Санкт-Петербургский государственный политехнический университет

Институт компьютерных наук и технологий

Высшая школа программной инженерии

Экзаменационная работа

Дисциплина: Системный анализ и принятие решений

Тема: Система поддержки принятия решения (СППР) на основе качественного подхода для выбора лучшего города для проживания

Выполнил студент гр. в3530904/00030 О.С. Клабукова

Руководитель В.В. Амосов

Санкт-Петербург
2022

СОДЕРЖАНИЕ

1 Введение

Экзаменационная работа посвящена разработке, программной реализации в виде веб-приложения и тестированию на примерах системы поддержки принятия решения (СППР) на основе качественного подхода к принятию решения по выбору лучшего города для проживания (ПР). В общем с необходимостью принятия решения сталкиваются и при разработке искусственного интеллекта, и при машинном обучении, и при разработке систем реального времени. Принятие решения может быть осуществлено несколькими способами, как предлагаемым, так и на основе количественного подхода. Качественный подход к принятию решения в данной статье подразумевает формализацию ситуации ПР с использованием аппарата бинарных отношений (БО) или предпочтений, задание для каждого БО весового коэффициента, проведение выбора решений с помощью аппарата функций выбора (ФВ), поиск оптимальных вариантов решений для каждого БО и сведения полученных результатов в обобщающую таблицу для наглядности и автоматизации выбора лицом, принимающим решение (ЛПР). Возможно автоматическое принятие решения.

2 Материалы и методы (математическая модель систе поддержки принятия решений, описание методов её р в программу)

Бинарные отношения задаются матрицами вручную с помощью экспертов, либо с помощью отдельной программы. Аппарат функций выбора задаётся для каждого БО механизмами доминирования, блокировки, турнирным механизмом и механизмом определения K-максимальных вариантов. Для каждого БО задаются весовые коэффициенты (к). Полученные по каждому механизму результаты ранжируются с учётом весовых коэффициентов БО. По механизмам доминирования и блокировки определяется сколько раз варианты решения доминировали и блокировали по разным БО, затем для каждого доминирующего или блокирующего по БО варианта формируем сумму из весовых коэффициентов соответствующих БО, и исходя из этого варианты решения ранжируются по убыванию с учётом весовых коэффициентов БО.

Механизм доминирования

Механизм, реализующий выбор того варианта, который лучше с точки зрения заданного предпочтения или БО, чем все остальные, но возможна и симметрия, то есть два варианта равноценны

$$CR(x) = \{x \in X \mid \forall y \in X,\ xRy\}$$

Механизм блокировки

Механизм, реализующий выбор того варианта, для которого нет варианта лучшего с точки зрения заданного предпочтения или БО

$$CR(x) = \{x \in X \mid \forall y \in X,\ yR\bar{x}\}$$

Турнирный механизм

Так как механизмы доминирования и блокировки не дают возможности учесть то, что некоторые варианты, не являясь лидерами могут иметь хорошие показатели по всем предпочтениям, воспользуемся турнирным механизмом, который это учитывает.

$$fR(x) = \Sigma fR(x, y),$$

где

$$fR(x, y) = \{1, \text{при}(xRy) \wedge (yR\bar{x})$$

$$0, \text{при}(yRx) \wedge (xR\bar{y})$$

$$1/2, \text{остальное}\},$$

- y - остальные варианты решения $\in X$,
- X – предъявленное множество вариантов,
- x и y – элементы множества вариантов X,
- $fR(x, y)$ – функция сравнения вариантов x и y,
- $fR(x)$ – результирующая функция варианта x,
- в каждом БО для каждого варианта x получаем своё число $fR(x)$,
- в каждом БО перемножаем эти числа $fR(x)$ на соответствующий этому БО весовой коэффициент (к*$fR(x)$),
- для каждого варианта x получаем сумму произведений (к*$fR(x)$) путём сложения этих произведений для варианта x в каждом БО.

Частные решения для каждой матрицы БО получаются из условия: чем больше сумма для данной альтернативы, тем данная альтернатива лучше подходит в качестве решения.

Для получения общего решения используется следующий механизм: для каждой альтернативы находится сумма по предпочтениям, где слагаемые равны значению турнирной функции по данному предпочтению, умноженному на значимость данного предпочтения. Среди полученных значений выбирается наибольшее, второе по величине, третье и т.д., это и определяет то место, которое займет данная альтернатива в общем решении.

Механизм К-максимальных

Перебираем для каждого БО все варианты x (x1, …, xn) и для каждого варианта x:

- определяем количество вариантов, подчинённых x по каждому БО R

HR0 (x) = {$y \in$ X, y $\neq x \mid xRy \wedge yR\bar{x}$},

- определяем количество вариантов, эквивалентных x по каждому БО R

ER (x) = {$y \in$ X, y $\neq x \mid xRy \wedge yRx$ },

- определяем количество вариантов, несравнимых с x по каждому БО R

NR (x) = {$y \in$ X, y $\neq x \mid xR\bar{y} \wedge yR\bar{x}$},

- определяем количество вариантов, подчинённых, эквивалентных и несравнимых x по каждому БО R

SR1(x) = HR0 (x)$\cup$ ER (x) $\cup$ NR (x),

- определяем количество вариантов, подчинённых и несравнимых x по каждому БО R

SR2 (x) = HR0 (x)$\cup$ NR (x),

- определяем количество вариантов подчинённых и эквивалентных x

по каждому БО R

SR3 (x) = HR0 (x)$\cup$ ER (x),

- определяем количество вариантов подчинённых x по каждому БО R

SR4 (x) = HR0 (x).

Для каждого БО перебираем все варианты x (x1, …, xn) и для каждого варианта x формируем вектор из 4 компонентов

x1 (SR1(x1), SR2 (x1), SR3 (x1), SR4 (x1))

x2 (SR1(x2), SR2 (x2), SR3 (x2), SR4 (x2))

. . .

xn (SR1(xn), SR2 (xn), SR3 (xn), SR4 (xn)),

где n – количество вариантов решения.

По каждому варианту от 1 до n для каждого БО получили числа SRi(xj), где

i=1,..., 4, а j=1,…,n. Затем в рамках каждого отдельного БО для каждого варианта (xj) получаем сумму по i от 1 до 4 ∑S*R*i (*x*j) и эту сумму умножаем на соответствующий этому БО весовой коэффициент «к», то есть Sj=к* ∑*SR*i(*x*j), где i=1,...,4. Получаем для каждого БО столбец из Sj, где j=1,…,n.

Определяем для каждого варианта «j» сумму из Sj по каждому БО: Sjp=∑ Sij, i=1, …, p, где p-количество БО. Получаем для всех БО один столбец сумм Sjp всех вариантов, которые ранжируем по убыванию.

Для каждого БО по каждому компоненту векторов всех вариантов по 1-му, 2-му, 3-му и 4-му компоненту находим максимальные значения!

Им будут соответствовать варианты, которые будут являться, соответственно, 1-максимальным, 2-максимальным, 3-максимальным и 4-максимальным вариантами!

После сравниваем численные значения 1,2,3,4-максимальных вариантов с общим количеством вариантов (n) и определяем являются ли эти к-максимальные варианты ещё и оптимальными вариантами (максимальными, строго максимальными, наибольшими и строго наибольшими). В результате находим оптимальные варианты для каждого БО.

Найденные оптимальные варианты также ранжируются с учётом весовых коэффициентов БО.

Для этого определяем сколько раз (число «M») в разных БО найденные варианты были оптимальными: максимальными, строго максимальными, наибольшими и строго наибольшими. В разных БО для оптимальных вариантов имеем разные значения Sj. Определяем для каждого оптимального варианта сумму из Sj по m от 1 до M: SjM=∑ Sjm, но только сумму тех Sj, которые соответствуют оптимальным вариантам.

Получаем для всех БО столбец сумм SjM всех оптимальных вариантов, которые ранжируем по убыванию.

Таким образом для механизма К-максимальных вариантов имеем два ранжирования: по суммам Sjp для всех вариантов и по суммам SjM для оптимальных вариантов.

Выбор лучшего варианта

Выбор лучшего варианта проводится по всем ранжированиям: по механизмам доминирования и блокировки, по турнирному механизму и по механизму определения К-максимальных вариантов, основываясь на балльной системе: балл («B») для каждого варианта при каждом механизме определяется как количество вариантов «n» плюс 1 минус номер места в ранжированном столбце («Lj»), соответствующем каждому механизму Bj=n+1-Lj. Затем баллы для каждого варианта по всем механизмам

складываются и по максимальной сумме выбирается лучший вариант.

3 Системный и архитектурный уровень разработки СП

На рисунке 1 представлен системный уровень разработки СППР, в котором приложения представлено чёрным ящиком с описание входных и выходных данных

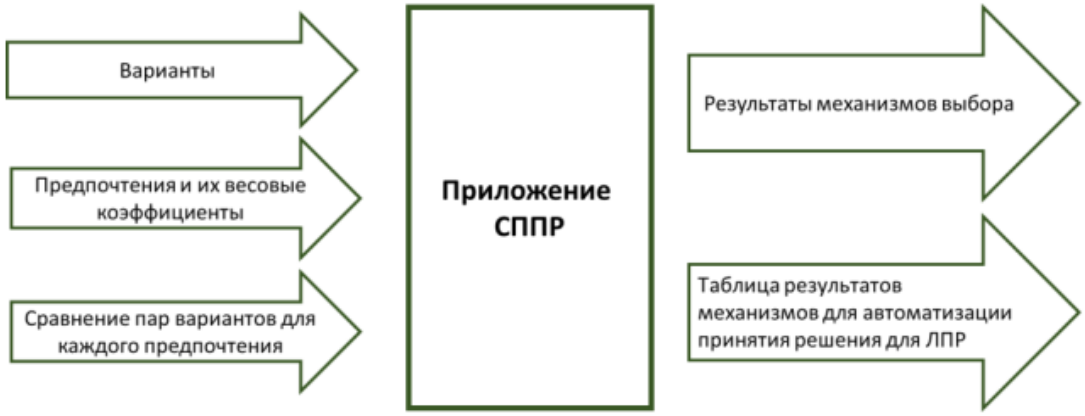


Рис. 1. Системный уровень разработки.

Архитектурный уровень разработки представлен на рисунке 2. В нём приложение СППР описывается структурной блок-схемой.

Рис. 2. Архитектурный уровень разработки.

Тестирование приложения будет выполняться в ручном режиме.
Для тестирования необходимо разработать вторую версия приложения.

На рисунке 3 представлены две версии приложения: версия разработчика и пользовательская версия.

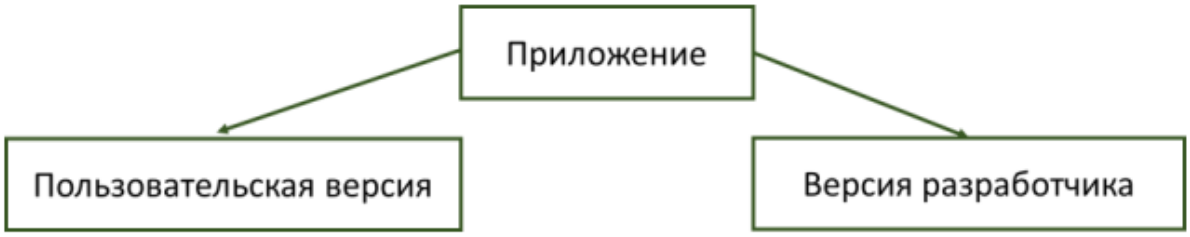


Рис. 3.

Версия разработчика:

• реализует полный функционал пользовательской версии, имея при этом дополнительные возможности.

• Предусмотрены дополнительные экранные управляющие кнопки, вызывающие выполнение операций, автоматически выполняемых в пользовательской версии, что позволяет разработчику контролировать правильность обработки данных в любой операции.

• Предусмотрен вывод в консоль браузера промежуточных данных, получаемых на различных этапах выполнения программы.

4 Результаты

Для демонстрации приложения произведём выбор, приняв для сравнения 6 разных городов. Обозначим данные о предпочтениях и их весовые коэффициенты, которые предоставлены в таблице 1. Весовые коэффициенты в сумме должны равняться единице.

Весовые коэффициенты		0,2	0,2	0,15	0,2	0,1	0,15
№ п/п	Город	Возможная ЗП	Семья и друзья	Уровень медицины	Отношение стоимости 1к квартиры к ЗП	Климат + экология	Доступность дачного участка
1	Санкт-Петербург	126 000	1	2	64	1	1
2	Москва	300 000	1	3	50	1	0
3	Екатеринбург	80 000	3	1	69	0	1
4	Хельсинки	126 000	1	2	108	2	2
5	Сочи	70 000	0	0	143	2	1
6	Асбест	40 000	1	0	25	0	2

Выбраны следующие БО:

- Возможная ЗП в области фармации/биотехнологии. Чем больше, тем лучше
- Семья и друзья - введено условное обозначение, где 0 - в данном городе нет таких человек, 1 - в данном городе до 10 человек, 2- в данном городе 10-20 человек, 3 - в данном городе более 20 человек. Чем больше, тем лучше
- Уровень медицины - введено условное обозначение, от 0 до 3 в порядке возрастания качества медицинских услуг. Чем больше, тем лучше
- Отношение стоимости 1к квартиры к ЗП. Введен данный коэффициент, как более логичный для сравнения, нежили стоимость квартир без привязки к уровню ЗП. Чем меньше, тем лучше

Город	Отношение стоимости 1к квартиры к ЗП	Стоимость 1к квартиры со сроком сдачи в 2022 году	ЗП	Ссылка
Санкт-Петербург	63	8 000 000	126 000	https://spb.cian.ru/sale/flat/265916403/
Москва	50	15 000 000	300 000	https://www.cian.ru/zhiloy-kompleks-green-park-moskva-7817/
Екатеринбург	69	5 500 000	80 000	https://ekb.cian.ru/zhiloy-kompleks-aston-sobytie-ekaterinburg-2804921/
Хельсинки	119	15 050 000	126 000	https://prian.ru/price/finland-11716-3229259.html
Сочи	143	10 000 000	70 000	https://sochi.cian.ru/zhiloy-kompleks-yuzhnoe-more-primore-mkr-46248/

Асбест	25	1 000 000	40 000	https://ekb.cian.ru/sale/flat/273485668/

- Климат+экология- введено ранжирование. Каждый вариант может набрать от 0 до 2 баллов (1 балл за хороший климат, 1 балл за хорошую экологию). Чем больше, тем лучше
- Доступность дачного участка - введено ранжирование, от 0 до 2 в порядке увеличения доступности. Чем больше, тем лучше

Далее представлены матрицы бинарных отношений, составленные с учетом выше озвученных предпочтений.

В каждом БО отмечены оранжевым города, которые являются лучшими вариантами в соответствии с методом доминирования, зеленым – в соответствии с методом блокировки

БО «Возможная ЗП»

№ п/п		Санкт-Петербург	Москва	Екатеринбург	Хельсинки	Сочи	Асбест
0	Санкт-Петербург	-	0	1	1	1	1
1	Москва	1	-	1	1	1	1
2	Екатеринбург	0	0	-	0	1	1
3	Хельсинки	1	0	1	-	1	1
4	Сочи	0	0	0	0	-	1
5	Асбест	0	0	0	0	0	-

БО «Семья и друзья»

№ п/п		Санкт-Петербург	Москва	Екатеринбург	Хельсинки	Сочи	Асбест
0	Санкт-Петербург	-	1	0	1	1	1
1	Москва	1	-	0	1	1	1
2	Екатеринбург	1	1	-	1	1	1
3	Хельсинки	1	1	0	-	1	1
4	Сочи	0	0	0	0	-	0
5	Асбест	1	1	0	1	1	-

БО «Уровень медицины»

№ п/п		Санкт-Петербург	Москва	Екатеринбург	Хельсинки	Сочи	Асбест
0	Санкт-Петербург	-	0	1	1	1	1
1	Москва	1	-	1	1	1	1
2	Екатеринбург	0	0	-	0	1	1
3	Хельсинки	1	0	1	-	1	1
4	Сочи	0	0	0	0	-	1
5	Асбест	0	0	0	0	1	-

БО «Отношение стоимости 1к квартиры к ЗП»

№ п/п		Санкт-Петербург	Москва	Екатеринбург	Хельсинки	Сочи	Асбест
0	Санкт-Петербург	-	0	1	1	1	0
1	Москва	1	-	1	1	1	0
2	Екатеринбург	0	0	-	1	1	0
3	Хельсинки	0	0	0	-	1	0
4	Сочи	0	0	0	0	-	0
5	Асбест	1	1	1	1	1	-

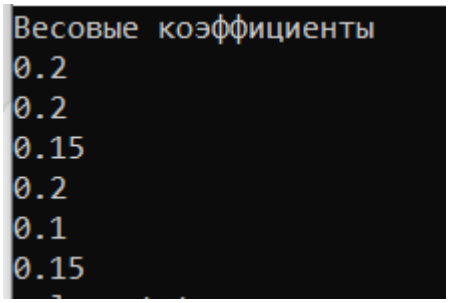
БО «Климат + экология»

№ п/п		Санкт-Петербург	Москва	Екатеринбург	Хельсинки	Сочи	Асбест
0	Санкт-Петербург	-	1	1	0	0	1
1	Москва	1	-	1	0	0	1
2	Екатеринбург	0	0	-	0	0	1
3	Хельсинки	1	1	1	-	1	1
4	Сочи	1	1	1	1	-	1
5	Асбест	0	0	1	0	0	-

БО «Доступность дачного участка»

№ п/п		Санкт-Петербург	Москва	Екатеринбург	Хельсинки	Сочи	Асбест
0	Санкт-Петербург	-	1	1	0	1	0
1	Москва	0	-	0	0	0	0
2	Екатеринбург	1	1	-	0	1	0
3	Хельсинки	1	1	1	-	1	1
4	Сочи	1	1	1	0	-	0
5	Асбест	1	1	1	1	1	-

Скриншоты работы программы



БО «Возможная ЗП»

```
salary.txt
-1      0       1       0       1       1
1       -1      1       1       1       1
0       0       -1      0       1       1
1       0       1       -1      1       1
0       0       0       0       -1      1
0       0       0       0       0       -1
Kopt
-842150451
4
-842150451
-842150451
-842150451
-842150451
+++++++
K-max механизм
3       3       3       3
5       5       5       5       строго наибольший
2       2       2       2
4       4       4       4
1       1       1       1
0       0       0       0
========
Доминирующий механизм
1
Блокирующий механизм
1
Турнирный механизм
0.6
1
0.4
0.8
0.2
0
```

БО «Семья и друзья»

```
family.txt
-1      1       0       1       1       1
1       -1      0       1       1       1
1       1       -1      1       1       1
1       1       0       -1      1       1
0       0       0       0       -1      0
1       1       0       1       1       -1
Kopt
-842150451
-842150451
4
-842150451
-842150451
-842150451
+++++++
K-max механизм
4       1       4       1
4       1       4       1
5       5       5       5       строго наибольший
4       1       4       1
0       0       0       0
4       1       4       1
========
Доминирующий механизм
2
Блокирующий механизм
2
Турнирный механизм
0.5
0.5
1
0.5
0
0.5
```

БО «Уровень медицины»

```
medicine.txt
-1      0       1       1       1       1
1       -1      1       1       1       1
0       0       -1      0       1       1
1       0       1       -1      1       1
0       0       0       0       -1      1
0       0       0       0       1       -1
Kopt
-842150451
4
-842150451
-842150451
-842150451
-842150451
+++++++
K-max механизм
4       3       4       3
5       5       5       5       строго наибольший
2       2       2       2
4       3       4       3
1       0       1       0
1       0       1       0
========
Доминирующий механизм
1
Блокирующий механизм
1
Турнирный механизм
0.525
0.75
0.3
0.525
0.075
0.075
```

БО «Отношение стоимости 1к квартиры к ЗП»

```
housecoef.txt
-1      0       1       1       1       0
1       -1      1       1       1       0
0       0       -1      1       1       0
0       0       0       -1      1       0
0       0       0       0       -1      0
1       1       1       1       1       -1
Kopt
-842150451
-842150451
-842150451
-842150451
-842150451
4
+++++++
K-max механизм
3       3       3       3
4       4       4       4
2       2       2       2
1       1       1       1
0       0       0       0
5       5       5       5       строго наибольший
========
Доминирующий механизм
5
Блокирующий механизм
5
Турнирный механизм
0.6
0.8
0.4
0.2
0
1
```

БО «Климат + экология»

```
ecology.txt
-1      1       1       0       0       1
1       -1      1       0       0       1
0       0       -1      0       0       1
1       1       1       -1      1       1
1       1       1       1       -1      1
0       0       1       0       0       -1
Kopt
-842150451
-842150451
-842150451
-842150451
-842150451
-842150451
+++++++
K-max механизм
3       2       3       2
3       2       3       2
1       0       1       0
5       4       5       4
5       4       5       4
1       0       1       0
========
Доминирующий механизм
3
4
Блокирующий механизм
Турнирный механизм
0.25
0.25
0.05
0.45
0.45
0.05
```

БО «Доступность дачного участка»

```
dacha.txt
-1      1       1       0       1       0
0       -1      0       0       0       0
1       1       -1      0       1       0
1       1       1       -1      1       1
1       1       1       0       -1      0
1       1       1       1       1       -1
Kopt
-842150451
-842150451
-842150451
-842150451
-842150451
-842150451
+++++++
K-max механизм
3       1       3       1
0       0       0       0
3       1       3       1
5       4       5       4
3       1       3       1
5       4       5       4
========
Доминирующий механизм
3
5
Блокирующий механизм
Турнирный механизм
0.3
0
0.3
0.675
0.3
0.675
```

Итоговая таблица: Выбор лучшего варианта

```
_____Механизм доминирования_____
Баллы вариантов с учетом весовых коэффициентов и места вариантов
 0       0       5
 1    0.35       1
 2     0.2       3
 3    0.25       2
 4     0.1       4
 5    0.35       1
_____Механизм блокировки______
Баллы вариантов с учетом весовых коэффициентов и места вариантов
 0       0       3
 1    0.35       1
 2     0.2       2
 3       0       3
 4       0       3
 5     0.2       2
______Турнирный механизм______
Баллы вариантов с учетом весовых коэффициентов и места вариантов
 0   2.775       3
 1     3.3       1
 2    2.45       4
 3    3.15       2
 4   1.025       6
 5     2.3       5
______Механизм K-MAX______
Баллы вариантов с учетом весовых коэффициентов и места вариантов
 0    11.1       3       0       3
 1    13.2       1       7       1
 2     9.8       4       4       2
 3    12.6       2       0       3
 4     4.1       6       0       3
 5     9.2       5       4       2
______Бальная система______
=================================================================
   ||  Блок  ||  Дом   ||  Тур  || Sjp || SjM ||  Сумма баллов  ||
=================================================================
 0       2       4         4       4       4      18
 1       6       6         6       6       6      30
 2       4       5         3       3       5      20
 3       5       4         5       5       4      23
 4       3       4         1       1       4      13
 5       6       5         2       2       5      20

C:\Users\Oks\source\repos\Сист.анализ и принятие решений\system_analysis
```

5 Вывод

В результате работы СППР в качестве наилучшего города для проживания выбирается город Москва, он набрал наибольшее количество баллов: 30 и вышел на 1 место.

На втором месте расположился город Хельсинки, который набрал 23 балла.

Также интерес представляют города Екатеринбург и Асбест, которые набрали одинаковое количество баллов по 20. Разрешению конфликта может помочь введение дополнительного предпочтения, например, по доступности сервиса, тогда город Екатеринбург будет на 3 месте опередив город Асбест.

6 Листинг

```
#include <iostream>
#include <fstream>
#include <vector>
#include <string>
#include <algorithm>
#include <iomanip>


const std::string PATH_TO_DATA0 = "salary.txt";
const std::string PATH_TO_DATA1 = "family.txt";
const std::string PATH_TO_DATA2 = "medicine.txt";
const std::string PATH_TO_DATA3 = "housecoef.txt";
const std::string PATH_TO_DATA4 = "ecology.txt";
const std::string PATH_TO_DATA5 = "dacha.txt";
const char PATH_TO_Power_DATA[] = "power.txt";
const int VARIANT = 6;
const int BO = 6;

//для машин LADA
//const std::string PATH_TO_DATA0 = "Cena_LADA.txt";
//const std::string PATH_TO_DATA1 = "GodV_LADA.txt";
//const std::string PATH_TO_DATA2 = "Probeg_LADA.txt";
//const char PATH_TO_Power_DATA[] = "powerLADA.txt";
//const int VARIANT = 3;
//const int BO = 3;

//выделение памяти дл¤ динамического двумерного массива
double** createArr(double n, double m);

//подсчет весовые коэффициенты
void readPower(double* arr, double n, std::ifstream& in);

//посчитать массив размером NxM из вход¤щего потока
void readFile(double** arr, double n, double m, std::ifstream& in);

//определение доминирующего варианта
std::vector<int> dominate(double** arr, double n, double m);

//определение блокирующего варианта
std::vector<int> block(double** arr, double n, double m);
```

```
//определение турнирного варианта
std::vector<double> turnir(double** arr, const double power[BO], double n,
double m, int number);

//составление массива дл¤ варианта в случае механизма K-max
double** createKarray(double** arr, double n, double m);

//определение K-opt вариантов
void createKopt(double** arr, double n, int* kopt_Array);

//вывести двумерный массив
void writeArr(double** arr, double n, double m);

//вывести двумерный массив K-opt механизм
void writeArrKopt(double** arr, double n, double m, int* opt);

//уничтожить двумерный массив
void distractionArray(double** arr, int n);

//расстановка мест
void  placeRating(const double arr[VARIANT], int A[VARIANT]);


int main()
{
    setlocale(LC_ALL, "RUS");
    double rating[VARIANT] = { 0 };
    double ratingBlock[VARIANT] = { 0 };
    double ratingTurnir[VARIANT] = { 0 };
    double kmax[VARIANT] = { 0 };
    double kopt[VARIANT] = { 0 };

    //считывание весовых коэффициентов
    std::ifstream in_power;
    in_power.open(PATH_TO_Power_DATA);
    double* powerArr = new double[BO];
    //powerArr - массив с весовыми коэффициентами
    readPower(powerArr, BO, in_power);
    std::cout << "Весовые коэффициенты" << std::endl;
    for (int i = 0; i < BO; ++i)
    {
        std::cout << powerArr[i] << std::endl;
    }
    in_power.close();

    for (int k = 0; k < BO; k++)
    {
        //подготовка данных
        std::ifstream in; //поток дл¤ считывани¤ основных данных
        std::string str;
        switch (k)
        {
        case 0:
            str = PATH_TO_DATA0;
            break;
        case 1:
            str = PATH_TO_DATA1;
            break;
        case 2:
            str = PATH_TO_DATA2;
            break;
        case 3:
```

```
            str = PATH_TO_DATA3;
            break;
        case 4:
            str = PATH_TO_DATA4;
            break;
        case 5:
            str = PATH_TO_DATA5;
            break;
        default:
            break;
        }
        std::cout << str << std::endl;
        in.open(str);
        double** Dom_data = createArr(VARIANT, VARIANT);
        readFile(Dom_data, VARIANT, VARIANT, in);
        writeArr(Dom_data, VARIANT, VARIANT);
        in.close();

        //обработка данных
        std::vector<int> dom_Array;
        std::vector<int> block_Array;
        std::vector<double> turnir_Array;
        //std::vector<int> kopt_Array;
        int* kopt_Array = new int[VARIANT];
        double** Karray = createKarray(Dom_data, VARIANT, VARIANT);
        createKopt(Karray, VARIANT, kopt_Array);
        std::cout << "Kopt" << std::endl;
        for (int i = 0; i < VARIANT; ++i)
        {
            std::cout << kopt_Array[i] << std::endl;
        }
        std::cout << "+++++++" << std::endl;
        std::cout << "K-max механизм " << std::endl;

        writeArrKopt(Karray, VARIANT, 4, kopt_Array);
        std::cout << "======== " << std::endl;
        dom_Array = dominate(Dom_data, VARIANT, VARIANT);//определение
доминирующих вариантов
        block_Array = block(Dom_data, VARIANT, VARIANT);//определение
блокирующих вариантов
        turnir_Array = turnir(Dom_data, powerArr, VARIANT, VARIANT,
k);//определение турнирных вариантов
        std::cout << "Доминирующий механизм" << std::endl;
        for (int i = 0; i < dom_Array.size(); ++i)
        {
            std::cout << dom_Array[i] << std::endl;
            rating[dom_Array[i]] += powerArr[k];
        }
        std::cout << "Блокирующий механизм " << std::endl;
        for (int i = 0; i < block_Array.size(); ++i)
        {
            std::cout << block_Array[i] << std::endl;
            ratingBlock[block_Array[i]] += powerArr[k];
        }
        //данные по турнирному механизму
        std::cout << "Турнирный механизм " << std::endl;
        for (int i = 0; i < turnir_Array.size(); ++i)
        {
            std::cout << turnir_Array[i] << std::endl;
            ratingTurnir[i] += turnir_Array[i];
        }
        //данные по K-max механизму
        for (int i = 0; i < VARIANT; ++i)
```

```
        {
            for (int j = 0; j < 4; j++)
            {
                kmax[i] += Karray[i][j] * powerArr[k];
            }
        }
        //данные по K-opt
        for (int i = 0; i < VARIANT; ++i)
        {
            if ((kopt_Array[i] == 1) || (kopt_Array[i] == 2) ||
(kopt_Array[i] == 3) || (kopt_Array[i] == 4))
            {
                for (int j = 0; j < 4; j++)
                {
                    kopt[i] += Karray[i][j] * powerArr[k];
                }
            }
        }
        distractionArray(Dom_data, VARIANT);
        distractionArray(Karray, VARIANT);
    }


    int rating_place[VARIANT] = { 0 };
    placeRating(rating, rating_place);
    std::cout << "_____Механизм доминирования_____" << std::endl;
    std::cout << "Баллы вариантов с учетом весовых коэффициентов и места
вариантов" << std::endl;
    for (int i = 0; i < VARIANT; ++i)
    {
        std::cout << std::setw(2) << i << std::setw(8) << rating[i] <<
std::setw(8) << rating_place[i] << std::endl;
    }

    int rating_place_block[VARIANT] = { 0 };
    placeRating(ratingBlock, rating_place_block);
    std::cout << "_____Механизм блокировки_____" << std::endl;
    std::cout << "Баллы вариантов с учетом весовых коэффициентов и места
вариантов" << std::endl;
    for (int i = 0; i < VARIANT; ++i)
    {
        std::cout << std::setw(2) << i << std::setw(8) << ratingBlock[i] <<
std::setw(8) << rating_place_block[i] << std::endl;
    }

    int rating_place_turnir[VARIANT] = { 0 };
    placeRating(ratingTurnir, rating_place_turnir);
    std::cout << "_____Турнирный механизм_____" << std::endl;
    std::cout << "Баллы вариантов с учетом весовых коэффициентов и места
вариантов" << std::endl;
    for (int i = 0; i < VARIANT; ++i)
    {
        std::cout << std::setw(2) << i << std::setw(8) << ratingTurnir[i] <<
std::setw(8) << rating_place_turnir[i] << std::endl;
    }

    int rating_place_kmax[VARIANT] = { 0 };
    int rating_place_kopt[VARIANT] = { 0 };
    placeRating(kmax, rating_place_kmax);
    placeRating(kopt, rating_place_kopt);
    std::cout << "_____Механизм K-MAX_____" << std::endl;
    std::cout << "Баллы вариантов с учетом весовых коэффициентов и места
вариантов" << std::endl;
```

```
    for (int i = 0; i < VARIANT; ++i)
    {
        std::cout << std::setw(2) << i << std::setw(8) << kmax[i] <<
std::setw(8) << rating_place_kmax[i] << std::setw(8) << kopt[i] <<
std::setw(8) << rating_place_kopt[i] << std::endl;
    }

    std::cout << "______Бальная система______" << std::endl;
    std::cout <<
"=================================================================" <<
std::endl;
    std::cout << "   ||  Блок  ||  Дом   ||  Тур  || Sjp || SjM ||  Сумма
баллов  ||" << std::endl;
    std::cout <<
"=================================================================" <<
std::endl;
    for (int i = 0; i < VARIANT; ++i)
    {
        int dom_value = VARIANT + 1 - rating_place[i];
        int block_value = VARIANT + 1 - rating_place_block[i];
        int turn_value = VARIANT + 1 - rating_place_turnir[i];
        int kmax_value = VARIANT + 1 - rating_place_kmax[i];
        int kopt_value = VARIANT + 1 - rating_place_kopt[i];
        int sum = dom_value + block_value + turn_value + kmax_value +
kopt_value;
        std::cout << std::setw(2) << i << std::setw(8) << block_value <<
std::setw(8) << dom_value << std::setw(10) << turn_value << std::setw(8) <<
kmax_value << std::setw(8) << kopt_value << std::setw(8) << sum << std::endl;
    }
    return 0;
}

//выделение памяти для динамического двухмерного массива
double** createArr(double n, double m)
{
    double** A;
    A = new double* [n];
    for (int i = 0; i < n; i++)
    {
        A[i] = new double[m];
    }
    return A;
}

//считать весовые коэффициенты
void readPower(double* arr, double n, std::ifstream& in)
{
    for (int j = 0; j < n; j++)
    {
        in >> arr[j];
    }
}

//считать массив размером NxM из входящего потока
void readFile(double** arr, double n, double m, std::ifstream& in)
{
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < m; j++)
        {
            in >> arr[i][j];
        }
    }
```

```
}

//определение доминирующего варианта
std::vector<int> dominate(double** arr, double n, double m)
{
    std::vector<int> dom_str_Array;
    bool dom_str;
    for (int i = 0; i < n; i++)
    {
        dom_str = true;
        for (int j = 0; j < m; j++)
        {
            if (i == j) continue;
            if (arr[i][j] != 1)
            {
                dom_str = false;
                break;
            }
        }
        if (dom_str) dom_str_Array.push_back(i);

    }
    return dom_str_Array;
}

//определение блокирующего варианта
std::vector<int> block(double** arr, double n, double m)
{
    std::vector<int> block_str_Array;
    bool block_str;
    for (int i = 0; i < n; i++)
    {
        block_str = true;
        for (int j = 0; j < m; j++)
        {
            if (i == j) continue;
            if (arr[j][i] != 0)
            {
                block_str = false;
                break;
            }
        }
        if (block_str) block_str_Array.push_back(i);

    }
    return block_str_Array;
}

//определение турнирного варианта
std::vector<double> turnir(double** arr, const double power[BO], double n,
double m, int number)
{
    std::vector<double> turnir_str_Array;
    bool turnir_str;
    for (int i = 0; i < n; i++)
    {
        double sum = 0;
        for (int j = 0; j < m; j++)
        {
            if (i == j) continue;
            if (arr[i][j] == 1)
            {
                if (arr[j][i] == 0)
```

```
                {
                    sum += power[number];
                }
                else if (arr[j][i] == 1)
                {
                    sum += power[number] / 2;
                }
            }
        }
        turnir_str_Array.push_back(sum);
    }
    return turnir_str_Array;
}

//оставление массива для варианта в случае механизма K-max
double** createKarray(double** arr, double n, double m)
{
    double** A = createArr(VARIANT, 4);
    for (int i = 0; i < n; i++)
    {
        double HR0 = 0;
        double ER = 0;
        double NK = 0;
        for (int j = 0; j < m; j++)
        {
            if (i == j) continue;
            if (arr[i][j] == 1)
            {
                if (arr[j][i] == 0)
                {
                    HR0 += 1;
                }
                else if (arr[j][i] == 1)
                {
                    ER += 1;
                }
            }
            if (arr[i][j] == -1)
            {
                NK += 1;
            }
        }
        for (int j = 0; j < 4; j++)
        {
            switch (j)
            {
            case 0:
                A[i][j] = HR0 + ER + NK;
                break;
            case 1:
                A[i][j] = HR0 + NK;
                break;
            case 2:
                A[i][j] = HR0 + ER;
                break;
            case 3:
                A[i][j] = HR0;
                break;
            default:
                break;
            }
        }
    }
```

```
    return A;
};

//определение K-opt вариантов
void createKopt(double** arr, double n, int* kopt_Array)
{
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < 4; j++)
        {
            switch (j)
            {
            case 0:
                if (arr[i][j] == n)
                {
                    kopt_Array[i] = 1;
                }
                break;
            case 1:
                if ((arr[i][j] == (n - 1)) && (arr[i][j] > arr[i][j + 2]))
                {
                    kopt_Array[i] = 2;
                }
                break;
            case 2:
                if ((arr[i][j] == n) && (arr[i][j] > arr[i][j + 1]))
                {
                    kopt_Array[i] = 3;
                }
                break;
            case 3:
                if ((arr[i][j] == (n - 1)) && (arr[i][j] == arr[i][j - 1]) &&
(arr[i][j] == arr[i][j - 2]))
                {
                    kopt_Array[i] = 4;
                }
                break;
            default:
                kopt_Array[i] = 0;
                break;
            }
        }
    }
}


//вывести двумерный массив
void writeArr(double** arr, double n, double m)
{
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < m; j++)
        {
            std::cout << arr[i][j] << "\t";
        }
        std::cout << std::endl;
    }
}

//вывести двумерный массив K-opt механизм
void writeArrKopt(double** arr, double n, double m, int* opt)
{
    for (int i = 0; i < n; i++)
```

```
    {
        for (int j = 0; j < m; j++)
        {
            std::cout << arr[i][j] << "\t";
        }
        switch (opt[i])
        {
        case 0:
            break;
        case 1:
            std::cout << "максимальный" << "\t";
            break;
        case 2:
            std::cout << "строго максимальный" << "\t";
            break;
        case 3:
            std::cout << "наибольший" << "\t";
            break;
        case 4:
            std::cout << "строго наибольший" << "\t";
            break;
        default:
            break;
        }
        std::cout << std::endl;
    }
}

//уничтожить двумерный массив
void distractionArray(double** arr, int n)
{
    for (int i = 0; i < n; ++i)
    {
        delete[] arr[i];
    }
    delete[] arr;
}

//расстановка мест
void  placeRating(const double arr[VARIANT], int A[VARIANT])
{
    double place[VARIANT] = { 0 };
    int number[VARIANT];
    for (int i = 0; i < VARIANT; ++i)
    {
        number[i] = i + 1;
    }
    for (int i = 0; i < VARIANT; i++)
    {
        place[i] = arr[i];
    }
    std::sort(std::begin(place), std::end(place));
    std::reverse(std::begin(place), std::end(place));
    int pl = 0;
    for (int i = 0; i < VARIANT; ++i)//по массиву place
    {
        if ((place[i] == place[i - 1]) && (i != 0))
        {
            continue;
        }
        for (int j = 0; j < VARIANT; j++)//по массиву arr
        {
            if (arr[j] == place[i])
```

```
            {
                //A[j] = i + 1;
                A[j] = number[pl];
            }
        }
        pl++;
        if (place[i] == 0) break;
    }
}
```