

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <clocale>

#include <cstdio>

#include <cstdlib>

#include <cstring>

#include <cmath>

#include <windows.h>

struct letter

{

double mark;

int indexOt;

char\* surnameOt;

char\* nameOt;

int indexPol;

char\* surnamePol;

char\* namePol;

letter\* next;

};

void ways()

{

printf\_s("\nДобавитьписьмо------------------1\n");

printf\_s("Вывод всей таблицы---------------2\n");

printf\_s("Удалить письма отправителя-------3\n");

printf\_s("Поиск писем отправителя----------4\n");

printf\_s("Поиск писем с большей оценкой----5\n");

printf\_s("Сортировка по алфавиту-----------6\n");

printf\_s("Выход + сохранение данных--------7\n");

}

void tablPrint(letter\* mass)

{

printf\_s("%-12s %-15s %-12s %-17s %-15s %-13s %-11s\n", "Оценка", "Индекс", "Фамилия", "Имя", "Индекс", "Фамилия", "Имя");

printf\_s("%-12s %-15s %-12s %-17s %-15s %-13s %-11s\n", "письма", "отправ.", "отправ.", "отправ.", "получ.", "получ.", "получ.");

printf\_s("----------------------------------------------------------------------------------------------\n");

while (mass != NULL)

{

printf\_s("%-12f %-15d %-12s %-18s", mass->mark, mass->indexOt, mass->surnameOt, mass->nameOt);

printf\_s("%-16d %-14s %-11s\n", mass->indexPol, mass->surnamePol, mass->namePol);

mass = mass->next;

}

printf\_s("----------------------------------------------------------------------------------------------\n");

}

//

void inserting(letter\*\* mass, int\* N)

{

printf\_s("\nПожалуйста, введите следующие параметры\n");

char str[100];

float Fnumber;

int Inumber;

(\*mass) = (letter\*)realloc((\*mass), (\*N + 1) \* sizeof(letter));

printf\_s("\nОценкаписьма:");

scanf\_s("%f", &Fnumber);

(\*mass)[\*N].mark = Fnumber;

printf\_s("Индексотправителя:");

scanf\_s("%d", &Inumber);

(\*mass)[\*N].indexOt = Inumber;

printf\_s("Фамилияотправителя:");

while (fgetc(stdin) != '\n');

gets\_s(str, 100);

(\*mass)[\*N].surnameOt = (char\*)malloc(strlen(str) + 1);

strcpy((\*mass)[\*N].surnameOt, str);

printf\_s("Имяотправителя:");

gets\_s(str, 100);

(\*mass)[\*N].nameOt = (char\*)malloc(strlen(str) + 1);

strcpy((\*mass)[\*N].nameOt, str);

printf\_s("Индексполучателя:");

scanf\_s("%d", &Inumber);

(\*mass)[\*N].indexPol = Inumber;

printf\_s("Фамилияполучателя:");

while (fgetc(stdin) != '\n');

gets\_s(str, 100);

(\*mass)[\*N].surnamePol = (char\*)malloc(strlen(str) + 1);

strcpy((\*mass)[\*N].surnamePol, str);

printf\_s("Имяполучателя:");

gets\_s(str, 100);

(\*mass)[\*N].namePol = (char\*)malloc(strlen(str) + 1);

strcpy((\*mass)[\*N].namePol, str);

(\*N)++;

}

void save(letter\* head)

{

FILE\* nameFile = fopen("Data save.txt", "w");

while (head != NULL)

{

fprintf\_s(nameFile, "%lf\n", head->mark);

fprintf\_s(nameFile, "%d\n", head->indexOt);

fprintf\_s(nameFile, "%s\n", head->surnameOt);

fprintf\_s(nameFile, "%s\n", head->nameOt);

fprintf\_s(nameFile, "%d\n", head->indexPol);

fprintf\_s(nameFile, "%s\n", head->surnamePol);

fprintf\_s(nameFile, "%s\n", head->namePol);

head = head->next;

}

fclose(nameFile);

}

void searchLetters(letter\* mass)

{

int t = 0, forSurname, forName;

char\* surnameOtprv, \* nameOtprv, buff[512];

letter\* m = mass;

while (fgetc(stdin) != '\n');

printf\_s("Введите фамилию отправителя:");

gets\_s(buff, 512);

surnameOtprv = (char\*)malloc(1 + strlen(buff));

strcpy(surnameOtprv, buff);

printf\_s("Введитеимяотправителя:");

gets\_s(buff, 512);

nameOtprv = (char\*)malloc(1 + strlen(buff));

strcpy(nameOtprv, buff);

printf\_s("\nПисьмаэтогоотправителя:\n");

while (mass != NULL)

{

forName = strcmp(mass->nameOt, nameOtprv);

forSurname = strcmp(mass->surnameOt, surnameOtprv);

if ((forName == 0) && (forSurname == 0))

{

printf\_s("%-15lf %-15d %-15d %-15s %-15s\n", mass->mark, mass->indexOt, mass->indexPol, mass->surnamePol, mass->namePol);

t++;

}

mass = mass->next;

}

if (t == 0)printf\_s("\nНенашлосьписемзаданногоотправителя\n");

free(surnameOtprv);

free(nameOtprv);

mass = m;

}

//

int searchRating (letter\* mass, int N, char\*\*\* addressees, double ratingMail)

{

int a = 0;

char str[100];

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

{

if (mass[i].mark > ratingMail)

{

(\*addressees) = (char\*\*)realloc((\*addressees), sizeof(char\*) \* (a + 1));

(\*addressees)[a] = mass[i].surnamePol;

a++;

}

}

if (a != 0)

{

for (int i = 0; i < (a - 1); i++)

{

for (int j = i + 1; j < a; j++)

{

if (strcmp((\*addressees)[i], (\*addressees)[j]) > 0)

{

strcpy(str, (\*addressees)[i]);

strcpy((\*addressees)[i], (\*addressees)[j]);

strcpy((\*addressees)[j], str);

}

}

}

printf\_s("\nПолучатели письма с такой оценкой(пофамильно):\n");

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

printf\_s("%s\n", (\*addressees)[i]);

return a;

}

else

{

printf\_s("\nПолучателейписьмасоценкойбольше %f ненайдено\n", &ratingMail);

return 0;

}

}

void receive(letter\*\* head, FILE\* files\_name)

{

if (files\_name == stdin) printf("Укажитеоценкуписьма:");

fscanf(files\_name, "%lf", &(\*head)->mark);

if (files\_name == stdin) printf("Укажите индекс отправителя:");

fscanf(files\_name, "%d", &((\*head)->indexOt));

if (files\_name == stdin) printf("Укажитефамилиюотправителя:");

while (fgetc(files\_name) != '\n');

(\*head)->surnameOt = (char\*)malloc(sizeof(char) \* 512);

fgets((\*head)->surnameOt, 512, files\_name);

(\*head)->surnameOt[strlen((\*head)->surnameOt) - 1] = '\0';

if (files\_name == stdin) printf("Укажитеимяотправителя:");

(\*head)->nameOt = (char\*)malloc(sizeof(char) \* 512);

fgets((\*head)->nameOt, 512, files\_name);

(\*head)->nameOt[strlen((\*head)->nameOt) - 1] = '\0';

if (files\_name == stdin) printf("Укажитеиндексполучателя:");

fscanf(files\_name, "%d", &(\*head)->indexPol);

if (files\_name == stdin) printf("Укажитефамилиюполучателя:");

while (fgetc(files\_name) != '\n');

(\*head)->surnamePol = (char\*)malloc(sizeof(char) \* 512);

fgets((\*head)->surnamePol, 512, files\_name);

(\*head)->surnamePol[strlen((\*head)->surnamePol) - 1] = '\0';

if (files\_name == stdin) printf("Укажитеимяполучателя:");

(\*head)->namePol = (char\*)malloc(sizeof(char) \* 512);

fgets((\*head)->namePol, 512, files\_name);

(\*head)->namePol[strlen((\*head)->namePol) - 1] = '\0';

}

void delete\_data(letter\*\* headList)

{

free((\*headList)->surnameOt);

free((\*headList)->nameOt);

free((\*headList)->surnamePol);

free((\*headList)->namePol);

}

void add\_list\_serial(letter\*\* head, int\* size, FILE\* files\_name)

{

(\*head)->next = (letter\*)malloc(sizeof(letter));

//letter\* m = (\*head)->next;

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

receive(&(\*head), files\_name);

(\*head)->next = NULL;

(\*size)++;

}

void create\_list(letter\*\* head, FILE\* files\_name)

{

(\*head) = (letter\*)malloc(sizeof(letter));

receive(&(\*head), files\_name);

(\*head)->next = NULL;

}

void delete\_otprav(letter\*\* head, int\* size)

{

int counter = 0, t = \*size, index\_del;

printf("Введитеиндекс:");

scanf("%d", &index\_del);

letter\* p = (\*head), \* pred = NULL, \* udal;

while ((\*head) != NULL)

{

if ((\*head)->indexOt == index\_del)

{

if (counter == 0)

{

udal = (\*head);

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

p = (\*head);

delete\_data(&udal);

free(udal);

}

else if (counter == (\*size - 1))

{

udal = (\*head);

(\*head) = pred;

(\*head)->next = NULL;

delete\_data(&udal);

free(udal);

}

else

{

udal = (\*head);

pred->next = (\*head)->next;

(\*head) = pred->next;

delete\_data(&udal);

free(udal);

}

t--;

counter++;

}

else

{

pred = (\*head);

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

counter++;

}

}

(\*head) = p;

(\*size) = t;

}

void collating(letter\*\* head, int size)

{

int counter = 0;

letter\* p = (\*head), \* sled = NULL, \* pred = NULL;

for (int i = 0; i < size - 1; i++)

{

(\*head) = p;

for (int j = i + 1; j < size; j++)

{

if ((\*head)->next != NULL)

{

if (strcmp((\*head)->surnamePol, (\*head)->next->surnamePol) > 0)

{

if (counter == 0)

{

pred = (\*head)->next;

sled = (\*head)->next->next;

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

(\*head)->next = sled;

p = pred;

}

else if (counter == (size - 1))

{

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

pred->next = (\*head)->next;

(\*head)->next = NULL;

}

else

{

pred->next = (\*head)->next;

sled = (\*head)->next->next;

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

(\*head)->next = sled;

pred = pred->next;

}

counter++;

}

else

{

pred = (\*head);

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

counter++;

}

}

else break;

}

}

(\*head) = p;

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int size = 1, parametr, t = 0;

double rating;

bool c0nt1nue = true;

letter\* begin = (letter\*)malloc(sizeof(letter));

letter\* end = (letter\*)malloc(sizeof(letter));

FILE\* data\_plus = fopen("Data capture.txt", "r");

create\_list(&begin, data\_plus);

end = begin;

while (!feof(data\_plus))

{

add\_list\_serial(&end, &size, data\_plus);

}

fclose(data\_plus);

while (c0nt1nue)

{

ways();

printf\_s("\nВыберите номер согласно действию:");

scanf("%d", &parametr);

//while (fgetc(stdin) != '\n');

switch (parametr)

{

case 1:

{

add\_list\_serial(&end, &size, stdin);

break;

}

case 2:

{

tablPrint(begin);

break;

}

case 3:

{

delete\_otprav(&begin, &size);

break;

}

case 4:

{

searchLetters(begin);

break;

}

case 5:

{

printf\_s("\nВведите оценку письма:");

scanf("%lf", &rating);

letter\* m = begin;

//while (fgetc(stdin) != '\n');

printf("\nПисьма с оценкой больше заданной:\n");

while (begin != NULL)

{

if (begin->mark > rating)

{

t++;

printf\_s("%-15d %-15s %-15s", begin->indexOt, begin->surnameOt, begin->nameOt);

printf\_s("%-15d %-15s %-15s\n", begin->indexPol, begin->surnamePol, begin->namePol);

}

begin = begin->next;

}

//if (t > 0) printf("----------------------------------------------------------------------------------------------\n");

if (t == 0)printf("\nНет писем с такой оценкой");

begin = m;

break;

}

case 6:

{

collating(&begin, size);

break;

}

case 7:

{

save(begin);

c0nt1nue = false;

break;

}

default:

{

printf("Ошибка!\nВы ввели число, для которого нет команд\nПожалуйста, введите корректное число");

break;

}

}

}

while (begin != NULL)

{

free(begin->nameOt);

free(begin->surnameOt);

free(begin->surnamePol);

free(begin->namePol);

begin = begin->next;

}

free(begin);

}

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описаниеИзображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, черный

Автоматически созданное описание