А.В. Макаров, С.Ю. Скоробогатов, А.М. Чеповский

Учебный курс “CIL и системное программирование в Microsoft .NET”

Лекция 12.

Синтаксис ILASM

Введение

- В составе .NET Framework SDK поставляется ассемблер ILASM, позволяющий компилировать текстовые файлы, содержащие CIL-код и метаданные.
- Будем называть IL-форматом формат файлов, поддерживаемый ассемблером ILASM, а программы, записанные в IL-формате, – IL-программами.

12.1. Основные элементы лексики

- Программы в IL-формате состоят из следующих лексических элементов:
  - идентификаторы;
  - метки;
  - константы;
  - зарезервированные слова;
  - специальные знаки;
  - комментарии.

Идентификаторы и метки

- Идентификаторы и метки чаще всего представляют последовательности символов, начинающиеся с латинской буквы (или с символов "_", "$", "@" и "?"), за которой следуют латинские буквы, цифры или символы "_", "$", "@" и "?".
- Для идентификаторов и меток существует особая форма записи в апострофах: она позволяет включать в идентификаторы любые символы Unicode.
- Примеры:

```
Label_1   $Name   'Идентификатор'
```

- Несколько идентификаторов могут быть объединены в один идентификатор с помощью точек. Например:

```
System.Console.WriteLine
```

Числовые константы

- Целочисленные константы записываются либо в десятичной системе счисления, либо в шестнадцатеричной (тогда перед ними ставится префикс "0x"). Например:

```
128   -10   0xFF10B000
```

- В вещественных константах точка используется для разделения целой и дробной части, а символы "e" и "E" служат для указания экспоненциальной части. Кроме того, поддерживается особая форма записи float32 (целая_константа) и float64(целая_константа), позволяющая представить целое число в виде числа с плавающей точкой. Например:

```
5.5   -1.05e10   float32(128)   float64(50)
```

Строковые константы

- Строковые константы записываются в двойных кавычках и могут содержать Escape-последовательности "\t", "\n" и "\xxx", где восьмеричное число xxx задает код символа от 0 до 255.
- Для переноса строковой константы на другую строку программы используется символ "\".
- Для строковых констант поддерживается операция конкатенации "+".
- Например:

```
"Alpha Beta Gamma"  "Hello, World\n"
"Concat"+"enation"
```

Комментарии

- Комментарии в IL-программах записываются так же, как в языке C#:
  - Если в строке программы встречается "//", то остаток строки считается комментарием.
  - Текст, начинающийся с "/*", оканчивающийся на "*/" и не содержащий "*/", считается комментарием.

12.2. Синтаксис

- IL-программа представляет собой последовательность объявлений. В этом разделе мы рассмотрим синтаксис объявлений следующих элементов IL-программы:
  - сборка;
  - модуль;
  - тип;
  - поле;
  - метод.

12.2.1. Сборки и модули

- Образец заголовка IL-файла для одномодульной сборки:

```
.assembly MyProgram { }
.module MyProgram.exe
.assembly extern mscorlib { }
```

  - Директива ".assembly" позволяет задать имя нашей сборки
  - Директива ".module" определяет имя модуля и совпадает с именем исполняемого файла, в который будет записана откомпилированная сборка
  - Директива ".assembly extern" указывает, что мы будем импортировать сборку mscorlib, в которой находится основная часть библиотеки классов .NET

12.2.2. Типы

- Объявление типа осуществляется с помощью директивы ".class" и состоит из четырех частей:
  - последовательность атрибутов типа;
  - имя типа;
  - базовый тип;
  - список реализуемых интерфейсов.

Атрибуты типов

- Последовательность атрибутов следует непосредственно после ключевого слова ".class".

Атрибут	Описание
abstract	Тип является абстрактным классом.
interface	Тип является интерфейсом.
private	Тип не экспортируется из сборки.
public	Тип экспортируется из сборки.
sealed	Тип не может являться базовым классом для другого типа (от него нельзя наследовать).
serializable	Экземпляры типа могут быть сериализованы.

Остальные элементы объявления типа

- После атрибутов следует идентификатор, задающий имя объявляемого типа.
- Если объявляемый тип наследует от какого-нибудь другого типа (базового класса), отличного от System.Object, то необходимо указать имя базового класса после ключевого слова "extends". При этом, если в качестве базового класса выбран System.ValueType, то объявляемый тип будет типом-значением.
- Если объявляемый тип реализует методы каких-либо интерфейсов, то должен быть приведен список этих интерфейсов после ключевого слова "implements".

Примеры

- Объявление экспортируемого абстрактного класса, реализующего интерфейс IEnumerable.

```
.class public abstract MyAbstractClass
  extends [mscorlib]System.Object
      implements
        [mscorlib]System.Collections.IEnumerable
      { }
```

- Объявление неэкспортируемого интерфейса.

```
.class private interface MyInterface { }
```

- Объявление экспортируемого типа-значения.

```
.class public sealed MyValueType
    extends [mscorlib]System.ValueType
    { }
```

12.2.3. Поля

- Поля объявляются внутри объявлений типов. Объявление поля осуществляется с помощью директивы ".field" и состоит из трех частей:
  - последовательность атрибутов поля;
  - тип;
  - имя поля.

Атрибуты полей

- Последовательность атрибутов следует непосредственно после ключевого слова ".field".

Атрибут	Описание
assembly	Поле видимо внутри сборки.
family	Поле видимо для наследников типа.
public	Поле видимо для всех.
private	Поле видимо только внутри типа.
static	Поле является статическим.

Примеры

- Объявление поля x типа массив.

```
.field private int32[] x
```

- Объявление поля table типа Hashtable.

```
.field public class
        [mscorlib]System.Collections.Hashtable table
```

12.2.4. Методы

- Методы объявляются внутри объявлений типов. Объявление метода осуществляется с помощью директивы ".method" и состоит из пяти частей:
  - последовательность атрибутов метода;
  - тип возвращаемого значения;
  - имя метода;
  - список параметров метода;
  - тело метода.

Атрибуты методов

- Последовательность атрибутов следует непосредственно после ключевого слова ".method".

Атрибут	Описание
assembly	Метод видим внутри сборки.
family	Метод видим для наследников типа.
public	Метод видим для всех.
private	Метод видим только внутри типа.
abstract	Метод является абстрактным.
virtual	Метод является виртуальным.
final	Метод не может переопределяться в наследниках.
static	Метод является статическим.

Сигнатура метода

- После атрибутов следует тип возвращаемого значения и идентификатор, задающий имя метода.
- Если метод не возвращает значения, в качестве типа возвращаемого значения указывается void.
- Конструкторы всегда имеют имя ".ctor", а статические конструкторы – ".cctor".
- Список параметров метода следует за именем метода и заключается в круглые скобки. Для каждого параметра указывается его тип и имя.

Примеры

- Объявление конструктора с двумя параметрами.

```
.method public void .ctor
        (int32 x, class [mscorlib]System.String s)
```

- Виртуальный метод с управляемым указателем в качестве параметра.

```
.method private virtual int32 myMethod(int32& pX)
```

- Статический метод, возвращающий массив:

```
.method public static int32[] MyStaticMethod()
```

Тело метода

- Тело метода заключается в фигурные скобки и содержит инструкции языка CIL.
- Каждая инструкция записывается на новой строке программы.
- Если нужно, то инструкции может предшествовать метка, отделяемая от инструкции двоеточием.
- Например:

```
Hello: ldstr "Hello, World!"
call void[mscorlib]System.Console.WriteLine(string)
```

Директивы тела метода

- Кроме инструкций CIL тело метода может содержать директивы тела метода.

Директива	Описание
`.entrypoint`	Показывает, что данный метод является точкой входа в сборку (метод должен быть статическим, возвращать int32 или ничего не возвращать, иметь в качестве параметров массив строк или вообще не иметь параметров).
`.locals (объявления)`	Определяет набор локальных переменных метода. Локальные переменные объявляются аналогично параметрам метода.
`.maxstack число`	Задает глубину стека вычислений.