Объектно-ориентированное программирование

Пацей Наталья Владимировна

кафедра Программной инженерии

n.patsei@belstu.by 327-1

- 1 семестр - 1 л. + 1лек (36+36) → экз
- 2 семестр – 1 л. + 1 лек – > (32 + 32) → экз + курс.

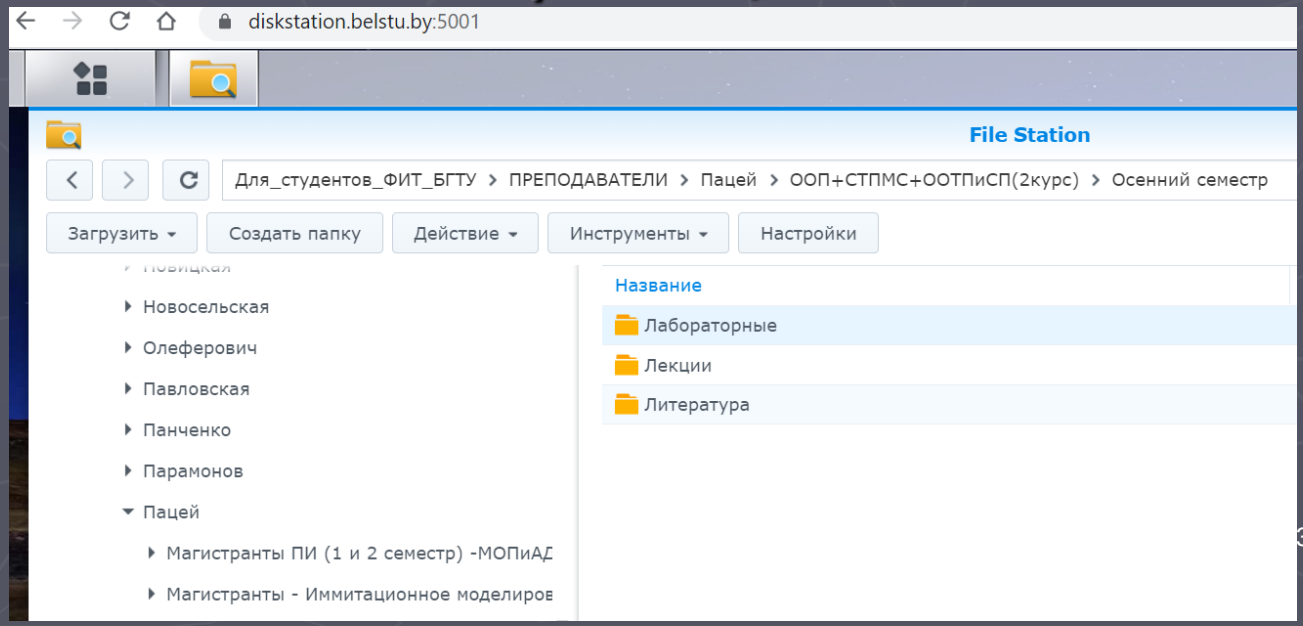
Необходимый материал

1) Лекции
2) Книги
3) Материалы

https://diskstation.belstu.by:5001/

Student

fitfit



Как будут выставляться оценки

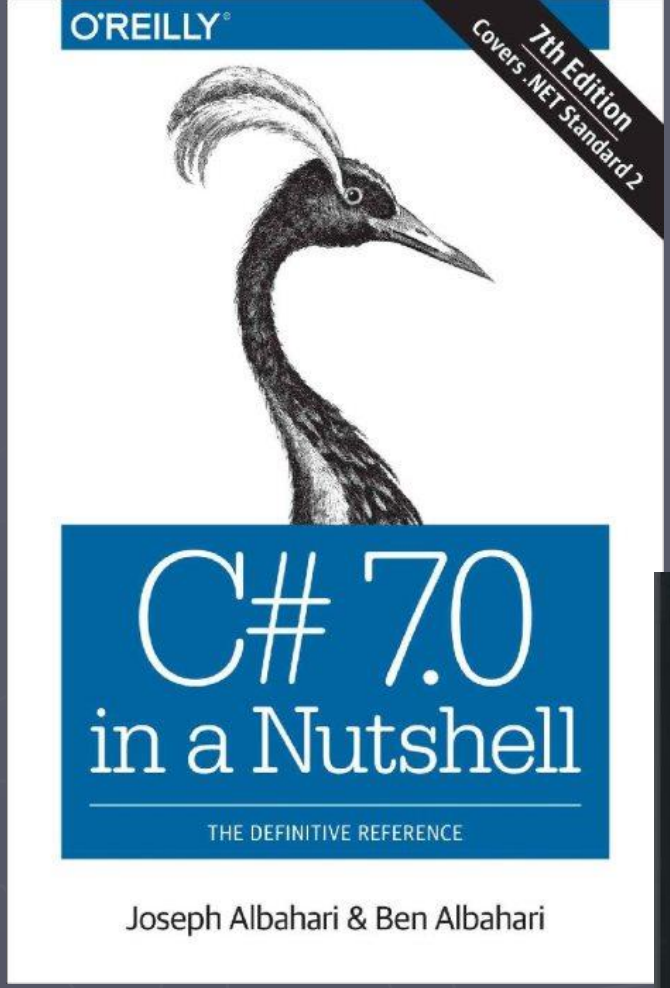
40% Оценка за лаборатор. + Оценка за аттестации

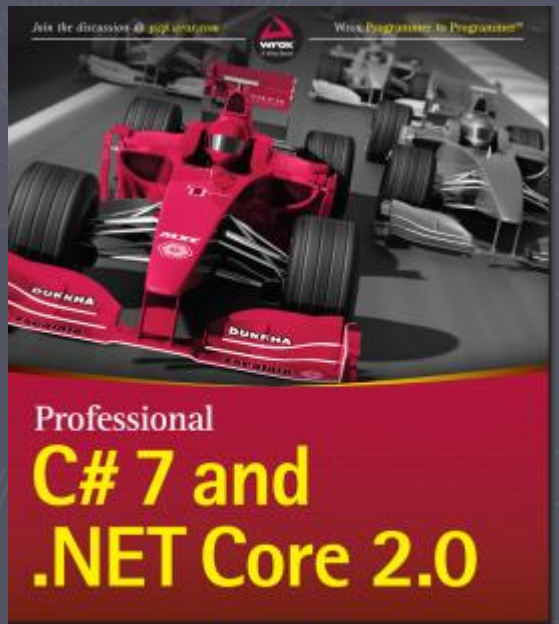
10% Оценка за тесты на лекции (теория)

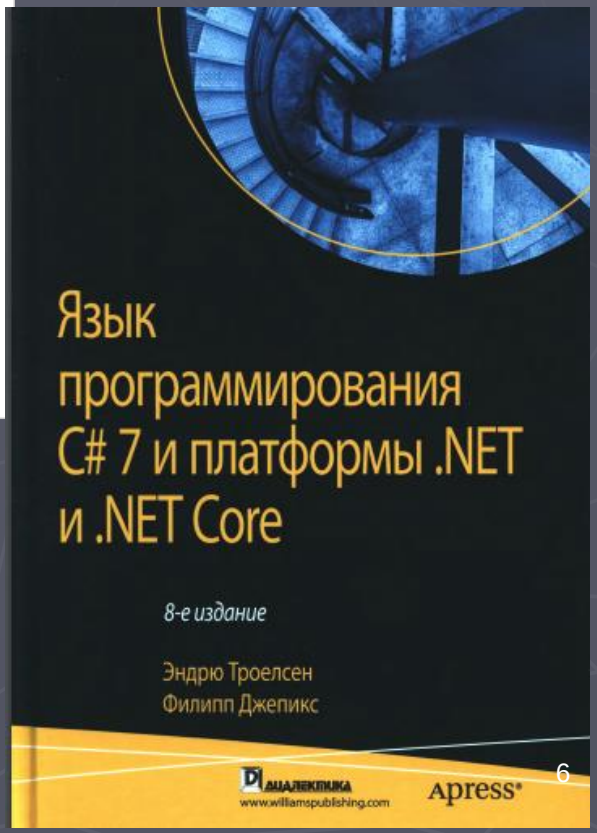
50% Экзамен

.NET, CLR, C#


Microsoft
МАСТЕР КЛАСС
CLR via C#
ПРОГРАММИРОВАНИЕ НА ПЛАТФОРМЕ MICROSOFT .NET FRAMEWORK 4.5 НА ЯЗЫКЕ C#
4-е издание
Джеффри Рихтер
ПИТЕР


O'REILLY
7th Edition
Covers .NET Standard 2
C# 7.0
in a Nutshell
THE DEFINITIVE REFERENCE
Joseph Albahari & Ben Albahari


Professional
C# 7 and
.NET Core 2.0


Язык программирования C# 7 и платформы .NET и .NET Core
8-е издание
Эндрю Троелсен
Филипп Джепикс
ДИАЛЕКТИКА
www.williamspublishing.com
apress

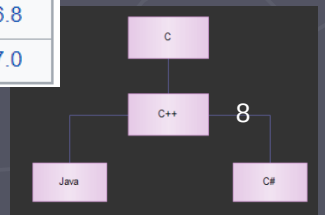
- Пацей, Н. В. Объектно-ориентированное программирование на C++/C#: учеб.-метод. пособие . В 2 ч., Ч. 1/ Н. В. Пацей – Минск.: БГТУ, 2014. – 191 с.
- Книги в папке на diskstation

- https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/
- https://github.com/dotnet/csharplang/blob/master/spec/README.md
- professorweb.ru
- https://metanit.com/sharp/general.php

C# - 2000г. Андрес Хэйлсберг

Версия	Спецификация языка			Дата	.NET	Visual Studio
	ECMA	ISO/IEC	Microsoft			
C# 1.0	Декабрь 2002	Апрель 2003 (недоступная ссылка)	Январь 2002	Январь 2002	.NET Framework 1.0	Visual Studio .NET (2002)
C# 1.1 C# 1.2			Октябрь 2003	Апрель 2003	.NET Framework 1.1	Visual Studio .NET 2003
C# 2.0	Июнь 2006	Сентябрь 2006	Сентябрь 2005[14]	Ноябрь 2005	.NET Framework 2.0 .NET Framework 3.0	Visual Studio 2005
C# 3.0	Отсутствует[15]		Август 2007	Ноябрь 2007	.NET Framework 2.0 (кроме LINQ) .NET Framework 3.0 (кроме LINQ) .NET Framework 3.5	Visual Studio 2008
C# 4.0			Апрель 2010	Апрель 2010	.NET Framework 4.0	Visual Studio 2010
C# 5.0	Декабрь 2017	Декабрь 2018	Июнь 2013	Август 2012	.NET Framework 4.5	Visual Studio 2012
C# 6.0	Отсутствует		Июль 2015		.NET Framework 4.6 .NET Core 1.0 .NET Core 1.1	Visual Studio 2015
C# 7.0			Март 2017		.NET Framework 4.6.2 .NET Framework 4.7	Visual Studio 2017 15.0
C# 7.1			Август 2017		.NET Core 2.0	Visual Studio 2017 15.3
C# 7.2			Ноябрь 2017			Visual Studio 2017 15.5
C# 7.3			Май 2018		.NET Core 2.1 .NET Core 2.2 .NET Framework 4.8	Visual Studio 2017 15.7
C# 8.0			Сентябрь 2019		.NET Core 3.0 .NET Core 3.1 .NET Framework 4.8	Visual Studio 2019 16.3
C# 9.0			Сентябрь 2020		.NET 5.0	Visual Studio 2019 16.8
C# 10.0			Июль 2021		.NET 6.0	Visual Studio 2022 17.0

- ECMA-334 и ISO/IEC 23271



Спецификация CLI

- ***CLI (Common Language Infrastructure)*** – спецификация общеязыковой инфраструктуры. Определяет архитектуру исполнительной системы и набор представляемых сервисов.

Стандарты: ECMA-335 и ISO/IEC 23271.

ISO/IEC 23271

- Концепция и архитектура. Здесь определены понятия: ***CTS (Common Type System)***, ***VES (Virtual Execution System***) и ***CLS (Common Language Specification)***
- Метаданные и семантика.
- Инструкции CIL. ***CIL(Common Intermediate Language)***.
- Библиотеки и профили. ***BCL(Base Class Library)***. Описание библиотеки поставляется в виде xml-файла ***CLILibraryTypes.xml***.
- Формат взаимодействия с отладчиком
- Приложения

.NET Framework

- Microsoft.NET (.NET Framework) – программная платформа. Содержит следующие основные компоненты:

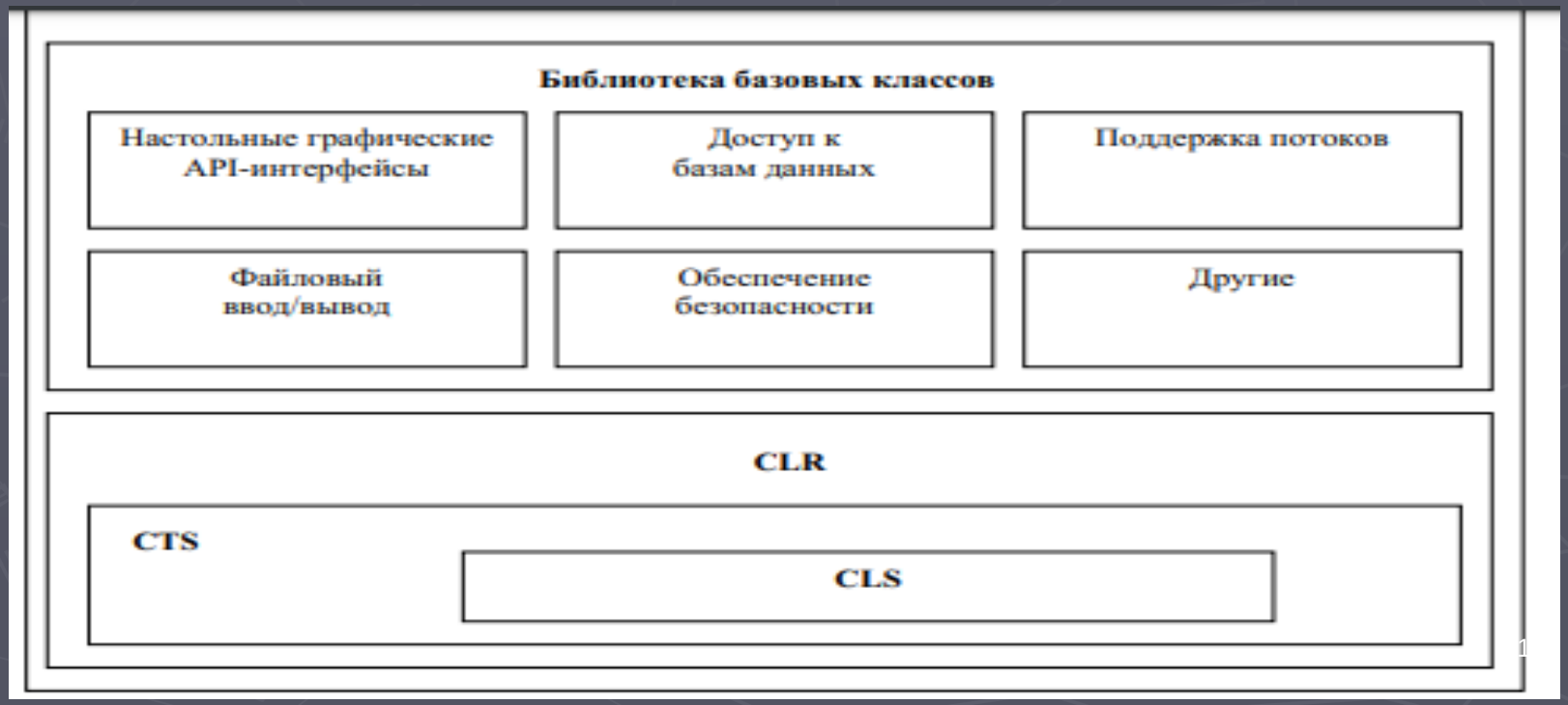
обеспечивает совместное использование разных языков программирования, а также безопасность, переносимость программ и общую модель программирования для платформы Windows

CLR (Common Language Runtime) – общеязыковая среда исполнения, виртуальная машина на которой исполняются все приложения, работающие в среде .NET. Реализация CLI VES компанией Microsoft. Компилятор ***JIT(Just in Time)***.

MSIL(Microsoft IL) – реализация CLI CIL компанией Microsoft.

FCL (Framework Class Library) – реализация CLI BCL компанией Microsoft. Можно рассматривать, как API CRL.

- CLR (Common Language Runtime) – Среда Времени Выполнения или Виртуальная Машина. Обеспечивает выполнение сборки (управление памятью, загрузка сборок, безопасность, обработка исключений, синхронизация)
- FCL (.NET Framework Class Library) – соответствующая CLS спецификации объектно-ориентированная библиотека классов, интерфейсов и системы типов (типов-значений)



Традиционные Windows-приложения

.NET-приложения
Базовые типы: Windows Application, Console Application Class Library

Visual Studio .NET

Языки программирования Microsoft (VB, C/C++, C#, J#, JScript) и независимых поставщиков

Common Language Spetification

Типы .NET-приложений (Console, Windows Forms, Components, ASP .NET, Web Services и пр.)

.NET Framework

Библиотеки классов

Базовые классы .NET Framework

Дополнительные классы .NET

Связь с COM-объектами

CLR (Common Language Runtime)

Windows

Сервисы операционной системы (Win32 API)

.NET FRAMEWORK – решение следующих проблем

- 1. **Интеграция языков программирования.**
- **CLS (Common Language Specification)** – общеязыковая спецификация, предназначенная для разработчиков компиляторов. →**CTS (Common Type Systems)**- спецификацию типов, которые должны поддерживаться всеми языками ориентированными на CLR. Microsoft выпустил несколько компиляторов соответствующих этой спецификации: C++/CLI (C++ с управляемыми расширениями), C#, VB .NET, JScript.

2. Работа на многих платформах.

- При компиляции кода компиляторы NET Framework генерируют код на промежуточном языке (**CIL**, **Common Intermediate Language**). При исполнении CLR транслирует CIL-код в команды соответствующего процессора. В принципе, однажды .NET Framework-приложение должно работать везде, где установлены и работают CLR и FCL.

- **3. Упрощенное повторное использование кода.**

- CLR позволяет типы разработанные на одном языке использовать в других языках.

- **4. Автоматическое управление памятью.**

- CLR автоматически отслеживает использование ресурсов. Сборщик мусора.

- **5. Проверка безопасности типов.**

- При работе в CLR практически исключена возможность записать (стереть) данные в область памяти, которая для этого не предназначена. Нет возможности передать управление в произвольную точку.

- **6. Единый принцип обработки сбоев.**

- Один из самых неприятных моментов в Window-программирование – это отсутствие единой системы обработки ошибок и сбоев: возврат функций, коды состояний, HRESULT, исключения и т.п. Для обработки ошибок и сбоев в CLR используется только механизм исключений.

- 7. **Взаимодействие с существующим кодом**.

- Поддерживаются функции Win32 DLL – библиотек.

8. Проблемы с версиями.

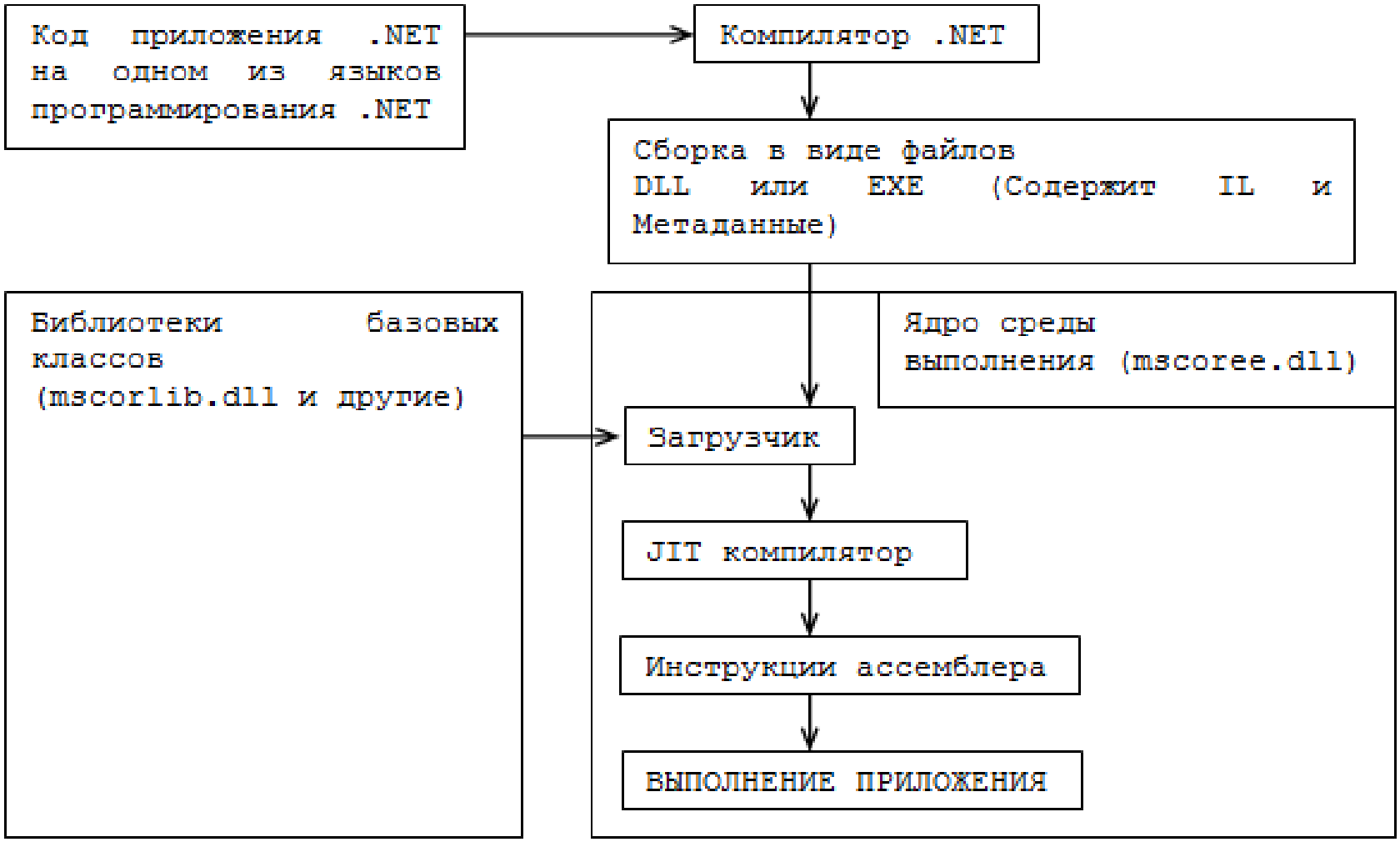
- В Windows возникают проблемы связанные с совместимостью DLL-библиотек. .NET Framework приложение всегда работает с компонентами с которыми компилировалось и тестировалось приложение.

CLR

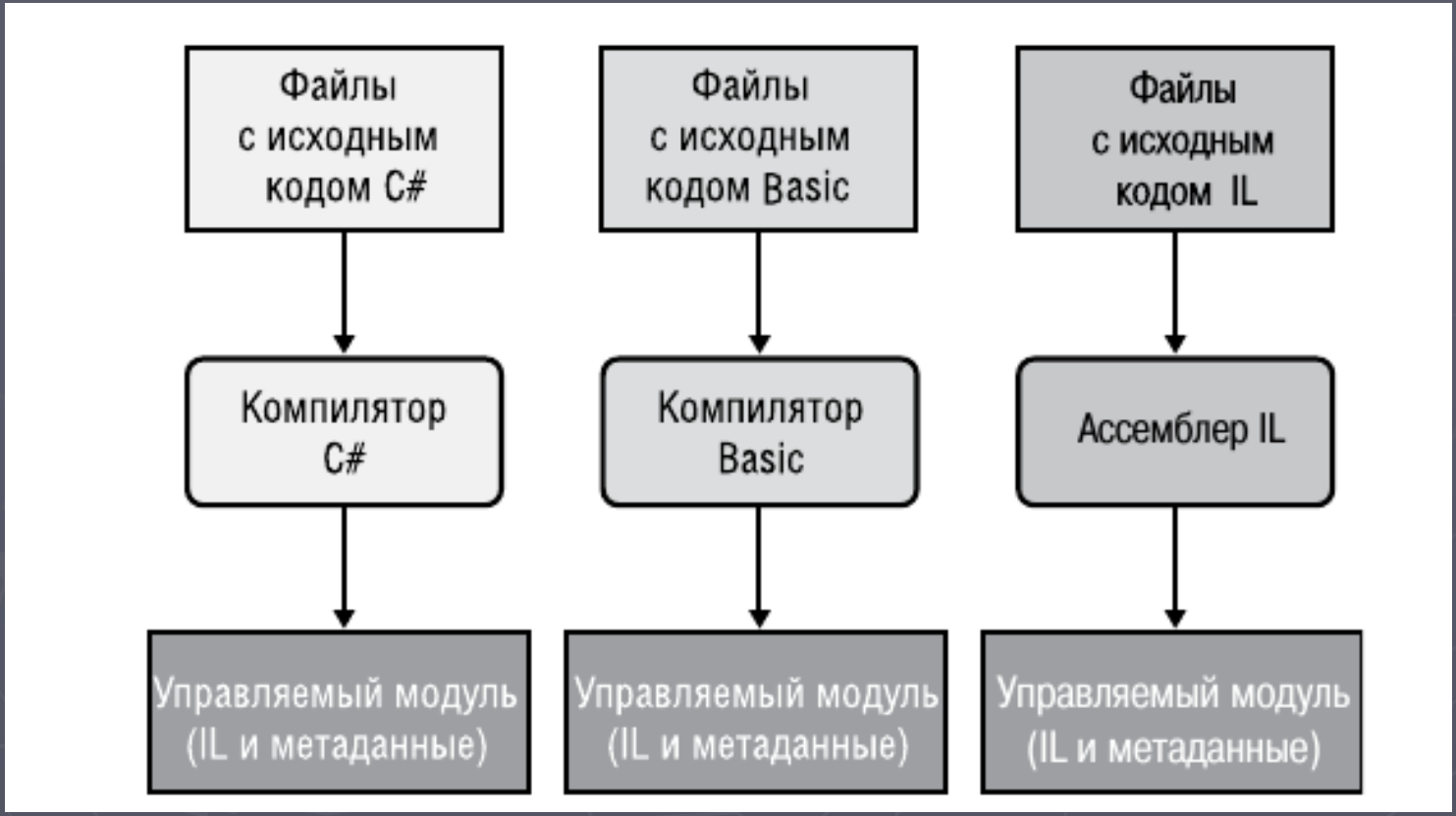
C++/CLI, Visual Basic, F#, Iron Python, Iron Ruby и ассемблер Intermediate Language (IL)

Ada, APL, Caml, COBOL, Eiffel, Forth, Fortran, Haskell, Lexico, LISP, LOGO, Lua, Mercury, ML, Mondrian, Oberon, Pascal, Perl, Php, Prolog, RPG, Scheme, Smalltalk

Структура среды выполнения CLR



1) Компиляция исходного кода в управляемые модули



assembly

2) Выполнения кода в среде CLR

CLR (Common Language Runtime – общеязыковая исполняющая среда)

IL

оперативная компиляция (Just In Time – JIT).

машинный код

IL-код компилятора	машинный JIT-код

Оптимизация

/optimize и /debug

NGen.exe IL->машинный код

IL

объектно-ориентированный машинный язык не зависящий от процессора

ILAsm.exe – ассемблер

ILDasm.exe - дизассемблер IL

Компиляция

csc.exe /out:Program.exe /t:exe /r:MSCorLib.dll Program.cs

csc.exe Program.cs

PE (portable executable

Управляемы модуль - portable executable (PE)

Заголовок PE32 или PE32+
Заголовок CLR
Метаданные
Код Intermediate Language (IL)

заголовок PE (32/64),

заголовок CLR (версия CLR, точки входа модуля, размеры и месторасположение ресурсов и метаданных),

метаданные (специальные таблицы, содержащие исходный код типов и членов данных);

код IL (код который CLR компилирует в команды процессора).

Метаданные

двоичный набор таблиц данных:

типы и их члены

портируемые типы и их члены

Назначение:

1)устраняют необходимость в заголовочных файлах (прототипы);

2) используются в VS для подсказок;

3) используется при верификации кода на предмет безопасных операций;

4) можно сериализовать объект одной машине и восстановить состояние объекта на другой машине;

5) используются при сборке мусора.

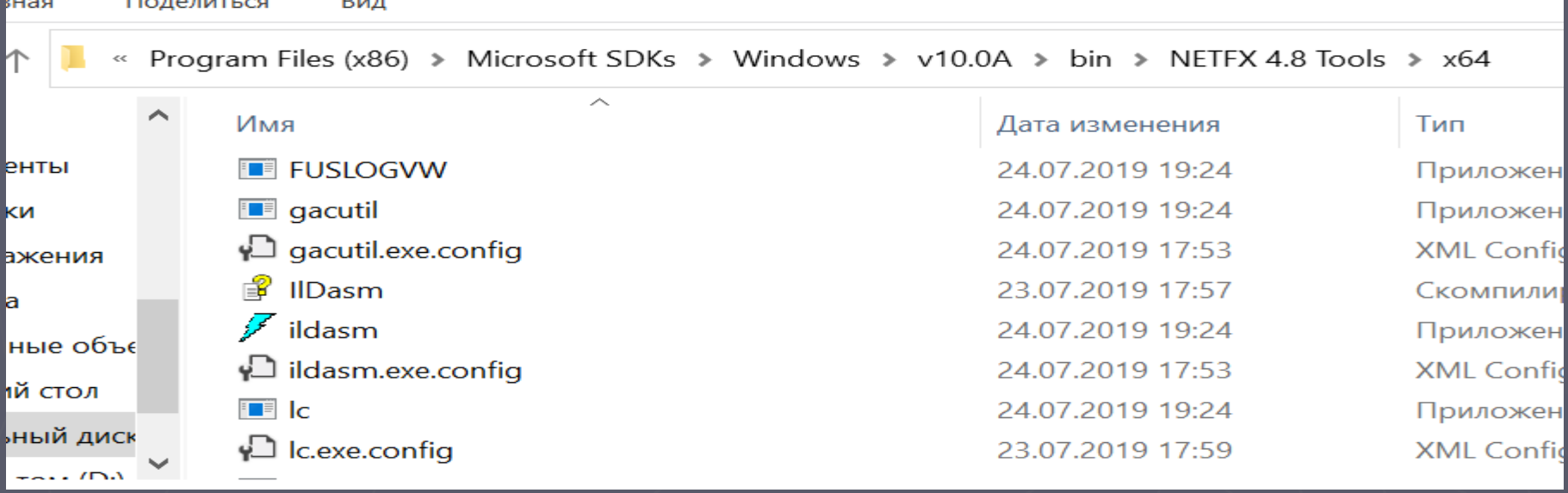
Таблицы определений:

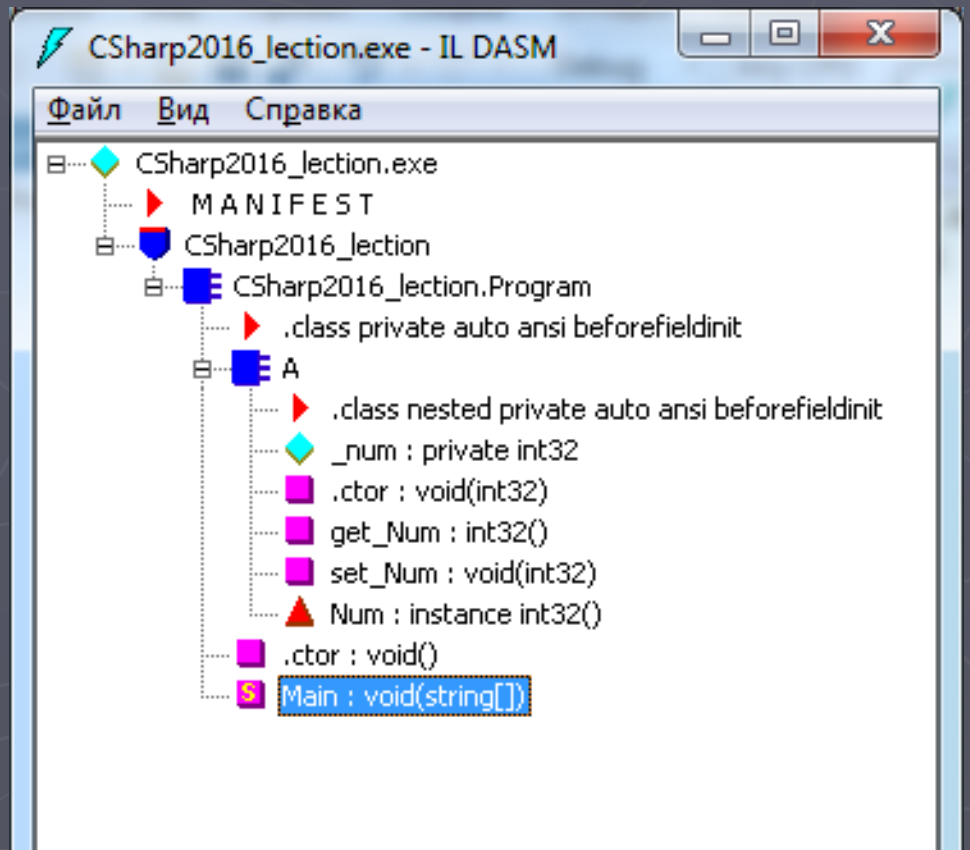
- **ModuleDef** – одна запись, идентифицирующая модуль (версия, имя, GUID).
- **TypeDef** – запись для каждого типа, определенного в модуле.
- **MethodDef** – запись для каждого метода (сигнатура, смещение в модуле кода MSIL, ссылка на **ParamDef**).
- **FieldDef** – запись для каждого поля.
- **ParamDef** – запись для каждого параметра.
- **PropertyDef** – запись для каждого свойства.
- **EventDef** –запись для каждого события.

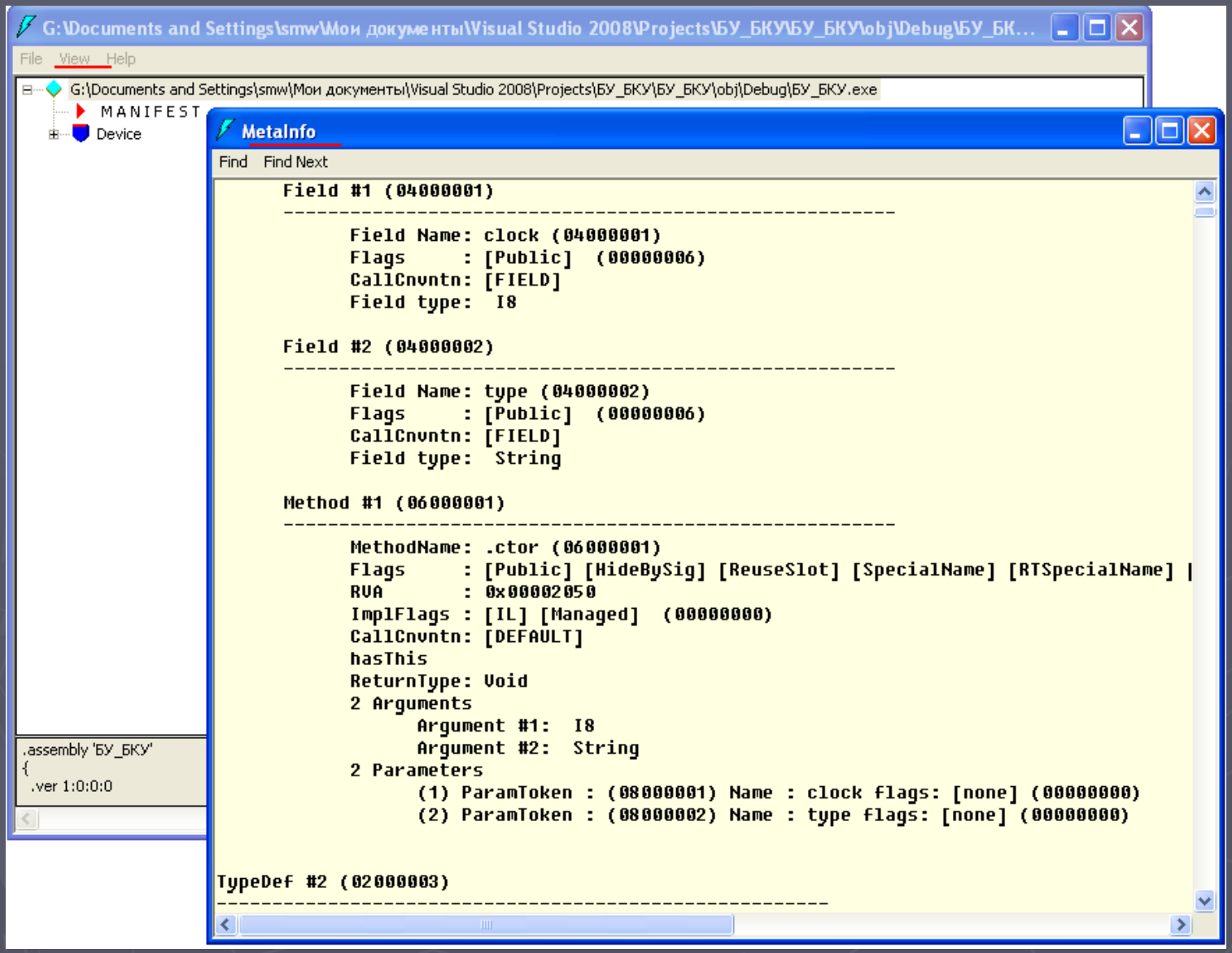
- **Таблицы ссылок**
- **AssemmlyRef** – запись для каждой сборки на которую ссылается модуль.
- **ModuleRef** – запись для каждого PE-модуля, на типы которого ссылается код модуля.
- **TypeRef** – запись для каждого типа на который ссылается модуль.
- **MemberType** - запись для каждого члена, на который ссылается модуль.

Таблицы метаданных декларации

- **AssemmlyDef** – идентифицирует сборку
- **FileDef** - по одной для каждого PE-файла и файла ресурсов.
- **ManifestResourceDef –** по одной для каждого файла ресурсов.
- **ExportedTypesDef –** записи для всех открытых (public) типов.


Program Files (x86) > Microsoft SDKs > Windows > v10.0A > bin > NETFX 4.8 Tools > x64
Имя
Дата изменения
Тип
FUSLOGVW 24.07.2019 19:24 Приложен
gacutil 24.07.2019 19:24 Приложен
gacutil.exe.config 24.07.2019 17:53 XML Config
IlDasm 23.07.2019 17:57 Скомпилир
ildasm 24.07.2019 19:24 Приложен
ildasm.exe.config 24.07.2019 17:53 XML Config
lc 24.07.2019 19:24 Приложен
lc.exe.config 23.07.2019 17:59 XML Config

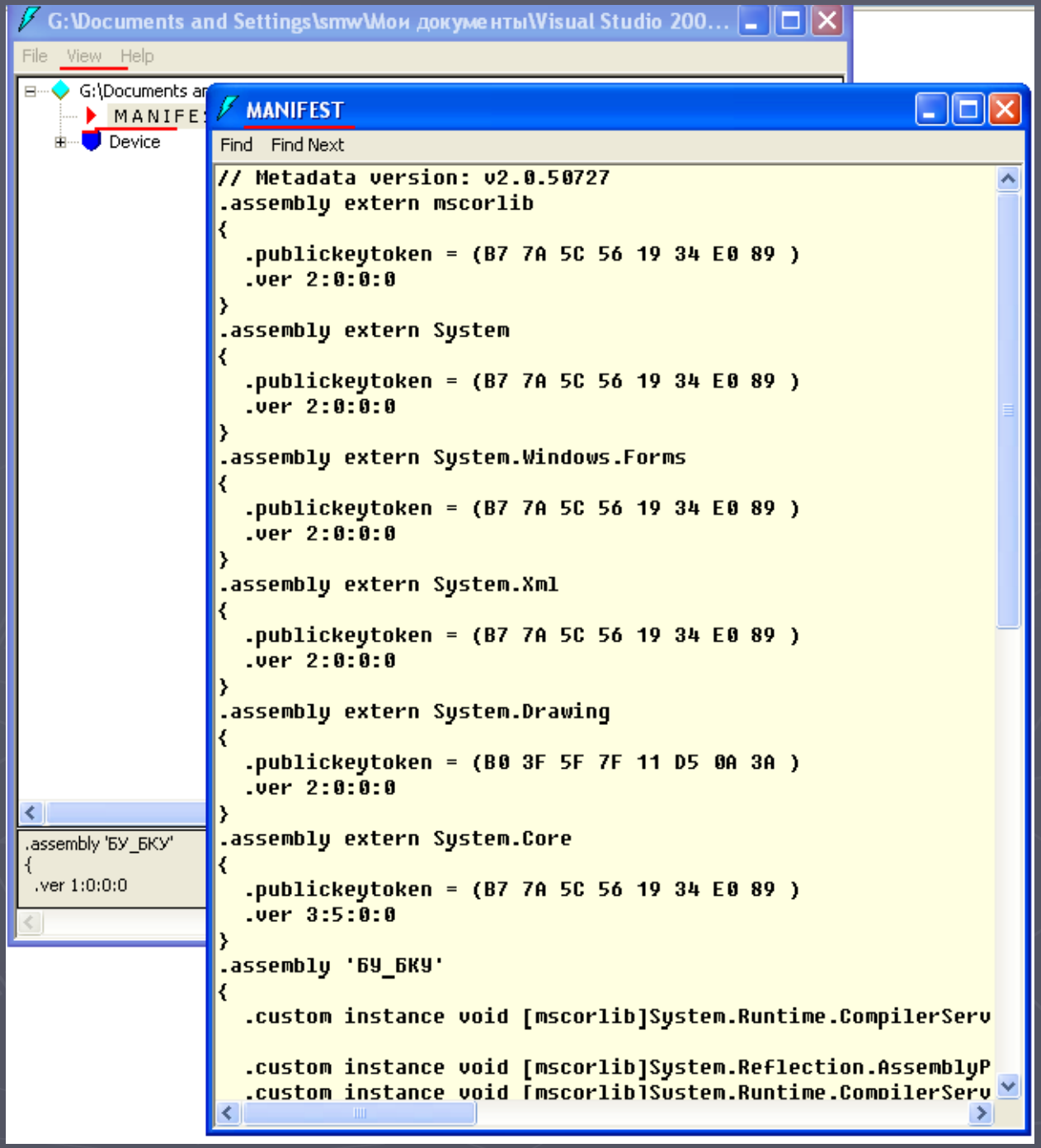

CSharp2016_lection.exe - IL DASM
Файл Вид Справка
CSharp2016_lection.exe
MANIFEST
CSharp2016_lection
CSharp2016_lection.Program
.class private auto ansi beforefieldinit
A
.class nested private auto ansi beforefieldinit
_num : private int32
.ctor : void(int32)
get_Num : int32()
set_Num : void(int32)
Num : instance int32()
.ctor : void()
Main : void(string[])


G:\Documents and Settings\smw\Мои документы\Visual Studio 2008\Projects\БУ_БКУ\БУ_БКУ\obj\Debug\БУ_БК...
File View Help
G:\Documents and Settings\smw\Мои документы\Visual Studio 2008\Projects\БУ_БКУ\БУ_БКУ\obj\Debug\БУ_БКУ.exe
MANIFEST
Device
.assembly 'БУ_БКУ'
{
.ver 1:0:0:0
MetaInfo
Find Find Next
Field #1 (04000001)
Field Name: clock (04000001)
Flags : [Public] (00000006)
CallCnvntn: [FIELD]
Field type: I8
Field #2 (04000002)
Field Name: type (04000002)
Flags : [Public] (00000006)
CallCnvntn: [FIELD]
Field type: String
Method #1 (06000001)
MethodName: .ctor (06000001)
Flags : [Public] [HideBySig] [ReuseSlot] [SpecialName] [RTSpecialName] |
RVA : 0x00002050
ImplFlags : [IL] [Managed] (00000000)
CallCnvntn: [DEFAULT]
hasThis
ReturnType: Void
2 Arguments
Argument #1: I8
Argument #2: String
2 Parameters
(1) ParamToken : (08000001) Name : clock flags: [none] (00000000)
(2) ParamToken : (08000002) Name : type flags: [none] (00000000)
TypeDef #2 (02000003)

Манифест

набор таблиц метаданных

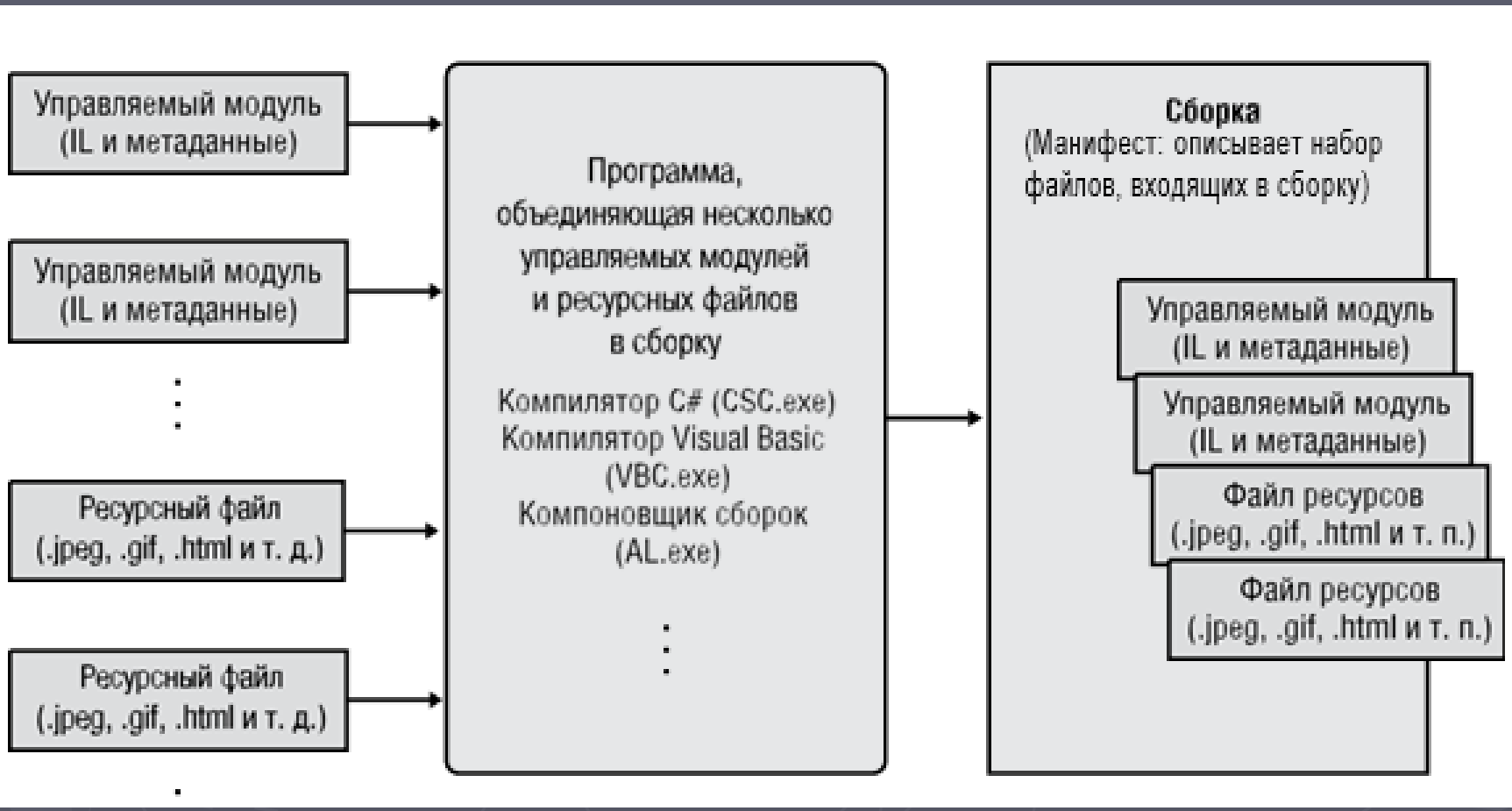
- файлы, которые входят в сборку,
- общедоступные экспортируемые типы,
- файлы ресурсов или данных


G:\Documents and Settings\smw\Мои документы\Visual Studio 200...
File View Help
G:\Documents ar
MANIFE
Device
.assembly 'БУ_БКУ'
{
.ver 1:0:0:0
MANIFEST
Find Find Next
// Metadata version: v2.0.50727
.assembly extern mscorlib
{
.publickeytoken = (B7 7A 5C 56 19 34 E0 89)
.ver 2:0:0:0
}
.assembly extern System
{
.publickeytoken = (B7 7A 5C 56 19 34 E0 89)
.ver 2:0:0:0
}
.assembly extern System.Windows.Forms
{
.publickeytoken = (B7 7A 5C 56 19 34 E0 89)
.ver 2:0:0:0
}
.assembly extern System.Xml
{
.publickeytoken = (B7 7A 5C 56 19 34 E0 89)
.ver 2:0:0:0
}
.assembly extern System.Drawing
{
.publickeytoken = (B0 3F 5F 7F 11 D5 0A 3A)
.ver 2:0:0:0
}
.assembly extern System.Core
{
.publickeytoken = (B7 7A 5C 56 19 34 E0 89)
.ver 3:5:0:0
}
.assembly 'БУ_БКУ'
{
.custom instance void [mscorlib]System.Runtime.CompilerServ
.custom instance void [mscorlib]System.Reflection.AssemblyP
.custom instance void [mscorlib]System.Runtime.CompilerServ

Сборки

- *Сборка* (assembly) — 1) это абстрактное понятие, для логической группировки одного или нескольких управляемых модулей или файлов ресурсов.
- 2) дискретная единица многократно используемого кода внутри CLR

Exe, dll


Управляемый модуль (IL и метаданные)
Управляемый модуль (IL и метаданные)
Ресурсный файл (.jpeg, .gif, .html и т. д.)
Ресурсный файл (.jpeg, .gif, .html и т. д.)
Программа, объединяющая несколько управляемых модулей и ресурсных файлов в сборку
Компилятор C# (CSC.exe)
Компилятор Visual Basic (VBC.exe)
Компоновщик сборок (AL.exe)
Сборка
(Манифест: описывает набор файлов, входящих в сборку)
Управляемый модуль (IL и метаданные)
Управляемый модуль (IL и метаданные)
Файл ресурсов (.jpeg, .gif, .html и т. п.)
Файл ресурсов (.jpeg, .gif, .html и т. п.)

Исполнение сборки

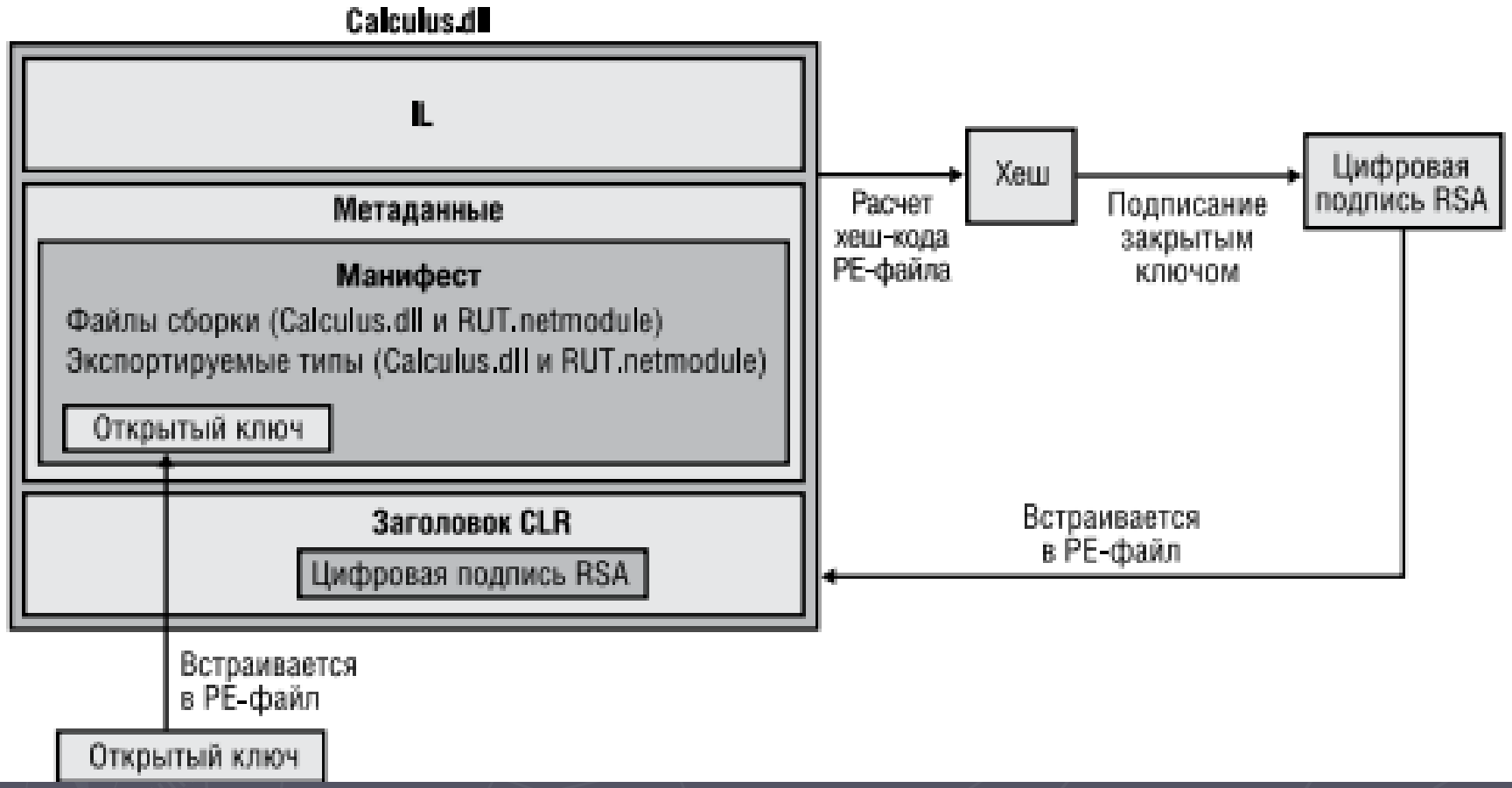
JIT-компилятор (Just-In-Time)

1) CLR ищет типы данных и загружает во внутренние структуры
2) Для каждого метода CLR заносит адрес внутренней CLR функции JITCompiler
3) JITCompiler ищет в метаданных соответствующей сборки IL-код вызываемого метода, проверяет и компилирует IL-код в машинные команды
4) Они хранятся в динамически выделенном блоке памяти.
5) JITCompiler заменяет адрес вызываемого метода адресом блока памяти, содержащего готовые машинные команды
6) JITCompiler передает управление коду в этом блоке памяти.

Типы сборок:

- с нестрогими именами (weakly named assemblies)
- со строгими именами (strongly named assemblies).
  - подписаны при помощи пары ключей, уникально идентифицирующей издателя сборки (безопасность, управление ее версиями, развертывание в любом месте пользовательского жесткого диска или в Интернете)
  - атрибуты: имя файла (без расширения), номер версии, идентификатор регионального стандарта и открытый ключ.

Подписание сборки



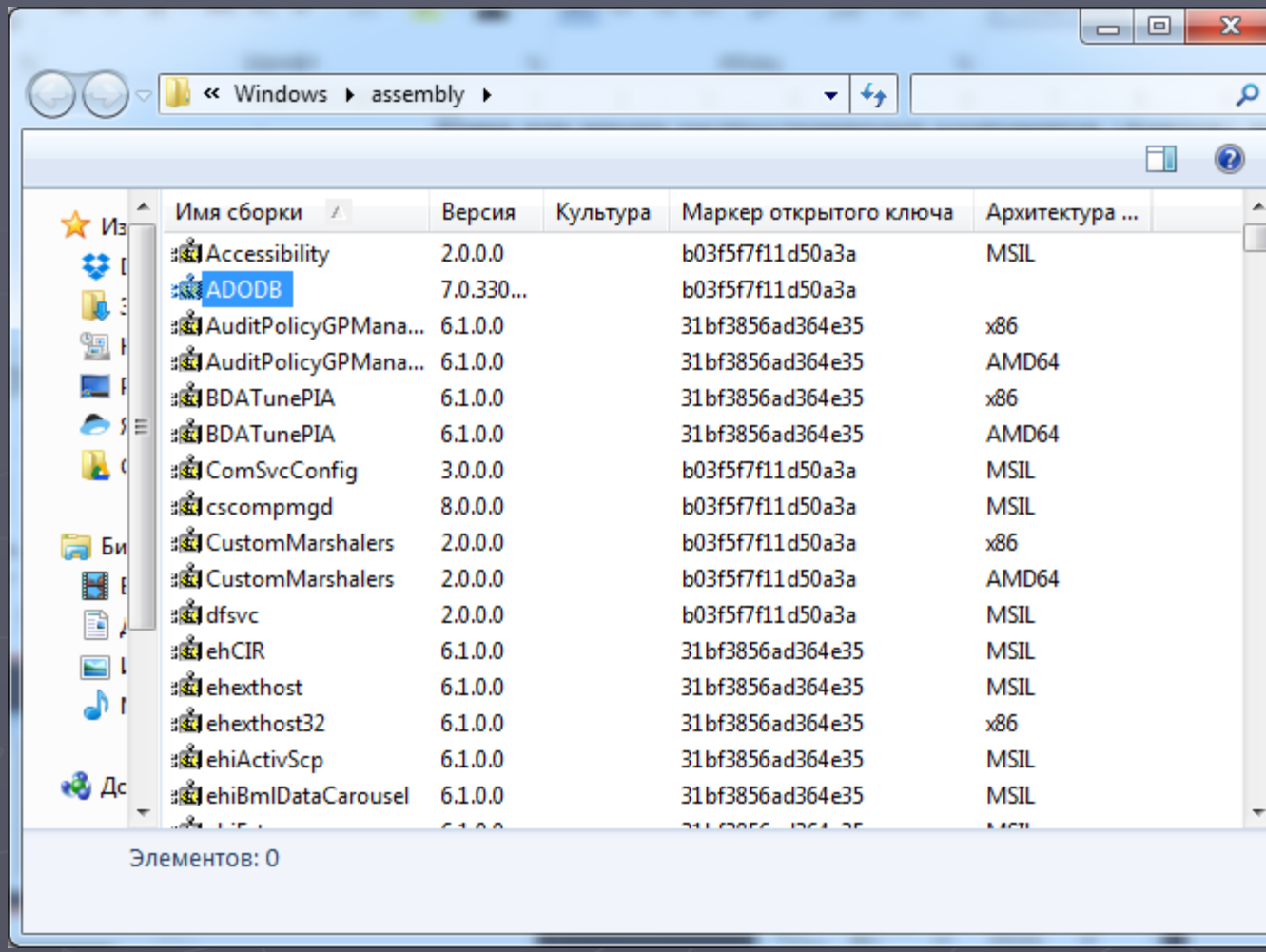
Развертывание сборки

- закрытым
  - развертываются в базовом каталоге приложения или в одном из его подкаталогов. Для сборки с нестрогим именем возможно лишь закрытое развертывание.
- глобальным

%systemroot%\assembly

GAC – Global Assembly Cache

GACUtil.exe



Windows Installer (MSI)

- Читаем часть 1

С 1 -120 стр.