ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ФАКУЛЬТЕТ АВТОМАТИКИ И ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

КАФЕДРА РАДИОЭЛЕКТРОННЫХ СРЕДСТВ

**ВИЗУАЛИЗАЦИЯ АЛГОРИТМА ПОИCКА В ШИРИНУ**

Пояснительная записка

Курсовая работа по дисциплине

"Языки программирования"

ТПЖА.18207-01 81 01 ПЗ

Разработал студент гр. ИКТб-2302-02-24 \_\_\_\_\_\_\_\_\_\_\_ /Глушков И.С./

Руководитель ст. преподаватель

Кафедры РЭС \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Т. В. Наумович/

Курсовая работа защищена с

оценкой «\_\_\_\_\_» «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_2019 г.

Киров 2019

ФГБОУ ВО «Вятский государственный университет»

Факультет автоматики и вычислительной техники

Кафедра радиоэлектронных средств

УТВЕРЖДАЮ

Зав. кафедрой РЭС

\_\_\_\_\_\_\_\_\_А. П. Метелев

**ЗАДАНИЕ**

**на курсовую работу**

**№ PL.2019. ИКТб-2302-02-24; v.07**

*Дисциплина*

**Языки программирования**

*Студент*

***Глушков Иван Сергеевич***

*Тема работы*

**Визуализация алгоритма поиска в ширину**

*Срок сдачи законченной работы* **02.12.2019 г.**

*Исходные данные к работе*

**1. Среда программирования Visual C++ 2017. 2. Дан связный неориентированный граф G =<V, E> (V – конечное множество вершин, E – набор неупорядоченных пар вершин). Граф описывается целочисленной матрицей смежности A [N][N]. 3. Описать различные алгоритмы поиска в ширину. 4. Ввод с помощью матрицы и графа. 5. Предусмотреть визуализацию алгоритма поиска в ширину.**

*Содержание пояснительной записки* **Программные документы "Описание программы", "Текст программы", "Руководство оператора" по ГОСТ 19.402-78, 19.401-78, 19.505-70.**

**Графический материал по согласованию с руководителем**

*Дата выдачи задания* **19.09.2019 г.**

*Руководитель работы* **Т. В. Наумович**

*Задание принял к исполнению* **19.09.2019г.**

*Подпись исполнителя*

**РЕФЕРАТ**

Глушков И.С. Визуализация алгоритма поиска в ширину: ТПЖА.18207-01 81 01ПЗ: Курс.работа/ ВятГУ, каф. РЭС; рук. Т.В. Наумович. - Киров, 2019.

ВИЗУАЛИЗАЦИЯ АЛГОРИТМА ПОИСКА В ШИРИНУ, ОСТОВНОЙ ГРАФ. ПОИСК В ШИРИНУ. МАТРИЦА СМЕЖНОСТИ. ВЕРШИНА ГРАФА. РЕБРО ГРАФА.

Объектом исследования является алгоритм поиска в ширину для нахождения остовного дерева графа.

Цель работы – изучить прнцип решения задачи нахождения остовного дерева при помощи алгоритма обхода графа в ширину.

Для достежения поставленной цели были определены следующие задачи:

1. Изучить суть алгоритма.
2. Разработать программу, реализующую алгоритм поиска в ширину.
3. Провести тестирование разработанной программы на различных примерах.

В качестве среды разработки в задании на курсовую работу указана Microsoft Visual Studio 2017 года, с framework Windows Forms, язык программирования C++.

В ходе выполнения работы была реализована программа, которая находит остовное дерево заданного графа при помощи алгоритма обхода графа в ширину.

В разработанной программе присутствуют возможность ввода начальных данных: количества вершин графа, матрицы смежности, с указанием веса ребер, начальной вершины для поиска.

Результаты работы могут быть использованы для демонстарции работы алгоритма обхода в ширину, решения прикладных зачач методом нахождения остовного дерева графа.

Содержание

[Введение 2](#_Toc24179611)

[1 Анализ задания и выбор метода решения 3](#_Toc24179612)

[1.1 Анализ задания 3](#_Toc24179613)

[1.2 Выбор метода решения 3](#_Toc24179614)

[2 Алгоритм поиска в ширину 4](#_Toc24179615)

[2.1 Описание алгоритма 4](#_Toc24179616)

[2.2 Проектирование программы 5](#_Toc24179617)

[2.3 Состав программных модулей 5](#_Toc24179618)

[2.4 Анализ результатов 7](#_Toc24179619)

[Заключение 8](#_Toc24179620)

Приложение А (обязательное) Визуализация алгоритма поиска в ширину. Описание программы. Лист утверждения ТПЖА.18207-01 13-01 9

[Приложение Б (обязательное) Визуализация алгоритма поиска в ширину. Руководство оператора. Лист утверждения ТПЖА.18207-01 34-01 15](#_Toc24179623)

Приложение В (обязательное) Визуализация алгоритма поиска в ширину. Текст программы. Лист утверждения ТПЖА.18207-01 12-01 20

Приложение Г (Справочное) Библиографический список 40

## Введение

В 2019/2020 году технический прогресс развивается настолько быстрыми темпами, что самые развитые экономики мира, да и просто люди не успевают следить за новыми открытиями и внедрить их в жизнь. Обработка огромнных массивов инфомации производится ежесекундно и днем и ночью. В связи с таким быстрым и расширяющимся информационным прогрессом появилась острая нужда в решении сложных вычислительных задач. Для это пишется огромное количество прикладных программ, например в нашей компании мы с коллегой написали программу для отслеживания нагрузки сотрудников, подсчет общего числа непрочитанных приоритетных e-mail, чатов.

Задача о нахождении остовного дерева может встречаться, например в подобной постановке: в ходе командировки по Европе нужно посетить пять различных мероприятий в проходящих в одно время в течение двух недель, требуется построить минимальный по стоимости перелета маршрут.

Эта задача может быть сформулирована в терминах теории графов как задача о нахождении минимального остовного дерева в графе, вершины которого представляют города, рёбра — это пары городов, а вес ребра равен стоимости перелета.

Остовным деревом (ОД) связного графа называется его связный подграф, состоящий из всех вершин исходного дерева и некоторых его ребер.

Цель данной работы состоит в реализации алгоритма поиска стягивающего дерева графа с использованием поиска в ширину в среде разработки Microsoft Visual Studio 2017, язык программирования С++, framework Windows Forms.

# **1 Анализ задания и выбор метода решения**

## 1.1 Анализ задания

В соответствии с заданием на курсовую работу требуется разработать программу, реализующую алгоритм поиска в ширину для построения основного дерева графа. Программа должна быть разработана в среде программирования Microsoft Visual C++ 2017 и должна работать в операционной системе Windows.

Программный продукт находит остовное дерево графа по матрице смежности при помощи алгоритма обхода графа в ширину. Он предоставляет возможность ввода начальных данных: количества вершин графа, матрицы смежности и номер начальной вершины для поиска.

## 1.2 Выбор метода решения

Программа, реализующая алгоритм поиска в ширину будет иметь вид оконного приложения, что облегчает ее разработку и в то же время упрощает ее использование оператором.

# **2 Алгоритм поиска в ширину**

## 2.1 Описание алгоритма

Граф – совокупность точек, соединенных линиями. Точки называются вершинами, или узлами, а линии – ребрами, или дугами.

Поиск в ширину – это алгоритм обхода вершин графа. Он подразумевает по уровневое исследование графа:

• в начале посещается корень – произвольно выбранный узел,

• затем – все потомки данного узла,

• после этого посещаются потомки потомков и т.д.

.

Поиск в ширину может применяться для решения множества задач. В данной работе рассматривается задача нахождения стягивающего дерева графа с использованием алгоритма поиска в ширину.

Стягивающее (остовное) дерево графа состоит из минимального подмножества рёбер графа, таких, что из любой вершины графа можно попасть в любую другую вершину, двигаясь по этим рёбрам.

В данном курсовом проекте будет разработана реализация алгоритма поиска стягивающего дерева с использованием поиска в ширину.

Поиск в ширину подразумевает по уровневое исследование графа:

* в начале посещается корень – произвольно выбранный узел,
* затем – все потомки данного узла,
* после этого посещаются потомки потомков и т.д.

Вершины просматриваются в порядке возрастания их расстояния от корня.

Алгоритм прекращает свою работу после обхода всех вершин графа, либо в случае выполнения требуемого условия (например, найти кратчайший путь из вершины 1 в вершину 6).

Применения алгоритма поиска в ширину

* Поиск кратчайшего пути в невзвешенном графе (ориентированном или неориентированном).
* Поиск компонент связности.
* Нахождения решения какой-либо задачи (игры) с наименьшим числом ходов.
* Найти все рёбра, лежащие на каком-либо кратчайшем пути между заданной парой вершин.
* Найти все вершины, лежащие на каком-либо кратчайшем пути между заданной парой вершин.

Алгоритм поиска в ширину работает как на ориентированных, так и на неориентированных графах.

Для реализации алгоритма удобно использовать очередь.

## 2.2 Проектирование программы

Программа будет иметь вид оконного приложения, все действия пользователя с программой будут выполняться при помощи визуальных полей ввода и элементов управления, размещённых на главном окне программы.

Программа будет состоять из двух модулей. Первый (программный) реализует взаимодействие с пользователем. Во втором модуле описаны дополнительные классы, необходимые для реализации алгоритма, а также реализация функций нахождения основного дерева при помощи алгоритма обхода графа в ширину.

## 2.3 Состав программных модулей

Название функции и ее описание представлены ниже.

1) Graph(intnode\_count) – конструктор класса графа.

Входным параметрами являются:

- node\_count - количество вершин графа;

2) Graph(array<array<int>^> ^matr) - конструктор класса графа.

Входным параметрами являются:

- array<array<int>^> ^matr – матрица смежности графа;

3) void init(array<array<int>^>^matr) – функция инициализации графа

Входным параметрами являются:

- array<array<int>^> ^matr – матрица смежности графа;

4) cliext::vector<par^> ^ run\_width\_next\_step() - функция выполнения одного шага алгоритма нахождения основного дерева при помощи обхода графа в ширину.

Выходным параметром является массив ребер полученного графа.

5) voidInitializeComponent(void). В данной функции происходит инициализация визуальных компонентов окна приложения.

6)VoidnumericUpDown1\_ValueChanged(System::Object^sender,System::EventArgs^e).Функция обрабатывает событие изменения количества вершин графа.

7) button1\_Click(System::Object^sender,System::EventArgs^e) Реализует инициализацию начальными условиями.

8) button2\_Click(System::Object^sender,System::EventArgs^e) Реализует алгоритм поиска остовного дерева минимальной длины.

Для того, чтобы лучше разобраться в структуре программы можно обратится к *Приложению В* (текст программы).

## 2.4 Анализ результатов

Данный пункт является необходимым в процессе выполнения работы, так как отображает результат работы (см. рисунок 2.1 и 2.2).

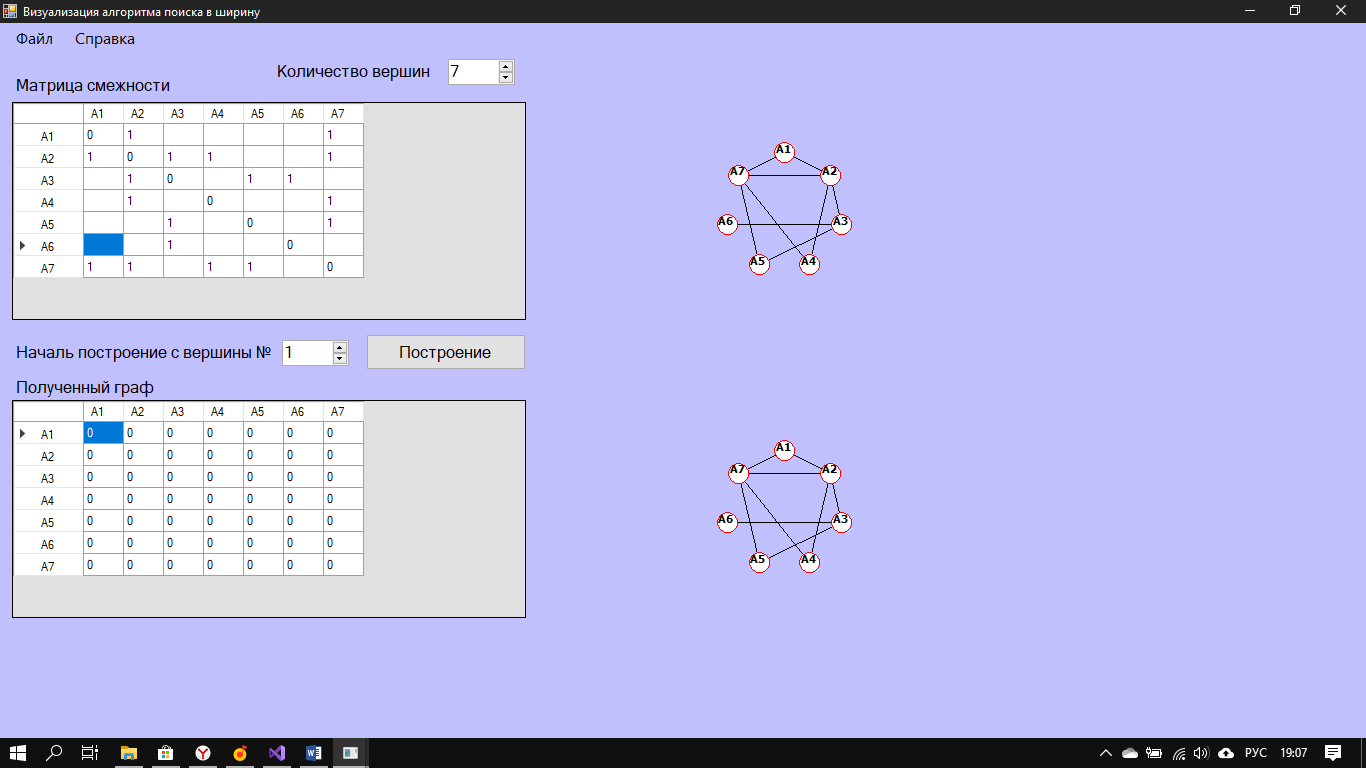


Рисунок 2.1 – Ввод данных

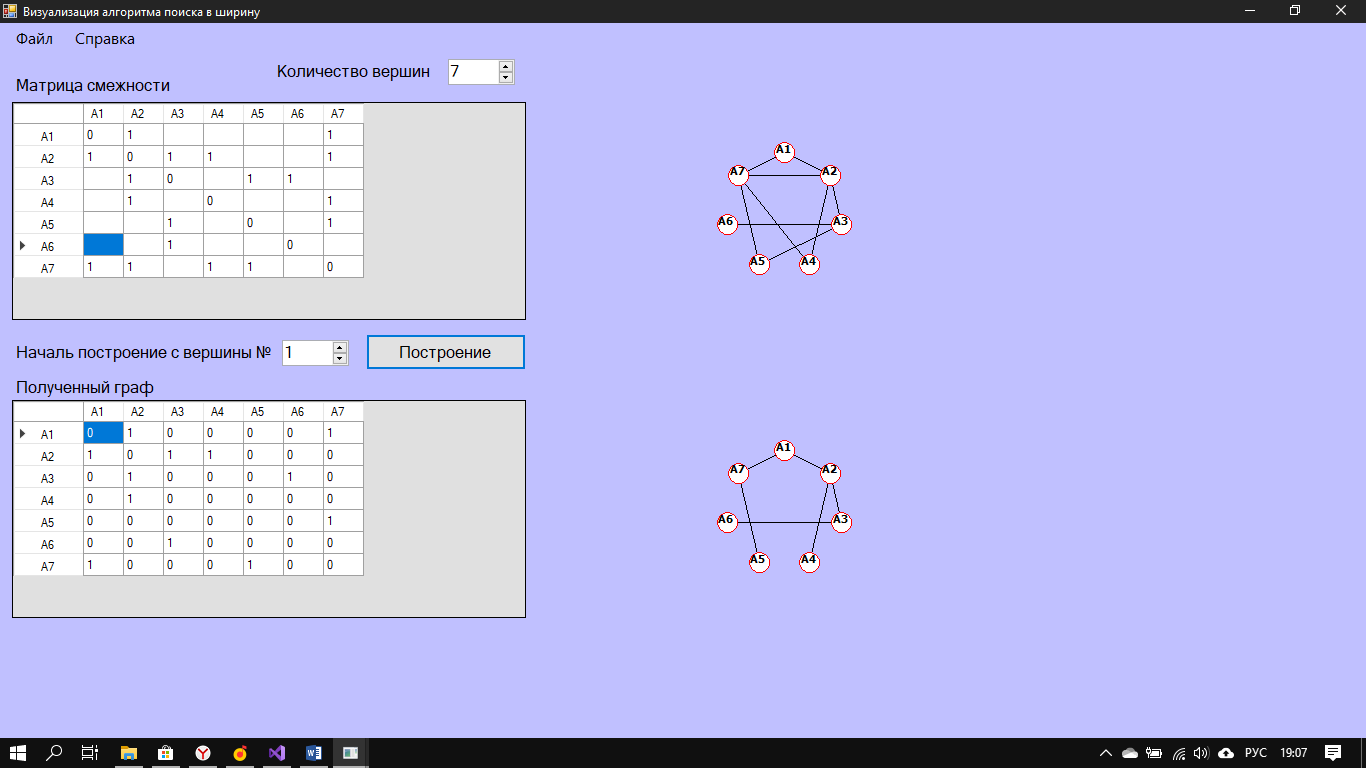


Рисунок 2.2 – Вывод решения

# **Заключение**

В ходе курсовой работы было разработано программное обеспечение, реализующее следующие функции:

1. Поиск остовного дерева при помощи алгоритма обзода графа в ширину;
2. Разработан доступный и простой интерфейс;
3. Программа работает на платформах Windows 7, Windows 10;
4. Организована возможность ввода количества вершин графа, матрицы смежности и номера начальной вершины для поиска.

Практическое использование данной программы возможно во всех случаях, требующих поиска остовного дерева.

Программа протестирована на системе Windows 10, и она является 32-разрядным приложением Windows. Возможно также использование на других платформах Windows.

Поставленная задача решена полностью. Программа готова для практического использования.

# Приложение А

(обязательное)

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ АВТОМАТИКИ И ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

КАФЕДРА РАДИОЭЛЕКТРОННЫХ СРЕДСТВ

УТВЕРЖДАЮ

Зав. Кафедрой РЭС

\_\_\_\_\_\_\_\_\_\_\_Петров Е.П.

**Визуализация алгоритма поиска в ширину**

Описание программы

Лист утверждения

ТПЖА.18207-01 13-01-ЛУ

Листов 2

Разработал: студент группы ИКТб-2302-02-24\_\_\_\_\_\_\_\_\_\_\_\_\_ /Глушков И.С./

Проверил: Руководитель

ст. преподаватель кафедры РЭС \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Наумович Т.В./

Киров 2019

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ АВТОМАТИКИ И ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

КАФЕДРА РАДИОЭЛЕКТРОННЫХ СРЕДСТВ

Утверждён:

ТПЖА.18207-01 13-01-ЛУ

**Визуализация алгоритма поиска в ширину**

Описание программы

ТПЖА.18207-01 13-01

Листов 5

Киров 2019

**Аннотация**

Данный программный документ содержит информацию, которая может помочь пользователю лучше понять логическую структуру и код программы.

В документе также приведены характеристики программы, её назначение, описание входных и выходных данных.

**Содержание**

Изм.

Лист

№ докум.

Подпись

Дата

Лист

3

ТПЖА.18207-01 13-01

Разраб.

Глушков И.С,

Провер.

Наумович Т.В.

Реценз.

Н. Контр.

Утверд.

Приложение А

(Обязательное)

Описание программы

Лит.

Листов

5

Кафедра РЭС,

группа ИКТб-2302-02-24

|  |  |  |
| --- | --- | --- |
| 1 | Общие сведения…………………………………………………… | 13 |
| 2 | Функциональное назначение…………………………………… | 13 |
| 3 | Используемые технические средства……………………………. | 13 |
| 4 | Вызов и загрузка………………………………………………… | 13 |
| 5 | Входные данные…………………………………………………... | 13 |
| 6 | Выходные данные………………………………………………… | 14 |
| 7 | Описание логической структуры………………………………… | 14 |

**1 Общие сведения**

Обозначение и наименование программы: Graph.exe «Визуализация алгоритма поиска в ширину».

Программное обеспечение, необходимое для функционирования программы: ОС Microsoft Windows 10, Microsoft Visual Studio 2017, SDK 10.137 для компиляции.

Языки программирования, на которых написана программа: C#

**2 Функциональное назначение**

Программа graph предназначена для поиска остовного дерева при помощи алгоритма обхода дерева в ширину.

**3 Используемые технические средства**

Программа эксплуатируется на персональном компьютере (ПК). Для работы в диалоговом режиме используется экран дисплея, клавиатура и манипулятор типа «мышь». Входные данные вводятся пользователем с клавиатуры. Программа работает под управлением ОС Microsoft Windows 10.

**4 Вызов и загрузка**

Загрузка программы осуществляется запуском исполняемого файла graph.exe либо запуском программы в среде разработки Microsoft Visual Studio 2017 и компиляцией кода, для работы требуется SDK 10.137, так же .Net Framework версии 4.x

**5 Входные данные**

Условно входные данные можно разделить на две категории:

1. управление ходом выполнения программы;
2. задание параметров для расчета.

Первая категория реализована посредством пользовательского интерфейса. Вторая категория входных данных представляет собой начальные параметры: количество вершин графа и матрица смежности графа, начальная вершина для поиска стягивающего дерева.

**6 Выходные данные**

Выходными данными являются:

* выводимый на экран результат поиска остовного дерева

**7 Описание логической структуры**

Название функции и ее описание представлены ниже.

1. Graph(intnode\_count) – конструктор класса графа.

Входным параметрами являются:

- node\_count - количество вершин графа;

1. Graph(array<array<int>^> ^matr) - конструктор класса графа.

Входным параметрами являются:

- array<array<int>^> ^matr – матрица смежности графа;

3) void init(array<array<int>>matr) – функция инициализации графа

Входным параметрами являются:

- array<array<int>^> ^matr – матрица смежности графа;

4) cliext::vector<par^> ^ run\_width\_next\_step() - функция выполнения одного шага алгоритма нахождения остовного дерева при помощи обхода графа в ширину.

1. voidInitializeComponent(void).

В данной функции происходит инициализация визуальных компонентов окна приложения.

1. VoidnumericUpDown1\_ValueChanged(System::Object^sender,System::EventArgs^e).Функция обрабатывает событие изменения количества вершин графа.
2. button1\_Click(System::Object^sender,System::EventArgs^e)

Реализует инициализацию начальными условиями.

1. Button2\_Click(System::Object^sender,System::EventArgs^e)

Реализует алгоритм поиска остовного дерева минимальной длины.

# Приложение Б

(обязательное)

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ АВТОМАТИКИ И ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

КАФЕДРА РАДИОЭЛЕКТРОННЫХ СРЕДСТВ

УТВЕРЖДАЮ

Зав. Кафедрой РЭС

\_\_\_\_\_\_\_\_\_\_\_Петров Е.П.

**Визуализация алгоритма поиска в ширину**

Руководство оператора

Лист утверждения

ТПЖА.18207-01 34-01-ЛУ

Листов 2

Разработал:

студент группы ИКТб-2302-02-24\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / Глушков И. С/

Проверил: Руководитель

ст. преподаватель кафедры РЭС \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Наумович Т.В./

Киров 2019

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ АВТОМАТИКИ И ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

КАФЕДРА РАДИОЭЛЕКТРОННЫХ СРЕДСТВ

Утверждён:

ТПЖА.18207-01 34-01-ЛУ

**Визуализация алгоритма поиска в ширину**

Руководство оператора

ТПЖА.18207-01 34-01

Листов 5

Киров 2019

**Аннотация**

Данный программный документ предназначен для того, чтобы помочь пользователю, работающему с программой, разобраться в её использовании, а также для демонстрации всех её возможностей.

**Содержание**

1 Назначение и условия применения программы……………….…...19

2 Работа с программой....……………………………………………....19

**1 Назначение и условия применения программы**

Программа разработана в качестве курсовой работы по дисциплине «Языки программирования» по техническому заданию преподавателя кафедры радиоэлектронных средств. Программный продукт представляет из себя реализацию алгоритма поиска стягивающего дерева путём обхода графа в ширину. Программу можно использовать, как прикладное решение поиске СД.

**2 Работа с программой**

Загрузка программы осуществляется запуском исполняемого файла grahp.exe либо запуском программы в среде разработки Microsoft Visual Studio 2017, обязательно наличие SDK 10.137 и Net.Framework 4.х (рисунок Б.1).

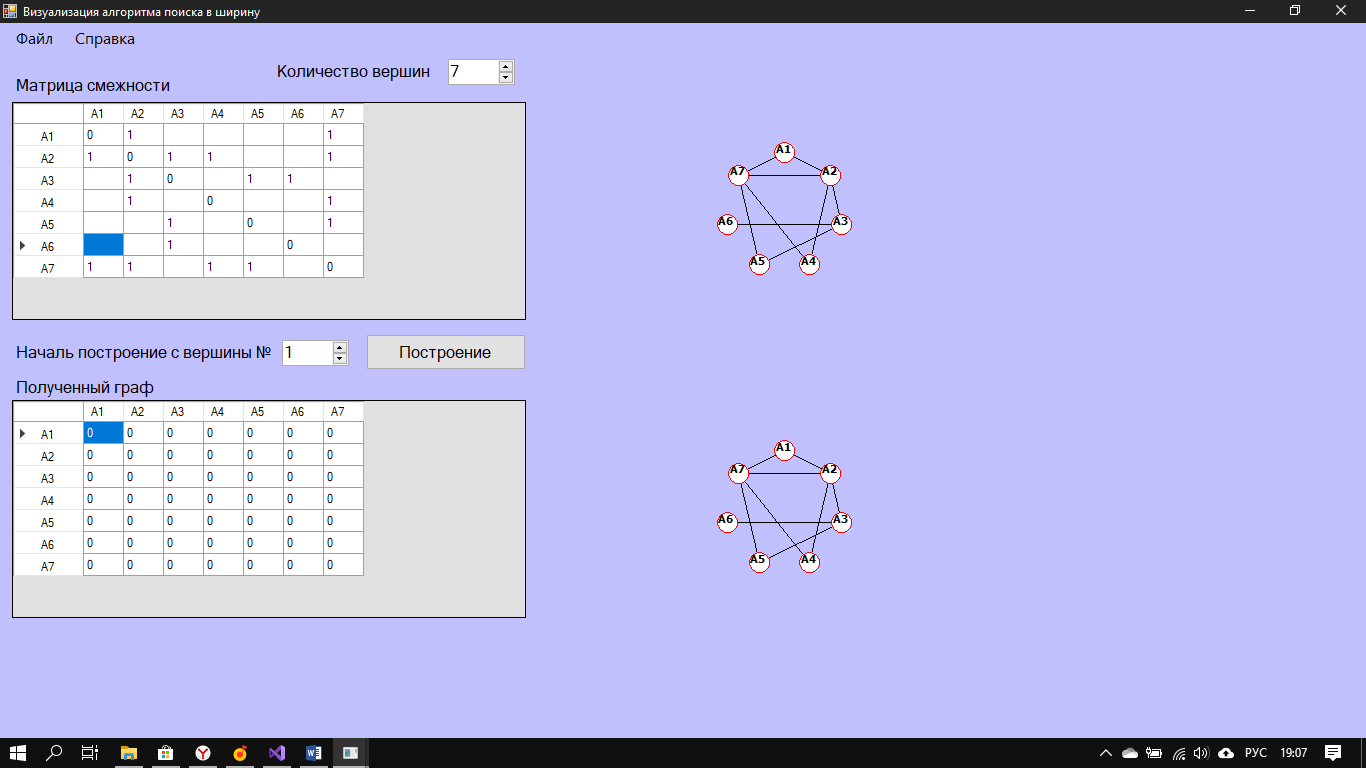


Рисунок Б.1 – Окно программы

Входными данными для программы являются вводимые пользователем с клавиатуры значения:

* Количество вершин графа;
* Матрица смежности;
* Начальная вершина для поиска стягивающего дерева.

Выходными данными являются:

* Список ребер остовного дерева.

Приложение В

(обязательное)

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ АВТОМАТИКИ И ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

КАФЕДРА РАДИОЭЛЕКТРОННЫХ СРЕДСТВ

УТВЕРЖДАЮ

Зав. Кафедрой РЭС

\_\_\_\_\_\_\_\_\_\_\_Петров Е.П.

**Визуализация алгоритма поиска в ширину**

Текст программы

Лист утверждения

ТПЖА.18207-01 12-01-ЛУ

Листов 2

Разработал:

студент группы ИКТб-2302-02-24\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Глушков И. С/

Проверил: Руководитель

ст. преподаватель кафедры РЭС \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Наумович Т.В./

Киров 2019

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ АВТОМАТИКИ И ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

КАФЕДРА РАДИОЭЛЕКТРОННЫХ СРЕДСТВ

Утверждён:

ТПЖА.18207-01 12-01-ЛУ

**Визуализация алгоритма поиска в ширину**

Текст программы

ТПЖА.18207-01 12-01

Листов 17

Киров 2019

**Аннотация**

Данный программный документ содержит полный исходный код программы на языке С++.

**Содержание**

1 MyForm.cpp……………………………………………………………......…….23

2 MyForm.h……………………………………………………………......……….24

3 graphClass.h……..…………………………………………………...……….......37

**1 MyForm.cpp**

#include "myForm.h"

using namespace graph;

[STAThreadAttribute]

int main(cli::array<System::String ^> ^args)

{

Application::EnableVisualStyles();

Application::SetCompatibleTextRenderingDefault(false);

Application::Run(gcnew myForm());

return 0;

}

**2MyForm.h**

#pragma once

#include "graphClass.h"

#include <cliext\vector>

#include "about.h"

namespace graph {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

/// <summary>

/// Summary for myForm

/// </summary>

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

{

public:

myForm(void)

{

InitializeComponent();

//

//TODO: Add the constructor code here

//

}

protected:

/// <summary>

/// Clean up any resources being used.

/// </summary>

~myForm()

{

if (components)

{

delete components;

}

}

private: System::Windows::Forms::DataGridView^ dataGridView1;

protected:

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

private: System::Windows::Forms::NumericUpDown^ numericUpDown1;

private: System::Windows::Forms::Label^ label2;

private: System::Windows::Forms::Button^ button1;

private: System::Windows::Forms::DataGridView^ dataGridView2;

private: System::Windows::Forms::Label^ label3;

private: System::Windows::Forms::Label^ label4;

private: System::Windows::Forms::NumericUpDown^ numericUpDown2;

private: System::Windows::Forms::PictureBox^ pictureBox1;

private: System::Windows::Forms::PictureBox^ pictureBox2;

public:

ref struct Vert{

int x;

int y;

int color;

};

Graph ^graph;

cliext::vector<Vert^> ^rebra = gcnew cliext::vector<Vert^>();

private: System::Windows::Forms::MenuStrip^ menuStrip1;

private: System::Windows::Forms::ToolStripMenuItem^ файлToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ выходToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ справкаToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ о ПрограммеToolStripMenuItem;

private: System::Windows::Forms::Timer^ timer1;

private: System::ComponentModel::IContainer^ components;

public:

/// <summary>

/// Required designer variable.

/// </summary>

#pragma region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

void InitializeComponent(void)

{

this->components = (gcnew System::ComponentModel::Container());

this->dataGridView1 = (gcnew System::Windows::Forms::DataGridView());

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

this->numericUpDown1 = (gcnew System::Windows::Forms::NumericUpDown());

this->label2 = (gcnew System::Windows::Forms::Label());

this->button1 = (gcnew System::Windows::Forms::Button());

this->dataGridView2 = (gcnew System::Windows::Forms::DataGridView());

this->label3 = (gcnew System::Windows::Forms::Label());

this->label4 = (gcnew System::Windows::Forms::Label());

this->numericUpDown2 = (gcnew System::Windows::Forms::NumericUpDown());

this->pictureBox1 = (gcnew System::Windows::Forms::PictureBox());

this->pictureBox2 = (gcnew System::Windows::Forms::PictureBox());

this->menuStrip1 = (gcnew System::Windows::Forms::MenuStrip());

this->файлToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->выходToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->справкаToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->оПрогрпаммеToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->timer1 = (gcnew System::Windows::Forms::Timer(this->components));

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView1))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->numericUpDown1))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView2))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->numericUpDown2))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureBox1))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureBox2))->BeginInit();

this->menuStrip1->SuspendLayout();

this->SuspendLayout();

//

// dataGridView1

//

this->dataGridView1->AllowUserToAddRows = false;

this->dataGridView1->BackgroundColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(224)),

static\_cast<System::Int32>(static\_cast<System::Byte>(224)), static\_cast<System::Int32>(static\_cast<System::Byte>(224)));

this->dataGridView1->ColumnHeadersHeightSizeMode = System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;

this->dataGridView1->Location = System::Drawing::Point(12, 79);

this->dataGridView1->Name = L"dataGridView1";

this->dataGridView1->Size = System::Drawing::Size(514, 218);

this->dataGridView1->TabIndex = 0;

this->dataGridView1->CellContentClick += gcnew System::Windows::Forms::DataGridViewCellEventHandler(this, &myForm::dataGridView1\_CellContentClick);

this->dataGridView1->CellValueChanged += gcnew System::Windows::Forms::DataGridViewCellEventHandler(this, &myForm::dataGridView1\_CellValueChanged);

//

// label1

//

this->label1->AutoSize = true;

this->label1->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->label1->Location = System::Drawing::Point(273, 38);

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

this->label1->Size = System::Drawing::Size(162, 20);

this->label1->TabIndex = 1;

this->label1->Text = L"Количество вершин";

//

// numericUpDown1

//

this->numericUpDown1->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->numericUpDown1->Location = System::Drawing::Point(448, 36);

this->numericUpDown1->Name = L"numericUpDown1";

this->numericUpDown1->Size = System::Drawing::Size(67, 26);

this->numericUpDown1->TabIndex = 2;

this->numericUpDown1->ValueChanged += gcnew System::EventHandler(this, &myForm::numericUpDown1\_ValueChanged);

//

// label2

//

this->label2->AutoSize = true;

this->label2->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->label2->Location = System::Drawing::Point(12, 52);

this->label2->Name = L"label2";

this->label2->Size = System::Drawing::Size(163, 20);

this->label2->TabIndex = 3;

this->label2->Text = L"Матрица смежности";

//

// button1

//

this->button1->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->button1->Location = System::Drawing::Point(366, 311);

this->button1->Name = L"button1";

this->button1->Size = System::Drawing::Size(160, 36);

this->button1->TabIndex = 4;

this->button1->Text = L"Построение";

this->button1->UseVisualStyleBackColor = true;

this->button1->Click += gcnew System::EventHandler(this, &myForm::button1\_Click);

//

// dataGridView2

//

this->dataGridView2->AllowUserToAddRows = false;

this->dataGridView2->BackgroundColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(224)),

static\_cast<System::Int32>(static\_cast<System::Byte>(224)), static\_cast<System::Int32>(static\_cast<System::Byte>(224)));

this->dataGridView2->ColumnHeadersHeightSizeMode = System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;

this->dataGridView2->Location = System::Drawing::Point(12, 377);

this->dataGridView2->Name = L"dataGridView2";

this->dataGridView2->Size = System::Drawing::Size(514, 218);

this->dataGridView2->TabIndex = 5;

//

// label3

//

this->label3->AutoSize = true;

this->label3->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->label3->Location = System::Drawing::Point(12, 354);

this->label3->Name = L"label3";

this->label3->Size = System::Drawing::Size(147, 20);

this->label3->TabIndex = 6;

this->label3->Text = L"Полученный граф";

//

// label4

//

this->label4->AutoSize = true;

this->label4->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->label4->Location = System::Drawing::Point(12, 319);

this->label4->Name = L"label4";

this->label4->Size = System::Drawing::Size(264, 20);

this->label4->TabIndex = 7;

this->label4->Text = L"Началь построение с вершины №";

//

// numericUpDown2

//

this->numericUpDown2->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

this->numericUpDown2->Location = System::Drawing::Point(282, 317);

this->numericUpDown2->Minimum = System::Decimal(gcnew cli::array< System::Int32 >(4) { 1, 0, 0, 0 });

this->numericUpDown2->Name = L"numericUpDown2";

this->numericUpDown2->Size = System::Drawing::Size(67, 26);

this->numericUpDown2->TabIndex = 8;

this->numericUpDown2->Value = System::Decimal(gcnew cli::array< System::Int32 >(4) { 1, 0, 0, 0 });

//

// pictureBox1

//

this->pictureBox1->Location = System::Drawing::Point(642, 79);

this->pictureBox1->Name = L"pictureBox1";

this->pictureBox1->Size = System::Drawing::Size(284, 218);

this->pictureBox1->TabIndex = 9;

this->pictureBox1->TabStop = false;

//

// pictureBox2

//

this->pictureBox2->Location = System::Drawing::Point(642, 377);

this->pictureBox2->Name = L"pictureBox2";

this->pictureBox2->Size = System::Drawing::Size(284, 218);

this->pictureBox2->TabIndex = 10;

this->pictureBox2->TabStop = false;

this->pictureBox2->Click += gcnew System::EventHandler(this, &myForm::pictureBox2\_Click);

//

// menuStrip1

//

this->menuStrip1->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(192)), static\_cast<System::Int32>(static\_cast<System::Byte>(192)),

static\_cast<System::Int32>(static\_cast<System::Byte>(255)));

this->menuStrip1->Items->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^ >(2) {

this->файлToolStripMenuItem,

this->справкаToolStripMenuItem

});

this->menuStrip1->Location = System::Drawing::Point(0, 0);

this->menuStrip1->Name = L"menuStrip1";

this->menuStrip1->Size = System::Drawing::Size(913, 29);

this->menuStrip1->TabIndex = 12;

this->menuStrip1->Text = L"menuStrip1";

//

// файлToolStripMenuItem

//

this->файлToolStripMenuItem->DropDownItems->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^ >(1) { this->выходToolStripMenuItem });

this->файлToolStripMenuItem->Font = (gcnew System::Drawing::Font(L"Segoe UI", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

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

this->файлToolStripMenuItem->Size = System::Drawing::Size(59, 25);

this->файлToolStripMenuItem->Text = L"Файл";

//

// выходToolStripMenuItem

//

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

this->выходToolStripMenuItem->Size = System::Drawing::Size(180, 26);

this->выходToolStripMenuItem->Text = L"Выход";

this->выходToolStripMenuItem->Click += gcnew System::EventHandler(this, &myForm::выходToolStripMenuItem\_Click);

//

// справкаToolStripMenuItem

//

this->справкаToolStripMenuItem->DropDownItems->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^ >(1) { this->оПрогрпаммеToolStripMenuItem });

this->справкаToolStripMenuItem->Font = (gcnew System::Drawing::Font(L"Segoe UI", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

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

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

this->справкаToolStripMenuItem->Size = System::Drawing::Size(82, 25);

this->справкаToolStripMenuItem->Text = L"Справка";

//

// оПрогрпаммеToolStripMenuItem

//

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

this->оПрогрпаммеToolStripMenuItem->Size = System::Drawing::Size(180, 26);

this->оПрогрпаммеToolStripMenuItem->Text = L"О программе";

this->оПрогрпаммеToolStripMenuItem->Click += gcnew System::EventHandler(this, &myForm::оПрогрпаммеToolStripMenuItem\_Click);

//

// timer1

//

this->timer1->Interval = 500;

this->timer1->Tick += gcnew System::EventHandler(this, &myForm::timer1\_Tick);

//

// myForm

//

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

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

this->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(192)), static\_cast<System::Int32>(static\_cast<System::Byte>(192)),

static\_cast<System::Int32>(static\_cast<System::Byte>(255)));

this->ClientSize = System::Drawing::Size(913, 591);

this->Controls->Add(this->pictureBox2);

this->Controls->Add(this->pictureBox1);

this->Controls->Add(this->numericUpDown2);

this->Controls->Add(this->label4);

this->Controls->Add(this->label3);

this->Controls->Add(this->dataGridView2);

this->Controls->Add(this->button1);

this->Controls->Add(this->label2);

this->Controls->Add(this->numericUpDown1);

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

this->Controls->Add(this->dataGridView1);

this->Controls->Add(this->menuStrip1);

this->MainMenuStrip = this->menuStrip1;

this->Name = L"myForm";

this->Text = L"Визуализация алгоритма поиска в ширину";

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView1))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->numericUpDown1))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView2))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->numericUpDown2))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureBox1))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureBox2))->EndInit();

this->menuStrip1->ResumeLayout(false);

this->menuStrip1->PerformLayout();

this->ResumeLayout(false);

this->PerformLayout();

}

#pragma endregion

private: void repaint(int count, cliext::vector <Vert^> ^reb)

{

int pW = pictureBox1->Width, pH = pictureBox1->Height;

Bitmap ^img = gcnew Bitmap(pW, pH);

Graphics ^g = Graphics::FromImage(img);

//находим середину

int mX = int(pW / 2);

int mY = int(pH / 2);

g->ScaleTransform(1, -1); //переворачиваем ось Y для удобства восприятия

g->TranslateTransform((float)mX, -(float)mY); //смещаем нулевую координату на пересечение осей

int n = count;

int R = pH / 2 - 50;

double alfa = 0;

SolidBrush^ drawBrush = gcnew SolidBrush(Color::White);

cli::array<Vert^>^v = gcnew cli::array<Vert^>(n);

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

{

double x = R \* Math::Sin(alfa);

double y = R \* Math::Cos(alfa);

alfa += 2 \* 3.14 / n;

v[i] = gcnew Vert;

v[i]->x = x;

v[i]->y = y;

}

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

{

g->DrawLine(Pens::Black, v[reb->at(i)->x]->x, v[reb->at(i)->x]->y, v[reb->at(i)->y]->x, v[reb->at(i)->y]->y);

}

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

{

g->FillEllipse(drawBrush, RectangleF(v[i]->x - 10, v[i]->y - 10, 20, 20));

g->DrawEllipse(Pens::Red, RectangleF(v[i]->x - 10, v[i]->y - 10, 20, 20));

}

alfa = 0;

SolidBrush^ drawBrush1 = gcnew SolidBrush(Color::Black);

System::Drawing::Font ^font = gcnew System::Drawing::Font("Tahoma", 8, FontStyle::Bold);

g->ScaleTransform(1, -1); //переворачиваем ось Y для удобства восприятия

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

{

double x = R \* Math::Sin(alfa);

double y = (R - R \* Math::Cos(alfa)) - R;

alfa += 2 \* 3.14 / n;

g->DrawString("A" + (i + 1), font, drawBrush1, x - 10, y - 10);

}

delete g; //освобождение ресурсов устройства рисования

this->pictureBox1->Image = img; //присвоение и отображение изображения в PictureBox

this->pictureBox2->Image = img;

}

private: void repaint2(int count, cliext::vector <Vert^> ^reb)

{

int pW = pictureBox2->Width, pH = pictureBox1->Height;

Bitmap ^img = gcnew Bitmap(pW, pH);

Graphics ^g = Graphics::FromImage(img);

//находим середину

int mX = int(pW / 2);

int mY = int(pH / 2);

g->ScaleTransform(1, -1); //переворачиваем ось Y для удобства восприятия

g->TranslateTransform((float)mX, -(float)mY); //смещаем нулевую координату на пересечение осей

int n = count;

int R = pH / 2 - 50;

double alfa = 0;

SolidBrush^ drawBrush = gcnew SolidBrush(Color::White);

cli::array<Vert^>^v = gcnew cli::array<Vert^>(n);

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

{

double x = R \* Math::Sin(alfa);

double y = R \* Math::Cos(alfa);

alfa += 2 \* 3.14 / n;

v[i] = gcnew Vert;

v[i]->x = x;

v[i]->y = y;

}

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

{

g->DrawLine(Pens::Black, v[reb->at(i)->x]->x, v[reb->at(i)->x]->y, v[reb->at(i)->y]->x, v[reb->at(i)->y]->y);

}

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

{

g->FillEllipse(drawBrush, RectangleF(v[i]->x - 10, v[i]->y - 10, 20, 20));

g->DrawEllipse(Pens::Red, RectangleF(v[i]->x - 10, v[i]->y - 10, 20, 20));

}

alfa = 0;

SolidBrush^ drawBrush1 = gcnew SolidBrush(Color::Black);

System::Drawing::Font ^font = gcnew System::Drawing::Font("Tahoma", 8, FontStyle::Bold);

g->ScaleTransform(1, -1); //переворачиваем ось Y для удобства восприятия

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

{

double x = R \* Math::Sin(alfa);

double y = (R - R \* Math::Cos(alfa)) - R;

alfa += 2 \* 3.14 / n;

g->DrawString("A" + (i + 1), font, drawBrush1, x - 10, y - 10);

}

delete g; //освобождение ресурсов устройства рисования

this->pictureBox2->Image = img;

}

private: System::Void numericUpDown1\_ValueChanged(System::Object^ sender, System::EventArgs^ e) {

int count = System::Convert::ToInt32(numericUpDown1->Value);

while (dataGridView1->RowCount >1)

{

dataGridView1->Rows->RemoveAt(0);

}

while (dataGridView2->RowCount >1)

{

dataGridView2->Rows->RemoveAt(0);

}

dataGridView1->Columns->Clear();

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

{

String ^name = "A" + Convert::ToString(i + 1);

String ^caption = "A" + Convert::ToString(i + 1);

dataGridView1->Columns->Add(caption, name);

dataGridView1->Columns[i]->Width = 40;

}

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

{

String ^nane = "X" + Convert::ToString(i + 1);

String ^caption = "A" + Convert::ToString(i + 1);

dataGridView1->Rows->Add();

dataGridView1->Rows[i]->HeaderCell->Value = "A" + Convert::ToString(i + 1) ;

dataGridView1->RowHeadersWidthSizeMode = DataGridViewRowHeadersWidthSizeMode::AutoSizeToAllHeaders;

}

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

dataGridView1->Rows[i]->Cells[i]->Value = 0;

dataGridView2->Columns->Clear();

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

{

String ^name = "A" + Convert::ToString(i + 1);

String ^caption = "A" + Convert::ToString(i + 1);

dataGridView2->Columns->Add(caption, name);

dataGridView2->Columns[i]->Width = 40;

}

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

{

String ^nane = "X" + Convert::ToString(i + 1);

String ^caption = "A" + Convert::ToString(i + 1);

dataGridView2->Rows->Add();

dataGridView2->Rows[i]->HeaderCell->Value = "A" + Convert::ToString(i + 1);

dataGridView2->RowHeadersWidthSizeMode = DataGridViewRowHeadersWidthSizeMode::AutoSizeToAllHeaders;

}

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

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

dataGridView2->Rows[i]->Cells[j]->Value = 0;

repaint(count, gcnew cliext::vector<Vert^> ());

}

private: System::Void dataGridView1\_CellContentClick(System::Object^ sender, System::Windows::Forms::DataGridViewCellEventArgs^ e) {

}

private: System::Void dataGridView1\_CellValueChanged(System::Object^ sender, System::Windows::Forms::DataGridViewCellEventArgs^ e) {

if (dataGridView1->RowCount != dataGridView1->ColumnCount) return;

if (e->ColumnIndex < 0 || e->RowIndex < 0) return;

dataGridView1->Rows[e->ColumnIndex]->Cells[e->RowIndex]->Value = dataGridView1->Rows[e->RowIndex]->Cells[e->ColumnIndex]->Value;

rebra->clear();

for (int i = 0; i < dataGridView1->RowCount; i++)

{

for (int j = 0; j < dataGridView1->ColumnCount; j++)

if (dataGridView1->Rows[i]->Cells[j]->Value != 0 && dataGridView1->Rows[i]->Cells[j]->Value != nullptr && i != j)

{

String ^str = gcnew String("");

str = dataGridView1->Rows[i]->Cells[j]->Value->ToString();

Vert ^a = gcnew Vert;

a->x = i;

a->y = j;

rebra->push\_back(a);

}

}

int count = System::Convert::ToInt32(numericUpDown1->Value);

repaint(count, rebra);

}

private: System::Void button1\_Click(System::Object^ sender, System::EventArgs^ e) {

int count = System::Convert::ToInt32(numericUpDown1->Value);

if (count <= 1) return;

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

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

dataGridView2->Rows[i]->Cells[j]->Value = 0;

cli::array<cli::array<int>^> ^matr = gcnew cli::array <cli::array<int>^>(dataGridView1->ColumnCount);

for (int i = 0; i < dataGridView1->ColumnCount; i++)

{

matr[i] = gcnew cli::array<int>(dataGridView1->ColumnCount);

}

for (int i = 0; i < dataGridView1->ColumnCount; i++)

{

for (int j = 0; j < dataGridView1->ColumnCount; j++)

matr[i][j] = System::Convert::ToDouble(dataGridView1->Rows[i]->Cells[j]->Value);

}

graph = gcnew Graph(matr);

vector<pair<int, int>> newGraph;

graph->SetFirstNode(System::Convert::ToInt32(numericUpDown2->Value) - 1);

rebra->clear();

// button2->Enabled = true;

timer1->Start();

}

private: System::Void button2\_Click(System::Object^ sender, System::EventArgs^ e) {

cliext::vector<Graph::par^> ^result\_graph = graph->run\_width\_next\_step();

int count = System::Convert::ToInt32(numericUpDown1->Value);

for (int i = 0; i < result\_graph->size(); i++)

{

Vert ^a = gcnew Vert;

a->x = result\_graph->at(i)->first;

a->y = result\_graph->at(i)->second;

rebra->push\_back(a);

}

repaint2(count, rebra);

int i = 0;

int first = 0;

for each(Graph::par^ a in result\_graph)

{

if (a->first == a->second)

{

first = a->second;

continue;

}

if (a->first != first)

{

i++;

first = a->first;

}

dataGridView2->Rows[a->first]->Cells[a->second]->Value = dataGridView1->Rows[a->first]->Cells[a->second]->Value;

dataGridView2->Rows[a->second]->Cells[a->first]->Value = dataGridView1->Rows[a->second]->Cells[a->first]->Value;

}

if (result\_graph->size() == 0) timer1->Stop();

}

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

Close();

}

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

about ^a = gcnew about();

a->Show();

}

private: System::Void timer1\_Tick(System::Object^ sender, System::EventArgs^ e) {

button2\_Click(sender, e);

}

private: System::Void pictureBox2\_Click(System::Object^ sender, System::EventArgs^ e) {

}

};

}

**3 graphClass.h**

#include <stdlib.h>

#include <vector>

#include <algorithm>

#include <utility>

#include <cliext\vector>

using namespace std;

public ref class Graph

{

private: int f\_node;

public:

ref struct par{

int first;

int second;

};

void SetFirstNode(int n) {

f\_node = n;

par ^p = gcnew par;

p->first = f\_node;

p->second = f\_node;

queue->push\_back(p);

}

Graph(int node\_count)

{

this->node\_count = node\_count;

graph\_matr = gcnew cli::array<cli::array<int>^>(node\_count);

for (int i = 0; i < node\_count; i++)

{

graph\_matr[i] = gcnew cli::array<int>(node\_count);

}

nodes = gcnew cli::array<int>(node\_count);

for (int i = 0; i < node\_count; i++)

nodes[i] = 0;

};

Graph(cli::array<cli::array<int>^> ^matr)

{

node\_count = matr[0]->Length;

graph\_matr = gcnew cli::array<cli::array<int>^>(node\_count);

for (int i = 0; i < node\_count; i++)

{

graph\_matr[i] = gcnew cli::array<int>(node\_count);

}

init(matr);

nodes = gcnew cli::array<int>(node\_count);

for (int i = 0; i < node\_count; i++)

nodes[i] = 0;

};

~Graph()

{

};

void init(cli::array<cli::array<int>^> ^matr)

{

if (matr->Length != node\_count) node\_count = matr->Length;

for (int i = 0; i < node\_count; i++)

for (int j = 0; j < node\_count; j++)

graph\_matr[i][j] = matr[i][j];

}

bool inGraph(int node, cliext::vector<par^> ^graph)

{

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

{

if (graph->at(i)->first == node || graph->at(i)->second == node) return true;

}

return false;

}

cliext::vector<par^> ^ run\_width\_next\_step()

{

if (!queue->empty())

{

int first = queue->front()->first;

int next = queue->front()->second;

queue->erase(queue->begin(), queue->begin() + 1);

if (!inGraph(next, result\_graph))

{

par ^p = gcnew par;

p->first = first;

p->second = next;

result\_graph->push\_back(p);

for (int i = 0; i < node\_count; i++)

{

if (next != i && graph\_matr[next][i] != 0)

{

if (!inGraph(i, queue) && !inGraph(i, result\_graph))

{

int newnode = i;

par ^p = gcnew par;

p->first = next;

p->second = i;

queue->push\_back(p);

}

}

}

}

}

return result\_graph;

};

private:

int node\_count;

cli::array<cli::array<int>^> ^graph\_matr;

cliext::vector<par^> ^queue = gcnew cliext::vector<par^>();

cliext::vector<par^> ^result\_graph = gcnew cliext::vector<par^>();

cli::array<int> ^nodes;

};

**Приложение Г**

(справочное)

**Библиографический список**

1. Турчак Л.И. Основы численных методов: Учеб. пособие / Л.И. Турчак. – М.: Наука. Гл.ред.физ.-мат.лит., 1987. - 320с.

2. Шилдт, Г. Полный справочник по С++, 4-е издание / Г. Шилдт. : Пер. с англ. - М. : Издательский дома "Вильямс", 2010. - 800с. - ISBN 978-5-8459-0489-8.

3. Наумович Т.В Языки программирования: методические указания к курсовой работе, ВятГУ, 2013 . 26с