



- **单元2 - C#语言基础编程：**在.NET平台上固然有数个编程语言可选，在以往可能有些开发人员会选择Visual Basic（VB），但VB在面对Java编程语言的攻势，显得有点力不从心。同时，微软曾多次企图挖角Anders Hejlsberg，1996年，Anders Hejlsberg终于加入微软，此后开发展出C#编程语言，以作为与Java对抗的主要语言。至此，各位学习的朋友应可体会出风向了。其实对开发人员也算好事吧，因为不管Java还是C#，都算是C/C++家族的一员了，而开发人员在这些编程语言里绕的弯可能会小一些。所以，打开心扉，我们来学C#编程语言吧！



## 1. C#简介及应用程序类型

LearnASP.NET-rev1.1 <http://www.coding24.org>

- **C#简介及应用程序类型：** 主要对C#编程语言作一些简单介绍，使开发人员略为熟悉这个编程言，能熟才能成为朋友！此外，也对安德斯·海尔斯伯格（Anders Hejlsberg）作些简介，他是这个编程语言的创始人。



## C#语言简介

- C#（读C Sharp）是微软公司在2001年发布的一种面向对象的编程语言、可运行于.NET Framework和.NET Core平台。
- 继承了C和C++强大功能的同时去掉了它们的一些复杂特性。
- 与Java语言高度相似，如在单一继承、接口、语言的语法和编译成中间代码再运行等。
- 在.NET Framework平台上，C#与VB（Visual Basic）共存；而在.NET Core平台上已成为主要编程语言。

LearnASP.NET-rev1.1 <http://www.coding24.org>

3

## C#语言简介

- C#（读作C Sharp）是微软公司在2001年发布的一种面向对象的编程语言、可运行于.NET Framework和.NET Core平台。
- 由微软公司研究员Anders Hejlsberg所开发出来的一种安全的、稳定的、简单的面向对象编程语言。继承了C和C++强大功能的同时去掉了一些它们的复杂特性。
- 与Java语言高度相似，如在单一继承、接口、语言的语法和编译成中间代码再运行等，均极为相似。
- 微软技术中，在.NET Framework时代，主要有C#与VB（Visual Basic）编程语言并立；在.NET Core时代，C#似乎已成为.NET平台主要的编程语言！也意味着C家族（C/C++/Java/C#）阵容愈加强大。



## 安德斯·海尔斯伯格 (Anders Hejlsberg)

安德斯·海尔斯伯格 (Anders Hejlsberg)，1960年12月出生于丹麦哥本哈根，曾在丹麦科技大学学习工程学（**但没有毕业**），计算机科学家。Turbo Pascal编译器的主要作者，Delphi、**C#**和TypeScript之父，**微软.NET技术**创立者。

主要成就：**发明了 Delphi、C# 两种著名编程语言**



LearnASP.NET-rev1.1 <http://www.coding24.org>

4

- 微软让我们最先想到的可能是比尔盖茨 (Bill Gates)，就读美国名校—哈佛大学，但没毕业。1996年微软力邀多年终于迎来了一位，对微软来说非常重要的人物—安德斯·海尔斯伯格，丹麦名校，但也是没毕业？！对微软来说，安德斯·海尔斯伯格发展了**C#**编程语言的重要性及角色，大约就相当于**Java**之父—詹姆斯·高斯林 (James Gosling) 了！不过加拿大人詹姆斯·高斯林在卡内基梅隆大学是有毕业的。



- 在本单元的C#编程实作或范例代码的编写，我采用建立ASP.NET Web Forms应用的方式去编写并测试代码；各位读者亦可建立Console类型应用，再编写代码的方式为之，差别在于前者代码大部份写在页面后置代码（.cs）的Page\_Load事件中，而后者为传统方式，大部份大码可能写在一个C#类的Main()方法中。



## 实作1- 在Web Form中编写C#

- 实作说明：在Web Forms类型项目中编写C#程序也可以尽快地理解Web的开发模式。在此模式中，我们大部部在页面的Page\_Load事件中编写代码并使用Response.Write()方法输出结果。
- 实作步骤：
  - 使用VS创建项目，类型：「ASP.NET Web应用程序(.NET Framework)」，项目/方案名为Lab-WebApp，框架：.NET Framework 4.8（选4.X亦可）
  - 在画面中选「空」，在添核心引用选「Web窗体」，取消「为HTTPS配置」
  - 在WebApp项目中添加「Web窗体」->命名为Hello.aspx
  - 在Hello后置代码中输入代码。
  - 运行程序：鼠标右击Hello.aspx-> 在浏览器中查看，运行结果如下。

LearnASP.NET-rev1.1 <http://www.coding24.org>

6

- 一个ASP.NET网页由两部份组成：
  - .aspx档：在教程中称页面或前端网页档，由HTML/CSS/JavaScript及ASP.NET标签组成。
  - .cs档：后置代码，为C#代码，主要由网页的事件程序组成。
- 进入后置代码编辑视窗的方式：
  - 在方案视窗中找到相应的.cs程序档。
  - 在页面设计画面中，鼠标右击->「查看代码」。
  - 在页面设计画面中，于空白处双击鼠标。
  - 在方案视窗中，鼠标点击「查看代码」图标。



- ASP.NET页面的3种运行方式:
  - 在方案视窗中，鼠标右击页面档名（.aspx）运行
  - 在代码编辑视窗中，前端页面（.aspx）上鼠标右击运行；或在设计画面上鼠标右击运行。
  - 在项目为「启动项目」（项目名为粗黑体字）时，打开欲运行的页面（.aspx）后，点击工具栏“IIS Express”运行。若在没打开页面下（会调用网站首页），需在运行后于浏览器输入网址。
- 前两种运行方式并不会重新生成web项目，因此若有更新相关资源（如添加了图档）或新增加class档并不会被编译并发布。但若只是更新.aspx档则前两种方式可行。
- Response.Write(): Response对象用于从服务器向用户发送信息，而Write()方法可以将传入的字符串参数输出到用户端的浏览器上。
- Hello.aspx.cs

```

using System;

namespace LearnASP.NET.Labs.Unit02
{
    public partial class Hello : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            //我的第1个C# Web Forms应用程序
            Response.Write("我的第1个C# Web Forms应用程序!");
        }
    }
}

```



## 2. C#程序及数据类型

LearnASP.NET-rev1.1 <http://www.coding24.org>

- 在后续小节中的C#代码示例将以Web Forms型式编写，各位读者若要编写成Console类型应用亦可。



## C# 程序结构 (Web Forms类型应用)

● 一个 C# 程序主要包括以下部分:

```

1  using System;
2
3
4  namespace Lab202_WebApp
5  {
6      public partial class Hello : System.Web.UI.Page
7      {
8          protected void Page_Load(object sender, EventArgs e)
9          {
10             //我的第1个C# Web Forms应用程序
11             Response.Write("我的第1个C# Web Forms应用程序!");
12          }
13      }
14
15  }

```

**Using:**命名空间的引用,在本例代码中可直接使用Console类,而不用全名.

**namespace:**命名空间的声明,后接{}, 在新版C#中将省略{}.

**class:**需有名称,若是继承, 需加: 父类名.

**method:**需有名称/参数/回传值型态等. Console项目的启动方法为Main(). 不同类型项目其启动方法不尽相同. (M是大写)

**注释:**善用注释可提高对代码的理解.

**代码:**在方法区块内写上你的C#语句 (Statement)以及表达式(Expression).若是成员变量或是另加方法则要写在class区块内.

←

←

←

←

←

←

9

### ● C#程序结构:

- Using: 命名空间的引用,在本例代码中可直接使用Console类,而不用全名.
- namespace: 命名空间的声明,后接{}, 在新版C#中将省略{}.
- class: 需有名称,若是继承, 需加: 父类名
- method: 需有名称/参数/回传值型态等. Console项目的启动方法为Main(). 不同类型项目其启动方法不尽相同. (M是大写)
- 注释: 善用注释可提高对代码的理解.
- 代码: 在方法区块内写上你的C#语句(Statement)以及表达式(Expression).若是成员变量或是另加方法则要写在class区块内.
- Partial class: C#的后置代码 (.cs) 与前端的页面 (.aspx) 在编译时会被整合成为一个class。
- Page类: 所有的Web Form页都继承自Page类, 透过继承获得了所需的基础功能。本例中的Hello类也是继承了Page类。
- Page\_Load事件: 是每一个Web Form表单一被载入就会执行的事件过程, 程序中代码须自订, 一般都是写些初值设置的代码。在本单元中大部份的C#代码都在这区块内。
- 单行注释: 使用符号//
- 多行注释: 仗用符号/\* .... \*/

### ● ASP.NET (C#)的命名规范 (非强制) 分为Pascal和Camel两种形式:

- Pascal形式将标识符的首字母大写和后面连接的每个单词的首字母都大写。如, 某个控件或类的属性名BackColor。

- Camel形式将标识符的首字母小写，而每个后面连接的单词的首字母都大写。如，变量名typeName。
- 控件其实是一个对象，是以Camel（小写开始）命名形式，其它如参数名，变量名，亦用此形。



## C# 的标识符和关键字

- 标识符 (Identifier) 是用来对类, 方法或变项的命名以便在程序中识别。C#标识符区别大小写, 且需以A-Z, a-z, \_ (下划线) 或数字组成, 但数字不能在开头。
- 关键字 (Keyword) 是保留标识符, 对编译器有特别意义。除非前面有 @ 前缀, 否则不能在程序中用作标识符。


abstract	as	base	bool	break	byte	case	catch
char	checked	class	const	continue	decimal	default	delegate
do	double	else	enum	event	explicit	extern	false
finally	fixed	float	for	foreach	goto	if	implicit
in	int	interface	internal	is	lock	long	namespace
new	null	object	operator	out	override	params	private
protected	public	readonly	ref	return	sbyte	sealed	short
Sizeof	stackalloc	static	string	struct	switch	this	throw
true	try	typeof	uint	ulong	unchecked	unsafe	ushort
using	virtual	void	volatile	while			

红字表较常用关键字

LearnASP.NET-rev1.1 <http://www.coding24.org>

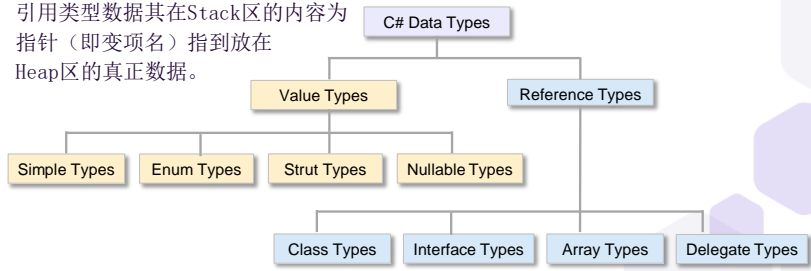
10

- 关键字或称保留字, 在各类编程语言中都相当的多, 我们要先认识一下, 不能用这些关键字作其它命名的用途。先认识常用的, 在编程时会较轻松些, 其它的等用到时再去了解。



## C#的数据类型

- C#主要将数据类型分为两种：**值类型 (Value Type)** 和 **引用类型 (Reference type)**。  
值类型包括：**简单类型**、**枚举类型**、**结构类型**和**Nullable**类型。引用类型包括：**类类型**、**接口类型**、**数组类型**和**委托类型**。
- 值类型数据存在Stack内存中，可直接存取；  
引用类型数据其在Stack区的内容为指针（即变量名）指向放在Heap区的真正数据。



```
graph TD; C[C# Data Types] --> V[Value Types]; C --> R[Reference Types]; V --> S[Simple Types]; V --> E[Enum Types]; V --> ST[Strut Types]; V --> N[Nullable Types]; R --> CT[Class Types]; R --> IT[Interface Types]; R --> AT[Array Types]; R --> DT[Delegate Types];
```

LearnASP.NET-rev1.1 <http://www.coding24.org> 11

### C#两种数据类型：

- **Value Type**：称值类型或实值类型，值类型存储实际值，在堆栈（**stack**）上分配。
- **Reference Type**：引用类型或称参考类型，引用类型在堆（**heap**）上分配，通常代表类实例。
- 在C#代码中还可以定义自己的值和引用类型。



## C#的简单类型


- 简单类型从类 `System.ValueType` 中派生，直接包含数据，类型及说明如下

类型	说明及范围	类型	说明及范围
bool	布尔值true/false	float	4 byte单精度浮点数
char	2 byte Unicode 字符	double	8 byte双精度浮点数
byte	1 byte无符号整数0 到 255	decimal	16 byte精确的十进制值，28-29 有效位数
sbyte	1 byte有符号整数-128 到 127	ushort	2 byte无符号整数
short	2 byte有符号整数-32,768 到 32,767	uint	4 byte无符号整数
int	4 byte有符号整数-2,147,483,648 到 2,147,483,647	ulong	8 byte无符号整数
long	8 byte有符号整数		

LearnASP.NET-rev1.1 <http://www.coding24.org>

12

- 简单类型从类 `System.ValueType` 中派生，直接包含数据。如 `int`、`char`、`float`，它们分别存储数字、字符、浮点数。当您声明一个 `int` 类型时，系统分配4 bytes内存来存储值。



## 变量和常量

● 在程序运行中通常需要一或多个存储区，若储存区内容可变为变量，若是初值完成后不可变则为常量。在声明语句中会给定名字以方便操作。

```

7 protected void Page_Load(object sender, EventArgs e)
8 {
9     short a = 5; //变量声明并赋值
10    int b; //变量声明
11    double c; //变量声明
12    const double d = 4.23; //常量声明并赋值
13    string s = "C# programming language "; //字符串变量
14    b = 10; //变量赋值
15    c = a + b; //变量赋值
16
17    Response.Write($"a = {a}, b = {b}, c = {c}<br/>");
18    //以下写法亦可
19    //Response.Write(string.Format("a = {0}, b = {1}, c = {2}<br/>", a, b, c));
20    Response.Write(d + "<br/>");
21    Response.Write(s + "<br/>");
22 }

```

a = 5, b = 10, c = 15  
 4.23  
 C# programming language

Examples/Unit02/VariableConst.aspx

LearnASP.NET-rev1.1 <http://www.coding24.org> 13

- 变量：就是在程序的运行过程中其值可以被改变的量，变量的类型可以是任何一种C#的数据类型。
- 常量：就是在程序的运行过程中其值不能被改变的量。常量的类型也可以是任何一种C#的数据类型。常量用保留字const来声明。
- 多个变量或常量串接为字符串的方式：Response.Write(String)的字符串参数只能有一个，因此在输出前需将要输出的信息串接起来，这些信息可能有常量，如“Hello World!”，这是字符串常量，也可能有变量，如上面的a,b或c变量，如何串接有以下几种方式：
  - 用+（加号）：如Response.Write(“Hello ” + “World!”);
  - 用\$及占位符：如Response.Write(\$"a = {a}, b = {b}, c = {c}<br/>"); 其中a, b, c是变量。
  - 用string.Format()方法：如 Response.Write(string.Format("a = {0}, b = {1}, c = {2}",a,b,c));
- VariableConst.aspx.cs

```

using System;

namespace LearnASPNET.Examples.Unit02
{
    public partial class VariableConst : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            short a = 5; //变量声明并赋值
            int b; //变量声明
            double c; //变量声明
            const double d = 4.23; //常量声明并赋值
            string s = "C# programming language"; //字符串变量
            b = 10; //变量赋值
        }
    }
}

```

```
c = a + b; //变量赋值

Response.Write($"a = {a}, b = {b}, c = {c}<br/>");
//以下写法亦可
//Response.Write(string.Format("a = {0}, b = {1}, c =
{2}<br/>", a, b, c));
Response.Write(d + "<br/>");
Response.Write(s + "<br/>");
    }
}
```



## 转型及格式化输出

```

7 protected void Page_Load(object sender, EventArgs e)
8 {
9     bool b = true;
10    double d = 1234.765;
11    int i = 15;
12    float f = 43.005f;
13
14    Response.Write(b.ToString() + "<br/>"); //转为字符串
15    Response.Write(d.ToString("#.##") + "<br/>"); //格式化输出1234.77
16    Response.Write(i + "<br/>"); //输出15
17    Response.Write($"{f:F2}<br/>"); //语法{索引[,宽度]:格式字符}, 格式化输出43.01
18    Response.Write((int)d); //强制转型为整数1234
19 }

```

True  
 1234.77  
 15  
 43.01  
 1234

Examples/Unit02/DataType.aspx

LearnASP.NET-rev1.1 <http://www.coding24.org>
14

### 占位符及格式化：（在\${}及string.Format中都可应用）

- C#的占位符
  - C#字符串可应用占位符，如string.Format("变项1={0}, 变项2={1}",变项1,变项2); 其中字符串中的{0},{1}就是占位符，先占位置然后以后面的变项值取代它。占位符的编号从0开始。
  - 在以Response.Write()输出前，可以用string.Format()整合要输出的内容。
- 位占符的格式：{index [,alignment] [:formatChar]}
  - 其中index指占位符索引，alignmen指对齐及宽度，:formatChar指输出的格式。
  - alignment是可选的，是一个带符号的整数，指示首选的格式化字段宽度。如果值小于格式化字符串的长度，会被忽略，并且使用格式化字符串的长度作为字段宽度；如果为正数，为向右对齐；如果为负数则为向左对齐。
- 字符串占位符及格式化示例：
 

```
int i=45123;
string.Format("{0:C}",i); //货币
string.Format("{0:D}",i); //十进制数
string.Format("{0:E}",i); //科学技术法
string.Format("{0:F}",i); // 浮点数表示法
string.Format("{0:G}",i); //G或g General 常用格式
string.Format("{0:N}",i); //N或n 用逗号分割千位的数字
//上述的格式化英文字母大小写相同
```

- **DataType.aspx.cs**


```
using System;

namespace LearnASPNET.Examples.Unit02
{
```



```
public partial class DataType : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        bool b = true;
        double d = 1234.765;
        int i = 15;
        float f = 43.005f;

        Response.Write(b.ToString() + "<br/>"); //转为字符串
        Response.Write(d.ToString("#.##") + "<br/>"); //格式化输出1234.77
        Response.Write(i + "<br/>"); //输出15
        Response.Write($"{f:F2}<br/>"); //语法{索引[,宽度]:格式字符}, 格式化输出
43.01
        Response.Write((int)d); //强制转型为整数1234
    }
}
```



## C#的string和StringBuilder

- string字符串类型:
  - 字符串是一个引用类型，支持常用的字符串操作及字符串格式化等功能。c#的string最后映射为.NET Framework的String（大写），但在编程中常使用string。

```
string s = "Hello"; //分配固定的内存大小
s = s + " World"; //创建新的内存代价较昂贵
```
- StringBuilder字符串类型
  - string对象是不可改变的，要修改时要重建，代价高。如果要修改字符串而不创建新的对象，可用System.Text.StringBuilder类。

```
StringBuilder sb = new StringBuilder(); //创建对象
Sb.Append("Hello"); //追加字符串
Sb.Append(" World"); //追加字符串
```

LearnASP.NET-rev1.1 <http://www.coding24.org> 15

string及StringBuilder的使用时机:

- string字符串一旦创建就无法修改大小，每次变动其内容时，都要在内存中创建一个新的字符串对象，为其分配新的内存空间。这代价会较为昂贵。
- 如果要修改字符串而不创建新的对象，则可以使用System.Text.StringBuilder类。例如，在一个while或for循环中将许多字符串连接在一起时，使用StringBuilder类可以提升性能。



### 3. C# 运算符和表达式

LearnASP.NET-rev1.1 <http://www.coding24.org>

- **C#的运算符和表达式：**包含算术运算符、关系运算符、逻辑运算符及位运算符以及其所构成的表达式。这在各类编程语言里都有，若有其它编程语言基础的朋友可快速浏览本节，大家也可以发现它与C家族的成员编程语言极为相似。



## C# 运算符和表达式


- 运算符包括

- 算术运算符: (一元): **++**、**--**、+、- (二元): \*、/、**%**(余数)、+、-
- 关系运算符: **==** (相等)、**!=** (不等)、<、>、<= 和 >=
- 逻辑运算符:
  - !(非)、& (且)、| (或)、^ (异或)
  - &&**(且)、**||** (或): 仅在必要时才做右侧运算
- 位运算符:
  - 使用整数类型或 **char** 类型的操作数执行位运算或移位运算
  - ~ (求补数)、<< (向左移位)、>> (向右移位)、& (且)、| (或)、^ (异或)

LearnASP.NET-rev1.1 <http://www.coding24.org>

17

- 包含了四类常见的运算符：算术运算符、关系运算符、逻辑运算符及位运算符，请特别注意标上红色之处，这些与数学里的一些符号不太一样，当然它们与Java或C语言是一样的。



## C# 运算符和表达式

```

8 protected void Page_Load(object sender, EventArgs e)
9 {
10     //赋值型表达式
11     int a, b, c;
12     a = 7; b = a; c = b++; //a=7, b=8, c=7
13     Response.Write($"a = {a}, b = {b}, c = {c}<br/>");
14     b = a + b * c;
15     c = a >= 100 ? b : c / 10;
16     a = (int)Math.Sqrt(b);
17     Response.Write($"a = {a}, b = {b}, c = {c}<br/>");
18
19     string s = "Hello World!";
20     char ch = s[s.Length - 2]; //倒数第2字符为d
21     var numbers = new List<int>(new[] { 1, 2, 3 }); //声明并建立数列
22     b = numbers.FindLast(n => n > 1); //在数列中找到最后1个大于1的值
23     Response.Write($"s = {s}, ch = {ch}, b = {b}");
24 }

```

a = 7, b = 8, c = 7  
a = 7, b = 63, c = 0  
s = Hello World!, ch = d, b = 3

Examples/Unit02/Expression1.aspx

LearnASP.NET-rev1.1 <http://www.coding24.org>
18

- C#的运算符和表达式范例，其中：

- 三元表达式（需要三个操作对象）：

语法为：（条件表达式）？表达式1：表达式2。

说明：问号前面的（）位置是判断的条件，要先运算，判断结果为true时调用表达式1，为false时调用表达式2。

三元表达式范例：

```

string result;
int i = 5, j = 10;
//三元表达式
result = i > j ? "success" : "fail";

```

- Math.Sqrt()：用于计算指定数字的平方根。
- 通常在查找List中的某个值，可以使用循环遍历对比方法，查找出结果。C#中提供了更方便的Find方法、FindLast方法及FindAll方法，可以直接使用，只要查找条件传入就可。
- numbers.FindLast(n => n > 1); 在括号中的是C#的Lambda表达式，要在numbers数列中找到最后1个值，而那个值的条件是要大于1。

Lambda表达式(n => n > 1)是一个匿名函数，是一种高效的类似于函数式编程的表达式。



## C# 运算符和表达式

```
8 protected void Page_Load(object sender,
9 {
10     //内插字符串表达式,类似占位符
11     var r = 2.3;
12     var msg = $"半径为{r}时,圆面积为{ Math.PI * r * r:F2}";
13     Response.Write(msg + "<br/>");
14     // 输出结果: 半径为2.3时,圆面积为16.62
15
16     //Lambda 表达式,可用于创建匿名函数
17     int[] numbers = { 2, 3, 4, 5 };
18     var maxSquare = numbers.Max(x => x * x);
19     Response.Write($"最大平方值:{maxSquare}");
20     //输出结果: 最大平方值:25
21 }
```

半径为2.3时,圆面积为16.62  
最大平方值:25

Examples/Unit02/Expression2.aspx

LearnASP.NET-rev1.1 <http://www.coding24.org>

19

- Lambda 表达式

使用 Lambda 表达式来创建匿名函数,其格式如下:

(input-parameters) => expression

左侧指定输入参数,然后在另一侧输入表达式或语句块。“=>”称为 lambda 运算符。

它将左侧的输入参数与右侧的 lambda 主体分开。



## C# 运算符和表达式

97 90 85

```
12 protected void Page_Load(object sender, EventArgs e)
13 {
14     //查询表达式, 可用于直接以 C# 使用查询功能
15     var scores = new[] { 90, 97, 78, 68, 85 };
16     IEnumerable<int> highScores =
17         from score in scores
18         where score > 80
19         orderby score descending
20         select score;
21     Response.Write(string.Join(" ", highScores)); //在元素间加空白
22     //输出结果: 97 90 85
23 }
```

Examples/Unit02/Expression3.aspx

LearnASP.NET-rev1.1 <http://www.coding24.org>

20

- **IEnumerable**这个接口是一个公开枚举数, 支持在非泛型集合上进行简单的迭代。换句话说, 它支持数组的遍历, 上面的范例就是利用这个特性。
- **string**的**Join()**方法: 用于连接数组的元素, 在每个元素之间使用指定的分隔符。它返回一个修改后的字符串。



## 4. C#各式语句范例

LearnASP.NET-rev1.1 <http://www.coding24.org>

- 本节将列出各类常见的编程语句，让各位读者在短时间内能熟悉C#的各式语句的编写方式。这些类型包含重复（循环）语句、选择（判断）语句及面向对向编程相关语句等。





## 重复语句

```
//Exercise1:
//重复语句 - for
for (int i = 0; i < 3; i++)
{
    Response.Write(i);
}
//输出结果:
// 012
```

```
// Exercise2:
//重复语句 - foreach
var Numbers =
    new List<int> { 0, 1, 1, 2, 3, 5, 8, 13 };
foreach (int n in Numbers)
{
    Response.Write($"{n} ");
}
//输出结果:
// 0 1 1 2 3 5 8 13
```

LearnASP.NET-rev1.1 <http://www.coding24.org>

22

- **for** 重复(循环)语句是一个允许您编写一个执行特定次数的循环的重复控制结构。
- **foreach** 用于列举出集合中所有的元素，**foreach** 语句中的表达式由关键字 **in** 隔开的两个项组成。**in** 右边的项是集合名，**in** 左边是变量名，用来存放该集合中的每个元素。
- **Exercise1.aspx.cs**

```
using System;


namespace LearnASPNET.Labs.Unit02
{
    public partial class Exercise1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            //Exercise1:
            //重复语句 - for
            for (int i = 0; i < 3; i++)
            {
                Response.Write(i);
            }
            //输出结果:
            // 012
        }
    }
}
```

- **Exercise2.aspx.cs**

```
using System;
using System.Collections.Generic;

namespace LearnASPNET.Labs.Unit02
{
    public partial class Exercise2 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            // Exercise2:
            //重复语句 - foreach
            var Numbers =
```

```
        new List<int> { 0, 1, 1, 2, 3, 5, 8, 13 };  
        foreach (int n in Numbers)  
        {  
            Response.Write($"{n} ");  
        }  
        //输出结果:  
        // 0 1 1 2 3 5 8 13  
    }  
}
```

 **重复语句**

```
// Exercise3:
//重复语句 - do
int n = 0;
do
{
    Console.Write(n);
    n++;
} while (n < 5);
//输出结果:
// 01234
```

```
//Exercise4:
//重复语句 - while
int n = 0;
while (n < 5)
{
    Response.Write(n);
    n++;
}
//输出结果:
// 01234
```

LearnASP.NET-rev1.1 <http://www.coding24.org> 23

- do...while 循环是在循环的尾部检查它的条件，与 while 循环类似，但是 do...while 循环会确保至少执行一次循环。While 语句是先检查条件，若为 true 才执行其内的语句。
- Exercise3.aspx.cs

```
using System;

namespace LearnASPNET.Labs.Unit02
{
    public partial class Exercise3 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            // Exercise3:重复语句 - do
            int n = 0;
            do
            {
                Response.Write(n);
                n++;
            } while (n < 5);
            //输出结果: 01234
        }
    }
}
```

- Exercise4.aspx.cs

```
using System;

namespace LearnASPNET.Labs.Unit02
{
    public partial class Exercise4 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            //Exercise4:
            //重复语句 - while
            int n = 0;
            while (n < 5)
            {
                Response.Write(n);
            }
        }
    }
}
```

```
        n++;  
    }  
    //输出结果:  
    // 01234  
}  
}
```



## 选择语句

```
// Exercise5: 选择语句- if...else
void DisplayWeatherReport(double t) //t:温度
{
    if (t < 20.0) {
        Response.Write("很冷!");
    } else {
        Response.Write("完美!");
    }
}
DisplayWeatherReport(15.0); //输出:很冷!
DisplayWeatherReport(24.0); //输出:完美!
```

LearnASP.NET-rev1.1 <http://www.coding24.org>

24

- 选择语句 if...else: 一个 if 语句后可跟一个可选的 else 语句, 如果布尔表达式为 true, 则执行 if 块内的代码。如果布尔表达式为 false, 则执行 else 块内的代码。
- Exercise5.aspx.cs

```
using System;

namespace LearnASPNET.Labs.Unit02
{
    public partial class Exercise5 : System.Web.UI.Page
    {
        // Exercise5: 选择语句 - if...else
        protected void Page_Load(object sender, EventArgs e)
        {
            DisplayWeatherReport(15.0); //输出:很冷!
            DisplayWeatherReport(24.0); //输出:完美!
        }
        void DisplayWeatherReport(double t) //t:温度
        {
            if (t < 20.0)
            {
                Response.Write("很冷!<br/>");
            }
            else
            {
                Response.Write("完美!<br/>");
            }
        }
    }
}
```

- Exercise6.aspx.cs

```
using System;

namespace LearnASPNET.Labs.Unit02
{
    public partial class Exercise6 : System.Web.UI.Page
    {
        // Exercise6: 选择语句 - switch
        protected void Page_Load(object sender, EventArgs e)
        {
            //以下为方法调用
        }
    }
}
```

```
DisplayByGrade('A');  
//输出: 成绩杰出! 你的成绩是 A  
DisplayByGrade('B');  
//输出: 成绩优良! 你的成绩是 B  
DisplayByGrade('C');  
//输出: 您通过了! 你的成绩是 C  
DisplayByGrade('F');  
//输出: 不合格! 你的成绩是 F  
DisplayByGrade('K');  
//输出: 无效的成绩! 你的成绩是 K  
}  
  
void DisplayByGrade(char grade)  
{  
    switch (grade)  
    {  
        case 'A':  
            Response.Write("成绩杰出! <br/>"); break;  
        case 'B':  
            Response.Write("成绩优良! <br/>"); break;  
        case 'C':  
        case 'D':  
            Response.Write("您通过了! <br/>"); break;  
        case 'F':  
            Response.Write("不合格! <br/>"); break;  
        default:  
            Response.Write("无效的成绩<br/>"); break;  
    }  
    Response.Write($"你的成绩是 {grade}<br/>");  
}  
}
```



## 选择语句

### // Exercise6: 选择语句- switch

```
void DisplayByGrade(char grade) {
    switch (grade) {
        case 'A':
            Response.Write("成绩杰出! "); break;
        case 'B':
            Response.Write("成绩优良! "); break;
        case 'C':
            Response.Write("您通过了! "); break;
        case 'D':
            Response.Write("不合格! "); break;
        default:
            Response.Write("无效的成绩"); break;
    }
    Response.Write($"你的成绩是 {grade}");
}
```

//以下为方法调用

```
DisplayByGrade('A');
//输出: 成绩杰出! 你的成绩是 A
DisplayByGrade('B');
//输出: 成绩优良! 你的成绩是 B
DisplayByGrade('C');
//输出: 您通过了! 你的成绩是 C
DisplayByGrade('F');
//输出: 不合格! 你的成绩是 F
DisplayByGrade('K');
//输出: 无效的成绩! 你的成绩是 K
```

LearnASP.NET-rev1.1 <http://www.coding24.org>

25

- 一个 switch 语句允许测试一个变量等于多个值时的情况。每个值称为一个 case，且被测试的变量会对每个 switch case 进行检查。switch 语句中的 expression 必须是一个整型或枚举类型，或者是一个 class 类型，其中 class 有一个单一的转换函数将其转换为整型或枚举类型。
- Exercise7.aspx.cs

```
using System;
namespace LearnASP.NET.Labs.Unit02
{
    // Exercise7:
    //面向对象编程 - 封装encapsulation
    public partial class Exercise7 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            Person p1 = new Person("张三", 6);
            Response.Write($"第1人, Name = { p1.Name } Age = { p1.Age}<br/>");
            // 声明第2人, 将p1指定给p2.
            Person p2 = p1;
            // 改变第2人的名字, 第1人名字亦更动.
            p2.Name = "李四";
            p2.Age = 16;
            Response.Write($"第2人, Name = { p2.Name } Age = { p2.Age}<br/>");
            Response.Write($"第1人, Name = { p1.Name } Age = { p1.Age}");
        }
    }

    public class Person
    {
        //属性Name, Age
        public string Name { get; set; }
        public int Age { get; set; }
        //构造方法
        public Person(string name, int age)
        {
            Name = name;
            Age = age;
        }
        // 其它的属性, 方法及事件...
    }
}
```

```
}  
}
```

- Exercise8.aspx.cs

```
using System;  
  
namespace LearnASPNET.Labs.Unit02  
{  
    // Exercise8:  
    // 面向对象编程 - 继承Inheritance  
    public partial class Exercise8 : System.Web.UI.Page  
    {  
        protected void Page_Load(object sender, EventArgs e)  
        {  
            Rectangle rect = new Rectangle();  
            rect.Width = 5;  
            rect.Height = 7;  
            Response.Write($"面积为:{ rect.getArea()}");  
        }  
    }  
  
    public class Shape  
    {  
        //属性宽度, 高度  
        public int Width { get; set; }  
        public int Height { get; set; }  
        // 其它的属性, 方法及事件...  
    }  
  
    public class Rectangle : Shape  
    {  
        public int getArea()  
        {  
            return Width * Height;  
        }  
    }  
}
```





## 面向对象编程语句

**// Exercise7:**  
**//面向对象编程-封装encapsulation**

```

public class Person {
    //属性Name, Age
    public string Name { get; set; }
    public int Age { get; set; }
    //构造方法
    public Person(string name, int age) {
        Name = name;
        Age = age;
    }
    // 其它的属性,方法及事件...
}

```

```

class Exercise7 {
    Page_Load () {
        Person p1 = new Person("张三", 6);
        Response.Write("第1人,Name = {0} Age = {1}",
            p1.Name, p1.Age);

        // 声明第2人, 将p1指定给p2.
        Person p2 = p1;
        // 改变第2人的名字,第1人名字亦更动.
        p2.Name = "李四";
        p2.Age = 16;

        Response.Write("第2人,Name = {0} Age = {1}",
            p2.Name, p2.Age);
        Response.Write("第1人,Name = {0} Age = {1}",
            p1.Name, p1.Age);
    }
}
//输出:
//第1人,Name = 张三 Age = 6
//第2人,Name = 李四 Age = 16
//第1人,Name = 李四 Age = 16

```

LearnASP.NET-rev1.1 <http://www.coding24.org>
26

- 在编写时注意 **Person** 是一个类，不要写在 **Page\_Load** 里面。在页面的后置代码中可放在同一个 **Namespace** 中，当然你也可以将它写在另一个档案中。
- **Exercise8.aspx.cs**

```

using System;
using System.Collections.Generic;

namespace LearnASPNET.Labs.Unit021
{
    // Exercise9:
    // 面向对象编程 - 多态Polymorphism
    public partial class Exercise9 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            var shapes = new List<Shape> {
                new Rectangle(3,5), new Circle(2)
            };
            foreach (var shape in shapes)
            {
                Response.Write(shape.area() + "<br/>");
            }
        }

        public abstract class Shape
        {
            //求面积,抽象方法
            public abstract double area();
        }

        public class Rectangle : Shape
        {
            //属性宽度, 高度
            public int Width { get; set; }
            public int Height { get; set; }
            public Rectangle(int w, int h)
            {
                Width = w;
                Height = h;
            }
            public override double area()
            {

```

```
        return Width * Height;
    }
}
public class Circle : Shape
{
    //属性:半径
    public int R { get; set; }
    public Circle(int r)
    {
        R = r;
    }
    public override double area()
    {
        return R * R * Math.PI;
    }
}
```



## 面向对象编程语句

// Exercise8:

// 面向对象编程- 继承Inheritance

```
public class Shape {  
    //属性宽度, 高度  
    public int Width { get; set; }  
    public int Height { get; set; }  
    // 其它的属性,方法及事件...  
}
```

```
public class Rectangle : Shape {  
    public int getArea() {  
        return Width*Height;  
    }  
}
```


```
class Exercise8 {  
    Page_Load () {  
        Rectangle rect = new Rectangle();  
        rect.Width = 5;  
        rect.Height = 7;  
        Response.Write ("面积为:{0}", rect.getArea());  
    }  
}
```

```
//输出:  
//面积为:35
```

LearnASP.NET-rev1.1 <http://www.coding24.org>

27

- 在编写时注意 Shape、Rectangle 都是类，不要写在 Page\_Load 里面。在页面的后置代码中可放在同一个 Namespace 中，当然你也可以将它写在另一个档案中。



## 面向对象编程语句

**// Exercise9:**  
**// 面向对象编程- 多态Polymorphism**

```
public abstract class Shape {  
    //求面积,抽象方法  
    public abstract double area();  
}  
  
public class Rectangle : Shape {  
    //属性宽度, 高度  
    public int Width { get; set; }  
    public int Height { get; set; }  
    public Rectangle(int w, int h) {  
        Width = w;  
        Height = h;  
    }  
    public override double area() {  
        return Width*Height;  
    }  
}
```

```
public class Circle : Shape {  
    //属性:半径  
    public int R { get; set; }  
    public Circle(int r){  
        R = r;  
    }  
    public override double area() {  
        return R*R*Math.PI;  
    }  
}
```

**class Exercise9 {**  
    **Page\_Load()** {  
        var shapes = new List<Shape> {  
            new Rectangle(3,5), new Circle(2)  
        };  
        foreach (var shape in shapes) {  
            Response.Write (shape.area());  
        }  
    }  
}

LearnASP.NET-rev1.1 <http://www.coding24.org>

28

**//输出:**  
**//15**  
**//12.5663706143592**

- 在 Exercise9 又重复出现 Shape、Rectangle 类，可能会与 Exercise8 同名冲突，要修一下它们的 Namespace 解决问题。



## 实作2- C#编程练习

Source:  
[Labs/Unit02/Exercise1.aspx](#) ...[Exercise9.aspx](#)

LearnASP.NET-rev1.1 <http://www.coding24.org>



## 实作3- C#编程练习

- 实作说明：将上一节的9类常见语句在VS2019中创建Web Forms类型项目进行编码并进行调试。
- 实作步骤：
  - 使用VS创建项目，类型：「ASP.NET Web应用程序(.NET Framework)」，项目/方案名为LabASPNET，框架：.NET Framework 4.8（选4.X亦可），在画面中选「空」，在添核心引用选「Web窗体」，取消「为HTTPS配置」。
  - 在项目上添加文件夹Labs，并于Labs下添加文件夹Unit02。
  - 在文件夹Unit02中添加「Web窗体」->命名为Exercise1.aspx
  - 在Exercise1.aspx后置代码中输入代码。
  - 运行程序：Exercise1.aspx，并查看结果是否正确。
  - 重复步骤3，完成Exercise2到Exercise9练习。方案及项目结构如下页：

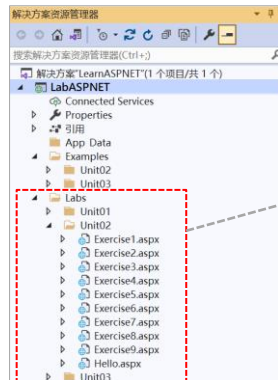
LearnASP.NET-rev1.1 <http://www.coding24.org>

30

- 注意：Exercise8及Exercise9都有Shape及Rectangle两个类名出现，且命名空间（namespace）都一致，在编写Exercise9时会出现错误。解决方法：将Exercise9后置代码（.cs）及页面（.aspx）的namespace都略作修改，如改为：namespace LearnASPNET.Labs.Unit021。



## 实作3- C#编程练习



各单元实作请都放在LabASPNET方案/项目中，并依次置入Labs\UnitXX文件夹中，方便您整理实作成果。除非必要不必另建项目及方案。

LearnASP.NET-rev1.1 <http://www.coding24.org>

31



## 单元小结

- 本单元从C#简介开始并以HelloWorld范例介绍了C#用于Console及Web项目的开发方式。
- 亦介绍了C#编程语言的：数据类型 / 运算符和表达式及各式语句范例。
- 单元中也以Web Forms项目型态实作练习各种 C#语法，透过这一单元的快速进入，以协助在未来单元在开发Web项目时能将C#编程能力应用于其中。