

Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

по курсу «Программирование»

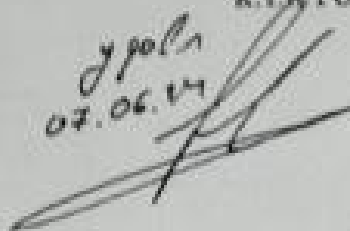
на тему: «База данных о студентах»

Выполнил:

Студент группы 23ВВВ3 Полиневский В. В.

Принят:

к.т.н. Голотенков Н. О.

уровень
07.06.24


Пенза, 2024

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 Митрохин В.В. Группа 23ВВВ

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

Исходные данные (технические требования) на проектирование

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. Экспериментальная часть

Отладка программы.

Срок выполнения проекта по разделам

В соответствии с графиком.

Дата выдачи задания "6" декабря

Дата защиты проекта "08" 06. 11

Руководитель Селезов Н.В.

Задание получил "08" декабря 2024г.

Студент Мелишевский Вадим Вадимович

Содержание

Введение	4
1. Постановка задачи	5
2. Выбор решения	5
5.1. Модули программы	5
5.2. Интерфейс программы	6
3. Описание разработки программы	7
4. Отладка и тестирование	8
1. Установка точки останова и запуск отладчика:	9
Переход по коду в отладчике с помощью пошаговых команд	10
2. Переход по коду в отладчике с помощью пошаговых команд	10
3. Шаг с обходом по коду для пропуска функций	11
4. Быстрое выполнение до точки в коде с помощью мыши	11
5. Описание программы	12
6. Руководство пользователя	18
Заключение	20
Список используемых источников	21
Приложение А	21
Приложение В	54

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.

1. 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.

2. Выбор решения

5.1. Модули программы

Программа состоит из трёх файлов: **mine.cpp** (файл с функций мейн), **source.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 ему не требуется в каждом модуле программы каждый раз подключать все требуемые библиотеки, достаточно подключить лишь «**header.h**».

5.2. Интерфейс программы

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

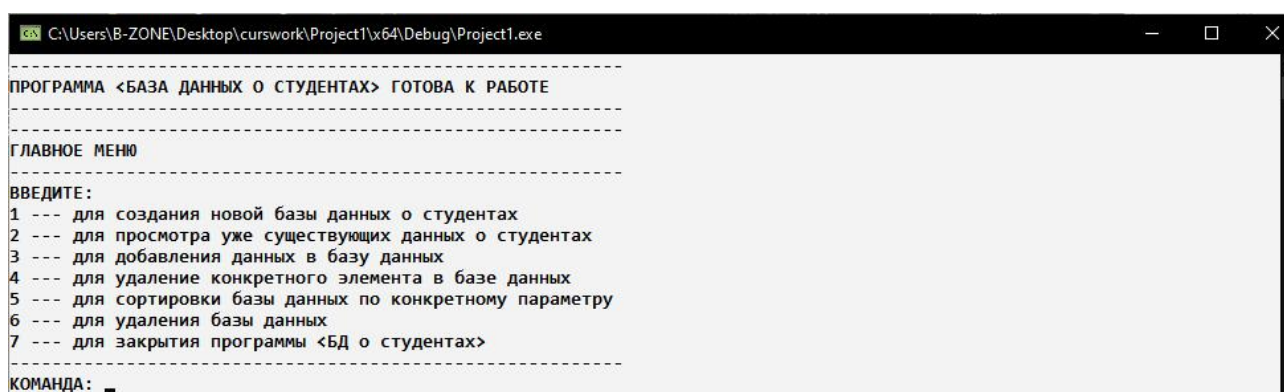


Рисунок 1

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 реализуется 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).

#	ФИО	Наз. Груп	№ в груп.	Дата Пост.	Дата изм.
0	Polinevskiy Vadim Vadimovich	23VV3	1	12.08.23	05/10/24
1	Aleckseev Victor Vadimovich	23VV4	4	13.08.23	05/10/24
2	Andreev Valentin Georgievich	23VVP3	22	12.05.07	05/05/20
3	Dubova Daria Anatolyevna	23VV3	15	12.05.07	03/05/21
4	Alexandrov Alexandr Wiktovorovich	23VN3	30	12.05.07	05/05/23
5	Debatov Maksim Afanasyevich	23VV3	19	12.05.07	03/05/22
6	Absolyamov Damir Casimovich	23VVP3	8	12.05.07	05/05/20
7	Polinevskiy Dmitriy Dmitrievich	23VV3	1	12.05.07	03/05/21

Рисунок 2

3. 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 соответствовало изучаемым за курс программирования на языке СИ навыкам.

В программе использованы языки Ассемблер, а также С 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 В)).

4. 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.

1. 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 распространённым 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 > запуск 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.

```

26     int number_function_menu = 0;
27     FILE* file_ptr;
28     char name_file[size_name_file];
29     printf(
30         "-----\n"
31         "Меню создания новой базы данных о студентах\n"
32         "-----\n");
33     printf("-----\n");
34     printf("Введите название файла. Учтите, что название файла может содержать только символы латинского и русского алфавита, а также знаки нижнего подчеркивания\n Название файла:");
35     scanf_s("%s", &name_file, sizeof(name_file));
36     printf("-----\n");
37     strcat(name_file, ".txt");
38     file_ptr = fopen(name_file, "a");
39     if (file_ptr != NULL) {
40         fclose(file_ptr);
41         printf(
42             "-----\n"

```

Рисунок 3

Переход по коду в отладчике с помощью пошаговых команд

2. 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.

```

26     int number_function_menu = 0;
27     FILE* file_ptr;
28     char name_file[size_name_file];
29     printf(
30         "-----\n"
31         "Меню создания новой базы данных о студентах\n"
32         "-----\n");
33     printf("-----\n");
34     printf("Введите название файла. Учтите, что название файла может содержать только символы латинского и русского алфавита, а также знаки нижнего подчеркивания\n Название файла:");
35     scanf_s("%s", &name_file, sizeof(name_file));
36     printf("-----\n");
37     strcat(name_file, ".txt");
38     file_ptr = fopen(name_file, "a");
39     if (file_ptr != NULL) {
40         fclose(file_ptr);
41         printf(

```

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.

3. 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.

4. 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).

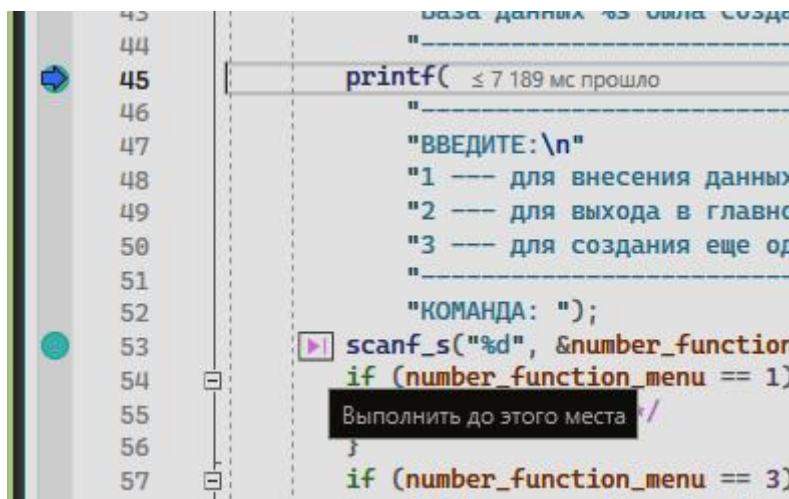


Рисунок 5

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.

5. 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 введенных 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()

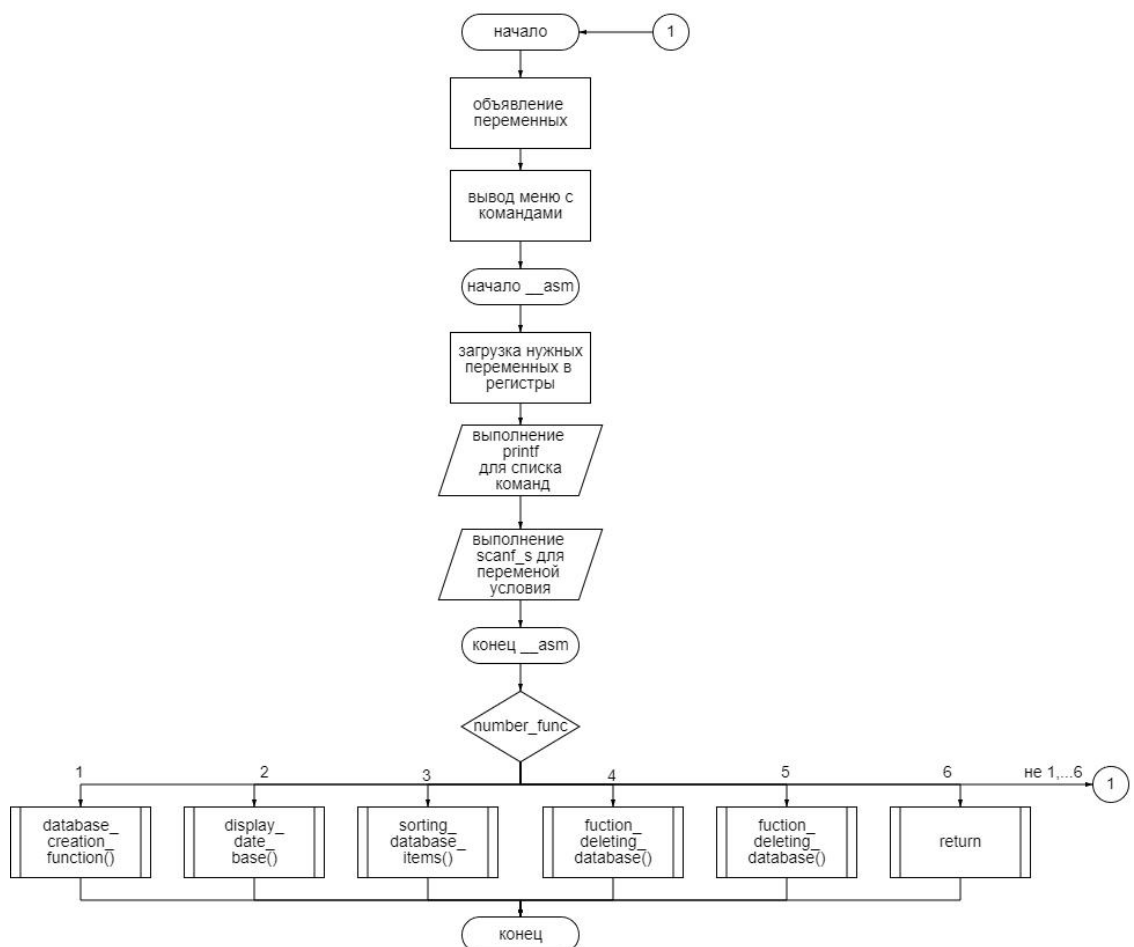


Рисунок 7. Схема recursive_menu()

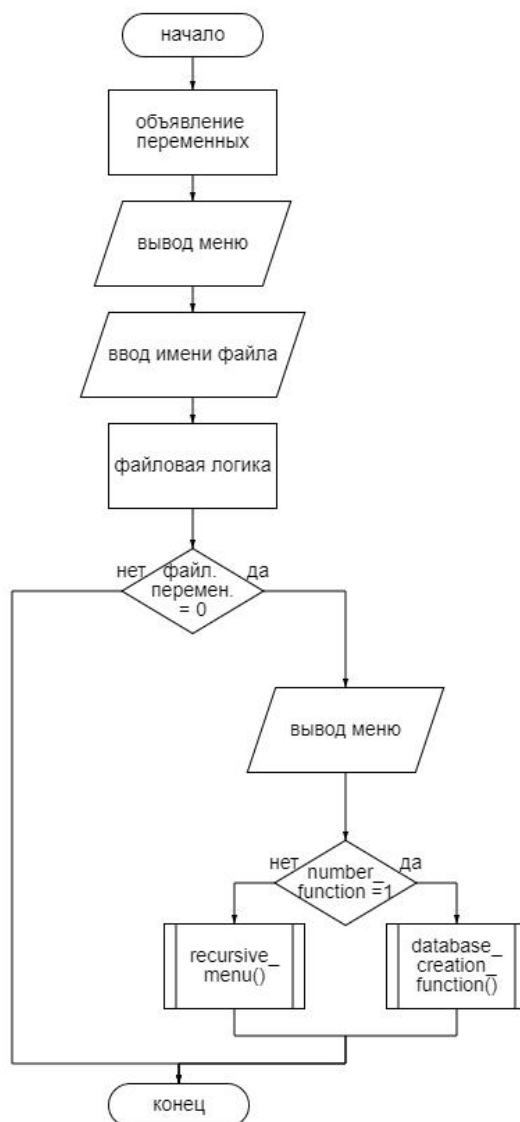


Рисунок 8.database_creation_function()

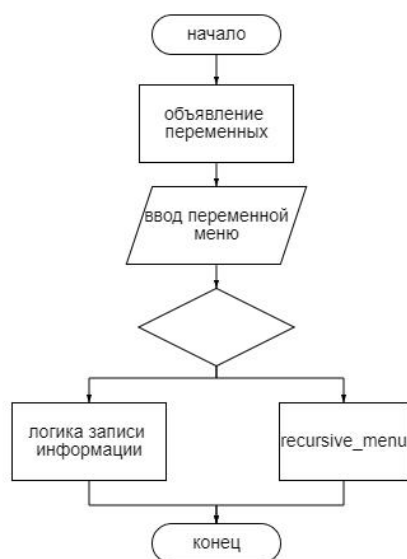


Рисунок 9.database_new_element()



Рисунок 10.sorting_database_items()

6. 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:

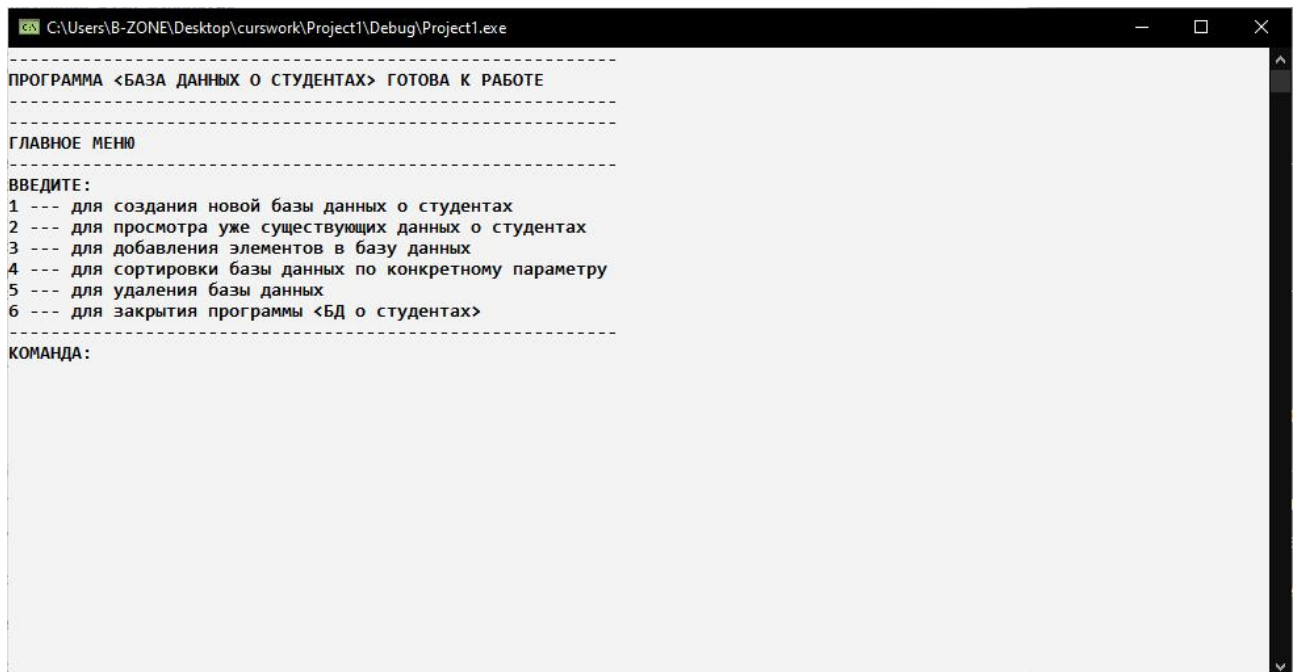


Рисунок 11. Главное меню

Меню создания новой БД.

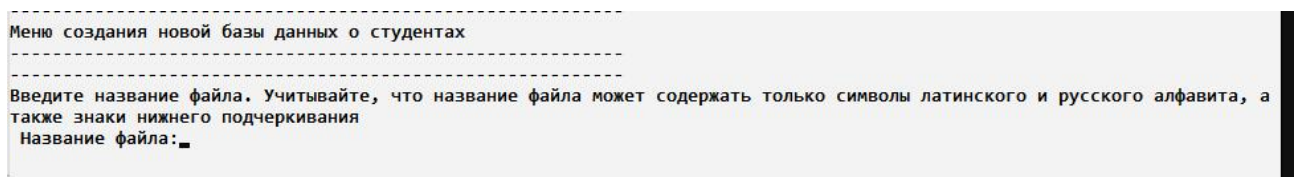


Рисунок 12. Меню создания новой БД

Меню Просмотра БД



Рисунок 13. Меню Просмотра БД

Меню Занесения данных в БД (Предварительный этап)

```
-----  
МЕНЮ ЗАНЕСЕНИЯ ДАННЫХ В БАЗУ ДАННЫХ  
-----
```

```
ВВЕДИТЕ:
```

```
1 --- для занесения данных
```

```
2 --- для выхода в главное меню  
-----
```

```
КОМАНДА:
```

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

Меню занесения данных (Основной этап)

```
-----  
МЕНЮ ЗАНЕСЕНИЯ ДАННЫХ В БАЗУ ДАННЫХ  
-----
```

```
ВВЕДИТЕ:
```

```
1 --- для занесения данных
```

```
2 --- для выхода в главное меню  
-----
```

```
КОМАНДА: 1  
-----
```

```
Введите название базы данных, в которую вы хотите занести данные:999  
-----
```

```
База данных существует и готова к записи
```

```
ВВЕДИТЕ:
```

```
1 -- если хотите продолжить
```

```
2 -- если хотите отменить операцию  
КОМАНДА:
```

```
1  
-----
```

```
Введите количество заносимых строк в базу данных. Учитывайте, что в каждую строку базы данных нужно занести: ФИО студент  
а, номер группы, дату поступления.
```

```
Кол-во строк:  
-----
```

Рисунок 15. Меню занесения данных (Основной этап)

Меню сортировки информации в БД (Предварительный этап)

```
-----  
Меню сортировки информации в БД  
-----
```

```
ВВЕДИТЕ:
```

```
1 --- для выполнения сортировки
```

```
2 --- для выхода в главное меню  
-----
```

```
КОМАНДА: █
```

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

Меню сортировки информации в БД (Основной этап 1):

```
[0]PolinevskiyVadimVadimovich|23VVV3|1|12.08.23|05/10/24|
[1]AleckseevVictorVadimovich|23VVV4|4|13.08.23|05/10/24|
[2]AndreevValentinGeorgievich|23VVP3|22|12.05.07|05/05/20|
[3]DubovaDariaAnatolyevna|23VVV3|15|12.05.07|03/05/21|
[4]AlexandrovAlexandrWiktovorovich|23VVN3|30|12.05.07|05/05/23|
[5]DebatovMaksimAfanasievich|23VVV3|19|12.05.07|03/05/22|
[6]AbsolyamovDamiirCasimovich|23VVP3|8|12.05.07|05/05/20|
[7]PolinevskiyDmitriyDmitrievich|23VVV3|1|12.05.07|03/05/21|
```

ВВЕДИТЕ:

- 1 --- для сортировки ФИО
- 2 --- для сортировки по дате поступления
- 3 --- для сортировки по названию группы
- 4 --- для сортировки по дате изменения
- 5 --- для выхода в главное меню

КОМАНДА:

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

Меню удаления база данных:

КОМАНДА: 5

Меню удаления БД

Введите название файла:

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(), stricmp(), 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 realloc(), 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 с.
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"
            -----\n"

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

            "-----\n", name_file);

        printf(

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

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

        "-----\n"
-----\n", file_ptr);

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

        "ВВЕДИТЕ:\n"

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

        "2 --- для выхода в главное меню\n"

        "-----\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"

                "-----\n"

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

                "-----\n"

                "-----\n");

        }

        fclose(file_ptr);

        file_ptr = fopen(file_name, "a");

        int number_of_repetitions;

        printf(

            "-----\n"

```

"Введите количество заносимых строк в базу данных. Учтите, 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 некорректная 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"
    -----\n"

    "Меню сортировки информации в БД\n"

    "-----\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 некорректная
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 сортировки 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"
-----\n"

            "Отсортированное значение:");
        for (int i = 0; i < size; i++) {

            //addSpacesBetweenCapitalLetters(dynamicArray[i]);

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

        }

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

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

        "Меню просмотра базы данных\n"

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

        "ВВЕДИТЕ:\n"

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

        "2 --- для выхода в главное меню\n"

        "-----\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("-----\n
        База данных <%s>не найдена\n
        -----\n", file_name);

        printf("ВВЕДИТЕ:\n"

            "1 --- для повтора попытки\n"

            "2 --- для выхода в главное меню\n"

            "-----\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 некорректная команда.
Повторите попытку\n");

        display_date_base();

    }

}

if (number_function_menu == 2) {

    recursive_menu();

}

else {

    printf("-----
\nБыла введена некорректная 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\n"
студента\n"

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

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

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

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

                "6   ---   если   хотите   закончить   ввод   и\n"
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 порядкового 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 некорректная 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("-----\n
        -----\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
    ----\n"

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

```

```

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

    recursive_menu();

    return 0;

}

```

Приложение В

Иерархическая структура программы:

