**ПРИЛОЖЕНИЕ А**

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

# КАФЕДРА ИНТЕЛЛЕКТУАЛЬНЫХ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

ОБРАБОТКА МАССИВОВ СТРУКТУРИРОВАННЫХ ДАННЫХ

**«**СТРАНЫ МИРА**»**

**ТЕКСТ ПРОГРАММЫ**

КР.АС59.200043-01 81 00

|  |  |
| --- | --- |
| Руководитель | И.Н. Аверина |
| Выполнил  студент 1 курса  группы АС-59 | И.В. Абоимов |
|  |  |
|  |  |
|  |  |

## 2021

**СОДЕРЖАНИЕ**

KursArbeit.cpp – основной модуль, в котором содержится меню для работы с программой. В ней же перед меню вызывается функция чтения из файла, описанная в другом файле.

lender.h – библиотека, содержащая структуру данных.

Dynamik.h, Dynamik.cpp – пользовательская библиотека, в которой содержатся функции, изменяющие содержание файла: функция чтения из файла, функция добавления записей, функция редактирования записей, функция удаления записей.

Statik.h, Statik.cpp – пользовательская библиотека, в который содержатся основные функции, не изменяющие содержания файла: функция вывода данных, функция выборки по указанному промежутку численности населения.

Sortierung.h, Sortierung.cpp – пользовательская функция, в которой содержатся все функции, относящиеся к сортировке.

Suchen.h, Suchen.cpp – пользовательская библиотека, в которой сожержатся все функции, относящиеся к поиску данных.

Peripherie.h, Peripherie.cpp – пользовательская библиотека, содержащая некоторые вспомогательные функции.

**KursArbeit.cpp**

// стандартные библиотеки

#include <iostream> // для ввода и вывода

#include <conio.h> // для \_getche()

// собственные библиотеки

#include "Peripherie.h" // библиотека вспомогательных функций

#include "lender.h" // библоитека структы

#include "Sortierung.h" // основная функция сортировки и сортировка при изменении данных

#include "Suchen.h" // основная функция поиска и поиск для удления данных

#include "Statik.h" // основные фукнции, которые не изменяют данные

#include "Dynamik.h" // основные функции, которые изменяют данные

using namespace std; // подключения пространства имён std

int main() // исполняемая функция

{

setlocale(LC\_ALL, "Russian"); // поддержка кириллицы

// начальные условия

lander\* land = new lander[0]; // массив структур

int Kapazitat = 0; // размерность массива

// чтение данных из текстового файла

LesenDaten(land, Kapazitat);

// массив указателей на функции

void((\*Arbeiter[8]))(lander\*& land , int& Kapazitat);

Arbeiter[0] = SreibenDaten; // без изменения основного файла

Arbeiter[1] = SortieranDaten; // без изменения основного файла

Arbeiter[2] = AddierenDaten; // с изменением основного и индексно-бинарного файлов

// с изменением основного и индексно-бинарного файлов

Arbeiter[3] = SubstragierenDaten;

Arbeiter[4] = UmSreibenData; // с изменением основного и индексно-бинарного файлов

Arbeiter[5] = WahlDaten; // без изменения основного файла

Arbeiter[6] = SuchenDaten; // без изменения основного файла

Arbeiter[7] = SpeichernDaten; // для сохранений данных в файл

// работа с данными

char ctrl = '1'; // переменая зацикливания для while

main\_menu:; // строки 56, 66

system("cls"); // очитска консоли

// меню для работы программы с полученными данными

cout << "Меню:\n\n"

<< "1 - вывод данных на экран\n"

<< "2 - меню сортировки\n"

<< "3 - добавление записи\n"

<< "4 - удаление записи\n"

<< "5 - редактирование записи\n"

<< "6 - вывод стран по указанному промежутку численности населения\n"

<< "7 - поиск информации\n"

<< "8 - сохранение данных в текстовый файл\n"

<< "0 - выход из программы\n"

<< "\nВыбор пункта меню: ";

ctrl = \_getche(); // ввод пункта меню

// проверка вводимых данных; сравнение идёт по номеру введённого символа в таблице ASCII

if ((ctrl < 48) || (ctrl > 56))

{

Prufung(); // оповещение о некорректных данных

goto main\_menu; // перенаправление на ввод новых данных; строка 37

}

// выход из программы

else if (ctrl == '0') // проверка условия для выхода из программы

{

delete[] land; // очистка основного буфера

return 0; // возврат нулевого значения в главную функцию

}

// -49 от кода символа в таблице ASCII, т.к. необходимо обеспечить попадание в нулевой элемент

массива

Arbeiter[ctrl - 49](land, Kapazitat); // распределение по меню

goto main\_menu; // строка 37

}

**lender.h**

#pragma once

struct lander // стркуктра данных о стране

{

// индекс для индексных файлов и некторой работы с массивом (4 символа)

unsigned int index;

char Name[26]; // название страны

char HauptStadt[26]; // название столицы

unsigned int Nas; // численность населения

unsigned int Terr; // площадь страны

char VAL[4]; // код валюты страны

unsigned int Nas\_pro\_Terr; // рассчитываемое поле - плотность населения

};

**Dynamik.h**

#pragma once

#include"lender.h" // подключение структуры

void AddierenDaten(lander\*& land, int& Kapazitat); // добавление записей

void SubstragierenDaten(lander\*& land, int& Kapazitat); // удаление записей

void UmSreibenData(lander\*& land, int& Kapazitat); // редактирвоание записей

void LesenDaten(lander\*& land, int& Kapazitat); // получение исходных данных

void SpeichernDaten(lander\*& land, int& Kapazitat); // для сохранений данных в файл

**Dinamik.cpp**

#include <iostream> // для ввода и вывода

#include <conio.h> // для \_getche()

#include <stdio.h> // для работы с файлами

#include <fstream> // для работы с файлами

#include <iomanip> // для setw()

#include <cmath> // для pow()

#include "Dynamik.h" // плдключение одноимённого заголовочного файла

#include "Peripherie.h" // подключение кода файла периферии

#include "Suchen.h" // подключение кода файла поиска

#include "Sortierung.h" // подключение кода файла сортировки

using namespace std; // подключение пространства имён std

ifstream wanderung; // глобальная переменная для чтения файла

ofstream reisen; // глобальная переменная для записи в файл

string DerNameDesTextes = "BinLender.txt"; // стандарное имя файла

void LesenDaten(lander\*& land, int& Kapazitat) // чтение данных из файла

{

// открытие файла для работы с ним//

wanderung.open(DerNameDesTextes, ios::binary | ios::out);

// проверка открытия файла

while (!wanderung.is\_open()) // пока не достигнет конца файла

{

// оповещение о неоткрытии файла

cout << "Файл не открыт, введите имя файла вручную: ";

cin >> DerNameDesTextes; // ввод полного имени файла вручную

wanderung.open(DerNameDesTextes); // открытие файла с введёным именем

system("cls"); // очистка консоли, чтобы её не загромождать

}

wanderung.read((char\*)&Kapazitat, sizeof(Kapazitat));

if (Kapazitat != 0)

{

delete[] land; // очистка прошлого буфера

land = new lander[Kapazitat]; // переопределение массива

for (int i = 0; i < Kapazitat; i++) // процесс считывания данных

wanderung.read((char\*)&land[i], sizeof(land[i]));

wanderung.close(); // закрытие файла после чтения

}

else

{

Kapazitat++; // увеличение размерности массива на первый элемент

delete[] land; // очистка прошлого буфера

land = new lander[Kapazitat]; // переопределение массива

ErsteSchrei:; // строки 58, 66, 77, 85, 96, 102, 114, 120, 128, 137, 144

cout << "Внимание, записи отсутствуют! Сделайте первую запись:\n\n";

char Zeitspanne[64];

// инддексация

land[0].index = 1;

// ввод названия страны

cout << "Введите название страны: ";

cin >> Zeitspanne;

if (strlen(Zeitspanne) > 25) // проверка по длине

{

Prufung(); // оповещение о некорректном вводе

goto ErsteSchrei; // строка 47

}

for (int i = 0; i < strlen(Zeitspanne); i++) // проверка на символы

if ((Zeitspanne[i] < 65) || (Zeitspanne[i] > 90))

if ((Zeitspanne[i] < 97) || (Zeitspanne[i] > 122))

if ((Zeitspanne[i] != 95) && (Zeitspanne[i] != 45))

{

Prufung(); // оповещение о некорректном вводе

goto ErsteSchrei; // строка 47

}

KorrigierenDaten(Zeitspanne, 1); // коррекция введённых данных

strcpy\_s(land[0].Name, Zeitspanne); // определеине названия страны

// ввод названия столицы

cout << "Введите название столицы: ";

cin >> Zeitspanne;

if (strlen(Zeitspanne) > 25) // проверка по длине

{

Prufung(); // оповещение о некорректном вводе

goto ErsteSchrei; // строка 47

}

for (int i = 0; i < strlen(Zeitspanne); i++) // проверка на символы

if ((Zeitspanne[i] < 65) || (Zeitspanne[i] > 90))

if ((Zeitspanne[i] < 97) || (Zeitspanne[i] > 122))

if ((Zeitspanne[i] != 95) && (Zeitspanne[i] != 45))

{

Prufung(); // оповещение о некорректном вводе

goto ErsteSchrei; // строка 47

}

KorrigierenDaten(Zeitspanne, 1); // коррекция введённых данных

strcpy\_s(land[0].HauptStadt, Zeitspanne); // определение названия столицы

// ввод численности населения

cout << "Введите численность населения: ";

cin >> Zeitspanne;

if (strlen(Zeitspanne) > 9) // проверка по длине

{

Prufung(); // оповещение о некорректном вводе

goto ErsteSchrei; // строка 47

}

for (int j = 0; j < strlen(Zeitspanne); j++) // проверка на символы

if ((static\_cast<int>(Zeitspanne[j]) - 48 < 0) || (static\_cast<int>(Zeitspanne[j]) - 48 > 9))

{

Prufung(); // оповещение о некорректном вводе

goto ErsteSchrei; // строка 47

}

land[0].Nas = 0; // обнуления для определения

// определение численности населения

for (int j = 0; j < strlen(Zeitspanne); j++)

land[0].Nas += (((static\_cast<int>(Zeitspanne[j])) - 48) \* pow(10,

(strlen(Zeitspanne) - 1 - j)));

// ввод площади

cout << "Введите площать: ";

cin >> Zeitspanne;

if (strlen(Zeitspanne) > 9) // проверка по длине

{

Prufung(); // оповещение о некорректном вводе

goto ErsteSchrei; // строка 47

}

for (int j = 0; j < strlen(Zeitspanne); j++) // проверка на символы

if ((static\_cast<int>(Zeitspanne[j]) - 48 < 0) || (static\_cast<int>(Zeitspanne[j]) - 48 > 9))

{

Prufung(); // оповещение о некорректном вводе

goto ErsteSchrei; // строка 47

}

land[0].Terr = 0; // обнуление для определения

for (int j = 0; j < strlen(Zeitspanne); j++) // определение площади

land[0].Terr += (((static\_cast<int>(Zeitspanne[j])) - 48) \* pow(10,

(strlen(Zeitspanne) - 1 - j)));

if (land[0].Terr == 0) // предотвращение деления на 0

{

Prufung(); // оповещение о некорректном вводе

goto ErsteSchrei; // строка 47

}

// ввод кода валюты страны

cout << "Введитье код валюты страны: ";

cin >> Zeitspanne;

if (strlen(Zeitspanne) != 3) // проверка по длине

{

Prufung(); // оповещение о некорректном вводе

goto ErsteSchrei; // строка 47

}

for (int i = 0; i < strlen(Zeitspanne); i++) // проверка на символы

if ((Zeitspanne[i] < 65) || (Zeitspanne[i] > 90))

if ((Zeitspanne[i] < 97) || (Zeitspanne[i] > 122))

{

Prufung(); // оповещение о некорректном вводе

goto ErsteSchrei; // строка 47

}

KorrigierenDaten(Zeitspanne, 2); // коррекция введённых данных

strcpy\_s(land[0].VAL, Zeitspanne); // определение валюты страны

// рассчёт плотности населения

land[0].Nas\_pro\_Terr = land[0].Nas / land[0].Terr;

// перезапись бинарного файла

// открытие бинарног файла

reisen.open(DerNameDesTextes, ios::binary | ios::out);

reisen.write((char\*)&Kapazitat, sizeof(Kapazitat)); // запись рабочей строки

// запись изменённой последовательности индексов

for (int i = 0; i < Kapazitat; i++)

reisen.write((char\*)&land[i], sizeof(land[i]));

reisen.close(); // закрытие файла

}

// изменение индексно-бинарных файлов

int K = Kapazitat; // переменная для передачи в нижестоящую функцию

SortierungNachVeranderungen(land, K); // функция изменения индексно-бинарных файлов

}

void AddierenDaten(lander\*& land, int& Kapazitat) // добавление записей

{

gt\_index:; // строки 185, 198, 208

// рабочии переменные

int metaindex = 0; // для выбора индекса, после которого надо будет добавить запись

char Zeitspanne[64]; // защитный буфер - защита от некорректных данных

int x = 0; // для задания количества новых записей

// вывод рабочих данных

system("cls"); // очистка консоли

cout << "Данные записей: \n\n";

Mutze(); // вывод шапки

for (int i = 0; i < Kapazitat; i++) // вывод самих данных через итерационный процесс

Schreiben(land, i);

// ввод номера записи

cout << "\nУкажите номер записи, после которой необходимо добавить информацию: ";

cin >> Zeitspanne; // ввод защитного буфера

// провека ввода, для преобразования в число

for (int i = 0; i < strlen(Zeitspanne); i++)

if ((Zeitspanne[i] < 48) || (Zeitspanne[i] > 57))

{

Prufung(); // оповещение о некорректном вводе

goto gt\_index; // строка 165

}

// получение индекса, после которого будет добавлена запись.

for (int i = 0; i < strlen(Zeitspanne); i++)

metaindex += ((Zeitspanne[i] - 48) \* pow(10, (strlen(Zeitspanne) - 1 - i)));

// провекра совпадений индекса

int leuchtturm = 0; // маячок для проверки индексов

for (int i = 0; i < Kapazitat; i++) // проверка существования введённого индекса

if (land[i].index == metaindex)

leuchtturm++;

if (leuchtturm == 0)

{

Prufung();

goto gt\_index; // строка 165

}

// ввод количества новых записей

cout << "\nСколько записей ВЫ хотите добавить записей? : ";

cin >> Zeitspanne;

for (int i = 0; i < strlen(Zeitspanne); i++) // проверка на символы

if ((Zeitspanne[i] < 48) || (Zeitspanne[i] > 57))

{

Prufung(); // оповещение о некорректном вводе

goto gt\_index; // строка 165

}

for (int i = 0; i < strlen(Zeitspanne); i++) // получение количества новых записей

x += ((Zeitspanne[i] - 48) \* pow(10, (strlen(Zeitspanne) - 1 - i)));

// расширение динамического массива

lander\* buffer = new lander[Kapazitat + x]; // создание переменного буфера

for (int i = 0; i < Kapazitat; i++) // сохранение данных в переменных буфер

buffer[i] = land[i];

delete[] land; // очистка основного буфера

land = new lander[Kapazitat + x]; // переопределение основного буфера

for (int i = 0; i < Kapazitat; i++) // сохранение данных в расширенный основной буфер

land[i] = buffer[i];

delete[] buffer; // очистка переменного буфера

cout << endl;

// блок ввода данных

if (metaindex == Kapazitat) // если ввёднный инт=декс совпадает с последним

{

// блок добавления записей

while (x > 0) // зацикливание до тех пор, пока не закончатся новые записи

{

gt\_VerSch:; // строки 239, 248, 260, 269, 281, 288, 301, 308, 317, 327, 335

// ввод названия страны

cout << "Введите название страны: ";

cin >> Zeitspanne;

if (strlen(Zeitspanne) > 25) // проверка на длину

{

Prufung(); // оповещение о некорректном вводе

system("cls"); // очистка консоли

goto gt\_VerSch; // строка 230

}

for (int i = 0; i < strlen(Zeitspanne); i++) // проверка на символы

if ((Zeitspanne[i] < 65) || (Zeitspanne[i] > 90))

if ((Zeitspanne[i] < 97) || (Zeitspanne[i] > 122))

if ((Zeitspanne[i] != 95) && (Zeitspanne[i] != 45))

{

// оповещение о некорректном вводе

Prufung();

system("cls"); // очистка консоли

goto gt\_VerSch; // строка 320

}

KorrigierenDaten(Zeitspanne, 1); // коррекция введённых данных

// определение названия страны

strcpy\_s(land[Kapazitat].Name, Zeitspanne);

// ввод названия столицы

cout << "Введите название столицы страны: ";

cin >> Zeitspanne;

if (strlen(Zeitspanne) > 25) // проверка на длину

{

Prufung(); // оповещение о некорректном вводе

system("cls"); // очистка консоли

goto gt\_VerSch; // строка 230

}

for (int i = 0; i < strlen(Zeitspanne); i++) // проверка на символы

if ((Zeitspanne[i] < 65) || (Zeitspanne[i] > 90))

if ((Zeitspanne[i] < 97) || (Zeitspanne[i] > 122))

if ((Zeitspanne[i] != 95) && (Zeitspanne[i] != 45))

{

// оповещение о некорректном вводе

Prufung();

system("cls"); // очистка консоли

goto gt\_VerSch; // строка 230

}

KorrigierenDaten(Zeitspanne, 1); // коррекция введённых данных

// определение названия столицы

strcpy\_s(land[Kapazitat].HauptStadt, Zeitspanne);

// ввод численности населения

cout << "Введите численность населения страны: ";

cin >> Zeitspanne;

if (strlen(Zeitspanne) > 10) // проверка на длину

{

Prufung(); // оповещение о некорректном вводе

system("cls"); // очистка консоли

goto gt\_VerSch; // строка 230

}

for (int i = 0; i < strlen(Zeitspanne); i++) // проверка на символы

if ((Zeitspanne[i] < 48) || (Zeitspanne[i] > 57))

{

Prufung(); // оповещение о некорректном вводе

system("cls"); // очистка консоли

goto gt\_VerSch; // строка 230

}

land[Kapazitat].Nas = 0; // обнуление для определения

// определение численности населения

for (int i = 0; i < strlen(Zeitspanne); i++)

land[Kapazitat].Nas += ((Zeitspanne[i] - 48) \* pow(10,

(strlen(Zeitspanne) - 1 - i)));

// ввод площади страны

cout << "Введите площадь страны: ";

cin >> Zeitspanne;

if (strlen(Zeitspanne) > 9) // проверка на длину

{

Prufung(); // оповещение о некорректном вводе

system("cls"); // очистка консоли

goto gt\_VerSch; // строка 230

}

for (int i = 0; i < strlen(Zeitspanne); i++) // проверка на символы

if ((Zeitspanne[i] < 48) || (Zeitspanne[i] > 57))

{

Prufung(); // оповещение о некорректном вводе

system("cls"); // очистка консоли

goto gt\_VerSch; // строка 230

}

land[Kapazitat].Terr = 0; // обнуление для определения

for (int i = 0; i < strlen(Zeitspanne); i++) // определение площади

land[Kapazitat].Terr += ((Zeitspanne[i] - 48) \* pow(10,

(strlen(Zeitspanne) - 1 - i)));

if (land[Kapazitat].Terr == 0)

{

Prufung(); // оповещение о некорректном вводе

system("cls"); // очистка консоли

goto gt\_VerSch; // строка 230

}

// ввод кода валюты страны

cout << "Введите код валюты страны: ";

cin >> Zeitspanne;

if (strlen(Zeitspanne) != 3) // проверка на длину

{

Prufung(); // оповещение о некорректном вводе

system("cls"); // очистка консоли

goto gt\_VerSch; // строка 230

}

for (int i = 0; i < strlen(Zeitspanne); i++) // проверка на символы

if ((Zeitspanne[i] < 65) || (Zeitspanne[i] > 90))

if ((Zeitspanne[i] < 97) || (Zeitspanne[i] > 122))

{

Prufung(); // оповещение о некорректном вводе

system("cls"); // очистка консоли

goto gt\_VerSch; // строка 230

}

KorrigierenDaten(Zeitspanne, 2); // коррекция введённых данных

strcpy\_s(land[Kapazitat].VAL, Zeitspanne); // определение кода валюты

// редакция плотности и индекса

// расчёт плотности населения

land[Kapazitat].Nas\_pro\_Terr = land[Kapazitat].Nas /

land[Kapazitat].Terr;

land[Kapazitat].index = Kapazitat + 1; // индексация

Kapazitat++; // увеличение размерности

// контроль оставшегося числа новых записей

x--; // уменьшение числа записей

if (x >= 1) // проверка количества введённых записей

cout << "\nСделайте ещё одну запись\n\n";

}

}

else // если введённый индекс не совпадает с последним

{

while (x > 0) // пока не закончатся новые записи

{

Kapazitat++; // расширения размерности массива

for (int i = Kapazitat - 1; i > 0; i--)

{

if (i - 1 == metaindex) // поиск необходимого элемента

{

// заключительный этап сортировки

land[i] = land[i - 1];

// индексация новой записи

land[i].index = metaindex + 2;

gt\_ZVerSch:; // строки 371, 380, 392, 401, 413, 420, 433, 440, 449, 459, 467

// ввод названия страны

cout << "Введите название страны: ";

cin >> Zeitspanne;

if (strlen(Zeitspanne) > 25) // проверка на длину

{

Prufung(); // оповещение о некорректном вводе

system("cls"); // очистка консоли

goto gt\_ZVerSch; // строка 362

}

// проверка на символы

for (int i = 0; i < strlen(Zeitspanne); i++)

if ((Zeitspanne[i] < 65) || (Zeitspanne[i] > 90))

if ((Zeitspanne[i] < 97) || (Zeitspanne[i] > 122))

if ((Zeitspanne[i] != 95) && (Zeitspanne[i] != 45))

{

Prufung(); // оповещение о некорректном вводе

system("cls"); // очистка консоли

goto gt\_ZVerSch; // строка 362

}

// коррекция введённых данных

KorrigierenDaten(Zeitspanne, 1);

// определение названия страны

strcpy\_s(land[metaindex].Name, Zeitspanne);

// ввод названия столицы

cout << "Введите название столицы страны: ";

cin >> Zeitspanne;

if (strlen(Zeitspanne) > 25) // проверка на длину

{

Prufung(); // оповещение о некорректном вводе

system("cls"); // очистка консоли

goto gt\_ZVerSch; // строка 362

}

// проверка на символы

for (int i = 0; i < strlen(Zeitspanne); i++)

if ((Zeitspanne[i] < 65) || (Zeitspanne[i] > 90))

if ((Zeitspanne[i] < 97) || (Zeitspanne[i] > 122))

if ((Zeitspanne[i] != 95) && (Zeitspanne[i] != 45))

{

Prufung(); // оповещение о некорректном вводе

system("cls"); // очистка консоли

goto gt\_ZVerSch; // строка 362

}

// коррекция введённых данных

KorrigierenDaten(Zeitspanne, 1);

// определение названия столицы

strcpy\_s(land[metaindex].HauptStadt, Zeitspanne);

// ввод численности населения страны

cout << "Введите численность населения страны: ";

cin >> Zeitspanne;

if (strlen(Zeitspanne) > 10) // проверка на длину

{

Prufung(); // оповещение о некорректном вводе

system("cls"); // очистка консоли

goto gt\_ZVerSch; // строка 362

}

// проверка на символы

for (int i = 0; i < strlen(Zeitspanne); i++)

if ((Zeitspanne[i] < 48) || (Zeitspanne[i] > 57))

{

Prufung(); // оповещение о некорректном вводе

system("cls"); // очистка консоли

goto gt\_ZVerSch; // строка 362

}

land[metaindex].Nas = 0; // обнуление для определения

// определение численности населения

for (int i = 0; i < strlen(Zeitspanne); i++)

land[metaindex].Nas += ((Zeitspanne[i] - 48) \*

pow(10, (strlen(Zeitspanne) - 1 - i)));

// ввод площади населения

cout << "Введите площадь старны: ";

cin >> Zeitspanne;

if (strlen(Zeitspanne) > 9) // проверка на длину

{

Prufung(); // оповещение о некорректном вводе

system("cls"); // очистка консоли

goto gt\_ZVerSch; // строка 362

}

// проверка на символы

for (int i = 0; i < strlen(Zeitspanne); i++)

if ((Zeitspanne[i] < 48) || (Zeitspanne[i] > 57))

{

Prufung(); // оповещение о некорректном вводе

system("cls"); // очистка консоли

goto gt\_ZVerSch; // строка 362

}

land[metaindex].Terr = 0; // обнуление для определения

// определение площади

for (int i = 0; i < strlen(Zeitspanne); i++)

land[metaindex].Terr += ((Zeitspanne[i] - 48)

\* pow(10, (strlen(Zeitspanne) - 1 - i)));

// предовтращения деления на 0

if (land[metaindex].Terr == 0)

{

Prufung(); // оповещение о некорректном вводе

system("cls"); // очистка консоли

goto gt\_ZVerSch; // строка 362

}

// ввод кода валюты

cout << "Введите код валюты страны: ";

cin >> Zeitspanne;

if (strlen(Zeitspanne) != 3) // проверка на длину

{

Prufung(); // оповещение о некорректном вводе

system("cls"); // очистка консоли

goto gt\_ZVerSch; // строка 362

}

// проверка на символы

for (int i = 0; i < strlen(Zeitspanne); i++)

if ((Zeitspanne[i] < 65) || (Zeitspanne[i] > 90))

if ((Zeitspanne[i] < 97) || (Zeitspanne[i] > 122))

{

Prufung(); // оповещение о некорректном вводе

system("cls"); // очистка консоли

goto gt\_ZVerSch; // строка 362

}

// коррекция введённых данных

KorrigierenDaten(Zeitspanne, 2);

// определение кода валюты

strcpy\_s(land[metaindex].VAL, Zeitspanne);

// рассчёт плотносит населения

land[metaindex].Nas\_pro\_Terr = land[metaindex].Nas /

land[metaindex].Terr;

metaindex++; // изменение рабочего индекса добавления

land[metaindex - 1].index = metaindex; // индексация

x--; // уменьшение числа записей

break; // выход из цикла for

}

else

{

swap(land[i], land[i - 1]); // сортировка

land[i].index++; // индексация

}

}

if (x >= 1) // проверка количества введённых записей

cout << "\nСделайте ещё одну запись\n\n";

}

}

// перезапись бинарного файла

reisen.open(DerNameDesTextes, ios::binary | ios::out); // открытие бинарног файла

reisen.write((char\*)&Kapazitat, sizeof(Kapazitat)); // запись рабочей строки

for (int i = 0; i < Kapazitat; i++) // запись изменённой последовательности индексов

reisen.write((char\*)&land[i], sizeof(land[i]));

reisen.close(); // закрытие файла

// изменение индексно-бинарных файлов

int K = Kapazitat; // для передачи в функцию

SortierungNachVeranderungen(land, K);

Wartung(); // организация задержки

}

**Statik. h**

#pragma once

#include "lender.h" // подключение структуры

void SreibenDaten(lander\*& land , int& Kapazitat); // вывод записей

void WahlDaten(lander\*& land , int& Kapazitat); // выборка данных

**Statik.cpp**

#include <iostream> // для ввода и вывода

#include <conio.h> // для \_getche()

#include "Statik.h" // подключение одноимённого заголовочного файла

#include "Peripherie.h" // подключение кода файла периферии

using namespace std; // подключение пространства имён std

void SreibenDaten(lander\*& land , int& Kapazitat) // для вывода элементов массива на экран

{

system("cls"); cout << "Вывод содержимого файла: \n\n";

Mutze(); // вывод шапки

for (int i = 0; i < Kapazitat; i++) // вывод элементов массива

Schreiben(land, i); // непосредственный вывод i-того элемента массива структур

cout << endl; // для эстетики

Wartung(); // организация зарежки

}

void WahlDaten(lander\*& land , int& Kapazitat) // функция выборки

{

Haupt\_Wahl:; // строки 35, 47, 56

// рабочие переменные

system("cls");

// переменные границ промежутка выборки

unsigned int NiedrigeGrenze = 0, HoheGrenze = 0;

int leuchtturm = 0; // маячок для подсчёта совпадений

char Zeitspanne[64]; // защитный буфер - защита от некорректного ввода

cout << "Укажите промежуток численности населения для выборки\n";

// ввод нижней границы

cout << "от "; cin >> Zeitspanne;

// проверка нижней границы

for (int i = 0; i < strlen(Zeitspanne); i++) // проверка на символы

if ((Zeitspanne[i] < 48) || (Zeitspanne[i] > 57))

{

Prufung(); // оповещение о некорректных данных

goto Haupt\_Wahl; // строка 20

}

for (int i = 0; i < strlen(Zeitspanne); i++) // определение нижней границы промежутка

NiedrigeGrenze += ((Zeitspanne[i] - 48) \* pow(10, (strlen(Zeitspanne) - 1 - i)));

// ввод верхней границы

cout << "до "; cin >> Zeitspanne;

// проверка верхней границы

for (int i = 0; i < strlen(Zeitspanne); i++) // проверка на символы

if ((Zeitspanne[i] < 48) || (Zeitspanne[i] > 57))

{

Prufung(); // оповещение о некорректных данных

goto Haupt\_Wahl; // строка 20

}

for (int i = 0; i < strlen(Zeitspanne); i++) // определение верхней границы промежутка

HoheGrenze += ((Zeitspanne[i] - 48) \* pow(10, (strlen(Zeitspanne) - 1 - i)));

// общая проверка

if (NiedrigeGrenze >= HoheGrenze)

{

Prufung(); // оповещение о некорректных данных

goto Haupt\_Wahl; // строка 20

}

// проверка совпадений

for (int i = 0; i < Kapazitat; i++)

if ((land[i].Nas > NiedrigeGrenze) && (land[i].Nas < HoheGrenze))

leuchtturm++; // при совпадении идёт увеличение маячка

// основной ход выборки

if (leuchtturm == 0) // в случае, если совпадений не было

cout << "\nНет записей, удовлетворяющих условию!\n\n";

else // в случае, если совпадения есть

{

cout << "\nСтраны, принадлежащие промежутку численности населения [" <<

NiedrigeGrenze << ";" << HoheGrenze << "]:\n\n";

Mutze(); // вывод шапки

for (int i = 0; i < Kapazitat; i++) // вывод элементов массива

// условия вывода элементов массива

if ((land[i].Nas > NiedrigeGrenze) && (land[i].Nas < HoheGrenze))

Schreiben(land, i); // непосредственный вывод i-той записи

// вывод количесва совпадений

cout << "\nВсего записей, удовлетворяющих условию: " << leuchtturm << "\n\n";

}

Wartung(); // задержка

}

**Sortierung.h**

#pragma once

#include "lender.h" // подключение структуры

void SortieranDaten(lander\*& land , int& Kapazitat); // сортировка

int HeranSortierung(lander\*& land , int i, int j); // прямой ход сортировки

void ZuruckSortierung(lander\*& land , int& Kapazitat); // обратный ход сортировки

void SortierungNachVeranderungen(lander\*& land , int& K); // изденение инд.-бин. файлов

**Sortierung.cpp**

#include <iostream> // для ввода и вывода

#include <iomanip> // для setw()

#include <conio.h> // для \_getche()

#include <stdio.h> // для работы с файлами

#include <fstream> // для работы с файлами

#include "Peripherie.h" // подключение кода файла периферии

#include "Sortierung.h" // подключение одноимённого заголовочного файла

using namespace std; // подключение пространства имён std

// вынесенные прототипы функций для взаимного использования функций, хранящихся в этом модуле

int HeranSortierung(lander\*& land , int i, int j); // прямой ход сортировки

void ZuruckSortierung(lander\*& land , int& Kapazitat); // обратный ход сортировки

void SortieranDaten(lander\*& land , int& Kapazitat) // сортировка данных

{

Haupt\_Sortierung:; // строка 151

// рабочие переменные

ifstream IndexFileR; // для чтения индексно-бинарных файлов

int K = Kapazitat; // для передачи размерности по ссылке в другие функции

int\* metamass = new int[Kapazitat]; // для чтения индексов из индексно-бинарных файлов

char laufgang = '8'; // для хода по меню

// ход сортировки

while (laufgang != '0') // организация повторяющегося меню

{

system("cls"); // очистка консоли

cout << "Меню сортировки:\n\n" // меню сортировки

<< "1 - сортировка по названию страны (по алфавиту)\n"

<< "2 - сортировка по названию столицы страны (по алфавиту)\n"

<< "3 - сортировка по численности населения страны (по убыванию)\n"

<< "4 - сортировка по площади страны (по убыванию)\n"

<< "5 - сортировка по плотности населения страны (по убыванию)\n"

<< "6 - сортировка по коду валюты страны (по алфавиту)\n"

<< "0 - выход из меню сортировки\n"

<< "\nВыбор пункта меню: ";

laufgang = \_getche(); // выбор пункта меню

switch (laufgang) // распределение по меню

{

case '1':

system("cls"); // очистка консоли

// открытие инд.-бин. файла в бинарном режиме для чтения

IndexFileR.open("Sort\_Name\_Bin.txt", ios::binary | ios::out);

for (int i = 0; i < Kapazitat; i++) // чтение данных из файла

IndexFileR.read((char\*)&metamass[i], sizeof(metamass[i]));

IndexFileR.close(); // закрытие индексно-бинарного файла

// вывод сортировки

Mutze(); // вывод шапки

for (int i = 0; i < Kapazitat; i++) // вывод отсортированного массива

for (int j = 0; j < Kapazitat; j++)

if (metamass[i] == land[j].index) // условие вывода

Schreiben(land, j);

cout << endl; // для эстетики

Wartung(); // организация задержки

break;

case '2':

system("cls"); // очистка консоли

// открытие инд.-бин. файла в бинарном режиме для чтения

IndexFileR.open("Sort\_Hauptstadt\_Bin.txt", ios::binary | ios::out);

for (int i = 0; i < Kapazitat; i++) // чтение данных из файла

IndexFileR.read((char\*)&metamass[i], sizeof(metamass[i]));

IndexFileR.close(); // закрытие индексно-бинарного файла

// вывод сортировки

Mutze(); // вывод шапки

for (int i = 0; i < Kapazitat; i++) // вывод отсортированного массива

for (int j = 0; j < Kapazitat; j++)

if (metamass[i] == land[j].index) // условие вывода

Schreiben(land, j);

cout << endl; // для эстетики

Wartung(); // организация задержки

break;

case '3':

system("cls"); // очистка консоли

// открытие инд.-бин. файла в бинарном режиме для чтения

IndexFileR.open("Sort\_Nas\_Bin.txt", ios::binary | ios::out);

for (int i = 0; i < Kapazitat; i++) // чтение данных из файла

IndexFileR.read((char\*)&metamass[i], sizeof(metamass[i]));

IndexFileR.close(); // закрытие индексно-бинарного файла

// вывод сортировки

Mutze(); // вывод шапки

for (int i = 0; i < Kapazitat; i++) // вывод отсортированного массива

for (int j = 0; j < Kapazitat; j++)

if (metamass[i] == land[j].index) // условие вывода

Schreiben(land, j);

cout << endl; // для эстетики

Wartung(); // организация задержки

break;

case '4':

system("cls"); // очистка консоли

// открытие инд.-бин. файла в бинарном режиме для чтения

IndexFileR.open("Sort\_Terr\_Bin.txt", ios::binary | ios::out);

for (int i = 0; i < Kapazitat; i++) // чтение данных из файла

IndexFileR.read((char\*)&metamass[i], sizeof(metamass[i]));

IndexFileR.close(); // закрытие индексно-бинарного файла

// вывод сортировки

Mutze(); // вывод шапки

for (int i = 0; i < Kapazitat; i++) // вывод отсортированного массива

for (int j = 0; j < Kapazitat; j++)

if (metamass[i] == land[j].index) // условие вывода

Schreiben(land, j);

cout << endl; // для эстетики

Wartung(); // организация задержки

break;

case '5':

system("cls"); // очистка консоли

// открытие инд.-бин. файла в бинарном режиме для чтения

IndexFileR.open("Sort\_Nas\_pro\_Terr\_Bin.txt", ios::binary | ios::out);

for (int i = 0; i < Kapazitat; i++) // чтение данных из файла

IndexFileR.read((char\*)&metamass[i], sizeof(metamass[i]));

IndexFileR.close(); // закрытие индексно-бинарного файла

// вывод сортировки

Mutze(); // вывод шапки

for (int i = 0; i < Kapazitat; i++) // вывод отсортированного массива

for (int j = 0; j < Kapazitat; j++)

if (metamass[i] == land[j].index) // условие вывода

Schreiben(land, j);

cout << endl; // для эстетики

Wartung(); // организация задержки

break;

case '6':

system("cls"); // очистка консоли

// открытие инд.-бин. файла в бинарном режиме для чтения

IndexFileR.open("Sort\_VAL\_Bin.txt", ios::binary | ios::out);

for (int i = 0; i < Kapazitat; i++) // чтение данных из файла

IndexFileR.read((char\*)&metamass[i], sizeof(metamass[i]));

IndexFileR.close(); // закрытие индексно-бинарного файла

// вывод сортировки

Mutze(); // вывод шапки

for (int i = 0; i < Kapazitat; i++) // вывод отсортированного массива

for (int j = 0; j < Kapazitat; j++)

if (metamass[i] == land[j].index) // условие вывода

Schreiben(land, j);

cout << endl; // для эстетики

Wartung(); // организация задержки

break;

case '0': break; // для быстрого выхода из меню

default:

Prufung(); // предупреждение о некорректном вводе

goto Haupt\_Sortierung; // строка 17

break;

}

}

system("cls"); // очистка консоли

}

int HeranSortierung(lander\*& land , int i, int j) // прямой ход ортировки

{

swap(land[i], land[j]); // перестановка значений

return 0; // возвращение обнулённого счётчика обхода

}

void ZuruckSortierung(lander\*& land , int& Kapazitat) // обратный ход сортировки

{

for (int i = 0; i < Kapazitat; i++) // сортировка по индексу

for (int j = i + 1; j < Kapazitat; j++) // другими словами

if (land[i].index > land[j].index) // возврат исходного порядка

i = HeranSortierung(land, i, j); // прямой ход в обратном ходе

cout << endl; // для эстетики

}

void SortierungNachVeranderungen(lander\*& land , int& K) // сортировка индексно-бинарных файлов после изменения данных

{

ofstream IndexReisen; // переменная для изменения индексных файлов

// сортировка по названию страны

for (int i = 0; i < K; i++)

for (int j = i + 1; j < K; j++)

// стравнение переменных для сортировки

if (strcmp(land[i].Name, land[j].Name) > 0)

// непосредственная сортировка

i = HeranSortierung(land, i, j);

// открытие соответствующего индексного-бинарного файла и его перезапись

IndexReisen.open("Sort\_Name\_Bin.txt", ios::binary | ios::out);

for (int i = 0; i < K; i++) // запись изменённой последовательности индексов

IndexReisen.write((char\*)&land[i].index, sizeof(land[i].index));

IndexReisen.close(); // закрытие файла

// сортировка по названию столицы

for (int i = 0; i < K; i++)

for (int j = i + 1; j < K; j++)

// стравнение переменных для сортировки

if (strcmp(land[i].HauptStadt, land[j].HauptStadt) > 0)

// непосредственная сортировка

i = HeranSortierung(land, i, j);

// открытие соответствующего индексного-бинарного файла и его перезапись

IndexReisen.open("Sort\_Hauptstadt\_Bin.txt", ios::binary | ios::out);

for (int i = 0; i < K; i++) // запись изменённой последовательности индексов

IndexReisen.write((char\*)&land[i].index, sizeof(land[i].index));

IndexReisen.close(); // закрытие файла

// сортировка по численности населения страны

for (int i = 0; i < K; i++)

for (int j = i + 1; j < K; j++)

if (land[i].Nas < land[j].Nas) // стравнение переменных для сортировки

// непосредственная сортировка

i = HeranSortierung(land, i, j);

// открытие соответствующего индексного-бинарного файла и его перезапись

IndexReisen.open("Sort\_Nas\_Bin.txt", ios::binary | ios::out);

for (int i = 0; i < K; i++) // запись изменённой последовательности индексов

IndexReisen.write((char\*)&land[i].index, sizeof(land[i].index));

IndexReisen.close(); // закрытие файла

// сортировка по площади страны

for (int i = 0; i < K; i++)

for (int j = i + 1; j < K; j++)

// стравнение переменных для сортировки

if (land[i].Terr < land[j].Terr)

// непосредственная сортировка

i = HeranSortierung(land, i, j);

// открытие соответствующего индексного-бинарного файла и его перезапись

IndexReisen.open("Sort\_Terr\_Bin.txt", ios::binary | ios::out);

for (int i = 0; i < K; i++) // запись изменённой последовательности индексов

IndexReisen.write((char\*)&land[i].index, sizeof(land[i].index));

IndexReisen.close(); // закрытие файла

// сортировка по плотности населения

for (int i = 0; i < K; i++)

for (int j = i + 1; j < K; j++)

// стравнение переменных для сортировки

if (land[i].Nas\_pro\_Terr < land[j].Nas\_pro\_Terr)

// непосредственная сортировка

i = HeranSortierung(land, i, j);

// открытие соответствующего индексного-бинарного файла и его перезапись

IndexReisen.open("Sort\_Nas\_pro\_Terr\_Bin.txt", ios::binary | ios::out);

for (int i = 0; i < K; i++) // запись изменённой последовательности индексов

IndexReisen.write((char\*)&land[i].index, sizeof(land[i].index));

IndexReisen.close(); // закрытие файла

// сортировка по коду валюты страны

for (int i = 0; i < K; i++)

for (int j = i + 1; j < K; j++)

// стравнение переменных для сортировки

if (strcmp(land[i].VAL, land[j].VAL) > 0)

// непосредственная сортировка

i = HeranSortierung(land, i, j);

// открытие соответствующего индексного-бинарного файла и его перезапись

IndexReisen.open("Sort\_VAL\_Bin.txt", ios::binary | ios::out);

for (int i = 0; i < K; i++) // запись изменённой последовательности индексов

IndexReisen.write((char\*)&land[i].index, sizeof(land[i].index));

IndexReisen.close(); // закрытие файла

// возвращение данных в исходный порядок

ZuruckSortierung(land, K);

}

**Suchen.h**

#pragma once

#include "lender.h" // подключение структуры

void SuchenDaten(lander\*& land , int& Kapazitat); // поиск данных

int SuchenFurSubstragierung(lander\*& land , int& Kapazitat); // поиск для удаления

**Suchen.cpp**

#include <iostream> // для ввода и вывода

#include <conio.h> // для \_getche()

#include "Peripherie.h" // подключение

#include "Suchen.h" // подключение одноимённого заголовочного файла

using namespace std; // подключение пространства имён std

void SuchenDaten(lander\*& land , int& Kapazitat) // поиск данных

{

char laufgang = 'd'; // для хода по меню

while (laufgang != '0')

{

Such:; // строка 279

// рабочие переменные

char Zeitspanne[64]; // защитный буфер - защита от некорректного ввода

int leuchtturm = 0; // маячок для подчсчётав совпадений

int metaint = 0; // переменная для числового выбора

// меню поиска для удаления

system("cls"); // очистка экрана

cout << "По какому полю будет проходить поиск?\n\n"

<< "1 - по индексу\n"

<< "2 - по названию страны\n"

<< "3 - по назманию столицы\n"

<< "4 - по численности населения\n"

<< "5 - по площади\n"

<< "6 - по плотности населения\n"

<< "7 - по валюте\n"

<< "0 - выход в меню\n\n"

<< "Пункт меню: ";

laufgang = \_getche(); // выбор пункта меню

// ход по меню

switch (laufgang)

{

case '1': // поиск по индексу

Such\_Ind:; // строка 48

metaint = 0; // обнуление для определения

system("cls"); // очистка консоли

cout << "Введите поисковую информацию по полю \"идекс поля\": ";

cin >> Zeitspanne;

for (int i = 0; i < strlen(Zeitspanne); i++) // проверка на символы

if ((Zeitspanne[i] < 48) || (Zeitspanne[i] > 57))

{

Prufung(); // оповещение о некорректном вводе

goto Such\_Ind; // строка 39

}

// определение переменной сравнения

for (int i = 0; i < strlen(Zeitspanne); i++)

metaint += ((Zeitspanne[i] - 48) \* pow(10, (strlen(Zeitspanne)

- 1 - i)));

for (int i = 0; i < Kapazitat; i++) // посчёт совпадений

if (metaint == land[i].index) // условие совпадений

leuchtturm++; // увеличение совпадений

if (leuchtturm == 0) // если совпадений нет

cout << "\nДаные не найдены!\n";

else // если совпадения есть

{

cout << endl;

Mutze(); // вывод элементов шапки

for (int i = 0; i < Kapazitat; i++) // вывод эдементов массива

if (metaint == land[i].index) // условия вывода

Schreiben(land, i); // вывод i-той записи

}

cout << endl;

Wartung(); // задержка

break;

case '2': // поиск по названию страны

Such\_Name:; // строка 78, 86

system("cls"); // очистка консоли

cout << "Введите поисковую информацию по полю \"название страны\": ";

cin >> Zeitspanne;

if (strlen(Zeitspanne) > 25) // проверка на длину

{

Prufung(); // оповещение о некорректном вводе

goto Such\_Name; // строка 71

}

for (int i = 0; i < strlen(Zeitspanne); i++) // проверка на символы

if ((Zeitspanne[i] < 65) || (Zeitspanne[i] > 90))

if ((Zeitspanne[i] < 97) || (Zeitspanne[i] > 122))

if ((Zeitspanne[i] != 95) && (Zeitspanne[i] != 45))

{

Prufung(); // оповещение о некорректном вводе

goto Such\_Name; // строка 71

}

KorrigierenDaten(Zeitspanne, 1); // коррекция введённых данных

for (int i = 0; i < Kapazitat; i++) // посчёт совпадений

// условие совпадений

if (strcmp(Zeitspanne, land[i].Name) == 0)

leuchtturm++; // увеличение совпадений

if (leuchtturm == 0) // если совпадений нет

cout << "\nДаные не найдены!\n";

else // если совпадения есть

{

cout << endl;

Mutze(); // вывод элементов шапки

for (int i = 0; i < Kapazitat; i++) // вывод эдементов массива

// условия вывода

if (strcmp(Zeitspanne, land[i].Name) == 0)

Schreiben(land, i); // вывод i-той записи

}

cout << endl;

Wartung(); // задержка

break;

case '3': // поиск по названию столицы

Such\_HauptStadt:; // строка 115, 123

system("cls"); // очистка консоли

cout << "Введите поисковую информацию по полю \"название столицы\": ";

cin >> Zeitspanne;

if (strlen(Zeitspanne) > 25) // проверка на длину

{

Prufung(); // оповещение о некорректном вводе

goto Such\_HauptStadt; // строка 108

}

for (int i = 0; i < strlen(Zeitspanne); i++) // проверка на символы

if ((Zeitspanne[i] < 65) || (Zeitspanne[i] > 90))

if ((Zeitspanne[i] < 97) || (Zeitspanne[i] > 122))

if ((Zeitspanne[i] != 95) && (Zeitspanne[i] != 45))

{

Prufung(); // оповещение о некорректном вводе

goto Such\_HauptStadt; // стркоа 108

}

KorrigierenDaten(Zeitspanne, 1); // коррекция введённых данных

for (int i = 0; i < Kapazitat; i++) // посчёт совпадений

// условие совпадений

if (strcmp(Zeitspanne, land[i].HauptStadt) == 0)

leuchtturm++; // увеличение совпадений

if (leuchtturm == 0) // если совпадений нет

cout << "\nДаные не найдены!\n";

else // если совпадения есть

{

cout << endl;

Mutze(); // вывод элементов шапки

for (int i = 0; i < Kapazitat; i++) // вывод эдементов массива

// условия вывода

if (strcmp(Zeitspanne, land[i].HauptStadt) == 0)

Schreiben(land, i); // вывод i-той записи

}

cout << endl;

Wartung(); // задержка

break;

case '4': // поиск по численности населения

Such\_Nas:; // строка 154

metaint = 0; // обнуление для определения

system("cls"); // очистка консоли

cout << "Введите поисковую информацию по полю \"идекс поля\": ";

cin >> Zeitspanne;

for (int i = 0; i < strlen(Zeitspanne); i++) // проверка на символы

if ((Zeitspanne[i] < 48) || (Zeitspanne[i] > 57))

{

Prufung(); // оповещение о некорректном вводе

goto Such\_Nas; // строка 145

}

// определение переменной сравнения

for (int i = 0; i < strlen(Zeitspanne); i++)

metaint += ((Zeitspanne[i] - 48) \* pow(10, (strlen(Zeitspanne)

- 1 - i)));

for (int i = 0; i < Kapazitat; i++) // посчёт совпадений

if (metaint == land[i].Nas) // условие совпадений

leuchtturm++; // увеличение совпадений

if (leuchtturm == 0) // если совпадений нет

cout << "\nДаные не найдены!\n";

else // если совпадения есть

{

cout << endl;

Mutze(); // вывод элементов шапки

for (int i = 0; i < Kapazitat; i++) // вывод эдементов массива

if (metaint == land[i].Nas) // условия вывода

Schreiben(land, i); // вывод i-той записи

}

cout << endl;

Wartung(); // задержка

break;

case '5': // поиск по площади

Such\_Terr:; // строка 186

metaint = 0; // обнуление для определения

system("cls"); // очистка консоли

cout << "Введите поисковую информацию по полю \"идекс поля\": ";

cin >> Zeitspanne;

for (int i = 0; i < strlen(Zeitspanne); i++) // проверка на символы

if ((Zeitspanne[i] < 48) || (Zeitspanne[i] > 57))

{

Prufung(); // оповещение о некорректном вводе

goto Such\_Terr; // строка 177

}

// определение переменной сравнения

for (int i = 0; i < strlen(Zeitspanne); i++)

metaint += ((Zeitspanne[i] - 48) \* pow(10, (strlen(Zeitspanne)

- 1 - i)));

for (int i = 0; i < Kapazitat; i++) // посчёт совпадений

if (metaint == land[i].Terr) // условие совпадений

leuchtturm++; // увеличение совпадений

if (leuchtturm == 0) // если совпадений нет

cout << "\nДаные не найдены!\n";

else // если совпадения есть

{

cout << endl;

Mutze(); // вывод элементов шапки

for (int i = 0; i < Kapazitat; i++) // вывод эдементов массива

if (metaint == land[i].Terr) // условия вывода

Schreiben(land, i); // вывод i-той записи

}

cout << endl;

Wartung(); // задержка

break;

case '6': // поиск по плотности населения

Such\_Nas\_pro\_Terr:; // строка 218

metaint = 0; // обнуление для определения

system("cls"); // очистка консоли

cout << "Введите поисковую информацию по полю \"идекс поля\": ";

cin >> Zeitspanne;

for (int i = 0; i < strlen(Zeitspanne); i++) // проверка на символы

if ((Zeitspanne[i] < 48) || (Zeitspanne[i] > 57))

{

Prufung(); // оповещение о некорректном вводе

goto Such\_Nas\_pro\_Terr; // строка 208

}

// определение переменной сравнения

for (int i = 0; i < strlen(Zeitspanne); i++)

metaint += ((Zeitspanne[i] - 48) \* pow(10, (strlen(Zeitspanne)

- 1 - i)));

for (int i = 0; i < Kapazitat; i++) // посчёт совпадений

if (metaint == land[i].Nas\_pro\_Terr) // условие совпадений

leuchtturm++; // увеличение совпадений

if (leuchtturm == 0) // если совпадений нет

cout << "\nДаные не найдены!\n";

else // если совпадения есть

{

cout << endl;

Mutze(); // вывод элементов шапки

for (int i = 0; i < Kapazitat; i++) // вывод эдементов массива

if (metaint == land[i].Nas\_pro\_Terr) // условия вывода

Schreiben(land, i); // вывод i-той записи

}

cout << endl;

Wartung(); // задержка

break;

case '7': // поиск по коду валюты

Such\_VAL:; // строка 248, 255

system("cls"); // очистка консоли

cout << "Введите поисковую информацию по полю \"код валюты\": ";

cin >> Zeitspanne;

if (strlen(Zeitspanne) != 3) // проверка на длину

{

Prufung(); // оповещение о некорректном вводе

goto Such\_VAL; // строка 241

}

for (int i = 0; i < strlen(Zeitspanne); i++) // проверка на символы

if ((Zeitspanne[i] < 65) || (Zeitspanne[i] > 90))

if ((Zeitspanne[i] < 97) || (Zeitspanne[i] > 122))

{

Prufung(); // оповещение о некорректном вводе

goto Such\_VAL; // строка 241

}

KorrigierenDaten(Zeitspanne, 2); // коррекция введённых данных

for (int i = 0; i < Kapazitat; i++) // посчёт совпадений

if (strcmp(Zeitspanne, land[i].VAL) == 0)

leuchtturm++; // увеличение совпадений

if (leuchtturm == 0) // если совпадений нет

cout << "\nДаные не найдены!\n";

else // если совпадения есть

{

cout << endl;

Mutze(); // вывод элементов шапки

for (int i = 0; i < Kapazitat; i++) // вывод эдементов массива

if (strcmp(Zeitspanne, land[i].VAL) == 0)

Schreiben(land, i); // вывод i-той записи

}

cout << endl;

Wartung(); // задержка

break;

case '0': break; // для выхода

default: // проверка ввода

Prufung(); // оповещение о некорректном вводе

goto Such; // строка 13

}

}

}

int SuchenFurSubstragierung(lander\*& land, int& Kapazitat) // поиск для удаления

{

Such\_Fur\_Sub\_Hauptablauf:; // стркоа 852

// рабочие переменные

char laufgang; // ход по меню

char Zeitspanne[64]; // защитный буфер - защита от некорректного ввода

char TeilDel[64]; // защитный буфер - защита для рабочего индекса

char Antwort; // переменная ответа

int leuchtturm = 0; // маячок для постчёта совпадений

int metaint = 0; //

int metaindex = 0; // для взврата индекса удаления

// меню поиска для удаления

system("cls"); // очистка экрана

cout << "По какому полю будет проходить поиск для удаления?\n\n"

<< "1 - по названию страны\n"

<< "2 - по назманию столицы\n"

<< "3 - по численности населения\n"

<< "4 - по площади\n"

<< "5 - по плотности населения\n"

<< "6 - по коду валюты\n"

<< "0 - выход из поиска\n\n"

<< "Пункт меню: ";

laufgang = \_getche(); // ввод пункта меню

if (laufgang == 0) return 0; // выход и возврат ноль-индекса

// ход по меню

switch (laufgang)

{

case '1': // поиск по названию страны

Such\_Fur\_Sub\_Anfang\_Name:; // строки 323, 321

system("cls"); // очистка консоли

cout << "Введите поисковую информацию по полю \"название страны\": ";

cin >> Zeitspanne;

if (strlen(Zeitspanne) > 25) // проверка на длину

{

Prufung(); // оповещение о некорректном вводе

goto Such\_Fur\_Sub\_Anfang\_Name; // строка 316

}

for (int i = 0; i < strlen(Zeitspanne); i++) // проверка на символы

if ((Zeitspanne[i] < 65) || (Zeitspanne[i] > 90))

if ((Zeitspanne[i] < 97) || (Zeitspanne[i] > 122))

if ((Zeitspanne[i] != 95) && (Zeitspanne[i] != 45))

{

Prufung(); // оповещение о некорректном вводе

goto Such\_Fur\_Sub\_Anfang\_Name; // строка 316

}

KorrigierenDaten(Zeitspanne, 1); // коррекция введённых данных

// работа с мачком совпадений

for (int i = 0; i < Kapazitat; i++)

// сравнение с действительными данными

if (strcmp(Zeitspanne, land[i].Name) == 0)

leuchtturm++; // увеличение маячка совпадений

if (leuchtturm == 0) // если совпадений нет

{

cout << "\nДаные не найдены!\n\n"; // оповещение о ненайденных данных

Wartung(); // задержка

return 0; // вовзрат ноль-индекса

}

else if (leuchtturm == 1) // если совпадение одно

{

cout << endl;

Such\_Fur\_Sub\_EinLT\_Name:; // строка 362

Mutze(); // вывод элементов шапки

for (int i = 0; i < Kapazitat; i++)

// нахождение строки сопадения

if (strcmp(Zeitspanne, land[i].Name) == 0)

{

Schreiben(land, i); // вывод cтроки совпадения

metaindex = land[i].index; // запоминание индекса

}

//запрос на удаление записи

cout << "\nУдалить эту запись? (y или n): ";

Antwort = \_getche(); // ответ

if ((Antwort != 'y') && (Antwort != 'n')) // проверка ввода

{

Prufung(); // оповещение о некорректном вводе

system("cls"); // очистка консоли

goto Such\_Fur\_Sub\_EinLT\_Name; // строка 348

}

else if (Antwort == 'y') // если ответ положительный

return metaindex; // возвращаем рабочий индекс

else if (Antwort == 'n') // если ответ отрицательный

return 0; // возвращаем ноль-индекс

}

else if (leuchtturm > 1) // если совпадений несколько

{

cout << endl;

Such\_Fur\_Sub\_VielLT\_Name:; // строки 384, 397

Mutze(); // вывод шапки

for (int i = 0; i < Kapazitat; i++) // вывод элементов массива

if (strcmp(Zeitspanne, land[i].Name) == 0)

Schreiben(land, i); // вывод cтроки совпадения

cout << "\nВведите индекс записи, которую необходимо удалить (0 –

никакую): "; // запрос на удаление

cin >> TeilDel;

for (int i = 0; i < strlen(TeilDel); i++) // проверка на символы

if ((TeilDel[i] < 48) || (TeilDel[i] > 57))

{

Prufung(); // оповещение о некорректном вводе

system("cls"); // очистка консоли

goto Such\_Fur\_Sub\_VielLT\_Name; // строка 372

}

// выход в меню удаления

if ((strlen(TeilDel) == 1) && (TeilDel[0] == 48)) return 0;

// определение рабочего сивола

for (int i = 0; i < strlen(TeilDel); i++)

metaindex += ((TeilDel[i] - 48) \* pow(10, (strlen(TeilDel) - 1

- i)));

leuchtturm = 0; // обнуление маячка совпадений

for (int i = 0; i < Kapazitat; i++) // подсчёт совпадений

if (strcmp(Zeitspanne, land[i].Name) == 0)

leuchtturm++; // увеличение маячка совпадений

if (leuchtturm == 0) // если совпадений нет

{

Prufung(); // оповещение о некорректном вводе

system("cls"); // очистка консоли

goto Such\_Fur\_Sub\_VielLT\_Name; // строка 372

}

else // если совпадения есть

for (int i = 0; i < Kapazitat; i++)

if (metaindex == land[i].index)

return metaindex; // возврат рабочего индекса

}

case '2': // поиск по названию столицы

Such\_Fur\_Sub\_Anfang\_Hauptstadt:; // строки 414, 422

system("cls"); // очистка консоли

cout << "Введите поисковую информацию по полю \"название столицы\": ";

cin >> Zeitspanne;

if (strlen(Zeitspanne) > 25) // проверка на длину

{

Prufung(); // оповещение о некорректном вводе

goto Such\_Fur\_Sub\_Anfang\_Hauptstadt; // стрка 407

}

for (int i = 0; i < strlen(Zeitspanne); i++) // проверка на символы

if ((Zeitspanne[i] < 65) || (Zeitspanne[i] > 90))

if ((Zeitspanne[i] < 97) || (Zeitspanne[i] > 122))

if ((Zeitspanne[i] != 95) && (Zeitspanne[i] != 45))

{

Prufung(); // оповещение о некорректном вводе

goto Such\_Fur\_Sub\_Anfang\_Hauptstadt; // стрка 407

}

KorrigierenDaten(Zeitspanne, 1); // коррекция введённых данных

// работа с мачком совпадений

for (int i = 0; i < Kapazitat; i++)

// сравнение с действительными данными

if (strcmp(Zeitspanne, land[i].HauptStadt) == 0)

leuchtturm++; // увеличение маячка совпадений

if (leuchtturm == 0) // если совпадений нет

{

cout << "\nДаные не найдены!\n\n"; // оповещение о ненайденных данных

Wartung(); // задержка

return 0; // вовзрат ноль-индекса

}

else if (leuchtturm == 1) // если совпадение одно

{

cout << endl;

Such\_Fur\_Sub\_EinLT\_Hauptstadt:; // строка 453

Mutze(); // вывод элементов шапки

for (int i = 0; i < Kapazitat; i++)

// нахождение строки сопадения

if (strcmp(Zeitspanne, land[i].HauptStadt) == 0)

{

Schreiben(land, i); // вывод cтроки совпадения

metaindex = land[i].index; // запоминание индекса

}

//запрос на удаление записи

cout << "\nУдалить эту запись? (y или n): ";

Antwort = \_getche(); // ответ

if ((Antwort != 'y') && (Antwort != 'n')) // проверка ввода

{

Prufung(); // оповещение о некорректном вводе

system("cls"); // очистка консоли

goto Such\_Fur\_Sub\_EinLT\_Hauptstadt; // строка 439

}

else if (Antwort == 'y') // если ответ положительный

return metaindex; // возвращаем рабочий индекс

else if (Antwort == 'n') // если ответ отрицательный

return 0; // возвращаем ноль-индекс

}

else if (leuchtturm > 1) // если совпадений несколько

{cout << endl;

Such\_Fur\_Sub\_VielLT\_Hauptstadt:; // строки 475, 488

Mutze(); // вывод шапки

for (int i = 0; i < Kapazitat; i++) // вывод элементов массива

if (strcmp(Zeitspanne, land[i].HauptStadt) == 0)

Schreiben(land, i); // вывод cтроки совпадения

cout << "\nВведите индекс записи, которую необходимо удалить (0 –

никакую): "; // запрос на удаление

cin >> TeilDel;

for (int i = 0; i < strlen(TeilDel); i++) // проверка на символы

if ((TeilDel[i] < 48) || (TeilDel[i] > 57))

{

Prufung(); // оповещение о некорректном вводе

system("cls"); // очистка консоли

goto Such\_Fur\_Sub\_VielLT\_Hauptstadt; // строка 463

}

// выход в меню удаления

if ((strlen(TeilDel) == 1) && (TeilDel[0] == 48)) return 0;

// определение рабочего сивола

for (int i = 0; i < strlen(TeilDel); i++)

metaindex += ((TeilDel[i] - 48) \* pow(10, (strlen(TeilDel) - 1

- i)));

leuchtturm = 0; // обнуление маячка совпадений

for (int i = 0; i < Kapazitat; i++) // подсчёт совпадений

if (strcmp(Zeitspanne, land[i].HauptStadt) == 0)

leuchtturm++;

if (leuchtturm == 0) // если совпадений нет

{

Prufung(); // оповещение о некорректном вводе

system("cls"); // очистка консоли

goto Such\_Fur\_Sub\_VielLT\_Hauptstadt; // строка 463

}

else

for (int i = 0; i < Kapazitat; i++)

if (metaindex == land[i].index)

return metaindex; // возврат рабочего индекса

}

case '3': // посик по численности населения

Such\_Fur\_Sub\_Anfang\_Nas:; // строка 511

system("cls"); // очистка консоли

cout << "Введите поисковую информацию по полю \"численность населения\": ";

cin >> Zeitspanne;

if (strlen(Zeitspanne) > 10) // проверка на длину

{

Prufung(); // оповещение о некорректном вводе

goto Such\_Fur\_Sub\_Anfang\_Nas;

}

for (int i = 0; i < strlen(Zeitspanne); i++) // проверка на символы

if ((Zeitspanne[i] < 48) || (Zeitspanne[i] > 57))

{

Prufung(); // оповещение о некорректном вводе

goto Such\_Fur\_Sub\_Anfang\_Nas; // строка 496

}

for (int i = 0; i < strlen(Zeitspanne); i++) // определение ключевого числа сравнения

metaint += ((Zeitspanne[i] - 48) \* pow(10, (strlen(Zeitspanne) - 1 –

i)));

for (int i = 0; i < Kapazitat; i++)

if (metaint == land[i].Nas) // условие увеличения числа совпадений

leuchtturm++; // увеличение маячка совпадений

if (leuchtturm == 0) // если совпадений нет

{

cout << "\nДаные не найдены!\n\n"; // опвещение, что данные не найдены

Wartung(); // задержка

return 0; // возврат ноль-индекса

}

else if (leuchtturm == 1) // если одно совпадение

{

cout << endl;

Such\_Fur\_Sub\_EinLT\_Nas:; // строка 541

Mutze(); // вывод элементов шапки

for (int i = 0; i < Kapazitat; i++)

if (metaint == land[i].Nas) // условие поиска

{

Schreiben(land, i); // вывод cтроки совпадения

// получение рабочего индекса

metaindex = land[i].index;

}

cout << "\nУдалить эту запись? (y или n): "; // запрос на уделание

Antwort = \_getche(); // ответ

if ((Antwort != 'y') && (Antwort != 'n')) // проверка ввода

{

Prufung(); // оповещение о некорректном вводе

system("cls"); // очистка консоли

goto Such\_Fur\_Sub\_EinLT\_Nas; // строка 527

}

else if (Antwort == 'y') // если ответ положительный

return metaindex; // возвращаем рабочий индекс

else if (Antwort == 'n') // если ответ отрицательный

return 0; // возвращаем ноль-индекс

}

else if (leuchtturm > 1) // если совпадений несколько

{

cout << endl;

Such\_Fur\_Sub\_VielLT\_Nas:; // строки 563, 576

Mutze(); // вывод шапки

for (int i = 0; i < Kapazitat; i++) // вывод элементов массива

if (metaint == land[i].Nas) // условие вывода

Schreiben(land, i); // вывод cтроки совпадения

cout << "\nВведите индекс записи, которую необходимо удалить (0 –

никакую): "; // запрос на удаление

cin >> TeilDel;

for (int i = 0; i < strlen(TeilDel); i++) // проверка на символы

if ((TeilDel[i] < 48) || (TeilDel[i] > 57))

{

Prufung(); // оповещение о некорректном вводе

system("cls"); // очистка консоли

goto Such\_Fur\_Sub\_VielLT\_Nas; // строка 551

}

// выход в меню удаления

if ((strlen(TeilDel) == 1) && (TeilDel[0] == 48)) return 0;

// определение рабочего сивола

for (int i = 0; i < strlen(TeilDel); i++)

metaindex += ((TeilDel[i] - 48) \* pow(10, (strlen(TeilDel) - 1 - i)));

leuchtturm = 0; // обнуление маячка совпадений

for (int i = 0; i < Kapazitat; i++) // подсчёт совпадений

// условие увеличения числа совпадений

if (metaint == land[i].Nas)

leuchtturm++;

if (leuchtturm == 0) // если совпадений нет

{

Prufung(); // оповещение о некорректном вводе

system("cls"); // очистка консоли

goto Such\_Fur\_Sub\_VielLT\_Nas; // строка 551

}

else

for (int i = 0; i < Kapazitat; i++)

if (metaindex == land[i].index)

return metaindex; // возврат рабочего индекса

}

case '4': // поиск по площади

Such\_Fur\_Sub\_Anfang\_Terr:; // строки 593

system("cls"); // очистка консоли

cout << "Введите поисковую информацию по полю \"площадь\": ";

cin >> Zeitspanne;

if (strlen(Zeitspanne) > 9) // проверка на длину

{

Prufung(); // оповещение о некорректном вводе

goto Such\_Fur\_Sub\_Anfang\_Terr; // строка 586

}

for (int i = 0; i < strlen(Zeitspanne); i++) // проверка на символы

if ((Zeitspanne[i] < 48) || (Zeitspanne[i] > 57))

{

Prufung(); // оповещение о некорректном вводе

goto Such\_Fur\_Sub\_Anfang\_Terr; // строка 586

}

// определение ключеговго числа сравнения

for (int i = 0; i < strlen(Zeitspanne); i++)

metaint += ((Zeitspanne[i] - 48) \* pow(10, (strlen(Zeitspanne) - 1 –

i)));

for (int i = 0; i < Kapazitat; i++)

if (metaint == land[i].Terr) // условие увеличения числа совпадений

leuchtturm++; // увеличение маячка совпадений

if (leuchtturm == 0) // если совпадений нет

{

cout << "\nДаные не найдены!\n\n"; // опвещение, что данные не найдены

Wartung(); // задержка

return 0; // возврат ноль-индекса

}

else if (leuchtturm == 1) // если одно совпадение

{

cout << endl;

Such\_Fur\_Sub\_EinLT\_Terr:; // строка 629

Mutze(); // вывод шапки

for (int i = 0; i < Kapazitat; i++)

if (metaint == land[i].Terr) // условие поиска

{

Schreiben(land, i); // вывод cтроки совпадения

// получение рабочего индекса

metaindex = land[i].index;

}

cout << "\nУдалить эту запись? (y или n): "; // запрос на уделание

Antwort = \_getche(); // ответ

if ((Antwort != 'y') && (Antwort != 'n')) // проверка ввода

{

Prufung(); // оповещение о некорректном вводе

system("cls"); // очистка консоли

goto Such\_Fur\_Sub\_EinLT\_Terr; // строка 615

}

else if (Antwort == 'y') // если ответ положительный

return metaindex; // возвращаем рабочий индекс

else if (Antwort == 'n') // если ответ отрицательный

return 0; // возвращаем ноль-индекс

}

else if (leuchtturm > 1) // если совпадений несколько

{

cout << endl;

Such\_Fur\_Sub\_VielLT\_Terr:; // строки 651, 664

Mutze(); // вывод шапки

for (int i = 0; i < Kapazitat; i++) // вывод элементов массива

if (metaint == land[i].Terr) // условие вывода

Schreiben(land, i); // вывод cтроки совпадения

cout << "\nВведите индекс записи, которую необходимо удалить (0 –

никакую): "; // запрос на удаление

cin >> TeilDel;

for (int i = 0; i < strlen(TeilDel); i++) // проверка на символы

if ((TeilDel[i] < 48) || (TeilDel[i] > 57))

{

Prufung(); // оповещение о некорректном вводе

system("cls"); // очистка консоли

goto Such\_Fur\_Sub\_VielLT\_Terr; // строка 664

}

// выход в меню удаления

if ((strlen(TeilDel) == 1) && (TeilDel[0] == 48)) return 0;

// определение рабочего сивола

for (int i = 0; i < strlen(TeilDel); i++)

metaindex += ((TeilDel[i] - 48) \* pow(10, (strlen(TeilDel) - 1

- i)));

leuchtturm = 0; // обнуление маячка совпадений

for (int i = 0; i < Kapazitat; i++) // подсчёт совпадений

// условие увеличения числа совпадений

if (metaint == land[i].Terr)

leuchtturm++;

if (leuchtturm == 0) // если совпадений нет

{

Prufung(); // оповещение о некорректном вводе

system("cls"); // очистка консоли

goto Such\_Fur\_Sub\_VielLT\_Terr; // строка 664

}

else

for (int i = 0; i < Kapazitat; i++)

if (metaindex == land[i].index)

return metaindex; // возврат рабочего индекса

}

case '5':

Such\_Fur\_Sub\_Anfang\_Nas\_pro\_Terr:; // строки 681, 687

system("cls"); // очистка консоли

cout << "Введите поисковую информацию по полю \"плотность населения\": ";

cin >> Zeitspanne;

if (strlen(Zeitspanne) > 8) // проверка на длину

{

Prufung(); // оповещение о некорректном вводе

goto Such\_Fur\_Sub\_Anfang\_Nas\_pro\_Terr; // строка 674

}

for (int i = 0; i < strlen(Zeitspanne); i++) // проверка на символы

if ((Zeitspanne[i] < 48) || (Zeitspanne[i] > 57))

{

Prufung(); // оповещение о некорректном вводе

goto Such\_Fur\_Sub\_Anfang\_Nas\_pro\_Terr; // строка 674

}

// определение ключевого числа сравнения

for (int i = 0; i < strlen(Zeitspanne); i++)

metaint += ((Zeitspanne[i] - 48) \* pow(10, (strlen(Zeitspanne) - 1 –

i)));

for (int i = 0; i < Kapazitat; i++)

// условие увеличения числа совпадений

if (metaint == land[i].Nas\_pro\_Terr)

leuchtturm++; // увеличение маячка совпадений

if (leuchtturm == 0) // если совпадений нет

{

cout << "\nДаные не найдены!\n\n"; // опвещение, что данные не найдены

Wartung(); // задержка

return 0; // возврат ноль-индекса

}

else if (leuchtturm == 1) // если одно совпадение

{

cout << endl;

Such\_Fur\_Sub\_EinLT\_Nas\_pro\_Terr:; // строка 717

Mutze(); // вывод элементов шапки

for (int i = 0; i < Kapazitat; i++)

if (metaint == land[i].Nas\_pro\_Terr) // условие поиска

{

Schreiben(land, i); // вывод cтроки совпадения

// получение рабочего индекса

metaindex = land[i].index;

}

cout << "\nУдалить эту запись? (y или n): "; // запрос на уделание

Antwort = \_getche(); // ответ

if ((Antwort != 'y') && (Antwort != 'n')) // проверка ввода

{

Prufung(); // оповещение о некорректном вводе

system("cls"); // очистка консоли

goto Such\_Fur\_Sub\_EinLT\_Nas\_pro\_Terr; // строка 703

}

else if (Antwort == 'y') // если ответ положительный

return metaindex; // возвращаем рабочий индекс

else if (Antwort == 'n') // если ответ отрицательный

return 0; // возвращаем ноль-индекс

}

else if (leuchtturm > 1) // если совпадений несколько

{

cout << endl;

Such\_Fur\_Sub\_VielLT\_Nas\_pro\_Terr:; // строки 739, 752

Mutze(); // вывод шапки

for (int i = 0; i < Kapazitat; i++) // вывод элементов массива

if (metaint == land[i].Nas\_pro\_Terr) // условие вывода

Schreiben(land, i); // вывод cтроки совпадения

cout << "\nВведите индекс записи, которую необходимо удалить (0 –

никакую): "; // запрос на удаление

cin >> TeilDel;

for (int i = 0; i < strlen(TeilDel); i++) // проверка на символы

if ((TeilDel[i] < 48) || (TeilDel[i] > 57))

{

Prufung(); // оповещение о некорректном вводе

system("cls"); // очистка консоли

goto Such\_Fur\_Sub\_VielLT\_Nas\_pro\_Terr; // строка 727

}

// выход в меню удаления

if ((strlen(TeilDel) == 1) && (TeilDel[0] == 48)) return 0;

// определение рабочего сивола

for (int i = 0; i < strlen(TeilDel); i++)

metaindex += ((TeilDel[i] - 48) \* pow(10, (strlen(TeilDel) - 1 - i)));

leuchtturm = 0; // обнуление маячка совпадений

for (int i = 0; i < Kapazitat; i++) // подсчёт совпадений

// условие увеличения числа совпадений

if (metaint == land[i].Nas\_pro\_Terr)

leuchtturm++; // увеличение маячка совпадений

if (leuchtturm == 0) // если совпадений нет

{

Prufung(); // оповещение о некорректном вводе

system("cls"); // очистка консоли

goto Such\_Fur\_Sub\_VielLT\_Nas\_pro\_Terr; // строка 727

}

else

for (int i = 0; i < Kapazitat; i++)

if (metaindex == land[i].index)

return metaindex; // возврат рабочего индекса

}

case '6': // поиск по коду валюты

Such\_Fur\_Sub\_Anfang\_VAL:; // строки 769, 776

system("cls"); // очистка консоли

cout << "Введите поисковую информацию по полю \"коду валюты\": ";

cin >> Zeitspanne;

if (strlen(Zeitspanne) != 3) // проверка на длину

{

Prufung(); // оповещение о некорректном вводе

goto Such\_Fur\_Sub\_Anfang\_VAL; // строка 762

}

for (int i = 0; i < strlen(Zeitspanne); i++) // проверка на символы

if ((Zeitspanne[i] < 65) || (Zeitspanne[i] > 90))

if ((Zeitspanne[i] < 97) || (Zeitspanne[i] > 122))

{

Prufung(); // оповещение о некорректном вводе

goto Such\_Fur\_Sub\_Anfang\_VAL; // строка 762

}

KorrigierenDaten(Zeitspanne, 2); // коррекция введённых данных

// работа с мачком совпадений

for (int i = 0; i < Kapazitat; i++)

if (strcmp(Zeitspanne, land[i].VAL) == 0) // сравнение с действительными данными

leuchtturm++; // увеличение маячка совпадений

if (leuchtturm == 0) // если совпадений нет

{

cout << "\nДаные не найдены!\n\n"; // оповещение о ненайденных данных

Wartung(); // задержка

return 0; // вовзрат ноль-индекса

}

else if (leuchtturm == 1) // если совпадение одно

{

cout << endl;

Such\_Fur\_Sub\_EinLT\_VAL:; // строка 807

Mutze(); // вывод элементов шапки

for (int i = 0; i < Kapazitat; i++)

if (strcmp(Zeitspanne, land[i].HauptStadt) == 0) // нахождение строки сопадения

{

Schreiben(land, i); // вывод cтроки совпадения

metaindex = land[i].index; // запоминание индекса

}

cout << "\nУдалить эту запись? (y или n): "; //запрос на удаление записи

Antwort = \_getche(); // ответ

if ((Antwort != 'y') && (Antwort != 'n')) // проверка ввода

{

Prufung(); // оповещение о некорректном вводе

system("cls"); // очистка консоли

goto Such\_Fur\_Sub\_EinLT\_VAL; // строка 793

}

else if (Antwort == 'y') // если ответ положительный

return metaindex; // возвращаем рабочий индекс

else if (Antwort == 'n') // если ответ отрицательный

return 0; // возвращаем ноль-индекс

}

else if (leuchtturm > 1) // если совпадений несколько

{

cout << endl;

Such\_Fur\_Sub\_VielLT\_VAL:; // строки 829, 842

Mutze(); // вывод шапки

for (int i = 0; i < Kapazitat; i++) // вывод элементов массива

if (strcmp(Zeitspanne, land[i].VAL) == 0)

Schreiben(land, i); // вывод cтроки совпадения

cout << "\nВведите индекс записи, которую необходимо удалить (0 - никакую): "; // запрос на удаление

cin >> TeilDel;

for (int i = 0; i < strlen(TeilDel); i++) // проверка на символы

if ((TeilDel[i] < 48) || (TeilDel[i] > 57))

{

Prufung(); // оповещение о некорректном вводе

system("cls"); // очистка консоли

goto Such\_Fur\_Sub\_VielLT\_VAL; // строка 817

}

if ((strlen(TeilDel) == 1) && (TeilDel[0] == 48)) return 0; // выход в меню удаления

for (int i = 0; i < strlen(TeilDel); i++) // определение рабочего сивола

metaindex += ((TeilDel[i] - 48) \* pow(10, (strlen(TeilDel) - 1 - i)));

leuchtturm = 0; // обнуление маячка совпадений

for (int i = 0; i < Kapazitat; i++) // подсчёт совпадений

if (strcmp(Zeitspanne, land[i].VAL) == 0)

leuchtturm++; // увеличение маячка совпадений

if (leuchtturm == 0) // если совпадений нет

{

Prufung(); // оповещение о некорректном вводе

system("cls"); // очистка консоли

goto Such\_Fur\_Sub\_VielLT\_VAL; // строка 817

}

else

for (int i = 0; i < Kapazitat; i++)

if (metaindex == land[i].index)

return metaindex; // возврат рабочего индекса

}

default: // указание несуществующего пункта меню

Prufung(); // оповещение о некорректном вводе

goto Such\_Fur\_Sub\_Hauptablauf; // строка 286

}

}

**Peripherie.h**

#pragma once

#include "lender.h" // подключение структуры

void Schreiben(lander\*& land , int i); // функция вывода элементов массива на экран

void Wartung(); // функция организации задержки и выхода в меню

void Prufung(); // реакция на неверно ведённые данные

void Mutze(); // прось шапки таблицы

void KorrigierenDaten(char Zeitspanne[], int gang); // корректировка вводимых данных

**Peripherie.cpp**

#include <iostream> // для ввода и вывода

#include <iomanip> // для setw ()

#include "Peripherie.h" // подключение одноимённого заголовочного файла

using namespace std; // подключение пространства имён std

void Mutze() // вывод шапки

{

// выравнивание по левой стороне и вывод элементов шапки с соответствующей резервацией

места в консоли

cout << setw(5) << setiosflags(ios::left) << "Инд."

<< setw(26) << "Страна"

<< setw(26) << "Столица"

<< setw(15) << "Население(чел)"

<< setw(15) << "Площадь(км^2)"

<< setw(15) << "Плотность(N/S)"

<< setw(3) << "Валюта";

cout << endl << endl; // для эстетики

}

void Prufung() // оповещение о некорректных данных

{

system("cls"); // очистка консоли

// непостредственно оповещение о некорректных данных

cout << "Неверный ввод данных!\n"

<< "Недопустимый символ \n"

<< " |(исп. для ввода латиницу и '\_' - для пробела - или '-')\n"

<< " |(исп. для ответов 'y' или 'n')\n"

<< " |(исп. для пунктов меню числа)\n"

<< "или превышение доспустимой длины!\n"

<< "Попробуйте ещё раз.\n\n";

system("pause"); // организация задержки

}

void Wartung() // функция задержки

{

system("pause"); // организация задержки

system("cls"); // очистка консоли

}

void Schreiben(lander\*& land , int i) // функция вывода i-того элемента массива структур

{

// выравнивание по левой стороне и вывод элементов массива с соответствующей

резервацией места в консоли

cout << setw(5) << setiosflags(ios::left) << land[i].index

<< setw(26) << land[i].Name

<< setw(26) << land[i].HauptStadt

<< setw(15) << land[i].Nas

<< setw(15) << land[i].Terr

<< setw(15) << land[i].Nas\_pro\_Terr

<< setw(3) << land[i].VAL << endl;

}

void KorrigierenDaten(char Zeitspanne[], int gang) // для коррекции введённых основных данных

{

if (gang == 1) // коррекция для названия страны и столицы

{

for (int i = 0; i < strlen(Zeitspanne); i++)

{

// уменьшение регистора всей строки

Zeitspanne[i] = tolower(Zeitspanne[i]);

// сравнение по коду таблицы ASCII

if ((Zeitspanne[i] == 95) || (Zeitspanne[i] == 45))

if (i + 1 < strlen(Zeitspanne)) // дополнительное условие

{

// увеличение регистора после знаков "\_" и "-"

Zeitspanne[i + 1] = toupper(Zeitspanne[i + 1]);

i = i + 2;

}

}

// увеличение регистора первого символа

Zeitspanne[0] = toupper(Zeitspanne[0]);

}

else if (gang == 2) // коррекция для кода валюты страны

for (int i = 0; i < 3; i++)

// увеличение регистора каждого символа

Zeitspanne[i] = toupper(Zeitspanne[i]);

}