

Правительство Санкт-Петербурга
Комитет по науке и высшей школе
Санкт-Петербургское государственное бюджетное
профессиональное образовательное учреждение
«Политехнический колледж городского хозяйства»

КУРСОВАЯ РАБОТА
по МДК

МДК 01.01 Разработка программных модулей
наименование дисциплины/ ПМ/ МДК

09.02.07 Информационные системы и программирование программист
№ и наименование специальности

Разработка приложения для посимвольного частотного анализа текста
тема курсовой работы

Студент группы ИП-19-3

Новоселов С.Д.

Руководитель

Андреев В.А.

Председатель П(Ц)К

Левит Л.В.

Оценка _____

« _____ » _____ 2022г.

подпись *расшифровка*

Санкт-Петербург
2022 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	3
1 Теоретические основы разработки	4
1.1 Описание предметной области	4
1.2 Анализ методов решения	4
1.3 Обзор средств программирования	5
1.4 Описание языка программирования	7
2 Практическая часть	8
2.1 Постановка задачи	8
2.2 Описание схем	8
2.4 Текст программы	13
2.5 Руководство оператора	13
2.4 Программа и методика испытаний	13
2.5 Протокол испытаний	14
ЗАКЛЮЧЕНИЕ	17
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	18
ПРИЛОЖЕНИЕ А	19
ПРИЛОЖЕНИЕ Б	25
ПРИЛОЖЕНИЕ В	31

ВВЕДЕНИЕ

До XX века в криптоанализе широко применялся метод частотного анализа для дешифровки текстов. Он предполагает, что частота появления заданной буквы алфавита в разных текстах одного и того же шифра (или языка) будет одинаковой. Если в шифротексте будет символ с аналогичной вероятностью появления, то можно сказать, что он и есть та заданная буква [1]. Большинство современных алгоритмов шифрования текстов стали устойчивыми к частотному криптоанализу, поэтому он применяется только для обучения молодых криптографов. Для более эффективного изучения частотного анализа, необходимо разработать программу, которая автоматизирует процесс подсчета символов и покажет наиболее часто встречающиеся из них, что и является целью данного курсового проекта. Для достижения поставленной цели, необходимо решить ряд задач:

- а) Проанализировать предметную область;
- б) Исследовать аналоги программно-инструментальных средств разработки;
- в) Спроектировать программный модуль;
- г) Разработать программный продукт;
- д) Провести испытания программного продукта.

Объект исследования – криптоанализ - подраздел криптографии. Предмет исследования – применение метода частотного анализа текста в криптографии. В результате выполнения данного курсового проекта, будет создана программа для посимвольного анализа текста, включающая в себя не только текстовый отчет, но и графический. Программа может быть использована криптографами для расшифровки текстов и для получения статистической информации.

1 Теоретические основы разработки

1.1 Описание предметной области

Частотный анализ текста проводится на основе предположения, что для каждого языка существует собственное распределение частоты повторения букв – частотность. Частотностью символов в тексте называют отношение количество вхождений данного экземпляра к общему количеству всех символов текста. Зная частотность символов шифротекста и языка, с которого он был написан можно провести криптоанализ для расшифровки заданного текста.

1.2 Анализ методов решения

Существует несколько методов реализации посимвольного частотного анализа:

а) Ручной

Данный способ является самым время- и ресурсозатратным, так как предполагает подсчет символов, используя человеческий ресурс.

б) Полуавтоматический

Данный способ предполагает использование программных средств для частотного анализа текста. Занимает значительно меньше времени, но присутствуют затраты на ввод текста пользователем.

в) Автоматический

Наиболее эффективный метод реализации криптоанализа относительно времязатрат. Способ предполагает ввод заранее подготовленного текстового файла в программу и автоматическое извлечение текста для анализа из него.

При использовании ручного способа криптоанализа текста, данные можно сохранять на бумажном носителе, в текстовых редакторах или

электронных таблицах, а с помощью остальных методов приемы получения данных частотного анализа зависят от структуры программы. Среди них выделяют: сохранение в текстовый файл, в электронную таблицу, в базу данных, печать на бумажный носитель и другие.

Для реализации курсового проекта наиболее подходящим является совмещение автоматического метода с полуавтоматическим, т.е. предусмотреть возможность ввода данных как с клавиатуры, так и чтением файла.

1.3 Обзор средств программирования

Для реализации поставленной цели, существует множество средств. Самые подходящие из них: Windows Forms (C#), Windows Presentation Foundation (WPF C#) и Flutter.

а) Windows Forms (C#)

Windows Forms – это платформа пользовательского интерфейса для создания классических приложений Windows. Является самой простой платформой для разработки. Приложения на Windows Forms состоят из одной или нескольких форм (окон на экране), которые содержат элементы управления, с которыми непосредственно взаимодействует пользователь. При нажатии на элементы управления вызываются события, которые отвечают за логику программы. Поэтому можно сказать, что Windows Forms – приложения в первую очередь управляются событиями. Платформа поддерживается с появления .Net Framework и доступна в новейших версиях, но только в режиме обслуживания. Это означает, что новая функциональность не добавляется, только исправляются старые ошибки. [2]

б) Windows Foundation Presentation (WPF C#)

Технология WPF является следующей ступенью эволюции Windows-

приложений. Платформа спроектирована для .NET под влиянием таких технологий отображения, как HTML и Flash, и использует аппаратное ускорение. Основное отличие WPF от Windows Forms заключается в способе отображения графики. В Windows Presentation Foundation реализован «новый» (на то время) графический механизм – пакет DirectX. Он позволяет полностью переложить визуальную часть приложения на видеокарту. Это открывает новые возможности для оформления пользовательского интерфейса приложений и рисования на страницах. Также в WPF реализована новая система построения структуры пользовательского интерфейса – расширяемый язык разметки приложений или XAML. Данный язык разметки позволяет создавать базовый пользовательский интерфейс в Microsoft Visual Studio, а затем по необходимости улучшать его графическую составляющую, используя сторонние программные средства, например, Expression Blend [3].

в) Flutter

Flutter – это современный бесплатный и открытый набор средств разработки мобильного пользовательского интерфейса. Совместно с Flutter используется язык программирования Dart. Он фокусируется на развитии верстки веб-страниц. Flutter & Dart предоставляют большое количество виджетов для создания многофункциональных приложений, а также широкие возможности для их персонализации для создания оригинального пользовательского интерфейса [4].

Исходя из данного подробного описания возможностей для реализации программы, была выбрана платформа WPF с языком программирования C#. Единственная среда разработки, которая предназначена для создания WPF-приложений – это Microsoft Visual Studio (Версия Community 2022).

1.4 Описание языка программирования

В пункте 1.3 был выбран язык программирования C# для реализации поставленной цели. C# - это современный объектно-ориентированный и типобезопасный язык программирования. На данный момент является одним из самых мощных, быстро развивающихся и востребованных языков в отрасли. Объектно-ориентированный подход позволяет решить задачи по построению крупных, но в тоже время гибких, масштабируемых и расширяемых приложений [5]. Синтаксис языка практически не отличается от языков семейства Си (C++, Java). Объектно-ориентированным языкам свойственно иметь две большие категории типов: типы которые присущи языку и классы, которые программист может создать самостоятельно [6]. Стандартные типы языка C# приведены в таблице 1.

Таблица 1 – Основные типы в языке программирования C#

Тип	Описание
object	Класс, базовый для всех типов
string	Строка
sbyte	8-ми разрядный байт (со знаком)
byte	8-ми разрядный байт (без знака)
short	16-и разрядное число (со знаком)
ushort	16-и разрядное число (без знака)
int	32-и разрядное число (со знаком)
uint	32-и разрядное число (без знака)
long	64-и разрядное число (со знаком)
ulong	64-и разрядное число (без знака)
char	16-и разрядный символ (Unicode)
float	32-ух разрядное число с плавающей точкой
double	64-ух разрядное число с плавающей точкой
bool	Булева переменная
decimal	128-и разрядное число с плавающей точкой (со знаком)

2 Практическая часть

2.1 Постановка задачи

Согласно цели курсового проекта, необходимо разработать программу для посимвольного частотного анализа текста. В программе должно быть реализована возможность ввода проверяемого текста с помощью текстового поля и с использованием стандартных средств Windows – диалогового окна выбора файла (текстовый файл в формате .txt). Выходная информация содержит полный список, содержащий все символы текста, и графическую информацию – столбчатую и круговую диаграммы.

2.2 Описание схем

На рисунке 1 изображена структурная схема приложения, которая наглядно показывает какие функции реализованы в нем.

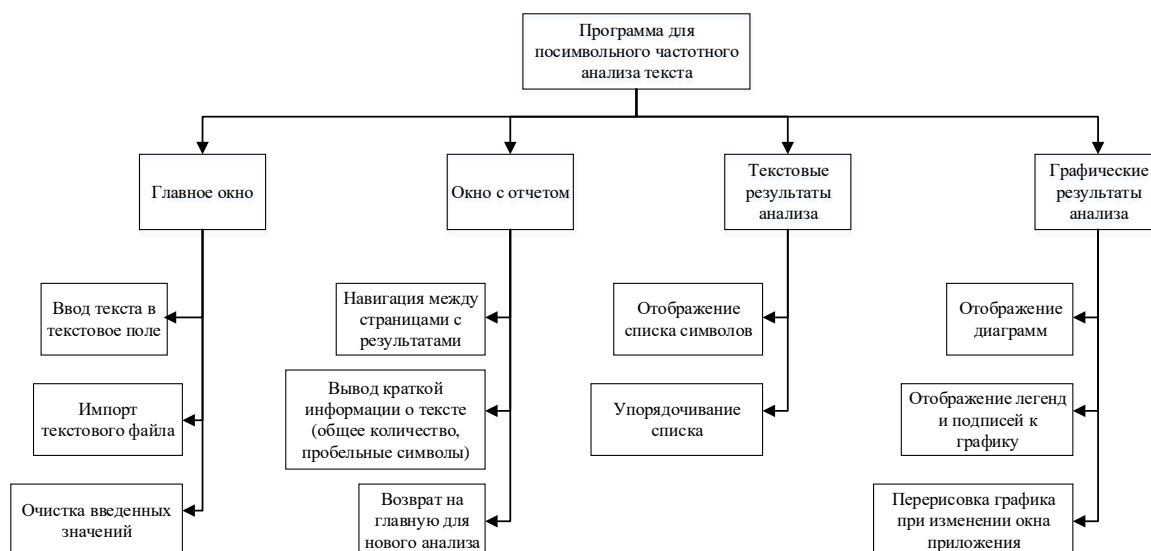


Рисунок 1 – Структурная схема

На рисунке 2 изображена диаграмма последовательностей, которая представляет все доступные функции в порядке их воспроизведения для пользователя.



Рисунок 2 – Диаграмма последовательностей

2.3 Описание программы

— Программное обеспечение необходимое для функционирования программы

Для полноценного функционирования программы, необходимо наличие ОС Microsoft Windows 7 или выше. Также необходимо наличие расширяемого пакета Microsoft Visual C++ 2003-2021 и пакета .NET версии 5.0.

— Языки программирования, на которых написана программа

Программа написана с использованием технологии Windows Presentation Foundation на языке программирования C# и расширяемом языке разметки приложений XAML.

— Описание логической структуры

Алгоритм программы представлен в виде блок-схемы на рисунке 3. Программа не имеет связей с другими программами.

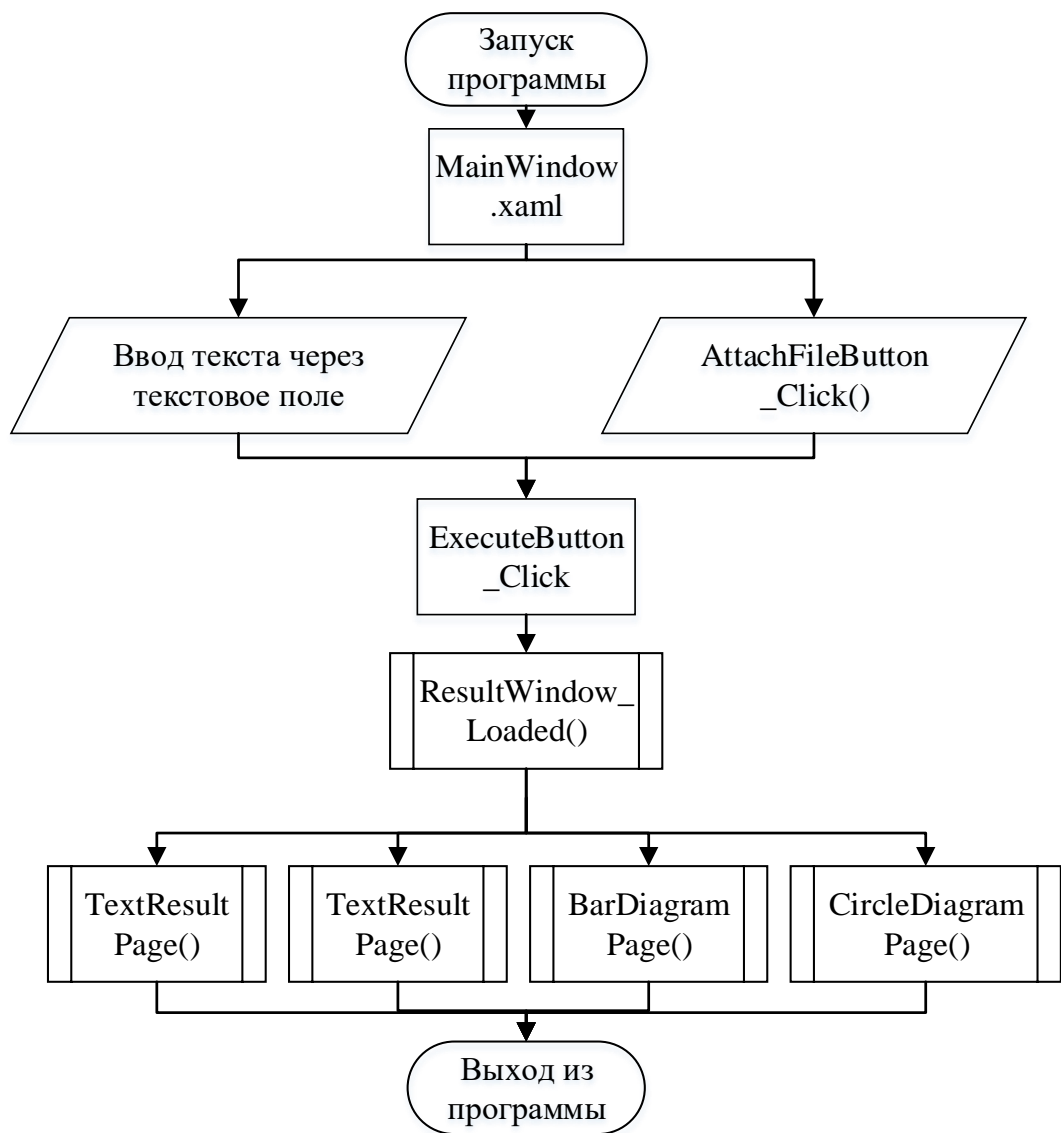


Рисунок 3 – Алгоритм программы

В процессе написания программного кода были созданы следующие методы:

а) MainWindow() – конструктор окна MainWindow.xaml

- 1) ClearTextAreaButton_Click() – событие, которое происходит при нажатии на кнопку «Очистить поле ввода»
- 2) CancelFileButton_Click() – событие, которое происходит при нажатии на кнопку «Открепить текстовый файл»
- 3) AttachFileButton_Click() – событие, которое происходит при

нажатии на кнопку «Прикрепить текстовый файл»

4) ExecuteButton_Click() – событие, которое происходит при нажатии на кнопку «Выполнить»

5) ClearInfo() – Очистка переменных, отвечающих за хранение текста.

б) ResultWindow() – конструктор окна ResultWindow.xaml

1) ShowTextResultButton_Click() – отобразить страницу TextResult.xaml

2) ShowBarDiagramButton_Click() – отобразить страницу BarDiagram.xaml

3) ShowCircleDiagramButton_Click() – отобразить страницу CircleDiagram.xaml

4) WindowLoaded() – событие, которое происходит при первом отображении окна

5) CreateReport() – создает текстовый файл с отчетом по анализу.

в) BarDiagramPage() – конструктор окна BarDiagramPage.xaml

г) CircleDiagramPage() – конструктор окна CircleDiagramPage.xaml

д) TextResultPage() – конструктор окна TextResultPage.xaml

— Структура программы с описанием функций составных частей и связи между ними

При запуске программы пользователю открывается окно MainWindow, с помощью которого он может ввести текст для анализа вручную, используя элемент InputTextBox, или прикрепить текстовый файл с расширением .txt, используя элемент OpenFileDialog.

По нажатию кнопки `ExecuteButton` открывается окно `ResultWindow`, главным элементом которого является `Frame`. Данный элемент позволяет выводить страницы внутри окна. Используя навигационные кнопки сверху окна можно выбрать, какую страницу отображать в данный момент. На выбор доступно всего три страницы: `TextResultPage`, `BarChartPage`, `PieChartPage`.

`TextResultPage` содержит элемент `ListView`, который показывает результаты анализа в виде списка (символ, его количество в тексте, частота появления в процентах). `BarChartPage` показывает гистограмму по первым 10 часто встречаемым символам. `PieChartPage` отображает аналогичную информацию в виде круговой диаграммы.

Также после выполнения криптоанализа текста, создается отчет в виде текстового файла (`.txt`), где дублируется текстовая информация, а также записывается дата и время выполнения анализа. Программа рассчитана на анализ нескольких текстов за один сеанс. Для этого необходимо закрыть окно `ResultWindow`, и ввести другой текст или прикрепить другой файл.

— Используемые технические средства

В состав используемых технических средств входит: 2-ух ядерный процессор Intel I3 или лучше, ОЗУ не менее 10Мб, свободное место на жестком диске не менее 20Мб, монитор, клавиатура и манипулятор «мышь».

— Вызов и загрузка

Вызов программы осуществляется с помощью иконки `TextAnalysis` на рабочем столе или приложения `TextAnalysis.exe` в корневой папке.

— Входные данные

Входными данными для программы служит текст, который пользователь может ввести в специальном текстовом поле или используя стандартное диалоговое окно `Windows` для открытия файлов.

— Выходные данные

Результатом работы программы является текстовый файл (txt) отчета, который содержит всю информацию об анализе текста.

2.4 Текст программы

Листинг программы, созданной для данного курсового проекта, приведен в соответствии с установленным ГОСТ ЕСПД 19.401-78 «Текст программы» в Приложении В.

2.5 Руководство оператора

Руководство оператора составлено в соответствии с установленным ГОСТ ЕСПД 19.505-79 «Руководство оператора» и приведено в Приложении Б.

2.4 Программа и методика испытаний

Чтобы наиболее эффективно протестировать программу для частотного анализа текста, необходимо воспользоваться тестированием черного ящика. Данный вид тестирования направлен на проверку функциональности программы и не предусматривает знание пользователем кода и логики программы. Программа испытаний включает в себя:

а) Анализ короткого текста (< 20 слов)

Данное испытание предназначено для проверки правильности счета программы. Перед началом тестирования пользователь может подсчитать количество символов вручную, а затем сравнить результат с результатом анализа программы.

б) Анализ больших документов (>10 страниц текста)

Данное испытание предназначено для проверки программы на

устойчивость к большим объемам данных и специальным прописным и пробельным символам.

в) Проверка составленных диаграмм

Данное испытание предназначено для проверки правильности составленных диаграмм в отчете к анализу текста. Пользователь может использовать онлайн ресурсы для проверки правильности отображения диаграмм.

Все испытания программы должны проводиться в соответствии с протоколом испытаний, приведенном ниже. Тестовое окружение состоит из операционной системы ОС Windows 10 и предустановленного пакета .NET 5.0 Runtime.

2.5 Протокол испытаний

а) Цель испытаний

Испытания программы для частотного анализа текста проводятся с целью проверки программы на отказоустойчивость, выявление изъянов в ее работе и определения соответствия программного продукта с техническим заданием.

б) Сведения о продолжительности испытаний

Испытания приложения должны проводится на завершающих этапах разработки программного продукта, т.е. с 16.05.2022 по 22.05.2022.

в) Программа испытаний

Программа испытаний программного продукта составлена в виде тест-кейсов и приведена в таблице 2.

Таблица 2 – Программа испытаний

Наименование испытываемого модуля	Суть испытания	Шаги по выполнению испытания	Ожидаемый результат
Главное меню	Импорт файла	а) Запустить программу б) Кликнуть по кнопке «Прикрепить текстовый файл» в) Выбрать любой файл в формате .txt г) Нажать кнопку «Выполнить»	Текстовое поле должно стать доступным только для чтения, а его содержание должно отображать путь указанного файла
Главное меню	Анализ текста, введенного вручную	а) Запустить программу б) Ввести случайный текст в поле в) Кликнуть по кнопке «Выполнить»	Программа должна считать текст из поля и перейти в следующее окно (с отчетом)
Окно с отчетом	Анализ короткого текста	а) Запустить программу б) Ввести короткий текст (<20 символов) с заранее известным количеством символов в) Кликнуть по кнопке «Выполнить»	Количество введенных символов должно совпадать с подсчитанным программой
Окно с отчетом	Анализ больших документов	а) Запустить программу б) Скопировать и вставить в текстовое поле текст данной пояснительной записки в) Кликнуть по кнопке «Выполнить»	В списке символов об анализе не должно содержаться «пустых» (специальных и пробельных) символов или «битых» символов (с неправильной кодировкой)

Продолжение таблицы 2

Наименование испытываемого модуля	Суть испытания	Шаги по выполнению испытания	Ожидаемый результат
Окно с отчетом	Просмотр диаграмм	а) Запустить программу б) Ввести текст с любым содержанием в поле или импортировать файл в) Кликнуть по кнопке «Выполнить» г) Открыть страницы с диаграммами	Диаграммы не должны заходить за края графиков, при изменении размеров окно графики соответственно меняют свой масштаб

г) Результаты испытаний

В процессе проведения испытаний программы для посимвольного анализа текста не было замечено серьезных проблем, приводящих к аварийному завершению программы. Все подсчеты проводятся верно, графики отображаются корректно.

ЗАКЛЮЧЕНИЕ

В результате выполнения исследовательской работы, была достигнута цель курсового проекта, а именно: создание программного продукта для посимвольного частотного анализа текста. Были поставлены основные задачи на реализацию продукта и доказана его актуальность. Проработаны функции приложения, которые помогут молодым криптографам обучаться более эффективно. Во время разработки программы для анализа текста были подробно изучены язык XAML и технология WPF, а именно новые способы рисования на страницах, паттерны для создания пользовательских оформлений элементов управления.

Для улучшения программы для посимвольного частотного анализа текстов, можно предусмотреть следующие улучшения:

- а) Возможность импорта не только текстовых файлов (.txt), но и документы MS Word (.docx)
- б) Возможность вывода на печать диаграмм или текстового отчета об анализе
- в) Предусмотреть возможность импорта файлов, использующих кодировку отличную от UTF-8

По итогам курсового проектирования была разработана программа для посимвольного частотного анализа текста, обеспечивающая необходимый функционал для быстрого и эффективного криптоанализа шифротекстов, который имеет потенциал на развитие и улучшение.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- 1) Википедия. Частотный анализ [Электронный ресурс]. Режим доступа -
https://ru.wikipedia.org/wiki/%D0%A7%D0%B0%D1%81%D1%82%D0%BE%D1%82%D0%BD%D1%8B%D0%B9_%D0%B0%D0%BD%D0%B0%D0%BB%D0%B8%D0%B7
- 2) Learn Tutorials. Учебник WinForms [Электронный ресурс]. Режим доступа -
<https://learntutorials.net/ru/winforms/topic/1018/%D0%BD%D0%B0%D1%87%D0%B0%D0%BB%D0%BE-%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D1%8B-%D1%81-winforms>
- 3) Мак-Дональд М. Windows Presentation Foundation в .NET 4 с примерами на C# 2010 для профессионалов / М. Мак-Дональд – Вильямс, 2011. – 1020 с.
- 4) Википедия. Flutter [Электронный ресурс]. Режим доступа -
<https://ru.wikipedia.org/wiki/Flutter>
- 5) Фаронов В.В. Создание приложений с помощью C# / В.В. Фаронов – Эксмо, 2008. – 573 с.
- 6) Шилдт Г. Полное руководство C#4.0 / Г. Шилдт – Вильямс, 2011. – 1055 с.
- 7) Полное руководство по языку программирования C# 10 и платформе .NET 6 [Электронный ресурс]. – Режим доступа: <https://metanit.com/sharp/tutorial/>
- 8) Документация по C# [Электронный ресурс]. – Режим доступа: <https://docs.microsoft.com/ru-ru/dotnet/csharp/>
- 9) Программирование на C, C# и Java [Электронный ресурс]. – Режим доступа: <http://vscode.ru/category/prog-lessons/c-sharp>

ПРИЛОЖЕНИЕ А
Техническое задание

Правительство Санкт-Петербурга
Комитет по науке и высшей школе
Санкт-Петербургское государственное бюджетное
профессиональное образовательное учреждение
«Политехнический колледж городского хозяйства»

СОГЛАСОВАНО
Председатель ПЦК
09.02.07 Информационные
системы и
программирование

_____ Левит Л.В.
_____._____. 2022

ПРОГРАММА ДЛЯ ПОСИМВОЛЬНОГО АНАЛИЗА ТЕКСТА
Техническое задание
Листов 5

ПРОВЕРИЛ
Преподаватель

_____ Андреев В.А.
21.04. 2022

ВЫПОЛНИЛ
Студент группы ИП-19-3

_____ Новоселов С.Д.
_____._____. 2022

2022

А.1. ВВЕДЕНИЕ

А.1.1. Полное наименование программной разработки: «ПРОГРАММА ДЛЯ ПОСИМВОЛЬНОГО АНАЛИЗА ТЕКСТА».

А.1.2. Программа для посимвольного анализа текста: человек вводит текст (или текстовый файл), программа выдает общее количество символов в нем, текстовую и графическую информацию об их содержании.

А.1.3. Программа предназначена для научно-исследовательских целей в области криптографии.

А.2. ОСНОВАНИЕ ДЛЯ РАЗРАБОТКИ

Разработка ведётся на основании задания к курсовому проекту по профессиональному модулю ПМ.01 «Разработка модулей программного обеспечения для компьютерных систем» МДК 01.01 «Разработка программных модулей» и согласовано Председателем предметно-цикловой комиссии отделения информационных технологий Политехнического колледжа городского хозяйства

А.3. НАЗНАЧЕНИЕ РАЗРАБОТКИ

А.3.1. Основное назначение программного продукта: компьютерная программа для анализа текстов.

А.3.2. Эксплуатационное назначение программного продукта: программа предназначена для широкого круга пользователей, без ограничения по возрасту, не требует внесения денежных средств или использования платёжных систем, предназначена для научно-исследовательских целей.

А.4. ТРЕБОВАНИЯ К РАЗРАБОТКЕ

А.4.1. Требования к функциональным характеристикам:

- Программа должна принимать на вход текст или текстовый файл;
- После проведения анализа программа должна выводить подробную информацию о тексте: общее количество символов, количество каждого символа по отдельности и их процентное соотношение;
- Программа должна предоставлять графическую информацию о содержании символов о введенном тексте: круговые и столбчатые диаграммы.
- Программа должна сохранять результаты анализа в текстовом файле с указанием исходного текста (или пути к исходному файлу) и даты анализа.

А.4.2. Требования к надежности:

- использование лицензированного программного обеспечения;
- проверка программы на наличие вирусов;
- организация бесперебойного питания.

А.4.3. Требования к составу и параметрам технических средств

Для нормального функционирования данной информационной системы необходим компьютер, клавиатура, мышь и следующие технические средства:

- 2-ух ядерный процессор Intel I3 или лучше;
- объем свободной оперативной памяти ~60 Мб;
- объем необходимой памяти на жестком диске ~20Мб;
- стандартный VGA-монитор или совместимый;
- стандартная клавиатура;

- манипулятор «мышь».

А.4.4. Требования к информационной и программной совместимости

Для полноценного функционирования данной системы необходимо наличие операционной системы Microsoft Windows 7 или выше. Также необходимо наличие Расширяемого пакета Microsoft Visual C++ 2003-2021 и пакета .NET версии 5.0. Язык интерфейса – русский.

А.4.5. Требования к маркировке и упаковке

Программа должна поставляться в виде проекта, исполняемого (exe) файла, установщика и документации.

А.4.6. Требования к транспортировке и хранению

Программа распространяется в электронном виде. Требования к транспортировке и хранению не предъявляются.

А.4.7. Специальные требования

Теоретическая часть включает подробное описание работы с приложением, включая UML-диаграммы, разработанные на этапе проектирования программной системы. Практическая часть включает разработку и реализацию программных модулей программного продукта с использованием среды программирования.

А.5. ТРЕБОВАНИЯ К ПРОГРАММНОЙ ДОКУМЕНТАЦИИ

А.5.1. Предварительный состав программной документации:

- «Техническое задание»;
- «Описание программы»;
- «Руководство пользователя»;
- разрабатываемые программные модули должны быть самодокументированы, т.е. тексты программ должны содержать все

- необходимые комментарии;
- разрабатываемое программное обеспечение должно включать справочную систему.

А.5.2. Перечень материалов пояснительной записки

Введение
1. Теоретические основы разработки
1.1. Описание предметной области
1.2. Анализ методов решения
1.3. Обзор средств программирования
1.4. Описание выбранного языка программирования
2. Практическая часть
2.1. Постановка задачи
2.2. Описание схем
2.3. Текст программы
2.4. Описание программы
2.5. Руководство оператора
2.6. Программа и методика испытаний
2.7. Протокол испытаний
Заключение
Список использованных источников
Приложения

Рисунок А.1 – Перечень материалов пояснительной записки

А.6. ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ

Технико-экономические показатели не рассчитываются.

А.7. СТАДИИ И ЭТАПЫ РАЗРАБОТКИ

Таблица А.1 – Стадии и этапы разработки

Содержание стадии	Содержание этапа	Срок 2022 г.		Форма отчетности
		начало	конец	
Техническое задание	Составление технического задания	16.04	23.04	Техническое задание
Эскизный проект	Разработка спецификаций	24.04	28.04	Спецификации программного обеспечения
Рабочий проект	Проектирование программы	22.04	30.04	Схема работы системы и спецификации компонентов
	Составление программы	25.04	15.05	Программная документация
	Приёмо-сдаточные испытания	16.05	22.05	Протокол испытаний (п. 2.7 пояснительной записки)
Приёмка	Защита курсового проекта	21.05	28.05	Оценка за курсовой проект

А.8. ПОРЯДОК КОНТРОЛЯ И ПРИЕМКИ

А.8.1. Порядок контроля

Контроль выполнения должен осуществляться руководителем курсового проекта (преподавателем) в соответствие с п.7.

А.8.2. Порядок приемки

Приемка должна осуществляться с участием руководителя после проведения приемо-сдаточных испытаний. В результате защиты курсового проекта должна быть выставлена оценка за курсовой проект.

ПРИЛОЖЕНИЕ Б
Руководство оператора

Правительство Санкт-Петербурга
Комитет по науке и высшей школе
Санкт-Петербургское государственное бюджетное
профессиональное образовательное учреждение
«Политехнический колледж городского хозяйства»

СОГЛАСОВАНО
Председатель ПЦК
09.02.07 Информационные
системы и
программирование

_____ Левит Л.В.
_____._____. 2022

ПРОГРАММА ДЛЯ ПОСИМВОЛЬНОГО АНАЛИЗА ТЕКСТА
Руководство оператора
Листов 5

ПРОВЕРИЛ
Преподаватель

_____ Андреев В.А.
07.05.2022

ВЫПОЛНИЛ
Студент группы ИП-19-3

_____ Новоселов С.Д.
_____._____. 2022

2022

Б.1 Назначение программы

Б.1.1 Функции программы

Основной функцией является упрощение криптоанализа текста путем подсчета всех символов в тексте и структурировании этой информации.

Б.1.2 Состав функций

Б.1.2.1 Ввод текста через текстовое поле

Программа для частотного анализа текста поддерживает возможность ручного ввода текста используя текстовое поле.

Б.1.2.2 Ввод текста, используя стандартные средства Windows

Приложение позволяет вводить текстовые файлы используя стандартное диалоговое окно Windows.

Б.1.2.3 Получение подробных результатов частотного анализа

После выполнения криптоанализа, приложение покажет три результата: список всех символов и количества их повторений в тексте, круговую и столбчатую диаграмму – графический способ отображения информации о символах.

Б.1.2.4 Составление отчета о проведенном исследовании

Программа обладает функцией автоматического создания отчетов о проведенном анализе, который сохраняется в текстовый файл. Отчет состоит из: даты и времени анализа, краткой информации о тексте, который был использован для анализа и список символов в этом тексте.

Б.2 Условия выполнения программы

Б.2.1 Минимальный состав аппаратных средств

Для нормального функционирования данного программного обеспечения необходим компьютер, обладающий следующими компонентами:

- 2-ух ядерный процессор Intel I3 или лучше;

- объем свободной оперативной памяти ~60 Мб;
- объем необходимой памяти на жестком диске ~20Мб;
- стандартный VGA-монитор или совместимый;
- стандартная клавиатура;
- манипулятор «мышь».

Б.2.2 Минимальный состав программных средств

Для обеспечения функционирования данной системы необходимо наличие операционной системы Microsoft Windows 7 или выше. Также необходимо наличие Расширяемого пакета Microsoft Visual C++ 2003-2021 и пакета .NET версии 5.0.

Б.2.3 Требования к пользователю

Конечный пользователь программы должен обладать практическими навыками работы с графическим пользовательским интерфейсом операционной системы.

Б.3 Выполнение программы

Б.3.1 Запуск программы

Запустив программное обеспечение с помощью ярлыка, откроется главное окно (рис. Б.1).

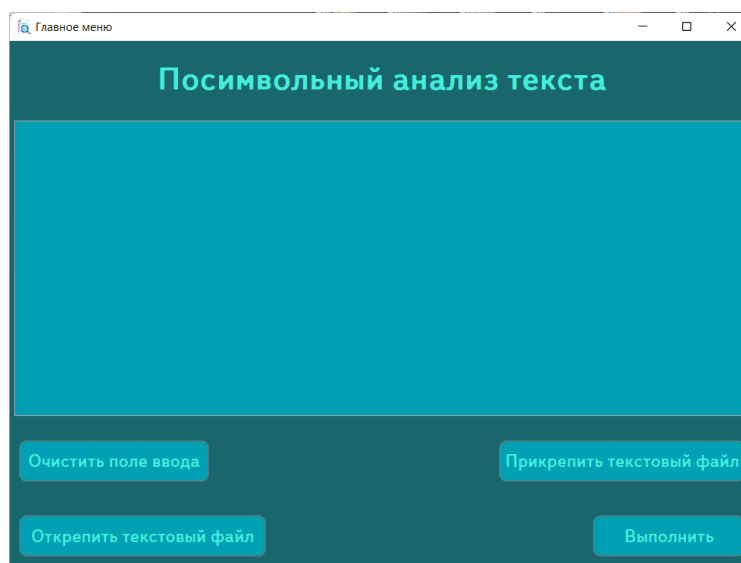


Рисунок Б.1 – Главное окно программы

Б.3.3 Ввод текста для анализа

Главное окно приложения состоит из текстового поля и кнопок управления. Для ввода текста можно воспользоваться текстовым полем, расположенным в центре окна. После набора текста необходимо нажать кнопку «Выполнить» для запуска анализа текста. Альтернативный способ ввода текста доступен по нажатию кнопки «Прикрепить текстовый файл», после чего откроется стандартное окно выбора файла для анализа. После выбора файла, необходимо нажать кнопку «Открыть», тогда выбранный текстовый файл загрузится в программу (рис. Б.2) и можно будет начать анализ.

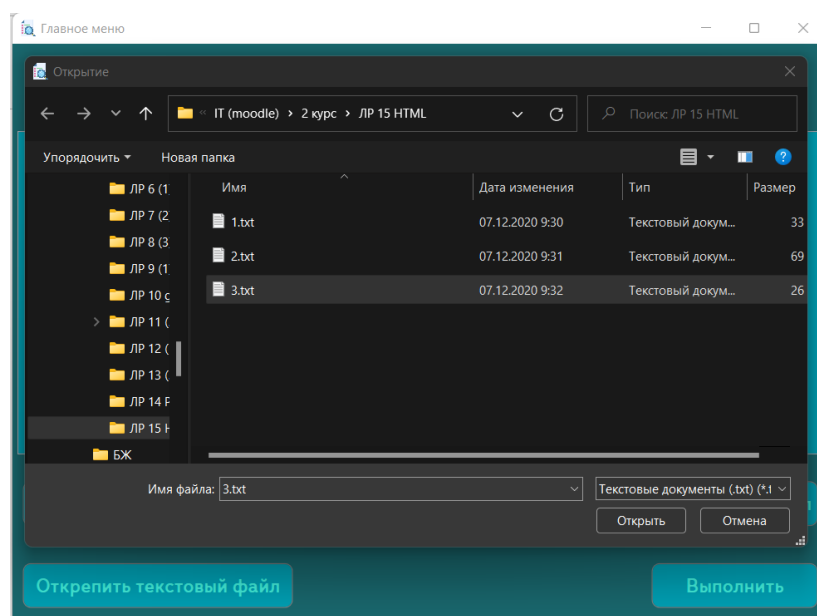


Рисунок Б.2 – Загрузка текстового файла в программу

Б.3.4 Получение отчета об анализе

После нажатия на кнопку «Выполнить» откроется новое окно, которое содержит навигационные кнопки и страницы с результатами. При нажатии навигационных кнопок показываются результаты анализа (рис. Б.3 а, б, в).

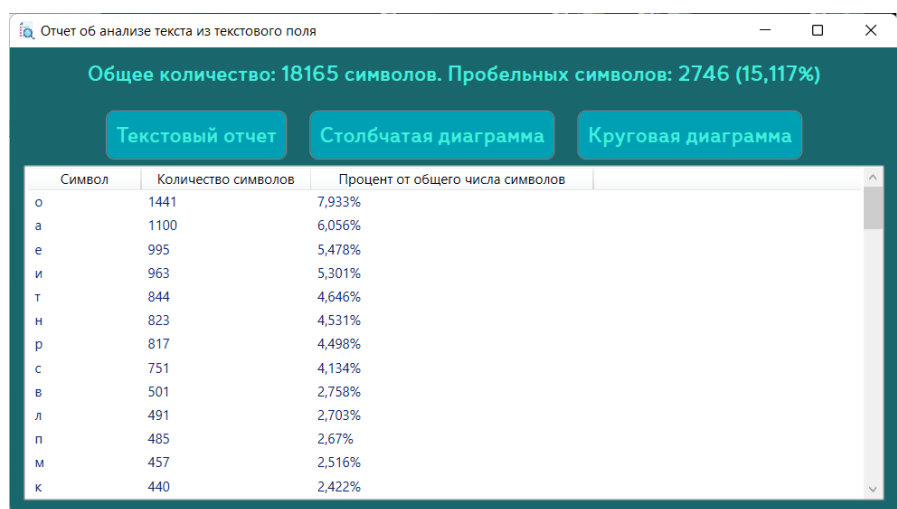


Рисунок Б.3а – Текстового отчета об ошибки

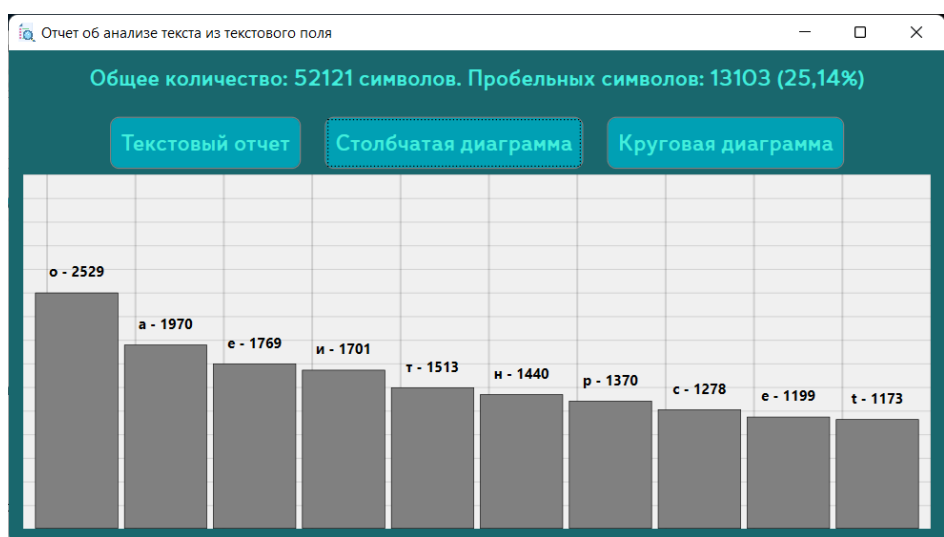


Рисунок Б.3б – Столбчатая диаграмма самых частых символов

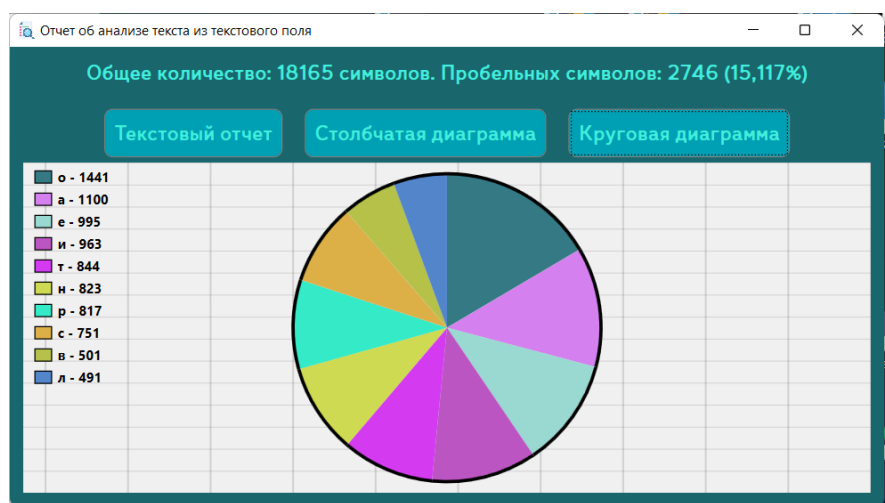
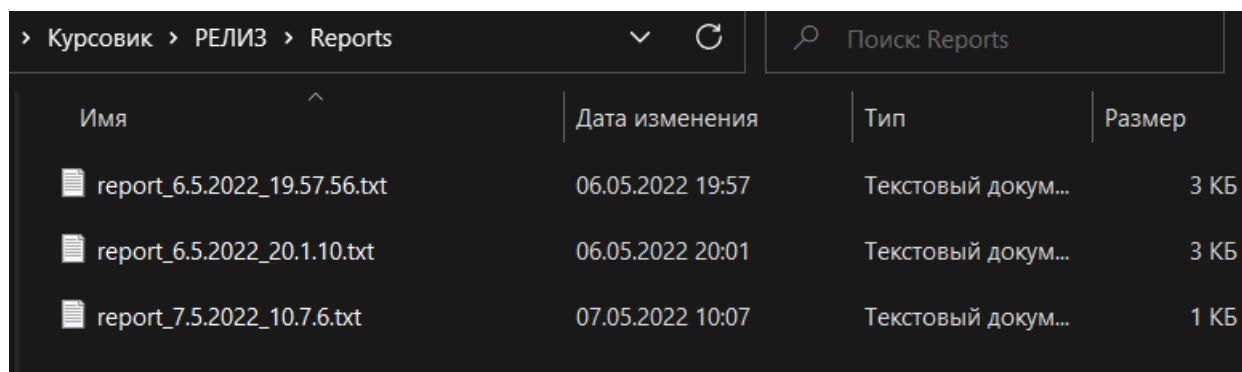


Рисунок Б.3в – Круговая диаграмма самых частых символов

В папке Reports содержатся отчеты о проверке с текстовой информацией о криптоанализе текстов (рис. Б.4)



Имя	Дата изменения	Тип	Размер
report_6.5.2022_19.57.56.txt	06.05.2022 19:57	Текстовый докум...	3 КБ
report_6.5.2022_20.1.10.txt	06.05.2022 20:01	Текстовый докум...	3 КБ
report_7.5.2022_10.7.6.txt	07.05.2022 10:07	Текстовый докум...	1 КБ

Рисунок Б.4 – Папка Reports, которая содержит все отчеты

Б.3.5 Завершение работы программы

Для завершения работы программы используются стандартные средства рабочего окна операционной системы, расположенные в правой части заголовка программы (рис. Б.5).

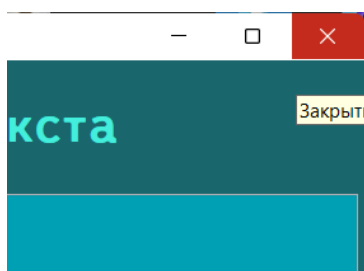


Рисунок Б.5 – Завершение работы программы

ПРИЛОЖЕНИЕ В

Текст программы

Логика окна MainWindow.xaml.cs:

```
using Microsoft.Win32;
using System;
using System.IO;
using System.Windows;

namespace TextAnalysis
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        private string TextForAnalysis = String.Empty;
        private string filePath = String.Empty;

        public MainWindow()
        {
            InitializeComponent();
        }
        private void ClearTextAreaButton_Click(object sender, RoutedEventArgs e)
        {
            if (TextForAnalysis == String.Empty)
            {
                InputTextBox.Text = String.Empty;
            }
        }
        private void CancelFileButton_Click(object sender, RoutedEventArgs e)
        {
            if (TextForAnalysis == String.Empty)
            {
                MessageBox.Show("Вы еще не прикрепили файл",
                                "Предупреждение!",
                                MessageBoxButton.OK,
                                MessageBoxImage.Warning);
                return;
            }
            if (MessageBox.Show("Вы действительно хотите открепить файл?",
                                "Подтверждение",
                                MessageBoxButton.OKCancel,
                                MessageBoxImage.Question) == MessageBoxResult.OK)
            {
                ClearInfo();
            }
        }

        private void ExecuteButton_Click(object sender, RoutedEventArgs e)
        {
            // Присутствует ли текст или файл
            if (InputTextBox.Text != String.Empty)
            {
                if (TextForAnalysis == String.Empty)
                {
                    TextForAnalysis = InputTextBox.Text;
                }
            }
        }
    }
}
```

```

    }
    else
    {
        MessageBox.Show("Вы должны сначала ввести текст или прикрепить файл!",
            "Внимание!",
            MessageBoxButton.OK,
            MessageBoxImage.Warning);
        return;
    }
    ResultWindow resultWindow = new ResultWindow(TextForAnalysis, filePath);
    resultWindow.Show();
    this.Hide();
    ClearInfo();
}

private void ClearInfo()
{
    InputTextBox.Text = String.Empty;
    TextForAnalysis = String.Empty;
    InputTextBox.IsReadOnly = false;
    filePath = String.Empty;
}

private void AttachFileButton_Click(object sender, RoutedEventArgs e)
{
    OpenFileDialog openFileDialog = new OpenFileDialog();
    openFileDialog.FileName = "Document";
    openFileDialog.DefaultExt = ".txt";
    openFileDialog.Multiselect = false;
    openFileDialog.Filter = "Текстовые документы (*.txt)|*.txt";

    if (openFileDialog.ShowDialog() == true)
    {
        using (StreamReader reader = new(openFileDialog.FileName))
        {
            TextForAnalysis = reader.ReadToEnd();
        }
        filePath = openFileDialog.FileName;
        InputTextBox.Text = "Вы прикрепили текстовый файл!\n Путь к файлу:\n" +
openFileDialog.FileName;
        InputTextBox.IsReadOnly = true;
    }
}
}

```

Логика окна **ResultWindow.xaml.cs**:

```

using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Windows;
using TextAnalysis.ResultPages;

namespace TextAnalysis
{
    /// <summary>
    /// Логика взаимодействия для ResultWindow.xaml
    /// </summary>
    public partial class ResultWindow : Window

```



```

{
    public int SymbolsCount;
    public Dictionary<char, int> symbols = new Dictionary<char, int>();
    private string text;
    private string filePath;

    public ResultWindow(string text, string filePath)
    {
        InitializeComponent();
        this.text = text.ToLower();
        this.filePath = filePath;
        if(filePath != String.Empty)
        {
            this.Title = "Отчет об анализе текста из файла: " + filePath;
        }
        else
            this.Title = "Отчет об анализе текста из текстового поля";
    }

    private void ShowTextResultButton_Click(object sender, RoutedEventArgs e)
    {
        TextResult textResult = new TextResult(symbols, SymbolsCount);
        frame.Content = textResult;
    }

    private void ShowBarDiagramButton_Click(object sender, RoutedEventArgs e)
    {
        BarDiagramPage barDiagramPage = new BarDiagramPage(symbols, SymbolsCount);
        frame.Content = barDiagramPage;
    }

    private void ShowCircleDiagramButton_Click(object sender, RoutedEventArgs e)
    {
        CircleDiagramPage circleDiagramPage = new CircleDiagramPage(symbols, SymbolsCount);
        frame.Content = circleDiagramPage;
    }

    private void Window_Loaded(object sender, RoutedEventArgs e)
    {
        int whiteSpaceCount = 0;
        foreach (var symbol in text)
        {
            SymbolsCount++;
            if (symbols.ContainsKey(symbol))
            {
                symbols[symbol]++;
            }
            else
            {
                if (char.IsWhiteSpace(symbol)) whiteSpaceCount++;
                else symbols.Add(symbol, 1);
            }
        }
        symbols = symbols.OrderByDescending(pair => pair.Value).ToDictionary(pair => pair.Key, pair =>
pair.Value);
        AllCharsCountLabel.Content = "Общее количество: "
            + SymbolsCount
            + " символов. Пробельных символов: "
            + whiteSpaceCount + $" ({Math.Round((float)whiteSpaceCount/(float)SymbolsCount*100, 3)}%)";
        CreateReport();
    }
}

```

```

        TextResult textResult = new TextResult(symbols, SymbolsCount);
        frame.Content = textResult;
    }

    private void CreateReport()
    {
        DateTime today = DateTime.Now;
        string reportName =
$"report_{today.Day}.{today.Month}.{today.Year}_{today.Hour}.{today.Minute}.{today.Second}";
        string reportPath = @"Reports\{reportName}.txt";

        using (FileStream writer = File.Create(reportPath))
        {
            string reportText = $"{this.Title}\n";
            reportText += $"{AllCharsCountLabel.Content}\n";
            reportText += "-----\n";
            int counter = 1;
            foreach (var symbol in symbols)
            {
                reportText += $"{counter} {symbol.Key} - {symbol.Value} - {Math.Round((float)symbol.Value /
(float)SymbolsCount * 100, 3)}%\n";
                counter++;
            }

            byte[] reportTextByteArray = new UTF8Encoding(true).GetBytes(reportText);
            writer.Write(reportTextByteArray, 0, reportTextByteArray.Length);
        }
    }

    private void Window_Closed(object sender, EventArgs e)
    {
        Application.Current.MainWindow.Show();
    }
}

```

Логика страницы CircleDiagramPage.xaml.cs:

```

using System.Collections.Generic;
using System.Windows.Controls;
using System.Windows.Input;
using Charts;

namespace TextAnalysis.ResultPages
{
    /// <summary>
    /// Логика взаимодействия для CircleDiagramPage.xaml
    /// </summary>
    public partial class CircleDiagramPage : Page
    {
        Chart chart = new PieChart();
        Dictionary<char, int> symbols = new Dictionary<char, int>();

        public CircleDiagramPage(Dictionary<char, int> symbols, int symbolsCount)
        {
            InitializeComponent();
            this.symbols = symbols;
            //Добавляем диаграмму на грид
            PaintingSurface.Children.Add(chart.ChartBg);
        }
    }
}

```

```

private void PaintingSurface_MouseUp(object sender, MouseButtonEventArgs e)
{
    CreateCircleGraphics();
    PaintingSurface.SizeChanged += PaintingSurface_SizeChanged;
}

private void PaintingSurface_SizeChanged(object sender, SizeChangedEventArgs e)
{
    CreateCircleGraphics();
}

private void CreateCircleGraphics()
{
    // Принудительно обновляем размеры контейнера для диаграммы
    PaintingSurface.UpdateLayout();
    PaintingSurface.ToolTip = null;

    chart.ChartBg.Children.Clear();
    chart.PathList.Clear();

    int counter = 0;
    foreach (var symbol in symbols)
    {
        chart.AddValue(symbol.Value, $"{symbol.Key} - {symbol.Value}");
        counter++;
        if (counter == 10) break;
    }
}
}
}

```

Логика страницы TextResultPage.xaml.cs:

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Windows.Controls;
using System.Windows.Data;

namespace TextAnalysis.ResultPages
{
    /// <summary>
    /// Логика взаимодействия для TextResult.xaml
    /// </summary>
    public partial class TextResult : Page
    {
        public List<SymbolClass> chars = new();

        public TextResult(Dictionary<char, int> symbols, int symbolsCount)
        {
            InitializeComponent();
            float percent = 0;
            foreach (var pair in symbols)
            {
                percent = (float)pair.Value / (float)symbolsCount * 100;
                SymbolClass sym = new(pair.Key, pair.Value, $"{Math.Round(percent,3)}%");
                chars.Add(sym);
            }
        }
    }
}

```

```

        this.TextResultListView.ItemsSource = chars;

        CollectionView view =
        (CollectionView)CollectionViewSource.GetDefaultView(TextResultListView.ItemsSource);
        view.SortDescriptions.Add(new SortDescription("symbolCount", ListSortDirection.Descending));
    }
}

public class SymbolClass
{
    public char symbol { get; set; }
    public int symbolCount { get; set; }
    public string symbolPercent { get; set; }

    public SymbolClass(char symbol, int symbolCount, string symbolPercent)
    {
        this.symbol = symbol;
        this.symbolCount = symbolCount;
        this.symbolPercent = symbolPercent;
    }
}
}

```

Абстрактный класс Chart.cs:

```

using System.Collections.Generic;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Media;

namespace Charts
{
    internal abstract class Chart
    {
        // Максимальная высота графика (значение 100%)
        private readonly double factor = 0.66666667;
        public readonly double PaddingChart = 10;
        public double Width;
        public double Height;
        public Canvas ChartBg = new();
        public Dictionary<string, double> BarsDict = new Dictionary<string, double>();
        public List<StoredValues> PathList = new List<StoredValues>();

        public Chart()
        {
            ChartBg.Margin = new Thickness(0);
            ChartBg.SizeChanged += ChartBg_SizeChanged;
        }

        public void ChartBg_SizeChanged(object sender, SizeChangedEventArgs e)
        {
            Width = e.NewSize.Width - (PaddingChart * 2);
            Height = e.NewSize.Height * factor;

            ChartBg.Background = DrawLines(e.NewSize.Width, Width, PaddingChart);
        }

        public abstract void AddValue(double value, string header);

        // Кисть для заднего фона
    }
}

```

```

private Brush DrawLines(double actualwidth, double widthchart, double padding)
{
    double totalWidth = widthchart / actualwidth;

    int numLines = 10;

    DrawingBrush brush = new()
    {
        TileMode = TileMode.Tile,
        Viewport = new Rect(1, 0, totalWidth / numLines, factor / numLines),
        // Рисуем прямоугольник, формирующий фоновую сетку.
        Drawing = new GeometryDrawing()
        {
            Pen = new(Brushes.Black, 0.05),
            Brush = new SolidColorBrush(Color.FromRgb(240, 240, 240)),
            Geometry = new RectangleGeometry(new Rect(0, 0, 45, 20))
        }
    };

    return brush;
}

public class StoredValues
{
    // Угол сектора диска
    public double Degree;

    // Угловое смещение
    public double Offset;

    public double Value;
    public string Header;
}
}

```

Класс PieChart.cs:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Media;
using System.Windows.Shapes;

namespace Charts
{
    internal class PieChart : Chart
    {
        public override void AddValue(double value, string header)
        {
            PathList = CalculateSectorAngle(value, header);

            ChartBg.Children.Clear();

            // Размещение секторов на канвасе
            for (int i = 0; i < PathList.Count; i++)
            {
                Path p = CreateSector(PathList[i]);
                ChartBg.Children.Add(p);
            }
        }
    }
}

```

```

// Числовые значения секторов диска.
Label label = new()
{
    Content = PathList[i].Header,
    FontWeight = FontWeights.Bold,
};

// Цветовые метки перед числовыми
Rectangle r = new()
{
    Width = 16,
    Height = 12,
    Fill = p.Fill,
    Stroke = Brushes.Black,
    StrokeThickness = 1
};

StackPanel sp = new()
{
    Orientation = Orientation.Horizontal
};
sp.Children.Add(r);
sp.Children.Add(label);
Canvas.SetLeft(sp, 10);
Canvas.SetTop(sp, 20 * i);
ChartBg.Children.Add(sp);
}
}

// Создание готового для вывода сектора
private Path CreateSector(StoredValues storedValues)
{
    Random random = new();

    Path path = new()
    {
        StrokeThickness = 3,
        Stroke = Brushes.Black,
        Fill = new SolidColorBrush(Color.FromArgb(255, (byte)random.Next(50, 256), (byte)random.Next(50, 256), (byte)random.Next(50, 256))),

        Data = new PathGeometry()
        {
            Figures = new PathFigureCollection()
            {
                {
                    SectorGeometry(storedValues.Degree, storedValues.Offset)
                }
            }
        },

        Tag = new StoredValues()
        {
            Degree = storedValues.Degree,
            Offset = storedValues.Offset,
            Value = storedValues.Value
        }
    };

    return path;
}

```

```

}

// Перерасчет переменных для создания сектора
private PathFigure SectorGeometry(double degree, double offset)
{
    double ChartRadius = ChartBg.ActualHeight / 2 - PaddingChart;

    bool isLarge = degree > 180 ? true : false;

    // Если начальная точка совпадет с конечной, сектор не отразится
    if (degree >= 360) degree = 359.999;

    Point centerPoint = new(ChartBg.ActualWidth / 2, ChartBg.ActualHeight / 2);

    Point startPoint = new(centerPoint.X, centerPoint.Y - ChartRadius);

    Point endPoint = startPoint;

    // Поворачиваем на угол смещения стартовую точку.
    RotateTransform rotateStartPoint = new(offset)
    {
        CenterX = centerPoint.X,
        CenterY = centerPoint.Y
    };
    startPoint = rotateStartPoint.Transform(startPoint);

    // Поворачиваем на заданный угол конечную точку
    RotateTransform rotateEndPoint = new(offset + degree)
    {
        CenterX = centerPoint.X,
        CenterY = centerPoint.Y
    };
    endPoint = rotateEndPoint.Transform(endPoint);

    PathFigure sector = new()
    {
        StartPoint = startPoint,
        Segments = new PathSegmentCollection()
        {
            new ArcSegment()
            {
                Point = endPoint,
                Size = new(ChartRadius, ChartRadius),
                SweepDirection = SweepDirection.Clockwise,
                IsLargeArc = isLarge,
                IsStroke = true
            },
            new PolyLineSegment()
            {
                Points = new PointCollection() { startPoint, centerPoint, endPoint },
                IsStroke = false,
            }
        }
    };

    return sector;
}

// Пересчитываем уже добавленные значения и записываем новые

```

```

private List<StoredValues> CalculateSectorAngle(double value, string header)
{
    StoredValues d = new()
    {
        Value = value,
        Header = header
    };

    PathList.Add(d);

    PathList = PathList.OrderByDescending(p => p.Value).ToList();

    double sum = PathList.Select(p => p.Value).Sum();

    // Общий знаменатель (значение, которое приходится на 1 градус)
    double denominator = sum / 360;

    for (int i = 0; i < PathList.Count; i++)
    {
        // Для исключения артефактов при рисовании снижаем точность
        PathList[i].Degree = Math.Round(PathList[i].Value / denominator, 2);

        double offset = 0;
        if (i > 0)
        {
            offset = PathList[i - 1].Degree + PathList[i - 1].Offset;
        }

        PathList[i].Offset = offset;
    }

    return PathList;
}
}

```

Класс BarChart.cs:

```

using System;
using System.Linq;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Media;
using System.Windows.Shapes;

namespace Charts
{
    internal class BarChart : Chart
    {
        // Расстояние между графиками
        private double gap = 5;

        public override void AddValue(double value, string header)
        {
            BarsDict.Add(header, value);

            // Вычисляем новую ширину бара, чтобы график поместился
            double widthBar = (Width - ((BarsDict.Count - 1) * gap)) / BarsDict.Count;

            // Ограничение высоты графика

```



```

double maxValue = BarsDict.Values.Max();
double denominator = maxValue / Height;
ChartBg.Children.Clear();

// Перерисовываем графики
foreach (var pair in BarsDict)
{
    int count = ChartBg.Children.OfType<Rectangle>().Count();

    double heightPoint = pair.Value / denominator;

    double BarX = (count * (widthBar + gap)) + (ChartBg.ActualWidth - Width) / 2;

    // Создание бара
    Rectangle bar = CreateBar(BarX, heightPoint, widthBar, pair.Value);
    ChartBg.Children.Add(bar);
    // Надпись над баром
    Label title = CreateTitle(BarX, bar.Height, widthBar, pair.Key);
    ChartBg.Children.Add(title);
}
}

//Создание столбика в диаграмме
private Rectangle CreateBar(double x, double height, double width, double value)
{
    Random random = new();
    Rectangle bar = new()
    {
        Stroke = Brushes.Black,
        Fill = new SolidColorBrush(Colors.Gray),
        Height = height,
        Width = width,
        StrokeThickness = 0.5,
        Tag = value
    };
    Canvas.SetLeft(bar, x);
    Canvas.SetBottom(bar, 0);
    return bar;
}

//Создание подписи
private Label CreateTitle(double x, double y, double width, string header)
{
    Label title = new()
    {
        Content = header,
        HorizontalContentAlignment = HorizontalAlignment.Center,
        Width = width,
        Padding = new Thickness(0, 0, 0, 10),
        FontWeight = FontWeights.Bold
    };

    Canvas.SetLeft(title, x);
    Canvas.SetBottom(title, y);
    return title;
}
}
}

```