Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«Уральский федеральный университет имени первого Президента России Б. Н. Ельцина» (УрФУ)
Институт радиоэлектроники и информационных технологий – РТф
Школа профессионального и академического образования.

Учебный научный центр” Информационная безопасность”

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовому проекту
по теме:
«Создание программной библиотеки, которая позволяет извлекать текстовую информацию из файлов формата XLS без использования библиотек Microsoft»
по дисциплине: «Языки и методы программирования»

Руководитель:	Хорьков Д.А., доцент учебно-научного центра «Информационная безопасность»	_____ _	21.12.2024
Исполнители:	Гирфанов Д.М., Гордеев А.Г., Добрых А.А., Костромин В.И., Миншина А.Д., Отставной Н.Ю., Теребенин А.С., Черников С.С., Шакуров Е.А., РИ-321055/56	_____ _	21.12.2024

Екатеринбург
2024

Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«Уральский федеральный университет имени первого Президента России Б. Н. Ельцина» (УрФУ)
Институт радиоэлектроники и информационных технологий – РТф
Школа профессионального и академического образования.
Учебный научный центр «Информационная безопасность»

Задание
на курсовой проект/работу

Студент_____
_____
группа_____
специальность/направление подготовки_____

1.Тема курсового проекта/работы___
_____
_____
_____
_____

2.Содержание проекта/работы, в том числе состав графических работ и расчетов
_____
_____
_____
_____

3.Дополнительные сведения
_____
_____
_____
_____

4. План выполнения курсового проекта/работы

Наименование элементов проектной работы	Сроки	Примечания	Отметка о выполнении

Руководитель _____/И.О. Фамилия/

Министерство образования и науки Российской Федерации
ФГАОУ ВО «УрФУ имени первого Президента России Б.Н.Ельцина»

РЕЦЕНЗИЯ

на курсовую работу (проект)

Студента ____ группы ____
(фамилия имя отчество)

Тема курсовой работы: _____
_____
_____
_____

Модуль/дисциплина ___
_____
_____

1 Соответствие результатов выполнения работы целям и задачам курсового проектирования, результатам обучения по дисциплине/модулю_____
_____
_____
_____

2 Оригинальность и самостоятельность выполнения работы_____
_____
_____
_____
_____

3 Полнота и глубина проработки разделов _____
_____
_____
_____
_____

4 Общая грамотность и качество оформления текстового документа и графических
материалов_____
_____
_____
_____
_____

5 Вопросы и замечания _____

_____

_____

_____

_____

_____

6 Общая оценка работы _____

_____

_____

_____

Сведения о рецензенте:

Ф.И.О. _____

Должность _____

Место работы _____

Уч. звание _____ Уч. степень _____

ПОДПИСЬ _____ **ДАТА_____**

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ 7
1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 8
1.1 Compound File (CF) 8
1.1.1 Заголовок (Header) 8
1.1.2 FAT (File Allocation Table) 10
1.1.3 MiniFAT 10
1.1.4 DIFAT 10
1.1.5 Directory Entry (каталог) 11
1.2 Directory Entry 11
1.3 Поток Workbook 13
1.4 Подпоток Globals Substream 15
1.5 Подпоток Worksheet Substream 17
1.6 Запись BUNDLESHEET 19
1.7 Запись SHAREDSTRINGS (SST) 20
1.8 Запись LABELSST 21
1.9 Теоретическая справка по DLL 23
1.9.1 Принцип работы 23
1.9.2 Основные элементы DLL 24
2. ПРОГРАММНЫЙ ПРОДУКТ 25
2.1 Архитектура 25
2.2 Разработка 25
2.2.1 Изучение спецификации Microsoft по XLS 25
2.2.2 Алгоритм программной реализации 27
2.2.3 Первый прототип программы 28
2.2.4 Чтение потока Workbook 29
2.2.5 Идентификация записей 30
2.2.6 Вывод текстовых данных 31
2.3 Тестирование 32
2.3.1 Unit-тест 32

2.3.2 Пользовательское тестирование 33
SELF - REVIEW 37
ЗАКЛЮЧЕНИЕ 41
СПИСОК МАТЕРИАЛОВ 42

ВВЕДЕНИЕ

Предлагаемая проектная документация разрабатываемого программного обеспечения демонстрирует процесс разработки библиотеки для считывания текстовых данных из файлов формата XLS, включая теоретическую часть и алгоритм программной реализации.

Библиотека для считывания текстовых данных из файлов формата XLS является важным инструментом в современной цифровой среде, так как этот формат, созданный корпорацией Microsoft для программы Excel, продолжает широко использоваться для хранения, анализа и обмена табличными данными в различных отраслях. Из этого следует то, что тема курсового проекта всё ещё актуальна.

Ручная обработка хранимых в файле данных требует много времени и связана с риском ошибок, поэтому инструменты, позволяющие автоматически извлекать и обрабатывать данные из таблиц, становятся всё более востребованными. Формат XLS обладает гибкостью и часто включает сложные элементы, такие как объединённые ячейки, особое форматирование, встроенные графики и формулы. Для работы с такими структурами стандартные методы, например, работа через CSV, оказываются недостаточными, что усиливает необходимость специализированных решений.

Разработанная библиотека для извлечения текстовых данных из файлов формата XLS может найти применение во множестве сфер, охватывая как частные задачи, так и сложные бизнес-процессы. Данная библиотека, несомненно, найдёт применение среди людей, занимающихся извлечением и дальнейшей обработкой данных, а также послужить «каркасом» для доработки или создания приложения/приложений со схожим функционалом.

1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Разработка библиотеки для чтения текстовых данных из файлов формата XLS предполагает создание программы, которая способна обрабатывать двоичные файлы, построенные на основе Compound File Binary Format (CFBF). Такая библиотека должна уметь извлекать и интерпретировать текстовое содержимое ячеек, следуя спецификациям формата XLS. Для реализации программы, способной извлекать текстовые данные из файлов формата XLS, необходимо провести всестороннее изучение спецификации данного формата. Это включает детальное понимание его внутренней структуры, основанной на Compound File Binary Format (CFBF), и особенностей хранения данных в бинарных потоках. Необходимо разобраться в том, как организованы потоки данных, а также каким образом в этих потоках сохраняется информация о ячейках, строках, таблицах и других элементах.

1.1 Compound File (CF)

Compound File или OLE Compound Document — это файловый формат, разработанный Microsoft, который используется для хранения сложных структурированных данных в одном файле. Этот формат представляет собой контейнер, содержащий потоки и подсекторы, организованные по принципу файловой системы, с каталогом и таблицами для навигации внутри контейнера. Compound File в XLS используется для организации данных, таких как текст, числовые значения, формулы и настройки рабочего листа. Содержимое файла организовано в потоки, обеспечивая доступ к данным каждого листа, настройкам формата, общей таблице строк (SST) и другим элементам.

Compound File имеет структуру, схожую с файловой системой, и состоит из следующих ключевых компонентов:

1.1.1 Заголовок (Header)

Заголовок 512 байт (размер стандартного заголовка) содержит ключевую информацию, которая помогает программе определить, является ли файл Compound File, и извлечь параметры, необходимые для дальнейшего чтения.

Структура заголовка:

1) Сигнатура (Signature). Первые 8 байт содержат уникальную сигнатуру (D0 CF 11 E0 A1 B1 1A E1), указывающую, что файл является Compound File. Это первое, что проверяет программа. Если сигнатура совпадает, файл может быть обработан как Compound File, если нет - программа завершает работу или сообщает об ошибке.
2) Идентификатор класса (CLSID). Байт с 8-го по 23-й. В большинстве случаев заполнен нулями и используется для уникальной идентификации файла.
3) Версия файла. Два байта, начиная с 24-го, определяют версию и минимальную версию Compound File, поддерживаемую этим файлом. Например, версия 3 указывает формат, используемый в файловых системах 512 байт на сектор, версия 4 — для 4096 байт на сектор.
4) Размер сектора и подсектора. Байты с 30 по 31: определяют размер стандартного сектора, который чаще всего составляет 512 байт (значение 0x0200) или 4096 байт (значение 0x1000. Байты с 32 по 33: определяют размер мини-сектора, который обычно составляет 64 байта (значение 0x0040).
5) Число секторов FAT (File Allocation Table) указывают общее количество секторов FAT, которые используются для распределения данных в потоке. Эти сектора содержат ссылки на другие секторы, создавая цепочки секторов для организации потоков и хранилищ.
6) Ссылки на дополнительные таблицы.
    - Главный сектор Directory: байты 48-51 указывают на начальный сектор Directory Entry, где хранится иерархия потоков и хранилищ.

- Сектор MiniFAT: байты 60-63 указывают на начальный сектор таблицы MiniFAT, который управляет малыми секторами для хранения мини-потоков.
- DIFAT (Double Indirect FAT): байты 76-511 содержат указатели на таблицы распределения, если данных много и требуется больше таблиц FAT для управления потоками.

1.1.2 FAT (File Allocation Table)

Таблица распределения секторов, которая хранит ссылки на последовательные секторы, объединяющиеся для создания потоков. FAT организует доступ к данным, указывая, в каких секторах находится информация для каждого потока и как связаны эти сектора и функционирует аналогично файловой системе в традиционных операционных системах.

1.1.3 MiniFAT

Используется для управления малыми потоками (меньше 4096 байт). Малые потоки хранятся в отдельной цепочке мини-секторов, и MiniFAT выполняет роль таблицы распределения для них. MiniFAT помогает эффективно использовать пространство для небольших объектов и записей.

1.1.4 DIFAT

DIFAT (Double Indirect File Allocation Table) в структуре Compound File используется для отслеживания расположения FAT-секторов, которые управляют потоками данных. Если число FAT-секторов превышает 109, информация о них хранится в дополнительных секторах DIFAT. В заголовке файла записаны первые 109 ссылок на FAT-секторы. Если их недостаточно, добавляются DIFAT-сектора, которые содержат ссылки на FAT и на следующий DIFAT-сектор, образуя цепочку. Последняя запись каждого сектора указывает на следующий сектор, либо принимает значение END_OF_CHAIN, если цепочка завершается. DIFAT обеспечивает масштабируемость структуры для хранения больших объемов данных.

1.1.5 Directory Entry (каталог)

Содержит структуру дерева для представления иерархии потоков и подсекторов. Каждый поток или объект в файле Compound File имеет запись в Directory Entry, которая содержит метаданные, такие как имя потока, его размер, начальный сектор и тип (например, обычный поток или хранилище). Каталог помогает программе находить нужные потоки, такие как Workbook в XLS-файлах, и управлять их доступом.

Compound File поддерживает универсальность в хранении данных, предоставляя возможность включать текстовый контент, мультимедиа, встроенные объекты и другие элементы. Логика работы формата напоминает стандартные файловые системы, что облегчает разработку и интеграцию приложений. Надежность хранения данных дополнена встроенными таблицами распределения (FAT, MiniFAT, SBAT), которые помогают восстанавливать данные при повреждениях. Масштабируемость формата позволяет эффективно работать как с небольшими, так и с большими файлами, благодаря механизмам Master FAT, обеспечивающим управление большими объемами информации.

1.2 Directory Entry

Directory Entry - это важный компонент формата Compound File, который представляет собой структурированную запись всех потоков и хранилищ, находящихся внутри файла. Он действует как таблица содержания для Compound File, указывая, где начинаются потоки, какова их структура, имена, иерархия и другие метаданные.

В файле XLS Directory Entry используется для описания и управления иерархией объектов, таких как потоки и хранилища. Каждая запись содержит полную информацию об объекте, включая его имя, тип, положение в иерархии, а также метаданные, такие как временные метки и состояние. Иерархическая организация записей, поддерживаемая ссылками на родительские, дочерние и соседние объекты, создает древовидную структуру, схожую с файловыми

системами. Это делает возможным логическое объединение данных, упрощая навигацию и управление.

Структура записи Directory Entry:

1) Имя объекта (64 байта). Первые 64 байта записи содержат имя потока или хранилища в формате Unicode. Это имя позволяет идентифицировать конкретный поток или хранилище, например, Workbook или Globals. Окончание имени помечено нулевым байтом, если его длина меньше 64 байт.
2) Длина имени (2 байта). Указывает длину имени в байтах. Это поле помогает программе определить фактическую длину имени потока или хранилища.
3) Тип объекта (1 байт). Определяет, является ли объект хранилищем (значение 0x01), потоком (значение 0x02), Root Entry (корневым объектом, 0x05), который находится на вершине иерархии Compound File.
4) Цвет узла (1 байт). Определяет цвет узла (красный или черный), используемый для балансировки дерева в структуре Directory Entry. Это поле помогает поддерживать оптимальную структуру хранения и доступа к записям.
5) Левая, правая и дочерняя ссылки (по 4 байта каждая). Эти ссылки организуют записи в иерархии, помогая построить древовидную структуру. Используются для доступа к соседним объектам (левому, правому) и дочерним узлам (детям), если объект является хранилищем.
6) Идентификатор CLSID (16 байт). Уникальный идентификатор (CLSID) для хранилищ, который может использоваться в специальных сценариях для их идентификации.
7) Состояние объекта (4 байта). Содержит различные флаги, характеризующие состояние объекта. Чаще всего используется для

хранения информации о размере потока или других специальных параметрах.

8) Дата создания и дата изменения (по 8 байт). Поля содержат временные метки создания и последнего изменения объекта. Эти значения сохраняются в виде 64-битных чисел, представляющих количество 100-наносекундных интервалов с 1601 года (в формате Windows FILETIME).
9) Начальный сектор (4 байта). Указывает на первый сектор потока, где хранятся данные. Этот указатель позволяет программе перейти к нужному потоку, чтобы прочитать или изменить его содержимое.
10) Размер потока (4 байта). Определяет длину потока в байтах. Если поток меньше, чем минимальный размер сектора (обычно 4096 байт), он хранится в MiniFAT (специальной области для маленьких потоков).

Directory Entry не только хранит структурные данные, но и обеспечивает связь с фактическим содержимым потока через указатель на начальный сектор и размер потока. Для малых потоков используется MiniFAT, оптимизирующий хранение мелких объектов, что значительно экономит дисковое пространство.

В дополнение к этому, поддержка временных меток и уникальных идентификаторов CLSID расширяет функциональные возможности Compound File, позволяя использовать его в разнообразных приложениях, от хранения документов до сложных мультимедийных или инженерных данных. Таким образом, Directory Entry представляет собой фундаментальную часть Compound File, обеспечивая высокую производительность, гибкость и надежность работы с данными.

1.3 Поток Workbook

Поток Workbook является одним из ключевых компонентов файлов в формате Microsoft Excel Binary File Format (XLS). Этот поток хранит основные данные рабочей книги, включая информацию о листах, форматах, формулах и

других аспектах, необходимых для работы с электронной таблицей. Поток расположен внутри контейнера Compound File и представлен в виде последовательности структурированных записей (BIFF - Binary Interchange File Format), каждая из которых отвечает за определенные элементы файла.

Поток Workbook состоит из ряда записей BIFF, организованных в последовательности, которая описывает содержание и настройки документа. BIFF-формат представляет собой двоичные данные, разбитые на записи, каждая из которых начинается с заголовка. Заголовок содержит идентификатор записи (2 байта) и длину данных (2 байта), за которыми следует фактическое содержимое.

Структура потока Workbook:

1) Общие свойства книги. Поток содержит записи, определяющие настройки рабочей книги, такие как версия Excel, параметры отображения и защита. Например, запись BOF (Beginning of File) указывает начало BIFF-потока, а EOF (End of File) завершает поток.
2) Данные листов. Каждый лист рабочей книги представлен отдельным набором записей BIFF. Эти записи включают информацию о содержимом ячеек (значения, формулы, текст), настройках отображения (ширина столбцов, форматирование), а также о типах данных, используемых в ячейках.
3) Глобальные настройки. Поток Workbook содержит глобальные параметры, которые применяются ко всей рабочей книге. Например, FONT определяет параметры шрифтов, FORMAT — числовые форматы, а XF (Extended Format) хранит информацию о стиле ячеек.
4) Общая таблица строк (Shared String Table, SST). Для оптимизации хранения текстовых данных используется запись SST, в которой текстовые строки сохраняются в виде уникальных значений. Это позволяет экономить место, так как вместо хранения одинаковых

строк несколько раз они записываются только один раз в SST и ссылаются из ячеек.

5) Настройки безопасности. BIFF-записи, такие как FILEPASS, могут использоваться для шифрования или защиты рабочей книги паролем.
6) Графические объекты и дополнительные элементы. Поток также может включать данные о диаграммах, изображениях, комментариях и других встроенных объектах.

Поток Workbook является центральной частью формата XLS, так как он содержит всю информацию, необходимую для отображения и обработки данных в электронной таблице. Его структура обеспечивает компактное и эффективное хранение данных, а использование BIFF-записей позволяет легко добавлять новые функции и совместимость с предыдущими версиями.

Эта организация позволяет Excel обрабатывать большие объемы данных и поддерживать множество функций, таких как формулы, фильтрация, сортировка, визуализация данных и работа с макросами, делая поток Workbook важным элементом системы Excel.

1.4 Подпоток Globals Substream

Globals Substream — это один из важнейших подпотоков, входящих в поток Workbook в формате Microsoft Excel Binary File Format (XLS). Этот подпоток содержит глобальные настройки рабочей книги, которые применяются ко всем листам и данным в файле. Globals Substream расположен в начале потока Workbook и состоит из последовательности BIFF-записей (Binary Interchange File Format), описывающих глобальные свойства книги, включая стили, шрифты, числовые форматы и другие параметры.

Структура Globals Substream:

1) BOF (Beginning of File). Начальная запись подпотока, которая указывает на начало Globals Substream и определяет его тип. Эта

запись помогает программе распознать, что она начала считывать глобальные настройки.

2) BUNDLESHEET. Эта запись содержит информацию о каждом листе в книге, включая его название и смещение, по которому можно найти данные листа в потоке Workbook. Также определяет состояние листа (видимый, скрытый или очень скрытый) и его тип (обычный лист, диаграмма или макрос).
3) SST (Shared String Table). Хранит уникальные строковые значения, которые используются на всех листах рабочей книги. Вместо хранения каждой строки в ячейках, программа использует ссылки на SST, что уменьшает объем данных. Состоит из количества строк и массива строк, каждая из которых имеет индекс для быстрого доступа.
4) STYLE и FONT. STYLE содержит предопределенные стили, такие как шрифты, цвета, форматирование и выравнивание, которые применяются к ячейкам. FONT описывает свойства шрифта, включая начертание, цвет и размер, и может быть использован в стилях. FORMAT определяет пользовательские форматы, применяемые к числовым, текстовым и датам в ячейках. Позволяет задать единый формат для разных ячеек, что упрощает их настройку и редактирование.
5) XF (Extended Format). Содержит расширенные параметры форматирования ячеек, такие как границы, цвет заливки и выравнивание. Применяется к ячейкам для унификации их внешнего вида.
6) EOF (End of File). Эта запись завершает Globals Substream и указывает, что в нем больше нет данных.

Globals Substream играет критически важную роль в организации данных и управлении настройками рабочей книги. Его структура позволяет Excel быстро получать доступ к глобальным данным, таким как стили и

шрифты, без необходимости повторного их определения для каждого листа или ячейки.

1.5 Подпоток Worksheet Substream

Подпоток Worksheet Substream представляет собой данные, которые непосредственно относятся к содержимому одного из листов рабочей книги, а именно к отдельным ячейкам, их значениям, форматам и метаданным. Каждый лист рабочей книги в формате XLS имеет свой собственный подпоток Worksheet Substream, который хранит все специфические данные, связанные с этим листом. Подпоток Worksheet Substream, размещается в потоке Workbook, где он является частью общей структуры данных файла. Подпоток Worksheet Substream состоит из последовательности BIFF-записей, каждая из которых выполняет свою функцию.

Структура Worksheet Substream:

1) BOF (Beginning of File). Начальная запись подпотока, указывающая на начало данных для конкретного листа. Включает информацию, помогающую программе идентифицировать, что начался новый рабочий лист.
2) CELL. Записи типа CELL содержат информацию о каждой ячейке на листе. В этих записях хранятся данные, такие как значение ячейки, формула (если она есть), и другие метаданные. Каждая запись CELL относится к определенной ячейке в заданном диапазоне и может включать данные, такие как строки, числа или ошибки. Также эта запись может хранить информацию о формате ячейки (ссылки на записи в таблице XF).
3) XF (Extended Format). Запись XF описывает форматирование ячеек. Каждая запись XF может содержать информацию о шрифте, границах, цветах, выравнивании и других параметрах отображения данных в ячейке. Эти записи используются для

унификации и применения одинаковых стилей форматирования к нескольким ячейкам.

4) SST (Shared String Table). В случае, если на листе используются повторяющиеся строки текста, то они могут быть записаны в таблицу SST, что позволяет экономить место. В этом случае в ячейках содержатся индексы, указывающие на строки в Shared String Table, а не сами строки. Это важно для оптимизации работы с текстовыми данными.
5) DIMENSIONS. Определяет диапазон используемых строк и столбцов на листе, указывая его верхнюю и нижнюю границы. Помогает программе понять, насколько велик диапазон данных, которые нужно обрабатывать.
6) ROW. Каждая запись ROW в подпотоке Worksheet Substream описывает строку данных на листе. В этой записи указаны номер строки, используемые ячейки и другие параметры, такие как скрытые или фиксированные строки.
7) BLANK, NUMBER, и LABELSST. BLANK представляет пустые ячейки, для которых требуется особое форматирование. NUMBER хранит числовые значения ячеек. LABELSST хранит строковые значения, используя ссылку на индекс в SST (Shared String Table).
8) FORMULA. Определяет ячейку, содержащую формулу, и включает результат её вычисления. Хранит формулу в скомпилированной форме, а также её текстовое представление.
9) BOOLERR. Используется для хранения логических значений (TRUE или FALSE) или ошибок (например, #DIV/0!). Полезна для проверки корректности значений в ячейках.
10) Merged Cells (MERGEDCELLS). Определяет диапазоны объединённых ячеек на листе. Позволяет программе корректно отображать объединённые ячейки.

11) EOF (End of File). Запись EOF указывает на конец подпотока Worksheet Substream. Она завершает данные листа и помогает системе определить, что информация о текущем листе завершена.

Worksheet Substream является неотъемлемой частью хранения и представления данных в файле XLS. Он организует все данные листа в структуру, которая позволяет быстро обращаться к ячейкам, вычислять значения формул, и отображать результаты пользователю. Также, благодаря разделению данных на ячейки, строки и столбцы, подпоток Worksheet Substream облегчает работу с большими объемами информации и помогает в обеспечении совместимости между различными версиями Excel.

Таким образом, подпоток Worksheet Substream представляет собой важный элемент структуры файлов XLS, обеспечивающий хранение всех данных, относящихся к отдельному рабочему листу, и их эффективное использование в процессе работы с файлом.

1.6 Запись BUNDLESHEET

BUNDLESHEET — это специализированная запись в формате Microsoft Excel Binary File Format (XLS), которая используется в потоке Workbook для хранения информации о листах рабочей книги. Она играет важную роль в идентификации листов, их свойствах и метаданных, которые необходимы для корректной работы и отображения листа в приложении Excel.

Запись BUNDLESHEET находится в Globals Substream и служит для предоставления важной информации о каждом листе в рабочей книге, такой как его название, состояние видимости, тип листа и другие ключевые параметры. Эта запись позволяет системе Excel эффективно управлять и организовывать данные листа, обеспечивая правильное отображение в интерфейсе программы и работу с данными.

Позиция листа указывает его порядковый номер в рабочей книге, что важно для правильного отображения и порядка листов. Кроме того, в записи содержится смещение, которое указывает на место начала данных этого листа

в потоке Workbook. Вся эта информация необходима для корректной работы Excel, чтобы программа могла правильно отображать и переключаться между листами, а также точно вычислять и показывать их содержимое. Записи BUNDLESHEET помогают эффективно организовать и структурировать листы рабочей книги, упрощая процесс загрузки и управления данными.

Содержимое записи BUNDLESHEET:

1) Тип записи 0x0085.
2) Положение листа. Смещение или номер сектора, где начинается соответствующий лист в потоке Workbook.
3) Состояние листа. Может указывать, видим ли лист, скрыт или очень скрыт.
4) Тип листа. Может быть обычным листом, диаграммой, модулем макросов и т.д.
5) Название листа. Название, отображаемое пользователю (например, "Лист1").

В целом, запись BUNDLESHEET обеспечивает правильную организацию метаданных для каждого листа рабочей книги и поддерживает эффективную работу Excel с файлами XLS.

1.7 Запись SHAREDSTRINGS (SST)

Запись SHAREDSTRINGS (или SST, что расшифровывается как Shared String Table) используется для хранения строковых данных, которые встречаются в рабочей книге, но повторяются на нескольких листах или в разных ячейках. Вместо того, чтобы дублировать строки в каждой ячейке, Excel использует таблицу Shared String Table, где каждая строка хранится только один раз, а в ячейках сохраняются ссылки на эти строки. Это помогает уменьшить размер файла и повысить его эффективность.

Запись SHAREDSTRINGS содержит таблицу уникальных строк, используемых во всей рабочей книге. Вместо того, чтобы хранить каждое текстовое значение отдельно в каждой ячейке, Excel хранит все уникальные

строки в одном месте - в SST, а в ячейках сохраняются индексы этих строк. Это значительное сокращение объема данных, особенно если одна и та же строка встречается много раз на разных листах.

Таблица SST сохраняет строки в виде массива, где каждая строка индексируется. Когда ячейка ссылается на строку, она сохраняет не саму строку, а ее индекс в таблице SST. Таким образом, повторяющиеся строки в рабочей книге занимают память только один раз, что значительно экономит пространство, особенно в больших документах с большим количеством текстовых данных.

Содержимое записи SHAREDSTRINGS (SST):

1) Тип записи 0x00FC.
2) Количество строк. Общее количество строк, включая дублирующиеся.
3) Уникальные строки. Количество уникальных строк.
4) Массив строк. Каждая строка хранится в формате Unicode и может включать дополнительные свойства, такие как форматирование.

SHAREDSTRINGS играет ключевую роль в оптимизации хранения строковых данных в формате XLS, эффективно уменьшая размер файла и улучшая производительность. Использование таблицы уникальных строк позволяет избежать дублирования данных, что особенно важно для больших документов с множеством повторяющихся текстов. С помощью индексации строк Excel значительно экономит память, сохраняя в ячейках только ссылки на строки, а не их полные копии. Это обеспечивает более компактное хранение данных и ускоряет обработку рабочих книг.

1.8 Запись LABELSST

Запись LABELSST в формате Microsoft Excel Binary File Format (XLS) используется для хранения текстовых данных в ячейках, которые ссылаются на таблицу общих строк SHAREDSTRINGS (SST). Это важная часть структуры, которая помогает оптимизировать хранение строковых данных в

рабочей книге, минимизируя повторение строк в ячейках и позволяя экономить место в файле.

Когда ячейка на листе содержит текст, который ранее был сохранен в SST вместо того, чтобы повторно хранить текст в каждой ячейке, Excel сохраняет только индекс этой строки в SST. Запись LABELSST указывает на строку, которая находится в SST, и хранит ссылку на эту строку с помощью её индекса.

Содержимое записи LABELSST:

1) Тип записи 0x00FD. Это поле идентифицирует запись как LABELSST. Тип записи помогает программе понять, что данные относятся к строке в SST, а не к обычному текстовому значению.
2) Размер записи. Значение указывает на размер записи в байтах, начиная с поля "SST Index". Это значение важно для корректной обработки записи.
3) Индекс строки в SST. Значение: индекс строки в таблице SHAREDSTRINGS (SST). Этот индекс указывает на конкретную строку в SST, которая должна быть извлечена и отображена в ячейке. Программа использует этот индекс для поиска строки в SST и отображения её в ячейке.
4) Флаги. Это поле может содержать флаги, которые указывают на дополнительные параметры записи, такие как форматирование или состояние строки.

Запись LABELSST используется в случае, когда текст, который должен быть отображен в ячейке, уже был сохранен в SST. Вместо того чтобы хранить сам текст в ячейке, в LABELSST сохраняется индекс строки в таблице SST. Это позволяет эффективно управлять памятью, так как одна и та же строка может быть использована на разных листах или в разных ячейках, при этом физически она хранится только один раз в SST.

1.9 Теоретическая справка по DLL

DLL (Dynamic Link Library) — это динамическая библиотека, представляющая собой набор функций и ресурсов, которые могут быть вызваны и использованы другими программами. DLL является важным компонентом операционных систем Windows, а также программного обеспечения, работающего в этой среде.

DLL-файлы обычно имеют расширение .dll, но также могут встречаться другие варианты, такие как .ocx (ActiveX) или .drv (драйверы устройств). Эти библиотеки позволяют разделять код между несколькими программами, обеспечивая модульность, удобство обслуживания и экономию ресурсов.

1.9.1 Принцип работы

Компиляция и использование:

Основная концепция DLL заключается в том, что они компилируются как независимые модули. Это позволяет разрабатывать библиотеки, которые можно повторно использовать в различных приложениях. В процессе компиляции создаётся два файла: .dll -сам файл библиотеки, содержащий скомпилированный код, .lib - файл импорта (опционально), который используется для статической линковки приложения с библиотекой.

Приложение использует экспортируемые функции библиотеки, подключая её через линковку. При статической линковке файл .lib добавляется к проекту на этапе компиляции. Программа при запуске автоматически подключает связанный DLL-файл. Этот метод проще в использовании, но требует, чтобы DLL была доступна в системе при запуске. При динамической линовке LoadLibrary загружает DLL во время выполнения программы, а GetProcAddress получает адрес экспортируемой функции из загруженной библиотеки.

Загрузка DLL:

При использовании статической линковки программа зависит от наличия библиотеки уже в момент запуска. Если файл DLL отсутствует или не

соответствует ожидаемой версии, операционная система выдаст ошибку, и приложение не запустится.

Динамическая загрузка предоставляет большую гибкость. Программа может проверить наличие библиотеки и подгрузить её только при необходимости. Это особенно полезно, если библиотека используется в редких случаях или имеет несколько версий.

Входная точка DLL:

Каждая DLL содержит функцию DllMain, которая вызывается системой в определённые моменты жизненного цикла библиотеки. Это стандартная точка входа для инициализации и завершения работы библиотеки.

1.9.2 Основные элементы DLL

Экспортируемые функции — это функции библиотеки, которые могут быть вызваны из других программ или DLL. Для их создания и экспорта требуется использовать специальные инструкции компилятора. Функции помечаются «__declspec(dllexport)» спецификатором, чтобы они были доступны за пределами библиотеки.

При работе с DLL важную роль играют таблицы импорта (Import Address Table, IAT) и экспорта (Export Address Table, EAT). Они обеспечивают связь между вызывающей программой и функциями библиотеки. Таблица импорта (Import Address Table, IAT) содержит адреса функций, импортируемых из внешней DLL, которые программа использует в своей работе. Таблица экспорта (Export Address Table, EAT) содержит список всех функций, которые DLL предоставляет для внешнего использования.

2. ПРОГРАММНЫЙ ПРОДУКТ

2.1 Архитектура

Библиотека для работы с XLS файлами имеет модульную архитектуру, что позволяет четко разделять ответственность между различными компонентами. Основной алгоритм работы включает в себя взаимодействие нескольких модулей для чтения и анализа содержимого XLS файла, обработка FAT и DIFAT, парсинг директории, обработка рабочей книги и организованный вывод данных.

Компонент чтения данных отвечает за открытие XLS файла и извлечение его содержимого. Он использует низкоуровневые операции для работы с файлами, включая чтение заголовка и получение информации о структуре документа. Этот компонент также обрабатывает различные секции файла, такие как DIFAT и FAT, для корректной интерпретации данных.

Компонент обработки данных выполняет анализ информации, полученной из XLS файла. Он включает функции для извлечения информации о листах, строках и ячейках, а также для работы с таблицей строк (SST), содержащей текстовые данные. Этот модуль обеспечивает правильное преобразование данных в удобный для пользователя формат.

В компоненте организованного вывода данных реализуется обработка и вывод данных из ячеек рабочей книги Excel. Он формирует структурированный вывод, который может включать названия листов и содержимое ячеек.

2.2 Разработка

2.2.1 Изучение спецификации Microsoft по XLS

Разработка библиотеки для считывания текстовых данных из файлов формата XLS началась с детального изучения спецификации формата и структуры этих файлов. Основным этапом было анализирование того, как организованы данные внутри файла и какие потоки содержат информацию о рабочих книгах, листах и их содержимом. Эти данные структурированы в

различных частях документа, и понимание того, как они взаимосвязаны, позволяло корректно извлекать информацию.

Одним из важных аспектов изучения было определение ключевых потоков и подпотоков, которые хранят основные данные файла. Например, поток «Workbook» содержит информацию о рабочей книге, такую как ее метаинформация и структура. Помимо этого, поток «Globals Substream» хранит глобальные настройки, относящиеся ко всей рабочей книге, например, параметры отображения или региональные настройки. Еще одним важным элементом является поток «Worksheet Substream», в котором содержатся данные о листах рабочей книги, такие как ссылки на ячейки и форматирование.

Кроме того, важными элементами спецификации являются запись «BUNDLESHEET», которая позволяет идентифицировать каждый лист рабочей книги и запись «SHAREDSTRINGS», которая хранит строковые данные, что позволяет избежать дублирования строк в документе и эффективно организовывать их хранение и извлечение.

Также была обнаружена запись «LABELSST», которая хранит текстовые данные непосредственно в ячейках рабочего листа. Важным моментом является то, что текстовые данные в ячейках часто хранятся в виде ссылок на строки в массиве SST (Shared Strings Table). Массив строк SST позволяет эффективно обрабатывать текстовые данные, извлекая их по индексам. Это упрощает как процесс чтения данных, так и обработку большого объема текстовой информации, ускоряя работу с файлами и уменьшая их размер.

Таким образом, изучение спецификации и структуры файлов XLS позволило выстроить эффективный подход к извлечению текстовых данных из ячеек. Понимание роли потока «Workbook», подпотоков «Globals Substream» и «Worksheet Substream», записей «BUNDLESHEET», «SHAREDSTRINGS» и «LABELSST», а также использование массива строк SST стало основой для разработки библиотеки, способной корректно обрабатывать и извлекать текстовые данные из Excel-файлов.

2.2.2 Алгоритм программной реализации

После изучения структуры XLS, потока рабочей книги, подпотоков рабочего листа и ключевых записей, необходимых для обработки рабочих листов, была поставлена задача создания алгоритма, обеспечивающего корректное извлечение текстового значения ячеек.

Программа для обработки файла в формате Compound File начинает работу с анализа первых 512 байт файла, чтобы проверить, соответствует ли он формату Compound File. Эти первые 512 байт содержат сигнатуру и параметры для дальнейшей обработки. Если сигнатура файла подтверждается как валидная, программа извлекает метаданные из заголовка и продолжает обработку, переходя к следующему этапу — чтению сектора Directory Entry (DE).

В секторах Directory Entry хранятся записи о потоках и хранилищах, которые используются в файле. Программа анализирует эти записи и ищет поток Workbook, который является основным потоком для работы с данными книги Excel. После того как найден поток Workbook, программа переходит к его началу.

На первом шаге анализа потока Workbook программа проверяет наличие заголовка BOF (Beginning of File), который подтверждает, что данный поток является валидным для дальнейшей обработки. Далее, программа продолжает чтение потока и извлекает данные Globals Substream, в котором находятся ключевые записи, такие как BUNDLESHEET (содержащие информацию о рабочих листах) и SHAREDSTRINGS (SST — таблица строк).

Если обнаружена таблица SST, программа выделяет память для массива строк в соответствии с количеством строк, указанных в таблице SST, и загружает в этот массив все строки из таблицы. Каждая строка из SST помещается в соответствующий элемент массива, индекс которого совпадает с индексом строки в таблице SST. Например, строка с индексом 0 в таблице SST будет находиться в массиве под индексом 0, строка с индексом 1 — под индексом 1 и так далее.

После обработки таблицы SST программа возвращается к записям BUNDLESHEET, которые содержат информацию о смещениях и типах всех рабочих листов в книге. Используя эти данные, программа переходит к анализу потоков Worksheet Substream для каждого рабочего листа.

При обработке данных ячеек на каждом рабочем листе программа анализирует записи, такие как LABELSST и LABEL. В случае, если встречается запись LABELSST, программа использует индекс в таблице SST для получения текстового значения из массива строк. Если же встречается запись LABEL, программа извлекает текст напрямую из самой записи.

После того как данные одного рабочего листа обработаны, программа переходит к следующему листу, используя смещения, указанные в записях BUNDLESHEET. Этот процесс продолжается до тех пор, пока не будут обработаны все рабочие листы в книге.

2.2.3 Первый прототип программы

Первый прототип программы для работы с файлами Excel формата XLS был сосредоточен исключительно на базовом считывании файла без извлечения текстовых данных или других значимых элементов. Основная цель прототипа заключалась в том, чтобы открыть файл, проверить его доступность и корректно интерпретировать его базовую структуру на уровне двоичных данных.

На начальном этапе программа открывала Excel-файл и проверяла его на соответствие формату XLS. Это включало чтение заголовка Compound File (CF), а именно первые 512 байт файла. Этот заголовок содержит ключевые метаданные, определяющие структуру файла и его принадлежность к формату Compound File Binary (CFB), используемому Excel. Если файл не соответствовал ожидаемому формату, программа выводила сообщение об ошибке и завершала работу.

Основными задачами первого прототипа программы стали:

1) Чтение содержимого файла в буфер. Размер буфера определяется на основе размера файла.

2) Извлечение заголовка Compound File, размер которого составляет 512 байт.
3) Сравнение так называемой "магической подписи" файла — это первые 8 байт заголовка, которые имеют строго определённое значение для формата Compound File: D0 CF 11 E0 A1 B1 1A E1. Если это значение не совпадало, программа считала файл невалидным и завершала выполнение.
4) Извлечение количества секторов FAT, которые используются для навигации по данным файла, и размер сектора, который задаётся в виде степени числа 2.

Заключительным этапом работы первого прототипа было подтверждение корректности структуры файла на основании извлечённых данных. После проверки "магической подписи" и извлечения параметров, таких как размер секторов и количество FAT-секторов, программа выводила информацию о валидности файла Compound File.

2.2.4 Чтение потока Workbook

Следующим этапом стало создание функций для считывания потока рабочей книги (Workbook), которое начиналось после успешного анализа заголовка файла и определения параметров, таких как размер сектора и структура FAT (File Allocation Table), обеспечивающих доступ к данным внутри Compound File.

Для чтения потока Workbook сначала находился соответствующий объект в каталоге (Directory Entry). Это осуществлялось следующим образом:

1) Цепочка секторов каталога Directory Entry извлекалась с использованием FAT. Эти сектора содержали метаданные обо всех потоках в файле.
2) Каждый сектор каталога сканировался на наличие записи с именем "Workbook". Когда такая запись находилась, извлекалась информация о её начальном секторе и размере потока.

После этого формировалась цепочка секторов, принадлежащих потоку Workbook, используя таблицу FAT. Эта цепочка представляла собой последовательность указателей на сектора, где хранились данные потока.

Когда цепочка секторов была извлечена, осуществлялось считывание данных из каждого сектора, принадлежащего потоку, и объединение их в непрерывный буфер. Этот буфер представлял собой полный содержимый поток Workbook в виде двоичных данных.

2.2.5 Идентификация записей

После успешного чтения потока Workbook, стояла задача определение ключевых записей, которые необходимы для считывание текстовых данных из каждого рабочего листа.

Данные потока Workbook организованы в виде записей, каждая из которых начинается с заголовка и включает содержимое. Заголовок записи занимает первые 4 байта и состоит из двух полей: тип записи (type), который занимает 2 байта и является идентификатором записи, и размер содержимого (size), также занимающий 2 байта, который указывает, сколько байт будет содержать запись. Содержимое записи идет непосредственно после заголовка и имеет длину, соответствующую значению поля size. Необходимо проходить по буферу, считывая заголовки записей и их содержимое. Тип записи (type) определяет, как именно нужно обработать содержимое записи.

Определение основных записей:

1) BoundSheet8 (тип 0x0085): запись, содержащая метаданные о листах книги, такие как имя листа и его положение в потоке. Для BoundSheet8 извлекалась информация о листах (позиция в потоке и имя), которая сохранялась для последующей работы.
2) SHAREDSTRINGS (тип 0x00FC): запись, представляющая таблицу строк (SST), которая содержит уникальные строки, используемые в ячейках листов. Для SHAREDSTRINGS создавалась структура SST, в которую заносились уникальные строки, извлечённые из потока.

3) LABELSST (тип 0x00FD): запись, которая используется для того, чтобы ячейки ссылались на строки из таблицы SST с помощью индекса. LABELSST идентифицируется по типу записи 253. Когда такая запись найдена, извлекается индекс строки из таблицы SST (поле isst).
4) LABEL (тип 0x0204): запись, хранящая текстовые значения ячеек. LABELSST идентифицируется по типу записи 6.

После определения записей была добавлена функция, которая проходит по всем рабочим листам и извлекает записи, содержащие ячейки с текстовыми значениями

2.2.6 Вывод текстовых данных

После обработки потока Workbook и извлечения необходимых данных начинается процесс вывода текстовых данных ячеек, который опирается на использование таблицы общих строк (SST). Эта структура хранит все текстовые строки, используемые в рабочей книге Excel, и предоставляет индексы для каждой строки, что позволяет сэкономить место за счет повторного использования строк в разных ячейках.

Для работы с текстом из SST программа сначала извлекает данные о строках. Эти данные хранятся в виде байтов, где каждый символ представлен одним или несколькими байтами в зависимости от кодировки, например UTF-8 или UTF-16. Для корректного отображения текста в консоли байты преобразуются в строку формата Unicode. Этот процесс включает анализ байтов строки, их интерпретацию в зависимости от кодировки и формирование итоговой строки. Если строка использует дополнительные байты для символов, это также учитывается.

После преобразования строка в формате Unicode собирается в объект типа wstring, который поддерживает широкий набор символов. Для вывода текста используется функция, которая принимает данные строки, находит их в таблице SST по индексу, извлекает и преобразует их в Unicode. Затем строка

выводится на экран с использованием стандартного механизма работы с текстом, например через wcout.

При обработке текстовых данных из ячейки программа обращается к её структуре (например, LabelSst), где хранится индекс строки в SST. С помощью этого индекса программа извлекает соответствующую строку, обрабатывает её, преобразует в Unicode и выводит в консоль. В результате каждая ячейка отображается с её текстовым содержимым.

2.3 Тестирование

2.3.1 Unit-тест

Данный процесс являлся неотъемлемым этапом процесса разработки. Он помог проверить корректность работы каждого модуля программного продукта на всех этапах его создания. Для достижения наилучшего результата была использована библиотека doctest, значительно упростившая процесс тестирования при разных входных данных.

Цели тестирования:

1) Проверка работоспособности модулей.
2) Контроль за корректностью работы каждой функции.
3) Выявление ошибок с целью последующего их устранения.

Большинство функций, непосредственно участвующих в открытии, считывании, проверке сигнатур и прочих процессах над файлом было протестировано описанным выше методом, что позволило выявить и в последующем устранить ошибки, возникшие на том или ином этапе разработки.

Примеры тестирования:

- Проверка сигнатур заголовка:

  Входные данные – пути к файлам xls. Для проверки корректности тестов на вход также подаётся файл с расширением .txt

Ожидаемый результат: каждый файл с расширением .xls будет успешно считан, попытки сверить сигнатуры COMPOUND FILE с сигнатурами файла .txt не увенчаются успехом.

Результат после проведения тестов: как и ожидалось, файлы XLS успешно прошли проверку сигнатур, а файл TXT нет.

- Считывание цепочки FAT

  Входные данные - пути к файлам xls.

  Ожидаемый результат: у каждого файла XLS будет успешно считана цепочка FAT

  Результат после проведения тестов: как и ожидалось, у каждого файла .XLS считалась цепочка FAT, попытка считать цепочку у файла .txt не увенчалась успехом.

- Ранее описанные тесты, но без .txt файла

  Ожидаемый результат: у каждого файла XLS будет успешно считана цепочка FAT

  Результат после проведения тестов: как и ожидалось, у каждого файла .XLS считалась цепочка FAT. Тесты завершились успешно.

- Проверка считывания WorkBook

  Входные данные те же.

  Ожидаемый результат: WorkBook каждой книги успешно считается

  Результат после проведения тестов: как и ожидалось, у каждого файла .XLS считался WorkBook. Тесты завершились успешно.

2.3.2 Пользовательское тестирование

В процессе реализации программного продукта был также использован метод так называемых «test-case», представляющих собой набор пошаговых инструкций, необходимых для проверки работоспособности программы в текущем состоянии и при внесении в неё изменений. Важно отметить, что для корректного срабатывания тестов необходимо придерживаться единой

структуры кода программного продукта и не отступать от неё в будущем. В таком случае, данный метод, представляющий собой набор из пошагово выполняемых тестов, у каждого из которых есть ожидаемый результат, позволит провести проверку работоспособности каждой из функций без каких-либо затруднений.

Цели:

1) Проверить корректность открытия файла.
2) Осуществить проверку сигнатуры считанного заголовка.
3) Проверить корректность считывания каждого из секторов.
4) Проверить, что выводимый программой результат читабелен и нет «слёта» кодировки.
5) Проверить корректность считывания каждой из цепочек.

В ходе создания программного продукта неоднократно проводилось тестирование разных его версий с соответствующими изменениями. Данный подход позволил выявить возникающие на каждом из этапов ошибки и устранить их.

В силу того, что test-cases охватывали основные аспекты программного продукта, возникающие ошибки оперативно обнаруживались и устранялись. Также стоит отметить, что за счёт наличия данных тестов, взаимодействие между разработчиками и тестировщиками не было затруднено, что позволило эффективно достигать поставленных целей.

Для проверки корректной работы разработанной программной библиотеки был проведён ряд тестов с различными типами данных и структурами файлов. Тестирование охватило наиболее распространённые случаи использования программы, включая обработку файлов с различным уровнем сложности.

Примеры тестирования:

- Простая таблица

  Входные данные: XLS-файл, содержащий одну таблицу без форматирования, включающую текстовые и числовые данные.

Результат: Программа успешно извлекла текстовую информацию из таблицы, сохранив порядок строк и столбцов. Кодировка текста была корректной.

- Файл с рисунками и диаграммами

  Входные данные: XLS-файл, содержащий таблицу с числовыми данными, а также вставленными изображениями и диаграммами.

  Результат: Программа корректно извлекла текстовую информацию из таблицы, проигнорировав изображения и диаграммы. Работа программы не прерывалась.

- Файл с несколькими листами

  Входные данные: XLS-файл, содержащий книгу с несколькими листами, каждый из которых включает текстовые и числовые данные в разных форматах.

  Результат: Программа успешно извлекла данные с каждого листа, сохранив порядок и указав имя листа перед текстом.

- Сложно оформленный отчет

  Входные данные: XLS-файл, представляющий собой отчет с разделением на страницы, сложным форматированием текста (разные шрифты, размеры, цвета), а также вставками изображений и диаграмм.

  Результат: Программа извлекла текстовую информацию, проигнорировав форматирование, изображения и диаграммы. Все данные извлечены в читабельном виде.

- Файл с нарушенной структурой

  Входные данные: XLS-файл, содержащий таблицу с неполными данными (пустые строки, пропуски значений, нарушение структуры данных).

  Результат: Программа корректно извлекла доступную текстовую информацию, пропустив пустые значения без сбоев.

- Файл большого объема

  Входные данные: XLS-файл с таблицей, содержащей несколько тысяч строк и сотни столбцов.

  Результат: Программа успешно обработала файл, извлекла текстовые данные в полном объеме. Производительность программы осталась стабильной.

SELF - REVIEW

Черников Святослав

В рамках курсового проекта я выполнял роль тимлида, что включало в себя координацию работы участников команды и контроль за процессом выполнения задач.

Кроме того, я принимал активное участие в подготовке чистового варианта документации проекта.

Также, в процессе работы, я изучил принципы написания unit-тестов и разработал несколько тестов для проверки правильности работы программного продукта.

Помимо этого, я изучил спецификацию файлов формата XLS и помогал с подготовкой к защите проекта.

Отставной Никита

В рамках курсового проекта я исполнял роль скрам-мастера. Мои основные обязанности заключались в организации работы команды, проведении скрам-мероприятий, таких как стендапы, планирование спринтов, ретроспективы и обзоры спринтов, а также в защите команды от внешних помех.

Кроме того, я следил за качеством написания документации проекта, подготовил для каждого участника листы с личными выполненными задачами и вел активную подготовку к защите нашего проекта.

Также я изучил руководство по Scrum и спецификацию формата XLS.

Гордеев Андрей

Моя роль заключалась в разработке программного продукта для курсового проекта.

В начале работы я собрал версии Excel 2003, 2007 и 2010.

Затем я изучил спецификацию формата XLS, чтобы понять его структуру и особенности.

После этого был разработан первый прототип программы для открытия файлов формата XLS. В дальнейшем программа была доработана, чтобы извлекать и выводить текстовую информацию из ячеек файлов. На финальном этапе программа была оформлена в виде библиотеки.

Шакуров Егор

В нашем курсовом проекте я выполнял роль технического писателя.

Сначала, совместно с Вадимом, я изучил правила составления технической документации и создал первоначальную структуру документа в Word.

Затем я освоил язык разметки Markdown и систему документирования Doxygen.

На основе изучения спецификации формата XLS я составил отчет, в котором выделил ключевые моменты и особенности этого формата.

Далее я подготовил введение, черновые версии теоретической части документации, касающейся формата XLS, а также черновой вариант раздела, описывающего продукт.

В конце я объединил все части документации курсового проекта в единый файл и оформил его в чистовом виде.

Миншина Аделина и Добрых Арина

В рамках курсового проекта мы, Миншина Аделина и Добрых Арина, отвечали за тестирование пользовательского функционала библиотеки, разработанной для извлечения текстовой информации из файлов формата XLS.

На первом этапе мы изучили различия между файлами, созданными в разных версиях Excel — 2003, 2007 и 2010 годов.

Затем мы ознакомились с методами тестирования, в том числе с функциональным тестированием.

Следующим шагом было детальное изучение спецификаций формата XLS.

После этого мы сосредоточились на разработке кода для сравнения sheridstrings в файлах формата XLS и XLSX.

Затем мы создали тестовую базу, которая включала различные типы файлов XLS.

Когда появился первый прототипы программы, мы провели тестирование, используя файлы из нашей базы данных для проверки корректности работы программы. То же самое мы сделали с последующими прототипами, тестируя их на основе новых и уже существующих файлов из базы данных.

Гирфанов Денис и Теребенин Артем

В рамках нашего курсового проекта мы отвечали за аналитику и архитектуру.

На первом этапе мы изучили, что такое программная библиотека, а также основные принципы и правил ее создания. Мы изучили ключевые аспекты разработки библиотек, такие как их структура, модульность и способы интеграции в проекты. Параллельно с этим мы ознакомились со спецификацией формата XLS и оказали помощь в написании введения для документации.

Затем, совместно с Вадимом, мы изучили основные принципы и подходы к архитектуре кода, которые играют ключевую роль в создании качественного и поддерживаемого программного обеспечения.

Наконец, после написания программы, извлекающей содержимое ячеек файлов формата XLS, мы вместе с Вадимом разработали архитектуру программного продукта. Это позволило создать код, который не только выполняет свою основную функцию, но и обладает гибкостью, легкостью в расширении и поддержке в будущем.

Костромин Вадим

В нашем проекте я изначально занимал роль технического писателя.

На первом этапе, совместно с Егором, я изучил правила составления технической документации и разработал первоначальную структуру документа в Word.

После изучения спецификации формата XLS я подготовил отчет, в котором отметил основные особенности и ключевые аспекты этого формата.

Затем я перешел к роли аналитика и архитектора.

Совместно с Артемом и Денисом мы изучили основные принципы и подходы к архитектуре кода, которые лежат в основе разработки качественного и легко поддерживаемого программного обеспечения.

Наконец, после написания программы, извлекающей содержимое ячеек файлов формата XLS, мы совместно разработали архитектуру программного продукта.

ЗАКЛЮЧЕНИЕ

В ходе выполненной курсовой работы была разработана библиотека для чтения текстовых данных из файлов формата XLS, обеспечивающая поддержку структурированного извлечения информации и совместимость с устаревшими системами. В процессе разработки была изучена спецификация формата XLS, включая его ключевые элементы: Compound File Binary Format (CFBF), потоки Workbook и подпотоки, такие как Globals Substream и Worksheet Substream. Это позволило создать эффективный алгоритм для анализа и обработки данных, а также разработать прототип, демонстрирующий базовую функциональность.

Проведённые этапы тестирования, включая модульное и пользовательское тестирование, обеспечили высокую надёжность программы и подтвердили её корректность при обработке данных в различных сценариях. Использование современных подходов, таких как работа с таблицей общих строк (SST) и унификация процессов чтения и анализа данных, позволило оптимизировать библиотеку для работы с файлами большого объёма и сложной структуры.

Данная библиотека имеет широкий спектр применений, от автоматизации рутинных операций в бухгалтерии и аналитике до интеграции в системы образовательных учреждений и маркетинга. Она упрощает обработку данных, минимизирует риск ошибок и повышает производительность, что делает её универсальным инструментом для современных задач обработки табличной информации.

Разработанный продукт обладает потенциалом для дальнейшего расширения функционала, включая поддержку дополнительных форматов, интеграцию с популярными платформами и улучшение пользовательского интерфейса для разработчиков.

СПИСОК МАТЕРИАЛОВ

1. https://okulord.github.io/programming_lib_project/documentation/html/index.html
2. https://github.com/Pokulord/programming_lib_project/tree/main
3. https://pokulord.github.io/programming_lib_project/BlockShemas/diagram.png
4. https://ru.yougile.com/board/57tfkdeszjpr