Министерство образования и науки Российской Федерации

Севастопольский государственный университет

Кафедра ИС

Лабораторная работа № 2

“ИССЛЕДОВАНИЕ ПРИМЕНЕНИЯ АППАРАТА ТЕОРИИ ПОЛЕЗНОСТИ ДЛЯ ОПИСАНИЯ БИНАРНЫХ ОТНОШЕНИЙ ПРИ ПРИНЯТИИ РЕШЕНИЙ”

Выполнил:

ст. гр. ИС-17-2о Горбенко К.Н.

Проверил

Кротов К.В.

Севастополь

2020

1 ЦЕЛЬ

Исследовать применение аппарата теории полезности при принятии решений по выбору альтернатив.

2 ПОСТАНОВКА ЗАДАЧИ

Для второго варианта задания предусматривается следующий порядок действий по выполнению лабораторной работы:

1) реализовать инициализацию матриц отношений строго предпочтения A_1 и эквивалентности A_2;

2) реализовать процедуру, формирующую на основе матрицы отношения эквивалентности A_2 классы эквивалентности $R(x_i)$ ($x_i \in X$);

3) реализовать процедуру, выполняющую сравнение полученных классов эквивалентности $R(x_i)$ ($x_i \in X$), исключение повторяющихся классов, формирующую множество $X/\sim$ уникальных классов эквивалентности решений k_l;

4) реализовать процедуру, выполняющую упорядочивание классов эквивалентности k_l с определение соответствующих им значений функции полезности $U(k_l)$;

5) реализовать процедуру, которая выполняет инициализацию значений функции полезности элементов (решений) $U(x_i)$ множества X, входящих в соответствующие классы эквивалентности k_l, значениями функции полезности этих классов $U(k_l)$; разрабатываемая процедура также выполняет упорядочивание решений $x_i \in X$ с точки зрения значений их функции полезности и определяет решение $x_i^* \in X$, для которого значение функции полезности является максимальным;

6) реализовать вывод исходных данных, промежуточных и конечных результатов: матриц отношений A_1 и A_2, классов эквивалентности $R(x_i)$ ($x_i \in X$), множества $X/\sim$ не повторяющихся ("уникальных") классов эквивалентности, полученных значений функции полезности $U(k_l)$ для каждого класса k_l, значений функции полезности для решений $x_i \in X$, соответствующих этим классам, эффективных решений с максимальным значением функции полезности.

3 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

Код класса TPRL2:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace Lab2_WpfApp1
{
    class TPRL2
    {
        public struct polezn
        {
            public int N;
            public int pol;

            public polezn(int _N, int _pol)
            {
                N = _N;
                pol = _pol;
            }
        }

        private bool[, ] EqlMatrix;
        private bool[, ] DomMatrix;
        private bool[,] DomClas;
        private List<int>[] EqlClasses;
        private List<int>[] UEqlClasses;
        private polezn[] Polezn;
        private polezn[] PoleznClas;


        public int EqlXCount
        {
            get
            {
                return(int) Math.Sqrt(EqlMatrix.Length);
            }
        }
        public int ClasCount
        {
            get
            {
                return (int)Math.Sqrt(DomClas.Length);
            }
        }
        public bool[,] PEqlMatrix
        {
            get
            {
                return EqlMatrix;
            }
        }
        public bool[,] PDomMatrix
        {
            get
            {
                return DomMatrix;
            }
        }
        public bool[,] PDomClas
        {
            get
            {
                return DomClas;
            }
        }
        public List<int>[] PEqlClasses
        {
            get
            {
                return EqlClasses;
```

```
    }
}
public List<int>[] PUEqlClasses
{
    get
    {
        return UEqlClasses;
    }
}
public polezn[] PPolezn
{
    get
    {
        return Polezn;
    }
}
public polezn[] PPoleznClas
{
    get
    {
        return PoleznClas;
    }
}

public int EnterEqlMatrix(string text)
{
    string temp = "";
    List<int> templist = new List<int>();
    foreach(char c in text)
    {
        if ((c == ',') || (c == ';'))
        {
            try { templist.Add(int.Parse(temp)); }
            catch (Exception) { return -1; }
            finally { temp = ""; }
        }
        else temp += c;
    }
    double temp2 = Math.Sqrt(templist.Count);
    if (temp2 - Math.Truncate(temp2) != 0 ) return -1;

    EqlMatrix = new bool[(int)temp2, (int)temp2];
    int itemp = 0;
    for (int i = 0; i < temp2; i++) for (int j = 0; j < temp2; j++)
        {
            EqlMatrix[i, j] = ((templist[itemp] == 0) ? (false) : (true));
            itemp++;
        }
    return 0;
}

public int EnterDomMatrix(string text)
{
    string temp = "";
    List<int> templist = new List<int>();
    foreach (char c in text)
    {
        if ((c == ',') || (c == ';'))
        {
            try { templist.Add(int.Parse(temp)); }
            catch (Exception) { return -1; }
            finally { temp = ""; }
        }
        else temp += c;
    }
```

```
            double temp2 = Math.Sqrt(templist.Count);
            if (temp2 - Math.Truncate(temp2) != 0) return -1;

            DomMatrix = new bool[(int)temp2, (int)temp2];
            int itemp = 0;
            for (int i = 0; i < temp2; i++) for (int j = 0; j < temp2; j++)
                {
                    DomMatrix[i, j] = ((templist[itemp] == 0) ? (false) : (true));
                    itemp++;
                }
            return 0;
        }

        public void Start()
        {
            GenEqlClases();
            GenUEqlClases();
            GenDomClas();
            GenPoleznClas();
            GenPolezn();
            SortPolezn();
        }

        private int GenEqlClases()
        {
            if (EqlMatrix == null) return -1;
            if (EqlMatrix.Length == 0) return -1;

            EqlClasses = new List<int>[(int)Math.Sqrt(EqlMatrix.Length)];
            for (int i = 0; i < EqlClasses.Length; i++) EqlClasses[i] = new List<int>();

            for(int i = 0; i < EqlClasses.Length; i++) for(int j = 0; j <
EqlClasses.Length; j++) if (EqlMatrix[i, j] == true) EqlClasses[i].Add(j);

            return 0;
        }

        private int GenUEqlClases()
        {
            if (EqlClasses == null) return -1;
            if (EqlClasses.Length == 0) return -1;

            List<List<int>> temp = new List<List<int>>();

            foreach (List<int> Li in EqlClasses)
            {
                if (!isIn(temp, Li)) temp.Add(Li);
            }

            UEqlClasses = new List<int>[temp.Count];
            for (int i = 0; i < temp.Count; i++) UEqlClasses[i] = new List<int>(temp[i]);
            return 0;
        }

        private int GenDomClas()
        {
            if (UEqlClasses == null) return -1;
            if (UEqlClasses.Length == 0) return -1;

            DomClas = new bool[UEqlClasses.Length, UEqlClasses.Length];

            for(int i = 0; i < UEqlClasses.Length; i++)
            {
                for(int j = 0; j < UEqlClasses.Length; j++)
                {
```

```
                DomClas[i, j] = isIn3(UEqlClasses[i], UEqlClasses[j]);
            }
        }

        return 0;
    }

    private int GenPoleznClas()
    {
        if (UEqlClasses == null) return -1;
        if (UEqlClasses.Length == 0) return -1;

        PoleznClas = new polezn[ClasCount];

        bool temp2 = false;
        int no = 0;
        for(int i = 0; i < ClasCount; i++)
        {
            for(int j = 0; j < ClasCount; j++)
            {
                if (DomClas[i, j])
                {
                    temp2 = true;
                    break;
                }
            }
            if (!temp2)
            {
                no = i;
                break;
            }
        }
        PoleznClas[no] = new polezn(0, 0);

        int temp3 = no;
        int temp4 = 0;
        while (true)
        {
            temp4 = temp3;
            temp3 = GetLuchChem(temp3);
            if (temp3 == -1) break;
            PoleznClas[temp3] = new polezn(PoleznClas[temp4].N, PoleznClas[temp4].pol
+ 1);
        }
        return 0;
    }

    private int GenPolezn()
    {
        if (PoleznClas == null) return -1;
        if (PoleznClas.Length == 0) return -1;

        Polezn = new polezn[EqlXCount];

        for (int i = 0; i < EqlXCount; i++)
        {
            Polezn[i] = new polezn(i, GetPolForResh(i));
        }

        return 0;
    }

    private int SortPolezn()
    {
        for (int i = 0; i < Polezn.Length; i++)
```

```
            {
                for (int j = 0; j < Polezn.Length - 1; j++)
                {
                    if (Polezn[j].pol > Polezn[j + 1].pol)
                    {
                        polezn z = Polezn[j];
                        Polezn[j] = Polezn[j + 1];
                        Polezn[j + 1] = z;
                    }
                }
            }
            return 0;
        }

        private int GetLuchChem(int n)
        {
            for(int i = 0; i < ClasCount; i++)
            {
                if(i != n) if (DomClas[i, n]) return i;
            }
            return -1;
        }

        private int GetPolForResh(int n)
        {
            for (int i = 0; i < UEqlClasses.Length; i++)
            {
                foreach (int j in UEqlClasses[i]) if (j == n) return PoleznClas[i].pol;
            }
            return -1;
        }

        private bool isIn3(List<int> list1, List<int> list2)
        {
            foreach(int i in list1)
            {
                foreach(int j in list2)
                {
                    if (DomMatrix[i, j]) return true;
                }
            }

            return false;
        }

        private bool isIn(List<List<int>> LL, List<int> L)
        {
            foreach(List<int> li in LL)
            {
                if (isIn2(li, L)) return true;
            }

            return false;
        }

        private bool isIn2(List<int> list1, List<int> list2)
        {
            foreach(int i in list1)
            {
                foreach (int j in list2) if (i == j) return true;
            }

            return false;
        }
    }
```

```
}
```

Код класса MainWindow.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace Lab2_WpfApp1
{
    /// <summary>
    /// Логика взаимодействия для MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        private TPRL2 tprl;

        public MainWindow()
        {
            InitializeComponent();
            tprl = new TPRL2();
        }

        private void B1_Click(object sender, RoutedEventArgs e)
        {
            if (tprl.EnterEqlMatrix(TB1.Text) == -1) MessageBox.Show("Ошибка ввода
матрица эквивалентности", "Ошибка");
            if (tprl.EnterDomMatrix(TB2.Text) == -1) MessageBox.Show("Ошибка ввода
матрицы доминирования", "Ошибка");
            tprl.Start();

            List<int>[] temp = tprl.PEqlClasses;
            if (temp == null) return;
            TB3.Text = "";
            for (int i = 0; i < temp.Length; i++)
            {
                TB3.Text += "R(X" + (i + 1) + ") = { ";
                foreach (int j in temp[i]) TB3.Text += "X" + (j + 1) + " , ";
                TB3.Text += " }" + '\n';
            }

            List<int>[] temp2 = tprl.PUEqlClasses;
            if (temp2 == null) return;
            TB4.Text = "";
            for (int i = 0; i < temp2.Length; i++)
            {
                TB4.Text += "K" + (i + 1) + " = { ";
                foreach (int j in temp2[i]) TB4.Text += "X" + (j + 1) + " , ";
                TB4.Text += " }" + '\n';
            }
```

```
            TB5.Text = "";
            for (int i = 0; i < tprl.ClasCount; i++)
            {
                for(int j = 0; j < tprl.ClasCount; j++)
                {
                    TB5.Text += (tprl.PDomClas[i, j]?(1):(0)) + ", ";
                }
                TB5.Text += '\n';
            }

            TB6.Text = "";
            for (int i = 0; i < tprl.ClasCount; i++)
            {
                TB6.Text += "U(K" + (tprl.PPoleznClas[i] .N+ 1) + ") = " +
tprl.PPoleznClas[i].pol + '\n';
            }

            TB7.Text = "";
            for (int i = 0; i < tprl.EqlXCount; i++)
            {
                TB7.Text += "U(X" + (tprl.PPolezn[i].N +1) + ") = " + tprl.PPolezn[i].pol
+ '\n';
            }
        }
    }
}
```

4 ПРИМЕРЫ ВЫПОЛНЕНИЯ

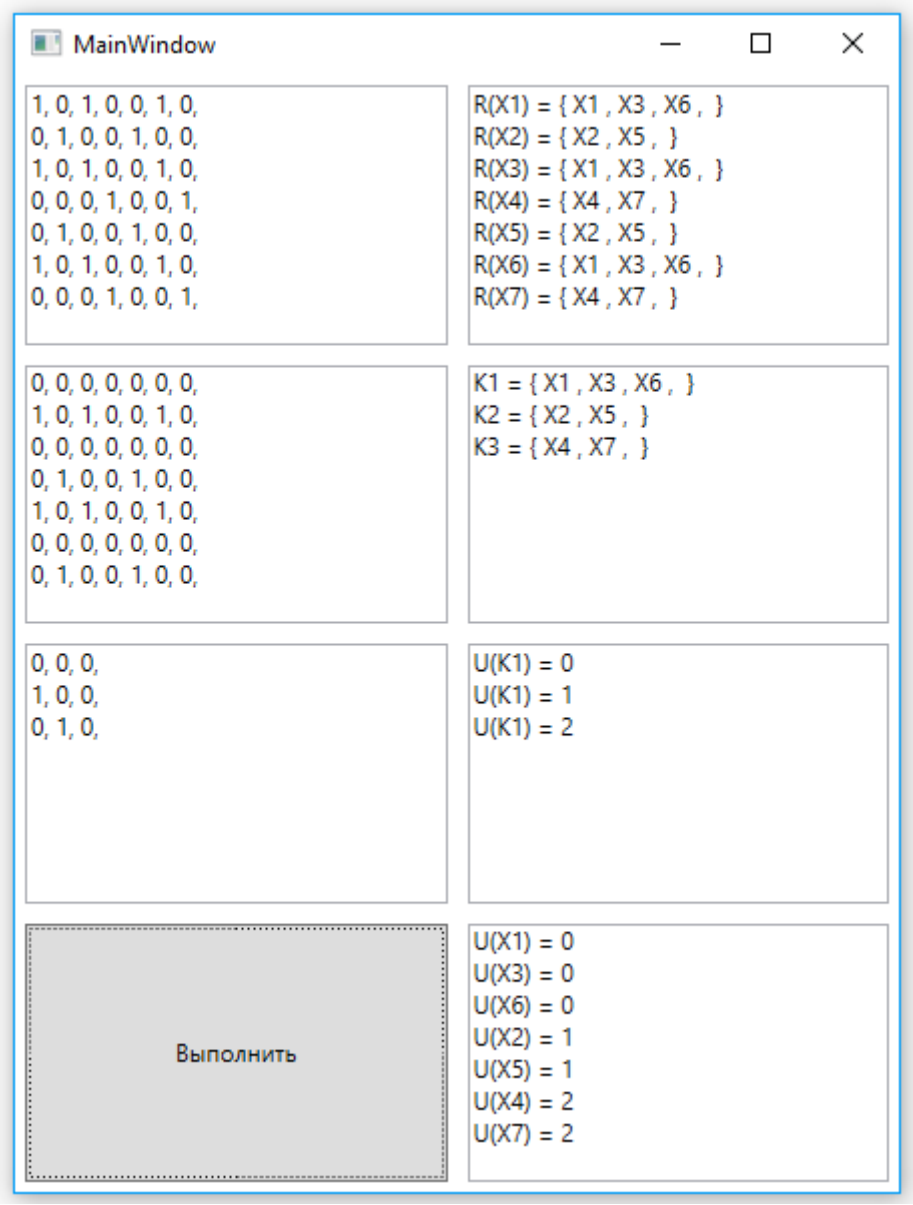

Рисунок 1 — Выполнение программы

ВЫВОДЫ

В ходе работы была написана и протестирована программа реализующей определение классов эквивалентности, уникальных классов эквивалентности, а так же значений полезности для каждого класса эквивалентности и решения. Программа показала свою работоспособность, которая была подтверждена вышеуказанными тестовыми примерами. Результаты программного и аналитического решения полностью совпали.