const 与 readonly知多少

尽管你写了很多年的C#的代码，但是可能当别人问到你const与readonly的区别时候，还是会小小的愣一会吧~

笔者也是在看欧立奇版的《.Net 程序员面试宝典》的时候，才发现自己长久以来竟然在弄不清出两者的情况下，混用了这么长的时间。的确，const与readonly 很像，都是将变量声明为只读，且在变量初始化后就不可改写。那么，const与readonly 这两个修饰符到底区别在什么地方呢？其实，这个牵扯出C#语言中两种不同的常量类型：静态常量（compile-time constants）和动态常量（runtime constants）。这两者具有不同的特性，错误的使用不仅会损失效率，而且还会造成错误。

首先先解释下什么是静态常量以及什么是动态常量。静态常量是指编译器在编译时候会对常量进行解析，并将常量的值替换成初始化的那个值。而动态常量的值则是在运行的那一刻才获得的，编译器编译期间将其标示为只读常量，而不用常量的值代替，这样动态常量不必在声明的时候就初始化，而可以延迟到构造函数中初始化。

当你大致了解上面的两个概念的时候，那么就可以来说明const与readonly了。const修饰的常量是上述中的第一种，即静态常量;而readonly则是第二种，即动态常量。那么区别可以通过静态常量与动态常量的特性来说明：

1）const修饰的常量在声明的时候必须初始化;readonly修饰的常量则可以延迟到构造函数初始化

2）const修饰的常量在编译期间就被解析，即常量值被替换成初始化的值;readonly修饰的常量则延迟到运行的时候

此外const常量既可以声明在类中也可以在函数体内，但是static readonly常量只能声明在类中。

可能通过上述纯概念性的讲解，对有些初学者有些晕乎。下面就一些例子来说明下：

```
using System;
class P
{
    static readonly int A=B*10;
    static readonly int B=10;
    public static void Main(string[] args)
    {
        Console.WriteLine("A is {0},B is {1} ",A,B);
    }
}
```

对于上述代码，输出结果是多少?很多人会认为是A is 100,B is 10吧！其实，正确的输出结果是A is 0,B is 10。好吧，如果改成下面的话：

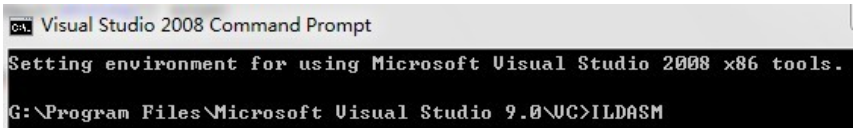
```
using System;
class P
{
    const int A=B*10;
```

--慢慢摸索

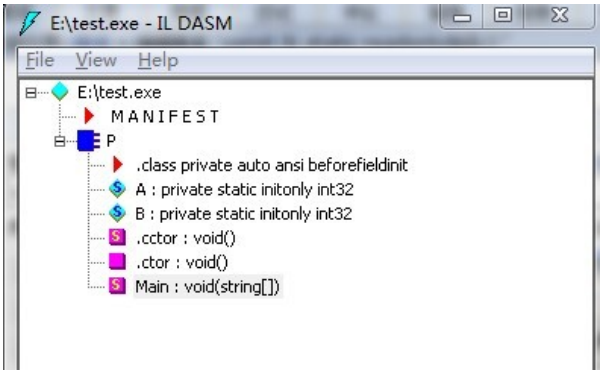
```
    const int B=10;
    public static void Main(string[] args)
    {
        Console.WriteLine("A is {0},B is {1} ",A,B);
    }
}
```

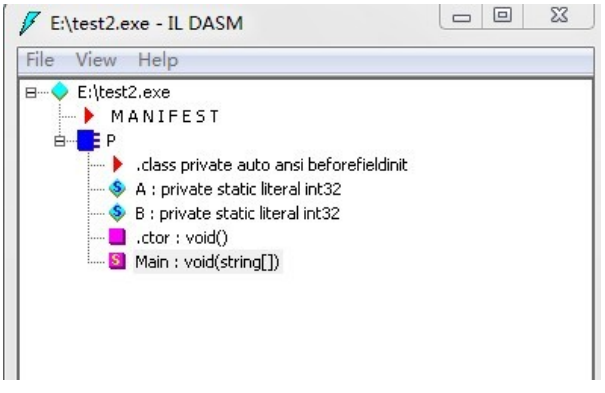
对于上述代码，输出结果又是多少呢？难道是A is 0,B is 10？其实又错了，这次正确的输出结果是A is 100,B is 10。

那么为什么是这样的呢？其实在上面说了，const是静态常量，所以在编译的时候就将A与B的值确定下来了（即B变量时10，而A=B*10=10*10=100）,那么Main函数中的输出当然是A is 100,B is 10啦。而static readonly则是动态常量，变量的值在编译期间不予以解析，所以开始都是默认值，像A与B都是int类型，故都是0。而在程序执行到A=B*10;所以A=0*10=0,程序接着执行到B=10这句时候，才会真正的B的初值10赋给B。如果，你还是不大清楚的话，我们可以借助于微软提供的ILDASM工具，只需在Vs 2008 Command下输入ILDASM就可以打开，如下所示：



分别打开上述两个代码编译后产生的可执行文件，如下图所示：

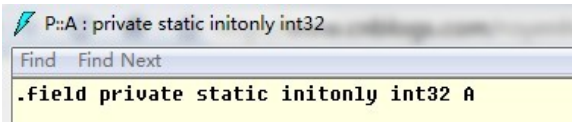


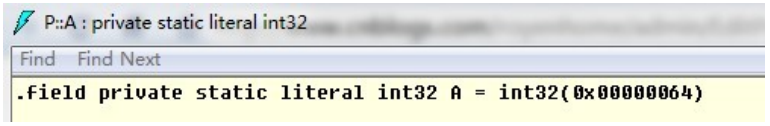


static readonly可执行程序的结构

const可执行程序的结构

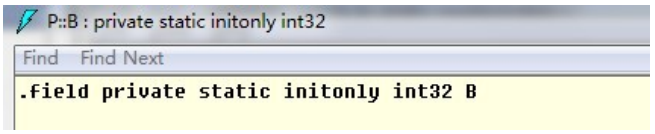
在上述两张图中都可以看到A与B常量，分别双击节点可以看出其中的差异：

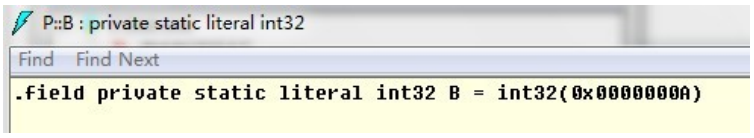




static readonly修饰的常量A

const修饰的常量A

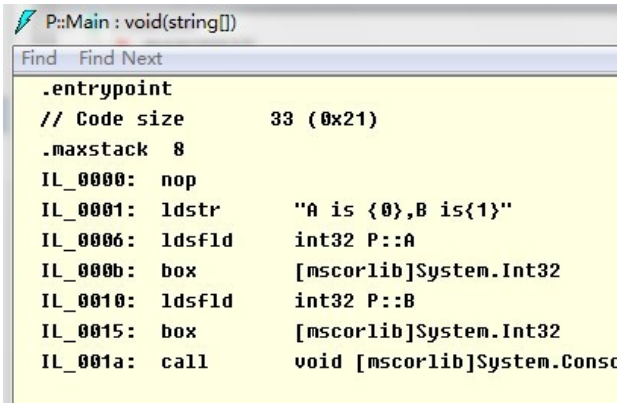


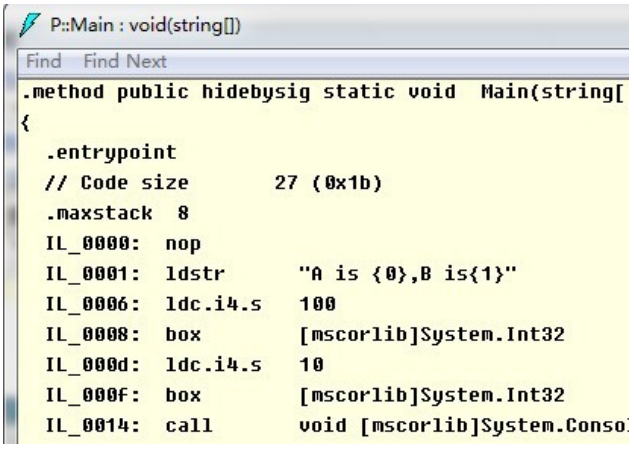


static readonly修饰的常量B

const修饰的常量B

从上图中可以看出，const修饰的常量在编译期间便已将A,B的字面值算出来了，而static readonly修饰的常量则未解析，所以在Main函数中有以下的区别：





static readonly程序的Main函数

const程序的Main函数

从Main函数中我们可以看出，const的那个程序的输出直接是100与10，而readonly在输出的时候确实P::A与P::B，即将A与B常量的值延迟到运行的时候才去确定，故输出是0与10。

那么对于静态常量以及动态常量还有什么特性呢?其实，静态常量只能被声明为简单的数据类型（int以及浮点型）、枚举、布尔或者字符串型，而动态常量则除了这些类型，还可以修饰一些对象类型。如DateTime类型，如下：

```
//错误
const DateTime time=new DateTime();
//正确
static readonly DateTime time=new DateTime();
```

上述错误在于不能使用new关键字初始化一个静态常量，即便是一个值类型，因为new将会导致到运行时才能确定值，与静态变量编译时就确定字面值有悖。

欧书上最后给出了对静态常量与动态常量之间的比较，如下表所示：

	静态常量	动态常量
内存消耗	无	因需保存常量，故有消耗
初始化	很少的简单类型，不能new ,必须在声明时候初始化	任意类型，可以在构造函数中初始化
何时发挥作用	编译时候进行替换	相当于类中的数据成员

最近随笔较少，希望这篇能给一些朋友带来一些帮助~

分类: [1].Net

royen
关注 - 9
粉丝 - 161
+加关注

38 0

« 上一篇：贴上前天柯达校园招聘笔试的题目
» 下一篇：Perl篇：获取操作系统的信息

posted @ 2010-05-22 16:45 royen 阅读(17282) 评论(47) 编辑 收藏

评论列表

#1楼 2010-05-22 17:22 王一一

用简单的方法把问题讲清楚是最好的。

支持(1) 反对(0)

#2楼[楼主] 2010-05-22 17:38 royen

@ 王一一
我说复杂了？

支持(0) 反对(0)

#3楼 2010-05-22 17:46 王一一

呵呵。没有。你用的就是简单的方法。还+1了

支持(0) 反对(0)

#4楼[楼主] 2010-05-22 17:55 royen

@ 王一一
呵呵，谢谢支持~等实习单位找好了，会给园子奉献更多更好的文章的~

支持(0) 反对(0)

#5楼 2010-05-22 17:56 说不得

常量貌似不是编译的时候确定值，而是编译的时候以值去替换所有常量。

比如：

引用

```
const int I = 100;
int i = 10;
```

```
int j = I * i;
```

编译结束之后使用Reflector或者IL工具查看代码，是没有I这个常量的，所有使用I这个常量的地方，都被替换成了值100。

基于以上原因，所有public修饰的变量，都不应该使用const修饰，因为，如果外部类引用了这个变量（常量），编译的时候，编译器会自动将该常量替换为值。后果就是，如果改变了该常量的值，就必须重新编译所有引用该常量的代码。

场景：该常量放在一个dll里，现在值变了，就必须更新所有引用了该dll的文件，而不是替换该dll就可以。

支持(0) 反对(0)

#6楼 2010-05-22 18:08 小AI

readonly为运行时常量，const为编译时常量
编译时常量被运行时常量快，性能好，但是缺乏灵活性（编译时常量需要重新编译应用程序）；
编译时常量（const）仅限于数值和字符串（基元类型），C#不允许使用new来初始化一个编译时常量；
const修饰的常量默认是静态的（类型）；
readonly修饰的字段可以在构造函数中被修改；
使用const较之使用readonly的唯一好处就是性能

http://www.cnblogs.com/heaiping/archive/2010/04/11/1709507.html

支持(0) 反对(0)

#7楼[楼主] 2010-05-22 18:13 royen

@ 说不得
首先需要明确的是，你这儿讨论的是静态常量的问题，而对于静态常量而言，准确的说法是在编译的时候将其替换成所对应的字面值。但是，你说通过const int I;定义的常量，在IL工具中看不到I这个常量，请看文中给出的截图，是存在这个常量的，不信你可以试下！你最后举得例子很好，前提是程序作为单个dll编译出来给其他模块用的时候，如果引用到了这个dll的const常量，需要重新编译dll并更新应用，但不是解决问题的关键~对于这种情况，则需要使用动态常量解决：
场景：static readonly修饰的常量放在dll中，现在常量改变了，不需更新该dll，只需将dll重新编译即可~

支持(0) 反对(0)

#8楼[楼主] 2010-05-22 18:22 royen

引用
Alex He：
readonly为运行时常量，const为编译时常量
编译时常量被运行时常量快，性能好，但是缺乏灵活性（编译时常量需要重新编译应用程序）；
编译时常量（const）仅限于数值和字符串（基元类型），C#不允许使用new来初始化一个编译时常量；
const修饰的常量默认是静态的（类型）；
readonly修饰的字段可以在构造函数中被修改；
使用const较之使用readonly的唯一好处就是性能

http://www.cnblogs.com/heaiping/archive/2010/04/11/1709507.html

归纳的很精辟哈~（编译时常量需要重新编译应用程序）这句你的意思应该是指编译后如果有对常量有引用的模块需更新引用吧~

支持(0) 反对(0)

#9楼 2010-05-22 18:51 小AI

引用
royen：
引用
Alex He：
readonly为运行时常量，const为编译时常量
编译时常量被运行时常量快，性能好，但是缺乏灵活性（编译时常量需要重新编译应用程序）；
编译时常量（const）仅限于数值和字符串（基元类型），C#不允许使用new来初始化一个编译时常量；
const修饰的常量默认是静态的（类型）；
readonly修饰的字段可以在构造函数中被修改；
使用const较之使用readonly的唯一好处就是性能

http://www.cnblogs.com/heaiping/archive/2010/04/11/1709507.html

归纳的很精辟哈~（编译时常量需要重新编译应用程序）这句你的意思应该是指编译后如果有对常量有引用的模块需更新引用吧~

就是大家得保持一致啊

支持(0) 反对(0)

#10楼 2010-05-22 19:14 W.SiMin

据我所理解，一个是分配内存，另一个不分配那样吧

支持(0) 反对(0)

#11楼[楼主] 2010-05-22 19:53 royen

呵呵，说的很通俗啊～

支持(0) 反对(0)

#12楼[楼主] 2010-05-22 20:01 royen

额，不是，都分配了内存，要说主要区别应该是在什么时候用字面值来替换常量的值

支持(0) 反对(0)

#13楼 2010-05-22 20:04 fyljf

讲的很清晰，学习了

支持(0) 反对(0)

#14楼[楼主] 2010-05-22 20:22 royen

呵呵，谢谢～

支持(0) 反对(0)

#15楼 2010-05-22 21:39 McJeremy&Fan

const是针对常量而言.
static readonly 是对变量而言.

支持(0) 反对(0)

#16楼[楼主] 2010-05-22 22:02 royen

@ McJeremy&Fan
不能苟同你的观点，两者都是针对常量吧~

支持(0) 反对(0)

#17楼 2010-05-22 23:18 skyaspnet

您好，我也在找实习，看你的闪存很熟悉，你也是软件学院的研究生吧？是在哪个学校呢？我也刚参加了腾讯的笔试，没考过

支持(0) 反对(0)

#18楼[楼主] 2010-05-22 23:24 royen

@ skyaspnet
额，我是南京东南大学软院大三的学生~当时腾讯临时来学院招聘，啥招聘信息没有，去了才知道找召测试人员，还一堆测试题目，悲剧的很啊~

支持(0) 反对(0)

#19楼 2010-05-22 23:28 skyaspnet

呵呵，我是中科大软件学院的在读研一学生，当时腾讯是去东南大学软件学院苏州校区，考得比较差，我兴趣是ASP.NET和WEB前端，希望多交流，刚才转载了您这篇文章，很不错

支持(0) 反对(0)

#20楼[楼主] 2010-05-23 00:01 royen

呵呵，其实我对ASP.NET以及3富互联网应用技术都感兴趣，由于时间关系，所以这方面写的比较少，很高兴与你学习交流～

支持(0) 反对(0)

#21楼 2010-05-23 00:19 W.SiMin

引用

royen：额，不是，都分配了内存，要说主要区别应该是在什么时候用字面值来替换常量的值

表格不是说了静态常量不消耗内存，那么何来分配内存呢？

支持(0) 反对(0)

#22楼[楼主] 2010-05-23 00:26 royen

表格中比较的是需不需要额外的内存空间保存那个常量值，因为动态常量在运行时候才被替换，所以常量初始化的值必须耗费额外的内存空间保存。而不是指常量本身占不占内存。

支持(0) 反对(0)

#23楼 2010-05-23 14:18 万雅虎

应该很简单的东西喔. 一个是编译时的东西.一个是编译后的东西.
就好比#if DEUNG 一样.
比如配置 就得用 static readONLY . 比如 数据库链接什么的.

支持(0) 反对(0)

#24楼[楼主] 2010-05-23 17:13 royen

呵呵，是基础的一些知识，只不过自己当初没学的那么细，所以现在总结下，希望对不知道的朋友有所帮助～

支持(0) 反对(0)

#25楼 2010-05-24 01:39 WizardWu

great

支持(0) 反对(0)

#26楼[楼主] 2010-05-24 07:41 royen

@ WizardWu
Thank you~

支持(0) 反对(0)

#27楼 2010-05-24 08:50 小 小 鸟

+1

支持(0) 反对(0)

#28楼[楼主] 2010-05-24 09:03 royen

@ 小 小 鸟
thank you~

支持(0) 反对(0)

#29楼 2010-05-24 09:24 自由泳

good

支持(0) 反对(0)

#30楼[楼主] 2010-05-24 09:57 royen

@ 自由泳
谢谢~

支持(0) 反对(0)

#31楼 2010-05-24 12:48 Kenny-Chen

学习了,希望有更好的文章出现

支持(0) 反对(0)

#32楼[楼主] 2010-05-24 13:05 royen

@ cycool
呵呵，会有的~

支持(0) 反对(0)

#33楼 2010-07-21 17:53 轩脉刃

学习了~~

支持(0) 反对(0)

#34楼[楼主] 2010-07-21 20:32 royen

@ 轩脉刃
呵呵，大家一起学嘛~

支持(0) 反对(0)

#35楼 2011-10-22 09:52 hutun

非常感谢!
我以前做C/C++时，常把常量数组生命为const，
如：const long array[] = {...};
这样编译器会把这部分数据存放在rom中。
但是在C#中不行，编译时通不过。
如你上面说描述的，readonly 会占用Ram，这样做在在嵌入式开发是很不可取的。
请教一下C#中如何能实现如C/C++把常来数组数据存放到Rom中？

支持(0) 反对(0)

#36楼 2012-06-20 17:30 初雪之恋

很好 - -

支持(0) 反对(0)

#37楼 2012-09-19 17:39 空葫芦

"const是静态常量，所以在编译的时候就将A与B的值确定下来了（即B变量时10，而A=B*10=10*10=100）"

A = B * 10;
B = 10;

B*10是在何时计算并赋值给A的呢

支持(0) 反对(0)

#38楼 2012-12-15 11:45 张承

这文章必须得转 太经典了 哥搞了几年的编程终于弄明白了

支持(0) 反对(0)

#39楼 2014-02-08 13:21 骑士归来

学习了，谢谢分享

支持(0) 反对(0)

#40楼 2014-06-06 10:41 ChuckLu

```
//readonly变量，怎么还可以在方法中进行操作呀？
class ChuckReadOnly<TKey,TValue>
{
private readonly Dictionary<TKey, TValue> dictinonary = new Dictionary<TKey, TValue>
();
public void DoTask(TKey key,TValue value)
{
dictinonary.Add(key,value);
}

}

class Program
{
static void Main(string[] args)
{
ChuckReadOnly<string, string> hello = new ChuckReadOnly<string, string>();
hello.DoTask("天","地");
Console.ReadLine();
}
```

支持(0) 反对(0)

#41楼 2014-06-10 09:45 Vampire_D

学习了。。。

支持(0) 反对(0)

#42楼 2014-11-11 16:41 _____Sam

彬哥脸一:"这玩意我都知道啊！"

支持(0) 反对(0)

#43楼 2014-11-11 16:41 _____Sam

彬哥脸一黑:"这玩意我都知道啊！"

支持(0) 反对(0)

#44楼 2015-05-20 09:15 ChuckLu

谷歌了一下
http://stackoverflow.com/questions/55984/what-is-the-difference-between-const-and-readonly

还有这个区别

支持(0) 反对(0)

#45楼 2015-05-22 10:44 ChuckLu

40楼问的问题，我找到答案了
只读，针对的是变量的引用。引用不能被改变[构造函数中可以改]

支持(0) 反对(0)

#46楼 2015-11-11 09:55 谷仁儿

学习了，谢过楼主！

支持(0) 反对(0)

#47楼 2016-07-22 14:51 别等时光非礼了梦想

写的很好，支持

支持(0) 反对(0)

刷新评论 刷新页面 返回顶部

注册用户登录后才能发表评论，请 登录 或 注册，访问网站首页。