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 системного программирования

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой, д.ф.-м.н.,
профессор

_____ Л.Б. Соколинский

«___»_____ 2024 г.

Разработка программной системы для вычисления и анализа степеней примитивных корней для начальных диапазонов простых чисел

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 09.03.04.2024.308-564.ВКР

Научный руководитель,
профессор кафедры СП, д.ф.-м.н.,
доцент
_____ Р.Ж. Алеев

Автор работы,
студент группы КЭ-433
_____ Д.А. Манов

Ученый секретарь
(нормоконтролер)
_____ И.Д. Володченко
«___»_____ 2024 г.

Челябинск, 2024 г.

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 системного программирования

УТВЕРЖДАЮ
Зав. кафедрой СП
_____ Л.Б. Соколинский
29.01.2024 г.

ЗАДАНИЕ
на выполнение выпускной квалификационной работы бакалавра
студенту группы КЭ-433
Манову Дмитрию Александровичу,
обучающемуся по направлению
09.03.04 «Программная инженерия»

1. **Тема работы** (утверждена приказом ректора от 22.04.2024 г. № 764-13/12)
Разработка программной системы для вычисления и анализа степеней примитивных корней для начальных диапазонов простых чисел.
2. **Срок сдачи студентом законченной работы:** 03.06.2024 г.
3. **Исходные данные к работе**

3.1. Грицук Д. В. Компьютерная алгебра. // Брестский государственный университет А.С. Пушкина. Брест, 2018. – 270 с.

3.2. Чандрасекхаран К. Введение в аналитическую теорию чисел. // Berlin Heidelberg New York 1968. – 92 с.

3.3. Нестеренко Ю.В. Теория чисел. // М. Академия, 2008. – 292 с.

4. **Перечень подлежащих разработке вопросов**

4.1. Составление списка простых чисел из диапазона [2..P] для заданного P.

4.2. Разработка функции для вычисления примитивных корней по модулю простых чисел из диапазона [2..P].

4.3. Разработка функции, которая будет находить для каждого примитивного корня количества простых чисел из диапазона [2..P], для которых это число будет примитивным корнем.

4.4. Создать графическое приложение, в которое будут включены все указанные функции.

5. Дата выдачи задания: 29.01.2024 г.

Научный руководитель,
профессор кафедры СП, д.ф.-м.н., доцент Р.Ж. Алеев

Задание принял к исполнению Д.А. Манов

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ 5
1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 7
2. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 12
3. ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЯ 14
4. ВЫЧИСЛЕНИЯ 17
5. РЕАЛИЗАЦИЯ ИНТЕРФЕЙСА СИСТЕМЫ 22
6. ТЕСТИРОВАНИЕ И ИСПОЛЬЗОВАНИЕ СИСТЕМЫ 26
ЗАКЛЮЧЕНИЕ 28
ЛИТЕРАТУРА 29
ПРИЛОЖЕНИЯ 31
Приложение А. Вычисления для P = 97 31
Приложение Б. Главная функция main 33

ВВЕДЕНИЕ

Актуальность

Примитивные корни имеют важное значение в криптографии, так как они лежат в основе многих алгоритмов шифрования, таких как RSA и эллиптические кривые. Автоматизация вычисления примитивных корней позволяет быстро и эффективно находить необходимые параметры для криптографических систем, что особенно важно при работе с большими числами.

Создание программы для вычисления примитивных корней для начальных диапазонов простых чисел является актуальным в связи с необходимостью решения задач, требующих большого количества вычислений, которые очень сложно провести вручную. В сфере математики, изучение примитивных корней способствует развитию алгебраической теории чисел и помогает в поиске новых методов шифрования и кодирования.

Постановка задачи

Целью выпускной квалификационной работы является разработка программной системы для вычисления и анализа степеней примитивных корней для начальных диапазонов простых чисел. Для достижения поставленной цели необходимо решить следующие задачи.

1. Составить список простых чисел из диапазона [2..P] для заданного P.
2. Разработать функцию для вычисления примитивных корней по модулю простых чисел из диапазона [2..P].
3. Разработать функцию, которая будет находить для каждого примитивного корня количества простых чисел из диапазона [2..P], для которых это число будет примитивным корнем.
4. Создать графическое приложение, в которое будут включены все указанные функции.

Структура и содержание работы

Работа состоит из введения, пяти глав, заключения и списка литературы. Объем работы составляет 34 страницы, объем списка литературы – 17 источников.

В первой главе «Анализ предметной области» описывается разрабатываемое приложение на языке python и сравнительный обзор аналогичных языков программирования.

Во второй главе «Теоретическая часть» описываются основные определения и теоремы для дальнейшего применения и создания основы для последующего понимания и применения математических концепций в рамках исследования, а также обеспечит качественное теоретическое обоснование для дальнейших практических выводов и методов анализа.

В третьей главе «Проектирование приложения» подробно представлены функциональные и нефункциональные требования так же построена диаграмма деятельности.

В четвертой главе «Вычисления» подробно представлено вычисление для конкретного числа, приведены графики возрастания чисел как примитивных корней.

В пятой главе «Реализация интерфейса системы» производится подробное описание разработки пользовательского интерфейса для приложения. Так же раскрываются ключевые аспекты создания интуитивно понятного и функционального интерфейса, обеспечивающего необходимые возможности для пользователей.

В шестой главе «Тестирование и использование системы» производится тестирование на корректность ввода и вывода. Также производится проверка соответствия программного кода установленным функциональным требованиям.

В приложении А представлено вычисление для числа P = 97.

В приложении Б представлен листинг главной функции `main`.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

Описание предметной области

Разрабатываемая в рамках выпускной квалификационной работы программная система предназначена для вычисления и анализа примитивных корней для начальных диапазонов простых чисел, обеспечивая высокую производительность, удобство использования и широкие возможности для исследовательских и учебных целей.

Обзор литературы

В области математики существует множество ценных литературных произведений, которые представляют собой источник знаний и вдохновения для ученых, студентов и любителей математики. Давайте ближе рассмотрим несколько значимых книг в данной области.

Книга [6] представляет собой масштабное исследование в области теории чисел. Она охватывает различные темы, начиная от основных принципов до более сложных концепций. Эта книга является неотъемлемым ресурсом для всех, кто интересуется глубинами математики и желает углубить свои знания в этой области.

Книга [8] является классическим учебником по теории чисел. Она охватывает различные темы, начиная от основных принципов до более сложных концепций. Эта книга является неотъемлемым ресурсом для всех, кто интересуется глубинами математики и желает углубить свои знания в этой области.

Электронная книга [9] предоставляет читателям обширный обзор теории вероятностей. Содержащая как базовые концепции, так и продвинутые темы, эта книга поможет читателям погрузиться захватывающий мир вероятностей и их приложений в различных областях.

Учебник [16] представляет систематизированный обзор вычислительно сложных задач в области теории чисел. Автор предлагает читателям глубокий взгляд на сложные аспекты математики, основанные на теории чисел, что делает данное пособие незаменимым инструментом для изучения и

понимания данной области. Также предлагаются читателям разнообразные задачи и примеры работы с примитивными корнями, что поможет углубить понимание и применение этих концепций. Книга также, исследует алгебраические и арифметические приложения примитивных корней и их применение в различных областях математики.

Книга [10] является пособием, предназначенным для изучения алгебры и логики через решение задач. Главной целью книги является развитие навыков аналитического мышления, работа с абстрактными объектами и решение разнообразных математических задач. В ней представлены задачи различного уровня сложности, от базовых до более продвинутых, что позволяет читателю постепенно углублять свои знания и навыки в области алгебры и логики. Книга также может быть полезна как учащимся и студентам, изучающим данные дисциплины, так и тем, кто интересуется математическим мышлением и хочет расширить свой кругозор в этой области.

Сравнительный анализ аналогов

Система компьютерной алгебры «GAP»

GAP [3] – свободно распространяемая на условиях лицензии GNU GPL кроссплатформенная система компьютерной алгебры для вычислительной дискретной алгебры с особым вниманием к вычислительной теории групп.

Основными особенностями GAP являются язык программирования, внешне напоминающий Паскаль, удобные типы переменных, в т.ч. оперативно изменяемые списки и записи, а также обширная библиотека данных, включая практически все группы, порядок которых не превосходит 1000. В GAP предусмотрен интерактивный режим работы, позволяющий пользователям вводить команды, быстро получать результаты и экспериментировать с различными подходами на лету. GAP интегрируется с другими системами компьютерной алгебры, такими как SageMath, расширяя свои функциональные возможности.

Таким образом, GAP представляет собой мощный и гибкий инструмент для математиков и исследователей в области вычислительной алгебры, предлагая широкие возможности для проведения сложных вычислений и анализа алгебраических структур.

Пример работы программы представлен на рисунке 1.

Рисунок 1 – Пример работы программы «GAP»

Приложение имеет следующие достоинства:

1) может быть запущено на различных операционных системах;
2) программа имеет большой набор функций, которые позволяют выполнять разнообразные вычисления;
3) бесплатная модель распространения товара;
4) понятный и простой язык программирования.

Приложение имеет следующие недостатки:

1) устаревший, невзрачный интерфейс;
2) ограниченность функционала.

Пакет прикладных программ «MATLAB»

MATLAB [1] – это среда и язык технических расчетов, предназначенный для решения широкого спектра инженерных и научных задач любой

сложности в любых отраслях.

Основные особенности MATLAB заключаются в наличии графических приложений, более сотни прикладных программ (приложения с графическим интерфейсом), а также в использовании как средства разработки программного обеспечения. MATLAB предоставляет широкий набор функций для выполнения линейной алгебры, статистического анализа, оптимизации, дифференциальных уравнений и многие другие. Это позволяет исследователям и инженерам решать задачи любой сложности.

Пример работы программы представлен на рисунке 2.

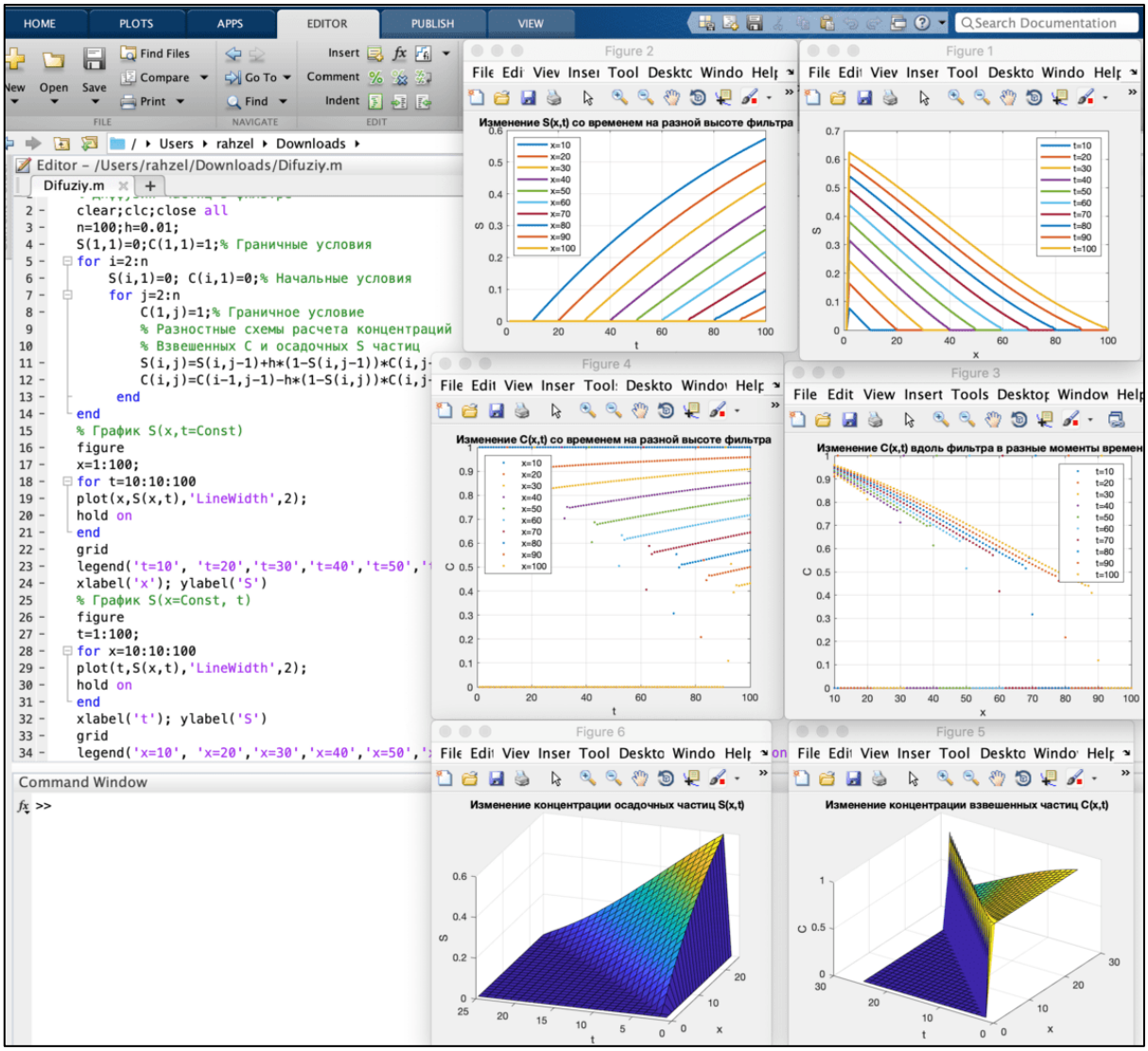

Рисунок 2 – Пример работы программы «MATLAB»

Среда разработки MATLAB имеет следующие достоинства:

1) язык легок для изучения, обладает простым и понятным синтаксисом;

2) частые обновления, как правило заметные положительные преобразования происходят не реже пары раз в год;

3) широкий функционал, включающий в себя продвинутую библиотеку для множества различных работ.

Также стоит упомянуть о недостатках:

1) платная модель распространения товара;

2) большое число команд и операторов существенно замедляют работу программ, написанных на MATLAB.

После анализа достоинств и недостатков MATLAB можно сделать вывод, что данное программное средство обладает рядом преимуществ, таких как мощные вычислительные возможности, простота использования, обширная документация и интеграция с другими инструментами. Однако, следует учитывать и некоторые недостатки, включая платную лицензию, неэффективность в определенных задачах, закрытый исходный код и ограниченные возможности веб-разработки.

В итоге, выбор использования MATLAB должен быть обоснован конкретными потребностями и целями пользователя.

Система компьютерной алгебры «Mathematica»

Mathematica [5] – система компьютерной алгебры, обеспечивающая цельную интегрированную и постоянно расширяющуюся систему, покрывающую широту и глубину технических вычислений. Простой дизайн и интуитивно понятные названия функций, состоящие из полных английских слов, позволяют с легкостью использовать программу.

Таким образом, в этом разделе был сделан обзор аналогичных проектов. Из полученных результатов можно сделать вывод, что для использования всех рассматриваемых программных систем необходимо знание языка программирования, а также только GAP распространяется бесплатно.

2. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Определение. Первообразным корнем по модулю n (primitive root mood n) называется такое число g, что все его степени по модулю n пробегают по всем числам, взаимно простым с n [7]. Математически это формулируется таким образом: есть такое целое k, что $g^k \equiv \text{a}(mod\ n)$.

Понимание свойств примитивных корней находит применение в теории автоматов [13], где эти математические концепции могут быть использованы для создания алгоритмов шифрования и моделирования различных вычислительных процессов.

Например: берется число 2 по модулю 5 и проверяется на примитивный корень на рисунке 3.

$$2^1 = 2 \equiv 2(mod\ 5)$$
$$2^2 = 4 \equiv 4(mod\ 5)$$
$$2^3 = 8 \equiv 3(mod\ 5)$$
$$2^4 = 16 \equiv 1(mod\ 5)$$

Рисунок 3 – Проверка числа 2 по модулю 5 на примитивный корень

Таким образом, 2 – примитивный корень, поскольку полученные результаты можно составить в цепочку от 1 до 4.

Теорема 2 [12]. Пусть p – простое нечетное число и α– натуральное число. Для каждого простого нечетного числа p существуют примитивные корни (Гаусс). Количество примитивных корней по модулю p равно $\varphi(p-1)$, где φ – функция Эйлера. Пусть g – примитивный корень по модулю p. Если p^2 не делит $g^{p-1} - 1 \leftrightarrow g^{p-1} - 1 \neq 0(mod\ p^2)$, то g – примитивный корень по модулю p^α.

Если g не является примитивным корнем по модулю p^α, то $g + p$ – примитивный корень по модулю p^α.

В данном примере рассматривается число 29.

Тогда примитивные корни: 2, 3, 8, 10, 11, 14, 15, 18, 19, 20, 21 в ходе всех вычислений будут выстроены в цепочку чисел (рисунок 4), где число 14 не удовлетворяет условиям теоремы и соответственно не будет являться примитивным корнем по модулю числа 29.

$$2^{29-1} - 1 \neq 0(mod 29^2)$$
$$3^{29-1} - 1 \neq 0(mod 29^2)$$
$$8^{29-1} - 1 \neq 0(mod 29^2)$$
$$10^{29-1} - 1 \neq 0(mod 29^2)$$
$$11^{29-1} - 1 \neq 0(mod 29^2)$$
$$14^{29-1} - 1 \equiv 0(mod 29^2)$$
$$15^{29-1} - 1 \neq 0(mod 29^2)$$
$$18^{29-1} - 1 \neq 0(mod 29^2)$$
$$19^{29-1} - 1 \neq 0(mod 29^2)$$
$$20^{29-1} - 1 \neq 0(mod 29^2)$$
$$21^{29-1} - 1 \neq 0(mod 29^2)$$

Рисунок 4 – Проверка числа 29 по теореме 2

Таким образом: 2, 3, 8, 10, 11, 15, 18, 19, 20, 21 – примитивные корни по модулю 29^{α} для любого натурального числа α.

Отметим также важную гипотезу.

Гипотеза Артина. Для любого целого числа α, не являющегося точным квадратом и отличного от -1, существует бесконечно много простых чисел, по модулю которых α является примитивным корнем.

Число 2 является первообразным корнем, в частности, по модулю 3 и по модулю 5, но не по модулю 7. Последовательность простых чисел, по модулю которых 2 является первообразным корнем, начинается так: 3, 5, 11, 13, 19, 29, 37, 53, 59, 61, 67, 83, 101 и т.д.

На данный момент остаётся открытым вопрос о бесконечности этой последовательности. Гипотеза Артина предполагает утвердительный ответ на этот вопрос.

3. ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЯ

Роль проектирования в создании качественного приложения

Проектирование приложения играет критически важную роль в разработке программного обеспечения. Во-первых, оно позволяет предвидеть и оптимизировать архитектуру приложения, учитывая масштабируемость, производительность и безопасность. Тщательно спроектированная архитектура может обеспечить удобство сопровождения и модификации приложения в будущем.

Кроме того, проектирование приложения способствует оптимальному использованию ресурсов, что в свою очередь приводит к повышению производительности и эффективности продукта. Тщательно спроектированный пользовательский интерфейс может улучшить пользовательский опыт и сделать приложение более удобным и привлекательным для конечных пользователей.

Также, проектирование приложения позволяет определить ключевые моменты взаимодействия с внешними системами, интеграцию с другими приложениями или API, что обеспечивает совместимость и гибкость продукта.

Более того, в процессе проектирования уделяется внимание безопасности приложения, что позволяет выявить потенциальные уязвимости и внедрить соответствующие меры защиты.

В итоге, проектирование приложения играет ключевую роль в обеспечении высокого качества программного продукта, удовлетворяющего потребности пользователей, обладающего высокой производительностью и безопасностью, и готового к развитию и масштабированию в будущем.

Функциональные требования

Функциональные требования системы описывают функционал программного обеспечения и поведение, которое должна предоставлять данная система. Для реализации приложения были выдвинуты следующие требования.

1. Пользователь должен иметь возможность вводить данные для расчета.

2. Пользователь должен иметь возможность просматривать предыдущие запросы.

Нефункциональные требования

Нефункциональные требования описывают свойства и ограничения, накладываемые на систему. Можно выделить следующие нефункциональные требования.

1. Приложение должно быть написано на языке Python.
2. Приложение должно работать на платформе Windows.

Варианты использования приложения

Диаграмма деятельности (рисунок 5) применяется для моделирования функциональных требований к программному продукту при его проектировании и разработке.

Диаграмма деятельности также способствует четкому определению ролей и ответственностей в рамках выполнения конкретного процесса. Она может быть использована для идентификации параллельных и последовательных действий, что помогает разработчикам лучше понять взаимодействие между различными компонентами системы.

Более того, диаграммы деятельности могут служить ценным инструментом коммуникации между членами команды разработки, заказчиками и другими заинтересованными сторонами, поскольку они предоставляют легко понимаемую визуализацию процессов системы. Таким образом, данная диаграмма не только помогает определить логику работы системы, но также облегчает командное взаимодействие и понимание сценариев выполнения процессов в рамках проекта разработки программного обеспечения.

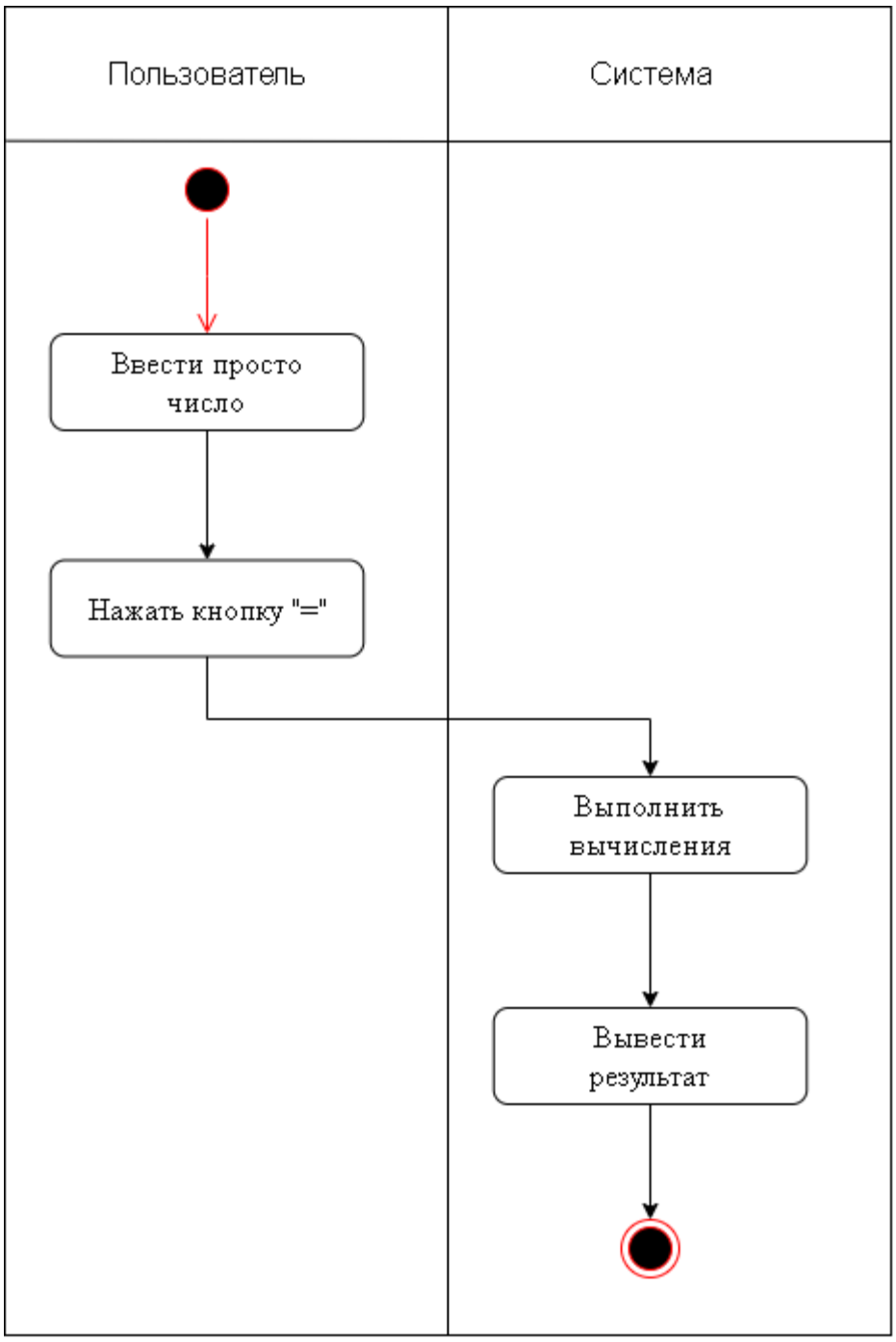

Рисунок 5 – Диаграмма деятельности

Данная диаграмма показывает, как один виток деятельности переходит к другому, при этом внимание фиксируется на результате деятельности. Она визуально отображает этапы выполнения, позволяет идентифицировать узкие места, ошибки или возможности для оптимизации.

4. ВЫЧИСЛЕНИЯ

Для разработки программной системы был выбран высокоуровневый язык программирования Python версии 3.12.2 и среда разработки PyCharm Community Edition 2020.1 [2]. При реализации программной системы использовались модули стандартной библиотеки Python Tkinter [4] (для разработки графического интерфейса).

Фиксируется простое число p = 23 [17]. Для него находятся все примитивные корни по модулю 23 в листинге 1, используя алгоритм из листинга для вычисления чисел, которые при возведении в степени дают все остатки от деления на 23.

Листинг 1 – Функция нахождения всех примитивных корней

```
def find_all_r(p):
    all_r = []
    exam = list(range(1, p))
    for i in range(2, p, +1):
        new_exam = exam.copy()
        for j in range(1, p+1, +1):
            ost = i**j % p
            try:
                new_exam.remove(ost)
            except ValueError:
                pass
        if new_exam == []:
            all_r.append(i)
    return all_r
```

В листинге 2 представлен алгоритм поиска всех простых чисел в заданном диапазоне [2..23]. Алгоритм проверяет каждое число в этом диапазоне на простоту, и если число является простым, оно добавляется в список простых чисел.

Листинг 2 – Функция нахождения всех простых чисел из диапазона

```
def find_prime(p):
    prime = []
    for i in range(2, p+1, +1):
        for j in range(2, i+1, +1):
            if i % j == 0 or i in prime:
                break
            else:
                prime.append(i)
    return(prime)
```

В листинге 3 представлен алгоритм, который вычисляет все примитивные корни для каждого простого числа из заданного диапазона [2..23].

Листинг 3 – Функция нахождения всех примитивных корней

```
def find_all_r(p):
    all_r = []
    exam = list(range(1, p))
    for i in range(2, p, +1):
        new_exam = exam.copy()
        for j in range(1, p+1, +1):
            ost = i**j % p
            try:
                new_exam.remove(ost)
            except ValueError:
                pass
        if new_exam == []:
            all_r.append(i)
    return all_r
```

Для всех простых чисел из диапазона [2..23] находятся все примитивные корни в листинге 4.

Листинг 4 – Функция нахождения всех примитивных корней по теореме 2

```
def find_all_rr(p):
    all_r = []
    exam = list(range(1, p))
    for i in range(2, p, +1):
        if (i**(p-1)-1) % p**2 != 0:
            new_exam = exam.copy()
            for j in range(1, p+1, +1):
                ost = i**j % p
                try:
                    new_exam.remove(ost)
                except ValueError:
                    pass
            if new_exam == []:
                all_r.append(i)
        else:
            print("Ушло", i)
    return all_r
```

В таблице 1, составленной на основе найденных результатов для каждого простого числа из диапазона [2..23], представлены все корни, включая примитивные, связанные с каждым числом. Эта подробная таблица не только обобщает примитивные корни, но также включает все другие корни.

Таблица 1 – Все примитивные корни числа p = 23

Корень	Числа	Количество
[2]	3, 5, 11, 13, 19	5
[3]	5, 7, 17, 19	4
[5]	7, 17, 23	3
[6]	11, 13, 17	3
[7]	11, 13, 17, 23	4
[8]	11	1
[10]	17, 19, 23	3
[11]	13, 17, 23	3
[12]	17	1
[13]	19	1
[14]	17, 19, 23	3
[15]	19, 23	2
[17]	23	1
[19]	23	1
[20]	23	1
[21]	23	1

В таблице 2, основанной на теореме 2, для каждого простого числа из расширенного диапазона [2..29], будут представлены все примитивные корни, соответствующие каждому числу.

Таблица 2 – Примитивные корни числа p = 29

Корень	Числа	Количество
[2]	3, 5, 11, 13, 19, 29	6
[3]	5, 7, 17, 19, 29	5
[5]	7, 17, 23	3
[6]	11, 13, 17	3
[7]	11, 13, 17, 23	4
[8]	11, 29	2
[10]	17, 19, 23, 29	4
[11]	13, 17, 23, 29	4
[12]	17	1
[13]	19	1
[14]	17, 19, 23	3
[15]	19, 23, 29	3
[17]	23	1
[18]	29	1
[19]	23, 29	2

Окончание таблицы 2

Корень	Числа	Количество
[20]	23	1
[21]	23, 29	2
[26]	29	1
[27]	29	1

Теперь основываясь на серии вычислений различных p, строятся графики возрастания чисел как примитивного корня.

На рисунке 6 изображен график возрастания количества 2 (как примитивного корня) в зависимости от возрастания простых чисел до тысячи.

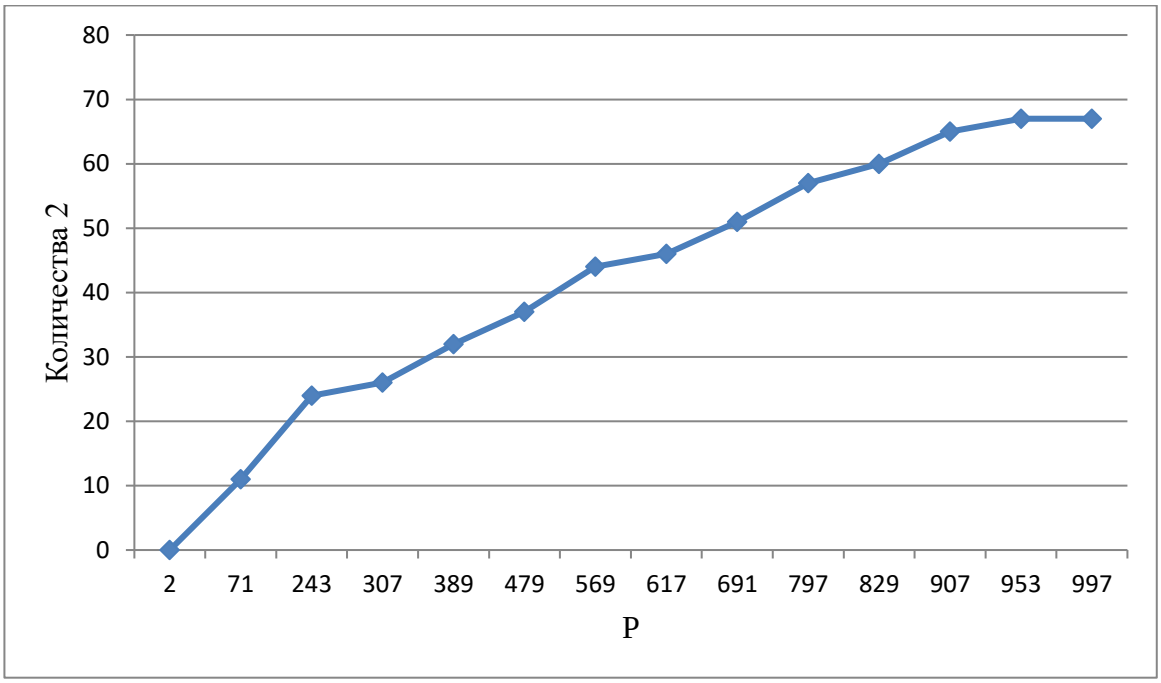

Рисунок 6 – График возрастания числа 2

На следующем графике (рисунок 7) рассматривается возрастание количества 3 (как примитивного корня) в зависимости от возрастания простых чисел до тысячи.

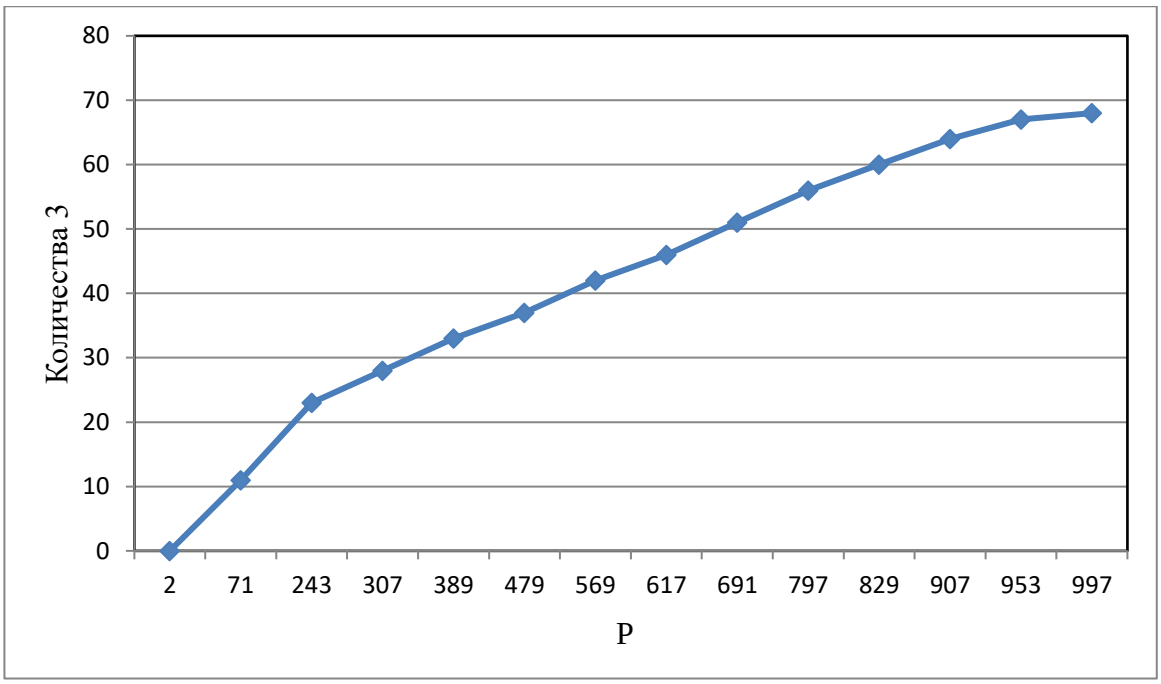

Рисунок 7 – График возрастания числа 3

Для наглядного примера сравниваются графики возрастания числа 2 и числа 3 (рисунок 8).

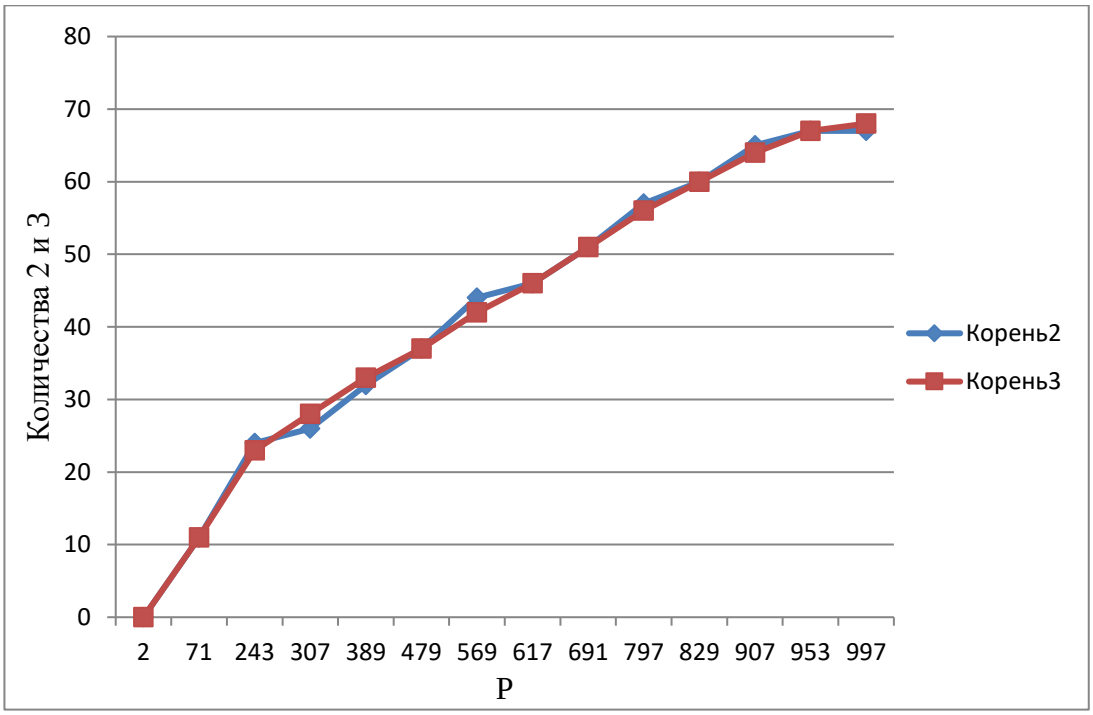

Рисунок 8 – График возрастания числа 2 и 3

Графики 6 – 8 показывают, что возрастание упомянутой ранее Гипотезы Артина подтверждается для простых чисел меньших 1000.

5. РЕАЛИЗАЦИЯ ИНТЕРФЕЙСА СИСТЕМЫ

Интерфейс пользователя (UI – англ. user interface) [15] – совокупность средств, при помощи которых пользователь общается с различными устройствами, чаще всего – с компьютером или бытовой техникой, либо иным сложным инструментарием (системой). Один из этапов разработки дизайна приложения – это создание макета экранов. Приложение должно иметь простой, интуитивно понятный пользовательский интерфейс, обеспечивающий необходимую функциональность.

При запуске программы нас встречает надпись «введите P» и поле для ввода.

На главном окне расположены все десять цифр (0..9), принцип их работы рассмотрен в листинге 5. Где для каждой кнопки заданы индивидуальные параметры («bg» – цвет фона, «row» – строчка, «column» – колонка, «stick» – ориентирование кнопки, «pad x, y» – соответствующие отступы по координатам).

Листинг 5 – Принцип работы цифр

```
def add_digit(digit):
    value = entry.get() + str(digit)
    entry.delete(0, END)
    entry.insert(0, value)
tk.Button(text='1', bd=3, bg = "gray", command=lambda
add_digit(1)).grid(row=2, column=0, stick='wens', padx=7, pady=7)
tk.Button(text='2', bd=3, bg = "gray", command=lambda :
add_digit(2)).grid(row=2, column=1, stick='wens', padx=7, pady=7)
tk.Button(text='3', bd=3, bg = "gray", command=lambda :
add_digit(3)).grid(row=2, column=2, stick='wens', padx=7, pady=7)
tk.Button(text='4', bd=3, bg = "gray", command=lambda :
add_digit(4)).grid(row=3, column=0, stick='wens', padx=7, pady=7)
tk.Button(text='5', bd=3, bg = "gray", command=lambda :
add_digit(5)).grid(row=3, column=1, stick='wens', padx=7, pady=7)
tk.Button(text='6', bd=3, bg = "gray", command=lambda :
add_digit(6)).grid(row=3, column=2, stick='wens', padx=7, pady=7)
tk.Button(text='7', bd=3, bg = "gray", command=lambda :
add_digit(7)).grid(row=4, column=0, stick='wens', padx=7, pady=7)
tk.Button(text='8', bd=3, bg = "gray", command=lambda :
add_digit(8)).grid(row=4, column=1, stick='wens', padx=7, pady=7)
tk.Button(text='9', bd=3, bg = "gray", command=lambda :
add_digit(9)).grid(row=4, column=2, stick='wens', padx=7, pady=7)
tk.Button(text='0', bd=3, bg = "gray", command=lambda :
add_digit(0)).grid(row=5, column=0, columnspan=2, stick='wens', padx=7,
pady=7)
```

Аналогичным образом на главной панели располагается кнопка «=», при нажатии на которую запускается основная функция `main`, производит все возможные вычисления и выводит результат. Ее функционал рассмотрен в приложении Б.

Кнопка «C» служит для удаления всех цифр из поля ввода, предоставляя пользователю возможность быстро очистить введенные данные и начать новый ввод с чистого листа. Ее использование приведено в листинге 6, где подробно описан механизм обработки нажатия этой кнопки для обеспечения максимальной удобности и эффективности работы с интерфейсом ввода данных.

Листинг 6 – Принцип работы кнопки «C»

```
def clear():
    entry.delete(0, END)
    entry.insert(0, '')
tk.Button(text='C', bd=3, bg = "#FF7400", command=lambda :
clear()).grid(row=3, column=3, stick='wens', padx=7, pady=7)
```

Кнопка «←» используется для удаления последней цифры в поле ввода. Ее назначение представлено в листинге 7, где подробно описан алгоритм обработки нажатия этой кнопки для улучшения взаимодействия с системой ввода данных.

Листинг 7 – Принцип работы кнопки «←»

```
def del_digit():
    value = entry.get()[:-1]
    entry.delete(0, END)
    entry.insert(0, value)
tk.Button(text='←', bd=3, bg = "#FF7400", command=lambda :
del_digit()).grid(row=2, column=3, stick='wens', padx=7, pady=7)
```

На рисунке 9 изображено главное окно приложения, обладающее полным набором необходимых кнопок для проведения вычислений. Этот интерфейс предоставляет пользователям доступ к широкому спектру математических функций, включая математические операции над диапазоном простых чисел и анализ данных. Такая многофункциональность обеспечивает удобство использования и позволяет эффективно проводить вычисления примитивных корней.

Рисунок 9 – Главное окно приложения

При вводе чисел открывается дополнительное окно, где пользователь может увидеть результаты вычислений программы для заданного числа P. Это обеспечивает удобство пользователей и позволяет им наглядно ознакомиться с выходными данными программы, улучшая взаимодействие с программой. Пример работы программы наглядно продемонстрирован на рисунке 10.

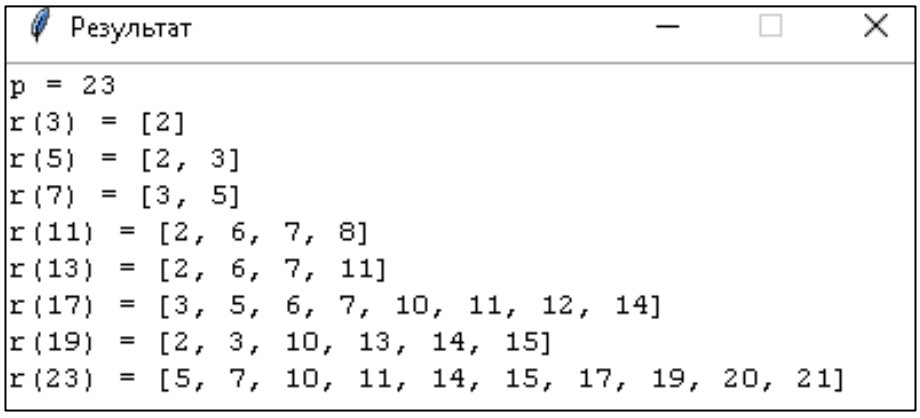

Рисунок 10 – Результат для числа P = 23

При вводе некорректных выражений программа автоматически открывает дополнительное окно, где отображается ошибка. Пример возникшей ошибки и процесса перезапуска программы наглядно продемонстрирован на рисунке 11.

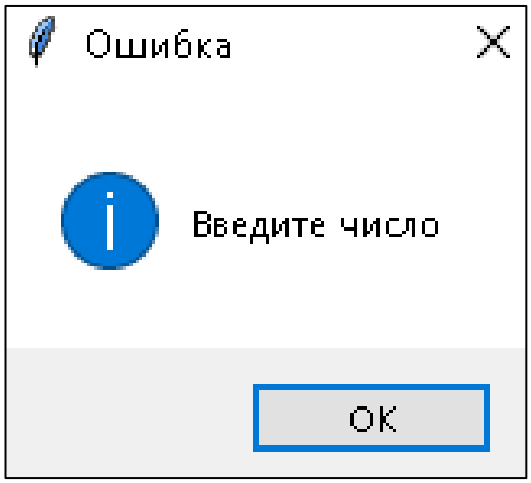

Рисунок 11 – Пример ошибки

При вводе чисел, для которых не существует примитивных корней, программа отображает вспомогательное окно с предупреждением. В качестве примера можно рассмотреть число 22 (рисунок 12), для которого не существует соответствующего примитивного корня. Это предупреждение помогает пользователям понять особенности вводимых чисел и предоставляет им информацию о работе программы в случае отсутствия примитивных корней для заданного числа.

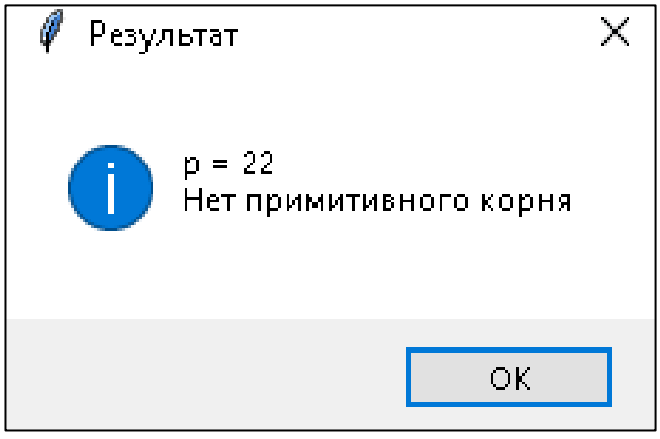

Рисунок 12 – Результат для числа P = 22

6. ТЕСТИРОВАНИЕ И ИСПОЛЬЗОВАНИЕ СИСТЕМЫ

Функциональное тестирование

Функциональное тестирование (functional testing) [14] – вид тестирования, направленный на проверку корректности работы функциональности приложения (корректность реализации функциональных требований).

Основной задачей функционального тестирования является подтверждение того, что разработанная программная система соответствует исходным функциональным требованиям.

Выполнено полноценное тестирование системы, в котором:

1) было проведено функциональное тестирование для проверки соответствия функциональным требованиям программы;

2) входные данные задаются в виде диапазона чисел [2…P] для заданного P;

3) были проведены тесты на корректность выходных данных.

В таблице 3 приведены результаты данного тестирования.

Таблица 3 – Результаты тестирования входных и выходных данных

№	Работа программы	Ожидаемый результат	Тест
1	Ввод: [23] Вывод: [[2, [ ]], [3, [2]], [5, [2, 3]], [7, [3, 5]], [11, [2, 6, 7, 8]], [13, [2, 6, 7, 11]], [17, [3, 5, 6, 7, 10, 11, 12]], [19, [2, 3, 10, 13, 14, 15]], [23, [5, 7, 10, 11, 14, 15, 17,19, 20, 21]]]	Ввод: [23] Вывод: [[2, [ ]], [3, [2]], [5, [2, 3]], [7, [3, 5]], [11, [2, 6, 7, 8]], [13, [2, 6, 7, 11]], [17, [3, 5, 6, 7, 10, 11, 12]], [19, [2, 3, 10, 13, 14, 15]], [23, [5, 7, 10, 11, 14, 15, 17,19, 20, 21]]]	Пройден
2	Ввод: [18] Вывод: [[1, 0], [2, 1], [3, 4], [4, 2], [5, 9], [6, 5], [7, 11], [8, 3], [9, 8], [10, 10],[11, 7], [12, 6]]	Ввод: [18] Вывод: [[1, 0], [2, 1], [3, 4], [4, 2], [5, 9], [6, 5], [7, 11], [8, 3], [9, 8], [10, 10],[11, 7], [12, 6]]	Пройден
3	Ввод: [21] Вывод: [[2, [ ]], [3, [2]], [5, [2, 3]], [7, [3, 5]], [11, [2, 6, 7, 8]], [13, [2, 6, 7, 11]], [17, [3, 5, 6, 7, 10, 11, 12]], [19, [2, 3, 10, 13, 14, 15]]	Ввод: [21] Вывод: [[2, [ ]], [3, [2]], [5, [2, 3]], [7, [3, 5]], [11, [2, 6, 7, 8]], [13, [2, 6, 7, 11]], [17, [3, 5, 6, 7, 10, 11, 12]], [19, [2, 3, 10, 13, 14, 15]]	Пройден

Было произведено функциональное тестирование разработанной системы, результаты которого представлены в таблице 4. Проведенное тестирование охватывало все ключевые аспекты функциональности системы, включая проверку корректности введения и обработки данных, а также работу различных интерфейсных элементов и кнопок. Результаты тестирования демонстрируют, насколько эффективно и бесперебойно система выполняет поставленные задачи, что подтверждается данными таблицы 4.

Таблица 4 – Функциональное тестирование

№	Цель теста	Действия	Ожидаемый результат	Тест пройден?
1	Запустить приложение	Ввести в окошке число P и нажать «=»	Открылось окно с результатами числа P	Да
2	Проверить функционал кнопок калькулятора	Нажать на кнопки от 1 до 9, кнопку «C» и «←»	При нажатии цифр в окно вводятся числа, а при нажатии «C» удаляются все числа и «←» удаляется последнее число.	Да
3	Ввести число, у которого нету примитивных корней	Ввести в окошке число P и нажать «=»	Открылось окно для заданного числа, где написано нету примитивных корней	Да
4	Ввести большое простое число	Ввести в окошке большое простое число P и нажать «=»	Благополучно открылось окно с результатами числа P	Да
5	Ввести буквы, знаки препинания, восклицания и др.	Ввести в окошке буквы, знаки препинания, восклицания и др.	Открылось окно ошибка введите число	Да

Таким образом, результаты тестирования позволили выделить ключевые аспекты, которые требуют мер по улучшению. Дальнейшее совершенствование и доработка приложения являются ключевыми факторами в обеспечении позитивного пользовательского опыта и достижении высоких стандартов качества.

ЗАКЛЮЧЕНИЕ

В рамках выпускной квалификационной работы было разработано приложение, которое при вводе диапазонов простых чисел позволяет определить количество примитивных корней для каждого значения, учитывая свойства примитивных корней. При этом были решены следующие задачи.

1. Произведен обзор литературы и существующих приложений по предметной области.
2. Вычислены примитивные корни по модулю простых чисел из диапазона [2..P] для заданного P.
3. Найдено количество примитивных корней для каждого значения.
4. Проведено тестирование приложения для корректной работы нахождения чисел из диапазона.
5. На основе предыдущих пунктов создано окончательное приложение, в котором содержится простой гайд по использованию, вычислению и анализу примитивных корней.

В будущем планируется продолжать разработку и улучшение приложения, уделяя особое внимание созданию интуитивно понятного и удобного интерфейса, который повысит удобство использования и удовлетворенность пользователей. Кроме того, будет внедрена функция построения графиков возрастания чисел, что позволит проводить более точный и наглядный анализ степеней чисел, а также других математических закономерностей и зависимостей. Помимо этого, рассматриваются возможности добавления новых функциональных возможностей, таких как интеграция с другими приложениями, расширенные аналитические инструменты, возможность экспорта данных и отчетов, а также оптимизация существующего функционала для повышения производительности, быстродействия и расширения области применения приложения.

ЛИТЕРАТУРА

1. MATLAB – MathWorks – MATLAB & Simulink. [Электронный ресурс] URL: https://www.mathworks.com/products/matlab.html (дата обращения: 20.05.2024 г.).

2. PyCharm. [Электронный ресурс] URL: https://www.jetbrains.com/pycharm/ (дата обращения: 20.05.2024 г.).

3. The GAP Group, GAP. // Programming Version 4.12.2, 2022. [Электронный ресурс] URL: https://www.gap-sytem.org/Doc/manuals (дата обращения: 21.04.2024 г.).

4. Tkinter – Python interface to Tcl/Tk. [Электронный ресурс] URL: https://docs.python.org/3. /library/tkinter.html (дата обращения: 24.05.2024 г.).

5. Wolfram Mathematica. [Электронный ресурс] URL: https://www.wolfram.com/mathematica/ (дата обращения: 20.04.2024 г.).

6. Агафонов В.Н. Математические основы обработки информации. // Новосибирск: НГУ, 1982. – 92 с.

7. Верещагин Н.К., Шень А. Работа с примитивными корнями. // М.: МЦНМО, 2000. – 128 с.

8. Виноградов И.М. Основы теории чисел. // М.–Л., Гостехиздат, 1952. – 180 с.

9. Попов В.А. Теория вероятностей. [Электронный ресурс] URL: https://kpfu.ru/docs/F1451194118/PLM_part2.pdf (дата обращения: 11.05.2024 г.).

10. Гиндикин С.Г. Алгебра и логика в задачах. // М.: Наука, 1972. – 288 с.

11. Гречников Е.А. Вычислительно сложные задачи теории чисел: учебное пособие // Москва: МГУ имени М.В.Ломоносова, 2012. – 312 с.

12. Ерусалимский Я.М. Дискретная математика: Теория, задачи, приложения. // М.: Вузовская книга, 2011. – 265 с.

13. Карпов Ю.Г. Теория автоматов. // М.: Питер, 2003. – 206 с.

14. Куликов С.С. Тестирование программного обеспечения. // Базовый курс. – Минск: Четыре четверти, 2017. – 312 с.

15. Новожилова Н. Особенности проектирования дружественных интерфейсов. [Электронный ресурс] URL: https://cyberleninka.ru/article/n/osobennosti-proektirovaniyadruzhestvennyh-interfeysov-dlya-polzovateley-ekonomistov/viewer (дата обращения: 10.05.2024 г.).

16. Нестеренко Ю.В. Теория чисел. // М. Академия, 2008. – 292 с.

17. Трост Э.В. Простые числа. // Москва, 1959. – 135 с.

ПРИЛОЖЕНИЯ

Приложение А. Вычисления для P = 97

В таблице 1 представлено подробное вычисление для числа 97.

Таблица 1 – Вычисления для P = 97

Корень	Числа	Количество
2	3, 5,11,13,19,29,37,53,59,61,67,83	12
3	5, 7,17,19,29,31,43,53,79,89	10
5	7,17,23,37,43,47,53,73,83,97	10
6	11,13,17,41,59,61,79,83,89	9
7	11,13,17,23,41,61,67,71,79,89,97	11
8	11,29,53,59,83	5
10	17,19,23,29,47,59,61,97	8
11	13,17,23,29,31,41,47,59,67,71,73	11
12	17,31,41,43,53,67	6
13	19,31,37,41,47,59,67,71,73,83,89,97	12
14	17,19,23,29,53,59,73,83,89,97	10
15	19,23,29,37,41,47,73,83,89,97	10
17	23,31,37,41,61,97	6
18	29,37,43,53,59,61,67,83	8
19	23,29,37,41,43,47,53,83,89	9
20	23,37,43,47,53,67,73,83	8
21	23,29,31,53,71,97	6
22	31,37,41,47,53,71,83	7
23	47,59,89,97	4
24	31,37,41,59,83,89	6
26	29,41,43,47,53,61,73,89,97	9
27	29,53,83	3
28	41,43,67,71,79,89	7
29	41,43,47,73,79,89,97	7
30	41,43,47,59,61,79,89	7
31	47,53,59,61,67,71,73,89	8
32	37,53,59,67,83	5
33	43,47,53,59,71,73,89	7
34	41,43,53,59,67,73,79,83	8
35	37,41,47,53,61,71,79,83,89	9
37	59,79,97	3
38	47,59,89,97	4
39	47,53,59,73,79,83,97	7
40	47,59,73,97	4
41	47,53,67,89,97	5
42	59,71,73,83	4
43	47,59,61,79,83,89	6
44	47,59,61,67,71,73	6
45	47,53,73,83	4
46	67,83,89	3
47	59,71,73,79,83	5
48	53,67,79,89	4

Окончание таблицы 1 приложения А

Корень	**Числа**	**Количество**
50	53,59,67,83	4
51	53,61,67,89	4
52	59,71,83	3
53	71, 73, 79, 83	4
54	59, 61, 79, 83, 89	5
55	59, 61, 71, 83	4
56	59, 71, 83, 89, 97	5
57	67, 83, 97	3
58	73, 83, 89, 97	4
59	61, 71, 73, 79, 89, 97	6
60	73, 79, 83, 89, 97	5
61	67, 71, 89	3
62	71, 73, 83, 89	4
63	67, 71, 79, 89	4
65	71, 89	2
66	79, 83, 89	3
67	71, 83	2
68	71, 73, 79, 97	4
69	71	1
70	79, 89	2
71	83, 97	2
72	83	1
73	83	1
74	79, 83, 89, 97	4
75	79, 89	2
76	83, 89, 97	3
77	79	1
79	83	1
80	83, 97	2
82	89, 97	2
83	89, 97	2
84	97	1
86	89	1
87	97	1
90	97	1
92	97	1

Приложение Б. Главная функция main

Принцип работы функции `main` представлен в листинге 1.

Листинг 1 – Принцип работы функции `main`

```
def main():

    # Нахождение всех простых чисел до p
    def find_prime(p):
        prime = []
        for i in range(2, p+1, +1):
            for j in range(2, i+1, +1):
                if i % j == 0 or i in prime:
                    break
                else:
                    prime.append(i)
        return(prime)

    # Нахождение всех примитивных корней r
    def find_all_r(p):
        all_r = []
        exam = list(range(1, p)) # готовая цепочка для проверки чисел на
примитивность
        for i in range(2, p, +1):
            new_exam = exam.copy()
            for j in range(1, p+1, +1):
                ost = i**j % p
                try:
                    new_exam.remove(ost)
                except ValueError:
                    pass
            if new_exam == []:
                all_r.append(i)
        return all_r

    # Нахождение всех примитивных корней r (с ограничением)
    def find_all_rr(p):
        all_r = []
        exam = list(range(1, p)) # готовая цепочка для проверки чисел на
примитивность
        for i in range(2, p, +1):
            if (i**(p-1)-1) % p**2 != 0:
                new_exam = exam.copy()
                for j in range(1, p+1, +1):
                    ost = i**j % p
                    try:
                        new_exam.remove(ost)
                    except ValueError:
                        pass
                if new_exam == []:
                    all_r.append(i)
            else:
                print("Ушло", i)
        return all_r

    # Задается p
    try:
        p = int(entry.get())
    except ValueError:
        result_text = "Введите число"
```

```
    messagebox.showinfo(title='Ошибка', message=result_text)
    exit()

result_text = f"p = {p}\n"

# Все простые до p
prime = find_prime(p)

# Проверка есть ли хотя бы один примитивный корень
r = find_all_rr(p)
if r == []:
    result_text += "Нет примитивного корня\n\n"
    messagebox.showinfo(title='Результат', message=result_text)
    exit()

# Все корни для всех простых до p
for i in prime:
    result_text += f"r({i}) = {find_all_rr(i)}\n"

result_window = tk.Toplevel()
result_window.title('Результат')

text_area = tk.Text(result_window, height=30, width=50)
text_area.insert(tk.END, result_text)
text_area.pack()
```