# 一、.NET的体系结构

## 1.JIT编译器，提高性能

JIT编辑器并不是编译整个程序，而是只编译运行部分的代码。

## 2.IL，.net编译语言

IL可以指Intermediate Language，同MSIL(Microsoft Intermediate Language),是将.NET代码转化为机器语言的一个中间语言的缩写。

使用IL不仅支持平台无关性，还支持语言的互操作性。

互操作性是指能将包括C#的任何语言编译成中间语言，如VB，VC++

### 主要特征

a)面向对象和使用接口

b)值类型和引用类型之间的显著差异：

引用类型的实例存储在“托管堆”中，而值类型存储在堆栈中（如果值类型在引用类型中声明为字段，它们内联存储在堆中）

c)强数据类型化：

通用类型系统(CTS)、公共语言规范(CLS)

垃圾回收：.net运行库采用了垃圾回收器，其工作原理是：检查对存储在托管堆上的对象的引用，无引用时将其回收。

进程不能直接访问物理内存，因此一个进程不可能访问分配给另一个进程的内存，也就是一个进程出错不会影响另一个进程。

每个应用程序域大致对应一个应用程序，不同的可执行文件都运行在同一个进程中，数据是可以共享的。

d)使用异常来处理错误

e)使用特性(attribute)

## 3.语言的互操作性

语言互操作性的真正含义是：用一种语言编写的类应能直接与用另一种语言编写的类通信。

1. 用一种语言编写的类应能继承用另一种语言编写的类
2. 一个类应能包含另一个类的实例，而不管两个类是使用什么语言编写的
3. 一个对象应能直接调用用其他语言编写的另一个对象的方法
4. 对象（或对象的引用）应能在方法之间传递
5. 在不同的语言之间调用方法时，应能在调试器中交替调试这些方法调用，即调试不同语言编写的源代码

.NET和中间语言已经实现了这个目标，VS IDE提供了这样的工具

公共语言规范和通用类型系统一起确保语言的互操作性

### 1）VB

VB.NET与C#之间的区别很小，主要是使用花括号还是END语句的个人喜好问题

### 2）VC++

VC++.NET要在.NET framework中运行，只需要在代码开头添加以下命令即可

#using <mscorlib.dll>

### 3）VF#

VF#是一门函数语言，而不是面向对象，有助于用来编写复杂的算法。

### 4）COM和COM+

### 5）Windows运行库

这个运行库可以在VB、C#、C++和JS中使用。它的多样性是通过使用语言投影实现的。

## 4.数据类型

### 1）值类型与引用类型

如果要把自己的类型定义为值类型，就应该把它声明为一个结构。

### 2）预定义类型

整型：sbyte、short、int、long、byte、ushort、uint、ulong

所有的整型都能被赋予十进制或者十六进制的值，后者需要0x前缀

若没有显式的声明，则默认为int，当然也可以直接指定值，如long l=1234L，ulong ul=1234UL。建议使用大写的U和L，否则小写l容易与数字1混淆。

浮点型：double和float，指定float值一般后面加F（f）

decimal：指定值加M

char：除了带单引号的字符字面量外，还可用4位十六进制的Unicode值，带有数据类型转换的整数值或者十六进制数，还可用转义序列

string：字符串中允许包含换行符，在代码中一个字符串可以分两行写，显示的时候也会出现换行

## 5.C#的其他基础

**goto**语句可以跳转到指定行，但不能跳入像for这样的代码块中，也不能跳出try-catch后面的finially。一般不允许使用goto

**Enum.Parse**可以将字符串转化为枚举，通常三个参数：第一个是typeof获得的枚举类型，第二个是字符串，第三个表示是否不区分大小写

**using**可以给命名空间取别名，using 别名=命名空间。当使用别名时，调用不再使用.，而是::出类

**多个Main方法**：csc DoubleMain.cs时会报错，因为分辨不出哪个Main方法为程序入口点。csc DoubleMain.cs /main:Wrox.MathExample即可指定哪个类的Main方法为入口点，从而避免报错。

Main方法的参数可以通过命令行来传送选项

**占位符{n}**：还可以调整文本宽度，{n,w}其中w就是文本宽度。另外在宽度后面使用某种格式字符串，即可实现可选精度

**global关键字**：如果直接写命名空间点出类，那么程序会就近选择匹配的命名空间，而若是global::命名空间点出类，那么程序将从全局开始搜索匹配的命名空间。

**内部注释**：即/\*\*/可以用在某句代码内部。

**预处理指令**：#define（定义类似于VS点开项目属性中的条件编译符号）、if#elif#endif、#warning编译到此句会提示但继续编译、#error编译到此句将抛出异常、#region#endregion、#line、#pragma

**标识符**：如果需要把一个保留字用作标识符，那么可以在标识符的前面加上前缀符号@。标识符也可以包含unicode字符，用语法\uXXXX来指定，如\_test与\u005ftest可以互换，因为\_的Unicode代码就是005f

## 6.csc命令

暂时没有看懂

## 7.C#的面向对象

### 1）类与结构

类是存在堆中的引用类型，而结构是存在栈上的值类型。

类中有个和构造函数差不多的东西，成为终结器，其名称也与类名一致，只是前面多了个~，当有某个对象不需要时调用它。

#### A）数据成员

按引用传递效率更高，因为在按值传递时，必须复制大量的数据。

#### B）函数成员

命名参数：void test(string a，string b)参时可以test(b=””,a=””）

属性的访问性，暂时看不懂。

C#可以给类编写无参数的静态构造函数。只执行一次。而且无参数的实例构造函数和静态构造函数可以在同一个类中同时定义！

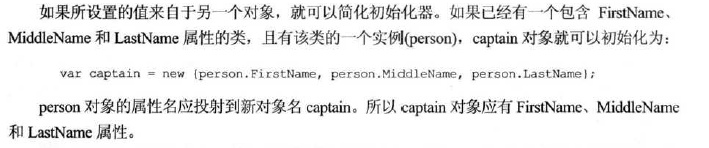
#### C）只读字段

readonly比const灵活，其规则是只可以在构造函数中给只读字段赋值，且不一定要是静态字段。

### 2）匿名类型

var不止能用于弱类型，还能用作隐式类型化的变量。

没用过，有点难理解，需要用一下才知道。



### 3）结构

结构是值类型，不是引用类型。存储在栈中或者存储为内联（如果它们是存储在堆中的另一个对象的一部分。内联函数是指用inline关键字修饰的函数。在类内定义的函数被默认成内联函数。暂时理解为，在类中定义的结构）。

结构的性能与类相比，有正面也有负面，正面是分配内存与释放内存都比较快，负面是传递时是整个复制，而不是指传递引用。当然，结构作为参数传递时，使用ref，也是传递引用

定义结构对象，可以new，也可以不new。因为这里的new仅仅是调用构造函数，不分配堆内存。

结构不允许自定义无参数的构造方法，且结构中实例字段不能初始化

### 4）弱引用

弱引用需要使用WeakReference类来创建，并且需要先判断对象是否存在（属性IsAlive）。

### 5）部分类partial

### 6）Object类

扩展方法：需要先创建一个静态类和一个静态方法。此静态方法要作扩展方法，第一个参数是要扩展的类型，放在this关键字的后面。

如果类中已经有与扩展方法同名的方法，则实例方法优先调用

## 8.继承

结构不能实现继承，但可以多重接口继承。

virtual虚方法：重写只对类中的实例函数成员有意义。

隐藏方法：派生类和基类中有同签名的方法，但不是虚方法，那么基类的此方法会被隐藏。如果需要使派生类的此方法隐藏，则需要在派生类的此方法的定义前加上new关键字。

抽象类（abstract）：不能实例化、不能有方法实现、必须在派生类中实现方法。

密封类（sealed）：不能被继承。

