

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 РГР:
”СОЗДАНИЕ ПРИМИТИВНОГО ПЕРЕВОДЧИКА С РУССКОГО НА
НЕМЕЦКИЙ”

Студент:
Группа: ИКС - 431

С.О. Бобокулов

Предподаватель:

А. И. Вейлер

Новосибирск 2025 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	3
1 ОСНОВНАЯ РАБОТА	4
ЗАКЛЮЧЕНИЕ	7

ВВЕДЕНИЕ

В данной работе разработан переводчик с русского на китайский язык. Программа принимает три файла:

- Исходный текст на русском языке (Russian.txt)
- Файл словаря (dict.txt) в формате ”русское слово немецкое слово”
- Файл для вывода перевода (Germany.txt)

Особенности реализации:

- Поддержка UTF-8 для русских и китайских символов
- Динамическое выделение памяти под словарь
- Сохранение исходного форматирования и пунктуации

1 ОСНОВНАЯ РАБОТА

```
typedef struct {  
    char Russian[50];  
    char Germany[50];  
} Translation;
```

Рисунок 1 — Объявление структуры

```
// Объявляю функцию для загрузки словаря  
int LoadDict (const char *filename, Translation **dict, int *size) {  
    FILE *fp = fopen(filename, "r"); // Открываю файл словаря с ключём для чтения  
    if (!fp) {  
        printf("Ошибка, не удалось открыть файл словаря %s\n", filename); // Если не уда  
        return -1;  
    }  
  
    *size = 0; // размер словаря  
    *dict = realloc(*dict, sizeof(Translation)); // Выделяю память под один элемент стру  
    if (!*dict) {  
        fclose(fp); // Если не удалось выделять память закрываю файл  
        return -1;  
    }  
  
    while (fscanf(fp, "%s %s", (*dict)[*size].Russian, (*dict)[*size].Germany) == 2) { //  
        (*size)++; // Увеличиваю счётчик  
        *dict = realloc(*dict, (*size + 1)*sizeof(Translation)); //Перевыделяю память дл  
        if (!*dict) {  
            fclose(fp); // Если не удалось перевыделит память закрываю файл  
            return -1;  
        }  
    }  
  
    fclose(fp); //Закрываю файл после чтения  
    return 0;  
}
```

Рисунок 2 — Функция загрузки файлов

```

// Объявляю функцию перевода текста
int TranslateFile (const char *input, const char *output, Translation *dict, int size) {
    FILE *in = fopen(input, "r"); // Открываю входной файл для чтения
    FILE *out = fopen(output, "w"); // Открываю выходной файл для записи
    if (!in || !out) { // Если не удалось открыть файлы вывожу ошибку и закрываю файлы
        printf("Ошибка, не удалось открыть один из файлов\n");
        if (!in) fclose(in);
        if (!out) fclose(out);
        return -1;
    }
    char word[50]; // Объявляю переменную для текущего слова
    while (fscanf(in, "%s ", word) == 1) { // Читаю слова из входного файла по одному
        int found = 0; // Объявляю переменную сигнализирующую найдено слово или нет
        for (int i = 0; i < size; i++) { // Прохожу по всем элементам словаря
            if (strcmp(word, dict[i].Russian) == 0) { // Если текущее слово == русское слов
                fprintf(out, "%s ", dict[i].Germany);
                found = 1; // Слово найдено
                break;
            }
        }

        if (!found) { // Если слово не найдено записываю само слово
            fprintf(out, "%s ", word);
        }
    }

    fclose(in); //Закрываю входной файл
    fclose(out); //Закрываю выходной файл
    return 0;
}

```

Рисунок 3 — Функция перевода

```

makefile

m_required(VERSION 3.10) # Указываю минимальную версию
slater) # Определяю имя проекта

STANDARD 11) # Устанавливаю стандарт C

translate STATIC translate.c) # Создаю статическую б

le(translater main.c) #Создаю исполняющий файл и свя
libraries(translater PRIVATE translate)

(CMocka REQUIRED) # Ищу пакет CMocka для юнит тестов
le(test_translate test_translate.c) # Создаю исполня
libraries(test_translate PRIVATE translate cmocka) #

```

Рисунок 4 — CMake файл для сборки проекта

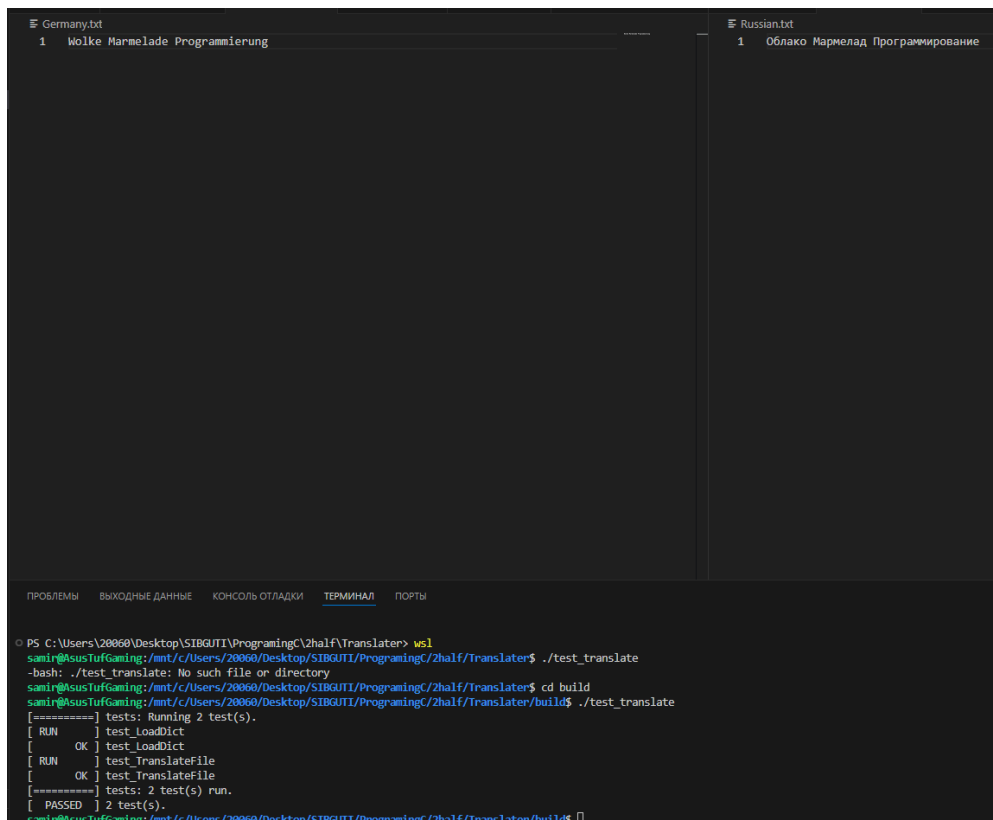


Рисунок 5 — Вывод перевода

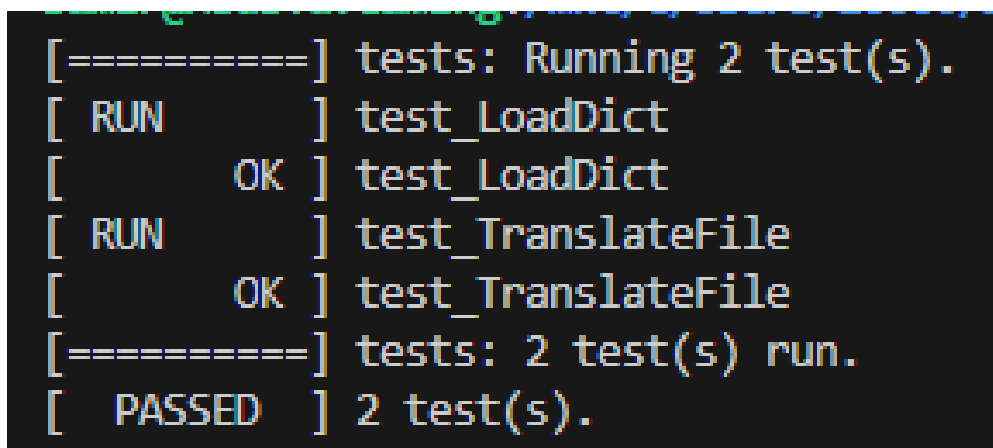


Рисунок 6 — Тестирование

ЗАКЛЮЧЕНИЕ

Разработанная программа соответствует всем критериям задания:

- Реализована корректная работа с UTF-8
- Обеспечено динамическое выделение памяти
- Сохранено исходное форматирование текста
- Поддержана сборка через CMake