Министерство образования Республики Беларусь

Министерство образования и науки Российской Федерации

МЕЖГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Высшая математика»

**Курсовой проект**

**по дисциплине «Оптимизация проектных решений**

**на тему «Экспертный анализ методом анализа иерархий»**

|  |  |  |
| --- | --- | --- |
| Выполнил: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_  «\_\_\_»\_\_\_\_\_\_\_\_\_2022 г. | студент гр. ИСИТ-191  Харкевич А. В. |
| Руководитель: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_  «\_\_\_»\_\_\_\_\_\_\_\_\_2022 г. | ст. преподаватель  Бондарев А. Н. |

Могилев 2022 г.

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc122306592)

[1 Методика применения МАИ 4](#_Toc122306593)

[1.1 Моделирование проблемы в виде иерархии 4](#_Toc122306594)

[1.2 Определение иерархической структуры 4](#_Toc122306595)

[1.3 Объяснение иерархических структур, используемых в МАИ 5](#_Toc122306596)

[1.4 Расстановка приоритетов 6](#_Toc122306597)

[1.5 Определение приоритетов и пояснения 6](#_Toc122306598)

[1.6 Практическое применение МАИ 7](#_Toc122306599)

[2 Использование Метода Анализа Иерархий 9](#_Toc122306600)

[2.1 Практическое применение Метода Анализа Иерархий 9](#_Toc122306601)

[ЗАКЛЮЧЕНИЕ 10](#_Toc122306602)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 11](#_Toc122306603)

[Приложение А 12](#_Toc122306604)

# **ВВЕДЕНИЕ**

Оптимизация как раздел математики существует достаточно давно. Оптимизация – это выбор, то есть то, чем постоянно приходится заниматься в повседневной жизни. Термином «оптимизация» в литературе обозначают процесс или последовательность операций, позволяющих получить уточненное решение. Хотя конечной целью оптимизации является отыскание наилучшего или «оптимального» решения, обычно приходится довольствоваться улучшением известных решений, а не доведением их до совершенства. Поэтому под оптимизацией понимают скорее стремление к совершенству, которое, возможно, и не будет достигнуто.

Одним из численных методов современной оптимизации является Метод Анализа Иерархий. Метод Анализа Иерархий (МАИ) – математический инструмент системного подхода к сложным проблемам принятия решений. МАИ не предписывает лицу, принимающему решение (ЛПР), какого-либо «правильного» решения, а позволяет ему в интерактивном режиме найти такой вариант (альтернативу), который наилучшим образом согласуется с его пониманием сути проблемы и требованиями к ее решению. Этот метод разработан Р. Беллманом, Б.Н. Бруком и В. Н. Бурковым, но получил широкую известность по работам Т. Саати, который и назвал процедуру методом анализа иерархий. Публикации Саати более полно раскрыли возможности процедуры, и с тех пор МАИ активно развивается и широко используется на практике. В его основе наряду с математикой заложены и психологические аспекты. МАИ позволяет понятным и рациональным образом структурировать сложную проблему принятия решений в виде иерархии, сравнить и выполнить количественную оценку альтернативных вариантов решения. Метод Анализа Иерархий используется во всем мире для принятия решений в разнообразных ситуациях: от управления на межгосударственном уровне до решения отраслевых и частных проблем в бизнесе, промышленности, здравоохранении и образовании.

Все это определяет актуальность темы курсового проекта.

Целью работы было изучение решения с помощью Метода Анализа Иерархий.

Задачи проекта:

- изучение основ Метода Анализа Иерархий;

- изучение использования Метода Анализа Иерархий;

- разработка компьютерной программы, реализующей Метод Анализа Иерархий на языке программирования высокого уровня.

При написании курсового проекта использовались следующие методы: анализ литературы по проблеме исследования, описание, расчеты, сравнение.

Проект состоит из введения, основной части, заключения, списка использованных источников, приложения с кодом программы, реализующей метод наискорейшего спуска.

# **1 Методика применения МАИ**

# **1.1 Моделирование проблемы в виде иерархии**

Метод анализа иерархий содержит процедуру синтеза приоритетов, вычисляемых на основе субъективных суждений экспертов. Число суждений может измеряться дюжинами или даже сотнями. Математические вычисления для задач небольшой размерности можно выполнить вручную или с помощью калькулятора, однако гораздо удобнее использовать программное обеспечение (ПО) для ввода и обработки суждений. Самый простой способ компьютерной поддержки – электронные таблицы, самое развитое ПО предусматривает применение специальных устройств для ввода суждений участниками процесса коллективного выбора. Порядок применения Метода Анализа Иерархий:

* Построение качественной модели проблемы в виде иерархии, включающей цель, альтернативные варианты достижения цели и критерии для оценки качества альтернатив.
* Определение приоритетов всех элементов иерархии с использованием метода парных сравнений.
* Синтез глобальных приоритетов альтернатив путем линейной свертки приоритетов элементов на иерархии.
* Проверка суждений на согласованность.
* Принятие решения на основе полученных результатов

Первый шаг МАИ – построение иерархической структуры, объединяющей цель выбора, критерии, альтернативы и другие факторы, влияющие на выбор решения. Построение такой структуры помогает проанализировать все аспекты проблемы и глубже вникнуть в суть задачи.

# **1.2 Определение иерархической структуры**

Иерархическая структура – это графическое представление проблемы в виде перевернутого дерева, где каждый элемент, за исключением самого верхнего, зависит от одного или более выше расположенных элементов. Часто в различных организациях распределение полномочий, руководство и эффективные коммуникации между сотрудниками организованы в иерархической форме.

Иерархические структуры используются для лучшего понимания сложной реальности: мы раскладываем исследуемую проблему на составные части; затем разбиваем на составные части получившиеся элементы и т.д. На каждом шаге важно фокусировать внимание на понимании текущего элемента, временно абстрагируясь от всех прочих компонентов. При проведении подобного анализа приходит понимание всей сложности и многогранности исследуемого предмета.

В качестве примера можно привести иерархическую структуру, которая используется при обучении в медицинских вузах. В рамках изучения анатомии отдельно рассматривается костно-мышечная система (которая включает такие элементы, как руки и их составляющие: мышцы и кости), сердечнососудистая система (и ее множественные уровни), нервная система (и ее компоненты и подсистемы) и т.д. Степень детализации доходит до клеточного и молекулярного уровня. В конце изучения приходит понимание системы организма в целом, а также осознание того, какую роль играет в нем занимает каждая часть С помощью подобного иерархического структурирования студенты приобретают всесторонние знания об анатомии.

Аналогичным образом, когда мы решаем сложную проблему, мы можем использовать иерархию как инструмент для обработки и восприятия больших объемов информации. По мере проектирования этой структуры у нас формируется все более полное понимание проблемы.

Простейшая иерархия МАИ. Чтобы избежать беспорядка в диаграммах МАИ, связи, соединяющие Альтернативы и их покрывающие Критерии, часто опускаются, или их количество искусственно уменьшается. Несмотря на такие упрощения в диаграмме, в самой иерархии каждая Альтернатива связана с каждым из покрывающих ее Критериев.

# **1.3 Объяснение иерархических структур, используемых в МАИ**

Иерархические структуры, используемые в МАИ, представляет собой инструмент для качественного моделирования сложных проблем. Вершиной иерархии является главная цель; элементы нижнего уровня представляют множество вариантов достижения цели (альтернатив); элементы промежуточных уровней соответствуют критериям или факторам, которые связывают цель с альтернативами. Существуют специальные термины для описания иерархической структуры МАИ. Каждый уровень состоит из узлов. Элементы, исходящие из узла, принято называть его детьми (дочерними элементами). Элементы, из которых исходит узел, называются родительскими. Группы элементов, имеющие один и тот же родительский элемент, называются группами сравнения. Родительские элементы Альтернатив, как правило, исходящие из различных групп сравнения, называются покрывающими Критериями. Используя эти термины для описания представленной ниже диаграммы, можно сказать, что четыре Критерия – это дети Цели; в свою очередь, Цель — это родительский элемент для любого из Критериев. Каждая Альтернатива – это дочерний элемент каждого из включающих ее Критериев. Всего на диаграмме присутствует две группы сравнения: группа, состоящая из четырех Критериев и группа, включающая три Альтернативы. Вид любой иерархии МАИ будет зависеть не только от объективного характера рассматриваемой проблемы, но и от знаний, суждений, системы ценностей, мнений, желаний и т. п. участников процесса. Опубликованные описания применений МАИ часто включают в себя различные схемы и объяснения представленных иерархий. Последовательное выполнение всех шагов МАИ предусматривает возможность изменения структуры иерархии, с целью включения в неё вновь появившихся, или ранее не считавшихся важными, Критериев и Альтернатив.

# **1.4 Расстановка приоритетов**

После построения иерархии участники процесса используют МАИ для определения приоритетов всех узлов структуры. Информация для расстановки приоритетов собирается со всех участников и математически обрабатывается. В данном разделе приведена информация, на простом примере поясняющая процесс вычисления приоритетов.

# **1.5 Определение приоритетов и пояснения**

Приоритеты – это числа, которые связаны с узлами иерархии. Они представляют собой относительные веса элементов в каждой группе. Подобно вероятностям, приоритеты – безразмерные величины, которые могут принимать значения от нуля до единицы. Чем больше величина приоритета, тем более значимым является соответствующий ему элемент. Сумма приоритетов элементов, подчиненных одному элементу вышележащего уровня иерархии, равна единице. Приоритет цели по определению равен 1.0. Рассмотрим простой пример, поясняющий методику вычисления приоритетов.

Простейшая иерархическая структура МАИ с приоритетами, определенными по умолчанию.

На рисунке показана иерархия, в которой приоритеты всех элементов не устанавливались ЛПР. В таком случае по умолчанию приоритеты элементов считаются одинаковыми, т. е. все четыре критерия имеют равную важность с точки зрения цели, а приоритеты всех альтернатив равны по всем критериями. Другими словами, альтернативы в этом примере неразличимы. Заметим, что сумма приоритетов элементов любого уровня, равна единице. Если бы альтернатив было две, то их приоритеты были бы равны 0.500, если бы критериев было 5, то приоритет каждого был бы равен 0.200. В этом простом примере приоритеты альтернатив по разным критериям могут не совпадать, что обычно и бывает на практике. Приведем пример, в котором локальные приоритеты альтернатив по разным критериям не совпадают. Глобальные приоритеты альтернатив относительно цели вычисляются путем умножения локального приоритета каждой альтернативы на приоритет каждого критерия и суммирования по всем критериям.

Более сложная иерархическая структура, содержащая глобальные и локальные значения приоритетов по умолчанию.

Если приоритеты критериев изменятся, то изменятся значения глобальных приоритетов альтернатив, следовательно, может измениться их порядок. На рисунке показано решение данной задачи с изменившимися значениями приоритетов критериев, при этом наиболее предпочтительной альтернативой становится A3.

# **1.6 Практическое применение МАИ**

МАИ может успешно использоваться для решения простых задач, однако его эффективность проявляется при поиске решения сложных проблем, требующих системного подхода и привлечения большого числа экспертов. Задачи, для решения которых может быть применен МАИ:

Проблема многокритериального выбора. Выбор одной альтернативы из имеющегося набора альтернатив на основе некоторых критериев.

Ранжирование. Многокритериальное упорядочивание заданного множества альтернатив.

Определение приоритетов альтернатив и критериев в задачах многокритериального выбора.

Распределение ресурсов. Распределение ресурсов между альтернативами из заданного множества.

Сопоставительный анализ. Разработка рекомендаций по оптимизации внутренних процессов организации на основе успешного опыта конкурентов.

Управление качеством. Анализ различных аспектов качества и пути улучшения качества.

Можно привести множество примеров успешного применения МАИ для решения сложных проблем, в частности, следующие:

Выработка стратегии, направленной на уменьшение негативного влияния глобального изменения климата (Fondazione Eni Enrico Mattei).

Вычисление показателя совокупного качества программных комплексов (Microsoft Corporation).

Выбор специализации при учебе в университете (Bloomsburg University of Pennsylvania).

Принятие решения о месторасположении оффшорных предприятий (University of Cambridge).

Оценка рисков, связанных с функционированием нефтяных трубопроводов, пролегающих на территории страны (American Society of Civil Engineers).

Разработка стратегии наиболее эффективного управления водоразделами США (U.S. Department of Agriculture).

Сфера образования и научных исследований.

Хотя для практического применения МАИ отсутствует необходимость специальной подготовки, основы метода преподают во многих учебных заведениях. Кроме того, этот метод широко применяется в сфере управления качеством и читается в рамках многих специализированных программ, таких как Six Sigma, Lean Six Sigma, и QFD. Около ста китайских университетов предлагают курсы по основам МАИ, и многие соискатели научных степеней выбирают МАИ в качестве объекта научных и диссертационных исследований. Опубликовано более 900 научных статей по данной тематике. Существует китайский научный журнал, специализирующийся в области МАИ. Раз в два года проводится Международный симпозиум, посвященный МАИ (International Symposium on Analytic Hierarchy Process, ISAHP), на котором встречаются как ученые, так и практики, работающие с МАИ. В 2007 году симпозиум проходил в Вальпараисо, Чили, где было представлено более 90 докладов ученых из 19 стран, включая США, Германию, Японию, Чили, Малайзию, и Непал.

# **2 Использование Метода Анализа Иерархий**

# **2.1 Практическое применение Метода Анализа Иерархий**

Применение Метода Анализа Иерархий на практике вызывает иногда определенные трудности, так как требует зачастую значительного объема вычислений. Использование современных компьютерных информационных технологий позволяет автоматизировать решение многих оптимизационных задач (в том числе и задач, решаемых с помощью Метода Анализа Иерархий).

Для реализации метода наискорейшего спуска была разработана программа на языке С#, код которой содержится в Приложении А.

Она включает основные этапы реализации метода:

- ввод критериев

- ввод альтернатив

- сравнение по критериям

- сравнение по альтернативам

- исследование

- синтез альтернатив;

- построение графика альтернатив с приоритетами.

Листинг результатов работы программы представлен на рисунке 8.

Table

Description automatically generated

Рисунок 1 – Задание критериев и альтернатив

Table

Description automatically generated

Рисунок 2 – Результат оценки критериев

Table

Description automatically generated

Рисунок 3 – Результат оценки альтернатив по цене

Table

Description automatically generated

Рисунок 4 – Результат оценки альтернатив по удобству

Table

Description automatically generated

Рисунок 5 – Результат оценки альтернатив по матрице

Table

Description automatically generated

Рисунок 6 – Результат оценки альтернатив по стабилизации

Chart, pie chart

Description automatically generated

Рисунок 7 – Результат работы программы нахождения глобальных приоритетов

# **ЗАКЛЮЧЕНИЕ**

В ходе выполнения проекта был изучен один из методов многомерной оптимизации – Метод Анализа Иерархий.

В работе проведен анализ литературы, затрагивающей вопросы использования Метода Анализа Иерархий при решении оптимизационных задач, изучены теоретические вопросы, связанные с обоснованием данного метода.

Рассмотрены вопросы практического применения Метод Анализа Иерархий в экспертном анализе. На конкретных примерах и задачах рассмотрены особенности использования Метода Анализа Иерархий.

Разработано программное средство, позволяющее произвести экспертный анализ Методом Анализа Иерархий.

В процессе выполнения проекта были закреплены знания по дисциплине «Оптимизация проектных решений». Также получены практические навыки по решению управленческих задач.

При решении управленческих задач с помощью метода анализа иерархий можно самостоятельно производить расчеты или воспользоваться программным приложением. Однако в любом случае необходимы экспертные оценки для составления исходной матрицы попарных сравнений, на основании которой затем происходит расчет основных показателей.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Саати Т.Л. Принятие решений при зависимостях и обратных связях: Аналитические сети. – М.: Издательство ЛКИ, 2008. – 360 с.

2. Саати Т.Л. Целочисленные методы оптимизации и связанные с ними экстремальные проблемы. – М.: Мир, 1973. – 302 с.

3. Басакер Р.Г., Саати Т.Л. Конечные графы и сети. – М.: Наука, 1974. – 366с.

4. Саати Т.Л. Принятие решений. Метод анализа иерархий. – М.: Радио и связь, 1989. – 316 с.

5. Саати Т., Кернс К. Аналитическое планирование. Организация систем. – М.: Радио и связь, 1991. – 224 с.

6. Андрейчиков А.В., Андрейчикова О.Н. Анализ, синтез, планирование решений в экономике. – М.: Финансы и статистика, 2004. – 464 с.

7. Андрейчиков А. В., Терелянский П. В., Андрейчикова О. Н. Информационные технологии прогнозирования технических решений на основе иерархических моделей. – Волгоград : РПК «Политехник», 2004.

8. Терелянский П. В. Непараметрическая экспертиза объектов сложной структуры. – М. : Изд.-торг. корп. «Дашков и Ко», 2009. – 221 с

9. Кини Р.Л., Райфа Х. принятие решений при многих критериях: предпочтения и замещения. - М.: Радио и связь, 1981.

10. Макаров И.М. и др. Теория выбора и принятия решений. - М.: Наука, 1982.

# Приложение А

Код программы

Файл Form1.cs

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Windows.Forms;

namespace CourseWork

{

public partial class Form1 : Form

{

int columnIndex = 0;

int rowIndex = 0;

ComboBox scaleComboBox = new ComboBox();

public List<Criterion> criterionsList;

public List<string> alternativesList;

Criterion allCriterionsComparing;

public List<double> WiList;

public Form1(List<Criterion> criterionsFromElements, List<string> alternativesFromElements)

{

InitializeComponent();

dataGridView1.ColumnHeadersHeightSizeMode = DataGridViewColumnHeadersHeightSizeMode.DisableResizing;

scaleComboBox.DropDownStyle = ComboBoxStyle.DropDownList;

criterionsList = new List<Criterion>();

alternativesList = new List<string>();

criterionsList.AddRange(criterionsFromElements.ToArray());

alternativesList.AddRange(alternativesFromElements.ToArray());

allCriterionsComparing = new Criterion("Object Criterions", criterionsList.Count);

FillDataGrid();

foreach(Criterion criteria in criterionsList)

{

criterionsComboBox.Items.Add(criteria.Name);

}

}

private void FillDataGrid()

{

dataGridView1.Rows.Clear();

dataGridView1.Columns.Clear();

dataGridView1.Columns.Add("Colunm0", "");

dataGridView1.Columns[0].Width = 30;

dataGridView1.Columns[0].SortMode = DataGridViewColumnSortMode.NotSortable;

for (int i = 1; i < criterionsList.Count + 1; i++)

{

dataGridView1.Columns.Add("Сolumn" + i.ToString(), "");

dataGridView1.Columns[i].AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;

dataGridView1.Columns[i].SortMode = DataGridViewColumnSortMode.NotSortable;

dataGridView1.Rows.Add();

dataGridView1.Rows[i - 1].Cells[0].ReadOnly = true;

}

ChangeDataGridHeaderText(elementComboBox.SelectedIndex);

DataGridValueLoad(allCriterionsComparing);

elementComboBox.SelectedIndex = 0;

}

private void DataGridValueLoad(Criterion element)

{

for (int i = 0; i < dataGridView1.Rows.Count; i++)

{

for (int j = 1; j < dataGridView1.Columns.Count; j++)

{

if (element.Matrix[j - 1, i].Equals("0"))

{

dataGridView1.Rows[i].Cells[j].Value = "";

}

else

{

dataGridView1.Rows[i].Cells[j].Value = element.Matrix[j - 1, i];

}

}

}

isTextBox.Text = element.IS.ToString();

osTextBox.Text = element.OS.ToString();

resultTextBox.Text = "";

}

private void ChangeDataGridHeaderText(int index)

{

if (index > 0)

{

for (int i = 1; i < criterionsList.Count + 1; i++)

{

dataGridView1.Columns[i].HeaderText = "A" + i.ToString();

dataGridView1.Rows[i - 1].Cells[0].Value = "A" + i.ToString();

label1.Visible = true;

criterionsComboBox.Visible = true;

criterionsComboBox.SelectedIndex = 0;

}

else if (index == 0)

{

for (int i = 1; i < criterionsList.Count + 1; i++)

{

dataGridView1.Columns[i].HeaderText = "K" + i.ToString();

dataGridView1.Rows[i - 1].Cells[0].Value = "K" + i.ToString();

}

label1.Visible = false;

criterionsComboBox.Visible = false;

}

}

private void Form1\_Load(object sender, EventArgs e)

{

scaleComboBox.Visible = false;

string[] arrayItem = new string[17];

for (int i = 0; i < 10; i++)

{

arrayItem[i] = (9 - i).ToString();

}

for (int i = 2; i < 10; i++)

{

arrayItem[i + 7] = ("1/" + i).ToString();

}

scaleComboBox.Items.AddRange(arrayItem);

scaleComboBox.SelectedValueChanged += scaleComboBox\_SelectedValueChanged;

dataGridView1.Controls.Add(scaleComboBox);

}

private void scaleComboBox\_SelectedValueChanged(object sender, EventArgs e)

{

if (scaleComboBox.SelectedIndex == -1)

{

return;

}

if (elementComboBox.SelectedIndex == 0)

{

allCriterionsComparing.flag = false;

}

else

{

criterionsList[criterionsComboBox.SelectedIndex].flag = false;

}

string newValue1 = scaleComboBox.SelectedItem.ToString();

string newValue2 = scaleComboBox.Items[16 - scaleComboBox.SelectedIndex].ToString();

dataGridView1[columnIndex, rowIndex].Value = newValue1;

dataGridView1[rowIndex + 1, columnIndex - 1].Value = newValue2;

scaleComboBox.Visible = false;

if (elementComboBox.SelectedIndex == 0)

{

allCriterionsComparing.UpdateMatrix(columnIndex - 1, rowIndex, newValue1, newValue2);

}

else

{

criterionsList[criterionsComboBox.SelectedIndex].UpdateMatrix(

columnIndex - 1, rowIndex, newValue1, newValue2);

}

}

private void dataGridView1\_CellClick(object sender, DataGridViewCellEventArgs e)

{

rowIndex = e.RowIndex;

columnIndex = e.ColumnIndex;

if (columnIndex == 0 || (columnIndex - 1) == rowIndex)

{

return;

}

//Получаем прямоугольник ячейки

Rectangle rectangle = dataGridView1.GetCellDisplayRectangle(columnIndex, rowIndex, true);

//Задаем размеры и месторасположение

scaleComboBox.Size = new Size(rectangle.Width, rectangle.Height);

scaleComboBox.Location = new Point(rectangle.X, rectangle.Y);

//Меняем индекс на изначальный (чтобы значение предыдущей ячейки не тянулось в следующие)

scaleComboBox.SelectedIndex = -1;

scaleComboBox.Visible = true;

}

private void dataGridView1\_ColumnHeaderMouseClick(object sender, DataGridViewCellMouseEventArgs e)

{

scaleComboBox.Visible = false;

}

private void criterionsComboBox\_SelectedIndexChanged(object sender, EventArgs e)

{

DataGridValueLoad(criterionsList[criterionsComboBox.SelectedIndex]);

dataGridView1.Enabled = true;

}

private void elementComboBox\_SelectedIndexChanged(object sender, EventArgs e)

{

ChangeDataGridHeaderText(elementComboBox.SelectedIndex);

if (elementComboBox.SelectedIndex == 0)

{

DataGridValueLoad(allCriterionsComparing);

}

else

{

DataGridValueLoad(criterionsList[criterionsComboBox.SelectedIndex]);

}

}

private void exploreButton\_Click(object sender, EventArgs e)

{

Criterion element;

if (elementComboBox.SelectedIndex == 0)

{

element = allCriterionsComparing;

}

else

{

element = criterionsList[criterionsComboBox.SelectedIndex];

}

if (!element.ExploreMatrix())

{

MessageBox.Show("Введите все значения!", "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

else

{

isTextBox.Text = element.IS.ToString();

osTextBox.Text = element.OS.ToString();

if (element.OS < 0.1)

{

resultTextBox.Text = "Матрица согласована, т.к. ОС < 0.1";

}

else

{

resultTextBox.Text = "Матрица не согласована, т.к. ОС > 0.1";

}

}

if (elementComboBox.SelectedIndex == 0)

{

allCriterionsComparing.flag = true;

}

else

{

criterionsList[criterionsComboBox.SelectedIndex].flag = true;

}

}

private void resultButton\_Click(object sender, EventArgs e)

{

if(!allCriterionsComparing.CheckExploring())

{

return;

}

foreach(Criterion element in criterionsList)

{

if(!element.CheckExploring())

{

return;

}

}

WiList = new List<double>();

double W = 0;

try

{

for (int i = 0; i < criterionsList.Count; i++)

{

for (int j = 0; j < criterionsList.Count; j++)

{

W += allCriterionsComparing.W\_i[j] \* criterionsList[j].W\_i[i];

}

WiList.Add(Math.Round(W, 4));

W = 0;

}

}

catch (Exception exc)

{

MessageBox.Show(exc.ToString(), "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

ResultForm resultForm = new ResultForm();

resultForm.Owner = this;

resultForm.Show();

}

}

}

Файл Elements.cs

using System;

using System.Collections.Generic;

using System.Windows.Forms;

namespace CourseWork

{

public partial class Elements : Form

{

Form1 mainForm;

List<Criterion> criterionsList;

List<string> alternativesList;

public Elements()

{

InitializeComponent();

criterionsList = new List<Criterion>();

alternativesList = new List<string>();

}

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

AddRows(CriterionsDataGridView, ElementsCountComboBox.SelectedIndex);

AddRows(AlternativesDataGridView, ElementsCountComboBox.SelectedIndex);

}

private void AddRows(DataGridView dataGridView, int count)

{

dataGridView.Rows.Clear();

dataGridView.Rows.Add(count + 2);

foreach(DataGridViewRow row in dataGridView.Rows)

{

row.Cells[0].Value = row.Index + 1;

}

}

private void OkButton\_Click(object sender, EventArgs e)

{

if (!CreateNewElements())

{

return;

}

mainForm = new Form1(criterionsList, alternativesList);

mainForm.ShowDialog();

}

private bool CreateNewElements()

{

foreach (DataGridViewRow row in CriterionsDataGridView.Rows)

{

if (!CheckInput(row))

{

return false;

}

Criterion criterion = new Criterion(row.Cells[1].Value.ToString(), CriterionsDataGridView.Rows.Count);

criterionsList.Add(criterion);

}

foreach (DataGridViewRow row in AlternativesDataGridView.Rows)

{

if (!CheckInput(row))

{

return false;

}

alternativesList.Add(row.Cells[1].Value.ToString());

}

return true;

}

private bool CheckInput(DataGridViewRow dataGridViewRow)

{

if (dataGridViewRow.Cells[1].Value == null)

{

MessageBox.Show("Введите все названия элементов!", "Внимание!", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return false;

}

return true;

}

}

}