|  |  |  |
| --- | --- | --- |
| ОТЧЕТ ЗАЩИЩЕН С ОЦЕНКОЙ: |  |  |

ПРЕПОДАВАТЕЛЬ:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| доцент, к.т.н., доцент | / |  | / |  | / | В. Н. Коромысличенко |
| (должность, учёная степень, звание) |  | (подпись) |  | (дата защиты) |  | (инициалы, фамилия) |

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №3

«Разработать требования к программе расчета чисел Фибоначчи»

ПО КУРСУ: «Разработка и анализ требований»

|  |  |  |  |
| --- | --- | --- | --- |
| РАБОТУ ВЫПОЛНИЛ СТУДЕНТ: | 4134К | / | Самарин Д.В. |
|  | (номер группы) |  | (инициалы, фамилия) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | / |  | / |  |
|  |  | (подпись студента) |  | (дата отчета) |

Разработка требования к программе расчета чисел Фибоначчи

# 1. Введение

Цель данного доклада – разработка требований к программе, выполняющей вычисление последовательности чисел Фибоначчи. Программа должна принимать на вход порядковый номер (или количество элементов последовательности), корректно рассчитывать требуемое число или последовательность, обрабатывать исключительные ситуации (например, некорректный ввод, отрицательные значения или слишком большие числа) и представлять результат пользователю. Спецификация составлена с учётом современных методик инженерии требований, использования сценарного подхода, трассировки и спецификации требований.

Особое внимание уделяется:

* Функциональным требованиям: обеспечение ввода данных, валидации, вычислительной логики и вывода результата.
* Нефункциональным требованиям: производительность, надёжность, удобство использования, безопасность и масштабируемость.
* Сценариям использования: последовательность действий пользователя и системы.
* Архитектурным решениям: модульное разделение компонентов и обеспечение трассировки требований.

# 2. Анализ предметной области и постановка задачи

Предметная область:

Программа расчёта чисел Фибоначчи предназначена для вычисления членов последовательности Фибоначчи – классической математической задачи, применимой в обучении, исследовательских целях и в задачах оптимизации. Задача состоит в том, чтобы, по введённому пользователем параметру (номер числа или количество элементов), вычислить соответствующее число Фибоначчи или всю последовательность.

Основные задачи системы:

* Приём и проверка корректности ввода (целое неотрицательное число).
* Вычисление числа Фибоначчи с использованием оптимальных алгоритмов (рекурсивного или итеративного метода, возможно, с мемоизацией).
* Вывод результата расчёта в удобном для пользователя виде.
* Обработка ошибок (например, неверный формат ввода, слишком большое значение, приводящее к переполнению).

Эта постановка задачи соответствует требованиям, изложенным в документе по инженерии требований, где выделяются процессы спецификации, валидации и трассировки требований.

# 3. Функциональные требования

Функциональные требования описывают функции, которые должна выполнять система:

# 3.1. Ввод и проверка данных

* Программа должна принимать ввод пользователя через графический интерфейс или консоль.
* Вводимые данные должны представлять собой целое неотрицательное число (номер члена последовательности или количество элементов).

Проверка корректности:

* Вводимые данные проверяются на соответствие числовому формату; в случае некорректного ввода (например, символы вместо цифр) генерируется сообщение об ошибке.

# 3.2. Вычисление чисел Фибоначчи

# Программа должна корректно вычислять i‑ый член последовательности Фибоначчи.

# Возможна опция расчёта всей последовательности до заданного номера.

# Реализовать выбор оптимального алгоритма: итеративный подход, рекурсия с мемоизацией или использование матричных методов.

# Обработка граничных случаев: нулевой и первый члены последовательности.

# 3.3. Вывод результатов

Интерфейс вывода:

Результат вычисления выводится пользователю в отдельном поле (или в консольном окне).

Логирование ошибок:

При наличии ошибки – вывод соответствующего сообщения с рекомендациями.

Опционально: возможность сохранения результата или истории вычислений в файл.

# 3.4. Дополнительные возможности (опционально)

# Графическое представление последовательности (например, построение графика).

# Возможность выбора метода расчёта (для демонстрационных целей).

# Калькуляция с поддержкой больших чисел (при использовании библиотек для работы с большими числами).

# 4. Нефункциональные требования

Нефункциональные требования определяют характеристики системы, не связанные с основной функциональностью:

# 4.1. Производительность

# Время вычисления должно быть минимальным, особенно для небольших значений входного параметра.

# При выборе алгоритма для больших значений обеспечить оптимизацию вычислений (время отклика не должно превышать допустимые рамки).

# 4.2. Надежность и устойчивость

# Корректная обработка исключений ввода и математических ограничений.

# Система не должна аварийно завершать работу при получении некорректных данных.

# Обработка переполнения и предупреждение пользователя о невозможности расчёта при слишком большом значении.

# 4.3. Удобство использования (Usability)

# Интуитивно понятный интерфейс с чёткими инструкциями по вводу данных.

# Сообщения об ошибках должны быть информативными и понятными.

# Возможность работы с программой как в графическом режиме, так и в режиме командной строки.

# 4.4. Безопасность

# Защита от внедрения вредоносных данных через ввод.

# При сохранении истории или результатов обеспечить контроль доступа к файлам.

# 4.5. Поддерживаемость и масштабируемость

# Модульная архитектура для упрощения расширения функционала (например, добавление новых алгоритмов вычисления).

# Чёткая документация к коду, позволяющая последующую модификацию и отладку.

# Возможность интеграции с другими программными компонентами (например, в образовательных системах).

# 4.6. Повторяемость

# Программа должна работать на различных операционных системах (Windows, Linux, macOS).

# Использование кроссплатформенных технологий и библиотек (например, PyQt, Tkinter или консольный режим на Python/Java/C#).

# 5. Требования к пользовательскому интерфейсу

Для обеспечения удобного взаимодействия с пользователем необходимо детально описать требования к интерфейсу:

# 5.1. Главное окно приложения

# Поле ввода для задания номера члена последовательности или количества элементов.

# Чёткие ярлыки (labels) с пояснениями к каждому полю ввода.

# Кнопка «Рассчитать», запускающая процесс вычисления.

# 5.2. Окно вывода результата

# Отдельное поле или диалоговое окно для отображения результата расчёта.

# При ошибке – вывод сообщения с описанием проблемы и инструкцией по повторному вводу.

# 5.3. Элементы управления

# Логичное расположение элементов управления для удобного взаимодействия.

# Возможность переключения между режимами (расчёт одного числа или всей последовательности).

# Поддержка адаптивного дизайна для различных разрешений экрана.

# 7. Системные требования (System Requirements)

Поддержка вычисления последовательности Фибоначчи для целых неотрицательных чисел.

Ограничение входных данных:

* Установить разумное верхнее значение для ввода, чтобы избежать длительного ожидания или переполнения.

Модульное разделение:

* Отдельные модули для ввода/вывода, валидации, вычислений и логирования ошибок.

# 8. Требования к атрибутам качества (Quality Attributes)

Надежность:

* Корректное вычисление и обработка ошибок ввода, предотвращение аварийных завершений.

Производительность:

* Оптимизированные алгоритмы вычисления, особенно для больших значений.

Портируемость:

* Работа на Windows, Linux, macOS.

Удобство:

* Понятный интерфейс, ясные сообщения и интуитивная навигация.

Безопасность:

* Защита от некорректных и потенциально вредоносных данных.

Масштабируемость:

* Возможность расширения функционала (например, добавление алгоритмов для расчёта других числовых последовательностей).

# 9. Схемы и визуальное представление

Для лучшего понимания процессов разработки требований приведены несколько схем.

# 9.1. Схема 1: Основные разделы разработки требований

Эта схема иллюстрирует основные этапы формирования требований, как представлено в лекционном материале:

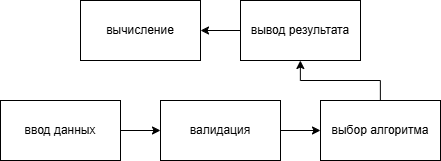


Схема 1 иллюстрирует последовательность этапов разработки требований, начиная от их сбора и заканчивая трассировкой, что позволяет обеспечить контроль корректности и полноты спецификации.

# 9.2. Схема 2: Компонентная диаграмма

Рассмотрим нашу диаграмму:

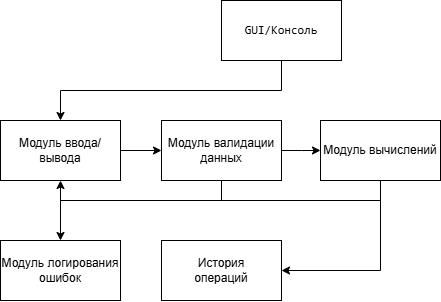


Схема 2 показывает, как система разделена на независимые модули, что обеспечивает удобство поддержки, тестирования и расширения функционала.

# 9.3. Схема 3: Use-case диаграмма

Ниже приведена схема основного сценария использования для программы деления чисел:



Схема 3 демонстрирует поток взаимодействия между пользователем и системой: от ввода данных до вывода результата с проверкой корректности на каждом этапе.

# 9.4. Схема 4: Трассировка требований

Данная схема иллюстрирует связь между функциональными требованиями, сценариями использования и тестированием:



Схема 4 демонстрирует механизм трассировки: каждое требование отображается в сценариях использования, что позволяет разработчикам и тестировщикам контролировать соответствие реализованной функциональности исходным требованиям.

# 9.5. Схема 5: Диаграмма классов

Разработка подробной диаграммы классов.

# 9.6. Схема 6: Пример интерфейса

Разработка интерфейса программы.

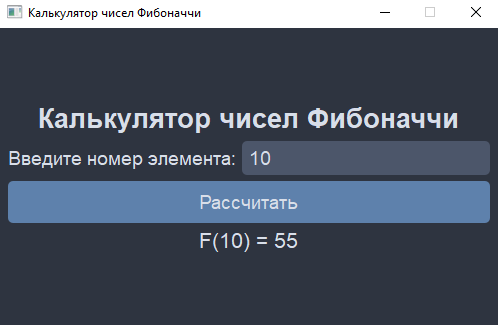


Схема 6 демонстрирует макет программы

# 10. Трассировка и спецификация требований

Для обеспечения качества и непротиворечивости требований применяется следующая система:

Матрица трассировки:

* Каждое требование получает уникальный идентификатор (например, FR-01, NFR-02) для обеспечения трассировки между функциональными требованиями, сценариями использования и тестовыми случаями.

Валидация и верификация:

* Экспертная оценка требований с привлечением потенциальных пользователей, прототипирование функций и согласование документа с заказчиком.

Обновление документа:

* Обновление документа сопровождается корректировкой матрицы трассировки, что позволяет отслеживать внесённые изменения и их влияние на компоненты системы.

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Тип | Сценарий использования | Тестовый случай |
| FR-01 |  |  |  |
| NFR-03 |  |  |  |

# 11. Технические аспекты реализации

# 11.1. Выбор платформы и языка программирования

Язык:

* Рекомендуется использовать кроссплатформенные языки (например, Java, C# или Python с библиотеками для создания GUI).

Инструменты:

* Использование современных средств разработки и фреймворков, позволяющих обеспечить модульность и масштабируемость:

Например:

PyQt5:

* + Для создания графического интерфейса с поддержкой анимаций и стилей.

Qt Designer:

* + Для визуального проектирования интерфейса.

Git:

* + Для контроля версий и совместной разработки.

Pytest:

* + Для модульного и интеграционного тестирования.

Sphinx:

* + Для автоматической генерации документации из комментариев в коде.

# 11.2. Архитектурные решения

Модульность:

* Четкое разделение логики ввода/вывода, проверки данных, вычислительной логики и обработки ошибок:

Модуль ввода/вывода:

* + Обрабатывает взаимодействие с пользователем через графический интерфейс.

Модуль проверки данных:

* + Проверяет корректность введенных данных.

Модуль вычислений;

Модуль обработки ошибок:

* + Логирует и отображает ошибки.

Модуль работы с файлами:

* + Сохраняет результаты и загружает историю операций.

Объектно-ориентированный подход:

* Использование классов и объектов для реализации основных функциональных блоков системы.
* Повторное использование кода.
* Упрощение тестирования.
* Возможность добавления новых алгоритмов или дополнительных режимов работы.

# 11.3. Документация

1. Подробные комментарии в исходном коде.
2. Разработка документации, описывающей архитектуру, логику работы и инструкции по сборке/развертыванию.

|  |
| --- |
| Архитектура системы:   * Описание модулей, их взаимодействия и диаграммы классов.   Логика работы:   * Пошаговое описание работы программы, включая обработку ошибок и анимации.   Инструкции по сборке/развертыванию:   * Установка зависимостей. * Запуск программы. |

1. Руководство пользователя

|  |
| --- |
| Описание интерфейса:   * Как вводить данные, выполнять операции и просматривать историю.   Обработка ошибок:   * Что делать при некорректном вводе. |

1. Руководство программиста

|  |
| --- |
| * Описание архитектуры и модулей. * Инструкции по добавлению новых функций (например, других математических операций). * Примеры тестов и их запуск. |

# 12. Тестирование и контроль качества

Для обеспечения высокого качества разрабатываемой программы необходимо провести комплексное тестирование:

# 12.1. Функциональное тестирование

# Проверка корректности вычисления для различных входных данных (0, 1, небольшие и граничные значения).

# Тестирование обработки ошибок: ввод нечисловых значений, отрицательных чисел, чисел, превышающих допустимые пределы.

# 12.2. Тестирование производительности

1. Измерение времени отклика для расчёта числа Фибоначчи при разных значениях входного параметра.
2. Тестирование оптимизированных алгоритмов для больших значений.
3. Оценка работы системы на различных платформах.

* Тестирование на Windows, Linux и macOS для проверки кроссплатформенности.
* Убедиться, что интерфейс и функциональность работают одинаково на всех платформах.

# 12.3. Тестирование удобства использования

1. Оценка интуитивности интерфейса.

* Провести тестирование с реальными пользователями.
* Убедиться, что интерфейс понятен и не требует дополнительных инструкций.

1. Проверка понятности сообщений об ошибках и инструкций для пользователя.

* Сообщения об ошибках должны быть четкими и информативными.

# 13. Заключение

В данном докладе представлена детальная спецификация требований для разработки программы расчёта чисел Фибоначчи. Спецификация охватывает все этапы – от сбора и анализа требований до архитектурных решений и тестирования. Основные выводы:

* Программа должна обеспечивать корректный ввод данных, их валидацию, вычисление требуемых чисел Фибоначчи и вывод результатов.
* Особое внимание уделяется оптимизации алгоритмов для обеспечения производительности при расчёте больших значений.
* Модульная архитектура и подробная документация позволят в будущем расширять функционал и интегрировать систему в более крупные проекты.

Данная спецификация может быть доработана с учётом конкретных требований заказчика и особенностей выбранной технологии реализации.