Санкт-Петербургский государственный политехнический университет

Кафедра “Прикладная математика”

**Отчет по лабораторной работе 1**

**“Алгоритмы и структуры данных”**

Студент группы № 5030102/20001

ФИО: Тишковец Сергей Евгеньевич

Выполнил (дата) 12.10.2023

Оглавление

[**Постановка задачи** 3](#_Toc148007496)

[**Описание алгоритма** 3](#_Toc148007497)

[**Текст программы** 5](#_Toc148007498)

[**Описание тестирования** 10](#_Toc148007499)

# **Постановка задачи**

*Вариант 4*

Файл содержит следующую информацию (построчно): фамилия, имя, отчество. Считать файл в список таким образом, чтобы он оставался отсортирован по фамилиям (при совпадении фамилий - по именам; при совпадении и фамилий и имен - по отчествам). Вывести список фамилий на экран. Ввести с клавиатуры начальные буквы фамилии, имени и отчества и проверить, есть ли люди с такой фамилией в списке.

Пример:

1. Ivanov Ivan Ivanovich
2. Petrov Ivan Petrovich

Запрос «I» «I» «» выдает первую фамилию в списке

Запрос «» «Iv» «» выдает обе фамилии в списке

# **Описание алгоритма**

Организация списка:

Сначала требовалось организовать список. Я реализовал его c помощью структуры. Структура представляет собой узел из фамилии, имени, отчества и указателя на следующий узел.

Функция для получения узла списка из массива:

1. Сначала считываются фамилия, имя и отчество в отдельные массивы.
2. Затем создается новый узел с помощью выделения памяти функцией malloc.
3. Далее идет проверка на успешность выделения памяти. В случае нехватки программа выводит сообщение об ошибке.
4. В конце вызывается функция для заполнения узла полученными данными.

Функция для вывода списка на экран:

Пробегается по всему списку и печатает фамилию, имя и отчество через пробел.

Функция для печати запроса пользователя:

1. Вызывается функция для сравнения элементов списка.
2. В случае наличия совпадений функция печатает на экране фамилию, имя и отчество через пробел.
3. Возвращает количество совпадений.

Функция для добавления элементов в список в алфавитном порядке:

1. С помощью функции strcmp побайтно сравниваются коды символов двух строк – той, которая уже есть в списке, и той, которую нужно добавить.
2. Сравнение начинается с фамилии. Затем, в случае совпадения фамилий, сравнивают имена и, аналогично, отчества.
3. Возвращает значение, соответствующее успешному добавлению элемента в список.

Функция для сравнения элементов списка:

1. Сначала вычисляется длина строки, которую ввел пользователь.
2. Далее идёт посимвольное сравнение элементов строки, введённой пользователем, и элементов списка.
3. Сравнение начинается с фамилии, далее для имени и отчества.
4. Возвращает либо значение, соответствующее наличию полного совпадения, либо значение, соответствующее отсутствию совпадений.

Функция для очищения памяти:

Освобождается выделенная ранее память, начиная с первого узла и заканчивая последним.

# **Текст программы**

#pragma warning(disable: 4996)

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <ctype.h>

#include <stdlib.h>

#include <string.h>

#include <locale.h>

#define SIZE 20

#define NOTSAME 1

#define SAME 0

#define ERROR\_OPENING\_FILE -1

#define SUCCESS\_ADD 0

#define ADDED 2

//структура List, представляющая собой узел

typedef struct List {

char surname[SIZE];

char name[SIZE];

char father[SIZE];

struct List\* next; //указатель на следующий узел

} List;

//копирует данные из строки surname в элемент узла

void GetStr(List\* man, char\* surname, char\* name, char\* father) {

strcpy(man->surname, surname);

strcpy(man->name, name);

strcpy(man->father, father);

}

//получает узел списка из массива

List\* TransformToList(char\* tmp) {

List\* result;

int i = 0, j = 0, k = 0;

//фамилия, имя, отчество

char surname[SIZE], name[SIZE], father[SIZE];

char q;

//считывание фамилии

while ((q = tmp[i]) != ' ') {

surname[i] = q;

i++;

}

//конец считывания

surname[i] = '\0';

i++;

//считывание имени

while ((q = tmp[i]) != ' ') {

name[k] = q;

i++;

k++;

}

//конец считывания

name[k] = '\0';

i++;

//считывание отчества

while ((q = tmp[i]) != '\0') {

father[j] = q;

i++;

j++;

}

//конец считывания

if (father[j - 1] == '\n' && j > 0) {

father[j - 1] = '\0';

}

else {

father[j] = '\0';

}

//создает новый узел

result = (List\*)malloc(sizeof(List));

//проверка успешного выделения памяти

if (result == NULL) {

printf("not enough memory");

return NULL;

}

GetStr(result, surname, name, father);

return result;

}

//печатает список

void PrintList(List\*\* top) {

List\* tmp = \*top;

while (tmp != NULL) {

//фамилия имя отчество (через пробел)

printf("%s ", tmp->surname);

printf("%s ", tmp->name);

printf("%s\n", tmp->father);

tmp = tmp->next; //переход к следующему узлу

}

}

//печатает запрос с консоли в случае совпадения

int PrintRequest(List\*\* top, List\* input) {

int count = 0;

List\* tmp = (\*top);

while (tmp != NULL) {

if (Compare(tmp, input) == SAME) {

printf("%s ", tmp->surname);

printf("%s ", tmp->name);

printf("%s\n", tmp->father);

count++;

}

tmp = tmp->next; //переход к следующему узлу

}

return count; //возвращает количество совпадений

}

//создает список, отсортированный по алфавиту

int AddToList(List\*\* top, List\* man) {

List\* tmp = (\*top);

List\* prev = NULL;

if (\*top == NULL) {

man->next = \*top;

\*top = man;

return SUCCESS\_ADD;

}

while (tmp != NULL) {

//побайтно сравнивает коды символов двух строк и возвращает:

//1. 0 – если сравниваемее строки идентичны

//2. Положительное число – если строки отличаются и код первого

//отличающегося символа в строке man больше кода символа на той же позиции в строке tmp

//3. Отрицательное число - наоборот (т.е. меньше)

int SurnameFlag = strcmp(man->surname, tmp->surname);

int NameFlag = strcmp(man->name, tmp->name);

int FatherFlag = strcmp(man->father, tmp->father);

//сравнение для фамилии

if (SurnameFlag < 0) {

if (prev == NULL) {

man->next = (\*top);

(\*top) = man;

return SUCCESS\_ADD;

}

prev->next = man;

man->next = tmp;

return SUCCESS\_ADD;

}

//сравнение для имени

else if (SurnameFlag == 0 && NameFlag < 0) {

if (prev == NULL) {

man->next = (\*top);

(\*top) = man;

return SUCCESS\_ADD;

}

prev->next = man;

man->next = tmp;

return SUCCESS\_ADD;

} //сравнение для отчества

else if (SurnameFlag == 0 && NameFlag == 0 && FatherFlag < 0) {

if (prev == NULL) {

man->next = (\*top);

(\*top) = man;

return SUCCESS\_ADD;

}

prev->next = man;

man->next = tmp;

return SUCCESS\_ADD;

}

prev = tmp;

tmp = tmp->next;

}

prev->next = man;

man->next = NULL;

return SUCCESS\_ADD;

}

//сравнивает элементы списка

int Compare(List\* man, List\* input) {

int length;

//сравнение для фамилии

if (input->surname != "") {

length = strlen(input->surname); //длина строки

//сверяет посимвольно элементы списка с тем, что ввели с консоли

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

if (man->surname[i] != input->surname[i]) {

return NOTSAME;

}

}

}

//сравнение для имени

if (input->name != "") {

length = strlen(input->name); //длина строки

//сверяет посимвольно элементы списка с тем, что ввели с консоли

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

if (man->name[i] != input->name[i]) {

return NOTSAME;

}

}

}

//сравнение для отчества

if (input->father != "") {

length = strlen(input->father); //длина строки

//сверяет посимвольно элементы списка с тем, что ввели с консоли

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

if (man->father[i] != input->father[i]) {

return NOTSAME;

}

}

}

//выводит 0 в случае совпадения

return SAME;

}

//очищает выделенную память

void FreeList(List\*\* top) {

List\* tmp = (\*top);

//конец, если и так пусто

if (tmp == NULL)

return;

while (\*top != NULL) {

tmp = \*top;

\*top = (\*top)->next;

free(tmp);

}

free(\*top);

}

int main() {

List\* top = NULL;

List\* input = NULL;

FILE\* file;

file = fopen("List.txt", "r"); //открывает файл с ФИО разных людей

//проверка успешного открытия файла

if (file == NULL) {

printf("cannot open file");

return ERROR\_OPENING\_FILE;

}

List\* person = NULL;

char\* buffer = (char\*)calloc(3 \* SIZE, sizeof(char));

//считывает данные из файла в массив, затем в список

while (!feof(file)) {

fgets(buffer, 3 \* SIZE, file);

person = TransformToList(buffer);

AddToList(&top, person);

}

//печатает отсортированный список

PrintList(&top);

printf("Enter the request in format <<surname>> <<name>> <<second name>> : ");

//проверяет, есть ли в списке люди с нужной фамилией

fgets(buffer, 3 \* SIZE, stdin);

input = TransformToList(buffer);

int count = PrintRequest(&top, input);

if (count == 0) { //если нет совпадений

printf("no matches found in list");

}

free(input);

free(buffer);

FreeList(&top);

fclose(file);

return 0;

}

# **Описание тестирования**

Для тестирования данной программы производился ее неоднократный запуск с вводом конкретных данных, при котором проверялось:

1. Стабильность работы программы при одинаковых входных данных;
2. Корректное выполнение всех заявленных процедур;
3. Корректное завершение программы;

Приведем пример тестирования на скриншотах, данных ниже:

Файл List.txt

Tishkovets Sergey Evgenevich

Ivanov Stepan Alekseevich

Veresov Daniil Aleksandrovich

Hramov Nikita Dmitrievich

Koroleva Darya Sergeevna

Tishkovets Timur Evgenevich

Drekalov Nikita Sergeevich

Tevrisov Nikolay Romanovich

Sergeev Kirill Sergeevich

Ivanov Alexander Mihaylovich

Экран:

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

Автоматически созданное описаниеВведено с консоли: <<T>> << >> << >>

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

Автоматически созданное описаниеВведено с консоли: << >> <<S>> << >>

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

Автоматически созданное описаниеВведено с консоли: <<K>> <<D>> <<S>>

При проведении тестирования такого рода никаких проблем обнаружено не было, что позволяет судить о корректности работы программы в целом.