ФЕДЕРАЛЬНОЕ АГЕНТСТВО ВОЗДУШНОГО ТРАНСПОРТА

-=BlackLynx=-

(РОСАВИАЦИЯ)

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«**МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ГРАЖДАНСКОЙ АВИАЦИИ» (МГТУ ГА)

Кафедра вычислительных машин, комплексов, систем и сетей

Отчет защищен с оценкой

(подпись преподавателя, дата)

ОТЧЕТ

по результатам выполнения практики

***Учебная 1. Ознакомительная практика***

Выполнил студент группы ЭВМ 1-2

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Ф.И.О., № зачётной книжки) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись, дата)

МОСКВА - 2022

Аннотация.

Разработана система для работы с файлами данных. Создания файлов, их обработки: сортировка, удаления записей. Так же просмотр данных в виде таблицы.

Программу для обработки текста. Поиск предложений и вывод их в консоль. Поиск слов и замена подстрок в них.

Оглавление:

Аннотация……………………………………………………………………………………………...2 Практическое занятие №1…………………………………………………………………………….4

Цель…………………………………………………………………………………………………..4

Техническое задание………………………………………………………………………………..4

Таблицы локальных переменных и схемы алгоритмов каждой функции программы………….….4

Практическое занятие №2……………………………………………………………………………12

Цель………………………………………………………………………………………………….12

Техническое задание…………………………………………………………………………...…..12

Таблицы локальных переменных и схемы алгоритмов каждой функции программы…………….12

Практическое занятие №3……………………………………………………………………………17

Цель………………………………………………………………………………………………….17

Техническое задание……………………………………………………………………………….17

Таблицы локальных переменных и схемы алгоритмов каждой функции программы…………….17

Практическое занятие №4……………………………………………………………………………31

Цель………………………………………………………………………………………………….31

Техническое задание……………………………………………………………………………….31

Таблицы локальных переменных и схемы алгоритмов каждой функции программы.……….…31

Список литературы…………………………………………………………………...………………35 Приложение 1. Программа…………………………………………………………………………...36

Практическое занятие №1…………………………………………………………………………36

Практическое занятие №2…………………………………………………………………………39

Практическое занятие №3…………………………………………………………………………43

Практическое занятие №4…………………………………………………………………………49

Приложение 2. Содержание файлов с данными для тестирования……………………………….51

Практическое занятие №1…………………………………………………………………………51

Практическое занятие №2…………………………………………………………………………52

Практическое занятие №3…………………………………………………………………………53

Практическое занятие №4…………………………………………………………………………54

Приложение 3. Результат выполнения……………………………...……………………………….55

Практическое занятие №1…………………………………………………………………………55

Практическое занятие №2…………………………………………………………………………56

Практическое занятие №3…………………………………………………………………………57

Практическое занятие №4…………………………………………………………………………59

Практическое задание №1

Цель:

Освоение методов работы с бинарными и текстовыми файлами, освоение алгоритмов удаления данных из бинарного файла.

Техническое задание:

Разработать схемы алгоритмов, структуру и текст программы для обработки структурных данных текстового и бинарного файлов с использованием функций создания файлов и удаления ряда структур из файлов.

Схемы алгоритмов должны быть разработаны для каждой функции.

Тестирование функции удаления проводить с помощью файла данных для удаления, при этом должны быть либо удалены отмеченные записи из бинарного файла и текстового файла, либо выведены правильные фразы, например об отсутствии искомых данных.

Вариант: 2

Если в бинарном файле найдена структура, которую следует удалить, то на ее место в файле копируется структура, следующая за ней, на которую, в свою очередь также копируется следующая структура и так далее до конца файла. В результате такого копирования удаляемая структура будет удалена (а именно - "закрыта" следующей за ней структурой), но в конце файла окажутся две одинаковые последние структуры, поэтому необходимо усечь файл на размер одной структуры. Реализация данного алгоритма должна базироваться на использовании функций ввода / вывода нижнего уровня (например, использовать сhsize().

Таблицы локальных переменных и схемы алгоритмов каждой функции программы.

Глобальные переменные

|  |  |
| --- | --- |
| fstream binf | Объявление двунаправленного потока |
| ifstream fin | Объявление входного потока |
| ofstream fout | Объявление выходного потока |
| int count | Количество структур |
| char filename[10] | Название текстового файла |
| char binfile[15] | Название бинарного файла |
| char delfile[11] | Название файла с данными для удаления |
| Cities ct | Экземпляр структуры |

Функции

|  |  |
| --- | --- |
| filtr | Функцию для удаления пробелов (ведущих и замыкающих) в строковых данных. |
| createFile | Создание файла с данными |
| createBin | Создание бинарного файла |
| RemoveLine | Удаление строк по данным |
| correction | Коррекция |
| rbinf | Чтение бинарного файла |

Алгоритмы:

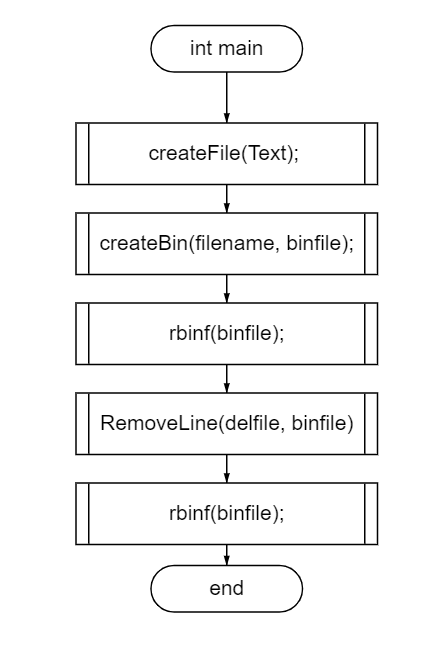
Структура программы:

Рис. 1

void flitr():

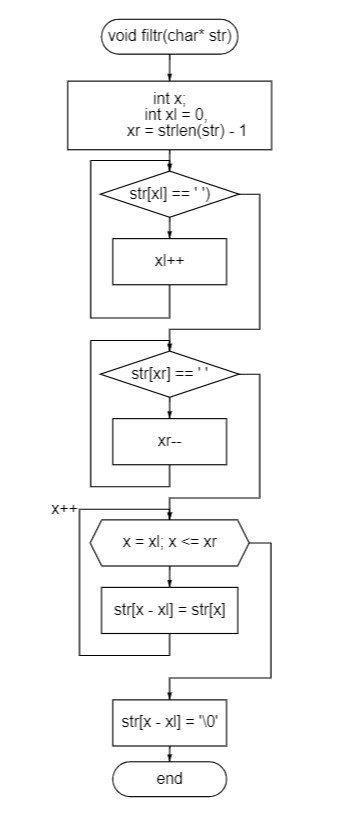


Рис. 2

void createFile():

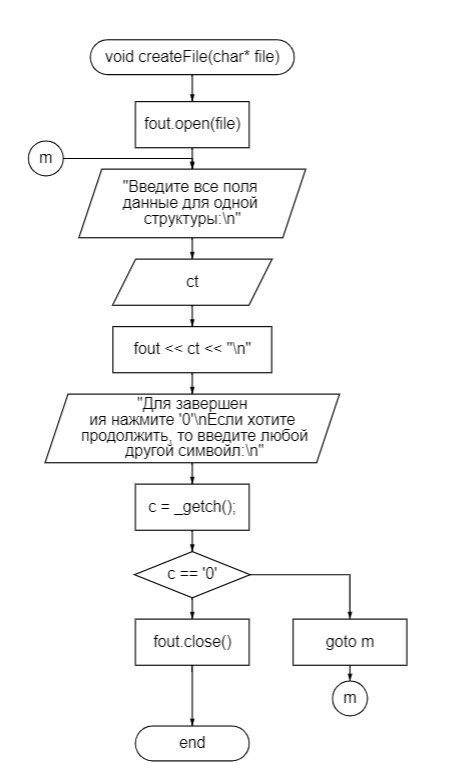


Рис. 3

void createBin():

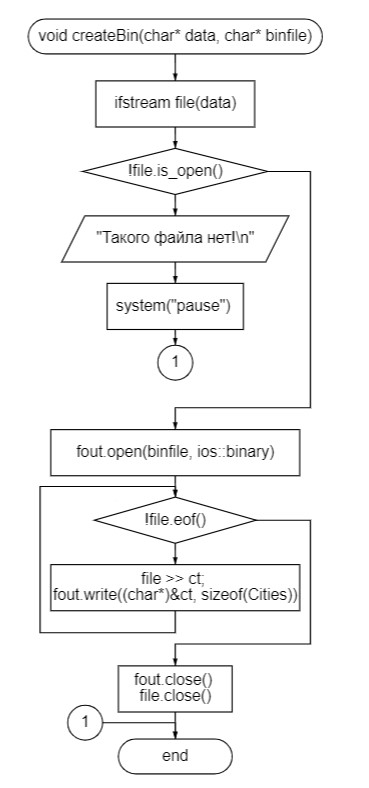


Рис. 4

void rbinf():

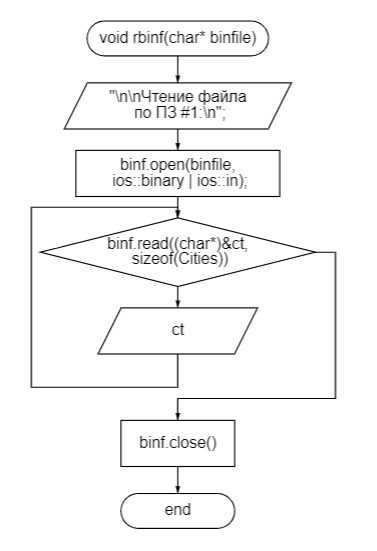


Рис. 5

void RemoveLine():

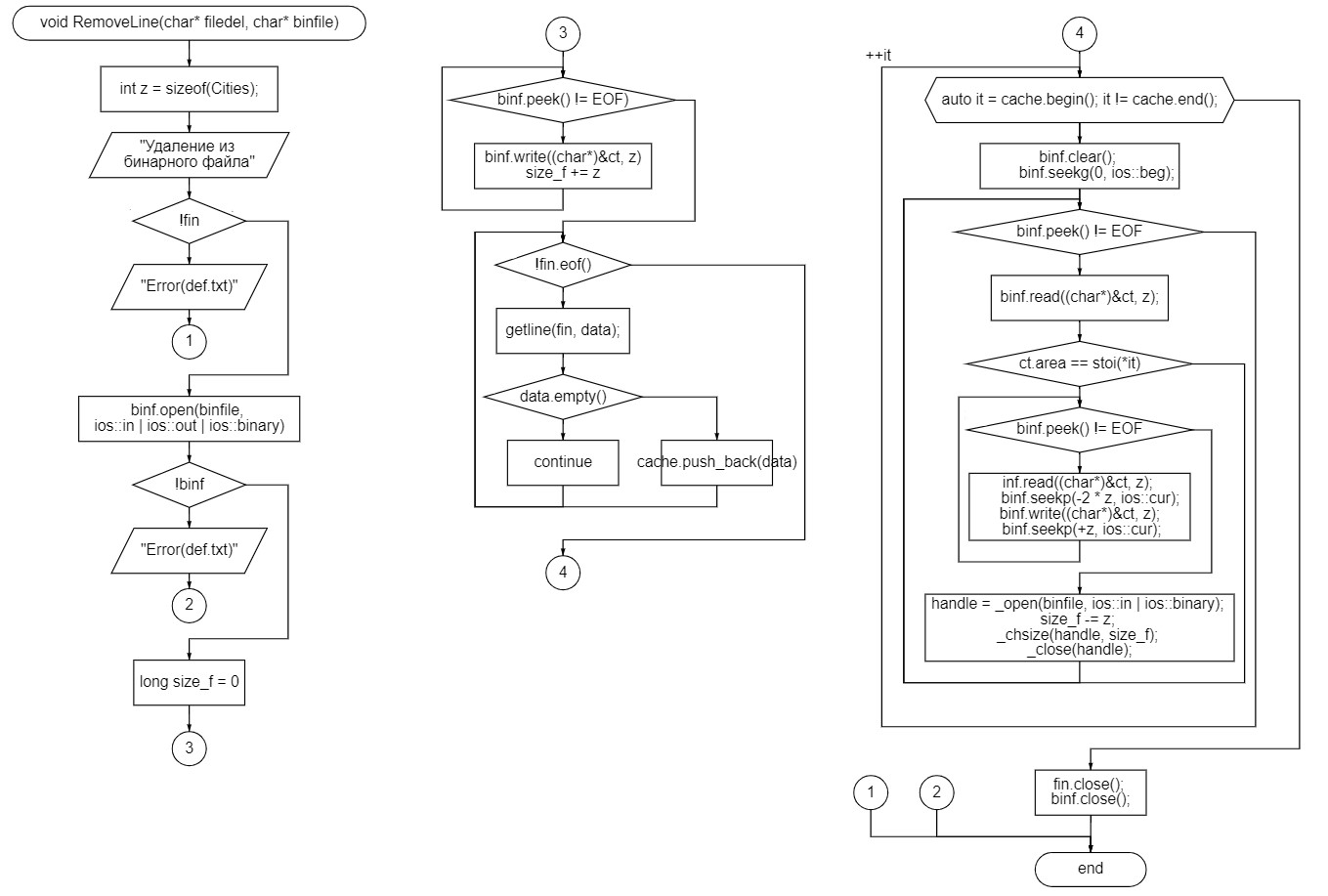


Рис. 6

Практическое задание №2

Цель:

1. Освоение методов внутренней сортировки.
2. Освоение алгоритмов сортировки структурированных данных бинарных файлов, с использованием динамических объектов - массивов, списков.

Техническое задание:

Разработать схему алгоритма, структуру и текст программы для обработки данных бинарного файла с использованием функции сортировки структурированных данных файла заданным методом.

Вариант: 3

Сортировка методом Шелла.

Таблицы локальных переменных и схемы алгоритмов каждой функции программы.

Глобальные переменные

|  |  |
| --- | --- |
| fstream binf | Объявление двунаправленного потока |
| ifstream fin | Объявление входного потока |
| ofstream fout | Объявление выходного потока |
| int count | Количество структур |
| char filename[10] | Название текстового файла |
| char binfile[15] | Название бинарного файла |
| char delfile[11] | Название файла с данными для удаления |
| Cities ct | Экземпляр структуры |

Функции

|  |  |
| --- | --- |
| filtr | Функцию для удаления пробелов (ведущих и замыкающих) в строковых данных. |
| createFile | Создание файла с данными |
| createBin | Создание бинарного файла |
| RemoveLine | Удаление строк по данным |
| correction | Коррекция |
| rbinf | Чтение бинарного файла |
| sortedBinary | Сортировка бинарного файла |
| sorted | Функция, сортирующая структуры медом Шелла |

Алгоритмы:

Структура программы:

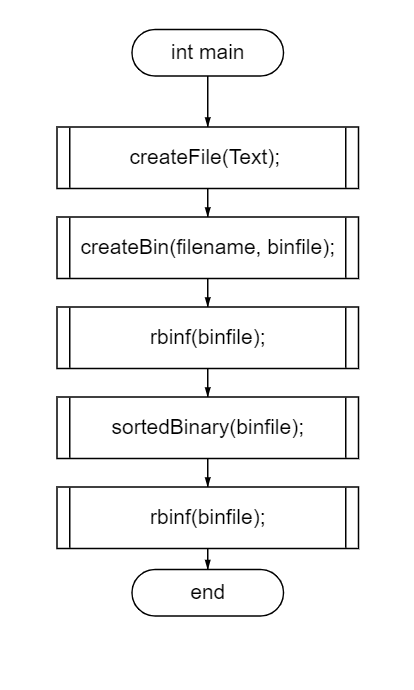


Рис. 7

void sortedBinary():

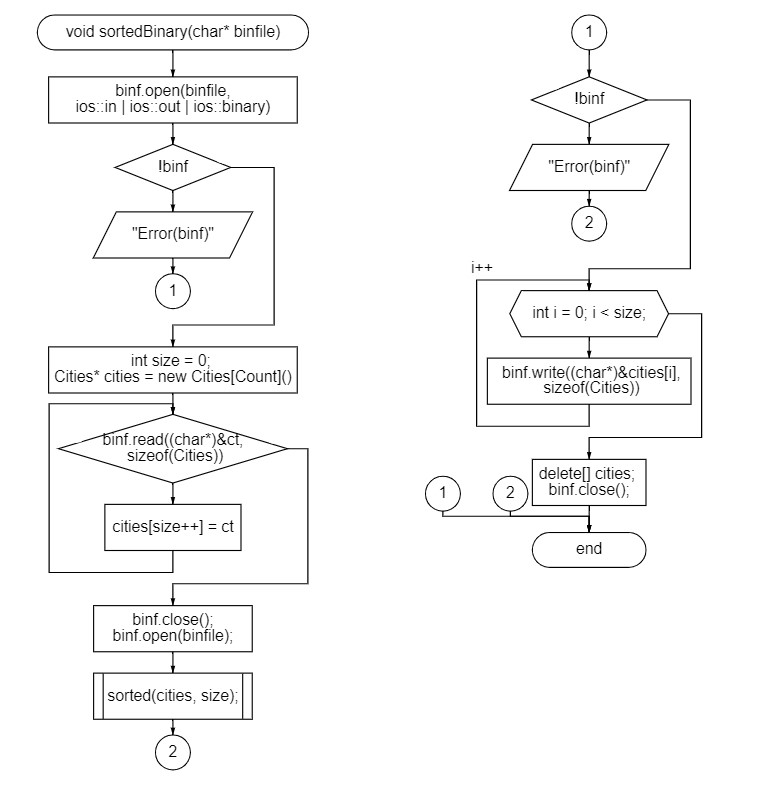


Рис. 8

void sorted():

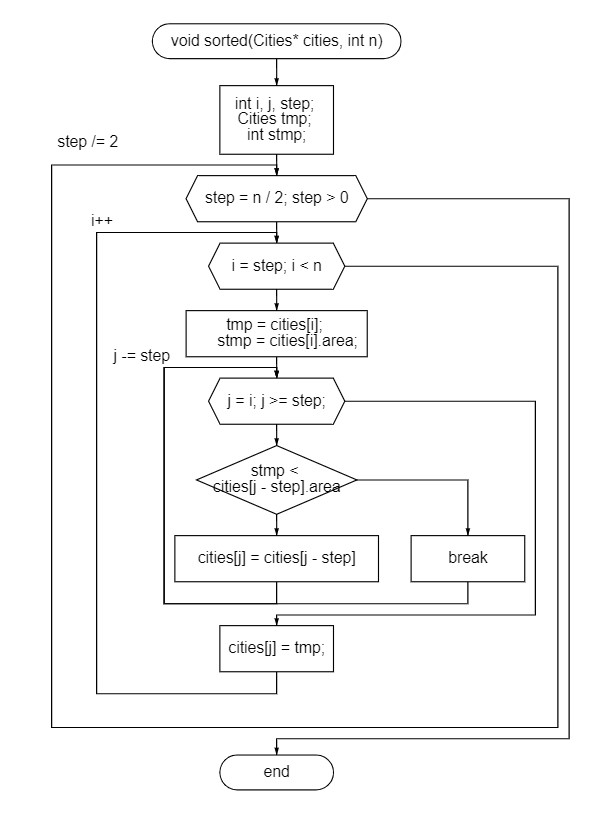


Рис. 9

Практическое задание №3

Цель:

1. Приобретение навыков разработки многомодульных программ.
2. Освоение методов разработки программы, управление которыми, осуществляется с помощью меню.
3. Освоение препроцессорные средств управления ходом компиляции программы.

Техническое задание:

Разработать многомодульную программу, включающую:

1. Заголовочный файл, с объявлением внешних переменных и прототипов используемых пользовательских функций;
2. Библиотеку функций пользователя – файл с определениями функций;
3. Файл с главной функцией, содержащий систему меню для выбора вариантов обработки данных.

Обрабатывать структурные данные, записи которых содержатся в бинарном и текстовом файлах.

Разработать структуру программы и схемы алгоритмов каждой функции, включая главную.

Вариант: 2

Алгоритм организации меню с помощью массива указателей на функции. В алгоритме отсутствует оператор выбора варианта.

Таблицы локальных переменных и схемы алгоритмов каждой функции программы.

Глобальные переменные

|  |  |
| --- | --- |
| fstream binf | Объявление двунаправленного потока |
| ifstream fin | Объявление входного потока |
| ofstream fout | Объявление выходного потока |
| int count | Количество структур |
| char filename[10] | Название текстового файла |
| char binfile[15] | Название бинарного файла |
| char delfile[11] | Название файла с данными для удаления |
| Cities ct | Экземпляр структуры |

Функции

|  |  |
| --- | --- |
| filtr | Функцию для удаления пробелов (ведущих и замыкающих) в строковых данных. |
| createFile | Создание файла с данными |
| createBin | Создание бинарного файла |
| RemoveLine | Удаление строк по данным |
| correction | Коррекция |
| rbinf | Чтение бинарного файла |
| sortedBinary | Сортировка бинарного файла |
| sorted | Функция, сортирующая структуры медом Шелла |
| outputMenu | Вывод меню |
| printHeaderTable | Вывод шапки таблицы |
| printFooterTable | Вывод завершающей строки таблицы |
| printLineTable | Вывод одной строки таблицы |
| CheckEdit | «Красивый» вывод данных файла |
| Create | Меню создание |
| Read | Меню чтение |
| Sorted | Меню сортировка |
| Remove | Меню удаление |
| Exit | Меню выход |
| inputNumber | Ввод числа |

Алгоритмы:

Структура программы:

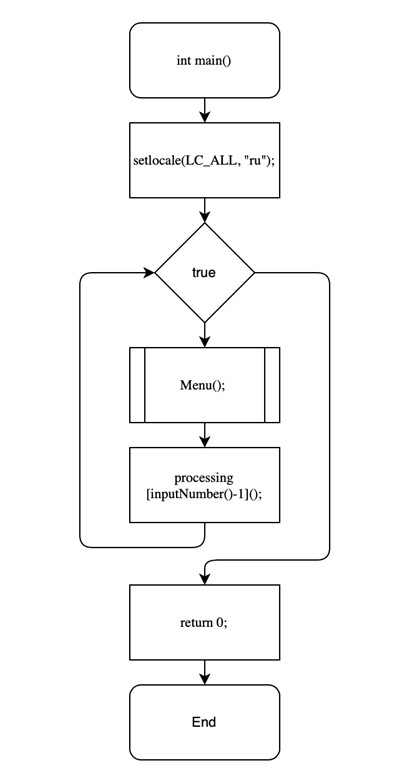


Рис. 10

void outputMenu():

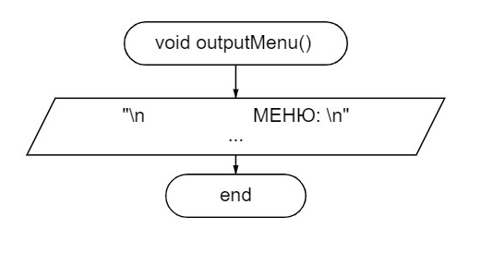


Рис. 11

void printHeaderTable():

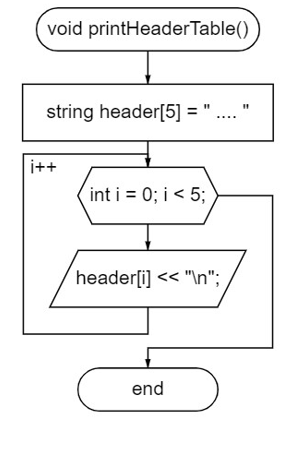


Рис. 12

void printFooterTabele():

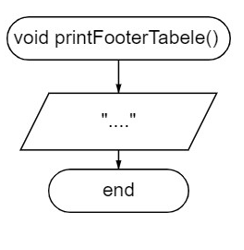


Рис. 14

void printLineTable(Cities city):

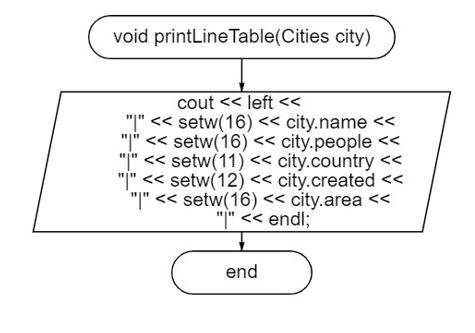


Рис. 15

void Create():

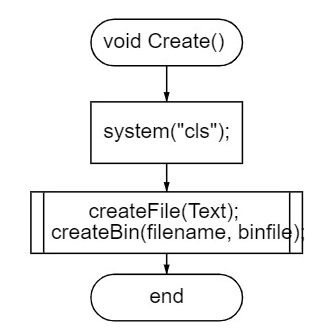


Рис. 16

void Read():

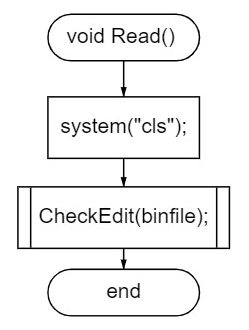


Рис. 17

void Sorted():

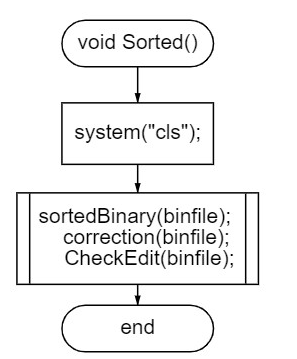


Рис. 18

void Remove():

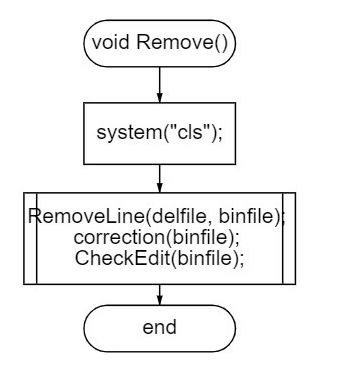


Рис. 19

Void Exit():

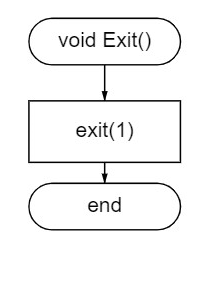


Рис. 20

int inputNumber():

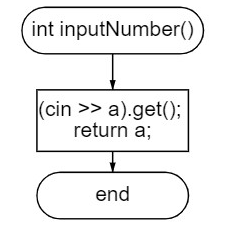


Рис. 21

void CheckEdit():

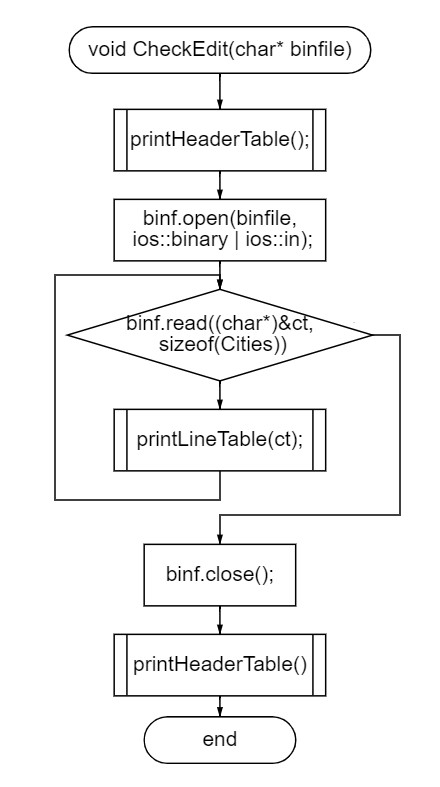


Рис. 22

Практическое задание №4

Цель:

1. Целью работы является получение навыков программирования с использованием методов стандартного класса string для обработки символьной информации, хранящейся в текстовых файлах.

Техническое задание:

Вариант 3:

1) Считывает текст из файла и вывести на экран только строки, содержащие двузначные числа.

2) Определить различные слова текста и их количество. Сформировать таблицу кодировки слов случайными целыми числами. Зашифровать заданный текст пословно. Вывести зашифрованный текст в рабочий файл. Расшифровать зашифрованный текст и вывести его в протокол.

Таблицы локальных переменных и схемы алгоритмов каждой функции программы.

Глобальные переменные

|  |  |
| --- | --- |
| ofstream write | Поток записи |
| Ifstream read | Поток чтения |
| const char ch[] | Строка для приятного вывода в консоль |

Функции

|  |  |
| --- | --- |
| TwoDigitOut() | Функция, выполняющая первое ТЗ |
| Number2() | Функция, выполняющая второе ТЗ |

Алгоритмы:

Структура программы:

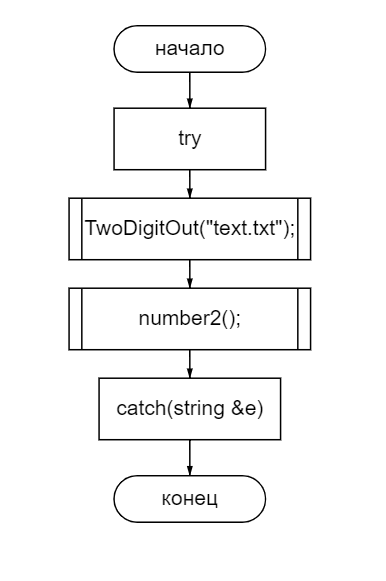


Рис. 23

TwoDigitOut(const char\* file):

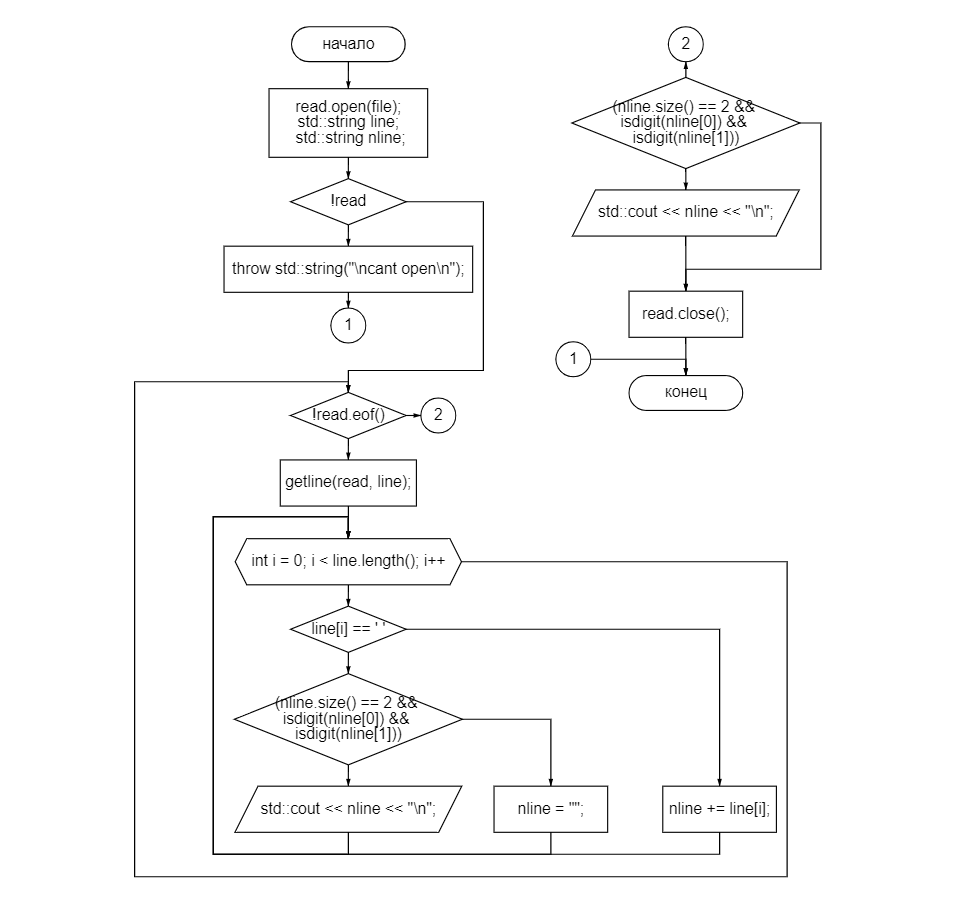


Рис. 24

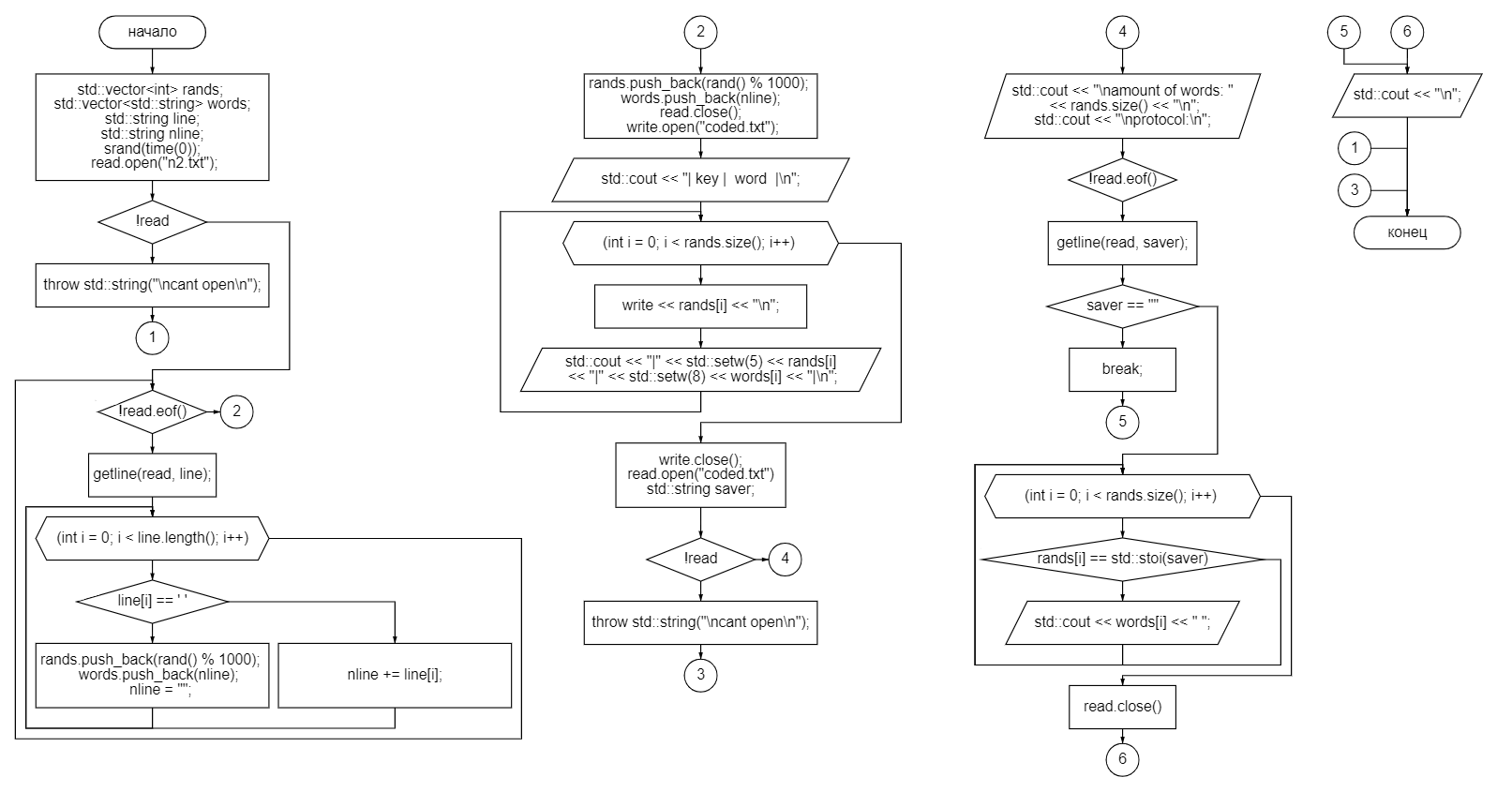
number2():

Рис. 25

Список литературы:

Л.А. Надейкина, Н.И.Черкасова ПОСОБИЕ к выполнению Вычислительной (учебной) практики по дисциплине «ПРОГРАММИРОВАНИЕ», 2014.

Приложение 1. Программы:

Практическое занятие №1:

#include <iostream>

#include <fstream>

#include <string>

#include <iomanip>

#include <vector>

#include <io.h>

#include <conio.h>

using namespace std;

ofstream fout;

ifstream fin;

fstream binf;

const int Count = 5;

char filename[10] = "test.txt";

char binfile[15] = "binfile.txt";

char delfile[11] = "def.txt";

struct Cities {

char name[10];

int people;

char country[10];

int created;

int area;

}ct;

void filtr(char\* str) {

int x;

int xl = 0,

xr = strlen(str) - 1;

while (str[xl] == ' ') xl++;

while (str[xr] == ' ') xr--;

for (x = xl; x <= xr; x++)

str[x - xl] = str[x];

str[x - xl] = '\0';

}

ostream& operator << (ostream& out, const Cities& Cities) {

out << left

<< setw(10) << Cities.name

<< setw(10) << Cities.people

<< setw(10) << Cities.country

<< setw(10) << Cities.created

<< setw(10) << Cities.area;

return out;

}

istream& operator >> (istream& in, Cities& Cities) {

char T[255];

in.get(Cities.name, 10);

in >> Cities.people;

in.get(Cities.country, 10);

in >> Cities.created >> Cities.area;

filtr(Cities.name);

filtr(Cities.country);

in.getline(T, 255);

return in;

}

void createFile(char\* file) {

fout.open(file);

m: cout << "Введите все поля данные для одной структуры:\n";

cin >> ct;

fout << ct << "\n";

cout << "Для завершения нажмите '0'\nЕсли хотите продолжить, то введите любой другой симвойл:\n";

char c;

c = \_getch();

if (c == '0'){

fout.close();

return;

}

else goto m;

}

void createBin(char\* data, char\* binfile){

ifstream file(data);

if (!file.is\_open()){

cout << "Такого файла нет!\n";

system("pause");

exit(0);

}

fout.open(binfile, ios::binary);

while (!file.eof()){

file >> ct;

fout.write((char\*)&ct, sizeof(Cities));

}

fout.close();

file.close();

}

void RemoveLine(char\* filedel, char\* binfile) {

int z = sizeof(Cities);

cout << "Удаление из бинарного файла" << endl;

fin.open(filedel);

if (!fin) {

cout << "Error(def.txt)";

exit(0);

}

binf.open(binfile, ios::in | ios::out | ios::binary);

if (!binf) {

cout << "Error(binf)";

exit(0);

}

long size\_f = 0;

while (binf.peek() != EOF) {

binf.write((char\*)&ct, z);

size\_f += z;

}

string data;

vector<string> cache;

while (!fin.eof()) {

getline(fin, data);

if (data.empty()) continue;

else cache.push\_back(data);

}

for (auto it = cache.begin(); it != cache.end(); it++) {

binf.clear();

binf.seekg(0, ios::beg);

while (binf.peek() != EOF) {

binf.read((char\*)&ct, z);

if (ct.area == stoi(\*it)) {

while (binf.peek() != EOF) {

binf.read((char\*)&ct, z);

binf.seekp(-2 \* z, ios::cur); //смещаем указатель файла на одну структуру назад

binf.write((char\*)&ct, z); //запись в файл откорректированной структуры

binf.seekp(+z, ios::cur);

};

int handle = \_open(binfile, ios::in | ios::binary);

size\_f -= z;

\_chsize(handle, size\_f);

\_close(handle);

}

}

}

fin.close();

binf.close();

};

void correction(char\* binfile) {

binf.open(binfile, ios::binary | ios::in);

fout.open("corr.txt");

while (binf.read((char\*)&ct, sizeof(Cities))) {

fout << ct << "\n";

}

fout.close();

binf.close();

}

void rbinf(char\* binfile) {

cout << "\n\nЧтение файла по ПЗ #1:\n";

binf.open(binfile, ios::binary | ios::in);

while (binf.read((char\*)&ct, sizeof(Cities))) {

cout << ct << "\n";

}

binf.close();

}

int main() {

setlocale(LC\_ALL, "ru");

createFile(filename);

createBin(filename, binfile);

rbinf(binfile);

RemoveLine(delfile, binfile);

rbinf(binfile);

}

Практическое занятие №2:

#include <iostream>

#include <fstream>

#include <string>

#include <iomanip>

#include <vector>

#include <io.h>

#include <conio.h>

using namespace std;

ofstream fout;

ifstream fin;

fstream binf;

const int Count = 5;

char filename[10] = "test.txt";

char binfile[15] = "binfile.txt";

char delfile[11] = "def.txt";

struct Cities {

char name[10];

int people;

char country[10];

int created;

int area;

}ct;

void sorted(Cities\* cities, int n);

void filtr(char\* str) {

int x;

int xl = 0,

xr = strlen(str) - 1;

while (str[xl] == ' ') xl++;

while (str[xr] == ' ') xr--;

for (x = xl; x <= xr; x++)

str[x - xl] = str[x];

str[x - xl] = '\0';

}

ostream& operator << (ostream& out, const Cities& Cities) {

out << left

<< setw(10) << Cities.name

<< setw(10) << Cities.people

<< setw(10) << Cities.country

<< setw(10) << Cities.created

<< setw(10) << Cities.area;

return out;

}

istream& operator >> (istream& in, Cities& Cities) {

char T[255];

in.get(Cities.name, 10);

in >> Cities.people;

in.get(Cities.country, 10);

in >> Cities.created >> Cities.area;

filtr(Cities.name);

filtr(Cities.country);

in.getline(T, 255);

return in;

}

void createFile(char\* file) {

fout.open(file);

m: cout << "Введите все поля данные для одной структуры:\n";

cin >> ct;

fout << ct << "\n";

cout << "Для завершения нажмите '0'\nЕсли хотите продолжить, то введите любой другой симвойл:\n";

char c;

c = \_getch();

if (c == '0'){

fout.close();

return;

}

else goto m;

}

void createBin(char\* data, char\* binfile){

ifstream file(data);

if (!file.is\_open()){

cout << "Такого файла нет!\n";

system("pause");

exit(0);

}

fout.open(binfile, ios::binary);

while (!file.eof()){

file >> ct;

fout.write((char\*)&ct, sizeof(Cities));

}

fout.close();

file.close();

}

void RemoveLine(char\* filedel, char\* binfile) {

int z = sizeof(Cities);

cout << "Удаление из бинарного файла" << endl;

fin.open(filedel);

if (!fin) {

cout << "Error(def.txt)";

exit(0);

}

binf.open(binfile, ios::in | ios::out | ios::binary);

if (!binf) {

cout << "Error(binf)";

exit(0);

}

long size\_f = 0;

while (binf.peek() != EOF) {

binf.write((char\*)&ct, z);

size\_f += z;

}

string data;

vector<string> cache;

while (!fin.eof()) {

getline(fin, data);

if (data.empty()) continue;

else cache.push\_back(data);

}

for (auto it = cache.begin(); it != cache.end(); it++) {

binf.clear();

binf.seekg(0, ios::beg);

while (binf.peek() != EOF) {

binf.read((char\*)&ct, z);

if (ct.area == stoi(\*it)) {

while (binf.peek() != EOF) {

binf.read((char\*)&ct, z);

binf.seekp(-2 \* z, ios::cur); //смещаем указатель файла на одну структуру назад

binf.write((char\*)&ct, z); //запись в файл откорректированной структуры

binf.seekp(+z, ios::cur);

};

int handle = \_open(binfile, ios::in | ios::binary);

size\_f -= z;

\_chsize(handle, size\_f);

\_close(handle);

}

}

}

fin.close();

binf.close();

};

void correction(char\* binfile) {

binf.open(binfile, ios::binary | ios::in);

fout.open("corr.txt");

while (binf.read((char\*)&ct, sizeof(Cities))) {

fout << ct << "\n";

}

fout.close();

binf.close();

}

void rbinf(char\* binfile) {

cout << "\n\nЧтение файла по ПЗ #1:\n";

binf.open(binfile, ios::binary | ios::in);

while (binf.read((char\*)&ct, sizeof(Cities))) {

cout << ct << "\n";

}

binf.close();

}

void sortedBinary(char\* binfile) {

binf.open(binfile, ios::in | ios::out | ios::binary);

if (!binf) {

cout << "Error(binf)";

exit(0);

}

int size = 0;

Cities\* cities = new Cities[Count]();

while (binf.read((char\*)&ct, sizeof(Cities))) cities[size++] = ct;

binf.close();

sorted(cities, size);

binf.open(binfile);

if (!binf) {

cout << "Error(binf)";

exit(0);

}

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

binf.write((char\*)&cities[i], sizeof(Cities));

delete[] cities;

binf.close();

}

void sorted(Cities\* cities, int n) {

int i, j, step;

Cities tmp;

int stmp;

for (step = n / 2; step > 0; step /= 2)

for (i = step; i < n; i++){

tmp = cities[i];

stmp = cities[i].area;

for (j = i; j >= step; j -= step){

if (stmp < cities[j - step].area)

cities[j] = cities[j - step];

else

break;

}

cities[j] = tmp;

}

}

int main() {

setlocale(LC\_ALL, "ru");

createFile(filename);

rbinf(binfile);

createBin(filename, binfile);

rbinf(binfile);

}

Практическое занятие №3:

#ifndef lab\_hpp

#define lab\_hpp

#include <iostream>

#include <fstream>

#include <string>

#include <iomanip>

#include <vector>

#include <io.h>

#include <conio.h>

using namespace std;

ofstream fout;

ifstream fin;

fstream binf;

const int Count = 5;

char filename[10] = "test.txt";

char binfile[15] = "binfile.txt";

char delfile[11] = "def.txt";

struct Cities {

char name[10];

int people;

char country[10];

int created;

int area;

}ct;

void sorted(Cities\* cities, int n);

void filtr(char\* str) {

int x;

int xl = 0,

xr = strlen(str) - 1;

while (str[xl] == ' ') xl++;

while (str[xr] == ' ') xr--;

for (x = xl; x <= xr; x++)

str[x - xl] = str[x];

str[x - xl] = '\0';

}

ostream& operator << (ostream& out, const Cities& Cities) {

out << left

<< setw(10) << Cities.name

<< setw(10) << Cities.people

<< setw(10) << Cities.country

<< setw(10) << Cities.created

<< setw(10) << Cities.area;

return out;

}

istream& operator >> (istream& in, Cities& Cities) {

char T[255];

in.get(Cities.name, 10);

in >> Cities.people;

in.get(Cities.country, 10);

in >> Cities.created >> Cities.area;

filtr(Cities.name);

filtr(Cities.country);

in.getline(T, 255);

return in;

}

void createFile(char\* file) {

fout.open(file);

m: cout << "Введите все поля данные для одной структуры:\n";

cin >> ct;

fout << ct << "\n";

cout << "Для завершения нажмите '0'\nЕсли хотите продолжить, то введите любой другой симвойл:\n";

char c;

c = \_getch();

if (c == '0'){

fout.close();

return;

}

else goto m;

}

void createBin(char\* data, char\* binfile){

ifstream file(data);

if (!file.is\_open()){

cout << "Такого файла нет!\n";

system("pause");

exit(0);

}

fout.open(binfile, ios::binary);

while (!file.eof()){

file >> ct;

fout.write((char\*)&ct, sizeof(Cities));

}

fout.close();

file.close();

}

void RemoveLine(char\* filedel, char\* binfile) {

int z = sizeof(Cities);

cout << "Удаление из бинарного файла" << endl;

fin.open(filedel);

if (!fin) {

cout << "Error(def.txt)";

exit(0);

}

binf.open(binfile, ios::in | ios::out | ios::binary);

if (!binf) {

cout << "Error(binf)";

exit(0);

}

long size\_f = 0;

while (binf.peek() != EOF) {

binf.write((char\*)&ct, z);

size\_f += z;

}

string data;

vector<string> cache;

while (!fin.eof()) {

getline(fin, data);

if (data.empty()) continue;

else cache.push\_back(data);

}

for (auto it = cache.begin(); it != cache.end(); it++) {

binf.clear();

binf.seekg(0, ios::beg);

while (binf.peek() != EOF) {

binf.read((char\*)&ct, z);

if (ct.area == stoi(\*it)) {

while (binf.peek() != EOF) {

binf.read((char\*)&ct, z);

binf.seekp(-2 \* z, ios::cur); //смещаем указатель файла на одну структуру назад

binf.write((char\*)&ct, z); //запись в файл откорректированной структуры

binf.seekp(+z, ios::cur);

};

int handle = \_open(binfile, ios::in | ios::binary);

size\_f -= z;

\_chsize(handle, size\_f);

\_close(handle);

}

}

}

fin.close();

binf.close();

};

void correction(char\* binfile) {

binf.open(binfile, ios::binary | ios::in);

fout.open("corr.txt");

while (binf.read((char\*)&ct, sizeof(Cities))) {

fout << ct << "\n";

}

fout.close();

binf.close();

}

void rbinf(char\* binfile) {

cout << "\n\nЧтение файла по ПЗ #1:\n";

binf.open(binfile, ios::binary | ios::in);

while (binf.read((char\*)&ct, sizeof(Cities))) {

cout << ct << "\n";

}

binf.close();

}

void sortedBinary(char\* binfile) {

binf.open(binfile, ios::in | ios::out | ios::binary);

if (!binf) {

cout << "Error(binf)";

exit(0);

}

int size = 0;

Cities\* cities = new Cities[Count]();

while (binf.read((char\*)&ct, sizeof(Cities))) cities[size++] = ct;

binf.close();

sorted(cities, size);

binf.open(binfile);

if (!binf) {

cout << "Error(binf)";

exit(0);

}

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

binf.write((char\*)&cities[i], sizeof(Cities));

delete[] cities;

binf.close();

}

void sorted(Cities\* cities, int n) {

int i, j, step;

Cities tmp;

int stmp;

for (step = n / 2; step > 0; step /= 2)

for (i = step; i < n; i++){

tmp = cities[i];

stmp = cities[i].area;

for (j = i; j >= step; j -= step){

if (stmp < cities[j - step].area)

cities[j] = cities[j - step];

else

break;

}

cities[j] = tmp;

}

}

void outputMenu() {

cout << "\n МЕНЮ: \n"

<< "\nСоздание файла------------------------------ 1"

<< "\nЧтение файла-------------------------------- 2"

<< "\nСортировка файла---------------------------- 3"

<< "\nУдаление данных из файла-------------------- 4"

<< "\nВыход--------------------------------------- 5";

cout << "\n\nДля выполнения операции введите нажмите нужную клавишу: ";

}

void printHeaderTable() {

string header[5] = { " Города ",

"|================|================|===========|============|================|",

"| Название | Численность |Страна | Год | Площадь |",

"| | населения | | создания | кв.км |",

"|================|================|===========|============|================|",

};

for (int i = 0; i < 5; i++) cout << header[i] << "\n";

}

void printFooterTabele() {

cout << "|================|================|===========|============|================|" << endl;

}

void printLineTable(Cities city) {

cout << left <<

"|" << setw(16) << city.name <<

"|" << setw(16) << city.people <<

"|" << setw(11) << city.country <<

"|" << setw(12) << city.created <<

"|" << setw(16) << city.area <<

"|" << endl;

}

void CheckEdit(char\* binfile) {

printHeaderTable();

binf.open(binfile, ios::binary | ios::in);

while (binf.read((char\*)&ct, sizeof(Cities))) {

printLineTable(ct);

}

binf.close();

printFooterTabele();

}

void Create() {

system("cls");

//createFile(Text);

createBin(filename, binfile);

}

void Read() {

system("cls");

CheckEdit(binfile);

}

void Sorted() {

system("cls");

sortedBinary(binfile);

correction(binfile);

CheckEdit(binfile);

}

void Remove() {

system("cls");

RemoveLine(delfile, binfile);

correction(binfile);

CheckEdit(binfile);

}

void Exit() { exit(1); }

int inputNumber() {

int a;

(cin >> a).get();

return a;

}

void(\*Processing[5])() = { Create, Read, Sorted, Remove, Exit };

#endif

#include "lab.hpp"

void filtr(char\* str);

ostream& operator << (ostream& out, const Cities& Cities);

istream& operator >> (istream& in, Cities& Cities);

void createFile(char\* file);

void createBin(char\* data, char\* binfile);

void RemoveLine(char\* filedel, char\* binfile);

void correction(char\* binfile);

void rbinf(char\* binfile);

void sortedBinary(char\* binfile);

void sorted(Cities\* cities, int n);

void outputMenu();

void printHeaderTable();

void printFooterTabele();

void printLineTable(Cities city);

void CheckEdit(char\* binfile);

void Create();

void Read();

void Sorted();

void Remove();

void Exit() { exit(1); }

int inputNumber();

void(\*Processing[5])() = { Create, Read, Sorted, Remove, Exit };

#include "lab.hpp"

int main() {

while (true) {

outputMenu();

Processing[inputNumber()-1]();

}

}

Практическое занятие №4:

#include <iostream>

#include <fstream>

#include <vector>

#include <iomanip>

#include <string>

std::ofstream write;

std::ifstream read;

const char ch[] = "-----------------";

void TwoDigitOut(const char\* file) {

read.open(file);

if (!read) throw std::string("\ncant open\n");

std::string line;

std::string nline;

while (!read.eof()) {

getline(read, line);

for (int i = 0; i < line.length(); i++) {

if (line[i] == ' ') {

if (nline.size() == 2 && isdigit(nline[0]) && isdigit(nline[1])) {

std::cout << nline << "\n";

}

nline = "";

}

else {

nline += line[i];

}

}

}

if (nline.size() == 2 && isdigit(nline[0]) && isdigit(nline[1])) {

std::cout << nline << "\n";

}

read.close();

}

void number2() {

std::vector<int> rands;

std::vector<std::string> words;

std::string line;

std::string nline;

srand(time(0));

read.open("n2.txt");

if (!read) throw std::string("\ncant open\n");

while (!read.eof()) {

getline(read, line);

for (int i = 0; i < line.length(); i++) {

if (line[i] == ' ') {

rands.push\_back(rand() % 1000);

words.push\_back(nline);

nline = "";

}

else {

nline += line[i];

}

}

}

rands.push\_back(rand() % 1000);

words.push\_back(nline);

read.close();

write.open("coded.txt");

std::cout << "| key | word |\n";

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

write << rands[i] << "\n";

std::cout << "|" << std::setw(5) << rands[i] << "|" << std::setw(8) << words[i] << "|\n";

}

write.close();

std::cout << "\namount of words: " << rands.size() << "\n";

read.open("coded.txt");

if (!read) throw std::string("\ncant open\n");

std::cout << "\nprotocol:\n";

std::string saver;

while (!read.eof()) {

getline(read, saver);

if (saver == "") break;

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

if (rands[i] == std::stoi(saver)) {

std::cout << words[i] << " ";

}

}

}

std::cout << "\n";

read.close();

}

int main() {

try {

std::cout << ch << "\n" << "\tn1\n";

TwoDigitOut("text.txt");

std::cout << ch << "\n" << "\tn2\n";

number2();

}

catch (std::string& e) {

std::cerr << e;

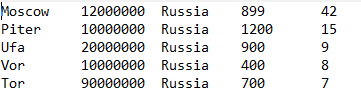
}

}

Приложение 2. Содержание файлов для тестирования:

Практическое занятие №1:

Файл: text.txt

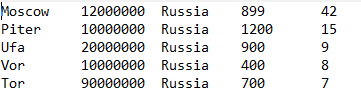


Файл: def.txt



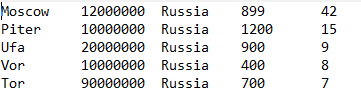
Практическое занятие №2:

Файл: text.txt



Практическое занятие №3:

Файл: text.txt

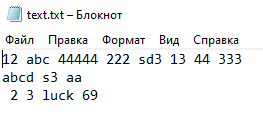


Файл: def.txt

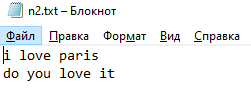


Практическое занятие №4:

Файл: text.txt



Файл: n2.txt



Приложение 3. Результат выполнения:

Практическое занятие №1:

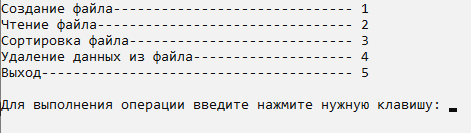


Практическое занятие №2:



Практическое занятие №3:

Главное меню:



Чтение файла:



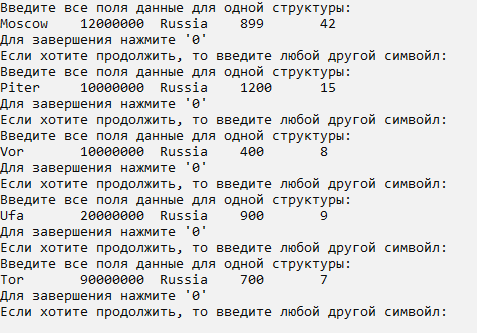
Сортировка файла:



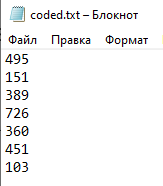
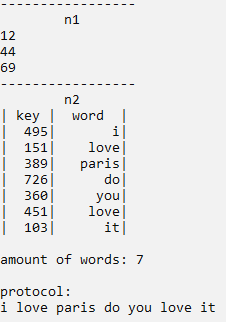
Удаление данных из файла:



Создание файла:



Практическое занятие №4:

Консоль: Закодированный текст: