

Министерство науки и высшего образования Российской Федерации
ФГБОУ ВО «Пензенский государственный университет»
Факультет вычислительной техники
Кафедра «Вычислительная техника»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе
по курсу «Программирование»
на тему «Разработка программы сложной структуры методом
нисходящего программирования. База данных по столицам стран

22.06.2026
Генералова

Выполнил: ст. гр. 25BVB1

Дедёкин К.И.

Проверил: к.т.н. доцент

Генералова А.А.

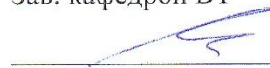
Пенза 2026

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Факультет Вычислительной техники

Кафедра "Вычислительная техника"

"УТВЕРЖДАЮ"

Зав. кафедрой ВТ



М.А. Митрохин

« 12 » июня 2026 г

ЗАДАНИЕ

на курсовое проектирование по курсу

Программирование

Студенту Дубинину Ксению Игоревичу Группа КБВВ 1

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

Исходные данные (технические требования) на проектирование

База данных школы госуниверситета.

Работа с данными: добавление данных, удаление, редактирование, сохранение.

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 "12" ноября 2026

Дата защиты проекта "12" ноября 2026

Руководитель П. Темурбаев А.А.

Задание получил "12" ноября 2026.

Студент Даркхон Кудрат Чороевич

Содержание

Введение	7
1 Постановка задачи.....	8
2 Выбор Решения	9
3 Описание разработки программы.....	11
4 Отладка и тестирование	12
5 Описание программы	12
6 Руководство пользователя.....	15
6.1 Заставка	15
6.2 Правила пользования	16
6.3 Просмотр всех записей.....	17
6.5 Удаление записи	18
6.6 Поиск.....	18
6.7 Сортировка	19
6.8 Сохранение и загрузка	20
Заключение	21
Список используемых материалов	22
Приложение А	23
Листинг программы	23
Файл main.c.....	23
Файл db.h.....	26
Файл db.c.....	26
Файл asm.h.....	33
Файл asm.c.....	33

Введение

Целью данной курсовой работы является разработка программы «Телефонный справочник» (База данных по столицам стран) на языке программирования Си с использованием ассемблерных вставок.

В процессе выполнения работы необходимо решить следующие задачи:

1. Разработать структуру данных для хранения информации о странах и их столицах.
2. Реализовать динамический массив для хранения записей с возможностью добавления, удаления и редактирования.
3. Обеспечить сохранение и загрузку данных в текстовом и бинарном форматах.
4. Разработать пользовательский интерфейс на основе текстового меню.
5. Реализовать ключевые алгоритмы (поиск и сортировку) с использованием ассемблерных вставок для оптимизации производительности.
6. Построить программу по многомодульному принципу для обеспечения удобства сопровождения и расширения.

Разрабатываемая программа должна функционировать в операционной системе Microsoft Windows, быть реализована в среде Microsoft Visual Studio и удовлетворять всем требованиям, предъявленным в задании на курсовую работу.

Программа предназначена для ведения справочной информации о столицах стран и демонстрации возможностей языка Си при работе с динамическими структурами данных, файловым вводом-выводом и низкоуровневым программированием.

В пояснительной записке представлены: описание структуры данных, схема алгоритмов, реализация программы на языке Си и Ассемблере, результаты тестирования и руководство пользователя. Графическая часть содержит схемы данных, ресурсов системы, работы системы, иерархической структуры, взаимодействия и алгоритма программы.

1 Постановка задачи

Целью данной работы является разработка консольной программы «Телефонный справочник» для ведения базы данных по столицам стран с использованием языков программирования Си и Ассемблер. В ходе выполнения работы необходимо реализовать следующий функционал:

1. Хранение записей в динамическом массиве структур с полями:

- страна;
- столица.

2. Реализация основных операций с базой данных:

- просмотр всех записей;
- добавление новой записи;
- удаление записи по номеру;
- поиск записи по названию страны;
- сортировка записей по названию страны.

3. Работа с файлами:

- сохранение базы данных в текстовый файл;
- загрузка базы данных из текстового файла;

- автоматическое сохранение и загрузка бинарного файла.
4. Использование ассемблерных вставок для оптимизации критических алгоритмов:
 - сравнение строк без учёта регистра.
 5. Реализация многомодульной архитектуры программы.
 6. Разработка текстового пользовательского интерфейса с меню.
 7. Наличие заставки при запуске программы.

Во время реализации проекта должны быть использованы сложные типы данных — структуры, массивы, файлы, алгоритмы обработки ввода и вывода и принципы модульного программирования.

2 Выбор Решения

В процессе разработки данной программы была выбрана среда разработки *Visual Studio 2022*. Она предоставляет все необходимые инструменты для комфортной и эффективной разработки, делая процесс написания кода, отладки и оптимизации максимально удобным.

Visual Studio также подходит для работы с языком ассемблера благодаря интеграции низкоуровневых возможностей с удобной средой разработки.

Главным преимуществом является поддержка встроенного ассемблера, что позволяет писать код в теле программы без необходимости использования дополнительных файлов. В данной работе ассемблерная вставка демонстрирует низкоуровневую реализацию сравнения строк.

Функция `asm_strcmp()` реализована на ассемблере и используется для сравнения строк без учёта регистра при поиске и сортировке.

Проект структурирован с использованием модулей, что позволяет логически разделить код на функциональные блоки, которые легко сопровождать. Схема данных приведена на рисунке 1.

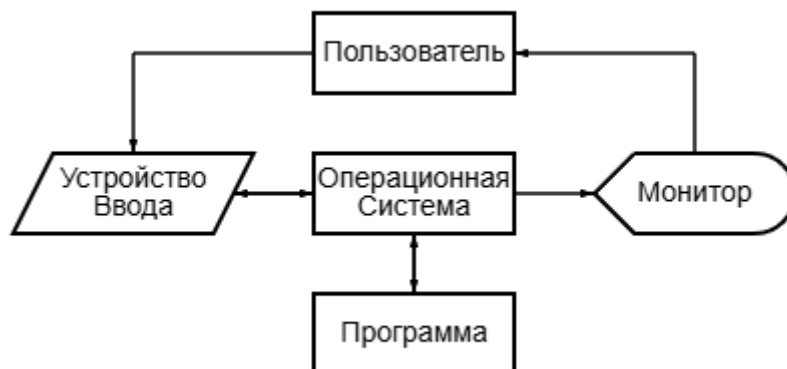


Рисунок 1 - Схема данных

Данные сохраняются в текстовый файл `phones.txt` для просмотра пользователем.

3 Описание разработки программы

Для реализации был выбран нисходящий метод разработки, потому что он позволяет сначала определить общую структуру программы, а затем постепенно детализировать её компоненты.

При запуске программы на экран выводится приветственная заставка с названием программы. Затем отображается главное меню с 8 пунктами:

1. Показать все записи;
2. Добавить запись;
3. Удалить запись;
4. Поиск (Ассемблер);
5. Сортировка (Ассемблер);
6. Сохранить в файл;
7. Загрузить из файла;
8. Выход.

Программа разделена на 3 модуля:

1. `main.c` — главный модуль, управляющий интерфейсом;
2. `db.c / db.h` — модуль работы с базой данных;
3. `asm.c / asm.h` — модуль с ассемблерными вставками.

Модуль `main.c` — главный файл программы, который инициализирует и запускает приложение, отображает заставку и меню, обрабатывает выбор пользователя.

Модуль `db.c` отвечает за работу с данными: добавление, удаление, поиск, сортировку, сохранение и загрузку записей.

Модуль `asm.c` содержит ассемблерную функцию сравнения строк.

4 Отладка и тестирование

В качестве среды разработки была выбрана программа *Visual Studio 2022*. Она оптимальна для разработки на языке программирования Си благодаря мощному функционалу и удобству работы.

Пошаговое выполнение дает возможность отслеживать, как программа инициализирует базу данных, добавляет записи, выполняет поиск и сортировку.

Точки останова позволяют исследовать состояние программы, с тем чтобы определить, как ведёт себя приложение.

Тестирование проходило в рабочем процессе, в процессе разработки и после завершения написания программы.

В ходе тестирования были проверены следующие сценарии:

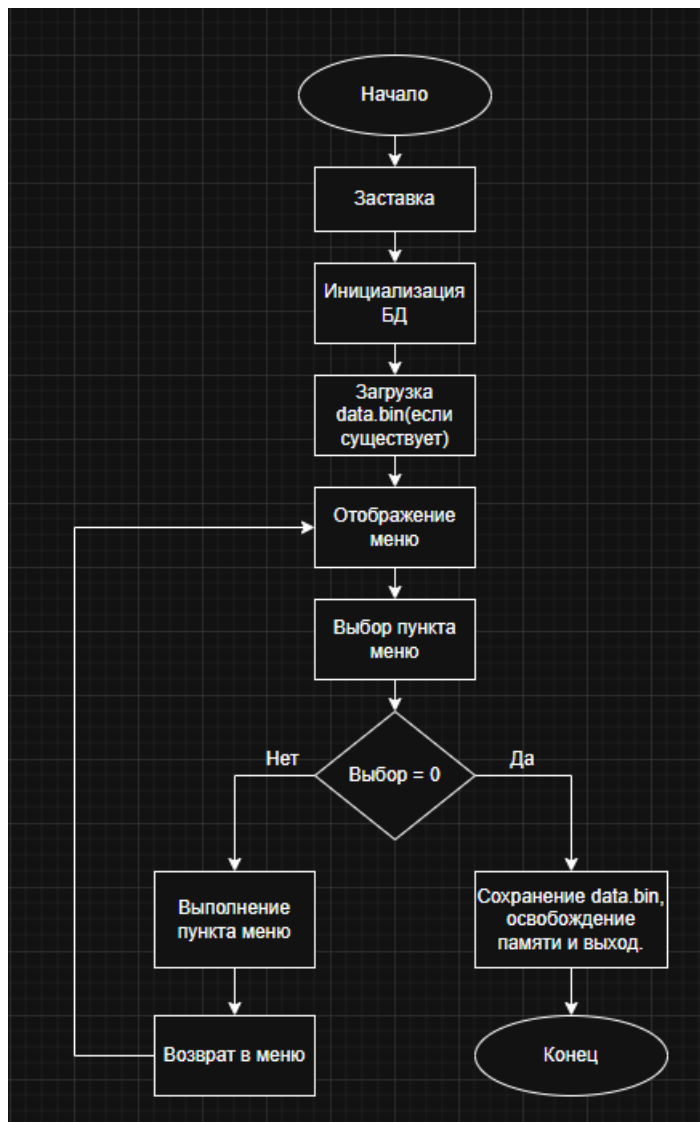
1. Добавление записи с корректными данными — запись успешно добавляется.
2. Добавление записи с пустыми полями — программа корректно обрабатывает ввод.
3. Поиск существующей страны — запись найдена.
4. Поиск несуществующей страны — сообщение об отсутствии совпадений.
5. Сортировка записей — записи отсортированы в алфавитном порядке.
6. Сохранение и загрузка из текстового файла — данные сохранены и загружены корректно.
7. Сохранение и загрузка из бинарного файла — данные сохранены и загружены корректно.
8. Удаление записи — запись удалена, остальные сдвинуты.

Все тесты пройдены успешно. Программа работает стабильно и корректно обрабатывает все сценарии.

5 Описание программы

5.1 Схема функции main

На рисунке 2 представлена схема функции main.



5.2 Описание модулей

Программа состоит из 3 основных модулей, которые упрощают процесс разработки и отладки, улучшают читаемость и ускоряют сборку проекта.

5.2.1 main.c — главный управляющий модуль

Файл main.c является точкой входа в программу, связывающей все компоненты:

- `main()` — функция, в которой вызывается заставка, инициализируется база данных, загружаются данные из бинарного файла, отображается меню и обрабатывается выбор пользователя, а при выходе данные сохраняются и память освобождается.
- `splash()` — функция, которая выводит на экран заставку с названием программы.
- `menu()` — функция, которая отображает главное меню.

- `process()` — функция-диспетчер, которая вызывает соответствующую функцию в зависимости от выбора пользователя.

5.2.2 db.h — заголовочный файл базы данных

В файле `db.h` объявляются структуры данных и прототипы функций:

- `Record` — структура, содержащая поля `country` (страна) и `capital` (столица).
- `Database` — структура, содержащая указатель на массив записей, текущее количество записей и ёмкость массива.
- Прототипы функций для работы с базой данных.

5.2.3 db.c — модуль работы с данными

Файл `db.c` содержит реализацию всех функций для работы с базой данных:

- `show_all()` — выводит все записи в виде таблицы.
- `add_record()` — добавляет новую запись, при необходимости расширяет массив.
- `delete_record()` — удаляет запись по номеру.
- `search_asm()` — выполняет поиск по названию страны с использованием ассемблерной функции сравнения.
- `sort_asm()` — выполняет сортировку записей по названию страны с использованием ассемблерной функции сравнения.
- `save_text()` — сохраняет базу данных в текстовый файл.
- `load_text()` — загружает базу данных из текстового файла.
- `save_binary()` — сохраняет базу данных в бинарный файл.
- `load_binary()` — загружает базу данных из бинарного файла.

5.2.4 asm.h — заголовочный файл ассемблерных функций

В файле `asm.h` объявляется прототип ассемблерной функции:

- `asm_strcmp()` — функция сравнения строк без учёта регистра.

5.2.5 asm.c — модуль ассемблерных вставок

Файл `asm.c` содержит реализацию ассемблерной функции:

- `asm_strcmp()` — функция сравнивает две строки без учёта регистра. Возвращает 0, если строки равны, 1, если первая строка больше второй, и -1, если первая строка меньше второй. Реализация использует регистры

ESI и EDI для хранения адресов строк, команды LODSB для загрузки байтов и CMP для сравнения. При сравнении символы приводятся к верхнему регистру.

Ассемблерная вставка находится в функции `asm_strncmp()`. Она сравнивает две строки без учёта регистра, используя регистры ESI, EDI и команды LODSB, CMP:

```
__asm {  
    push esi  
    push edi  
  
    mov esi, s1  
    mov edi, s2  
  
loop_start:  
    mov al, [esi]  
    mov ah, [edi]  
    // ... сравнение символов с приведением к верхнему регистру ...  
}
```

6 Руководство пользователя

Программа предназначена для ведения справочной информации о столицах стран. Она интуитивно понятна в управлении.

6.1 Заставка

На рисунке 3 представлена игровая заставка, которая появляется при запуске приложения.

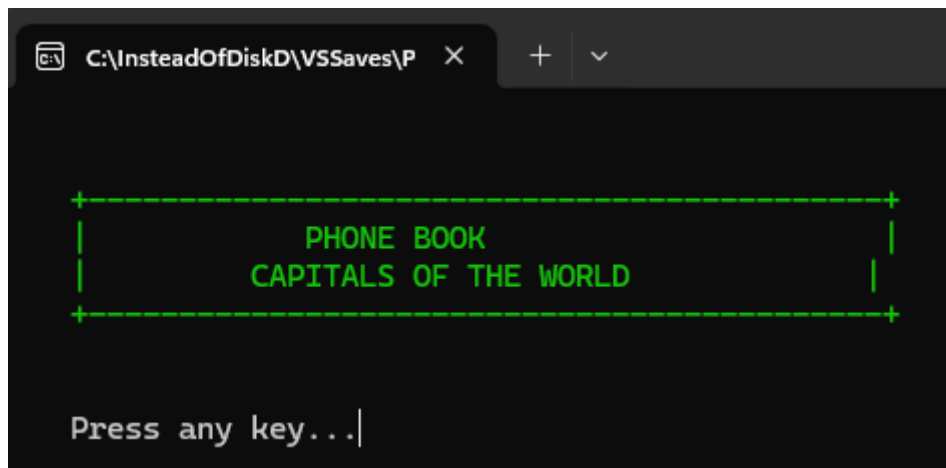


Рисунок 3 - Заставка

6.2 Правила пользования

После заставки появляется главное меню (рисунок 4), в котором можно выбрать одно из 7 действий или выйти из программы.

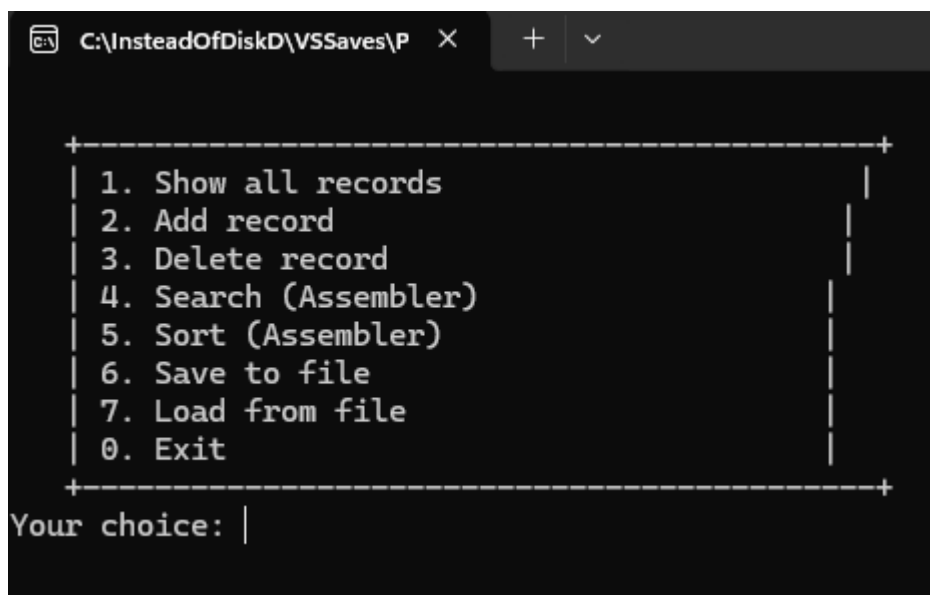
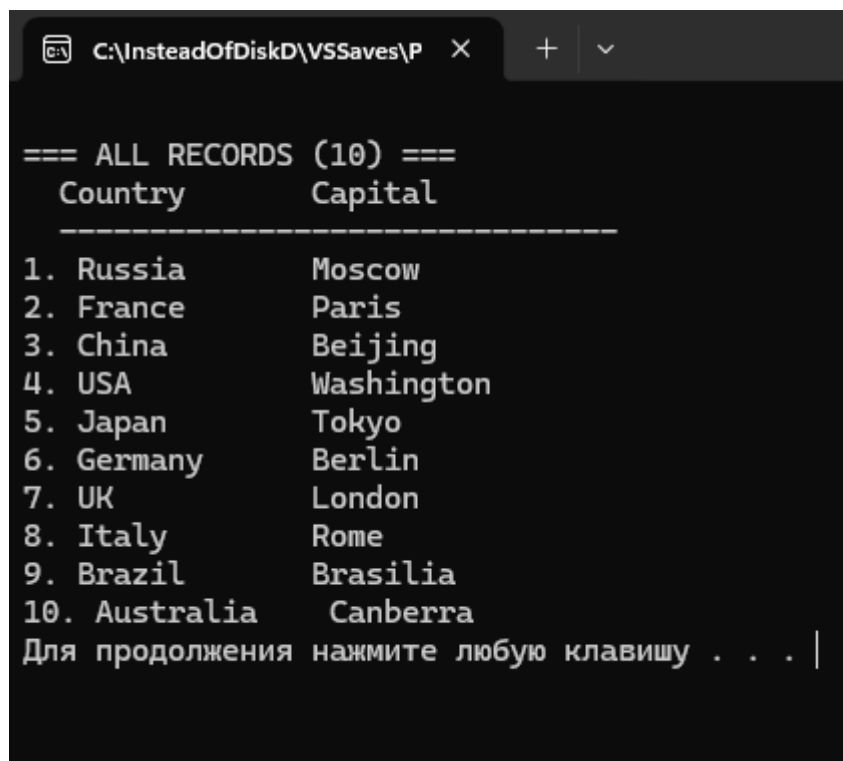


Рисунок 4 – Меню программы

6.3 Просмотр всех записей

При выборе пункта 1 на экран выводится таблица со всеми записями (рисунок 5).



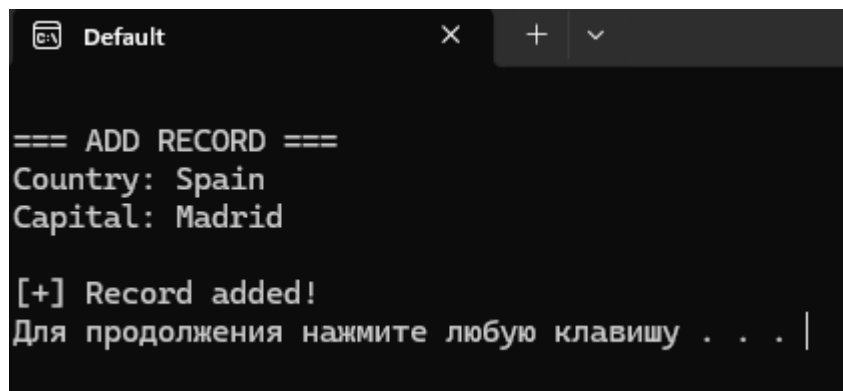
```
C:\InsteadOfDiskD\VSSaves\P x + v

=== ALL RECORDS (10) ===
Country      Capital
-----
1. Russia    Moscow
2. France    Paris
3. China     Beijing
4. USA       Washington
5. Japan     Tokyo
6. Germany   Berlin
7. UK        London
8. Italy      Rome
9. Brazil    Brasilia
10. Australia Canberra
Для продолжения нажмите любую клавишу . . . |
```

Рисунок 5 – Просмотр записей

6.4 Добавление записи

При выборе пункта 2 программа последовательно запрашивает ввод данных: страну и столицу (рисунок 6).



```
Default x + v

=== ADD RECORD ===
Country: Spain
Capital: Madrid

[+] Record added!
Для продолжения нажмите любую клавишу . . . |
```

Рисунок 6 – Добавление записи

6.5 Удаление записи

При выборе пункта 3 программа сначала отображает все записи, затем запрашивает номер записи для удаления (рисунок 7).



```
Default x + v

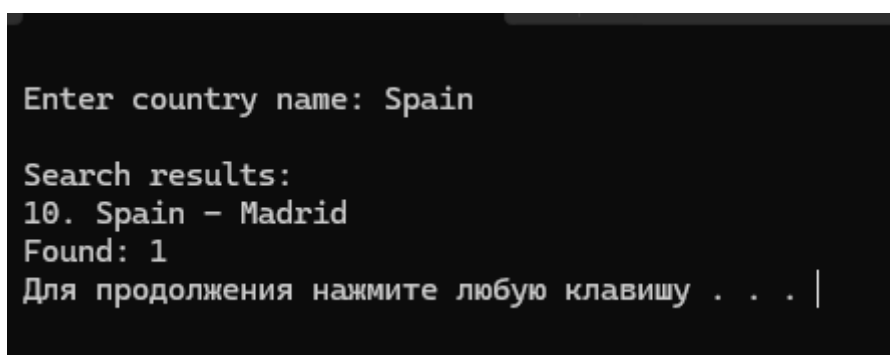
=== ALL RECORDS (11) ===
Country      Capital
-----
1. Russia    Moscow
2. France    Paris
3. China     Beijing
4. USA       Washington
5. Japan     Tokyo
6. Germany   Berlin
7. UK        London
8. Italy      Rome
9. Brazil    Brasilia
10. Australia Canberra
11. Spain    Madrid

Record number to delete (0-cancel): 4
[+] Record deleted!
Для продолжения нажмите любую клавишу . . . |
```

Рисунок 7 – Удаление записи

6.6 Поиск

При выборе пункта 4 программа запрашивает название страны для поиска (рисунок 8). Поиск выполняется с использованием ассемблерной функции, которая сравнивает строки без учёта регистра.



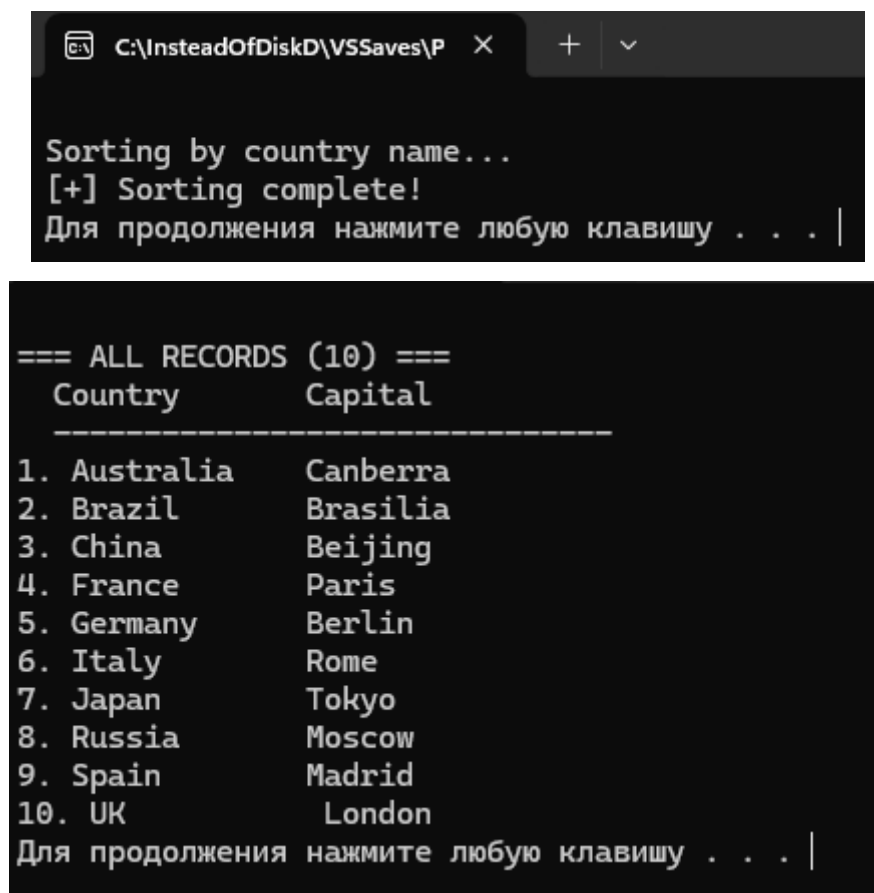
```
Enter country name: Spain

Search results:
10. Spain - Madrid
Found: 1
Для продолжения нажмите любую клавишу . . . |
```

Рисунок 8 - Поиск

6.7 Сортировка

При выборе пункта 5 программа выполняет сортировку всех записей по названию страны в алфавитном порядке с использованием ассемблерной функции сравнения (рисунок 9-10).



```
C:\InsteadOfDiskD\VSSaves\P X + v

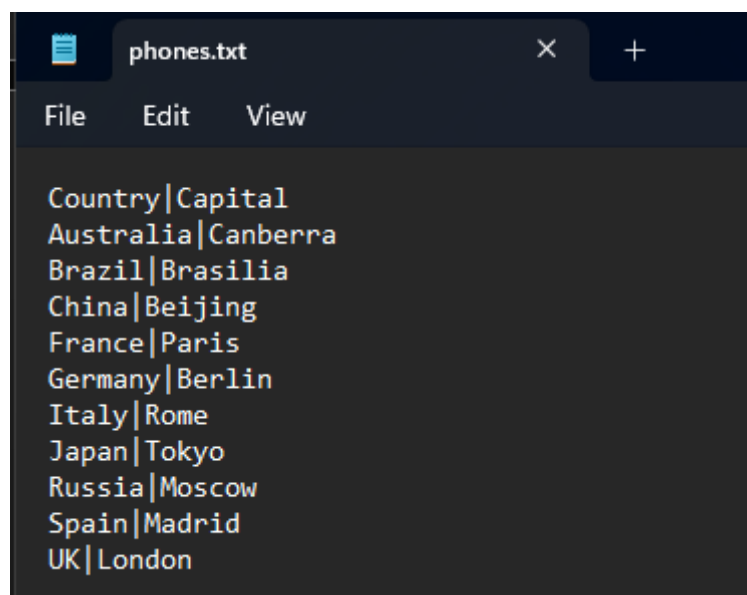
Sorting by country name...
[+] Sorting complete!
Для продолжения нажмите любую клавишу . . . |

=== ALL RECORDS (10) ===
Country      Capital
-----
1. Australia  Canberra
2. Brazil     Brasilia
3. China      Beijing
4. France     Paris
5. Germany    Berlin
6. Italy       Rome
7. Japan      Tokyo
8. Russia     Moscow
9. Spain      Madrid
10. UK        London
Для продолжения нажмите любую клавишу . . . |
```

Рисунок 9-102 - Сортировка

6.8 Сохранение и загрузка

Пункты 6 и 7 позволяют сохранить базу данных в текстовый файл phones.txt и загрузить её из этого файла.

A screenshot of a text editor window with a dark theme. The title bar shows a tab labeled 'phones.txt' with a close button (X) and a new file button (+). The menu bar contains 'File', 'Edit', and 'View'. The text area displays a list of countries and their capitals, separated by a vertical bar. The data is as follows:

Country	Capital
Australia	Canberra
Brazil	Brasilia
China	Beijing
France	Paris
Germany	Berlin
Italy	Rome
Japan	Tokyo
Russia	Moscow
Spain	Madrid
UK	London

Заключение

В ходе выполнения курсовой работы была разработана программа «Телефонный справочник» (База данных по столицам стран), реализованная на языке программирования Си с использованием ассемблерных вставок.

Основные достижения и реализованные функции:

1. Разработана структура данных для хранения информации о странах и их столицах.
2. Реализован динамический массив для хранения записей с возможностью добавления, удаления и редактирования.
3. Обеспечено сохранение и загрузка данных в текстовом и бинарном форматах.
4. Разработан пользовательский интерфейс на основе текстового меню.
5. Реализованы ключевые алгоритмы (поиск и сортировка) с использованием ассемблерных вставок для оптимизации производительности.
6. Программа построена по многомодульному принципу.

Выполненная работа продемонстрировала эффективность языка Си для разработки приложений с динамическими структурами данных. Программа успешно реализует все основные функции справочника, демонстрируя при этом высокую производительность и надёжность. Приобретённые в ходе работы навыки имеют важное значение для дальнейшего профессионального роста в области системного программирования.

Список используемых материалов

1. CppReference — <https://en.cppreference.com/w/c>
2. Microsoft C/C++ Documentation — <https://docs.microsoft.com/cpp/c-language/>
3. GeeksforGeeks C Programming — <https://www.geeksforgeeks.org/c-programming-language/>
4. Stack Overflow — <https://stackoverflow.com/questions/tagged/c>
5. GCC Inline Assembly HOWTO — <https://www.ibiblio.org/gferg/ldp/GCC-Inline-Assembly-HOWTO.html>
6. C Programming Subreddit — https://reddit.com/r/C_Programming

Приложение А

Листинг программы

Файл main.c

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#include <windows.h>
#include <conio.h>
#include "db.h"
#include "asm.h"

Database db = { NULL, 0, 0 };

void splash(void);
void menu(void);
void process(int choice);

int main(void) {
    splash();

    db.items = (Record*)malloc(10 * sizeof(Record));
    db.count = 0;
    db.capacity = 10;

    load_binary("data.bin");

    int choice;
    do {
        menu();
        printf("Your choice: ");
        scanf_s("%d", &choice);
        system("cls");
        if (choice >= 1 && choice <= 7) process(choice);
    } while (choice != 0);
}
```

```

    save_binary("data.bin");
    free(db.items);

    printf("\nGoodbye!\n");
    return 0;
}

void splash(void) {
    system("cls");
    HANDLE h = GetStdHandle(STD_OUTPUT_HANDLE);
    SetConsoleTextAttribute(h, 10);

    printf("\n\n");
    printf("  +-----+ \n");
    printf("  |                PHONE BOOK                | \n");
    printf("  |                CAPITALS OF THE WORLD        | \n");
    printf("  +-----+ \n");

    SetConsoleTextAttribute(h, 7);
    printf("\n\n  Press any key...");
    _getch();
    system("cls");
}

void menu(void) {
    printf("\n");
    printf("  +-----+ \n");
    printf("  | 1. Show all records                        | \n");
    printf("  | 2. Add record                             | \n");
    printf("  | 3. Delete record                          | \n");
    printf("  | 4. Search (Assembler)                     | \n");
    printf("  | 5. Sort (Assembler)                       | \n");
    printf("  | 6. Save to file                           | \n");
    printf("  | 7. Load from file                         | \n");
    printf("  | 0. Exit                                   | \n");
    printf("  +-----+ \n");
}

```

```
}

void process(int choice) {
    switch (choice) {
        case 1: show_all(); break;
        case 2: add_record(); break;
        case 3: delete_record(); break;
        case 4: search_asm(); break;
        case 5: sort_asm(); break;
        case 6: save_text("phones.txt"); break;
        case 7: load_text("phones.txt"); break;
    }
    system("pause");
    system("cls");
}
```

Файл db.h

```
#ifndef DB_H
#define DB_H

typedef struct {
    char country[30];
    char capital[30];
} Record;

typedef struct {
    Record* items;
    int count;
    int capacity;
} Database;

extern Database db;

void show_all(void);
void add_record(void);
void delete_record(void);
void search_asm(void);
void sort_asm(void);
void save_text(const char* fn);
void load_text(const char* fn);
void save_binary(const char* fn);
void load_binary(const char* fn);

#endif
```

Файл db.c

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

```

#include "db.h"
#include "asm.h"

extern Database db;

void show_all(void) {
    if (db.count == 0) {
        printf("\nDATABASE IS EMPTY\n");
        return;
    }

    printf("\n=== ALL RECORDS (%d) ===\n", db.count);
    printf("  Country          Capital\n");
    printf("  -----\n");

    for (int i = 0; i < db.count; i++) {
        Record* r = &db.items[i];
        printf("%d. %-12s %-12s\n",
            i + 1, r->country, r->capital);
    }
}

void add_record(void) {
    if (db.count >= db.capacity) {
        db.items = (Record*)realloc(db.items, db.capacity * 2 *
sizeof(Record));
        db.capacity *= 2;
    }

    Record* r = &db.items[db.count];

    printf("\n=== ADD RECORD ===\n");
    getchar();
}

```

```

printf("Country: ");
gets_s(r->country, sizeof(r->country));

printf("Capital: ");
gets_s(r->capital, sizeof(r->capital));

db.count++;
printf("\n[+] Record added!\n");
}

void delete_record(void) {
    if (db.count == 0) {
        printf("\nDATABASE IS EMPTY\n");
        return;
    }

    show_all();
    printf("\nRecord number to delete (0-cancel): ");
    int n;
    scanf_s("%d", &n);

    if (n < 1 || n > db.count) {
        printf("[!] Cancelled\n");
        return;
    }

    for (int i = n - 1; i < db.count - 1; i++) {
        db.items[i] = db.items[i + 1];
    }
    db.count--;
    printf("[+] Record deleted!\n");
}

```

```

void search_asm(void) {
    if (db.count == 0) {
        printf("\nDATABASE IS EMPTY\n");
        return;
    }

    char s[30];
    getchar();
    printf("\nEnter country name: ");
    gets_s(s, sizeof(s));

    printf("\nSearch results:\n");
    int found = 0;

    for (int i = 0; i < db.count; i++) {
        if (asm_strcmp(db.items[i].country, s) == 0) {
            printf("%d. %s - %s\n",
                i + 1, db.items[i].country, db.items[i].capital);
            found++;
        }
    }

    if (found == 0) {
        printf("No matches found.\n");
    }
    else {
        printf("Found: %d\n", found);
    }
}

void sort_asm(void) {
    if (db.count < 2) {

```

```

        printf("\nNot enough records to sort\n");
        return;
    }

    printf("\nSorting by country name...\n");

    int n = db.count;
    Record temp;

    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (asm_strcmp(db.items[j].country, db.items[j +
1].country) > 0) {
                temp = db.items[j];
                db.items[j] = db.items[j + 1];
                db.items[j + 1] = temp;
            }
        }
    }

    printf("[+] Sorting complete!\n");
}

void save_text(const char* fn) {
    FILE* f;
    fopen_s(&f, fn, "w");
    if (!f) {
        printf("Error opening file!\n");
        return;
    }

    fprintf(f, "Country|Capital\n");
    for (int i = 0; i < db.count; i++) {

```

```

        fprintf(f, "%s|%s\n",
                db.items[i].country,
                db.items[i].capital);
    }

    fclose(f);
    printf("[+] Saved to: %s\n", fn);
}

void load_text(const char* fn) {
    FILE* f;
    fopen_s(&f, fn, "r");
    if (!f) {
        printf("File not found!\n");
        return;
    }

    char buf[200];
    fgets(buf, sizeof(buf), f);

    int count = 0;
    Record temp[100];
    char* context = NULL;

    while (fgets(buf, sizeof(buf), f) != NULL) {
        char* token = strtok_s(buf, "|\\n", &context);
        if (!token) continue;
        strcpy_s(temp[count].country, sizeof(temp[count].country),
token);

        token = strtok_s(NULL, "|\\n", &context);
        if (!token) continue;
    }
}

```

```

        strcpy_s(temp[count].capital, sizeof(temp[count].capital),
token);

        count++;
        if (count >= 100) break;
    }

    fclose(f);

    if (count == 0) {
        printf("File is empty!\n");
        return;
    }

    free(db.items);
    db.items = (Record*)malloc(count * sizeof(Record));
    db.count = count;
    db.capacity = count;

    for (int i = 0; i < count; i++) {
        db.items[i] = temp[i];
    }

    printf("[+] Loaded %d records\n", count);
}

void save_binary(const char* fn) {
    FILE* f;
    fopen_s(&f, fn, "wb");
    if (!f) return;

    fwrite(&db.count, sizeof(int), 1, f);
    fwrite(db.items, sizeof(Record), db.count, f);
}

```

```

        fclose(f);
    }

void load_binary(const char* fn) {
    FILE* f;
    fopen_s(&f, fn, "rb");
    if (!f) return;

    int count;
    fread(&count, sizeof(int), 1, f);

    if (count == 0) {
        fclose(f);
        return;
    }

    free(db.items);
    db.items = (Record*)malloc(count * sizeof(Record));
    db.count = count;
    db.capacity = count;
    fread(db.items, sizeof(Record), count, f);

    fclose(f);
}

```

Файл asm.h

```

#ifndef ASM_H
#define ASM_H

int asm_strcmp(const char* s1, const char* s2);

#endif

```

Файл asm.c

```

#include "asm.h"

int asm_strncmp(const char* s1, const char* s2) {
    int result = 0;

    __asm {
        push esi
        push edi

        mov esi, s1
        mov edi, s2

loop_start:
        mov al, [esi]
        mov ah, [edi]

        cmp al, 0
        je check_end
        cmp ah, 0
        je s1_greater

        cmp al, 'a'
        jb skip1
        cmp al, 'z'
        ja skip1
        sub al, 32

skip1:
        cmp ah, 'a'
        jb skip2
        cmp ah, 'z'
        ja skip2
        sub ah, 32
    }

    if (result < 0)
        result = -1;
    else if (result > 0)
        result = 1;
    else
        result = 0;

    return result;
}

```

```

skip2:
    cmp al, ah
    jg s1_greater
    jl s1_less

    inc esi
    inc edi
    jmp loop_start

check_end:
    cmp ah, 0
    je equal
    jmp s1_less

s1_greater:
    mov result, 1
    jmp done

s1_less:
    mov result, -1
    jmp done

equal:
    mov result, 0

done:
    pop edi
    pop esi
}

return result;
}

```