в геномику позволит усовершенствовать методы оказания онкологической помощи за счет оперативного проведения анализа ДНК и получения персонализированных рекомендаций по лечению для каждого пациента. Изучая особенности организма и генотип определенного человека, врачи могут назначать в наибольшей степени эффективные медикаменты и процедуры. Работники медицинской отрасли считают, что существующие сегодня ограничения для использования всей доступной информации снижают степень их уверенности в правильности принятых решений.

В кулинарии такого рода системы могут предложить нечто совсем неожиданное, открыть новую область, добавить необычные сочетания продуктов. Уже на данный момент некоторые когнитивные системы могут составлять рецепты блюд, исходя из предварительно заданного набора продуктов. Также когнитивные системы могут использоваться для оценки свежести еды, анализируя такие факторы, как внешний вид и запах.

Когнитивные системы используются в метеорологии. Прогнозы солнца и ветра, сделанные при помощи обучающихся машин и других технологий когнитивных вычислений, оказываются на 30 процентов точнее, чем прогнозы, получаемые при использовании традиционных подходов. Улучшая точность прогнозов, энергетические компании могут работать эффективнее, увеличивая свои доходы. В конечном счете, это приведет к росту использования возобновляемой энергии.

В сфере образования когнитивные технологии используются для персонализации учебной методики и улучшения опыта научной работы для студентов и преподавателей.

Над созданием когнитивных систем работают многие организации и правительства разных стран. Но на текущий момент наиболее совершенной когнитивной системой, включающей огромное число подсистем и элементов, является IBM Watson.

Я.А. Юницкий (УО «ГГУ им. Ф. Скорины», Гомель)
Науч. рук. **А.В. Воруев**, канд. техн. наук, доцент

ОПРЕДЕЛЕНИЕ И РЕАЛИЗАЦИЯ СТАТИЧЕСКИХ МЕТОДОВ ИНТЕРФЕЙСА В .NET

Как известно, статические методы – это методы, принадлежащие собственно типу, а не конкретному экземпляру данного типа. Использование таких методов оправдано, если реализация метода предполагает выполнение некоторых действий, независящих от полей экземпляра класса. Другими словами, это некоторый общий метод.

В языках программирования, совместимых с платформой .NET, запрещено определять сигнатуру статических методов в интерфейсе (так как получение доступа к статическому объекту осуществляется по имени класса, а интерфейсы реализуются посредством таблицы виртуальных методов, т.е. как переопределяемые методы). Тем не менее, в платформе .NET такая возможность есть. Более того, в интерфейс можно поместить реализацию такого статического метода. Это связано с возможностью дальнейшего расширения .NET совместимых языков.

Для создания статического метода в интерфейсе необходимо использовать промежуточный язык программирования CIL (Common Intermediate Language, ранее MSIL), представляющий собой высокоуровневый язык ассемблера, в который компилируются все программы, предназначенные для платформы .NET.

Далее определим простейшую реализацию статического метода в интерфейсе. Первым шагом создания любого CIL-проекта является перечисление внешних сборок, используемых в текущей. В рассматриваемом примере применяются только типы из сборки mscorlib.dll, а значит потребуется добавить ссылку только на эту сборку. Для этого необходимо указать в новом файле директиву assembly с уточняющим атрибутом extern.

```
.assembly extern mscorlib {}
```

Следующий шаг состоит в определении интересующей сборки с использованием директивы assembly. Директива module определяет название и тип сборки. В нашем случае это DLL библиотека.

```
.assembly TestStatic {}
.module TestStatic.dll
```

Далее, используя директиву namespace, определяется пространство имён сборки. В блок пространства имён помещается определение типа данных, в нашем случае интерфейса, с применением директивы class.

В дополнение к директиве class применяются различные уточняющие атрибуты, определяющие область видимости, или уточняющие природу типа данных. Атрибут extends объявляет базовый класс, и если он не указан явно, автоматически будет использован тип System.Object в качестве родительского.

```
.namespace
{
 .class interface public TestStatic.ITestInterface {}
}
```

На данном этапе непосредственно определяются члены типов данных. В нашем случае это статический метод. Для примера определим его без входных параметров, с возвращаемым типом void и с именем StaticMethod. Полностью простейшая реализация поставленной задачи на языке CIL может выглядеть следующим образом:

```
.assembly extern mscorlib {}
.assembly TestStatic {}
.module TestStatic.dll
.namespace TestLibrary
{
 .class interface public TestStatic.ITestInterface
 {
 .method public hidebysig static void StaticMethod() cil man-
aged
 {
 .maxstack  8
 nop
 ldstr "Hello from Test Interface"
 call void [mscorlib]System.Console::WriteLine(string)
 nop
 ret
 }}
}
```

Для компилирования данной программы применяется компилятор ilasm.exe из пакета .NET Framework, например, в командной строке:

```
iliasm /dll C:\TestStatic.il /output=C:\TestStatic.dll
```

Для проверки работоспособности созданной библиотеки необходимо написать простое приложение, использующее её. К сожалению, поскольку в языке C# заблокирована возможность использовать статический метод в интерфейсе, непосредственный их вызов по имени интерфейса или класса, реализующего интерфейс, не представляется возможным, т. к. это приведёт к ошибке на стадии компиляции. Альтернативным решением является использование механизмов рефлексии для получения доступа к методу.

```
using System;
using System.Reflection;
using TestLibrary.TestStatic;
namespace Program
{
 class TestClass : ITestInterface {}
 class Program
 {
 static void Main()
 {
 //TestClass.StaticMethod() //Ошибка компиляции
 //ITestInterface.StaticMethod() //Ошибка компиляции
 Type type = typeof(ITestInterface);
 MethodInfo method = type.GetMethod ("StaticMethod");
 method.Invoke(null, null);
 Console.ReadLine();
 }
 }
}
```

Скомпилировать приложение можно входящим в состав платформы .NET Framework компилятором csc.exe, с указанием ранее созданной библиотеки в качестве параметра.

```
csc /r:C:\TestStatic.dll /out:C:\program.exe C:\program.cs
```

Приложение выводит на консоль значение строковой переменной, определённой в статическом методе интерфейса (рисунок 1):

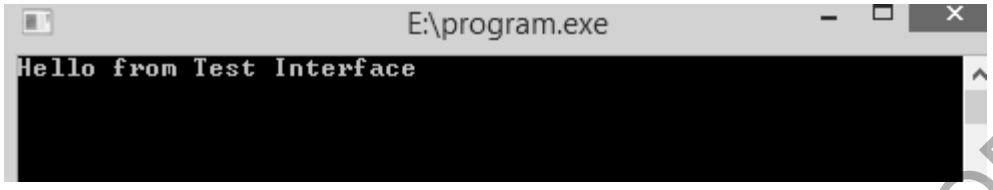

Рисунок 1 – Результат работы приложения

В.О. Ярошенко (Национальная Металлургическая Академия Украины, Днепропетровск)
Науч. рук. **А.В. Жаданос**, канд. техн. наук, доцент

ПРОГНОЗИРОВАНИЕ ХИМИЧЕСКОГО СОСТАВА КОНСТРУКЦИОННОЙ СТАЛИ В АГРЕГАТЕ КОВШ-ПЕЧЬ ДЛЯ СОЗДАНИЯ АВТОМАТИЗИРОВАННОЙ ИНФОРМАЦИОННОЙ СИСТЕМЫ

Введение. Постоянное увеличение требований к качеству выплавляемых сталей обуславливает широкое внедрение внепечной обработки. Одним из основных агрегатов внепечной обработки является установка ковш-печь (УКП), которая предназначена для десульфурации, легирования, раскисления металла и подогрева его перед последующими технологическими операциями. При этом необходимо обеспечить стабильный регламентированный химический состав металла при рациональном расходе легирующих материалов. Так как химический состав металла в процессе внепечной обработки стали контролируется путем периодических замеров, целесообразно прогнозировать его при помощи математических моделей.

Целью данной работы была разработка математической модели прогнозирования содержания легирующих элементов в конструкционных сталях при обработке расплава на УКП.

Разработка математической модели. Для раскисления и легирования конструкционной стали для железнодорожных колес (ГОСТ 10791-2011) применяются следующие материалы: ферросилиций ФС65, ферросиликомарганец МнС17 и углерод. С целью построения математических моделей