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

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

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой ЭВМ
_____ Д.В. Топольский
«___»_____ 2023 г.

Разработка программы для поиска заимствований по файлам электронных таблиц

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУРГУ-090301.2023.025 ПЗ ВКР

Руководитель работы,
к.пед.н., доцент каф. ЭВМ
_____ М.А. Алтухова
«___» _____2023 г.

Автор работы,
студент группы КЭ-405
_____Е.Е. Иванов
«___» _____2023 г.

Нормоконтролёр,
ст. преп. каф. ЭВМ
_____С.В. Сяськов
«___» _____2023 г.

Челябинск-2023

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

УТВЕРЖДАЮ
Заведующий кафедрой ЭВМ
_____Д.В. Топольский
«___» _____2023 г.

ЗАДАНИЕ

на выпускную квалификационную работу бакалавра

студенту группы КЭ-405
Иванову Егору Евгеньевичу,
обучающемуся по направлению
09.03.01 «Информатика и вычислительная техника»

1. **Тема работы: «**Разработка программы для поиска заимствований по файлам электронных таблиц» утверждена приказом по университету от

25 апреля 2023 г. № 753-13/12

2. **Срок сдачи студентом законченной работы:** 01 июня 2023 г.

3. **Исходные данные к работе:** задан формат табличного документа с фиксированным количеством содержащихся в нём информационных полей (ячеек). Предоставлен образец правильного выполнения задания, с которым будет производиться проверка других работ.

Входные данные: файлы электронных таблиц с организованными определённым образом данными.

Программа должна соответствовать следующим функциональным требованиям:

1) сопоставление проверяемой работы с образцом и формирование вывода о степени соответствия форматов и содержимого определённых ячеек;

2) показ информации обо всех обнаруженных несоответствиях;

3) возможность ручной проверки ячеек, информация в которых должна отличаться в зависимости от исполнителя работы (содержит данные, которые будут уникальны для каждого исполнителя);

4) сохранение статистики проверенных работ (степень соответствия, обнаруженные ошибки);

5) выполнение анализа проверенных работ на наличие маркеров, указывающих на возможное заимствование;

6) наличие возможности внесения незначительных изменений в исходный табличный документ, например, добавление или изменение содержимого информационной ячейки.

4. **Перечень подлежащих разработке вопросов:**

1) аналитический обзор аналогов разрабатываемой программы и имеющихся методов решения;

2) определение требований;

3) проектирование функциональной модели программы;

4) реализация и тестирование программы.

5. **Дата выдачи задания:** 1 декабря 2022 г.

Руководитель работы _____/ *М.А. Алтухова /*

Студент _____/ *Е.Е. Иванов /*

КАЛЕНДАРНЫЙ ПЛАН

Этап	Срок сдачи	Подпись руководителя
Аналитический обзор аналогов разрабатываемой программы и имеющихся методов решения	27.02.2023	
Определение требований	1.04.2023	
Проектирование функциональной модели программы	15.04.2023	
Реализация и тестирование программы	01.05.2023	
Компоновка текста работы и сдача на нормоконтроль	11.05.2023	
Подготовка презентации и доклада	30.05.2023	

Руководитель работы _____/ *М.А. Алтухова* /

Студент _____/ *Е.Е. Иванов* /

АННОТАЦИЯ

Е.Е. Иванов. Разработка программы для поиска заимствований по файлам электронных таблиц. — Челябинск: ФГАОУ ВО «ЮУрГУ (НИУ)», ВШ ЭКН; 2023, 37с., 24 ил., библиогр. список — 9 наим., 1 прил.

В рамках выпускной квалификационной работы разрабатывается программа для проверки студенческих работ, выполненных в программе MS Excel формата .xlsx. Программа сравнивает работу студента с заданным шаблоном (образцом) и выявляет возможные ошибки, которые были допущены студентом в ходе выполнения работы. Далее, загружается следующая студенческая работа. Она проходит те же этапы проверки на наличие ошибок. Затем, идёт работа с ошибками. Программа сравнивает ошибки, и по характеру этих ошибок программа делает некоторый вывод о наличии заимствования одной работы студента у другой, или об его отсутствии.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В настоящее время с развитием возможностей интернета обостряется проблема плагиата. Эта проблема сказалась и на образовании. Студенты и школьники зачастую выбирают наиболее простой способ выполнения некоторых работ, а именно заимствовать у товарищей или найти готовые решения в Интернете. По итогу стали изобретаться специальные сервисы, которые проверяют ту или иную загруженную работу на наличие заимствований с огромной базой других готовых работ. Но и этого решения оказалось недостаточно: помимо текстовых работ выполняются работы в файлах электронных таблиц и иных форматах. Доступные бесплатные сервисы не имеют функционала проверки файлов, а платные версии и подписки стоят достаточно много, и позволить такое себе может не каждый проверяющий.

Актуальным направлением решения данной проблемы может послужить разработка программы, которая проверяет содержимое файлов на наличие заимствований из других файлов такого же формата.

Целью представленной выпускной квалификационной работы является разработка программы для персонального компьютера, которая будет проверять файлы электронных таблиц на наличие различных заимствований и делать определённые выводы о наличии этих заимствований.

Для достижения поставленной цели, необходимо решить следующие поставленные задачи:

1. Рассмотреть существующие сервисы для проверки файлов на наличие заимствований.

2. Рассмотреть современные методы решения проблемы с проверкой на плагиат.

3. Разработать функциональную модель будущей программы.

4. Реализовать программу и проверить её работоспособность в различных ситуациях.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Обзор аналогов

К сожалению, в настоящее время нет широкоизвестных программ или веб-сервисов по поиску заимствований, которые специализировалась бы на плагиате в файлах электронных таблиц. Однако, существует несколько инструментов, которые могут помочь в этом процессе. К таким инструментам относят:

1) Unicheck;

2) Plagiarism Checker X;

3) DupliChecker.

Unicheck - это сервис, который используется для проверки оригинальности текстов и документов. Он может работать с различными форматами файлов, включая электронные таблицы. Unicheck использует алгоритмы машинного обучения для сравнения загруженного файла с базой данных других файлов и определения наличия заимствований [1].

Plagiarism Checker X - это программа, которая также может использоваться для поиска заимствований в электронных таблицах. Она работает аналогично Unicheck, сравнивая загруженный файл с базой данных других файлов и определяя наличие заимствований. Plagiarism Checker X также может помочь учителям и преподавателям проверять работы на соответствие требованиям и оценивать качество написания [2].

DupliChecker - это онлайн-инструмент, который может использоваться для поиска заимствований в электронных таблицах. Он работает аналогично Unicheck и Plagiarism Checker X, сравнивая загруженный файл с базой данных других файлов и определяя наличие заимствований. DupliChecker также может помочь владельцам сайтов и блогов защитить свои материалы от копирования [3].

1.2 Обзор имеющихся методов решения

Современные антиплагиат сервисы используют такие методы для поиска заимствований, как методы машинного обучения. Существует 6 таких методов:

1) векторное представление текста;

2) методы машинного обучения на основе графов;

3) методы машинного обучения на основе нейронных сетей;

4) алгоритмы кластеризации и классификации;

5) методы машинного обучения на основе деревьев решений;

6) методы машинного обучения на основе статистических моделей.

Наиболее используемыми и подходящими для решения моих задач являются первые два метода, поэтому про них стоит рассказать подробнее.

Метод векторного представления заключается в том, что каждому слову(объекту) в документе присваивается числовое значение (вектор), которое отражает его семантическое значение и контекст использования. Это может быть выполнено с помощью методов, таких как TF-IDF или Word2Vec.

Word2Vec – это алгоритм машинного обучения, используемый для создания векторных представлений слов на основе их контекста в текстовых корпусах. Алгоритм был разработан в 2013 году и стал одним из самых популярных методов для работы с естественным языком.

Алгоритм Word2Vec использует нейронную сеть, которая обучается на большом корпусе текстовых данных. Она принимает на вход последовательности слов и строит векторные представления каждого слова на основе его контекста. Векторы слов, полученные с помощью Word2Vec, имеют свойство близости: слова, которые часто появляются в одном контексте, имеют близкие векторные представления [4].

Существует два метода обучения Word2Vec: Continuous Bag-of-Words (CBOW) и Skip-Gram. В методе CBOW модель пытается предсказать центральное слово на основе контекста, а в методе Skip-Gram - наоборот, пытается предсказать контекст на основе центрального слова. Оба метода имеют свои преимущества и недостатки, и выбор конкретного метода зависит от задачи, которую необходимо решить.

После обучения Word2Vec можно использовать полученные векторные представления слов для различных задач, таких как классификация текстов, анализ тональности и многие другие. Векторы слов могут быть использованы для поиска синонимов, антонимов и связанных слов, что делает Word2Vec очень полезным инструментом для работы с естественным языком.

После преобразования векторы слов объединяются в векторное представление всего текста. При сравнении двух документов алгоритмы машинного обучения сравнивают их векторные представления, определяют степень сходства и выдают результаты проверки на наличие плагиата. Затем используется алгоритм сравнения векторов, такой как косинусное сходство, чтобы определить степень сходства между двумя векторами. Чем более похожи векторы, тем более вероятно, что два документа являются плагиатом.

Методы машинного обучения на основе графов для выявления плагиата основаны на анализе структуры текста. Сначала каждый текст преобразуется в граф, где каждое слово представлено узлом, а связи между узлами отражают связи между словами в тексте. Затем используется алгоритм анализа графов, такой как PageRank или HITS, чтобы определить степень важности каждого узла в графе. Далее сравниваются графы двух текстов, используя алгоритмы сравнения графов, такие как Graph Edit Distance или Subgraph Isomorphism. Чем более похожи графы, тем более вероятно, что два текста являются плагиатом. Для выявления плагиата используются специализированные программы, которые сравнивают графы и выдают результаты сравнения в виде процентного соотношения двух

документов. Если процент сходства превышает заданный порог, то это может указывать на наличие плагиата [5].

В целом, хотя инструменты для поиска заимствований в электронных таблицах не так широко распространены, как для текстовых документов, все же существует несколько программ и сервисов, которые могут помочь в этом процессе.

2 ОПРЕДЕЛЕНИЕ ТРЕБОВАНИЙ

2.1 Функциональные требования

Программа должна соответствовать следующим требованиям:

1) сопоставление проверяемой работы с образцом и формирование вывода о степени соответствия форматов и содержимого определённых ячеек;

2) показ информации обо всех обнаруженных несоответствиях;

3) возможность ручной проверки ячеек, информация в которых должна отмечаться в зависимости от исполнителя работы (содержит данные, которые будут уникальны для каждого исполнителя);

4) сохранение статистики проверенных работ (степень соответствия, обнаруженные ошибки);

5) наличие функции внесения незначительных изменений в исходный табличный формат;

6) выполнение анализа проверенных работ на наличие маркеров, указывающих на возможное заимствование.

Все вышеописанные требования можно представить в виде диаграммы вариантов использования. Данная диаграмма представлена на рисунке 1.

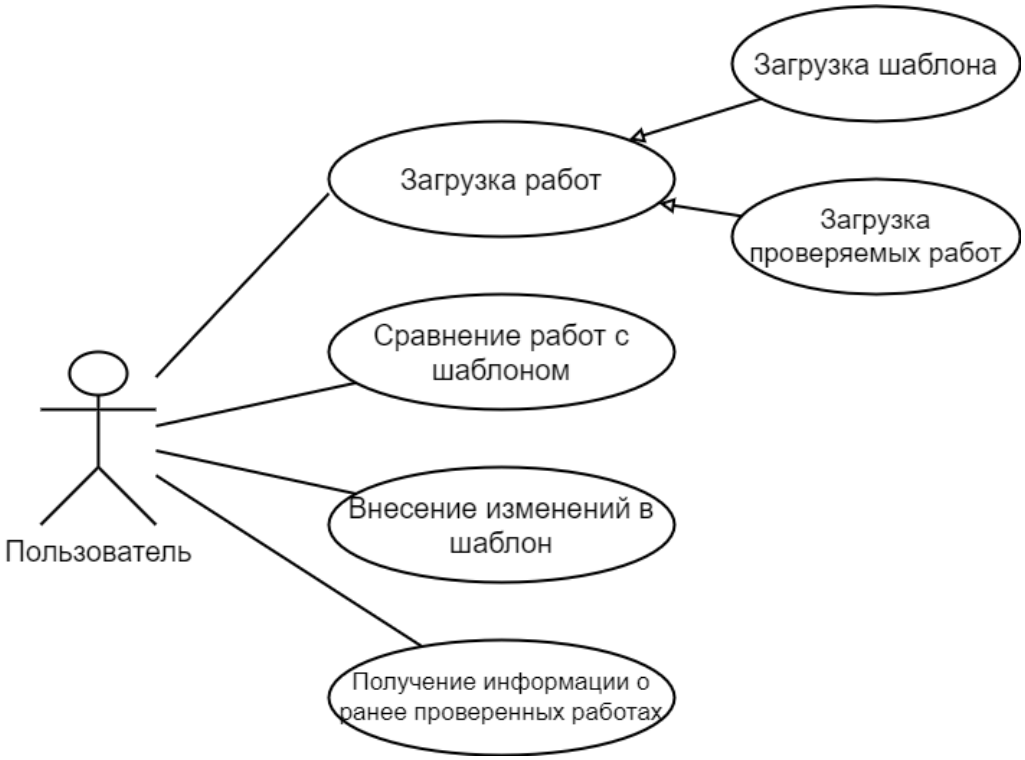


Рисунок 1 – UML диаграмма вариантов использования

Пользователь может:

а) загружать работы;

б) выполнять проверку загруженных работ;

в) вносить незначительные изменения в исходный табличный формат;

г) получать статистику раннее проверенных работ.

Пользователь может загружать как шаблон, так и студенческие работы. Для загрузки необходимо указать полный путь к самой работе.

Внесения изменений в шаблон проводятся вручную пользователем. Для сохранения изменений шаблона при запуске программы, необходимо заново указать пусть к файлу.

2.2 Нефункциональные требования

Нефункциональные требования к программе:

1) наличие кнопочного меню для удобной работы с программой;

2) программа должна работать на следующей ОС: Windows;

3) программа не должна требовать и собирать личные данные пользователя.

3 ПРОЕКТИРОВАНИЕ ФУНКЦИОНАЛЛЬНОЙ МОДЕЛИ ПРОГРАММЫ

Для решения задачи была разработана блок-схема алгоритма программы, которая показывает, как будет выполняться проверка работ и работа программы в целом. Схема алгоритма показана на рисунках 2 и 3.

После запуска программы появляется меню с выбором одного из трёх пунктов:

1) загрузить шаблон;
2) загрузить работу студента;
3) выход из программы.

При выборе первого пункта пользователь загружает шаблон работы, с которым будут сравниваться все последующие работы и одну из проверяемых работ. Далее при помощи методов и функций, описанных в главе 4, выполняется сравнение проверяемой работы с шаблоном, и в местах несоответствий устанавливается маркер, к которому можно получить доступ в любой момент времени. Затем пользователю предлагается внести незначительные изменения в шаблон. Если пользователь хочет внести изменения, то он вносит все необходимые изменения и работа проверяется заново. В противном случае, пользователю предлагается выполнить те же операции, но с другой проверяемой работой. После проверки обеих работ происходит сравнение маркеров ошибок. Если маркеры ошибок похожие, то можно предполагать, что ошибки по своему характеру одинаковые, что может говорить о том, что в работе есть заимствования.

При выборе второго пункта пользователю предоставляется возможность только загрузки работы студента. Затем ему будет выведено сообщение о том, что дальнейшая работа программы без шаблона не имеет смысла, и программа завершится.

При выборе третьего пункта программа завершает свою работу и закрывается.

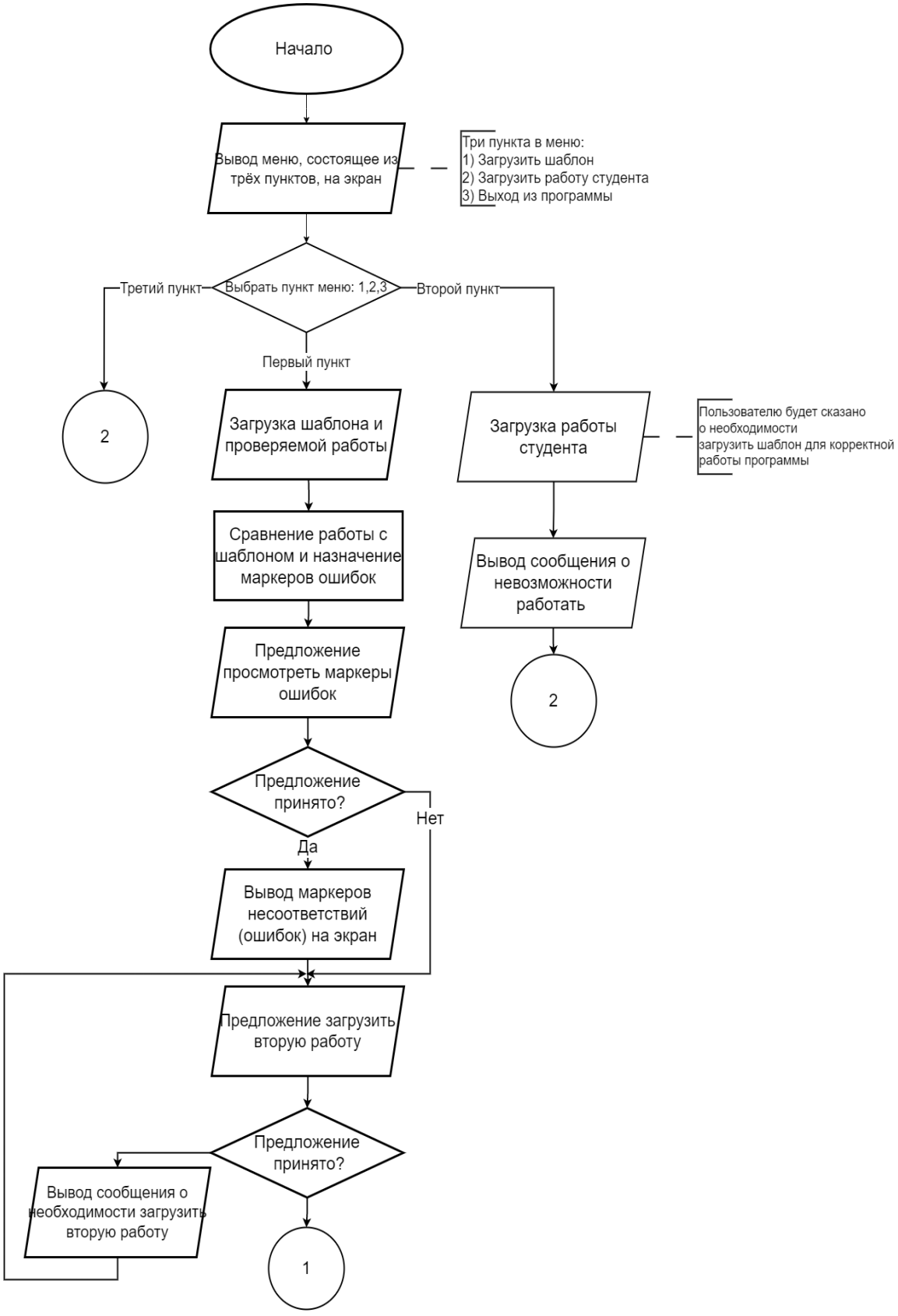


Рисунок 2 – Первая часть блок-схемы алгоритма работы программы

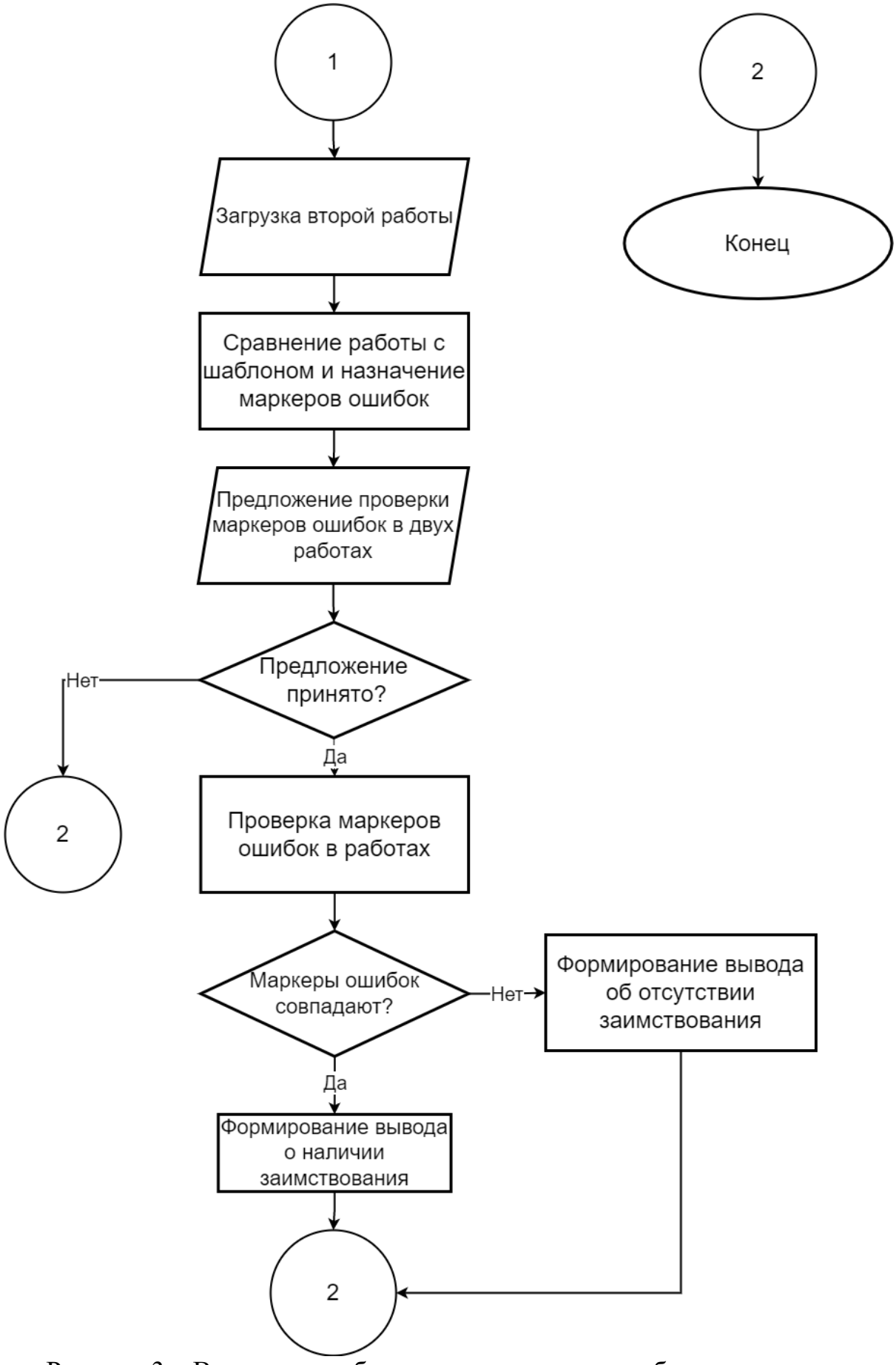


Рисунок 3 – Вторая часть блок-схемы алгоритма работы программы

При выполнении сравнения работы с шаблоном используются функции программы, описанные в главе 4. Принцип установления маркеров несоответствий (ошибок) происходит по одному и тому же алгоритму, блок-схема которого показана на рисунке 4.

Вначале вызывается функция проверки данных шаблона и студенческой работы. Результаты работы функции сохраняются в массивы. Для шаблона и студенческой работы создаётся свой массив. Затем создаётся массив маркеров ошибок, в который записываются все несовпадения студенческой работы с шаблоном. Далее создаётся цикл, в котором сравниваются массивы данных поэлементно. В случае, если какой-то элемент массива данных студенческой работы не соответствует элементу данных шаблона, то такой элемент записывается в массив маркеров ошибок.

В конце в результате возвращается массив ошибок. Если он пустой, то данные шаблона и студенческой работы одинаковы.

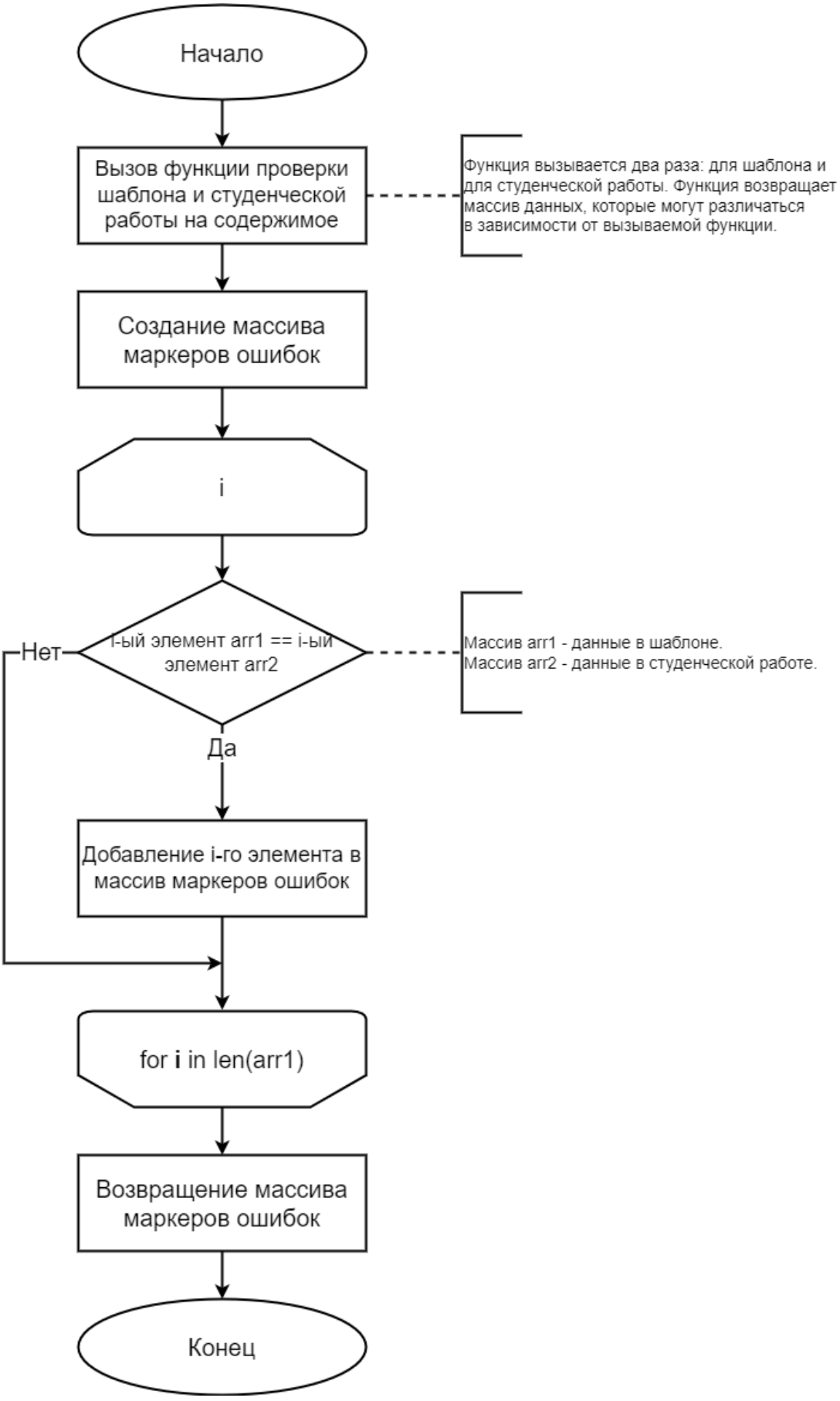


Рисунок 4 – Блок-схема работы алгоритма установления маркеров

4 РЕАЛИЗАЦИЯ И ТЕСТИРОВАНИЕ ПРОГРАММЫ

Для разработки программы был выбран язык программирования Python в среде PyCharm Community Edition. Данный язык был выбран по нескольким причинам:

1) широкая поддержка библиотек для работы с файлами формата .xlsx, таких как openpyxl, xlsxwriter, xlrd, xlwt, pandas и др.;

2) простота и удобство работы с данными в формате .xlsx благодаря возможности использования объектов и методов библиотек для работы с таблицами;

3) возможность автоматизации процессов обработки данных в формате .xlsx, например, с помощью скриптов (функций) на Python;

4) гибкость и масштабируемость Python позволяют работать с файлами .xlsx любой сложности и объема.

4.1 Используемые библиотеки

Для работы с электронными таблицами были использованы следующие библиотеки:

- pandas;
- openpyxl.

Для использования векторного представления был использован метод Word2Vec из библиотеки genism.models. Для дальнейшего их сравнения косинусного сходства был взят метод spatial из библиотеки scipy.

Для удобства использования методов были использованы библиотеки numpy для массивов и datetime для преобразования времени в удобный формат.

Библиотека openpyxl - это мощный инструмент для работы с файлами формата .xlsx на языке программирования Python. Она позволяет создавать,

редактировать и анализировать таблицы Excel, а также осуществлять автоматическую обработку данных.

Среди основных возможностей openpyxl можно выделить:

- чтение и запись данных в ячейки таблицы;
- создание и удаление листов;
- форматирование ячеек, строк и столбцов;
- работа с формулами и функциями Excel;
- сохранение и загрузка файлов формата .xlsx [6].

Библиотека openpyxl имеет простой и удобный интерфейс, который позволяет быстро освоить основные функции. Кроме того, она активно поддерживается сообществом разработчиков, что гарантирует ее стабильную работу и обновление до последних версий.

Openpyxl также обладает высокой производительностью, что позволяет обрабатывать большие объемы данных без задержек и ошибок. Это делает ее идеальным выбором для автоматизации процессов работы с таблицами Excel в Python [7].

Библиотека pandas - это высокоуровневая Python библиотека для анализа данных. Она называется высокоуровневой, потому что построена она поверх более низкоуровневой библиотеки NumPy (написана на Си), что является большим плюсом в производительности. В экосистеме Python, pandas является наиболее продвинутой и быстроразвивающейся библиотекой для обработки и анализа данных. В библиотеке pandas определены два класса объектов для работы с данными:

- Series;
- DataFrame [8].

Series — структура/объект Series представляет из себя объект, похожий на одномерный массив (питоновский список, например), но отличительной его чертой является наличие ассоциированных меток, т.н. индексов, вдоль каждого элемента из списка. Такая особенность превращает его в ассоциативный массив или словарь в Python. В строковом представлении

объекта Series, индекс находится слева, а сам элемент справа. Если индекс явно не задан, то pandas автоматически создаёт RangeIndex от 0 до N-1, где N общее количество элементов. Также стоит обратить, что у Series есть тип хранимых элементов. У объекта Series есть атрибуты, через которые можно получить список элементов и индексы, это values и index соответственно.

DataFrame — объект DataFrame лучше всего представлять себе в виде обычной таблицы и это правильно, ведь DataFrame является табличной структурой данных. В любой таблице всегда присутствуют строки и столбцы. Столбцами в объекте DataFrame выступают объекты Series, строки которых являются их непосредственными элементами. Объект DataFrame имеет 2 индекса: по строкам и по столбцам. Если индекс по строкам явно не задан (например, колонка по которой нужно их строить), то pandas задаёт целочисленный индекс RangeIndex от 0 до N-1, где N это количество строк в таблице [9].

4.2 Описание функций

Для решения поставленной задачи было реализовано несколько функций, которые проверяют выбранную работу с шаблоном.

Функция color_check() показывает цвет ячеек в указанном диапазоне и возвращает массив значений цветов ячеек. Например, на рисунке 5 показан пример в листе «Форматирование». По заданию требуется закрасить ячейки в произвольный цвет.

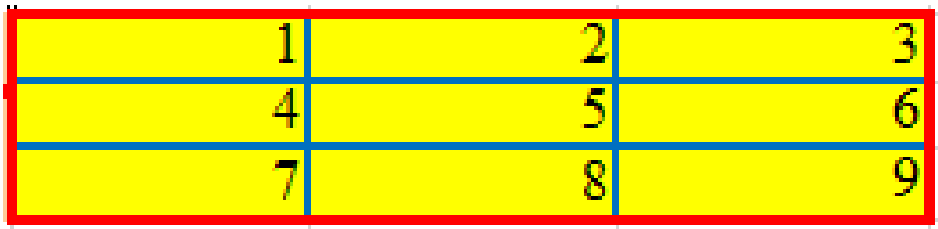


Рисунок 5 – Закрашенные ячейки в листе «Форматирование»

Для такого цвета функция возвращает массив, который показан на рисунке 6. Каждое значение равняется «FFFFFF00», что соответствует цвету «Желтый».

Рисунок 6 – Массив цветов

Функция font_size_check() проверяет на выбранном диапазоне ячеек выбранный шрифт и размер текста. По заданию требуется установить в листе «Форматирование» шрифт Times New Roman и размер текста 14пт. Если задание выполнено верно, то функция вернёт массив с одним значением шрифта и размера. Если задание выполнено неверно, т.е. будет найден другой шрифт или размер, то функция вернёт весь массив значений, где каждый элемент имеет значение «[Шрифт, размер]».

Результат работы функции показан на рисунке 7. В данном случае работа выполнена верно, и результатом будет массив с одним значением.

Рисунок 7 – Работа функции font_size_check()

Функция check_data() возвращает массив значений из заданного диапазона листа «Автозаполнение». Результат работы функции показан на рисунке 8. В данном случае выводится массив дат, которые были указаны в столбце.

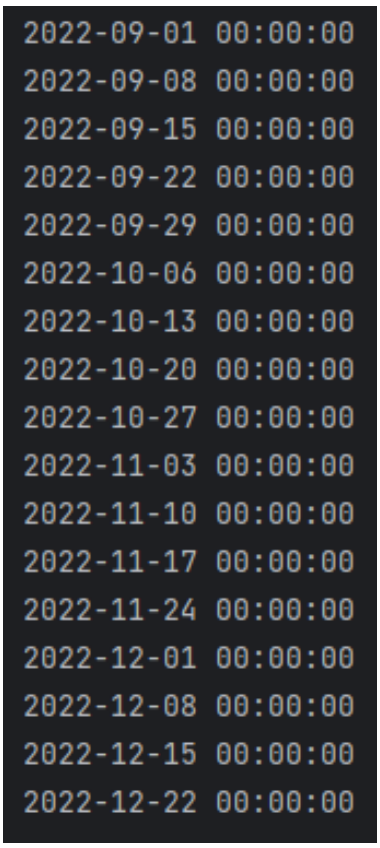


Рисунок 8 – Результат работы функции check_data()

Функция check_format_data() принимает на вход путь до файла и название листа, в котором требуется проверить корректность формата. Возвращает массив форматов каждой ячейки из указанного диапазона. Результаты работы функции показаны на рисунках 9-11.

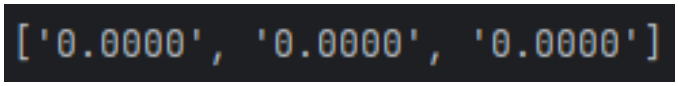


Рисунок 9 – Числовой формат данных

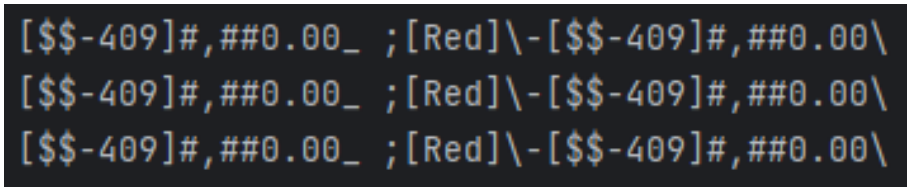


Рисунок 10 – Денежный формат данных

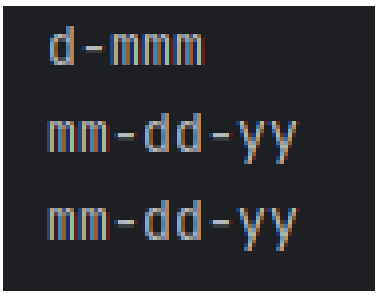


Рисунок 11 – Формат даты для данных

Функция get_create_modified_time() позволяет получить информацию о дате и времени создания файла и последних изменений в файле. Результат работы функции показан на рисунке 12. Для тестирования были внесены незначительные изменения в файл для получения даты изменения файла. Вверху будет показана дата создания файла, внизу – дата последнего изменения файла.

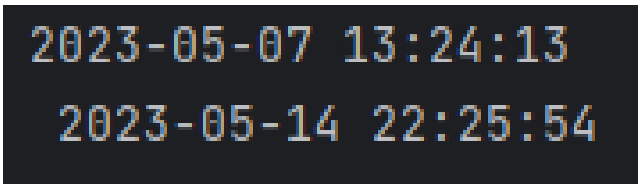


Рисунок 12 – Результат работы функции get_create_modified_time()

Все вышеописанные функции нужны для того, чтобы при проверке работ студентов при помощи этих функций можно было установить маркер в местах несовпадений или ошибках. Для выявления заимствования будут сравниваться маркеры ошибок и несоответствий.

4.3 Тестирование программы

На данном этапе будет происходить тестирование работоспособности программы. На рисунке 13 представлено меню, с помощью которого осуществляется работа с программой.

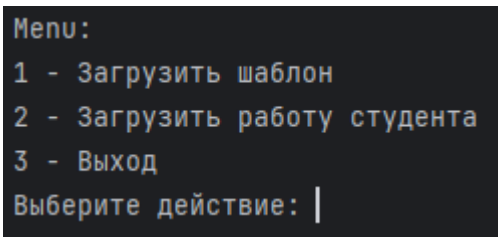


Рисунок 13 – Меню программы

В меню у нас есть на выбор три пункта. При выборе первого пункта нам предложит указать путь до нашего шаблона с расширением .xlsx (рисунок 14). При выборе второго пункта нам предложит указать путь до работы студента и выйдет из программы, потому что дальнейшая работа без шаблона невозможна (рисунок 15). Последний пункт - это выход из программы и завершение работы (рисунок 16).

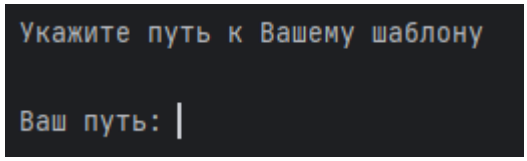


Рисунок 14 – Выбор первого пункта меню

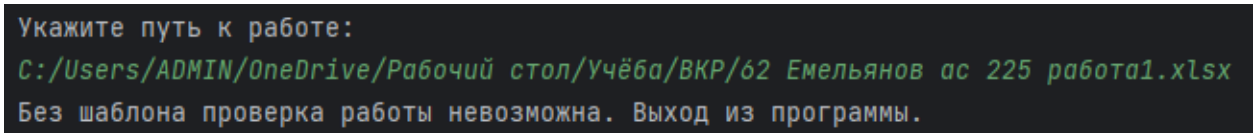


Рисунок 15 – Выбор второго пункта меню

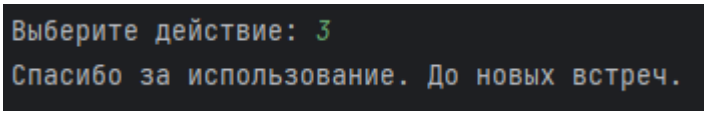


Рисунок 16 – Выбор третьего пункта меню

Так как второй и третий пункт подразумевают собой выход из программы для дальнейшего тестирования воспользуемся первым пунктом и загрузим шаблон (рисунок 17).

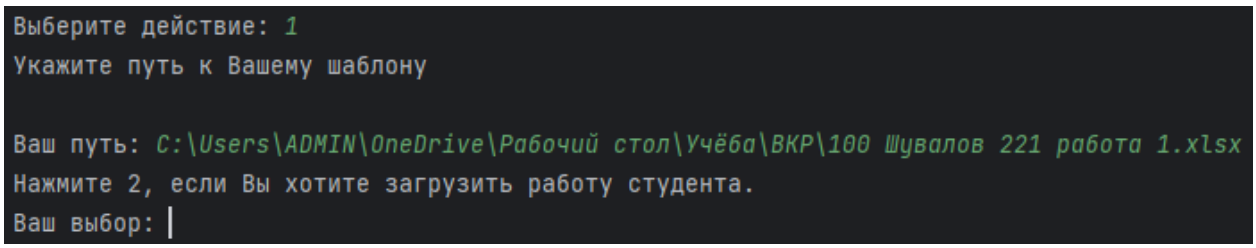


Рисунок 17 – Загрузка шаблона работы

Далее нам предлагают загрузить работу студента, которая будет сверяться с шаблоном. Для подтверждения действия нужно нажать на цифру 2.

После загрузки работы студента нам будет предложено начать проверку работы. Для подтверждения нужно нажать на цифру 1 на клавиатуре (рисунок 18).

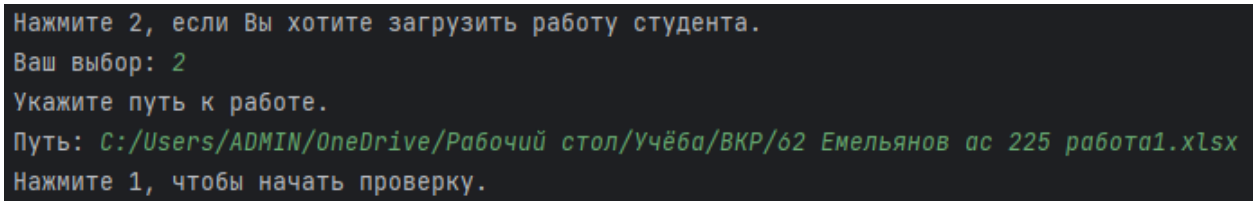


Рисунок 18 – Загрузка работы студента

После нажатия на цифру 1 при помощи описанных выше функций происходит проверка работы с шаблоном. После выполнения проверки работы нам предлагается посмотреть на имеющиеся несовпадения (рисунок 19).

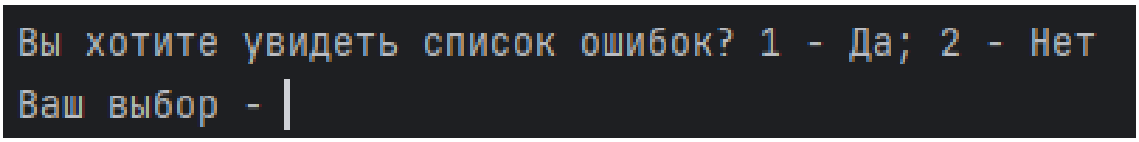


Рисунок 19 – Работа программы после выполнения проверки

Для тестирования выберем первый пункт и посмотрим на имеющиеся ошибки в работе студента. На рисунке 20 показаны все несовпадения, которые были найдены в ходе выполнения работы.

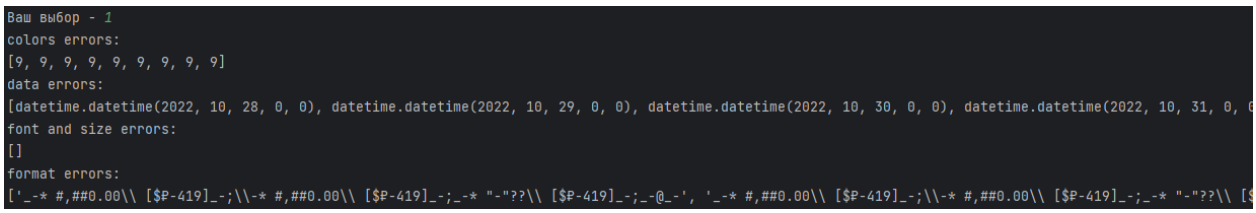


Рисунок 20 – Показ ошибок

Здесь мы видим ошибки по 4 критериям:

1) color errors;
2) data errors;
3) font and size errors;
4) format errors.

Критерий «color errors» показывает различия в заливке цветом таблицы в листе «Форматирование».

Критерий «data errors» показывает ошибки при выполнении задания в листе «Автозаполнение». В данном случае вывелся список ошибок при выполнении задания с датами.

Критерий «format and size errors» показывает ошибки при выполнении задания в листе «Форматирование». По заданию требуется, чтобы все данные были размером 14пт. И шрифтом Times New Roman. В данном случае это задание выполнилось студентом правильно, поэтому в этом критерии пустой список.

Критерий «format errors» показывает ошибки при выполнении задания в листе «Форматирование». По заданию требуется определённый формат данных, поэтому если формат студента не совпадает с шаблоном, то он появляется в этом списке ошибок.

Если мы не хотим видеть список ошибок, то выбираем пункт под номером 2. Нам предлагает загрузить вторую работу студента для проведения такой же проверки (рисунок 21).

```
Вы хотите увидеть список ошибок? 1 - Да; 2 - Нет
Ваш выбор - 2
Загрузить другую работу? 1 - Да; 2 - Нет
Ваш выбор - 1
Укажите путь к другой работе:
C:\Users\ADMIN\OneDrive\Рабочий стол\Учёба\ВКР\92 Рахматулин Артём 221 работа 1.xlsx
```

Рисунок 21 – Загрузка второй работы студента

После загрузки второй работы и дальнейшей её проверки нам предлагается так же посмотреть на ошибки второй работы. Так как мы уже видели, как работает данная функция, в этот раз выбирать её не будем. На рисунке 22 нам предлагается сравнить маркеры ошибок из двух работ.

```
Укажите путь к другой работе:
C:\Users\ADMIN\OneDrive\Рабочий стол\Учёба\ВКР\92 Рахматулин Артём 221 работа 1.xlsx
Начать проверку второй работы? 1 - Да; 2 - Нет:
1
Elements are same
Elements are same
Elements are same
Показать ошибки? 1 - Да; 2 - Нет
2
Сравнить ошибки двух работ? 1 - Да; 2 - Нет
```

Рисунок 22 – Работа программы после проверки первой работы студента

Нажимаем на цифру 1, чтобы посмотреть, имеются ли одинаковые ошибки в работах студентов. На рисунке 23 после выбора нужного пункта нам выводит сообщение о том, что в данном случае одинаковых ошибок не имеется, и программа начинает свою работу заново. На основе этого можно предполагать, что работы двух студентов выполнены самостоятельно.

```
Сравнить ошибки двух работ? 1 - Да; 2 - Нет
1
Одинаковых ошибок нет.
```

Рисунок 23 – Сообщение об отсутствии одинаковых ошибок

Теперь протестируем второй способ, в котором будет наличие одинаковых ошибок. Для тестирования внесём изменения во вторую работу, взяв данные из первой работы студента. В качестве примера сделаем одинаковый цвет таблицы из листа «Форматрование». На рисунке 24 показан результат выполнения проверки.

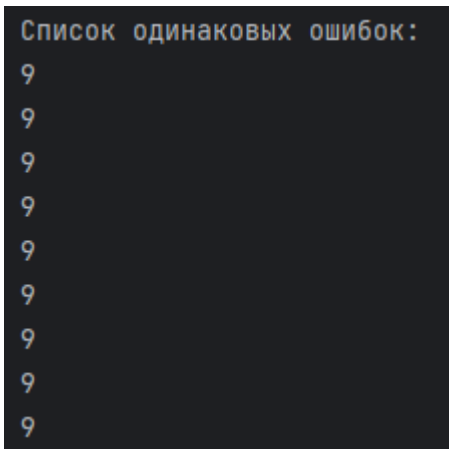


Рисунок 24 – Результат проверки с наличием одинаковой ошибки

В результате вывелся список ошибок, состоящий из цифр 9. Это говорит о том, что для закрашивания ячеек в листе «Форматирование» был использован цвет с таким кодом. Так как коды цвета в двух работах совпадают, они вывелись в список. На основе этого можно сделать вывод о том, что одна работа была позаимствована от другой. Полный листинг программы предоставлен в приложении.

ЗАКЛЮЧЕНИЕ

В рамках моей выпускной работы была реализована программа для поиска заимствований по электронным таблицам.

Для достижения цели были выполнены следующие задачи:

- выполнен анализ предметной области и имеющихся решений, в результате которого было выбрано наиболее подходящий метод достижения поставленной цели;

- уточнены требования к программе;

- спроектирована модель программы;

- реализована и протестирована программа.

В итоге можно сказать, что программа работает корректно, свои задачи выполняет.

Реализованная программа может быть полезна для преподавателей, потому что она позволит слегка облегчить работу при выполнении проверок работ в среде MS Excel.

В дальнейшем планируется реализация графической части приложения в виде удобного меню с кнопками и показом ошибок не только в текстовом виде, но и в графическом.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Unicheck [Электронный ресурс]. – URL: https://dev.abcdef.wiki/wiki/Unicheck (дата обращения 20.02.2023).

2 Программа для проверки на плагиат: Plagiarism Checker X [Электронный ресурс]. – URL: https://kakdelateto.ru/programma-dlya-proverki-na-plagiat-plagiarism-checker-x/ (дата обращения: 20.02.2023).

3 Топ-5 самых надежных инструментов для проверки на плагиат [Электронный ресурс]. – URL: https://smodin.io/ru/blog/best-plagiarism-checker-top-5-plagiarism-checker/ (дата обращения: 20.02.2023).

4 Word2vec в картинках [Электронный ресурс]. – URL: https://habr.com/ru/articles/446530/https://lala.lanbook.com/5-besplatnyh-analogov-antiplagiata-dlya-prepodavatelej (дата обращения: 22.02.2023).

5 Word2Vec – Википедия [Электронный ресурс]. – URL: https://ru.wikipedia.org/wiki/Word2vec (дата обращения: 24.02.2023).

6 Модуль openpyxl в Python, работа с файлами XLSX [Электронный ресурс]. – URL: https://docs-python.ru/packages/modul-openpyxl/ (дата обращения: 3.05.2023).

7 openpyxl PyPI [Электронный ресурс]. – URL: https://pypi.org/project/openpyxl/ (дата обращения: 3.05.2023).

8 Модуль.padnas [Электронный ресурс]. – URL: https://academy.yandex.ru/handbook/python/article/modul-pandas (дата обращения: 26.04.2023).

9 Введение в pandas: анализ данных на Python [Электронный ресурс]. – URL: https://khashtamov.com/ru/pandas-introduction (дата обращения: 02.05.2023).

ПРИЛОЖЕНИЕ

Листинг программы для поиска заимствований по файлам электронных таблиц.

```
# -*- coding: utf-8 -*-
import numpy as np
import pandas as pd
import openpyxl
import os
from datetime import datetime
from gensim.models import Word2Vec
from scipy import spatial
df = pd.read_excel("C:/Users/ADMIN/OneDrive/Рабочий
стол/Учёба/ВКР/100 Шувалов 221 работа 1.xlsx")
total = 0
def color_check(way):
    colors = []
    workbook = openpyxl.load_workbook(way)
    sheet = workbook['Форматирование']
    cell_range = sheet['G30':'I32']
    for row in cell_range:
        for cell in row:
            g_color = cell.fill.start_color.index
            colors.append(g_color)
    return colors
def func(a):
    sentences = []
    for row in a.iterrows():
        sentence = [str(cell) for cell in row[1]]
        sentences.append(sentence)
    model = Word2Vec(sentences, min_count=1)
    vectors = []
 for row in a.iterrows():
        vector = []
        for cell in row[1]:
            cell_vector = model.wv[str(cell)]
            vector.append(cell_vector)
        vectors.append(vector)
    vectors_np = np.array(vectors)
    return vectors_np
def cosine_similarity(path):
    df_2 = pd.read_excel(path)
    df_arr = func(df)
    df_2_arr = func(df_2)
    a = df_arr.shape
```

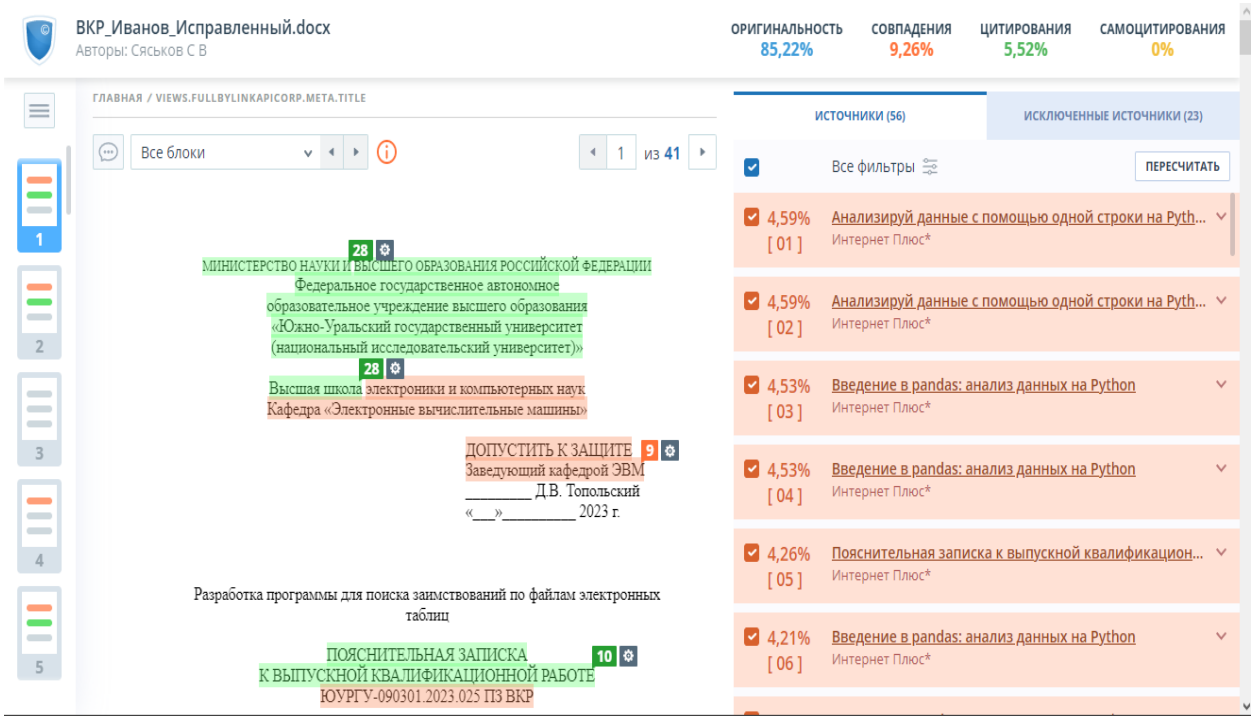
```
    b = df_2_arr.shape
    similarity_matrix = 1 - spatial.distance.cosine(a, b)
    return similarity_matrix
def font_size_check(path):
    arr = []
    wb = openpyxl.load_workbook(path)
    sheet = wb["Форматирование"]
    cell_range = sheet['G17':'I28']
    for row in cell_range:
        for cell in row:
            arr.append(cell.font.name)
            arr.append(cell.font.size)
            for i in range(len(arr)):
                if arr[i] == arr[i + 1]:
                    return arr[0]
                else:
     return arr
def check_data(path):
    wb = openpyxl.load_workbook(path)
    sh = wb["Автозаполнение"]
    list = []
    for row in sh.iter_rows(min_row=7, max_row=23, min_col=12,
max_col=12):
        for cell in row:
            list.append(cell.value)
    return list
def check_format_data(path):
    f_list = []
    wb = openpyxl.load_workbook(path)
    sh = wb['Форматирование']
   cell_range = sh['G20':'I20']
    for row in cell_range:
        for cell in row:
            f_list.append(cell.number_format)
    return f_list
def get_create_modified_time(way):
    created_time = os.path.getctime(way)
    created_time_formatted =
datetime.fromtimestamp(created_time).strftime('%Y-%m-%d %H:%M:%S')
    modified_time = os.path.getmtime(way)
    modified_time_formatted =
datetime.fromtimestamp(modified_time).strftime('%Y-%m-%d
%H:%M:%S')
```

```
    print(created_time_formatted,'\n',modified_time_formatted)
def check_data_marker(way):
arr1 = check_data(path_shab)
arr2 = check_data(way)
    error_list = []
    for i in range(len(arr1)):
        if arr1[i] != arr2[i]:
            error_list.append(arr2[i])
    if len(error_list) == 0:
        print("Elements are same")
        return error_list
    else:
        return error_list
def color_check_marker(way):
    arr1 = color_check(path_shab)
    arr2 = color_check(way)
    error_list = []
    for i in range(len(arr1)):
        if arr2[i] != arr1[i]:
            error_list.append(arr2[i])
    if len(error_list) == 0:
        print("Elements are same")
        return error_list
    else:
        return error_list
def font_size_check_marker(way):
    arr1 = font_size_check(path_shab)
    arr2 = font_size_check(way)
    err_list = []
    for i in range(len(arr1)):
        if arr2[i] != arr1[i]:
            err_list.append(arr2[i])
    if len(err_list) == 0:
        print("Elements are same")
        return err_list
    else:
        return err_list
def check_format_data_marker(way):
    arr1 = check_format_data(path_shab)
    arr2 = check_format_data(way)
    err_list = []
    for i in range(len(arr1)):
        if arr1[i] != arr2[i]:
```

```
            err_list.append(arr2[i])
    if len(err_list) == 0:
        print("Elements are same")
        return err_list
    else:
        return err_list
while True:
    print("Menu:\n"
          "1 - Загрузить шаблон\n"
          "2 - Загрузить работу студента\n"
          "3 - Выход")
    number = int(input("Выберите действие: "))
    if number == 3:
        break
    elif number == 1:
        print("Укажите путь к Вашему шаблону\n")
        path_shab = input("Ваш путь: ")
        print("Нажмите 2, если Вы хотите загрузить работу студента.")
        n = int(input("Ваш выбор: "))
        if n == 2:
            print("Укажите путь к работе.")
            path_work = input("Путь: ")
            print("Нажмите 1, чтобы начать проверку.")
            y = int(input("Ваш выбор: "))
            if y == 1:
                m_a = color_check_marker(path_work)
                m_b = check_data_marker(path_work)
                m_c = font_size_check_marker(path_work)
                m_d = check_format_data_marker(path_work)
                m_errors = []
                m_errors.extend(m_a)
                m_errors.extend(m_b)
                m_errors.extend(m_c)
                m_errors.extend(m_d)
                print("Вы хотите увидеть список ошибок? 1 - Да; 2 - Нет")
                choose = int(input("Ваш выбор - "))
                if choose == 1:
                    print("colors errors: \n"
                          f"{m_a}\n"
                          "data errors:\n"
                          f"{m_b}\n"
                          "font and size errors:\n"
                          f"{m_c}\n"
```

```
                "format errors:\n"
                f"{m_d}")
        elif choose == 2:
            print("Загрузить другую работу? 1 - Да; 2 - Нет")
            choose2 = int(input("Ваш выбор - "))
            if choose2 == 1:
                path_work_2 = input("Укажите путь к другой работе: \n")
                choose_2_work = int(input("Начать проверку второй
работы? 1 - Да; 2 - Нет: \n"))
                if choose_2_work == 1:
                    m_a_2 = color_check_marker(path_work_2)
                    m_b_2 = check_data_marker(path_work_2)
                    m_c_2 = font_size_check_marker(path_work_2)
                    m_d_2 = check_format_data_marker(path_work_2)
                    m_2_errors = []
                    m_2_errors.extend(m_a_2)
                    m_2_errors.extend(m_b_2)
                    m_2_errors.extend(m_c_2)
                    m_2_errors.extend(m_d_2)
                    err = int(input("Показать ошибки? 1 - Да; 2 - Нет\n"))
                    if err == 1:
                        print("colors errors: \n"
                            f"{m_a_2}\n"
                            "data errors:\n"
                            f"{m_b_2}\n"
                            "font and size errors:\n"
                            f"{m_c_2}\n"
                            "format errors:\n"
                            f"{m_d_2}")
                        check_errors = int(input("Сравнить ошибки двух
работ? 1 - Да; 2 - Нет\n"))
                        if check_errors == 1:
                            show_errors = []
                            for i in range(len(m_2_errors)):
                                if m_errors[i] == m_2_errors[i]:
                                    show_errors.append(m_2_errors[i])
                            if len(show_errors) == 0:
                                print("Одинаковых ошибок нет.")
                            else:
                                print("Список одинаковых ошибок: ")
                                for i in range(len(show_errors)):
                                    print(show_errors[i])
                    elif err == 2:
```

```
                        check_errors = int(input("Сравнить ошибки двух
работ? 1 - Да; 2 - Нет\n"))
                        if check_errors == 1:
                            show_errors = []
                            for i in range(len(m_2_errors)):
                                if m_errors[i] == m_2_errors[i]:
                                    show_errors.append(m_2_errors[i])
                            if len(show_errors) == 0:
                                print("Одинаковых ошибок нет.")
                            else:
                                print("Список одинаковых ошибок: ")
                                for i in range(len(show_errors)):
                                    print(show_errors[i])
                        else:
                            print("Две работы были проверены на ошибки.
Выход из программы.")
                            break
                    elif choose_2_work == 2:
                        print("Вторая работа загружена. Выход из
программы.")
                        break
                elif choose == 2:
                    print("Дальнейшая работа программы не имеет смысла.
Выход из программы.")
                    break
    elif number == 2:
        path_work = input("Укажите путь к работе:\n")
        print("Без шаблона проверка работы невозможна. Выход из
программы.")
        break
print("Спасибо за использование. До новых встреч.")
print("Нажмите любую клавишу для выхода из программы.")
input()
```


ВКР_Иванов_Исправленный.docx
Авторы: Сяськов С В
ОРИГИНАЛЬНОСТЬ 85,22%
СОВПАДЕНИЯ 9,26%
ЦИТИРОВАНИЯ 5,52%
САМОЦИТИРОВАНИЯ 0%
ГЛАВНАЯ / VIEWS.FULLBYLINKAPICORP.META.TITLE
Все блоки
1 из 41
МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»
ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой ЭВМ
_____ Д.В. Топольский
«__»_____ 2023 г.
Разработка программы для поиска заимствований по файлам электронных таблиц
ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУРГУ-090301.2023.025 ПЗ ВКР
ИСТОЧНИКИ (56)
ИСКЛЮЧЕННЫЕ ИСТОЧНИКИ (23)
Все фильтры
ПЕРЕСЧИТАТЬ
4,59% [01] Анализируй данные с помощью одной строки на Pyth... Интернет Плюс*
4,59% [02] Анализируй данные с помощью одной строки на Pyth... Интернет Плюс*
4,53% [03] Введение в pandas: анализ данных на Python Интернет Плюс*
4,53% [04] Введение в pandas: анализ данных на Python Интернет Плюс*
4,26% [05] Пояснительная записка к выпускной квалификацион... Интернет Плюс*
4,21% [06] Введение в pandas: анализ данных на Python Интернет Плюс*