Санкт-Петербургский государственный политехнический университет

Кафедра “Прикладная математика”

**Отчет по лабораторной работе 1**

**“Алгоритмы и структуры данных”**

Студент группы № 5030102/20003

ФИО: Королёва Дарья Сергеевна

Выполнил (дата) 14.10.2023

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:

Список реализован c помощью структуры, которая представляет собой узел из персональных данных (фамилии, имени, отчества) и указателя на следующий узел.

Функция для получения узла списка из массива:

1. Считывание ФИО в отдельные строковые массивы.
2. Создание нового узла через выделение памяти функцией malloc и проверка успешности выполнения действия

Функция для вывода списка на экран:

Идет по всему списку и печатает ФИО.

Функция для сравнения элементов списка:

1. Вычисление длины строки, которую ввел пользователь.
2. Посимвольное сравнение элементов введенной строки и элементов списка (Сравнение начинается с фамилии, далее для имени и отчества).
3. Возвращается либо значение, соответствующее наличию полного совпадения, либо значение, соответствующее отсутствию совпадений.

Функция для печати запроса на поиск совпадений:

1. При наличии совпадений функция печатает на экране фамилию, имя и отчество через пробел.
2. Возвращает количество совпадений.

Функция для добавления элементов в список в алфавитном порядке:

1. Побайтно сравниваются коды символов двух строк – уже имеющейся и текущей (которую надо добавить).
2. Сравнение начинается с фамилии. Затем, в случае совпадения фамилий, сравнивают имена. В случае совпадения и второго - отчества.
3. Возвращается значение, соответствующее успешному добавлению элемента в список.

Функция для очищения памяти:

Освобождается выделенная ранее память под все узлы.

# **Текст программы**

#pragma warning(disable: 4996)

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <locale.h>

#include <ctype.h>

#define SIZE 20

typedef struct List {

char surname[SIZE];

char name[SIZE];

char lastname[SIZE];

struct List\* next;

} List;

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

strcpy(man->surname, surname);

strcpy(man->name, name);

strcpy(man->lastname, lastname);

}

List\* ToList(char\* tmp) {

List\* result;

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

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

char q;

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

surname[i] = q;

i++;

}

surname[i] = '\0';

i++;

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

name[k] = q;

i++;

k++;

}

name[k] = '\0';

i++;

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

lastname[j] = q;

i++;

j++;

}

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

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

}

else {

lastname[j] = '\0';

}

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

if (result == NULL) {

printf("not enough memory");

return NULL;

}

GetStr(result, surname, name, lastname);

return result;

}

void PrintList(List\*\* top) {

List\* tmp = \*top;

while (tmp != NULL) {

printf("%s %s %s\n", tmp->surname, tmp->name, tmp->lastname);

tmp = tmp->next;

}

}

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

int length;

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

length = strlen(input->surname);

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

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

//printf("not smth same");

return 0;

}

}

}

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

length = strlen(input->name);

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

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

// printf("not smth same");

return 0;

}

}

}

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

length = strlen(input->lastname);

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

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

//printf("not smth same");

return 0;

}

}

}

return 1;

}

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

int count = 0;

List\* tmp = (\*top);

while (tmp != NULL) {

if (Compare(tmp, input) == 1) {

printf("%s %s %s\n", tmp->surname, tmp->name, tmp->lastname);

count++;

}

tmp = tmp->next;

}

return count; //количество совпадений

}

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

List\* tmp = (\*top);

List\* prev = NULL;

if (\*top == NULL) {

man->next = \*top;

\*top = man;

printf("that's all right");

return 1;

}

while (tmp != NULL) {

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

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

int LastnameFlag = strcmp(man->lastname, tmp->lastname);

if (SurnameFlag < 0) {

if (prev == NULL) {

man->next = (\*top);

(\*top) = man;

printf("that's all right");

return 1;

}

prev->next = man;

man->next = tmp;

printf("that's all right");

return 1;

}

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

if (prev == NULL) {

man->next = (\*top);

(\*top) = man;

printf("that's all right");

return 1;

}

prev->next = man;

man->next = tmp;

printf("that's all right");

return 1;

}

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

if (prev == NULL) {

man->next = (\*top);

(\*top) = man;

printf("that's all right");

return 1;

}

prev->next = man;

man->next = tmp;

printf("that's all right");

return 1;

}

prev = tmp;

tmp = tmp->next;

}

prev->next = man;

man->next = NULL;

}

void FreeList(List\*\* top) {

List\* tmp = (\*top);

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");

if (file == NULL) {

printf("error opening file");

exit(1);

}

List\* man = NULL;

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

while (!feof(file)) {

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

man = ToList(buffer);

AddToList(&top, man);

}

PrintList(&top);

printf("<<surname>> <<name>> <<second name>> : ");

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

input = ToList(buffer);

int count = PrintRequest(&top, input);

if (count == 0) {

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

Koroleva Darya Sergeevna

Ivanov Ivan Ivanovich

Yurkova Aleksandra Kirillovna

Sidorov Nikolay Vladimirovich

Evtutova Tatiana Aleksandrovna

Maybe Eva Olegovna

Almenchenko Margo Sergeevna

Petrov Ivan Petrovich

Тесты:

Введено с консоли: «I» «I» «»

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Введено с консоли: «» «I» «»

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Введено с консоли: «» «Iv» «»

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Введено с консоли: «I» «I» «I»

Изображение выглядит как текст, снимок экрана, дизайн

Автоматически созданное описание

Введено с консоли: «» «» «S»

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

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