Министерство образования и науки Российской Федерации

НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Расчетно-графическая работа

По программированию в языке СИ

Вариант №15

Группа: АВТ–019 Преподаватель:

Студент: Москвин Т.А. Ядрышников О.Д

НОВОСИБИРСК

2021

**Задание к варианту:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **15** | **B** | **200** | **2** | **Динамически** | **Убывает** | **Экран** |

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

Память под массив структур выделяется динамически.

Максимальное количество записей – 200.

Поля базы данных: Наименование детали, количество, вес

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 <math.h>

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

#include <string.h>

#include <stdbool.h>

#define NOTES 200

void input\_by\_number(int number); // Ввод записи с произвольным номером

void print\_by\_number(int number); // Вывод записи с заданным номером

void sort\_by\_weight(); // Сортировка деталей по весу в порядке убывания

void print\_sorted(); // Вывод всех записей в отсортированном порядке

void save\_to\_file(char\* path); // Сохранение всех записей в файле

void read\_from\_file(char\* path); // Чтение записей из файла

int ask\_number();

char\* ask\_path();

struct Detail {

char\* name;

int amount;

int weight;

int number;

};

static struct Detail\* database;

static bool sorted;

int main() {

setlocale(LC\_ALL, "ru");

database = malloc(sizeof(struct Detail) \* NOTES);

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

database[i].name = NULL;

printf("\n### Database ###\n");

printf("1) Ввод записи с произвольным номером\n");

printf("2) Вывод записи с заданным номером\n");

printf("3) Сортировка деталей по весу в порядке убывания\n");

printf("4) Вывод всех записей в отсортированном порядке\n");

printf("5) Сохранение всех записей в файле\n");

printf("6) Чтение записей из файла\n");

printf("0) Выход из программы\n");

int command = -1;

while (command != 0) {

printf("Введите номер команды: ");

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

switch (command)

{

case 1:

input\_by\_number(ask\_number());

break;

case 2:

print\_by\_number(ask\_number());

break;

case 3:

sort\_by\_weight();

break;

case 4:

print\_sorted();

break;

case 5:

save\_to\_file(ask\_path());

break;

case 6:

read\_from\_file(ask\_path());

break;

case 0:

break;

default:

printf("Неверный номер команды\n");

break;

}

}

return 0;

}

void input\_by\_number(int number) {

database[number - 1].name = calloc(20, sizeof(char));

printf("\n[Ввод параметров деталей]\n");

printf("\tВведите название: ");

scanf\_s("%s", database[number - 1].name, 64);

printf("\tВведите количество: ");

scanf\_s("%d", &database[number - 1].amount);

printf("\tВведите вес: ");

scanf\_s("%d", &database[number - 1].weight);

database[number - 1].number = number;

}

void print\_by\_number(int number) {

if (database[number - 1].name != NULL) {

printf("\n[Деталь №%d]\n", number);

printf("\tНазвание: %s\n", database[number - 1].name);

printf("\tКоличество: %d\n", database[number - 1].amount);

printf("\tВес: %d\n", database[number - 1].weight);

}

else

printf(">> Информация о детали №%d отсутствует\n", number);

}

void sort\_by\_weight() {

if (!sorted) {

int start\_from = 0;

int index\_max;

int max\_weight;

while (start\_from < NOTES) {

index\_max = -1;

max\_weight = -1;

for (int i = start\_from; i < NOTES; i++) {

if (database[i].weight > max\_weight) {

max\_weight = database[i].weight;

index\_max = i;

}

}

struct Detail temp = database[start\_from];

database[start\_from] = database[index\_max];

database[index\_max] = temp;

start\_from++;

}

sorted = true;

}

else

printf("\n>> База данных уже отсортированна\n");

}

void print\_sorted() {

if (!sorted)

sort\_by\_weight();

for (int number = 1; number <= NOTES; number++)

{

if (database[number - 1].name != NULL)

print\_by\_number(number);

}

}

void save\_to\_file(char\* path) {

FILE\* file = malloc(sizeof(FILE));

fopen\_s(&file, path, "w");

if (file != 0) {

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

if (database[i].name != NULL) {

fprintf(file, "{n=%s;a=%d;w=%d;№=%d;}\n", database[i].name, database[i].amount, database[i].weight, database[i].number);

}

}

fclose(file);

}

}

void read\_from\_file(char\* path) {

// data template: {n=<char[20]>;a=<int>;w=<int>;№=<int>;e=<char>;}

FILE\* file = malloc(sizeof(FILE));

fopen\_s(&file, path, "r");

const int size = sizeof(char) \* 21 \* sizeof(int) \* 2 \* 200;

char\* data = malloc(size);

struct Detail\* detail = malloc(sizeof(struct Detail));

bool read\_value = false;

bool read\_name = false;

bool read\_amount = false;

bool read\_weight = false;

bool read\_number = false;

int rev = 0;

int num = 0;

int ltr = 0;

double rank = 0;

if (file != 0) {

fscanf\_s(file, "%s", data, size);

fclose(file);

}

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

switch (data[i]) {

case '{':

if (detail != 0) {

free(detail);

detail = malloc(sizeof(struct Detail));

}

break;

case '}':

database[num] = \*detail;

num++;

break;

case '=':

read\_value = true;

break;

case ';':

read\_value = false;

int value = 0;

while (rev > 0) {

rank--;

value += (rev % 10) \* pow(10, rank);

rev /= 10;

}

if (read\_amount) {

detail->amount = value;

}

else if (read\_weight) {

detail->weight = value;

}

break;

case 'n':

if (!read\_value) {

ltr = 0;

detail->name = calloc(20, sizeof(char));

read\_weight = false;

read\_number = false;

read\_amount = false;

read\_name = true;

}

break;

case 'a':

if (!read\_value) {

rev = 0;

read\_name = false;

read\_number = false;

read\_weight = false;

read\_amount = true;

}

break;

case 'w':

if (!read\_value) {

rev = 0;

read\_name = false;

read\_amount = false;

read\_number = false;

read\_weight = true;

}

break;

case '№':

if (!read\_value) {

rev = 0;

read\_name = false;

read\_amount = false;

read\_weight = false;

read\_number = true;

}

break;

default:

if (!read\_value)

continue;

break;

}

if (read\_value) {

if (read\_name && data[i] != '=')

detail->name[ltr++] = data[i];

else if (data[i] >= '0' && data[i] <= '9') {

rev += (data[i] - '0') \* pow(10, rank);

rank++;

}

}

}

free(data);

free(detail);

}

int ask\_number() {

int number;

printf("Введите число: ");

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

return number;

}

char\* ask\_path() {

char\* path = calloc(20, sizeof(char));

printf("Введите путь к базе данных: ");

scanf\_s("%s", path, 20);

return path;

}

**Результат:**

