C# 字符串（String） C# 枚举（Enum）

C# 结构体（Struct）

在 C# 中，结构体是值类型数据结构。它使得一个单一变量可以存储各种数据类型的相关数据。**struct** 关键字用于创建结构体。

结构体是用来代表一个记录。假设您想跟踪图书馆中书的动态。您可能想跟踪每本书的以下属性：

- Title
- Author
- Subject
- Book ID

定义结构体

为了定义一个结构体，您必须使用 struct 语句。struct 语句为程序定义了一个带有多个成员的新的数据类型。

例如，您可以按照如下的方式声明 Book 结构：

```
struct Books
{
   public string title;
   public string author;
   public string subject;
   public int book_id;
};
```

下面的程序演示了结构的用法：

实例

```
using System;
using System.Text;

struct Books
{
   public string title;
   public string author;
   public string subject;
   public int book_id;
};

public class testStructure
{
   public static void Main(string[] args)
   {
```

```
      Books Book1;        /* 声明 Book1，类型为 Book */
      Books Book2;        /* 声明 Book2，类型为 Book */

      /* book 1 详述 */
      Book1.title = "C Programming";
      Book1.author = "Nuha Ali";
      Book1.subject = "C Programming Tutorial";
      Book1.book_id = 6495407;

      /* book 2 详述 */
      Book2.title = "Telecom Billing";
      Book2.author = "Zara Ali";
      Book2.subject =  "Telecom Billing Tutorial";
      Book2.book_id = 6495700;

      /* 打印 Book1 信息 */
      Console.WriteLine( "Book 1 title : {0}", Book1.title);
      Console.WriteLine("Book 1 author : {0}", Book1.author);
      Console.WriteLine("Book 1 subject : {0}", Book1.subject);
      Console.WriteLine("Book 1 book_id :{0}", Book1.book_id);

      /* 打印 Book2 信息 */
      Console.WriteLine("Book 2 title : {0}", Book2.title);
      Console.WriteLine("Book 2 author : {0}", Book2.author);
      Console.WriteLine("Book 2 subject : {0}", Book2.subject);
      Console.WriteLine("Book 2 book_id : {0}", Book2.book_id);

      Console.ReadKey();

   }
}
```

当上面的代码被编译和执行时，它会产生下列结果：

```
Book 1 title : C Programming
Book 1 author : Nuha Ali
Book 1 subject : C Programming Tutorial
Book 1 book_id : 6495407
Book 2 title : Telecom Billing
Book 2 author : Zara Ali
Book 2 subject : Telecom Billing Tutorial
Book 2 book_id : 6495700
```

C# 结构的特点

您已经用了一个简单的名为 Books 的结构。在 C# 中的结构与传统的 C 或 C++ 中的结构不同。C# 中的结构有以下特点：

- 结构可带有方法、字段、索引、属性、运算符方法和事件。
- 结构可定义构造函数，但不能定义析构函数。但是，您不能为结构定义默认的构造函数。默认的构造函数是自动定义的，且不能被改变。

- 与类不同，结构不能继承其他的结构或类。
- 结构不能作为其他结构或类的基础结构。
- 结构可实现一个或多个接口。
- 结构成员不能指定为 abstract、virtual 或 protected。
- 当您使用 **New** 操作符创建一个结构对象时，会调用适当的构造函数来创建结构。与类不同，结构可以不使用 New 操作符即可被实例化。
- 如果不使用 New 操作符，只有在所有的字段都被初始化之后，字段才被赋值，对象才被使用。

类 vs 结构

类和结构有以下几个基本的不同点：

- 类是引用类型，结构是值类型。
- 结构不支持继承。
- 结构不能声明默认的构造函数。

针对上述讨论，让我们重写前面的实例：

实例

```
using System;
using System.Text;

struct Books
{
   private string title;
   private string author;
   private string subject;
   private int book_id;
   public void getValues(string t, string a, string s, int id)
   {
      title = t;
      author = a;
      subject = s;
      book_id =id;
   }
   public void display()
   {
      Console.WriteLine("Title : {0}", title);
      Console.WriteLine("Author : {0}", author);
      Console.WriteLine("Subject : {0}", subject);
      Console.WriteLine("Book_id :{0}", book_id);
   }

};

public class testStructure
{
   public static void Main(string[] args)
```

```
    {

        Books Book1 = new Books(); /* 声明 Book1，类型为 Book */
        Books Book2 = new Books(); /* 声明 Book2，类型为 Book */

        /* book 1 详述 */
        Book1.getValues("C Programming",
        "Nuha Ali", "C Programming Tutorial",6495407);

        /* book 2 详述 */
        Book2.getValues("Telecom Billing",
        "Zara Ali", "Telecom Billing Tutorial", 6495700);

        /* 打印 Book1 信息 */
        Book1.display();

        /* 打印 Book2 信息 */
        Book2.display();

        Console.ReadKey();

    }
}
```

当上面的代码被编译和执行时，它会产生下列结果：

```
Title : C Programming
Author : Nuha Ali
Subject : C Programming Tutorial
Book_id : 6495407
Title : Telecom Billing
Author : Zara Ali
Subject : Telecom Billing Tutorial
Book_id : 6495700
```

← C# 字符串（ String ）　　C# 枚举（ Enum ） →

2 篇笔记

写笔记

补充：类与结构体的区别

1、结构体中声明的字段无法赋予初值，类可以:

```
struct test001
{
    private int aa = 1;
}
```

执行以上代码将出现“**结构中不能实例属性或字段初始值设定**”的报错，而类中无此限制，代码如下：

```
class test002
{
    private int aa = 1;
}
```

2、结构体的构造函数中，必须为结构体所有字段赋值，类的构造函数无此限制:

补充：类与结构的选择

首先明确，类的对象是存储在堆空间中，结构存储在栈中。堆空间大，但访问速度较慢，栈空间小，访问速度相对更快。故而，当我们描述一个轻量级对象的时候，结构可提高效率，成本更低。当然，这也得从需求出发，假如我们在传值的时候希望传递的是对象的引用地址而不是对象的拷贝，就应该使用类了。

轩哥 11个月前 (04-17)

C# 中结构类型和类类型在语法上非常相似，他们都是一种数据结构，都可以包括数据成员和方法成员。

结构和类的区别：

- 1、结构是值类型，它在栈中分配空间；而类是引用类型，它在堆中分配空间，栈中保存的只是引用。
- 2、结构类型直接存储成员数据，让其他类的数据位于对中，位于栈中的变量保存的是指向堆中数据对象的引用。

  C# 中的简单类型，如int、double、bool等都是结构类型。如果需要的话，甚至可以使用结构类型结合运算符运算重载，再为 C# 语言创建出一种新的值类型来。

  由于结构是值类型，并且直接存储数据，因此在一个对象的主要成员为数据且数据量不大的情况下，使用结构会带来更好的性能。

  因为结构是值类型，因此在为结构分配内存，或者当结构超出了作用域被删除时，性能会非常好，因为他们将内联或者保存在堆栈中。当把一个结构类型的变量赋值给另一个结构时，对性能的影响取决于结构的大小，如果结构的数据成员非常多而且复杂，就会造成损失，接下来使用一段代码来说明这个问题。

  结构和类的适用场合分析：
- 1、当堆栈的空间很有限，且有大量的逻辑对象时，创建类要比创建结构好一些；
- 2、对于点、矩形和颜色这样的轻量对象，假如要声明一个含有许多个颜色对象的数组，则CLR需要为每个对象分配内存，在这种情况下，使用结构的成本较低；
- 3、在表现抽象和多级别的对象层次时，类是最好的选择，因为结构不支持继承。
- 4、大多数情况下，目标类型只是含有一些数据，或者以数据为主。

King 9个月前 (06-08)