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,
к.т.н., доцент
_____ Д.В. Топольский
«___»_____ 2023 г.

АВТОМАТИЗИРОВАННЫЙ СБОР ДАННЫХ ДЛЯ БИБЛИОМЕТРИЧЕСКОГО АНАЛИЗА ПУБЛИКАЦИЙ

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУрГУ-090301.2023.049 ПЗ ВКР

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

Автор работы,
студент группы КЭ-406
_____ А.Е. Лекомцев
«___»_____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-406
Лекомцеву Александру Евгеньевичу,
обучающемуся по направлению
09.03.01 «Информатика и вычислительная техника»

1. Тема работы: «Автоматизированный сбор данных для библиометрического анализа публикаций» утверждена приказом по университету от «25» апреля 2023 г. №753-13/12, приложение №308/10

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

3. Исходные данные к работе:

3.1 Разрабатываемое приложение предназначено для упрощения работы с электронной библиотекой eLIBRARY.ru, интегрированной с Российским индексом научного цитирования, для научных работников.

3.2 Приложение должно работать на операционной системе Microsoft Windows, а также обеспечивать следующий функционал:

– поиск научных публикаций по заданным параметрам;

– сбор информации о публикациях, выбранных пользователем с веб-сайта (парсинг) и вывод ее на экран;

– наличие функции предварительного просмотра публикаций и возможность дополнительной обработки данных;

– наличие функции перевода результатов в различные форматы для программ наукометрического анализа;

– наличие функции сохранения результатов поиска.

4. Перечень подлежащих разработке вопросов:

– изучить поисковую систему в сервисе eLIBRARY: возможности, ограничения, виды запросов;

– провести сравнительный анализ существующих программных решений, реализующих похожий функционал;

– провести выбор средств реализации на основе сравнительного анализа и разработать приложение;

– протестировать разработанное программное обеспечение.

Дата выдачи задания: ___ декабря 2022 г.

Руководитель работы _____/ М.А. Алтухова /

Студент _____/ А.Е. Лекомцев /

КАЛЕНДАРНЫЙ ПЛАН

Этап	Срок сдачи	Подпись руководителя
Введение и обзор предметной области	01.03.2023	
Сравнительный анализ существующих программных решений	01.04.2023	
Выбор средств и реализация системы	01.05.2023	
Тестирование и отладка	15.05.2023	
Компоновка текста работы и сдача на нормоконтроль	24.05.2023	
Подготовка презентации и доклада	30.05.2023	

Руководитель работы _____/ М.А. Алтухова /

Студент _____/ А.Е. Лекомцев /

АННОТАЦИЯ

Лекомцев А.Е. Автоматизированный сбор данных для библиометрического анализа публикаций. – Челябинск: ФГАОУ ВО «ЮУрГУ (НИУ)», ВШ ЭКН; 2023г., 47 с., 16 ил., библиогр. список – 15 наим.

Целью данной работы являлась разработка программы для автоматизации сбора данных для библиометрического анализа из библиографической и реферативной базы eLIBRARY. Разработано программное решение, которое автоматизирует процесс сбора данных по заданным параметрам из базы eLIBRARY, обеспечивая ученым доступ к необходимой информации для проведения анализа и исследований в области наукометрии и библиометрии.

В выпускной квалификационной работе проведена автоматизация сбора информации о каждой публикации выданной системой eLIBRARY и разработан программный продукт, выполняющий сбор данных, и их экспорт для дальнейшего библиометрического анализа.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Библиометрия – это наука, которая изучает качественные и количественные характеристики научных публикаций и авторов, используя математические и статистические методы. Она была разработана в начале 1960-х годов и с тех пор широко используется в различных областях, включая технологии и медицину. Библиометрия помогает оценить научный вклад авторов, идентифицировать наиболее цитируемые работы и публикации, а также определить наиболее эффективные научные журналы.

Одной из важных областей, в которых применяется библиометрия, является наукометрия – это дисциплина, которая изучает научную деятельность и ее результаты. Она использует методы библиометрии, чтобы оценить научную продуктивность и влияние авторов, научных групп и организаций [1].

Один из основных инструментов библиометрии и наукометрии – это библиометрический анализ. Он позволяет оценить научную продуктивность и влияние научных публикаций, используя различные параметры, такие как число цитирований, индекс Хирша, графы цитирования и другие [2].

Научные публикации, требуемые для проведения наукометрического или библиометрического анализа, хранятся в специальных цифровых базах данных, а именно библиографических и реферативных. В подобных базах кроме названия и имени автора публикации, могут присутствовать следующие поля: ключевые слова, аннотация, данные о цитировании и тематике. Самые известные международные библиографические и реферативные базы: Web of Science и Scopus; а в РФ – eLIBRARY [3]. Это крупнейшая в России электронная библиотека научных публикаций, обладающая богатыми возможностями поиска и анализа научной информации. Библиотека интегрирована с Российским индексом научного цитирования (РИНЦ) – созданным по заказу Минобрнауки РФ бесплатным общедоступным инструментом измерения публикационной активности ученых и организаций [4].

В последние два десятка лет наукометрия широко распространилась благодаря данным из баз Web of Science и Scopus. Из этих библиографических и реферативных баз можно выгружать информацию о публикациях, в различных форматах, а затем пользоваться этими данными для исследований в специализированных программах для анализа данных, но данный функционал доступен только после подтверждения своей принадлежности к университету или НИИ на сайте. В то же время, eLIBRARY не обладает функционалом для экспортирования данных для анализа, а также не предоставляет доступ к API никому, кроме организаций, которые оплатят право пользования. Это очень сильно усложняет сбор данных о публикациях для простого ученого, потому что всю информацию приходится собирать вручную [5].

Так как для проведения анализа может потребоваться достаточно большой объем данных, то собирать информацию о публикациях вручную будет затруднительно. Исходя из этого факта, требуется разработка программы, которая будет самостоятельно собирать данные для анализа по заданным параметрам из библиографической и реферативной базы eLIBRARY.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

eLIBRARY – крупнейшая в России электронная библиотека научных публикаций, обладающая богатыми возможностями поиска и анализа научной информации. Библиотека интегрирована с Российским индексом научного цитирования (РИНЦ). Также система обладает обширной базой данных, которая охватывает различные области знаний, включая естественные и точные науки, гуманитарные и социальные науки, медицину, технику и многое другое. Сервис содержит как российские, так и зарубежные публикации, но его основной акцент сделан на российских исследованиях и научной литературе [6].

eLIBRARY предоставляет различные возможности для поиска, изучения и получения доступа к научным публикациям. Он включает в себя функциональности, такие как полнотекстовый поиск, фильтрацию результатов, поиск по ключевым словам, поиск по авторам и журналам, а также инструменты для анализа цитирований и изучения влияния научных работ.

Рассмотрим подробнее возможности, ограничения, виды запросов и другие аспекты, связанные с системой поиска в этой системе.

Поисковая система в eLIBRARY представлена в двух видах, стандартный поисковой модуль (рисунок 1) и расширенный (рисунок 2).

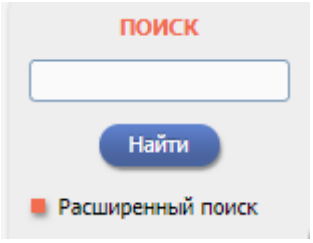

Рисунок 1 – Стандартный поиск в системе eLIBRARY

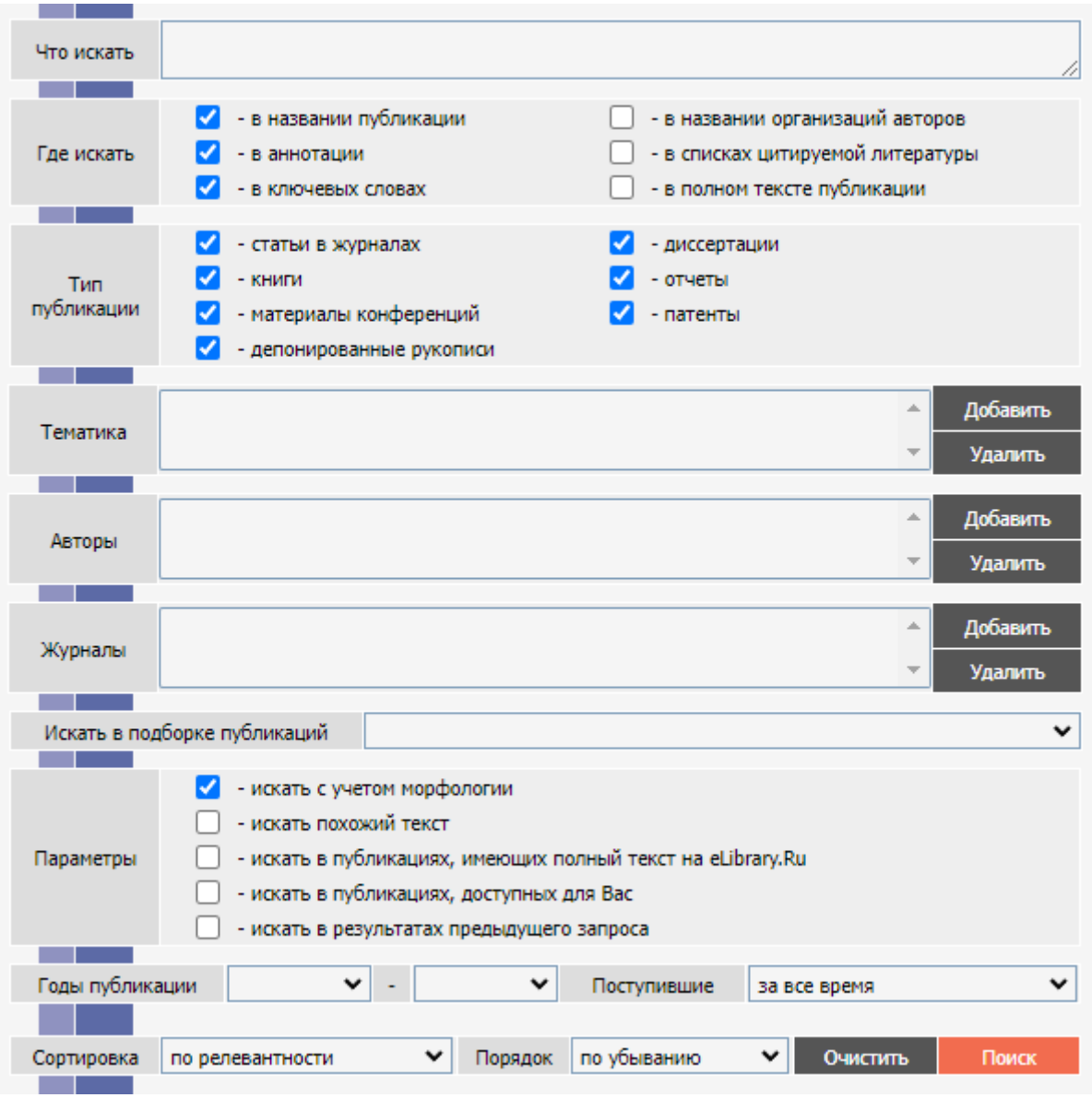

Рисунок 2 – Расширенный поиск в системе eLIBRARY

Расширенный поиск состоит из нескольких блоков давайте рассмотрим их подробнее.

Блок «Что искать» представлен в виде поля ввода, где пользователь вводит ключевые слова, фразы или запросы для поиска информации. Это основное поле, в которое пользователь может вводить свой запрос. В сущности, данный блок представляет собой стандартный поисковой модуль системы eLIBRARY.

Блоки «Где искать» представлен некоторым количеством «флажков выбора», отметив которые, можно выбрать, где именно будет происходить поиск.

Блок «Тип публикации» также, как и предыдущий блок состоит из «флажков», но в этом блоке уже будут отмечаться, что именно ищет пользователь.

В блоках «Тематика», «Авторы» и «Журналы» представлены две кнопки «Добавить» и «Удалить» и не редактируемое поле ввода. При нажатии на кнопку «Добавить» открывается новое окно браузера, в котором пользователь сможет выбрать тему публикации (рисунок 3), ее автора (рисунок 4) или журнал, в котором будет публикация (рисунок 5). После добавления критериев в эти блоки пользователь сможет удалить ненужные данные нажав на кнопку «Удалить».

Блок «Искать в подборке публикаций» представлен в виде «выпадающего списка», где пользователь выбирает подборку публикаций, которую он мог отложить в своем аккаунте на сайте eLIBRARY.

Блок «Параметры» состоит из «флажков», отметив которые, пользователь указывает как будет искаться информация, которую он ввел в блоках выше.

Блоки «Годы публикации» и «Поступившие» представлены «выпадающими списками» в которых пользователь выбирает в каком временном диапазоне система должна искать информацию.

Блоки «Сортировка» и «Порядок» также состоят из «выпадающих списков» в которых пользователь указывает в каком виде будут представлены результаты поиска.

Рисунок 3 – Страница выбора темы публикации в системе eLIBRARY

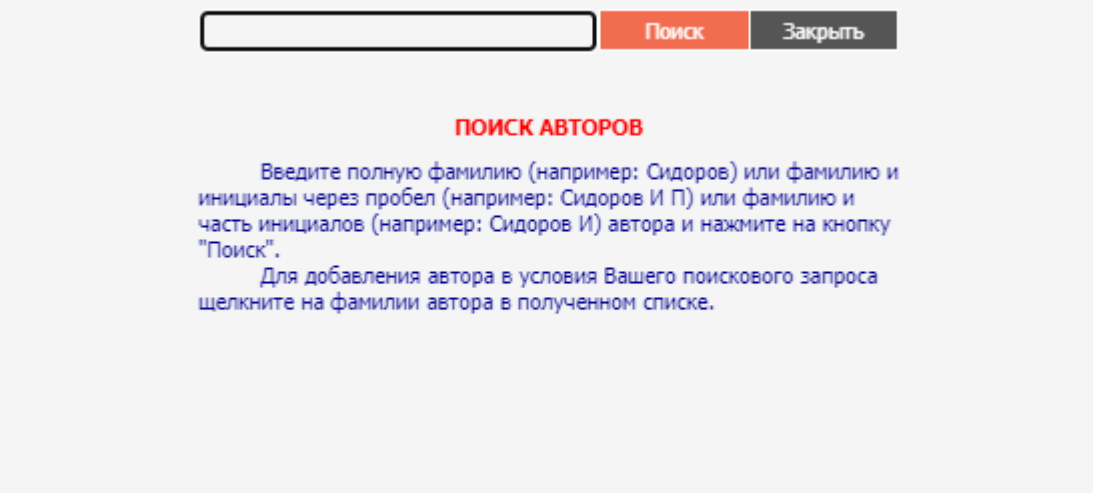

Рисунок 4 – Страница выбора автора публикации в системе eLIBRARY

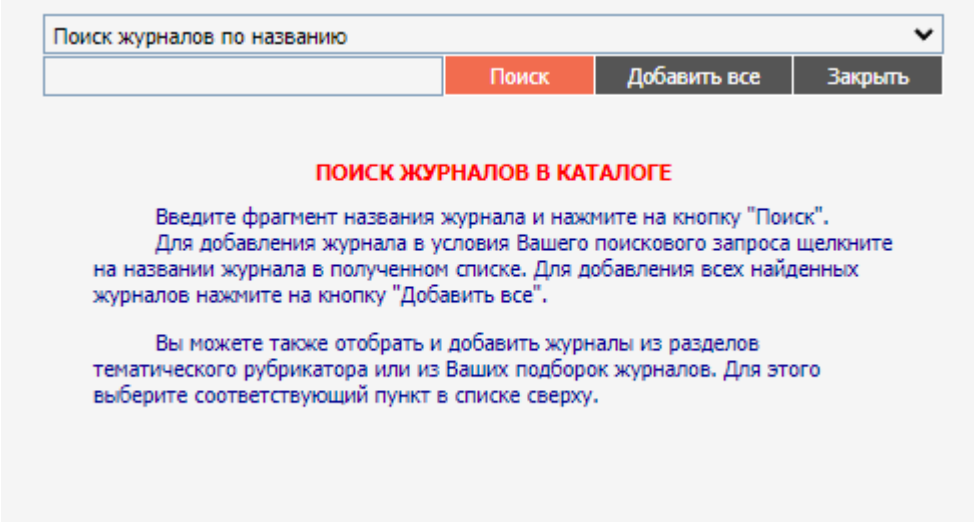

Рисунок 5 – Страница выбора журнала в системе eLIBRARY

Также, в расширенном поиске в системе eLIBRARY, есть кнопка «Очистить», при нажатии на которую поисковой запрос очищается, и кнопка «Поиск», нажимая на которую пользователь отправляет критерии своего запроса на сайт и в ответ получает результаты поиска.

В системе eLIBRARY предоставляется обширный объем результатов поиска, включающий значительное количество публикаций (на сентябрь 2021 года их число превышало 27 миллионов). Однако стандартным пользователям системы не разрешено выводить более 10 тысяч публикаций в одном запросе.

2 ОБЗОР АНАЛОГОВ

С каждым годом к исследованиям в сфере наукометрии растет интерес, проводится систематические обзоры литературы, выполняется систематизация и структурирование метаданных. Несмотря на то, что данные темы вызывают все больший интерес, существуют определенные ограничения, включая недостаток информации о российском сегменте научных публикаций и отсутствие сравнительных исследований в международном и российском контексте.

Возможно, в России цифровизация произошла с некоторым опозданием по сравнению с другими странами, что объясняет отсутствие возможности экспорта данных о публикациях прямо с сайта eLIBRARY без использования сторонних программ. Тем не менее, предвидится покупка доступа к API eLIBRARY и дальнейшая разработка автоматических сборщиков данных, но эта функциональность доступна только для организаций.

Лучший способ проведения автоматического сбора информации об интересующих публикациях с eLIBRARY – это разработка программы-парсера.

Электронная библиотека eLIBRARY не обладает функционалом экспорта результатов поиска и не содержит многих возможностей, которые присутствуют в других библиографических и реферативных базах. С появлением потребности в получении данных о научных публикациях прямо с сайта, разработчики eLIBRARY все равно не добавили эту функцию, а только предоставили возможность доступа к своему API за деньги, но такая возможность есть только у организаций, университетов и научно-исследовательских институтов (НИИ).

На сегодняшний день нет приложения, которое будет выводить информацию с сайта eLIBRARY по запросу пользователя, а если и есть, то это «кустарные» программы-парсеры. При поиске аналогов была обнаружена только одна рабочая программа, которая была разработана специально для eLIBRARY – «СЕБЗЕР», к сожалению, остальные разработки потеряли свою актуальность или были не доделаны и в последствии заброшены, поэтому в качестве аналогов

рассмотрим встроенные функции популярнейших библиографических и реферативных баз, таких как Web of Science и Scopus.

Программы и электронные сервисы со схожими функциями были выбраны по следующим критериям:

1) возможность бесплатного вывода данных из сайта,

2) экспорт данных в форматах xlsx, enw, BibTex, ris,

3) возможность редактирования и сортировки,

4) удобство и простота использования,

5) возможность использовать на eLIBRARY.ru.

Эти критерии отбора программ были выбраны автором, потому что нам нужна программа, которая будет бесплатно экспортировать данные о научных публикациях из сайтов библиографических и реферативных баз данных; выводить данные о публикации в наиболее часто используемых форматах (xlsx, enw, BibTex, ris), чтобы можно было использовать любую программу для библиографического анализа; должна быть удобной и простой, чтобы научный работник не смог запутаться в различных функциях программы; всё вышеперечисленное должно иметь возможность применения для системы eLIBRARY.

Рассмотрим программы, реализующие похожий функционал, как и у разрабатываемого приложения, выбранные автором:

1) средство экспорта библиографических записей из eLIBRARY.ru «СЕБЗЕР» [7],

2) система обработки данных на сайте Web of Science (WoS) [8],

3) система обработки данных на сайте Scopus [9].

Достоинством приложения «СЕБЗЕР» является то, что она позволяет выводить данные с сайта eLIBRARY.ru и делает это бесплатно.

Однако при этом «СЕБЗЕР» имеет следующие недостатки:

– не является полноценным приложением, а лишь дополнением к расширению браузера «Tampermonkey» [10];

– больше не поддерживается разработчиком;

– неудобный интерфейс и сложность освоения для «рядового» пользователя;

– позволяет вывести результаты только с типом публикации «статьи в журналах» и только в формате BibTeX;

– не позволяет редактировать результаты поиска перед их экспортом.

Далее рассмотрим систему обработки данных на сайте Web of Science (WoS), преимущество которой заключается в том, что она встроена в систему Web of Science, а не является сторонним приложением или дополнением.

В то же время данная система обладает следующими недостатками:

– позволяет экспортировать результаты поиска с сайта WoS в различных форматах (xlsx, enw, ris и несколько своих форматов), но только после подтверждения принадлежности к университету или НИИ;

– не может быть использована на других сайтах, кроме WoS;

– не имеет возможности редактировать данные в приложении.

Рассмотрим достоинства системы обработки данных на сайте Scopus. Основными достоинствами этой системы является то, что она позволяет больше углубиться в исследование, потому что Scopus предоставляет больше информации, чем остальные реферативные базы, а также данная система интегрирована на сайт Scopus, что делает сбор данных гораздо удобнее для пользователей.

Однако при этом система имеет следующие недостатки:

– позволяет экспортировать результаты поиска с сайта Scopus в различных форматах (xlsx, ris, enw, BibTex), но только после подтверждения принадлежности к университету или НИИ;

– не может быть использована на других сайтах, кроме Scopus.

– не имеет возможности редактировать данные в приложении.

Результаты сравнения рассмотренных программ по выбранным ранее критериям сведены в таблицу 1.

Таблица 1 – Преимущества и недостатки аналогов

Название / Критерий	СЕБЗЕР	Web of Science	Scopus
Возможность вывода данных из сайта бесплатно	+	-	-
Экспорт данных в форматах xlsx, enw, BibTex, ris	-	+	+
Возможность редактирования и сортировки	-	-	-
Удобство и простота использования	-	+	+
Возможность использовать на eLIBRARY.ru	+	-	-

3 ВЫБОР СРЕДСТВ РЕАЛИЗАЦИИ И РАЗРАБОТКА ПРИЛОЖЕНИЯ

3.1 Сравнительный анализ средств реализации

Приложение, которое мы разрабатываем, будет основываться на парсинге (синтаксическом анализе). Парсер – это программа, которая анализирует текстовые данные и извлекает соответствующую информацию на основе определенного набора правил или грамматики. Программа обычно принимает строку текста в качестве входных данных, а затем применяет алгоритмы синтаксического анализа для анализа структуры текста и определения его составных частей. Объектом парсинга может быть справочник, интернет-магазин, форум, блог и абсолютно любой интернет-ресурс.

Первым параметром для выбора средства реализации является язык программирования. В данном случае мы рассматриваем несколько языков программирования, которые наиболее подходят для создания парсера, таких как:

1) Python,

2) Java,

3) C/C++.

Критерии, по которым были выбраны именно эти языки:

1) производительность (скорость выполнения программы является ключевым фактором при работе с большим объемом данных);

2) простота использования: интерфейс и синтаксис языка программирования должны быть простыми и интуитивно понятными, чтобы упростить процесс разработки программы-парсера;

3) наличие библиотек и инструментов: языки программирования должны иметь подходящие библиотеки и инструменты для работы с различными типами данных и форматов.

Каждый язык программирования имеет свои преимущества и недостатки.

Python – популярный язык для создания программ-парсеров благодаря простоте использования, гибкости и удобочитаемости. Он имеет несколько популярных библиотек парсеров, таких как NLTK, BeautifulSoup и Selenium. NLTK – это комплексный набор инструментов для обработки естественного языка, который предоставляет широкий спектр инструментов для анализа текста, включая синтаксические анализаторы. BeautifulSoup – это библиотека веб-скрейпинга, которая полезна для извлечения данных из документов HTML и XML, а Selenium в Python – библиотека, которая позволяет автоматизировать взаимодействие с веб-браузерами, позволяя разработчикам выполнять различные действия, такие как заполнение форм, нажатие кнопок, переходы по ссылкам и сбор данных с веб-страниц. Она широко используется для тестирования веб-приложений, сбора данных и создания ботов. В Python имеются библиотеки, которые позволяют создавать графические интерфейсы. Одной из таких библиотек является Tkinter, которая является стандартной и предоставляет разнообразные инструменты и виджеты для построения окон, кнопок, полей ввода и других элементов пользовательского интерфейса. Благодаря своей простоте и распространенности, Tkinter является популярным выбором для разработки простых приложений с графическим интерфейсом.

Основные преимущества Python – это его простота, удобочитаемость и широкий спектр библиотек. Кроме того, это интерпретируемый язык, а это означает, что код можно быстро тестировать и модифицировать. Однако производительность Python может быть не такой высокой, как у других скомпилированных языков, а его динамическая типизация может привести к ошибкам в крупномасштабных проектах [11].

Java – популярный язык для создания синтаксических анализаторов благодаря сильной поддержке объектно-ориентированного программирования, производительности. В Java есть несколько популярных библиотек для парсинга, таких как ANTLR, JavaCC и Jison. ANTLR, например, – это мощный генератор синтаксических анализаторов, который поддерживает синтаксический анализ нескольких языков, включая Java, C и Python. Он имеет большое сообщество

пользователей и поставляется с визуальным отладчиком для устранения неполадок. JavaCC – это еще один генератор синтаксических анализаторов, который использует контекстно-свободную грамматику для создания синтаксических анализаторов, а Jison – это генератор синтаксических анализаторов JavaScript, который может генерировать синтаксические анализаторы для использования как в браузере, так и на стороне сервера. Для языка Java, библиотеки, такие как Jsoup, имеют высокую производительность, а также предоставляют больше возможностей для манипуляции с данными, что может быть полезно для сложных задач парсинга.

К основным преимуществам Java относятся масштабируемость и простота обслуживания. Однако некоторым разработчикам он может показаться слишком многословным или громоздким для написания и компиляции.

C/C++ – являются популярными языками для создания парсеров из-за их скорости и низкоуровневого доступа к системе. У них есть несколько популярных библиотек парсеров, таких как Bison, Flex и ANTLR. Bison – популярный генератор синтаксических анализаторов, который широко используется при разработке компиляторов, а Flex – популярный генератор сканеров, который можно использовать в сочетании с Bison для создания полного синтаксического анализатора. ANTLR – это генератор синтаксических анализаторов, который поддерживает широкий спектр языков программирования и может генерировать синтаксические анализаторы в нескольких выходных форматах.

Основные преимущества C/C++ включают их производительность, низкоуровневый доступ к системе и совместимость со многими платформами. Однако их сложнее изучать и использовать, чем другие языки, а их синтаксис может быть более сложным [12].

Для реализации поставленной задачи был выбран язык Python, поскольку он имеет простой синтаксис, множество библиотек, подходящих для нашей цели. Python является языком с открытым исходным кодом, что означает, что разработчики могут использовать и изменять код в соответствии со своими потребностями. Кроме того, наличие открытого исходного кода привлекает

большое количество разработчиков, которые вносят свой вклад в развитие языка и его библиотек. Это один из самых популярных языков программирования. Также, мы уже были знакомы с этим языком, что облегчит написание кода.

В качестве среды разработки будет использоваться официальная IDE (Интегрированная среда разработки) JetBrains PyCharm Community, в этой среде разработки присутствуют инструменты для анализа кода, графический отладчик, а также это программное обеспечение было создано именно для языка Python.

3.2 Разработка приложения

Алгоритм работы разрабатываемой программы представлен на рисунке 6.

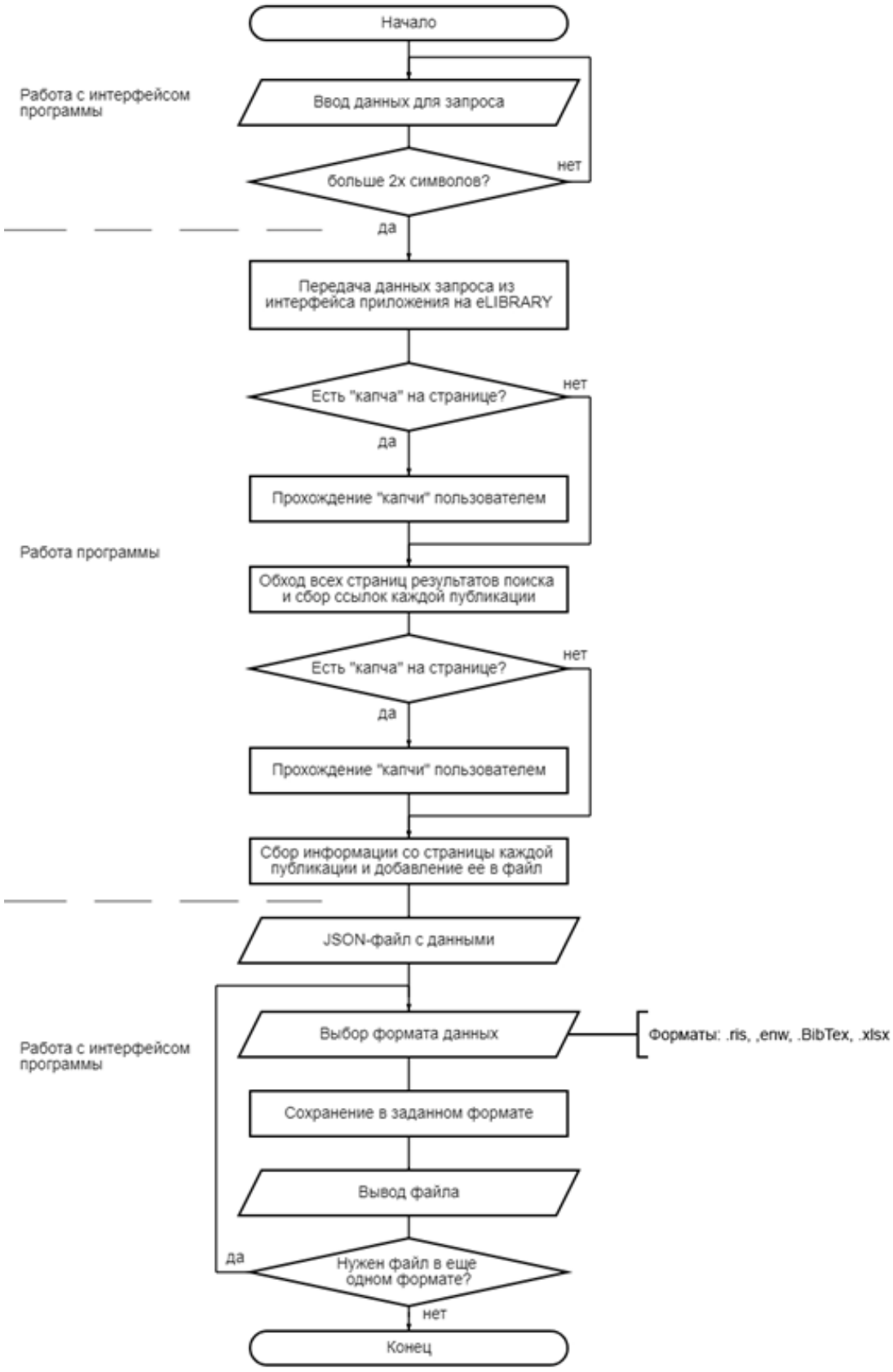

Рисунок 6 – Алгоритм работы программы

Библиотеки BeautifulSoup и Selenium. При запуске приложения пользователю предлагается ввести все необходимые критерии для поиска публикаций. Затем, с использованием библиотек, эти критерии передаются на сайт, создавая запрос [13]. После нажатия кнопки «Поиск» приложение ожидает загрузки результатов поиска с сайта. Когда сайт предоставляет результаты, приложение собирает информацию о количестве страниц, содержащих результаты, для того чтобы извлечь ссылки на каждую публикацию со всех страниц с результатами поиска [14].

После сбора всех ссылок на публикации, программа переходит к обходу каждой ссылки и извлекает данные о соответствующей публикации. Полученная информация о всех публикациях сохраняется в формате JSON в файле. Затем этот JSON-файл может быть преобразован в различные форматы, такие как xlsx, ris, enw, BibTex, для дальнейшего библиометрического анализа.

В ходе разработки приложения было решено реализовать интерфейс поисковой системы схожий с расширенным поисковым модулем системы eLIBRARY. В разрабатываемом приложении автор отказался от блоков «Тематика», «Авторы», «Журналы», которые описывались в первой главе, так же, как и от блока «Искать в подборке публикаций» и нескольких флагов в блоке «Параметры», чтобы оставаться незаметнее для системы безопасности сайта. В интерфейс программы был добавлен блок из четырех кнопок, при нажатии на которые можно перевести, полученный в ходе работы программы, JSON-файл, в форматы, соответствующие надписям на кнопках: “to Bibtex”, “to RIS”, “to ENW”, “to Excel” [15]. Интерфейс разработанного приложения представлен на рисунке 7.

3.3 Описание данных

Входные данные приложения представлены фильтрами поиска, которые пользователь указывает путем выбора соответствующих критериев. После ввода критериев пользовательские данные передаются в приложение. После этого, дополнительные данные от пользователя не требуются.

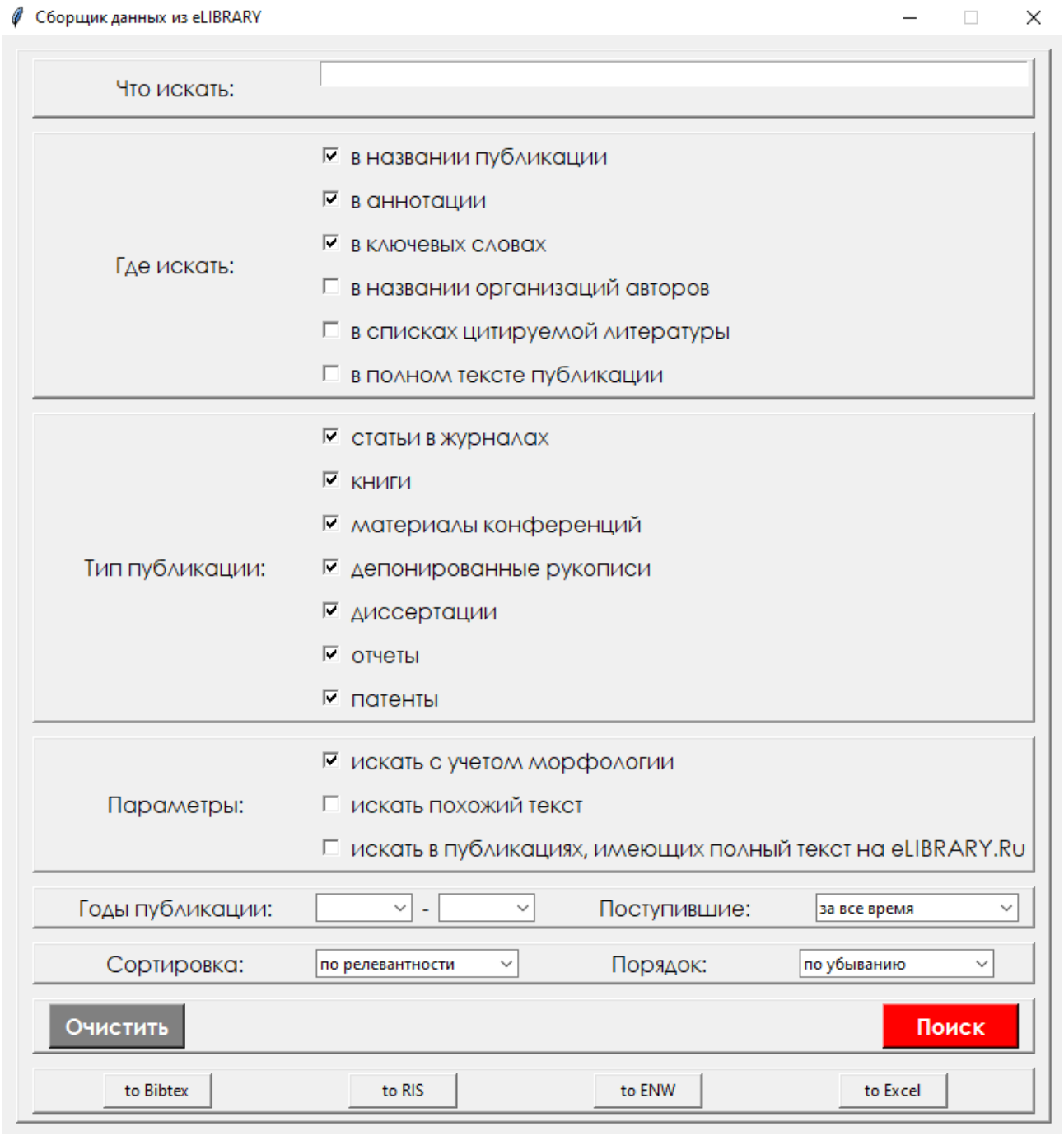

Рисунок 7 – Интерфейс программы

Промежуточные данные представлены HTML-кодом страницы, который приложение получает в ответ от сайта. Основываясь на промежуточных данных, программа формирует выходные данные, которые хранят в себе информацию о публикациях, и передает их пользователю.

4 ТЕСТИРОВАНИЕ РАЗРАБОТАННОГО ПРИЛОЖЕНИЯ

Поскольку приложение разрабатывалось в качестве выпускной квалификационной работы, было проведено только альфа-тестирование разработчиком. Для предотвращения ошибок в коде были проведены модульные тесты.

4.1 Тестирование корректности ввода

Для исключения пользовательских ошибок ввода информации в поле «Что искать» оно было настроено таким образом, что ввести меньше двух символов в него нельзя. Пример ввода показан на рисунках 8 и 9.

Рисунок 8 – Пример правильного ввода данных

Рисунок 9 – Ввод одного символа в поле ввода

4.2 Тестирование сбора данных

Тестирование работы проводится на основе выбора разных фильтров поиска. Например, будем искать «Генетический код» за 2023-2024г. и выведем эти данные в Excel. Количество результатов на eLIBRARY представлено на рисунке 10.

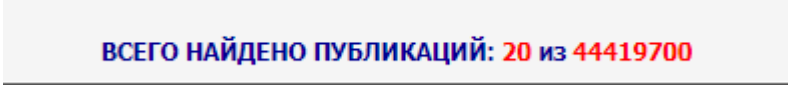

Рисунок 10 – Результаты на eLIBRARY

Результаты работы разрабатываемого приложения представлены на рисунке 11.

	ID	EDN	Type	Title	Author	Data	Lang	Year	Pages	Annotation
2	50473754	CXNOFX	статья в ж	ПРИОРИТ	ШАТИЛОІ	Финансов	русский			В условиях внешнеполитического давления сохранение и укр
3	50410199	XEJSPC	статья в сб	О МИНИМ	ЗОЛЯН СУ	Институт	русский	2023	42-56	Успехи языкознания XX века были достигнуты за счет разграни
4	50242465	MGWAZQ	статья в сб	КЛОНИРС	ДРОБЫШ	ФГБОУ ВС	русский	2023	65-68	В статье рассматривается роль клонирование собак в жизни че
5	50342494	XXYNIY	статья в сб	ТРАНСФО	ПАХОМОВА В. К.1		русский	2023	266-268	В материале исследуется трансформация имиджа Испании к
6	50349956	YOKTCT	статья в ж	СИЦИЛИЙ	БОТОВА В.И.		русский		48-71	Статья содержит результаты исследования родственной связи
7	50733700	QNTBGR	статья в ж	«ЦИРКУМ	МОНАСТЫ	Санкт-Пет	русский			«Циркумбалтийское пространство» презентировано в работе
8	50759644	UCKHXJ	статья в ж	ЗАКОНОМ	КУБЕЦКАЯ	Централь	русский		59-70	В статье рассматривается сходство закономерностей формирс
9	52295102	VYEMCX	свидетел	«ПРОГРАІ	РУСАКОВ	АЛЕКСЕЙ	МИХАЙЛС	2023		Программное обеспечение представлено в виде законченног
10	50450024	LBYBYP	статья в ж	СТРУКТУР	СИЛИН Я.	Уральски	русский		26-49	Методологическая база исследования процессов трансформа
11	50342345	WGCZUO	статья в сб	ОХРАНЯЕІ	АГАПОВ А	ЮУрГГПУ	русский	2023	13-16	Генетический код геобренда включает в себя природные ресу
12	50342354	ASWBNV	статья в сб	ДОКУДРА	ЛИНЬКОВ	ЧелГУ	русский	2023	34-35	Генетический код геобренда включает в себя реальную истор
13	50170042	MHZCVB	статья в ж	К ИСТОРИ	ДАВЫДОІ	Институт	русский		67-69	В статье рассматривается многообразие «генетического» кода
14	53703149	IJKPUQ	статья в ж	ПРИМЕНЕ	СУЛЕЙМАНОВА Э. И		русский			Аминокисло́ты представляют собой органические соединения
15	50516347	LRENGQ	статья в о	ЦВЕТ В СЕ	ЦАЦЕНКС	Кубански	русский			В основу представленной базы данных легла философия книги
16	50211271	LTAHHW	статья в сб	ХИМИЧЕС	МИХЕЕВА	Ульяновс	русский	2023	109-115	Аминокислоты - это важная группа органических соединений.
17	50494117	FYFDMM	статья в ж	ПРОГРАМ	ДАНИЛОІ	Белорусс	русский		63-71	В статье рассматривается процесс социальной эволюции как п
18	50438485	MVQEPR	статья в ж	АДАПТИР	ЛЕВАКОВА О. В.		русский		42-49	Цель исследования - выявить и представить наиболее ценные
19	50134017	DHXNCA	свидетел	ПРОГРАМ	МУКБА САБИНА АНАТОЛЬЕВН			2023		Программа предназначена для автоматического создания вхо
20	50758094	KOZFFA	статья в ж	ВАРИАЦИ	СОФРОНС	Федераль	русский		3-13	В плотной сети городских дорог влияние светофоров на сосед
21	52657374	JQGKLX	статья в ж	МОЛЕКУЛ	БАЗАНОВ Т.А.		русский			Актуальность. При исследовании генетического разнообразия

Рисунок 11 – Результаты работы приложения в программе Excel

Как видно из рисунка, приложение нашло и обработало все публикации, выведенные сайтом eLIBRARY по нашему запросу.

Для второго теста будем собирать данные по запросу «Квантовая суперпозиция», Тип публикации: статьи в журналах, материалы конференций, диссертации, отчеты. В параметрах укажем: искать с учетом морфологии, искать в публикациях, имеющих полный текст на eLIBRARY.ru. Количество результатов на eLIBRARY представлено на рисунке 12.

ВСЕГО НАЙДЕНО ПУБЛИКАЦИЙ: 11 из 44419700

Рисунок 12 – Результаты на eLIBRARY

Результаты работы программы по второму запросу представлены на рисунке 13. Для полной проверки соответствия сбора данных перейдем на страницу публикации. Страницы публикаций представлены на рисунках 14, 15, 16.

Информация, которую мы получили в файле, и информация со страницы публикации полностью совпадают. Следовательно, приложение работает корректно.

```
%0 статья в журнале - научная статья
%A ЛУНИН Н.В.
%T НЕЕВКЛИДОВ ПРИНЦИП СУПЕРПОЗИЦИИ И ЗАКОНЫ СОХРАНЕНИЯ В ФЕЙНМАНОВСКОЙ СХЕМЕ КВАНТОВОЙ МЕХ
%D 2002
%P 120-141
%I Институт прикладной физики РАН, 603950, Нижний Новгород, Ульянова, 46
%N Марковские свойства пропагаторов и принцип суперпозиции в квантовой механике с теоретик
ных согласно евклидову принципу суперпозиции, выполняются не всюду, тогда как неевклидов к

%0 статья в журнале - аннотация
%A ЯКОВЛЕВ В.А.
%T 2002.03.012. МЕРРИАМ П. ОБ ОТНОСИТЕЛЬНОСТИ КВАНТОВЫХ СУПЕРПОЗИЦИЙ. MERRIAM P. ON THE RE
%D 2002
%P 63-65

%0 статья в журнале - научная статья
%A АРУСТАМЯН М.Г.
%T ВОЗБУЖДЕНИЕ АТОМА ВОДОРОДА БИХРОМАТИЧЕСКИМ ИЗЛУЧЕНИЕМ С ОТНОШЕНИЕМ ЧАСТОТ 1:2 В ПРИСУТС
%D 2006
%P 20-28
%I Московский физико-технический институт
%N В рамках теории возмущений рассчитана вероятность возбуждения атома водорода из основно
```

Рисунок 13 – Часть результатов работы приложения

НЕЕВКЛИДОВ ПРИНЦИП СУПЕРПОЗИЦИИ И ЗАКОНЫ СОХРАНЕНИЯ В ФЕЙНМАНОВСКОЙ СХЕМЕ КВАНТОВОЙ МЕХАНИКИ

ЛУНИН Н.В.[1]

[1] Институт прикладной физики РАН, 603950, Нижний Новгород, Ульянова, 46

Тип: статья в журнале - научная статья Язык: русский

Номер: 1 Год: 2002 Страницы: 120-141

УДК: 512.54; 530.145

ЖУРНАЛ:

ВЕСТНИК НИЖЕГОРОДСКОГО УНИВЕРСИТЕТА ИМ. Н.И. ЛОБАЧЕВСКОГО. СЕРИЯ: МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ И ОПТИМАЛЬНОЕ УПРАВЛЕНИЕ
Учредители: Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского
ISSN: 1811-5977

АННОТАЦИЯ:

Марковские свойства пропагаторов и принцип суперпозиции в квантовой механике с теоретико-групповой точки зрения состоят в противоречии, согласно теореме Нетер, это может привести к нарушению законов сохранения. Для стационарного уравнения Шредингера получен полный набор наблюдаемых, их - четыре, из которых три независимы. Теоретико-групповые требования к полному пропагатору получены из требования выполнения законов сохранения. Парциальные пропагаторы принадлежат той же группе, что и полный, пространством их логарифмов является плоскость Лобачевского. Получена коммутативная бинарная операция на неабелевых группах Ли. Она является неевклидовым принципом суперпозиции, учитывающим некоммутативность пропагаторов и локально переходящим в евклидов принцип суперпозиции. Полный пропагатор строится из парциальных, он является мультипликативным интегралом по путям, для него найдены мера и вариационный принцип. В эксперименте с двумя щелями на границе двух сред показано, что законы сохранения для наблюдаемых, вычисленных согласно евклидову принципу суперпозиции, выполняются не всюду, тогда как неевклидов к их выполнению приводит везде.

Рисунок 14 – Страница первой публикации

2002.03.012. МЕРРИАМ П. ОБ ОТНОСИТЕЛЬНОСТИ КВАНТОВЫХ СУПЕРПОЗИЦИЙ. MERRIAM P. ON THE RELATIVITY OF QUANTUM SUPERPOSITIONS//HTTP://WWW.META.INH.EDU. (METAPHYSICAL REV.: ESSAYS ON THE FOUNDATIONS OF PHYSICS. - 1997. - VOL. 4, N 5)

ПОЛИЩУК С.Е., ЯКОВЛЕВ В.А.

Тип: статья в журнале - аннотация Язык: русский

Номер: 3 Год: 2002 Страницы: 63-65

ЖУРНАЛ:

СОЦИАЛЬНЫЕ И ГУМАНИТАРНЫЕ НАУКИ. ОТЕЧЕСТВЕННАЯ И ЗАРУБЕЖНАЯ ЛИТЕРАТУРА. СЕРИЯ 3: ФИЛОСОФИЯ. РЕФЕРАТИВНЫЙ ЖУРНАЛ
Учредители: Институт научной информации по общественным наукам РАН
ISSN: 2219-8555

КЛЮЧЕВЫЕ СЛОВА:

ИЗМЕРЕНИЕ, НАБЛЮДАЕМОСТЬ, ОТНОСИТЕЛЬНОСТЬ, ФИЛОСОФСКИЕ ПРОБЛЕМЫ ФИЗИКИ КВАНТОВОЙ

Рисунок 15 – Страница второй публикации

ВОЗБУЖДЕНИЕ АТОМА ВОДОРОДА БИХРОМАТИЧЕСКИМ ИЗЛУЧЕНИЕМ С ОТНОШЕНИЕМ ЧАСТОТ 1:2 В ПРИСУТСТВИИ ЭЛЕКТРИЧЕСКОГО ПОЛЯ

АРУСТАМЯН М.Г.[1], АСТАПЕНКО В.А.[1]

[1] Московский физико-технический институт

Тип: статья в журнале - научная статья Язык: русский

Том: 49 Номер: 8 Год: 2006 Страницы: 20-28

УДК: 621.373

ЖУРНАЛ:

ИЗВЕСТИЯ ВУЗОВ. ФИЗИКА
Учредители: Национальный исследовательский Томский государственный университет
ISSN: 0021-3411

АННОТАЦИЯ:

В рамках теории возмущений рассчитана вероятность возбуждения атома водорода из основного в первое возбужденное состояние под действием бихроматического лазерного излучения с отношением частот 1:2 в присутствии электрического поля с учетом квантовой интерференции двух каналов процесса. В рассматриваемом случае когерентная суперпозиция *2s*- и 2*p*-состояний, необходимая для возникновения квантовой интерференции, формируется под действием внешнего электрического поля. Проанализирована зависимость вероятности возбуждения от относительной фазы монохроматических компонент излучения (когерентный фазовый контроль) в двух предельных случаях: сильного электрического поля, когда штарковское расщепление верхнего уровня энергии существенно превышает спин-орбитальное, и в противоположном пределе слабого поля.

Рисунок 16 – Страница третьей публикации

ЗАКЛЮЧЕНИЕ

В результате выполнения выпускной квалификационной работы удалось достичь решения всех поставленных задач.

Были изучены основные возможности поисковой системы в сервисе eLIBRARY, включая виды запросов и ограничения. В процессе исследования было выявлено, что eLIBRARY обладает широким набором функций для поиска научных публикаций. Пользователям предоставляется возможность использовать различные параметры и фильтры при формулировке запросов, такие как ключевые слова, авторы, названия журналов, год публикации и другие.

Проведенный сравнительный анализ существующих программных решений позволил выявить, что большинство приложений реализующих похожий функционал не могут работать с eLIBRARY, так как большинство из них перестали поддерживаться разработчиками или были предназначены для других сервисов.

На основе полученных данных был осуществлен выбор средств реализации, и разработано приложение, которое позволяет автоматизировать процесс сбора данных из библиографической и реферативной базы eLIBRARY. В ходе выполнения выпускной квалификационной работы было разработано бесплатное приложение, позволяющее указать большое количество фильтров для составления подробного запроса на сайте eLIBRARY, а также собрать информацию об интересующих публикациях и вывести данные в различные форматы для библиометрического анализа.

Также разработанное программное обеспечение было протестировано на корректность ввода данных для поиска и точность сбора данных из системы и их дальнейшее преобразование в различные форматы. Результаты тестирования подтвердили работоспособность и надежность приложения.

Однако, в ходе разработки столкнулись с некоторыми ограничениями. В частности, не удалось реализовать функцию предварительного просмотра публикаций и возможность дополнительной обработки данных, так как разработка данной функции требует большего времени, чем отведено на выполнение

выпускной квалификационной работы, поэтому она будет разработана в дальнейшем.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Низомутдинов, Б.А. Автоматизированный сбор данных для наукометрического анализа / Б.А. Низомутдинов, А.С. Тропников // Научный сервис в сети Интернет: труды XXI Всероссийской научной конференции (23-28 сентября 2019 г., г. Новороссийск). – ИПМ им. М.В.Келдыша. – 2019. – С. 523-531. [Электронный ресурс]. URL: http://keldysh.ru/abrau/2019/theses/76.pdf. (Дата обращения: 27.02.2023).

2 Меньшиков, Я.С. Преимущества автоматического сбора данных в сети интернет над ручным сбором данных / Я.С. Меньшиков // Universum: технические науки. – 2022. №10-1 (103). [Электронный ресурс]. URL: https://cyberleninka.ru/article/n/preimuschestva-avtomaticheskogo-sbora-dannyh-v-seti-internet-nad-ruchnym-sborom-dannyh (Дата обращения: 3.03.2023).

3 Горный, Е. Развитие электронных библиотек: мировой и российский опыт, проблемы, перспективы / Е. Горный, К. Вигурский – 2002. [Электронный ресурс]. URL: https://ifap.ru/library/book004.pdf. (Дата обращения: 3.03.2023).

4 Земсков, А.И. Электронные библиотеки и развитие Информационного общества в России // Электронные библиотеки. – Вып. 4 – № 6 (1). [Электронный ресурс]. URL: https://rdl-journal.ru/article/view/194. (Дата обращения: 27.02.2023).

5 Рожков, В.Ю. Электронное правительство – перспективы внедрения новых информационно-коммуникационных сервисов / В.Ю. Рожков // Вестник Московского университета. – 2010. – №4. [Электронный ресурс]. URL: https://cyberleninka.ru/article/n/elektronnoe-pravitelstvo-perspektivy-vnedreniya-novyh-informatsionno-kommunikatsionnyh-servisov (Дата обращения: 27.02.2023).

6 Elibrary [Электронный ресурс]. Elibrary: [сайт]. URL: https://elibrary.ru/. (Дата обращения: 19.05.2023).

7 Xs-sebzer [Электронный ресурс]. GitHub: [сайт]. URL: https://github.com/p1m-ortho/xs-sebzer. (Дата обращения: 27.02.2023).

8 Web of Science [Электронный ресурс]. Web of Science: [сайт]. URL: https://www.webofscience.com/wos/woscc/basic-search. (Дата обращения: 27.02.2023).

9 Scopus [Электронный ресурс]. Scopus: [сайт]. URL: https://www.scopus.com/search. (Дата обращения: 27.02.2023).

10 Tampermonkey [Электронный ресурс]. Tampermonkey: [сайт]. URL: https://www.tampermonkey.net/. (Дата обращения: 27.02.2023).

11 GitHub [Электронный ресурс]. GitHub: [сайт]. URL: https://github.com/. (Дата обращения: 19.05.2023).

12 Stack Overflow [Электронный ресурс]. Stack Overflow: [сайт]. URL: https://stackoverflow.com/. (Дата обращения: 19.05.2023).

13 Митчелл, Р. Скрапинг веб-сайтов с помощью Python: сбор данных из современного интернета: практическое руководство / Р. Митчелл. – Москва: ДМК Пресс, 2016. – 280 с.

14 Митчелл, Р. Современный скрапинг веб-сайтов с помощью Python: практическое руководство / Р. Митчелл. – 2-е межд. изд. – Санкт-Петербург: Питер, 2021. – 336 с.

15 Прохоренок, Н.А. Python 3 и PyQt 5. Разработка приложений: практическое руководство / Н.А. Прохоренок, В.А. Дронов – Санкт-Петербург: БХВ-Петербург, 2016. – 832 с.

ПРИЛОЖЕНИЕ

Исходный код программы

```
import json
import os
import re
import random
import sys
from urllib.parse import urlparse
import pandas as pd
from bs4 import Tag, BeautifulSoup
from selenium import webdriver
from selenium_stealth import stealth
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.common.by import By
from selenium.webdriver.support.select import Select
import time
import tkinter as tk
import tkinter.font as tkfont
import datetime
from tkinter import ttk, messagebox, filedialog


# Функция отправки формы запроса(передачи данных в программу)
def submit_form():
    keyword = entry.get()
    checkbox1_state = checkbox1_var.get()
    checkbox2_state = checkbox2_var.get()
    checkbox3_state = checkbox3_var.get()
    checkbox4_state = checkbox4_var.get()
    checkbox5_state = checkbox5_var.get()
    checkbox6_state = checkbox6_var.get()
    checkbox7_state = checkbox7_var.get()
    checkbox8_state = checkbox8_var.get()
    checkbox9_state = checkbox9_var.get()
    checkbox10_state = checkbox10_var.get()
    checkbox11_state = checkbox11_var.get()
    checkbox12_state = checkbox12_var.get()
    checkbox13_state = checkbox13_var.get()
    checkbox14_state = checkbox14_var.get()
    checkbox15_state = checkbox15_var.get()
    checkbox16_state = checkbox16_var.get()
    combo_state = combo.get()
    combo2_state = combo2.get()
    combo3_state = combo3.get()
    combo4_state = combo4.get()
    combo5_state = combo5.get()

    if len(keyword) >= 2:
        entry.config(bg="white")
        error_label.config(text="")
        print("Форма отправлена!")
    else:
```

```
        error_label.config(text="Введите 2 и более символа", fg="red")  #
Отображение сообщения об ошибке
        entry.config(bg='red')
        return

    options = webdriver.ChromeOptions()
    options.add_argument("start-maximized")

    # options.add_argument("--headless")

    options.add_experimental_option("excludeSwitches", ["enable-automation"])
    options.add_experimental_option('useAutomationExtension', False)

    # Функция указания папки с ChromeDriver
    def resource_path(relative_path):
        try:
            base_path = sys._MEIPASS
        except Exception:
            base_path = os.path.dirname(__file__)
        return os.path.join(base_path, relative_path)

    driver =
webdriver.Chrome(service=Service(resource_path('./driver/chromedriver.exe')),
options=options)

    stealth(driver,
            languages=["en-US", "en"],
            vendor="Google Inc.",
            platform="Win32",
            webgl_vendor="Intel Inc.",
            renderer="Intel Iris OpenGL Engine",
            fix_hairline=True,
            )

    # Очистка/создание JSON-файла
    empty_data = {}
    with open('data.json', 'w') as file:
        json.dump(empty_data, file)

    # Функция отправки сообщения о завершении работы программы
    def open_message_box():
        messagebox.showinfo("Завершение программы", "Программа завершена!")

    # Функция сбора ссылок на публикации
    def get_urls() -> list:
        urls = []
        url_pattern = "https://elibrary.ru/item.asp?id="
        block = driver.find_element(By.ID, "restab")
        pubs = block.find_element(By.TAG_NAME, "tbody").find_elements(By.TAG_NAME,
"tr")
        for pub in pubs:
            try:
                id = pub.get_attribute("id")
                id = ''.join([i for i in id if i.isdigit()])
                urls.append(url_pattern + id)
            except Exception as err:
                continue
```

```
        del urls[0]
        return urls

    # Основная функция программы. Сбор данных со страницы публикации
    def parser(i):
        title = driver.find_element(By.CLASS_NAME, "bigtext").text

        content = driver.page_source
        soup = BeautifulSoup(content, 'lxml')
        owner = ""
        p_ind = ""
        tr = soup.body.table.tbody.tr.td.table.tbody.tr
        td = ""
        for t in tr.children:
            if isinstance(t, Tag):
                td = t
        tbody = td.table.tbody
        for t in tbody.children:
            if isinstance(t, Tag):
                tr = t
        td = tr.td

        id, edn = td.table.tbody.tr.find_all('a')[:2]
        id, edn = id.text, edn.text

        t = 0
        table = ""
        div = ""
        for tag in td.contents:
            if isinstance(tag, Tag):
                t += 1
                if t == 2:
                    table = tag
                elif t == 3:
                    div = tag

        try:
            p_ind = div.tbody.tr.find('span', class_='help1 pointer')
            owner = div.tbody.tr.find('span', class_='help pointer')
            if p_ind is None:
                pass
            else:
                p_ind = p_ind.text + str(p_ind.next_sibling)
                p_ind = p_ind.replace("<br/>", "")
                p_ind = p_ind.replace("\n\xE2\x96\xBCПоказать полностью", "")
            if owner is None:
                owner = ''.join([i.text for i in
div.table.tbody.tr.find_all('div')])
                owner = owner.replace("\u00A0", " ")
            else:
                owner = owner.text
                owner = owner.replace("\u00A0", " ")
                owner = re.sub(r'\d', '', owner)
        except Exception as err:
            print(err, 71)

        p_data = driver.find_elements(By.XPATH,
```

```
"/html/body/table/tbody/tr/td/table[1]/tbody/tr/td[2]/table/tbody/tr[2]/td[1]/div/
table")
        datas = ""
        for data in p_data:
            temp = data.find_element(By.TAG_NAME,
"tbody").find_elements(By.TAG_NAME, "tr")
            for d in temp:
                datas += d.text
        type_r = re.search(r"Тип:\s*(.*?)(?=[А-Я]|$)", datas)
        type_p = type_r.group(1).strip() if type_r else None

        lang_r = re.search(r"Язык:\s*(.*?)(?=[А-Я]|$)", datas)
        lang_p = lang_r.group(1).strip() if lang_r else None

        year_r = re.search(r"(?:Год(?:\sиздания)?:|Год
публикации:)\s*(\d{4})(?=[А-Я]|$)", datas)
        year_p = year_r.group(1).strip() if year_r else None

        page_r = re.search(r"Страницы:\s*(.*?)(?=[А-Я]|$)", datas)
        page_p = page_r.group(1).strip() if page_r else None

        p_data = driver.find_elements(By.XPATH,

"/html/body/table/tbody/tr/td/table[1]/tbody/tr/td[2]/table/tbody/tr[2]/td[1]/div/
table")
        ann_tr = ""
        kw_tr = ""
        for an in p_data:
            ann_title = an.find_element(By.TAG_NAME,
"tbody").find_element(By.TAG_NAME, "tr").find_element(By.TAG_NAME,

"td").text
            if ann_title == "АННОТАЦИЯ:":
                ann_tr = an.text
            if ann_title == "КЛЮЧЕВЫЕ СЛОВА:":
                kw_tr = an.text
        ann_p = ann_tr[ann_tr.find("АННОТАЦИЯ: \n  ") + len("АННОТАЦИЯ: \n  "):]
        ann_p = ann_p.replace("<br/>", "")
        ann_p = ann_p.replace("\n▼Показать полностью", "")
        kw_p = kw_tr[ann_tr.find("КЛЮЧЕВЫЕ СЛОВА: \n  ") + len("КЛЮЧЕВЫЕ СЛОВА: \n
"):]
        kw_p = [keyw.strip() for keyw in kw_p.split(",")]

        data_dict = {
            "ID": id,
            "EDN": edn,
            "Type": type_p,
            "Title": title,
            "Author": owner,
            "Data": p_ind,
            "Lang": lang_p,
            "Year": year_p,
            "Pages": page_p,
            "Keywords": kw_p,
            "Annotation": ann_p
        }
```

```
        data_dict = {k: v for k, v in data_dict.items() if v not in ("", None,
"Null", [""])}

        data = json.load(open('data.json', 'r', encoding='UTF-8'))
        data[i] = data_dict

        # Запись в файл
        with open("data.json", "w", encoding="utf-8") as file:
            json.dump(data, file, indent=4, ensure_ascii=False)

    url = "https://elibrary.ru/querybox.asp?scope=newquery"
    driver.get(url)
    time.sleep(3)
    curr = driver.current_url
    if ("https://elibrary.ru/page_captcha.asp" in curr) and (curr != url):
        time.sleep(90)

    # Передача данных из программы на форму поиска сайта eLIBRARY
    driver.find_element(By.NAME, "where_name").click()
    driver.find_element(By.NAME, "where_abstract").click()
    driver.find_element(By.NAME, "where_keywords").click()

    if checkbox6_state:
        driver.find_element(By.NAME, "where_fulltext").click()  # on
        driver.find_element(By.NAME, "where_name").click()
        driver.find_element(By.NAME, "where_abstract").click()
        driver.find_element(By.NAME, "where_keywords").click()
        driver.find_element(By.NAME, "where_affiliation").click()
        driver.find_element(By.NAME, "where_references").click()

    if checkbox1_state:
        driver.find_element(By.NAME, "where_name").click()

    if checkbox2_state:
        driver.find_element(By.NAME, "where_abstract").click()

    if checkbox3_state:
        driver.find_element(By.NAME, "where_keywords").click()

    if checkbox4_state:
        driver.find_element(By.NAME, "where_affiliation").click()

    if checkbox5_state:
        driver.find_element(By.NAME, "where_references").click()

    if not checkbox7_state:
        driver.find_element(By.NAME, "type_article").click()

    if not checkbox8_state:
        driver.find_element(By.NAME, "type_book").click()

    if not checkbox9_state:
        driver.find_element(By.NAME, "type_conf").click()

    if not checkbox10_state:
        driver.find_element(By.NAME, "type_preprint").click()
```

```
if not checkbox11_state:
    driver.find_element(By.NAME, "type_disser").click()

if not checkbox12_state:
    driver.find_element(By.NAME, "type_report").click()

if not checkbox13_state:
    driver.find_element(By.NAME, "type_patent").click()

if not checkbox14_state:
    driver.find_element(By.NAME, "search_morph").click()

if checkbox15_state:
    driver.find_element(By.NAME, "search_freetext").click()

if checkbox16_state:
    driver.find_element(By.NAME, "search_fulltext").click()

key = keyword
driver.find_element(By.CLASS_NAME, "inputr").send_keys(key)

year_f = combo_state
year_t = combo2_state
ddf = Select(driver.find_element(By.NAME, "begin_year"))
ddf.select_by_visible_text(year_f)
ddt = Select(driver.find_element(By.NAME, "end_year"))
ddt.select_by_visible_text(year_t)

pst = combo3_state
ddp = Select(driver.find_element(By.NAME, "issues"))
ddp.select_by_visible_text(pst)
sort_p = combo4_state
dds = Select(driver.find_element(By.NAME, "orderby"))
dds.select_by_visible_text(sort_p)
order_p = combo5_state
ddo = Select(driver.find_element(By.NAME, "order"))
ddo.select_by_visible_text(order_p)

driver.find_element(By.LINK_TEXT, "Поиск").click()
time.sleep(5)

# Сбор ссылок на публикации
urls = []
urls.extend(get_urls())
curr = driver.current_url
if ("https://elibrary.ru/page_captcha.asp" in curr) and (curr != url):
    time.sleep(90)
try:
    end = driver.find_element(By.LINK_TEXT, "В конец")
    end.click()
    url = driver.current_url
    parsed_url = urlparse(url)
    query_string = parsed_url.query
    page_number = None
    for param in query_string.split('&'):
        param_name, param_value = param.split('=')
        if param_name == 'pagenum':
```

```
                page_number = int(param_value)
        if page_number >= 100:
            page_number = 99

        url = "https://www.elibrary.ru/query_results.asp"
        driver.get(url)
        for i in range(2, page_number + 1):
            driver.get(url + f'?pagenum={i}')
            curr = driver.current_url
            if ("https://elibrary.ru/page_captcha.asp" in curr) and (curr != url):
                time.sleep(90)
            urls.extend(get_urls())
    except Exception as err:
        pass

    print(f'Найдено {len(urls)} ссылок для обработки')

    # Сбор данных из публикаций из списка
    for i, url in enumerate(urls, 1):
        driver.get(url)
        curr = driver.current_url
        if ("https://elibrary.ru/page_captcha.asp" in curr) and (curr != url):
            time.sleep(90)
        print(f'Сейчас обрабатывается {i} публикация по адресу: {url}')
        if i % 7 == 0:
            second = random.randrange(5, 15)
            time.sleep(second)
        parser(i)

    open_message_box()
    driver.quit()


# Функции конвертации JSON-файла в .BibTex
def generate_bibtex_key(author, year, title):
    author_lastname = author.split()[0]
    first_word = title.split()[0]
    bibtex_key = f'{author_lastname}{year}{first_word}'

    return bibtex_key


def json_to_bibtex(json_file_path, bibtex_file_path):
    with open(json_file_path, 'r', encoding='utf-8') as json_file:
        data = json.load(json_file)

    bibtex_entries = []

    for item_id, item in data.items():
        bibtex_type = 'article'
        bibtex_key = generate_bibtex_key(item.get('Author', ''), item.get('Year',
'', item.get('Title', ''))

        bibtex_entry = f'@{bibtex_type}{{{bibtex_key},\n'
        bibtex_entry += f'    title = {{{item.get("Title", "")}}},\n'
        if "Author" in item:
```

```
            bibtex_entry += f'    author = {{{item["Author"]}}},\n'
        if "Year" in item:
            bibtex_entry += f'    year = {{{item["Year"]}}},\n'
        if "Pages" in item:
            bibtex_entry += f'    pages = {{{item["Pages"]}}},\n'
        if "Data" in item:
            bibtex_entry += f'    institution = {{{item["Data"]}}},\n'
        if "Annotation" in item:
            bibtex_entry += f'    note = {{{item["Annotation"]}}},\n'

        keywords = item.get('Keywords', [])
        keyword_string = ', '.join(keywords)

        if keyword_string:
            bibtex_entry += f'    keywords = {{{keyword_string}}},\n'

        bibtex_entry += '}\n'

        bibtex_entries.append(bibtex_entry)

    bibtex_string = "\n".join(bibtex_entries)

    with open(bibtex_file_path, 'w', encoding='utf-8') as bibtex_file:
        bibtex_file.write(bibtex_string)

    print(f'BibTeX file "{bibtex_file_path}" created successfully.')


def convert_json_to_bibtex():
    json_file_path = filedialog.askopenfilename(filetypes=[('JSON Files',
'*.json')])
    bibtex_file_path = filedialog.asksaveasfilename(defaultextension='.bib',
filetypes=[('Bibtex Files', '*.bib')])
    if bibtex_file_path:
        json_to_bibtex(json_file_path, bibtex_file_path)


# Функции конвертации JSON-файла в .ris
def json_to_ris(json_file_path, ris_file_path):
    with open(json_file_path, 'r', encoding='utf-8') as json_file:
        data = json.load(json_file)

    ris_entries = []

    for item in data.values():
        ris_entry = ""

        if "Type" in item:
            ris_entry += "TY  - " + item["Type"] + "\n"

        if "Author" in item:
            ris_entry += "AU  - " + item["Author"] + "\n"

        if "Title" in item:
            ris_entry += "TI  - " + item["Title"] + "\n"

        if "Year" in item:
```

```
            ris_entry += "PY  - " + item["Year"] + "\n"

        if "Pages" in item:
            pages = item["Pages"]
            if "-" in pages:
                ris_entry += "SP  - " + pages.split("-")[0] + "\n"
                ris_entry += "EP  - " + pages.split("-")[1] + "\n"
            else:
                ris_entry += "SP  - " + pages + "\n"

        if "Data" in item:
            ris_entry += "PB  - " + item["Data"] + "\n"

        if "Annotation" in item:
            ris_entry += "N2  - " + item["Annotation"] + "\n"

        if "Keywords" in item:
            keywords = item["Keywords"]
            for keyword in keywords:
                ris_entry += "KW  - " + keyword + "\n"

        ris_entries.append(ris_entry)

    ris_string = "\n".join(ris_entries)

    with open(ris_file_path, 'w', encoding='utf-8') as ris_file:
        ris_file.write(ris_string)

    print(f'RIS file "{ris_file_path}" created successfully.')


def convert_json_to_ris():
    json_file_path = filedialog.askopenfilename(filetypes=[('JSON Files',
'*.json')])
    ris_file_path = filedialog.asksaveasfilename(defaultextension='.ris',
filetypes=[('RIS Files', '*.ris')])
    if ris_file_path:
        json_to_ris(json_file_path, ris_file_path)


# Функции конвертации JSON-файла в .enw
def json_to_enw(json_file_path, enw_file_path):
    with open(json_file_path, 'r', encoding='utf-8') as json_file:
        data = json.load(json_file)

    enw_entries = []

    for item in data.values():
        enw_entry = ''

        if "Type" in item:
            enw_entry += f'%0 {item["Type"]}\n'

        if "Author" in item:
            enw_entry += f'%A {item["Author"]}\n'

        if "Title" in item:
```

```
            enw_entry += f'%T {item["Title"]}\n'

        if "Year" in item:
            enw_entry += f'%D {item["Year"]}\n'

        if "Pages" in item:
            enw_entry += f'%P {item["Pages"]}\n'

        if "Data" in item:
            enw_entry += f'%I {item["Data"]}\n'

        if "Annotation" in item:
            enw_entry += f'%N {item["Annotation"]}\n'

        if "Keywords" in item:
            keywords = item["Keywords"]
            for keyword in keywords:
                enw_entry += f'%K {keyword}\n'

        enw_entries.append(enw_entry)

    enw_string = "\n".join(enw_entries)

    with open(enw_file_path, 'w', encoding='utf-8') as enw_file:
        enw_file.write(enw_string)

    print(f'ENW file "{enw_file_path}" created successfully.')


def convert_json_to_enw():
    json_file_path = filedialog.askopenfilename(filetypes=[('JSON Files',
'*.json')])
    enw_file_path = filedialog.asksaveasfilename(defaultextension='.enw',
filetypes=[('ENW Files', '*.enw')])
    if enw_file_path:
        json_to_enw(json_file_path, enw_file_path)


# Функции конвертации JSON-файла в .xlsx
def json_to_excel(json_file_path, excel_file_path):
    with open(json_file_path, 'r', encoding='utf-8') as json_file:
        data = json.load(json_file)

    rows = []
    columns = ['ID', 'EDN', 'Type', 'Title', 'Author', 'Data', 'Lang', 'Year',
'Pages', 'Keywords', 'Annotation']

    for item_id, item in data.items():
        row = [
            item.get('ID', ''),
            item.get('EDN', ''),
            item.get('Type', ''),
            item.get('Title', ''),
            item.get('Author', ''),
            item.get('Data', ''),
            item.get('Lang', ''),
            item.get('Year', ''),
```

```
            item.get('Pages', ''),
            ', '.join(item.get('Keywords', [])),
            item.get('Annotation', '')
        ]
        rows.append(row)

    df = pd.DataFrame(rows, columns=columns)

    df.to_excel(excel_file_path, index=False)

    print(f'Excel file "{excel_file_path}" created successfully.')


def convert_json_to_excel():
    json_file_path = filedialog.askopenfilename(filetypes=[('JSON Files',
'*.json')])
    excel_file_path = filedialog.asksaveasfilename(defaultextension='.xlsx',
filetypes=[('Excel Files', '*.xlsx')])
    if excel_file_path:
        json_to_excel(json_file_path, excel_file_path)


# Функция обработки кнопки "enter" в форме
def on_enter_pressed(event):
    submit_button.invoke()


# Функция для работы кнопки "в полном тексте публикации" в форме запроса
def update_checkboxes():
    if checkbox1_var.get():
        checkbox1_var.set(True)
        checkbox2_var.set(True)
        checkbox3_var.set(True)
        checkbox4_var.set(True)
        checkbox5_var.set(True)


# Функция для очистки формы запроса
def clear():
    entry.delete(0, tk.END)

    checkbox1_var.set(True)
    checkbox2_var.set(True)
    checkbox3_var.set(True)
    checkbox4_var.set(False)
    checkbox5_var.set(False)
    checkbox6_var.set(False)
    checkbox7_var.set(True)
    checkbox8_var.set(True)
    checkbox9_var.set(True)
    checkbox10_var.set(True)
    checkbox11_var.set(True)
    checkbox12_var.set(True)
    checkbox13_var.set(True)
    checkbox14_var.set(True)
    checkbox15_var.set(False)
    checkbox16_var.set(False)
```

```
    combo.set(years[0])
    combo2.set(years[0])
    combo3.set(values[0])
    combo4.set(sort[0])
    combo5.set(order[0])


# Интерфейс программы
window = tk.Tk()

# Заголовок окна
window.title("Сборщик данных из eLIBRARY")
window.resizable(width=False, height=False)

# шрифт
custom_font = tkfont.Font(family="Century Gothic", size=12)
custom_fontB = tkfont.Font(family="Century Gothic", size=12, weight="bold")

# Основной фрейм
frame0 = tk.Frame(window, borderwidth=2, relief=tk.RAISED)
frame0.pack(padx=10, pady=10)

# блок "Что искать"
frame = tk.Frame(frame0, borderwidth=2, relief=tk.RAISED)
frame.pack(padx=10, pady=5, anchor="w", fill="x", expand=True)

label = tk.Label(frame, text="Что искать:", font=custom_font, width=20)
label.pack(side='left')

entry = tk.Entry(frame)
entry.pack(fill="x", expand=True, padx=3)

error_label = tk.Label(frame, fg="red")
error_label.pack()

# блок "Где искать"
frame2 = tk.Frame(frame0, borderwidth=2, relief=tk.RAISED)
frame2.pack(padx=10, pady=5, anchor="w", fill="x", expand=True)

label2 = tk.Label(frame2, text="Где искать:", font=custom_font, width=20)
label2.pack(side='left')

checkbox1_var = tk.BooleanVar()
checkbox1_var.set(True)
checkbox1 = tk.Checkbutton(frame2, text="в названии публикации",
variable=checkbox1_var, font=custom_font)
checkbox1.pack(anchor="w")

checkbox2_var = tk.BooleanVar()
checkbox2_var.set(True)
checkbox2 = tk.Checkbutton(frame2, text="в аннотации", variable=checkbox2_var,
font=custom_font)
checkbox2.pack(anchor="w")

checkbox3_var = tk.BooleanVar()
checkbox3_var.set(True)
```

```
checkbox3 = tk.Checkbutton(frame2, text="в ключевых словах",
variable=checkbox3_var, font=custom_font)
checkbox3.pack(anchor="w")

checkbox4_var = tk.BooleanVar()
checkbox4 = tk.Checkbutton(frame2, text="в названии организаций авторов",
variable=checkbox4_var, font=custom_font)
checkbox4.pack(anchor="w")

checkbox5_var = tk.BooleanVar()
checkbox5 = tk.Checkbutton(frame2, text="в списках цитируемой литературы",
variable=checkbox5_var, font=custom_font)
checkbox5.pack(anchor="w")

checkbox6_var = tk.BooleanVar()
checkbox6 = tk.Checkbutton(frame2, text="в полном тексте публикации",
variable=checkbox6_var, command=update_checkboxes,
                           font=custom_font)
checkbox6.pack(anchor="w")

# блок "Тип публикации"
frame3 = tk.Frame(frame0, borderwidth=2, relief=tk.RAISED)
frame3.pack(padx=10, pady=5, anchor="w", fill="x", expand=True)

label3 = tk.Label(frame3, text="Тип публикации:", font=custom_font, width=20)
label3.pack(side='left')

checkbox7_var = tk.BooleanVar()
checkbox7_var.set(True)
checkbox7 = tk.Checkbutton(frame3, text="статьи в журналах",
variable=checkbox7_var, font=custom_font)
checkbox7.pack(anchor="w")

checkbox8_var = tk.BooleanVar()
checkbox8_var.set(True)
checkbox8 = tk.Checkbutton(frame3, text="книги", variable=checkbox8_var,
font=custom_font)
checkbox8.pack(anchor="w")

checkbox9_var = tk.BooleanVar()
checkbox9_var.set(True)
checkbox9 = tk.Checkbutton(frame3, text="материалы конференций",
variable=checkbox9_var, font=custom_font)
checkbox9.pack(anchor="w")

checkbox10_var = tk.BooleanVar()
checkbox10_var.set(True)
checkbox10 = tk.Checkbutton(frame3, text="депонированные рукописи",
variable=checkbox10_var, font=custom_font)
checkbox10.pack(anchor="w")

checkbox11_var = tk.BooleanVar()
checkbox11_var.set(True)
checkbox11 = tk.Checkbutton(frame3, text="диссертации", variable=checkbox11_var,
font=custom_font)
checkbox11.pack(anchor="w")
```

```
checkbox12_var = tk.BooleanVar()
checkbox12_var.set(True)
checkbox12 = tk.Checkbutton(frame3, text="отчеты", variable=checkbox12_var,
font=custom_font)
checkbox12.pack(anchor="w")

checkbox13_var = tk.BooleanVar()
checkbox13_var.set(True)
checkbox13 = tk.Checkbutton(frame3, text="патенты", variable=checkbox13_var,
font=custom_font)
checkbox13.pack(anchor="w")

# блок "параметры"
frame4 = tk.Frame(frame0, borderwidth=2, relief=tk.RAISED)
frame4.pack(padx=10, pady=5, anchor="w", fill="x", expand=True)

label4 = tk.Label(frame4, text="Параметры:", font=custom_font, width=20)
label4.pack(side='left')

checkbox14_var = tk.BooleanVar()
checkbox14_var.set(True)
checkbox14 = tk.Checkbutton(frame4, text="искать с учетом морфологии",
variable=checkbox14_var, font=custom_font)
checkbox14.pack(anchor="w")

checkbox15_var = tk.BooleanVar()
checkbox15 = tk.Checkbutton(frame4, text="искать похожий текст",
variable=checkbox15_var, font=custom_font)
checkbox15.pack(anchor="w")

checkbox16_var = tk.BooleanVar()
checkbox16 = tk.Checkbutton(frame4, text="искать в публикациях, имеющих полный
текст на eLIBRARY.ru",
                            variable=checkbox16_var, font=custom_font)
checkbox16.pack(anchor="w")

# блок "Годы и поступившие"
frame5 = tk.Frame(frame0, borderwidth=2, relief=tk.RAISED)
frame5.pack(padx=10, pady=5, anchor="w", fill="x", expand=True)

label5 = tk.Label(frame5, text="Годы публикации:", font=custom_font, width=20)
label5.pack(side='left')

current_year = datetime.date.today().year
years = [''] + [str(year) for year in range(2024, 1899, -1)]

combo = ttk.Combobox(frame5, values=years, width=8, state="readonly")
combo.pack(side='left')
combo.current(0)

label5 = tk.Label(frame5, text=" - ", font=custom_font)
label5.pack(side='left')

combo2 = ttk.Combobox(frame5, values=years, width=8, state="readonly")
combo2.pack(side='left')
combo2.current(0)
```

```
label6 = tk.Label(frame5, text="Поступившие:", font=custom_font, width=20)
label6.pack(side='left')

values = ["за все время", "за последний месяц", "за последние 3 месяца", "за
последние полгода", "за последний год"]
combo3 = ttk.Combobox(frame5, values=values, width=21, state="readonly")
combo3.pack(side='left')
combo3.current(0)

# блок "сортировка и порядок"
frame6 = tk.Frame(frame0, borderwidth=2, relief=tk.RAISED)
frame6.pack(padx=10, pady=5, anchor="w", fill="x", expand=True)

label7 = tk.Label(frame6, text="Сортировка:", font=custom_font, width=20)
label7.pack(side='left')

sort = ["по релевантности", "по дате выпуска", "по дате добавления", "по названию
статьи", "по названию журнала",
        "по числу цитирований"]
combo4 = ttk.Combobox(frame6, values=sort, width=21, state="readonly")
combo4.pack(side='left')
combo4.current(0)

label8 = tk.Label(frame6, text="Порядок:", font=custom_font, width=20)
label8.pack(side='left')

order = ["по убыванию", "по возрастанию"]
combo5 = ttk.Combobox(frame6, values=order, width=20, state="readonly")
combo5.pack(side='left')
combo5.current(0)

# блок "Кнопки"
frame7 = tk.Frame(frame0, borderwidth=2, relief=tk.RAISED)
frame7.pack(padx=10, pady=5, anchor="w", fill="x", expand=True)
# Кнопка очистки формы
clear_button = tk.Button(frame7, text="Очистить", command=clear, bg="gray",
fg="white", font=custom_fontB, width=10,
                         height=1)
clear_button.pack(padx=10, pady=2, side="left")
# Кнопка отправки формы
submit_button = tk.Button(frame7, text="Поиск", command=submit_form, bg="red",
fg="white", font=custom_fontB, width=10,
                          height=1)
submit_button.pack(padx=10, pady=2, side="right")

# блок "Продолжить"
frame8 = tk.Frame(frame0, borderwidth=2, relief=tk.RAISED)
frame8.pack(padx=10, pady=5, anchor="w", fill="x", expand=True)

b_bib = tk.Button(frame8, text="to Bibtex", command=convert_json_to_bibtex,
width=10, height=1)
b_bib.pack(padx=50, pady=2, side="left")
b_ris = tk.Button(frame8, text="to RIS", command=convert_json_to_ris, width=10,
height=1)
b_ris.pack(padx=50, pady=2, side="left")
b_enw = tk.Button(frame8, text="to ENW", command=convert_json_to_enw, width=10,
height=1)
```

```
b_enw.pack(padx=50, pady=2, side="left")
b_xlsx = tk.Button(frame8, text="to Excel", command=convert_json_to_excel,
width=10, height=1)
b_xlsx.pack(padx=50, pady=2, side="left")

entry.bind('<Return>', on_enter_pressed)

window.mainloop()
```