Министерство образования и науки Российской Федерации

НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Расчетно-графическая работа

По программированию в языке СИ

Вариант №16

Группа: АВТ–019 Преподаватель:

Студент: Москвичев С.А. Ядрышников О.Д

НОВОСИБИРСК

2021

**Задание к варианту:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **16** | **С** | **200** | **1** | **Статически** | **Убывает** | **Экран** |

Написать программу, выполняющую действия с базой данных. База данных должна быть организована в виде массива структур.

Память под массив структур выделяется статически.

Максимальное количество записей – 200.

Поля базы данных: Название пункта, расстояние, количество рейсов

Программа должна обеспечивать следующий набор операций:

1) Ввод записи с произвольным номером;

2) Вывод записи с заданным номером;

3) Сортировка записей по заданному полю в порядке убывания или возрастания;

4) Вывод всех записей в отсортированном порядке на экран или принтер;

5) Сохранение всех записей в файле;

6) Чтение записей из файла.

**Программный код:**

#include <stdio.h>  
#include <stdbool.h>  
#include <string.h>  
  
#define NOTES 200  
#define NAME\_LENGTH 30  
#define FILE\_PATH "database.txt"  
  
typedef struct {  
 char name[NAME\_LENGTH];  
 int distance;  
 int flightsCount;  
} Flight;  
  
void add\_note();  
void out\_note();  
void sort();  
void print\_db();  
void write\_db();  
void read\_db();  
  
int btr\_str(const char\* s1, const char\* s2, int l);  
  
Flight flights[NOTES];  
bool isSorted;  
  
int main() {  
 isSorted = false;  
 int command = 0;  
 do {  
 printf("\n\t[Menu]\n");  
 printf("<1> Input note\n");  
 printf("<2> Output note\n");  
 printf("<3> Sort notes by destination\n");  
 printf("<4> Output sorted notes\n");  
 printf("<5> Save notes in file\n");  
 printf("<6> Read notes from file\n");  
 printf("<0> Exit\n");  
  
 printf("\nInput number of command here: ");  
 scanf("%d", &command);  
  
 if (command == 0)  
 break;  
 else if (command == 1)  
 add\_note();  
 else if (command == 2)  
 out\_note();  
 else if (command == 3)  
 sort();  
 else if (command == 4)  
 print\_db();  
 else if (command == 5)  
 write\_db();  
 else if (command == 6)  
 read\_db();  
 else  
 printf("Incorrect command, try again:\n");  
  
 printf("################\n");  
 } while (command != 0);  
  
 return 0;  
}  
  
void add\_note() {  
 int num;  
  
 printf("Number of note: ");  
 scanf("%d", &num);  
 num--;  
  
 printf("> Input new note here:\n");  
  
 printf(" Destination: ");  
 scanf("%s", flights[num].name);  
  
 printf(" Distance: ");  
 scanf("%d", &flights[num].distance);  
  
 printf(" Number of flights: ");  
 scanf("%d", &flights[num].flightsCount);  
  
 isSorted = false;  
}  
  
void out\_note() {  
 int num;  
  
 printf("Number of note: ");  
 scanf("%d", &num);  
 num--;  
  
 if (flights[num].distance > 0) {  
 printf("> Note №%d\n", num+1);  
 printf("\tDestination: %s\n", flights[num].name);  
 printf("\tDistance: %d\n", flights[num].distance);  
 printf("\tNumber of flights: %d\n", flights[num].flightsCount);  
 } else {  
 printf("There is not note %d\n", num+1);  
 }  
}  
  
void sort() {  
 if (!isSorted) {  
 for (int loop = 0; loop < NOTES/2; loop++) {  
 for (int number = 0; number < NOTES-1; number++) {  
 if (flights[ number ].name == NULL ||  
 (flights[number+1].name != NULL &&  
 btr\_str(flights[number].name, flights[number+1].name, 0) > 0))  
 {  
 Flight temp = flights[number+1];  
 flights[number+1] = flights[number];  
 flights[number] = temp;  
 }  
 }  
 }  
  
 printf("Database was sorted\n");  
 isSorted = true;  
 }  
  
}  
  
void print\_db() {  
 if (!isSorted)  
 sort();  
  
 for (int num = 0; num < NOTES; num++) {  
 if (flights[num].distance > 0) {  
 printf("> Note № %d", num+1);  
 printf(" Destination: %s\n", flights[num].name);  
 printf(" Distance: %d\n", flights[num].distance);  
 printf(" Number of flights: %d\n", flights[num].flightsCount);  
 }  
 }  
}  
  
void write\_db() {  
 FILE\* database = fopen(FILE\_PATH, "w");  
  
 if (database != NULL) {  
 int added\_flights = 0;  
 for (int note = 0; note < NOTES; note++) {  
 if (flights[note].distance > 0)  
 added\_flights++;  
 }  
 fprintf(database, "#%d\n", added\_flights);  
  
 for (int num = 0; num < NOTES; num++) {  
 if (flights[num].distance > 0) {  
 fprintf(database, "%s %d %d\n",  
 flights[num].name, flights[num].distance, flights[num].flightsCount);  
 }  
 }  
 printf("Notes was saved\n");  
 } else {  
 printf("Error\n");  
 }  
  
 fclose(database);  
}  
  
void read\_db() {  
 FILE\* database = fopen(FILE\_PATH, "r");  
  
 if (database != NULL) {  
 for (int note = 0; note < NOTES; note++) {  
 flights[note].distance = -1;  
 }  
  
 int added\_flights = 0;  
 fscanf(database, "#%d\n", &added\_flights);  
  
 for (int num = 0; num < added\_flights; num++) {  
 fscanf(database, "%s %d %d\n",  
 flights[num].name, &flights[num].distance, &flights[num].flightsCount);  
 }  
  
 printf("Notes was loaded\n");  
 } else {  
 printf("Error\n");  
 }  
  
 fclose(database);  
 isSorted = false;  
}  
  
int btr\_str(const char\* s1, const char\* s2, int l) {  
 int ltr = l;  
 int max\_ltr = strlen(s1) < strlen(s2) ? strlen(s1) : strlen(s2);  
  
 if (s1[ltr] > s2[ltr])  
 return -1;  
 else if (s2[ltr] > s1[ltr])  
 return 1;  
 else {  
 if (ltr < max\_ltr)  
 return btr\_str(s1, s2, ltr+1);  
 else  
 return (strlen(s1) > strlen(s2) ? -1 : 1);  
 }  
}

**Результат:**

