

**Содержание**

[Введение 4](#_Toc22695)

[1. Постановка задачи 5](#_Toc27148)

[2. Выбор решения 5](#_Toc23931)

[5.1. Модули программы 5](#_Toc5124)

[5.2. Интерфейс программы 6](#_Toc26329)

[3. Описание разработки программы 7](#_Toc20540)

[4. Отладка и тестирование 8](#_Toc24288)

[1. Установка точки останова и запуск отладчика: 9](#_Toc16703)

[Переход по коду в отладчике с помощью пошаговых команд 10](#_Toc25389)

[2. Переход по коду в отладчике с помощью пошаговых команд 10](#_Toc1651)

[3. Шаг с обходом по коду для пропуска функций 11](#_Toc5350)

[4. Быстрое выполнение до точки в коде с помощью мыши 11](#_Toc1643)

[5. Описание программы 12](#_Toc2607)

[6. Руководство пользователя 18](#_Toc10522)

[Заключение 20](#_Toc29733)

[Список используемых источников 21](#_Toc26705)

[Приложение А 21](#_Toc26067)

[Приложение B 54](#_Toc4457)

# Введение

База данных (БД) — это набор информации, которая хранится упорядоченно в электронном виде.

Базы данных позволяют обрабатывать, хранить и структурировать намного большие объёмы информации, чем таблицы.

Удалённый доступ и система запросов позволяет множеству людей одновременно использовать базы данных. С электронными таблицами тоже можно работать онлайн всей командой, но системы управления базами данных делают этот процесс организованнее, быстрее и безопаснее.

Объём информации в базах данных может быть огромным и не влиять на скорость работы. А в Google Таблицах уже после нескольких сотен строк или тысяч символов страница будет загружаться очень медленно.

Современные БД проектируются по принципу «получить данные прямо сейчас», чтобы пользователь не ждал отклик на запрос.

Какой бы высокой ни была скорость, это бессмысленно, если нужно сделать много сложных операций, чтобы получить, обновить или добавить данные в базу.

Изменения в любом количестве и качестве информации не должны влиять на структуру базы данных. Также изменения не должны касаться программного обеспечения и средств хранения, например жёсткого диска.

Аналогично свойству независимости структуры: при обновлении программного обеспечения или СУБД (сокр. от «системы управления базами данных») база данных не должна менять свою структуру или свойства.

# Постановка задачи

Необходимо разработать базу данных, которая будет хранить данные о студентах, а именно: ФИО, группа, номер в группе, дата поступления, а также дату изменения определённых данных.

Многомодульность программы. Программа должна быть поделена на логические модули. Это упростит поиск ошибок при отладке и тестировании

консольного приложения.

Использование сложных типов данных - массивов, структур, файлов. Это необходимо для более простой и интуитивной обработки данных в коде программы.

Режим работы видеосистемы – текстовый/графический. Для начала необходимо определиться с типом интерфейса и с элементами управления, затем необходимо изучить способы их реализации.

Программа должна поддерживать функции ввод данных о студентах, их хранение, вывод, сортировку. Весь этот комплекс условий осуществляется при помощи многочисленных функций.

Устройство ввода-вывода – клавиатура и мышь. Необходимо различать и идентифицировать действия, произведённые с их помощью, это облегчит использование программы.

# **Выбор решения**

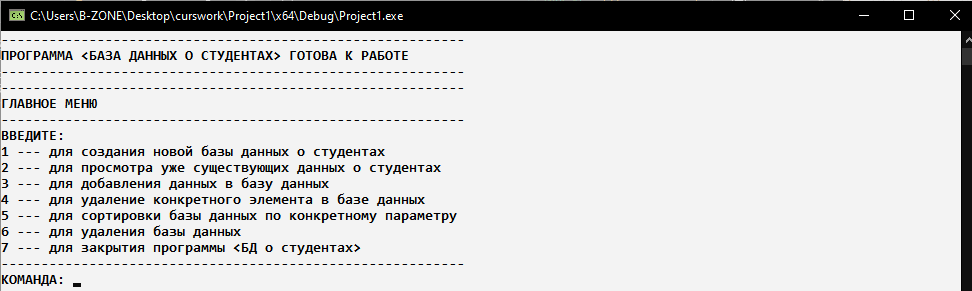
### Модули программы

Программа состоит из трёх файлов: **mine.cpp** (файл с функций мейн), **sourse.cpp** (файл с описанием структуры и всех функций) и **header.h** (заголовочный файл.). **Header.h** был разработан для данной курсовой работы и содержит в себе следующие элементы: подключение всех нужных библиотек, объявление функций и переменных.

Данный файл необходим для связывания всех модулей программы между собой, для ускорения вызова функций и работы программы в целом, во избежание многократного объявления переменных в разных модулях программы, что могло привести к ошибке в работе программы. Также благодаря нему не требуется в каждом модуле программы каждый раз подключать все требуемые библиотеки, достаточно подключить лишь «**header.h**».

### Интерфейс программы

Интерфейс программы выполнен максимально минимализирован. Ввод и вывод необходимых данных осуществляется в консоли. Перед каждым «важным» для понимания выполнения программы действии на экран выводится список допустимых команд (пример представлен на рисунке 1 - функция главного меню программы, которая содержит в себе вызов вспомогательных функций).



**Рисунок 1**

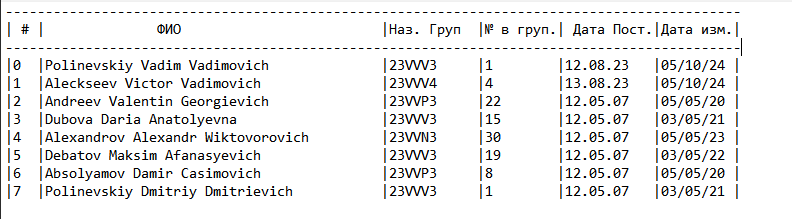
Основное предназначения интерфейса, как уже отмечалось, демонстрация возможных действий над функциями, которые описаны в пункте: «Постановка задачи».

Интерфейс не отличается такими свойствами как использование различных цветов, задержек в выводе и вводе данных, что с одной стороны является слишком «простым» и «устаревшим» способом оформления приложений, но с другой стороны, он не содержит ничего «лишнего», что затрудняло бы восприятия функционала программы.

Цветовая схема представлена лишь двумя стандартными цветами консоли Операционной системы Windows и консольных средств среды Visual Studio цветами: белый и чёрный.

Кроме того, чтобы избавить потенциального пользователя от ошибок, связанным с неверным восприятием данных ввода и вывода, все важные места программы отделены в консоли при помощи разделителей, которым является либо табулирование и перенос каретки (), либо литералы типа «-», «|», что реализовывается при помощи функции-вывода стандартной библиотеки языка си (*printf()*).

Для лёгкости восприятия информации пользователем, файлы, которые являются неким «Хранилищем» базы данных, тоже имели базовые графические разделения, приведённые к табличному виду(пример базы данных из 8 «потенциальных» данных о студентах представлен на рисунке 2).



**Рисунок 2**

# **Описание разработки программы**

Определившись с решением вопроса о графическом интерфейсе программы, появилась решить комплекс проблем, связанным с определением возможных способах реализации необходимого функционала, в основном связанным с хранением и обработкой данных некоторого «хранилища». Была идея, связанная с организацией Базы данных в среде Microsoft Exel, для наиболее прикладного использования программы позднее, но этот вариант не подошёл по причине необходимости написания и использования новых библиотек и ,соответственно, новых функций, что не было чем-то «сложным» и «труднодоступным» для изучения, но не отвечало требованием, связанным с высокой, насколько это возможно, скоростью разработки и простотой «осознания» логики. Поэтому был выбран вариант с реализацией базы данных в виде обычного .txt- файла, что соответсоволо изучаемым за курс программирования на языке СИ навыкам.

В программе использованы языки Ассемблер, а также C с его стандартными библиотеками: <stdio.h>, <string.h>, <locale.h> , <stdlib.h>, <time.h>, <windows.h>, с включённым в них комплексом функций.

Разработка началась с реализации набора функций ,основных и «зависимых» от прикладного использования знания курса. Ими являются комплекс функций вычислений и выполнения необходимых действий: выделения памяти, разделения логики вместе с функциями *database\_creation\_function()* и *sorting\_database\_items()*. Вначале, до завершающих этапов разработки, они не были связаны многомодульностью.

Вначале программа была написана на «чистом» си. Ближе к концу, когда была точно определена логика программы, были добавлены ассемблерные вставки.

Программа содержит 12 функций, связанных между собой (Иерархическая структура программы (Приложение B)).

# **Отладка и тестирование**

В качестве среды разработки была выбрана программа Visual Studio 2022. Программа обладает всеми средствами, необходимыми при разработке и отладке, а также позволяет использовать более эффективные способы разработки. Для отладки использовались несколько возможностей программы Visual Studio: точка останова, анализ содержимого переменных и трассировку.

Тестирование проводилось только мной одним. Оно было связано с использованием различных вариантов «ввода» переменных и анализировании, требующих «особого внимания» переменных в функциях, фрагментах кода, а также. В такие моменты возникали идеи различных вариантов оформления логики программы, в зависимости от простоты и эффективности.

### **Установка точки останова и запуск отладчика:**

Точки останова полезны, если вам известны строка или раздел кода, которые вы хотите подробно изучить в среде выполнения. Дополнительные сведения о различных типах точек останова, которые можно задать, например об условных точках останова и точках останова в функциях.

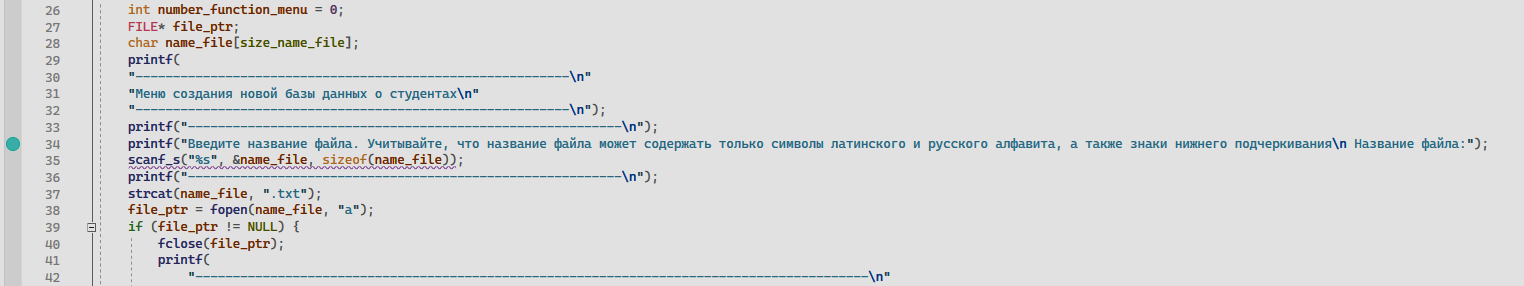
Для отладки нужно запустить приложение с отладчиком, подключённым к процессу приложения. Для этого:

Нажмите клавишу **F5** (**Отладка** > **Начать отладку**), которая является наиболее распространенным методом.

Однако сейчас у вас, возможно, не задано ни одной точки останова для проверки кода приложения, поэтому мы сначала зададим их, а затем начнём отладку. Точки останова — это самая основная и важная функция надёжной отладки. Точка останова указывает, где Visual Studio следует приостановить выполнение кода, чтобы вы могли проверить значения переменных или поведение памяти либо выполнение ветви кода.

Если вы открыли файл в редакторе кода, точку останова можно задать, щёлкнув в поле слева от строки кода.

Нажмите **клавишу** **F5** (**Отладка** > **запуска отладки**),а отладчик запускается в первую точку останова, с которой она сталкивается. Если приложение ещё не запущено, при нажатии **клавиши F5** запускается отладчик и выполняется остановка в первой точке останова.



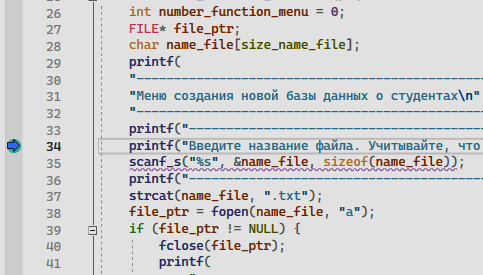
**Рисунок 3**

### Переход по коду в отладчике с помощью пошаговых команд

### **Переход по коду в отладчике с помощью пошаговых команд**

Мы указываем сочетания клавиш для большинства команд, так как они ускоряют навигацию по коду вашего приложения. Дополнительные сведения об использовании команд пошагового.

Для запуска приложения с подключённым отладчиком нажмите клавишу F11 (Отладка > Шаг с заходом). F11 — это команда Шаг с заходом, которая выполняет приложение с переходом к следующему оператору. При запуске приложения с помощью клавиши F11 отладчик останавливается на первом выполняемом операторе.



**Рисунок 4**

Жёлтая стрелка представляет оператор, на котором приостановлен отладчик. В этой же точке приостанавливается выполнение приложения (этот оператор пока не выполнен). Клавишу F11 удобно использовать для более детальной проверки потока выполнения.

### **Шаг с обходом по коду для пропуска функций**

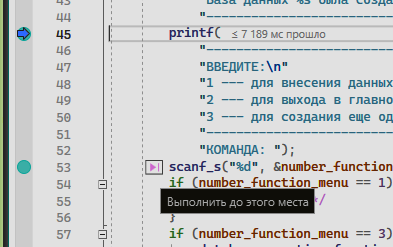
Когда вы находитесь в строке кода, представляющей собой вызов функции или метода, можно нажать клавишу F10 (Отладка > Шаг с обходом) вместо F11.

Клавиша F10 продолжает выполнение отладчика без захода в функции или методы в коде приложения (код продолжает выполняться). Нажав клавишу F10, вы можете обойти код, который вас не интересует. Так можно быстро перейти к важному для вас коду.

### **Быстрое выполнение до точки в коде с помощью мыши**

Использование кнопки Выполнение до щёлкнутого аналогично установке временной точки останова. Кроме того, эта команда удобна для быстрой работы в видимой области кода приложения. Выполнение до щёлкнутого можно использовать в любом открытом файле.

Пока в отладчике наведите указатель мыши на строку кода, пока не появится кнопка "Запустить" (выполнить выполнение здесь).



**Рисунок 5**

Нажмите кнопку выполнения до щёлкнутого (Выполнить до этого места). Отладчик продолжает выполнение до строки кода, которую вы щёлкнули.

Тестирование проводилось во время разработки и также после завершения разработки.

Отслеживание значение переменных в основном использовалось для проверки пунктов, связанных с обработкой строк.

# **Описание программы**

При запуске программы происходит вывод доступных команд, после чего пользователю необходимо выбрать пункт меню для дальнейшего использования базы данных. После выбора пользователем нужного пункта осуществляется вызов другой функции, которая ответственна за выполнение определённого действия.

|  |  |  |
| --- | --- | --- |
| Клавиши, вызывающие событие | Действие пользователя | Действие программы |
| 1, Enter | Выбор пункта «для Создание » новый базы данных о студентах | Вызывается диалоговое окно, ответственное за создание БД |
| 2, Enter | Выбор пункта «Для просмотра уже существующих данных о студентах | Вызывается диалоговое окно, ответственное за показ БД |
| 3, Enter | Выбор меню « для добавления элемента в существующую БД» | Вызывается диалоговое окно, ответственное за добавление элемента в БД |
| 4, Enter |  |  |
| 5, Enter | Вызов меню сортировки в БД | Вызывается диалоговое окно, ответственное за показ БД |
| 6, Enter | Вызов меню удаления БД | Вызывается диалоговое окно, ответственное за удаление БД |
| 7, Enter | Вызов меню завершения программы | Завершение программы |
| Не 1...7, Enter | Повторный вызов меню с командами | Рекурсивный вызов функции главного меню с командами. |

Таблица 1 Описание работы функции главного меню (*recursive\_menu()*).

|  |  |  |
| --- | --- | --- |
| 1, Enter | Пункт «Сортировка по ФИО» | Выполнение алгоритма «Сортировка по ФИО» |
| 2, Enter | Пункт «Сортировка по дате поступления» | Выполнение алгоритма «Сортировка по дате поступления» |
| 3, Enter | Пункт «Сортировка по названию группы» | Выполнение алгоритма «Сортировка по названию группы» |
| 4, Enter | Пункт «Сортировка по названию по дате изменения значения» | Выполнение алгоритма «Сортировка по дате изменения» |
| 5,Enter | Пункт «Главное меню» | Вызов функции главного меню с командами |
| не 1…5 | Пункт «Сортировки в БД» | Рекурсивный вызов функции |

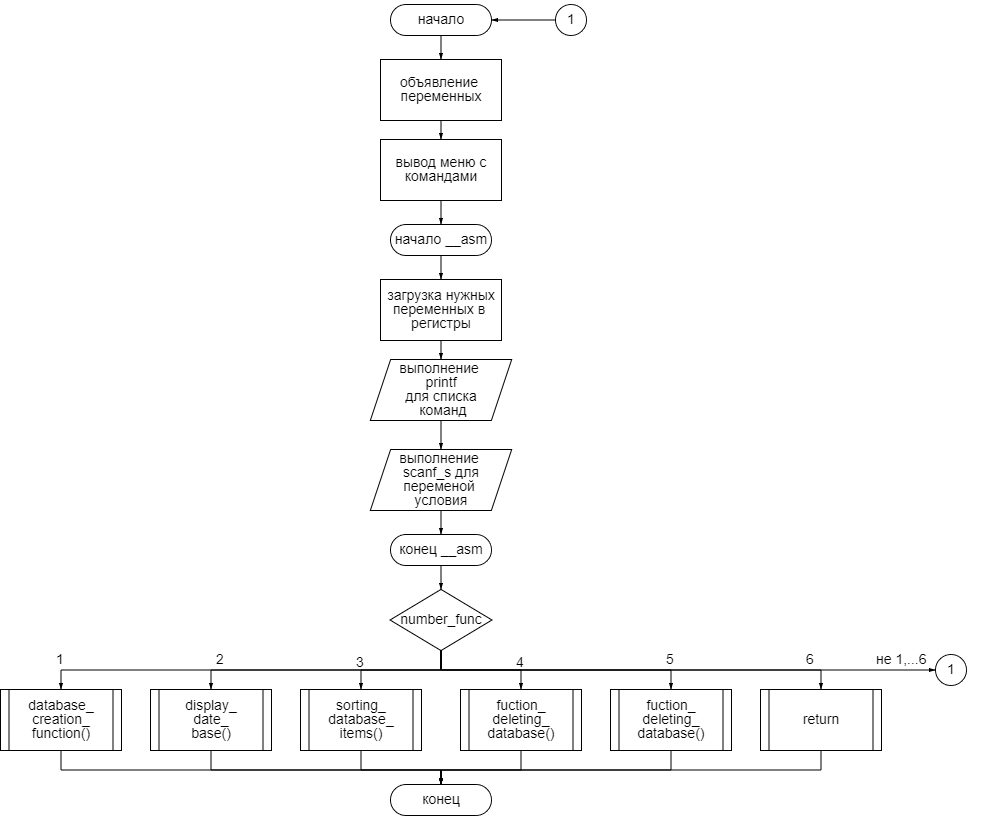
Таблица2. Описание функции сортировки (*sorting\_database\_items()*).

|  |  |  |
| --- | --- | --- |
| 1, Enter | Пункт занесения информации в БД при корректно введеннных данных | Выполнения алгоритма «Занесение информации в Бд» |
| 2, Enter | Пункт «отмены введения и перезаписи вводимых данных» | Выполнения алгоритма перезаписи данных, которые необходимо ввести в строку |
| 3, Enter | Пункт «изменение конкретного параметра» | Выполнение алгоритма перезаписи определённого параметра |
| 4, Enter | Пункт «Отмена занесения ранее введённых параметров и возвращение в Главное меню» | Вызов функции главного меню с командами. |

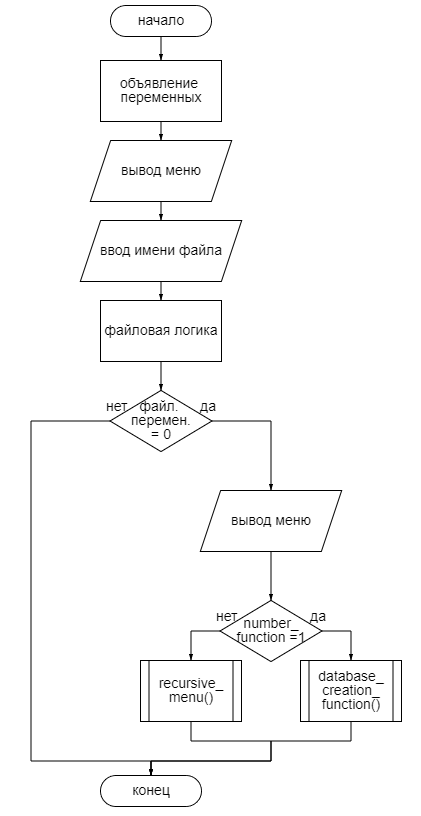
Таблица 3. Описание функции добавления информации(*enter\_data()*)



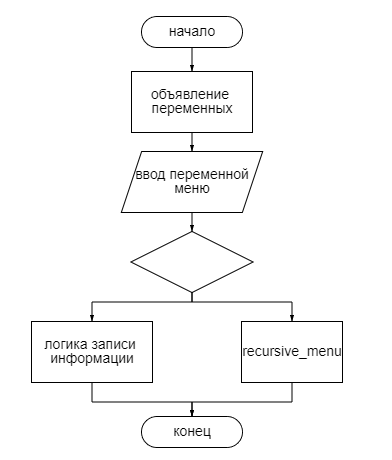
**Рисунок 6.Схема функции main()**



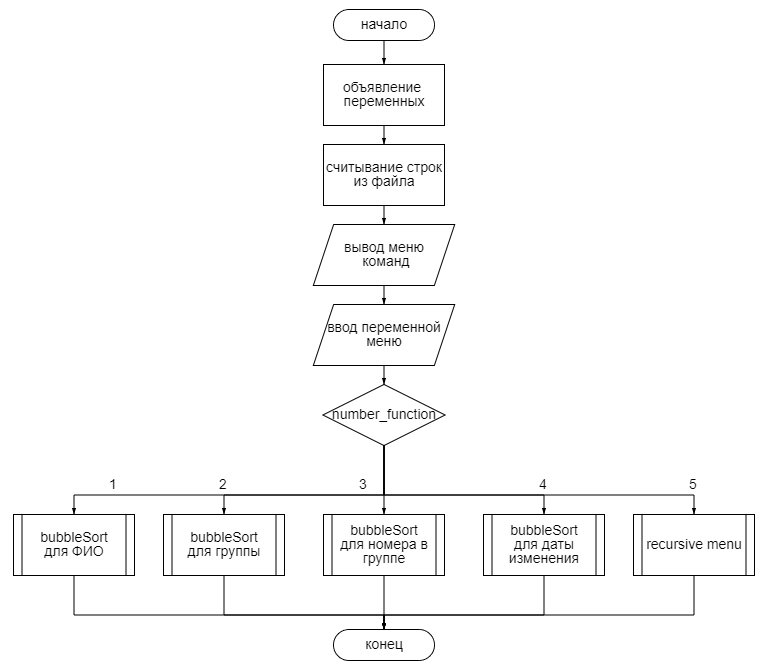
**Рисунок 7. Схема recursive\_menu()**



**Рисунок 8.database\_creation\_function()**



**Рисунок 9.database\_new\_element()**

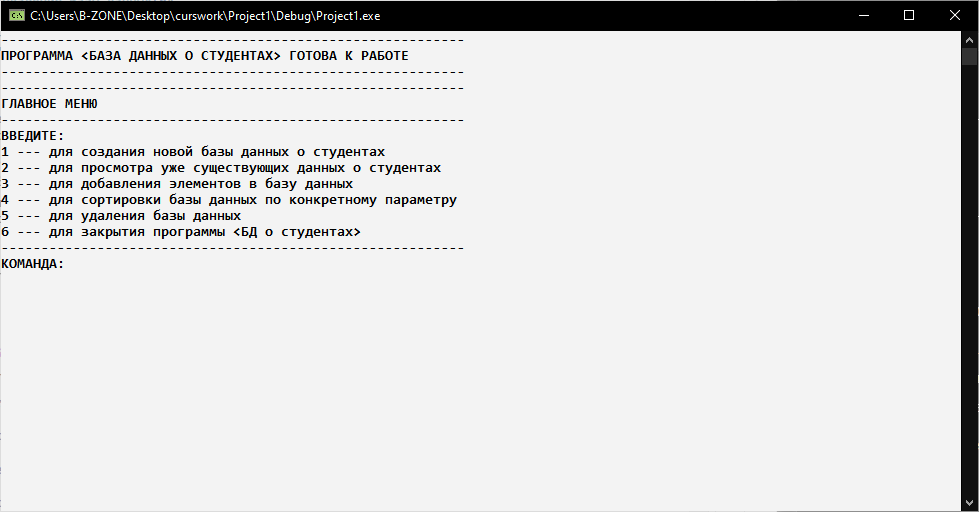


**Рисунок 10.sorting\_database\_items()**

# **Руководство пользователя**

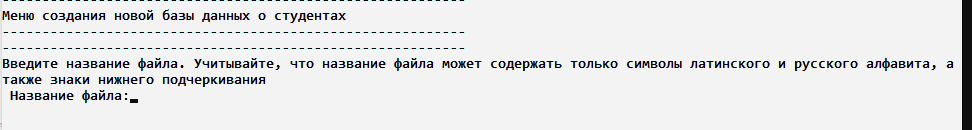
Программа предназначена для работы с базой данных и имеет интуитивно понятный интерфейс, поддерживающий вывод списка команд, информации о вводе значений, а также данных из файла БД.

**Главное меню:**



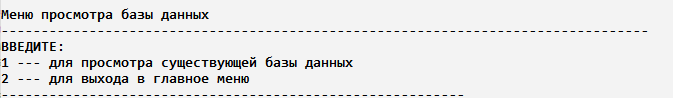
**Рисунок 11. Главное меню**

**Меню создания новой БД.**



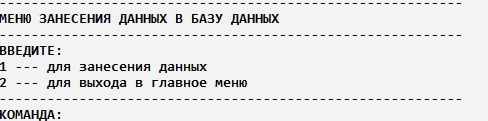
**Рисунок 12. Меню создания новой БД**

**Меню Просмотра БД**



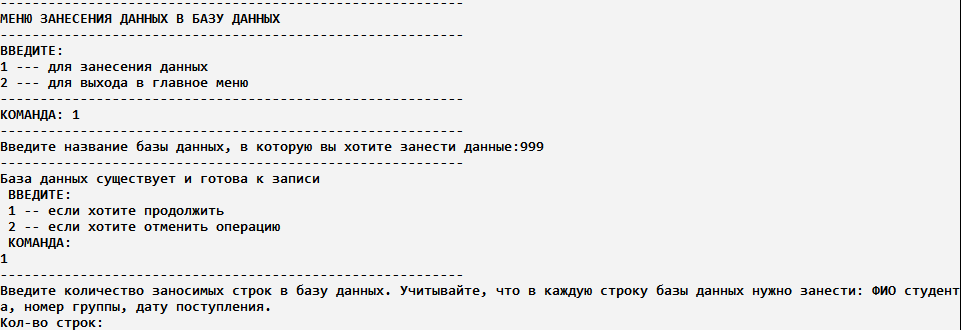
**Рисунок 13. Меню Просмотра БД**

**Меню Занесения данных в БД (Предварительный этап)**



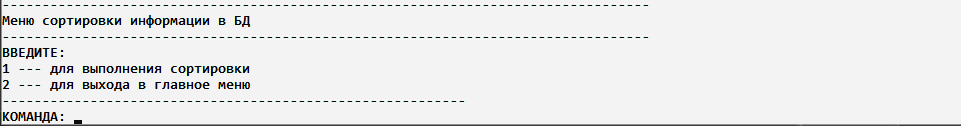
**Рисунок 14. Меню Занесения данных в БД (Предварительный этап)**

**Меню занесения данных (Основной этап)**



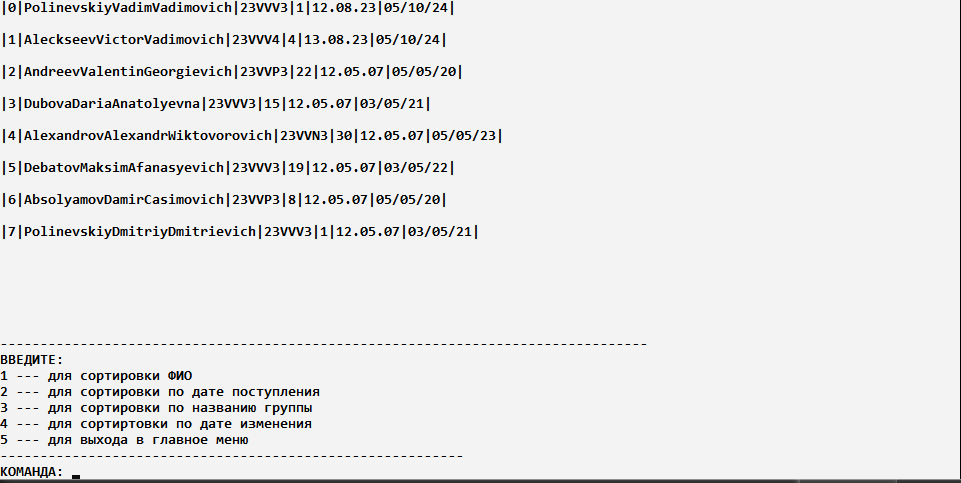
**Рисунок 15. Меню занесения данных (Основной этап)**

**Меню сортировки информации в БД (Предварительный этап)**



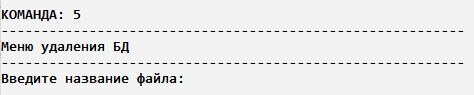
**Рисунок 16. Меню сортировки информации в БД (Предварительный этап)**

**Меню сортировки информации в БД (Основной этап 1):**



**Рисунок 17. Меню сортировки информации в БД (Основной этап 1):**

**Меню удаления база данных:**



**Рисунок 18. Меню удаления база данных:**

# **Заключение**

В результате выполнения Курсовой работы, я обобщил знания языка си, полученного за курс различных дисциплин: изучил и разобрался во многих функциях с способах программирование, которые раньше мне не удавалось «воспринять» из-за «отдалённости» таковых от прикладного программирования. Так, например, я в подробностях изучил функции strtok(), srtcmp(), углубленные знания которых пригодились мне для записи в .txt-файл и считывания данных о студентах из «Хранилища». Также я подробнее ознакомился с выделением, распределением и удалением динамической памяти(функции malloc() и relloc(), free()). Также я лучше стал разбираться в принципах работы с массивами и указателями на них соответственно. Многомодульность, которую я раньше избегал в связи с отсутствием необходимости применения в лабораторных работах, теперь изучена мной. Кроме того, я получил достаточно важные в «век отчётности и документации» навыки оформления

# **Список используемых источников**

1. Microsoft Corporation. Разработка приложений на Microsoft Visual C++ 6.0. Учебный курс: Официальное пособие Microsoft для самостоятельной подготовки / Пер. с англ.- М.: Издательско-торговый дом «Русская Редакция», 2000. – 576 стр.: илл
2. Ашарина, И.В. Основы программирования на языках С и С++: Курс лекций для высших учебных заведений / И.В. Ашарина. — М.: Гор. линия-Телеком, 2018. — 208 c.
3. Белоцерковская, И. Е. Алгоритмизация. Введение в язык программирования С++ : краткий учебный курс / И. Е. Белоцерковская, Н. В. Галина, Л. Ю. Катаева. - Москва : ИНТУИТ, 2016. - 141 с. - Текст : электронный. - URL: https://znanium.ru/catalog/product/2137409
4. Гагарина, Л. Г. Основы программирования на языке С : учебное пособие / Л.Г. Гагарина, Е.Г. Дорогова. — 2-е изд., испр. и доп. — Москва : ИНФРА-М, 2023. — 269 с. — (Высшее образование: Бакалавриат). — DOI 10.12737/1035562. - ISBN 978-5-16-015470-1. - Текст : электронный. - URL: https://znanium.ru/catalog/product/1907425
5. Руслан Аблязов. Программирование на ассемблере на платформе x86-64. 2016 г.

# **Приложение А**

Листинг программы:  
**Header.h:**

import pygame

#define \_CRT\_SECURE\_NO\_WARNINGS

#pragma once

#include <stdio.h>

#include <string.h>

#include <locale.h>

#include <stdlib.h>

#include <time.h>

#include <windows.h>

typedef struct Node {

int number\_in\_list;

char name\_student[50];

char name\_group[8];

int number\_student\_in\_group;

char date\_of\_receipt[12];

char date\_of\_entry[9];

struct Node\* next;

};

#define size\_name\_file 15

#define size\_name\_file\_copy 21

void database\_new\_element();

void removeSpaces(char str[]);

void enter\_data(FILE\* file\_ptr, char file\_name[], int number\_of\_repetitions);

void recursive\_menu();

**Source.cpp:**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include "Header.h"

void database\_new\_element();

void enter\_data(FILE\* file\_ptr, char file\_name[], int number\_of\_repetitions);

void database\_creation\_function();

void recursive\_menu();

void fuction\_deleting\_database();

void database\_creation\_function() {

int number\_function\_menu = 0;

FILE\* file\_ptr;

char name\_file[size\_name\_file];

printf(

"----------------------------------------------------------\n"

"Меню создания новой базы данных о студентах\n"

"----------------------------------------------------------\n");

printf("----------------------------------------------------------\n");

printf("Введите название файла. Учитывайте, что название файла может содержать только символы латинского и русского алфавита, а также знаки нижнего подчеркивания\n Название файла:");

scanf\_s("%s", &name\_file, sizeof(name\_file));

printf("----------------------------------------------------------\n");

strcat(name\_file, ".txt");

file\_ptr = fopen(name\_file, "a");

if (file\_ptr != NULL) {

fclose(file\_ptr);

printf(

"------------------------------------------------------------------------------------------\n"

"База данных %s была создана и размещена в каталоге данной программы\n"

"------------------------------------------------------------------------------------------\n", name\_file);

printf(

"----------------------------------------------------------\n"

"ВВЕДИТЕ:\n"

"1 --- для внесения данных в эту базу данных\n"

"2 --- для выхода в главное меню\n"

"3 --- для создания еще одной базы данных"

"----------------------------------------------------------\n"

"КОМАНДА: ");

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

if (number\_function\_menu == 1) {

/\*enter\_data();\*/

}

if (number\_function\_menu == 3) {

database\_creation\_function();

}

else {

recursive\_menu();

}

}

else {

printf(

"----------------------------------------------------------------------------------------\n"

"---Файл %s не был создан. Возможно был превышен допустимый размер названия данных или использован недопустимый символ\n"

"----------------------------------------------------------------------------------------\n", file\_ptr);

printf("----------------------------------------------------------\n"

"ВВЕДИТЕ:\n"

"1 --- для повтора попытки создания базы данных\n"

"2 --- для выхода в главное меню\n"

"----------------------------------------------------------\n"

"КОМАНДА: ");

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

if (number\_function\_menu == 1) {

database\_creation\_function();

}

else {

recursive\_menu();

}

}

}

void database\_new\_element() {

char temp\_str[100];

int isEmpty = 1;

char file\_name[size\_name\_file];

int number\_function\_menu = 0;

FILE\* file\_ptr;

printf(

"----------------------------------------------------------\n"

"МЕНЮ ЗАНЕСЕНИЯ ДАННЫХ В БАЗУ ДАННЫХ\n"

"----------------------------------------------------------\n"

"ВВЕДИТЕ:\n"

"1 --- для занесения данных\n"

"2 --- для выхода в главное меню\n"

"----------------------------------------------------------\n"

"КОМАНДА: ");

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

if (number\_function\_menu == 1) {

printf(

"----------------------------------------------------------\n"

"Введите название базы данных, в которую вы хотите занести данные:");

scanf\_s("%s", &file\_name, sizeof(file\_name));

strcat(file\_name, ".txt");

file\_ptr = fopen(file\_name, "r+");

if (file\_ptr != NULL) {

printf(

"----------------------------------------------------------\n"

"База данных существует и готова к записи \n ВВЕДИТЕ:\n 1 -- если хотите продолжить\n 2 -- если хотите отменить операцию\n КОМАНДА: \n");

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

if (number\_function\_menu == 1) {

if (fgets(temp\_str, 100, file\_ptr) == NULL) {

fprintf(file\_ptr,

"-------------------------------------------------------------------------------------------\n"

"| # | ФИО |Название группы | Номер в группе | Дата Поступления|\n"

"-------------------------------------------------------------------------------------------\n");

}

fclose(file\_ptr);

file\_ptr = fopen(file\_name, "a");

int number\_of\_repetitions;

printf(

"----------------------------------------------------------\n"

"Введите количество заносимых строк в базу данных. Учитывайте, что в каждую строку базы данных нужно занести: ФИО студента, номер группы, дату поступления.\n"

"Кол-во строк:");

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

enter\_data(file\_ptr, file\_name, number\_of\_repetitions);

}

if (number\_function\_menu == 2) {

database\_new\_element();

}

else {

printf("Была введена некоректная команда. Вы возвращены в главное меню\n");

recursive\_menu();

}

}

}

if (number\_function\_menu == 2) {

recursive\_menu();

}

else {

printf("База данных не существует.");

recursive\_menu();

}

}

void fuction\_deleting\_database() {

int number\_fuction\_menu;

char name\_file[size\_name\_file];

printf("Введите название файла:");

scanf\_s("%s", &name\_file, sizeof(name\_file));

strcat(name\_file, ".txt");

char\* filePath = name\_file;

if (remove(name\_file) == 0) {

printf(

"----------------------------------------------------------\n"

"Файл %s успешно удален \n"

"----------------------------------------------------------\n", name\_file);

printf(

"----------------------------------------------------------\n"

"ВВЕДИТЕ:\n"

"1 --- для выхода в главное меню\n"

"----------------------------------------------------------\n"

"КОМАНДА: "

);

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

if (number\_fuction\_menu == 1) {

recursive\_menu();

}

}

else {

printf(

"----------------------------------------------------------\n"

"Не удалось удалить файл %s\n"

"----------------------------------------------------------\n", name\_file);

printf("----------------------------------------------------------\n"

"ВВЕДИТЕ:\n"

"1 --- для повтора попытки удаления базы данных\n"

"2 --- для выхода в главное меню\n"

"----------------------------------------------------------\n"

"КОМАНДА: ");

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

if (number\_fuction\_menu == 1) {

database\_creation\_function();

}

else {

recursive\_menu();

}

}

}

void removeSpaces(char str[]) {

int i, j = 0;

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

if (str[i] != ' ') {

str[j++] = str[i];

}

}

str[j] = '\0';

}

void appendStringToArray(char\* temp\_str, char\*\*\* array, int\* size) {

(\*array) = (char\*\*)realloc((\*array), ((\*size) + 1) \* sizeof(char\*));

(\*array)[\*size] = \_strdup(temp\_str);

(\*size)++;

}

void bubbleSort(char\*\* arr, int n, int number\_skip) {

char\* token\_1, \* token\_2, \* a = NULL, \* b = NULL;

char\* str1 = 0, \* str2 = 0;

int k;

for (int i = 0; i < n - 1; i++) {

for (int j = 0; j < n - i - 1; j++) {

k = 0;

str1 = \_strdup(arr[j]);

str2 = \_strdup(arr[j + 1]);

token\_1 = strtok(str1, "|");

while (token\_1 != NULL) {

token\_1 = strtok(NULL, "|");

if (k == number\_skip) {

//a = &token\_1[0];

a = \_strdup(token\_1);

}

k++;

}

k = 0;

token\_2 = strtok(str2, "|");

while (token\_2 != NULL) {

token\_2 = strtok(NULL, "|");

if (k == number\_skip) {

//b = &token\_2[0];

b = \_strdup(token\_2);

}

k++;

}

if (strcmp(a, b) > 0) {

char\* temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

}

}

}

}

void addSpacesBetweenCapitalLetters(char\* str) {

int i = 0;

while (str[i] != '\0') {

if (str[i] >= 'A' && str[i] <= 'Z') {

printf(" ");

}

printf("%c", str[i]);

i++;

}

}

void sorting\_database\_items() {

int number\_in\_list = 0, number\_student\_in\_group = 0, number\_function\_menu, line = 0, count\_line\_in\_file = 0, i = 0;

char name\_student[50], name\_group[8], date\_of\_receipt[12], temp\_str[100], date\_of\_entry[9], file\_name[size\_name\_file], temp\_str\_elemets\_DB[50], \* token, \* array\_str;

//char\* token, \*str;

FILE\* file\_ptr;

printf("---------------------------------------------------------------------------------\n"

"Меню сортировки информации в БД\n"

"---------------------------------------------------------------------------------\n"

"ВВЕДИТЕ:\n"

"1 --- для выполнения сортировки\n"

"2 --- для выхода в главное меню\n"

"----------------------------------------------------------\n"

"КОМАНДА: ");

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

if (number\_function\_menu == 1) {

printf("---------------------------------------------------------------------------------\n"

"Введите название базы данных, в которой следует выполнить сортировку:");

scanf\_s("%14s", &file\_name, sizeof(file\_name));

strcat(file\_name, ".txt");

file\_ptr = fopen(file\_name, "r");

if (file\_ptr != NULL) {

while (fgets(temp\_str, 100, file\_ptr) != NULL) {

count\_line\_in\_file++;

}

fclose(file\_ptr);

file\_ptr = fopen(file\_name, "a+");

if (count\_line\_in\_file <= 3) {

printf("---------------------------------------------------------------------------------\n"

"База данных пуста\n"

"---------------------------------------------------------------------------------\n"

"ВВЕДИТЕ:\n"

"1 --- для повторения попытки\n"

"2 --- для выхода в главное меню\n"

"Команда: ");

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

while (1) {

switch (number\_function\_menu) {

case 1:

sorting\_database\_items();

case 2:

recursive\_menu();

default:

printf("Была введена некоректная команда. Повторите попытку\n");

}

}

}

else {

char\*\* dynamicArray = NULL;

int size = 0;

while (fgets(temp\_str, 100, file\_ptr) != NULL) {

line++;

if (line >= 4) {

removeSpaces(temp\_str);

appendStringToArray(temp\_str, &dynamicArray, &size);

}

}

printf("---------------------------------------------------------------------------------\nЭлементы БД:\n---------------------------------------------------------------------------------\n");

for (i = 0; i < size; i++) {

printf("%s\n", dynamicArray[i]);

}

printf("\n---------------------------------------------------------------------------------\n"

"ВВЕДИТЕ:\n"

"1 --- для сортировки ФИО\n"

"2 --- для сортировки по дате поступления\n"

"3 --- для сортировки по названию группы\n"

"4 --- для сортиртовки по дате изменения \n"

"5 --- для выхода в главное меню\n"

"----------------------------------------------------------\n"

"КОМАНДА: ");

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

if (number\_function\_menu == 1) {

bubbleSort(dynamicArray, i, 0);

}

if (number\_function\_menu == 2) {

bubbleSort(dynamicArray, i, 4);

}

if (number\_function\_menu == 3) {

bubbleSort(dynamicArray, i, 1);

}

if (number\_function\_menu == 4) {

bubbleSort(dynamicArray, i, 5);

}

if (number\_function\_menu == 5) {

recursive\_menu();

}

printf("---------------------------------------------------------------------------------\n"

"Отсортированное значение:");

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

//addSpacesBetweenCapitalLetters(dynamicArray[i]);

printf("%s\n", dynamicArray[i]);

}

printf("--------------------------------------------------------------------------------\n");

printf("\n---------------------------------------------------------------------------------\n"

"ВВЕДИТЕ:\n"

"1 --- для записи данного значения в новый файл\n"

"2 --- для выхода в главное меню\n"

"3 --- для продолжения сортировки другой базы данных\n"

"----------------------------------------------------------\n"

"КОМАНДА: ");

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

while (1) {

switch (number\_function\_menu) {

case 1: {

printf("Введите название файла. Учитывайте, что название файла может содержать только символы латинского и русского алфавита, а также знаки нижнего подчеркивания\n Название файла:");

scanf\_s("%s", &file\_name, sizeof(file\_name));

printf("----------------------------------------------------------\n");

strcat(file\_name, "copy.txt");

file\_ptr = fopen(file\_name, "a");

if (file\_ptr == NULL) {

printf("При работе с файлом произошла ошибка, пожалуйста повторите попытку\n");

break;

}

else {

for (i = 0; i < size; i++) {

fprintf(file\_ptr, "%s", dynamicArray[i]);

}

fclose(file\_ptr);

printf("Запись в файл %s прошла успешно", file\_name);

free(dynamicArray);

recursive\_menu();

}

}

case 2: {

free(dynamicArray);

recursive\_menu();

}

case 3: {

sorting\_database\_items();

free(dynamicArray);

}

default:

printf("Была введена некоректная команда. Повторите попытку\n");

break;

}

}

}

}

}

if (number\_function\_menu == 2) {

recursive\_menu();

}

if (number\_function\_menu == 3) {

;;

}

if (number\_function\_menu != 1 && number\_function\_menu != 2) {

printf("Была введена некоректная команда. Повторите попытку\n");

sorting\_database\_items();

}

}

Node\* createNode(int number\_in\_list, char name\_student[], char name\_group[], int number\_student\_in\_group, char date\_of\_receipt[], char date\_of\_entry[]) {

Node\* newNode = (Node\*)malloc(sizeof(Node));

newNode->number\_in\_list = number\_in\_list;

strcpy(newNode->name\_student, name\_student);

strcpy(newNode->name\_group, name\_group);

newNode->number\_student\_in\_group = number\_student\_in\_group;

strcpy(newNode->date\_of\_receipt, date\_of\_receipt);

strcpy(newNode->date\_of\_entry, date\_of\_entry);

newNode->next = NULL;

return newNode;

}

void addNode(FILE\* file\_ptr, char file\_name[], Node\*\* head, int number\_in\_list, char name\_student[], char name\_group[], int number\_student\_in\_group, char date\_of\_receipt[], char date\_of\_entry[], int free\_condition) {

Node\* newNode = createNode(number\_in\_list, name\_student, name\_group, number\_student\_in\_group, date\_of\_receipt, date\_of\_entry);

if (free\_condition == 0) {

file\_ptr = fopen(file\_name, "a");

fprintf(file\_ptr, "|%-3d|", newNode->number\_in\_list);

fprintf(file\_ptr, "%-49s|", newNode->name\_student);

fprintf(file\_ptr, "%-9s|", newNode->name\_group);

fprintf(file\_ptr, "%-3d|", newNode->number\_student\_in\_group);

fprintf(file\_ptr, "%-9s|", newNode->date\_of\_receipt);

fprintf(file\_ptr, "%-9s|", newNode->date\_of\_entry);

fprintf(file\_ptr, "\n");

fclose(file\_ptr);

free(newNode);

}

else {

if (\*head == NULL) {

\*head = newNode;

}

else {

Node\* temp = \*head;

while (temp->next != NULL) {

temp = temp->next;

}

temp->next = newNode;

}

}

}

void display\_date\_base() {

char file\_name[size\_name\_file];

char temp\_str[100];

int number\_function\_menu;

FILE\* file\_ptr;

printf("---------------------------------------------------------------------------------\n"

"Меню просмотра базы данных\n"

"---------------------------------------------------------------------------------\n"

"ВВЕДИТЕ:\n"

"1 --- для просмотра существующей базы данных\n"

"2 --- для выхода в главное меню\n"

"----------------------------------------------------------\n"

"КОМАНДА: ");

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

if (number\_function\_menu == 1) {

printf("Введите название базы данных для чтения:");

scanf\_s("%s", &file\_name, sizeof(file\_name));

strcat(file\_name, ".txt");

file\_ptr = fopen(file\_name, "r");

if (file\_ptr != NULL) {

printf("--------------------------------------------------------\n БАЗА ДАННЫХ <%s>\n --------------------------------------------------------\n", file\_name);

while (fgets(temp\_str, 100, file\_ptr) != NULL) {

printf("%s", temp\_str);

}

printf("\n");

fclose(file\_ptr);

}

else {

printf("-------------------------------------------------\nБаза данных <%s>не найдена\n-------------------------------------------------\n", file\_name);

printf("ВВЕДИТЕ:\n"

"1 --- для повтора попытки\n"

"2 --- для выхода в главное меню\n"

"----------------------------------------------------------\n"

"КОМАНДА: ");

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

if (number\_function\_menu == 1) {

display\_date\_base();

}

if (number\_function\_menu == 2) {

recursive\_menu();

}

else {

printf("Была введена некоректная команда. Повторите попытку\n");

display\_date\_base();

}

}

}

if (number\_function\_menu == 2) {

recursive\_menu();

}

else {

printf("-------------------------------------------------\nБыла введена некоректная команда. Повторите попытку\n-------------------------------------------------\n");

display\_date\_base();

}

}

void enter\_data(FILE\* file\_ptr, char file\_name[], int number\_of\_repetitions) {

const char head3[] =

"----------------------------------------------------------\n"

"ВВЕДИТЕ:"

"1 --- если данные введены корректно и вы готовы внести их в базу\n"

"2 --- если хотите внести все введенные ранее в строку данные заново \n"

"3 --- если хотите изменить конкретный параметр\n"

"4 --- если хотите отменить введение и вернуться в главное меню\n"

"----------------------------------------------------------\n";

int number\_in\_list = 0;

char name\_student[50];

char name1[20];

char name2[15];

char name3[15];

char name\_group[8];

char date\_of\_entry[9];

int number\_student\_in\_group = 0;

char date\_of\_receipt[12];

int number\_function\_menu;

for (number\_in\_list; number\_in\_list < number\_of\_repetitions; number\_in\_list++) {

Node\* head = NULL;

printf("----------------------------------------------------------\n");

printf("ФИО студента: \n ");

printf("Фамилия:");

scanf\_s("%s", name1, sizeof(name1));

printf("Имя: ");

scanf\_s("%s", name2, sizeof(name2));

printf("Отчество:");

scanf\_s("%s", name3, sizeof(name3));

printf("Название группы: ");

scanf\_s("%s", name\_group, sizeof(name\_group));

printf("Порядковый номер студента в группе ");

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

printf("Дата поступления: ");

scanf\_s("%s", date\_of\_receipt, sizeof(date\_of\_receipt));

printf("----------------------------------------------------------\n");

printf("ВВЕДЕННЫЕ ДАННЫЕ: %d | %s %s %s | %s | %d | %s |\n", number\_in\_list + 1, name1, name2, name3, name\_group, number\_student\_in\_group, date\_of\_receipt);

printf("----------------------------------------------------------\n");

printf("%s", head3);

printf("КОМАНДА: ");

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

if (number\_function\_menu == 1) {

time\_t mytime = time(NULL);

struct tm\* now = localtime(&mytime);

strftime(date\_of\_entry, sizeof(date\_of\_entry), "%D", now);

strcpy(name\_student, name1);

strcat(name\_student, " ");

strcat(name\_student, name2);

strcat(name\_student, " ");

strcat(name\_student, name3);

addNode(file\_ptr, file\_name, &head, number\_in\_list, name\_student, name\_group, number\_student\_in\_group, date\_of\_receipt, date\_of\_entry, 0);

}

if (number\_function\_menu == 2) {

number\_in\_list--;

}

if (number\_function\_menu == 3) {

while (number\_function\_menu != 5) {

printf(

"----------------------------------------------------------\n"

"ВВЕДИТЕ : \n"

"1 --- если хотите изменить ФИО студента\n"

"2 --- если хотите изменить название группы\n"

"3 --- ecли хотите изменить порядковый номер студента в группе\n"

"4 --- если хотите изменить дату поступления\n"

"5 --- если хотите вернуться к заполнению\n"

"6 --- если хотите закончить ввод и вернуться в главное меню"

"----------------------------------------------------------\n"

"КОМАНДА:");

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

switch (number\_function\_menu)

{

case 1:

printf("------------------------------------------------------------- \n Старое значение ФИО студента: %s \n ------------------------------------------------------------\n", name\_student);

printf("Новое значение ФИО студента: ");

printf("Фамилия:");

scanf\_s("%s", name1, sizeof(name1));

printf("Имя: ");

scanf\_s("%s", name2, sizeof(name2));

printf("Отчество:");

scanf\_s("%s", name3, sizeof(name3));

case 2:

printf("----------------------------------------------------------\n Старое значение названия группы: %s \n", name\_group);

printf("Новое значение названия группы: ");

scanf\_s("%s", &name\_group);

case 3:

printf("----------------------------------------------------------\n Старое значение порядкового номера студента в группе: %d \n", number\_student\_in\_group);

printf("Новое значение порядкого номера студента: ");

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

case 4:

printf("----------------------------------------------------------\nСтарое значение даты поступления студента: %s \n", date\_of\_receipt);

printf("Новое значение названия группы: ");

scanf\_s("%s", &date\_of\_receipt, sizeof(date\_of\_receipt));

case 5:

printf("----------------------------------------------------------\n");

printf("ВВЕДЕННЫЕ ДАННЫЕ: %d | %s %s %s | %s | %d | %s |\n", number\_in\_list + 1, name1, name2, name3, name\_group, number\_student\_in\_group, date\_of\_receipt);

printf("----------------------------------------------------------\n");

break;

case 6:

recursive\_menu();

default:

printf("----------------------------------------------------------\nБыла введена некоректная команда\n----------------------------------------------------------\n");

}

printf("----------------------------------------------------------\n");

printf("ВВЕДЕННЫЕ ДАННЫЕ: %d | %s %s %s | %s | %d | %s |\n", number\_in\_list + 1, name1, name2, name3, name\_group, number\_student\_in\_group, date\_of\_receipt);

printf("----------------------------------------------------------\n");

}

}

if (number\_in\_list == number\_of\_repetitions) {

printf(

"----------------------------------------------------------\n"

"Вы успешно ввели %d строк в базу данных\n"

"----------------------------------------------------------\n"

"ВВЕДИТЕ:\n"

"1 --- для внесения дополнительных данных в базу\n"

"2 --- для просмотра базы данных\n"

"3 --- для выхода в главное меню"

"----------------------------------------------------------\n"

"КОМАНДА:", number\_in\_list);

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

if (number\_function\_menu == 1) {

printf(

"----------------------------------------------------------\n"

"Введите количество заносимых в базу данных строк: \n");

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

/\*enter\_data(file\_ptr, file\_name, number\_of\_repetitions);\*/

}

if (number\_function\_menu == 2) {

}

else {

break;

}

}

}

}

void recursive\_menu() {

int number\_function\_menu = 0;

char str1[] = "%s";

char str2[] = "%d";

char head2[] =

"----------------------------------------------------------\n"

"ГЛАВНОЕ МЕНЮ\n"

"----------------------------------------------------------\n"

"ВВЕДИТЕ:\n"

"1 --- для создания новой базы данных о студентах\n"

"2 --- для просмотра уже существующих данных о студентах\n"

"3 --- для добавления элементов в базу данных\n"

"4 --- для сортировки базы данных по конкретному параметру\n"

"5 --- для удаления базы данных\n"

"6 --- для закрытия программы <БД о студентах>\n"

"----------------------------------------------------------\n"

"КОМАНДА: ";

\_\_asm {

push esi;

push eax;

lea esi, head2;

lea eax, str1;

push esi;

push eax;

call printf;

lea esi, number\_function\_menu;

lea eax, str2;

push esi;

push eax;

call scanf\_s;;

pop eax;

pop esi;

}

/\*printf("%s", head2);

scanf\_s("%d", &number\_function\_menu);\*/

switch (number\_function\_menu) {

case 1:

database\_creation\_function();

case 2:

display\_date\_base();

case 3:

database\_new\_element();

case 4:

sorting\_database\_items();

case 5:

fuction\_deleting\_database();

case 6:

return;

default:

printf("----------------------------------------------------------\nБыла введена некоректная команда, повторите попытку:\n----------------------------------------------------------\n");

recursive\_menu();

}

}

**main.cpp:**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include "Header.h"

int main() {

setlocale(LC\_ALL, "russian");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

printf("----------------------------------------------------------\n"

"ПРОГРАММА <БАЗА ДАННЫХ О СТУДЕНТАХ> ГОТОВА К РАБОТЕ\n"

"----------------------------------------------------------\n");

recursive\_menu();

return 0;

}

# **Приложение B**

Иерархическая структура программы:

