Образец оформления отчёта по Лабораторной работе.

|  |  |
| --- | --- |
| **К Г Э У** | МИНИСТЕРСТВО ВЫСШЕГО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«КАЗАНСКИЙ ГОСУДАРСТВЕННЫЙ ЭНЕРГЕТИЧЕСКИЙ УНИВЕРСИТЕТ»**  (ФГБОУ ВО «КГЭУ») |

**Кафедра Информатики и информационных управляющих систем**

**ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 12**

**ДИНАМИЧЕСКОЕ МАССИВЫ СТУКТУРНЫХ ПЕРЕМЕННЫХ. РАБОТА С ФАЙЛАМИ.**

|  |  |
| --- | --- |
| **Исполнитель:** | Банницин Дмитрий |
| **Группа:** | ТРП-3-22 |
| **Вариант:** | 35 |
|  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

Казань -2022

**Задания для самостоятельной работы**

**Задача №1**

Составить программу, выводящую на экран ведомость начисленной заработной платы (Ф.И.О., должность, года и дата рождения, заработная плата). Вывести фамилию сотрудника с самой большой зарплатой.

**Код**

#include <iostream>

#include <time.h>

#include <fstream>

using namespace std;

struct statement {

string FIO;

string job;

int birthdayYear;

string birthday;

int pay;

};

string getRandFio();

string getRandJob();

string getRandDate();

void GetWorkers(statement\* workers, int count);

int getMaxPay(statement\* workers);

int main()

{

setlocale(LC\_ALL, "Russian");

srand(time(0));

statement\* workers = new statement[10];

GetWorkers(workers, 10);

ofstream fout;

fout.open("result.txt");

for (size\_t i = 0; i < 10; i++)

{

cout << i + 1 << ") FIO: " << workers[i].FIO << endl;

cout << "\t Job: " << workers[i].job << endl;

cout << "\t Birthday: " << workers[i].birthday << " " << workers[i].birthdayYear << "г." << endl;

cout << "\t Pay: " << workers[i].pay << endl;

fout << i + 1 << ") FIO: " << workers[i].FIO << endl;

fout << "\t Job: " << workers[i].job << endl;

fout << "\t Birthday: " << workers[i].birthday << " " << workers[i].birthdayYear << "г." << endl;

fout << "\t Pay: " << workers[i].pay << endl;

}

cout << "\nMax pay = " << getMaxPay(workers) << endl;

fout << "\nMax pay = " << getMaxPay(workers) << endl;

}

string getRandFio() {

string fios[10] = { "Пестов Давид Лукьянович",

"Авдеев Виктор Игоревич",

"Филатов Борис Михаилович",

"Самойлов Вольдемар Альбертович",

"Русаков Максимилиан Матвеевич",

"Харитонов Михаил Святославович",

"Галкин Лаврентий Святославович",

"Лобанов Рудольф Кириллович",

"Шарапов Велорий Михаилович",

"Корнилов Любовь Дамирович" };

return fios[rand() % 10];

}

string getRandJob() {

string jobs[10] = { "разработчик",

"архитектор",

"аналитик",

"веб-дизайнер",

"тестировщик",

"инженер",

"системный администратор",

"специалист по работе с Big Data",

"специалисты в области искусственного интеллекта",

"юрист" };

return jobs[rand() % 10];

}

string getRandDate() {

string dates[10] = { "1 декабря",

"2 января",

"3 февраля",

"4 марта",

"5 апреля",

"6 мая",

"7 июня",

"8 июля",

"9 августа",

"10 сентября" };

return dates[rand() % 10];

}

void GetWorkers(statement\* workers, int count) {

for (size\_t i = 0; i < 10; i++)

{

workers[i].FIO = getRandFio();

workers[i].job = getRandJob();

workers[i].birthdayYear = rand() % 50 + 1950;

workers[i].birthday = getRandDate();

workers[i].pay = rand() % 50000 + 30000;

}

}

int getMaxPay(statement\* workers) {

int maxPay = 0;

for (size\_t i = 0; i < 10; i++)

{

if (workers[i].pay > maxPay)

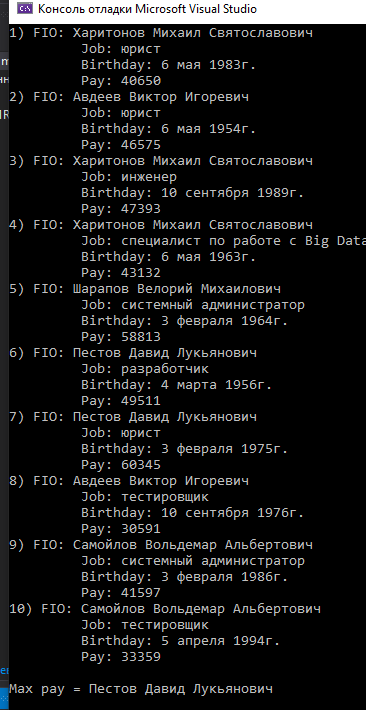
maxPay = workers[i].pay;

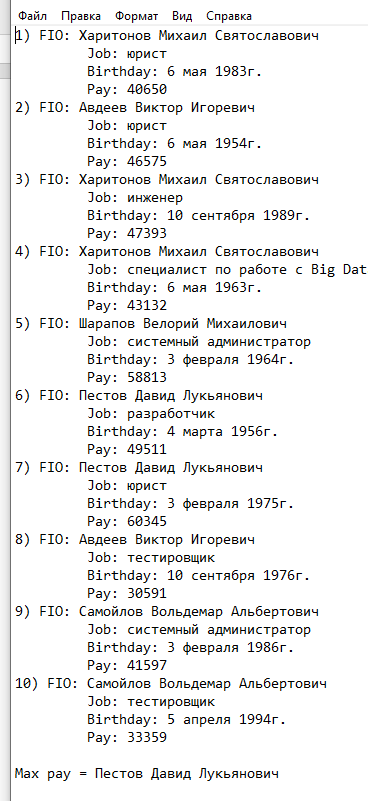
}

return maxPay;

}

**Решение**

****

****

**Задача №2**

Создать структурный тип, описывающий гостиничный номер (название гостиницы, номер, комфортность (люкс, полулюкс, стандарт, эконом), количество человек, стоимость). Заполнить структурный массив N>2 записями. Переписать из исходного массива в другой массив, информацию только о тех гостиничных номерах, название гостиницы которых начинается на букву . Затем новый массив отсортировать по возрастанию стоимости.

**Код**

#include <iostream>

#include <time.h>

#include <list>

#include <fstream>

using namespace std;

ofstream fout;

struct hostel {

string Name;

string comfort;

int number;

int capacity;

int price;

};

string getRandName();

string getRandComfort();

void GetHostels(hostel\* hostels, int count);

list<hostel> GetHostelsP(hostel\* hostels);

void Sort(hostel\* hostels, int size);

void printHostels(hostel\* hostels, int size);

int main()

{

fout.open("result.txt");

setlocale(LC\_ALL, "Russian");

srand(time(0));

hostel\* hostels = new hostel[10];

GetHostels(hostels, 10);

cout << "---------------------------------------\n";

cout << "Hotels:\n";

cout << "---------------------------------------\n";

fout << "---------------------------------------\n";

fout << "Hotels:\n";

fout << "---------------------------------------\n";

printHostels(hostels, 10);

list<hostel> hostelsList = GetHostelsP(hostels);

hostel\* sortedHostels = new hostel[hostelsList.size()];

int j = 0;

for ( auto i = hostelsList.begin(); i != hostelsList.end(); i++)

{

sortedHostels[j++] = \*i;

}

Sort(sortedHostels, hostelsList.size());

cout << "---------------------------------------\n";

cout << "Sorted hotels:\n";

cout << "---------------------------------------\n";

fout << "---------------------------------------\n";

fout << "Sorted hotels:\n";

fout << "---------------------------------------\n";

printHostels(sortedHostels, hostelsList.size());

}

string getRandName() {

string fios[10] = { "Джузеппе",

"Ибис",

"Парк Инн Рэдиссон",

"Максим Горький",

"Хаял",

"Раймонд Отель",

"Шаляпин Палас Отель",

"Берисон Худякова",

"Давыдов",

"Рубин Апарт"

};

return fios[rand() % 10];

}

string getRandComfort() {

string jobs[10] = { "люкс",

"полулюкс",

"стандарт",

"эконом"

};

return jobs[rand() % 4];

}

void GetHostels(hostel\* hostels, int count) {

for (size\_t i = 0; i < 10; i++)

{

hostels[i].Name = getRandName();

hostels[i].comfort = getRandComfort();

hostels[i].number = rand() % 100 + 1;

hostels[i].capacity = rand() % 5 + 1;

hostels[i].price = rand() % 2500 + 500;

}

}

list<hostel> GetHostelsP(hostel\* hostels) {

list<hostel> hostelsList;

for (size\_t i = 0; i < 10; i++)

{

int temp = (int)hostels[i].Name[0];

int answer = -48;

if (temp == answer) {

hostelsList.push\_front(hostel{ hostels[i].Name,

hostels[i].comfort,

hostels[i].number,

hostels[i].capacity,

hostels[i].price

});

}

}

return hostelsList;

}

void Sort(hostel\* hostels, int size) {

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

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

if (hostels[j].price > hostels[j + 1].price) {

swap(hostels[j], hostels[j + 1]);

}

}

}

}

void printHostels(hostel\* hostels, int size) {

for (size\_t i = 0; i < size; i++)

{

cout << i + 1 << ") Name: " << hostels[i].Name << endl;

cout << "\t Number: " << hostels[i].number << endl;

cout << "\t Comfort: " << hostels[i].comfort << endl;

cout << "\t Capacity: " << hostels[i].capacity << " чел." << endl;

cout << "\t Price: " << hostels[i].price << " в день" << endl;

fout << i + 1 << ") Name: " << hostels[i].Name << endl;

fout << "\t Number: " << hostels[i].number << endl;

fout << "\t Comfort: " << hostels[i].comfort << endl;

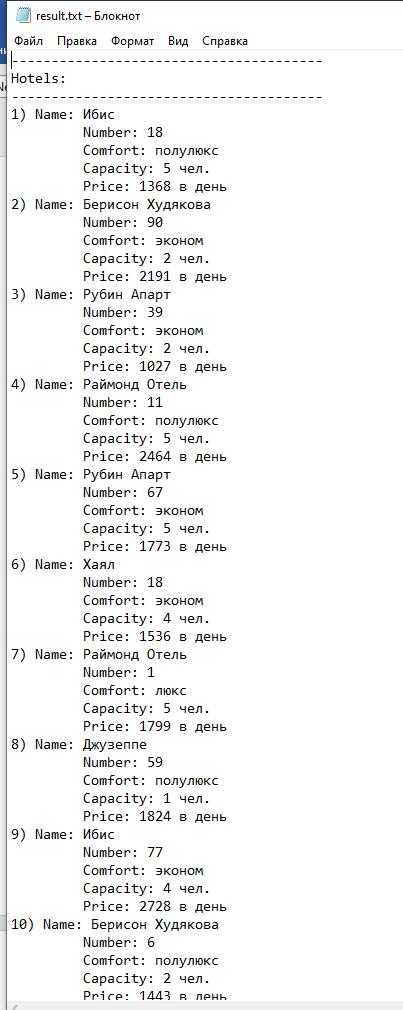
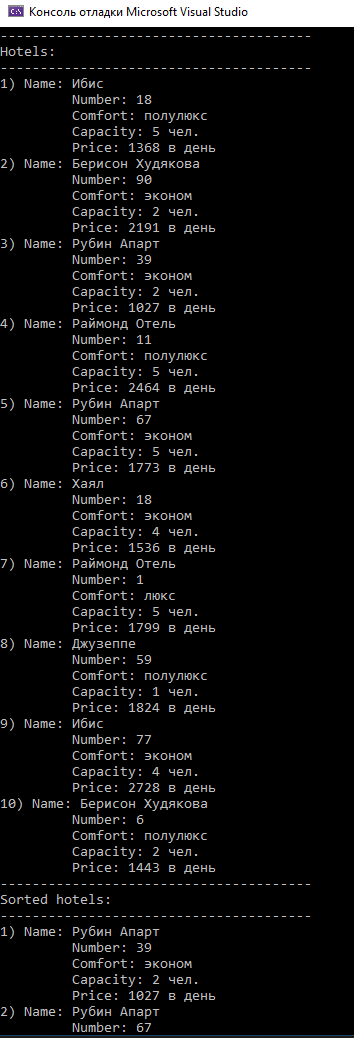
fout << "\t Capacity: " << hostels[i].capacity << " чел." << endl;

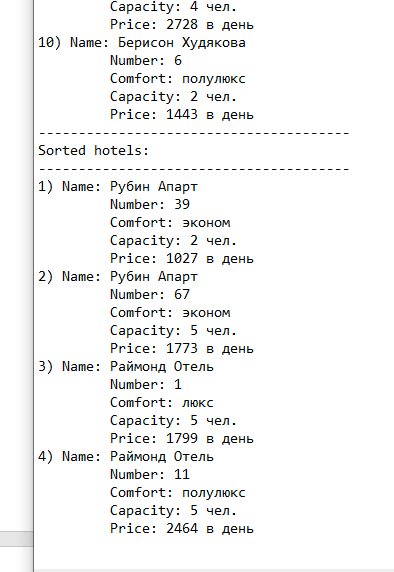
fout << "\t Price: " << hostels[i].price << " в день" << endl;

}

}

**Решение**

****

****

**Задача №3**

Дана информация о комнатах в общежитии, записанная в файле. Запись имеет вид: фамилии, номер комнаты, факультет, площадь. Составить список студентов с заданного факультета и найти среди них тех, кто живет в комнате с наименьшей площадью. Новая структура: фамилия, площадь комнаты.

**Код**

#include <iostream>

#include <time.h>

#include <list>

#include <fstream>

using namespace std;

ofstream fout;

struct room {

string surname;

string faculty;

int number;

int square;

};

struct smallestRoom {

string surname;

int square;

};

string getRandSurname();

string getRandFaculty();

void getRandRooms(room\* rooms, int count);

void printRooms(room\* rooms, int size);

void getSmallestRoom(room\* rooms, smallestRoom\* room);

int main()

{

fout.open("result.txt");

setlocale(LC\_ALL, "Russian");

srand(time(0));

room\* rooms = new room[10];

getRandRooms(rooms, 10);

cout << "---------------------------------------\n";

cout << "Rooms:\n";

cout << "---------------------------------------\n";

fout << "---------------------------------------\n";

fout << "Rooms:\n";

fout << "---------------------------------------\n";

printRooms(rooms, 10);

smallestRoom smallRoom;

getSmallestRoom(rooms, &smallRoom);

cout << "---------------------------------------\n";

cout << "Smallest room:\n";

cout << "---------------------------------------\n";

cout << "Surname: " << smallRoom.surname << endl;

cout << "Square: " << smallRoom.square << " кв.м." << endl;

fout << "---------------------------------------\n";

fout << "Smallest room:\n";

fout << "---------------------------------------\n";

fout << "Surname: " << smallRoom.surname << endl;

fout << "Square: " << smallRoom.square << " кв.м." << endl;

}

string getRandSurname() {

string surnames[10] = { "Пестов",

"Авдеев",

"Филатов",

"Самойлов",

"Русаков",

"Харитонов",

"Галкин",

"Лобанов",

"Шарапов",

"Корнилов"

};

return surnames[rand() % 10];

}

string getRandFaculty() {

string faculties[10] = { "Педагогический",

"Медицинский",

"Политехнический",

"Экономический",

"Исторический"

};

return faculties[rand() % 5];

}

void getRandRooms(room\* rooms, int count) {

for (size\_t i = 0; i < 10; i++)

{

rooms[i].surname = getRandSurname();

rooms[i].faculty = getRandFaculty();

rooms[i].number = rand() % 100 + 1;

rooms[i].square = rand() % 20 + 20;

}

}

void printRooms(room\* rooms, int size) {

for (size\_t i = 0; i < size; i++)

{

cout << i + 1 << ") Surname: " << rooms[i].surname << endl;

cout << "\t Number: " << rooms[i].number << endl;

cout << "\t Faculty: " << rooms[i].faculty << endl;

cout << "\t Square: " << rooms[i].square << " кв.м." << endl;

fout << i + 1 << ") Surname: " << rooms[i].surname << endl;

fout << "\t Number: " << rooms[i].number << endl;

fout << "\t Faculty: " << rooms[i].faculty << endl;

fout << "\t Square: " << rooms[i].square << " кв.м." << endl;

}

}

void getSmallestRoom(room\* rooms, smallestRoom\* room) {

room->square = 99;

for (size\_t i = 0; i < 10; i++)

{

if (rooms[i].square < room->square)

{

room->square = rooms[i].square;

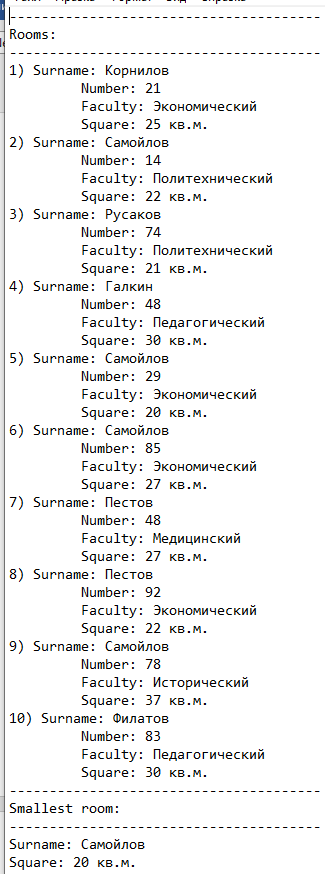
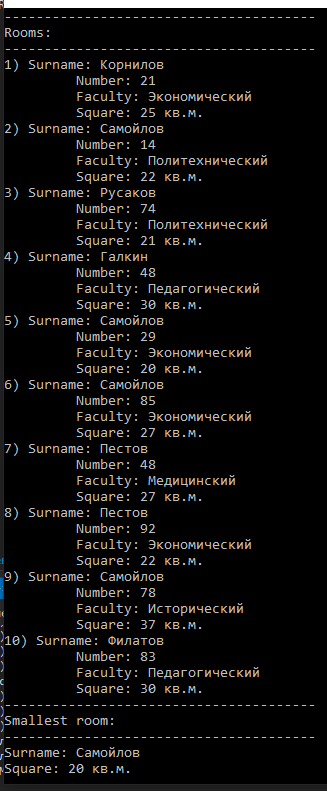
room->surname = rooms[i].surname;

}

}

}

**Решение**

****

**Домашнее задание.**

**Задание №1.**

В рамках структурного похода создать программу по обстреливанию мишени-шара, заданного своими координатами центра и радиусом, расположенного внутри куба, стороны которого параллельны осям координат. Вывести количество сделанных выстрелов и количество промахов.

**Код**

#include <iostream>

#include <time.h>

using namespace std;

struct point {

int x;

int y;

int z;

};

struct sphere {

int a;

int b;

int c;

int R;

};

void getSphere(sphere\* sphere);

void getPoint(point\* point);

void Shots(sphere sphere);

void main() {

srand(time(0));

sphere sphere;

int maxX = 50;

int maxY = 50;

int maxZ = 50;

getSphere(&sphere);

cout << "Sphere:\n";

cout << "a = " << sphere.a << endl;

cout << "b = " << sphere.b << endl;

cout << "c = " << sphere.c << endl;

cout << "R = " <<sphere.R << endl << endl;

Shots(sphere);

}

void getSphere(sphere\* sphere) {

sphere->R = rand() % 20 + 5;

sphere->a = rand() % (50 - 2 \* sphere->R) + sphere->R;

sphere->b = rand() % (50 - 2 \* sphere->R) + sphere->R;

sphere->c = rand() % (50 - 2 \* sphere->R) + sphere->R;

}

void getPoint(point\* point) {

point->x = rand() % 50;

point->y = rand() % 50;

point->z = rand() % 50;

}

void Shots(sphere sphere) {

point point;

int hit = 0;

int miss = 0;

for (int i = 0; i < 100; i++)

{

getPoint(&point);

if (pow(point.x - sphere.a, 2) + pow(point.y - sphere.b, 2) + pow(point.z - sphere.c, 2) <= pow(sphere.R, 2))

hit++;

else

miss++;

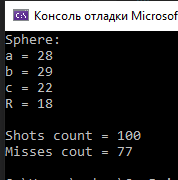
}

cout << "Shots count = " << hit + miss << endl;

cout << "Misses cout = " << miss << endl;

}

**Решение**

****

**Задание №2.**

Создать структуру для представления информации о кости домино, состоящей из левой половинки и правой половинки. Поля «левая» и «правая» половинки хранят информацию о количестве точек на половинках. Описать массив из 28 элементов (кости домино). Заполнить массив случайными числами или ввести его значения из файла. Исключить из массива одинаковые кости, если они в нем появились в ходе инициализации. Отсортировать оставшиеся элементы массива так, чтобы правая цифра очередной кости равнялась левой цифре следующей кости.

**Код**

#include <iostream>

#include <fstream>

using namespace std;

ofstream fout;

struct domino {

int left;

int right;

};

void getRandDominoes(domino\* dominos, int size);

void printDominoes(domino\* dominos, int size);

void removeDomino(domino\* dominos, int index, int& size);

void removeCollision(domino\* dominos, int& size);

void addRandDominoes(domino\* dominos, int& size);

void sortDominoes(domino\* dominos, int size);

int main()

{

fout.open("result.txt");

int size = 28;

domino\* dominos = new domino[size];

getRandDominoes(dominos, size);

cout << "---------------------------------------\n";

cout << "Dominos:\n";

cout << "---------------------------------------\n";

fout << "---------------------------------------\n";

fout << "Dominos:\n";

fout << "---------------------------------------\n";

printDominoes(dominos, size);

removeCollision(dominos, size);

while (size != 28)

{

addRandDominoes(dominos, size);

removeCollision(dominos, size);

}

cout << "---------------------------------------\n";

cout << "Unique dominos:\n";

cout << "---------------------------------------\n";

fout << "---------------------------------------\n";

fout << "Unique dominos:\n";

fout << "---------------------------------------\n";

printDominoes(dominos, size);

sortDominoes(dominos, size);

cout << "---------------------------------------\n";

cout << "Sorted dominos:\n";

cout << "---------------------------------------\n";

fout << "---------------------------------------\n";

fout << "Sorted dominos:\n";

fout << "---------------------------------------\n";

printDominoes(dominos, size);

}

void getRandDominoes(domino\* dominos, int size) {

for (size\_t i = 0; i < size; i++)

{

dominos[i].left = rand() % 7;

dominos[i].right = rand() % 7;

}

}

void printDominoes(domino\* dominos, int size) {

for (size\_t i = 0; i < size; i++)

{

cout << "#" << i + 1 << endl;

cout << "-----\n";

cout << "|" << dominos[i].left << "|" << dominos[i].right << "|\n";

cout << "-----\n";

fout << "#" << i + 1 << endl;

fout << "-----\n";

fout << "|" << dominos[i].left << "|" << dominos[i].right << "|\n";

fout << "-----\n";

}

}

void removeDomino(domino\* dominos, int index, int& size) {

for (size\_t i = index; i < size; i++)

{

dominos[i] = dominos[i + 1];

}

size--;

}

void removeCollision(domino\* dominos, int& size) {

for (size\_t i = 0; i < size; i++)

{

for (size\_t j = i + 1; j < size; j++)

{

if (dominos[i].left == dominos[j].left && dominos[i].right == dominos[j].right ||

dominos[i].left == dominos[j].right && dominos[i].right == dominos[j].left) {

removeDomino(dominos, i, size);

i--;

}

}

}

}

void addRandDominoes(domino\* dominos, int& size) {

for (size\_t i = size; i < 28; i++)

{

dominos[i].left = rand() % 7;

dominos[i].right = rand() % 7;

}

size = 28;

}

void sortDominoes(domino\* dominos, int size) {

for (size\_t i = 0; i < size; i++)

{

for (size\_t j = i + 1; j < size; j++)

{

if (dominos[i].right == dominos[j].left)

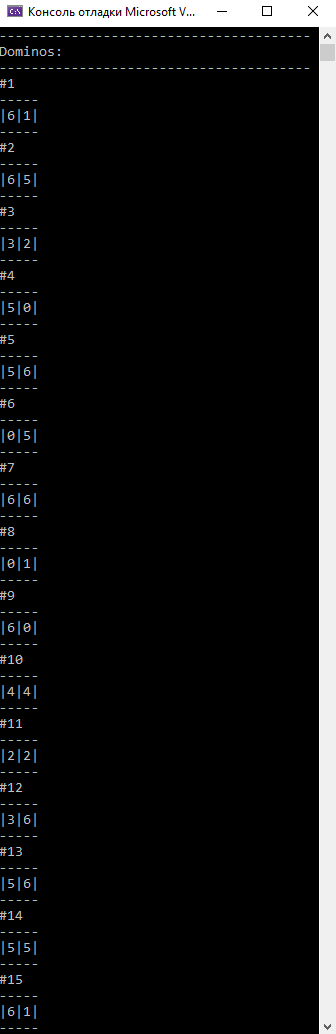
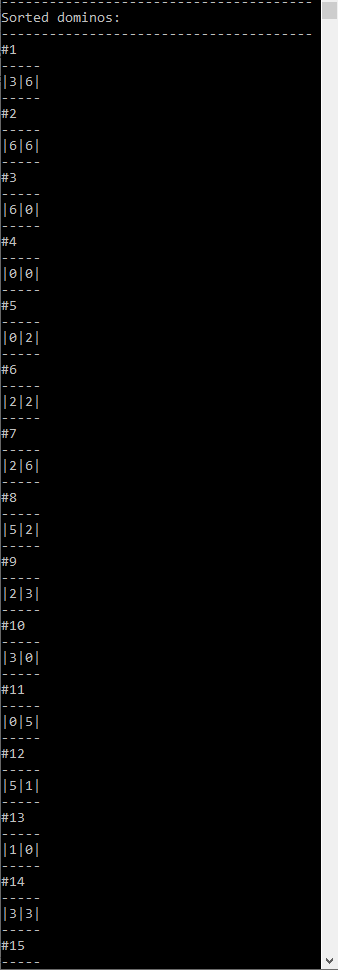
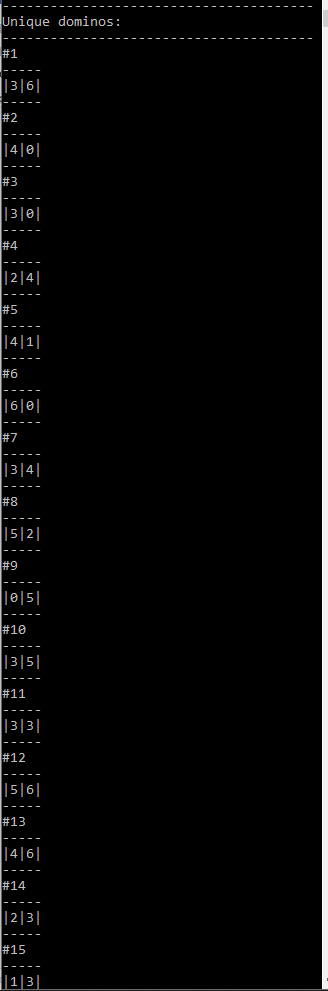
swap(dominos[i + 1], dominos[j]);

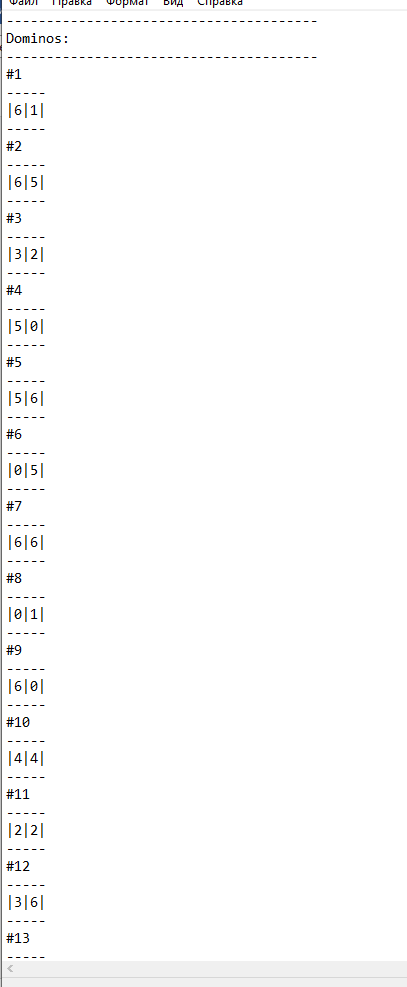
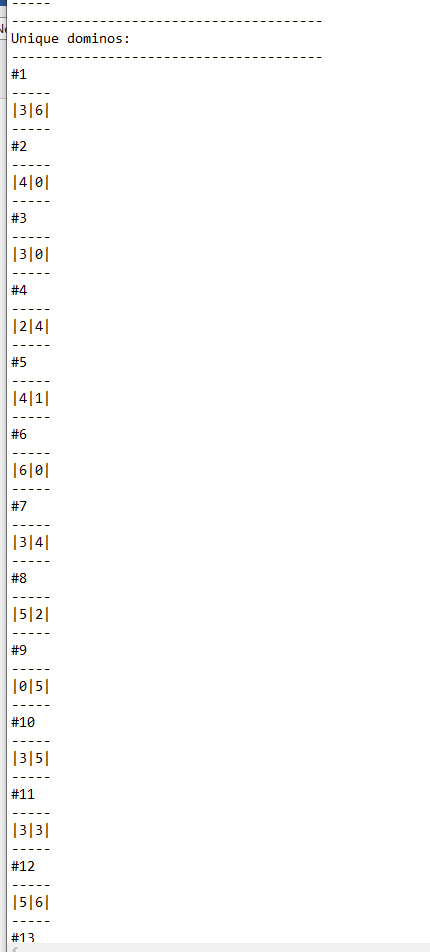
}

}

}

**Решение**

**** ****

**** **** 