();

}

private void alph3\_Click(object sender, EventArgs e) //Русский алфавит

{

mas = new char[67];

int j = 0;

for (char i = 'А'; i <= 'Я'; i++, j++)

{

mas[j] = i;

}

for (char i = 'а'; i <= 'я'; i++, j++)

{

mas[j] = i;

}

mas[64] = 'ё';

mas[65] = 'Ё';

mas[66] = 'λ';

flag = true;

Hide();

}

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

{

string s = CreateMasText.Text;

char[] bufMas = s.ToCharArray();

Array.Sort(bufMas); //сортировка для удобства

bufMas = bufMas.Distinct().ToArray(); //избавление от пробелов лишних

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

if (bufMas[i] == ' ')

bufMas[i] = 'λ';

mas = bufMas;

flag = true;

Hide();

}

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

{

try

{

OpenFileDialog ofd = new OpenFileDialog();

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

{

StreamReader sr = new StreamReader(ofd.FileName, Encoding.Default);

string s = sr.ReadToEnd();

char[] bufMas = new char[s.Length];

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

bufMas[i] = s[i];

Array.Sort(bufMas);

bufMas = bufMas.Distinct().ToArray();

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

if (bufMas[i] == ' ')

bufMas[i] = 'λ';

mas = bufMas;

flag = true;

Hide();

sr.Close();

}

}

catch

{

MessageBox.Show("Ваш алфавит был введён неверно!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

mas = null;

}

}

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

{

Hide();

flag = false;

}

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

{

textBoxEmpty.Text = "λ";

}

}

[Serializable]

public partial class WriteRead

{

public string[] mas1;

public void CreateFile(string[,] mas, string path)

{

FileStream fs = new FileStream(path, FileMode.Create);

BinaryFormatter w = new BinaryFormatter();

w.Serialize(fs, mas); // Запись

fs.Close();

}

public string[,] OpenFile(string path)

{

FileStream f2 = new FileStream(path, FileMode.Open);

BinaryFormatter bf = new BinaryFormatter();

string[,] mas = (string[,])bf.Deserialize(f2);// восстановление объекта

f2.Close();

return mas;

}

}

}

# Заключение

За время работы над интерпретатором мы узнали о различных способах формализации алгоритмов. Мы прошли все основные начальные этапы разработки программного обеспечения, разделив роли в командном проекте между собой. Во время работы мы усовершенствовали свои навыки анализа задач, разработки алгоритмов и их решений, работу со структурой данных и работы в команде над проектом. Нам удалось осуществить почти все задачи, которые мы ставили перед и собой

# Библиографический список

1. Методические рекомендации по подготовке и защите курсовых работ: Методические указания / НИУ ВШЭ – Пермь; Викентьева О.Л., Дерябин А.И., Лебедев В.В., Лядова Л.Н., Шестакова Л.В.,Корчагина В.П. – Пермь, 2013. – 18 с.
2. Лебедев В.В.Информационные технологии в офисной работе. Создание электронных документов с помощью текстового процессора MicrosoftWord 2007. НИУ ВШЭ – Пермь, Кафедра информационных технологий в бизнесе, 2010 г. – 218 с.
3. Т. А. Павловская "C# Программирование на языке высокого уровня" + cайт: [https://msdn.microsoft.com/ru-ru/library/67ef8sbd.aspx](https://vk.com/away.php?utf=1&to=https%3A%2F%2Fmsdn.microsoft.com%2Fru-ru%2Flibrary%2F67ef8sbd.aspx)

# Приложения









