

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждениевысшего образования

**"МИРЭА - Российский технологический университет" РТУ МИРЭА**

**Институт** Информационных Технологий **Кафедра** Вычислительной Техники

**ПРАКТИЧЕСКАЯ РАБОТА №3**

**по дисциплине**

**«Теория принятия решений» Метод анализа иерархий**

Студент группы:ИКБО-04-22 Карева А.А.

*(Ф. И.О. студента)*

Преподаватель Железняк Л.М.

*(Ф.И.О. преподавателя)*

Москва 2024

**СОДЕРЖАНИЕ**

[1 МЕТОД АНАЛИЗА ИЕРАРХИЙ 4](#_Toc80594)

[1.1 Постановка задачи 4](#_Toc80595)

[1.2 Представление проблемы в виде иерархии 4](#_Toc80596)

[1.3 Установка приоритетов критериев 5](#_Toc80597)

[1.4 Синтез приоритетов 6](#_Toc80598)

[1.5 Согласованность локальных приоритетов 14](#_Toc80599)

[3.6 Синтез альтернатив 21](#_Toc80600)

[3.7 Вывод 22](#_Toc80601)

[1.8 Результаты работы программы 23](#_Toc80602)

**ВВЕДЕНИЕ**

Метод Анализа Иерархий (Analitic Hierarchy Process) разработан американским математиком Томасом Саати в 70-х годах прошлого века. Метод анализа иерархий (МАИ) является замкнутой логической конструкцией, которая обеспечивает с помощью простых и хорошо обоснованных правил, решение задач МКО, включающих как качественные, так и количественные факторы, причем количественные факторы могут иметь разную размерность. Метод основан на декомпозиции задачи и представлении ее в виде иерархической структуры, что позволяет включить в иерархию все имеющиеся у лица, принимающего решение знания по решаемой проблеме и последующей обработке суждений. В результате может быть выявлена относительная степень взаимодействия элементов в иерархии, которые затем выражаются численно. МАИ включает процедуры синтеза множественных суждений, получения приоритетности критериев и нахождения альтернативных решений. Весь процесс решения подвергается проверке и переосмыслению на каждом этапе, что позволяет проводить оценку качества полученного решения. Решение многокритериального выбора основано на трех основных этапах: Первый этап – представление системы критериев (целей) в виде иерархической структуры. Второй этап – оценки приоритетов (весов) критериев с учётом их места в иерархии относительной важности. Третий этап – определение лучшей альтернативы по значениям её характеристик и важности критериев. Первый этап – предусматривает представление проблемы в виде иерархии. Иерархия – некоторая абстракция структуры системы, предназначенная для изучения функциональных взаимодействий ее компонент и их воздействий на систему в целом.

# 1 МЕТОД АНАЛИЗА ИЕРАРХИЙ

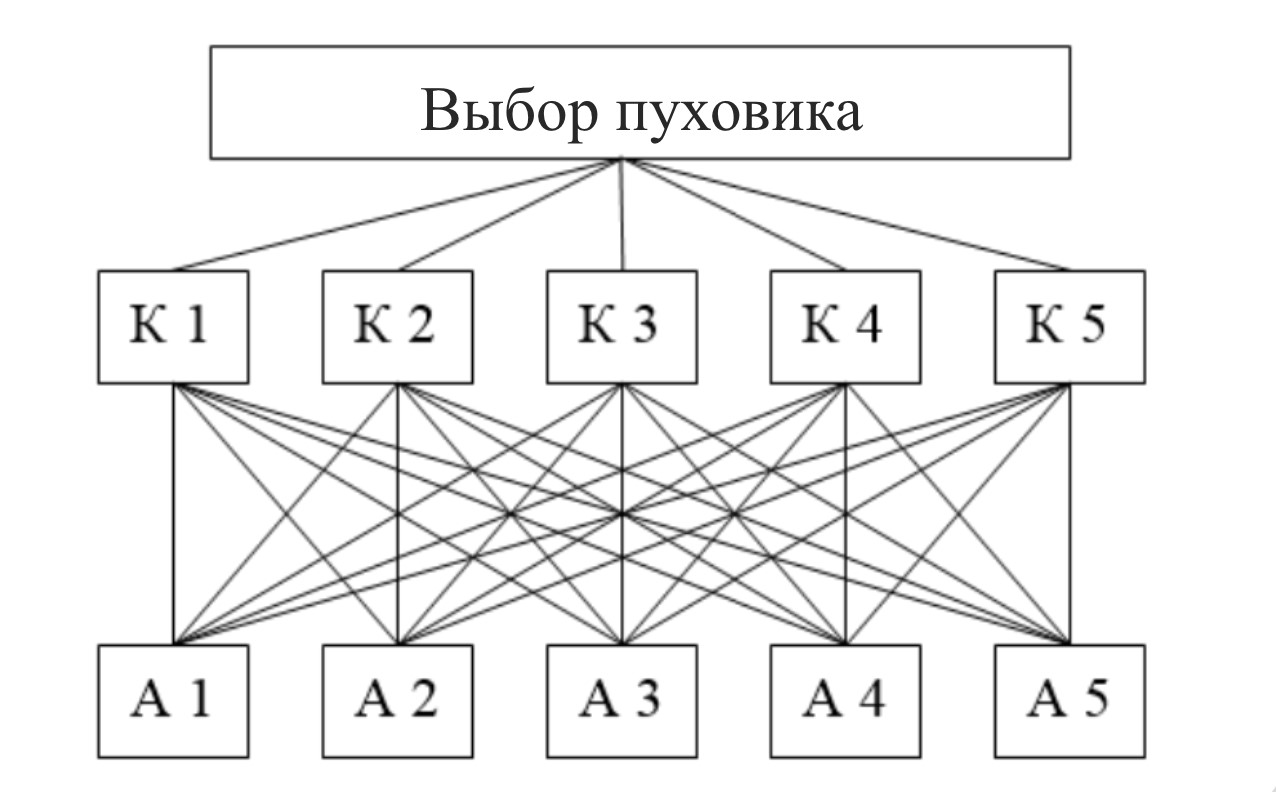
## 1.1 Постановка задачи

Задача практической работы: выбрать своя предметная область.

## 1.2 Представление проблемы в виде иерархии

Первый этап – представление проблемы в виде иерархии или сети. В простейшем случае, иерархия строится, начиная с цели, которая помещается в вершину иерархии. Через промежуточные уровни, на которых располагаются критерии и от которых зависят последующие уровни, к самому низкому уровню, который содержит перечень альтернатив.

Иерархия считается полной, если каждый элемент заданного уровня является критерием для всех элементов нижнего уровня



**Рисунок 1 – Полная доминантная иерархия.**

Критерии:

К 1 – Средний чек;

К 2 – Рейтинг;

К 3 – Количество услуг;

К 4 – Удалённость локации;

Альтернативы:

А 1 - Метод;

А 2 -FIDEL;

А 3 - БородаВайб;

А 4 – Бритва;

А 5 – BomboKlak.

## 1.3 Установка приоритетов критериев

После иерархического представления задачи установлены приоритеты критериев и оценена каждая из альтернатив по критериям, определена наиболее важная их них. В методе анализа иерархий элементы сравниваются попарно по отношению к их влиянию на общую для них характеристику. Парные сравнения приводят к записи характеристик сравнений в виде квадратной таблицы чисел, которая называется матрицей. Для облегчения работы введена шкала относительной важности (Таблица 1).

*Таблица.1 – Шкала относительной важности.*

|  |  |  |
| --- | --- | --- |
| Интенсивность относительной важности | Определение | Объяснение |
| 1 | Равная важность | Равный вклад двух критериев в цель. |
| 3 | Слабое превосходство | Дают легкое превосходство одной альтернативы над  другой |
| 5 | Умеренное превосходство | Опыт и суждения дают умеренное превосходство |
| 7 | Сильное превосходство | Одному из критериев дается настолько сильное предпочтение. |
| 9 | Абсолютное превосходство | Очевидность превосходства одного критерия над другим |
| 2,4,6,8 | Промежуточные решения между двумя соседними суждениями | Применяется в компромиссных случаях |

Шкала содержит соответствующие обратные значения.

## 1.4 Синтез приоритетов

После построения иерархии и определения величин парных субъективных суждений следует этап, на котором иерархическая декомпозиция и относительные суждения объединяются для получения осмысленного решения многокритериальной задачи принятия решений. Из групп парных сравнений формируется набор локальных критериев, которые выражают относительное влияние элементов на элемент, расположенный на уровне выше. Составлена обратно симметричная матрица для парного сравнения критериев (Таблица 2).

*Таблица 2 – Матрица парного сравнения критериев.*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Цель | К 1 | К 2 | К 3 | К 4 | К 5 | Vi | W2i |
| К 1 | 1 | 3 | 3 | 5 | 7 | 3.1598 | 0,4453 |
| К 2 | 0.3333 | 1 | 3 | 5 | 7 | 2.0361 | 0,2869 |
| К 3 | 0.3333 | 0.3333 | 1 | 3 | 5 | 1.1075 | 0,1561 |
| К 4 | 0.2 | 0.2 | 0.3333 | 1 | 3 | 0.5253 | 0,0740 |
| К 5 | 0.1429 | 0.1429 | 0.2 | 0.3333 | 1 | 0.2672 | 0,0377 |
|  |  | ∑Vi | |  |  | 7,0959 |  |

Для определения относительной ценности каждого элемента необходимо найти геометрическое среднее и с этой целью перемножить n элементов каждой строки и из полученного результата извлечь корни n-й степени (размерность матрицы n=5).

Строка № 1

V1=(1x3x3x5x7)1/5 = 3.1598 ;

Строка № 2

V2=(1/3x1x3x5x7)1/5 = 2.0361;

Строка № 3

V3=(1/3x1/3x1x3x5)1/5 = 1.1075;

Строка № 4

V4=(1/5x1/5x1/3x1x3)1/5 = 0.5253;

Строка № 5

V5=(1/7x1/7x1/5x1/3x1)1/5 = 0.2672.

Проведена нормализация полученных чисел. Для этого определен нормирующий коэффициент ∑Vi.

∑Vi = V1 + V2 + V3 + V4 + V5 = 3.1598 + 2.0361 + 1.1075 + 0.5253 + 0.2672 + = 7,0959.

Найдена важность приоритетов W2i, для этого каждое из чисел Vi разделено на ∑Vi.

***Строка № 1***

W21= 3.1598 / ∑Vi =0,4453;

Строка № 2

W22= 2.0361 / ∑Vi = 0,2869;

***Строка № 3***

W23= 1.1075 / ∑Vi = 0,1561;

***Строка № 4***

W24= 0.5253 / ∑Vi = 0,074;

***Строка № 5***

W25= 0.2672 / ∑Vi = 0,0377.

В результате получен вектор приоритетов:

W2i = (0,4453; 0,2869; 0,1561; 0,074; 0,0377), где индекс 2 означает, что вектор приоритетов относится ко второму уровню иерархии. К 1 – % пуха (Таблица 3);

*Таблица 3 – Матрица сравнения по критерию 1.*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| К1 | А1 | А2 | А3 | А4 | А5 | VК1Y | W3К1Y |
| А1 | 1 | 9 | 9 | 1 | 1 | 2.4082 | 0,3087 |
| А2 | 0.1111 | 1 | 0.1429 | 0.1111 | 0.1111 | 0.1813 | 0,0232 |
| А3 | 0.1111 | 7 | 1 | 0.1111 | 0.1111 | 0.3949 | 0,0506 |
| А4 | 1 | 9 | 9 | 1 | 1 | 2.4082 | 0,3087 |
| А5 | 1 | 9 | 9 | 1 | 1 | 2.4082 | 0,3087 |
|  |  | ∑VК1Y | |  |  | 7.8008 |  |

Определена относительная ценность каждого элемента.

Строка № 1

VК11=(1x9x9x1x1)1/5= 2.4082;

Строка № 2

VК12=(1/9x1x1/7x1/9x1/9)1/5= 0.1813;

Строка № 3

VК13=(1/9x7x1x1/9x1/9)1/5= 0.3949;

Строка № 4

VК14=(1x9x9x1x1)1/5= 2.4082; Строка № 5

VК15=(1x9x9x1x1)1/5= 2.4082.

Проведена нормализация полученных чисел. Для этого определен нормирующий коэффициент ∑VK1Y.

∑VК1Y = VК11 + VК12 + VК13 + VК14 + VК15 = 7.8008.

Найдена важность приоритетов W3К1Y, для этого каждое из чисел VK1Y разделено на ∑VK1Y.

Строка № 1

W3К11= 2.4082 / ∑Vi = 0,3087; Строка № 2

W3К12= 0.1813 / ∑Vi = 0,0232;

Строка № 3

W3К13= 0.3949 / ∑Vi = 0,0506;

Строка № 4

W3К14= 2.4082 / ∑Vi = 0,3087;

Строка № 5

W3К15= 2.4082 / ∑Vi = 0,3087;

В результате получаем вектор приоритетов:

W3К1Y = (0,3087; 0,0232; 0,0506; 0,3087; 0,3087),

где индекс 3 означает, что вектор приоритетов относится к третьему

уровню иерархии критерия К1.

К 2 – цена (Таблица 3.4.3);

*Таблица 3.4.3 – Матрица сравнения по критерию 2.*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| К2 | А1 | А2 |  | А3 | А4 | А5 | VК2Y | W3К2Y |
| А1 | 1 | 0.3333 |  | 7 | 2 | 0.1111 | 0.8769 | 0,0996 |
| А2 | 3 | 1 |  | 7 | 6 | 0.3333 | 2.5017 | 0,2842 |
| А3 | 0.1429 | 0.1429 |  | 1 | 0.1429 | 0.1111 | 0.1901 | 0,0216 |
| А4 | 0.5 | 0.1667 |  | 7 | 1 | 0.1111 | 0.5786 | 0,0657 |
| А5 | 9 | 3 |  | 9 | 9 | 1 | 4.6555 | 0,5289 |
|  |  | ∑VК2Y | |  |  | | 8.8028 |  |

Определена относительная ценность каждого элемента.

Строка № 1

VК21=(1x1/3x7x2x1/9)1/5= 0.8769;

Строка № 2

VК22=(3x1x7x6x1/3)1/5= 2.5017;

Строка № 3

VК23=(1/7x1/7x1x1/7x1/9)1/5= 0.1901;

Строка № 4

VК24=(1/2x1/6x7x1x1/9)1/5= 0.5786;

Строка № 5

VК25=(9x3x9x9x1)1/5= 4.6555.

Проведена нормализация полученных чисел. Для этого определен нормирующий коэффициент ∑VK2Y.

∑VК2Y = VК21 + VК22 + VК23 + VК24 + VК25 = 8.8028.

Найдена важность приоритетов W3К2Y, для этого каждое из чисел VK2Y разделено на ∑VK2Y.

Строка № 1

W3К21= 0.8769 / ∑Vi = 0,0996;

Строка № 2

W3К22= 2.5017 / ∑Vi = 0,2842;

Строка № 3

W3К23= 0.1901 / ∑Vi = 0,0216;

Строка № 4

W3К24= 0.5786 / ∑Vi = 0,0657; Строка № 5

W3К25= 4.6555 / ∑Vi = 0,5289.

В результате получаем вектор приоритетов:

W3К2Y = (0,0996; 0,2842; 0,0216; 0,0657; 0,5289),

где индекс 3 означает, что вектор приоритетов относится к третьему

уровню иерархии критерия К2.

К 3 – сочетаемость (Таблица 3.4.4);

*Таблица 3.4.4 – Матрица сравнения по критерию 3.*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| К3 | А1 | А2 | А3 | А4 | А5 | VК3Y | W3К3Y |
| А1 | 1 | 2 | 1 | 1 | 0.2 | 0.8326 | 0,1262 |
| А2 | 0.5 | 1 | 0.5 | 0.5 | 0.2 | 0.4782 | 0,0725 |
| А3 | 1 | 2 | 1 | 1 | 0.2 | 0.8326 | 0,1262 |
| А4 | 1 | 2 | 1 | 1 | 0.2 | 0.8326 | 0,1262 |
| А5 | 5 | 5 | 5 | 5 | 1 | 3.6239 | 0,5491 |
|  |  | ∑VК35 | |  |  | 6.5999 |  |

Определена относительная ценность каждого элемента.

Строка № 1

VК31=(1x2x1x1x0.2)1/5= 0.8326;

Строка № 2

VК32=(1/2x1x1/2x1/5)1/5= 0.4782;

Строка № 3

VК33=(1x2x1x1x0.2)1/5= 0.8326;

Строка № 4

VК34=(1x2x1x1x0.2)1/5= 0.8326;

Строка № 5

VК35=(5x5x5x5x1)1/5= 3.6239 .

Проведена нормализация полученных чисел. Для этого определен нормирующий коэффициент ∑VK3Y.

∑VК3Y = VК31 + VК32 + VК33 + VК34 + VК35 = 6.5999.

Найдена важность приоритетов W3К2Y, для этого каждое из чисел VK2Y разделено на ∑VK2Y.

Строка № 1

W3К31= 0.8326 / ∑Vi = 0,1262;

Строка № 2

W3К32= 0.4782 / ∑Vi = 0,0725;

Строка № 3

W3К33= 0.8326 / ∑Vi = 0,1262;

Строка № 4

W3К34= 0.8326 / ∑Vi = 0,1262; Строка № 5

W3К35= 3.6239 / ∑Vi = 0,5491.

В результате получаем вектор приоритетов:

W3К3Y = (0,1262; 0,0725; 0,1262; 0,1262; 0,5491),

где индекс 3 означает, что вектор приоритетов относится к третьему

уровню иерархии критерия К3.

К 4 – длина рукава (Таблица 3.4.5);

*Таблица 3.4.5 – Матрица сравнения по критерию 4.*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| К4 | А1 | А2 | А3 | А4 | А5 | VК4Y | W3К4Y |
| А1 | 1 | 7 | 5 | 2 | 0.3333 | 1.8775 | 0,2447 |
| А2 | 0.1429 | 1 | 0.5 | 0.1667 | 0.1111 | 0.2657 | 0,0346 |
| А3 | 0.2 | 2 | 1 | 0.25 | 0.125 | 0.4163 | 0,0543 |
| А4 | 0.5 | 6 | 4 | 1 | 0.25 | 1.2457 | 0,1624 |
| А5 | 3 | 9 | 8 | 4 | 1 | 3.8664 | 0,5040 |
|  |  | ∑VК4Y | |  |  | 7.6716 |  |

Определена относительная ценность каждого элемента.

Строка № 1

VК41=(1x7x5x2x1/3)1/5 = 1.8775;

Строка № 2

VК42=(1/7x1x1/2x1/6x1/9)1/5 = 0.2657;

Строка № 3

VК43=(1/5x2x1x1/4x1/8)1/5 = 0.4163;

Строка № 4

VК44=(1/2x6x4x1x1/4)1/5 = 1.2457;

Строка № 5

VК45=(3x9x8x4x1)1/5 = 3.8664.

Проведена нормализация полученных чисел. Для этого определен нормирующий коэффициент ∑VK4Y.

∑VК4Y = VК41 + VК42 + VК43 + VК44 + VК45 = 7.6716.

Найдена важность приоритетов W3К4Y, для этого каждое из чисел VK4Y разделено на ∑VK4Y.

Строка № 1

W3К41= 1.8775 / ∑Vi = 0,0346;

Строка № 2

W3К42= 0.2657 / ∑Vi = 0,0346;

Строка № 3

W3К43= 0.4163 / ∑Vi = 0,0543;

Строка № 4

W3К44= 1.2457 / ∑Vi = 0,1624;

Строка № 5

W3К45= 3.8664 / ∑Vi = 0,504.

В результате получаем вектор приоритетов:

W3К4Y = (0,0346; 0,0346; 0,0543; 0,1624; 0,504),

где индекс 3 означает, что вектор приоритетов относится к третьему

уровню иерархии критерия К4.

К 5 – длина (Таблица 3.4.6).

*Таблица 3.4.6 – Матрица сравнения по критерию 5.*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| К5 | А1 | А2 | А3 | | А4 | А5 | VК5Y | W3К5Y |
| А1 | 1 | 7 | 6 | | 4 | 0.5 | 2.4258 | 0,3328 |
| А2 | 0.1429 | 1 |  | 0.5 | 0.25 | 0.1429 | 0.3030 | 0,0416 |
| А3 | 0.1667 | 2 |  | 1 | 0.5 | 0.1667 | 0.4884 | 0,0670 |
| А4 | 0.25 | 4 |  | 2 | 1 | 0.25 | 0.8706 | 0,1194 |
| А5 | 2 | 7 |  | 6 | 4 | 1 | 3.2009 | 0,4392 |
|  |  | ∑VК5Y | |  |  |  | 7.2887 |  |

Определена относительная ценность каждого элемента.

Строка № 1

VК51=(1x7x6x4x1/2)1/5 = 2.4258;

Строка № 2

VК52=(1/7x1x1/2x1/4x1/7)1/5 = 0.3030;

Строка № 3

VК53=(1/6x2x1x1/2x1/6)1/5 = 0.4884;

Строка № 4

VК54=(1/4x4x2x1x1/4)1/5 = 0.8706;

Строка № 5

VК55=(2x7x6x4x1)1/5 = 3.2009.

Проведена нормализация полученных чисел. Для этого определен нормирующий коэффициент ∑VK5Y.

∑VК5Y = VК51 + VК52 + VК53 + VК54 + VК55 = 7.2887.

Найдена важность приоритетов W3К5Y, для этого каждое из чисел VK5Y разделено на ∑VK5Y.

Строка № 1

W3К51= 2.4258 / ∑Vi = 0,3328;

Строка № 2

W3К52= 0.3030 / ∑Vi = 0,0416;

Строка № 3

W3К53= 0.4884 / ∑Vi = 0,067;

Строка № 4

W3К54= 0.8706 / ∑Vi = 0,1194;

Строка № 5

W3К55= 3.2009 / ∑Vi = 0,4392.

В результате получаем вектор приоритетов:

W3К5Y = (0,3328; 0,0416; 0,067; 0,1194; 0,4392),

где индекс 3 означает, что вектор приоритетов относится к третьему

уровню иерархии критерия К5.

## 1.5 Согласованность локальных приоритетов

Любая матрица суждений в общем случае не согласована, так как суждения отражают субъективные мнения ЛПР, а сравнение элементов, которые имеют количественные эквиваленты, может быть несогласованным из-за присутствия погрешности при проведении измерений. Совершенной согласованности парных сравнений даже в идеальном случае на практике достичь трудно. Нужен способ оценки степени согласованности при решении конкретной задачи.

Метод анализа иерархий дает возможность провести такую оценку.

Вместе с матрицей парных сравнений есть мера оценки степени отклонения от согласованности. Когда такие отклонения превышают установленные пределы тем, кто проводит решение задачи, необходимо их пересмотреть.

В таблице приведены средние значения индекса случайной согласованности (СИ) для случайных матриц суждений разного порядка.

В нашей задаче размерность матрицы n=5, тогда среднее значение индекса случайной согласованности СИ = 1,12.

Определены индекс согласованности и отношение согласованности для матрицы «Выбор лучшего пуховика» (Таблица 3.5.1).

*Таблица 3.5.1 – Матрица «Выбор лучшего пуховика».*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Цель | К 1 | К 2 | К 3 | К 4 | К 5 | W2i |
| К 1 | 1 | 3 | 3 | 5 | 7 | 0,4453 |
| К 2 | 0.3333 | 1 | 3 | 5 | 7 | 0,2869 |
| К 3 | 0.3333 | 0.3333 | 1 | 3 | 5 | 0,1561 |
| К 4 | 0.2 | 0.2 | 0.3333 | 1 | 3 | 0,0740 |
| К 5 | 0.1429 | 0.1429 | 0.2 | 0.3333 | 1 | 0,0377 |

Определена сумма каждого столбца матрицы суждений.

S1 = 1 + 1/3 + 1/3 + 1/5 + 1/7 = 2.0095;

S2 = 3 + 1 + 1/3 + 1/5 + 1/7 = 4.6765;

S3 = 3 + 3 + 1 + 1/3 + 1/5 = 7.5339;

S4 = 5 + 5 + 3 + 1 + 1/3 = 14.3336; S5 = 7 + 7 + 5 + 3 + 1 = 22.9961.

Полученный результат умножен на компоненту нормализованного вектора приоритетов, т.е. сумму суждений первого столбца на первую компоненту, сумму суждений второго столбца - на вторую и т.д.

Р1 = S1 х W21 = 0.8948;

Р2 = S2 х W22 = 1.3417;

Р3 = S3 х W23 = 1.1760;

Р4 = S4 х W24 = 1.0607; Р5 = S5 х W25 = 0.8670.

Сумма чисел Рj отражает пропорциональность предпочтений, чем ближе эта величина к n (числу объектов и видов действия в матрице парных сравнений), тем более согласованны суждения.

λmax = Р1 + Р2 + Р3 + Р4 + Р5 = 5.3402.

Отклонение от согласованности выражается индексом согласованности.

ИС = (λmax - n)/(n - 1) = 1,3608.

Отношение индекса согласованности ИС к среднему значению случайного индекса согласованности СИ называется отношением согласованности ОС.

ОС = ИС/СИ = 0.0759.

Значение ОС меньше или равное 0.10 считается приемлемым, значит матрица «Выбор лучшего спортсмена» согласована.

Определнены индекс согласованности К 1 – % пуха (Таблица 3.5.2).

*Таблица 3.5.2 – Матрица сравнения по критерию 1.*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| К1 | А1 | А2 | А3 | А4 | А5 | W3К1Y |
| А1 | 1 | 9 | 9 | 1 | 1 | 0,3087 |
| А2 | 0.1111 | 1 | 0.1429 | 0.1111 | 0.1111 | 0,0232 |
| А3 | 0.1111 | 7 | 1 | 0.1111 | 0.1111 | 0,0506 |
| А4 | 1 | 9 | 9 | 1 | 1 | 0,3087 |
| А5 | 1 | 9 | 9 | 1 | 1 | 0,3087 |

Определяется сумма каждого столбца матрицы суждений.

S1 К1 = 3.2222;

S2 К1 = 35.0006;

S3 К1 = 28.1456;

S4 К1 = 3.2222;

S5 К1 = 3.2222.

Затем полученный результат умножен на компоненту нормализованного вектора приоритетов.

Р1 К1 = S1 х W3К11 = 0.9947;

Р2 К1 = S2 х W3К12 = 0.8120;

Р3 К1 = S3 х W3К13 = 1.4242;

Р4 К1 = S1 х W3К14 = 0.9947;

Р5 К1 = S1 х W3К15 = 0.9947.

Найдена пропорциональность предпочтений.

λmax К1 = Р1К1 + Р2К1 + Р3К1 + Р4К1 + Р5К1 = 5.2203.

Отклонение от согласованности выражается индексом согласованности.

ИС К1 = (λmax К1 - n)/(n - 1) = 0,8812. отношение согласованности

ОС К1 = = 0.0492.

ОС меньше или равное

1. (% пуха) согласована.
2. – цена (Таблица 3.5.3).

*3.5.3 – Матрица сравнения по критерию 2.*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| К2 | А1 | А2 | А3 | А4 | А5 | W3К2Y |
| А1 | 1 | 0.3333 | 7 | 2 | 0.1111 | 0,0996 |
| А2 | 3 | 1 | 7 | 6 | 0.3333 | 0,2842 |
| А3 | 0.1429 | 0.1429 | 1 | 0.1429 | 0.1111 | 0,0216 |
| А4 | 0.5 | 0.1667 | 7 | 1 | 0.1111 | 0,0657 |
| А5 | 9 | 3 | 9 | 9 | 1 | 0,5289 |

Определена сумма каждого столбца матрицы суждений.

S1К2 = 13.6440;

S2 К2 = 4.6432;

S3 К2 = 31; S4 К2 = 18.1426;

S5 К2 = 1.6667.

Затем полученный результат умножен на компоненту нормализованного вектора приоритетов.

Р1 К2 = S1 х W3 К21 = 1.359;

Р2 К2 = S2 х W3 К22 = 1.32;

Р3 К2 = S3 х W3 К23 = 0.6694;

Р4 К2 = S4 х W3 К24 = 1.192;

Р5 К2 = S5 х W3 К25 = 0.8815.

Найдена пропорциональность предпочтений.

λmax К2 = Р1К2 + Р2К2 + Р3К2 + Р4К2 + Р5К2 = 5.421.

Отклонение от согласованности выражается индексом согласованности.

ИС К2 = (λmax К2 - n)/(n - 1) = 1.684.

отношение согласованности

ОС К2 = = 0.0940.

ОС меньше или равное

1. (цена) согласована.
2. – сочетаемость (Таблица 3.5.4).

*3.5.4 – Матрица сравнения по критерию 3.*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| К3 | А1 | А2 | А3 | А4 | А5 | W3К3Y |
| А1 | 1 | 2 | 1 | 1 | 0.2 | 0,1262 |
| А2 | 0.5 | 1 | 0.5 | 0.5 | 0.2 | 0,0725 |
| А3 | 1 | 2 | 1 | 1 | 0.2 | 0,1262 |
| А4 | 1 | 2 | 1 | 1 | 0.2 | 0,1262 |
| А5 | 5 | 5 | 5 | 5 | 1 | 0,5491 |

Определена сумма каждого столбца матрицы суждений.

S1К3 = 8.5;

S2 К3 = 12;

S3 К3 = 8.5;

S4 К3 = 8.5;

S5 К3 = 1.8.

Затем полученный результат умножен на компоненту нормализованного вектора приоритетов.

Р1 К3 = S1 х W3 К31 = 1.0727;

Р2 К3 = S2 х W3 К32 = 0.87;

Р3 К3 = S3 х W3 К33 = 1.0727;

Р4 К3 = S4 х W3 К34 = 1.0727;

Р5 К3 = S5 х W3 К35 = 0.9884.

Найдем пропорциональность предпочтений.

λmax К3 = Р1К3 + Р2К3 + Р3К3 + Р4К3 + Р5К3 = 5.0765.

Отклонение от согласованности выражается индексом согласованности.

ИС К3 = (λmax К3 - n)/(n - 1) = 0.306.

отношение согласованности

ОС К3 = = 0.0171.

ОС меньше или равное

1. (сочетаемость) согласована.
2. – длина рукава (Таблица 3.5.5).  *3.5.5 – Матрица сравнения по критерию 4.*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| К4 | А1 | А2 | А3 | А4 | А5 | W3К4Y |
| А1 | 1 | 7 | 5 | 2 | 0.3333 | 0,2447 |
| А2 | 0.1429 | 1 | 0.5 | 0.1667 | 0.1111 | 0,0346 |
| А3 | 0.2 | 2 | 1 | 0.25 | 0.125 | 0,0543 |
| А4 | 0.5 | 6 | 4 | 1 | 0.25 | 0,1624 |
| А5 | 3 | 9 | 8 | 4 | 1 | 0,5040 |

Определена сумма каждого столбца матрицы суждений.

S1К4 = 4.8432;

S2К4 = 24.9976;

S3К4 = 18.5; S4К4 = 7.4167;

S5К4 = 1.8194.

Затем полученный результат умножен на компоненту нормализованного вектора приоритетов.

Р1К4 = S1 х W3 К41 = 1.1851;

Р2К4 = S2 х W3 К42 = 0.8649;

Р3К4 = S3 х W3 К43 = 1.0046;

Р4К4 = S4 х W3 К44 = 1.2045;

Р5К4 = S5 х W3 К45 = 0.917.

Найдена пропорциональность предпочтений.

λmax К4 = Р1К4 + Р2К4 + Р3К4 + Р4К4 + Р5К4 = 5.1761.

Отклонение от согласованности выражается индексом согласованности.

ИС К4 = (λmax К4 - n)/(n - 1) = 0.7044. отношение согласованности ОС.

ОС К4 = ИС/СИ = 0.0393.

ОС меньше или равное 0,10 4 (длина рукава) согласована.

5 – длина (Таблица 3.5.6).

*3.5.6 – Матрица сравнения по критерию 5.*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| К5 | А1 | А2 | А3 | А4 | А5 | W3К5Y |
| А1 | 1 | 7 | 6 | 4 | 0.5 | 0,3328 |
| А2 | 0.1429 | 1 | 0.5 | 0.25 | 0.1429 | 0,0416 |
| А3 | 0.1667 | 2 | 1 | 0.5 | 0.1667 | 0,0670 |
| А4 | 0.25 | 4 | 2 | 1 | 0.25 | 0,1194 |
| А5 | 2 | 7 | 6 | 4 | 1 | 0,4392 |

Определена сумма каждого столбца матрицы суждений.

S1К5 = 3.5595;

S2К5 = 20.9958;

S3К5 = 15.4976;

S4К5 = 9.75;

S5К5 = 2.0595.

Затем полученный результат умножен на компоненту нормализованного вектора приоритетов.

Р1К5 = S1 х W3 К41 = 1.1846;

Р2К5 = S2 х W3 К42 = 0.8734;

Р3К5 = S3 х W3 К43 = 1.0383;

Р4К5 = S1 х W3 К44 = 1.1642;

Р5К5 = S1 х W3 К45 = 0.9045.

Найдена пропорциональность предпочтений.

λmax К5 = Р1К5 + Р2К5 + Р3К5 + Р4К5 + Р5К5 = 5.1651.

Отклонение от согласованности выражается индексом согласованности.

ИС К5 = (λmax К5 - n)/(n - 1) = 0.6604. отношение согласованности ОС.

ОС К5 = ИС/СИ = 0.0368.

ОС меньше или равное 0,10 5 (длина) согласована.

## 3.6 Синтез альтернатив

Векторы приоритетов и отношения согласованности определяются для всех матриц суждений, начиная со второго уровня.

Для определения приоритетов альтернатив локальные приоритеты умножены на приоритет соответствующего критерия на высшем уровне и найдены суммы по каждому элементу в соответствии с критериями, на которые воздействует этот элемент.

W2i = (0,4453; 0,2869; 0,1561; 0,074; 0,0377);

W3К1Y = (0,3087; 0,0232; 0,0506; 0,3087; 0,3087);

W3К2Y = (0,0996; 0,2842; 0,0216; 0,0657; 0,5289));

W3К3Y = (0,1262; 0,0725; 0,1262; 0,1262; 0,5491);

W3К4Y = (0,0346; 0,0346; 0,0543; 0,1624; 0,504);

W3К5Y = (0,3328; 0,0416; 0,067; 0,1194; 0,4392).

Приоритеты альтернатив получены следующим образом:

W1 = W21 х W3К11 + W22 х W3К21 + W23 х W3К31 + W24 х W3К41 + W25 х W3К51 = = 0,2164.

W2 = W21 х W3К12 + W22 х W3К22 + W23 х W3К32 + W24 х W3К42 + W25 х W3К52 = = 0,1073.

W3 = W21 х W3К13 + W22 х W3К23 + W23 х W3К33 + W24 х W3К43 + W25 х W3К53 = = 0,055

W4 = W21 х W3К14 + W22 х W3К24 + W23 х W3К34 + W24 х W3К44 + W25 х W3К54 = = 0,1925.

W5 = W21 х W3К15 + W22 х W3К25 + W23 х W3К35 + W24 х W3К45 + W25 х W3К55 = = 0,4288.

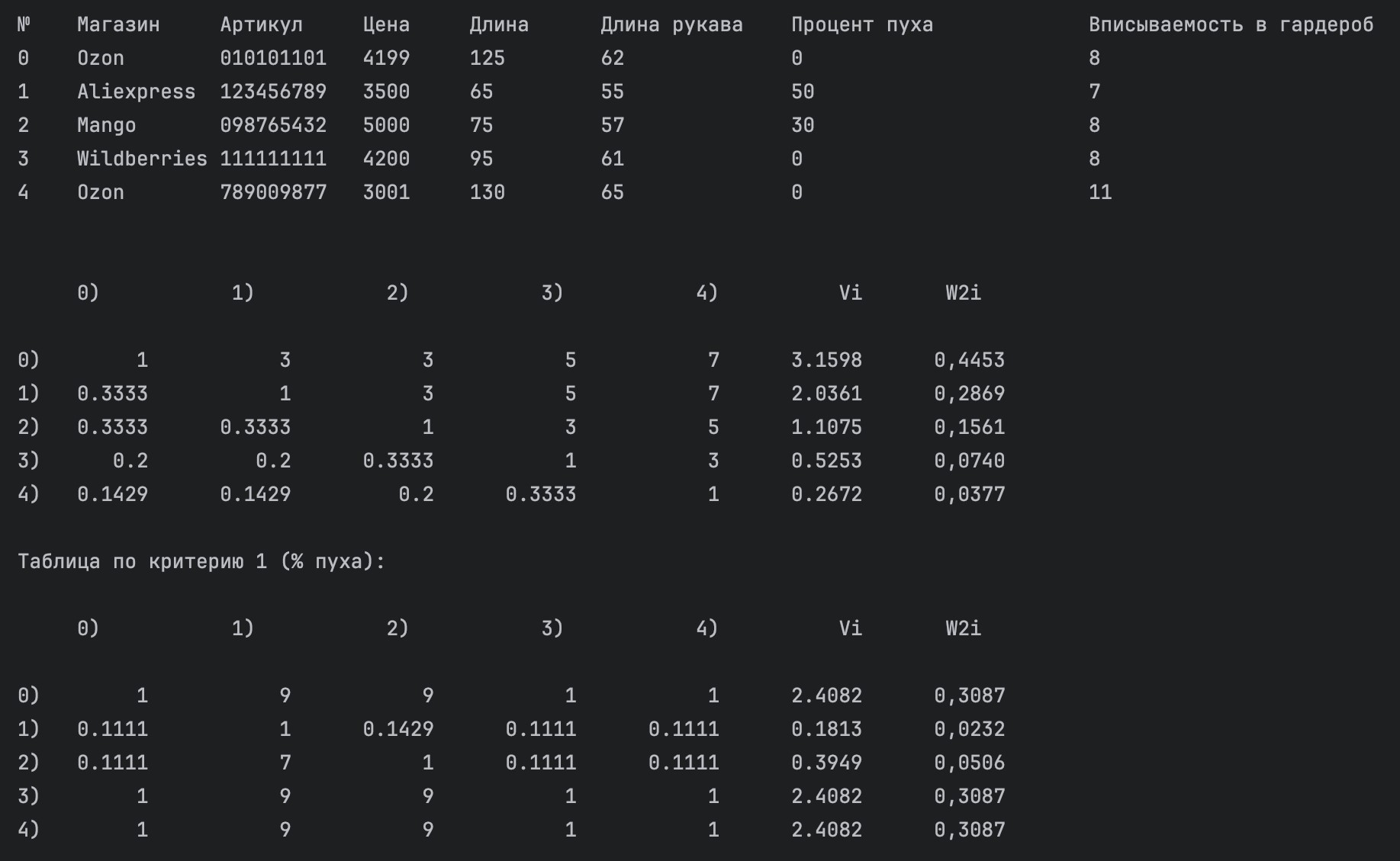
Таким образом, приоритеты альтернатив равны:

альтернатива А1 (010101101) - W1 приоритет равен = 0,2164; альтернатива А2 (123456789)- W2 приоритет равен = 0,1073; альтернатива А3 (098765432) - W3 приоритет равен = 0,055; альтернатива А4 (111111111) – W4 приоритет равен = 0,1925; альтернатива А5 (789009877) - W5 приоритет равен = 0,4288.

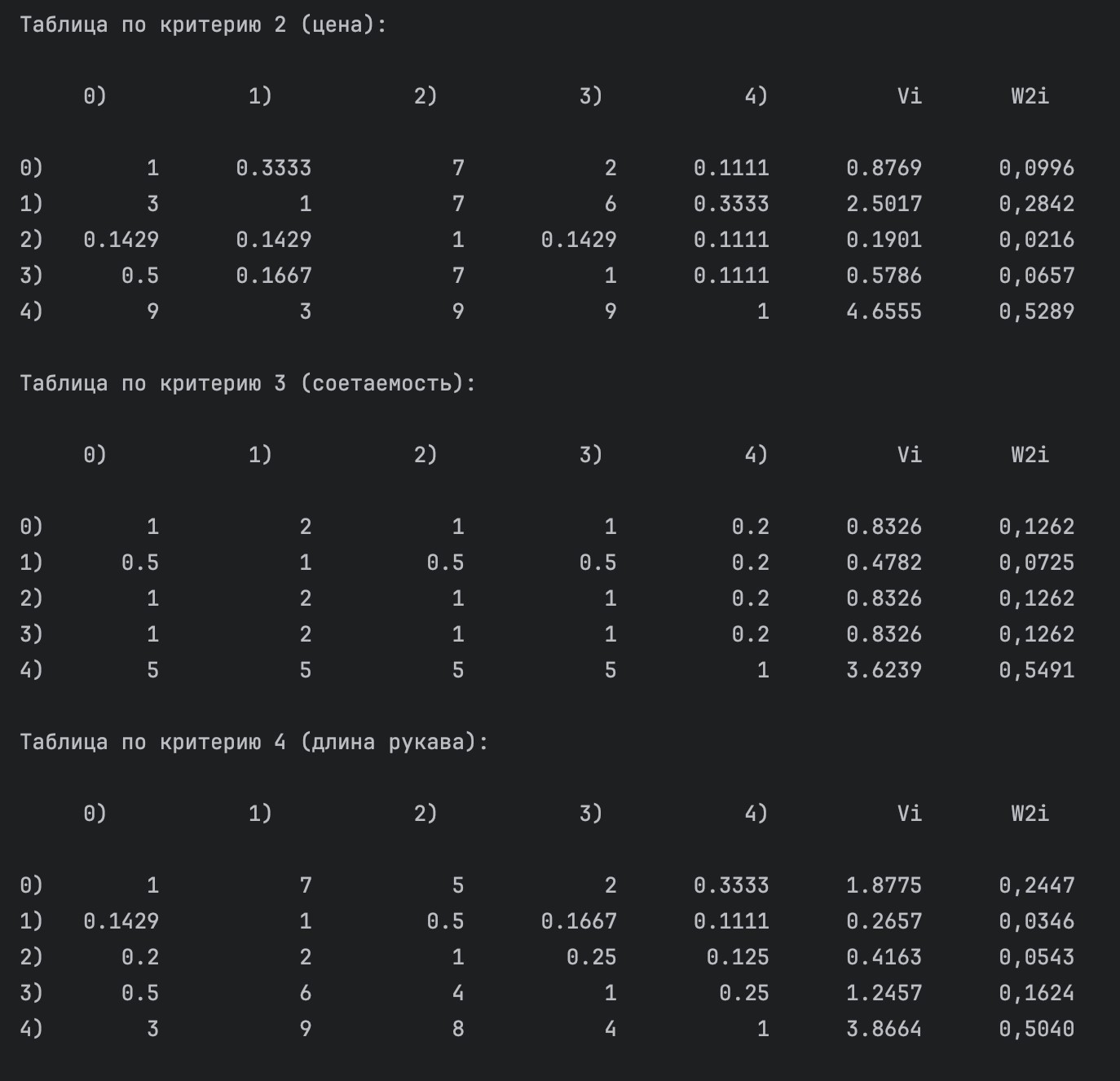
## 3.7 Вывод

Наиболее перспективным с позиции метода анализа иерархий признается выбор А5.

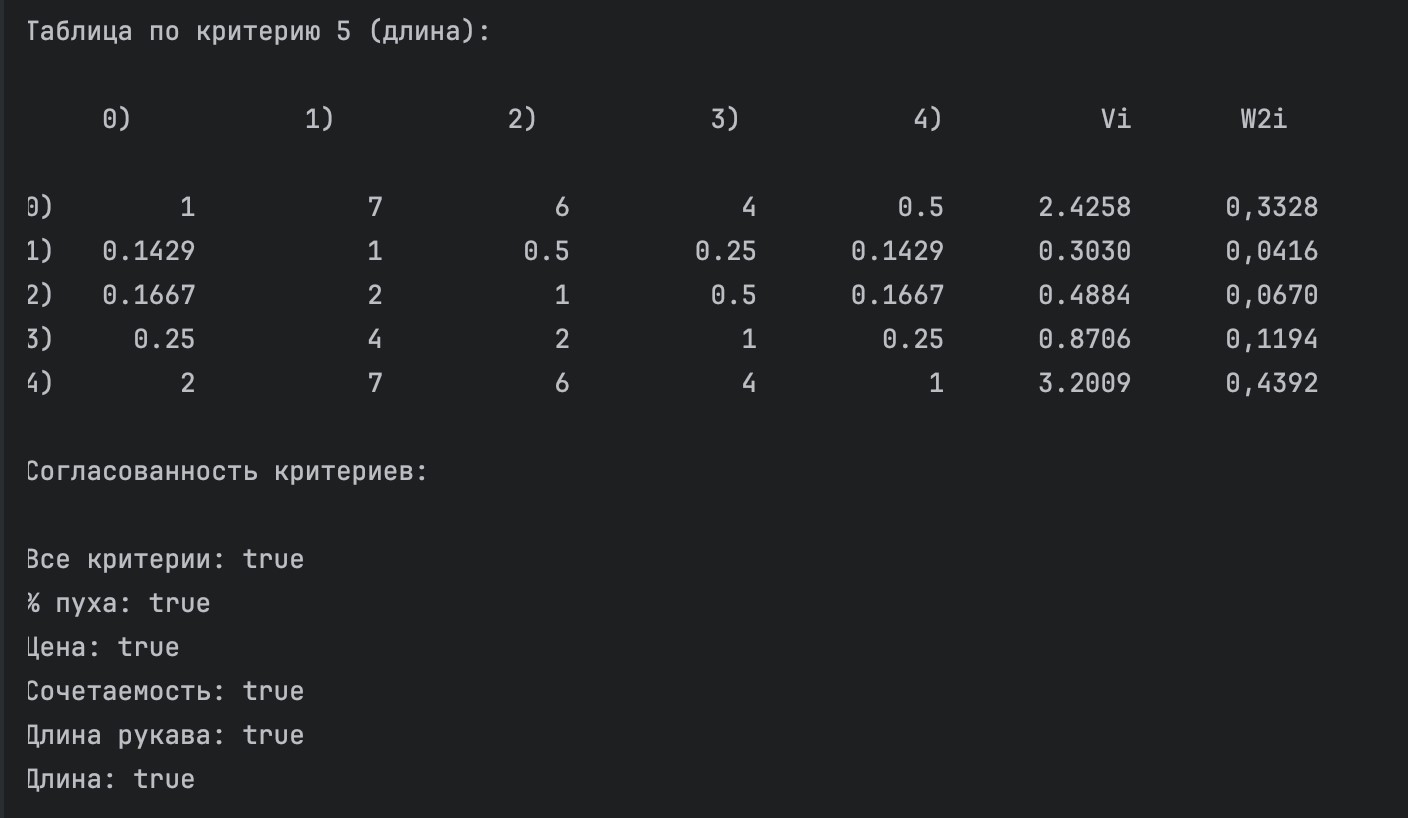
## 1.8 Результаты работы программы



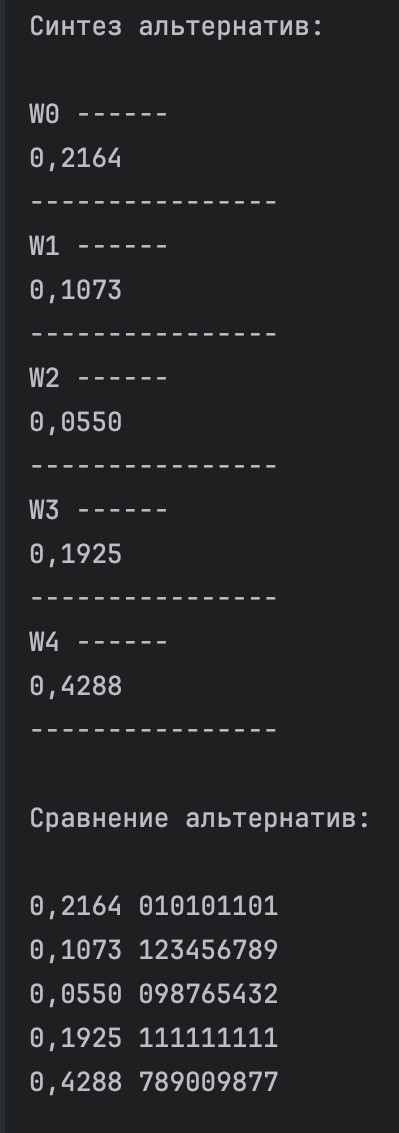
**Рисунок 2 – Вывод программы**



**Рисунок 3 – Вывод программы**



**Рисунок 4 – Вывод программы**



**Рисунок 5 – Вывод программы**

**ЗАКЛЮЧЕНИЕ**

В ходе выполнения практической работы, было приведено представление системы критериев (целей) в виде иерархической структуры, приведена оценка приоритетов (весов) критериев с учётом их места в иерархии относительной важности, определена лучшая альтернатива по значениям её характеристик и важности критериев. Преимущества использования иерархий в качестве описания задачи принятия решений:

* иерархическое представление задачи позволяет описывать влияние элементов иерархии одного уровня на элементы другого уровня;
* построение иерархии исходит из способа мышления человека

(определение объектов и установление связей между ними);

* иерархия устойчива и гибка в том смысле, что малые ее изменения

(удаление и добавление элементов) не разрушают ее характеристик;

**СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ**

1. Болотова Л. С. Многокритериальная оптимизация. Болотова Л. С., Сорокин А. Б. [Электронный ресурс] / Метод. указания по вып. курсовой работы — М.: МИРЭА, 2015.
2. Сорокин А. Б. Методы оптимизации: гибридные генетические алгоритмы. Сорокин А. Б. [Электронный ресурс] / Учебно-метод. пособие — М.: МИРЭА, 2016.
3. Сорокин А. Б. Линейное программирование: практикум. Сорокин А. Б., Бражникова Е. В., Платонова О. В. [Электронный ресурс] / Учебнометод. пособие — М.: МИРЭА, 2017.

**ПРИЛОЖЕНИЯ**

Приложение А – Код реализации МАИ на языке Java.

**Приложение А** Код реализации МАИ на языке Java.

|  |
| --- |
| package Electro;  import java.awt.\*; import java.util.ArrayList; import static java.awt.Color.\*; public class DownJacket {  int length; // + int sleeveLength; // + int downPercentage; // -  int price; // -  int baseCoefficient = 0; // +  String storeCode; String storeName; int number;  private final int hasHood; // 0 - true, 1 - false  private final int width; private final Color color; private final int designDetails;  public DownJacket(String storeName, String storeCode, int length, int sleeveLength, int downPercentage, int price, int hasHood, int width, Color  color, int designDetails, int number) { this.storeCode = storeCode; this.storeName = storeName;  this.baseCoefficient = calculateBaseCoefficient(width, hasHood,  designDetails, color); this.length = length;  this.sleeveLength = sleeveLength; this.downPercentage = downPercentage;  this.price = price; this.hasHood = hasHood; this.width = width; this.color = color;  this.designDetails = designDetails;  this.number = number;  }  private int calculateBaseCoefficient(int width, int hasHood, int  designDetails, Color color) { int w = 0, c = 0;  if (width > 47 && width < 65)  w++;  if (color == *WHITE* || color == *GRAY* || color == *lightGray* || color == *BLACK*) c++; package MAI;  import java.awt.\*;  import static java.awt.Color.\*; public class DownJacket {  int length; // + int sleeveLength; // + int downPercentage; // -  int price; // -  int baseCoefficient = 0; // +  String storeCode; String storeName; int number; |

|  |
| --- |
| private final int hasHood; // 0 - true, 1 - false  private final int width; private final Color color; private final int designDetails;  public DownJacket(String storeName, String storeCode, int length, int sleeveLength, int downPercentage, int price, int hasHood, int width,  Color color, int designDetails, int number) {  this.storeCode = storeCode; this.storeName = storeName;  this.baseCoefficient = calculateBaseCoefficient(width, hasHood,  designDetails, color); this.length = length;  this.sleeveLength = sleeveLength; this.downPercentage = downPercentage;  this.price = price; this.hasHood = hasHood; this.width = width; this.color = color;  this.designDetails = designDetails;  this.number = number;  }  private int calculateBaseCoefficient(int width, int hasHood, int  designDetails, Color color) { int w = 0, c = 0;  if (width > 47 && width < 65)  w++;  if (color == *WHITE* || color == *GRAY* || color == *lightGray* ||  color == *BLACK*) c++;  return 10 + w + c - designDetails - hasHood;  }  public Color getColor() {  return color;  }  public int getBaseCoefficient() { return baseCoefficient;  }  public int getPrice() { return price;  }  public int getDownPercentage() { return downPercentage;  }  public int getSleeveLength() { return sleeveLength;  }  public int getLength() { return length;  } |

|  |
| --- |
| public String getStoreCode() { return storeCode;  }  public int getDesignDetails() { return designDetails;  }  public int getWidth() { return width;  }  public int getHasHood() { return hasHood;  }  public int getNumber() { return number;  }  public String getStoreName() {  return storeName;  } }  package MAI;  import java.text.DecimalFormat; import java.util.ArrayList;  import static java.util.Collections.*swap*;  public class Mai {  public static String[][] formCreteriaTable() {  DecimalFormat df = new DecimalFormat("0.0000");  String[][] comparedCreteria = {{"1", "3", "3", "5", "7", df.format(Math.*pow*(3 \* 3 \* 5 \* 7, 0.2)).replace(",", "."), ""},  {"0.3333", "1", "1", "5", "7", df.format(Math.*pow*(0.3333  \* 1 \* 1 \* 5 \* 7, 0.2)).replace(",", "."), ""}, {"0.3333", "0.3333", "1", "3", "3",  df.format(Math.*pow*(0.3333 \* 0.3333 \* 1 \* 3 \* 3, 0.2)).replace(",", "."),  ""},  {"0.2", "0.2", "0.3333", "1", "3",  df.format(Math.*pow*(0.2 \* 0.2 \* 0.3333 \* 1 \* 3, 0.2)).replace(",", "."),  ""},  {"0.1429", "0.1429", "0.3333", "0.3333", "1",  df.format(Math.*pow*(0.1429 \* 2 \* 0.3333 \* 2 \* 1, 0.2)).replace(",", "."),  ""}};  double Evi = 0.00;  for (int i = 0; i < 5; i++) {  Evi += Double.*parseDouble*(comparedCreteria[i][5]);  }  for (int i = 0; i < 5; i++) { comparedCreteria[i][6] =  df.format(Double.*parseDouble*(comparedCreteria[i][5].replace(",", ".")) /  Evi);  } |

|  |
| --- |
| return comparedCreteria;  }  public static double compareByPrice(DownJacket a, DownJacket b) {  int delta = 0; if (a.price < b.price) { delta = b.price - a.price;  if (delta <= 300) return 3; else if (delta <= 500)  return 5; else if (delta <= 700) return 7;  return 9;  }  else if (a.price > b.price) { delta = a.price - b.price;  if (delta <= 300) return (double) 1 /3; else if (delta <= 500) return (double) 1 /5; else if (delta <= 700) return (double) 1 /7;  return (double) 1 /9;  }  return 1;  }  public static double compareByLength(DownJacket a, DownJacket b) {  int delta = 0;  if (a.length > b.length) { delta = a.length - b.length;  if (delta <= 5) return 3; else if (delta <= 10)  return 5; else if (delta <= 20) return 7;  return 9;  }  else if (a.length < b.length) { delta = b.length - a.length;  if (delta <= 5) return (double) 1 /3; else if (delta <= 10) return (double) 1 /5; else if (delta <= 20) return (double) 1 /7;  return (double) 1 /9;  }  return 1;  } |

|  |
| --- |
| public static double compareBySleeveLength(DownJacket a, DownJacket b)  {  int delta = 0;  if (a.sleeveLength > b.sleeveLength) { delta = a.sleeveLength - b.sleeveLength;  if (delta <= 2) return 3; else if (delta <= 3) return 5; else if (delta <= 4) return 7;  return 9;  }  else if (a.sleeveLength < b.sleeveLength) { delta = b.sleeveLength - a.sleeveLength;  if (delta <= 2) return (double) 1 /3; else if (delta <= 3) return (double) 1 /5; else if (delta <= 4) return (double) 1 /7;  return (double) 1 /9;  }  return 1;  }  public static double compareByDownPercentage(DownJacket a, DownJacket b) { int delta = 0;  if (a.downPercentage < b.downPercentage) { delta = b.downPercentage - a.downPercentage;  if (delta <= 5) return 3; else if (delta <= 10)  return 5; else if (delta <= 20) return 7;  return 9;  }  else if (a.downPercentage > b.downPercentage) { delta = a.downPercentage - b.downPercentage;  if (delta <= 5) return (double) 1 /3; else if (delta <= 10) return (double) 1 /5; else if (delta <= 20) return (double) 1 /7;  return (double) 1 /9;  }  return 1;  } |

|  |
| --- |
| public static double compareByBaseCoefficient(DownJacket a, DownJacket b) {  int delta = 0;  if (a.baseCoefficient > b.baseCoefficient) { delta = a.baseCoefficient - b.baseCoefficient;  if (delta <= 2) return 3; else if (delta <= 5) return 5; else if (delta <= 7) return 7;  return 9;  }  else if (a.baseCoefficient < b.baseCoefficient) { delta = b.baseCoefficient - a.baseCoefficient;  if (delta <= 2) return (double) 1 /3; else if (delta <= 5) return (double) 1 /5; else if (delta <= 7) return (double) 1 /7;  return (double) 1 /9;  }  return 1;  }  public static void addViEVi(String[][] table){  DecimalFormat df = new DecimalFormat("0.0000");  double Vi = 1.00; double EVi = 0.00;  for (int i = 0; i < table.length; i++) {  for (int j = 0; j < table[0].length - 2; j++) {  Vi \*= Double.*parseDouble*(table[i][j].replace(",", "."));  }  Vi = Math.*pow*(Vi, 0.2); table[i][5] = df.format(Vi);  EVi += Vi;  // table[i][6] = df.format(EVi);  Vi = 1.00;  }  double W2i;  for (int i = 0; i < table.length; i++) {  W2i = Double.*parseDouble*(table[i][5].replace(",", "."));  table[i][6] = df.format(W2i/EVi);  }  System.*out*.println("EVi: " + df.format(EVi)); } |

|  |
| --- |
| public static String[][] formPriceCriteriaTable(ArrayList<DownJacket>  downJackets) {  DecimalFormat df = new DecimalFormat("0.0000");  // double Vi, Evi = 0.00;  String[][] priceCriteriaTable = new String[downJackets.size()]  [downJackets.size() + 2];  for (int i = 0; i < downJackets.size(); i++) {  // Vi = 1.00;  for (int j = i; j < downJackets.size(); j++) {  if (i != j) {  priceCriteriaTable[i][j] = df.format(*compareByPrice*(downJackets.get(i), downJackets.get(j))).replace(",", ".");  priceCriteriaTable[j][i] = df.format(1.0 /  Double.*parseDouble*(priceCriteriaTable[i][j].replace(",",  "."))).replace(",", ".");  } else {  priceCriteriaTable[i][j] = "1";  }  // Vi \*= Double.parseDouble(priceCriteriaTable[i][j]);  // System.out.println(i + " " + j + "------");  // System.out.println("Vi = " + Vi);  }  // priceCriteriaTable[i][downJackets.size()] =  df.format(Math.pow(Vi, 0.2)).replace(",", ".");  // Evi += Vi;  }  // for (int i = 0; i < downJackets.size(); i++) { // priceCriteriaTable[i][downJackets.size() + 1] =  df.format(Double.parseDouble(priceCriteriaTable[i]  [downJackets.size()].replace(",", ".")) / Evi).replace(",", ".");  // }  *addViEVi*(priceCriteriaTable);  return priceCriteriaTable;  }  public static String[][] formLengthCretriaTable(ArrayList<DownJacket>  downJackets) {  DecimalFormat df = new DecimalFormat("0.0000");  // double Vi, Evi = 0.00;  String[][] lengthCriteriaTable = new String[downJackets.size()]  [downJackets.size() + 2];  for (int i = 0; i < downJackets.size(); i++) {  // Vi = 1.00;  for (int j = i; j < downJackets.size(); j++) {  if (i != j) {  lengthCriteriaTable[i][j] = df.format(*compareByLength*(downJackets.get(i), downJackets.get(j))).replace(",", ".");  lengthCriteriaTable[j][i] = df.format(1.0 / Double.*parseDouble*(lengthCriteriaTable[i][j])).replace(",", ".");  } else {  lengthCriteriaTable[i][j] = "1";  } |

|  |
| --- |
| // Vi \*= Double.parseDouble(lengthCriteriaTable[i][j]);  }  // lengthCriteriaTable[i][downJackets.size()] =  df.format(Math.pow(Vi, 0.2));  // Evi += Vi;  }  // for (int i = 0; i < downJackets.size(); i++) { // lengthCriteriaTable[i][downJackets.size() + 1] =  df.format(Double.parseDouble(lengthCriteriaTable[i]  [downJackets.size()].replace(",", ".")) / Evi);  // }  *addViEVi*(lengthCriteriaTable);  return lengthCriteriaTable;  }  public static String[][]  formSleeveLengthCretriaTable(ArrayList<DownJacket> downJackets) {  DecimalFormat df = new DecimalFormat("0.0000");  // double Vi, Evi = 0.00;  String[][] sleeveLengthCriteriaTable = new String[downJackets.size()][downJackets.size() + 2]; for (int i = 0; i < downJackets.size(); i++) {  // Vi = 1.00;  for (int j = i; j < downJackets.size(); j++) {  if (i != j) {  sleeveLengthCriteriaTable[i][j] = df.format(*compareBySleeveLength*(downJackets.get(i),  downJackets.get(j))).replace(",", ".");  sleeveLengthCriteriaTable[j][i] = df.format(1.0 / Double.*parseDouble*(sleeveLengthCriteriaTable[i][j])).replace(",", ".");  } else {  sleeveLengthCriteriaTable[i][j] = "1";  }  // Vi \*= Double.parseDouble(sleeveLengthCriteriaTable[i]  [j]);  }  // sleeveLengthCriteriaTable[i][downJackets.size()] =  df.format(Math.pow(Vi, 0.2));  // Evi += Vi;  }  // for (int i = 0; i < downJackets.size(); i++) {  // sleeveLengthCriteriaTable[i][downJackets.size() + 1] =  df.format(Double.parseDouble(sleeveLengthCriteriaTable[i]  [downJackets.size()].replace(",", ".")) / Evi);  // }  *addViEVi*(sleeveLengthCriteriaTable);  return sleeveLengthCriteriaTable;  }  public static String[][] formDownCretriaTable(ArrayList<DownJacket>  downJackets) {  DecimalFormat df = new DecimalFormat("0.0000");  // double Vi, Evi = 0.00; |

|  |
| --- |
| String[][] downCriteriaTable = new String[downJackets.size()]  [downJackets.size() + 2];  for (int i = 0; i < downJackets.size(); i++) {  // Vi = 1.00;  for (int j = i; j < downJackets.size(); j++) {  if (i != j) {  downCriteriaTable[i][j] = df.format(*compareByDownPercentage*(downJackets.get(i),  downJackets.get(j))).replace(",", ".");  downCriteriaTable[j][i] = df.format(1.0 / Double.*parseDouble*(downCriteriaTable[i][j])).replace(",", ".");  } else {  downCriteriaTable[i][j] = "1";  }  // Vi \*= Double.parseDouble(downCriteriaTable[i][j]);  }  // downCriteriaTable[i][downJackets.size()] =  df.format(Math.pow(Vi, 0.2));  // Evi += Vi;  }  // for (int i = 0; i < downJackets.size(); i++) { // downCriteriaTable[i][downJackets.size() + 1] =  df.format(Double.parseDouble(downCriteriaTable[i]  [downJackets.size()].replace(",", ".")) / Evi);  // }  *addViEVi*(downCriteriaTable);  return downCriteriaTable;  }  public static String[][] formBaseCretriaTable(ArrayList<DownJacket>  downJackets) {  DecimalFormat df = new DecimalFormat("0.0000");  // double Vi, Evi = 0.00;  String[][] BaseCriteriaTable = new String[downJackets.size()]  [downJackets.size() + 2];  for (int i = 0; i < downJackets.size(); i++) {  // Vi = 1.00;  for (int j = i; j < downJackets.size(); j++) {  if (i != j) {  BaseCriteriaTable[i][j] = df.format(*compareByBaseCoefficient*(downJackets.get(i),  downJackets.get(j))).replace(",", ".");  BaseCriteriaTable[j][i] = df.format(1.0 /  Double.*parseDouble*(BaseCriteriaTable[i][j])).replace(",", ".");  } else {  BaseCriteriaTable[i][j] = "1";  }  // Vi \*= Double.parseDouble(BaseCriteriaTable[i][j]);  }  // BaseCriteriaTable[i][downJackets.size()] =  df.format(Math.pow(Vi, 0.2));  // Evi += Vi; |

|  |
| --- |
| }  // for (int i = 0; i < downJackets.size(); i++) { // BaseCriteriaTable[i][downJackets.size() + 1] =  df.format(Double.parseDouble(BaseCriteriaTable[i]  [downJackets.size()].replace(",", ".")) / Evi);  // }  *addViEVi*(BaseCriteriaTable);  return BaseCriteriaTable;  }  public static void displayTable(ArrayList<DownJacket> downJackets) {  System.*out*.println("\n");  System.*out*.printf("%-4s %-11s %-11s %-8s %-10s %-15s %-24s %-18s  %n",  "№", "Магазин", "Артикул", "Цена", "Длина", "Длина  рукава", "Процент пуха", "Вписываемость в гардероб"); for (DownJacket jacket : downJackets) {  System.*out*.printf("%-4s %-11s %-11s %-8s %-10s %-15s %-24s  %-18s %n",  jacket.getNumber(), jacket.getStoreName(), jacket.getStoreCode(), jacket.getPrice(), jacket.getLength(),  jacket.getSleeveLength(), jacket.getDownPercentage(),  jacket.getBaseCoefficient());  } }  public static void displayComparisonTable(String[][]  comparisonTable) {  System.*out*.printf("%4s", " ");  for (int i = 0; i < comparisonTable[0].length - 2; i++) {  System.*out*.printf("%3s", i + ")");  System.*out*.printf("%10s", " ");  }  System.*out*.printf("%2s", "Vi");  System.*out*.printf("%-7s", " ");  System.*out*.printf("%2s", "W2i");  System.*out*.println("\n");  for (int i = 0; i < comparisonTable.length; i++) { for (int j = 0; j < comparisonTable[i].length; j++) {  if (j == 0) {  System.*out*.printf("%-1s %6s", i + ") ",  comparisonTable[i][j]); } else {  System.*out*.printf("%12s", comparisonTable[i][j]);  }  }  System.*out*.println();  } }  public static boolean harmonization(String[][] comparisonTable) { //  поправить  Double[] S = {0.00, 0.00, 0.00, 0.00, 0.00};  double LambdaMax = 0.00; |

|  |
| --- |
| for (int i = 0; i < 5; i++) { for (int j = 0; j < 5; j++) {  S[i] += (double) 1/Double.*parseDouble*(comparisonTable[i]  [j].replace(",", "."));  // System.out.println(i + ": 1/ai = " + (double) 1/  Double.parseDouble(comparisonTable[i][j].replace(",", ".")));  }  // System.out.println(i + ": Si = " + S[i]);  }  for (int i = 0; i < 5; i++) {  double Wi = Double.*parseDouble*(comparisonTable[i]  [6].replace(",", "."));  // System.out.println(i + ": Wi = " + Wi);  // System.out.println(i + ": Pi = " + S[i] \* Wi);  LambdaMax += S[i] \* Wi;  }  // System.out.println("LambdaMax = " + LambdaMax);  //ИС = (λmax – n)/(n-1), ОС = ИС/СИ  double OS = ((LambdaMax - 5) / 4) / 1.12;  // System.out.println("OS = " + OS); return OS <= 0.1;  }  public static Double[] alternativesSynthesis(String[][]  PriceCriteriaTable,  String[][]  LengthCretriaTable,  String[][]  SleeveLengthCretriaTable,  String[][]  DownCretriaTable,  String[][]  BaseCretriaTable,  String[][]  comparedCreteria){  Double[] W = new Double[5]; for (int i = 0; i < 5; i++) {  W[i] = Double.*parseDouble*(comparedCreteria[0]  [6].replace(",", ".")) \* Double.*parseDouble*(DownCretriaTable[i] [6].replace(",", "."))  + Double.*parseDouble*(comparedCreteria[1]  [6].replace(",", ".")) \* Double.*parseDouble*(PriceCriteriaTable[i]  [6].replace(",", "."))  + Double.*parseDouble*(comparedCreteria[2]  [6].replace(",", ".")) \* Double.*parseDouble*(BaseCretriaTable[i] [6].replace(",", "."))  + Double.*parseDouble*(comparedCreteria[3]  [6].replace(",", ".")) \* Double.*parseDouble*(SleeveLengthCretriaTable[i]  [6].replace(",", "."))  + Double.*parseDouble*(comparedCreteria[4]  [6].replace(",", ".")) \* Double.*parseDouble*(LengthCretriaTable[i] [6].replace(",", ".")); |

|  |
| --- |
| System.*out*.println(comparedCreteria[0][6] + " " + DownCretriaTable[i][6]  + " "  + comparedCreteria[1][6] + " " +  PriceCriteriaTable[i][6] + " "  + comparedCreteria[2][6] + " " + BaseCretriaTable[i]  [6] + " "  + comparedCreteria[3][6] + " " +  SleeveLengthCretriaTable[i][6] + " "  + comparedCreteria[4][6] + " " +  LengthCretriaTable[i][6]);  }  return W;  }  public static void displayPriorityTable(Double[] W,  ArrayList<DownJacket> downJackets){  System.*out*.println("\nСравнение альтернатив: \n");  for (int i = 0; i < W.length; i++) { System.*out*.println(W[i] + " " +  downJackets.get(i).storeCode);  }  } }  package MAI;  import java.awt.\*; import java.util.ArrayList; import static MAI.Mai.\*;  public class Main {  public static void main(String[] args) { ArrayList<DownJacket> downJackets = new ArrayList<>();  DownJacket jacket0 = new DownJacket("Ozon", "010101101", 125,  62, 0, 4199, 0, 59, Color.*PINK*, 3, 0);  DownJacket jacket1 = new DownJacket("Aliexpress", "123456789",  65, 55, 50, 3500, 0, 52, Color.*BLUE*, 4, 1);  DownJacket jacket2 = new DownJacket("Mango", "098765432", 75,  57, 30, 5000, 0, 95, Color.*GRAY*, 3, 2);  DownJacket jacket3 = new DownJacket("Wildberries", "111111111",  95, 61, 0, 4200, 1, 46, Color.*BLACK*, 2, 3);  DownJacket jacket4 = new DownJacket("Ozon", "789009877", 130,  65, 0, 3001, 0, 50, Color.*WHITE*, 1, 4); downJackets.add(jacket0); downJackets.add(jacket1); downJackets.add(jacket2); downJackets.add(jacket3); downJackets.add(jacket4); *displayTable*(downJackets); |

String [][] creterias = *formCreteriaTable*();

String [][] price = *formPriceCriteriaTable*(downJackets);

String [][] length = *formLengthCretriaTable*(downJackets);

String [][] sleeveLength = *formSleeveLengthCretriaTable*(downJackets);

String [][] down = *formDownCretriaTable*(downJackets);

String [][] matchCoefficient = *formBaseCretriaTable*(downJackets);

System.*out*.println("\n"); *displayComparisonTable*(creterias);

System.*out*.println("\nТаблица по критерию 1 (% пуха): \n");

*displayComparisonTable*(down);

System.*out*.println("\nТаблица по критерию 2 (цена): \n");

*displayComparisonTable*(price);

System.*out*.println("\nТаблица по критерию 3 (соетаемость): \n");

*displayComparisonTable*(matchCoefficient);

System.*out*.println("\nТаблица по критерию 4 (длина рукава):

\n");

*displayComparisonTable*(sleeveLength);

System.*out*.println("\nТаблица по критерию 5 (длина): \n"); *displayComparisonTable*(length);

System.*out*.println("Согласованность критериев:\n");

System.*out*.println("Все критерии: " + *harmonization*(creterias) + "\n");

System.*out*.println("\n% пуха: " + *harmonization*(down) + "\n");

System.*out*.println("\nЦена: " + *harmonization*(price) + "\n");

System.*out*.println("\nСочетаемость: " + *harmonization*(matchCoefficient) + "\n");

System.*out*.println("\nДлина рукава: " +

*harmonization*(sleeveLength) + "\n"); System.*out*.println("\nДлина: " + *harmonization*(length) + " \n");

System.*out*.println("Синтез альтернатив:\n");

Double[] W = *alternativesSynthesis*(price, length, sleeveLength,

down, matchCoefficient, creterias);

*displayPriorityTable*(W, downJackets);

}

}