Министерство образования и науки Российской Федерации

федеральное государственное бюджетное образовательное

учреждение высшего образования

«Алтайский государственный технический университет

им. И.И. Ползунова»

Факультет \_\_информационных технологий\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кафедра \_\_\_прикладной математики\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Специальность (направление, профиль) \_\_ПИ\_\_\_\_\_\_\_\_\_\_\_\_\_

Курсовой проект

защищен с оценкой\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Е.В. Егорова

(подпись руководителя проекта) (инициалы, фамилия)

“\_\_\_\_”\_\_\_\_\_\_\_\_\_\_\_ 2024 г.

Курсовой проект

\_\_\_\_\_\_\_\_\_Разработка программного обеспечения для хранения и анализа

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_списка видеоигр\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(тема курсового проекта)

Пояснительная записка

по дисциплине \_\_\_\_\_\_Программирование\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_КП 09.03.04.61.000 ПЗ\_\_\_\_\_\_\_

(обозначение документа)

Студент группы ПИ32 Самохвалов Д.А. \_\_\_\_\_\_\_\_\_\_\_\_01.04.2024

(фамилия, имя, отчество) (подпись) (дата)

Руководитель

проекта доцент, к.т.н.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Е.В. Егорова

(должность, ученое звание) (подпись) (инициалы, фамилия)

БАРНАУЛ 2024

Министерство науки и высшего образования Российской Федерации

ФГБОУ ВО «Алтайский государственный технический университет

имени И.И. Ползунова»

Факультет информационных технологий

Кафедра «Прикладная математика»

З А Д А Н И Е

на курсовой проект по дисциплине «Программирование»

студенту группы ПИ-32 Самохвалову Даниилу Алексеевичу

Тема курсового проекта: «Разработка программного обеспечения для хранения и анализа списка видеоигр».

Календарный план работы:

|  |  |  |
| --- | --- | --- |
| № этапа | Содержание этапа | Недели семестра |
| 1 | Получение задания | 1 |
| 2 | Постановка задачи. Работа с документацией | 2 |
| 3 | Проектирование программы | 3-4 |
| 4 | Реализация программы | 5-13 |
| 5 | Оформление пояснительной записки | 14 |
| 6 | Защита курсового проекта | 15-16 |

Руководитель проекта\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Егорова Е.В., доцент

подпись

Дата выдачи задания 12 марта 2024г.

число месяц год

Задание принял к исполнению­­  Самохвалов Д.А.

подпись

Содержание

[Условные обозначения 6](#_Toc163033957)

[Введение 7](#_Toc163033958)

[1 Обзор предметной области и постановка задачи 8](#_Toc163033959)

[2 Проектирование 9](#_Toc163033960)

[3 Реализация 10](#_Toc163033961)

[Заключение 14](#_Toc163033962)

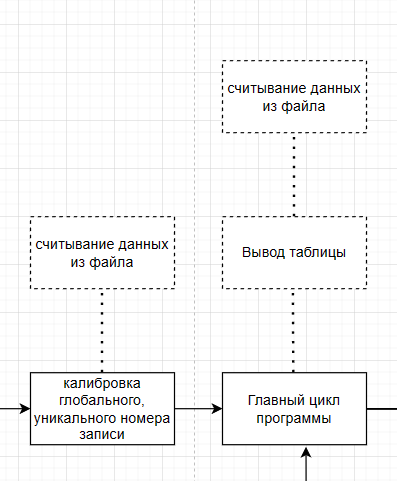
[Список использованной литературы 14](#_Toc163033963)

[Приложение А 15](#_Toc163033964)

[Приложение Б 34](#_Toc163033965)

# Условные обозначения

Обозначения в блок схеме:



Пунктиром обозначены дополнительные вызовы функций.

Стрелка указывает иерархический спуск (какие функции вызываются из определенной функции)

# **Введение**

Проблема хранения и структуризации информации всегда была актуальной для различных проектов. Сегодня эта задача может быть упрощена с помощью работы с электронными таблицами и базами данных. Работа с пользовательскими данными в информационных технологиях требует от разработчиков программного обеспечения опыта и ответственности. Цель данной работы - изучение принципов создания консольных приложений для работы с большими наборами данных. Главная задача - написать программу для работы в консольном режиме, которая поможет организовать сбор, хранение и обработку структурированных данных определенной направленности.

# 1 Обзор предметной области и постановка задачи

1.1 Обзор предметной области

*База данных* – это **упорядоченный набор структурированной информации или данных, которые обычно хранятся в электронном виде в компьютерной системе**. База данных обычно управляется системой управления базами данных (СУБД). Данные вместе с СУБД, а также приложения, которые с ними связаны, называются системой баз данных, или, для краткости, просто базой данных.

Базы данных используются:

* Для хранения данных
* Для взаимодействия с данными

Режимы работы с базами данных:

* *Получение данных*– к примеру, запрос определенных записей, удовлетворяющих некоторому условию
* *Редактирование*– создания или изменения структуры базы и создания ее объектов

*СУБД* – программное обеспечение, предназначенное для работы с базами данных.

Основные функции СУБД:

* управление данными во внешней памяти (на дисках);
* управление данными в оперативной памяти;

1.2 Постановка задачи

Организовать работу с базой данных, состоящих из записей. Каждая запись содержит:

-номер игры

− название игры; − размер на диске (в байтах);

− возрастной рейтинг (6+, 12+, 18+, и т.д.);

− графика (2 -хорошая, 1 -средняя, 0 -плохая);

− ограничение на число игроков, которые могут одновременно играть (0- нет ограничений, 1- индивидуальная игра);

− средний рейтинг критиков (по 100-бальной шкале).

Например: Крестики-нолики 1000 6+ 1 1 57

В программе следует организовать необходимые защиты от неправильного ввода данных:

* ввод букв вместо чисел;
* ввод чисел, значение которых выходит за требуемый диапазон;
* попытка открыть для чтения несуществующий файл;
* открытие файла в режиме “w” без предварительной проверки на возможное существование файла;
* попытка открыть файла с запрещенными для использования именами типа “con”, “prn” и т.д.

В программе должен быть реализован дружественный интерфейс, предполагающий диалоговое меню для пользователя и удобный, однозначно понятный ввод данных и вывод данных.

# 2 Проектирование

2.1 Алгоритм решения

Программа работает в диалоговом режиме.

При входе в программу пользователя спрашивают о базе данных, которую он хотел бы использовать, далее основное меню программы, которое включает в себя:

* ввод и корректировка базы данных;
* вывод содержимого базы данных;
* запрос;
* выбор файла
* сохранение базы данных
* выход из программы.

При входе в каждый блок можно вернуться назад, написав команду, либо после выполнения определенного действия, программа спросит: остаться в данном блоке или подняться выше на один уровень.

2.2 Структура данных

Для решения поставленной задачи в файле реализована база данных, структура которой представлена следующим образом:

* number – уникальный номер записи
* *name –* наименование игры;
* *memoryD* – вес игры;
* ageR – возрастной рейтнг
* *graphic* – качество графики
* *plrL* – ограничение количества игроков
* *arate* – средний рейтинг критиков

# 3 Реализация

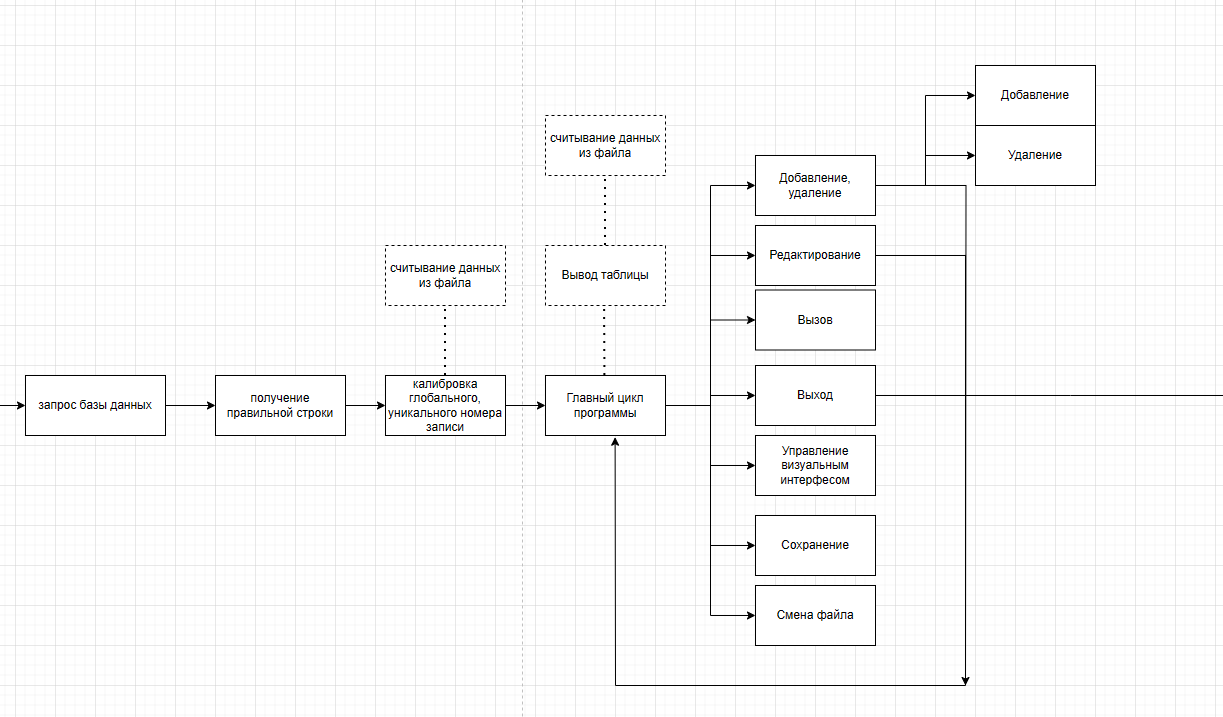
3.1 Выбор средств реализации

Данное ПО реализовано на языке С с использованием среды разработки программного Microsoft Visual Studio Code. Язык С – компилируемый статически слаботипизированный язык программирования общего назначения.

Для полноценного функционирования программы требуется:

* ОС Windows XP или выше;
* Оперативная память 10МБ или выше;
* Жесткий диск: 10МБ для хранения программы нужно 100КБ; дополнительное место для хранения файлов баз данных;
* .NET Framework версии 2.0 или выше;
* Наличие мыши (или её аналоги), клавиатуры, монитора.

3.2 Структура программы



3.3 Состав программы

Функции и процедуры:

1. Интерфейс, предоставляющий работу с файлом

void closedb(struct DataBase \* db); - представляет из себя процедуру деинициализации передаваемого ей аргумента типа DataBase, по средствам сброса файлового указателя и закрытия файла.

void resetPtr(struct DataBase \* db) – сброс файлового указателя, перевод указателя на начало файла.

char \* getLine(struct DataBase \* db) – получение строки из файла; используется как построчное считывания строк из фала; возвращает дескриптор локальной переменной.

void writeLine(struct DataBase \* db, char \* data) – запись строки в файл.

void dbconnect(struct DataBase \* db)- инициализирует базу данных, открывает файл.

void dbcpy(struct DataBase \* dbt, struct DataBase \* dbf) – нетривиальное копирование баз данных. Процедура не копирует указатель, а открывает файл, пользуясь данными, которые мы хотим скопировать

void initDbWithFile(struct DataBase \* db, const char \* filename, char mode [])- инициализирует базу данных, открывая файл с указанным именем и в указанном режиме.

void initDefault(struct DataBase \* db) – инициализация по умолчанию.

1. Интерфейс, контролирующий работу с базами данных

char isFileExistInCDir(char \* f) – проверяет есть ли файл директории, где была запущена программа.

void appendFileName(char \* fn)- при создании файла добавляет его название в список имен созданных файлов.

void openReadonlyConnect(struct DataBase \* base) – подключается к базе данных в режиме “только чтение”

void openOutputAndReadConnect(struct DataBase \* base) – подключается к базе данных в режиме “добавление и чтение”

void openWriteConnect(struct DataBase \* base) – подключается к базе данных в режиме “только запись”

void closeConnect(struct DataBase \* b)- отключается от базы данных.

void pushRecord(struct DataBase \* b) - приводит все типы данных в строки, собирает из них одну и записывает ее в файл, с определенным форматом.

int nextRecord(struct DataBase \* b) – итерирует по записям(в глубине: по строкам файла) записывает найденную запись в глобальную переменную. Возвращает NULL, если записи кончились, 1 – в случае успешного считывания.

void outRecord(const struct Record \*rec) – выводит запись в определенном формате.

void openBufferDb(char \* name) – создает буфферную базу данных. К примеру, для удаления записи из базы данных, по средствам переноса всех записей в другую базу данных, кроме удаляемой. Добавление и чтение.

void openBufferDbW(char \* name) – создает буфферную базу данных только для записи.

void closeBufferDb() – закрывает буффер.

void deleteRecordByNumber(int nn) – удаляет запись из базы данных по номеру записи.

struct Record \* findRecordByNumber (int nn) - находит запись по номеру; возвращает дескриптор локального объекта, которому присваивается найденная запись; такой подход не использует глобальную переменную recordFromFile в качестве результата, что позволяет без опаски потери данных использовать ее после вызова функции. Есть другой подход: использование глобальной переменной recordFromFile, но тогда, после вызова функции, пользователь обязан сразу обработать результат, записанный в recordFromFile.

1. Интерфейс, задающий сценарий программы

void outFileBar() – вывод панели управления порождаемыми файлами.

void event\_add() – сценарий добавления записи.

void event\_remove() – сценарий удаления записи.

void event\_outTable() – сценарий вывода таблицы.

void event\_addRemove() – блок ввода/вывода.

void event\_edit() – сценарий редактирования записи.

void inputFieldsForCall(struct Call \*cl) – ввод полей для запроса.

void call\_init\_default(struct Call \*cl) – инициализация запроса по умолчанию.

void event\_call() – алгоритм, реализующий запрос.

void event\_exit() – намеренный выход из программы.

int init\_focusfile(int val) – инициализация переменной, которая указывает на файл, выделенный пользователем.

void event\_swapfile\_left() – позволяет пользователю выбрать другой файл на панели управления фалами.

void event\_swapfile\_right() аналогично.

void event\_run\_focus\_file() – меняет имя запускаемого файла. ???

void event\_save() – сценарий сохранения изменений в базе данных.

void mainLoop() – основной цикл программы, он же и задает сценарий главного меню.

void setGlobalNumberByFile() – выставляет глобальный номер записи, так, что у каждой записи будет уникальный номер.

# Заключение

Был создан программный продукт, позволяющий эффективно и безопасно работать с базой данных видеоигр. Реализованы защиты на ввод некорректных данных, эффективная обработка запроса, редактирование базы данных.

В процессе выполнения задания были подробно изучены основы языка С.

Преимущества приложения:

* Интуитивно понятный интерфейс;
* Быстрая и эффективная обработка запроса;
* Возможность оперативного редактирования/ добавления/ удаления записей;
* Минималистический дизайн;
* Легкая возможность переключения базы данных, почти не влияющая на производительность

Недочёты приложения:

* Работа только с одной базой одновременно;
* Нету поддержки русского языка

Возможны дальнейшие усовершенствования программного продукта с течением времени:

* Работа с несколькими базами одновременно;

*Примечание: в какой-то мере это было реализовано: пользователь может переключаться между порожденными базами данных. Для работы с несколькими базами данных программа должна будет принимать названия этих баз на диске, алгоритм реализации переключения между ними будет схож.*

* Сохранение результатов запроса в файлы;
* Усовершенствование визуального интерфейса и полный отказ от команд.
* Добавить русский язык.
* Доработка кода; некоторые из типов не отслеживают свой инвариант.

# Список использованной литературы

1. [Stack Overflow - Where Developers Learn, Share, & Build Careers](https://stackoverflow.com/)
2. Егорова Е.В. Программирование на языке СИ. Учебное пособие/Алт. Госуд. Технич. Ун-т им. И.И.Ползунова.-Барнаул:2013.-184с.
3. [**cppreference.com**](https://en.cppreference.com/)

# Приложение А

**Код программы**

**/\***

**main.c**

**Основной модуль, драйвер**

**\*/**

**#include <stdio.h>**

**#include <conio.h>**

**#include <locale.h>**

**#include "controlBase.c"**

**#include "inputFuncs.c"**

**#define DEFAULT\_INIT\_STRING "\_\_DFINIT\0"**

**#define DEFAULT\_INIT\_INT -1**

**#define RESET "\033[0m"**

**#define RED "\033[1;31m"**

**#define YELLOW "\033[1;33m"**

**int globalNumberOfRecord = 0;**

**#ifdef \_WIN32**

**void clearConsole()**

**{ system("cls"); }**

**#endif**

**#ifdef linux**

**void clearConsole()**

**{ system("clear") }**

**#endif**

**// FILE BAR INTERFACE**

**int focusfile = DEFAULT\_INIT\_INT;**

**void outFileBar()**

**{**

**int ptrid = 0;**

**char \*begin = createdFiles[ptrid];**

**char \*end = lastcratedFile;**

**while (begin != end)**

**if (focusfile != ptrid)**

**printf("%s %c ", (begin = createdFiles[ptrid++]) , 29 );**

**else**

**printf("%s%s%s %c ", YELLOW, (begin = createdFiles[ptrid++]), RESET, 29 );**

**if (focusfile == DEFAULT\_INIT\_INT)**

**printf("%spress 8 and 9 for move between files%s", YELLOW, RESET);**

**else**

**printf("press 8 and 9 for move between files");**

**printf("\n\nActive file now = %s ", curr\_base.\_fName);**

**}**

**///**

**// zone ADD REMOVE**

**void event\_add()**

**{**

**openOutputAndReadConnect(&curr\_base);**

**printf("\n\nAdd record.\n");**

**recordToFile.number = globalNumberOfRecord++;**

**fillRecord(&recordToFile);**

**pushRecord(&curr\_base);**

**closeConnect(&curr\_base);**

**}**

**void event\_remove()**

**{**

**openReadonlyConnect(&curr\_base);**

**int num;**

**// файл пуст**

**if (nextRecord(&curr\_base) == NULL)**

**{**

**printf("\nHmm.. file is empty.\n");**

**closeConnect(&curr\_base);**

**return;**

**}**

**resetPtr(&curr\_base);**

**printf("\nEnter the number of record > ");**

**waitInt(&num);**

**if (findRecordByNumber(num) == NULL)**

**{**

**printf("Oops.. record with this number do not exist.");**

**waitButtom();**

**return;**

**}**

**printf("\nAre you sure? Enter 'y' for accept..\n");**

**if (getchar() == 'y')**

**deleteRecordByNumber(num);**

**closeConnect(&curr\_base);**

**}**

**///**

**void event\_outTable()**

**{**

**openReadonlyConnect(&curr\_base);**

**printf("\n Number\t\t Name(0) \t Memory(1) \t Age(2) \t Graphic(3) \t Players(4) \t Rating(5)\n");**

**printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n\n");**

**while (nextRecord(&curr\_base))**

**{**

**outRecord(&recordFromFile);**

**putchar('\n');**

**}**

**closeConnect(&curr\_base);**

**}**

**void event\_addRemove()**

**{**

**int inputcmd;**

**do**

**{**

**clearConsole();**

**event\_outTable();**

**printf("\n\n1) Add / 2) Remove / 3) Exit > ");**

**waitInt(&inputcmd);**

**if (inputcmd == 1) event\_add();**

**if (inputcmd == 2) event\_remove();**

**if (inputcmd == 3) return;**

**printf("\n\nYou can press ESC, for exit the block.");**

**} while (\_getch() != 27);**

**}**

**//**

**void event\_edit()**

**{**

**struct Record \*save\_rec;**

**int num;**

**int field;**

**openOutputAndReadConnect(&curr\_base);**

**if (nextRecord(&curr\_base) == NULL) {**

**printf("\nHmm.. file is empty.\n");**

**closeConnect(&curr\_base);**

**waitButtom();**

**return;**

**}**

**do**

**{**

**clearConsole();**

**event\_outTable();**

**openOutputAndReadConnect(&curr\_base);**

**printf("\n\nEnter the number of record or \"-1\" for exit: ");**

**waitInt(&num);**

**if (num == -1) return;**

**printf("\n\nEnter ID of field: ");**

**waitInt(&field);**

**// Подготавливаем новую запись на основе старой**

**save\_rec = findRecordByNumber(num);**

**if (save\_rec == NULL) {**

**printf("Oops.. record with this number do not exist.");**

**waitButtom();**

**return;**

**}**

**fillField(save\_rec, field);**

**printf("\nAfter the changes, you will get this record:\n\n");**

**outRecord(save\_rec);**

**printf("\n\nEnter '1' for accept, '0' for decline\n> ");**

**if (getchar() == '1')**

**deleteRecordByNumber(num);**

**else { closeConnect(&curr\_base); continue; }**

**recordToFile = \*save\_rec;**

**pushRecord(&curr\_base);**

**closeConnect(&curr\_base);**

**} while ((printf("\n\nYou can press ESC, for exit the block.")), \_getch() != 27);**

**}**

**void inputFieldsForCall(struct Call \*cl)**

**{**

**char fields[8];**

**printf("\n\nCall block.\n\n- For exit enter 'exit' or '-1'. \n");**

**printf("\nintrested field> ");**

**gets\_s(fields, 7);**

**if (stricmp("exit", fields) == 0 || stricmp("-1", fields) == 0) return;**

**int inputField = 0;**

**for (int i = 0; fields[i] != '\0'; ++i)**

**{**

**inputField = fields[i] - '0';**

**if (inputField <= 5 && inputField >= 0) printf("\n\033[0;36mEntering data for the call. Field < %d >%s\n", inputField, RESET);**

**switch (inputField)**

**{**

**case NAME: inputNameForCall(cl);**

**break;**

**case MEM: inputMemForCall(cl);**

**break;**

**case AGER: inputAgerateForCall(cl);**

**break;**

**case GRAPHIC: inputGraphic(&cl->graphic);**

**break;**

**case PLAYERS: inputPlayerAm(&cl->plrL);**

**break;**

**case AVERATE: inputAverForCall(cl);**

**break;**

**default: printf("\n\nWarning. %sIncorrect%s input was ignored : %s%c%s\n", RED, RESET, RED, fields[i], RESET);**

**}**

**}**

**}**

**void call\_init\_default(struct Call \*cl)**

**{**

**strcpy(cl->name, DEFAULT\_INIT\_STRING);**

**cl->mem = DEFAULT\_INIT\_INT;**

**cl->ageR = DEFAULT\_INIT\_INT;**

**cl->graphic = DEFAULT\_INIT\_INT;**

**cl->plrL = DEFAULT\_INIT\_INT;**

**cl->arate = DEFAULT\_INIT\_INT;**

**}**

**void event\_call()**

**{**

**struct Call cl;**

**call\_init\_default(&cl);**

**inputFieldsForCall(&cl);**

**const struct Record \*rec;**

**char isNeedRec;**

**openReadonlyConnect(&curr\_base);**

**printf("\n \033[44;1m -\*- Games for you:%s\n", RESET);**

**while (nextRecord(&curr\_base))**

**{**

**isNeedRec = 1;**

**rec = &recordFromFile;**

**if (strcmp(cl.name, DEFAULT\_INIT\_STRING) != 0)**

**if (cl.nc == cl.ABSOLUTE)**

**isNeedRec = (stricmp(cl.name, rec->gmst.name) == 0);**

**else**

**isNeedRec = (strstr(rec->gmst.name, cl.name) != NULL);**

**if (cl.mem != DEFAULT\_INIT\_INT && isNeedRec)**

**if (cl.bmem == cl.LESS) isNeedRec = (rec->gmst.memoryD < cl.mem);**

**else isNeedRec = (rec->gmst.memoryD > cl.mem);**

**if (cl.ageR != DEFAULT\_INIT\_INT && isNeedRec)**

**isNeedRec = (cl.ageR >= rec->gmst.ageR);**

**if (cl.graphic != DEFAULT\_INIT\_INT && isNeedRec)**

**isNeedRec = (cl.graphic == rec->gmst.graphic);**

**if (cl.plrL != DEFAULT\_INIT\_INT && isNeedRec)**

**isNeedRec = (cl.plrL == rec->gmst.plrL);**

**if (cl.arate != DEFAULT\_INIT\_INT && isNeedRec)**

**if (cl.barate == cl.MORE) isNeedRec = (cl.arate < rec->gmst.arate);**

**else isNeedRec = (cl.arate >= rec->gmst.arate);**

**if (isNeedRec){**

**outRecord(rec);**

**printf("\n");**

**}**

**}**

**waitButtom();**

**closeConnect(&curr\_base);**

**}**

**void event\_exit()**

**{**

**exit(EXIT\_SUCCESS);**

**}**

**void event\_invalidcmd(){}**

**int init\_focusfile(int val)**

**{**

**static int init = (focusfile = val);**

**return 0;**

**}**

**void event\_swapfile\_left()**

**{**

**static int init = init\_focusfile(amfiles);**

**if (focusfile != 0) focusfile -= 1;**

**}**

**void event\_swapfile\_right()**

**{**

**static int init = init\_focusfile(0);**

**if (focusfile < amfiles - 1) focusfile += 1;**

**}**

**void event\_run\_focus\_file()**

**{**

**if (amfiles > 0 && focusfile != -1)**

**strcpy(curr\_base.\_fName, createdFiles[focusfile]);**

**}**

**void event\_save()**

**{**

**if (stricmp(createdFiles[0], curr\_base.\_fName) == 0) return;**

**openReadonlyConnect(&curr\_base);**

**openBufferDbW(createdFiles[0]);**

**while (nextRecord(&curr\_base))**

**{**

**recordToFile = recordFromFile;**

**pushRecord(&temp\_base);**

**}**

**closeBufferDb();**

**closeConnect(&curr\_base);**

**}**

**void mainLoop()**

**{**

**clearConsole();**

**printf("\n");**

**outFileBar();**

**printf("\n\n");**

**event\_outTable();**

**printf("\nSelect the block you need.");**

**printf("\n1) Add / Remove");**

**printf("\n2) Edit");**

**printf("\n3) Call");**

**printf("\n4) Exit");**

**printf("\nR) Run focused file");**

**printf("\nS) Save changes");**

**printf("\nThe button is expected to be pressed to go to the block...");**

**switch (\_getch())**

**{**

**case 49: event\_addRemove(); // '1'**

**break;**

**case 50: event\_edit();**

**break;**

**case 51: event\_call();**

**break;**

**case 52: event\_exit();**

**break;**

**case 56: event\_swapfile\_left(); // '8'**

**break;**

**case 57: event\_swapfile\_right(); // '9'**

**break;**

**case 114: event\_run\_focus\_file(); // 'R'**

**break;**

**case 115: event\_save(); // 'S'**

**break;**

**default:**

**event\_invalidcmd();**

**break;**

**}**

**}**

**void intro()**

**{**

**printf("\n\n\t\t\tWelcome!");**

**printf("\n\t\t\tDeveloper -> https://github.com/eastpriSs");**

**\_getch();**

**}**

**void setGlobalNumberByFile()**

**{**

**openReadonlyConnect(&curr\_base);**

**while (nextRecord(&curr\_base))**

**{**

**if (globalNumberOfRecord < recordFromFile.number)**

**globalNumberOfRecord = recordFromFile.number;**

**}**

**++globalNumberOfRecord;**

**closeConnect(&curr\_base);**

**}**

**int main(){**

**setlocale(LC\_ALL, "rus");**

**intro();**

**clearConsole();**

**enterFileName(curr\_base.\_fName); // ???**

**clearStdin();**

**setGlobalNumberByFile();**

**while (1) mainLoop();**

**}**

**/\***

**controlBase.c**

**Реализация работы с базой данных**

**\*/**

**#include <stdio.h>**

**#include <string.h>**

**#include <stdlib.h>**

**#define MAX\_FILENAME 255**

**#define MAX\_FILEMODE 10**

**#define MAX\_LINE\_SIZE 500**

**#define CORRECT\_RETURN 0**

**#define FAILURE\_RETURN 1**

**#define UNDEFINED\_STRING "!UNDF\0"**

**struct DataBase**

**{**

**char \_fName [MAX\_FILENAME];**

**char \_fMode [MAX\_FILEMODE];**

**FILE \*\_ptrFile;**

**char \_isDef;**

**};**

**void closedb(struct DataBase \* db);**

**void initDefault(struct DataBase \* db)**

**{**

**db->\_ptrFile = NULL;**

**db->\_isDef = 0;**

**strcpy(db->\_fName, UNDEFINED\_STRING);**

**strcpy(db->\_fMode, UNDEFINED\_STRING);**

**}**

**void initDbWithFile(struct DataBase \* db, const char \* fileName,**

**char mode [])**

**{**

**if (db->\_isDef) fclose(db->\_ptrFile);**

**strcpy(db->\_fName, fileName);**

**strcpy(db->\_fMode, mode);**

**db->\_isDef = 1;**

**}**

**void changeFMode(struct DataBase \* db, char mode [])**

**{**

**if (db->\_isDef) fclose(db->\_ptrFile);**

**else return;**

**strcpy(db->\_fMode, mode);**

**}**

**void changeFName(struct DataBase \* db, char name [])**

**{**

**if (db->\_isDef) fclose(db->\_ptrFile);**

**else return;**

**strcpy(db->\_fName, name);**

**}**

**void dbcpy(struct DataBase \* dbt, struct DataBase \* dbf)**

**{**

**dbt->\_ptrFile = fopen(dbf->\_fName, dbf->\_fMode);**

**strcpy(dbt->\_fMode, dbf->\_fMode);**

**strcpy(dbt->\_fName, dbf->\_fName);**

**dbt->\_isDef = dbf->\_isDef;**

**}**

**void dbconnect(struct DataBase \* db)**

**{**

**db->\_ptrFile = fopen(db->\_fName, db->\_fMode);**

**}**

**void \_nextLine(struct DataBase \* db)**

**{**

**fputc('\n', db->\_ptrFile);**

**}**

**void writeLine(struct DataBase \* db, char \* data)**

**{**

**fputs(data, db->\_ptrFile);**

**fputc('\n', db->\_ptrFile);**

**}**

**char \* getLine(struct DataBase \* db)**

**{**

**static char line [MAX\_LINE\_SIZE];**

**if (fgets(line, MAX\_LINE\_SIZE, db->\_ptrFile)) return line;**

**return NULL;**

**}**

**void resetPtr(struct DataBase \* db)**

**{**

**fseek(db->\_ptrFile, 0, SEEK\_SET);**

**}**

**void closedb(struct DataBase \* db)**

**{**

**resetPtr(db);**

**fclose(db->\_ptrFile);**

**}**

**// DATABASE**

**#define MAX\_SIZE\_GNAME 255**

**#define MAX\_BASEPOOL 10**

**#define FILE\_LINE\_SEPORATOR\_S ";"**

**#define MAX\_CHANGES 100**

**char createdFiles [MAX\_CHANGES][MAX\_FILENAME];**

**char \*lastcratedFile = \*createdFiles;**

**int amfiles = 0;**

**typedef struct DataBase DataBase;**

**struct DataBase curr\_base;**

**struct DataBase temp\_base;**

**struct Record**

**{**

**int number;**

**struct GameStat**

**{**

**char name [MAX\_SIZE\_GNAME];**

**int memoryD;**

**int ageR;**

**int graphic;**

**int plrL;**

**int arate;**

**} gmst;**

**} recordToFile, recordFromFile;**

**struct Call**

**{**

**enum nameCoincedence {ABSOLUTE, INTER};**

**int nc;**

**char name[MAX\_SIZE\_GNAME];**

**enum borders {MORE, LESS};**

**int bmem;**

**int mem;**

**int ageR;**

**int graphic;**

**int plrL;**

**int barate;**

**int arate;**

**};**

**char isFileExistInCDir(char \* f)**

**{**

**if (fopen(f, "r") == NULL) return 0;**

**return 1;**

**}**

**void appendFileName(char \* fn)**

**{**

**for (int i = 0; i < amfiles; ++i)**

**if (strcmp(createdFiles[i], fn) == 0) return;**

**lastcratedFile = createdFiles[amfiles++];**

**strcpy(lastcratedFile, fn);**

**}**

**void openReadonlyConnect(struct DataBase \* base)**

**{**

**appendFileName(base->\_fName);**

**initDbWithFile(base, base->\_fName, "r");**

**dbconnect(base);**

**}**

**void openOutputAndReadConnect(struct DataBase \* base)**

**{**

**appendFileName(base->\_fName);**

**initDbWithFile(base, base->\_fName, "a+");**

**dbconnect(base);**

**}**

**void openWriteConnect(struct DataBase \* base)**

**{**

**appendFileName(base->\_fName);**

**initDbWithFile(base, base->\_fName, "w");**

**dbconnect(base);**

**}**

**void closeConnect(struct DataBase \* b)**

**{**

**closedb(b);**

**}**

**void pushRecord(struct DataBase \* b)**

**{**

**char line [MAX\_LINE\_SIZE] = {0};**

**char strNumber [30]; // макс. число разрядов**

**snprintf(strNumber, 30, "%d", recordToFile.number);**

**strcat(line, strNumber);**

**strcat(line, FILE\_LINE\_SEPORATOR\_S);**

**strcat(line, recordToFile.gmst.name);**

**strcat(line, FILE\_LINE\_SEPORATOR\_S);**

**snprintf(strNumber, 30, "%d", recordToFile.gmst.memoryD);**

**strcat(line, strNumber);**

**strcat(line, FILE\_LINE\_SEPORATOR\_S);**

**snprintf(strNumber, 30, "%d", recordToFile.gmst.ageR);**

**strcat(line, strNumber);**

**strcat(line, FILE\_LINE\_SEPORATOR\_S);**

**snprintf(strNumber, 30, "%d", recordToFile.gmst.graphic);**

**strcat(line, strNumber);**

**strcat(line, FILE\_LINE\_SEPORATOR\_S);**

**snprintf(strNumber, 30, "%d", recordToFile.gmst.plrL);**

**strcat(line, strNumber);**

**strcat(line, FILE\_LINE\_SEPORATOR\_S);**

**snprintf(strNumber, 30, "%d", recordToFile.gmst.arate);**

**strcat(line, strNumber);**

**strcat(line, FILE\_LINE\_SEPORATOR\_S);**

**writeLine(b, line);**

**}**

**int nextRecord(struct DataBase \* b)**

**{**

**char \* line = getLine(b);**

**if (line == NULL) return NULL;**

**recordFromFile.number = atoi(strtok(line, FILE\_LINE\_SEPORATOR\_S));**

**strcpy(recordFromFile.gmst.name, strtok(NULL, FILE\_LINE\_SEPORATOR\_S));**

**recordFromFile.gmst.memoryD = atoi(strtok(NULL, FILE\_LINE\_SEPORATOR\_S));**

**recordFromFile.gmst.ageR = atoi(strtok(NULL, FILE\_LINE\_SEPORATOR\_S));**

**recordFromFile.gmst.graphic = atoi(strtok(NULL, FILE\_LINE\_SEPORATOR\_S));**

**recordFromFile.gmst.plrL = atoi(strtok(NULL, FILE\_LINE\_SEPORATOR\_S));**

**recordFromFile.gmst.arate = atoi(strtok(NULL, FILE\_LINE\_SEPORATOR\_S));**

**return 1;**

**}**

**void outRecord(const struct Record \*rec)**

**{**

**printf(" %10d ", rec->number);**

**printf(" \t%20.10s ", rec->gmst.name);**

**printf(" %10d ", rec->gmst.memoryD);**

**printf(" %10d+ ", rec->gmst.ageR);**

**printf(" \t%10d ", rec->gmst.graphic);**

**printf(" \t%10d ", rec->gmst.plrL);**

**printf(" \t%10d ", rec->gmst.arate);**

**}**

**void openBufferDb(char \* name)**

**{**

**strcpy(temp\_base.\_fName, name);**

**openOutputAndReadConnect(&temp\_base);**

**}**

**void openBufferDbW(char \* name)**

**{**

**strcpy(temp\_base.\_fName, name);**

**openWriteConnect(&temp\_base);**

**}**

**void closeBufferDb()**

**{**

**closeConnect(&temp\_base);**

**}**

**void deleteRecordByNumber(int nn)**

**{**

**static char nameFile[MAX\_FILENAME];**

**sprintf(nameFile, "after\_remove\_%d", nn);**

**if (isFileExistInCDir(nameFile)) strcat(nameFile, "\_");**

**openBufferDb(nameFile);**

**while (nextRecord(&curr\_base)) {**

**if (recordFromFile.number != nn) {**

**recordToFile = recordFromFile;**

**pushRecord(&temp\_base);**

**}**

**}**

**closeConnect(&curr\_base);**

**dbcpy(&curr\_base, &temp\_base);**

**closeBufferDb();**

**resetPtr(&curr\_base);**

**}**

**struct Record \* findRecordByNumber (int nn)**

**{**

**static struct Record foundRecord;**

**while (nextRecord(&curr\_base))**

**{**

**if (recordFromFile.number == nn) {**

**foundRecord = recordFromFile;**

**resetPtr(&curr\_base);**

**return &foundRecord;**

**}**

**}**

**resetPtr(&curr\_base);**

**return NULL;**

**}**

**/\***

**Реализация работы с базой данных**

**\*/**

**#include <stdio.h>**

**#include <string.h>**

**#include <stdlib.h>**

**#define MAX\_FILENAME 255**

**#define MAX\_FILEMODE 10**

**#define MAX\_LINE\_SIZE 500**

**#define CORRECT\_RETURN 0**

**#define FAILURE\_RETURN 1**

**#define UNDEFINED\_STRING "!UNDF\0"**

**struct DataBase**

**{**

**char \_fName [MAX\_FILENAME];**

**char \_fMode [MAX\_FILEMODE];**

**FILE \*\_ptrFile;**

**char \_isDef;**

**};**

**void closedb(struct DataBase \* db);**

**void initDefault(struct DataBase \* db)**

**{**

**db->\_ptrFile = NULL;**

**db->\_isDef = 0;**

**strcpy(db->\_fName, UNDEFINED\_STRING);**

**strcpy(db->\_fMode, UNDEFINED\_STRING);**

**}**

**void initDbWithFile(struct DataBase \* db, const char \* fileName,**

**char mode [])**

**{**

**if (db->\_isDef) fclose(db->\_ptrFile);**

**strcpy(db->\_fName, fileName);**

**strcpy(db->\_fMode, mode);**

**db->\_isDef = 1;**

**}**

**void changeFMode(struct DataBase \* db, char mode [])**

**{**

**if (db->\_isDef) fclose(db->\_ptrFile);**

**else return;**

**strcpy(db->\_fMode, mode);**

**}**

**void changeFName(struct DataBase \* db, char name [])**

**{**

**if (db->\_isDef) fclose(db->\_ptrFile);**

**else return;**

**strcpy(db->\_fName, name);**

**}**

**void dbcpy(struct DataBase \* dbt, struct DataBase \* dbf)**

**{**

**dbt->\_ptrFile = fopen(dbf->\_fName, dbf->\_fMode);**

**strcpy(dbt->\_fMode, dbf->\_fMode);**

**strcpy(dbt->\_fName, dbf->\_fName);**

**dbt->\_isDef = dbf->\_isDef;**

**}**

**void dbconnect(struct DataBase \* db)**

**{ // инвариант**

**db->\_ptrFile = fopen(db->\_fName, db->\_fMode);**

**}**

**void \_nextLine(struct DataBase \* db)**

**{**

**fputc('\n', db->\_ptrFile);**

**}**

**void writeLine(struct DataBase \* db, char \* data)**

**{**

**fputs(data, db->\_ptrFile);**

**fputc('\n', db->\_ptrFile);**

**}**

**char \* getLine(struct DataBase \* db)**

**{**

**static char line [MAX\_LINE\_SIZE];**

**if (fgets(line, MAX\_LINE\_SIZE, db->\_ptrFile)) return line;**

**return NULL;**

**}**

**void resetPtr(struct DataBase \* db)**

**{**

**fseek(db->\_ptrFile, 0, SEEK\_SET);**

**}**

**void closedb(struct DataBase \* db)**

**{**

**resetPtr(db);**

**fclose(db->\_ptrFile);**

**}**

**// DATABASE**

**/\***

**Модуль ввода inputFunc.c**

**\*/**

**#include <stdio.h>**

**#include <ctype.h>**

**#include <conio.h>**

**#define INVALID\_INPUT -1**

**#define MAX\_SIZE\_GNAME 255**

**typedef struct Record Record;**

**void waitButtom()**

**{**

**printf("\n\nWaiting for the button...");**

**\_getch();**

**}**

**void clearStdin()**

**{**

**fseek(stdin, 0, SEEK\_SET);**

**}**

**void waitInt(int \* ref)**

**{**

**while (scanf("%d", ref) != 1)**

**clearStdin();**

**clearStdin();**

**}**

**int inputGraphic(int \* g)**

**// Коды ошибки: -1**

**{**

**int temp = 0;**

**printf("\ngraphics > ");**

**waitInt(&temp);**

**if ( !(temp >= 0 && temp <= 2) ) return INVALID\_INPUT;**

**\*g = temp;**

**return 0;**

**}**

**int inputPlayerAm(int \* pa)**

**// Коды ошибки: -1**

**{**

**int temp = 0;**

**printf("\nplayers > ");**

**waitInt(&temp);**

**if (temp != 0 && temp != 1) return INVALID\_INPUT;**

**\*pa = temp;**

**return 0;**

**}**

**int inputArate(int \* r)**

**// Коды ошибки: -1**

**{**

**int temp = 0;**

**printf("\nRange of ratings [0 ; 100]\nrating of critics > ");**

**waitInt(&temp);**

**if (!(temp >= 0 && temp <= 100)) return INVALID\_INPUT;**

**\*r = temp;**

**return 0;**

**}**

**int countCharsstrstr(char \* countHere, char \* chFrom){**

**size\_t cfl = strlen(chFrom);**

**size\_t chl = strlen(countHere);**

**size\_t count = 0;**

**char ischfrom = 0;**

**for (int i = 0; i < chl; ++i)**

**{**

**ischfrom = 0;**

**for (int j = 0; j < cfl; ++j)**

**{**

**if ( countHere[i] == chFrom[j] && chFrom[j] != '\0')**

**ischfrom = 1;**

**}**

**if (ischfrom) count += 1;**

**}**

**return count;**

**}**

**int inputNameAndCheckOrThrowEx(char n [])**

**{**

**char sep[] = " <>-,/.!?():;\'\"\n\t";**

**char str [MAX\_SIZE\_GNAME];**

**const int MAX\_SEP\_AMOUNT = 7;**

**printf("\nThe name of the game > ");**

**gets\_s(str, MAX\_SIZE\_GNAME);**

**// Если знаков-разделитей неестественно много**

**if (countCharsstrstr(str, sep) > MAX\_SEP\_AMOUNT) {**

**printf( "Many seporators" );**

**return -1;**

**}**

**strcpy(n, str);**

**return 0;**

**}**

**int inputMemoryD(int \* m)**

**// Коды ошибки: -1**

**{**

**int temp = 0;**

**printf("\nmemory > ");**

**waitInt(&temp);**

**if (temp <= 0) return INVALID\_INPUT;**

**\*m = temp;**

**return 0;**

**}**

**int checkAgeROrThrowEx(const char s[])**

**{**

**char ds [] = "----";**

**int ps = 0;**

**char isCorrectInput = 1;**

**// Синтакическая правильность**

**if (s[ps] == '+' ) {**

**printf("\nExpected digit or \"+\"\n");**

**isCorrectInput = 0;**

**}**

**while (s[ps] != '+' && isCorrectInput)**

**{**

**if ( !isdigit( s[ps++] ) ){**

**printf("\n:nExpected \"+\" or digit.");**

**printf("\n:%.\*s", ps - 1, s);**

**printf("\n:%.\*s^\n", ps - 1, ds);**

**isCorrectInput = 0;**

**}**

**}**

**return isCorrectInput;**

**}**

**char \* numInStr(char\* s)**

**// Находит число от начала строки до символа,**

**// который не является цифрой. Пример. 16+**

**// Вывод. 16**

**{**

**char \* num = (char\*)calloc(strlen(s) + 1, sizeof(char));**

**size\_t i = 0;**

**for (; isdigit(s[i]); ++i){**

**num[i] = s[i];**

**}**

**num[i + 1] = '\0';**

**return num;**

**}**

**int inputAgeR(int \* ag)**

**// Коды ошибки: -1**

**{**

**char str[20];**

**char \* n;**

**printf("\n(format: 6+,7+,..,18+,....)\n age rate > ");**

**gets\_s(str, 20);**

**if (checkAgeROrThrowEx(str) == 0)**

**return -1;**

**n = numInStr(str);**

**\*ag = atoi( n );**

**free(n);**

**return 0;**

**}**

**enum field\_codes {NAME, MEM, AGER, GRAPHIC, PLAYERS, AVERATE};**

**void fillField(struct Record \*rec, int numberf)**

**{**

**switch (numberf)**

**{**

**case NAME:**

**while( inputNameAndCheckOrThrowEx(rec->gmst.name) == INVALID\_INPUT);**

**break;**

**case MEM:**

**while( inputMemoryD(&rec->gmst.memoryD) == INVALID\_INPUT);**

**break;**

**case AGER:**

**while ( inputAgeR(&rec->gmst.ageR) == INVALID\_INPUT);**

**break;**

**case GRAPHIC:**

**while (inputGraphic(&rec->gmst.graphic) == INVALID\_INPUT);**

**break;**

**case PLAYERS:**

**while ( inputPlayerAm(&rec->gmst.plrL) == INVALID\_INPUT);**

**break;**

**case AVERATE:**

**while ( inputArate(&rec->gmst.arate) == INVALID\_INPUT);**

**break;**

**}**

**}**

**void fillRecord(struct Record \*rec)**

**{**

**fillField(rec, NAME);**

**fillField(rec, MEM);**

**fillField(rec, AGER);**

**fillField(rec, GRAPHIC);**

**fillField(rec, PLAYERS);**

**fillField(rec, AVERATE);**

**}**

**void inputNameForCall(struct Call \*cl)**

**{**

**do {**

**printf("\nAbsolute coincidence = %d, Part coincidence = %d\n: ", cl->ABSOLUTE, cl->INTER);**

**waitInt(&cl->nc);**

**} while (cl->nc != cl->ABSOLUTE && cl->nc != cl->INTER);**

**while ( inputNameAndCheckOrThrowEx(cl->name) == INVALID\_INPUT );**

**}**

**void inputMemForCall(struct Call \*cl)**

**{**

**do {**

**printf("\nMore = %d, less = %d\n: ", cl->MORE, cl->LESS);**

**waitInt(&cl->bmem);**

**} while (cl->bmem != cl->MORE && cl->bmem != cl->LESS);**

**while ( inputMemoryD(&cl->mem) == INVALID\_INPUT );**

**}**

**void inputAverForCall(struct Call \*cl)**

**{**

**do {**

**printf("\nMore = %d, less = %d\n: ", cl->MORE, cl->LESS);**

**waitInt(&cl->barate);**

**} while (cl->barate != cl->MORE && cl->barate != cl->LESS);**

**while ( inputArate(&cl->arate) == INVALID\_INPUT );**

**}**

**void inputAgerateForCall(struct Call \*cl)**

**{**

**do {**

**printf("\nEnter your age > ");**

**waitInt(&cl->ageR);**

**} while(cl->ageR < 0 && cl->ageR > 100);**

**}**

**void enterFileName(char \* fn) // ?? inputFuncs**

**{**

**printf("\n- \* - Enter name of file with database - \* - \n");**

**printf("\n - If the program finds the file, it will be able to open it. ");**

**printf("\n - Else, the program can create it.\n");**

**char loop;**

**do {**

**loop = 1;**

**printf("\nEnter \"-1\" for exit OR name of file");**

**printf("\n > ");**

**gets\_s(fn, MAX\_FILENAME);**

**if ( strpbrk(fn, "\\/:\*?\"<>|") != NULL ) continue;**

**if ( (strcmp(fn, "con") == 0 ) || (strcmp(fn, "prn") == 0) )**

**continue;**

**if (strcmp(fn, "-1") == 0) exit(EXIT\_SUCCESS);**

**if (!isFileExistInCDir(fn))**

**{**

**printf("\nThere is no such file.\n");**

**printf("\nMaybe we'll create one?\n");**

**}**

**else**

**printf("\nThe file has been found. Open it?\n");**

**printf("\nEnter \"y\" fore positive answer > ");**

**if (getchar() == 'y') loop = 0;**

**clearStdin();**

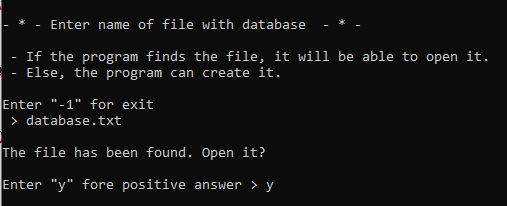
**} while(loop);**

**}**

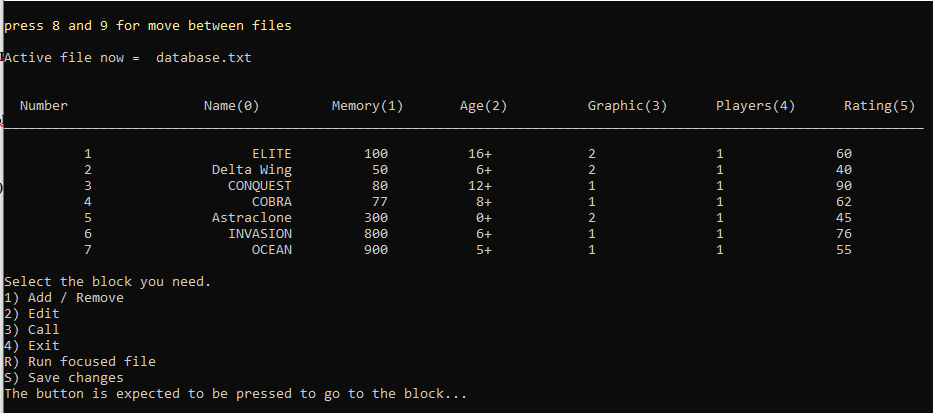
# Приложение Б

***Тесты программы***

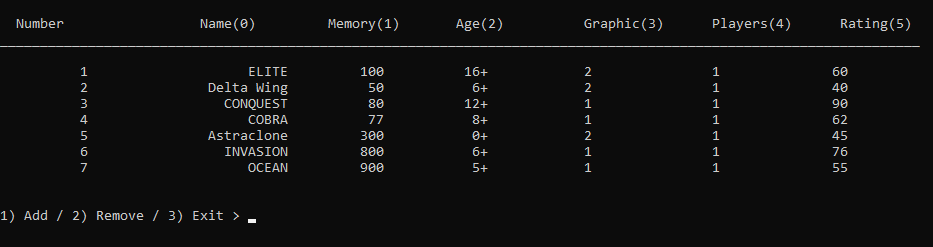
*Запрос файла*



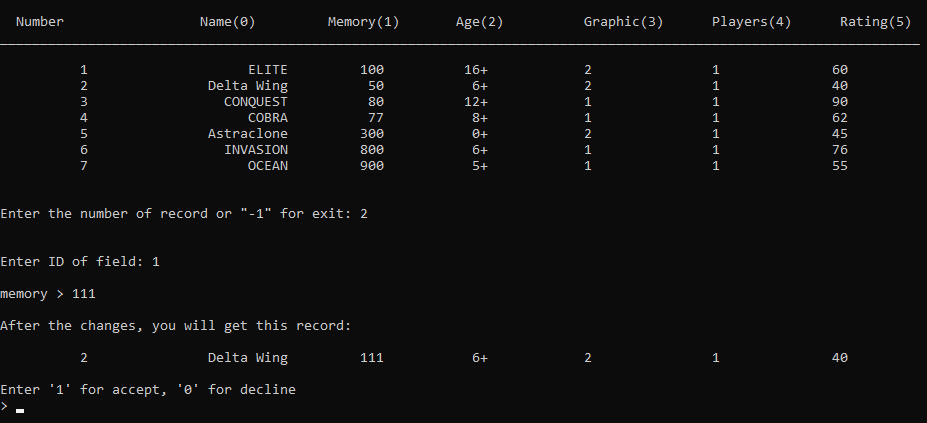
*Главное меню*



*Блок добавления, удаления*



*Редактирование*



*Запрос*

