**Правительство Российской Федерации**

**Федеральное государственное автономное образовательное учреждение высшего профессионального образования   
"Национальный исследовательский университет   
"Высшая школа экономики"**

Московский институт электроники и математики Национального

исследовательского университета "Высшая школа экономики"

Департамент прикладной математики

**ОТЧЕТ**

**По лабораторной работе №11**

**База данных**

**По курсу «Алгоритмизация и программирование»**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  |  | | --- | --- | |  | ФИО студента | | Номер группы | Дата |
| Кармаев Александр Андреевич | БПМ211 | 15.10.2021 |
|  |
|  |
|  |

**Москва – 2021 г.**

**ЗАДАНИЕ (вариант №13)**

*1. Данные должны храниться в бинарном файле прямого доступа.*

*2. Каждая операция с данными базы должна быть реализована как функция или*

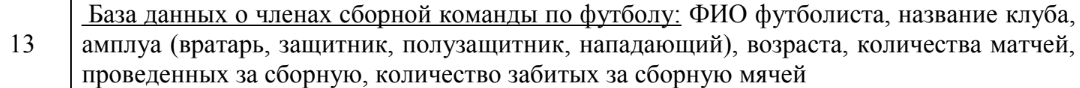
*набор функций.*

*3. Выбор и запуск требуемого режима (действия) осуществляется через меню.*

*4. Реализовать следующие функции обработки базы данных:*

* *Добавление записи в базу*
* *Удаление заданной записи из базы*
* *Поиск записей по заданному пользователем (любому) полю*
* *Редактирование заданной записи в базе*
* *Сортировка данных в базе по заданному пользователем (любому) полю*
* *Вывод на экран содержимого базы в табличном виде*

*5. Решение должно быть многофайловым проектом.*

**

**РЕШЕНИЕ**

**header\_11.h**

struct member { // структура члена сборной команды по футболу

char full\_name[60]; // ФИО

char club[30]; // размеры массивов char выбраны достаточно большими для вмещения требуемых строк

char role[11]; // goalkeeper, defender, midfielder, striker

int age;

int matches;

int goals\_scored;

};

// прототипы функций для работы с базой данных

void add\_data(); // Добавление записи в базу

void remove\_data(); // Удаление заданной записи из базы

void search\_data(); // Поиск записей по заданному пользователем (любому) полю

void edit\_data(); // Редактирование заданной записи в базе

void sort\_database(); // Сортировка данных в базе по заданному пользователем (любому) полю

void output\_database(); // Вывод на экран содержимого базы в табличном виде

int contain\_latin\_letters(char\* str, int len) { // функция проверяет, является ли символ допустимым для использования (латинские буквы и цифры)

char aval[] = "qwertyuiopasdfghjklzxcvbnmQWERTYUIOPASDFGHJKLZXCVBNM "; // массив допустимых символов

int aval\_length = strlen(aval); // подсчитываем длину строки aval c помощью функции из библиотеки string.h

int f = 0;

for (int j = 0; j < len; j++) {

for (int i = 0; i < aval\_length; i++) {

if (aval[i] == str[j]) // если находим совпадение, то возвращаем 1 (правда - символ является допустимым)

f++;

}

}

if(f == len)

return 1;

return 0; // если не будет возвращена 1, то возвращаем 0 (ложь)

}

void number\_input(char\* str, int\* p) { // функция присвоит необходимое(числовое) значение переменной в основной функции, находящейся по адресу(2-ой аргумент)

char string[256];

printf("%s", str);

fgets(string, sizeof(string), stdin);

while (sscanf(string, "%i", p) != 1) {

printf("Incorrect input, try again!! %s", str);

fgets(string, sizeof(string), stdin);

}

}

int num\_len(int num) { // функция возвращает длину числа (~ 7 -> 1; 145 -> 3)

int ans = 0;

do {

ans++;

if (num < pow(10, ans))

return ans;

} while (1);

}

**main.c**

#include <stdio.h>

#include <string.h>

#include <math.h>

#include <stdlib.h>

#include "header\_11.h" // подключаем заголовочный файл со структурой, прототипами функций для работы с БД и вспомогательными функциями

// создаём глобальные переменные, для упрощения работы с функциями, обрабатывающими БД

FILE\* fp; // указатель на поток

struct member\* arr = NULL; // массив, в котором будут храниться структуры участников, для упрощения работы с данными

int arr\_len = 0; // длина массива arr - текущее количество членов сборной

int max\_sizes[] = { 0, 0, 0, 0, 0, 0, 0 }; // массив хранит максимальные длины столбцов по порядку; нужен для вывода БД в табличной форме

void change\_max\_sizes() { // функция редактирует значения максимальных размеров столбцов

for (int i = 0; i < arr\_len; i++) { // проходимся по каждой структуре и изменяем размеры столбцов

if (strlen(arr[i].full\_name) > max\_sizes[1])

max\_sizes[1] = strlen(arr[i].full\_name);

if (strlen(arr[i].club) > max\_sizes[2])

max\_sizes[2] = strlen(arr[i].club);

if (strlen(arr[i].role) > max\_sizes[3])

max\_sizes[3] = strlen(arr[i].role);

if (num\_len(arr[i].age) > max\_sizes[4])

max\_sizes[4] = num\_len(arr[i].age);

if (num\_len(arr[i].matches) > max\_sizes[5])

max\_sizes[5] = num\_len(arr[i].matches);

if (num\_len(arr[i].goals\_scored) > max\_sizes[6])

max\_sizes[6] = num\_len(arr[i].goals\_scored);

}

}

void input\_data(int i) { // функция для ввода данных в элемент массива(структуру) по выбору

printf("\n");

do {

printf("Full name - ");

gets(arr[i].full\_name); // записываем строку с консоли в строковую переменную

} while (!contain\_latin\_letters(arr[i].full\_name, strlen(arr[i].full\_name)));

do {

printf("Club - ");

gets(arr[i].club);

} while (!contain\_latin\_letters(arr[i].club, strlen(arr[i].club)));

do {

printf("Role (goalkeeper, defender, midfielder, striker) - ");

gets(arr[i].role);

} while (!(!strcmp(arr[i].role, "goalkeeper") || !strcmp(arr[i].role, "defender") || !strcmp(arr[i].role, "midfielder") || !strcmp(arr[i].role, "striker")));

number\_input("Age of member - ", &arr[i].age);

number\_input("Number of matches played for team - ", &arr[i].matches);

number\_input("Number of scored goals - ", &arr[i].goals\_scored);

change\_max\_sizes();

}

void add\_data() {

fp = fopen("database.bin", "ab"); // открываем поток на наш файл в режиме добавления данных в конец файла

arr\_len++;

arr = realloc(arr, sizeof(struct member) \* arr\_len);

printf(" Input information about new member:");

getchar();

input\_data(arr\_len - 1);

char\* c = (char\*)&(arr[arr\_len - 1]); // устанавливаем указатель на начало структуры

// посимвольно дописываем в файл структуру

for (int i = 0; i < sizeof(struct member); i++)

{

putc(\*c++, fp);

}

fclose(fp);

}

void remove\_data() {

if (arr\_len == 0) {

printf("There is no elements to remove\n");

return 0;

}

fp = fopen("database.bin", "wb");

int n = 0; // переменная отвечает за позицию члена сборной, чьи данные хотят удалить

output\_database(); // выводим БД, чтобы пользователь мог видеть её перед собой для удобной работы

getchar(); // делаем, чтобы избавиться от \n оставшегося после последнего scanf()

do {

number\_input("Input number of member in the database who you want to remove: ", &n);

} while (!(n > 0 && n <= arr\_len));

arr\_len--;

for (int i = n - 1; i < arr\_len; i++) { // смещаем влево элементы массива, идущие после выбранного - таким образом удаляем его из массива

arr[i] = arr[i + 1];

}

arr = realloc(arr, sizeof(struct member) \* arr\_len);

char\* c = (char\*)arr; // устанавливаем указатель на начало массива структур

// посимвольно записываем в файл массив структур

for (int i = 0; i < sizeof(struct member) \* arr\_len; i++)

{

putc(\*c++, fp);

}

fclose(fp);

// приводим значения ширины столбцов к длине заголовков, чтобы учесть удаление данных

max\_sizes[0] = strlen("#");

max\_sizes[1] = strlen("Full name");

max\_sizes[2] = strlen("Club");

max\_sizes[3] = strlen("Role");

max\_sizes[4] = strlen("Age");

max\_sizes[5] = strlen("Matches");

max\_sizes[6] = strlen("Goals");

change\_max\_sizes();

}

void search\_data() {

if (arr\_len == 0) {

printf("There is nothing to search\n");

return 0;

}

output\_database();

getchar();

int n = 0;

do {

number\_input("\n Input number of column by which you want to search(1 - Full name; 2 - Club...) -> ", &n);

} while (!(n > 0 && n <= 6));

place2:;

int f = 0; // флаг, который будет отвечать за наличие введённого значения в столбце в БД, чтобы можно было попросить пользователя заново ввести значение

printf("Input value from chosen column: ");

if (n > 3) { // разбиваем задачу отдельно на работу с числами и строками для экономии памяти

int value = 0; // значение, по которому будем искать структуру

number\_input("", &value);

printf("\n");

for (int i = 0; i < arr\_len; i++) {

if (n == 4 && value == arr[i].age || n == 5 && value == arr[i].matches || n == 6 && value == arr[i].goals\_scored) { // выводим все данные по члену сборной, у которого найдено соответствие

printf("%d -> Full name - %s; Club - %s; Role - %s; Age - %d; Matches number - %d; Goals number - %d.\n", i+1, arr[i].full\_name, arr[i].club, arr[i].role, arr[i].age, arr[i].matches, arr[i].goals\_scored);

f = 1;

}

}

}

else {

char value[60];

gets(value);

printf("\n");

for (int i = 0; i < arr\_len; i++) {

if (n == 1 && !strcmp(value, arr[i].full\_name) || n == 2 && !strcmp(value, arr[i].club) || n == 3 && !strcmp(value, arr[i].role)) {

printf("%d -> Full name - %s; Club - %s; Role - %s; Age - %d; Matches number - %d; Goals number - %d.\n", i + 1, arr[i].full\_name, arr[i].club, arr[i].role, arr[i].age, arr[i].matches, arr[i].goals\_scored);

f = 1;

}

}

}

if (f == 0) {

printf(" There is no such value in this column. ");

goto place2;

}

else {

printf(" It's all members of national team with inputed value in the chosen column\n");

}

}

void edit\_data() {

if (arr\_len == 0) {

printf("There are no elements to edit\n");

return 0;

}

fp = fopen("database.bin", "wb");

int n = 0;

output\_database();

getchar();

do {

number\_input("Input number of member in the database whose data you want to edit: ", &n);

} while (!(n > 0 && n <= arr\_len));

input\_data(n - 1);

fclose(fp);

}

void sort\_database() {

if (arr\_len == 0) {

printf("There is no data to sort\n");

return 0;

}

output\_database();

getchar();

fp = fopen("database.bin", "wb");

int choise = 0;

do {

number\_input("\n Input number of column(1 - Full name; 2 - Club...) -> ", &choise);

} while (!(choise > 0 && choise <= 6));

for (int i = 1; i < arr\_len; i++) { // сортировка чисел по убыванию и строк в алфавитном порядке (sort algorithm - insertsort)

int j = i;

while (j > 0 && (strcmp(arr[j - 1].full\_name, arr[j].full\_name) > 0 && choise == 1 || strcmp(arr[j - 1].club, arr[j].club) > 0 && choise == 2 || strcmp(arr[j - 1].role, arr[j].role) > 0 && choise == 3 || arr[j - 1].age < arr[j].age && choise == 4 || arr[j - 1].matches < arr[j].matches && choise == 5 || arr[j - 1].goals\_scored < arr[j].goals\_scored && choise == 6)) {

struct member k = arr[j - 1];

arr[j - 1] = arr[j];

arr[j] = k;

j--;

}

}

fwrite(arr, sizeof(struct member), arr\_len, fp);

fclose(fp);

}

void output\_database() {

fp = fopen("database.bin", "rb");

if (arr == NULL) {

printf("There aren't any elements in database. First of all input data\n");

return 0;

}

fread(arr, sizeof(struct member), arr\_len, fp); // записываем данные из потока в массив

if (num\_len(arr\_len) > max\_sizes[0]) // обрабатываем размер БД для возможного увеличения первого столбца

max\_sizes[0] = num\_len(arr\_len);

printf(" Members of national team:\n\n");

// выводим БД в табличном виде путём настраивания ширины поля и выравнивания по левому краю

printf("%-\*s | %-\*s | %-\*s | %-\*s | %-\*s | %-\*s | %-\*s\n", max\_sizes[0], "#", max\_sizes[1], "Full name", max\_sizes[2], "Club", max\_sizes[3], "Role", max\_sizes[4], "Age", max\_sizes[5], "Matches", max\_sizes[6], "Goals");

for (int i = 0; i < arr\_len; i++)

printf("%-\*d | %-\*s | %-\*s | %-\*s | %-\*d | %-\*d | %-\*d\n", max\_sizes[0], i + 1, max\_sizes[1], arr[i].full\_name, max\_sizes[2], arr[i].club, max\_sizes[3], arr[i].role, max\_sizes[4], arr[i].age, max\_sizes[5], arr[i].matches, max\_sizes[6], arr[i].goals\_scored);

fclose(fp);

}

int main(void) {

remove("database.bin"); // удаляем созданный в прошлый раз файл с базой данных

// приводим значения ширины столбцов к длине заголовков

max\_sizes[0] = strlen("#");

max\_sizes[1] = strlen("Full name");

max\_sizes[2] = strlen("Club");

max\_sizes[3] = strlen("Role");

max\_sizes[4] = strlen("Age");

max\_sizes[5] = strlen("Matches");

max\_sizes[6] = strlen("Goals");

int f; // переменная отвечает за выбранный пользователем режим из меню

printf("\tMenue:\n1 -> Add data\n2 -> Remove data\n3 -> Search data\n4 -> Edit data\n5 -> Sort database\n6 -> Output database\n0 -> End process\n");

do {

place1:;

printf("\nInput number of the option from menue: ");

scanf("%d", &f);

switch (f) {

case 0:

return 0;

break;

case 1:

add\_data();

break;

case 2:

remove\_data();

break;

case 3:

search\_data();

break;

case 4:

edit\_data();

break;

case 5:

sort\_database();

break;

case 6:

output\_database();

break;

default:

printf("Wrong input, make it again!! ");

goto place1;

break;

}

} while (f != 0);

return 0;

}

**ТЕСТЫ**

**Тест № 1** // обычный пример (добавление данных и вывод)

**Изображение выглядит как текст

Автоматически созданное описание**

**Тест № 2** // пример содержит использование функции удаления записи из БД

**Изображение выглядит как текст

Автоматически созданное описание**

**Тест № 3** // пример содержит использование функции поиска записи по полю // данные были введены такие же, как и в первом примере

**Изображение выглядит как текст

Автоматически созданное описание**

**Тест № 4** // пример содержит использование функции редактирования выбранной записи // данные были введены такие же, как и в первом примере

**Изображение выглядит как текст

Автоматически созданное описание**

**Тест № 4** // пример содержит использование функции сортировки по столбцу **Изображение выглядит как текст

Автоматически созданное описание** **Изображение выглядит как текст, табличка

Автоматически созданное описание** Изображение выглядит как текст, экран, черный, снимок экрана

Автоматически созданное описание

**Тест № 6** // пример содержит обработку исключений при попытке провести операции с пустой БД

**Изображение выглядит как текст

Автоматически созданное описание**

**// также программа стабильно работает при использовании функций меню в произвольном порядке**

**// обрабатываются различные исключения по нарушению ввода требуемых значений**

**Тест № 7**

**Изображение выглядит как текст

Автоматически созданное описание**