МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра «Измерительно-вычислительные комплексы»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | |  | |  |  |
| |  |  | | --- | --- | | ***Подп. и дата*** |  | | ***Инв. № дубл.*** |  | | ***Взам. инв. №*** |  | | ***Подп. и дата*** |  | | ***Инв. № подл*** |  | |  | | ПОЯСНИТЕЛЬНАЯ ЗАПИСКА  на лабораторную работу №12  по дисциплине «Алгоритмы и структуры данных»  Тема «Разработка компьютерной программы ‘Менеджер задач’» | | | | | | |
|  | | |  | |  | | | |
|  | | Исполнитель  студент гр. ИСТбд-21 Казаров Д.  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 2024 г. | | | |
|  | | 2024 | | | | | | |

**Введение**

Разрабатываемое приложение представляет собой менеджер задач, созданный с использованием языка программирования Python и библиотеки Tkinter для графического интерфейса. Целью приложения является организация и управление задачами пользователя, включая их создание, редактирование, удаление и отслеживание статуса выполнения. Программа демонстрирует навыки программирования и создания пользовательских интерфейсов для практического применения в повседневной жизни.

**Основания для разработки:**

Задание для лабораторной работы по учебному плану направления «Информационные системы и технологии».

**Постановка задачи**

Цель разработки заключается в создании удобного и функционального менеджера задач, который позволяет пользователям эффективно управлять своими задачами, категоризировать их, устанавливать дедлайны и отслеживать прогресс выполнения.

**Структура программы**

**Классы:**

1. TaskManager: основной класс приложения, управляющий всей функциональностью.

Основные функции:

1. create\_widgets - создание элементов интерфейса.

2. add\_task - добавление новой задачи.

3. mark\_complete - отметка задачи как выполненной.

4. delete\_task - удаление задачи.

5. edit\_task - редактирование существующей задачи.

6. update\_task\_list - обновление списка задач в интерфейсе.

7. save\_tasks - сохранение задач в файл.

8. load\_tasks - загрузка задач из файла.

**Функциональное назначение:**

1. Создание, редактирование и удаление задач.

2. Категоризация задач.

3. Установка дедлайнов для задач.

4. Отметка задач как выполненных.

5. Сортировка задач по различным критериям.

6. Сохранение и загрузка задач между сеансами работы.

**Проектная часть**

**1.1 Математические методы**

Приложение использует базовые операции с датами и временем для управления дедлайнами задач. Также применяются алгоритмы сортировки для организации задач по различным критериям.

**1.2 Архитектура и алгоритмы**

**1.2.1 Архитектура**

Архитектура приложения включает следующие основные компоненты:

• Главное окно приложения с элементами управления.

• Список задач, отображающий информацию о каждой задаче.

• Формы для добавления и редактирования задач.

• Система категорий для организации задач.

• Механизм сохранения и загрузки данных.

**1.2.2 Алгоритмы**

• Алгоритм добавления задачи: Проверка введенных данных, создание объекта задачи, добавление в список и обновление интерфейса.

• Алгоритм редактирования задачи: Получение выбранной задачи, отображение формы редактирования, обновление данных задачи и интерфейса.

• Алгоритм сортировки задач: Сортировка списка задач по выбранному критерию (категория, дата создания, дедлайн, статус).

**Стадии и этапы разработки**

1. Определение требований: постановка задач и определение функциональности программы.

2. Проектирование: разработка структуры классов и функций.

3. Реализация: написание кода программы.

4. Тестирование: проверка работоспособности всех функций.

5. Оптимизация: улучшение производительности и пользовательского интерфейса.

6. Документирование: создание пояснительной записки и комментариев к коду.

**Тестирование**

**Цель тестирования**

Цель тестирования заключается в проверке корректности работы всех функций менеджера задач и удобства пользовательского интерфейса.

**Методика тестирования**

Тестирование проводилось методом "черного ящика", где проверялись функциональные возможности приложения при различных сценариях использования.

**Тестирование**

1. Модульное тестирование: проверка работы отдельных функций.

2. Интеграционное тестирование: проверка взаимодействия между различными компонентами программы.

3. Функциональное тестирование: проверка соответствия программы заданным требованиям.

4. Тестирование пользовательского интерфейса: проверка удобства использования и отзывчивости интерфейса.

5. Проверка добавления, редактирования и удаления задач.

6. Проверка работы категорий и сортировки задач.

7. Проверка корректности работы с датами и временем.

8. Проверка сохранения и загрузки данных.

**Результаты тестирования**

Были выявлены и исправлены ошибки в логике работы с датами и временем, а также улучшен пользовательский интерфейс для более удобного взаимодействия с приложением.

**Заключение**

Разработанная программа полностью соответствует поставленным задачам и обеспечивает эффективное управление задачами пользователя. Программа имеет удобный пользовательский интерфейс и возможность настройки параметров задач.

**Источники, использованные при разработке**

1. Лутц М. Изучаем Python. – СПб.: Символ-Плюс, 2011.

2. "Документация Tkinter" [Электронный ресурс] // Официальная документация Python: https://docs.python.org/3/library/tkinter.html – Режим доступа: свободный.

3. Саммерфилд М. Python на практике. – М.: ДМК Пресс, 2014.

4. ГОСТ 19.701-90 «Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения». – М.: Издательство стандартов, 1990.

5. Бизли Д. Python. Подробный справочник. – СПб.: Символ-Плюс, 2010.