МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Сибирский государственный университет телекоммуникаций и информатики»

Кафедра телекоммуникационных систем и вычислительных средств (ТС и ВС)

ОТЧЕТ
по дисциплине
«Программирование»

по теме:
РГЗ

Студент:
Группа: ИКС - 431 *П.С. Волевач*

Предподаватель:
А.И. Вейлер

Новосибирск 2025 г.

СОДЕРЖАНИЕ

1 ОСНОВНАЯ ЧАСТЬ

1.1 Задание

Вариант 6. Шифрование текста шифром Цезаря
Задание

Разработать программу Cezar, выполняющую шифрование в заданном тексте и DeCezar – дешифровку текста. Текст до шифрования, после шифрования и после дешифровки должен выводиться на экран.

1.2 Анализируем задание и критерии оценки

Внимательно читаем:

1. Функционал:
   - Программа Cezar для шифрования.
   - Программа DeCezar для дешифровки.
   - Вывод текста: до шифрования, после шифрования, после дешифровки.
   - Шифр Цезаря: c = (m + k) mod n и m = (c - k) mod n.
   - k — ключ (смещение).
   - n — мощность алфавита.
   - Важно: Упоминается расширение алфавита (знаки препинания, заглавные, цифры).
2. Критерии оценки (это наша дорожная карта!):
   - ”Удовлетворительно”:
     - Проверка совпадения исходного и дешифрованного текста.
     - Нет динамического выделения памяти.
     - Функции в статической библиотеке.
   - ”Хорошо”:
     - Вход: 2 файла (первый - текст на русском, второй - для зашифрованного).

    - Обязательно динамическое выделение памяти.
    - Функции в статической библиотеке.
- ”Отлично”:
    - Оценить криптостойкость шифра.
    - Обязательно динамическое выделение памяти.
    - Функции в динамической библиотеке.

Вывод из анализа: Чтобы получить **”отлично”**, нам нужно сфокусироваться на: **динамическом выделении памяти** (это критично для больших файлов и является обязательным для ”хорошо”и ”отлично”), **работе с файлами** (чтение из одного, запись в другой), **русском языке** (это означает, что нужно будет работать с широкими символами (‘wchar_t‘) и корректной локалью (‘setlocale‘) для поддержки UTF-8), **динамической библиотеке** (это требование для ”отлично вместо статической), **криптостойкости** (это самая ”сложная”часть для ”отлично которую нужно будет продумать отдельно), и **тестировании** (хотя явно не прописано в критериях, автоматическая проверка совпадения текста (пункт ”удовлетворительно”) идеально реализуется через модульные тесты. Это также поможет отладить алгоритм шифрования/дешифрования).

—

1.3 Планируем структуру проекта

Разделим проект на логические части сразу, ориентируясь на динамическую библиотеку и тесты.

Папка ‘cezar/‘ будет содержать нашу библиотеку: ‘cezar.h‘ — заголовочный файл с объявлениями функций шифрования, дешифрования, чтения/-записи, и ‘cezar.c‘ — реализация этих функций. Папка ‘app/‘ будет содержать основную программу, которая использует библиотеку: ‘main.c‘ — логика взаимодействия с пользователем и файлами. Папка ‘tests/‘ будет содержать тесты для нашей библиотеки: ‘test_cezar.h‘ — объявления тестовых функций, ‘test_cezar.c‘ — реализация самих тестов (используя cezar.h), и ‘run_tests.c‘ — точка входа для запуска всех тестов. Также будут использоваться файлы ‘CMakeLists.txt‘ (корневой и в каждой подпапке) для автоматизации сборки.

2 ТЕСТОВЫЕ ДАННЫЕ И СКРИНШОТЫ

2.1 Тестовые данные

- Тест: Шифрование русских заглавных
  - Исходный текст: ”ПРИВЕТ”
  - Ожидаемый зашифрованный текст (с ключом 3): ”ТУЛЕИХ”
- Тест: Шифрование русских строчных
  - Исходный текст: ”привет”
  - Ожидаемый зашифрованный текст (с ключом 3): ”тулеих”
- Тест: Шифрование английских заглавных
  - Исходный текст: ”HELLO”
  - Ожидаемый зашифрованный текст (с ключом 3): ”KHOOR”
- Тест: Шифрование английских строчных
  - Исходный текст: ”hello”
  - Ожидаемый зашифрованный текст (с ключом 3): ”khoor”
- Тест: Неалфавитные символы при шифровании
  - Исходный текст: ”123!#$%&()”
  - Ожидаемый зашифрованный текст (с ключом 3): ”123!#$%&ˆ×()”(неалфавитные символы остаются без изменений)

—

- Тест: Дешифрование русских заглавных
  - Исходный (зашифрованный) текст: ”ТУЛЕИХ”
  - Ожидаемый дешифрованный текст (с ключом 3): ”ПРИВЕТ”
- Тест: Дешифрование русских строчных
  - Исходный (зашифрованный) текст: ”тулеих”
  - Ожидаемый дешифрованный текст (с ключом 3): ”привет”
- Тест: Дешифрование английских заглавных
  - Исходный (зашифрованный) текст: ”KHOOR”

- Ожидаемый дешифрованный текст (с ключом 3): ”HELLO”

– Тест: Дешифрование английских строчных
  - Исходный (зашифрованный) текст: ”khoor”
  - Ожидаемый дешифрованный текст (с ключом 3): ”hello”

– Тест: Дешифрование смешанного текста
  - Исходный (зашифрованный) текст: ”Тулеих, Zruog!”
  - Ожидаемый дешифрованный текст (с ключом 3): ”Привет, World!”

2.2 Результаты тестов и работы программ

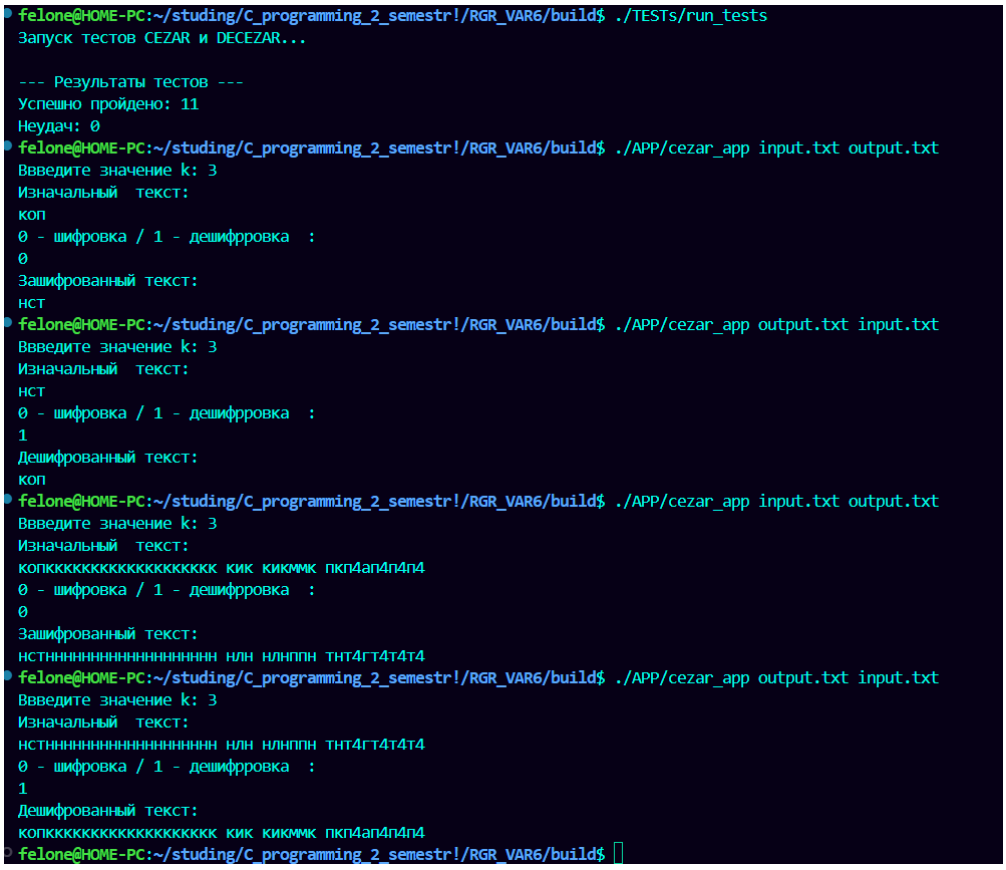

Рисунок 1 — Результаты

2.3 Ссылка на github

github