Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Санкт-Петербургский политехнический университет Петра Великого»

—

Институт компьютерных наук и кибербезопасности

**Высшая школа кибербезопасности**

**ЛАБОРАТОРНАЯ РАБОТА № 1**

«**Работа с файлами**»

по дисциплине «Структуры данных»

Выполнили

студенты гр. 5131001/30002 Мишенев Н. С.

Квашенникова В. М.

<*подпись*>

Руководитель

программист Вагисаров В. Б.

<*подпись*>

Санкт-Петербург

2024г.

СОДЕРЖАНИЕ

[ЦЕЛЬ РАБОТЫ 3](#_Toc160748129)

[ХОД РАБОТЫ 4](#_Toc160748130)

[ВЫВОД 6](#_Toc160748131)

[ПРИЛОЖЕНИЕ А 7](#_Toc160748132)

ЦЕЛЬ РАБОТЫ

Реализовать программу, осуществляющую замену слов в тексте и вывод полученного текста в выходной файл.

ХОД РАБОТЫ

1. **Подключение необходимых библиотек и определение констант**

В программе будут использованы библиотеки stdio.h, stdlib.h, string.h. Определены константы: WORDLEN (длина слова), PATHLEN (максимальная длина пути к файлу), STRLEN (максимальная длина строки)

1. **Определение функции clear и get\_n\_word**

Функция clear используется для обнуления содержимого строки. Она принимает указатель на строку str и длину этой строки strlen.

Функция get\_n\_word используется для записи в строку слова, содержащегося в файле.

1. **Загрузка файлов**

На первом этапе процесса программы происходит получение путей к файлам из параметров командной строки, с которыми вызывалась программа, а затем определение соответствующих потоков для чтения и записи данных.

1. **Чтение данных из файлов**

После получения путей к файлам программа начинает чтение содержимого. С помощью цикла программа определяет количество слов, подлежащих замене, и динамически выделяет память для создания двух массивов строк: один содержит слова, которые будут заменены, а другой - слова, на которые нужно заменить. Полученные массивы заполняются данными из соответствующих файлов. Затем происходит посимвольное чтение текста из файла и формирование строки, содержащей считываемое на данный момент слово.

1. **Сравнение и замена**

При обнаружении символа алфавита программа начинает формировать слово, считывая последующие символы, и сохраняет его в строку word\_compare.

Когда слово завершено (встречен символ, не являющийся буквой), программа сравнивает его с каждым словом из массива words\_comparewith.

Если слово совпадает с одним из слов в words\_comparewith, оно заменяется соответствующим словом из words\_toput. В противном случае слово остается без изменений.

Результат записывается в файл file\_4.

1. **Заключительные работы и тестирование**

После завершения чтения и обработки данных необходимо закрыть все открытые файлы с помощью функции fclose() и освободить выделенную динамическую память с помощью функции free().

Программа тестировалась на различных наборах данных, чтобы убедиться в ее корректности и правильности работы.

ВЫВОД

В ходе работы были получены навыки работы с файлами и аргументами командной строки, была реализована программа, которая выполняет замену слов в тексте и выводит полученный текст в выходной файл.

ПРИЛОЖЕНИЕ А

Листинг 1 - исходный код программы на языке С

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define WORDLEN 100

#define PATHLEN 260

#define STRLEN 50

void clear(char\* str, int strlen) {

for (int i = 0; i < strlen; i++) {

str[i] = '\0';

}

}

void get\_n\_word(FILE\* file, char\* word) {

int count\_slash = 0;

int j = 0;

char c;

for (int i = 0;; i++) {

c = fgetc(file);

if ((c == '\n') || (feof(file) != 0)) {

count\_slash++;

break;

}

if (count\_slash == 0) {

word[j++] = c;

}

}

word[j] = '\0';

}

int main(void)

{

char name\_1[PATHLEN]={0}, name\_2[PATHLEN] = { 0 }, name\_3[PATHLEN] = { 0 }, name\_4[PATHLEN] = { 0 };

FILE\* file\_1 = fopen(argv[1], "r"), \* file\_2 = fopen(argv[2], "r"), \* file\_3 = fopen(argv[3], "r"), \* file\_4 = fopen(argv[4], "w");

if (file\_1 == NULL || file\_2 == NULL || file\_3 == NULL || file\_4 == NULL) {

printf("Error\n");

return 1;

}

int words\_count = 1;

while (1)

{

if (fgetc(file\_2) == '\n') words\_count++;

if (feof(file\_2) != 0) break;

}

fseek(file\_2, 0, SEEK\_SET);

char\*\* words\_comparewith = (char\*\*)malloc(words\_count \* WORDLEN \* sizeof(char));

for (int i = 0; i < words\_count; i++) {

char\* word = (char\*)malloc(sizeof(char) \* WORDLEN);

clear(word, WORDLEN-1);

get\_n\_word(file\_2, word);

words\_comparewith[i] = word;

}

char\*\* words\_toput = (char\*\*)malloc(words\_count \* WORDLEN \* sizeof(char));

for (int i = 0; i < words\_count; i++) {

char\* word = (char\*)malloc(sizeof(char) \* WORDLEN);

clear(word, WORDLEN);

get\_n\_word(file\_3, word);

words\_toput[i] = word;

}

int flag = 0,i=0;

char word\_compare[WORDLEN],c;

while (1)

{

c = fgetc(file\_1);

if (('a' <= c && c<= 'z' ) || ('A' <= c && c <= 'Z'))

{

flag = 1;

word\_compare[i] = c;//начало слова

i++;

}

else

{

if (flag == 1)

{

flag = 0;

word\_compare[i] = '\0';

int replaced = 0;

for (int j = 0; j < words\_count; j++)

{

if (strcmp(word\_compare, words\_comparewith[j]) == 0)

{

fputs(words\_toput[j], file\_4);

replaced = 1;

break;

}

}

if (replaced==0) fputs(word\_compare, file\_4);

}

if (feof(file\_1) != 0) break;

fputc(c, file\_4);

i = 0;

}

}

fclose(file\_1);

fclose(file\_2);

fclose(file\_3);

fclose(file\_4);

for (int i = 0; i < words\_count; i++) {

free(words\_comparewith[i]);

free(words\_toput[i]);

}

free(words\_comparewith);

free(words\_toput);

return 0;

}