单元2–

# C#语言基础编程

1. C#简介及应用程序类型

实作1- 在Web Form中编写C#

2. C#程序及数据类型

3. C# 运算符和表达式

4.C#各式语句范例

实作2- C#编程练习

# 1. C#简介及应用程序类型

# 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。

# 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# 两种着名编程语言

# 实作1- 在Web Form中编写C#

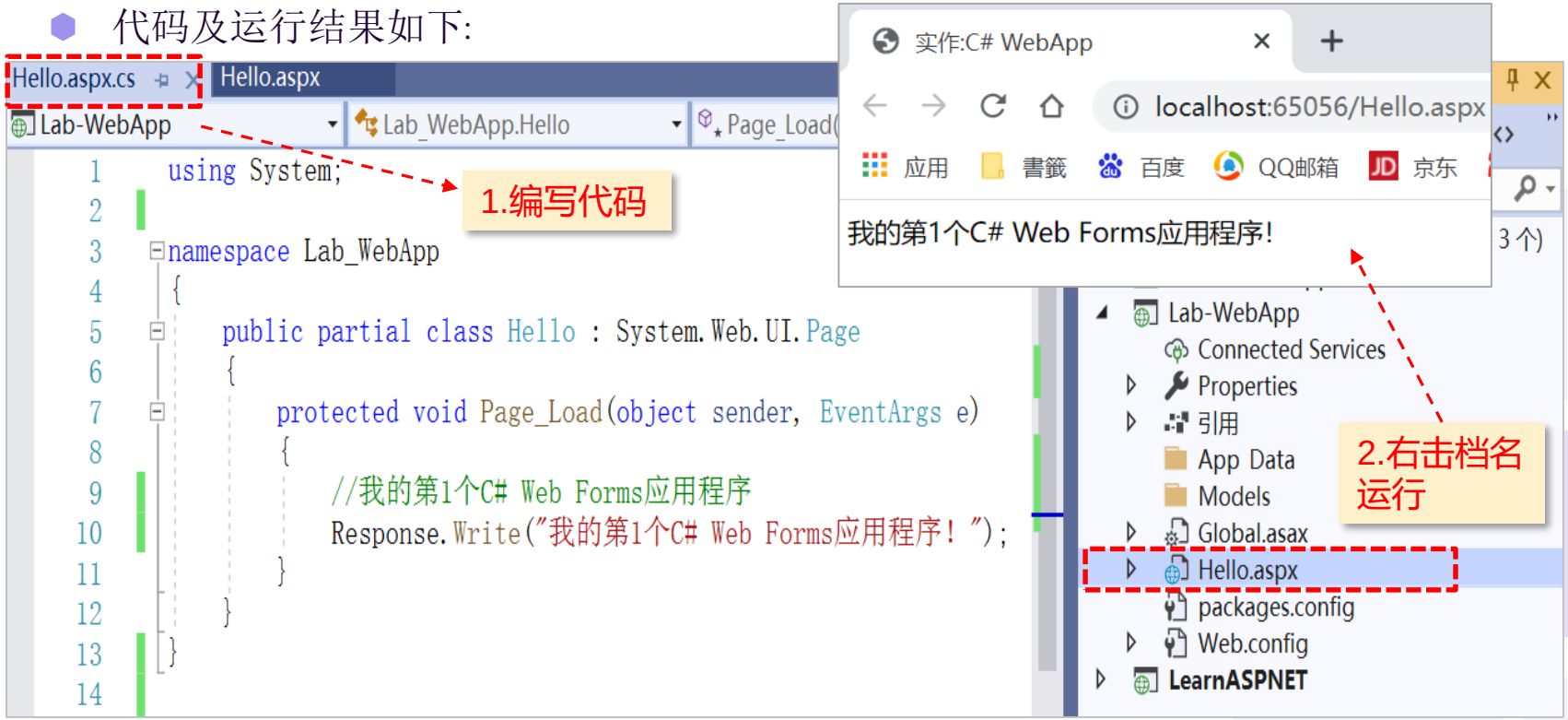
Source: Labs/Unit02/Hello.aspx

# 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。

# 实作1- 在Web Form中编写C#

- 代码及运行结果如下:



# 2. C#程序及数据类型

# 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
2   using System;
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.

# C# 的标识符和关键字

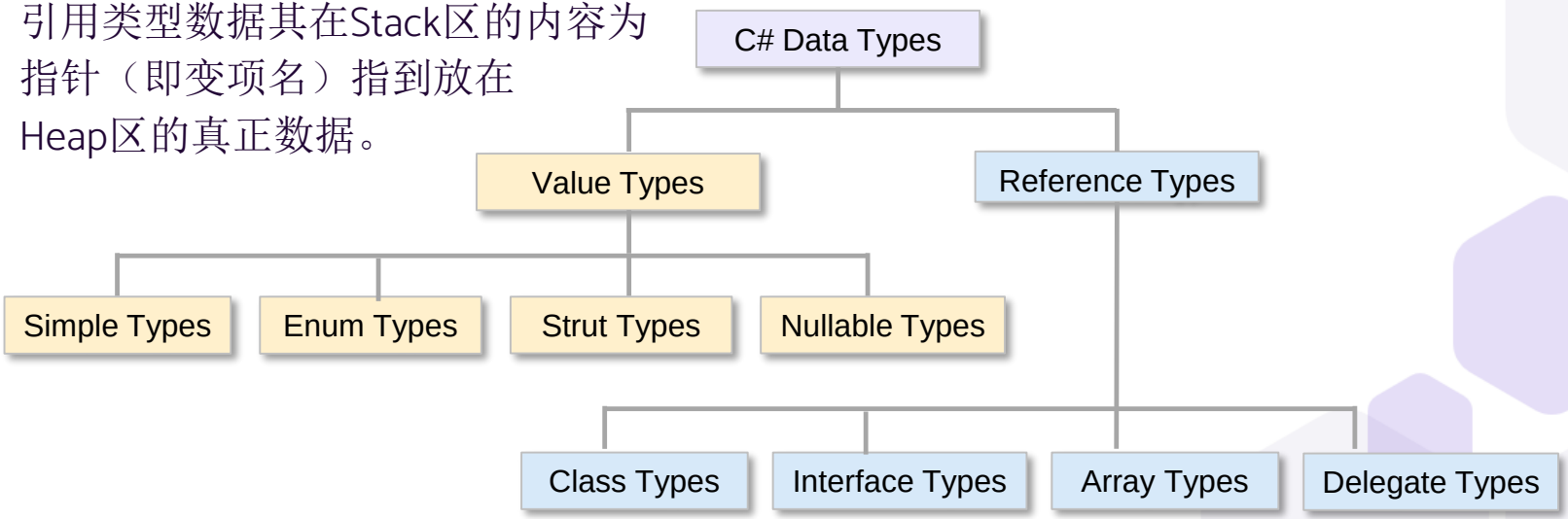
- 标识符（Identifier）是用来对类，方法或变项的命名以便在程序中识别。C#标识符区别大小写，且需以A-Z，a-z，_（下划线）或数字组成，但数字不能在开头。
- 关键字（Keyword）是保留标识符，对编译器有特别意义。 除非前面有 @ 前缀，否则不能在程序中用作标识符。

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 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

# 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内储中，可直接存取；引用类型数据其在Stack区的内容为指针（即变项名）指到放在Heap区的真正数据。



# C#的简单类型

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

| 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 | | |

# 变量和常量

- 在程序运行中通常需要一或多个存储区，若储存区内容可变则为变量，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。

```
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/>");
}
```

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

Examples/Unit02/VariableConst.aspx

# 转型及格式化输出

```
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
}
```

```
True
1234.77
15
43.01
1234
```

Examples/Unit02/DataType.aspx

# C#的string程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
```

# 3. C# 运算符和表达式

# 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)

# C# 运算符和表达式

```
protected void Page_Load(object sender, Even
{
    //赋值型表达式
    int a, b, c;
    a = 7; b = a; c = b++;  //a=7,b=8,c=7
    Response.Write($"a = {a}, b = {b}, c = {c}<br/>");
    b = a + b * c;
    c = a >= 100 ? b : c / 10;
    a = (int)Math.Sqrt(b);
    Response.Write($"a = {a}, b = {b}, c = {c}<br/>");

    string s = "Hello World!";
    char ch = s[s.Length - 2]; //倒数第2字符为d
    var numbers = new List<int>(new[] { 1, 2, 3 });   //声明并建立串列
    b = numbers.FindLast(n => n > 1);    //在串列中找到最后1个大于1的值
    Response.Write($"s = {s}, ch = {ch}, b = {b}");
}
```

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

Examples/Unit02/Expression1.aspx

# C# 运算符和表达式

```
protected void Page_Load(object sender,
{
    //内插字符串表达式,类似占位符
    var r = 2.3;
    var msg = $"半径为{r}时,圆面积为{ Math.PI * r * r:F2}";
    Response.Write(msg + "<br/>");
    // 输出结果: 半径为2.3时,圆面积为16.62

    //Lambda 表达式，可用于创建匿名函数
    int[] numbers = { 2, 3, 4, 5 };
    var maxSquare = numbers.Max(x => x * x);
    Response.Write($"最大平方值:{maxSquare}");
    //输出结果: 最大平方值:25
}
```

半径为2.3时,圆面积为16.62
最大平方值:25

Examples/Unit02/Expression2.aspx

# 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

# 4.C#各式语句范例

# 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
```

# 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
```

# 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!
```

# 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':
        case 'D':
            Response.Write("您通过了！");    break;
        case 'F':
            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
```

# 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
```

# 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
```

# 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());
        }
     }
}
//输出:
//15
//12.5663706143592
```

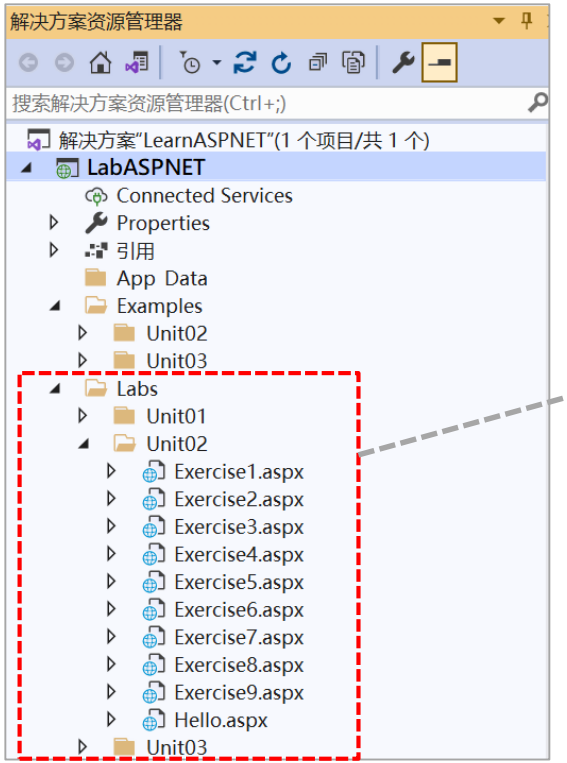
# 实作2- C#编程练习

Source:
Labs/Unit02/Exercise1.aspx …Exercise9.aspx

# 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：

# 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。

# 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。