МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования «Сибирский государственный университет телекоммуникаций и информатики» (СибГУТИ)

Расчетно-графическая работа по дисциплине «Программирование»

Вариант 6

Студент: Группа ИКС-433 Синица М.А.

Преподаватель: Вейлер А.И.

Новосибирск 2025

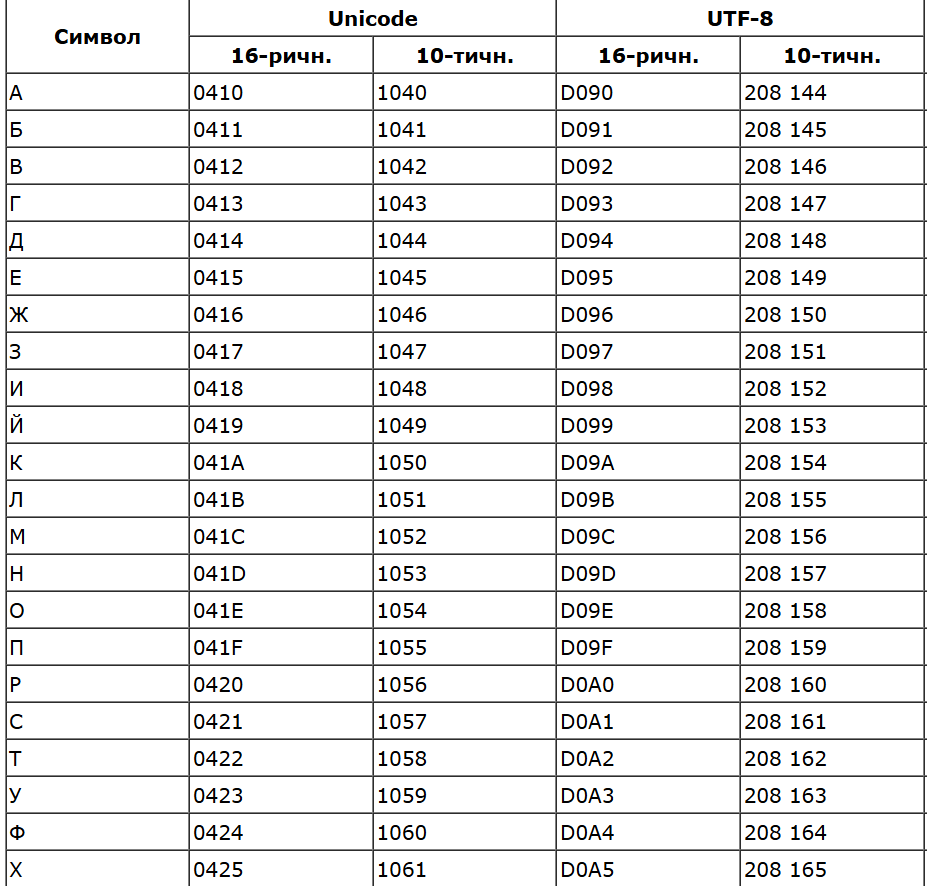
**Задание:**

Разработать программу Cezar, выполняющую шифрование в заданном тексте и DeCezar – дешифровку текста. Текст до шифрования, после шифрования и после дешифровки должен выводиться на экран. На вход программы подается 2 файла. Первый файл содержит текст на русском языке. Второй файл будет содержать зашифрованный текст. Обязательно динамическое выделение памяти под входные данные. Функции записаны в статическую библиотеку.

**Методы решения задачи:**

Шифр цезаря: циклический сдвиг букв алфавита на заданное число позиций

Для русских букв будем использовать кодировку, UTF-8 в которой русские буквы кодируются 2 байтами.



Псевдокод шифра цезаря:

Для каждого символа в тексте:

Если это русская буква:

Определить её позицию в алфавите

Применить сдвиг (позиция + ключ) mod 32

Заменить байты на новые

Иначе:

Оставить без изменений

На вход программе будет подаваться 4 аргумента строки. Второй аргумент имя файла с исходным текстом, третий аргумент имя файла в который нужно записать зашифрованный текст, четвертый аргумент — это ключ для сдвига.

Будем побайтово считывать текст и через кодировку UTF-8 проверять является ли символ русской буквой, далее проверяем заглавная буква или строчная. Далее будем определять её позицию, применять сдвиг и записывать новые байты. Далее выводим результат в командную строку и записываем зашифрованный текст в файл.

В случае дешифровки алгоритм тот же, только внутри алгоритма ключ будет не прибавляться а вычитаться.

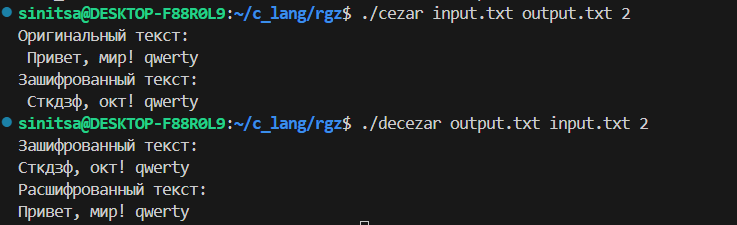
**Текстовые данные:**

Вход : Привет, мир!

Выход: Сткдзф, окт!

Вход : qwerty

Выход: qwerty



**Исходные коды:**

**Cheasar.h**

void encrypt(char \*text, int key);

void decrypt(char \*text, int key);

char \*read\_file(const char \*filename);

void write\_file(const char \*filename, const char \*text);

**main\_cezar.c:**

#include <stdio.h>

#include <locale.h>

#include "caesar.h"

int main(int argc, char \*argv[]) {

    setlocale(LC\_ALL, "ru\_RU.UTF-8");

    if (argc != 4) {

        printf("Использование: входной\_файл выходной\_файл ключ\n");

        return 1;

    }

    char \*in = read\_file(argv[1]);

    if (!in) {

        printf("Ошибка чтения файла %s\n", argv[1]);

        return 1;

    }

    int key = atoi(argv[3]);

    printf("Оригинальный текст:\n %s\n", in);

    encrypt(in, key);

    printf("Зашифрованный текст:\n %s\n", in);

    write\_file(argv[2], in);

    free(in);

    return 0;

}

**Main\_decezar.c:**

#include <stdio.h>

#include <locale.h>

#include "caesar.h"

int main(int argc, char \*argv[]) {

    setlocale(LC\_ALL, "ru\_RU.UTF-8");

    if (argc != 4) {

        printf("Использование: %s <входной\_файл> <выходной\_файл> <ключ>\n", argv[0]);

        return 1;

    }

    char \*in = read\_file(argv[1]);

    if (!in) {

        printf("Ошибка чтения файла: %s\n", argv[1]);

        return 1;

    }

    int key = atoi(argv[3]);

    printf("Зашифрованный текст:\n %s\n", in);

    decrypt(in, key);

    printf("Расшифрованный текст:\n %s\n", in);

    write\_file(argv[2], in);

    free(in);

    return 0;

}

**Cheasar.c:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "caesar.h"

void caesar(char \*text, int key) {

    for (int i=0; text[i];){

        unsigned char f = (unsigned char)text[i];

        unsigned char s = (unsigned char)text[i+1];

        if ((f == 0xD0 && s >= 0x90 && s <= 0xBF && s != 0x81)||(f == 0xD1 && s >= 0x80 && s <= 0x8F && s != 0x91)){

            int upper =(f == 0xD0 && s <= 0xAF);

            int index;

            if (upper) {

                index = s - 0x90;

                index = (index+ key+32)%32;

                text[i] = 0xD0;

                text[i+1] = 0x90 + index;

            } else {

                if (f == 0xD0) {

                    index = s - 0xB0;

                } else {

                    index = s - 0x80 + 16;

                }

                index = (index + key + 32) % 32;

                if (index < 16) {

                    text[i] = 0xD0;

                    text[i+1] = 0xB0 + index;

                } else {

                    text[i] = 0xD1;

                    text[i+1] = 0x80 + (index - 16);

                }

            }

            i+=2;

        } else {

            i++;

        }

    }

}

void encrypt(char \*text, int key){

    caesar(text, key);

}

void decrypt(char \*text, int key){

    caesar(text, -key);

}

char \*read\_file(const char \*filename) {

    FILE \*f = fopen(filename, "rb");

    if (!f) return NULL;

    fseek(f, 0, SEEK\_END);

    long size = ftell(f);

    rewind(f);

    char \*buffer = malloc(size + 1);

    if (!buffer) {

        fclose(f);

        return NULL;

    }

    fread(buffer, 1, size, f);

    buffer[size] = '\0';

    fclose(f);

    return buffer;

}

void write\_file(const char \*filename, const char \*text) {

    FILE \*f = fopen(filename, "wb");

    if (f) {

        fwrite(text, 1, strlen(text), f);

        fclose(f);

    }

}

Список литературы:

1. http://blog.kislenko.net/show.php?id=2045