

Министерство образования Российской Федерации
Пензенский государственный университет
Кафедра «Вычислительная техника»

Пояснительная записка

К курсовой работе
По курсу «Программирование»
На тему «База данных»

Выполнила:
студентка группы 21BB1
Ананьева А.В.

Принял:
к.т.н., Токарев А. Н.

201
7.06.22



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 Анисьевой Ане Витальевне Группа ИВВ1

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

Исходные данные (технические требования) на проектирование

Наименование программы: База данных Столиц стран
Задача: Программа должна выполнять следующие функции:
- Создание новой базы данных. Дополнение записей.
- Удаление записей. Редактирование записей.
- Поиск и сортировка записей. Фильтрация записей по определенному критерию.
- Сохранение текущей базы данных в файл. Загрузка сохраненной базы данных из файла.

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 “22” февраль

Дата защиты проекта “ ”

Руководитель Токарев И.И.

Задание получил “22” февраль 2022 г.

Студент Ску (Ананьева А.В.)

Содержание

Введение	5
Постановка задачи	6
Выбор решения	7
Описание разработки программы	8
Отладка и тестирование	9
Описание программы	9
Руководство пользователя	16
Заключение	21
Приложение А	23
Приложение В	37
Приложение С	38

Введение

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

База данных представляет собой упорядоченную, структурированную совокупность данных, подобранных таким образом, что эти данные могут легко и быстро обрабатываться на ЭВМ.

В наше время базы данных являются неотъемлемой частью любой организации, любого предприятия. Динамические сайты используют базу данных, чтобы хранить данные пользователя, как и стриминговые сервисы с подписочной моделью. Банки хранят в огромных базах данных данные пользователей.

Данная программа является информационно-поисковой системой и позволяет работать с простейшей базой данных столиц стран.

Постановка задачи

Необходимо разработать программу для работы с базой данных столиц стран. Программа должна иметь интуитивно понятный интерфейс для пользователя. Необходимо определить названия пунктов меню, построить алгоритмы работы с массивами структур для фильтрации и сортировки записей по определенным критериям, изучить функции для работы с файлами, чтобы сохранять базу данных в файл и загружать из файла.

Многомодульность программы. Программа должна быть поделена на логические модули. Это упростит поиск ошибок при отладке и тестировании программы, а также позволит легко расширять функционал программы.

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

Режим работы видеосистемы – текстовый/графический. Для ввода данных и ориентирования в меню используется клавиатура.

Заставка дает пользователю общую информацию о программе.

Выбор решения

При запуске программы появляется заставка. После ее закрытия уже открывается главное меню, разделенное на пункты:

1. Создать новый список
 - 1.1. Загрузить из файла. Программа открывает файл по заданному адресу и читает элементы списка из него.
 - 1.2. Ввести вручную. Пользователь вводит число вводимых элементов, потом последовательно заполняет список.
2. Добавить запись. Добавляет новую запись в конец списка с данными, которые ввел пользователь.
3. Просмотр списка. Позволяет переключаться между элементами списка и удалять их по желанию пользователя.
 - 3.1. Вверх
 - 3.2. Вниз
 - 3.3. Удалить
 - 3.4. Выйти
4. Очистить список. Полное удаление списка.
5. Выборка. Меню с функциями фильтрации элементов в списке, а также функциями сортировки. Создает копию списка, которая будет подвергнута изменениям.
 - 5.1. Поиск по названию. Выборка всех элементов с подходящим названием.
 - 5.2. Сортировка по названию. Упорядочивание списка.
 - 5.3. Сортировка по численности. Упорядочивание списка.
 - 5.4. Отбор по численности. Выборка элементов подходящих по вводимому условию.
 - 5.4.1. Больше или равно.
 - 5.4.2. Меньше чем.
 - 5.5. Восстановить данные. Восстанавливает список в начальное состояние до каких-либо изменений.

5.6. Сохранить изменения. Сохраняет изменения в основной список

5.7. Выход

6. Сохранить данные в файл. Сохранение списка в файл в формате, в котором программа сможет его потом считать.

7. Выход из программы.

Описание разработки программы

Разработанная программа состоит из следующих модулей:

1. main_data_base.cpp – основной файл программы, запускает все остальные.
2. add_last.cpp – добавление элемента в конец списка.
3. create_elem.cpp – создание элемента списка.
4. read_base.cpp – Чтение односвязного списка из файла.
5. write_base.cpp – Заполнение односвязного списка вручную.
6. del_elems.cpp – Удаляет помеченные элементы из списка.
7. clear_list.cpp – Полностью очищает список.
8. copy_list.cpp – Возвращает копию списка.
9. filter_list_by_population.cpp – Возвращает список с выполненным отбором.
10. search_by_name.cpp - Возвращает список с выполненным отбором.
11. sort_by_name.cpp – Сортирует список по именам городов.
12. sort_by_population.cpp – Сортирует список по их численности.
13. save_to_file.cpp – Сохраняет файл в папку ресурсов программы.
14. first_screen.cpp – Печатает заставку на экран пользователя.

15. menu.cpp – Организует работу всех функций.

В ходе реализации сначала создавались основные компоненты программы, а затем дорабатывались ее мелкие детали. Каждый модуль тестируется сразу после его реализации, что помогает в создании более структурированной программы.

Отладка и тестирование

В качестве среды разработки была выбрана программа Microsoft Visual Studio 2019. Программа обладает всеми средствами необходимыми при разработке и отладке программы. Для отладки использовались несколько возможностей Visual Studio: точка останова, трассировка, анализ содержимого переменных.

Тестирование проводилось во время разработки и также после завершения разработки. В ходе нее было выявлено огромное количество проблем, связанных с работой с файлами, работой с памятью, работой с полями структуры.

Описание программы

При запуске программы происходит вывод заставки. После чего перед пользователем возможность навигации в меню. Описание состояний программы приведено в таблицах ниже:

Таблица 1

Клавиши, вызывающее событие	Действие пользователя	Действие программы
1, Enter	Выбран пункт «Создать новый список»	Открывается новое меню см. Таблица 2
2, Enter	Выбран пункт «Добавить запись»	Запускается диалог добавления записи

3, Enter	Выбран пункт «Просмотр списка»	Открывается новое меню
4, Enter	Выбран пункт «Очистить список»	Удаление всех элементов списка
5, Enter	Выбран пункт «Выборка»	Открывается новое меню см. Таблица 4
6, Enter	Выбран пункт «Сохранить данные в файл»	Список сохраняется в файл ресурсов
7, Enter	Выбран пункт «Выход из программы»	Выполняется закрытие программы

Таблица 2

Клавиши, вызывающее событие	Действие пользователя	Действие программы
1, Enter	Выбран пункт «Загрузить из файла»	Чтение списка данных из файла
2, Enter	Выбран пункт «Ввести»	Запускается диалог добавления записей

Таблица 3

Клавиши, вызывающее событие	Действие пользователя	Действие программы
1, Enter	Выбран пункт «Вверх»	Перемещение селектора списка вверх
2, Enter	Выбран пункт «Вниз»	Перемещение селектора списка вниз
3, Enter	Выбран пункт «Удалить»	Удаление выбранной записи
4, Enter	Выбран пункт «Выйти»	Возврат в главное меню см. Таблица 1

Таблица 4

Клавиши, вызывающее событие	Действие пользователя	Действие программы
1, Enter	Выбран пункт «Поиск по названию»	Открывается диалог ввода названия города.
2, Enter	Выбран пункт «Сортировка по названию»	Сортирует список по названию городов
3, Enter	Выбран пункт «Сортировка по численности»	Сортирует список по численности жителей городов
4, Enter	Выбран пункт «Отбор по численности»	Открывается новое меню см. Таблица 5 после чего происходит выборка по выбранному условию
5, Enter	Выбран пункт «Восстановить данные»	Все изменения в списке отменяются
6, Enter	Выбран пункт «Сохранить изменения»	Изменения сохраняются в основном списке
7, Enter	Выбран пункт «Выход»	Возврат в главное меню см. Таблица 1

Таблица 5

Клавиши, вызывающее событие	Действие пользователя	Действие программы
1, Enter	Выбран пункт «Больше или равно»	Запускается диалог ввода числа
2, Enter	Выбран пункт «Меньше чем»	Запускается диалог ввода числа

Далее представлены схемы:

sort_by_name.cpp – Сортировка списка по названию столиц

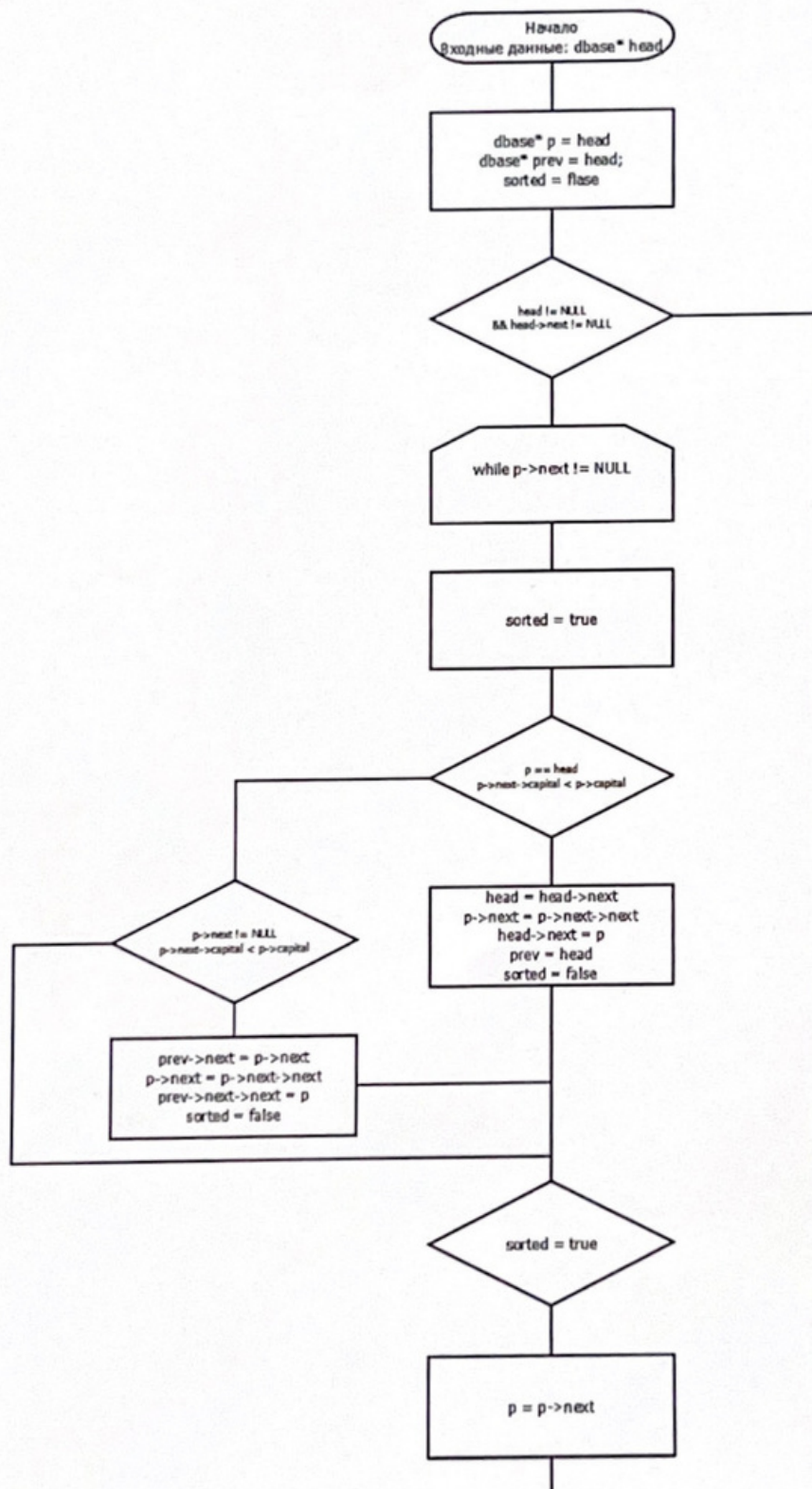


Рисунок 1 sort_by_name() часть 1

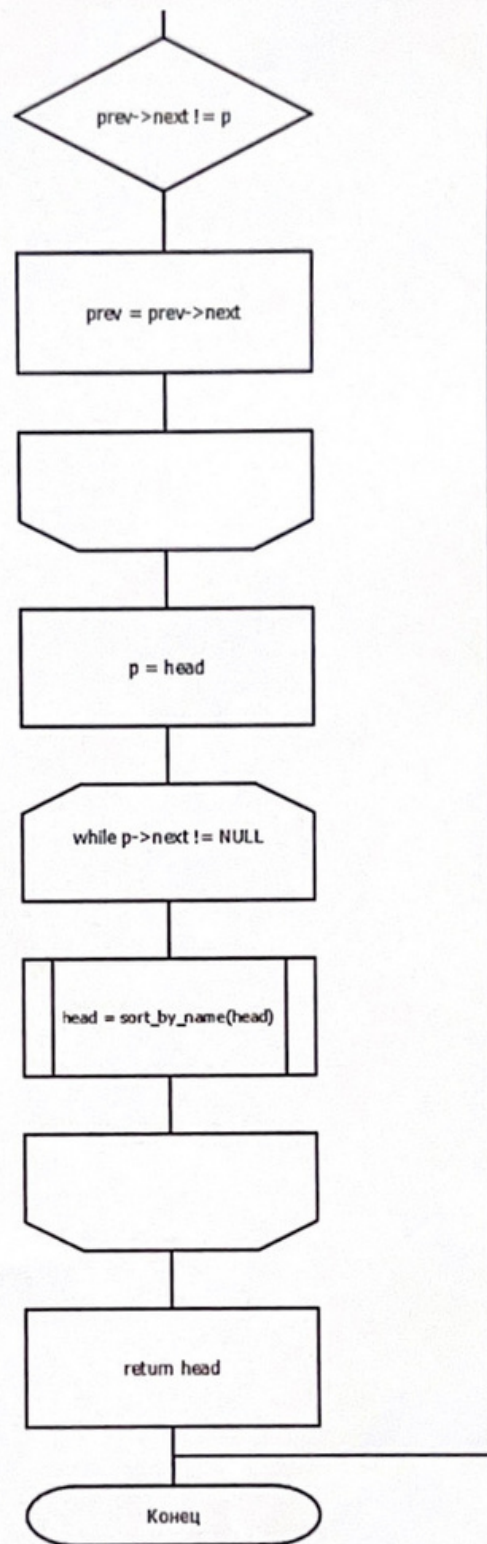


Рисунок 2 `sort_by_name()` часть 2

Структурная схема главного меню:

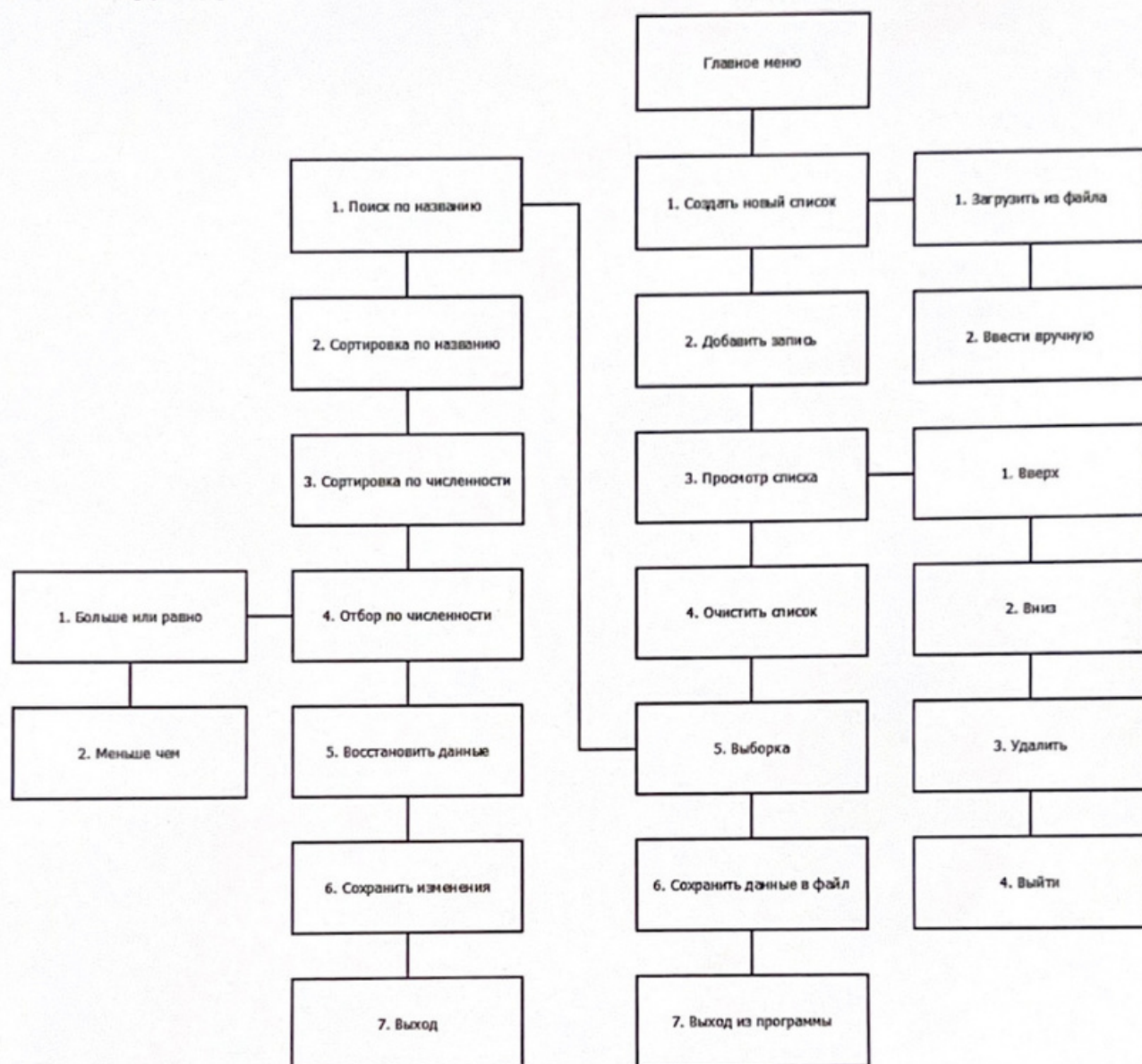


Рисунок 3 Структурная схема меню программы

Описание функции на языке Assembler:

Известно, что функции, разработанные на языке ассемблера работают быстрее и занимают меньше места. В качестве такой функции была выбрана функция сравнения.

Для реализации функции сравнения были использованы следующие ассемблерные инструкции:

- mov выполняет копирование содержимого источника в приемник;
- cmp выполняет сравнение двух операндов;
- jmp выполняет безусловный переход на метку;
- ja выполняет переход на метку, если первый операнд был больше второго при выполнении инструкции cmp.

Руководство пользователя

Программа предназначена для хранения информации данных о столицах стран. Интерфейс программы интуитивно понятен. Программа позволяет создавать новый список вручную или читать его из файла, добавлять и удалять новые записи, а также производить отбор нужной информации путем функций выборки, поиска и фильтрации.

Для просмотра созданного списка нужно открыть соответствующее меню:

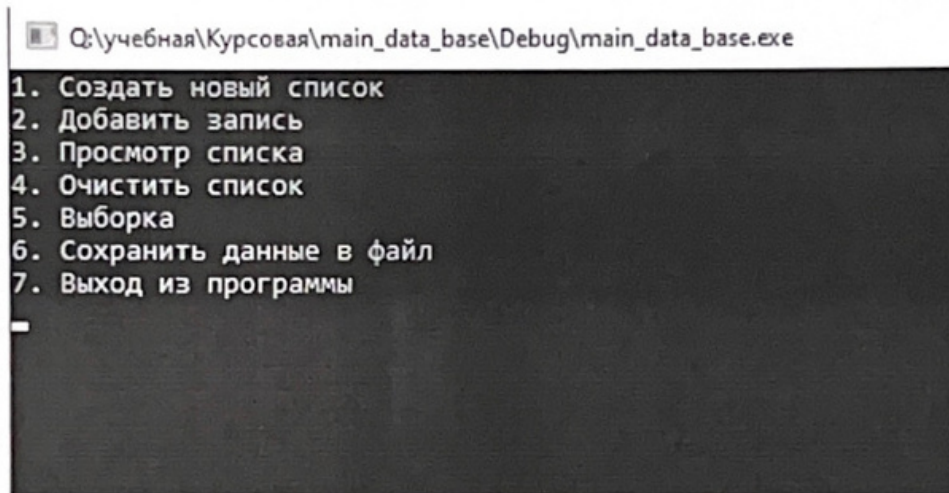


Рисунок 4

Навигация по списку осуществляется путем ввода соответствующих команд:

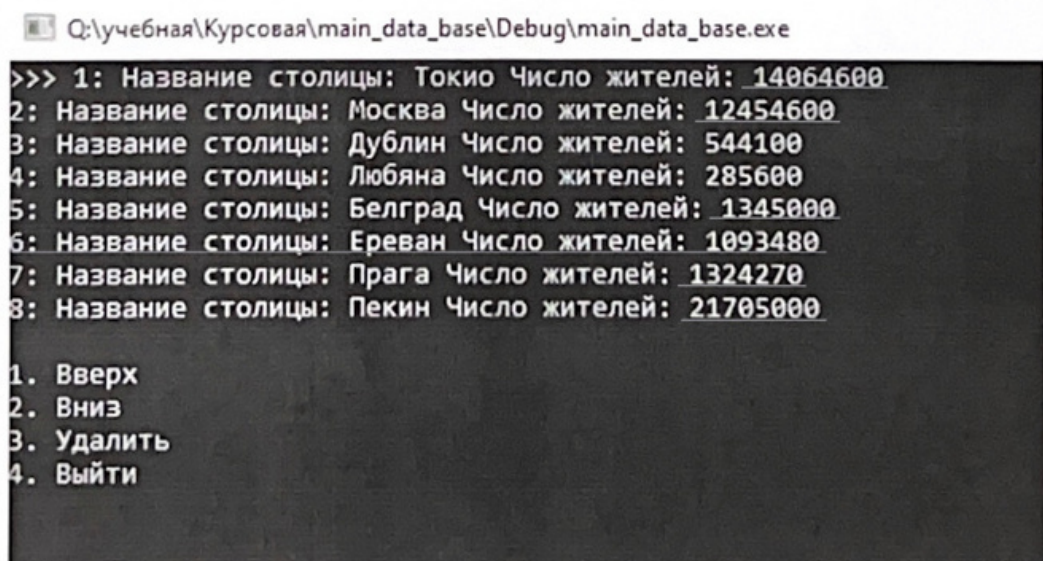


Рисунок 5

Удаление выбранного элемента возвращает указатель выбора в начальное положение:

Q:\учебная\Курсовая\main_data_base\Debug\main_data_base.exe

```
>>> 1: Название столицы: Токио Число жителей: 14064600
2: Название столицы: Москва Число жителей: 12454600
3: Название столицы: Дублин Число жителей: 544100
4: Название столицы: Любяна Число жителей: 285600
5: Название столицы: Белград Число жителей: 1345000
6: Название столицы: Ереван Число жителей: 1093480
7: Название столицы: Прага Число жителей: 1324270
8: Название столицы: Пекин Число жителей: 21705000

1. Вверх
2. Вниз
3. Удалить
4. Выйти
```

Рисунок 6

Q:\учебная\Курсовая\main_data_base\Debug\main_data_base.exe

```
>>> 1: Название столицы: Токио Число жителей: 14064600
2: Название столицы: Москва Число жителей: 12454600
3: Название столицы: Дублин Число жителей: 544100
4: Название столицы: Любяна Число жителей: 285600
5: Название столицы: Ереван Число жителей: 1093480
6: Название столицы: Прага Число жителей: 1324270
7: Название столицы: Пекин Число жителей: 21705000

1. Вверх
2. Вниз
3. Удалить
4. Выйти
```

Рисунок 7

Добавление записи вызывает диалоговое окно, в котором пользователь вводит нужные данные:

Q:\учебная\Курсовая\main_data_base\Debug\main_data_base.exe

```
Введите название города:
Белгад
Введите число жителей:
1345000_
```

Рисунок 8

По завершении добавления программа автоматически возвращается в главное меню.

Во вкладке «Выборка» пользователь может выбрать нужные ему функции для преобразования списка данных:

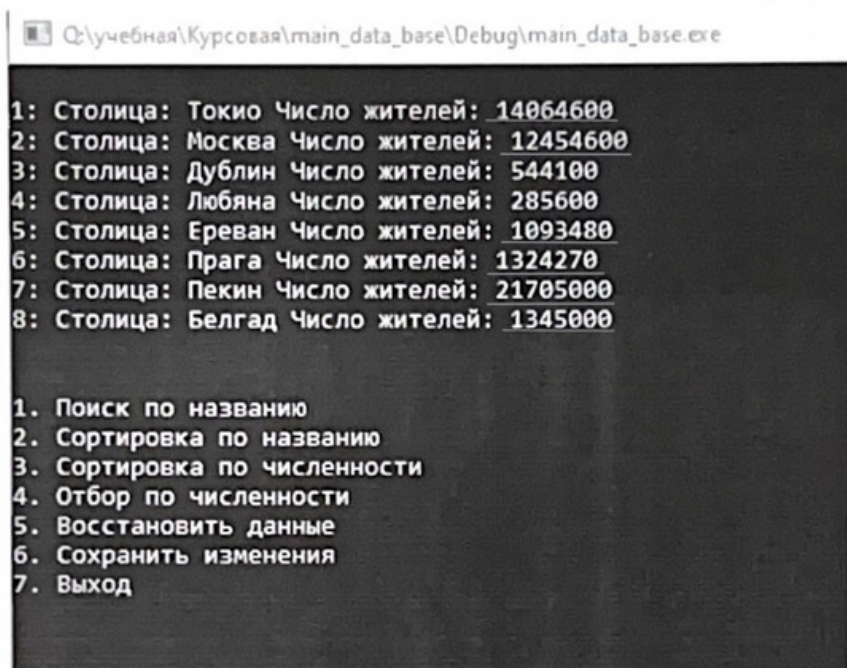


Рисунок 9

После выбора сортировки по названию список станет упорядочен по именам городов:

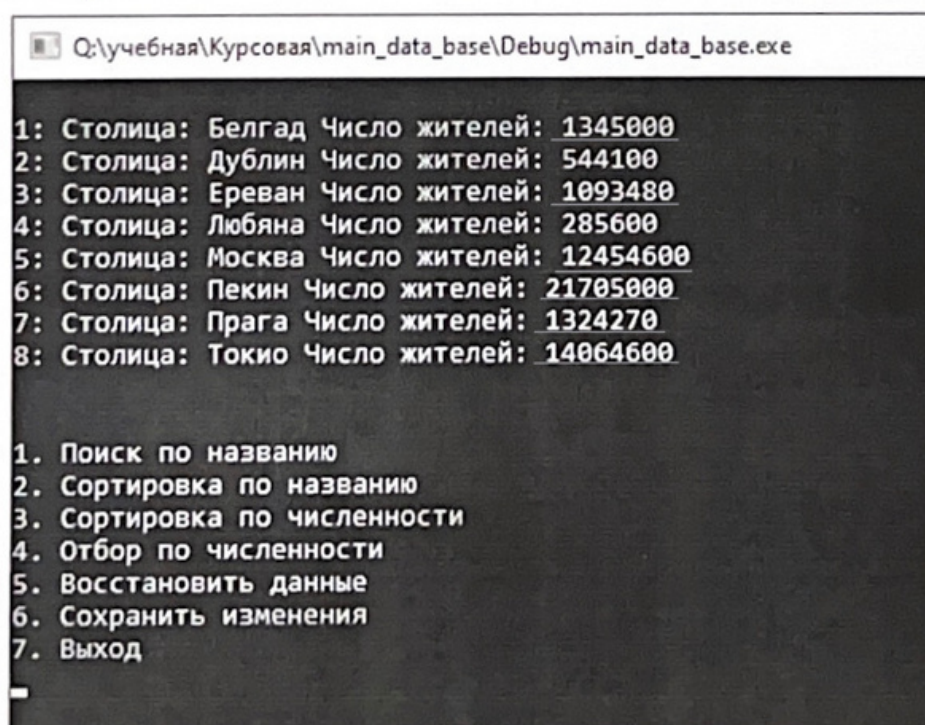


Рисунок 10

При сортировке по численности список изменится соответственно:

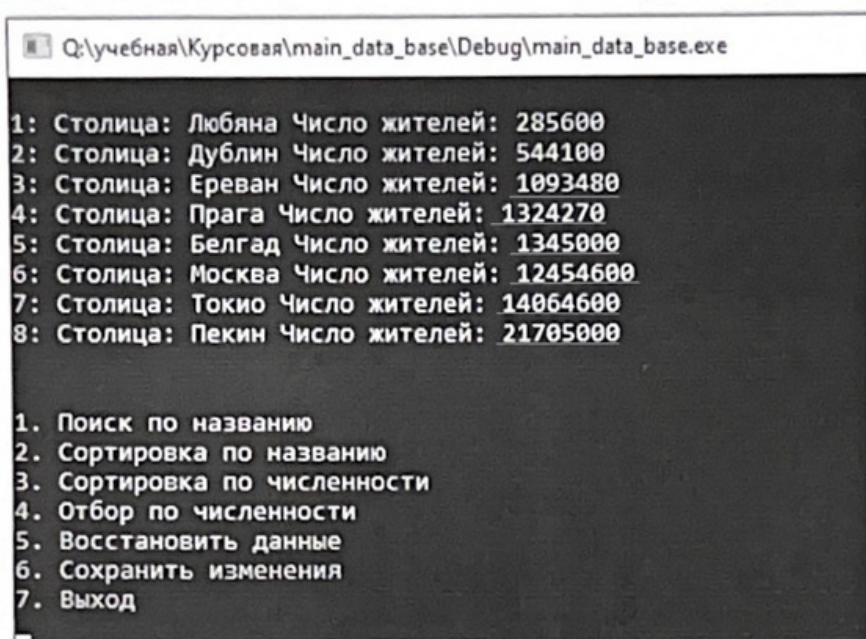


Рисунок 11

Поиск по названию позволяет найти информацию об искомом городе если запись о таком существует:

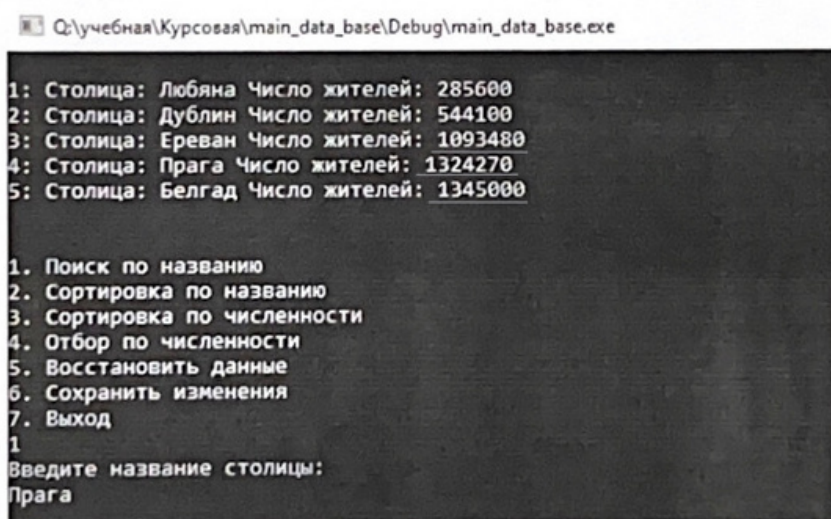


Рисунок 12

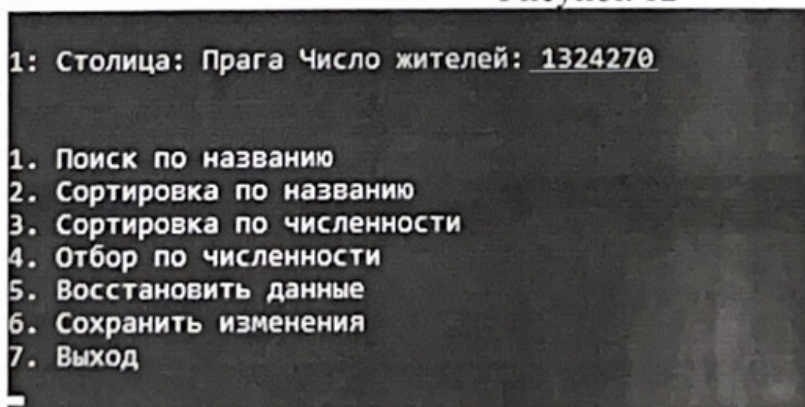


Рисунок 13

Отбор по численности предлагает пользователю задать условие, по которому будут отобраны элементы списка:

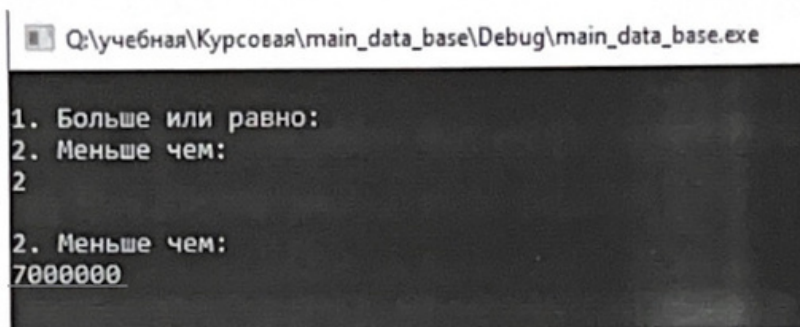


Рисунок 14

В результате список изменится:

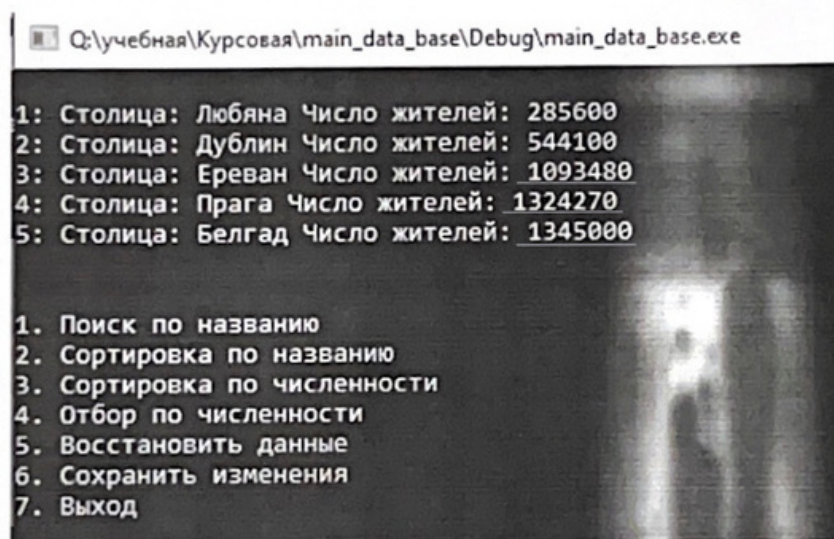


Рисунок 15

Пользователь может восстановить первоначальный вид списка в случае ошибки, или если сохранение результатов ему не требуется. В случае если пользователь хочет сохранить список в данном виде, он может вызвать пункт «Сохранить изменения».

В конце всех преобразований пользователь может сохранить конечный вид списка в файл, из которого потом можно будет снова считать данные:

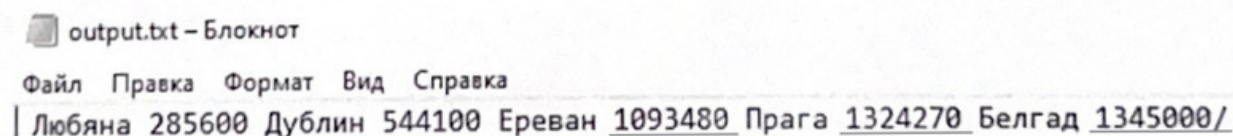


Рисунок 16 Результат работы программы

Для выхода из программы нужно выбрать пункт главного меню «Выход из программы», после чего программа прекратит работу.

Заключение

При выполнении курсовой работы были получены навыки разработки многомодульных программ. Была освоена работа с файлами, оперативной памятью и консольным интерфейсом. В ходе отладки были получены навыки тестирования программ и выявления причин появления ошибок.

В рамках данной курсовой работы была написана программа базы данных столиц стран. Данная программа имеет все необходимые функции работы с базами данных.

В дальнейшем разработанную программу можно улучшить, путем расширения ее функционала и добавления более удобного графического интерфейса с использованием управления мышью, что в целом позволит улучшить опыт пользователя при ее использовании.

Список используемых источников

1. К. Джемса «Учимся программировать на языке C++». 1997 г.
2. Л. Скэнлон «Программирование на языке ассемблера». 1991 г.
3. MSDN
4. Рудольф Марек. «Ассемблер на примерах». 2005 г.
5. В. И. Юров «Assembler». 2010 г.

Приложение А

Листинг программы

Main_data_base.cpp

```
#include <string>
#include <iostream>
#include <fstream>
#include "menu.h"

using namespace std;

void main()
{
    system("chcp 1251");
    system("cls");
    setlocale(LC_ALL, "Russian_Russia.1251");

    menu();

    system("pause");
}
```

Menu.cpp

```
#include "menu.h"

void menu() {
    first_screen();
    system("pause");
    dbase* head = NULL;
    dbase* filtered = NULL;
    bool quit_menu = 0;
    int msg;

    while (!quit_menu) {
        system("cls");
        cout << "1. Создать новый список\n2. Добавить запись\n3. Просмотр списка\n4. Очистить список\n5. Выборка\n6. Сохранить данные в файл\n7. Выход из программы\n";
```

```

cin >> msg;
switch (msg)
{
case 1: {
    system("cls");
    int msg1;
    cout << "1. Загрузить из файла\n2. Ввести вручную\n";
    cin >> msg1;
    switch (msg1)
    {
    case 1: {
        if (head != NULL) {
            head = clear_list(head);
            head = read_base("source/data_base.txt");
        }
        else {
            head = read_base("source/data_base.txt");
        }
        break;
    }
    case 2: {
        if (head != NULL) {
            head = clear_list(head);
            head = write_base(head);
        }
        else {
            head = write_base(head);
        }
        break;
    }
    default:
        break;
    }
    break;
}
case 2: {
    system("cls");
    int pop;
    string city;
    if (head == NULL) {
        cout << "Введите название города:\n";
        cin >> city;
        cout << "Введите число жителей:\n";
        cin >> pop;
        head = create_elem(city, pop);
    }
}
}

```

```

    }
    else {
        cout << "Введите название города:\n";
        cin >> city;
        cout << "Введите число жителей:\n";
        cin >> pop;
        add_last(create_elem(city, pop), head);
    }
    break;
}
case 3: {
    int line = 0;
    bool quit = 0;
    while (quit == 0) {
        system("cls");
        dbase* selected = head;
        dbase* p = head;
        int msg2;
        int k = 0;
        while (p != NULL) {
            if (line == k) {
                selected = p;
                cout << ">>> ";
            }
            cout << k + 1 << ": Название столицы: " << p->capital << " Число
жителей: " << p->population << endl;
            k++;
            p = p->next;
        }
        cout << "\n1. Вверх\n2. Вниз\n3. Удалить\n4. Выйти\n";
        cin >> msg2;
        switch (msg2)
        {
            case 1: {
                _asm {
                    mov eax, line
                    mov ebx, 0
                    cmp eax, ebx
                    ja M7
                    jmp R
                M7:
                    sub line, 1
                R:
            }
        }
    }
}

```

```

        break;
    }
    case 2: {
        if (selected != NULL) {
            if (selected->next != NULL) {
                line++;
            }
        }
        break;
    }
    case 3: {
        if (selected != NULL) {
            selected->del_flag = 1;
            head = del_elems(head);
        }
        line = 0;
        break;
    }
    case 4: {
        quit = 1;
        break;
    }
    default:
        break;
    }
}
break;
}
case 4: {
    head = clear_list(head);
    break;
}
case 5: {
    int msg3;
    bool quit = 0;
    dbase* filtered = NULL;
    filtered = copy_list(head);
    while (quit == 0) {
        system("cls");
        print_list(filtered);
    }
}

```

cout << "\n1. Поиск по названию\n2. Сортировка по названию\n3. Сортировка по численности\n4. Отбор по численности\n5. Восстановить данные\n6. Сохранить изменения\n7. Выход\n";

```

cin >> msg3;
switch (msg3)
{
case 1: {
    cout << "Введите название столицы:\n";
    string name;
    cin >> name;
    filtered = search_by_name(filtered, name);
    break;
}
case 2: {
    filtered = sort_by_name(filtered);
    break;
}
case 3: {
    filtered = sort_by_population(filtered);
    break;
}
case 4: {
    system("cls");
    int msg4;
    cout << "\n1. Больше или равно:\n2. Меньше чем:\n";
    cin >> msg4;
    switch (msg4)
    {
    case 1: {
        cout << "\n1. Больше или равно:\n";
        int number;
        cin >> number;
        filtered = filter_list_by_population(filtered, number, 0);
        break;
    }
    case 2: {
        cout << "\n2. Меньше чем:\n";
        int number;
        cin >> number;
        filtered = filter_list_by_population(filtered, number, 1);
        break;
    }
    default:
        break;
    }
    break;
}
case 5: {

```

```

        filtered = clear_list(filtered);
        filtered = copy_list(head);
        break;
    }
    case 6: {
        head = clear_list(head);
        head = copy_list(filtered);
        break;
    }
    case 7: {
        quit = 1;
        break;
    }
    default:
        break;
    }
}
filtered = clear_list(filtered);
break;
}
case 6: {
    save_to_file(head);
    break;
}
case 7: {
    quit_menu = 1;
    break;
}
default:
    break;
}
}
}

```

Read_base.cpp

```

#include "read_base.h"

dbase* read_base(string name) {
    dbase* head = NULL;
    ifstream file;
    string city;
    int pop = 0;
    char c;
    file.open(name);

```

```

if (file.get(c) && c != '/') {
    file >> city;
    file >> pop;
    head = create_elem(city, pop);
}

while (file.get(c)) {
    if (c != '/') {
        file >> city;
        file >> pop;
        add_last(create_elem(city, pop), head);
    }
}
file.close();
return head;
}

```

Write_base.cpp

```

#include "write_base.h"

dbase* write_base(dbase* head) {
    int n, i;
    string city;
    int pop = 0;
    cout << "Введите число элементов:\n";
    cin >> n;
    if (n > 0) {
        cout << "Введите название города:\n";
        cin >> city;
        cout << "Введите число жителей:\n";
        cin >> pop;
        head = create_elem(city, pop);
    }
    for (i = 1; i < n; i++) {
        cout << "Введите название города:\n";
        cin >> city;
        cout << "Введите число жителей:\n";
        cin >> pop;
        add_last(create_elem(city, pop), head);
    }
    return head;
}

```

Create_elem.cpp

```
#include "create_elem.h"

dbase* create_elem(string city, int pop) {
    dbase* p = new dbase;
    p->capital = city;
    p->population = pop;
    p->next = NULL;
    p->del_flag = 0;
    return p;
}
```

Add_last.cpp

```
#include "add_last.h"

void add_last(dbase* elem, dbase* head) {
    dbase* p = head;
    while (p->next != NULL) {
        p = p->next;
    }
    p->next = elem;
}
```

Clear_list.cpp

```
#include "clear_list.h"

dbase* clear_list(dbase* head) {
    dbase* p = head;
    dbase* buf;
    if (head != NULL) {
        while (p->next != NULL) {
            if (p->next->del_flag == 0) {
                buf = p->next;
                p->next = p->next->next;
                delete buf;
            }
            else {
                p = p->next;
            }
        }
        if (head->del_flag == 0) {
            buf = head;
        }
    }
}
```

```

        head = head->next;
        delete buf;
    }
}
return head;
}

```

Copy_list.cpp

```

#include "copy_list.h"

dbase* copy_list(dbase* head) {
    dbase* copied_head = NULL;
    if (head != NULL) {
        copied_head = create_elem(head->capital, head->population);
        dbase* p = head->next;
        while (p != NULL) {
            add_last(create_elem(p->capital, p->population), copied_head);
            p = p->next;
        }
    }
    return copied_head;
}

```

Del_elems.cpp

```
#include "del_elems.h"

dbase* del_elems(dbase* head) {
    dbase* p = head;
    dbase* buf;
    if (head != NULL) {
        while (p->next != NULL) {
            if (p->next->del_flag == 1) {
                buf = p->next;
                p->next = p->next->next;
                delete buf;
            }
            else {
                p = p->next;
            }
        }
        if (head->del_flag == 1) {
            buf = head;
            head = head->next;
            delete buf;
        }
    }
    return head;
}
```

Filter_list_by_population.cpp

```
#include "filter_list_by_population.h"

dbase* filter_list_by_population(dbase* head, int number, bool less) {
    dbase* p = head;
    while (p != NULL) {
        if ((p->population < number) != less) {
            p->del_flag = 1;
        }
        p = p->next;
    }
    head = del_elems(head);
    return head;
}
```

Print_list.cpp

```
#include "print_list.h"
```

```
void print_list(dbase* head) {  
    dbase* p = head;  
    int i = 1;  
    cout << endl;  
    while (p != NULL) {  
        cout << i << ": Столица: " << p->capital << " Число жителей: " << p->  
population << endl;  
        p = p->next;  
        i++;  
    }  
    cout << endl;  
}
```

Save_to_file.cpp

```
#include "save_to_file.h"
```

```
void save_to_file(dbase* head) {  
    ofstream file;  
    file.open("source/output.txt");  
    if (head != NULL) {  
        dbase* p = head;  
        while (p != NULL) {  
            file << " " << p->capital << " " << p->population;  
            p = p->next;  
        }  
    }  
    file << '/';  
    file.close();  
}
```

Search_by_name.cpp

```
#include "search_by_name.h"
```

```
dbase* search_by_name(dbase* head, string name)  
{  
    dbase* p = head;  
    while (p != NULL) {  
        if (p->capital != name) {  
            p->del_flag = 1;  
        }  
    }  
}
```

```

    }
    p = p->next;
}
head = del_elems(head);
return head;
}

```

Sort_by_name.cpp

```

#include "sort_by_name.h"

dbase* sort_by_name(dbase* head) {
    dbase* p = head;
    dbase* prev = p;
    bool sorted = 0;
    if (head != NULL) {
        if (head->next != NULL) {
            while (p->next != NULL) {
                sorted = 1;
                if (p == head) {
                    if (p->next->capital < p->capital) {
                        head = head->next;
                        p->next = p->next->next;
                        head->next = p;
                        prev = head;
                        sorted = 0;
                    }
                }
            }
        }
        else {
            if (p->next != NULL) {
                if (p->next->capital < p->capital) {
                    prev->next = p->next;
                    p->next = p->next->next;
                    prev->next->next = p;
                    sorted = 0;
                }
            }
        }
    }
    if (sorted == 1) {
        sorted = 0;
        p = p->next;
    }
    if (prev->next != p) {
        prev = prev->next;
    }
}

```

```

    }
    p = head;
    while (p->next != NULL) {
        if (p->next->capital < p->capital) {
            head = sort_by_name(head);
        }
        p = p->next;
    }
}
}
return head;
}

```

Sort_by_population.cpp

```

#include "sort_by_population.h"

dbase* sort_by_population(dbase* head) {
    dbase* p = head;
    dbase* prev = p;
    bool sorted = 0;
    if (head != NULL) {
        if (head->next != NULL) {
            while (p->next != NULL) {
                sorted = 1;
                if (p == head) {
                    if (p->next->population < p->population) {
                        head = head->next;
                        p->next = p->next->next;
                        head->next = p;
                        prev = head;
                        sorted = 0;
                    }
                }
                else {
                    if (p->next != NULL) {
                        if (p->next->population < p->population) {
                            prev->next = p->next;
                            p->next = p->next->next;
                            prev->next->next = p;
                            sorted = 0;
                        }
                    }
                }
                p = p->next;
            }
        }
        if (sorted == 1) {

```

```

        sorted = 0;
        p = p->next;
    }
    if (prev->next != p) {
        prev = prev->next;
    }
}
p = head;
while (p->next != NULL) {
    if (p->next->population < p->population) {
        head = sort_by_population(head);
    }
    p = p->next;
}
}
return head;
}

```

First_screen.cpp

```

#include "first_screen.h"

void first_screen() {
    cout << "-----СТОЛИЦЫ СТРАН-----\n";
    cout << "\n
    столиц.\n
    ст.гр.\t21BB1\t Ананьева А.В.\n\n";
}

```

Работа:\tБаза данных
Выполнила:\n

Data_base.h

```

#pragma once
#include <string>
#include <iostream>
using namespace std;

struct dbase {
    string capital;
    int population;
    dbase* next;
    bool del_flag;
};

```

Приложение В

Информационная заставка:

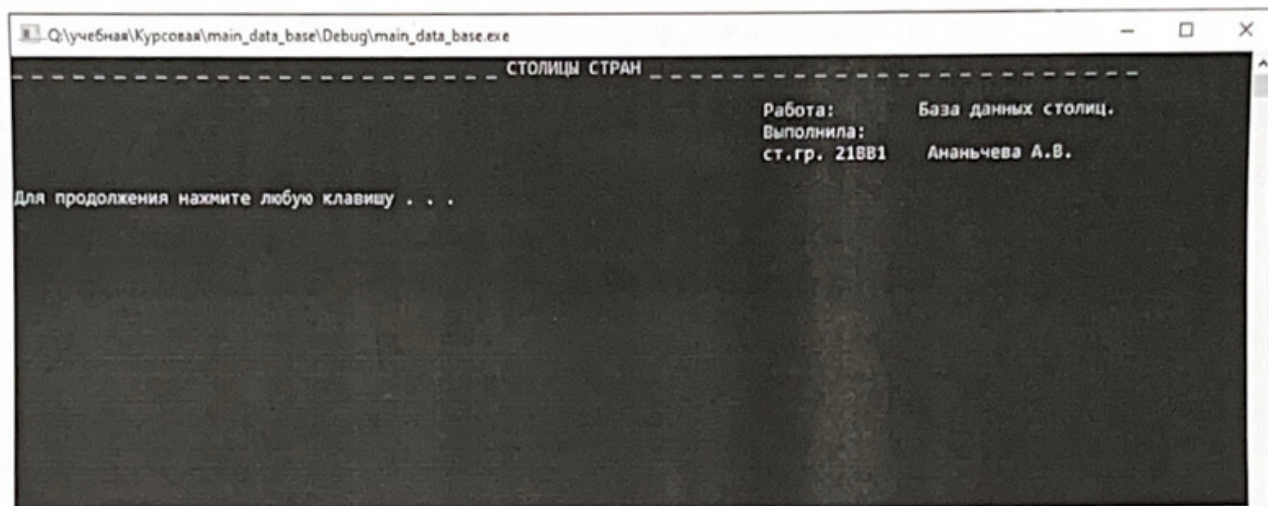


Рисунок 17 Информационная заставка

Приложение С

Результат работы программы

Output.txt

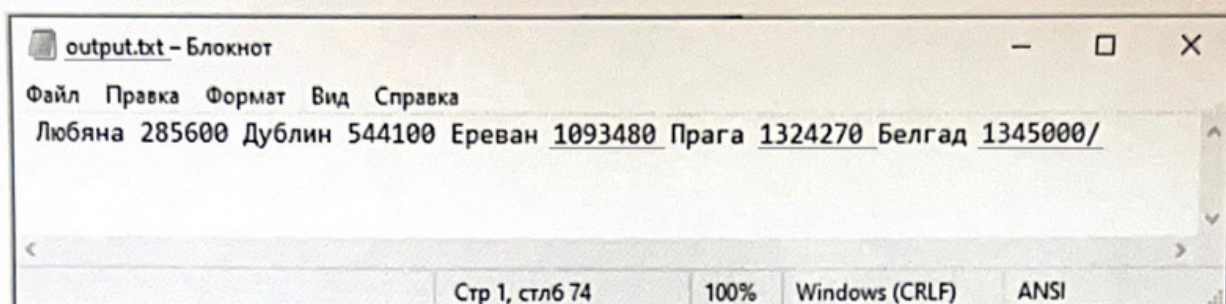


Рисунок 18 Результат работы программы