Филиал федерального государственного бюджетного образовательного учреждения высшего профессионального образования«Национальный исследовательский университет «МЭИ»в г. Смоленске

Кафедра вычислительной техники

Отчёт о выполнении лабораторной работы №2

по курсу: «Интеллектуальные системы»

Студент: Батулев А.И.

Восканян К. К.

Группа: ВМ-16 (маг)

Бригада: 8

Преподаватель: Зернов М. М.

Смоленск, 2016

Лабораторная работа проводится с использованием оболочки экспертной системы продукционного типа **ExpertSystem**. С помощью **ExpertSystem** создать базу знаний, состоящую минимум из 30 правил и написать программу на любом языке высокого уровня, реализующую вывод на основе созданной базы знаний. Тематика базы знаний приводится для каждого варианта в задании.

Тема – Строительство (выбор материалов);

Метод вывода – обратный;

Метод разрешения конфликтов – выбор последнего правила из конфликтного множества;

Метод вычисления достоверности – минимум из достоверности правила и достоверности посылки.



Рисунок 1 – Схема базы знаний

Исходные:

1 – Особенность - Требуется ремонт

2 – Особенность - Требуются материалы для внутренней отделки

55 - Особенность – Требуется напольное покрытие

Промежуточные:

3 - Тип – Материалы по внутренней отделке

4 – Особенность – Ремонт стен

5 – Особенность – Ремонт потолка

6 – Особенность – Ремонт пола

7 – Подтип - Материалы по внутренней отделке стен

8 – Особенность – Влагостойкие

9 – Особенность – Невлагостойкие

10 – Подтип – Влагостойкие материалы по внутренней отделке стен

11 – Подтип – Невлагостойкие материалы по внутренней отделке стен

12 – Подтип – Натуральные материалы по внутренней отделке стен

13 – Подтип – Искусственные материалы по внутренней отделке стен

15 – Особенность – Невлагостойкие и натуральные материалы по внутренней отделке стен (дерево)

16 – Особенность – Влагостойкие и ненатуральные материалы по внутренней отделке стен (плитка, керамогранит)

17 – Особенность – Невлагостойкие и ненатуральные материалы по внутренней отделке стен (обои, панели, краска)

18 – Особенность –Материалы по внутренней отделке стен, обладающие высокой износостойкостью

19 – Особенность – Материалы по внутренней отделке стен, не обладающие высокой износостойкостью

41 - Подтип - Материалы для отделки потолка

24 – Особенность – Высокая стоимость

25 – Особенность - Низкая стоимость

26 – Подтип – Материалы для отделки потолка с высокой стоимостью

27 – Подтип – Материалы для отделки потолка с низкой стоимостью

28 - Особенность – Материалы для многоуровневой отделки потолка

29 - Особенность – Материалы для не многоуровневой отделки потолка

32 - Особенность – Простота отделки потолка

33 - Особенность – Широкая цветовая гамма отделки потолка

34 - Особенность – Рельефная отделка потолка

38 - Подтип - Материалы для отделки пола

39 - Особенность – Повышенная износостойкость

40 - Особенность – Обычная износостойкость

41 - Подтип – Материалы с повышенной износостойкостью

42 - Подтип – Материалы с обычной износостойкостью

43 – Особенность – Высокая стоимость

44 – Особенность - Низкая стоимость

47 – Особенность – Не износостойкие материалы с высокой стоимостью

48 – Особенность – Не износостойкие материалы с низкой стоимостью

49 – Подтип – Натуральные напольные материалы

50 – Подтип – Искусственные напольные материалы

Результирующие:

14 – Результат – Натуральный камень

20 – Результат – Керамогранит

21 – Результат – Керамическая настенная плитка

22 – Результат – Панели

23 – Результат – Обои/краска

30 – Результат – Гипсокартон

31 – Результат – Натяжной потолок

35 – Результат – Побелка потолочная

36 – Результат – Краска

37 – Результат – Потолочная плитка

45 – Результат – Керамическая напольная плитка

46 – Результат – Бетонная напольная плитка

51 – Результат – Паркет

52 – Результат – Ламинат

53 – Результат – Доска

54 – Результат – Линолеум.

Пример работы программы приведен на рисунке 2.

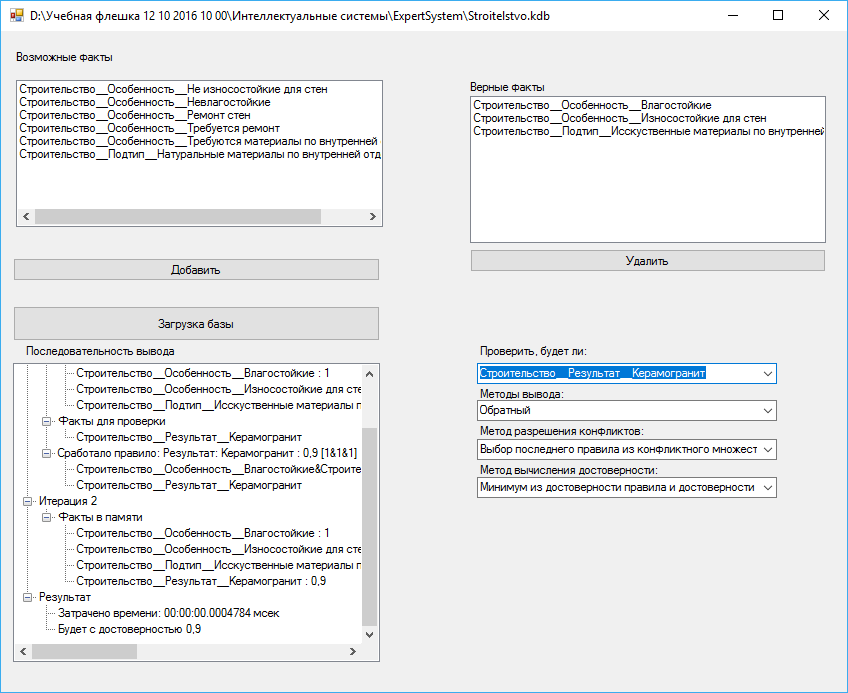
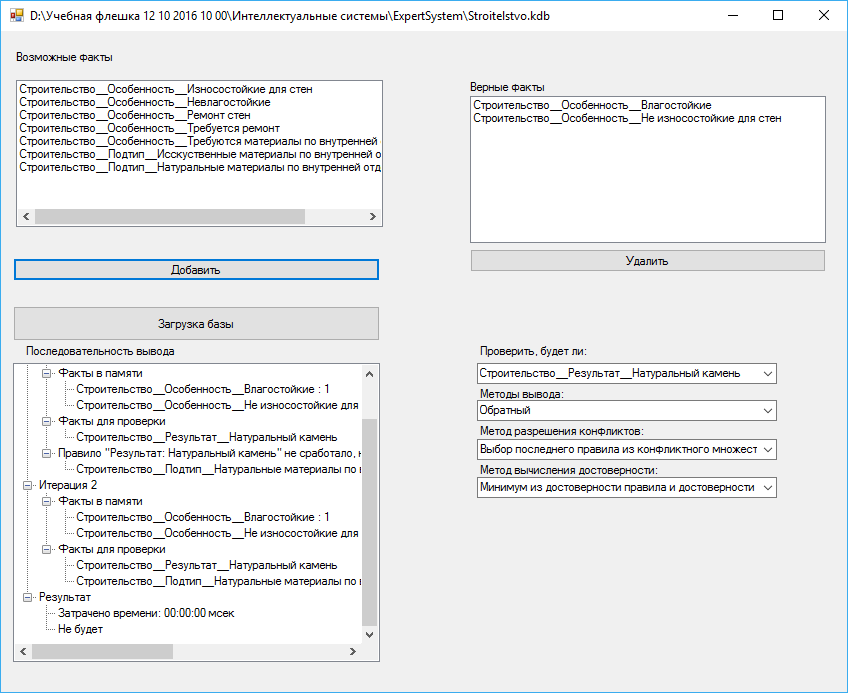
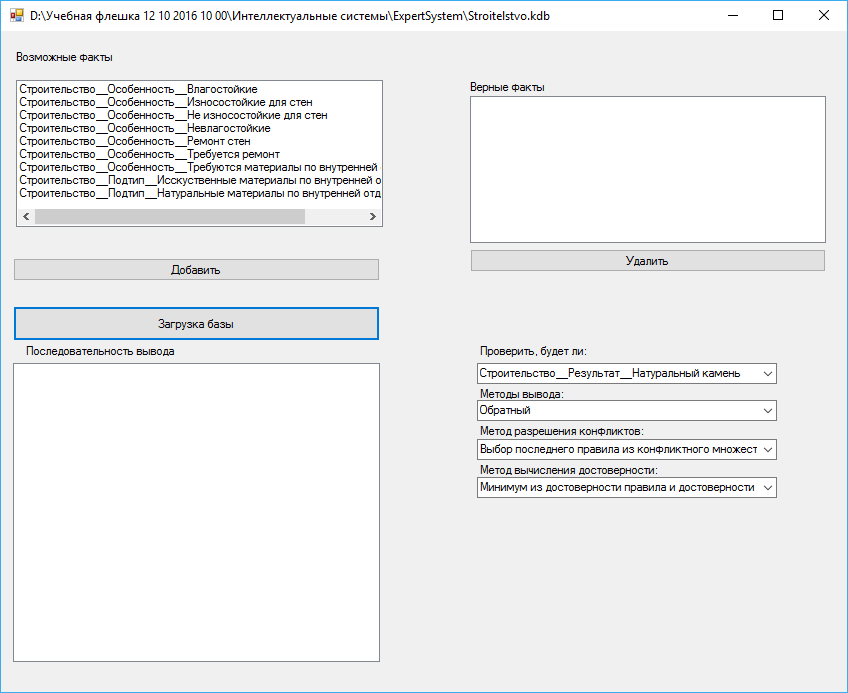
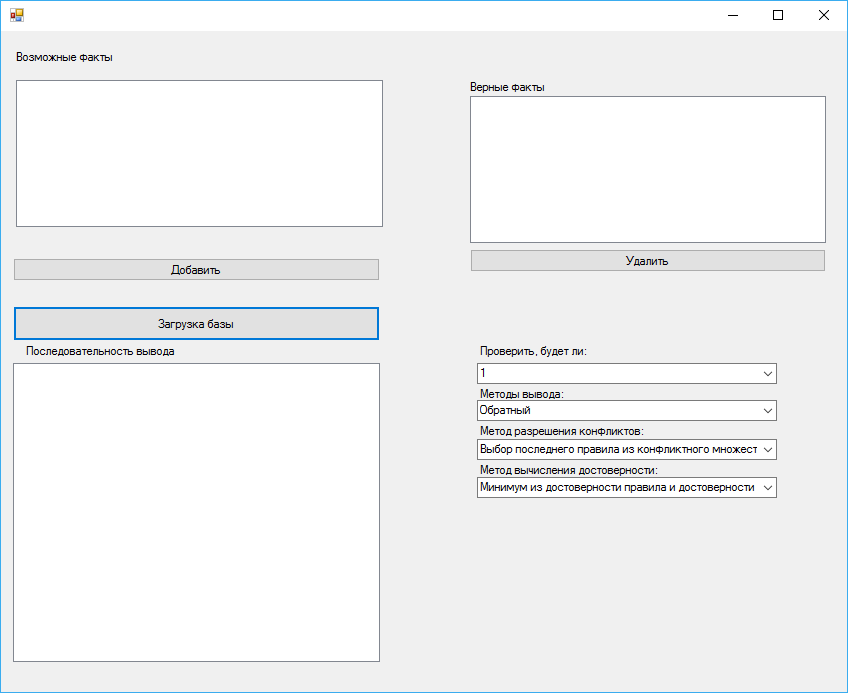


Рисунок 2 - Пример работы программы

Текст программы

using System;

using System.Collections;

using System.IO;

using System.Windows.Forms;

namespace Lab2\_8var

{

class MainFormPresenter

{

private readonly MainForm \_mainForm;

private ArrayList facts = new ArrayList();

private ArrayList rules\_Copy = new ArrayList();

private ArrayList rules = new ArrayList();

public MainFormPresenter(MainForm mainForm)

{

\_mainForm = mainForm;

}

public void LoadData()

{

if (\_mainForm.LoadDataDialog.ShowDialog() != DialogResult.OK)

return;

\_mainForm.Text = \_mainForm.LoadDataDialog.FileName;

var fileStream = new FileStream(\_mainForm.LoadDataDialog.FileName, FileMode.Open);

try

{

BinaryReader br = new BinaryReader(fileStream);

if (br.ReadInt32() != 2000590686)

throw new Exception("неверный заголовок.");

if (br.ReadInt32() != 268435456)

throw new Exception("неверная версия базы знаний.");

int num1 = br.ReadInt32();

facts.Clear();

for (int index = 0; index < num1; ++index)

{

Fact fact = new Fact(0, "", "", "", 0.0, FactType.Intermediate);

fact.ReadFromFile(br);

facts.Add(fact);

}

int num2 = br.ReadInt32();

rules.Clear();

for (int index = 0; index < num2; ++index)

{

Rule rule = new Rule(0);

rule.ReadFromFile(br);

rules.Add(rule);

}

rules\_Copy = new ArrayList(rules);

\_mainForm.UpdateAllFactsList(facts);

}

catch (Exception ex)

{

int num = (int)MessageBox.Show("Ошибка при загрузке файла: " + ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Hand);

}

fileStream.Close();

\_mainForm.UpdateTrueFactsList(facts);

\_mainForm.UpdateFactToCheck(facts);

// GetWorkResult();

}

public void GetWorkResult()

{

\_mainForm.RSTree.Nodes.Clear();

if (\_mainForm.TrueFactsList.Items.Count <= 0 || \_mainForm.FactToCheck.Items.Count <= 0)

{

return;

}

var now = DateTime.Now;

var work\_facts = new ArrayList();

var work\_truths = new ArrayList();

foreach (Fact fact in \_mainForm.TrueFactsList.Items)

{

work\_facts.Add(fact.ToString());

work\_truths.Add(fact.Truth);

}

// Отключаю перерисовку TreeView.

\_mainForm.RSTree.BeginUpdate();

var node1 = new TreeNode("Проверяем, будет ли:");

\_mainForm.RSTree.Nodes.Add(node1);

node1.Nodes.Add(new TreeNode(\_mainForm.FactToCheck.Text));

rules = new ArrayList(rules\_Copy);

GetReverseWorkIterations(work\_facts, work\_truths);

TreeNode node2 = new TreeNode($"Затрачено времени: {DateTime.Now - now} мсек");

TreeNode node3 = new TreeNode("Результат");

\_mainForm.RSTree.Nodes.Add(node3);

TreeNode node4 = new TreeNode();

int index = work\_facts.IndexOf(\_mainForm.FactToCheck.SelectedItem.ToString());

if (index == -1)

{

node4.Text = "Не будет";

}

else

{

node4.Text = "Будет с достоверностью " + work\_truths[index];

}

node3.Nodes.Add(node2);

node3.Nodes.Add(node4);

\_mainForm.RSTree.ExpandAll();

\_mainForm.RSTree.EndUpdate();

}

/// <summary>

/// Алгоритм обратного вывода.

/// </summary>

/// <param name="workFacts"></param>

/// <param name="workTruths"></param>

private void GetReverseWorkIterations(ArrayList workFacts, ArrayList workTruths)

{

ArrayList arrayList1 = new ArrayList();

ArrayList arrayList2 = new ArrayList();

ArrayList arrayList3 = new ArrayList();

ArrayList arrayList4 = new ArrayList();

ArrayList arrayList5 = new ArrayList();

arrayList2.Add(\_mainForm.FactToCheck.SelectedItem.ToString());

int num1 = 0;

bool flag1 = true;

while (flag1)

{

var node1 = new TreeNode($"Итерация {num1 + 1}");

\_mainForm.RSTree.Nodes.Add(node1);

var node2 = new TreeNode("Факты в памяти");

node1.Nodes.Add(node2);

for (int index = 0; index < workFacts.Count; ++index)

{

var treeNode = new TreeNode((string)workFacts[index] + " : " + ((double)workTruths[index]));

node2.Nodes.Add(treeNode);

}

if (arrayList2.Count > 0)

{

var node3 = new TreeNode("Факты для проверки");

node1.Nodes.Add(node3);

foreach (string str in arrayList2)

{

var treeNode = new TreeNode(str);

node3.Nodes.Add(treeNode);

}

}

if (workFacts.Contains(\_mainForm.FactToCheck.SelectedItem.ToString()))

break;

arrayList1.Clear();

arrayList3.Clear();

arrayList4.Clear();

arrayList5.Clear();

flag1 = false;

foreach (string str1 in arrayList2)

{

foreach (Rule rule in rules)

{

bool flag2 = false;

for (int index = 0; index < rule.Conclusions.Count; ++index)

{

int num2 = (int)rule.Conclusions[index];

foreach (Fact fact in facts)

{

if (fact.ID == num2 && fact.ToString() == str1)

{

flag2 = true;

break;

}

}

}

if (flag2)

{

ArrayList arrayList6 = new ArrayList();

string exp = "";

string str2 = "";

for (int index = 0; index < rule.Condition.Length; ++index)

{

char ch = rule.Condition[index];

switch (ch)

{

case '(':

case ')':

case '&':

case '|':

if (str2 != "")

{

string sFact = str2.Trim();

exp += (workFacts.Contains(sFact) ? '1' : '0').ToString();

if (IsValidFact(sFact))

arrayList6.Add(sFact);

str2 = "";

}

char[] chars = { ch };

string exp111 = new string(chars);

exp += exp111;

break;

default:

char[] chars1 = { ch };

string str2111 = new string(chars1);

str2 += str2111;

break;

}

}

if (str2 != "")

{

string sFact = str2.Trim();

char[] chars2 = { (char)(workFacts.Contains(sFact) ? 49 : 48) };

string exp3 = new string(chars2);

exp += exp3;

if (IsValidFact(sFact))

arrayList6.Add(sFact);

}

if (CheckExpression(exp))

{

bool flag3 = true;

for (int index = 0; index < rule.Conclusions.Count; ++index)

{

Fact factById = GetFactById((int)rule.Conclusions[index]);

if (factById != null && !workFacts.Contains(factById.ToString()))

{

flag3 = false;

break;

}

}

if (!flag3)

{

arrayList4.Add(rule);

arrayList5.Add(exp);

}

}

else

{

TreeNode node3 = null;

foreach (string text in arrayList6)

{

if (!arrayList2.Contains(text) && !arrayList3.Contains(text) && !workFacts.Contains(text))

{

if (node3 == null)

{

node3 = new TreeNode("Правило \"" + rule.Name + "\" не сработало, но из посылки следующие факты были добавлены в список фактов для проверки:");

node1.Nodes.Add(node3);

}

arrayList3.Add(text);

TreeNode node4 = new TreeNode(text);

node3.Nodes.Add(node4);

flag1 = true;

}

}

}

}

}

}

foreach (string str in arrayList3)

{

if (!arrayList2.Contains(str))

arrayList2.Add(str);

}

Rule rule1 = arrayList4.Count > 0 ? (Rule)arrayList4[arrayList4.Count - 1] : null;

string str3 = arrayList5.Count > 0 ? (string)arrayList5[0] : null;

if (arrayList4.Count > 1)

{

rule1 = ResolveConflict(node1, arrayList4, arrayList5, rule1, ref str3);

}

if (rule1 != null)

{

var node3 = new TreeNode("Сработало правило: " + rule1.Name + " : " + rule1.Truth + " [" + str3 + "]");

node1.Nodes.Add(node3);

var node4 = new TreeNode(rule1.Condition);

node3.Nodes.Add(node4);

for (int index = 0; index < rule1.Conclusions.Count; ++index)

{

Fact factById = GetFactById((int)rule1.Conclusions[index]);

if (factById != null)

{

TreeNode node5 = new TreeNode(factById.ToString());

node3.Nodes.Add(node5);

}

}

foreach (int fact\_id in rule1.Conclusions)

{

Fact factById = GetFactById(fact\_id);

if (factById != null)

{

var fact = new Fact(-1, factById.Object, factById.Attribute, factById.Value, rule1.Truth, factById.Type);

if (arrayList1.IndexOf(fact) == -1)

arrayList1.Add(fact);

}

}

}

foreach (Fact fact in arrayList1)

{

int index = workFacts.IndexOf(fact.ToString());

if (index == -1)

{

workFacts.Add(fact.ToString());

workTruths.Add(fact.Truth);

flag1 = true;

arrayList2.Remove(fact.ToString());

}

else

{

switch (\_mainForm.TruthMethod.SelectedIndex)

{

case 0:

if (fact.Truth < (double)workTruths[index])

{

workTruths[index] = fact.Truth;

}

continue;

case 1:

workTruths[index] = fact.Truth;

continue;

default:

continue;

}

}

}

++num1;

}

}

/// <summary>

/// Метод разрешения конфликтов.

/// </summary>

/// <param name="node1"></param>

/// <param name="arrayList4"></param>

/// <param name="arrayList5"></param>

/// <param name="rule1"></param>

/// <param name="str3"></param>

/// <returns></returns>

private Rule ResolveConflict(TreeNode node1, ArrayList arrayList4, ArrayList arrayList5, Rule rule1, ref string str3)

{

var node3 = new TreeNode("Конфликтное множество");

node1.Nodes.Add(node3);

for (int index1 = 0; index1 < arrayList4.Count; ++index1)

{

Rule rule2 = (Rule)arrayList4[index1];

var node4 = new TreeNode(rule2.Name + " : " + rule2.Truth + " [" + (string)arrayList5[index1] + "]");

node3.Nodes.Add(node4);

var node5 = new TreeNode(rule2.Condition);

node4.Nodes.Add(node5);

for (int index2 = 0; index2 < rule2.Conclusions.Count; ++index2)

{

Fact factById = GetFactById((int)rule2.Conclusions[index2]);

if (factById != null)

{

var node6 = new TreeNode(factById.ToString());

node4.Nodes.Add(node6);

}

}

switch (\_mainForm.ConflictMethod.SelectedIndex)

{

case 0:

if (rule2.Truth > rule1.Truth)

{

rule1 = rule2;

str3 = (string)arrayList5[index1];

}

break;

case 1:

rule1 = rule2;

str3 = (string)arrayList5[index1];

break;

case 2:

int num2 = 0;

int num3 = 0;

for (int index2 = 0; index2 < rule2.Condition.Length; ++index2)

{

if (rule2.Condition[index2] == 124 || rule2.Condition[index2] == 38)

++num2;

}

for (int index2 = 0; index2 < rule2.Condition.Length; ++index2)

{

if (rule2.Condition[index2] == 124 || rule2.Condition[index2] == 38)

++num3;

}

if (num2 > num3)

{

rule1 = rule2;

str3 = (string)arrayList5[index1];

}

break;

}

}

return rule1;

}

/// <summary>

/// Вычисление достоверности.

/// </summary>

/// <param name="exp"></param>

/// <returns></returns>

private bool CheckExpression(string exp)

{

Stack stack = new Stack();

char ch1 = ' ';

for (int index = 0; index < exp.Length; ++index)

{

char ch2 = exp[index];

if (ch2 == ')')

{

string exp1 = "";

while (stack.Count > 0 && (ch1 = (char)stack.Pop()) != '(')

exp1 += ch1.ToString();

if (ch1 != '(')

return false;

stack.Push((CheckExpression(exp1) ? '1' : '0'));

}

else

stack.Push(ch2);

}

string str1 = "";

while (stack.Count > 0)

str1 += Convert.ToChar(stack.Pop()).ToString();

string str2 = "";

string str3 = "";

int index1;

for (index1 = 0; index1 < str1.Length; ++index1)

{

ch1 = str1[index1];

switch (ch1)

{

case '|':

case '&':

goto label\_18;

default:

str2 += str1[index1].ToString();

continue;

}

}

label\_18:

string str4 = str2.Trim();

if (index1 == str1.Length)

return str4 == "1";

char ch3 = ch1;

for (int index2 = index1 + 1; index2 < str1.Length; ++index2)

{

char ch2 = str1[index2];

switch (ch2)

{

case '|':

case '&':

ch3 = ch2;

switch (ch2)

{

case '&':

str4 = !(str4 == "1") || !(str3 == "1") ? "0" : "1";

break;

case '|':

str4 = !(str4 == "0") || !(str3 == "0") ? "1" : "0";

break;

}

str3 = "";

break;

default:

str3 += ch2.ToString();

break;

}

}

string str5 = str3.Trim();

switch (ch3)

{

case '&':

return str4 == "1" && str5 == "1";

case '|':

return !(str4 == "0") || !(str5 == "0");

default:

return false;

}

}

private bool IsValidFact(string sFact)

{

bool flag = false;

foreach (Fact fact in facts)

{

if (fact.ToString() == sFact)

{

flag = true;

break;

}

}

return flag;

}

private Fact GetFactById(int factId)

{

foreach (Fact fact in facts)

{

if (fact.ID == factId)

return fact;

}

return null;

}

}

}