МИНИСТЕРСТВО ОБРАЗОВАНИЯ РФ  
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
КАФЕДРА САПР

Курсовая работа  
по дисциплине “Информационное обеспечение САПР”

на тему «Проектирование базы данных сайта оказания строительных услуг»

Направление подготовки – 09.03.01 Информатика и вычислительная техника Профиль подготовки – «Системы автоматизированного проектирования»

Работу выполнили  
студенты группы 17ВВ3:  
Горшенин Лев   
Тютеньков Константин  
Шагалин Константин  
Работу приняла:  
К.т.н , доцент кафедры САПР Глотова Т.В.

Пенза 2020

# Оглавление

[Введение 5](#_Toc59616681)

[1. Анализ ТЗ 6](#_Toc59616682)

[2. Анализ методов и алгоритмов 7](#_Toc59616683)

[2.1 Постановка задачи 7](#_Toc59616684)

[2.2 Венгерский алгоритм 7](#_Toc59616685)

[2.3 Алгоритм МаксиМин 8](#_Toc59616686)

[2.4 Алгоритм на основе принципа минимального риска 8](#_Toc59616687)

[3. Разработка диаграмм прецедентов и последовательностей 9](#_Toc59616688)

[3.1 Разработка диаграммы вариантов использования 9](#_Toc59616689)

[3.2 Разработка диаграммы последовательностей 10](#_Toc59616690)

[4. Структура подсистемы 11](#_Toc59616691)

[5. Разработка структуры диалога 12](#_Toc59616692)

[5.1 Общее описание 12](#_Toc59616693)

[5.2 Структура системного меню 13](#_Toc59616694)

[5.3 Структура диалогового окна настроек 14](#_Toc59616695)

[5.4 Структура окна управления решением 14](#_Toc59616696)

[5.5 Структура панели инструментов 15](#_Toc59616697)

[5.6 Горячие клавиши 15](#_Toc59616698)

[6. Информационное обеспечение 16](#_Toc59616699)

[6.1 Инфологическая модель 16](#_Toc59616700)

[6.2 Логическая модель 16](#_Toc59616701)

[6.3 Физическая модель 17](#_Toc59616702)

[6.4 Реализация 18](#_Toc59616703)

[7. Разработка ПО прототипа подсистемы 19](#_Toc59616704)

[7.1 Общее описание разработки диаграммы классов 19](#_Toc59616705)

[7.2 Структура СУБД 19](#_Toc59616706)

[7.3 Структура рисователя 19](#_Toc59616707)

[7.4 Структура решателя 20](#_Toc59616708)

[7.5 Общее описание разработки прототипа 21](#_Toc59616709)

[7.6 Работа пользователя с подсистемой 21](#_Toc59616710)

[Заключение 23](#_Toc59616711)

[Список литературы 24](#_Toc59616712)

[Приложение А. Диаграмма прецедентов 25](#_Toc59616713)

[Приложение Б. Диаграмма последовательностей 26](#_Toc59616714)

[Приложение В. Граф диалога 27](#_Toc59616715)

[Приложение Г. Диаграмма классов 28](#_Toc59616716)

[Приложение Д. Листинг 30](#_Toc59616717)

[Example.cpp 30](#_Toc59616718)

Введение

Автоматизация проектирования занимает особое место среди информационных технологий. Автоматизация проектирования - синтетическая дисциплина, ее составными частями являются многие другие современные информационные технологии. Так, техническое обеспечение систем автоматизированного проектирования (САПР) основано на использовании вычислительных сетей и телекоммуникационных технологий, в САПР используются персональные компьютеры и рабочие станции.

Знание основ автоматизации проектирования и умение работать со средствами САПР требуется любому инженеру разработчику. Компьютерами насыщены проектные подразделения, конструкторские бюро и офисы. Расчеты с помощью логарифмической линейки или оформление отчета на пишущей машинке стали анахронизмом. Предприятия, ведущие разработки без САПР или лишь с малой степенью их использования, оказываются неконкурентоспособными из - за больших материальных и временных затрат на проектирование и из - за невысокого качества проектов.

Малых временных и материальных затрат при решении проектных задач, добиваются за счет специализации систем. При этом растет число различных САПР. Чтобы снизить расходы на разработку многих специализированных САПР, разумно строить при максимальном использования унифицированных составных частей. Условием унификации является поиск общих черт в моделировании, анализе и синтезе разнородных технических объектов.

Задача о назначениях - вид задачи линейного программирования, с помощью которой решаются вопросы типа: как распределить рабочих по станкам, чтобы общая выработка была наибольшей или затраты на заработную плату наименьшими.

1. Анализ ТЗ

Задача данной работы состоит в проектировании и частичной реализации учебной подсистемы САПР для решения задачи назначения. Система должна иметь графический редактор и уметь работать с примерами (для чего необходимо создать специализированную базу данных). Систему решено было разрабатывать под систему Windows (программа теоретически должна работать в версиях Windows начиная с XP) на языке C++.

2. Анализ методов и алгоритмов

2.1 Постановка задачи

Традиционная формулировка задачи о назначениях такова.

Пусть имеется квадратная матрица a порядка n.

Решением задачи о назначениях для этой матрицы называется такой набор из n элементов матрицы (по одному в каждой строке и каждом столбце), что сумма этих элементов минимальна по всем наборам.

2.2 Венгерский алгоритм

Алгоритм был разработан и опубликован Гарольдом Куном (Harold Kuhn) в 1955 г. Сам Кун дал алгоритму название "венгерский", потому что он был в значительной степени основан на более ранних работах двух венгерских математиков: Денеша Кёнига (Dénes Kőnig) и Эйгена Эгервари (Jenő Egerváry).

В 1957 г. Джеймс Манкрес (James Munkres) показал, что этот алгоритм работает за (строго) полиномиальное время (т.е. за время порядка полинома от n, не зависящего от величины стоимостей).

Поэтому в литературе данный алгоритм известен не только как "венгерский", но и как "алгоритм Куна-Манкреса" или "алгоритм Манкреса".

Впрочем, недавно (в 2006 г.) выяснилось, что точно такой же алгоритм был изобретён за век до Куна немецким математиком Карлом Густавом Якоби (Carl Gustav Jacobi). Дело в том, что его работа "About the research of the order of a system of arbitrary ordinary differential equations", напечатанная посмертно в 1890 г., содержавшая помимо прочих результатов и полиномиальный алгоритм решения задачи о назначениях, была написана на латыни, а её публикация прошла незамеченной среди математиков.

Также стоит отметить, что первоначальный алгоритм Куна имел асимптотику O(n^4), и лишь позже Джек Эдмондс (Jack Edmonds) и Ричард Карп (Richard Karp) (и независимо от них Томидзава (Tomizawa)) показали, каким образом улучшить его до асимптотики O(n^3).[3]

Венгерский алгоритм даёт точное решение и имеет асимптотическую сложность – n^3.

2.3 Алгоритм МаксиМин

Алгоритм основан на принципе назначении максимального значения из минимальных значений в строках и столбцах. Алгоритм даёт приближенное решение и имеет сложность n^3.[2]



Рисунок №1. Схема алгоритма МаксиМин

2.4 Алгоритм на основе принципа минимального риска

Алгоритм аналогичен алгоритму МаксиМин за тем лишь исключением, что тут для каждой строки и столбца находится не один минимальный элемент, а пара минимальных элементов и разность между ними. И сам максимум считается именно между разностей. Алгоритм даёт приближенное немного более точное решение и имеет асимптотическую сложность n^3.[2]

3. Разработка диаграмм прецедентов и последовательностей

3.1 Разработка диаграммы вариантов использования

Диаграмма прецедентов(Вариантов использования) была разработака в CASE системе ArgoUml. Саму диаграмму можно посмотреть в Приложении А. На диаграмме присутствует два действующих лица – Пользователь и Администратор. Пользователь становиться администратором когда запускает приложение от имени администратора компьютера. Разделение на пользователя и администратора сделано для того чтобы обычный пользователь с умыслом или без не смог удалить или испортить файл примера.

Пользователь имеет ряд возможностей:

* Загрузка примера
* Выбор представления задачи
* Изменения подробности решения
* Выбор алгоритма
* Решение задачи
* Получение справки

Выбор представления состоит в выборе изображения данных задачи из двух вариантов: в виде двудольного графа и в виде матрицы. Представление в виде графа удобно для задач с небольшим числом назначаемых и мест. В ином случае удобней представление в виде матрицы (так как она компактней).

Подробность решения выбирается также между двумя вариантами: решение сразу и пошаговое решение.

В приложении планируется реализовать 3 алгоритма решения: Венгерский, МаксиМин и алгоритм наименьшего риска.

Получение справки состоит описании каждого шага при пошаговом решении задачи. Кроме того в пунтке меню «Помощь» можно просмотреть цельное объяснение каждого из трёх алгоритмов.

Администратор имеет дополнительные возможности:

* Создание примера
* Редактирование примера
* Сохранение примера
* Удаления примера

Редактирование заключается в изменении условий задачи: добавление/удаление назначаемых, добавление/удаление мест, изменение цен назначения, изменение цветовой схемы, добавление текстовых пометок, а также самого названия и описания задачи.

3.2 Разработка диаграммы последовательностей

Разработанную диаграмму последовательностей можно посмотреть в Приложении Б. В диаграмме выделено 7 сущностей:

* Пользователь
* Интерфейс
* СУБД
* Пример
* Рисователь
* Решатель
* Справчник

Пользователь взаимодействует со всеми остальными сущностями только через интерфейс. Создание, сохранение и удаление примеров происходит через СУБД. Рисователь отрисовывает загруженный пример, а решатель сразу его решает (Сохраняя каждый шаг решения в специальный вектор).

4. Структура подсистемы



Рисунок №2. Структура подсистемы для решения задачи назначения

Подсистема подразделяется на следующие модули: интерфейс пользователя, графический редактор, решатель задачи, СУБД, базы данных и обучающего модуля.

Пользовательский интерфейс необходим для взаимодействия с пользователем. Благодаря использованию интерфейса и графического редактора в решение задачи назначения вносится обучающий элемент.

Графический редактор дает возможность отображать граф или матрицу, необходимые при решении задачи назначения и имеет возможность переключиться к другому виду в любой момент. Графический редактор включает в себя возможность добавления, выделения, перемещения и удаления связей и вершин, так же позволяет в совокупности с модулем решения задачи пошагово визуализировать решение задачи назначения разными алгоритмами.

Модуль обучения открывает окно, включающее в себя заголовок шага, описание шага, кнопки далее/назад.

Задача назначения решается следующими алгоритмами: Венгерским, Минмакса, Наименьшего риска.

Специализированная система управления БД позволяет сохранять и загружать данные о примерах.

БД хранит информацию об исходных данных(в виде матрицы назначения) и результатах.

5. Разработка структуры диалога

5.1 Общее описание

Диалог пользователя с приложением осуществляется через элементы графического интерфейса Windows: системного меню и панелей инструментов, для элементов меню есть возможность использования горячих клавиш. Граф диалога можно посмотреть в Приложении В. Также с помощью щелчка мыши можно выделять элемент непосредственно в графическом редакторе (к выделяемым элементам относятся вершины, связи и надписи). Для организации работы с просмотром решения/обучением создано вспомогательное окно ManageToolWnd, в котором отображается информация о текущем шаге решения. Для организации работы с графическим редактором реализована панель инструментов для рисования DrawToolWnd. Эти два окна можно сворачивать и разворачивать либо с помощью меню, либо с помощью двойного щелчка по окну. Эти два окна не могут быть развёрнуты одновременно: первое развёрнуто только в режиме решения, а второе – в режиме редактирования.

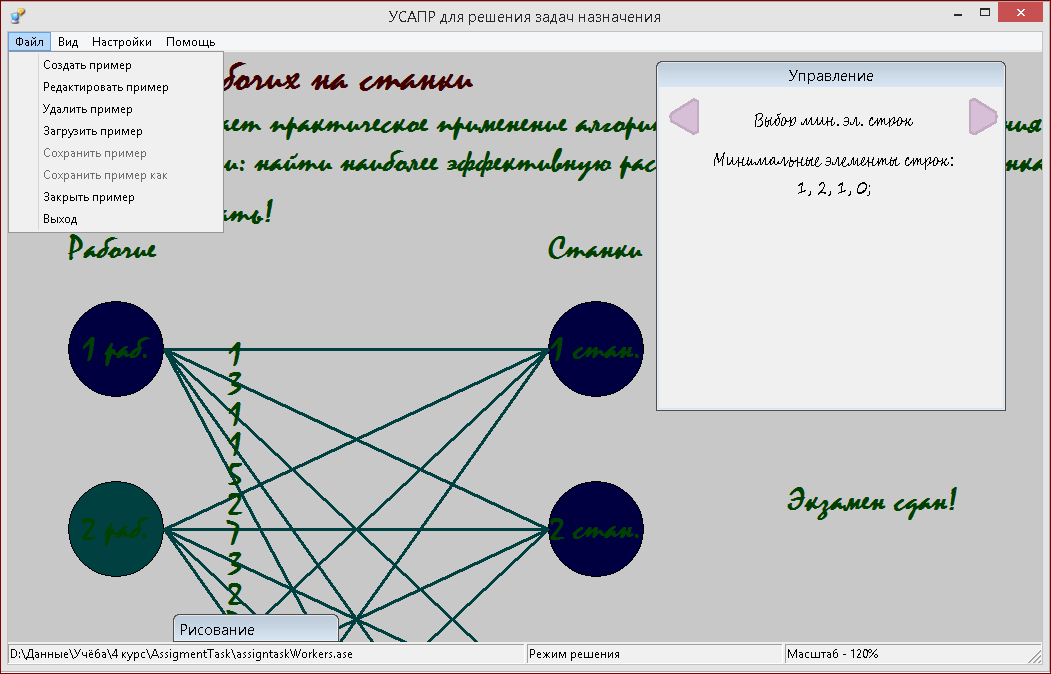


Рисунок №3. Графический интерфейс

5.2 Структура системного меню

Системное меню состоит из 4 статических пунктов:

* Файл
* Вид
* Настройки
* Помощь

Пункт меню Файл включает в себя следующие всплывающие пункты:

* Создать пример\*(включает режим редактирования)
* Редактировать пример(включает режим редактирования)
* Удалить пример
* Загрузить пример\*(включает режим решения)
* Сохранить пример
* Сохранить пример как
* Закрыть пример\*(включает нейтральный режим)
* Выход\*

При выборе пунктов помеченных звёздочкой, в случае если пример не сохранён, всплывает окно с предупреждением, что у вас есть открытый несохранённый пример и вариантами действий: закрыть не сохранив или отменить действие и сначала сохраниться.

Пункт меню Вид имеет следующие всплывающие пункты:

* Окно управления
* Окно рисования

Эти пункты просто сворачивают/разворачивают соответствующие окна.

Пункт меню Настройки просто вызывает диалоговое окно настроек.

Пункт меню Помощь состоит из следующих всплывающих пунктов:

* О программе
* Горячие клавиши
* Венгерский алгоритм
* Алгоритм МаксиМин
* Алгоритм минимального риска

5.3 Структура диалогового окна настроек

В этом диалоговом окне можно настроить 3 параметра: Подробность решения(сразу или пошагово), представление(граф или матрица), алгоритм(венгерский, максимин, минималный риск). Для подтверждения изменений нужно нажать кнопку ОК, а для отмены – кнопку ОТМЕНА. При смене алгоритма пример заново перерешивается.

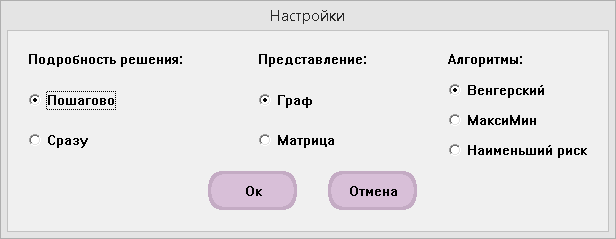


Рисунок №4. Диалоговое окно настроек

5.4 Структура окна управления решением

Окно отображает информацию о текущем шаге решения. Оно содержит поля для заголовка и описания шага. Кроме того имеется две треугольные кнопки для перехода к следующему шагу или возврата к предыдущему.

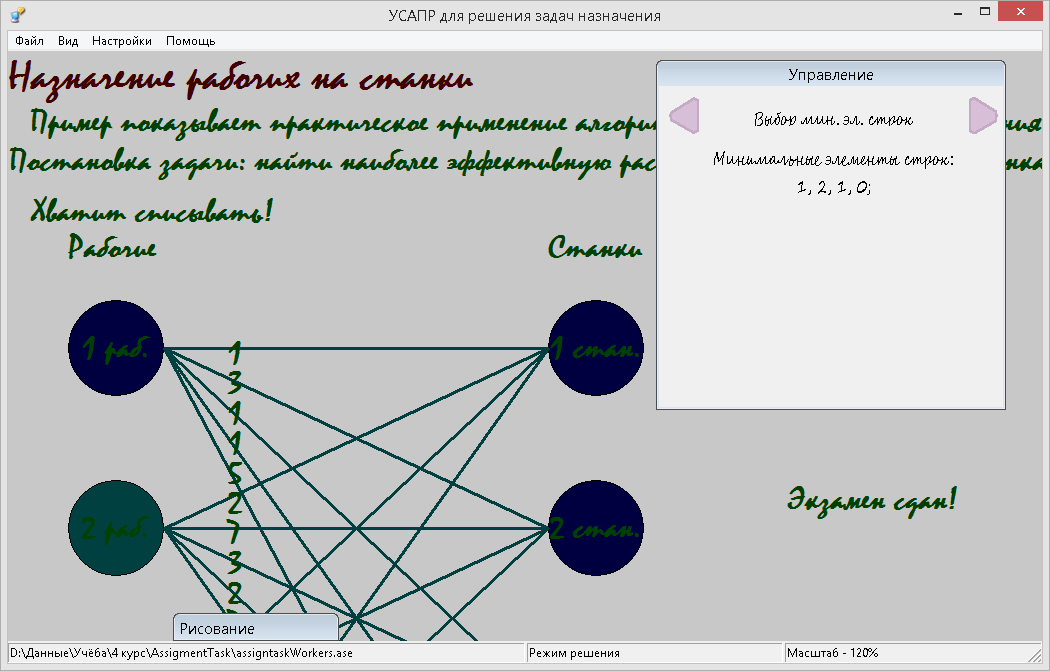


Рисунок №5. Окно управления

5.5 Структура панели инструментов

Панель состоит из 3 кнопок. Первая отвечает за создание вершины, вторая за создание надписи, а третья за настройку цветовой схемы.

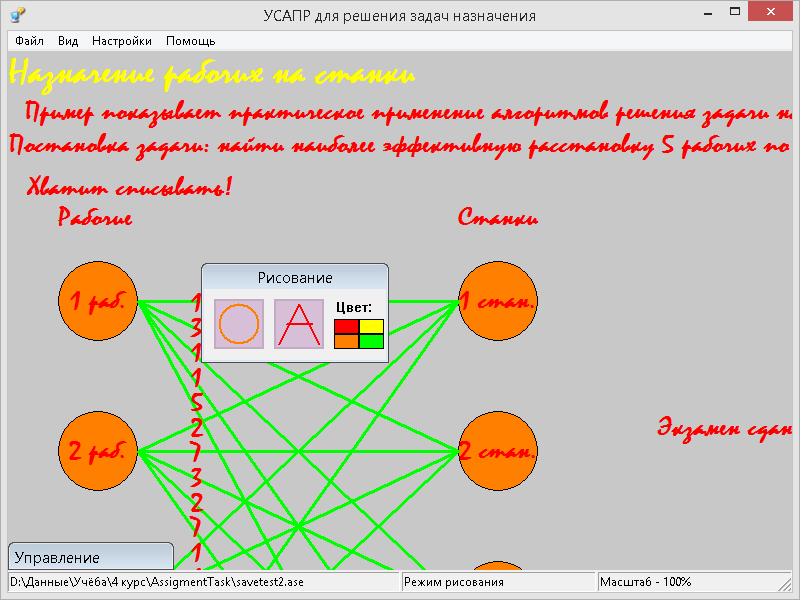


Рисунок №6. Панель инструментов для рисования

5.6 Горячие клавиши

Список горячих клавиш:

* ALT + N – Создать пример
* ALT + E - Редактировать пример
* ALT + L – Загрузить пример
* ALT + S – Сохранить пример
* ALT + C – Закрыть пример
* ALT + Q – Выйти
* ALT + M - Скрыть/Показать окно управления решением
* ALT + D – Скрыть/Показать панель инструментов для рисования
* ALT + T – Настройка
* ALT + H – Горячие клавиши
* COMMA(запятая) – Уменьшить масштаб
* SHIFT + COMMA – Намного уменьшить масштаб
* PERIOD(точка) – Увеличить масштаб
* SHIFT + PERIOD – Намного увеличить масштаб
* ARROWS - Движение

6. Информационное обеспечение

6.1 Инфологическая модель

Инфологическая модель представлена сделанной в MySqlWorkbench диаграммой сущность-связь.

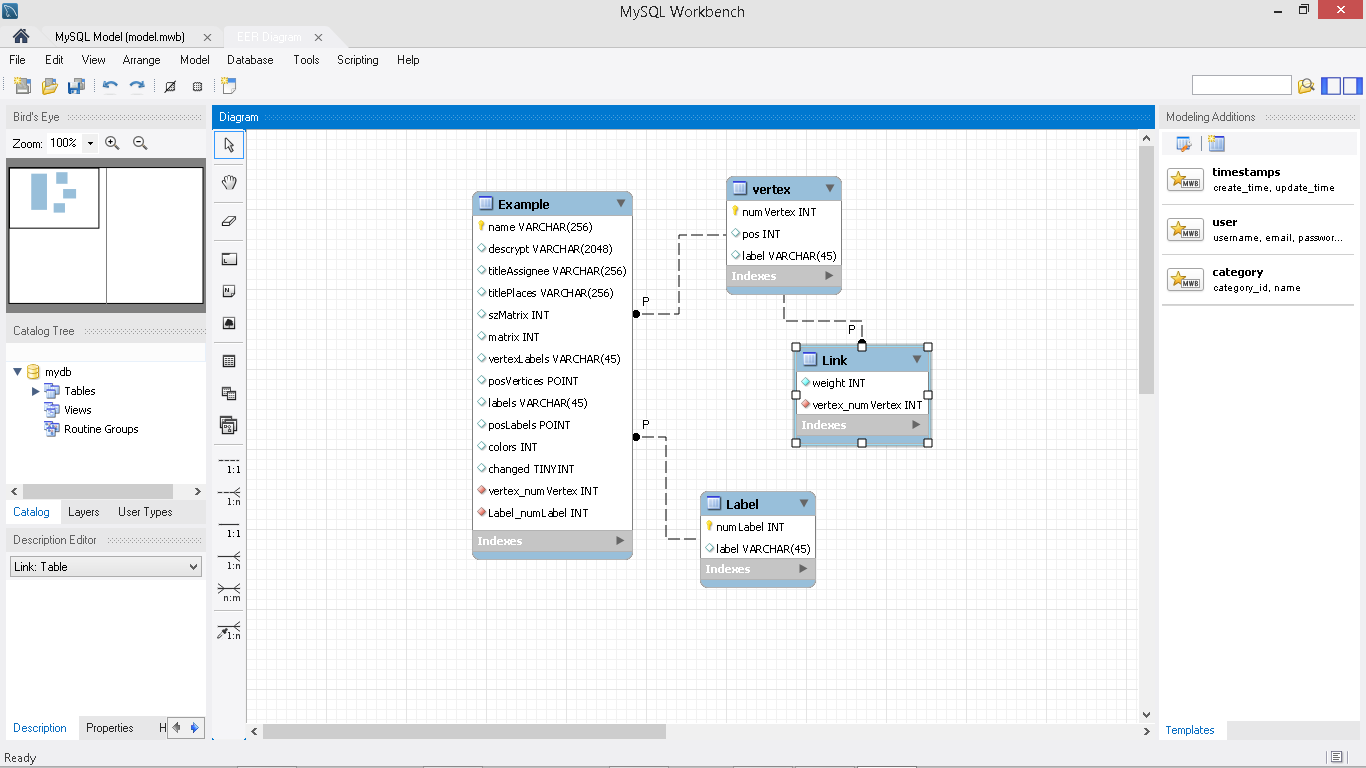


Рисунок №7. Диаграмма сущность-связь

6.2 Логическая модель

База данных представлена в объектно-ориентированном виде. В программе она состоит из класса Example, и набора классов Vertex, Link, Label в которые преобразуется матрица примера для удобства работы рисовальщика. Примеры сохраняются в файловой системе компьютера, откуда их можно многократно загружать и изменять.

6.3 Физическая модель

В программе пример будет представлять собой структуру данных. Пример можно будет загрузить с диска и сохранить на диск. Формат файла примера представлен на рисунке №8:

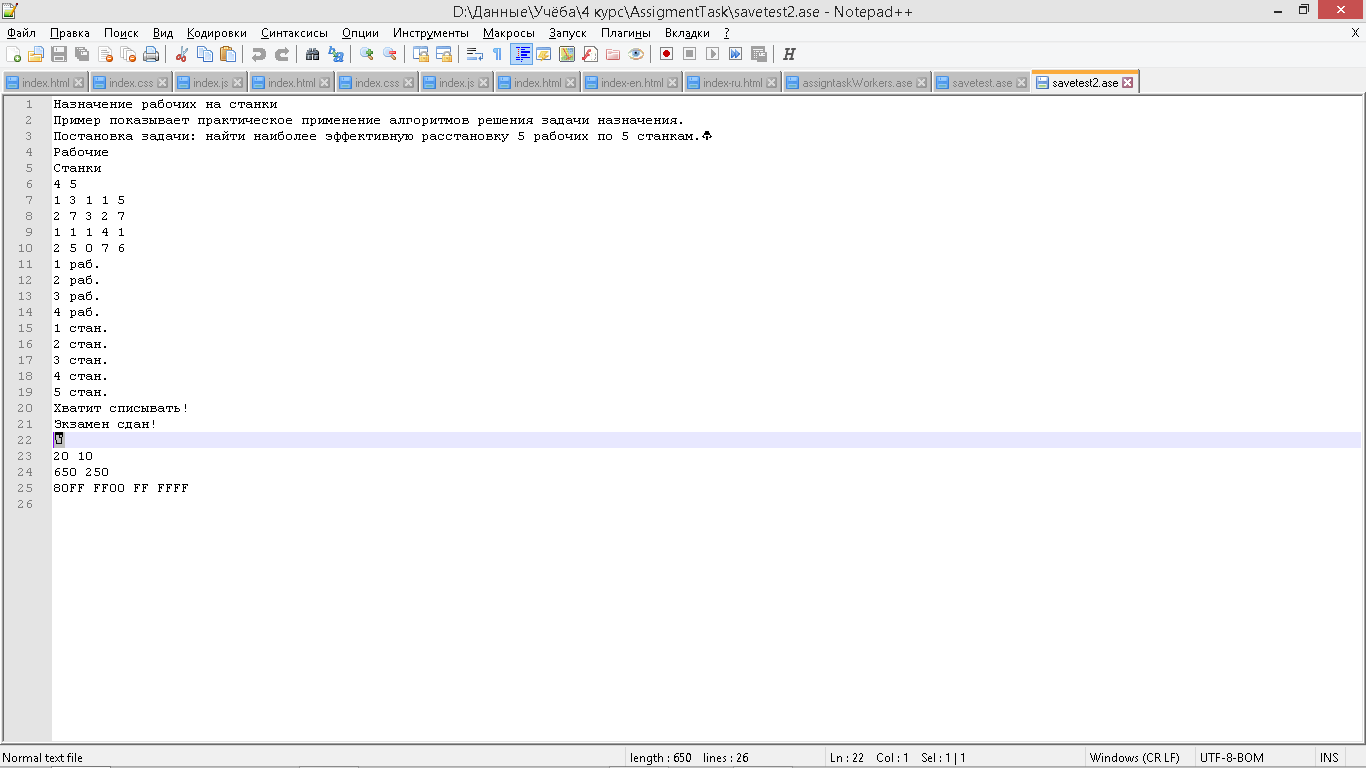


Рисунок №8. Формат файла

Файл состоит из Названия примера(одна строка), далее идёт многострочное Описание примера(завершается юникод символом мухомора). В следующих строках записаны: Название назначаемых и Название мест назначения. После этого записываются размеры матрицы, и с новой строки сама матрица. Далее записаны надписи назначаемых и мест назначения каждая с новой строка. После идёт список пометок оканчивающийся юникод символом пивной кружки и их координаты. В конце в одной строке описаны цвета заголовков, надписей, вершин и связей.

6.4 Реализация

Для хранения названия проекта, его описания и заголовков используются массивы символов. Для хранения матрицы используется указатель на динамически выделяемый массив int. Для надписей используются вектора строк. А для позиций указатели на динамически выделяемые массивы точек. Также присутствует булева переменная changed, которая имеет значение true если пример был изменён в редакторе.

struct Example {

TCHAR name[256];

TCHAR descrypt[2048];

TCHAR titleAssignee[256];

TCHAR titlePlaces[256];

unsigned szMatrix;

int \* matrix;

vector<LPCTSTR> vertexLabels;

LPPOINTS posVertices;

vector<LPCTSTR> labels;

LPPOINTS posLabels;

COLORREF colorVertex;

COLORREF colorLink;

COLORREF colorLabel;

COLORREF colorTitle;

bool changed;

Example();

Example(LPCTSTR filename);

void Save(LPCTSTR filename) const;

~Example();

};

7. Разработка ПО прототипа подсистемы

7.1 Общее описание разработки диаграммы классов

В процессе проектирования учебной подсистемы САПР в ArgoUml была разработана UML диаграмма классов, которая приведена в приложении Г. Подсистему решено было разбить на 13 классов, часть классов группируется в 2 иерархии по 4 класса. Все основые компоненты сосредоточены в классе main.[1]

7.2 Структура СУБД

СУБД состоит из одного класса – структуры Example, которая содержит в себе переменные для хранения информации о примере, и методы для сохранения, и загрузки. Удаления реализуется просто вызовом функции remove для выбранного файла. Подробное описание смотрите в разделе Информационное обеспечение.

7.3 Структура рисователя

Сам рисователь реализован через класс Drawer и вложенной в него иерархии отрисовываемых компонентов. Класс Drawer обеспечивает возможность отрисовки, изменения масштаба, изменения положения, выделения. Кроме того с помощью панели инструментов для рисования Drawer даёт возможность добавлять вершину, надпись, а также менять цветовую схему. Также имеется возможность менять представление примера с графа на матрицу. В режиме матрицы пользовательские надписи не отображаются.

Drawer содержит в себе вектора вершин, связей и надписей, а также текущее представление отрисовки(матрица или граф). Также есть прочие члены класса хранящии информацию о шрифтах, перьях, контекстах рисования.

Первая иерархия – это иерархия отрисовываемых компонентов с помощью объекта класса Drawer.

Список классов иерархии:

* DrawableComponent
* Vertex
* Link
* Label

DrawableComponent – абстрактный класс, в нём есть защищённая переменная drawer, которая хранит ссылку на рисователя, и 4 виртуальных метода присущих всем отрисовываемым компонентам: IsPointIn(проверяет находится ли точка внутри фигуры компонента), OnMouseClick(для обработки щелчка мыши по фигуре компонента), Normalize(для приведения компонента в стандартное состояние), Paint(Отрисовка компонента).

Vertex – класс вершины, в нём хранится информация о номере вершины, позиции, надписи и пометки(нужны для работы венгерского алгоритма). Также есть статическая переменная хранящая ширину.

Link – класс связи, в объекте класса присутствуют ссылки на вершины, которые соединяет связь, цена соединения и пометка.

Label – класс надписи, содержит в себе информацию о положении надписи, её текста и ограничивающего прямоугольника.

7.4 Структура решателя

Вторая иерархия состоит из алгоритмов решения задачи назначения. Список классов иерархии:

* AssignTaskAlgorithm
* VengerAlg
* MaxiMinAlg
* MinRiskAlg

AssignTaskAlgorithm – абстрактный класс алгоритма решения задачи назначения, имеет вектор хранящий информацию о каждом шаге решения. Также класс имеет функции для очистки решения(то есть вектора), и получении информации об определённом шаге. Кроме того есть виртуальная функция для решения задачи.

VengerAlg – реализация венгерского алгоритма решения.

MaxiMinAlg – реализация алгоритма МаксиМин

MinRiskAlg – реализация алгоритма минимального риска

Помимо этого в структуре решателя есть ещё и классы хранящие информацию о каждом шаге полученного решения: StepInfo и ActionInfo.

StepInfo – структура хранящяя информацию о шаге решения. В ней определены переменные для хранения заголовка шага и описания шага. Кроме того здесь присутствует информаци о действии совершаемом на этом шаге. Это нужно для реализации интерактивности. То есть чтобы пользователю по ходу решения могли даваться задания которые он должен решить.

ActionInfo – структура хранящяя информацию о совершаемом на этом шаге действии. Содержит переменную перечисление с типом действия и 3 параметра для этого дейтствия.

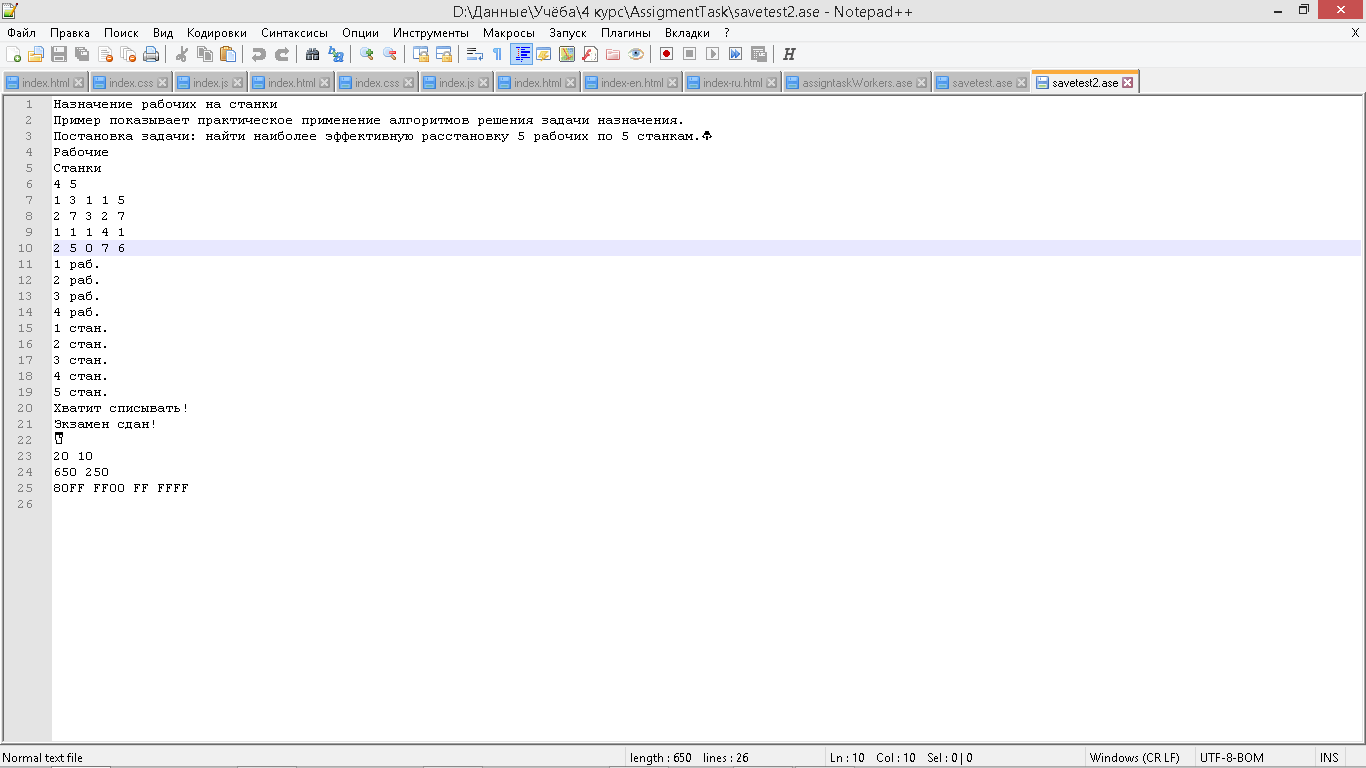
7.5 Общее описание разработки прототипа

Разработка прототипа велась на языке С++ в среде CodeBlocs. Для разработки использовался интерфейс системы windows – win32api. Программа теоретически должна работать на всех версиях Windows начиная с Windows XP. Приложение работает на UNICODE. В том числе и кодировка файлов примеров должна быть UTF8. Для большей понятности интерфейса все кнопки не имеющие надписей снабжены всплывающими подсказками.

На текущий момент реализована работа с системным меню. Полностью реализована работа окна управления решением и частично панели инструментов для рисования(не работают функции добавления вершины и надписи). Также почти полностью реализована работа графического редактора, в нём не работают лишь функции связанные с изменением графа примера. В решателе реализована общая логика решения и один из трёх алгоритмов – МаксиМин.

7.6 Работа пользователя с подсистемой

Здесь описан пример решения задачи назначения с использованием разработанной подсистемы.

  
Рисунок №9. Исходные данные

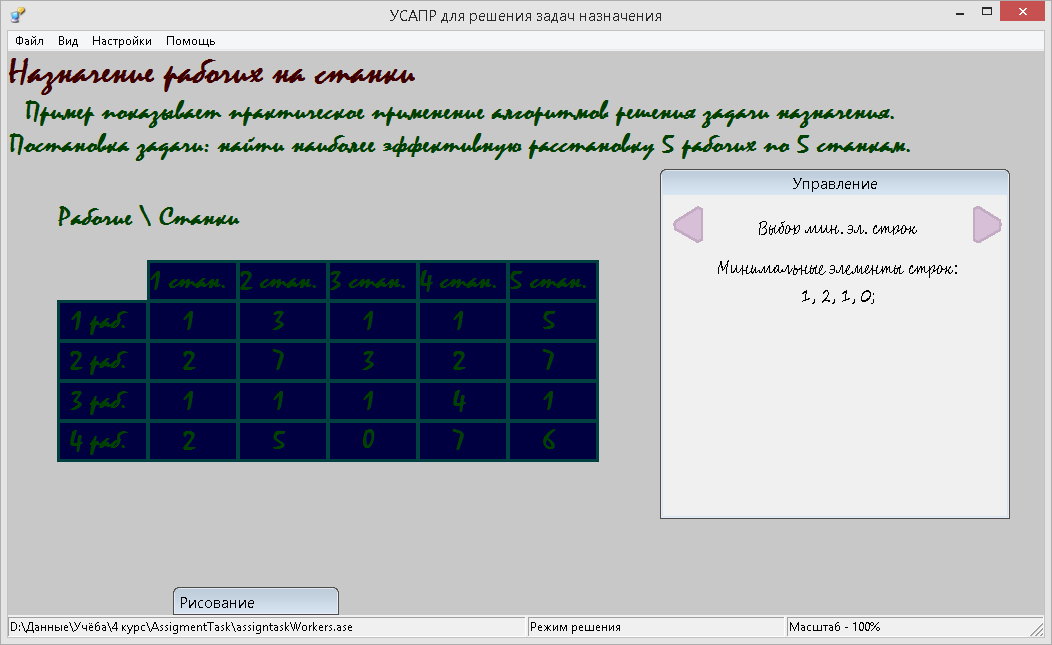


Рисунок №10. Выбор минимальных элементов строк

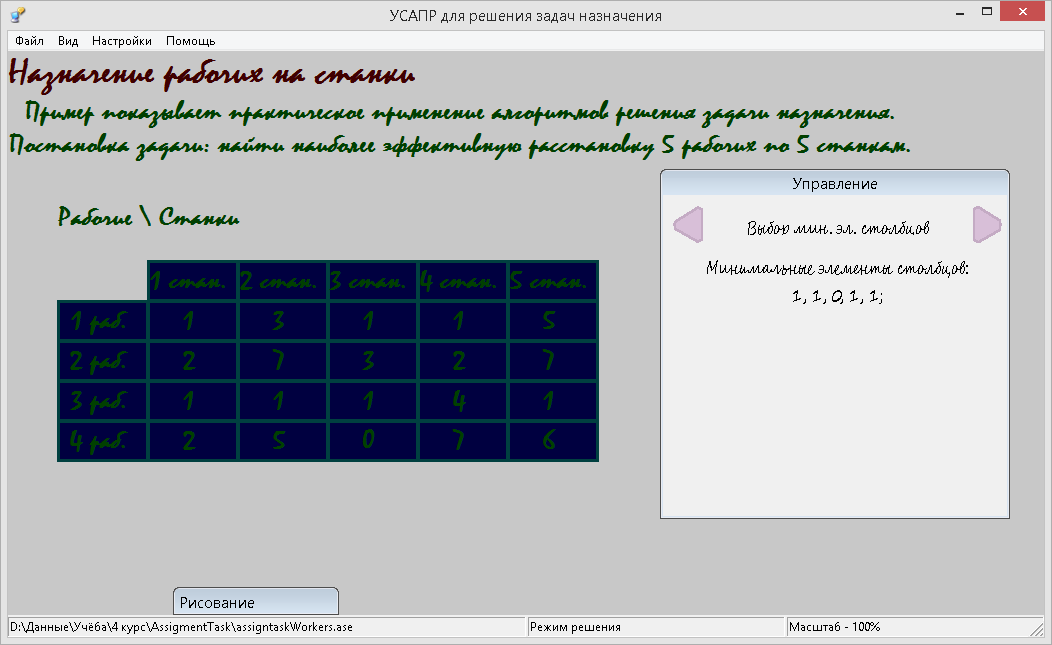


Рисунок №11. Выбор минимальных элементов столбцов

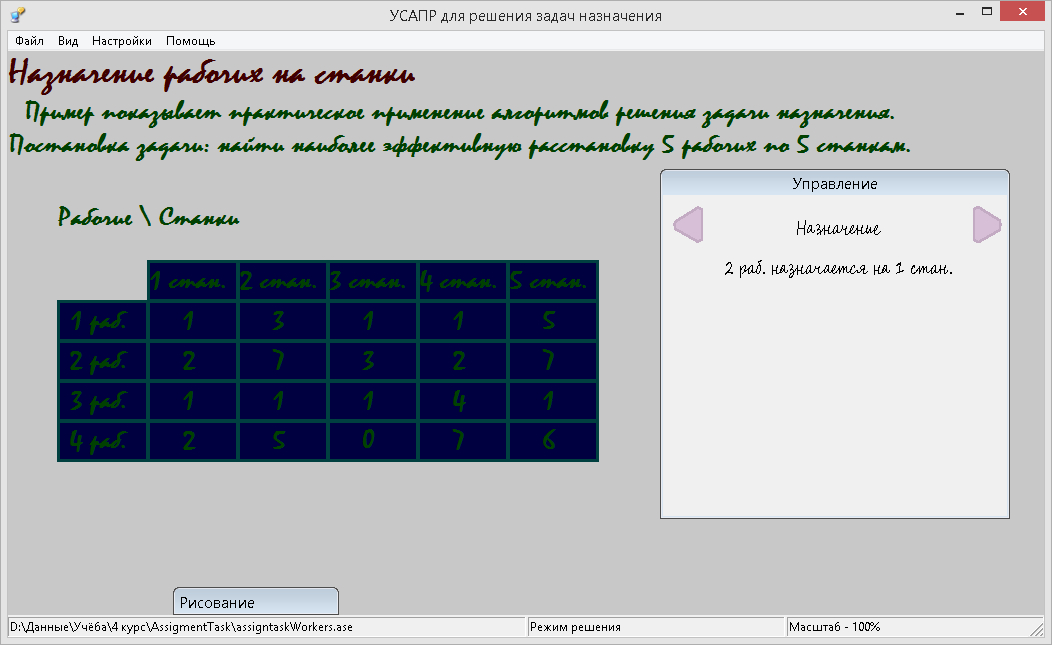


Рисунок №12. Первое назначение

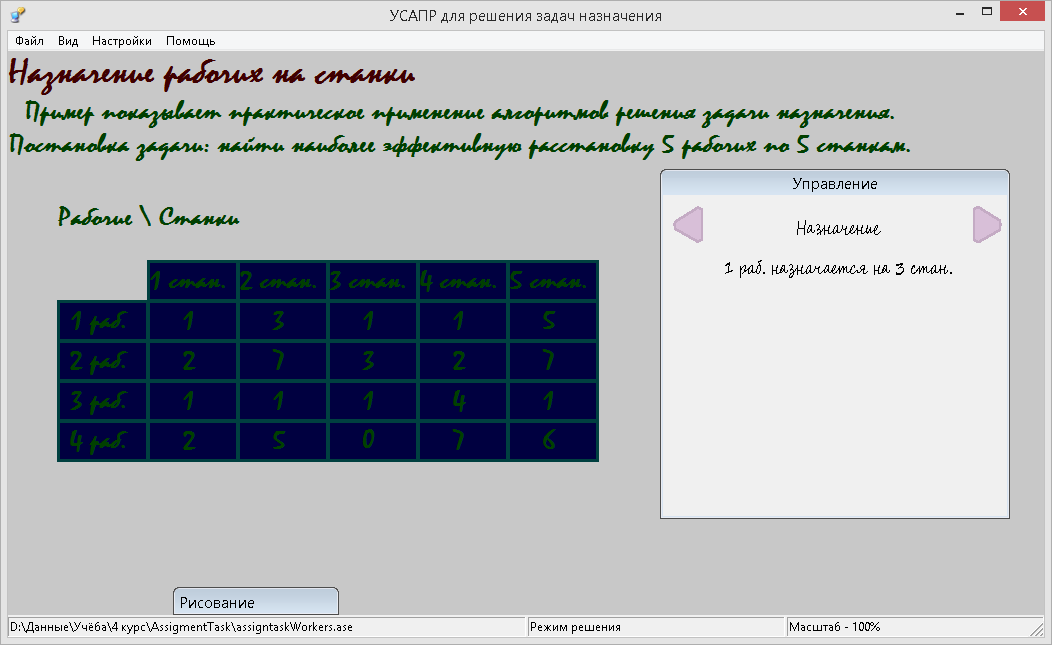


Рисунок №13. Второе назначение

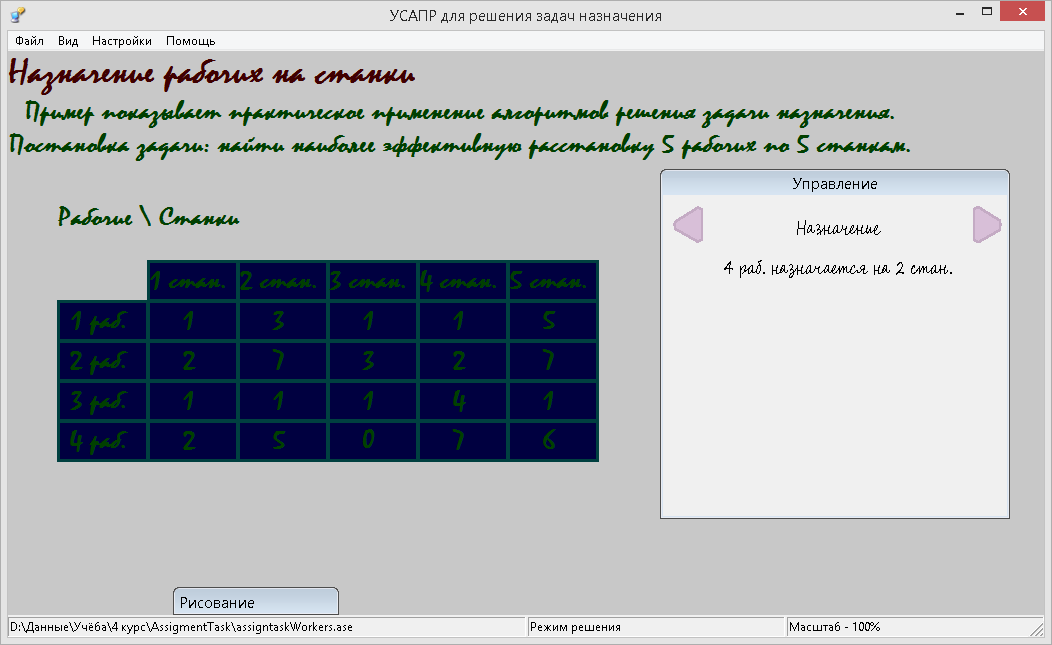


Рисунок №14. Третье назначение

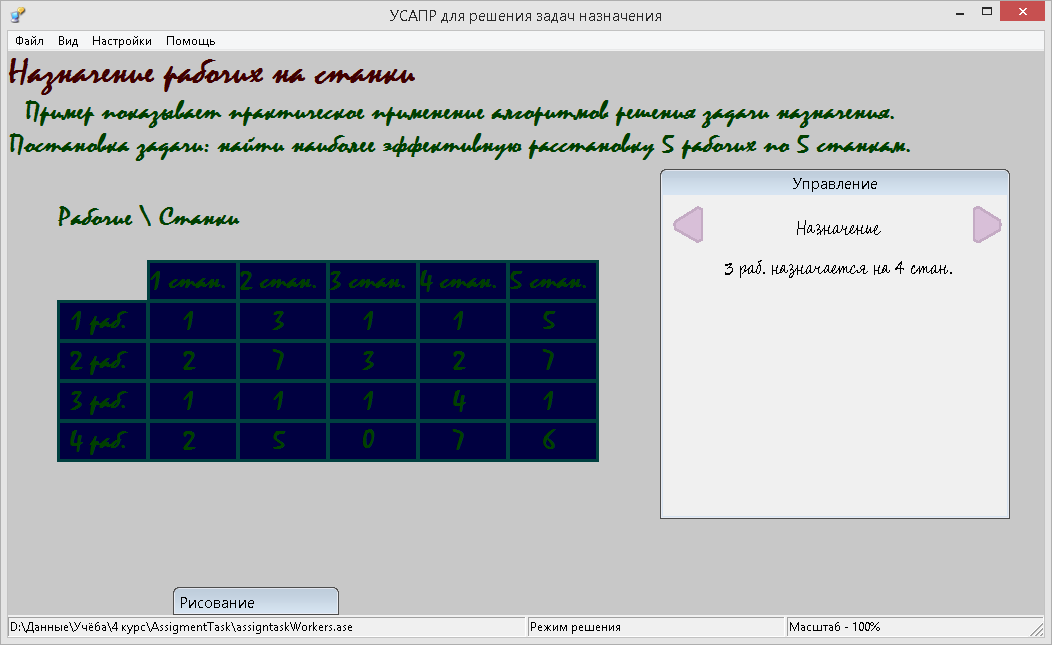


Рисунок №15. Четвёртое назначение

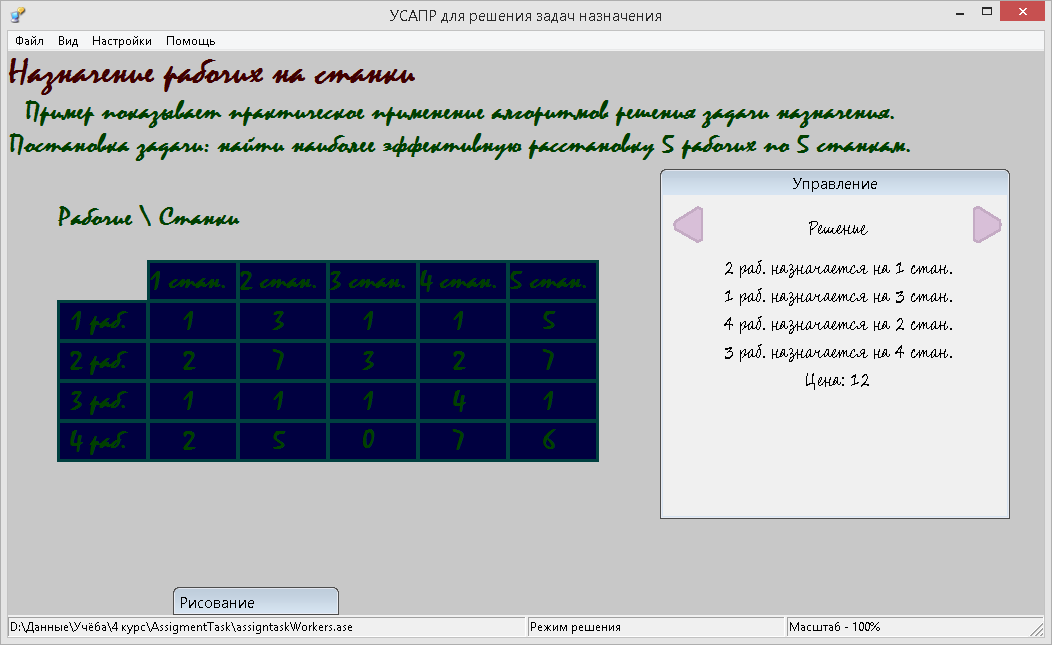


Рисунок №16. Решение

Заключение

В ходе работы были разработаны UML диаграммы прецедентов, последовательностей, классов в CASE-системе ArgoUml. Также была разработана структура подсистемы и выполнено проектирование ИО и ПО. После был разработан прототип учебной подсистемы САПР для решения задачи назначения. Прототип протестирован в системе Windows 8.1.

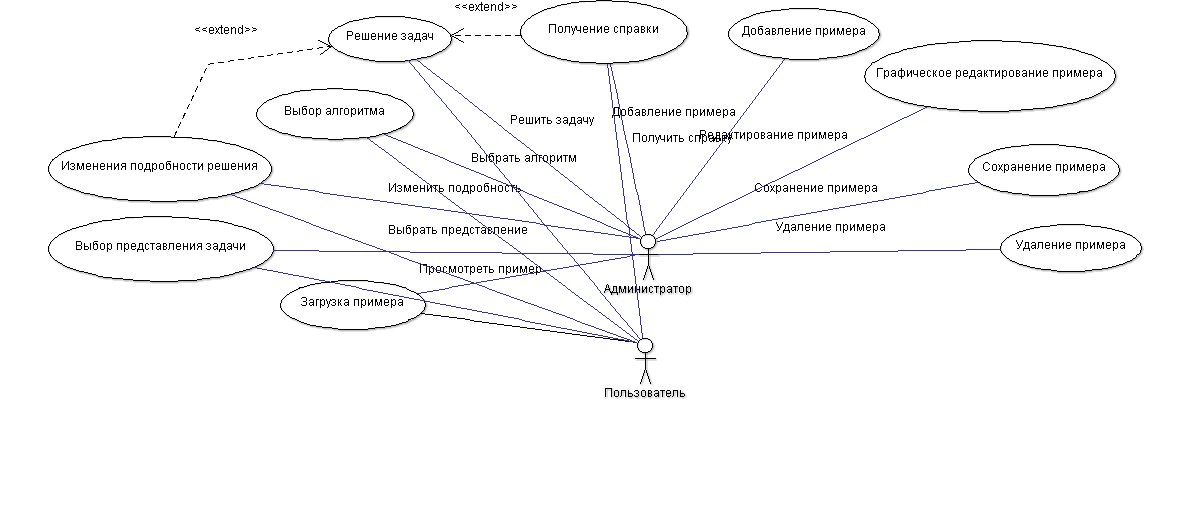
Список литературы

**Буч Гради** Объектно-ориентированный анализ и проектирование с примерами приложений [Книга]. - Москва : Издательский дом "Вильямс", 2008. - 3-е издание.

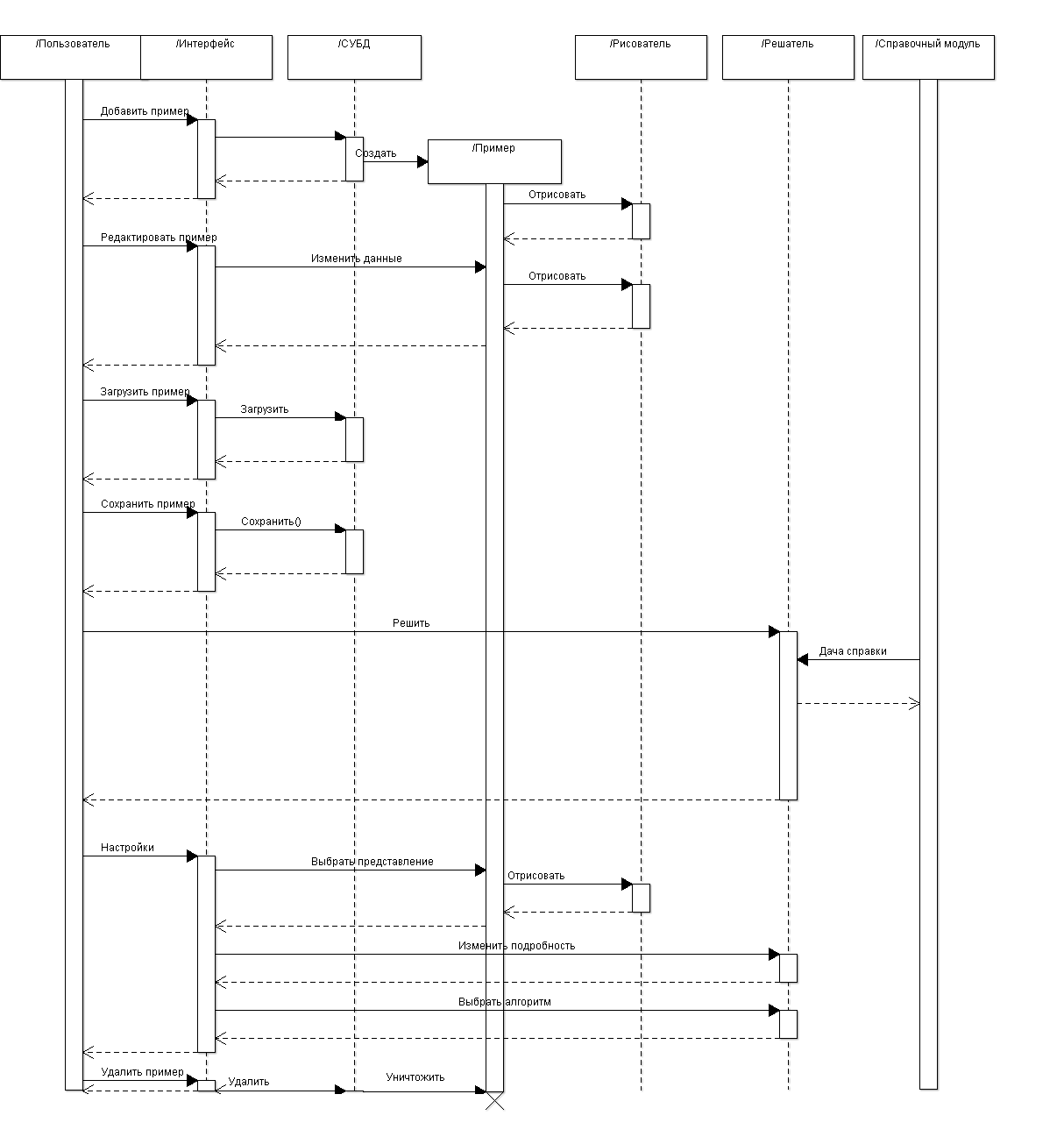
**Е. Г. Шереметьева, Т. А. Макурина, Л. В. Игошина** Решение оптимизационных задач методами линейного программирования [Книга]. - Пенза : Издательство Пензенского Государственного Университета, 2005.

**Максим Иванов** Венгерский алгоритм решения задачи о назначениях [https://e-maxx.ru/algo/assignment\_hungary]. - 2012 г..

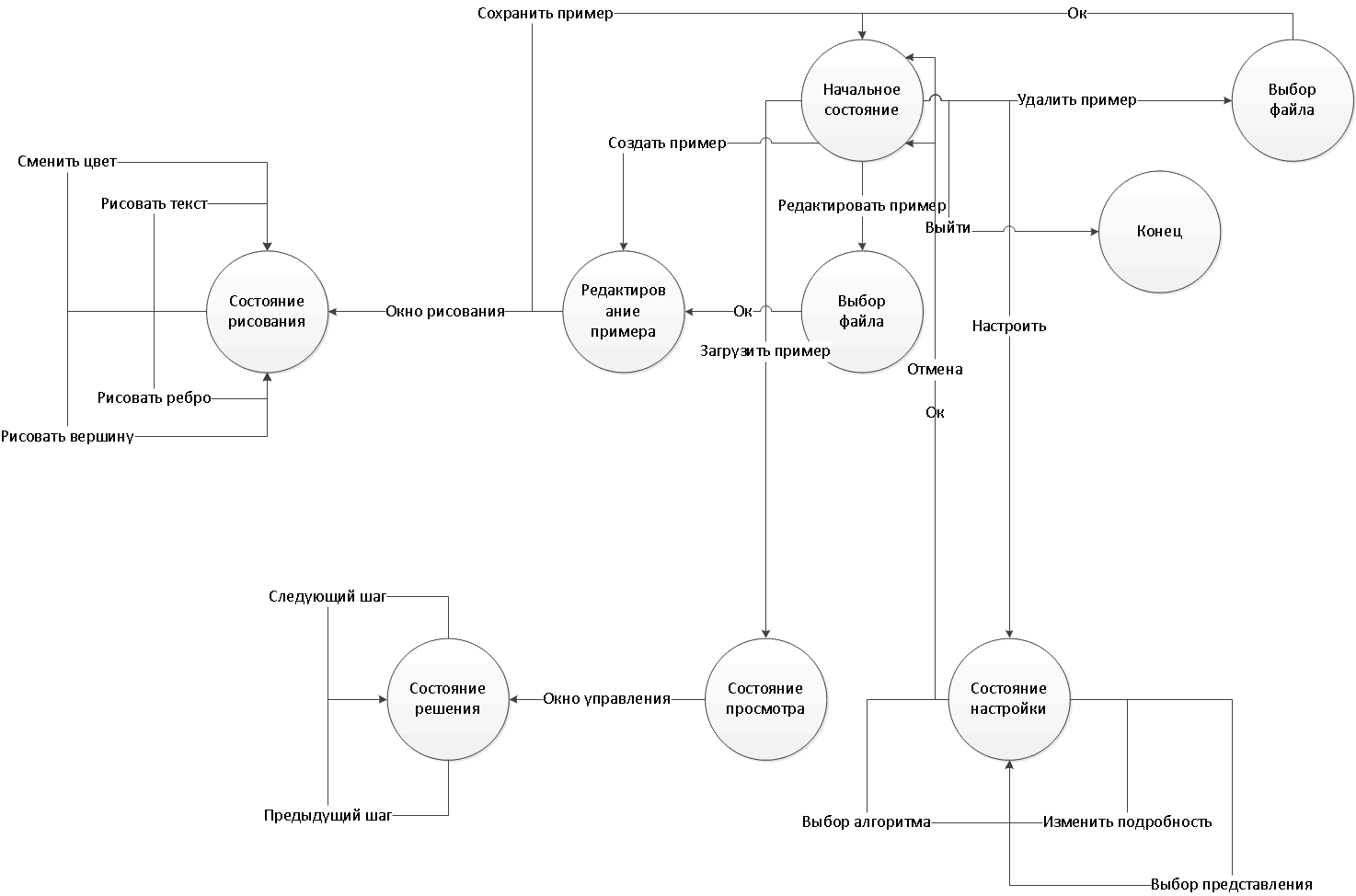
Приложение А. Диаграмма прецедентов



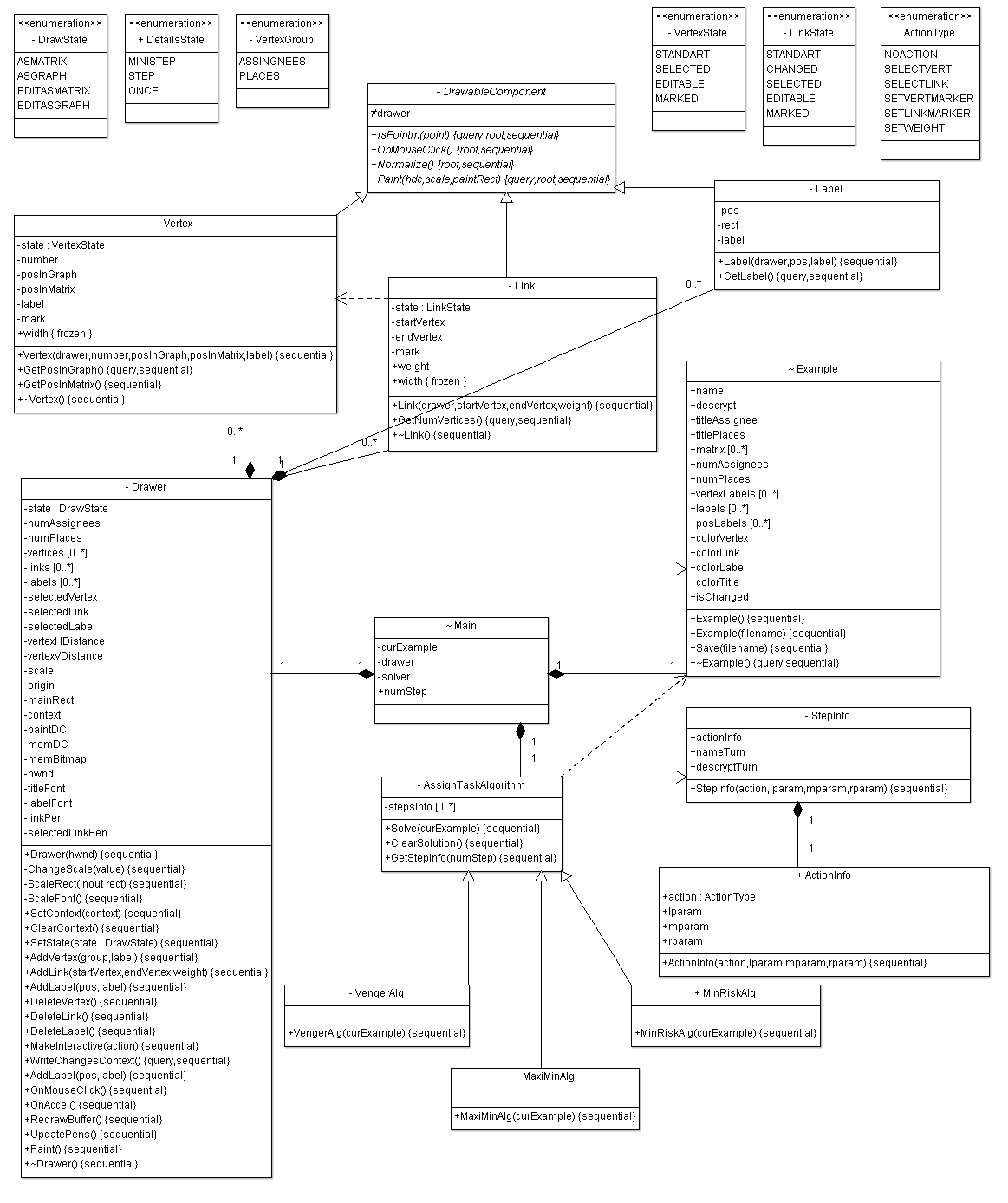
Приложение Б. Диаграмма последовательностей

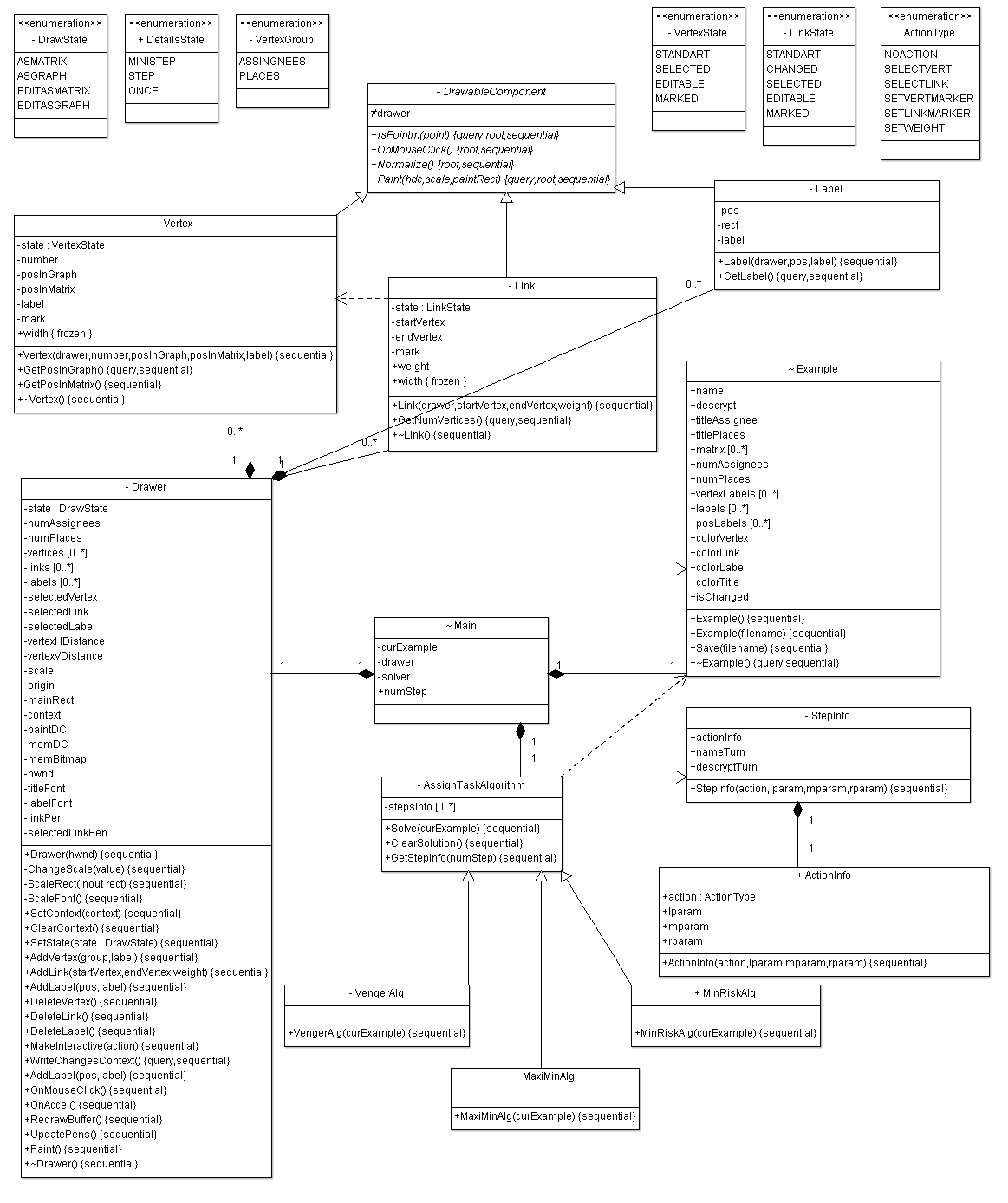


Приложение В. Граф диалога



Приложение Г. Диаграмма классов





Приложение Д. Листинг

Example.cpp

Example::Example()

: numAssignees(0), numPlaces(0), matrix(NULL), posLabels(NULL), colorVertex(colors[VERTEXCOLOR]), colorLink(colors[LINKCOLOR]),

colorLabel(colors[LABELCOLOR]), colorTitle(colors[TITLECOLOR]), changed(false)

{

name[0] = L'\0';

descrypt[0] = L'\0';

titleAssignees[0] = L'\0';

titlePlaces[0] = L'\0';

}

Example::Example(LPCTSTR filename)

: changed(false)

{

TCHAR shortBuffer[256], buffer[2048];

unsigned i,j;

FILE\* file = \_tfopen(filename, \_T("r, ccs=UTF-8"));

//Получение основных сведений о примере

if(\_ftscanf(file, \_T("%255[^\n]\n%2047[^🍄]🍄\n%255[^\n]\n%255[^\n]\n"), name, descrypt, titleAssignees, titlePlaces) != 4)

MessageBox(NULL, \_T("Ошибка при чтении общих сведений"), \_T("Ошибка"), MB\_OK | MB\_ICONERROR);

//Получение сведений об исходной матрице

if(\_ftscanf(file, \_T("%u %u\n"), &numAssignees, &numPlaces) != 2)

MessageBox(NULL, \_T("Ошибка при чтении размеров матрицы"), \_T("Ошибка"), MB\_OK | MB\_ICONERROR);

matrix = (int\*) malloc(sizeof(int) \* numAssignees \* numPlaces);

if(matrix == NULL)

MessageBox(NULL, \_T("Ошибка выделения памяти под матрицу"), \_T("Ошибка"), MB\_OK | MB\_ICONERROR);

for(i = 0; i < numAssignees; i++)

{

for(j = 0; j < numPlaces - 1; j++)

if(\_ftscanf(file, \_T("%d "), &matrix[i \* numPlaces + j]) != 1)

MessageBox(NULL, \_T("Ошибка при чтении элемента матрицы"), \_T("Ошибка"), MB\_OK | MB\_ICONERROR);

if(\_ftscanf(file, \_T("%d\n"), &matrix[i \* numPlaces + j]) != 1)

MessageBox(NULL, \_T("Ошибка при чтении элемента матрицы"), \_T("Ошибка"), MB\_OK | MB\_ICONERROR);

}

//Получение сведений о надписях вершин

for(i = 0; i < (numAssignees + numPlaces); i++)

{

if(\_ftscanf(file, \_T("%7[^\n]\n"), shortBuffer) != 1)

MessageBox(NULL, \_T("Ошибка при чтении надписи вершины"), \_T("Ошибка"), MB\_OK | MB\_ICONERROR);

vertexLabels.push\_back(\_tcsdup(shortBuffer));

}

//Получение сведений о надписях

while(1)

{

if(\_ftscanf(file, \_T("%255[^\n]\n"), shortBuffer) != 1)

MessageBox(NULL, \_T("Ошибка при чтении надписи"), \_T("Ошибка"), MB\_OK | MB\_ICONERROR);

if(!\_tcscmp(shortBuffer, \_T("🍺")))

break;

labels.push\_back(\_tcsdup(shortBuffer));

}

//Получение сведений о координатах надписей

posLabels = (LPPOINT)malloc(sizeof(POINT) \* labels.size());

if(posLabels == NULL)

MessageBox(NULL, \_T("Ошибка выделения памяти под координаты надписей"), \_T("Ошибка"), MB\_OK | MB\_ICONERROR);

for(i = 0; i < labels.size(); i++)

if(\_ftscanf(file, \_T("%d %d\n"), &posLabels[i].x, &posLabels[i].y) != 2)

MessageBox(NULL, \_T("Ошибка при чтении координат надписи"), \_T("Ошибка"), MB\_OK | MB\_ICONERROR);

//Получение сведений о цветах

if(\_ftscanf(file, \_T("%x %x %x %x\n"), &colors[VERTEXCOLOR], &colors[LINKCOLOR], &colors[LABELCOLOR], &colors[TITLECOLOR]) != 4)

MessageBox(NULL, \_T("Ошибка при чтении информации о цветах"), \_T("Ошибка"), MB\_OK | MB\_ICONERROR);

fclose(file);

}

void Example::Save(LPCTSTR filename) const

{

unsigned i,j;

FILE\* file = \_tfopen(filename, \_T("w, ccs=UTF-8"));

\_ftprintf(file, \_T("%s\n%s🍄\n%s\n%s\n"), name, descrypt, titleAssignees, titlePlaces);

\_ftprintf(file, \_T("%u %u\n"), numAssignees, numPlaces);

for(i = 0; i < numAssignees; i++)

{

for(j = 0; j < numPlaces - 1; j++)

\_ftprintf(file, \_T("%d "), matrix[i \* numPlaces + j]);

\_ftprintf(file, \_T("%d\n"), matrix[i \* numPlaces + j]);

}

for(LPTSTR label : vertexLabels)

\_ftprintf(file, \_T("%s\n"), label);

for(LPTSTR label : labels)

\_ftprintf(file, \_T("%s\n"), label);

\_ftprintf(file, \_T("🍺\n"));

for(i = 0; i < labels.size(); i++)

\_ftprintf(file, \_T("%d %d\n"), posLabels[i].x, posLabels[i].y);

\_ftprintf(file, \_T("%X %X %X %X\n"), colors[VERTEXCOLOR], colors[LINKCOLOR], colors[LABELCOLOR], colors[TITLECOLOR]);

fclose(file);

}

Example::~Example()

{

free(matrix);

for(LPTSTR label : vertexLabels)

free((void\*)label);

for(LPTSTR label : labels)

free((void\*)label);

free(posLabels);

}