Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Высшая школа программной инженерии

КУРСОВАЯ РАБОТА

по дисциплине «Современные проблемы программной инженерии»

по теме:

«Применение СППР для выбора инструментария для систем оркестрации и IaC»

Выполнил студент:
гр. № 3540904/10101 Томилин И. С.

Руководитель:
к.т.н. Амосов В. В.

Санкт-Петербург
2022 г.

ОГЛАВЛЕНИЕ

1. Введение

Качественный подход к принятию решения подразумевает формализацию ситуации с использованием аппарата бинарных отношений или предпочтений, задание для каждого бинарного отношения весового коэффициента, проведение выбора решений с помощью аппарата функций выбора, поиск оптимальных вариантов решений для каждого бинарного отношения и сведение полученных результатов в обобщающую таблицу для наглядности и автоматизации выбора лицом, принимающим решение.

В системе применяются для оценки предпочтений бинарные отношения «больше» и «меньше»

Аппарат функций выбора задаётся для каждого бинарного отношения механизмами доминирования, блокировки, турнирным механизмом и механизмом определения К - максимальных вариантов. Для каждого бинарного отношения задаются весовые коэффициенты. Полученные по каждому механизму результаты ранжируются с учётом весовых коэффициентов бинарных отношений.

2. Исходные данные для проверки

Для тестирования системы используются данные, загружаемые из файлов, как стандартные тестовые данные шаблона системы поддержки принятия решения на основе качественного подхода, так и приведенные ниже данные.

В качестве тестирования корректности расчета и вывода результатов был взят шаблон выбора наилучшей модели автомобиля LADA (ВАЗ). Цель тестирования, убедиться в корректности работы приложения.

Варианты

Сравниваются три автомобиля:

LADA (ВАЗ) Priora I, 2008 (1вариант)	LADA (ВАЗ) Priora I, 2009 (2 вариант)	LADA (ВАЗ) Priora I, 2009 (3 вариант)
143000 руб.	150000 руб.	148000 руб.
Пробег 170000 км	Пробег 140000 км	Пробег 150000 км
1.6 л / 98 л. с. / Бензин	1.6 л / 81 л. с. / Бензин	1.6 л / 81 л. с. / Бензин
Механическая	Механическая	Механическая
Серый, Передний, Седан	Серый, Передний, Седан	Белый, Передний, Хэтчбек 5 дверей

Названия вариантов вводим в файл следующим образом:

```
1 lada_1_variant
2 lada_2_variant
3 lada_3_variant
```

Каждый вариант с новой строки через нижнее подчеркивание.

Предпочтения

Выбраны следующие данные о предпочтениях:

Поля для ввода предпочтений		
Цена	Год выпуска	Пробег
Ввести предпочтения	**Добавить предпочтение**	**Обработать добавление**
Поля для ввода весовых коэффициентов предпочтений		
0.3	0.3	0.4
Ввести весовые коэффициенты		

Файл для ввода предпочтений нужно ввести в файл следующим образом:

```
1 Цена 0.3
2 Год_выпуска 0.3
3 Пробег 0.4
```

Названия предпочтений вводим через нижнее подчеркивание, затем через пробел вводим коэффициент.
Чтение коэффициентов и вариантов указывается в коде, количество бинарных отношений и вариантов нужно также задать:

```
#ifdef TEST
const char PATH_TO_Power_DATA[] = "./data/lada_power.txt";
const std::string PROD_NAMES = "./data/lada_names.txt";

const int VARIANT = 3;
const int BO = 3;
#endif
```

Бинарные отношения

Матрицы бинарных отношений генерируются автоматически, нужно только ввести два вектора:
1) варианты предпочтения;
2) вектор определяющий преобладание предпочтения (например чем новее год выпуска автомобиля тем лучше, 2009 предпочтительнее чем 2008 или наоборот чем меньше пробег тем лучше, 170000км менее предпочтительней чем 140000км)

```
std::vector<std::vector<int>> prep
{
    { 143000, 150000, 148000 },
    { 2008, 2009, 2009 },
    { 170000, 140000, 150000 },
};

// false если меньшее значение преобладает над большим.
// true если большее значение предобладает над меньшим.
std::vector<bool> prep_bm
{
    false,
    true,
    false,
};
```

Таблица предпочтений вариантов			
	Цена	**Год выпуска**	**Пробег**
LADA Priora 1	143000	2008	170000
LADA Priora 2	150000	2009	140000
LADA Priora 3	148000	2009	150000
Создать таблицу	**Ввести данные таблицы**		**Удалить таблицу**

3. Результаты работы программы по теме «Применение СППР для выбора инструментария для систем оркестрации и IaC»

Результат работы на основе тестовых данных:

```
Предпочтения:
Цена: 0.3
Год_выпуска: 0.3
Пробег: 0.4
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Цена
-1      1       1
0       -1      0
0       1       -1
Kopt
[0] = 4
[1] = -1
[2] = -1
+++++++
K-max механизм
2       2       2       2       строго наибольший
0       0       0       0
1       1       1       1
========
Доминирующий механизм
0
Блокирующий механизм
0
Турнирный механизм
0.6
0
0.3
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Год_выпуска
-1      0       0
1       -1      1
1       1       -1
Kopt
[0] = -1
[1] = -1
[2] = -1
+++++++
K-max механизм
0       0       0       0
2       1       2       1
2       1       2       1
========
Доминирующий механизм
1
2
Блокирующий механизм
Турнирный механизм
0
0.45
0.45
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
```

```
Пробег
-1      0       0
1       -1      1
1       0       -1
Kopt
[0] = -1
[1] = 4
[2] = -1
+++++++
K-max механизм
0       0       0       0
2       2       2       2       строго наибольший
1       1       1       1
========
Доминирующий механизм
1
Блокирующий механизм
1
Турнирный механизм
0
0.8
0.4
_____Механизм доминирования_____
Баллы вариантов с учетом весовых коэффициентов и места вариантов
lada_1_variant        0.3       2
lada_2_variant        0.7       1
lada_3_variant        0.3       2
_____Механизм блокировки______
Баллы вариантов с учетом весовых коэффициентов и места вариантов
lada_1_variant        0.3       2
lada_2_variant        0.4       1
lada_3_variant          0       3
______Турнирный механизм______
Баллы вариантов с учетом весовых коэффициентов и места вариантов
lada_1_variant        0.6       3
lada_2_variant       1.25       1
lada_3_variant       1.15       2
______Механизм K-MAX______
Баллы вариантов с учетом весовых коэффициентов и места вариантов
lada_1_variant        2.4       3     2.4       2
lada_2_variant          5       1     3.2       1
lada_3_variant        4.6       2       0       3
______Бальная система______
================================================================================
             ||  Блок  ||  Дом  ||  Тур  ||  Sjp  ||  SjM  ||  Сумма баллов  ||
================================================================================
lada_1_variant     2       2       1       1       2           8
lada_2_variant     3       3       3       3       3          15
lada_3_variant     1       2       2       2       1           8
```

Результат работы на основе собственных данных:

Выбор системы IaC:

```
Предпочтения:
Синтаксис_yaml: 0.2
Поддержка_сообществом: 0.3
Безагентнотсь: 0.2
Идемпотентность: 0.1
Качество_документации: 0.2
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
./data/iac_1_file_type.txt
-1      1       1       1       1
0       -1      0       0       0
0       0       -1      0       0
0       0       0       -1      0
0       0       0       0       -1
Kopt
[0] = 4
[1] = -1
[2] = -1
[3] = -1
[4] = -1
+++++++
K-max механизм
4       4       4       4       строго наибольший
0       0       0       0
0       0       0       0
0       0       0       0
0       0       0       0
========
Доминирующий механизм
0
Блокирующий механизм
0
Турнирный механизм
0.8
0
0
0
0
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
./data/iac_2_com.txt
-1      1       1       1       1
0       -1      1       1       1
0       0       -1      0       1
0       1       0       -1      1
0       0       0       0       -1
Kopt
[0] = 4
[1] = -1
[2] = -1
[3] = -1
[4] = -1
+++++++
K-max механизм
4       4       4       4       строго наибольший
3       2       3       2
1       1       1       1
2       1       2       1
0       0       0       0
========
Доминирующий механизм
0
Блокирующий механизм
0
Турнирный механизм
```

```
1.2
0.75
0.3
0.45
0
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
./data/iac_3_noagent.txt
-1      1       1       1       1
0       -1      1       0       1
0       0       -1      0       0
0       1       1       -1      1
0       0       0       0       -1
Kopt
[0] = 4
[1] = -1
[2] = -1
[3] = -1
[4] = -1
+++++++
K-max механизм
4       4       4       4       строго наибольший
2       2       2       2
0       0       0       0
3       3       3       3
0       0       0       0
========
Доминирующий механизм
0
Блокирующий механизм
0
Турнирный механизм
0.8
0.4
0
0.6
0
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
./data/iac_4_iden.txt
-1      1       1       1       1
1       -1      1       1       1
0       0       -1      1       0
0       1       0       -1      0
0       0       0       0       -1
Kopt
[0] = -1
[1] = -1
[2] = -1
[3] = -1
[4] = -1
+++++++
K-max механизм
4       3       4       3
4       2       4       2
1       1       1       1
1       0       1       0
0       0       0       0
========
Доминирующий механизм
0
1
Блокирующий механизм
Турнирный механизм
0.35
```

```
0.3
0.1
0.05
0
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
./data/iac_5_doc.txt
-1      1       1       1       1
1       -1      1       1       1
0       0       -1      0       0
1       0       1       -1      0
0       0       0       1       -1
Kopt
[0] = -1
[1] = -1
[2] = -1
[3] = -1
[4] = -1
+++++++
K-max механизм
4       2       4       2
4       3       4       3
0       0       0       0
2       1       2       1
1       1       1       1
========
Доминирующий механизм
0
1
Блокирующий механизм
Турнирный механизм
0.6
0.7
0
0.3
0.2
_____Механизм доминирования_____
Баллы вариантов с учетом весовых коэффициентов и места вариантов
ansible                  1       1
salt                   0.3       2
puppet                   0       3
chef                     0       3
pulumi                   0       3
_____Механизм блокировки______
Баллы вариантов с учетом весовых коэффициентов и места вариантов
ansible                0.7       1
salt                     0       2
puppet                   0       2
chef                     0       2
pulumi                   0       2
______Турнирный механизм______
Баллы вариантов с учетом весовых коэффициентов и места вариантов
ansible               3.75       1
salt                  2.15       2
puppet                 0.4       4
chef                   1.4       3
pulumi                 0.2       5
______Механизм K-MAX______
Баллы вариантов с учетом весовых коэффициентов и места вариантов
ansible                 15       1    11.2       1
salt                   8.6       2       0       2
puppet                 1.6       4       0       2
chef                   5.6       3       0       2
pulumi                 0.8       5       0       2
```

```
______Бальная система______
==============================================================================
             ||  Блок  ||  Дом  ||  Тур  ||  Sjp  ||  SjM  ||  Сумма баллов  ||
==============================================================================
ansible            5        5        5        5        5          25
salt               4        4        4        4        4          20
puppet             4        3        2        2        4          15
chef               4        3        3        3        4          17
pulumi             4        3        1        1        4          13
```

Выбор оркестратора:

```
Предпочтения:
Простота_использования: 0.2
Поддержка_сообществом: 0.1
Масштабируемость: 0.1
Интеграция_vault_из_коробки: 0.2
Документация: 0.3
Поддержка_ui_из_коробки: 0.1
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
./data/orch_1_using.txt
-1      0       0       0       0
1       -1      1       1       1
1       1       -1      1       1
1       0       0       -1      1
1       0       0       1       -1
Kopt
[0] = -1
[1] = -1
[2] = -1
[3] = -1
[4] = -1
+++++++
K-max механизм
0       0       0       0
4       3       4       3
4       3       4       3
2       1       2       1
2       1       2       1
========
Доминирующий механизм
1
2
Блокирующий механизм
Турнирный механизм
0
0.7
0.7
0.3
0.3
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
./data/orch_2_com.txt
-1      1       1       1       1
1       -1      1       1       1
0       0       -1      0       0
0       0       0       -1      1
0       1       1       1       -1
Kopt
[0] = -1
[1] = -1
[2] = -1
[3] = -1
[4] = -1
+++++++
K-max механизм
4       3       4       3
4       2       4       2
0       0       0       0
1       0       1       0
3       1       3       1
========
```

```
Доминирующий механизм
0
1
Блокирующий механизм
Турнирный механизм
0.35
0.3
0
0.05
0.2
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
./data/orch_3_scaling.txt
-1      1       1       1       1
0       -1      1       1       1
0       0       -1      0       0
0       1       1       -1      1
0       1       1       1       -1
Kopt
[0] = 4
[1] = -1
[2] = -1
[3] = -1
[4] = -1
+++++++
K-max механизм
4       4       4       4       строго наибольший
3       1       3       1
0       0       0       0
3       1       3       1
3       1       3       1
========
Доминирующий механизм
0
Блокирующий механизм
0
Турнирный механизм
0.4
0.2
0
0.2
0.2
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
./data/orch_4_vault.txt
-1      0       0       0       0
1       -1      1       1       1
0       0       -1      0       0
0       0       0       -1      0
0       0       0       0       -1
Kopt
[0] = -1
[1] = 4
[2] = -1
[3] = -1
[4] = -1
+++++++
K-max механизм
0       0       0       0
4       4       4       4       строго наибольший
0       0       0       0
0       0       0       0
0       0       0       0
========
```

```
Kopt
[0] = -1
[1] = 4
[2] = -1
[3] = -1
[4] = -1
+++++++
K-max механизм
0       0       0       0
4       4       4       4       строго наибольший
0       0       0       0
0       0       0       0
0       0       0       0
========
Доминирующий механизм
1
Блокирующий механизм
1
Турнирный механизм
0
0.8
0
0
0
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
./data/orch_5_doc.txt
-1      0       0       0       0
1       -1      1       1       1
0       0       -1      0       0
1       0       1       -1      0
1       1       1       1       -1
Kopt
[0] = -1
[1] = -1
[2] = -1
[3] = -1
[4] = -1
+++++++
K-max механизм
0       0       0       0
4       3       4       3
0       0       0       0
2       2       2       2
4       3       4       3
========
Доминирующий механизм
1
4
Блокирующий механизм
Турнирный механизм
0
1.05
0
0.6
1.05
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
./data/orch_6_ui.txt
-1      0       0       0       0
1       -1      1       1       1
0       0       -1      1       1
1       1       1       -1      1
1       1       1       1       -1
```

```
Kopt
[0] = -1
[1] = -1
[2] = -1
[3] = -1
[4] = -1
+++++++
K-max механизм
0       0       0       0
4       2       4       2
2       0       2       0
4       1       4       1
4       1       4       1
========
Доминирующий механизм
1
3
4
Блокирующий механизм
Турнирный механизм
0
0.3
0.1
0.25
0.25
_____Механизм доминирования_____
Баллы вариантов с учетом весовых коэффициентов и места вариантов
k8s                   0.2       3
Nomad                 0.9       1
docker_swarm          0.2       3
mesos                 0.1       4
openshift             0.4       2
_____Механизм блокировки______
Баллы вариантов с учетом весовых коэффициентов и места вариантов
k8s                   0.1       2
Nomad                 0.2       1
docker_swarm            0       3
mesos                   0       3
openshift               0       3
______Турнирный механизм______
Баллы вариантов с учетом весовых коэффициентов и места вариантов
k8s                  0.75       5
Nomad                3.35       1
docker_swarm          0.8       4
mesos                 1.4       3
openshift               2       2
______Механизм K-MAX______
Баллы вариантов с учетом весовых коэффициентов и места вариантов
k8s                     3       5     1.6       2
Nomad                13.4       1     3.2       1
docker_swarm          3.2       4       0       3
mesos                 5.6       3       0       3
openshift               8       2       0       3
______Бальная система______
=============================================================================
             ||  Блок  ||  Дом  ||  Typ  ||  Sjp  ||  SjM  ||  Сумма баллов  ||
=============================================================================
k8s                4       3       1       1       4          13
Nomad              5       5       5       5       5          25
docker_swarm       3       3       2       2       3          13
mesos              3       2       3       3       3          14
openshift          3       4       4       4       3          18
```

4. Вывод

Произведена модификация исходного кода студента гр. в3530904/00030 О.С. Клабукова, исправлена часть недочетов, введены новые возможности. Работа программы протестирована на данных шаблона СППР, в результате работы приложения, были получены данные как в шаблоне, что может сказать о правильности работы программы.
Также результат работы программы проверен на собственных примерах для принятия решения в выборе инструментария оркестратора и системы IaC.

5. Листинг программы

Файл сборки **CMakeLists.txt**:

```
cmake_minimum_required(VERSION 3.5)

project(sppr LANGUAGES CXX)

set(CMAKE_CXX_STANDARD 11)
set(CMAKE_CXX_STANDARD_REQUIRED ON)

option(USE_IAC  "IAC SPPR" OFF)
option(USE_ORCH "ORCH SPPR" OFF)
option(USE_TEST "TEST SPPR" OFF)

if (USE_IAC)
   add_definitions(-DIAC)
endif()

if (USE_ORCH)
   add_definitions(-DORCH)
endif()

if (USE_TEST)
   add_definitions(-DTEST)
endif()
add_executable(sppr main.cpp)
```

Сценарий запуска **build.sh**:

```
#!/usr/bin/env bash

set -e

MODE=$1
if [[ -z $MODE ]]; then
   echo "[ ERROR ] First arg must be a mode (IAC|ORCH)."
   exit 1
fi

rm -Rf ./build
mkdir build && cd build

if [[ $MODE == "IAC" ]]; then
   cmake -DUSE_IAC=ON .. && make
elif [[ $MODE == "ORCH" ]]; then
   cmake -DUSE_ORCH=ON .. && make
elif [[ $MODE == "TEST" ]]; then
   cmake -DUSE_TEST=ON .. && make
else
   echo "[ ERROR ] Not handled mode."
   exit 1
fi
```

```
cp sppr ../
cd ..
./sppr
```

Исходный код программы **main.cpp**:

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

#ifdef TEST
const char PATH_TO_Power_DATA[] = "./data/lada_power.txt";
const std::string PROD_NAMES = "./data/lada_names.txt";

const int VARIANT = 3;
const int BO = 3;
#endif

//выделение памяти дл¤ динамического двумерного массива
double** createArr(int n, int m);
//подсчет весовые коэффициенты
void readPower(double* arr, double n, std::ifstream& in,
        std::vector<std::string> &pref_names);
//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]);

void read_from_file( std::vector<std::string> &vec, std::ifstream& in )
{
```

```
    std::string data {};
    while( in >> data )
    {
        vec.emplace_back( data );
    }
}

void build_matrix( std::vector<int> vars, bool greater, double** matrix )
{
    for ( int i = 0; i < vars.size(); ++i )
    {
        for ( int j = 0; j < vars.size(); ++j )
        {
            if ( i == j )
            {
                matrix[i][j] = -1;
                continue;
            }

            if ( !greater && vars.at(i) <= vars.at(j) )
            {
                matrix[i][j] = 1;
                continue;
            }

            if ( greater && vars.at(i) >= vars.at(j) )
            {
                matrix[i][j] = 1;
            }
        }
    }
}

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 };

    std::vector<std::vector<int>> prep
    {
        { 143000, 150000, 148000 },
        { 2008, 2009, 2009 },
        { 170000, 140000, 150000 },
    };

    // false если меньшее значение преобладает над большим.
    // true если большее значение предобладает над меньшим.
    std::vector<bool> prep_bm
```

```
    {
       false,
       true,
       false,
    };

    // Read names.
    std::vector<std::string> names;
    std::ifstream in_names;
//    in_names.exceptions(std::ios::failbit | std::ios::badbit);
    in_names.open(PROD_NAMES.c_str());
    read_from_file( names, in_names );

    //считывание весовых коэффициентов
    std::ifstream in_power;
    in_power.exceptions(std::ios::failbit | std::ios::badbit);
    in_power.open(PATH_TO_Power_DATA);

    double* powerArr = new double[BO];
    //powerArr - массив с весовыми коэффициентами
    std::vector<std::string> pref_names;
    readPower(powerArr, BO, in_power, pref_names);

    double sum { 0 };
    sum = std::accumulate(powerArr, powerArr + BO, sum );
    std::cout << "Sum of weight coeff: " << sum << std::endl;
    std::cout << "Предпочтения:" << std::endl;
    for (int i = 0; i < BO; ++i)
    {
       std::cout << pref_names[i] << ": " << powerArr[i] << std::endl;
    }
    in_power.close();
    for (int k = 0; k < prep.size(); k++)
    {
       std::cout <<
"<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>" <<
std::endl;
       std::cout << pref_names[k] << std::endl;

       double** Dom_data = createArr(VARIANT, VARIANT);
       build_matrix( prep[k], prep_bm[k], Dom_data );
       //readFile(Dom_data, VARIANT, VARIANT, in);
       writeArr(Dom_data, VARIANT, VARIANT);

       //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];
       std::fill_n( kopt_Array, VARIANT, -1 );
       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 << '[' << i << "] = " << 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(0) << names[i] << std::setw(25 - names[i].size()) << 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(0) << names[i] << std::setw(25 - names[i].size()) <<
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(0) << names[i] << std::setw(25 - names[i].size()) <<
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(0) << names[i] << std::setw(25 - names[i].size()) << 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;

   int winner = VARIANT + 1 - rating_place[0] +
            VARIANT + 1 - rating_place_block[0] +
         VARIANT + 1 - rating_place_turnir[0] +
         VARIANT + 1 - rating_place_kmax[0] +
         VARIANT + 1 - rating_place_kopt[0]
         ;
   int index { 0 };
   for (int i = 1; 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;
      if ( winner < sum ) {
         winner = sum;
         index = i;
      }
   }

   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;
      if ( i == index ) {
         // Unix only (light text).
         std::cout << "\033[1;31m"  << std::setw(0) << names[i] << std::setw(20 -
names[i].size()) << block_value <<
                  std::setw(8) << dom_value << std::setw(9) << turn_value << std::setw(9) <<
                  kmax_value << std::setw(9) << kopt_value << std::setw(13) << sum << "\
033[0m" << std::endl;
      } else {
         std::cout << std::setw(0) << names[i] << std::setw(20 - names[i].size()) <<
block_value <<
                  std::setw(8) << dom_value << std::setw(9) << turn_value << std::setw(9) <<
```

```
                    kmax_value << std::setw(9) << kopt_value << std::setw(13) << sum <<
std::endl;
        }
    }
    return 0;
}
//выделение памяти для динамического двухмерного массива
double** createArr(int n, int m)
{
    double** A = new double* [n];
    for (int i = 0; i < n; i++)
    {
        A[i] = new double[m];
    }
    // Fill zeros.
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            A[i][j] = 0;
        }
    }
    return A;
}
//считать весовые коэффициенты
void readPower(double* arr, double n, std::ifstream& in,
               std::vector<std::string> &pref_names)
{
    std::string name {""};
    for (int j = 0; j < n; j++)
    {
        in >> name;
        pref_names.emplace_back( name );
        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;
   }
}
```