Изображение выглядит как текст, небо, внешний

Автоматически созданное описание

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**   
**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**   
**імені ІГОРЯ СІКОРСЬКОГО»**

**Інститут прикладного системного аналізу**   
**Кафедра системного проектування**  
   
   
 

**Звіт**

**про виконання лабораторної роботи №13**   
**«****Текстові файли** **в С»**

Виконав:   
студент I курсу, групи ДА-12   
Панасенко Олександр Андрійович

Прийняв:

к.т.н., доцент Безносик О. Ю.

**Мета роботи**

Вивчити особливості використання текстових файлів, навчитися застосовувати текстові файли в програмуванні.

**Завдання**

Створити текстовий файл, вводячи дані з клавіатури. У кожному рядку тексту знайти слово максимальної довжини і записати знайдені слова та їх довжини у новий текстовий файл.

**Аналіз умови задачі.**

Виходячи з умови задачі, я створив функцію для запису вводу тексту з клавіатури користувача до текстового файлу, а також функцію, що буде шукати найдовше слово у кожному рядку тексту. Для реалізації останньої функції, мені знадобилась допоміжна функція для запису рядка з найдовшим словом та його довжиною у новий текстовий файл.

Ввод користувача я опрацьовував функцією fgets(), адже кожного разу користувач прагне записати рядок у файл, а за допомогою функції fputs(), записував текст у файл.

**Код програми**

#include <stdio.h>

#include <string.h>

#include <ctype.h>

#define TRUE 1

#define FALSE 0

*/\**

*10 - linefeed - used to switch to a new line*

*13 - carriage return - used to reset a device's position to the beginning of a line of text*

*26 - substitute - EOF*

*\*/*

void pflush(const char \*str)

{

puts(str);

fflush(stdin);

}

// flush after scanf

// in order for puts to work correctly

void \_flush\_scanf()

{

int f = 0;

while ((f = getchar()) != '**\n**' && f != EOF); // <- after use of scanf() there is linefeed left in stdin buffer, so we gotta clear it

}

FILE\* restrict readbuffer;

FILE\* restrict writebuffer;

char filename[16];

char words\_filename[16];

void write\_file(FILE\* restrict writebuffer, const char\* filename);

void read\_file(FILE\* restrict readbuffer, const char\* filename);

void write\_max\_word\_file(FILE\* restrict writebuffer, FILE\* restrict readbuffer, const char\* \_newfilename, const char\* filename);\

// writebuffer should be closed or segfault will be throwed

// will open writebuffer again with mode "a" and close at the end of function

void write\_line(FILE\* restrict writebuffer, const char\* filename, const char\* line);

// converts string to lowercase not depending on '\0'

// returns a pointer to the same part of memory

char\* strlower(char\* str, size\_t len);

int main(void)

{

pflush("[LOG] Enter the name of the file:");

char \_filepath[32] = "../data/13/"; // where data of 13-text-files-c will be saved

scanf("%s", &filename);

\_flush\_scanf();

strcat(\_filepath, filename);

pflush("[LOG] Creating blank file... Listening for input:");

write\_file(writebuffer, \_filepath);

pflush("[LOG] Reading a file... Result:");

read\_file(readbuffer, \_filepath);

pflush("[LOG] Enter the name of the second file:");

char \_wfilepath[32] = "../data/13/";

scanf("%s", &words\_filename);

\_flush\_scanf();

strcat(\_wfilepath, words\_filename);

write\_max\_word\_file(writebuffer, readbuffer, \_wfilepath, \_filepath);

}

void write\_max\_word\_file(FILE\* restrict writebuffer, FILE\* restrict readbuffer, const char\* \_newfilename, const char\* filename)

{

// should check if the name is correct

readbuffer = fopen(filename, "r");

char \_rline[128];

char delim[] = " ,.;:**\"\n**!?"; // <- \n will be recorded by fgets, so we are forced to check and change it

char\* restrict \_wpointer;

char target\_word[128];

size\_t wlen;

while(TRUE)

{

fgets(\_rline, sizeof(\_rline), readbuffer); // get next line

if (feof(readbuffer))

break;

// split word

\_wpointer = strtok(\_rline, delim); // get first word

// size\_t \_actsize\_tmp = strlen(\_wpointer);

wlen = strlen(\_wpointer); // check if this is the only word in line and \n is next

strcpy(target\_word, \_wpointer);

// get all words and check them

while (\_wpointer != NULL)

{

size\_t curr\_wlen = strlen(\_wpointer);

if (wlen <= curr\_wlen)

{

wlen = curr\_wlen;

strcpy(target\_word, \_wpointer);

}

\_wpointer = strtok(NULL, delim);

}

char len[4]; // will convert wlen to char\* with sprintf

sprintf(len, " %d**\n**", wlen);

strcat(target\_word, len);

write\_line(writebuffer, \_newfilename, target\_word);

}

fclose(readbuffer);

}

void write\_file(FILE\* restrict writebuffer, const char\* filename)

{

writebuffer = fopen(filename, "w"); // create new file for write and read

char answ;

char line[128];

pflush("Keep entering strings (type EXIT to stop entering):");

while (TRUE) {

*/\**

*Note: The gets() function can also be to take input from the user. However, it is removed from the C/CXX standard.*

*It's because gets() allows you to input any length of characters. Hence, there might be a buffer overflow.*

*\*/*

fgets(line, sizeof(line), stdin); // <- \n will be recorded by fgets, so we are forced to check for exit\n underneath

// line[sizeof(line) - 1] = '\0'; // < for strlower() to work correctly

if (strcmp(strlower(line, sizeof(line)), "exit**\n**") == 0) // 0 if equal

break;

fputs(line, writebuffer);

}

// fputs("\0", writebuffer); // why? odd

fclose(writebuffer);

}

void read\_file(FILE\* restrict readbuffer, const char\* filename)

{

readbuffer = fopen(filename, "r");

if (readbuffer == NULL)

{

pflush("failed to open a file. Returning");

return;

}

char ch;

char line[128];

while (TRUE)

{

fgets(line, sizeof(line), readbuffer);

if (feof(readbuffer)) // we should check EOF here as the line is updated so that it will try to print EOF

break;

printf("%s", line);

}

fclose(readbuffer);

}

void write\_line(FILE\* restrict writebuffer, const char\* filename, const char\* line)

{

writebuffer = fopen(filename, "a");

if (writebuffer == NULL)

{

pflush("no writebuffer found. Returning");

return;

}

fputs(line, writebuffer);

fclose(writebuffer);

}

char\* strlower(char\* str, size\_t len)

{

for (int i = 0; i < len; i++)

str[i] = tolower(str[i]);

return str;

}

Результати виконання програми зображено на рисунках 1,2,3.

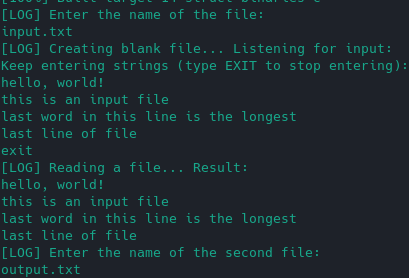


Рисунок 1

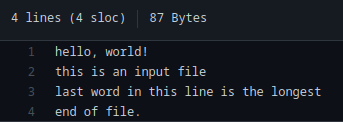


Рисунок 2. Input.txt

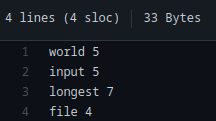
****

Рисунок 3. Output.txt

**Висновки**

У ході виконання лабораторної роботи №13 я користувався типом данних FILE та функціями, що відносяться до нього, такі як fputs(), fgets(), fopen(). Для ділення речення на слова я використовував функцію strtok(). Найдовше з отриманих слів я зберігав в окремий текстовий документ, додаючи до нього його довжину.