**АННОТАЦИЯ**

В данном программном документе приведена пояснительная записка к программе «Агрегатор кулинарных рецептов на Android».

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:

1. постановка задачи на разработку программы;
2. описание алгоритма и функционирования программы с обоснованием выбора схемы алгоритма решения задачи и возможные взаимодействия программы с другими программами;
3. описание и обоснование выбора метода организации входных и выходных данных;
4. 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].

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 – «Агрегатор кулинарных рецептов на Android».

Наименование программы на английском языке – «Aggregator of Recipes for Android»

Краткое наименование программы – «На плите».

* 1. Документы, на основании которых ведется разработка

Разработка ведется на основании приказа Национального исследовательского университета "Высшая школа экономики" № 2.3-02/2004-04 от 20.04.2020.

1. НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ
2. 1. Назначение программы
      1. Функциональное назначение

Функциональным назначением программы является поиск кулинарных рецептов с различных интернет-сайтов с помощью встроенных фильтров, разделов или с помощью названия рецепта, а также отображение их в более подробном формате.

К функциональным возможностям серверной части относится сбор и хранение информации о кулинарных рецептах, а также взаимодействием с мобильным приложением.

* + 1. Эксплуатационное назначение

Решение поиска кулинарных рецептов на Android является востребованным на сегодняшний день, так как с каждым годом увеличивается рост посещения интернет-сайтов с мобильных устройств, а, зачастую, кулинарные сайты не поддерживают мобильную версию, что приводит к некомфортной работе с материалами сайта.

Основная цель программы – предоставлять пользователю кулинарные рецепты по его запросу, а также облегчить работу с кулинарными рецептами на операционной системе Android.

Программа предназначена для пользователей на операционной системе Android.

* 1. Краткая характеристика области применения

«Агрегатор кулинарных рецептов на Android» – программа, позволяющая находить и отображать кулинарные рецепты, взятые с различных кулинарных сайтов или порталов, а это в свою очередь позволит пользователям экономить своё время на поиске и просмотре кулинарных рецептов.

1. ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ
2. 1. Постановка задачи на разработку программы
      1. Задачи работы (Android-клиент)
3. Реализовать возможность просмотра списка кулинарных рецептов;
4. Реализовать постепенную загрузку кулинарных рецептов для экономии трафика и меньшей нагрузки на мобильное устройство;
5. Реализовать отображение индикатора процесса загрузки данных с сервера;
6. Реализовать фильтрацию кулинарных рецептов:
7. Популярные рецепты;
8. Новые рецепты;
9. Случайные рецепты;
10. Реализовать отображение кулинарных рецептов по разделам;
11. Реализовать просмотр расширенной информации о рецепте;
12. Реализовать просмотр списка ингредиентов и списка шагов приготовления рецепта;
13. Реализовать просмотр видеоматериала, приложенного к рецепту;
14. Реализовать возможность добавления рецепта в раздел «Избранное»;
15. Реализовать возможность удаление рецепта из раздела «Избранное»;
16. Реализовать возможность добавление ингредиента в раздел «Корзина продуктов»;
17. Реализовать возможность удаление ингредиента из раздела «Корзина продуктов»;
18. Реализовать возможность перехода на веб-страницу источника рецепта;
19. Реализовать возможность поделиться информацией о рецепте в сторонних приложениях;
    * 1. Задачи работы (серверная часть)
20. Реализовать сбор списка рецептов с кулинарных сайтов, а также сбор информации о конкретном рецепте по запросу пользователя:
21. Реализовать получение данных для каждого сайта с помощью HTTP GET или HTTP POST запросов;
22. Реализовать сбор информации с помощью «CSS Selectors»;
23. Реализовать временное кэширование рецептов в базе данных MySQL;
24. Выдавать информацию о рецепте при получении HTTP-GET запроса в формате JSON;
    1. Описание алгоритма и функционирования программы
       1. Описание построения Android приложения

В Android приложении задумано 7 активностей (см. терминологию) и 3 фрагмента (см. терминологию)

1. MainActivity (Рис. 1). Активность, формирующая интерфейс для отображения списка рецептов, где каждый рецепт подкреплён названием рецепта, изображением, а также названием источника. Является самой первой активностью, которая загружается при запуске приложения. В этой активности есть одно поле для поиска кулинарного рецепта по названию, а также кнопка для запуска поиска, выпадающий список для выбора фильтрации (Рис. 2), кнопка для открытия бокового меню слева (Рис. 3).
2. RecipeActivity. Активность, формирующая интерфейс для отображения информации о рецепте. Имеет кнопку «назад», расположенную в левом верхнем углу, которая возвращает пользователя к предыдущей активности, кнопку «поделиться», кнопку для перехода на сайт источника, а также плавающую кнопку для добавления рецепта в раздел «Избранное» или удаление рецепта из раздела «Избранное».

Фрагменты:

* 1. RecipeDescriptionFragment (Рис. 4). Фрагмент, который содержит краткое описание рецепта и какую-либо информацию о нём, а также видеоматериал при его наличии.
  2. RecipeIngredientsFragment (Рис. 5). Фрагмент, который содержит список ингредиентов. Каждый ингредиент подкреплён названием и количеством, а также элементом checkbox для добавления ингредиента в раздел «корзина продуктов» или для удаления ингредиента из раздела «корзина продуктов».
  3. RecipeStepsFragment (Рис. 6). Фрагмент, который содержит в себе список шагов для приготовления рецепта, где каждый элемент списка содержит в себе описание и изображение при его наличии.

1. PlatesActivity (Рис. 7). Активность, формирующая интерфейс для отображения списка разделов, где каждый элемент списка – изображение и название поверх этого изображения. Содержит в себе кнопку, позволяющую вернуться к предыдущей активности.
2. SectionRecipesActivity (Рис. 8). Активность, формирующая интерфейс для отображения списка рецептов, где каждый рецепт подкреплён названием рецепта, изображением, а также названием источника. Содержит в себе название раздела, расположенное на вернем части активности, а также кнопки, позволяющей вернуться к предыдущей активности.
3. SavedIngredientsActivity (Рис. 9). Активность, формирующая интерфейс для отображения списка сохранённых ингредиентов (наз. «Корзина продуктов»), каждый элемент списка содержит в себе название ингредиента, его количества, название рецепта, которому он принадлежит, а также кнопку для удаления ингредиента из данного раздела. Содержит в себе кнопку, позволяющую вернуться к предыдущей активности.
4. SavedRecipesActivity (Рис. 10). Активность, формирующая интерфейс для отображения списка сохранённых рецептов (наз. «Избранное»). Элемент списка содержит в себе название рецепта, изображение, а также название источника. Активность также содержит элемент SearchView, который позволяет фильтровать список по введённому в него названию рецепта, а также кнопку «Назад», чтобы пользователь мог вернуться к предыдущей активности.
5. InfoActivity. Активность, содержащая в себе информацию о приложении, а также кнопку «Назад», чтобы пользователь мог вернуться к предыдущей активности.

Изображение выглядит как снимок экрана, еда

Автоматически созданное описание

Рис. 1-3: MainActivity.



Рис. 4-6: RecipeActivity.

RecipeDescriptionFragment

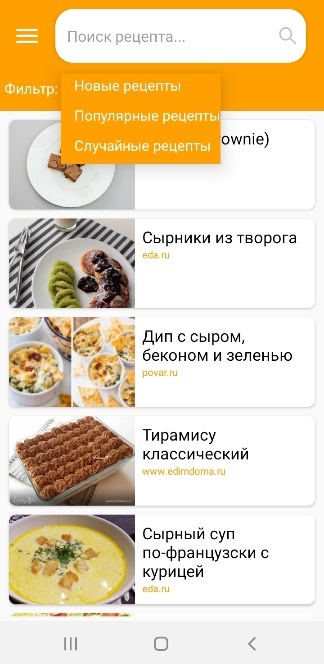


Рис. 2: Выезжающий список.

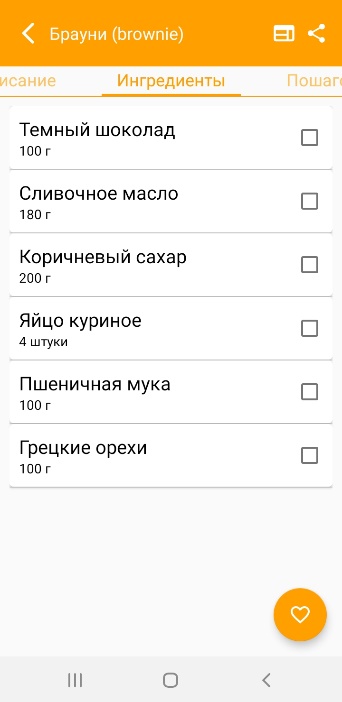


Рис. 5: RecipeIngredientsFragment

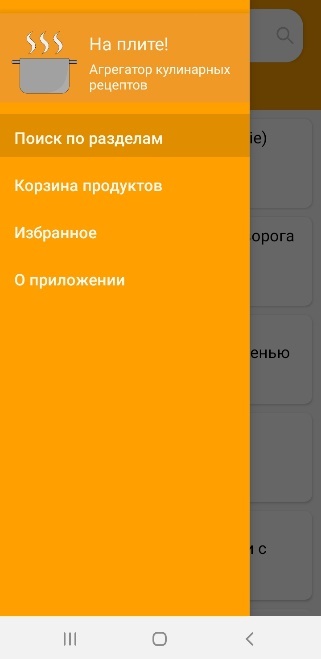


Рис. 3: Боковое меню

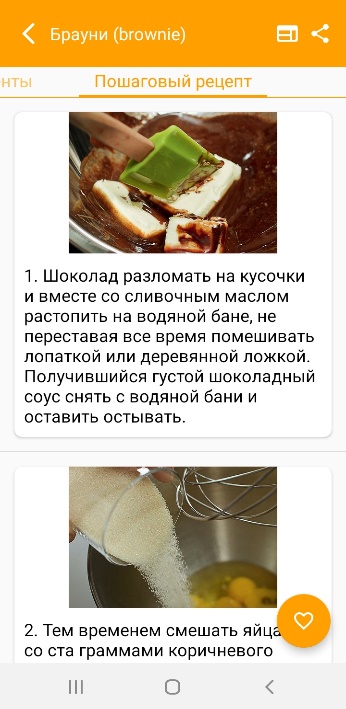


Рис. 6: RecipeStepsFragment

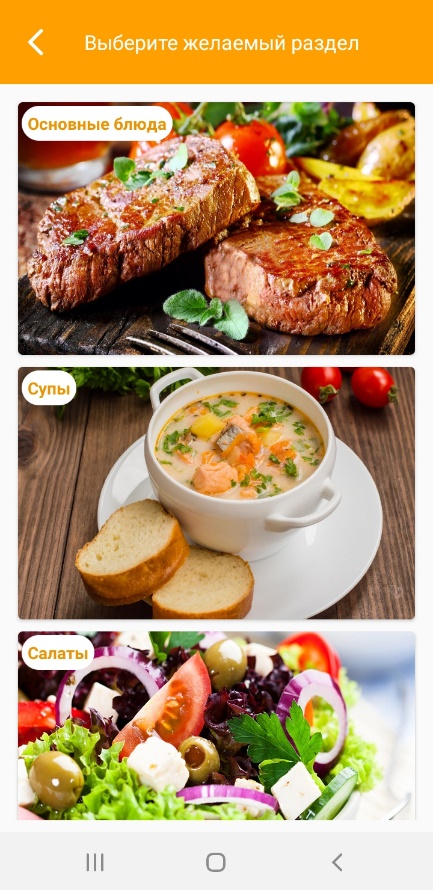


Рис. 7: PlatesActivity.

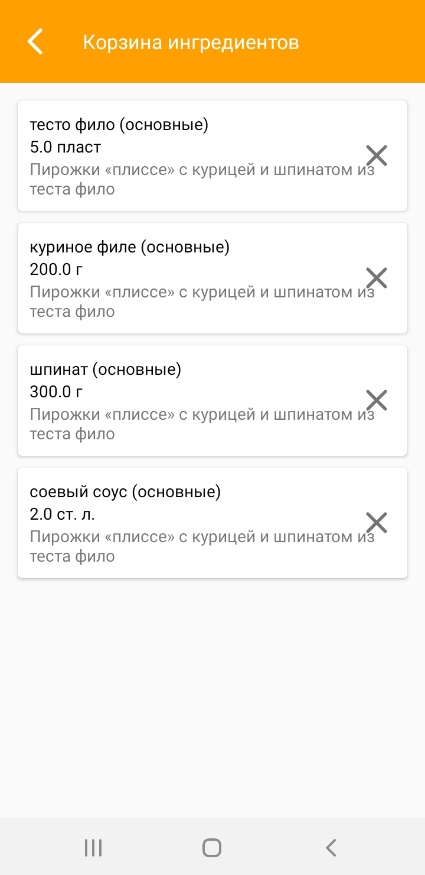


Рис. 9: SavedIngredientsActivity

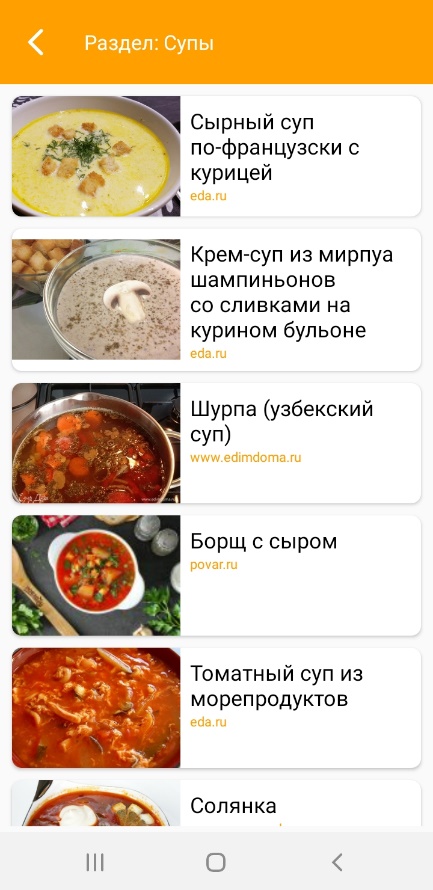


Рис. 8: SectionRecipesActivity.

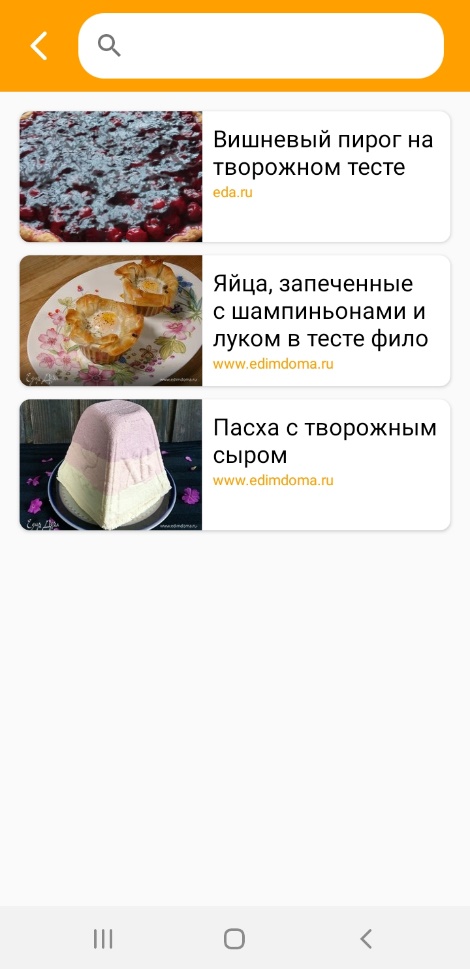


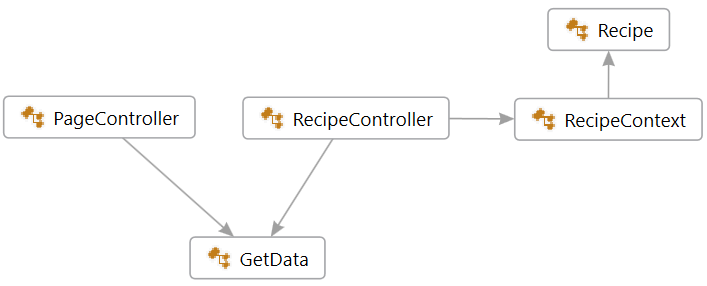
Рис. 10: SavedRecipesActivity

* + 1. Описание построения серверной части
       1. Структура серверной части

Структура серверной части содержит 2 контроллера:

1. PageController – контроллер, отвечающий за обработку входящих HTTP-GET запросов для получения списка рецептов в формате JSON.
2. RecipeController – контроллер, отвечающих за обработку входящих HTTP-GET запросов для получения информации о рецепте в формате JSON.

А также класс RecipeContext, наследующий класс DbContext, предоставляемый библиотекой Entity Framework Core для связки базы данных MySql и ASP NET Core.

Класс Recipe описывает структуру кэшируемого объекта в базе данных.

* + - 1. Контроллеры

**PageController:** Данный контроллер принимает информацию о разделе, номере страницы (необязательный параметр), а также названии рецепта (необязательный параметр):

Параметр section (раздел) поддерживает следующие значения:

1. Popular – популярные рецепты;
2. New – новые рецепты;
3. Random – случайные рецепты;
4. Recipe – поиск по названию;
5. Горячее;
6. Супы;
7. Салаты;
8. Выпечка;
9. Десерты;
10. Закуски;
11. Соусы;

Параметр в виде номера страницы – положительное целочисленное значение, является необязательным параметром для ввода. Значение по умолчанию – 1.

Если был выбран раздел Recipe, то ещё одним параметром следует указать параметр recipeName, который принимает название рецепта, по которому следует начать поиск.

**RecipeController:** Данный контроллер принимает информацию в виде одного параметре – url - адреса кулинарного рецепта.

* + - 1. Логирование информации

Каждый из контроллеров производит логирование данных как в консольное приложение, так и в текстовые документы, фильтруя их по уровню:

1. Debug
2. Information
3. Warning

Так, в файл «nlog-all-yyyy-mm-dd.log» поступают все данные, в файл «warning.log» поступает информация уровня Warning, а в файл «info.log» информация уровня Information.

В консольное приложению поступает информация уровня Information или Warning.

Пример логирования из класса RecipeController представлен на языке C#:

\_logger.LogDebug($"Рецепт существует в БД. Id = {recipeDb.Id}, Date = {recipeDb.Date}");

// Если рецепт нужно обновить:

if ((startTime - recipeDb.Date).TotalHours > diffHours)

{

\_logger.LogDebug($"Требуется обновление рецепта.");

try

{

recipe = await GetData.GetRecipe(url);

recipeDb.RecipeFull = recipe;

recipeDb.Date = startTime;

db.SaveChanges();

\_logger.LogInformation($"Статус: Ok.");

LogTime(startTime);

return Ok(recipe);

}

catch(Exception e)

{

\_logger.LogWarning(e, $"Произошла ошибка при парсинге рецепта! Выдан рецепт из БД.");

\_logger.LogInformation($"Статус: Ok.");

LogTime(startTime);

return Ok(recipeDb.RecipeFull);

}

}

* + - 1. Кэширование данных.

Кэширование данных, а в частности информации о полном рецепте происходит каждые 24 часа, это сделано для того, чтобы при многочисленных запросах от пользователей не тратить ресурсы сервера, а также чтобы обновлять информацию о новых рецептах, которые не всегда после публикации имеют совершенный вид.

Пример представлен на языке C#:

using(RecipeContext db = new RecipeContext())

{

// Если в БД есть рецепт:

if((recipeDb = db.Recipes.FirstOrDefault(x => x.Url == url)) != null)

{

…

// Если рецепт нужно обновить:

if ((startTime - recipeDb.Date).TotalHours > diffHours)

{

// Происходит обновление существующего в БД рецепта и возвращается новый.

…

}

// Если рецепт есть в БД и обновлять его ненужно:

else

{

// Возвращается рецепт из БД.

…

}

}

// Если в БД нет рецепта:

else

{

// Получаем рецепт с помощью алгоритма парсинга.

\_logger.LogDebug($"Рецепта не существует в БД.");

try

{

…

}

catch (Exception e)

{

// Если происходит ошибка – уведомляет пользователя об этом.

…

}

}

}

* + - 1. Описание построения парсера страниц со списком рецептов

Для парсера веб-страниц со списком рецептов предусмотрены следующие классы: ParserPage, HtmlLoader, PovarPageParser, EdaPageParser, PovarenokPageParser, EdimDomaPageParser, EdaPageSettings, EdimDomaPageSettings, PovarPageSettings, PovarenokPageSettings.

1. **ParserPage –** класс, который направляет полученные от пользователя параметры на получение страниц с рецептами в класс HtmlLoader, а после направляет его результат в соответствующий парсер.
2. **HtmlLoader –** класс, который собирает внутри себя правильную ссылку на страницу с рецептами в соответствии с заданными параметрами и на выходе выдаёт полученный с помощью HTTP-GET или HTTP-POST запросами код страницы кулинарного сайта.
3. **PovarPageParser,** унаследованный от **IParserPage**<RecipeShort[]>. Отвечает за преобразование исходного кода страницы сайта povar.ru в коллекцию типа RecipeShort[]. При помощи библиотеки AngleSharp проводит поиск информации по нужным тегам и формирует из них объект.

На языке C# это выглядит следующим образом:

public RecipeShort[] Parse(IHtmlDocument document, IParserPageSettings settings)

{

var recipesBody = document.QuerySelectorAll("div.recipe\_list > div.recipe");

double indexStartPopularity = settings.IndexPopularity;

return (from recipe in recipesBody

let anyBody = recipe.QuerySelector("h3 > a")

let title = anyBody.TextContent

let url = "https://povar.ru" + anyBody.Attributes[0].Value

let pictureBody = recipe.QuerySelector("img")

let imageUrl = pictureBody.Attributes[0].Value

let indexPopularity = indexStartPopularity -= settings.IndexStep

select new RecipeShort(title, new Image(imageUrl), url, indexPopularity)).ToArray();

}

1. **EdaPageParser**, **EdimDomaParser**, **PovarenokPageParser** функционал идентичен с **PovarPageParser**, только действует на сайте «eda.ru», «edimdoma.ru», «povarenok.ru» соответственно.
2. Классы **EdaPageSettings**. **EdimDomaPageSettings**, **PovarPageSettings**, **PovarenokPageSettings** наследуются от **IParserPageSettings** и задают каждому сайту индивидуальные настройки, используемые при формировании ссылки.
   * + 1. Описание построения парсера страниц с рецептом.

Необходимые классы: HtmlLoader, ParserRecipe, PovarenokRecipeParser, EdaRecipeParser, EdimDomaRecipeParser, PovarRecipeParser, PovarenokRecipeSettings, PovarRecipeSettings, EdaRecipeSettings, EdimDomaRecipeSettings.

1. **ParserRecipe** - класс, который направляет полученные от пользователя параметры на получение страниц с рецептами в класс **HtmlLoader**, а после направляет его результат в соответствующий парсер.
2. **HtmlLoader –** класс, который собирает внутри себя правильную ссылку на страницу с рецептами в соответствии с заданными параметрами и на выходе выдаёт полученный с помощью HTTP-GET или HTTP-POST запросами код страницы кулинарного сайта.
3. **PovarRecipeParser,** унаследованный от **IParserRecipe<RecipeFull>**. Отвечает за преобразование исходного кода страницы сайта povar.ru с рецептом в объект типа RecipeFull. При помощи библиотеки AngleSharp проводит поиск информации по нужным тегам и формирует из них объект

На языке C# часть алгоритма выглядит следующим образом:

var recipeBody = document.QuerySelector("div.cont\_area");

if (recipeBody is null)

return new RecipeFull();

Title = recipeBody.QuerySelector("h1.detailed[itemprop='name']").TextContent;

TitleImage = new Image(recipeBody.QuerySelector("div.bigImgBox")?

.QuerySelector("a").FirstElementChild.Attributes[0].Value);

Description = String.Empty;

foreach (var textLine in recipeBody.QuerySelectorAll("span.detailed\_full").Select(x => x.TextContent))

Description += textLine.Trim() + Environment.NewLine;

Description.Replace("\t", "").Replace(" ", "");

var ingredientsBody = recipeBody.QuerySelector("ul.detailed\_ingredients").QuerySelectorAll("li");

int countIngredients = ingredientsBody.Length;

Ingredient[] ingredients = new Ingredient[countIngredients];

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

{

string title = ingredientsBody[i].Attributes[1].Value;

string unit = ingredientsBody[i].TextContent

.Replace(title, string.Empty)

.Remove(0, 35);

ingredients[i] = new Ingredient(title, unit, Title);

}

Ingredients = ingredients;

1. **EdaRecipeParser**, **EdimRecipeParser**, **PovarenokRecipeParser** функционал идентичен с **PovarRecipeParser**, только действует на сайте «eda.ru», «edimdoma.ru», «povarenok.ru» соответственно.
2. Классы **EdaRecipeSettings**. **EdimDomaRecipeSettings**, **PovarRecipeSettings**, **PovarenokRecipeSettings** наследуются от **IParserRecipeSettings** и задают каждому сайту индивидуальные настройки, используемые при формировании ссылки.
   1. Описание и обоснование выбора метода организации входных и выходных данных
      1. Описание метода организации входных и выходных данных

Входными и выходными данными в Android-приложении является интерфейс.

Входными данными серверной части является параметры, содержащиеся в теле HTTP-GET запроса, а выходными данными является документ формата JSON.

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

Формат вывода данных на серверной части был выбран JSON, а не его похожий аналог – XML, по ряду следующих причин:

1. JSON поддерживает: строки, числа, объекты, коллекции.
2. Размер документа формата JSON значительно меньше, а значит повышается скорость передачи данных.
   1. Описание и обоснование выбора состава технических и программных средств
      1. Состав технических и программных средств

Технические требования сервера:

1. Операционная система: Windows 7 SP1 и выше/Mac OS X 10.12+ x64/Ubuntu 16.04+
2. Наличие Microsoft .NET Core SDK 3.1.
3. Наличие MySQL Server 8.0 и выше.
4. Наличие веб-сервера.

Технические требования Android-приложения:

1. Операционная система мобильного устройства должна бывать Android версии 6.0 «Marshmallow» и выше.
2. Постоянное подключение к сети интернет;
3. Для использования кнопок требуется сенсорный экран;
4. Память устройства должна быть не менее 80 Мб (рекомендуется более 120 Мб).
   * 1. Обоснование выбора технических и программных средств

Серверная часть:

Чтобы поддерживать высокую производительность и стабильность серверной части – используется последняя версия Net Core, которая имеет системные требования:

1. Операционная система: Windows 7 SP1 и выше/Mac OS X 10.12+ x64/Ubuntu 16.04+

Наличие MySQL Server необходимо, чтобы создавать и работать с базой данных. При этом выбор MySQL аргументирован так:

1. Поддержка SQL
2. Поддержка многих операционных систем**.**
3. **Высокая скорость и производительность.**

Наличие веб-сервера необходимо, чтобы ретранслировать запросы пользователей из внешней сети на сервер.

Android приложение:

Приложение разрабатывалось под минимальную версию операционной системы Android версии 6.0 «Marshmallow», потому что согласно официальному отчёту компании Google от 10 апреля 2020 года устройств с операционной системой 6.0 и выше – 84.9% от общего количества.

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

Программу может использовать любой человек, желающий проверить знания других людей.

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

Существует как отечественные, так и зарубежные аналоги, однако данное приложение имеет следующие преимущества:

1. Распространяется бесплатно;
2. Не требует вложения денежных средств во время использования;
3. Не имеет рекламных баннеров;
4. Имеет неограниченный срок службы;
5. СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ
6. ГОСТ 19.101-77 Виды программ и программных документов. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
7. ГОСТ 19.102-77 Стадии разработки. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
8. ГОСТ 19.103-77 Обозначения программ и программных документов. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
9. ГОСТ 19.104-78 Основные надписи. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
10. ГОСТ 19.105-78 Общие требования к программным документам. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
11. ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
12. ГОСТ 19.404-79 Пояснительная записка. Требования к содержанию и оформлению. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
13. ГОСТ 19.603-78 Общие правила внесения изменений. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
14. ГОСТ 19.604-78 Правила внесения изменений в программные документы, выполненные печатным способом. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
15. Серия учебников по работе с ASP.NET Core MVC и EF Core. [Электронный ресурс]// URL: <https://docs.microsoft.com/ru-ru/aspnet/core/data/ef-mvc/?view=aspnetcore-3.1> (Дата обращения: многократно c 01.03.2020 по 11.05.2020)
16. Руководство по ASP.NET Core 3. [Электронный ресурс]// URL: <https://metanit.com/sharp/aspnet5/> (Дата обращения: многократно c 01.03.2020 по 11.05.2020)
17. Руководство по Entity Framework Core [Электронный ресурс]// URL: <https://metanit.com/sharp/entityframeworkcore/> (Дата обращения: многократно c 01.03.2020 по 11.05.2020)
18. Build anything on Android [Электронный ресурс]// URL: <https://developer.android.com/> (Дата обращения: многократно c 01.03.2020 по 11.05.2020)

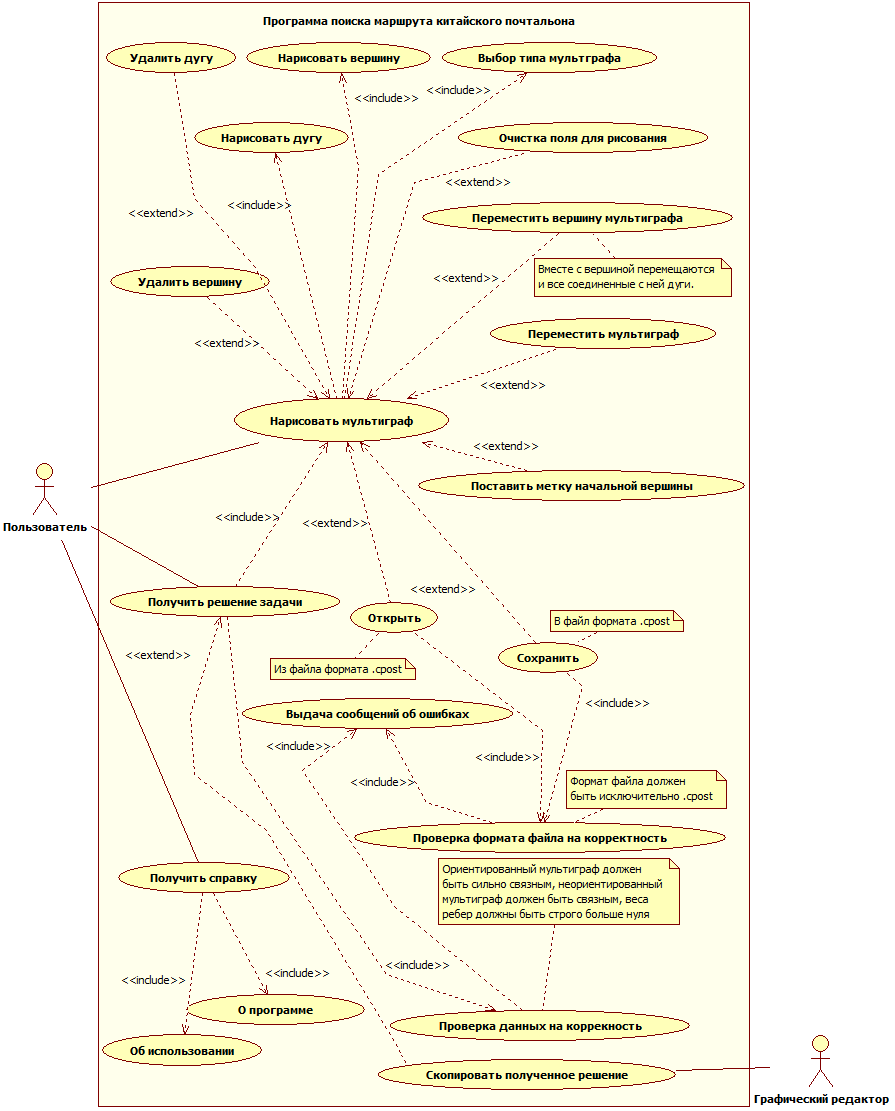
# ПРИЛОЖЕНИЕ 1 ТЕРМИНОЛОГИЯ

1. **Контроллер (англ. Controller)** – интерпретирует действия пользователя, оповещая модель о необходимости изменения
2. **Кэш (англ. *cache*)** – промежуточный буфер с быстрым доступом к нему, содержащий информацию, которая может быть запрошена с наибольшей вероятностью.
3. **Обратный прокси-сервер (англ. *reverse proxy*)** — тип проекси-сервера, который ретранслирует запросы клиентов из внешней сети на один или несколько серверов, логически расположенных во внутренней сети.
4. **Лог (*протокол*, *журнал*, *лог, англ.* *log*)** — файл с записями о событиях в хронологическом порядке, простейшее средство обеспечения журналирования.
5. **Логирование** – процесс журналирования.
6. **Активность (Activity)** – это компонент приложения, который выдает экран, и с которым пользователи могут взаимодействовать для выполнения каких-либо действий.
7. **Фрагмент (Fragment)** – Фрагмент представляет поведение или часть пользовательского интерфейса в операции (класс Activity). В одной активности может быть несколько фрагментов.
8. **RecyclerView** – Элемент визуальной группы для отображения списков объектов, переиспользующий место на экране устройства. То есть хранится столько объектов, сколько может уместиться на экране, а при прокрутке элемента, существующие объекты перерисовываются, а не продолжаются хранится в памяти.
9. **SearchView** – элемент визуальной группы представляет собой изображение лупы, которое разворачивается в текстовое поле при нажатии. Используется для поиска, например, в базе данных.
10. **База данных (БД)** - совокупность данных, хранимых в соответствии со схемой данных, манипулирование которыми выполняют в соответствии с правилами средств моделирования данных.
11. **JSON (**англ. *JavaScript Object Notation*)– Текстовый формат обмена данными.

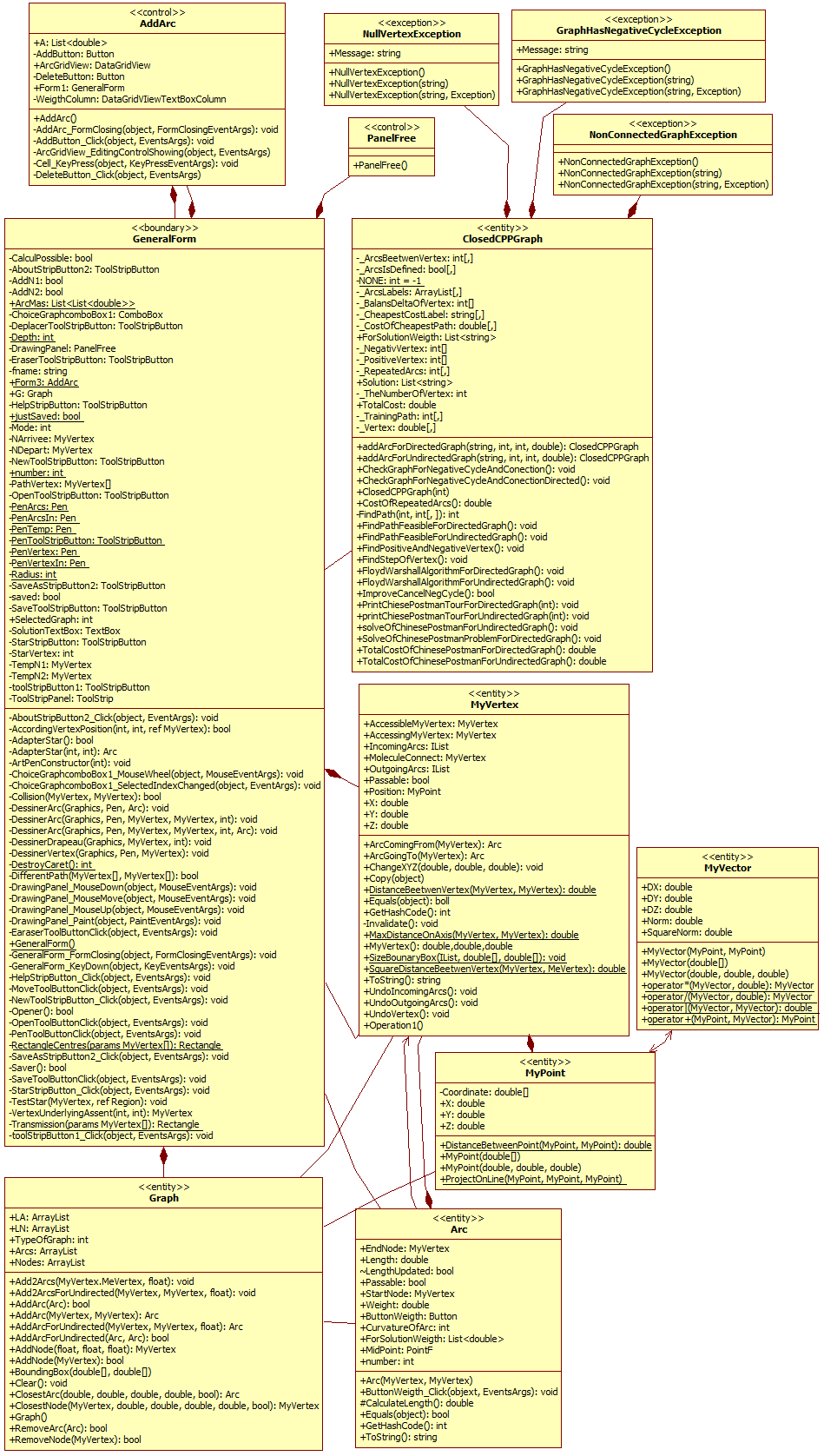
# 

# 

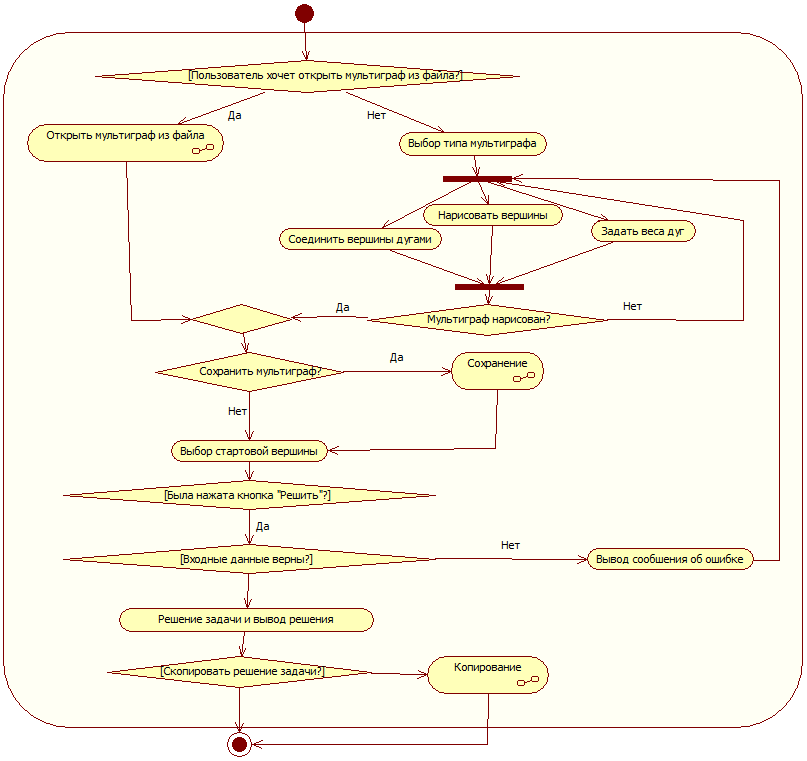
# ПРИЛОЖЕНИЕ 3 ДИАГРАММА ВАРИАНТОВ ИСПОЛЬЗОВАНИЯ

****

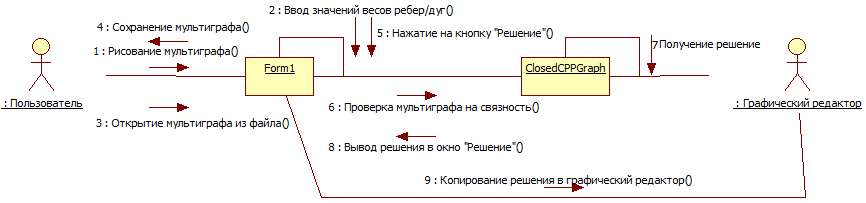
**ПРИЛОЖЕНИЕ 4  
ДИАГРАММА КЛАССОВ**

****

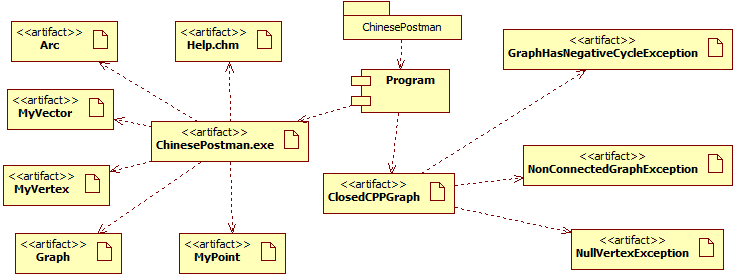
**ПРИЛОЖЕНИЕ 5  
ДИАГРАММА ДЕЯТЕЛЬНОСТИ**

****

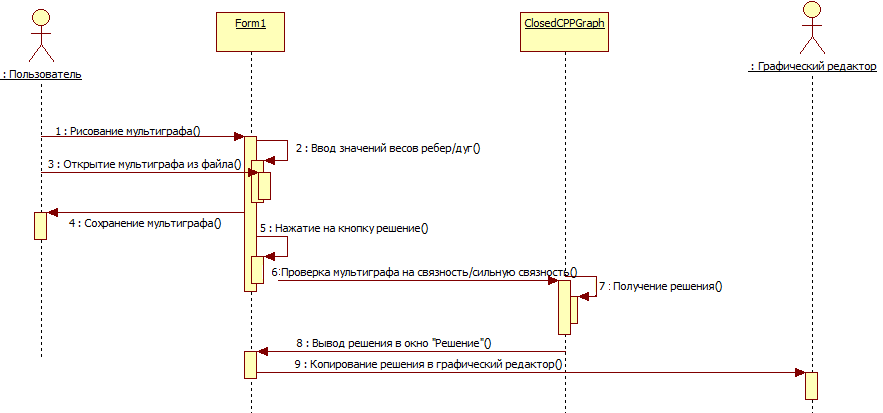
**ПРИЛОЖЕНИЕ 6  
ДИАГРАММА КООПЕРАЦИИ**

****

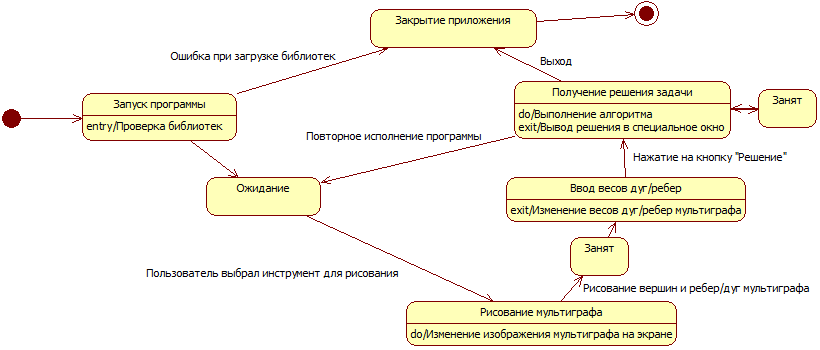
**ПРИЛОЖЕНИЕ 7  
ДИАГРАММА КОМПОНЕНТОВ**

****

**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 |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |