Министерство образования Республики Беларусь

Учреждение образования

Брестский государственный технический университет

Кафедра ИИТ

Лабораторная работа №12

За 2 семестр

По дисциплине: ОАиП

Тема: “Бинарные и текстовые файлы”

Подготовил:

Студент 1 курса

Группы АС-59

Левоцкий Н.

Цыганов Е.

Проверил:

Гирель Т.Н.

Брест 2021

Цель работы: Изучить принципы программирования с использованием

бинарных файлов. Ознакомиться с основными функциями в Си для работы с

бинарными файлами.

Задание:

В программу разработанную в прошлой лабораторной работе добавить чтение и сохранение данных массива структур при помощи бинарных файлов следующим образом:

1. При первом запуске программы должен создаваться бинарный или текстовый файл на выбор пользователя для хранения данных

из массива структур.

2.При добавлении новой записи в массив структур в файл должна дописываться новая запись, без изменения остальных записей.

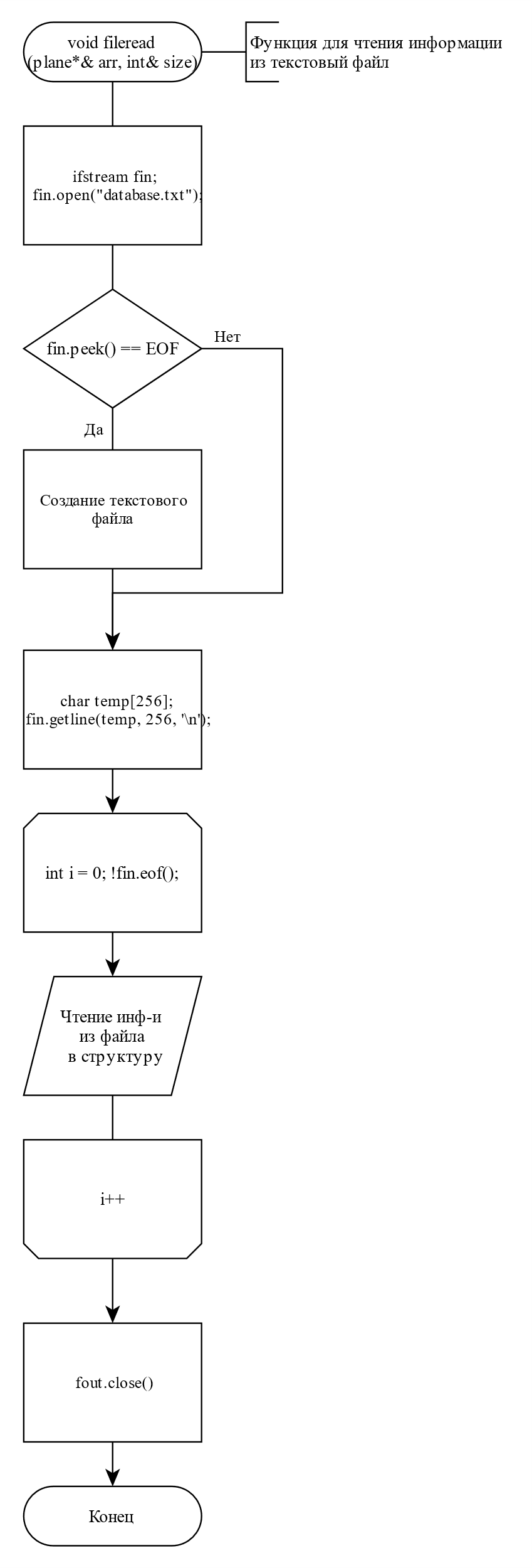
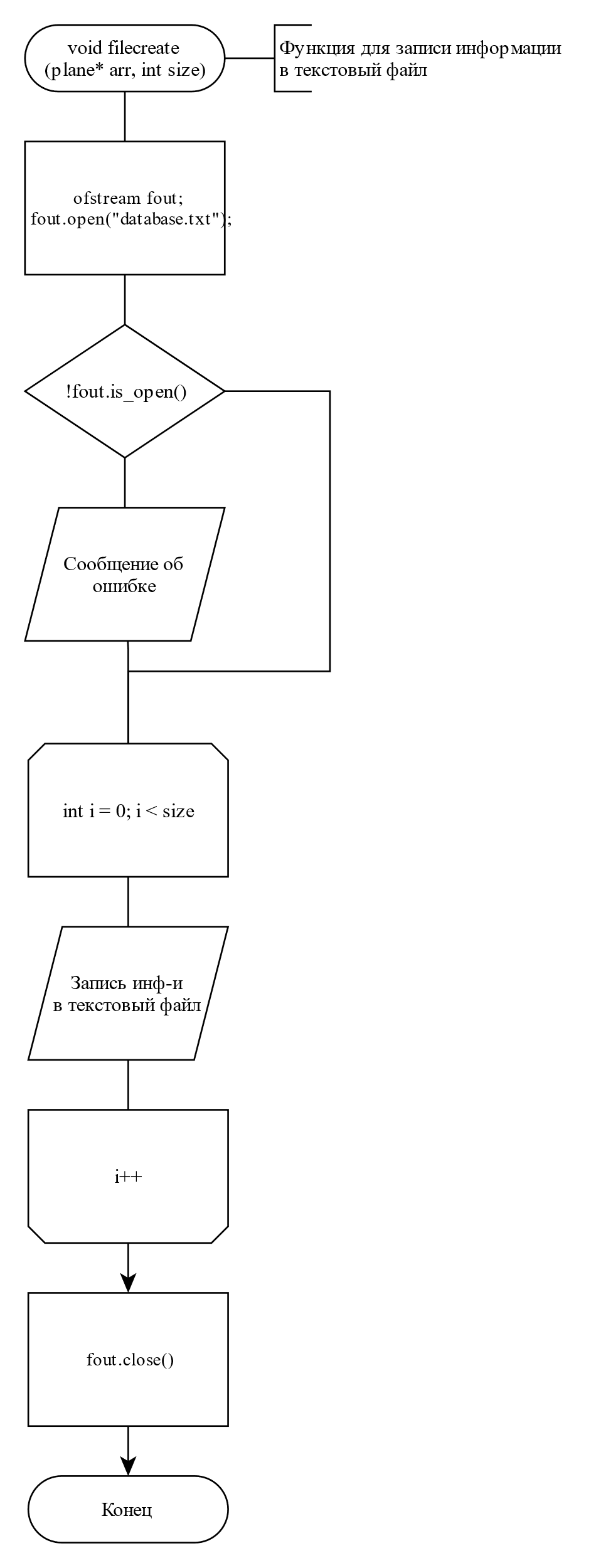
3. При повторном запуске программы, если файл уже существует, то информация в массив структур должна читаться из этого файла. Если файл отсутствует, то он должен создаваться (см. Пункт 1).

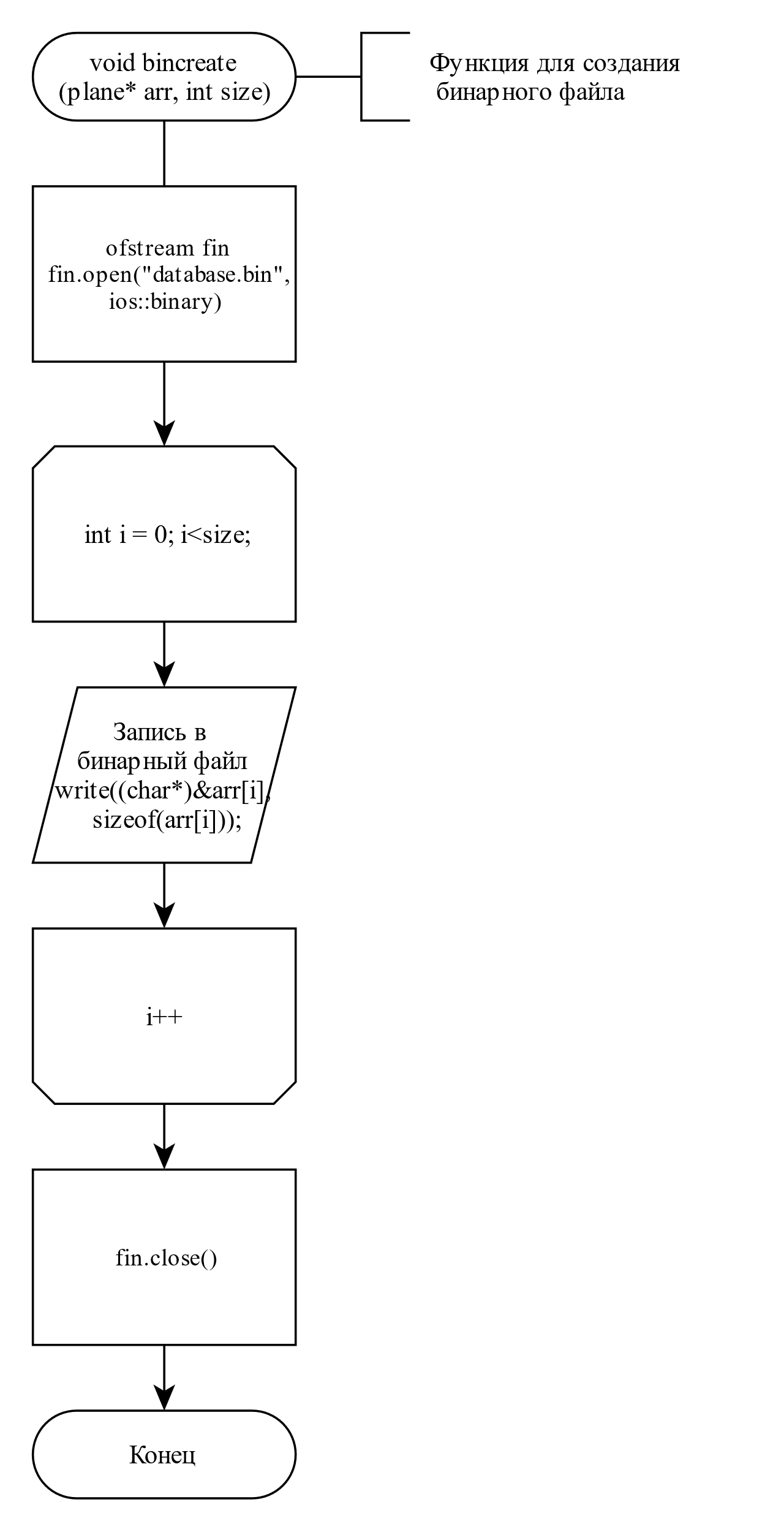
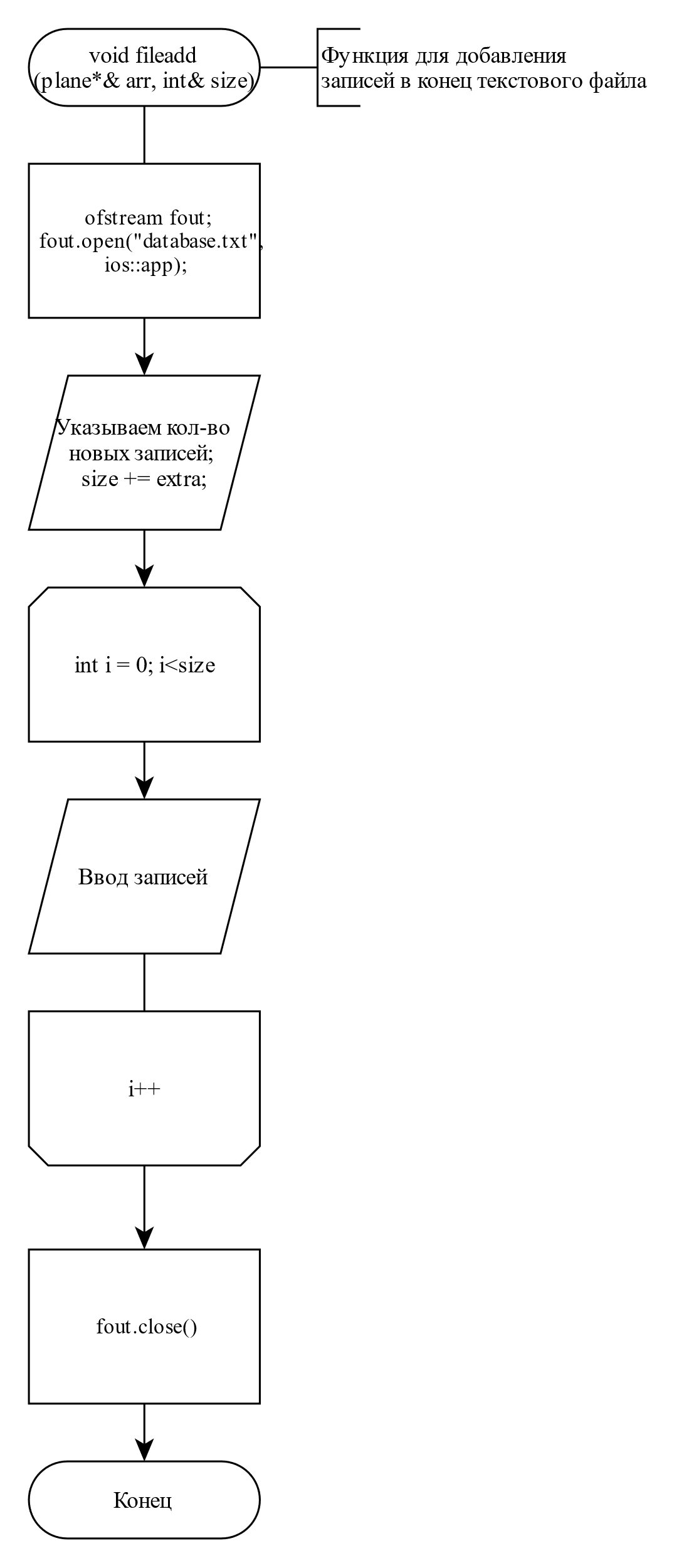
4. Все изменения (сортировка, изменения полей записи, удаление записи) – сохраняются в файле при помощи полной перезаписи содержимого.

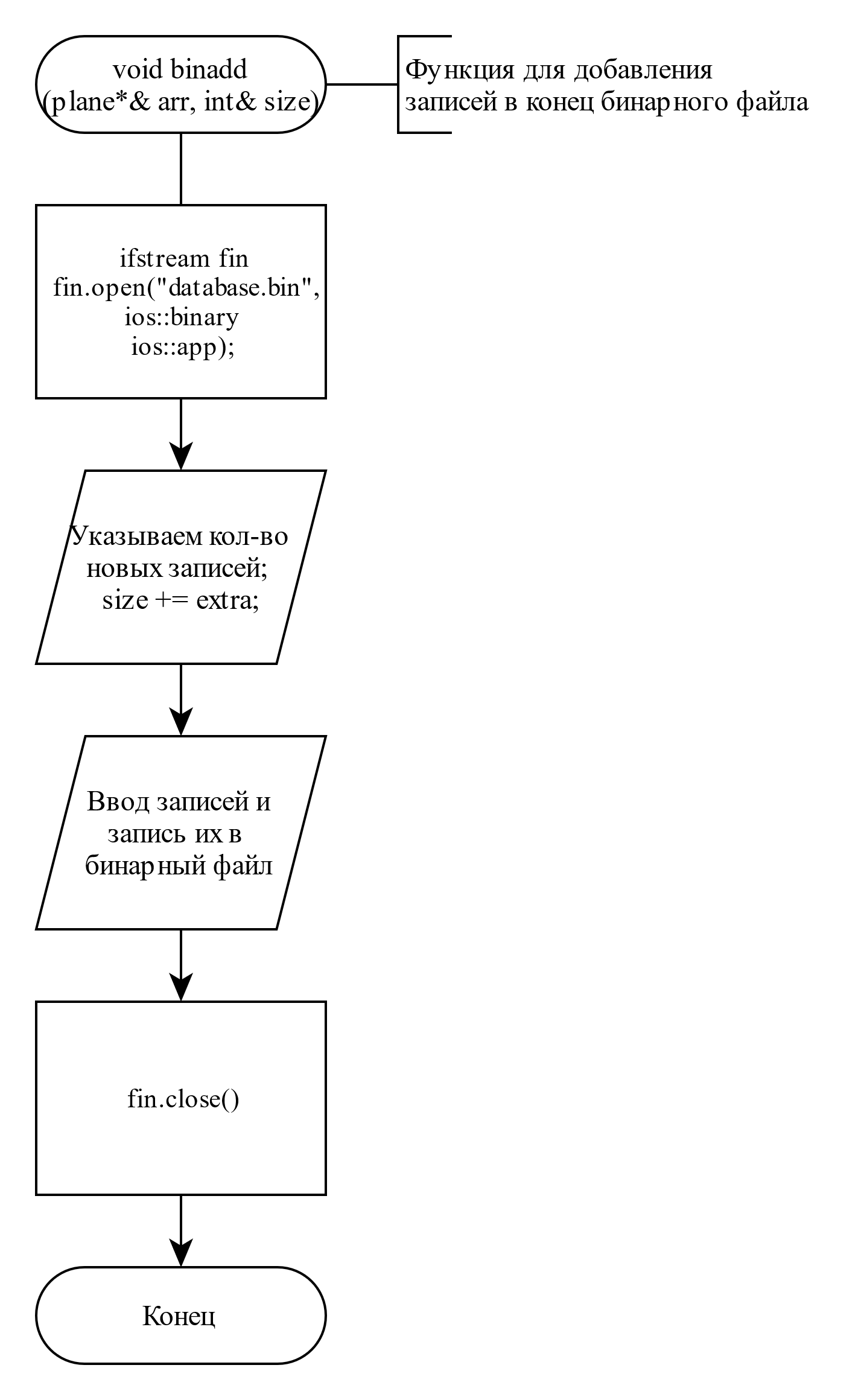
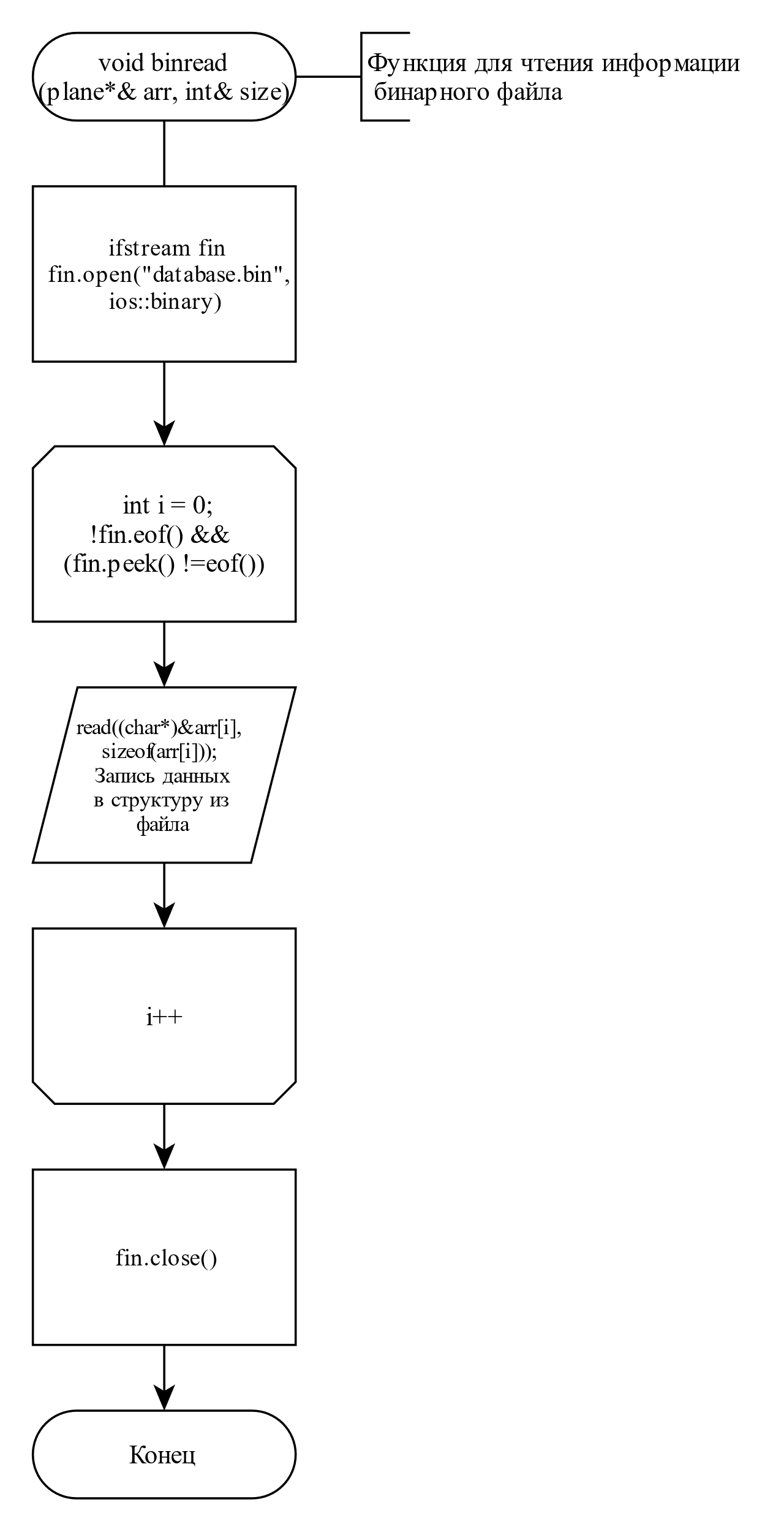
5. Сделать вывод о том, какие преимущества использования конкретного типа файлов (бинарные или текстовые) в решаемой вами задаче.

**Вариант 12**

Блок-схемы:







**Код программы:**

#include<iostream>

#include<Windows.h>

#include<iomanip>

#include<stdlib.h>

#include<fstream>

using namespace std;

struct plane

{

int num;

union inFlight //объединение

{

int key;

char ans[4];

}inf;

enum Type //перечисление

{

pass = 1,

trans,

mail,

cour

};

char typedes[15];

short hour;

short min;

char loc[30];

short time;

};

void menu(plane\* arr, int size); //меню

void input(plane\*& arr, int &size); //ввод

void output(plane\* arr, int size); //вывод

void del(plane\*& arr, int& size); //удаление

void sort(plane\* arr, int size); //сортировка

void task(plane\* arr, int size); //задание

void fileadd(plane\*& arr, int& size); //добавление

void filecreate(plane\* arr, int size);

void fileread(plane\*& arr, int& size);

void bincreate(plane\* arr, int size);

void binread(plane\*& arr, int& size);

void binadd(plane\*& arr, int& size);

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int size = 0;

plane\* mass = new plane[size];

menu(mass,size);

delete[] mass;

return 0;

}

void menu(plane\* arr, int size)

{

int ch;

int fileCh;

cout << "Тип файла: 1-текстовый \\ 2-бинарный\n";

cin >> fileCh;

do {

cout << "Меню: \n";

cout << "1 - чтение данных\n";

cout << "2 - вывод данных\n";

cout << "3 - удаление записи по номеру\n";

cout << "4 - сортировать по времени вылета\n";

cout << "5 - Вывод номеров рейсов, вылетающих в данный пункт назначения\n";

cout << "6 - добавление записи\n";

cout << "0 - завершить работу с меню\n";

cout << "Выберите пункт: "; cin >> ch;

switch (ch)

{

case 1:

if (fileCh == 1)

{

system("cls");

fileread(arr, size);

break;

}

if (fileCh == 2)

{

system("cls");

binread(arr, size);

break;

}

case 2:

system("cls");

output(arr, size);

break;

case 3:

if (fileCh == 1)

{

system("cls");

del(arr, size);

filecreate(arr, size);

break;

}

if (fileCh == 2)

{

system("cls");

del(arr, size);

bincreate(arr, size);

break;

}

case 4:

if (fileCh == 1)

{

system("cls");

sort(arr,size);

filecreate(arr, size);

break;

}

if (fileCh == 2)

{

system("cls");

sort(arr, size);

bincreate(arr, size);

break;

}

case 5:

system("cls");

task(arr, size);

break;

case 6:

if (fileCh == 1)

{

system("cls");

fileadd(arr, size);

break;

}

if (fileCh == 2)

{

system("cls");

binadd(arr, size);

break;

}

case 0:break;

default: cout << "Неверный ввод";

break;

}

} while (ch != 0);

}

void input(plane\*& arr, int& size)

{

int N;

cout << "Введите кол-во записей: "; cin >> N;

size = size + N;

plane\* tmp = new plane[size ];

for (int j = 0; j < size - N; j++)

tmp[j] = arr[j];

arr = new plane[size];

for (int j = 0; j < size - N; j++)

arr[j] = tmp[j];

delete[] tmp;

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

{

arr[i].num = i + 1;

cout << "Номер самолёта: " << i+1 << endl;

int type;

cout << "Тип самолёта (1-Пассажирский, 2-Транспортный, 3-Почтовый, 4-Курьерский): "; cin >>type;

switch (type)

{

case (arr[i].pass):

{

strcpy(arr[i].typedes, "Пассажирский");

}break;

case (arr[i].trans):

{

strcpy(arr[i].typedes, "Транспортный");

}break;

case (arr[i].mail):

{

strcpy(arr[i].typedes, "Почтовый");

}break;

case (arr[i].cour):

{

strcpy(arr[i].typedes, "Курьерский");

}break;

}

cin.ignore();

cout << "Пункт назначения: ";

gets\_s(arr[i].loc);

cout << "Время вылета (часы : минуты): \n";

cin >> arr[i].hour;

cin >> arr[i].min;

cout << "В полёте (1-Да, 2-Нет): "; cin >> arr[i].inf.key;

if (arr[i].inf.key == 1)

strcpy(arr[i].inf.ans, "Да");

if (arr[i].inf.key == 2)

strcpy(arr[i].inf.ans, "Нет");

cout << endl;

}

}

void output(plane\* arr, int size)

{

cout << left << setw(8) << "|Номер" << setw(14) << left << "|Тип самолёта" << setw(19) << left << "|Пункт назначения" << setw(14) << left << "|Время вылета" << setw(10) << left << "|В полёте|" << endl;

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

{

cout << left << setw(9) << arr[i].num << setw(15) << left << arr[i].typedes << setw(20) << left << arr[i].loc << setw(2) << left <<arr[i].hour<<":"<<setw(12)<<left<<arr[i].min << setw(10) << left << arr[i].inf.ans << endl;

}

}

void del(plane\*& arr, int& size)

{

int val;

cout << "Номер записи, которую удалить- "; cin >> val;

unsigned int l = 0, removed = 0;

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

{

if (arr[i].num != val)

arr[l++] = arr[i];

else

removed++;

}

size -= removed;

plane\* tmp = new plane[size];

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

tmp[j] = arr[j];

arr = new plane[size];

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

arr[j] = tmp[j];

delete[] tmp;

}

void sort(plane\* arr, int size)

{

for (int i = 0; i < size-1; i++)

{

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

{

if (arr[i].hour > arr[j].hour)

{

swap(arr[i], arr[j]);

}

}

}

}

void task(plane\* arr, int size)

{

char check[30];

cout << "Введит пункт назначения- ";

cin.ignore();

gets\_s(check);

cout << "Номера рейсов:\n";

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

{

if (strcmp(check, arr[i].loc) == 0)

cout << arr[i].num << endl;

}

}

void fileadd(plane\*& arr, int& size)

{

ofstream fout;

fout.open("database.txt", ios\_base::app);

int extra;

cout << "Сколько элементов добавить: "; cin >> extra;

size += extra;

if (fout.is\_open())

{

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

{

if (i >= (size - extra))

{

arr[i].num = size;

cout << "Номер самолёта: " << i + 1 << endl;

int type;

cout << "Тип самолёта (1-Пассажирский, 2-Транспортный, 3-Почтовый, 4-Курьерский): "; cin >> type;

switch (type)

{

case (arr[i].pass):

{

strcpy(arr[i].typedes, "Пассажирский");

}break;

case (arr[i].trans):

{

strcpy(arr[i].typedes, "Транспортный");

}break;

case (arr[i].mail):

{

strcpy(arr[i].typedes, "Почтовый");

}break;

case (arr[i].cour):

{

strcpy(arr[i].typedes, "Курьерский");

}break;

}

cin.ignore();

cout << "Пункт назначения: ";

gets\_s(arr[i].loc);

cout << "Время вылета (часы : минуты): \n";

cin >> arr[i].time;

arr[i].hour = arr[i].time;

cin >> arr[i].time;

arr[i].min = arr[i].time;

cout << "В полёте (1-Да, 2-Нет): "; cin >> arr[i].inf.key;

if (arr[i].inf.key == 1)

strcpy(arr[i].inf.ans, "Да");

if (arr[i].inf.key == 2)

strcpy(arr[i].inf.ans, "Нет");

fout << left << setw(9) << (arr[i].num) << setw(15) << left << arr[i].typedes << setw(20) << left << arr[i].loc << setw(2) << left << arr[i].hour << " " << setw(12) << left << arr[i].min << setw(10) << left << arr[i].inf.ans << endl;

cout << endl;

}

}

}

fout.close();

}

void filecreate(plane\* arr, int size)

{

ofstream fout;

fout.open("database.txt");

if (!fout.is\_open())

{

cerr << "Oшибка открытия" << endl;

}

fout << left << setw(8) << "|Номер" << setw(14) << left << "|Тип самолёта" << setw(19) << left << "|Пункт назначения" << setw(14) << left << "|Время вылета" << setw(10) << left << "|В полёте|" << endl;

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

{

fout << left << setw(9) << arr[i].num << setw(15) << left << arr[i].typedes << setw(20) << left << arr[i].loc << setw(2) << left << arr[i].hour << " " << setw(12) << left << arr[i].min << setw(10) << left << arr[i].inf.ans << endl;

}

fout.close();

cout << "Информация записана в файл" << endl;

}

void fileread(plane\*& arr, int& size)

{

size = 0;

ifstream fin;

fin.open("database.txt");

if (fin.peek() == EOF)

{

cout << "Файл пуст или отстутсвует\n";

input(arr, size);

filecreate(arr, size);

}

else

{

arr = new plane[size];

char temp[256];

fin.getline(temp, 256, '\n');

for (int i = 0; !fin.eof(); i++) {

size++;

plane\* tempArr = new plane[size - 1];

for (int j = 0; j < size - 1; j++) {

tempArr[j] = arr[j];

}

arr = new plane[size];

for (int j = 0; j < size - 1; j++) {

arr[j] = tempArr[j];

}

delete tempArr;

fin >> arr[i].num;

fin >> arr[i].typedes;

fin >> arr[i].loc;

fin >> arr[i].hour;

fin >> arr[i].min;

fin >> arr[i].inf.ans;

}

size-= 1;

cout << "Данные считаны из файла.\n";

cin.ignore();

fin.close();

}

}

void bincreate(plane\* arr, int size)

{

ofstream fout;

fout.open("database.bin", ios\_base::binary);

if (!fout.is\_open())

{

cerr << "Oшибка открытия" << endl;

}

else

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

{

fout.write((char\*)&arr[i], sizeof(arr[i]));

}

fout.close();

}

void binread(plane\*& arr, int& size)

{

size = 0;

ifstream fin;

fin.open("database.bin", ios::binary);

if (!fin.is\_open())

{

input(arr, size);

bincreate(arr, size);

}

else

{

for (int i = 0; !fin.eof() && (fin.peek() != std::ifstream::traits\_type::eof()); i++)

{

size++;

plane\* tempArr = new plane[size - 1];

for (int j = 0; j < size - 1; j++) {

tempArr[j] = arr[j];

}

arr = new plane[size];

for (int j = 0; j < size - 1; j++) {

arr[j] = tempArr[j];

}

delete tempArr;

fin.read((char\*)&arr[i], sizeof(arr[i]));

}

cout << "Данные считаны из файла.\n";

cin.ignore();

fin.close();

}

}

void binadd(plane\*& arr, int& size)

{

ofstream fout;

fout.open("database.bin", ios\_base::binary | ios::app);

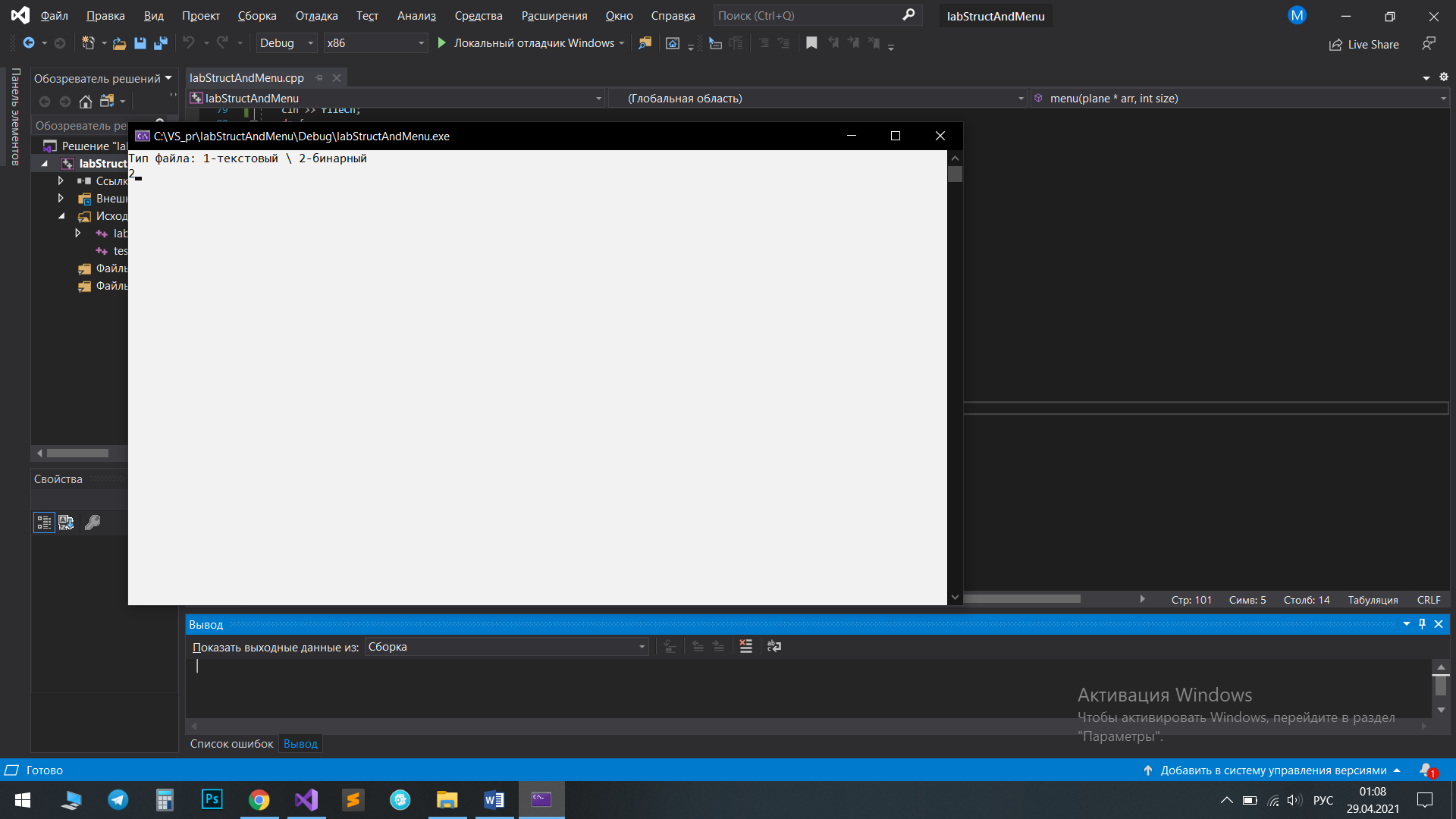
input(arr, size);

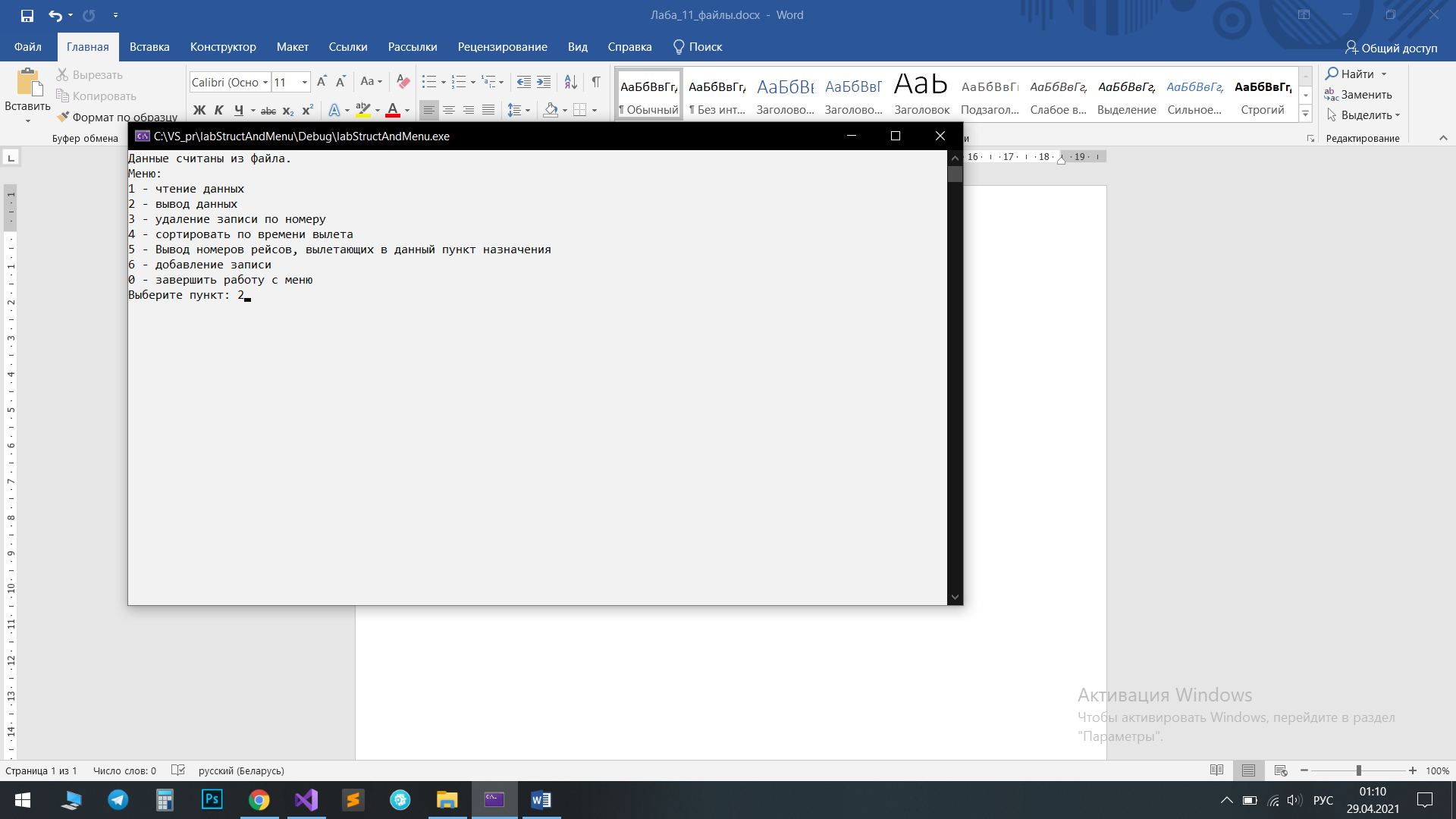
bincreate(arr, size);

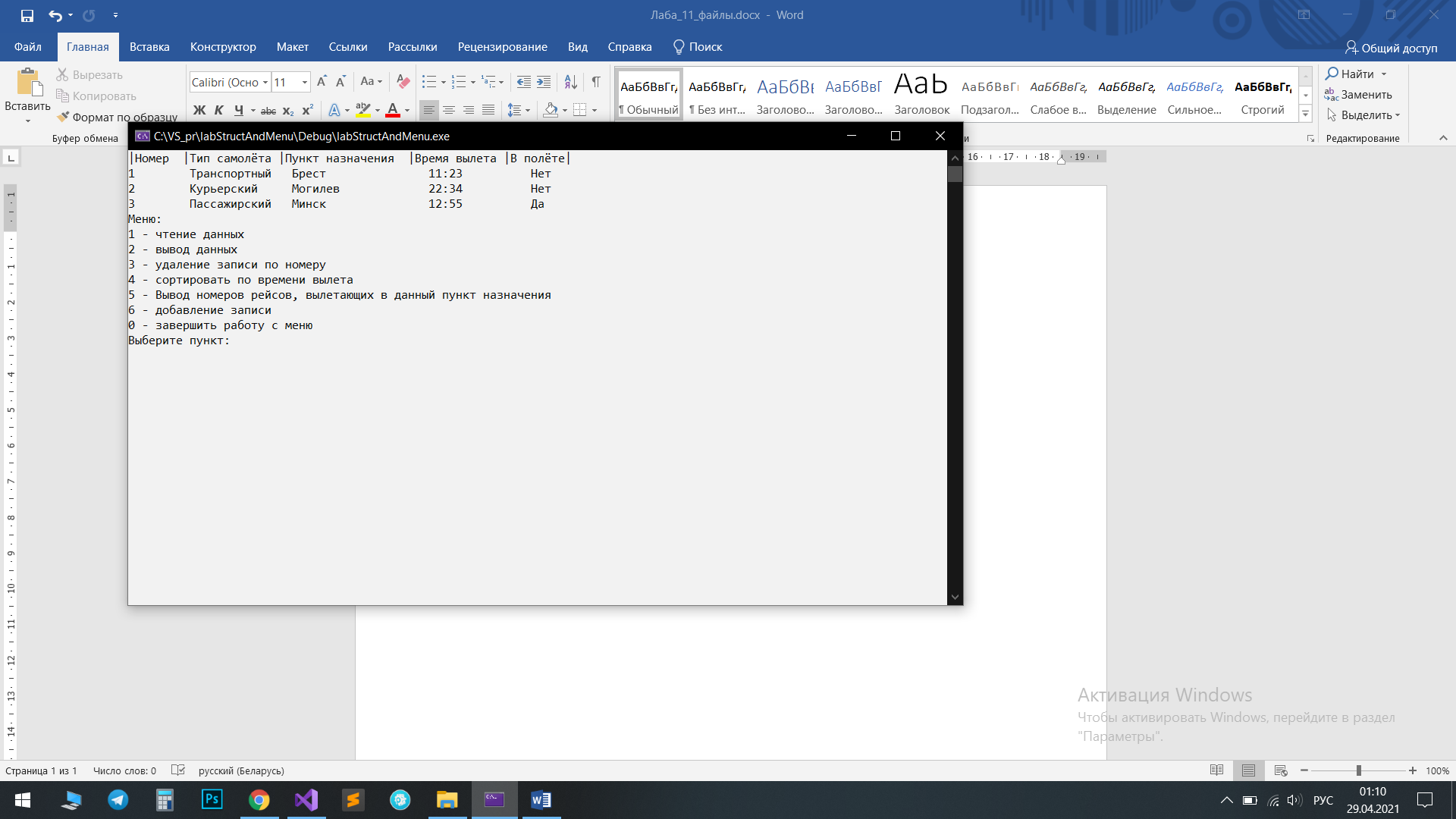
fout.close();

}

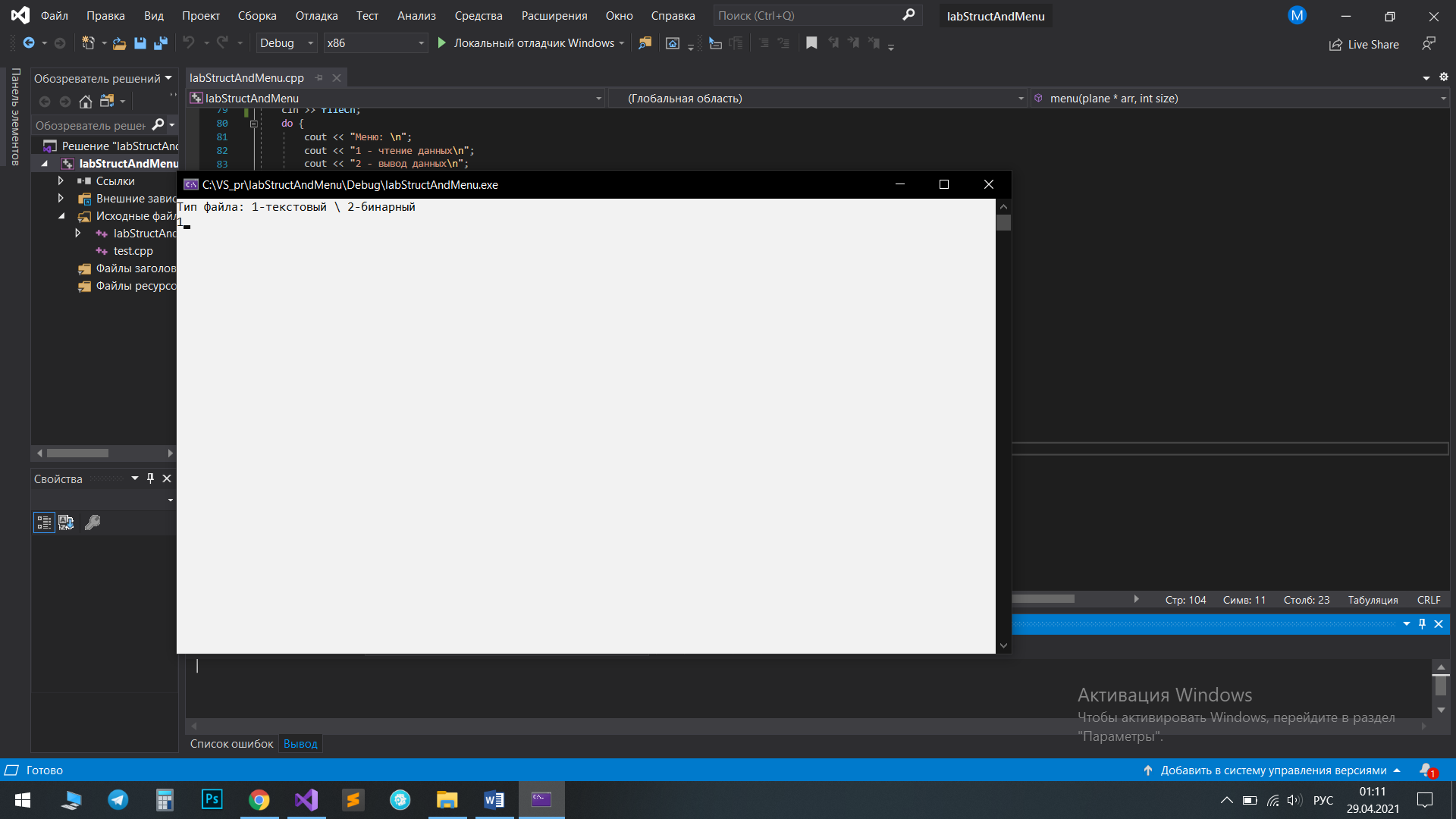
Бинарный

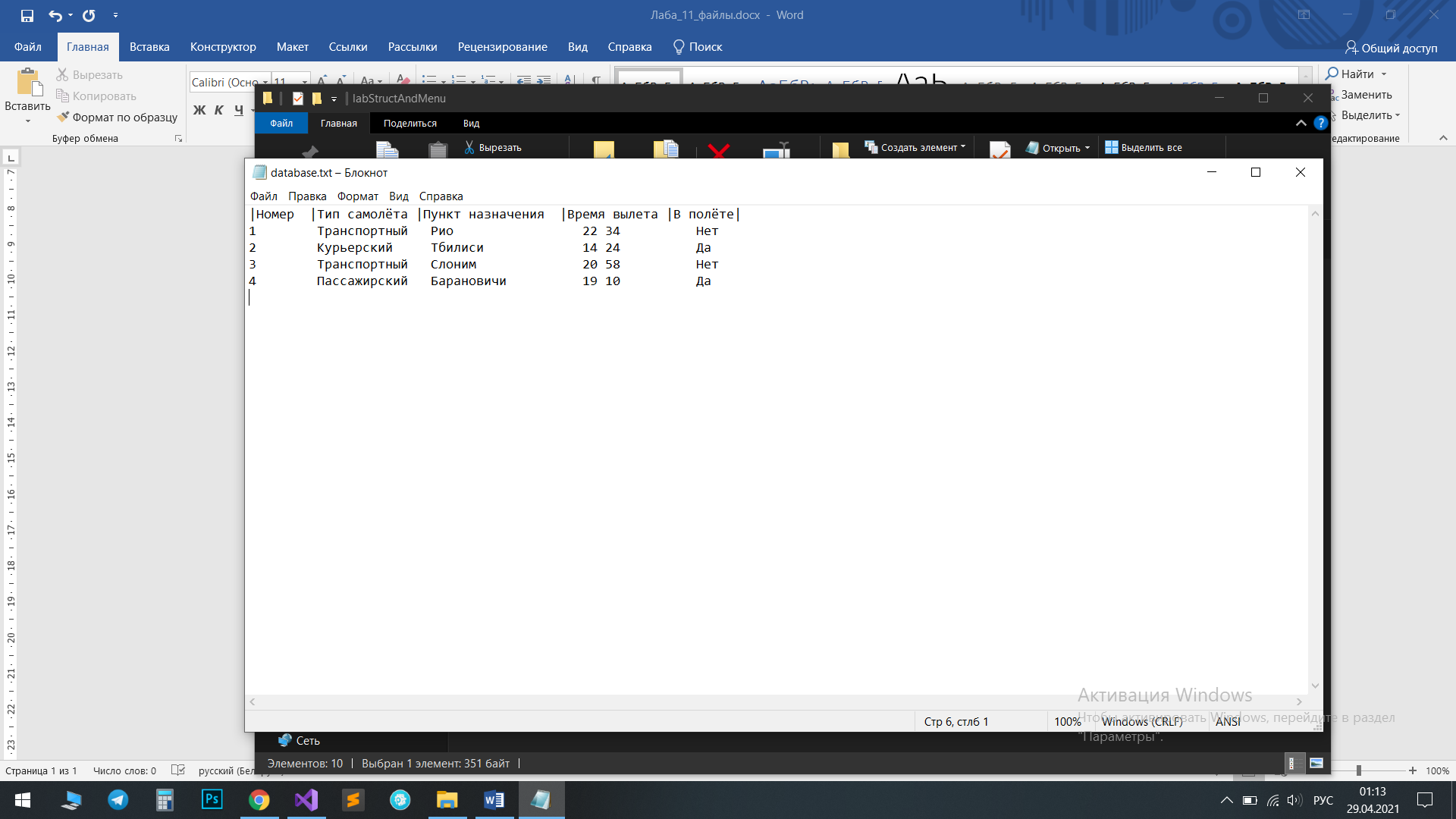


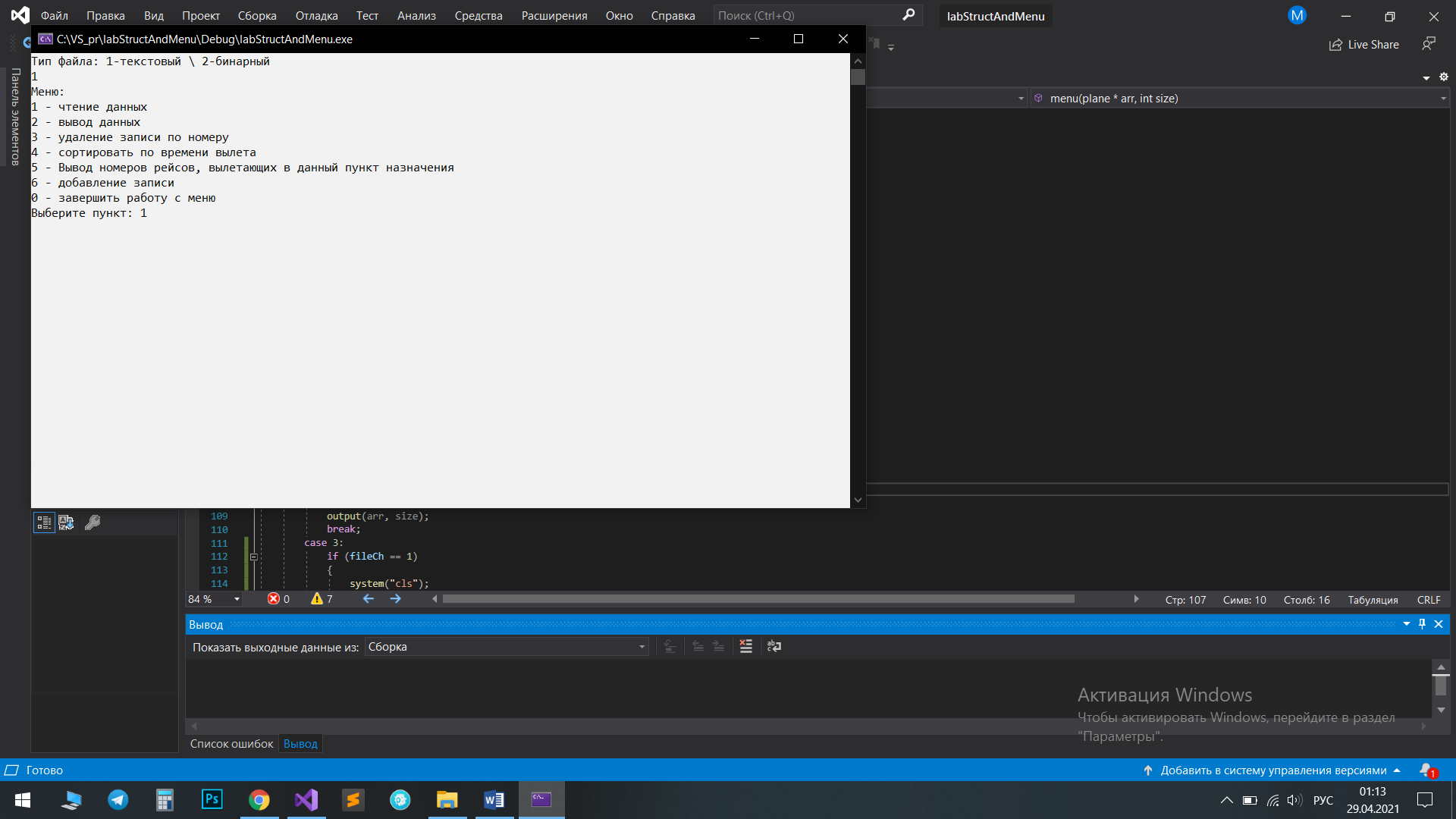


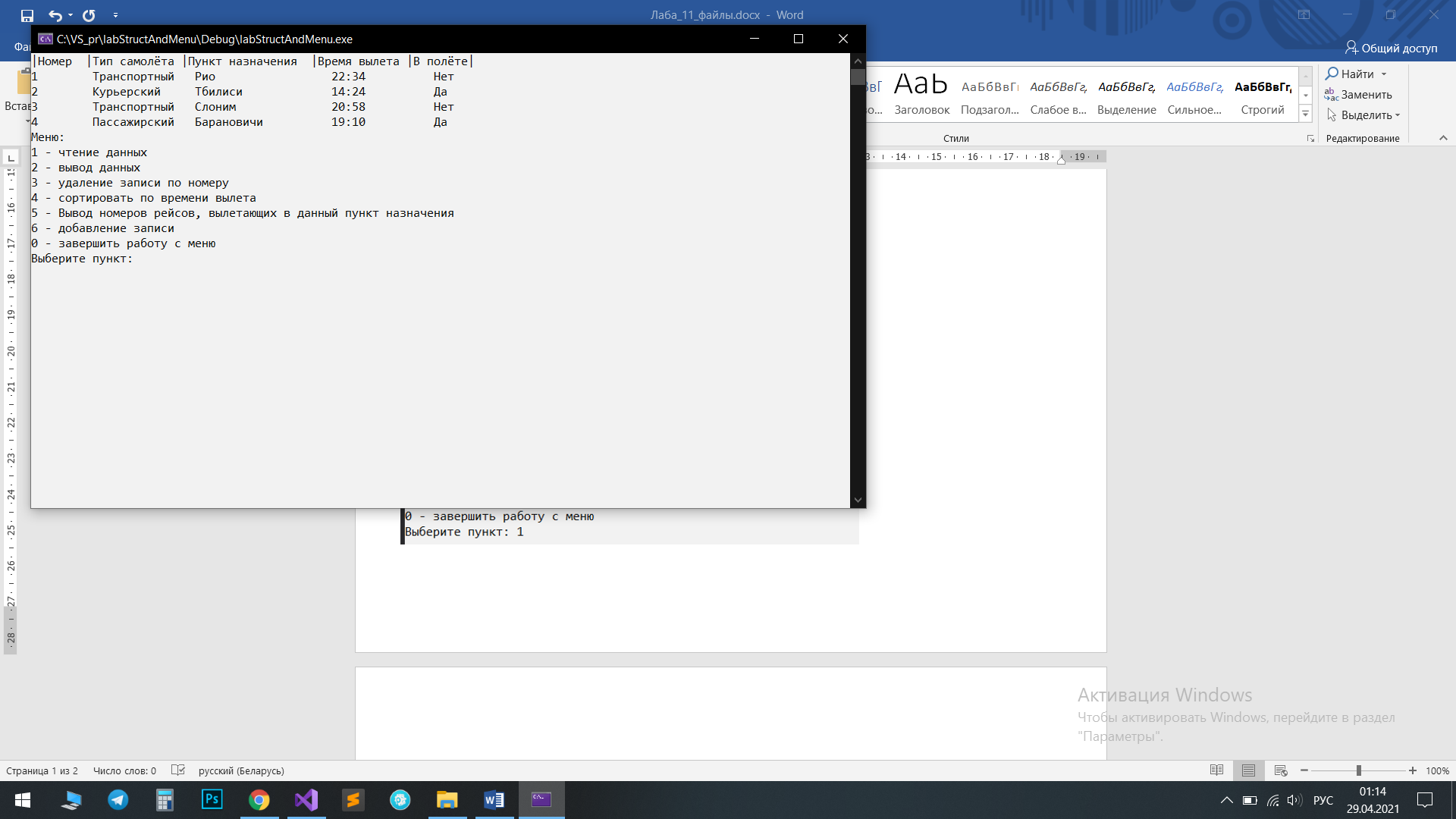


Текстовый









**Вывод**: Бинарный файл удобно использовать, когда нужно ограничить доступ пользователя к манипуляциям над базой данных, в частности массивом структур. Текстовый файл, однако,позволяет просмотреть и отредактировать файл без сторонних программ, а, например, с помощью блокнота.