Лабораторная работа №2

**Графы**

7 вариант

***Составить программы, решающие следующие задачи. Входные данные находятся в файле INPUT.TXT. Для каждой задачи изобразить на экране соответствующий граф.***

В компьютерной игре герою требуется пройти через все залы, уничтожив в каждом по одному монстру. Монстры имеют различную силу, и убить каждого из них можно, если сила монстра не превосходит силы героя. Сила героя каждый раз возрастает на величину силы убитого монстра. Известны, сила каждого монстра, начальная сила героя, а также между какими залами имеются двери (через дверь можно пройти только в одном направлении). Определить все пути, двигаясь по которым, герой сможет убить всех монстров, побывав в каждом зале только один раз.

Технические требования:

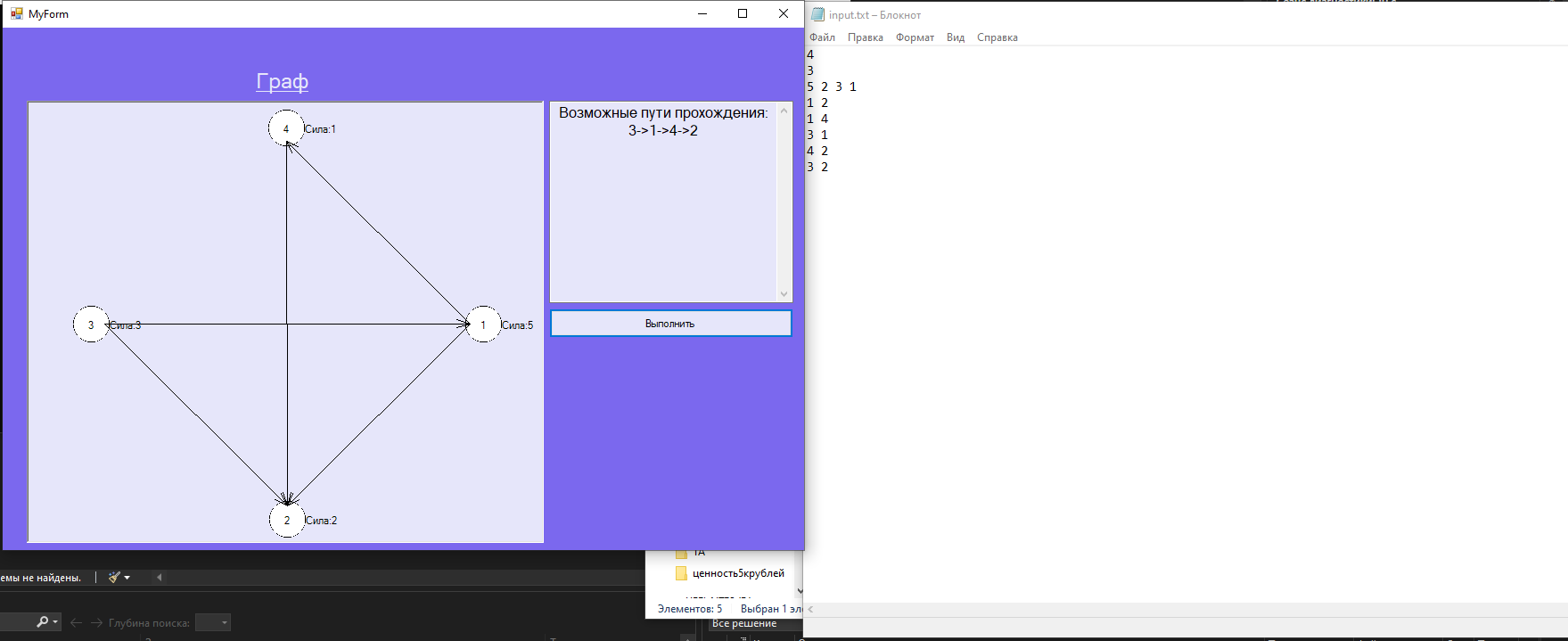
Количество залов N≤10. Залы пронумерованы числами от 1 до N.

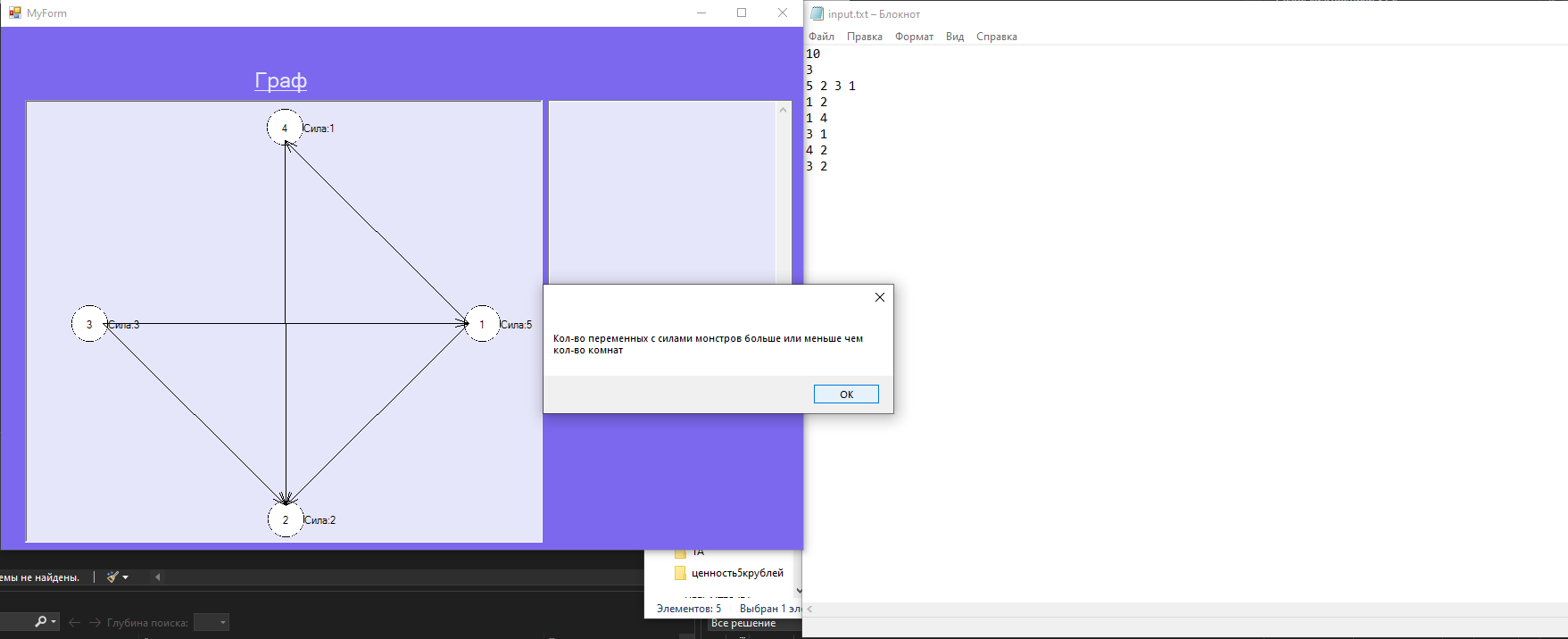
Формат файла INPUT.TXT:

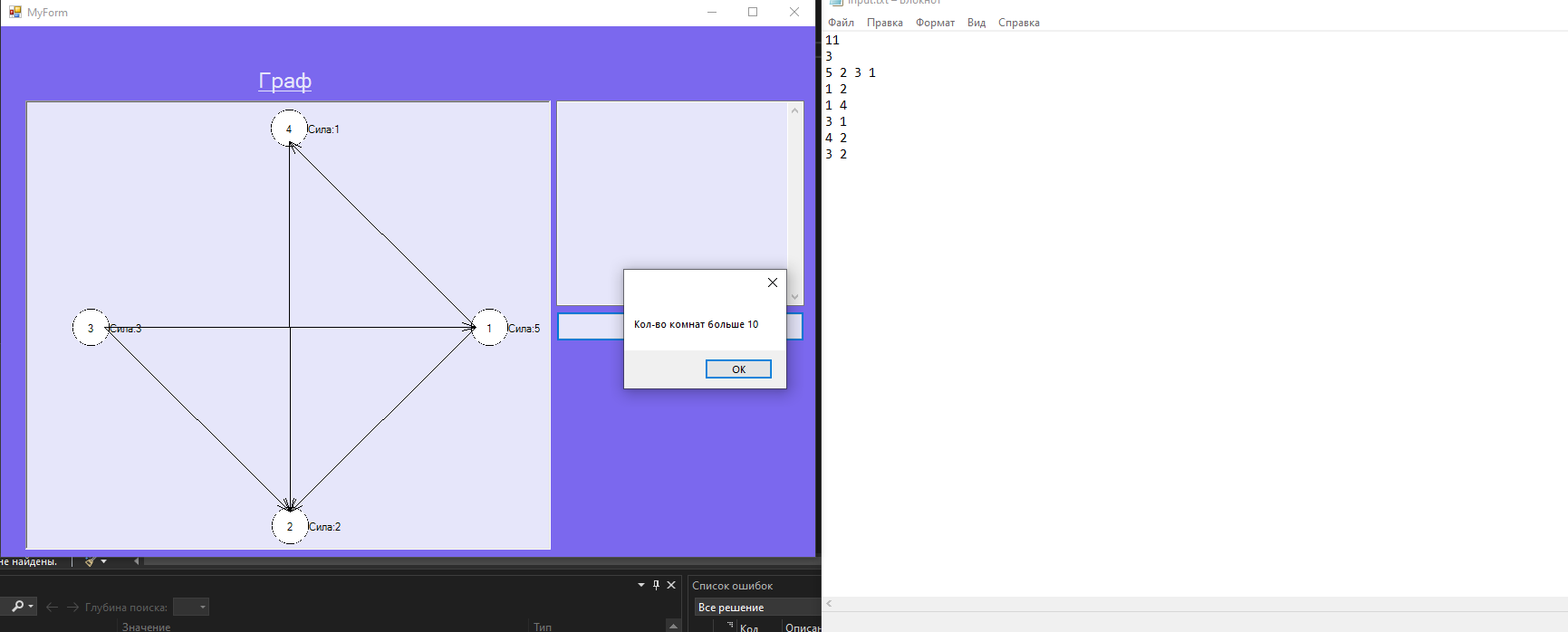
1-я строка – количество залов; 2-я - начальная сила героя; 3-я – N натуральных чисел через пробел – силы монстров в каждом зале; последующие строки содержат по два натуральных числа – номера залов X и Y, между которыми существует дверь из X в Y.

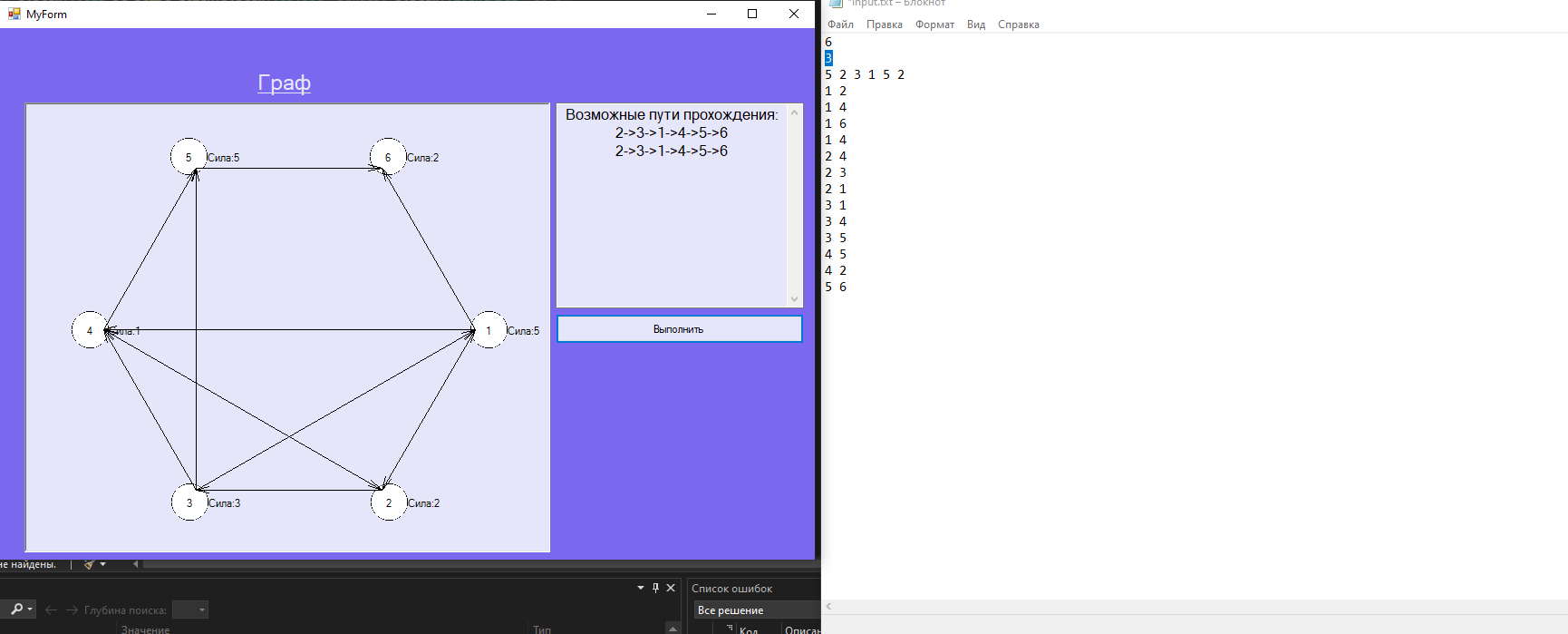
**Описание алгоритма**

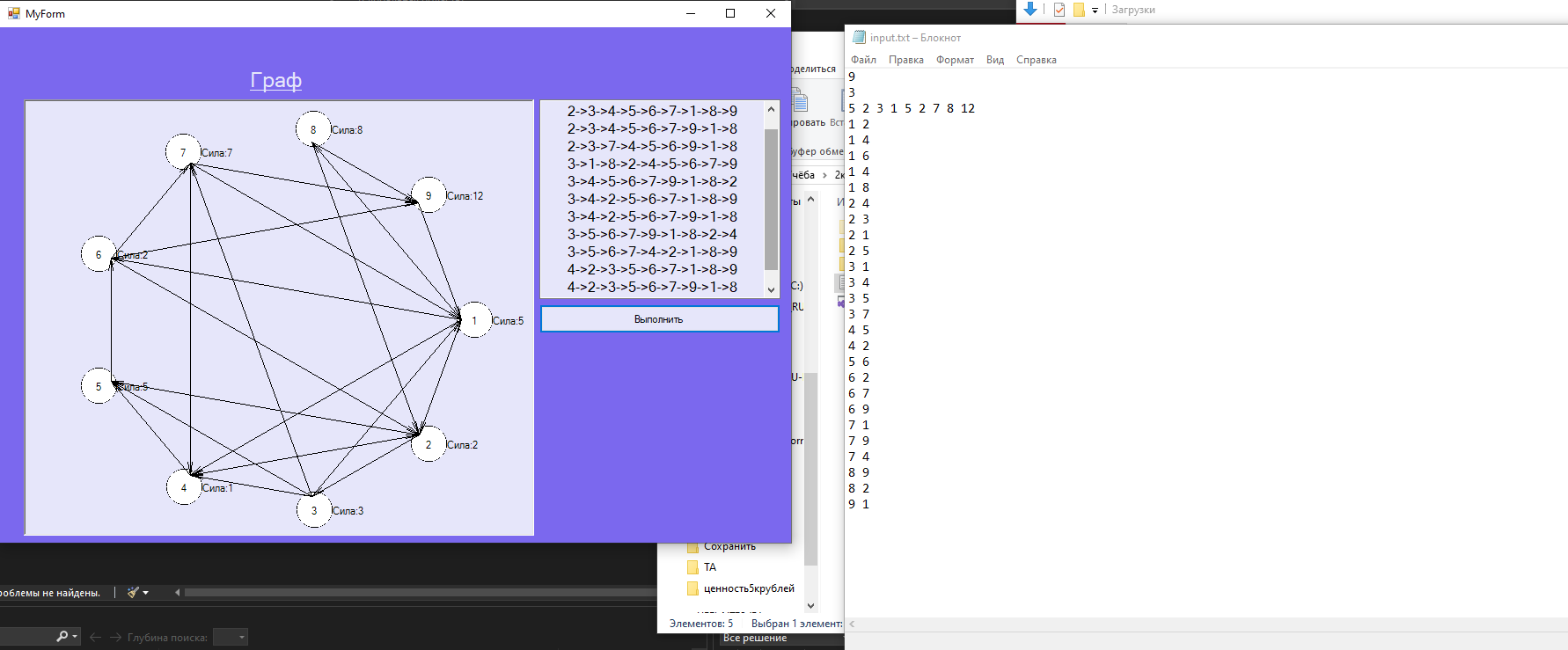
Тестирование:











Код программы:

#pragma once

#include <iostream>

#include <string>

#include <vector>

#include <string>

#include <math.h>

namespace Project1 {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

using namespace System::IO;

struct Room {

int power; // Сила монстра в зале

std::vector<int> doors; // Номера залов, с которыми есть двери

};

public ref class MyForm : public System::Windows::Forms::Form

{

public:

MyForm(void)

{

InitializeComponent();

}

protected:

~MyForm()

{

if (components)

{

delete components;

}

}

private: System::Windows::Forms::OpenFileDialog^ openFileDialog1;

private: System::Windows::Forms::Button^ bt\_start;

private: System::Windows::Forms::TextBox^ textBox1;

private: System::Windows::Forms::Panel^ ptr;

private: System::Windows::Forms::Label^ label1;

private:

System::ComponentModel::Container^ components;

#pragma region Windows Form Designer generated code

void InitializeComponent(void)

{

this->openFileDialog1 = (gcnew System::Windows::Forms::OpenFileDialog());

this->bt\_start = (gcnew System::Windows::Forms::Button());

this->textBox1 = (gcnew System::Windows::Forms::TextBox());

this->ptr = (gcnew System::Windows::Forms::Panel());

this->label1 = (gcnew System::Windows::Forms::Label());

this->SuspendLayout();

//

// openFileDialog1

//

this->openFileDialog1->FileName = L"openFileDialog1";

//

// bt\_start

//

this->bt\_start->BackColor = System::Drawing::Color::Lavender;

this->bt\_start->Location = System::Drawing::Point(613, 315);

this->bt\_start->Name = L"bt\_start";

this->bt\_start->Size = System::Drawing::Size(274, 33);

this->bt\_start->TabIndex = 3;

this->bt\_start->Text = L"Выполнить";

this->bt\_start->UseMnemonic = false;

this->bt\_start->UseVisualStyleBackColor = false;

this->bt\_start->Click += gcnew System::EventHandler(this, &MyForm::bt\_start\_Click);

//

// textBox1

//

this->textBox1->BackColor = System::Drawing::Color::Lavender;

this->textBox1->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->textBox1->Location = System::Drawing::Point(613, 82);

this->textBox1->Multiline = true;

this->textBox1->Name = L"textBox1";

this->textBox1->ReadOnly = true;

this->textBox1->ScrollBars = System::Windows::Forms::ScrollBars::Vertical;

this->textBox1->Size = System::Drawing::Size(274, 227);

this->textBox1->TabIndex = 6;

this->textBox1->TextAlign = System::Windows::Forms::HorizontalAlignment::Center;

//

// ptr

//

this->ptr->BackColor = System::Drawing::Color::Lavender;

this->ptr->BorderStyle = System::Windows::Forms::BorderStyle::Fixed3D;

this->ptr->Location = System::Drawing::Point(27, 82);

this->ptr->Name = L"ptr";

this->ptr->Size = System::Drawing::Size(580, 496);

this->ptr->TabIndex = 10;

//

// label1

//

this->label1->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 18, System::Drawing::FontStyle::Underline, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label1->ForeColor = System::Drawing::Color::Lavender;

this->label1->Location = System::Drawing::Point(245, 39);

this->label1->Name = L"label1";

this->label1->Size = System::Drawing::Size(141, 40);

this->label1->TabIndex = 12;

this->label1->Text = L"Граф";

this->label1->TextAlign = System::Drawing::ContentAlignment::MiddleCenter;

//

// MyForm

//

this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);

this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;

this->BackColor = System::Drawing::Color::MediumSlateBlue;

this->ClientSize = System::Drawing::Size(899, 586);

this->Controls->Add(this->label1);

this->Controls->Add(this->ptr);

this->Controls->Add(this->textBox1);

this->Controls->Add(this->bt\_start);

this->Name = L"MyForm";

this->Text = L"MyForm";

this->ResumeLayout(false);

this->PerformLayout();

}

#pragma endregion

bool readData(int& heroPower, int& nRooms, std::vector<Room>& rooms) {

String^ FileName = "";

if (openFileDialog1->ShowDialog() == Windows::Forms::DialogResult::OK) {

FileName = openFileDialog1->FileName;

}

try

{

StreamReader^ f = gcnew System::IO::StreamReader(FileName);

String^ line = f->ReadLine();

nRooms = Convert::ToInt32(line);

if (nRooms > 10) {

MessageBox::Show("Кол-во комнат больше 10");

f->Close();

return 0;

}

line = f->ReadLine();

heroPower = Convert::ToInt32(line);

line = f->ReadLine();

array<String^>^ tmp = line->Split(' ');

if (tmp->Length != nRooms) {

MessageBox::Show("Кол-во переменных с силами монстров больше или меньше чем кол-во комнат");

f->Close();

return 0;

}

rooms.resize(nRooms);

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

rooms[i].power = Convert::ToInt32(tmp[i]);

}

int x, y;

while ((line = f->ReadLine()) != nullptr) {

array<String^>^ tmp = line->Split(' ');

x = Convert::ToInt32(tmp[0]);

y = Convert::ToInt32(tmp[1]);

rooms[x - 1].doors.push\_back(y - 1);

}

f->Close();

}

catch (Exception^ ex) {

MessageBox::Show("Ошибка чтения файла");

return 0;

}

}

void findPaths(int currentRoom, int heroPower, const std::vector<Room>& rooms, std::vector<int>& path, std::vector<std::vector<int>>& allPaths, std::vector<bool>& visited) {

if (visited[currentRoom]) {

return;

}

visited[currentRoom] = true;

path.push\_back(currentRoom);

if (path.size() == rooms.size()) {

allPaths.push\_back(path);

visited[currentRoom] = false;

path.pop\_back();

return;

}

if (heroPower >= rooms[currentRoom].power) {

heroPower += rooms[currentRoom].power;

for (int nextRoom : rooms[currentRoom].doors) {

findPaths(nextRoom, heroPower, rooms, path, allPaths, visited);

}

heroPower -= rooms[currentRoom].power;

}

visited[currentRoom] = false;

path.pop\_back();

}

void DrawGraph(int n, std::vector<Room>& rooms) {

Graphics^ g = ptr->CreateGraphics();

g->Clear(Color::Lavender);

int radius = 20;

int centerX = ptr->Width / 2;

int centerY = ptr->Height / 2;

double angle = 2 \* 3.14 / n;

Pen^ pen = gcnew Pen(Color::Black, 1);

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

int x = (int)(centerX + 220 \* cos(i \* angle));

int y = (int)(centerY + 220 \* sin(i \* angle));

g->DrawEllipse(pen, x - radius, y - radius, 2 \* radius, 2 \* radius);

g->FillEllipse(Brushes::White, x - radius, y - radius, 2 \* radius, 2 \* radius);

g->DrawString(Convert::ToString(i + 1) + " Сила:" + rooms[i].power, DefaultFont, Brushes::Black, x - 5, y - 5);

}

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

for (int j : rooms[i].doors) {

int x1 = (centerX + 205 \* cos(i \* angle));

int y1 = (centerY + 205 \* sin(i \* angle));

int x2 = (centerX + 205 \* cos(j \* angle));

int y2 = (centerY + 205 \* sin(j \* angle));

double lineAngle = Math::Atan2(y2 - y1, x2 - x1);

int ax = x2 - 15 \* cos(lineAngle + angle / 5);

int ay = y2 - 15 \* sin(lineAngle + angle / 5);

int ax1 = x2 - 15 \* cos(lineAngle - angle / 5);

int ay1 = y2 - 15 \* sin(lineAngle - angle / 5);

g->DrawLine(pen, x1, y1, x2, y2);

g->DrawLine(pen, ax, ay, x2, y2);

g->DrawLine(pen, ax1, ay1, x2, y2);

}

}

}

private: System::Void bt\_start\_Click(System::Object^ sender, System::EventArgs^ e) {

textBox1->Clear();

int Hpw, nR;

std::vector<Room> rooms;

if (readData(Hpw, nR, rooms))

{

DrawGraph(nR, rooms);

std::vector<int> path;

std::vector<std::vector<int>> allPaths;

std::vector<bool> visited(rooms.size() + 1, false);

for (int t = 0; t < nR; t++) {

findPaths(t, Hpw, rooms, path, allPaths, visited);

}

if (allPaths.empty()) {

textBox1->Text += "Путей прохождения нет";

return;

}

textBox1->Text += "Возможные пути прохождения:\r\n";

for (const auto& path : allPaths) {

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

textBox1->Text += (path[i] + 1);

if (i != path.size() - 1) {

textBox1->Text += "->";

}

}

textBox1->Text += "\r\n";

}

}

}

};

}