**Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования** **«Новгородский государственный университет имени Ярослава Мудрого»**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кафедра ИТиС

«Продукционная модель представления знаний»

Лабораторная работа №2

По дисциплине: Системы искусственного интеллекта

Отчет

Выполнил: студент гр. 9091

\_\_\_\_\_\_\_ \_\_\_\_Мирошкин О.А.

Проверил: Преподаватель

\_\_\_\_\_\_\_\_\_\_\_Михайлов Д. В.

Великий Новгород

2022

1. **Формулировка цели и задач**

Целью данной работы является приобретение студентами умений и навыков реализации пополняемой динамической базы знаний, не включаемой непосредственно в текст программы.

Необходимые для достижения поставленной цели задачи состоят в следующем:

· научиться оценивать возможности применения современных языков высокого уровня для реализации баз знаний экспертных систем (ЭС);

· изучение технических аспектов реализации продукционной модели представления знаний.

1. **Конфигурация системы продукций**

Предметная область для ЭС – Материал для пошива одежды.

*Таблица 1. Описание предметной области*

|  |  |  |
| --- | --- | --- |
| **Тип сырья** | **Название** | **Характеристики** |
| Натуральный | Хлопок | Цена за метр 1137  Качество 8  Уровень аллергии 2  Уровень поглощения влаги 3  Уровень воздухопроницаемости 3  Уровень терморегуляции 3 |
| Лён | Цена за метр 1669  Качество 6  Уровень аллергии 2  Уровень поглощения влаги 2  Уровень воздухопроницаемости 5  Уровень терморегуляции 3 |
| Шерсть | Цена за метр 1200  Качество 8  Уровень аллергии 0  Уровень поглощения влаги 4  Уровень воздухопроницаемости 3  Уровень терморегуляции 3 |
| Искусственный | Бамбук | Цена за метр 600  Качество 4  Уровень аллергии 3  Уровень поглощения влаги 2  Уровень воздухопроницаемости 5  Уровень терморегуляции 2 |
| Ацетат | Цена за метр 800  Качество 4  Уровень аллергии 4  Уровень поглощения влаги 1  Уровень воздухопроницаемости 2  Уровень терморегуляции 4 |
| Синтетический | Акрил | Цена за метр 400  Качество 2  Уровень аллергии 2  Уровень поглощения влаги 0  Уровень воздухопроницаемости 1  Уровень терморегуляции 2 |
| Джордан | Цена за метр 300  Качество 3  Уровень аллергии 3  Уровень поглощения влаги 0  Уровень воздухопроницаемости 1  Уровень терморегуляции 3 |

Для описания объекта или его отдельных свойств выбираются некоторые характеристики – величины, которые могут принимать либо количественные, либо качественные значения.

В свою очередь, совокупность всех характеристик некоторого объекта образует так называемый список разрешенных характеристик данного объекта. Списки разрешенных характеристик и разрешенных значений этих характеристик охватывают множество всех имеющихся фактов, подлежащих хранению в базе знаний экспертной системы.

В экспертных системах правила, по которым решаются проблемы в конкретной предметной области, хранятся в базе знаний. *Проблемная область ЭС* описывается посредством фактов и правил. Проблемы ставятся перед системой в виде совокупности фактов, описывающих некоторую ситуацию, и система с помощью базы знаний пытается вывести заключение из этих фактов.

Для данной лабораторной работы в качестве базы знаний был выбран текстовый файл.

Список атрибутов и принимаемых значений в []:

*Файл LocalBase.txt*

*Тип сырья [Натуральный, Искусственный, Синтетический]*

*Максимальная Цена за метр [значения больше 0]*

*Минимальное Качество [0, 1, 2, 3, 4, 5, 6, 7, 8]*

*Максимальный Уровень аллергии [0, 1, 2, 3, 4, 5, 6]*

*Минимальный Уровень поглощения влаги [0, 1, 2, 3, 4]*

*Минимальный Уровень воздухопроницаемости [0, 1, 2, 3, 4, 5, 6]*

*Минимальный Уровень терморегуляции [0, 1, 2, 3, 4, 5, 6]*

Также в базе знаний хранятся правила, синтаксис которых:

IF A THEN S, где А – условие; S – заключение.

Пример правил:

*Файл GlobalBase.txt*

*IF Тип сырья = Натуральный & Цена за метр >= 1137 & Качество <= 8 & Уровень аллергии >= 2 & Уровень поглощения влаги <= 3 & Уровень воздухопроницаемости <= 3 & Уровень терморегуляции <= 3 THEN Материал = Хлопок*

*IF Тип сырья = Натуральный & Цена за метр >= 1669 & Качество <= 7 & Уровень аллергии >= 2 & Уровень поглощения влаги <= 2 & Уровень воздухопроницаемости <= 5 & Уровень терморегуляции <= 5 THEN Материал = Лён*

Некоторые термы дают положительные результаты при нескольких значениях соответствующих атрибутов. Например, терм “*Цена за метр >= 1137*” будет давать положительный результат при значении атрибута “*Максимальная Цена за метр”* от значений 1137 и более.

1. **Описание способа организации поиска на графе**

Поиск на графе организован с помощью последовательного отброса заведомо ложных путей и проверки доступных с помощью перебора.

1. **Описание конфликтного набора и алгоритм разрешения конфликта при логическом выводе**

В данной лабораторной работе используется обратный порядок вывода. В системах с обратным выводом сначала выдвигается гипотеза, а затем механизм вывода в процессе работы переходит от выдвинутой гипотезы к фактам и пытается найти среди них те, которые подтверждают эту гипотезу.

Конфликтный набор:

Чтобы вызвать конфликтный набор, необходимо указать параметры, которые не будут соблюдать ни одному из термов, необходимых для определения успешного результата.

Пример:   
Натуральный  
100

Ответ системы: База знаний не содержит удовлетворяющий критериям результат.

1. **Тестовый набор правил**

Хлопок

Натуральный

2000

6

2

2

1

2

Ответ системы: Гипотеза подтвердилась! Материал хлопок вам подходит!

Бамбук

Синтетический

300

5

2

4

1

3

Ответ системы: Гипотеза не подтвердилась! Материал бамбук вам не подходит!

**Вывод**

Мною была изучена реализация пополняемой динамической базы знаний, не включаемой непосредственно в текст программы.

Так же я изучил реализацию продукционной модели представления знаний.

Так же была построена база знаний, предметной области - “Выбор материала для изготовления одежды”.

**Приложение:**

1. **Текст программы**

Program.cs

1. namespace Lab2
2. {
3. public class Program
4. {
5. private const string MaterialsFilePath = "Materials.json";
6. private const string LocalBasePath = "LocalBase.txt";
7. private const string GlobalBasePath = "GlobalBase.txt";
8. static void Main(string[] args)
9. {
10. var bootstrapper = new Bootstrapper(MaterialsFilePath, GlobalBasePath, LocalBasePath);
11. bootstrapper.Bootstrap();
12. new ProductionSystem(bootstrapper.Materials, Bootstrapper.Conditions,
13. Bootstrapper.OnConditionValueRecieved, bootstrapper.UserInterface)
14. .StartLooping();
15. }
16. }
17. }

ProductionSystem.cs  
using System;

using System.Collections.Generic;

using System.Linq;

using Lab2.Data;

using Lab2.UI;

using static Lab2.Predicates.MaterialConditions;

namespace Lab2

{

public class ProductionSystem

{

private readonly IEnumerable<Material> \_materials;

private readonly List<Material> \_suitableMaterials;

private readonly IConditionsUserInterface \_userInterface;

private readonly List<string> \_userConditions = new List<string>();

private readonly MaterialStats \_userMaterialStats;

private readonly IEnumerable<MaterialCondition> \_materialConditions;

private readonly IReadOnlyDictionary<MaterialCondition, Action<string, MaterialStats>> \_onConditionValueRecieved;

public ProductionSystem(IEnumerable<Material> materials,

IEnumerable<MaterialCondition> conditions,

IReadOnlyDictionary<MaterialCondition, Action<string, MaterialStats>> OnConditionValueRecieved,

IConditionsUserInterface conditionsInterface)

{

\_materials = materials;

\_suitableMaterials = new List<Material>(\_materials);

\_userMaterialStats = new MaterialStats();

\_materialConditions = conditions;

\_onConditionValueRecieved = OnConditionValueRecieved;

\_userInterface = conditionsInterface;

}

public void StartLooping()

{

while (true)

{

DoLoop();

}

}

private void DoLoop()

{

string requestedMaterialName = \_userInterface.RequestHypothesis();

bool conditionsChecked = \_userConditions.Count == \_materialConditions.Count();

if (!conditionsChecked)

{

RequestConditions();

}

CheckHypothesis(requestedMaterialName);

}

private void RequestConditions()

{

foreach (MaterialCondition condition in \_materialConditions)

{

string conditionValueStr = \_userInterface.RequestConditionValue(condition);

\_onConditionValueRecieved[condition](conditionValueStr, \_userMaterialStats);

ApplyCondition(condition);

\_userInterface.PrintNewLine();

\_userInterface.PrintMemoryPool(\_userConditions);

\_userInterface.PrintNewLine();

}

}

private void CheckHypothesis(string requestedMaterialName)

{

bool confirmed = \_suitableMaterials.Select(mat => mat.MaterialName.ToLower())

.Contains(requestedMaterialName);

\_userInterface.PrintHypothesisConclusion(confirmed, requestedMaterialName);

\_userInterface.PrintNewLine();

}

private void ApplyCondition(MaterialCondition toApply)

{

\_userConditions.Add(StringProvider.GetParametrizedConditionString(toApply, \_userMaterialStats));

\_suitableMaterials.RemoveAll(mat => toApply(mat.Stats, \_userMaterialStats) == false);

}

}

}

Bootstrapper.cs  
  
using Lab2.Data;

using Lab2.UI;

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using static Lab2.Predicates.MaterialConditions;

namespace Lab2

{

public class Bootstrapper

{

private readonly string \_materialsFilePath;

private readonly string \_globalBaseFilePath;

private readonly string \_localBaseFilePath;

private MaterialSerializer \_materialSerializer;

public IEnumerable<Material> Materials { get; private set; }

public UserInterface UserInterface { get; private set; }

public static readonly IReadOnlyDictionary<MaterialCondition, Action<string, MaterialStats>> OnConditionValueRecieved;

public static readonly IEnumerable<MaterialCondition> Conditions;

static Bootstrapper()

{

OnConditionValueRecieved = new Dictionary<MaterialCondition, Action<string, MaterialStats>>

{

{ IsRawMaterialTypeEquals, (str, stats) => stats.RawMaterialType = StringProvider.StrToMaterialType(str) },

{ IsPriceNotHigher, (str, stats) => stats.PricePerMeter = ulong.Parse(str) },

{ IsQualityNotLower, (str, stats) => stats.Quality = uint.Parse(str) },

{ IsAllergicNotHigher, (str, stats) => stats.AllergicLevel = uint.Parse(str) },

{ IsAbsorptionNotLower, (str, stats) => stats.MoistureAbsorption = uint.Parse(str) },

{ IsBreathabilityNotLower, (str, stats) => stats.Breatability = uint.Parse(str) },

{ IsThermoregulationNotLower, (str, stats) => stats.Thermoregulation = uint.Parse(str) }

};

Conditions = OnConditionValueRecieved.Keys;

}

public Bootstrapper(string materialsFilePath, string globalBaseFilePath, string localBaseFilePath)

{

\_materialsFilePath = materialsFilePath;

\_globalBaseFilePath = globalBaseFilePath;

\_localBaseFilePath = localBaseFilePath;

}

public void Bootstrap()

{

\_materialSerializer = new MaterialSerializer(\_materialsFilePath);

bool firstCreation = !File.Exists(\_materialsFilePath) || !File.Exists(\_globalBaseFilePath) ||

!File.Exists(\_localBaseFilePath);

if (firstCreation)

{

SaveMaterials(CreateStandartMaterials());

}

Materials = \_materialSerializer.ReadAll();

var materialNames = Materials.Select(mat => mat.MaterialName);

UserInterface = new UserInterface(materialNames);

if (firstCreation)

{

SaveBases();

}

}

private void SaveBases()

{

var condtitionsSerializer = new ConditionsSerialzer(\_localBaseFilePath,

\_globalBaseFilePath, UserInterface);

condtitionsSerializer.SaveLocalBase(Conditions);

condtitionsSerializer.SaveGlobalBase(Conditions, Materials);

}

private void SaveMaterials(IEnumerable<Material> materials)

{

foreach (Material material in materials)

{

\_materialSerializer.Add(material, forceSave: false);

}

\_materialSerializer.SaveAll();

}

private static Material[] CreateStandartMaterials()

{

return new[]

{

new Material("Хлопок", new MaterialStats(RawMaterialType.Natural,

pricePerMeter: 1137, quality: 8, allergicLevel: 2, moistureAbsorption: 3, breatability: 3, thermoregulation: 3)),

new Material("Лён", new MaterialStats(RawMaterialType.Natural,

pricePerMeter: 1669, quality: 7, allergicLevel: 2, moistureAbsorption: 2, breatability: 5, thermoregulation: 5)),

new Material("Шерсть", new MaterialStats(RawMaterialType.Natural,

pricePerMeter: 1200, quality: 8, allergicLevel: 0, moistureAbsorption: 4, breatability: 3, thermoregulation: 3)),

new Material("Бамбук", new MaterialStats(RawMaterialType.Artificial,

pricePerMeter: 600, quality: 4, allergicLevel: 3, moistureAbsorption: 2, breatability: 5, thermoregulation: 2)),

new Material("Ацетат", new MaterialStats(RawMaterialType.Artificial,

pricePerMeter: 800, quality: 4, allergicLevel: 4, moistureAbsorption: 1, breatability: 2, thermoregulation: 4)),

new Material("Акрил", new MaterialStats(RawMaterialType.Synthetic,

pricePerMeter: 400, quality: 2, allergicLevel: 2, moistureAbsorption: 0, breatability: 1, thermoregulation: 2)),

new Material("Джордан", new MaterialStats(RawMaterialType.Synthetic,

pricePerMeter: 300, quality: 3, allergicLevel: 3, moistureAbsorption: 0, breatability: 1, thermoregulation: 3))

};

}

}

}

ConditionsSerializer.cs  
  
using Lab2.UI;

using System;

using System.Collections.Generic;

using System.IO;

using System.Text;

using static Lab2.Predicates.MaterialConditions;

namespace Lab2.Data

{

public class ConditionsSerialzer

{

private readonly string \_localBasePath;

private readonly string \_globalBasePath;

private readonly IGlobalConditionStringProvider \_globalConditionProvider;

private readonly FileMode \_openMode;

public ConditionsSerialzer(string localBaseFilePath, string globalBaseFilePath,

IGlobalConditionStringProvider globalConditionProvider, FileMode openMode = FileMode.OpenOrCreate)

{

\_localBasePath = localBaseFilePath;

\_globalBasePath = globalBaseFilePath;

\_globalConditionProvider = globalConditionProvider;

\_openMode = openMode;

}

public void SaveLocalBase(IEnumerable<MaterialCondition> conditions)

{

using StreamWriter writer = new StreamWriter(new FileStream(\_localBasePath, \_openMode,

FileAccess.Write));

writer.BaseStream.SetLength(0);

foreach (MaterialCondition condition in conditions)

{

writer.WriteLine(StringProvider.GetNonParametrizedConditionString(condition));

}

}

public void SaveGlobalBase(IEnumerable<MaterialCondition> conditions, IEnumerable<Material> materials)

{

using StreamWriter writer = new StreamWriter(new FileStream(\_globalBasePath, \_openMode,

FileAccess.Write));

writer.BaseStream.SetLength(0);

foreach (Material material in materials)

{

writer.WriteLine(\_globalConditionProvider.GetMaterialConditions(conditions, material));

}

}

}

}

Material.cs  
  
using System;

namespace Lab2.Data

{

[Serializable]

public struct Material

{

public string MaterialName;

public MaterialStats Stats;

public Material(string materialName, MaterialStats stats)

{

MaterialName = materialName;

Stats = stats;

}

}

}

MaterialSerializer.cs  
  
using Newtonsoft.Json;

using System.Collections.Generic;

using System.IO;

using System.Linq;

namespace Lab2.Data

{

public class MaterialSerializer

{

private readonly string \_filePath;

private readonly FileMode \_openMode;

private readonly List<Material> \_materials;

public MaterialSerializer(string filePath, FileMode openMode = FileMode.OpenOrCreate)

{

\_filePath = filePath;

\_openMode = openMode;

\_materials = new List<Material>();

}

public void Add(Material material, bool forceSave)

{

\_materials.Add(material);

if (forceSave)

{

SaveAll();

}

}

public IReadOnlyList<Material> ReadAll(bool refreshOld = true)

{

if (refreshOld)

{

\_materials.Clear();

}

using (StreamReader reader = new StreamReader(new FileStream(\_filePath, \_openMode, FileAccess.Read)))

{

string materialsStr = reader.ReadLine();

var uniqueMaterials = JsonConvert.DeserializeObject<Material[]>(materialsStr);

\_materials.AddRange(uniqueMaterials);

}

return \_materials;

}

public void SaveAll()

{

using StreamWriter writer = new StreamWriter(new FileStream(\_filePath, \_openMode, FileAccess.Write));

writer.BaseStream.SetLength(0);

string materialsStr = JsonConvert.SerializeObject(\_materials.ToArray());

writer.WriteLine(materialsStr);

}

}

}

MaterialStats.cs  
  
using System;

using System.Text.Json.Serialization;

namespace Lab2.Data

{

public enum RawMaterialType

{

Natural, Artificial, Synthetic

}

[Serializable]

public class MaterialStats

{

public RawMaterialType RawMaterialType;

public ulong PricePerMeter;

public uint Quality;

public uint AllergicLevel;

public uint MoistureAbsorption;

public uint Breatability;

public uint Thermoregulation;

public readonly static Range QualityRange = new Range(0, 9);

public readonly static Range AllergicRange = new Range(0, 7);

public readonly static Range AbsorptionRange = new Range(0, 5);

public readonly static Range BreatabilityRange = new Range(0, 7);

public readonly static Range ThermoregulationRange = new Range(0, 7);

public MaterialStats()

{ }

public MaterialStats(RawMaterialType rawMaterialType,

ulong pricePerMeter,

uint quality,

uint allergicLevel,

uint moistureAbsorption,

uint breatability,

uint thermoregulation)

{

RawMaterialType = rawMaterialType;

PricePerMeter = pricePerMeter;

Quality = ClampStat(quality, QualityRange);

AllergicLevel = ClampStat(allergicLevel, AllergicRange);

MoistureAbsorption = ClampStat(moistureAbsorption, AbsorptionRange);

Breatability = ClampStat(breatability, BreatabilityRange);

Thermoregulation = ClampStat(thermoregulation, ThermoregulationRange);

}

private uint ClampStat(uint stat, Range range) =>

Math.Clamp(stat, (uint)range.Start.Value, (uint)range.End.Value);

}

}

MaterialConditions.cs  
  
using Lab2.Data;

namespace Lab2.Predicates

{

public static class MaterialConditions

{

public delegate bool MaterialCondition(MaterialStats materialStats,

MaterialStats userStats);

public static bool IsRawMaterialTypeEquals(MaterialStats materialStats,

MaterialStats userStats) => materialStats.RawMaterialType == userStats.RawMaterialType;

public static bool IsPriceNotHigher(MaterialStats materialStats,

MaterialStats userStats) => materialStats.PricePerMeter <= userStats.PricePerMeter;

public static bool IsQualityNotLower(MaterialStats materialStats,

MaterialStats userStats) => materialStats.Quality >= userStats.Quality;

public static bool IsAllergicNotHigher(MaterialStats materialStats,

MaterialStats userStats) => materialStats.AllergicLevel <= userStats.AllergicLevel;

public static bool IsAbsorptionNotLower(MaterialStats materialStats,

MaterialStats userStats) => materialStats.MoistureAbsorption >= userStats.MoistureAbsorption;

public static bool IsBreathabilityNotLower(MaterialStats materialStats,

MaterialStats userStats) => materialStats.Breatability >= userStats.Breatability;

public static bool IsThermoregulationNotLower(MaterialStats materialStats,

MaterialStats userStats) => materialStats.Thermoregulation >= userStats.Thermoregulation;

}

}

ICondtitionsUserInterface.cs  
  
using System.Collections.Generic;

using static Lab2.Predicates.MaterialConditions;

namespace Lab2.UI

{

public interface IConditionsUserInterface

{

string RequestHypothesis();

string RequestConditionValue(MaterialCondition condition);

void PrintNewLine();

void PrintMemoryPool(IEnumerable<string> memoryPool);

void PrintHypothesisConclusion(bool confirmed, string materialName);

}

}

}  
  
IGlobalConditionStringProvider.cs  
  
using Lab2.Data;

using System.Collections.Generic;

using static Lab2.Predicates.MaterialConditions;

namespace Lab2.UI

{

public interface IGlobalConditionStringProvider

{

string GetMaterialConditions(IEnumerable<MaterialCondition> conditions, Material material);

}

}

StringProvider.cs  
  
using Lab2.Data;

using System;

using System.Collections.Generic;

using System.Linq;

using static Lab2.Predicates.MaterialConditions;

namespace Lab2.UI

{

public static class StringProvider

{

public const string Material = "Материал";

public const string MaterialTypeName = "Тип сырья";

public const string PriceName = "Цена за метр";

public const string QualityName = "Качество";

public const string AllergicName = "Уровень аллергии";

public const string AbsorbtionName = "Уровень поглощения влаги";

public const string BreathabilityName = "Уровень воздухопроницаемости";

public const string ThermoregulationName = "Уровень терморегуляции";

private static readonly Dictionary<MaterialCondition, Func<MaterialStats, string>> \_parametrizedConditionStrings;

private static readonly Dictionary<MaterialCondition, string> \_nonParametrizedConditionStrings;

static StringProvider()

{

\_parametrizedConditionStrings = new Dictionary<MaterialCondition, Func<MaterialStats, string>>

{

{ IsRawMaterialTypeEquals, IsRawMaterialTypeEqualsToString },

{ IsPriceNotHigher, IsPriceNotHigherToString },

{ IsQualityNotLower, IsQualityNotLowerToString },

{ IsAllergicNotHigher, IsAllergicNotHigherToString },

{ IsAbsorptionNotLower, IsAbsorptionNotLowerToString },

{ IsBreathabilityNotLower, IsBreathabilityNotLowerToString },

{ IsThermoregulationNotLower, IsThermoregulationNotLowerToString }

};

\_nonParametrizedConditionStrings = new Dictionary<MaterialCondition, string>

{

{ IsRawMaterialTypeEquals, $"{MaterialTypeName} " +

$"[{MaterialTypeValues.ToSingleStr()}]"},

{ IsPriceNotHigher, $"Максимальная {PriceName} " +

$"[значения больше 0]"},

{ IsQualityNotLower, $"Минимальное {QualityName} " +

$"[{RangeToString(MaterialStats.QualityRange)}]"},

{ IsAllergicNotHigher, $"Максимальный {AllergicName} " +

$"[{RangeToString(MaterialStats.AllergicRange)}]"},

{ IsAbsorptionNotLower, $"Минимальный {AbsorbtionName} " +

$"[{RangeToString(MaterialStats.AbsorptionRange)}]"},

{ IsBreathabilityNotLower, $"Минимальный {BreathabilityName} " +

$"[{RangeToString(MaterialStats.BreatabilityRange)}]"},

{ IsThermoregulationNotLower, $"Минимальный {ThermoregulationName} " +

$"[{RangeToString(MaterialStats.ThermoregulationRange)}]"}

};

}

public static string[] MaterialTypeValues => new[] { "Натуральный", "Искусственный", "Синтетический" };

public static string RangeToString(Range range)

{

string[] values = new string[range.End.Value - range.Start.Value];

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

{

values[i] = $"{range.Start.Value + i}";

}

return values.ToSingleStr();

}

public static string GetParametrizedConditionString(MaterialCondition condition, MaterialStats userStats) =>

\_parametrizedConditionStrings[condition](userStats);

public static string GetNonParametrizedConditionString(MaterialCondition condition) =>

\_nonParametrizedConditionStrings[condition];

public static string IsRawMaterialTypeEqualsToString(MaterialStats userStats) =>

$"{MaterialTypeName} = {RawMaterialTypeToStr(userStats.RawMaterialType)}";

public static string IsPriceNotHigherToString(MaterialStats userStats) =>

$"Максимальная {PriceName} = {userStats.PricePerMeter}";

public static string IsQualityNotLowerToString(MaterialStats userStats) =>

$"Минимальное {QualityName} = {userStats.Quality}";

public static string IsAllergicNotHigherToString(MaterialStats userStats) =>

$"Максимальный {AllergicName} = {userStats.AllergicLevel}";

public static string IsAbsorptionNotLowerToString(MaterialStats userStats) =>

$"Минимальный {AbsorbtionName} = {userStats.MoistureAbsorption}";

public static string IsBreathabilityNotLowerToString(MaterialStats userStats) =>

$"Минимальный {BreathabilityName} = {userStats.Breatability}";

public static string IsThermoregulationNotLowerToString(MaterialStats userStats) =>

$"Минимальный {ThermoregulationName} = {userStats.Thermoregulation}";

public static IEnumerable<string> GetMaterialsNames(IEnumerable<Material> materials) =>

materials.Select(mat => mat.MaterialName);

public static RawMaterialType StrToMaterialType(string materialType)

{

materialType = materialType.ToLower();

return materialType switch

{

"натуральный" => RawMaterialType.Natural,

"искусственный" => RawMaterialType.Artificial,

"синтетический" => RawMaterialType.Synthetic,

\_ => throw new ArgumentException(),

};

}

public static string RawMaterialTypeToStr(RawMaterialType materialType)

{

return materialType switch

{

RawMaterialType.Natural => "натуральный",

RawMaterialType.Artificial => "искусственный",

RawMaterialType.Synthetic => "синтетический",

\_ => throw new NotImplementedException(),

};

}

}

}

UserInterface.cs  
  
using Lab2.Data;

using System;

using System.Collections.Generic;

using System.Linq;

using static Lab2.Predicates.MaterialConditions;

namespace Lab2.UI

{

public class UserInterface : IGlobalConditionStringProvider, IConditionsUserInterface

{

private readonly IEnumerable<string> \_materialNames;

public UserInterface(IEnumerable<string> materialNames)

{

\_materialNames = materialNames;

}

/// <returns>String representing material name</returns>

public string RequestHypothesis()

{

Console.WriteLine($"Пожалуйста, выдвините гепотезу, о подходящем материале, " +

$"например \"{\_materialNames.First()}\"");

string materialsString = \_materialNames.ToSingleStr();

Console.WriteLine($"Список материалов: {materialsString}");

string materialName = Console.ReadLine().ToLower();

bool materialNotExists = !\_materialNames.Select(name => name.ToLower()).Contains(materialName);

if (materialNotExists)

{

Console.WriteLine($"Материала {materialName} не существует! Пожалуйста, попробуйте ещё раз!");

return RequestHypothesis();

}

return materialName;

}

public string RequestConditionValue(MaterialCondition condition)

{

string reqConditionMessage = StringProvider.GetNonParametrizedConditionString(condition);

Console.WriteLine($"Введите значение атрибута {reqConditionMessage}: ");

string conditionValue = Console.ReadLine().ToLower();

if (!ValidateConditionValue(condition, conditionValue))

{

Console.WriteLine($"Неправильный ввод! Попробуйте ещё раз.");

return RequestConditionValue(condition);

}

return conditionValue;

}

public void PrintNewLine() => Console.Write(Environment.NewLine);

public void PrintMemoryPool(IEnumerable<string> memoryPool) =>

Console.WriteLine($"Основная память: {Environment.NewLine}" +

$"{string.Join($",{Environment.NewLine}", memoryPool)}");

public void PrintHypothesisConclusion(bool confirmed, string materialName)

{

if (confirmed)

{

Console.WriteLine($"Гепотеза подтвердилась! Материал {materialName} вам подходит!");

}

else

{

Console.WriteLine($"Гепотеза не подтвердилась! Материал {materialName} вам не подходит!");

}

}

public string GetMaterialConditions(IEnumerable<MaterialCondition> conditions, Material material)

{

const string andSepatator = " & ";

string conditionStr = "IF ";

foreach (MaterialCondition condition in conditions)

{

conditionStr += StringProvider.GetParametrizedConditionString(condition, material.Stats)

+ andSepatator;

}

conditionStr = conditionStr.Substring(0, conditionStr.LastIndexOf(andSepatator));

conditionStr += $" THEN {StringProvider.Material} = {material.MaterialName}";

return conditionStr;

}

private bool ValidateConditionValue(MaterialCondition condition, string conditionValue)

{

if (condition == IsRawMaterialTypeEquals)

{

try

{

StringProvider.StrToMaterialType(conditionValue);

}

catch (Exception)

{

return false;

}

}

else

{

bool inputIsRight = uint.TryParse(conditionValue, out uint printedNumber);

if (condition != IsPriceNotHigher)

inputIsRight |= IsInRange(printedNumber, new Range(0, 8)); // got lazy here

if (!inputIsRight)

{

return false;

}

}

return true;

}

private bool IsInRange(uint number, Range range) =>

number >= range.Start.Value && number <= range.End.Value;

}

}

LocalBase.txt  
  
Тип сырья [Натуральный, Искусственный, Синтетический]

Максимальная Цена за метр [значения больше 0]

Минимальное Качество [0, 1, 2, 3, 4, 5, 6, 7, 8]

Максимальный Уровень аллергии [0, 1, 2, 3, 4, 5, 6]

Минимальный Уровень поглощения влаги [0, 1, 2, 3, 4]

Минимальный Уровень воздухопроницаемости [0, 1, 2, 3, 4, 5, 6]

Минимальный Уровень терморегуляции [0, 1, 2, 3, 4, 5, 6]

GlobalBase.txt  
  
IF Тип сырья = Натуральный & Цена за метр >= 1137 & Качество <= 8 & Уровень аллергии >= 2 & Уровень поглощения влаги <= 3 & Уровень воздухопроницаемости <= 3 & Уровень терморегуляции <= 3 THEN Материал = Хлопок

IF Тип сырья = Натуральный & Цена за метр >= 1669 & Качество <= 7 & Уровень аллергии >= 2 & Уровень поглощения влаги <= 2 & Уровень воздухопроницаемости <= 5 & Уровень терморегуляции <= 5 THEN Материал = Лён

IF Тип сырья = Натуральный & Цена за метр >= 1200 & Качество <= 8 & Уровень аллергии >= 0 & Уровень поглощения влаги <= 4 & Уровень воздухопроницаемости <= 3 & Уровень терморегуляции <= 3 THEN Материал = Шерсть

IF Тип сырья = Искусственный & Цена за метр >= 600 & Качество <= 4 & Уровень аллергии >= 3 & Уровень поглощения влаги <= 2 & Уровень воздухопроницаемости <= 5 & Уровень терморегуляции <= 2 THEN Материал = Бамбук

IF Тип сырья = Искусственный & Цена за метр >= 800 & Качество <= 4 & Уровень аллергии >= 4 & Уровень поглощения влаги <= 1 & Уровень воздухопроницаемости <= 2 & Уровень терморегуляции <= 4 THEN Материал = Ацетат

IF Тип сырья = Синтетический & Цена за метр >= 400 & Качество <= 2 & Уровень аллергии >= 2 & Уровень поглощения влаги <= 0 & Уровень воздухопроницаемости <= 1 & Уровень терморегуляции <= 2 THEN Материал = Акрил

IF Тип сырья = Синтетический & Цена за метр >= 300 & Качество <= 3 & Уровень аллергии >= 3 & Уровень поглощения влаги <= 0 & Уровень воздухопроницаемости <= 1 & Уровень терморегуляции <= 3 THEN Материал = Джордан