**АННОТАЦИЯ**

В данном программном документе приведена пояснительная записка к программе «GraphFormerForCPP.ехе» («Программа поиска маршрута китайского почтальона»), предназначенной для поиска маршрута китайского почтальона на ориентированном и неориентированном мультиграфе без петель.

В разделе «Введение» указано наименование программы, краткое наименование программы и документы, на основании которых ведется разработка.

В разделе «Назначение и область применения» указано функциональное назначение программы, эксплуатационное назначение программы и краткая характеристика области применения программы.

В разделе «Технические характеристики» содержатся следующие подразделы:

* постановка задачи на разработку программы;
* описание алгоритма и функционирования программы с обоснованием выбора схемы алгоритма решения задачи и возможные взаимодействия программы с другими программами;
* описание и обоснование выбора метода организации входных и выходных данных;
* описание и обоснование выбора состава технических и программных средств.

В разделе «Ожидаемые технико-экономические показатели» указана предполагаемая потребность и экономические преимущества разработки по сравнению с отечественными и зарубежными образцами или аналогами

Настоящий документ разработан в соответствии с требованиями:

1. ГОСТ 19.101-77 Виды программ и программных документов [1];
2. ГОСТ 19.102-77 Стадии разработки [2];
3. ГОСТ 19.103-77 Обозначения программ и программных документов [3];
4. ГОСТ 19.104-78 Основные надписи [4];
5. ГОСТ 19.105-78 Общие требования к программным документам [5];
6. ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом [6];
7. ГОСТ 19.404-79 Пояснительная записка. Требования к содержанию и оформлению [7].

Изменения к Пояснительной записке оформляются согласно ГОСТ 19.603-78 [8], ГОСТ 19.604-78 [9].

Перед прочтением данного документа рекомендуется ознакомиться с терминологией, приведенной в Приложении 1 настоящей пояснительной записки. В данном документе под мультиграфом подразумеваем мультиграф без петель.

СОДЕРЖАНИЕ

[1. ВВЕДЕНИЕ 5](#_Toc451722317)

[1.1. Наименование программы 5](#_Toc451722318)

[1.2. Документы, на основании которых ведется разработка 5](#_Toc451722319)

[2. НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ 6](#_Toc451722320)

[2.1. Назначение программы 6](#_Toc451722321)

[2.1.1. Функциональное назначение 6](#_Toc451722322)

[2.1.2. Эксплуатационное назначение 6](#_Toc451722323)

[2.2. Краткая характеристика области применения 6](#_Toc451722324)

[3. ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ 7](#_Toc451722325)

[3.1. Постановка задачи на разработку программы 7](#_Toc451722326)

[3.2. Описание алгоритма и функционирования программы 7](#_Toc451722327)

[3.2.1. Описание алгоритма и функционирования программы для ориентированного мультиграфа 7](#_Toc451722328)

[3.2.1.1. Нахождение расстояний между всеми парами вершин алгоритмом Флойда-Уоршелла 8](#_Toc451722329)

[3.2.1.2. Алгоритм проверки ориентированного мультиграфа на сильную связность и на наличие циклов нулевой и отрицательной длины 9](#_Toc451722330)

[3.2.1.3. Алгоритм нахождения несбалансированных вершин ориентированного мультиграфа 9](#_Toc451722331)

[3.2.1.4. Нахождение дуги между парой несбалансированных вершин с минимальным весом и добавление этой дуги в мультиграф 10](#_Toc451722332)

[3.2.2. Описание алгоритма и функционирования программы для неориентированного мультиграфа 11](#_Toc451722333)

[3.2.2.1. Нахождение расстояний между всеми парами вершин алгоритмом Флойда-Уоршелла 12](#_Toc451722334)

[3.2.2.2. Алгоритм проверки неориентированного мультиграфа на связность и на наличие циклов нулевой и отрицательной длины 12](#_Toc451722335)

[3.2.2.3. Алгоритм нахождения нечетных вершин неориентированного мультиграфа 12](#_Toc451722336)

[3.2.2.4. Нахождение дуги между парой несбалансированных вершин с минимальным весом и добавление этой дуги в мультиграф 13](#_Toc451722337)

[3.2.3. Обоснование выбора алгоритма решения задачи 14](#_Toc451722338)

[3.2.4. Возможные взаимодействия программы с другими программами 14](#_Toc451722339)

[3.3. Описание и обоснование выбора метода организации входных и выходных данных 14](#_Toc451722340)

[3.3.1. Описание метода организации входных и выходных данных 14](#_Toc451722341)

[3.3.2. Обоснования выбора метода организации входных и выходных данных 15](#_Toc451722342)

[3.4. Описание и обоснование выбора состава технических и программных средств 15](#_Toc451722343)

[3.4.1. Состав технических и программных средств 15](#_Toc451722344)

[3.4.2. Обоснование выбора технических и программных средств 15](#_Toc451722345)

[4. ОЖИДАЕМЫЕ ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ 17](#_Toc451722346)

[4.1. Предполагаемая потребность 17](#_Toc451722347)

[4.2. Экономические преимущества разработки по сравнению с отечественными и зарубежными образцами или аналогами 17](#_Toc451722348)

[5. СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 18](#_Toc451722349)

[ПРИЛОЖЕНИЕ 1 ТЕРМИНОЛОГИЯ 19](#_Toc451722350)

[ПРИЛОЖЕНИЕ 2 ОПИСАНИЕ ФОРМАТА .cpost 20](#_Toc451722351)

[ПРИЛОЖЕНИЕ 3 ДИАГРАММА ВАРИАНТОВ ИСПОЛЬЗОВАНИЯ 21](#_Toc451722352)

[ПРИЛОЖЕНИЕ 4 ДИАГРАММА КЛАССОВ 22](#_Toc451722353)

[ПРИЛОЖЕНИЕ 5 ДИАГРАММА ДЕЯТЕЛЬНОСТИ 23](#_Toc451722354)

[ПРИЛОЖЕНИЕ 6 ДИАГРАММА КООПЕРАЦИИ 24](#_Toc451722355)

[ПРИЛОЖЕНИЕ 7 ДИАГРАММА КОМПОНЕНТОВ 25](#_Toc451722356)

[ПРИЛОЖЕНИЕ 8 ДИАГРАММА ПОСЛЕДОВАТЕЛЬНОСТЕЙ 26](#_Toc451722357)

[ПРИЛОЖЕНИЕ 9 ДИАГРАММА СОСТОЯНИЙ 27](#_Toc451722358)

[ПРИЛОЖЕНИЕ 10 ОПИСАНИЕ И ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ КЛАССОВ 28](#_Toc451722359)

[ПРИЛОЖЕНИЕ 11 ОПИСАНИЕ И ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ ПОЛЕЙ МЕТОДОВ И СВОЙСТВ 29](#_Toc451722360)

1. **ВВЕДЕНИЕ**
   1. **Наименование программы**

Наименование программы – «Программа поиска маршрута китайского почтальона».

Краткое наименование программы – «GraphFormerForCPP».

Добавить Наименование программы на английском языке

* 1. **Документы, на основании которых ведется разработка**

Разработка ведется на основании приказа Национального исследовательского университета "Высшая школа экономики" № \_\_\_\_\_\_\_\_\_\_\_\_ от \_\_\_\_\_\_\_\_\_\_\_\_.

Номер приказа – см. в ЛМС.

Кроме того, надо записать наименование приказа (тоже в ЛМС)

1. **НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ**

81 – код Пояснительной записки

04.01 – вместо этого ставите свой класс ПО (из ТЗ)

* 1. **Назначение программы**
     1. **Функциональное назначение**

Функциональным назначением программы является поиск во взвешенном связном неориентированном или во взвешенном сильно связном ориентированном мультиграфе без петель кратчайшего пути, начинающегося и заканчивающегося в одной и той же вершине и проходящего через каждое ребро (в неориентированном)/дугу (в неориентированном) мультиграфа без петель, по крайней мере, один раз. Если перейти от понятий теории графов к повседневной терминологии, то функциональное значение программы заключается в том, чтобы пройти все улицы заданного маршрута с определенной длиной каждой дороги, входящей в маршрут, и вернуться в начальную точку, пройдя при этом как можно меньшее расстояние [10, c.219].

* + 1. **Эксплуатационное назначение**

Решение задачи китайского почтальона является востребованным на сегодняшний день, так как существует множество процессов, для выполнения которых необходимо знать кратчайший путь, проходящий через каждую дорогу заданного маршрута хотя бы один раз, и который при этом вернется в исходную точку. Задача китайского почтальона и ее решение могут быть использованы при реализации следующих потенциально полезных приложений [10, c.219]:

Для студентов ПМИ штамп в нижнем колонтитуле не нужен (не обязателен)

Какие функции должна выполнять программа

Где, когда, кем, с какой целью может использоваться программа

1) минимизация затрат (физических, материальных, временных) при доставке молока, писем, посылок;

2) расчет оптимального пути (по километражу, времени или стоимости перевозки) сбора мусора в определенном районе города при наличии только одной мусороуборочной машины на участок, патрулирования улиц, ремонтных и проверяющих бригад, для сельскохозяйственных машин ведущих посев чего-либо;

3) проверка распределительных систем, например, телефонных, железнодорожных, электрических, которые требуют проверки всех своих «компонент».

Кроме того программа может использоваться в рамках учебного процесса дисциплины «Алгоритмы и структуры данных». Библиотека программы может использоваться разработчиками для создания потенциально полезных приложений.

* 1. **Краткая характеристика области применения**

«Программа поиска маршрута китайского почтальона» – программа, позволяющая во взвешенном связном неориентированном или во взвешенном сильно связном ориентированном мультиграфе без петель найти кратчайший путь, проходящий через каждое ребро (в неориентированном)/дугу (в ориентированном) мультиграфа без петель, по крайней мере, один раз, начинающийся и заканчивающийся в одной и той же вершине.

Если перейти от понятий теории графов к повседневной терминологии, то задача китайского почтальона заключается в том, чтобы пройти все улицы заданного маршрута с определенной длиной каждой дороги, входящей в маршрут, и вернуться в начальную точку, пройдя при этом как можно меньшее расстояние [10, c.219].

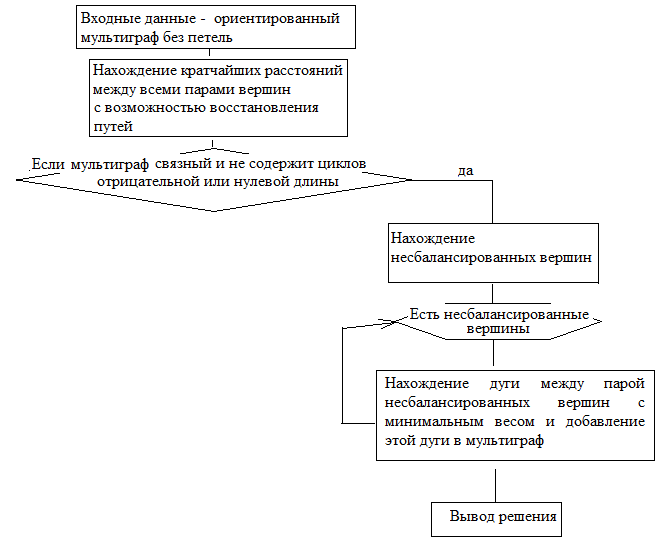
Задача китайского почтальона и ее решение имеют много потенциальных приложений:

1. доставка почты, молока и т.д.;
2. сбор мусора;
3. проверка электрических, телефонных или железнодорожных линий;
4. патрулирование улиц определенного района;
5. наилучший маршрут для движения сельскохозяйственных машин по полю при посеве чего-либо;
6. минимизация пути обхода ремонтными или проверяющими бригадами.
7. **ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ**
   1. **Постановка задачи на разработку программы**

Разрабатываемая программа должна:

1. решать задачу китайского почтальона на ориентированном мультиграфе без петель и на неориентированном мультиграфе без петель;
2. иметь отдельное поле для рисования мультиграфа и осуществлять отрисовку мультиграфа без петель и его редактирование;
3. выполнять сохранение созданного мультиграфа в файл формата .cpost [см. Приложение 2];
4. выполнять открытие ранее созданного в этой программе мультиграфа из файла формата .cpost [см. Приложение 2].
   1. **Описание алгоритма и функционирования программы**
      1. **Описание алгоритма и функционирования программы для ориентированного мультиграфа**

Алгоритм решения задачи китайского почтальона для ориентированного мультиграфа без петель приведен ниже (рис. 1):



Здесь должна быть ветка «нет»

*Рисунок 1.* Алгоритм решения задачи китайского почтальона для ориентированного мультиграфа без петель

Общая сложность алгоритма .

* + - 1. **Нахождение расстояний между всеми парами вершин алгоритмом Флойда-Уоршелла**

Алгоритм Флойда-Уоршелла – динамический алгоритм для нахождения кратчайших расстояний между всеми вершинами мультиграфа.

Для определенности положим, что вершины графа последовательно пронумерованы от 0 до \_TheNumberOfVertex-1. Алгоритм Флойда использует матрицу \_CostOfCheapestPath размера [\_TheNumberOfVertex,\_TheNumberOfVertex], в которой вычисляются длины кратчайших путей. Вначале \_CostOfCheapestPath[i, j] равен наименьшему значению из параллельных дуг между i и j для всех i не равных j.

Над матрицей \_CostOfCheapestPath выполняется \_TheNumberOfVertex итераций. После k-й итерации \_CostOfCheapestPath [i, j] содержит значение наименьшей длины путей из вершины i в вершину j, которые не проходят через вершины с номером, большим k. Другими словами, между концевыми вершинами пути i и j могут находиться только вершины, номера которых меньше или равны k. На k-й итерации для вычисления матрицы \_CostOfCheapestPath применяется следующая формула (1):

(1)

где \_CostOfCheapestPath [i,j] – матрица, хранящая информацию о минимальной стоимости пути между вершинами i и j.

Нижний индекс k обозначает значение матрицы \_CostOfCheapestPath после k-й итерации. Для вычисления \_CostOfCheapestPath [i, j] проводится сравнение величины \_CostOfCheapestPath [i, j] (т.е. стоимость пути от вершины i к вершине j без участия вершины k или другой вершины с более высоким номером) с величиной \_CostOfCheapestPath [i, k] + \_CostOfCheapestPath [k, j] (стоимость пути от вершины i до вершины k плюс стоимость пути от вершины k до вершины j). Если путь через вершину k дешевле, чем \_CostOfCheapestPath [i, j], то величина \_CostOfCheapestPath [i, j] изменяется.

В этом алгоритме также заполняется матрица \_ArcsIsDefined[i, j], которая содержит значение true или false, в зависимости от того определен ли путь между вершинами i и j.

Алгоритм Флойда-Уоршелла на языке C#:

public void FloydWarshallAlgorithmForDirectedGraph()

{

for (int k = 0; k < \_TheNumberOfVertex; k++)

{

for (int i = 0; i < \_TheNumberOfVertex; i++)

{

if (\_ArcsIsDefined[i, k])

{

for (int j = 0; j < \_TheNumberOfVertex; j++)

{

if (\_ArcsIsDefined[k, j] && (!\_ArcsIsDefined[i, j] || \_CostOfCheapestPath[i, j] > \_CostOfCheapestPath[i, k] + \_CostOfCheapestPath[k, j]))

{

\_TrainingPath[i, j] = \_TrainingPath[i, k];

\_CostOfCheapestPath[i, j] = \_CostOfCheapestPath[i, k] + \_CostOfCheapestPath[k, j];

\_ArcsIsDefined[i, j] = true;

if (i == j && \_CostOfCheapestPath[i, j] < 0)

return;

}

}

}

}

}

}

Время выполнения этого алгоритма, очевидно, имеет порядок , поскольку в ней практически нет ничего, кроме вложенных друг в друга трех циклов.

* + - 1. **Алгоритм проверки ориентированного мультиграфа на сильную связность и на наличие циклов нулевой и отрицательной длины**

При выполнении алгоритма Флойда-Уоршелла была заполнена матрица \_ArcsIsDefined, содержащая значение true или false, в зависимости от того определен ли путь между вершинами i и j. Если все значения матрицы равны true, то ориентированный мультиграф сильно связан. Если все значения матрицы \_CostOfCheapestPath больше нуля, то ориентированный мультиграф не имеет циклов отрицательной или нулевой длины.

Алгоритм проверки приведен ниже на языке C#:

public void CheckGraphForNegativeCycleAndConection()

{

for (int i = 0; i < \_TheNumberOfVertex; i++)

{

for (int j = 0; j < \_TheNumberOfVertex; j++)

{

if (!\_ArcsIsDefined[i, j])

throw new NonConnectedGraphException("Мультиграф не связан!");

}

if (\_CostOfCheapestPath[i, i] < 0)

throw new GraphHasNegativeCycleException("Негативные циклы в мультиграфе");

}

}

Время выполнения этого алгоритма, очевидно, имеет порядок .

* + - 1. **Алгоритм нахождения несбалансированных вершин ориентированного мультиграфа**

При задании мультиграфа заполняется матрица \_BalansDeltaOfVertex для каждой вершины, она содержит баланс каждой вершины: положительное, отрицательное или нулевое число. Если число равно нулю, то для данной вершины полустепень захода равна полустепени исхода – вершина сбалансированна. Если число отрицательное, то полустепень исхода больше полустепени захода, если же число положительное, то полустепень захода больше полустепени исхода. Эти значения позволяют найти вершины с положительной степенью исхода и захода. Необходимо посчитать количество значений в \_BalansDeltaOfVertex больших нуля и меньших нуля. Создать два массива \_NegativVertex и \_PositiveVertex, которые будут содержать количество элементов равное числу элементов в \_BalansDeltaOfVertex меньших и больших нуля соответственно. Затем заполнить массивы \_NegativVertex и \_PositiveVertex номерами соответствующих вершин.

На C# алгоритм выглядит следующим образом:

public void FindPositiveAndNegativeVertex()

{

int NumerOfNeg = 0, NumberOfPos = 0;

for (int i = 0; i < \_TheNumberOfVertex; i++)

{

if (\_BalansDeltaOfVertex[i] < 0)

{

NumerOfNeg++;

}

else

{

if (\_BalansDeltaOfVertex[i] > 0)

NumberOfPos++;

}

}

\_NegativVertex = new int[NumerOfNeg];

\_PositiveVertex = new int[NumberOfPos];

NumerOfNeg = NumberOfPos = 0;

for (int i = 0; i < \_TheNumberOfVertex; i++)

{

if (\_BalansDeltaOfVertex[i] < 0)

{

\_NegativVertex[NumerOfNeg++] = i;

}

else

{

if (\_BalansDeltaOfVertex[i] > 0)

{

\_PositiveVertex[NumberOfPos++] = i;

}

}

}

}

Время выполнения этого алгоритма, очевидно, имеет порядок О(\_TheNumberOfVertex).

* + - 1. **Нахождение дуги между парой несбалансированных вершин с минимальным весом и добавление этой дуги в мультиграф**

Ищем баланс каждой вершины с учетом повторяющихся дуг.

На C# это выглядит следующим образом:

public void FindPathFeasibleForDirectedGraph()

{

int[] delta = new int[\_TheNumberOfVertex];

for (int i = 0; i < \_TheNumberOfVertex; i++)

delta[i] = this.\_BalansDeltaOfVertex[i];

for (int OutVertex = 0; OutVertex < \_NegativVertex.Length; OutVertex++)

{

int i = \_NegativVertex[OutVertex];

for (int IncomeVertex = 0; IncomeVertex < \_PositiveVertex.Length; IncomeVertex++)

{

int j = \_PositiveVertex[IncomeVertex];

\_RepeatedArcs[i, j] = -delta[i] < delta[j] ? -delta[i] : delta[j];

delta[i] += \_RepeatedArcs[i, j];

delta[j] -= \_RepeatedArcs[i, j];

}

}

}

Время выполнения этого алгоритма, очевидно, имеет порядок .

Для просмотра необходимости дальнейшего поиска несбалансированных вершин и вспомогательных дуг в мультиграфе построим новый мультиграф, содержащий вспомогательные дуги и посмотрим сбалансированы ли его вершины. Если все вершины сбалансированы – то новый мультиграф является эйлеровым и больше проводить операций с ним не требуется, если же вершины не сбалансированы – то снова выполняется алгоритм для поиска дуги с наименьшей стоимостью между парой несбалансированных вершин.

public bool ImproveCancelNegCycle()

{

ClosedCPPGraph Vestigial = new ClosedCPPGraph(\_TheNumberOfVertex);

for (int u = 0; u < \_NegativVertex.Length; u++)

{

int i = \_NegativVertex[u];

for (int v = 0; v < \_PositiveVertex.Length; v++)

{

int j = \_PositiveVertex[v];

Vestigial.addArcForDirectedGraph(null, i, j, \_CostOfCheapestPath[i, j]);

if (\_RepeatedArcs[i, j] != 0)

Vestigial.addArcForDirectedGraph(null, j, i, -\_CostOfCheapestPath[i, j]);

}

}

Vestigial.FloydWarshallAlgorithmForDirectedGraph();

for (int i = 0; i < \_TheNumberOfVertex; i++)

if (Vestigial.\_CostOfCheapestPath[i, i] < 0)

{

int k = 0, u, v;

bool kunset = true;

u = i;

do

{

v = Vestigial.\_TrainingPath[u, i];

if (Vestigial.\_CostOfCheapestPath[u, v] < 0 && (kunset || k > \_RepeatedArcs[v, u]))

{

k = \_RepeatedArcs[v, u];

kunset = false;

}

} while ((u = v) != i);

u = i;

do

{

v = Vestigial.\_TrainingPath[u, i];

if (Vestigial.\_CostOfCheapestPath[u, v] < 0) \_RepeatedArcs[v, u] -= k;

else \_RepeatedArcs[u, v] += k;

} while ((u = v) != i);

return true;

}

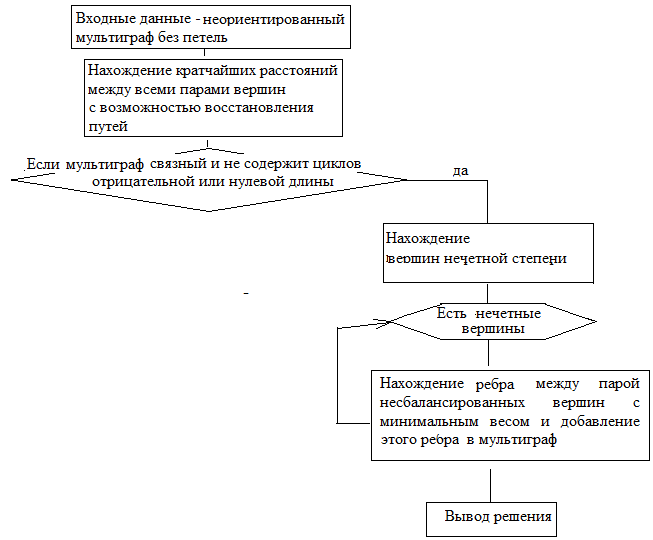
return false;

}

Время выполнения этого алгоритма, очевидно, имеет порядок О().

* + 1. **Описание алгоритма и функционирования программы для неориентированного мультиграфа**

Алгоритм решения задачи китайского почтальона для неориентированного графа приведен ниже (рис. 2):



Оформление рисунка.

Подпись – после рисунка.

Рисунок + номер, точка, затем наименование рисунка.

Ссылка на рисунок в тексте «рис. + номер»

*Рисунок 2.* Алгоритм решения задачи китайского почтальона для неориентированного графа

Общая сложность алгоритма .

* + - 1. **Нахождение расстояний между всеми парами вершин алгоритмом Флойда-Уоршелла**

Алгоритм Флойда-Уоршелла для нахождения расстояний между всеми парами вершин неориентированного мультиграфа аналогичен алгоритму нахождения расстояний между всеми парами вершин для ориентированного мультиграфа, приведенном в п. 3.3.1.1.

Временная сложность алгоритма

где \_TheNumberOfVertex – количество вершин в мультиграфе.

* + - 1. **Алгоритм проверки неориентированного мультиграфа на связность и на наличие циклов нулевой и отрицательной длины**

Нежелательно отрывать заголовки от текста.

Меню «Абзац», вкладка «Положение на странице», check-box «не отрывать от следующего»

Алгоритм проверки неориентированного мультиграфа на связность и на наличие циклов нулевой и отрицательной длины аналогичен алгоритму проверки ориентированного мультиграфа на связность и на наличие циклов нулевой длины, приведенном в п. 3.3.1.2.

Время выполнения этого алгоритма имеет порядок О()

где \_TheNumberOfVertex – количество вершин в мультиграфе.

* + - 1. **Алгоритм нахождения нечетных вершин неориентированного мультиграфа**

При создании мультиграфа для каждой вершины выполнялся подсчет степени вершины и записывался в массив \_BalansDeltaOfVertex. Вершина нечетная, если соответствующее число в массиве \_BalansDeltaOfVertex нечетное.

На языке C# нахождение нечетных вершин следующим образом:

public void FindStepOfVertex()

{

int nn = 0;

for (int i = 0; i < \_TheNumberOfVertex; i++)

if (\_BalansDeltaOfVertex[i] % 2 != 0)

nn++;

\_NegativVertex = new int[nn];

nn = 0;

for (int i = 0; i < \_TheNumberOfVertex; i++)

if (\_BalansDeltaOfVertex[i] % 2 != 0)

\_NegativVertex[nn++] = i;

}

Время выполнения этого алгоритма имеет порядок О(\_TheNumberOfVertex)

где \_TheNumberOfVertex – количество вершин в мультиграфе.

* + - 1. **Нахождение дуги между парой несбалансированных вершин с минимальным весом и добавление этой дуги в мультиграф**

Этот алгоритм также похож на алгоритм, описанный в п. 3.3.1.4.

Ищем между парой несбалансированных вершин путь минимального веса. На языке C# это выглядит следующим образом:

public void FindPathFeasibleForUndirectedGraph()

Алгоритм можно привести на псевдокоде или на языке программирования.

Если нет комментариев в коде, надо привести описание алгоритма в тексте

{

double min = double.MaxValue;

int a = 0;

int b = 1;

for (int i = 0; i < \_NegativVertex.Length; )

{

min = double.MaxValue;

for (int j = i + 1; j < \_NegativVertex.Length - 1; j++)

{

if (\_CostOfCheapestPath[\_NegativVertex[i], \_NegativVertex[j]] < min)

{

min = \_CostOfCheapestPath[\_NegativVertex[i], \_NegativVertex[j]];

a = i;

b = j;

}

}

if (a < \_NegativVertex.Length && b < \_NegativVertex.Length)

{

\_RepeatedArcs[\_NegativVertex[a], \_NegativVertex[b]]++;

}

\_BalansDeltaOfVertex[\_NegativVertex[a]]++;

\_BalansDeltaOfVertex[\_NegativVertex[b]]++;

FindStepOfVertex();

}

for (int i = 0; i < \_RepeatedArcs.GetLength(0); i++)

for (int j = 0; j < \_RepeatedArcs.GetLength(1); j++)

\_RepeatedArcs[j, i] = \_RepeatedArcs[i, j];

}

Время выполнения этого алгоритма, очевидно, имеет порядок .

где \_TheNumberOfVertex – количество вершин в мультиграфе.

Затем смотрим необходимость дальнейшего поиска путей между нечетными вершинами. Строим новый мультиграф, содержащий вспомогательные дуги и посмотрим четны ли его вершины. Если все вершины четны – то новый мультиграф является эйлеровым и больше проводить операций с ним не требуется, если же вершины не все четны – то снова выполняется алгоритм для поиска дуги с наименьшей стоимостью между парой несбалансированных вершин. Алгоритм просмотра приведен в п. 3.3.1.4.

* + 1. **Обоснование выбора алгоритма решения задачи**

При разработке программы был использован алгоритм, приведенный в [10]. Других алгоритмов на момент создания приложения найдено не было. Согласно этому источнику для решения задачи китайского почтальона необходимо найти путь между всеми парами вершин мультиграфа и найти минимальный путь между несбалансированными/нечетными вершинами данного мультиграфа.

Следующие алгоритмы позволяют найти кратчайший путь между вершинами мультиграфа:

* алгоритм Шимбела;
* алгоритм Флойда-Уоршелла;
* алгоритм Дейкстры.

Таблица 1

Сравнение алгоритмов поиска кратчайшего пути между вершинами мультиграфа

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Временная сложность | Простота реализации | Простота восстановления путей | Находит путь между всеми парами вершин |
| Алгоритм Шимбела |  | сложный | простой | да |
| Алгортм Флойда-Уоршелла |  | простой | простой  Оформление таблиц:  1) Слово «Таблица» и номер без точки в конце прижать к правому краю  2) по центру на следующей строке – наименование таблицы  3) Далее – сама таблица | да |
| Алгоритм Дейкстры |  | простой | средний | нет |

Для поиска минимального пути между несбалансированными/нечетными вершинами данного мультиграфа был использован алгоритм из [13], т.к. остальные алгоритмы наподобие венгерского имеют сложную реализацию и большую временную сложность.

* + 1. **Возможные взаимодействия программы с другими программами**

В целом программа работает самостоятельно. Однако для сохранения, редактирования записи и печати решения задачи необходим текстовый редактор, например Microsoft Word. Можно скопировать решение задачи из программы в графический редактор и выполнять необходимые действия по сохранению, редактированию и печати.

* 1. **Описание и обоснование выбора метода организации входных и выходных данных**
     1. **Описание метода организации входных и выходных данных**

Входные данные могут быть внесены пользователем вручную и быть получены из файла формата .cpost [см. Приложение 2].

Вручную пользователь может нарисовать мультиграф без петель, задать веса его дуг/ребер и получить решение в поле «Решение».

Выходные данные представляются в виде вывода решения задачи китайского почтальона в поле «Решение» и сохранения созданного мультиграфа без петель в файл формата .cpost [см. Приложение 2].

При работе с файлами формата .cpost [см. Приложение 2] используется встроенный механизм сериализации в двоичный формат и десериализации из двоичного формата.

* + 1. **Обоснования выбора метода организации входных и выходных данных**

Механизм сериализации был выбран из-за того что сериализация позволяет:

1. сохранить объект в файле и обращаться к нему в любое время;
2. воссоздать объект в его первоначальную форму.

Итогом работы пользователя является нарисованый мультиграф. Но промежуточный результат работы – это вершины, дуги/ребра, значения весов дуг/ребер, все это имеет параметры типа координат, цвета, значения и пр. Для сохранения и последующего редактирования мультиграфа создается свой формат файла .cpost [см. Приложение 2] и последовательно выгружаются данные, которые создал пользователь. Сериализиция позволяет выгружать объекты класса, вместе со всеми свойствами автоматически. Соответственно, когда пользователь захочет открыть сохраненный файл формата .cpost [см. Приложение] потребуется только лишь десериализовать файл.

Механизм сериализации и десериализации позволяет не потерять данные при сохранении и открытии файла. А также сократить количество программного кода.

* 1. **Описание и обоснование выбора состава технических и программных средств**
     1. **Состав технических и программных средств**

Для работы программы необходим следующий состав программных средств:

1) операционная система Microsoft Windows XP SP3 или более поздняя версия;

2) установленный Microsoft .NET Framework 2.0, требующий Windows Installer 3.1 или более поздняя версия;

3) Internet Explorer 6.0 или более поздняя версия.

Для работы программы необходим следующий состав технических средств:

Характеристики программных средств тоже надо указывать аккуратно. Каждую характеристику вы должны суметь обосновать.

Характеристики технических средств надо указывать аккуратно. Каждую характеристику вы должны суметь обосновать.

1) персональный компьютер, оснащенный 32-разрядным (x86) или 64-разрядным (x64) процессором Pentium с тактовой частотой 400 MГц и выше или аналогичный процессор (рекомендуется Pentium с тактовой частотой 1 ГГц и выше или аналогичный процессор);

2) 96 МБ оперативной памяти или больше (рекомендуется 256 МБ оперативной памяти или больше);

3) не менее 1,5 ГБ свободного места на жестком диске;

4) видеокарта и монитор, поддерживающие режим Super VGA с разрешением не менее чем 850x600 точек;

5) мышь Microsoft Mouse или совместимое указывающее устройство;

6) клавиатура;

7) CD-ROM привод;

8) звуковая плата;

9) динамики или наушники.

* + 1. **Обоснование выбора технических и программных средств**

При реализации программы использован тип List<T>, где T – тип элементов в списке, из пространства имен System.Collections.Generic, который впервые был представлен в платформе Microsoft .NET Framework 2.0. Остальные используемые в программе пространства имен были представлены и в более ранних версиях Microsoft .NET Framework.

Microsoft .NET Framework 2.0 в свою очередь требует [21,22]:

1. операционную систему Windows XP SP3 и выше;
2. процессор Pentium с тактовой частотой 400 MГц и выше или аналогичный процессор (рекомендуется Pentium с тактовой частотой 1 ГГц и выше или аналогичный процессор);
3. 96 МБ оперативной памяти или больше (рекомендуется 256 МБ оперативной памяти или больше);
4. установщик Windows 3.1 или более поздней версии;
5. Internet Explorer 6.0 или более поздней версии;
6. 0,28 гигабайт (ГБ) (для 32-разрядной системы) или 0,61 ГБ (для 64-разрядной системы) пространства на жестком диске и выше.

Для установки Windows XP необходимы [16]:

1. процессор Pentium с частотой 233 МГц или более быстрый (рекомендуется не менее 300 МГц);
2. не менее 64 МБ оперативной памяти (рекомендуется не менее 128 МБ);
3. не менее 1,5 ГБ свободного места на жестком диске;
4. дисковод для компакт- или DVD-дисков;
5. клавиатура, мышь Microsoft Mouse или совместимое указывающее устройство;
6. видеокарта и монитор, поддерживающие режим Super VGA с разрешением не менее чем 800x600 точек;
7. звуковая плата;
8. динамики или наушники.

Главное окно программы имеет разрешение 850x490 точек, поэтому разрешение монитора должно быть не менее чем 850x490 точек.

Здесь характеристики основаны на требованиях MS .NET Framework 2.0, это неудачная идея.

OC Windows указанных версий давно не поддерживается.

Оперативная память слишком мала, такую придется поискать

Дисководы тоже можно считать устаревшими.

Красным шрифтом выделены «нехорошие» характеристики

1. **ОЖИДАЕМЫЕ ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ**
   1. **Предполагаемая потребность**

Задача поиска маршрута китайского почтальона и ее решение довольно востребованы в экономической и информационной областях. Программу могут использовать все, кто нуждается в поиске оптимального (по времени, километражу, трудозатратам) пути, проходящего через каждую дорогу заданного маршрута хотя бы один раз. Программа может использоваться почтальонами, курьерами, людьми занимающимися доставкой молока, ремонтными и проверочными бригадами, патрульными службами, а также всеми теми, кто хочет оптимизировать свой маршрут данным способом.

* 1. **Экономические преимущества разработки по сравнению с отечественными и зарубежными образцами или аналогами**

Быстрый поиск в сети Интернет на момент создания приложения не выявил аналогов данной программы.

Данное приложение:

1. распространяется бесплатно;
2. не требует вложения денежных средств во время использования;
3. имеет неограниченный срок службы.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ГОСТ 19.101-77 Виды программ и программных документов. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
2. ГОСТ 19.102-77 Стадии разработки. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
3. ГОСТ 19.103-77 Обозначения программ и программных документов. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
4. ГОСТ 19.104-78 Основные надписи. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
5. ГОСТ 19.105-78 Общие требования к программным документам. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
6. ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
7. ГОСТ 19.404-79 Пояснительная записка. Требования к содержанию и оформлению. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
8. ГОСТ 19.603-78 Общие правила внесения изменений. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
9. ГОСТ 19.604-78 Правила внесения изменений в программные документы, выполненные печатным способом. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.

в описании книг надо указывать количество страниц. Здесь не указано, это ошибка

1. Майника Э. Алгоритмы оптимизации на сетях и графах. – М.: Мир, 1981
2. Скиена С. Алгоритмы. Руководство по разработке. – СПб.: БХВ-Петербург, 2011.
3. Mineka E. The Chinese postman problem for mixed networks [Электронный ресурс]. –U.S.A.: Management Science, 1979.
4. Harold Thimbleby. The directed Chinese Postman Problem [Электронный ресурс]. London.: University College London Interaction Centre.

Список источников лучше упорядочить по алфавиту.

1. ГОСТ 15150-69 Машины, приборы и другие технические изделия. Исполнения для различных климатических районов. Категории, условия эксплуатации, хранения и транспортирования в части воздействия климатических факторов внешней среды. – М.: Изд-во стандартов, 1997.
2. Устинов В.. Надежность оптических дисков: как их правильно хранить и использовать. //Журнал «625» №7. М.: Издательство «625», 2005.
3. Системные требования ОС Windows ХР. [Электронный ресурс]// URL: <http://support.microsoft.com/kb/314865/ru> (Дата обращения: 13.04.2014, режим доступа: свободный).
4. ГОСТ Р 7.02-2006 Консервация документов на компакт-дисках. Общие требования. – М.: ИПК Издательство стандартов, 2006.
5. ГОСТ 19.301-79 Программа и методика испытаний. Требования к содержанию и оформлению. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
6. Толковый словарь по теории графов [Электронный ресурс]// URL:<http://pco.iis.nsk.su/grapp/> (Дата обращения: 13.04.2014, режим доступа: свободный).
7. Словарь терминов теории графов [Электронный ресурс]// URL:<http://ru.wikipedia.org/wiki/Словарь_терминов_теории_графов> (Дата обращения: 13.04.2014, режим доступа: свободный).
8. Microsoft .NET Framework 2. [Электронный ресурс]// URL:<http://www.microsoft.com/ru-ru/download/details.aspx?id=1639> (Дата обращения: 13.04.2014, режим доступа: свободный).
9. Требования к системе для .NET Framework. [Электронный ресурс]// URL:<http://msdn.microsoft.com/ru-ru/library/vstudio/8z6watww(v=vs.100).aspx> (Дата обращения: 13.04.2014, режим доступа: свободный).

# ПРИЛОЖЕНИЕ 1 ТЕРМИНОЛОГИЯ

**Вершина** – базовое понятие. Точка, где могут сходиться/выходить рёбра и/или дуги.

**Ребро** – базовое понятие. Ребро соединяет две вершины графа.

**Дуга** – это ориентированное ребро.

**Граф** – базовое понятие. Включает множество вершин и множество рёбер, являющееся подмножеством декартова квадрата множества вершин (то есть каждое ребро соединяет ровно две вершины).

**Мультиграф** – граф, в котором может быть пара вершин, которая соединена более чем одним ребром (ненаправленным), либо более чем двумя дугами противоположных направлений.

**Петля** — ребро, начало и конец которого находятся в одной и той же вершине.

**Мультиграф** **без** **петель** – мльтиграф не имеющий в своем составе петель. По тексту документа под мультиграфом будем подразумеваем мультиграф без петель.

**Взвешенный граф** – граф, каждому ребру которого поставлено в соответствие некое значение.

**Орграф**, ориентированный граф G = (V,E) есть пара множеств, где V - множество вершин (узлов), E - множество дуг (ориентированных рёбер). Дуга - это упорядоченная пара вершин (v, w), где вершину v называют началом, а w - концом дуги. Можно сказать, что дуга v → w ведет от вершины v к вершине w, при этом вершина w смежная с вершиной v.

**Неориентированный граф** G = (V, Е) состоит из конечного множества вершин V и мно­жества ребер Е. В отличие от ориентированного графа, здесь каждое ребро (v,w) соот­ветствует неупорядоченной паре вершин: если (v,w) - неориентированное ребро, то (v, w) = (w, v).

**Задача китайского почтальона** - задача нахождения пути, заканчивающегося и начинающегося в одной и той же вершине, проходящего через каждое ребро (в неориентированном мультиграфе) или каждую дугу (в ориентированном мультиграфе), по крайней мере, один раз и имеющего минимальную суммарную длину.

**Связный граф** – граф, в котором все вершины связаны, т.е. существует соединяющая их цепь.

**Сильно связный граф** – ориентированный граф, в котором все вершины сильно связаны, т.е. существует путь из первой во вторую и из второй в первую и т.д.

**Орграф называется слабым** (или слабосвязным), если связным графом является его неориентированный дубликат.

**Путь** - последовательность рёбер (в неориентированном графе) и/или дуг (в ориентированном графе), такая, что конец одной дуги (ребра) является началом другой дуги (ребра).

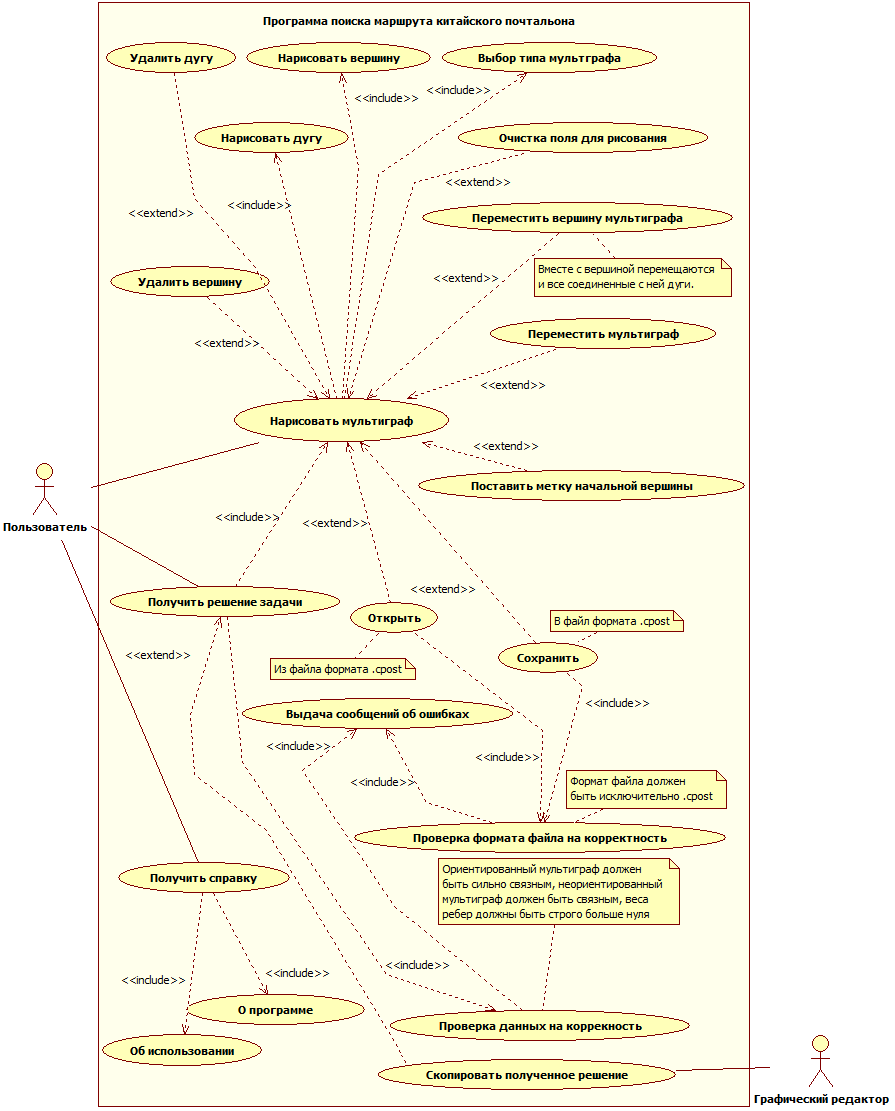
# ПРИЛОЖЕНИЕ 2 ОПИСАНИЕ ФОРМАТА .cpost

Файл формата .cpost содержит информацию о мультиграфе, созданном в программе GraphFormerForCPP, в сереализованном виде.

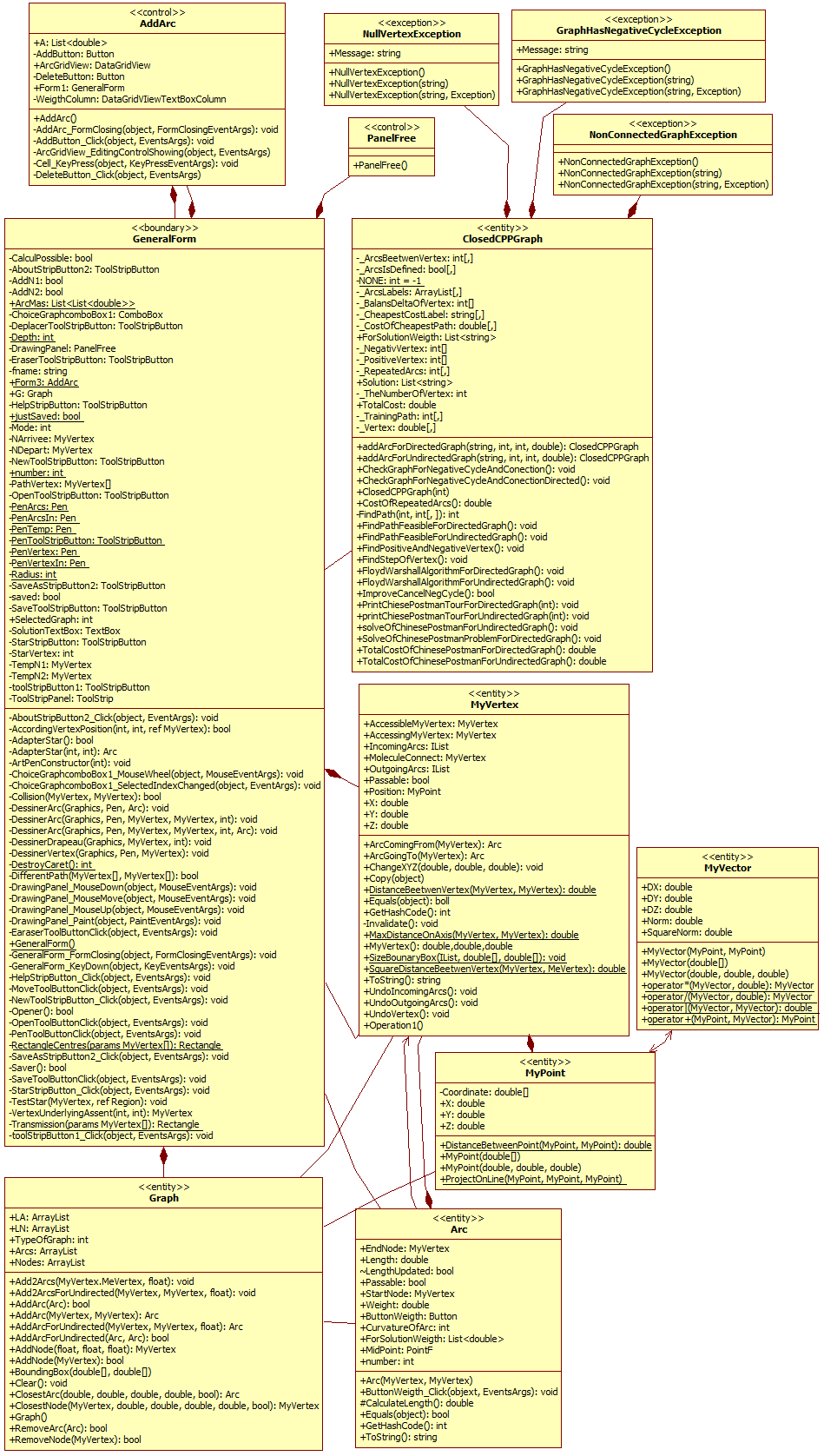
Открыть файл формата .cpost можно программой GraphFormerForCPP.

Программа GraphFormerForCPP при этом осуществляет корректное открытие только файлов формата .cpost, созданных или измененных в этой программе.

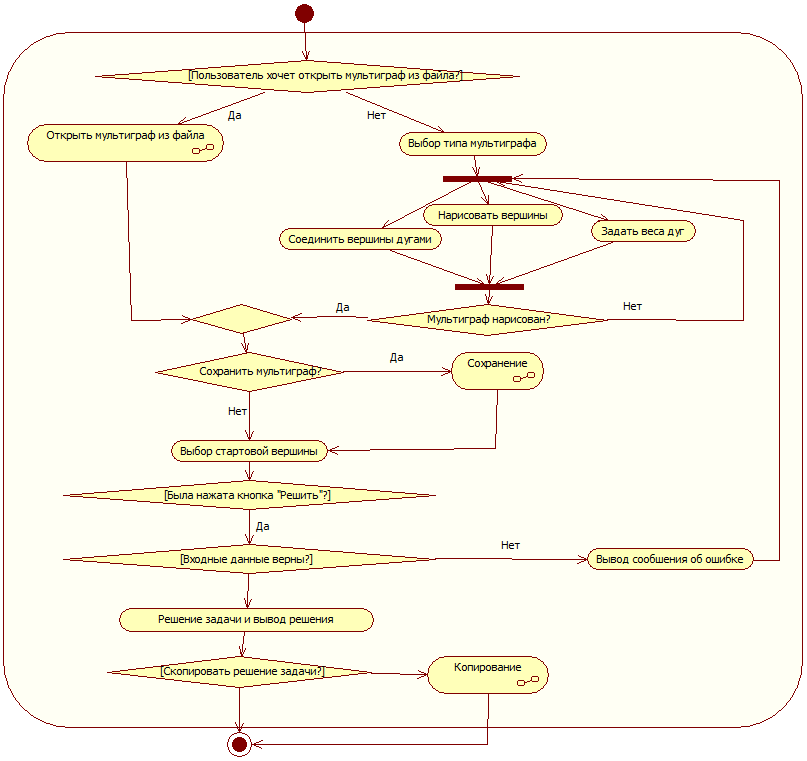
# ПРИЛОЖЕНИЕ 3 ДИАГРАММА ВАРИАНТОВ ИСПОЛЬЗОВАНИЯ

****

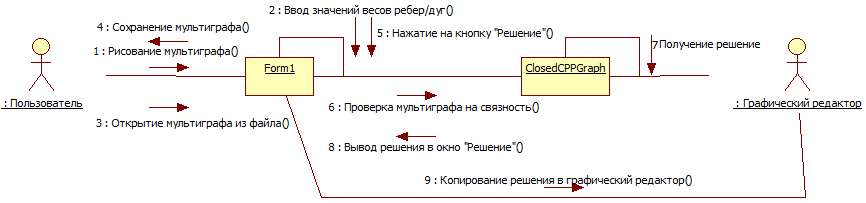
**ПРИЛОЖЕНИЕ 4  
ДИАГРАММА КЛАССОВ**

****

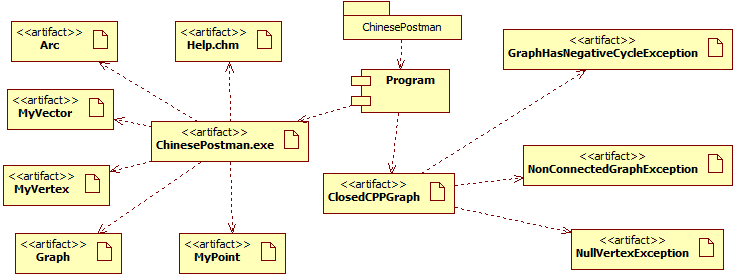
**ПРИЛОЖЕНИЕ 5  
ДИАГРАММА ДЕЯТЕЛЬНОСТИ**

****

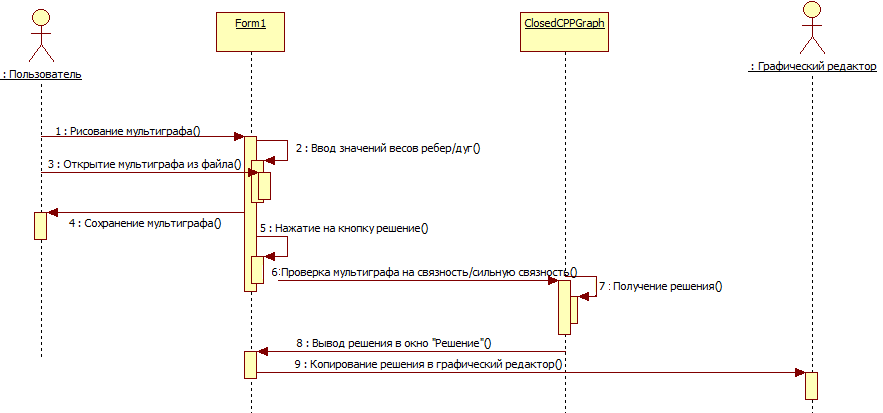
**ПРИЛОЖЕНИЕ 6  
ДИАГРАММА КООПЕРАЦИИ**

****

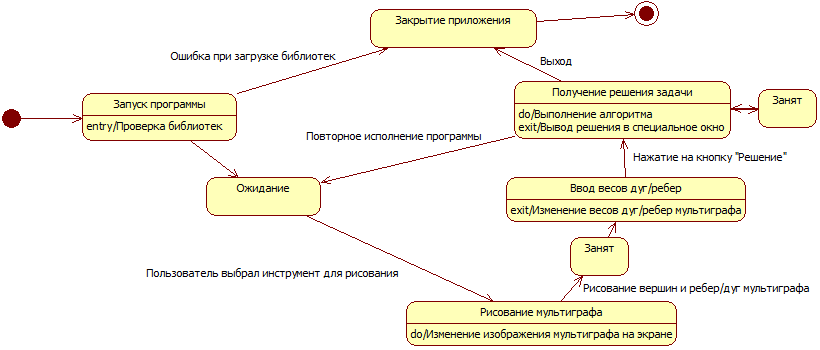
**ПРИЛОЖЕНИЕ 7  
ДИАГРАММА КОМПОНЕНТОВ**

****

**ПРИЛОЖЕНИЕ 8  
ДИАГРАММА ПОСЛЕДОВАТЕЛЬНОСТЕЙ**

****

**ПРИЛОЖЕНИЕ 9  
ДИАГРАММА СОСТОЯНИЙ**

****

**ПРИЛОЖЕНИЕ 10  
ОПИСАНИЕ И ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ КЛАССОВ**

Таблица 10.1

Описание и функциональное назначение классов

| Класс | Назначение |
| --- | --- |
| ClosedCPPGraph | Класс, содержащий структуру ориентированного и неориентированного графа, а также содержащий поля и методы для решения задачи китайского почтальона на ориентированном или неориентированном мультиграфе без петель. |
| GraphHasNegativeCycleException | Класс исключения, которое возникает если мультиграф имеет негативный цикл. |
| NonConnectedGraphException | Класс исключения, которое возникает если мультиграф не связан/несильно связан. |
| NullVertexException | Класс исключения, которое возникает если мультиграф имеет нуль вершин. |
| Arc | Класс представляющий структуру дуги/ребра для отрисовки. Содержит поля и методы необходимые для рисования дуги/ребра. Производный от MyVector. |
| AddArc | Представляет окно для добавления/удаления дуг/ребер и их весов. Содержит обработчики пользовательских событий. |
| Form1 | Представляет главное окно приложения. Содержит обработчики пользовательских событий, в нем так же содержатся и взаимодействуют объекты других классов и поля и методы необходимые для взаимодействия этих объектов. Производный от PanelFree. |
| Graph | Класс, представляющий структуру графа для отрисовки, и содержащий необходимые поля и методы для создания и редактирования графа. |
| MyPoint | Класс точки, являющийся базовым для MyVertex. Содержит поля и методы необходимые для расчёта расстояний между точками. |
| MyVector | Класс представляющий вектор, являющийся базовый классом для Arc. Содержит поля, методы, переопределенные операторы, необходимые для проведения операций с векторами. |
| MyVertex | Класс представляющий структуру вершины для отрисовки. Производный от MyPoint. Содержит поля и методы, необходимы для рисования вершины. |
| PanelFree | Класс, представляющий форму с заданным стилем и поведением элементов управления. Базовый для Form1. |

**ПРИЛОЖЕНИЕ 11  
ОПИСАНИЕ И ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ ПОЛЕЙ МЕТОДОВ И СВОЙСТВ**

Таблица 11.1

Описание полей методов и свойств класса ClosedCPPGraph.cs

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Поля** | | | | | | | | |
| Имя | | Модификатор доступа | | Тип | | | Назначение | |
| \_TheNumberOfVertex | | private | | int | | | Количество вершин мультиграфа. | |
| \_BalansDeltaOfVertex | | private | | int[] | | | Массив, содержащий баланс вершин мультиграфа. | |
| \_NegativVertex | | public | | int[]; | | | Массив, содержащий вершины с отрицательной степенью, их числовое значение. | |
| \_PositiveVertex | | public | | int[] | | | Массив, содержащий вершины с положительной степенью, их числовое значение. | |
| \_ArcsBeetwenVertex | | private | | int[,] | | | Матрица дуг между вершинами, содержащая информацию о количестве параллельных дуг. | |
| \_ArcsLabels | | private | | ArrayList[,] | | | Список названий дуг. | |
| \_RepeatedArcs | | private | | int[,] | | | Массив, содержащий информацию о повторяющихся дугах/ребрах в маршруте. | |
| \_CostOfCheapestPath | | private | | double[,] | | | Массив, содержащий информацию о минимальной стоимости путей между заданными вершинами. | |
| \_CheapestCostLabel | | private | | String[,] | | | Массив, содержащий информацию о названиях дуг/ребер с минимальной стоимости путей между заданными вершинам. | |
| \_ArcsIsDefined | | private | | bool[,] | | | Массив содержащий информацию о возможности пути между заданными вершинами.. | |
| \_TrainingPath | | private | | int[,] | | | Остовное дерево мультиграфа | |
| \_TotalCost | | private | | double | | | Общая стоимость маршрута китайского почтальона. | |
| \_Vertex; | | private | | double[,] | | | Массив значений весов между вершинами. | |
| NONE | | private | | int | | | Константа со значением -1. | |
| \_Solution | | private | | List<string> | | | Список для хранения решения задачи почтальона. | |
| \_ForSolutionWeigth | | private | | List<string> | | | Содержит информацию о весах ребер/дуг. | |
| **Методы** | | | | | | | | |
| Имя | Модификатор доступа | | Тип | | Аргументы | | | Назначение |
| ClosedCPPGraph | public | | конструктор | | int VertexNumber | | | Конструктор мультигрфа с заданным количеством вершин. |
| Продолжение таблицы 11.1 | | | | | | | | |
| addArcForDirectedGraph | public | | ClosedCPPGraph | | String Names, int OutgoingVertex, int IncomingVertex, double PathCost | | | Метод для добавления дуг в ориентированный мультиграф между заданными вершинами, нахождения дуг между заданными вершинами с наименьшей стоимостью и подсчета баланса вершин между заданными вершинами. |
| addArcForUndirectedGraph | public | | ClosedCPPGraph | | String Names, int OutgoingVertex, int IncomingVertex, double PathCost | | | Метод для добавления ребер в неориентированный мультиграф между заданными вершинами, нахождения ребер между заданными вершинами с наименьшей стоимостью и подсчета баланса вершин между заданными вершинами. |
| FloydWarshallAlgorithmForDirectedGraph | public | | void | | – | | | Метод для нахождения кратчайших путей между вершинами ориентированного мультиграфа алгоритмом Флойда-Варшела и восстановления этих путей. |
| CheckGraphForNegativeCycleAndConection | public | | void | | – | | | Метод для проверки мультиграфа на связность и наличие отрицательных дуг. |
| FindStepOfVertex | public | | void | | – | | | Метод для нахождения нечетных вершин мультиграфа. |
| FindPositiveAndNegativeVertex | public | | void | | – | | | Метод для поиска несбалансированных (негативных и позитивных) вершин. |
| FindPathFeasibleForDirectedGraph | public | | void | | – | | | Метод ищет повторяющиеся дуги. |
| FindPathFeasibleForUndirectedGraph | public | | void | | – | | | Метод ищет повторяющиеся ребра. |
| ImproveCancelNegCycle | public | | bool | | – | | | Цикл "улучшения" остаточного мультиграфа. Удаляет появившиеся негативные циклы. |
| TotalCostOfChinesePostmanForDirectedGraph | public | | double | | – | | | Метод подсчета стоимости обхода ориентированного мультиграфа для китайского почтальона |
| CheckGraphForNegativeCycleAndConectionDirected | public | | void | | – | | | Метод для проверки мультиграфа на сильную связность и наличие отрицательных дуг. |
|  |  | |  | |  | | |  |
| Продолжение таблицы 11.1 | | | | | | | | |
| TotalCostOfChinesePostmanForUndirectedGraph | public | | double | | – | | | Метод подсчета стоимости обхода неориентированного мультиграфа для китайского почтальона |
| CostOfRepeatedArcs | public | | double | | – | | | Метод подсчитывает стоимость повторяющих в маршруте дуг. |
| FloydWarshallAlgorithmForUndirectedGraph | public | | void | | – | | | Метод для нахождения кратчайших путей между вершинами неориентированного мультиграфа алгоритмом Флойда-Варшела и восстановления этих путей. |
| FindPath | private | | int | | – | | | Метод для нахождения пути из заданной вершины в другие. |
| printChinesePostmanTourForUndirectedGraph | public | | void | | int startVertex | | | "Печать" (сохранение) решения задачи китайского почтальона для неориентированного мультиграфа. |
| PrintChiesePostmanTourForDirectedGraph | public | | void | | int startVertex | | | "Печать" (сохранение) решения задачи китайского почтальона для ориентированного мультиграфа. |
| SolveOfChinesePostmanProblemForDirectedGraph | public | | void | | – | | | Метод, объединяющий "вспомогательные методы" для решения задачи китайского почтальона на ориентированном мультиграфе. |
| solveOfChinesePostmanForUndirectedGraph | public | | void | | – | | | Метод, объединяющий "вспомогательные методы" для решения задачи китайского почтальона на неориентированном мультиграфе. |
| **Свойства** | | | | | | | | |
| Имя | Модификатор доступа | | Тип | | | Доступ | | Назначение |
| ForSolutionWeigth | public | | List<string> | | | get | | Свойство возвращающее список весов ребер |
| Solution | public | | List<string> | | | get | | Свойство возвращающее значение списка с решением задачи. |
| TotalCost | public | | double | | | get | | Стоимость заданных путей между вершинами |

Таблица 11.2

Описание полей методов и свойств класса GraphHasNegativeCycleException.cs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Методы** | | | | |
| Имя | Модификатор доступа | Тип | Аргументы | Назначение |
| GraphHasNegativeCycleException | public | конструктор | – | Конструктор исключения без параметров. |
| GraphHasNegativeCycleException | public | конструктор | string message | Конструктор исключения с сообщением от программиста |
| GraphHasNegativeCycleException | public | конструктор | string message, Exception innerException | Инициализирует исключения с сообщением об ошибке и ссылкой на внутреннее исключение, которое стало причиной данного исключения. |
| **Свойства** | | | | |
| Имя | Модификатор доступа | Тип | Доступ | Назначение |
| Message | public | string | get | Свойство для получения сообщения об исключении. Переопределено. |

Таблица 11.3

Описание полей методов и свойств класса NonConnectedGraphException.cs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Методы** | | | | |
| Имя | Модификатор доступа | Тип | Аргументы | Назначение |
| GraphHasNegativeCycleException | public | конструктор | – | Конструктор исключения без параметров. |
| GraphHasNegativeCycleException | public | конструктор | string message | Конструктор исключения с сообщением от программиста |
| GraphHasNegativeCycleException | public | конструктор | string message, Exception innerException | Инициализирует исключения с сообщением об ошибке и ссылкой на внутреннее исключение, которое стало причиной данного исключения. |

Таблица 11.4

Описание полей методов и свойств класса NullVertexException.cs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Методы** | | | | |
| Имя | Модификатор доступа | Тип | Аргументы | Назначение |
| GraphHasNegativeCycleException | public | конструктор | – | Конструктор исключения без параметров. |
| GraphHasNegativeCycleException | public | конструктор | string message | Конструктор исключения с сообщением от программиста |
|  |  |  |  |  |
| Продолжение таблицы 11.4 | | | | |
| GraphHasNegativeCycleException | public | конструктор | string message, Exception innerException | Инициализирует исключения с сообщением об ошибке и ссылкой на внутреннее исключение, которое стало причиной данного исключения. |
| **Свойства** | | | | |
| Имя | Модификатор доступа | Тип | Доступ | Назначение |
| Message | public | string | get | Свойство для получения сообщения об исключении. Переопределено. |

Таблица 11.5

Описание полей методов и свойств класса Arc.cs

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Поля** | | | | | | | |
| Имя | | Модификатор доступа | | Тип | | Назначение | |
| \_StartNode | | private | | MyVertex | | Вершина, из которой выходит дуга/ребро. | |
| \_EndNode | | private | | MyVertex | | Вершина, в которую входит дуга/ребро. | |
| \_Weight | | private | | double | | Вес дуги/ребра, по умолчанию 1. | |
| \_Passable; | | private | | bool | | Метка "возможности" дуги. | |
| \_Length | | private | | double | | Длина отрисованной дуги. | |
| \_LengthUpdated | | private | | bool | | Метка изменения длины отрисованной дуги. | |
| ButtonWeigth | | public | | Button | | Кнопка для вызова формы для изменения количества и весов дуг. | |
| MidPoint | | public | | PointF | | Серединная точка дуги. | |
| CurvatureOfArc | | public | | int | | Кривизна дуги. | |
| ForSolutionWeigth | | public | | List<double> | | Список весов дуг/ребер для отрисованной дуги/ребра | |
| number | | public | | int | | Номер дуги в мультиграфе | |
| **Методы** | | | | | | | |
| Имя | Модификатор доступа | | Тип | | Аргументы | | Назначение |
| Arc | public | | конструктор | | MyVertex Start, MyVertex End | | Конструктор дуги для отрисовки. |
| ButtonWeigth\_Click | public | | void | | object sender, EventArgs e | | Метод обработки события нажатия на кнопку, находящуюся посредине дуги/ребра. |
|  |  | |  | |  | |  |
| Продолжение таблицы 11.5 | | | | | | | |
| CalculateLength | protected | | double | | – | | Метод возвращает посчитанную длину дуги/ребра между стартовой и конечной точкой. Может быть переопределен. |
| ToString | public | | string | | – | | Метод возвращает строку с описание дуги. Переопределен. |
| Equals | public | | bool | | object O | | Метод для сравнения двух вершин. Переопределен. |
| GetHashCode | public | | int | | – | | Переопределение Object.GetHashCode. |
| **Свойства** | | | | | | | |
| Имя | Модификатор доступа | | Тип | | Доступ | | Назначение |
| StartNode | public | | MyVertex | | get/set | | Свойство, возвращающее и задающее значение вершины, из которой выходит дуга/ребро. |
| EndNode | public | | MyVertex | | get/set | | Свойство, возвращающее и задающее значение вершины, в которую входит дуга/ребро. |
| Weight | public | | double | | get/set | | Свойство, возвращающее и задающее значение веса дуги/ребра. |
| Passable | public | | bool | | get/set | | Свойство, задающее и возвращающее значение возможности отрисовки дуги/ребра. |
| LengthUpdated | internal | | bool | | get/set | | Свойство, возвращающее и задающее значение метки изменения длины отрисованной дуги/ребра. |
| Length | public | | double | | get/set | | Свойство возвращающее длину отрисованной дуги/ребра. |

Таблица 11.6

Описание полей методов и свойств класса AddArc.cs

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Поля** | | | | | | | |
| Имя | | Модификатор доступа | | Тип | | Назначение | |
| A | | public | | List<double> | | Список весов дуг/ребер. | |
| Form1 | | public | | GeneralForm | | Ссылка на главную форму | |
| **Методы** | | | | | | | |
| Имя | Модификатор доступа | | Тип | | Аргументы | | Назначение |
| AddArc | public | | конструктор | | – | | Конструктор формы для добавления весов дуг/ребер без параметров. |
| AddButton\_Click | private | | void | | object sender, EventArgs e | | Метод для события, которое возникает при нажатии кнопки "Добавить". |
| DeleteButton\_Click | private | | void | | object sender, EventArgs e | | Метод для события, которое возникает при нажатии кнопки "Удалить". |
| ArcGridView\_EditingControlShowing | private | | void | | object sender, DataGridViewEditingControlShowingEventArgs e | | Метод для события, которое возникает при изменении ячейки. |
| Cell\_KeyPress | private | | void | | object sender, KeyPressEventArgs press | | Метод, который позволяет обрабатывать нажатие клавиш. Допускаются только десятичные цифры. |
| AddArc\_FormClosing | private | | void | | object sender, FormClosingEventArgs e | | Метод для события закрытия формы. |

Таблица 11.7

Описание полей методов и свойств класса Form1.cs

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Поля** | | | | | | | | |
| Имя | | Модификатор доступа | | Тип | | | Назначение | |
| number | | public | | int | | | Номер текущей дуги в графе. | |
| SelectedGraph | | public | | int | | | Тип выбранного графа. 0-ориентированный, 1-неориентированный. | |
| justSaved | | public | | bool | | | Хранит информацию о сохранении файла. | |
| Radius | | private | | int | | | Радиус вершины. | |
| Depth | | private | | int | | | Ширина линии отрисовки. | |
| PenVertex | | private | | Pen | | | Карандаш для вершины. | |
| PenArcs | | private | | Pen | | | Карандаш для дуги. | |
| PenVertexIn | | private | | Pen | | | Карандаш для отрисовки следа вершины. | |
| PenArcsIn | | private | | Pen | | | Карандаш для отрисовки следа дуги | |
| PenTemp | | private | | Pen | | | Временный карандаш. | |
| Mode | | private | | int | | | Выбранный тип инструмента. 0-карандаш, 1-ластик, 2-перемещение (деплейсер). | |
| G | | public | | Graph | | | Объект графа, предназначен для отрисовки. | |
| TempN1 | | private | | MyVertex | | | Временная вершина из которой выходит дуга/ребро. | |
| TempN2 | | private | | MyVertex | | | Временная вершина в которую входит дуга/ребро. | |
| Продолжение таблицы 11.7 | | | | | | | | |
| AddN1 | | private | | bool | | | Метка о возможности добавления вершины, из которой выходит дуга/ребро.. | |
| AddN2 | | private | | bool | | | Метка о возможности добавления вершины, в которую входит дуга/ребро. | |
| NDepart | | private | | MyVertex | | | Исходящая вершина. | |
| NArrivee | | private | | MyVertex | | | Входящая вершина. | |
| PathVertex | | private | | MyVertex[] | | | Путь из вершин. | |
| TempP | | private | | Point | | | Временная точка. | |
| StarVertex | | private | | int | | | Номер начальной вершины. | |
| saved | | private | | bool | | | Метка о сохранении файла, был ли когда-то сохранен. | |
| fname | | private | | string | | | Имя файла для сохранения. | |
| Form3 | | public | | AddArc | | | Ссылка на форму для добавления весов дуг | |
| ArcMas | | public | | List<List<double>> | | | Хранит список весов текущей дуги | |
| **Методы** | | | | | | | | |
| Имя | Модификатор доступа | | Тип | | | Аргументы | | Назначение |
| DestroyCaret | public | | int | | | – | | Метод убирает каретку из поля для вывода решения задачи. |
| GeneralForm | public | | конструктор | | | – | | Конструктор главного окна программы. |
| ChoiceGraphcomboBox1\_MouseWheel | private | | void | | | object sender, MouseEventArgs e | | Метод для события, при прокрутке колесика для ComboBox в котором находится тип выбранного графа. Не изменяет тип. |
| GeneralForm\_KeyDown | private | | void | | | object sender, KeyEventArgs e | | Метод обработки события нажатия клавиш. Обрабатывает «горячие клавиши». |
| ChoiceGraphcomboBox1\_SelectedIndexChanged | private | | void | | | object sender, EventArgs e | | Метод для обработки события при изменения ComboBox в котором находится тип выбранного графа. |
| GeneralForm | private | | конструктор | | | – | | Статический конструктор для инциализации карандашей. |
| PenToolButtonClick | private | | void | | | object sender, EventArgs e | | Метод для события нажатия на карандаш. Делает кнопку «Карандаш» нажатой. |
| EaraserToolButtonClick | private | | void | | | object sender, EventArgs e | | Метод для события нажатия на ластик. Делает кнопку «Ластик» нажатой. |
|  |  | |  | | |  | |  |
| Продолжение таблицы 11.7 | | | | | | | | |
| MoveToolButtonClick | private | | void | | | object sender, EventArgs e | | Метод для события нажатия на перемещение. Делает кнопку «Деплейсер» нажатой. |
| Transmission | private | | Rectangle | | | params MyVertex[] EncompassVertex | | Метод для увеличения области с захваченными вершинами. |
| RectangleCentres | private | | Rectangle | | | params MyVertex[] EncompassVertex | | Метод для вычисления прямоугольника с заданными захваченными вершинами. |
| AccordingVertexPosition | private | | bool | | | int X, int Y, ref MyVertex N | | Метод просмотра согласованности вершины и точки. |
| VertexUnderlyingAssent | private | | MyVertex | | | int X, int Y | | Метод для вычисления близкой вершины. |
| AdapterStar | private | | Arc | | | int X, int Y | | AdapterStar |
| Collision | private | | bool | | | MyVertex N1, MyVertex N2 | | Метод для вычисления возможности столкновения(слияния) точек. |
| DrawingPanel\_Paint | private | | void | | | object sender, PaintEventArgs e | | Метод для события, происходящего при рисовании на панели. Рисование вершин и дуг или их удаления |
| DrawingPanel\_MouseDown | private | | void | | | object sender, MouseEventArgs e | | Метод для события, возникающего в момент нажатия на кнопку мыши. (Удаление, рисование и перемещение дуг и вершин). |
| DessinerVertex | private | | void | | | Graphics Grfx, Pen P, MyVertex N | | Метод для отрисовки вершины. |
| DessinerArc | private | | void | | | Graphics Grfx, Pen P, Arc A | | Метод для отрисовки дуги (вспомогательный). |
| DessinerArc | private | | void | | | Graphics Grfx, Pen P, MyVertex N1, MyVertex N2, int choiceofcurv | | Метод для отрисовки следа дуги. |
|  | | | | | | | | |
| Продолжение таблицы 11.7 | | | | | | | | |
| DessinerArc | private | | void | | | Graphics Grfx, Pen P, MyVertex N1, MyVertex N2, int choiceofcurve, Arc AA | | Метод для отрисовки дуги. |
| DrawingPanel\_MouseMove | private | | void | | | object sender, MouseEventArgs e | | Метод для события, которое возникает при передвижении мыши. (рисование, удаление, перемещение). |
| ArtPenConstructor | private | | void | | | int index | | Метод, конструирующий наконечник для карандаша дуги, в зависимости от типа мультиграфа. |
| DrawingPanel\_MouseUp | private | | void | | | object sender, MouseEventArgs e | | Метод для события, которое возникает в момент отпускании кнопки мыши. |
| AdapterStar | private | | bool | | | – | | Можно ли поставить звезду - метку начальной вершины. |
| DifferentPath | private | | bool | | | MyVertex[] C1, MyVertex[] C2 | | Метод для определения возможности различных путей |
| TestStar | private | | void | | | MyVertex N, ref Region ZoneInvalide | | Метод для звезды - метка начальной вершины, нужен для поиска ближайшей вершины. |
| SaveToolButtonClick | private | | void | | | object sender, EventArgs e | | Метод для события, возникающего при нажатии на кнопку "Сохранить". Проверяет на пустоту и сохраненность графа. И сохраняет при необходимости. |
| OpenToolButtonClick | private | | void | | | object sender, EventArgs e | | Метод для события, возникающего при нажатии на кнопку "Открыть". Открытие файла. |
| NewToolStripButton\_Click | private | | void | | | object sender, EventArgs e | | Метод для события, возникающего при нажатии на кнопку "Новый". Очищает поле, предлагает сохранить граф. |
| Opener | private | | bool | | | – | | Метод для проверки возможности открытия файла. |
| Saver | private | | bool | | | – | | Метод для проверки возможности сохранения файла |
| Продолжение таблицы 11.7 | | | | | | | | |
| toolStripButton1\_Click | private | | void | | object sender, EventArgs e | | | Метод для события, возникающего при нажатии на кнопку "Решение". Ищет решение задачи китайского почтальона. Выводит результат в поле «Решение» или сообщает о том, что граф был задан некорректно. |
| GeneralForm\_FormClosing | private | | void | | object sender, FormClosingEventArgs e | | | Метод для события, возникающего при закрытии формы. Сохранение графа. |
| AboutStripButton2\_Click | private | | void | | object sender, EventArgs e | | | Метод для события, возникающего при нажатии на кнопку "О программе. Информация о программе. |
| StarStripButton\_Click | private | | void | | object sender, EventArgs e | | | Метод для события нажатия на звезду. Делает кнопку «Звездочка (пометить вершину)» нажатой. |
| DessinerDrapeau | private | | void | | Graphics Grfx, MyVertex N, int Numero | | | Метод для рисования звезды – метки начальной вершины. |
| HelpStripButton\_Click | private | | void | | object sender, EventArgs e | | | Метод обработки события нажатия на кнопку «Справка». Открывает справку программы. |
| SaveAsStripButton2\_Click | private | | void | | object sender, EventArgs e | | | Метод обработки события нажатия на кнопку «Сохранить как». Сохраняет в новый файл мультиграф. |
| **Свойства** | | | | | | | | |
| Имя | Модификатор доступа | | Тип | | | Доступ | | Назначение |
| CalculPossible | private | | bool | | | get | | Свойства для получения метки о возможности входящей и исходящей вершины. |

Таблица 11.8

Описание полей методов и свойств класса Graph.cs

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Поля** | | | | | | | |
| Имя | | Модификатор доступа | | Тип | | Назначение | |
| TypeOfGraph | | public | | int | | Тип графа, 0-ориентированный, 1-неориентированный. | |
| LN | | public | | ArrayList | | Список вершин. | |
| LA | | public | | ArrayList | | Список дуг/ребер. | |
| Продолжение таблицы 11.8 | | | | | | | |
| **Методы** | | | | | | | |
| Имя | Модификатор доступа | | Тип | | Аргументы | | Назначение |
| Graph | public | | конструктор | | – | | Конструктор для мультиграфа. |
| Clear | public | | void | | – | | Метод для очистки мультиграфа от ребер/дуг и вершин. |
| AddNode | public | | bool | | MyVertex NewNode | | Проверка возможности добавления вершины в муьтиграф. |
| AddNode | public | | MyVertex | | float x, float y, float z | | Метод для добавления вершины в граф для отрисовки. |
| AddArc | public | | bool | | Arc NewArc | | Проверка возможности добавления дуги/ребра в мультиграф. |
| AddArc | public | | Arc | | MyVertex StartNode, MyVertex EndNode, float Weight | | Метод для создания дуги между двумя вершинами. |
| Add2Arcs | public | | void | | MyVertex Node1, MyVertex Node2, float Weight | | Метод для добавления двух дуг в мультиграф. |
| AddArcsForUndirected | public | | Arc | | MyVertex Node1, MyVertex Node2, float Weight | | Добавление ребра в неориентированном мультиграфе. |
| AddArcForUndirected | public | | bool | | Arc NewArc, Arc NewArcRet | | Метод для проверки возможности добавления дуги в неориентированный мультиграф. |
| RemoveNode | public | | bool | | MyVertex NodeToRemove | | Метод для проверки возможности удаления вершины из графа. |
| RemoveArc | public | | bool | | Arc ArcToRemove | | Метод проверки возможности для удаления дуги/ребра из графа. |
| BoundingBox | public | | void | | out double[] MinPoint, out double[] MaxPoint | | Метод определяет ограничительную рамку всего графа. |
|  |  | |  | |  | |  |
| Продолжение таблицы 11.8 | | | | | | | |
| ClosestNode | public | | MyVertex | | double PtX, double PtY, double PtZ, out double Distance, bool IgnorePassableProperty | | Метод для нахождения ближайшей вершины к заданной точке. |
| ClosestArc | public | | Arc | | double PtX, double PtY, double PtZ, out double Distance, bool IgnorePassableProperty | | Метод для нахождения ближайшей дуги к заданной точке |
| **Свойства** | | | | | | | |
| Имя | Модификатор доступа | | Тип | | Доступ | | Назначение |
| Nodes | public | | ArrayList | | get | | Свойство возвращает список вершин мультиграфа. |
| Arcs | public | | ArrayList | | get | | Свойство возвращает список дуг/ребер мультиграфа. |

**ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Лист регистрации изменений | | | | | | | | | |
| Номера листов (страниц) | | | | | Всего листов (страниц в докум.) | № документа | Входящий № сопроводительного докум. и дата | Подп. | Дата |
| Изм. | Измененных | Замененных | Новых | Аннулированх |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |