Федеральное агентство по образованию

Пермский национальный исследовательский политехнический университет

Кафедра вычислительной математики и механики

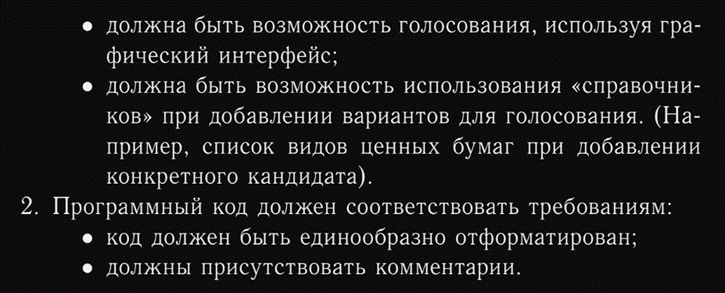
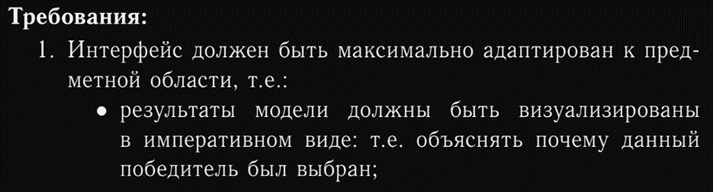
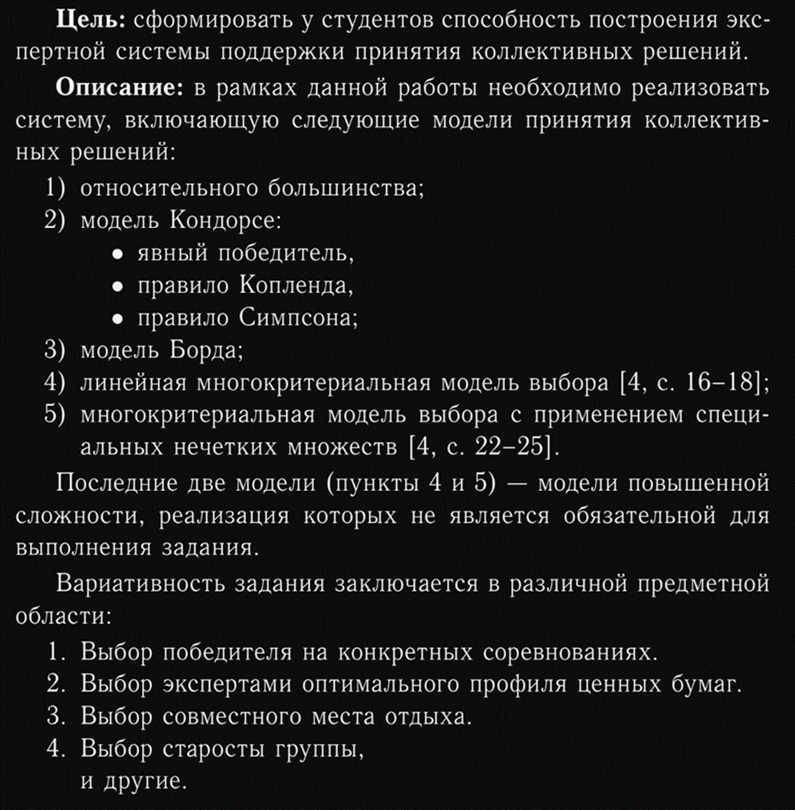
ЛАБОРАТОРНАЯ РАБОТА № 4

Реализация моделей принятия коллективных решений

Выполнил:

студент гр. ИСТ-19-2б А.А. Михайлишин

Пермь 2022



1. **Описание логики**

Сначала спрашиваются и заполняются кандидаты и выборщики. Потом предлагается на выбор один из 2 способов ПР, это модель Борда или модель относительного большинства. После чего проводится голосование, в соответствии с выбранной моделью и подсчитываются голоса. По анализу подсчитанных голосов делается вывод о победителе, если сумма голосов будет равна, то выбрать победителя не получится и придётся проводить голосование заново.

1. **Модель Борда**

Всем голосующим, предлагается про ранжировать кандидатов, выставив им места с 1 по p. За первое место даётся p-1 балл, за последние место даётся 0 баллов. Потом голоса для каждого из кандидатов суммируются и выбирается кандидат с наибольшим количеством баллов. В случаях равенства голосов, необходимо проводить голосование заново.

1. **Модель относительного большинства**

Всем голосующим предлагается выбрать одно из кандидатов, потом подсчитывается количество голосов за каждого из кандидатов. И выбирается кандидат с наибольшим количеством голосов. В случаях равенства голосов, необходимо проводить голосование заново.

**4. Код оболочки на языке C#**

internal class Program

{

public static List<String> listCandidates;

public static List<String> listElectors;

static void Main(string[] args)

{

Console.WriteLine("Приветсвую вас Странник!");

// Кандидаты

Console.WriteLine("Введите число кандидатов для голосования");

int candidates = Vvod(9);

Console.WriteLine();

// Список кандидатов

Console.WriteLine("Заполните список кандидатов");

listCandidates = VvodList(candidates);

Console.WriteLine();

// Выборщики

Console.WriteLine("Введите число выборщиков для голосования");

int electors = Vvod(9);

Console.WriteLine();

// Список выборщиков

Console.WriteLine("Заполните список выборщиков");

listElectors = VvodList(electors);

Console.WriteLine();

// Использование соответсвующей модели

while (true)

{

// Выбор модели ПР

Console.WriteLine("Выберите модель для ПР:");

Console.WriteLine("1) Модель Борда");

Console.WriteLine("2) Модель относительного большинства");

Console.WriteLine("3) Закрыть программу");

int model = Vvod(3);

Console.WriteLine();

switch (model)

{

case 1:

Bord();

break;

case 2:

Greater();

break;

case 3:

Console.WriteLine("До встречи, всего хорошего!!!");

Environment.Exit(0);

break;

}

}

}

public static int Vvod(int max)

{

while (true)

{

string input = Console.ReadLine();

for (int i = 1; i <= max; i++)

{

if (input.Equals(Convert.ToString(i))) {

Console.ForegroundColor = ConsoleColor.Green;

Console.WriteLine("Значение сохранено!");

Console.ForegroundColor = ConsoleColor.Black;

return Convert.ToInt16(input);

}

}

Console.WriteLine("Ошибка ввода, попробуйте снова");

}

}

public static List<String> VvodList(int count)

{

List<String> list = new List<String>();

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

{

Console.WriteLine( (i+1) + " член");

list.Add(Console.ReadLine());

Console.ForegroundColor = ConsoleColor.Green;

Console.WriteLine("Значение сохранено!");

Console.ForegroundColor = ConsoleColor.Black;

}

return list;

}

public static void Bord()

{

Console.WriteLine("Проведём голосование");

int[,] mass = new int[listElectors.Count, listCandidates.Count];

int[] sum = new int[listCandidates.Count];

// Опрос

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

{

Console.WriteLine("Опрашивается " + listElectors[i]);

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

{

Console.WriteLine("Выставьте место для " + listCandidates[j] + " от {1 до " + listCandidates.Count + "}");

mass[i, j] = Vvod(listCandidates.Count);

}

}

Console.WriteLine();

// Вывод голосов

Console.WriteLine("Матрица голосов");

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

{

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

{

Console.Write(listCandidates.Count - mass[i, j] + "\t");

}

Console.WriteLine();

}

// Подсчёт голосов

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

{

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

{

sum[j] += listCandidates.Count - mass[i, j];

}

}

// Вывод суммы баллов

Console.WriteLine("Сумма голосов");

for (int i = 0; i < sum.Length; i++)

{

Console.Write(sum[i] + "\t");

}

Console.WriteLine();

int winnerId = Array.IndexOf(sum, sum.Max());

if (Array.FindAll(sum, value => value == sum.Max()).Length > 1)

Console.WriteLine("Невозможно определить победителя, так как количество голосов совпадает");

else

Console.WriteLine("Поздравляю " + listCandidates[winnerId] + " с победой!!!");

}

public static void Greater()

{

Console.WriteLine("Проведём голосование");

int[,] mass = new int[listElectors.Count, listCandidates.Count];

int[] sum = new int[listCandidates.Count];

// Опрос

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

{

Console.WriteLine("Опрашивается " + listElectors[i]);

Console.WriteLine("Отдайте свой голос за одного из кандидатов, от {1 до " + listCandidates.Count + "}");

int index = Vvod(listCandidates.Count);

mass[i, index-1] = 1;

}

Console.WriteLine();

// Вывод голосов

Console.WriteLine("Матрица голосов");

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

{

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

{

Console.Write(mass[i, j] + "\t");

}

Console.WriteLine();

}

// Подсчёт голосов

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

{

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

{

sum[j] += mass[i, j];

}

}

// Вывод суммы баллов

Console.WriteLine("Сумма голосов");

for (int i = 0; i < sum.Length; i++)

{

Console.Write(sum[i] + "\t");

}

Console.WriteLine();

int winnerId = Array.IndexOf(sum, sum.Max());

if (Array.FindAll(sum, value => value == sum.Max()).Length > 1)

Console.WriteLine("Невозможно определить победителя, так как количество голосов совпадает");

else

Console.WriteLine("Поздравляю " + listCandidates[winnerId] + " с победой!!!");

}

}