**Исходный текст программы**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<ctype.h>

#define MAXLEN 256

typedef struct userStruct {

int id;

char \*fullName;

int age;

char \*profession;

float friendsRating;

float publicRating;

int friendsCount;

int friendsId[MAXLEN];

} User;

char \*\*simpleSplit(char \*str, int length, char sep);

void simpleSplitInt(const char \*str, char sep, int arr[]);

User \*fillStruct(char \*\*str);

int cmp(const void \*a, const void \*b);

void sortStructs(User \*\*users, int count);

void printHeader();

void printUser(User \*user);

void printAllUsers(User \*\*users, int count);

void trim(char \*str);

void clearConsole();

void addUser(User \*\*\*usersPtr, int \*count, int \*n);

int startsWithIgnoreCase(const char \*str, const char \*prefix);

void clearStruct(User \*user);

int main() {

User \*\*users = NULL;

int slen, i, n, count, j;

char sep;

char temp[MAXLEN];

char \*\*splitArray;

char ask;

FILE \*file;

file = fopen("index.csv", "r");

if (file != NULL) {

n = 0;

while ((fgets(temp, MAXLEN, file)) != NULL) n++;

rewind(file);

users = (User \*\*)malloc((n + 50) \* sizeof(User \*));

if (users != NULL) {

sep = ';';

puts("Initial array:");

printHeader();

for (i = 0, count = 0; i < n; i++, count++) {

fgets(temp, MAXLEN, file);

slen = strlen(temp);

temp[slen - 1] = '\0';

splitArray = simpleSplit(temp, slen, sep);

if (splitArray != NULL) {

users[i] = fillStruct(splitArray);

if (users[i] != NULL) printUser(users[i]);

else {

puts("Structure not allocated!");

i = n;

}

}

else {

puts("Error data reading");

i = n;

}

}

}

else puts("Out of memory!");

fclose(file);

}

else perror("Failed to open file");

if (users && n && count == n) {

do {

printf("\nDo you want to add another user? (y/n): ");

scanf(" %c", &ask);

getchar();

if (ask == 'y' || ask == 'Y') {

clearConsole();

addUser(&users, &count, &n);

}

} while (ask == 'y' || ask == 'Y');

clearConsole();

printf("Press ENTER to see all users sorted by number of friends ");

getchar();

sortStructs(users, count);

printAllUsers(users, count);

printf("\nYou can now sort users by either name or profession. Choose one option (1 or 2): ");

scanf("%c", &ask);

if (ask != '1' && ask != '2') {

printf("invalid option");

} else if (ask == '1') {

clearConsole();

printf("Enter the user name: ");

scanf("%s", temp);

getchar();

printf("\n");

printHeader();

j = 0;

for (i = 0; i < count; i++) {

if (startsWithIgnoreCase(users[i]->fullName, temp)) {

printUser(users[i]);

j++;

}

}

if (j == 0) {

printf("\nNo user seems to match your input.\n");

}

} else {

clearConsole();

printf("Enter the name of profile image : ");

scanf("%s", temp);

getchar();

printf("\n");

printHeader();

j = 0;

for (i = 0; i < count; i++) {

if (startsWithIgnoreCase(users[i]->profession, temp)) {

printUser(users[i]);

j++;

}

}

if (j == 0) {

printf("\nNo user seems to match your input.\n");

}

}

for (i = 0; i < count; i++) clearStruct(users[i]);

free(users);

users = NULL;

} else puts("No data found!");

return 0;

}

char \*\*simpleSplit(char \*str, int length, char sep) {

int count = 0;

int i = 0;

int start = 0;

int j = 0;

int wordLen = 0;

char \*\*result = NULL;

char \*newStr = NULL;

int allocError = 0;

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

if (str[i] == sep) count++;

}

count++;

result = malloc(count \* sizeof(char \*));

if (result == NULL) {

perror("Memory allocation failed");

} else {

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

if (str[i] == sep || str[i] == '\0') {

wordLen = i - start;

newStr = malloc((wordLen + 1) \* sizeof(char));

if (newStr == NULL) {

perror("Memory allocation failed");

allocError = 1;

i = length;

} else {

strncpy(newStr, str + start, wordLen);

newStr[wordLen] = '\0';

result[j++] = newStr;

start = i + 1;

}

}

}

if (allocError) {

for (i = 0; i < j; i++) {

free(result[i]);

}

free(result);

result = NULL;

}

}

return result;

}

void simpleSplitInt(const char \*str, char sep, int arr[]) {

int count = 0;

int start = 0;

int i, len;

char tempStr[MAXLEN];

for (i = 0; i < MAXLEN; i++) {

arr[i] = 0;

}

for (i = 0; str[i] != '\0'; i++) {

if (str[i] == sep || str[i + 1] == '\0') {

len = (str[i] == sep) ? (i - start) : (i - start + 1);

strncpy(tempStr, str + start, len);

tempStr[len] = '\0';

arr[count++] = atoi(tempStr);

start = i + 1;

}

}

}

User \*fillStruct(char \*\*str) {

User \*user = NULL;

user = (User\*)malloc(sizeof(User));

if (user != NULL) {

user->id = atoi(str[0]);

free(str[0]);

user->fullName = str[1];

user->age = atoi(str[2]);

free(str[2]);

user->profession = str[3];

user->friendsRating = atof(str[4]);

free(str[4]);

user->publicRating = atof(str[5]);

free(str[5]);

user->friendsCount = atoi(str[6]);

free(str[6]);

simpleSplitInt(str[7], ',', user->friendsId);

free(str[7]);

free(str);

}

return user;

}

void printHeader() {

printf("%-3s %-20s %-5s %-15s %-15s %-15s %-15s %-20s\n",

"ID", "Full Name", "Age", "Profession", "Friends Rating", "Public Rating", "Friends Count", "Friends IDs");

}

void printUser(User \*user) {

int i;

printf("%-3d %-20s %-5d %-15s %-15.1f %-15.1f %-15d ",

user->id, user->fullName, user->age, user->profession, user->friendsRating, user->publicRating, user->friendsCount);

printf("[");

for (i = 0; i < user->friendsCount; i++) {

printf("%d", user->friendsId[i]);

if (i < user->friendsCount - 1) {

printf(", ");

}

}

printf("]\n");

}

void addUser(User \*\*\*usersPtr, int \*count, int \*n) {

User \*\*newUsers;

User \*newUser;

char tempStr[MAXLEN];

int tempId, tempAge;

float tempFriendsRating, tempPublicRating;

int tempFriendsCount;

char \*tempFullName, \*tempProfileImg;

if (\*(count) == \*(n)) {

newUsers = realloc(\*usersPtr, ((\*count) \* 2) \* sizeof(User \*));

(\*n) = (\*count) \* 2;

} else {

newUsers = \*usersPtr;

}

if (newUsers == NULL) {

perror("Memory allocation failed");

} else {

\*usersPtr = newUsers;

newUser = malloc(sizeof(User));

if (newUser == NULL) {

perror("Memory allocation failed");

} else {

(\*usersPtr)[\*count] = newUser;

printf("Enter user ID: ");

scanf("%d", &tempId);

getchar();

newUser->id = tempId;

printf("Enter full name: ");

newUser->fullName = malloc(MAXLEN \* sizeof(char));

if (newUser->fullName == NULL || fgets(newUser->fullName, MAXLEN, stdin) == NULL) {

perror("Failed to read full name or allocate memory");

free(newUser);

} else {

trim(newUser->fullName);

printf("Enter age: ");

scanf("%d", &tempAge);

getchar();

newUser->age = tempAge;

printf("Enter profession: ");

newUser->profession = malloc(MAXLEN \* sizeof(char));

if (newUser->profession == NULL || fgets(newUser->profession, MAXLEN, stdin) == NULL) {

perror("Failed to read image filename or allocate memory");

free(newUser->fullName);

free(newUser);

} else {

trim(newUser->profession);

printf("Enter friends rating: ");

scanf("%f", &tempFriendsRating);

newUser->friendsRating = tempFriendsRating;

printf("Enter public rating: ");

scanf("%f", &tempPublicRating);

newUser->publicRating = tempPublicRating;

printf("Enter friends count: ");

scanf("%d", &tempFriendsCount);

getchar();

newUser->friendsCount = tempFriendsCount;

printf("Enter friends IDs (example: 1,2,3,4): ");

scanf("%s", tempStr);

getchar();

simpleSplitInt(tempStr, ',', newUser->friendsId);

printf("\nNew user successfully added!\n");

(\*count)++;

}

}

}

}

}

void clearStruct(User \*user) {

if (user != NULL) {

free(user->fullName);

user->fullName = NULL;

free(user->profession);

user->profession = NULL;

free(user);

}

}

int cmp(const void \*a, const void \*b) {

User \*userA = \*(User\*\*)a;

User \*userB = \*(User\*\*)b;

return userB->friendsCount - userA->friendsCount;

}

void sortStructs(User \*\*users, int count) {

qsort(users, count, sizeof(User\*), cmp);

}

void printAllUsers(User \*\*users, int count) {

int i;

printHeader();

for (i = 0; i < count; i++) {

printUser(users[i]);

}

}

void clearConsole() {

#if defined(\_WIN32) || defined(\_WIN64)

system("cls");

#else

system("clear");

#endif

}

void trim(char \*str) {

int i = 0;

for (i = 0; i < MAXLEN; i++) {

if (str[i] == '\n' || str[i] == '\r') {

str[i] = '\0';

i = MAXLEN;

}

}

}

int startsWithIgnoreCase(const char \*str, const char \*prefix) {

int isPrefix = 1;

while (\*str && \*prefix && isPrefix) {

if (tolower(\*str) != tolower(\*prefix)) {

isPrefix = 0;

}

str++;

prefix++;

}

if (\*prefix != '\0') {

isPrefix = 0;

}

return isPrefix;

}

**Контрольные примеры**

**Пример:**

Initial array:

ID Full Name Age Profession Friends Rating Public Rating Friends Count Friends IDs

1 John Doe 30 teacher 4.5 3.9 3 [2, 5, 7]

2 Jane Smith 25 engeneer 3.8 4.1 2 [1, 3]

3 Alice Johnson 28 driver 4.2 3.7 4 [1, 2, 6, 8]

4 Michael Brown 33 pilot 3.9 4.0 5 [3, 6, 9, 10, 2]

5 Emily Davis 27 dentist 4.1 3.8 3 [1, 2, 3]

6 David Wilson 35 actor 4.0 4.2 2 [5, 2]

7 Linda Martinez 32 actor 3.9 3.7 4 [4, 6, 5, 1]

8 Robert White 29 teacher 4.3 3.8 3 [1, 2, 3]

9 Sarah Taylor 31 teacher 4.0 4.1 5 [8, 5, 6, 3, 1]

10 James Anderson 34 pilot 4.2 3.9 2 [1, 2]

11 Davidios Morgan 20 teacher 2.0 1.0 0 []

Do you want to add another user? (y/n): y

Enter user ID: 12

Enter full name: newUser

Enter age: 12

Enter profession: pilot

Enter friends rating: 4.3

Enter public rating: 4.0

Enter friends count: 3

Enter friends IDs (example: 1,2,3,4): 1,4,6

New user successfully added!

Do you want to add another user? (y/n): n

Press ENTER to see all users sorted by number of friends

ID Full Name Age Profession Friends Rating Public Rating Friends Count Friends IDs

9 Sarah Taylor 31 teacher 4.0 4.1 5 [8, 5, 6, 3, 1]

4 Michael Brown 33 pilot 3.9 4.0 5 [3, 6, 9, 10, 2]

3 Alice Johnson 28 driver 4.2 3.7 4 [1, 2, 6, 8]

7 Linda Martinez 32 actor 3.9 3.7 4 [4, 6, 5, 1]

5 Emily Davis 27 dentist 4.1 3.8 3 [1, 2, 3]

8 Robert White 29 teacher 4.3 3.8 3 [1, 2, 3]

1 John Doe 30 teacher 4.5 3.9 3 [2, 5, 7]

12 newUser 12 pilot 4.3 4.0 3 [1, 4, 6]

10 James Anderson 34 pilot 4.2 3.9 2 [1, 2]

2 Jane Smith 25 engeneer 3.8 4.1 2 [1, 3]

6 David Wilson 35 actor 4.0 4.2 2 [5, 2]

11 Davidios Morgan 20 teacher 2.0 1.0 0 []

You can now sort users by either name or profession. Choose one option (1 or 2): 1

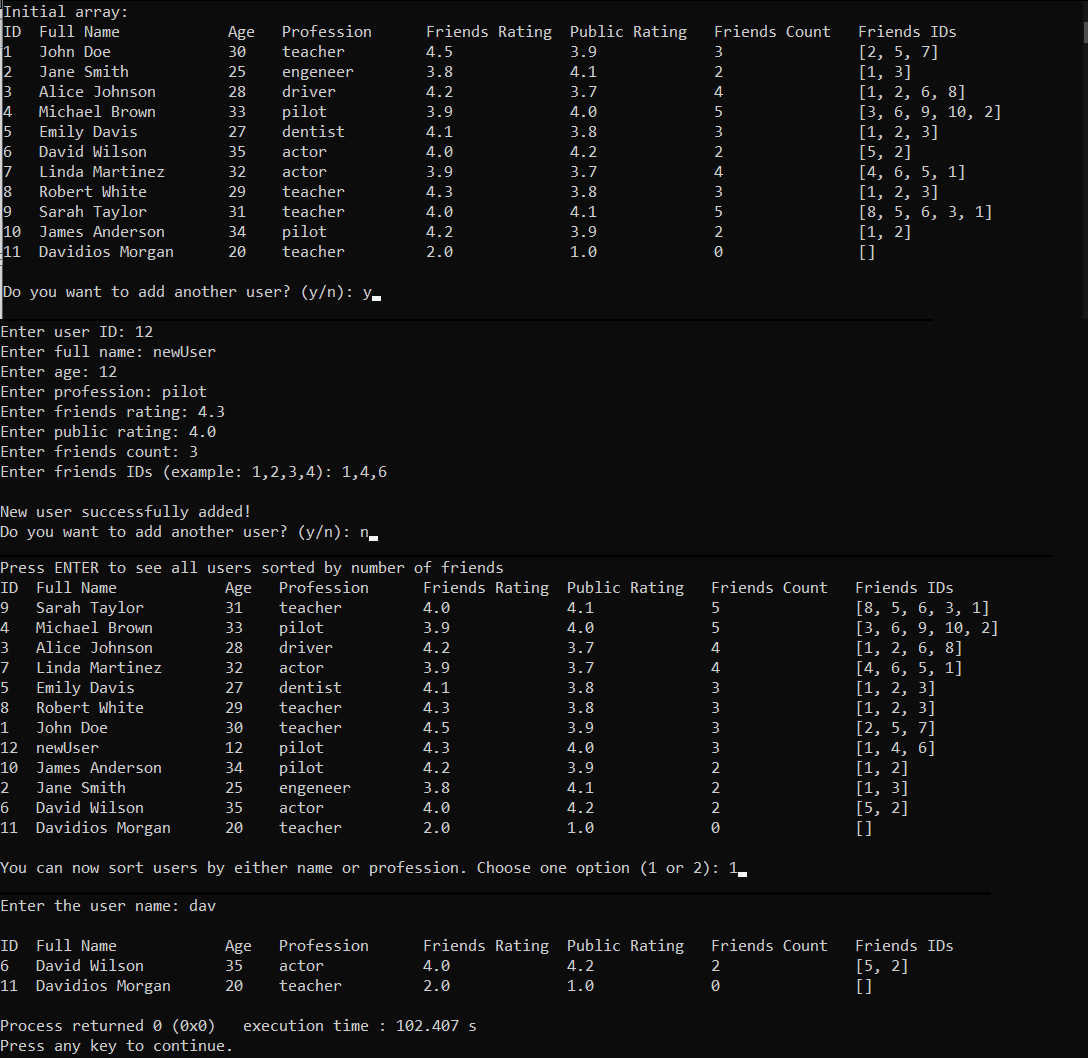
Enter the user name: dav

ID Full Name Age Profession Friends Rating Public Rating Friends Count Friends IDs

6 David Wilson 35 actor 4.0 4.2 2 [5, 2]

11 Davidios Morgan 20 teacher 2.0 1.0 0 []

**Примеры выполнения программы**

****

**Выводы.**

В результате выполнения лабораторной работы были получены навыки работы с указателями на структуры и функциями в C.