Лабораторная работа №1

Тема: Изучение основ работы с общеязыковой средой выполнения – Common Language Runtime платформы NET

*Цель работы* – получить основные навыки работы со средой CLR

*Теоретические положения.*

Платформа .NET Framework является надстройкой над операционной системой, в качестве которой может выступать любая версия Windows. Самым важным компонентом платформы .NET Framework является CLR (Common Language Runtime), предоставляющая среду, в которой выполняются программы. Главная ее роль заключается в том, чтобы обнаруживать и загружать типы .NET и производить управление ими в соответствии с полученными командами. CLR включает в себя виртуальную машину. На верхнем уровне среда активизирует объекты, производит проверку безопасности, размещает объекты в памяти, выполняет их, а также запускает сборщик мусора. Под сборкой мусора понимается освобождение памяти, занятой объектами, которые стали бесполезными и не используются в дальнейшей работе приложения. В CLR задача сборки мусора (и другие вопросы, связанные с использованием памяти) решается в нужное время и в нужном месте исполнительной средой, ответственной за выполнение вычислений.

Таким образом, компиляция .NET делится на два этапа с целью предоставления разработчикам удобных условий и мобильности. Перед созданием низкоуровневого машинного кода компилятору необходимо знать, в какой операционной системе и на каком базовом оборудовании будет функционировать приложение.

**Этап 1.**

При компиляции в управляемый код компилятор преобразует исходный код в промежуточный язык Microsoft (MSIL), представляющий собой независимый от процессора набор инструкций, который можно эффективно преобразовать в машинный код. Язык MSIL включает инструкции для загрузки, сохранения, инициализации и вызова методов для объектов, а также инструкции для арифметических и логических операций, потоков управления, прямого доступа к памяти, обработки исключений и других операций. Перед выполнением код MSIL необходимо преобразовать в код для конкретного процессора, обычно с помощью JIT-компилятора Поскольку среда CLR предоставляет для каждой поддерживаемой компьютерной архитектуры один или несколько JIT-компиляторов, один набор инструкций MSIL можно компилировать и выполнять в любой поддерживаемой архитектуре.

Когда компилятор создает код MSIL, одновременно создаются метаданные. Метаданные содержат описание типов в коде, включая определение каждого типа, подписи каждого члена типа, члены, на которые есть ссылки в коде, а также другие сведения, используемые средой выполнения во время выполнения. Код MSIL и метаданные содержатся в переносимом исполняемом (PE) файле, который представляет собой расширение форматов Microsoft PE и COFF, ранее использовавшихся для исполняемого содержимого. Этот формат файлов, позволяющий размещать код MSIL или машинный код, а также метаданные, позволяет операционной системе распознавать образы среды CLR. Наличие в файле метаданных наряду с кодом MSIL позволяет размещать описание внутри самого кода. Это устраняет необходимость в использовании библиотек типов или языка определения интерфейса (IDL). Среда выполнения находит и извлекает метаданные из файла по мере необходимости при выполнении.

**Этап 2.**

Перед запуском промежуточного языка MSIL его необходимо скомпилировать в машинный код для среды CLR для архитектуры конечного компьютера. Платформа .NET Framework предоставляет два способа такого преобразования:

– JIT-компилятор платформы .NET Framework;

–.NET Framework Генератор образов в машинном коде (Ngen.exe).

**Компиляция с помощью JIT-компилятора.**

При JIT-компиляции язык MSIL преобразуется в машинный код во время выполнения приложения по требованию, когда загружается и исполняется содержимое сборки. При JIT-компиляции принимается во внимание, что некоторый код может не вызываться во время выполнения. Вместо того, чтобы расходовать время и память на преобразование всего кода MSIL из переносимого исполняемого (PE) файла в машинный код, код MSIL преобразуется в машинный код по мере необходимости при выполнении и полученный машинный код сохраняется в памяти для доступа при дальнейших вызовах в контексте процесса.

**Создание кода во время установки с помощью NGen.exe.**

Поскольку JIT-компилятор преобразует код MSIL сборки в машинный код при вызове отдельных методов, определенных в этой сборке, то это обязательно приводит к снижению производительности во время выполнения.

Работа Ngen.exe отличается от JIT-компилятора в трех аспектах.

1. Она выполняет преобразование из языка MSIL в машинный код перед выполнением приложения, а не во время.

2. Она компилирует всю сборку, а не отдельные методы.

3. Она сохраняет созданный код в кэше образа машинного кода в виде файла на диске.

*Ход работы:*

1. Создаем консольное приложение со следующим кодом:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace lek001

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Test1");

Console.WriteLine("Test2");

Console.ReadLine();

}

}

}

Выполняем его построение – преобразование в IL код.

2. Запускаем командную строку Visual Studio

Пуск – Microsoft Visual Studio 2012 – Visual Studio Tools – Командная строка VS2012 x86 Native Tools

3. Переходим с помощью команды (CD) в папку с приложением. Для справки по командам строки необходимо набрать в строке после знака « >» help и нажать enter. Получаем следующее:

D:\lek\lk01\lk01\bin\Debug>help

Для получения сведений об определенной команде наберите HELP <имя команды>

…

CD Вывод имени либо смена текущей папки.

CHCP Вывод либо установка активной кодовой страницы.

CHDIR Вывод имени либо смена текущей папки.

…

CLS Очистка экрана.

…

DIR Вывод списка файлов и подпапок из указанной папки.

…

HELP Выводит справочную информацию о командах Windows.

…

TREE Графическое отображение структуры каталогов диска или папки.

…

VER Вывод сведений о версии Windows.

…

Дополнительные сведения о программах приведены в описании программ командной строки в справке.

4. Просматриваем содержание директории с помощью DIR

D:\lek\lk01\lk01\bin\Debug>dir

Том в устройстве D имеет метку DATA

Серийный номер тома: EC1B-EF11

Содержимое папки D:\lek\lk01\lk01\bin\Debug

26.09.2014 22:35 <DIR> .

26.09.2014 22:35 <DIR> ..

26.09.2014 22:35 24 576 lk01.exe

19.09.2014 23:07 187 lk01.exe.config

26.09.2014 22:35 36 352 lk01.pdb

19.09.2014 23:42 180 lk01.Resources.resources

20.09.2014 18:26 22 984 lk01.vshost.exe

19.09.2014 23:07 187 lk01.vshost.exe.config

05.03.2013 03:06 490 lk01.vshost.exe.manifest

26.09.2014 22:35 648 lk01.xml

…

16 файлов 553 840 байт

2 папок 5 797 728 256 байт свободно

5. Находим исполняемый файл с именем проекта и запускаем его на выполнение.

D:\lek\lk01\lk01\bin\Debug>lk01.exe

Test 1 NET

Test 2 NET

6. Запускаем программу предварительной компиляции ngen.exe

D:\lek\lk01\lk01\bin\Debug> ngen

Получаем на экране список параметром с которыми запускается эта программа.

D:\lek\lk01\lk01\bin\Debug>ngen

Microsoft (R) CLR Native Image Generator - Version 4.0.30319.19408

Copyright (c) Microsoft Corporation. All rights reserved.

Usage: ngen <action> [args] [/nologo] [/silent] [/verbose]

ngen /? or /help

/nologo - Prevents displaying of logo

/silent - Prevents displaying of success messages

/verbose - Displays verbose output for debugging

ngen install <assembly name> [scenarios] [config] [/queue[:[1|2|3]]

Generate native images for an assembly and its dependencies

and install them in the Native Images Cache

If /queue is specified compilation job is queued up. If a priority

is not specified, the default priority used is 3.

ngen uninstall <assembly name> [scenarios] [config]

Delete the native images of an assembly and its dependencies from

the Native Images Cache.

ngen update [/queue]

Update native images that have become invalid

If /queue is specified compilation jobs are queued up.

7. Компилируем с помощью этой программы наш проект:

D:\lek\lk01\lk01\bin\Debug>ngen install lk01.exe

Microsoft (R) CLR Native Image Generator - Version 4.0.30319.19408

Copyright (c) Microsoft Corporation. All rights reserved.

1> Compiling assembly D:\lek\lk01\lk01\bin\Debug\lk01.exe (CLR v4.0.30319) ..

.

8. Запускаем откомпилированный проект

D:\lek\lk01\lk01\bin\Debug>lk01.exe

Test 1 NET

Test 2 NET

9. Переходим на системный диск

D:\lek\lk01\lk01\bin\Debug>C:

C:\Program Files (x86)\Microsoft Visual Studio 11.0\VC> cd c:\

10. Находим каталог с откомпилированной программой следующим образом:

C:\>cd windows

C:\Windows>cd assembly

C:\Windows\assembly>cd NativeImages\_v4.0.30319\_32

C:\Windows\assembly\NativeImages\_v4.0.30319\_32>

11. Просматриваем помощь для DIR

C:\Windows\assembly\NativeImages\_v4.0.30319\_32>help dir

Вывод списка файлов и подкаталогов в указанном каталоге.

DIR [диск:][путь][имя файла] [/A[[:]атрибуты]] [/B] [/C] [/D] [/L] [/N]

[/O[[:]порядок сортировки]] [/P] [/Q] [/R] [/S] [/T[[:]время]] [/W] [/X] [/4]

[диск:][путь][имя файла]

Диск, каталог или имена файлов для включения в список.

/A Отображение файлов с указанными атрибутами.

атрибуты D Каталоги. R Файлы, доступные только для чтения.

H Скрытые файлы. A Файлы, готовые для архивирования.

S Системные файлы. I Файлы с неиндексированным содержимым.

L Точки повторной обработки. - Префикс "-" имеет значение НЕ.

/B Вывод только имен файлов.

/C Применение разделителя групп разрядов при выводе размеров файлов.

Используется по умолчанию. Чтобы отключить применение

разделителя групп разрядов, задайте ключ /-C.

/D Вывод списка в нескольких столбцах с сортировкой по столбцам.

/L Использовать нижний регистр для имен файлов.

/N Новый формат длинного списка, имена файлов выводятся в крайнем

правом столбце.

/O Сортировка списка отображаемых файлов.

порядок N По имени (по алфавиту)

сортировки S По размеру (начиная с минимального)

E По расширению (по алфавиту)

D По дате и времени (начиная с самого старого)

…

/W Вывод списка в несколько столбцов.

12. Выводим список в несколько столбцов и находим имя откомпилированного проекта.

C:\Windows\assembly\NativeImages\_v4.0.30319\_32>dir /w

Том в устройстве C имеет метку SYSTEMX64

Серийный номер тома: EA12-3C49

Содержимое папки C:\Windows\assembly\NativeImages\_v4.0.30319\_32

[.] [..] [Accessibility]

[AspNetMMCExt] [aspnet\_intern] [aspnet\_merge]

[AxImp] [Blend] [ComSvcConfig]

[ConsoleApplication2] [CustomMarshalers] [dfsvc]

[Fsc] [FSharp.Build] [FSharp.Comp8f056e1a#]

[FSharp.Compc65252bb#] [FSharp.Compiler] [FSharp.Core]

[FSharp.Data087ef7a1#] [FSharp.Lang31116b57#] [FSharp.Projd4f911f5#]

[FSharp.VS.FSI] [Fsi] [FsiAnyCPU]

[GalaSoft.Mv2ad69d8b#] [GalaSoft.Mv954dba8c#] [GFExperience]

[GFExperienceControls] [GFExperienceCore] [GridService]

[InstallerService] [InstallerUIExtension] [IntelliTrace]

[lc] **[lk01]** [log4net]

13. Входим в эту папку и просматриваем ее содержимое. В папке выделенной жирным находится откомпилированный образ.

C:\Windows\assembly\NativeImages\_v4.0.30319\_32>cd lk01

C:\Windows\assembly\NativeImages\_v4.0.30319\_32\lk01>dir

Том в устройстве C имеет метку SYSTEMX64

Серийный номер тома: EA12-3C49

Содержимое папки C:\Windows\assembly\NativeImages\_v4.0.30319\_32\lk01

26.09.2014 23:00 <DIR> .

26.09.2014 23:00 <DIR> ..

26.09.2014 23:00 <DIR> **4355e9f9b3e07e9784933647f28e8223**

0 файлов 0 байт

3 папок 26 510 950 400 байт свободно

C:\Windows\assembly\NativeImages\_v4.0.30319\_32\lk01>

14. Удаляем откомпилированный образ

D:\lek\lk01\lk01\bin\Debug>ngen uninstall lk01.exe

Microsoft (R) CLR Native Image Generator - Version 4.0.30319.19408

Copyright (c) Microsoft Corporation. All rights reserved.

Uninstalling assembly D:\lek\lk01\lk01\bin\Debug\lk01.exe

15. Запускаем программу просмотра бинарного исполняемого файла

D:\lek\lk01\lk01\bin\Debug>dumpbin

Microsoft (R) COFF/PE Dumper Version 11.00.61030.0

Copyright (C) Microsoft Corporation. All rights reserved.

использование: DUMPBIN [параметры] [файлы]

параметры:

/ALL

/ARCHIVEMEMBERS

/CLRHEADER

/DEPENDENTS

/DIRECTIVES

/DISASM[:{BYTES|NOBYTES}]

/ERRORREPORT:{NONE|PROMPT|QUEUE|SEND}

/EXPORTS

/FPO

/HEADERS

/IMPORTS[:имя\_файла]

/LINENUMBERS

/LINKERMEMBER[:{1|2}]

/LOADCONFIG

/OUT:имя\_файла

16. Выводим бинарное представление в файл текста – ttx

D:\lek\lk01\lk01\bin\Debug>dumpbin lk01.exe /all /out:ttx

Microsoft (R) COFF/PE Dumper Version 11.00.61030.0

Copyright (C) Microsoft Corporation. All rights reserved.

17. Находим в текстовом файле данные для исполняющей среды. В заголовке CLR Header мы видим импортируемую функцию \_CorExeMain, реализованную в mscoree.dll — основном модуле среды выполнения. Для реализации возможности выполнения .NET-файлов под управлением операционных систем Windows 98, Windows Me и Windows 2000 фирма Microsoft изменила код загрузчика: теперь он распознает CLR-элементы в исполняемых файлах и передает управление CLR через точку входа — \_CorExeMain. Функция Main() нашей программы вызывается самой средой выполнения.

clr Header:

48 cb

2.05 runtime version

28BC [ 1D1C] RVA [size] of MetaData Directory

20003 flags

IL Only

32-Bit Required

32-Bit Preferred

6000020 entry point token

2803 [ B8] RVA [size] of Resources Directory

0 [ 0] RVA [size] of StrongNameSignature Directory

0 [ 0] RVA [size] of CodeManagerTable Directory

0 [ 0] RVA [size] of VTableFixups Directory

0 [ 0] RVA [size] of ExportAddressTableJumps Directory

0 [ 0] RVA [size] of ManagedNativeHeader Directory

Section contains the following imports:

mscoree.dll

402000 Import Address Table

404600 Import Name Table

0 time date stamp

0 Index of first forwarder reference

0 \_CorExeMain

18. Компания Microsoft включила в .NET Framework SDK утилиту командной строки CLRVer.exe, которая выводит список всех версий CLR, установленных на машине, а также сообщает, какая именно версия среды CLR используется текущими процессами. Для этого нужно указать параметр –all или идентификатор интересующего процесса.

D:\lek\lk01\lk01\bin\Debug>clrver

Microsoft (R) .NET CLR Version Tool Version 4.0.30319.17929

Copyright (c) Microsoft Corporation. All rights reserved.

Versions installed on the machine:

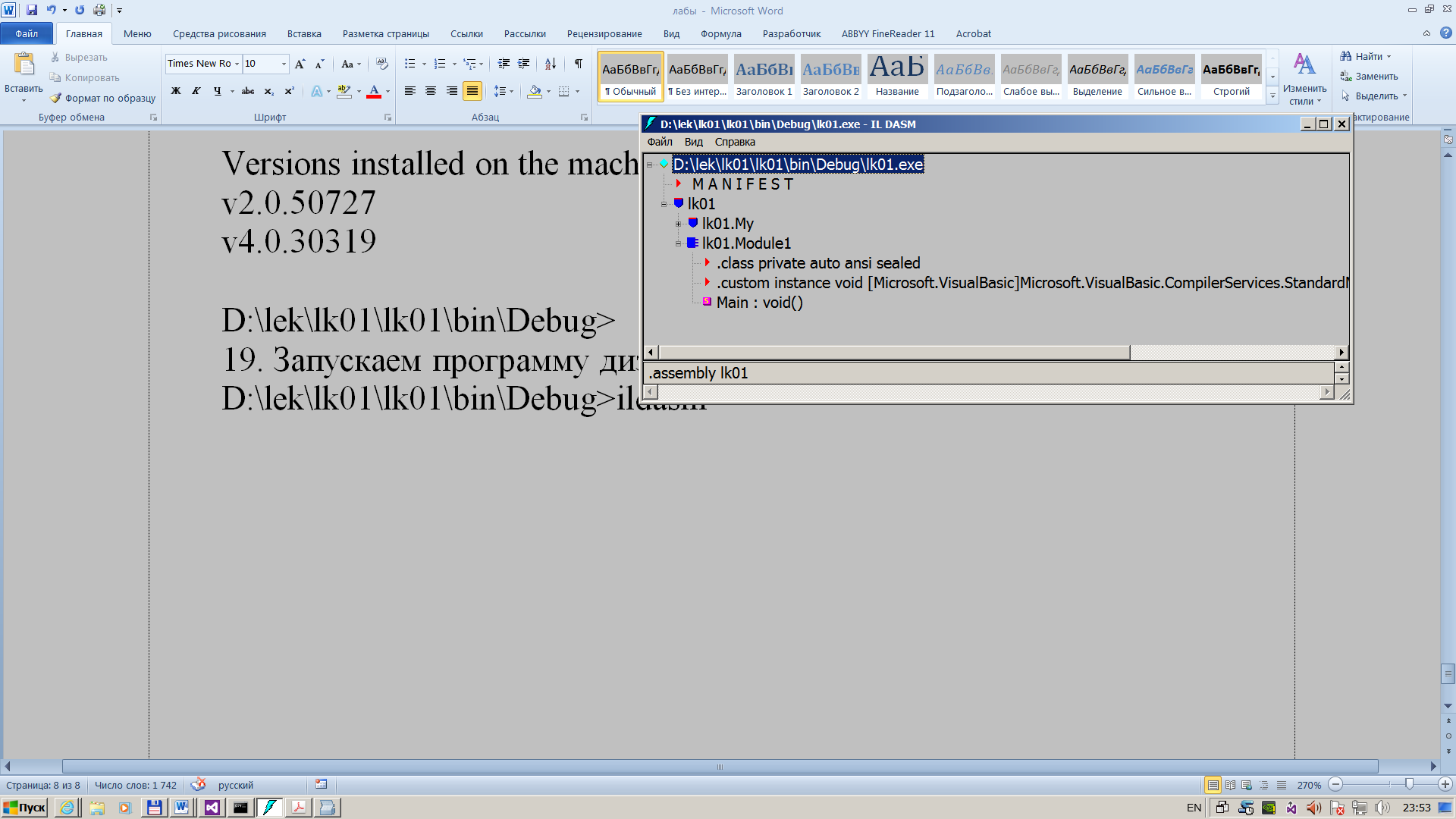
v2.0.50727

v4.0.30319

D:\lek\lk01\lk01\bin\Debug>

19. Запускаем программу дизассемблирования файла

D:\lek\lk01\lk01\bin\Debug>ildasm



.method public static void Main() cil managed

{

.entrypoint

.custom instance void [mscorlib]System.STAThreadAttribute::.ctor() = ( 01 00 00 00 )

// Размер кода: 25 (0x19)

.maxstack 8

IL\_0000: nop

IL\_0001: ldstr "Test 1 NET"

IL\_0006: call void [mscorlib]System.Console::WriteLine(string)

IL\_000b: nop

IL\_000c: ldstr "Test 2 NET"

IL\_0011: call void [mscorlib]System.Console::WriteLine(string)

IL\_0016: nop

IL\_0017: nop

IL\_0018: ret

} // end of method Module1::Main

Лабораторная работа №2

Тема: Разработка класса для выполнения действий над матрицами

Лабораторная работа №3

Тема: Разработка консольного приложения, обеспечивающего ввод/вывод матриц на консоль и работу с классом

Лабораторная работа №4

Тема: Разработка консольного приложения, обеспечивающего ввод/вывод матриц в текстовый файл и работу с классом

Лабораторная работа №5

Тема: Разработка Windows Form приложения, обеспечивающего ввод/вывод матриц в форме или формах ввода, а также работу с классом

Лабораторная работа №6

Тема: Разработка приложения Windows Form для параллельного выполнения задач в одном окне. Например умножения и сложения матриц параллельно.

Лабораторная работа №7

Тема: Разработка консольного приложения, обеспечивающего ввод/вывод матриц в XML файл с помощью LinQ и работу с классом

Лабораторная работа №8

Тема: Разработка веб-старницы для умножения матриц

Лабораторная работа №9

Тема: Разработка сайта для умножения, сложения, вычитания матриц с использованием мастер-страниц и веб-меню, предоставляемого платформой

Лабораторная работа №10

Тема: Добавление к сайту, разработанному в работе 9, возможности хранения и отображения результатов вычислений над матрицами в базе данных по выбору.

Лабораторная работа №11

Тема: Разработка WPF приложения, обеспечивающего ввод/вывод матриц в форме или формах ввода, а также работу с классом

Лабораторная работа №12

Тема: Разработка приложения WPF для параллельного выполнения задач в одном окне. Например умножения и сложения матриц параллельно.

Лабораторная работа №13

Тема: Разработка WPF приложения, обеспечивающего ввод/вывод матриц в форме или формах ввода, а также работу с классом на основе шаблона MVVM