ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИСCЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Факультет безопасности информационных технологий

**Дисциплина:**

«Программирование»

**ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3**

**Вариант 8-2-3-3**

**Выполнил:**

Студент гр. N3151, Мочеков Семён Сергеевич



**Проверил:**

Гирик Алексей Валерьевич

Санкт-Петербург

2023

1. **Задание.**

Составить программу, находящую в последовательностях строк, полученных из файла или ввода с клавиатуры, вхождения даты и времени в формате:

HH:MM:SS dd.mm.yyyy

и модифицирующую разметкой Markdown Код: `текст`.

Так же поддерживающую опцию -с, при указании которой разметка Markdown заменяется на выделение текста жёлтым цветом, в соответствие управляющей последовательности ANSI: \e[33mтекст\e[m – текст.

Указание опции -v выводит ФИО и группу студента, составившего программу, и номер варианта.

Поддерживаемая опция -n модифицирует только те вхождения, что находятся в пределах одной строки.

Дополнительные опции -f и -t задают номера вхождений, с которых надо начать модификацию и закончить её соответственно.

1. **Makefile**

.PHONY: all clean

APP=lab3ssmN3151

CFLAGS=-Wall -Wextra -Werror -g

all: $(APP)

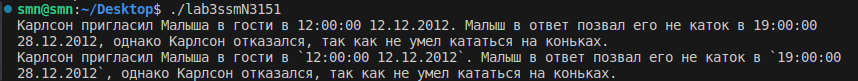
$(APP):

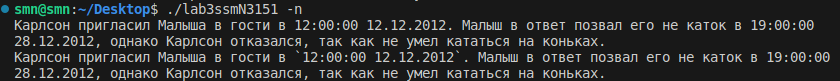
gcc -o $(APP) $(CFLAGS) $(APP).c

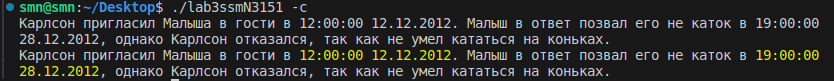
clean:

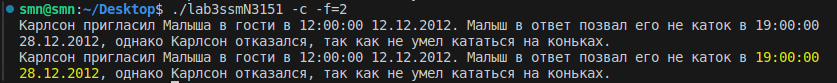
rm $(APP)

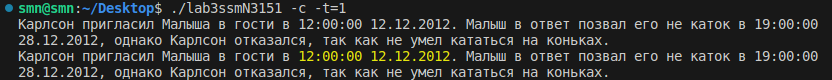
1. **Работа алгоритма**





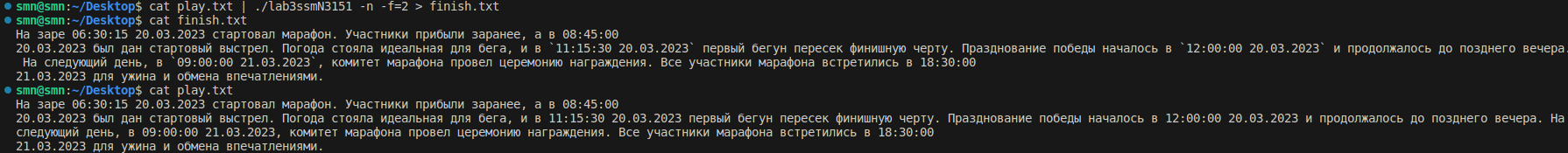






Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Автоматически созданное описание



Изображение выглядит как текст, снимок экрана, диаграмма, Шрифт

Автоматически созданное описаниеИзображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

**IV. Исходный код алгоритма**

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <ctype.h>

#include <stdbool.h>

#include <errno.h>

enum exit\_codes

{

OK = 0,

WRONG\_FLAG = 1,

FILE\_ERROR = 2,

TYPE\_ERROR = 3,

MALLOC\_ERROR = 4

};

typedef struct

{

bool n;

bool c;

unsigned long f;

unsigned long t;

} flags;

flags flag = {false, false, 1, -1};

FILE \*input\_file = NULL, \*output\_file = NULL;

bool is\_num(char \*str)

{

char \*end\_of\_str;

long val = strtol(str, &end\_of\_str, 10);

if (\*end\_of\_str == '\0' && val >= 0) {

return true;

}

return false;

}

bool validation(char \*pattern, char \*str)

{

size\_t len = strlen(pattern);

for (size\_t i = 0; i < len; i++) {

if (pattern[i] == 'd') {

if (isdigit(str[i]) == 0) {

return false;

}

}

else if (pattern[i] != str[i]) {

return false;

}

}

return true;

}

bool is\_date(char \*str)

{

if (!validation("dd:dd:dd dd.dd.dddd", str))

return false;

int day, month, year;

int seconds, minutes, hours;

sscanf(str, "%d:%d:%d %d.%d.%d", &hours, &minutes, &seconds, &day, &month, &year);

if (day < 1 || day > 31 || month < 1 || month > 12 || year < 1 || year > 9999 || hours > 23 || minutes > 59 || seconds > 59)

return false;

if (day > 30 && (month == 4 || month == 6 || month == 9 || month == 11 || month == 2))

return false;

if (month == 2 && day > 29)

return false;

if (month == 2 && day == 29 && !(year % 4 == 0 && (year % 100 != 0 || year % 400 == 0)))

return false;

return true;

}

void flag\_check(int argc, char \*\*argv)

{

char \*temp = (char \*)malloc(256);

if (!temp) {

perror("Не удалось выделить память");

exit(MALLOC\_ERROR);

}

for (int i = 1; i < argc; i++) {

if (strcmp(argv[i], "-v") == 0) {

printf("Семён, N3151\nВариант: 8-2-3-3\n");

exit(OK);

}

else if (strcmp(argv[i], "-n") == 0) {

flag.n = true;

}

else if (strcmp(argv[i], "-c") == 0) {

flag.c = true;

}

else if (argv[i][0] == '-' && argv[i][1] == 'f' && argv[i][2] == '=') {

strncpy(temp, argv[i] + 3, strlen(argv[i]) - 3);

if (is\_num(temp)) {

flag.f = strtol(temp, NULL, 10);

}

else {

perror("Неверный тип данных для -f.\n");

exit(TYPE\_ERROR);

}

}

else if (argv[i][0] == '-' && argv[i][1] == 't' && argv[i][2] == '=') {

strncpy(temp, argv[i] + 3, strlen(argv[i]) - 3);

if (is\_num(temp)) {

flag.t = strtol(temp, NULL, 10);

}

else {

perror("Неверный тип данных для -f.\n");

exit(TYPE\_ERROR);

}

}

else if (input\_file == NULL) {

input\_file = fopen(argv[i], "r");

if (input\_file == NULL) {

perror("Не удалось открыть файл.\nПроверьте корректность указанного пути");

exit(FILE\_ERROR);

}

}

else if (output\_file == NULL) {

output\_file = fopen(argv[i], "w+");

if (output\_file == NULL) {

perror("Некая проблема c выводным файлом.\nПередайте это программисту");

exit(FILE\_ERROR);

}

}

else {

printf("Неподходящие аргументы запуска программы!\n");

exit(WRONG\_FLAG);

}

}

free(temp);

}

int main(int argc, char \*\*argv)

{

flag\_check(argc, argv);

if (input\_file == NULL) {

input\_file = fopen("tmp", "w+");

if (input\_file == NULL) {

perror("Ошибка при создании временного файла");

exit(FILE\_ERROR);

}

int chars;

while ((chars = getchar()) != EOF) {

fputc(chars, input\_file);

}

rewind(input\_file);

}

unsigned long long file\_size;

fseek(input\_file, 0, SEEK\_END);

file\_size = ftell(input\_file) + 1;

fseek(input\_file != NULL ? input\_file : stdin, 0, SEEK\_SET);

char \*buffer = (char \*)malloc(file\_size), \*pattern\_buffer = (char \*)malloc(19);

if (!buffer || !pattern\_buffer) {

perror("Не удалось выделить память");

exit(MALLOC\_ERROR);

}

fread(buffer, 1, file\_size, input\_file);

unsigned long objects = 0;

unsigned long long i = 0;

while (i < file\_size) {

strncpy(pattern\_buffer, buffer + i, 19);

if (!flag.n && pattern\_buffer[8] == '\n') {

pattern\_buffer[8] = ' ';

}

if (is\_date(pattern\_buffer) && (++objects >= flag.f ) &&(flag.t == (unsigned long long)-1 || objects <= flag.t)) {

strncpy(pattern\_buffer, buffer + i, 19);

fprintf(output\_file == NULL ? stdout : output\_file, flag.c ? "\e[33m%s\e[m" : "`%s`", pattern\_buffer);

i+=18;

}

else {

fprintf(output\_file == NULL ? stdout : output\_file, "%c", buffer[i]);

}

i++;

}

free(buffer);

free(pattern\_buffer);

if (input\_file != NULL) {

fclose(input\_file);

remove("tmp");

}

if (output\_file != NULL){

fclose(output\_file);

}

return 0;

}