Пензенский государственный университет
Кафедра «Вычислительная техника»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе
по курсу «Программирование»
на тему «Разработка программы сложной структуры методом нисходящего программирования»

Выполнила:
студентка группы 21ВВ3 Михальцова П.В.

Принял:
Патунин Д.В.

Пенза 2022

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Факультет Вычислительной техники

Кафедра "Вычислительная техника"

"УТВЕРЖДАЮ"

Зав. кафедрой ВТ

_____ М.А. Митрохин

«___»_____20__г

ЗАДАНИЕ

на курсовое проектирование по курсу

Программирование

Студенту Михальцовой Полине Валерьевне Группа 21ВВ3

Тема проекта: ***Разработка программы сложной структуры методом нисходящего программирования.***

Исходные данные (технические требования) на проектирование

Наименование приложения: База данных по студентам

Задание:
- Добавление записей
- Удаление записей
- Редактирование записей
- Поиск и сортировка записей
- Создание новой базы данных
- Загрузка сохраненных баз данных
- Сохранение текущей базы данных
- Фильтрация записей по определенному критерию

Обязательные требования к программе:

1. Многомодульность.
2. Использование сложных типов данных – структур, списков, файлов.
3. Режим работы видеосистемы – текстовый / графический.
4. Устройство ввода информации – клавиатура, мышь.
5. Пользовательский интерфейс должен быть построен на основе меню и панели инструментов.
6. Наличие заставки.
7. Операционная система MS Windows.
8. Язык программирования – Си и Ассемблер.
9. Среда разработки ПО – Microsoft Visual Studio.

Объем работы по курсу

1. **Расчетная часть**

Разработка программы.

2. **Графическая часть**

Схема данных, схема ресурсов системы, схема работы системы, иерархическая структура программы, схема взаимодействия программы.

3. **Экеперимантальная часть**

Отладка программы.

Срок выполнения проекта по разделам

В соответствии с графиком.

Дата выдачи задания "04" 03.2022

Дата защиты проекта "14" 06.2022

Руководитель Бакулин Д.В.

Задание получил "04" марта 2022г.

Студент Михальцова П.В.

Содержание

Введение 5
1 Постановка задачи 6
2 Выбор решения 7
2.1 Определение необходимых модулей программы 7
2.2 Определение структуры файла базы данных 8
3 Описание разработки программы 9
4 Отладка и тестирование 12
5 Описание программы 13
5.1 Разработка модуля main.c 13
5.2 Разработка меню 14
5.3 Разработка функции на языке Assembler 14
6 Руководство пользователя 15
Заключение 19
Список используемых источников 20
Приложение А Листинги программы 21
Приложение А.1 Файл «vvv.c» 21
Приложение А.2 Файл «main.c» 22
Приложение А.3 Файл «inkl.h» 30
Приложение В Снимки экрана 31

Введение

На сегодняшний день информация – ценный источник знаний, систематизации которой, позволяет оптимизировать работу как человека, так и общества в целом. Компьютерные технологии во многом упрощают нашу жизнь в самых разнообразных сферах нашей деятельности. И именно компьютерные технологии позволяют грамотно взаимодействовать с информацией, обрабатывать её и хранить на компактных носителях.

Базы данных — это совокупность структур, предназначенных для хранения больших объемов информации, и программных модулей, осуществляющих управление данными, выборку, сортировку и другие подобные действия над записями. Создавая базу данных, пользователь стремится упорядочить информацию по различным признакам и быстро извлекать данные, при необходимости делая выборку с произвольным сочетанием признаков.

Информация базы данных хранится в одной или нескольких таблицах. Любая таблица с данными состоит из набора однотипных записей, расположенных друг за другом, которые можно добавлять, изменять и удалять. Каждая запись является набором именованных полей или ячеек, которые могут хранить самую разнообразную информацию (фамилия, дата рождения и пр.).

С появлением и расширением компьютерных технологий появились противоречия между ограниченными возможностями человека по восприятию и переработке информации и существующими массивами хранящейся и передаваемой информации. Для решения подобных проблем применяются автоматизированные базы данных. Они стали неотъемлемой частью практически всех компьютерных систем – от глобальной отрасли до отдельного предприятия. Для персональных компьютеров было разработано множество различных программ, предназначенных для решения различных потребностей человека. Данная программа является информационно-поисковой системой, позволяющей любому пользователю обратиться к данным о студентах и произвести необходимые действия над записями – от сортировки по алфавиту до поиска по группе.

Данная программа является информационно-поисковой системой и позволяет работать с простейшей базой данных по студентам, позволяющей любому человеку просматривать и добавлять информацию.

1 Постановка задачи

Необходимо разработать приложение «База данных по студентам». Программа содержит следующие поля: № группы, Фамилия, Имя, Отчество, средний балл и дата рождения.

Программа должна включать в себя функции добавления записей в базу данных (далее – БД), удаления записей из БД, редактирования записей БД, поиска и сортировки записей БД, фильтрации записей БД по определенному критерию, создания новой БД, сохранения текущей БД, загрузки сохраненных БД. Для реализации данных требований необходимо разработать интуитивно понятный интерфейс пользователя и определить названия пунктов в меню. Нужно разработать алгоритм функционирования меню и обработки событий клавиатуры в консоли, продумать структуру работы программы и механизм взаимодействия входящих в нее функций.

Многомодульность программы. Требуется поделить программу на логические модули, что повысит расширяемость, упростит редактирование и исправление возникающих ошибок при отладке или тестировании программы.

Использование сложных типов данных – структур, списков, файлов. Требуется изучить базовые и сложные типы данных, что позволит организовать структурирование БД.

Режим работы системы – текстовый / графический. Требуется определить тип интерфейса, изучить способы его реализации и механизмы работы.

Устройство ввода информации – клавиатура, мышь. Требуется однозначное распознавание действия пользователя, реализованного через клавиатуру или мышь, и выполнение соответствующего действия программы.

Пользовательский интерфейс должен быть построен на основе меню и панели инструментов. Требуется изучить способы разработки меню, выбрать наиболее подходящий вариант для данного задания и условий. Меню и панель инструментов являются основной задачей разработки интуитивно понятного интерфейса.

Наличие заставки. Заставка необходима для сообщения пользователю достаточной визуальной информации о программе и возможности перехода к работе с программой.

Операционная система MS Windows. Язык программирования – C, Assembler. Среда разработки – MS Visual Studio 2019.

2 Выбор решения

2.1 Определение необходимых модулей программы

Разработанная программа состоит из трех модулей:

1. inkl.h
2. vvv.c
3. main.c

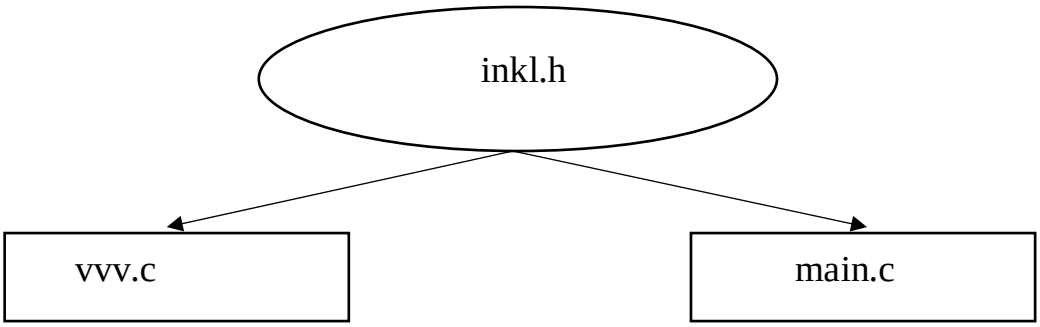


Рисунок 1 – структура модулей программы

Заголовочный файл inkl.h необходим для объявления глобальных переменных и подключения стандартных библиотек языка С. Данная библиотека была создана специально для данной курсовой работы.

Модуль vvv.c содержит функцию сортировки записей.

Модуль main.c – основной модуль данной курсовой работы. Модуль содержит все требуемые функции программы: добавление записей в БД, удаление записей из БД, редактирование записей БД, поиск записей БД, фильтрация записей БД по определенному критерию, создание новой БД, сохранение текущей БД, загрузка сохраненных БД.

Все модули с расширением .c используют заголовочные файлы (main.c еще глобальную переменную) из библиотеки inkl.h. Тем самым выполняется требование модульности данной курсовой работы. Управление приложением осуществляется с помощью клавиатуры. Одновременная работа двух программ с одной базой данных не допускается. Данные из БД отображаются в программе в виде таблицы из 6 столбцов: № группы, Фамилия, Имя, Отчество, средний балл, дата рождения. Навигация по таблице организована через события клавиатуры, а именно через клавиши – цифры от 0 до 8.

2.2 Определение структуры файла базы данных

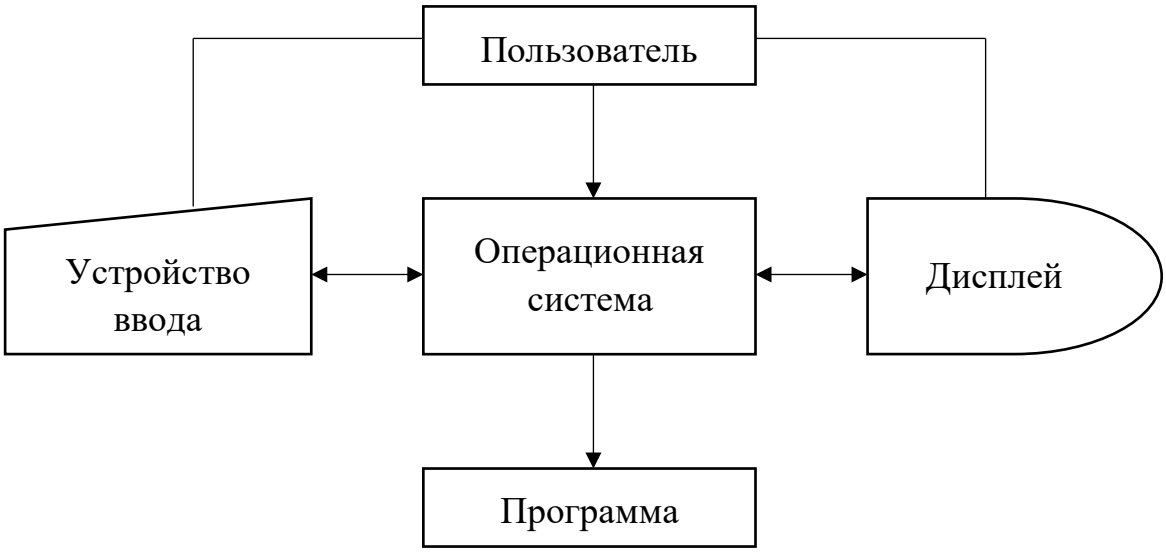
Для структурирования данных внутри файла была выбрана следующая структура данных:

```
//Структура данных
struct stundent {
    char name[20]; // имя
    char surname[20]; // фамилия
    char patr[20];  // отчество
    char group[8];  // № группы
    double averageScore;  //средний балл
    double dateOfBirn[];  // дата рождения
};
```

Поля структуры «Номер группы», «Фамилия», «Имя», «Отчество» имеют текстовый формат. Под Фамилию, Имя, Отчество выделено 20 символов, под группу – 8 символа. Поле «Средний бал» и «Дата рождения» определена переменной типа double.

База данных реализована в виде массива структур. Количество записей в файле неограниченно.

Ниже представлена схема данных:



3 Описание разработки программы

В написании данной программы использован достаточно распространенный язык программирования С. При разработке языка был принят компромисс между низким уровнем языка ассемблера и высоким уровнем других языков. Язык С даёт программисту возможность выражать свои алгоритмы и данные наиболее подходящими для этого средствами.

В качестве среды программирования был выбран программный продукт Visual Studio 2019.

Интерфейс информационной системы в силу своего удобства и наименьшего времени доступа к данным представлен в виде меню, перемещение по пунктам которого осуществлено с помощью клавиш управления.

Исходя из того, что данные о студентах (например, средний балл) со временем могут изменяться, в БД предусмотрена функция редактирования записей и соответствующий пункт меню «Редактировать». Для пополнения БД предусмотрен пункт меню «Добавить».

Для быстрого поиска полной информации об имеющихся записях (студентах) меню содержит пункт «Поиск». Причем поиск осуществляется по всем категориям данных, что позволяет пользователю без труда и потери времени получить доступ к нужной записи.

Для упорядочения БД предусмотрен пункт меню «Сортировка», в котором можно выстроить записи БД по среднему баллу (по возрастанию) и по группе.

Перечисленные выше операции можно выполнять как в уже созданной БД, так и в создаваемой в настоящем времени БД. Для открытия существующей базы предусмотрен пункт меню «Загрузить», для создания новой БД – «Добавить». Для сохранения созданных записей предусмотрен одноименный пункт меню.

Основная программа выполняется функцией main(). Функция отвечает за инициализацию консоли, вывод меню на экран, передачу управления в функции обработки событий меню. После ввода некоторого значения с клавиатуры функция обработки событий меню получает идентификатор выбранного пункта меню, в соответствии с которым организуется передача управления в нужную функцию. Блок-схема функции main() и функции обработки событий меню представлена на рисунке 3.

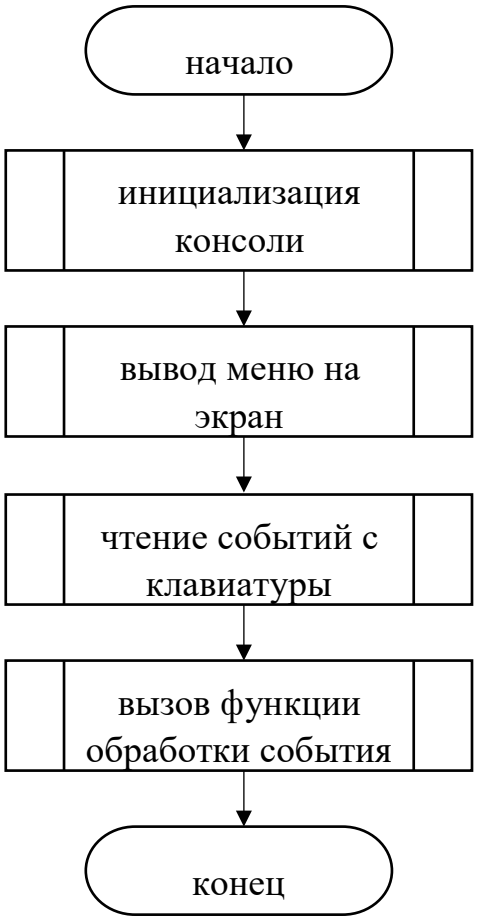


Рисунок 3 – схема программы, функция обработки событий меню

Когда главная функция обнаруживает событие от клавиатуры, вызывается функция обработки события. В зависимости от активного элемента выполняются те или иные действия.

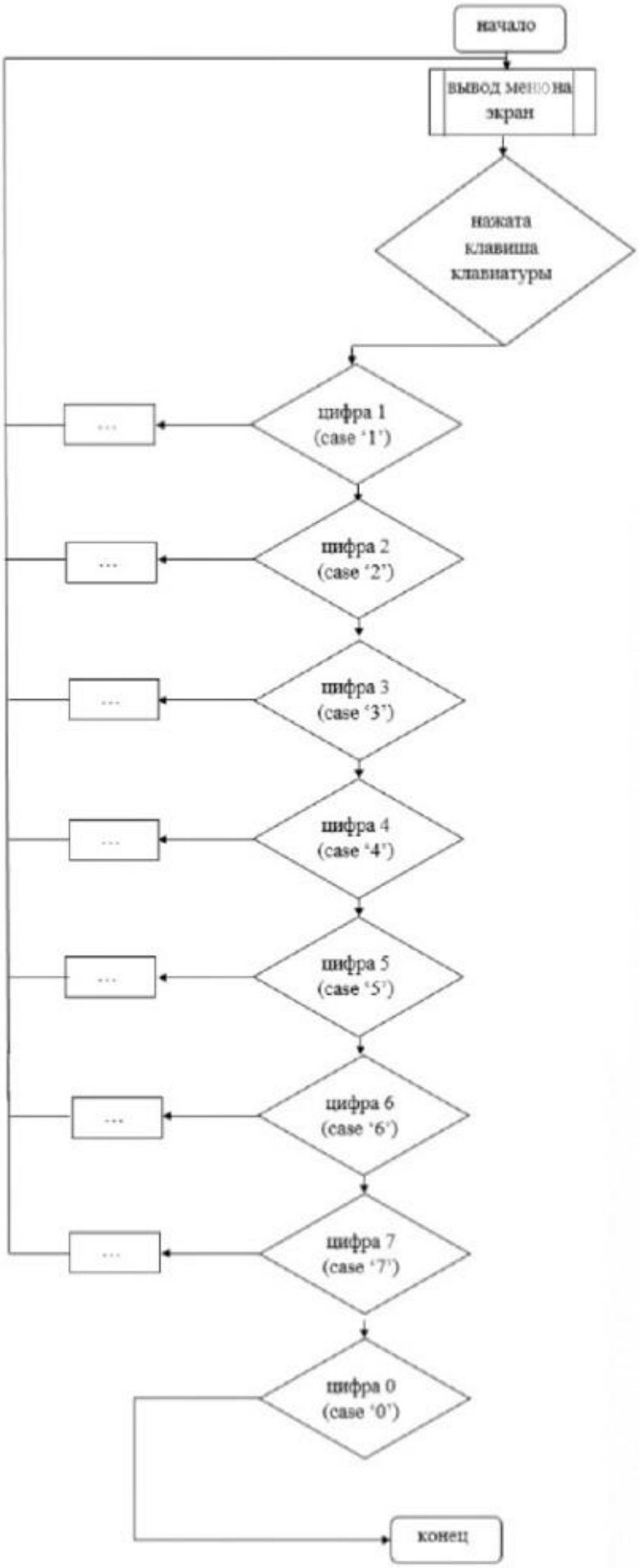


Рисунок 4 – алгоритм работы меню

4 Отладка и тестирование

Средой разработки была выбрана программа Visual Studio 2019. Программа содержит в себе все необходимые средства для успешной отладки и тестирования модулей программы, а также всей программы в целом. Данная среда разработки выпущена относительно недавно, что позволяет проводить необходимые операции над разработанной программой наиболее производительно и эффективно. Для выполнения отладки применялись средства Visual Studio, такие как: точки останова, анализ содержимого переменных, выполнение кода по шагам (по строкам), анализ памяти.

Тестирование проводилось в рабочем порядке, на всех этапах разработки программы, в т.ч. после завершения написания кода. Тестирование позволило выявить множество проблем и недочетов по части внешнего вида. Для эффективной работы программы были созданы проверки, не позволяющие проводить обработку некоторых частей программы в определенных условиях.

5 Описание программы

5.1 Разработка модуля main.c

Модуль main.c является основным модулем программы. Вспомогательный модуль vvv.cpp содержит в себе одну функцию, используемую в main.c. В модуле происходит полный процесс работы с БД, начиная от ее создания и вывода на экран и заканчивая сохранением готовой БД. При запуске программы на экран выводится меню, содержащее восемь функций, готовых выполниться по нажатию соответствующей клавиши. Присутствует клавиша 0, которая служит в программе возвратом назад («шаг назад») при нахождении во вторичном меню, выходом из программы.

Описание всех состояний программы приведено в таблице 1:

Таблица 1 – Описание состояний программы

Клавиша, вызывающая событие	Действие пользователя	Действие программы
0	Выбран пункт «Выход»	При нахождении во вторичном меню – возвращает в основное меню; выход из программы
1	Выбран пункт «Вывод»	Вывод шаблона таблицы(если уже загружена)
2	Выбран пункт «Добавить»	Выводится форма для добавления записи о новом студенте
3	Выбран пункт «Редактировать»	Выводится вторичное меню, где представлены действия редактирования, вывод БД
4	Выбран пункт «Удалить»	Выпадает запрос на подтверждение действия, при подтверждении – удаление выбранного студента
5	Выбран пункт «Сортировка»	Выводится вторичное меню, где представлены действия (параметры) сортировки
0 (5)	Выбран пункт «Назад»	Выход в основное меню
1 (5)	Выбран пункт «Группа»	Выводится диалоговая строка «Введите значение поиска», осуществляется соответствующий поиск (фильтрация)
2 (5)	Выбран пункт «Средний балл»	

6	Выбран пункт «Поиск»	Выводится вторичное меню, где представлены действия поиска
0 (6)	Выбран пункт «Назад»	Выход в основное меню
1 (6)	Выбран пункт «Группа»	Выводится диалоговая строка «Введите значение поиска», Осуществляется соответствующий поиск
2 (6)	Выбран пункт «Средний балл»	
7	Выбран пункт «Загрузить»	Чтение БД из файла
8	Выбран пункт «Сохранить»	Запись БД в файл

5.2 Разработка меню

Меню реализовано в файле main.c. Состоит из следующих пунктов: «Выход», «Вывод», «Добавить», «Редактировать», «Удалить», «Сортировка», «Поиск», «Загрузить», «Сохранить». Главной задачей функций файла main.c является обеспечение возможности выбора пользователю одного из пунктов меню и возврата программе идентификатора выбранного пункта, в соответствии с которым происходит передача управления в нужную часть программы.

Основное меню реагирует на нажатие клавиш: цифры от 0 до 8.

5.3 Разработка функции на языке Assembler

Функции на языке ассемблера выполняются процессором гораздо быстрее, что в данной программе имеет прикладное значение в выводе меню и считывании символа, введенного пользователем в меню (выбор пункта меню). Visual Studio 2019 позволяет включать в программу ассемблерную вставку (имеет для этого компилятор) следующим образом: в теле программы пишется ключевое слово «__asm» (двойное нижнее подчеркивание), открывается фигурная скобка, в которой реализуется ассемблерный код, после завершения написания – закрывающая фигурная скобка.

Для реализации функции вывода меню и считывания символа были использованы ассемблерные инструкции:

– *Push* вталкивает используемый регистр в стек;

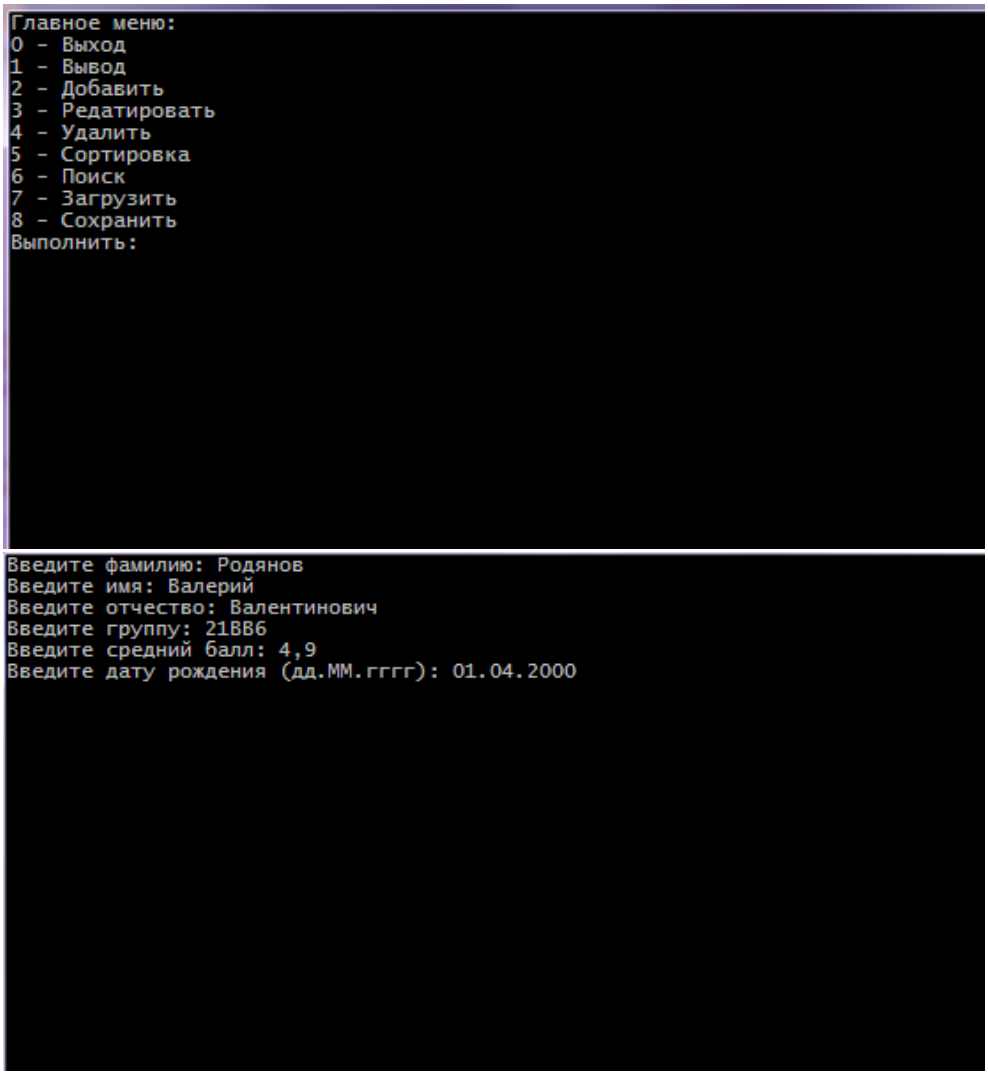
– *Call* вызывает функции языка С;

– *Pop* удаляет верхний элемент стека.

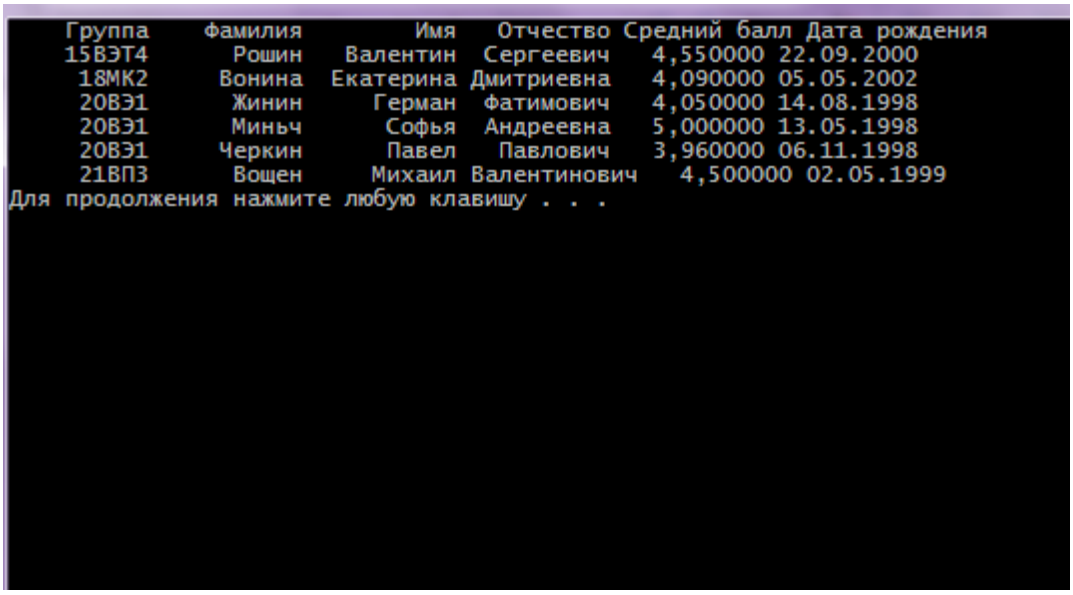
Код функции представлен в приложении.

6 Руководство пользователя

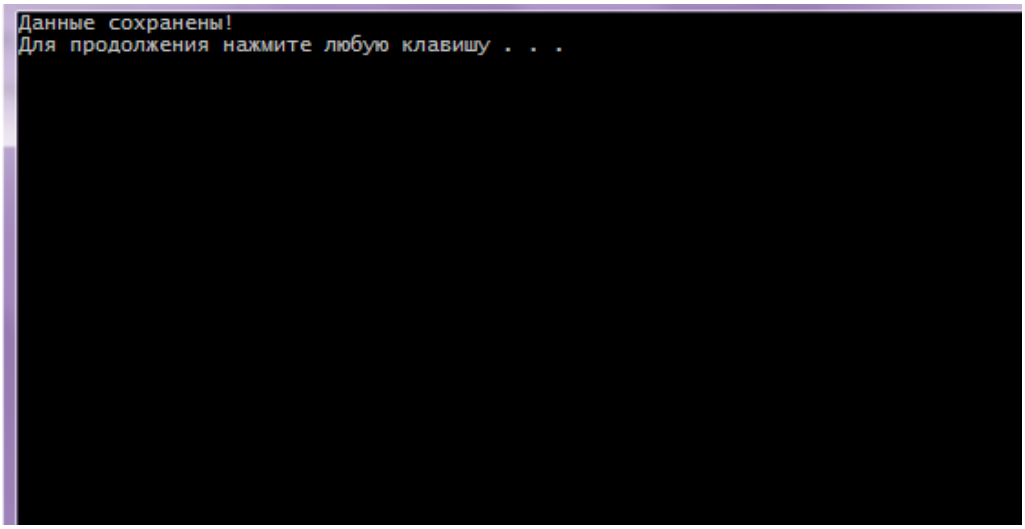
Для создания новой БД необходимо выбрать пункт меню «Добавить»:



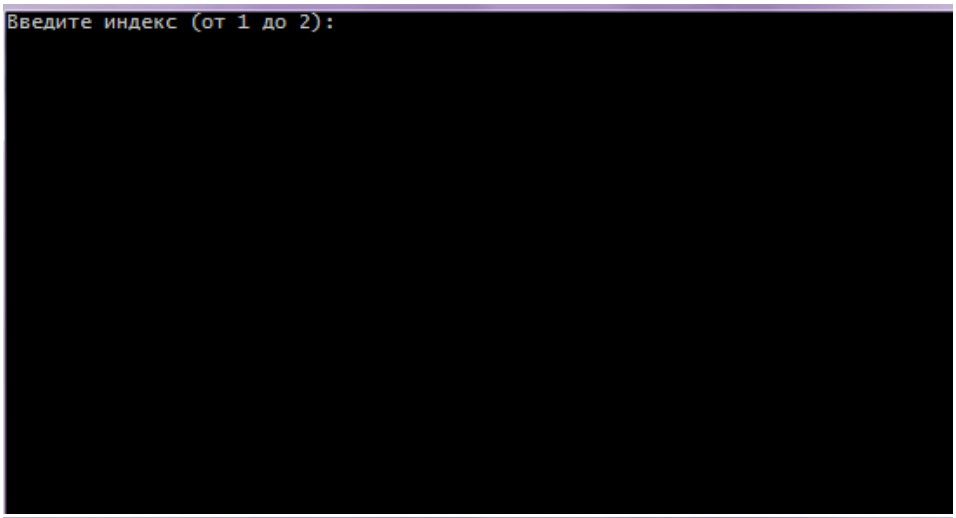
Для открытия и вывода на экран существующей БД необходимо выбрать пункт меню «Вывод»:

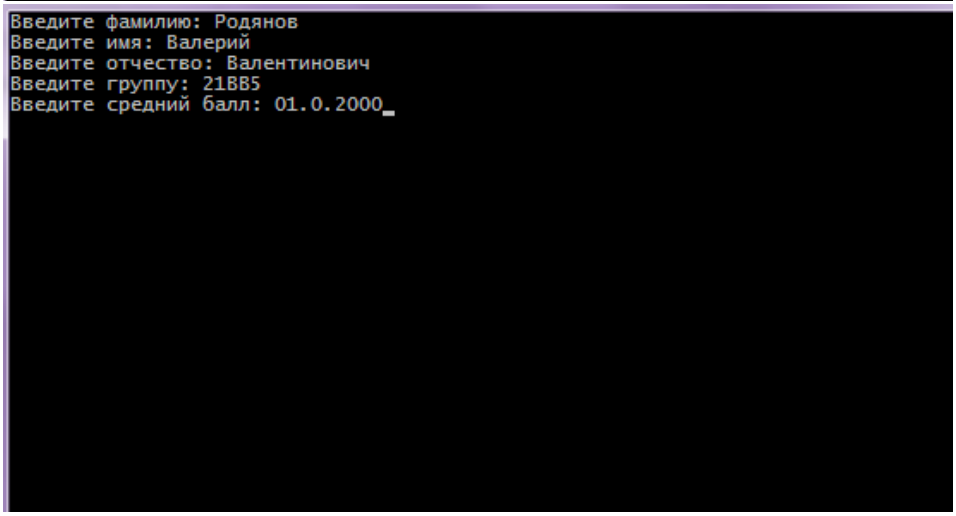


Для сохранения существующей (созданной) БД необходимо выбрать пункт «Сохранить»:

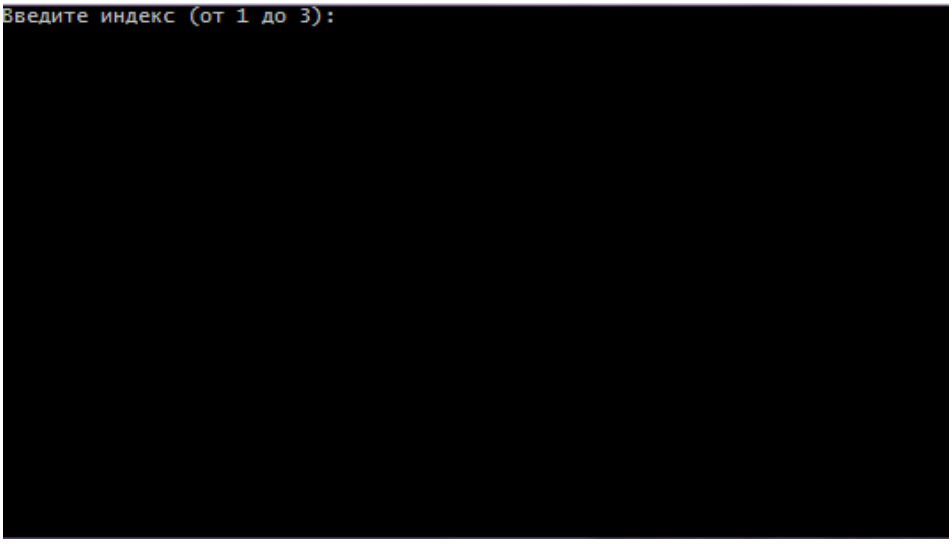


Для редактирования БД необходимо выбрать пункт «Редактировать»:

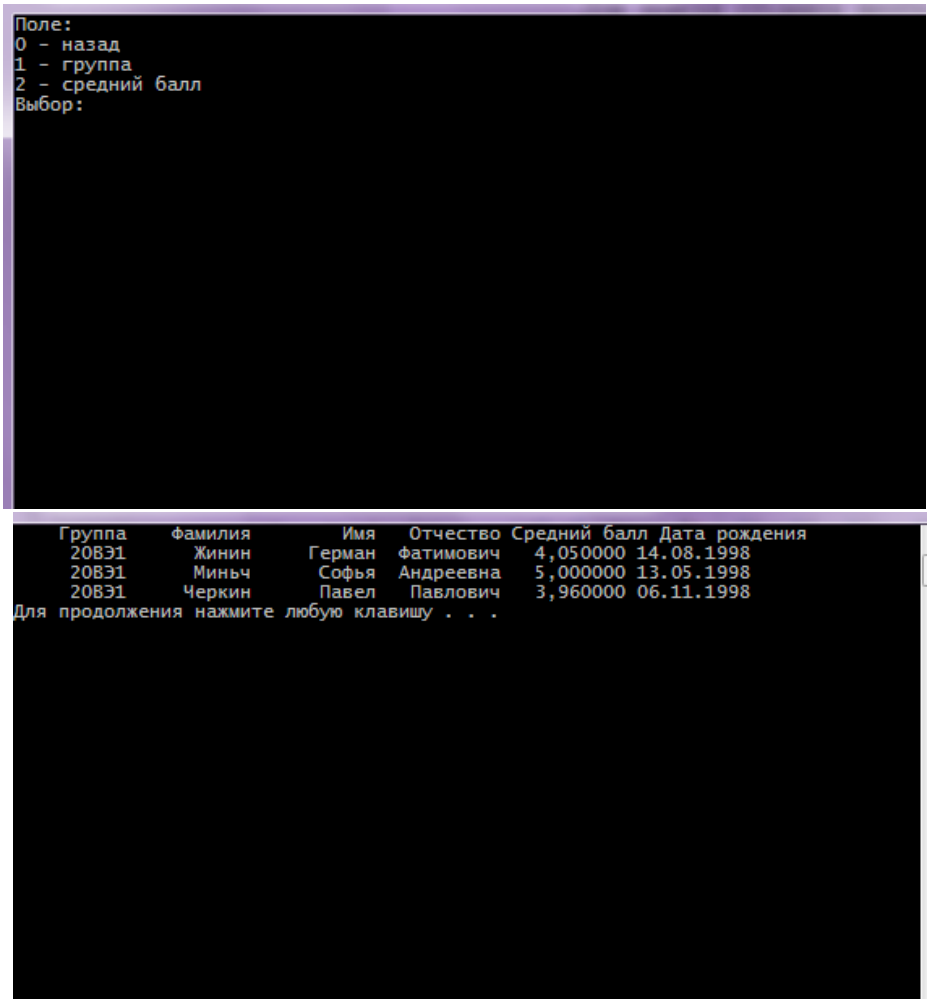




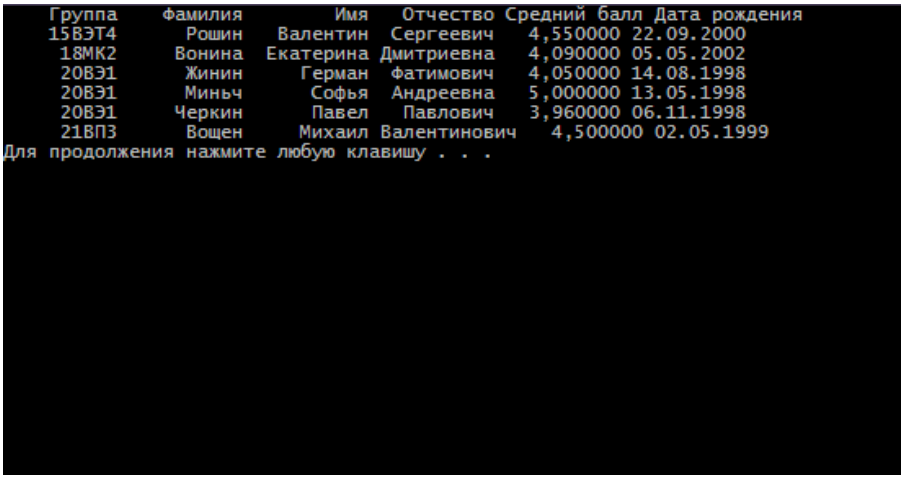
Для удаления выбранного студента необходимо во вторичном меню выбрать пункт «Удалить текущего»:



Для поиска студента необходимо выбрать пункт «Поиск».



Для сортировки студентов по определенному критерию необходимо выбрать пункт меню «Сортировка». Выбор критерия представлен во вторичном меню пункта «Сортировка»:



Заключение

В ходе выполнения курсовой работы были получены и расширены навыки разработки многомодульных программ на языке С, освоены функции работы с консолью, способы интерпретации событий от клавиатуры; изучены различные принципы реализации меню. Освоены на практике в большом объеме функции для работы с файлами. Закреплены приемы работы в компиляторе Visual Studio 2019, отточены навыки программирования на языках С, Assembler, изучены возможности взаимодействия между языками.

В данной курсовой работе была написана программа, используемая для хранения информации о студентах, «база данных по студентам». Программа представляет ограниченный функционал, содержит базовые средства для выполнения операций в базе данных, которые в будущем можно усовершенствовать. Соответственно, программа не требует большого количества системных ресурсов, благодаря, в том числе использованию языка ассемблера, обеспечивает быструю обработку информации.

Список используемых источников

1. Земсков Ю.В. Программирование на языке C/C++. 2012 г.
2. http://programmersforum.ru
3. https://cyberforum.ru
4. Давыдов В.Г. Программирование и основы алгоритмизации. 2003 г.

Приложение А Листинги программы

Приложение А.1 Файл «vvv.c»

```
#include "inkl.h"

void Sort(struct list* lst) {
    if (lst == NULL) {
        printf("Данные пусты!\n");
        return;
    }

    int mode;
    printf("Поле:\n0 - назад\n1 - группа\n2 - средний балл\nВыбор: ");
    scanf("%d", &mode);
    system("cls");

    if (mode == 0)
        return;

    int sorted = 0;
    struct list* p = lst;
    struct list* p1 = lst;
    struct list* p2 = lst->ptr;

    switch (mode)
    {
    case 1:
        // Сортировка пузырьком
        do {
            while (p2 != NULL) {
                if (strcmp(p1->field.group, p2->field.group) > 0) {
                    struct stundent buf = p1->field;
                    p1->field = p2->field;
                    p2->field = buf;
                }

                p2 = p2->ptr;
            }

            p1 = p1->ptr;

            if (p1 != NULL)
                p2 = p1->ptr;
        } while (p1 != NULL);

        sorted = 1;
        break;

    case 2:
        // Сортировка пузырьком
        do {
            while (p2 != NULL) {
```

```
                if (p1->field.averageScore > p2->field.averageScore) {
                    struct stundent buf = p1->field;
                    p1->field = p2->field;
                    p2->field = buf;
                }

                p2 = p2->ptr;
            }

            p1 = p1->ptr;

            if (p1 != NULL)
                p2 = p1->ptr;
        } while (p1 != NULL);

        sorted = 1;
        break;

    default:
        printf("Неверный выбор!");
        break;
    }

    if (sorted == 1)
        printf("Данные отсортированы!\n");
}
```

Приложение А.2 Файл «main.c»

```
#define _CRT_SECURE_NO_DEPRECATE
#include "inkl.h"

extern void Sort(struct list* lst);

int size;

// Структура данных

struct list
{
    struct stundent field;  // поле данных
    struct list* ptr;       // указатель на следующий элемент
};

struct list* Init(struct stundent a) // a- значение первого узла
{
    struct list* lst;
    // выделение памяти под корень списка
    lst = (struct list*)malloc(sizeof(struct list));
    lst->field = a;
    lst->ptr = NULL; // это последний узел списка
    size = 1;
    return(lst);
}

struct list* Add(struct list* lst, struct stundent a)
```

```
{
    struct list* temp, * p = lst;
    temp = (struct list*)malloc(sizeof(struct list));

    while (p->ptr != NULL)
        p = p->ptr;

    temp->ptr = NULL;
    temp->field = a;    // сохранение поля данных добавляемого узла
    p->ptr = temp;
    size++;
    return(temp);
}

struct list* DeletElem(struct list* lst, struct list* root)
{
    if (root == NULL) {
        printf("Данные пусты!\n");
        return;
    }

    struct list* temp;
    temp = root;
    while (temp->ptr != lst) // просматриваем список начиная с корня
    { // пока не найдем узел, предшествующий lst
        temp = temp->ptr;
    }
    temp->ptr = lst->ptr; // переставляем указатель
    free(lst); // освобождаем память удаляемого узла
    size--;

    return(temp);
}

struct list* DeleteHead(struct list* root)
{
    if (root == NULL) {
        printf("Данные пусты!\n");
        return;
    }

    struct list* temp;
    temp = root->ptr;
    free(root); // освобождение памяти текущего корня
    size--;

    return(temp); // новый корень списка
}

struct list* GetNode(struct list* root, int index)
{
    if (root == NULL) {
        printf("Данные пусты!\n");
        return;
    }
```

```
    struct list* p;
    p = root;

    while (p != NULL) {
        if (index == 0)
            break;

        index--;
        p = p->ptr;
    }

    return p;
}

void Clear(struct list* root) {
    while (root != NULL) {
        DeleteHead(root);
    }
}

void PrintHeader() {
    printf("%10s %10s %10s %10s %10s %10s\n", "Группа", "Фамилия", "Имя",
"Отчество", "Средний балл", "Дата рождения");
}

void Print(struct stundent d) {
    printf("%10s %10s %10s %10s %10f %10s\n", d.group, d.surname, d.name,
d.patr, d.averageScore, d.dateOfBirn);
}

void PrintAll(struct list* lst)
{
    if (lst == NULL) {
        printf("Данные пусты!\n");
        return;
    }

    struct list* p;
    p = lst;

    PrintHeader();
    do {
        Print(p->field);// вывод значения элемента p
        p = p->ptr;     // переход к следующему узлу
    } while (p != NULL);
}


void Search(struct list* lst) {
    if (lst == NULL) {
        printf("Данные пусты!\n");
        return;
    }

    int mode;
    printf("Поле:\n0 - назад\n1 - группа\n2 - средний балл\nВыбор: ");
```

```
    scanf("%d", &mode);
    system("cls");

    if (mode == 0)
        return;

    char group[100];
    float averageScore;
    int found = 0;

    struct list* p = lst;

    switch (mode)
    {
    case 1:
        printf("Введите группу: ");
        scanf("%s", &group);
        system("cls");

        PrintHeader();
        while (p != NULL) {
            if (strcmp(p->field.group, group) == 0) {
                found = 1;
                Print(p->field);
            }

            p = p->ptr;
        }
        break;

    case 2:
        printf("Введите средний балл: ");
        scanf("%f", &averageScore);
        system("cls");

        PrintHeader();
        while (p != NULL) {
            if (p->field.averageScore == averageScore) {
                found = 1;
                Print(p->field);
            }

            p = p->ptr;
        }
        break;

    default:
        printf("Неверный выбор!");
        break;
    }

    if (found == 0)
        printf("Совпадений не найжено!\n");
}

void Save(struct list* lst) {
```

```
    if (lst == NULL) {
        printf("Данные пусты!\n");
        return;
    }

    FILE* fp;

    if ((fp = fopen("data.txt", "w")) == NULL)
    {
        printf("Не удалось открыть файл!\n");
        return;
    }

    struct list* p = lst;
    while (p != NULL) {
        fprintf(fp, "%f\n", p->field.averageScore);
        fprintf(fp, "%s\n", p->field.dateOfBirn);
        fprintf(fp, "%s\n", p->field.group);
        fprintf(fp, "%s\n", p->field.name);
        fprintf(fp, "%s\n", p->field.patr);

        if(p->ptr != NULL)
            fprintf(fp, "%s\n", p->field.surname);
        else
            fprintf(fp, "%s", p->field.surname);

        p = p->ptr;
    }

    fclose(fp);
    printf("Данные сохранены!\n");
}

struct list* Load(struct list* lst) {
    FILE* fp;

    if ((fp = fopen("data.txt", "r")) == NULL)
    {
        printf("Не удалось открыть файл!\n");
        return;
    }

    if (lst != NULL)
        Clear(lst);

    struct list* p = lst;

    while (1) {
        struct stundent data;

        fscanf(fp, "%f", &data.averageScore);
        fscanf(fp, "%s", &data.dateOfBirn);
        fscanf(fp, "%s", &data.group);
        fscanf(fp, "%s", &data.name);
        fscanf(fp, "%s", &data.patr);
        fscanf(fp, "%s", &data.surname);
```

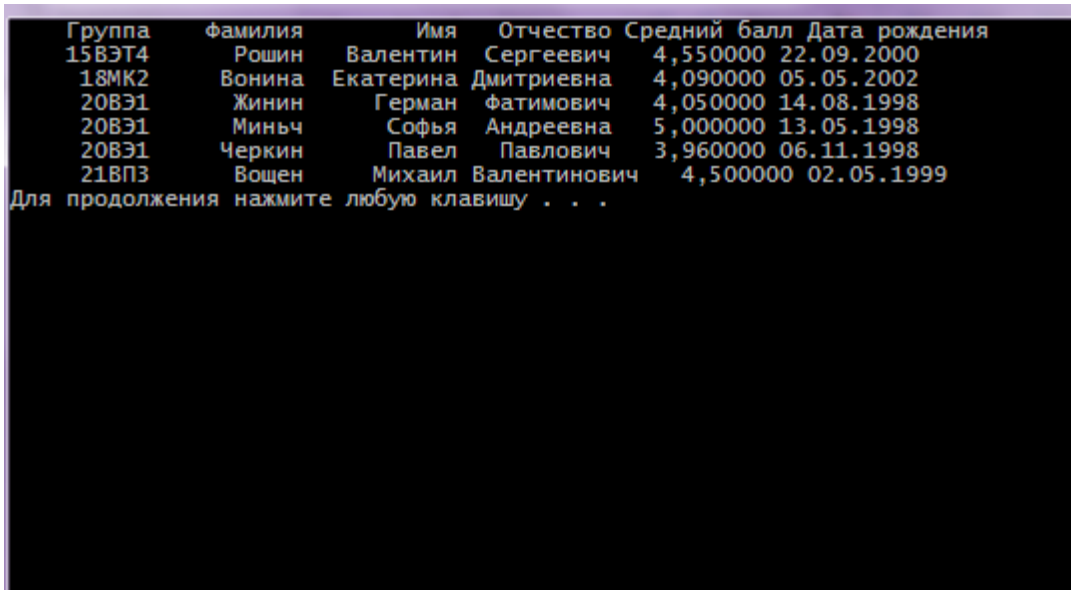
```
        if (p == NULL) {
            p = Init(data);
        }
        else {
            Add(p, data);
        }

        if (feof(fp))
            break;
    }
    /*
    for(int i = 0; i < size; i++){
        struct stundent data;

        fscanf(fp, "%f", &data.averageScore);
        fscanf(fp, "%s", &data.dateOfBirn);
        fscanf(fp, "%s", &data.group);
        fscanf(fp, "%s", &data.name);
        fscanf(fp, "%s", &data.patr);
        fscanf(fp, "%s", &data.surname);

        if (p == NULL) {
            p = Init(data);
        }
        else {
            Add(p, data);
        }
    }
    */
    fclose(fp);
    printf("Данные загружены!\n");
    return p;
}

char headerMenu[] = "Главное меню:";
int Menu() {
    __asm
    {
        mov  eax, offset headerMenu
        push eax
        call printf
        pop  ebx
    }

    int act;
    printf("\n0 - Выход\n1 - Вывод\n2 - Добавить\n3 - Редатировать\n4 -
Удалить\n5 - Сортировка\n6 - Поиск\n7 - Загрузить\n8 - Сохранить\nВыполнить: ");
    scanf("%d", &act);
    system("cls");

    return act;
}

int main()
{
```

```
char* locale = setlocale(LC_ALL, "");
SetConsoleCP(1251);
SetConsoleOutputCP(1251);

struct list* root = NULL;
struct list* p = NULL;

struct stundent data;
size = 0;
int index;
int act = 0;

do {
    act = Menu();

    switch (act)
    {
    case 0:
        break;

    case 1:
        PrintAll(root);
        break;

    case 2:
        printf("Введите фамилию: ");
        scanf("%s", &data.surname);
        printf("Введите имя: ");
        scanf("%s", &data.name);
        printf("Введите отчество: ");
        scanf("%s", &data.patr);
        printf("Введите группу: ");
        scanf("%s", &data.group);
        printf("Введите средний балл: ");
        scanf("%f", &data.averageScore);
        printf("Введите дату рождения (дд.ММ.гггг): ");
        scanf("%s", &data.dateOfBirn);

        system("cls");

        if (root == NULL) {
            root = Init(data);
        }
        else {
            Add(root, data);
        }

        printf("Данные добавлены!\n");
        break;

    case 3:
        printf("Введите индекс (от 1 до %d): ", size);
        scanf("%d", &index);
        index--;
        system("cls");
```

```
            p = GetNode(root, index);

            printf("Введите фамилию: ");
            scanf("%s", &p->field.surname);
            printf("Введите имя: ");
            scanf("%s", &p->field.name);
            printf("Введите отчество: ");
            scanf("%s", &p->field.patr);
            printf("Введите группу: ");
            scanf("%s", &p->field.group);
            printf("Введите средний балл: ");
            scanf("%f", &p->field.averageScore);
            printf("Введите дату рождения (дд.ММ.гггг): ");
            scanf("%s", &p->field.dateOfBirn);

            system("cls");

            printf("Данные изменены!\n");
            break;

        case 4:
            printf("Введите индекс (от 1 до %d): ", size);
            scanf("%d", &index);
            index--;
            system("cls");

            p = GetNode(root, index);
            DeletElem(p, root);

            printf("Данные удалены!\n");
            break;

        case 5:
            Sort(root);
            break;

        case 6:
            Search(root);
            break;

        case 7:
            root = Load(root);
            break;

        case 8:
            Save(root);
            break;

        default:
            printf("Неверный выбор!\n");
            break;
        }

        system("pause");
        system("cls");
    } while (act != 0);
```

```
    //Clear(root);
    return 0;
}
```

Приложение А.3 Файл «inkl.h»

```
#include <stdio.h>
#include <stdlib.h>
#include < string.h >
#include <conio.h>
#include <malloc.h>
#include <locale.h>
#include <Windows.h>

//Структура данных
struct stundent {
    char name[20];
    char surname[20];
    char patr[20];
    char group[8];
    double averageScore;
    double dateOfBirn[];
                                        };
```

Приложение В Снимки экрана