

C#程序设计及应用

唐大仕

dstang2000@263.net

北京大学

Copyright © by ARTCOM PT All rights reserved.



第2章 C#语言基础 —数据运算、流程控制、数组

唐大仕

dstang2000@263.net

<http://www.dstang.com>



本章内容

- 第1节 数据类型、变量与常量
- 第2节 运算符与表达式
- 第3节 流程控制语句
- 第4节 数组



第1节 数据类型、变量与常量



变量

- 命名变量名要遵守如下的规则：

- (1) 不能是C#关键字。
- (2) 由字母、数字、下划线构成。
- (3) 第一个字符必须是字母或下划线。
- (4) 不要太长，一般不超过31个字符为宜。
- (5) 变量名最好不要与库函数名、类名相同。

■注： C#是大小写敏感的



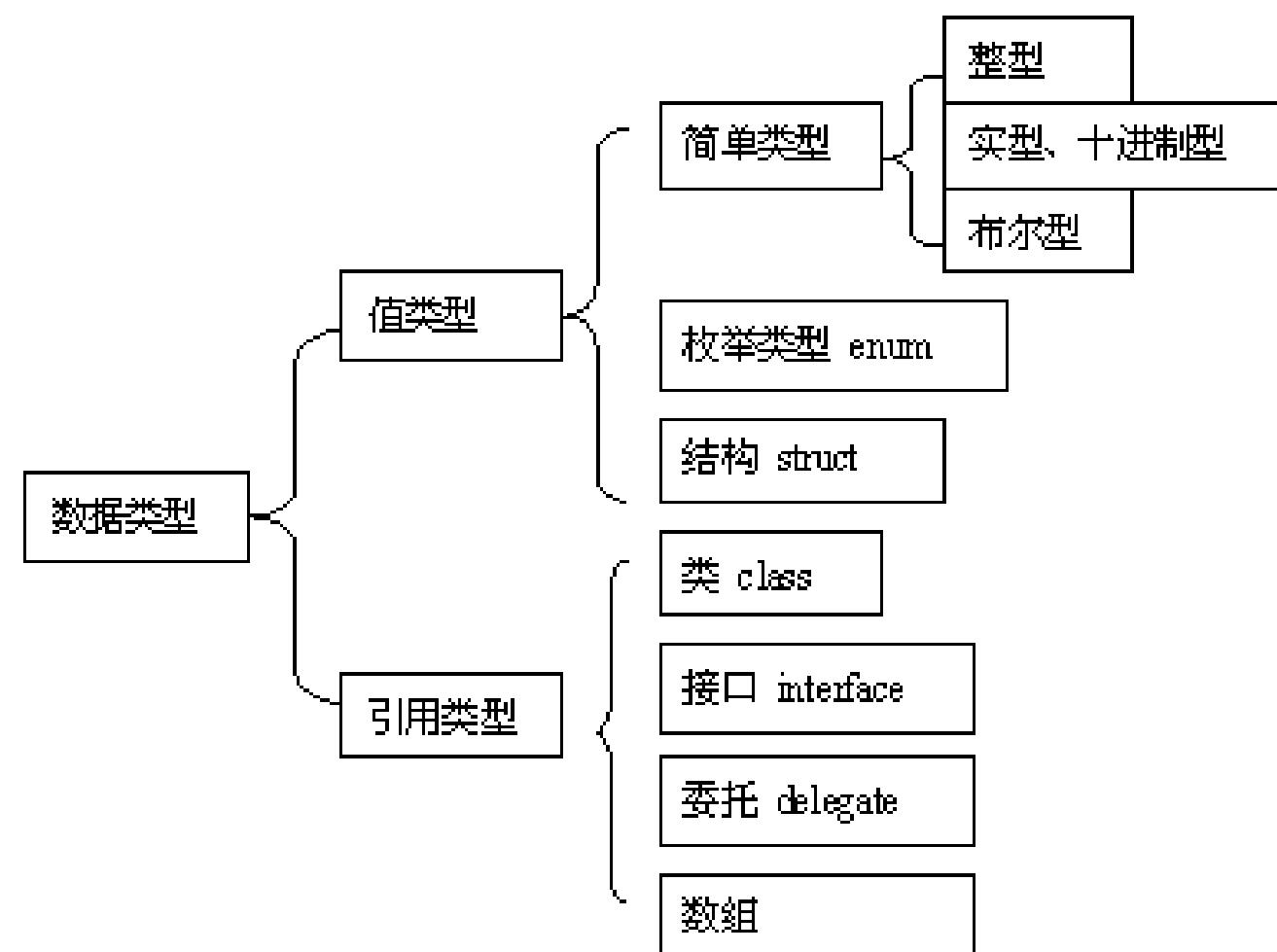
常量

- 变量是常数或代表固定不变值的名字。程序中如果想让变量的内容初始化后一直保持不变，可以定义一个常量。



C#数据类型

- C#的数据类型分值类型 (Value Type) 和引用类型 (Reference Type) 两大类
- 值类型包括
 - 简单类型 (Simple Type)
 - 结构类型 (Struct Type)
 - 枚举类型 (Enum Type)
- 引用类型包括
 - 类类型 (Class Type)
 - 接口类型 (Interface Type)
 - 委托类型 (Delegate)
 - 数组类型 (Array Type)
- 值类型与引用类型的区别
 - 前者一般是存在栈中，后者则是有一个引用变量，一个对象内存（存入堆中）





简单类型

- 整数类型

- 有符号 sbyte short int long 如 87L, 0x1F (注：没有八进制写法)
 - 无符号 byte ushort uint ulong 如 87UL
 - 字符类型 char 如 'a' '\uA0B1' '\n' (回车)

- 实数类型

- float 如 3.14F
 - double 如 3.14 3.14D (后面这个D可以省略)

- 十进制类型

- Decimal 如 120.50M

- 布尔类型

- bool 如 true false (小写)



等价类型

- 注意：每种数据类型都有一个关键词
 - int 相当于 System.Int32
 - double 相当于 System.Double
 - bool 相当于 System.Boolean
 - string 相当于 System.String
 - (如果using System, 则 string相当于String)



使用数据类型要注意

- 针对VB程序员
 - int 为32位长
 - 字符 (char)与字符串不同
- 定义变量
 - int a, b;
 - float pi = 3.14F;
 - 类型是严格的，比VB严格



逻辑型

- bool类型适于逻辑运算，一般用于程序流程控制
- bool类型数据只允许取值true或false，不可以0或非0的整数替代true和false。



字符型

- char型数据用来表示通常意义上“字符”
- 字符常量是用单引号括起来的单个字符
 - `char c = 'A';`
- C#字符采用Unicode编码，每个字符占两个字节，因而可用十六进制编码形式表示
 - `char c1 = '\u0061';`
- C#语言中还允许使用转义字符'\'来将其后的字符转变为其它的含义
 - `char c2 = '\n'; //代表换行符`



转义符

- 转义字符含 义
- \uxxxx 1到4位十六进制数所表示的字符(xxxx)
- \' 单引号字符
- \" 双引号字符
- \\ 反斜杠字符
- \r 回车
- \n 换行
- \f 走纸换页
- \t 横向跳格
- \b 退格



字符串类型

- String

- 是对象类型，但对字符串常量有特殊处理

- “abcd1234”

- @“abcd”

- Pqrst“”

- 字符串前使用@， aa则可以不进行\转义，可以换行，双引号则用两个双引号表示一个双引号



基本数据类型变量声明和赋值

```
public class Test {  
    public static void Main () {  
        bool b = true;          //声明bool型变量并赋值  
        int x, y=8;            // 声明int型变量  
        float f = 4.5f;         // 声明float型变量并赋值  
        double d = 3.1415;      //声明double型变量并赋值  
        char c;                //声明char型变量  
        c = '\u0031';           //为char型变量赋值  
        x = 12;                //为int型变量赋值  
    }  
}
```



标识符

- 任何一个变量、常量、方法、对象和类都需要有名字，这些名字就是标识符。标识符可以由编程者自由指定，但是需要遵循一定的语法规规定。
- 标识符要满足如下的规定：
 - (1) 标识符可以由字母、数字和下划线(_)、美元符号(\$)组合而成；
 - (2) 标识符必须以字母、下划线或美元符号开头，不能以数字开头。
- 在实际应用标识符时，应该使标识符能一定程度上反映它所表示的变量、常量、对象或类的意义，这样程序的可读性会更好。
- 同时，应注意C#是大小写敏感的语言。



C#新版本中的特殊类型

- 推断类型(C#3.0)
 - var a = 1+2;
 - 与javascript中不同，其类型由编译器推断，在编译时就确定了的。
- Nullable类型(C#3.0)
 - int? a = 32;
 - if(a.HasValue).....
- Dynamic (C#4.0) 由DLR 支持
 - dynamic x = new Cell();
 - 编译时不检查，运行时才确定，主要用于与COM组件或其他语言交互



第2节 运算符与表达式



运算符

- 算术运算符: +, -, *, /, %, ++, --
- 关系运算符: >, <, >=, <=, ==, !=
- 逻辑运算符: !, &, |, ^, &&, ||
- 位运算符: &, |, ^, ~, >>, <<
- 赋值运算符: = 扩展赋值运算符: +=, -=, *=, /=
- 字符串连接运算符: +



常用运算符

• 算术运算符

□ + - * / %

- 其中%表示求余，没有乘方运算

□ ++ --

• 关系运算符

□ > < >= <= == !=

- 注意等于及不等于的写法，与VB不同

• 逻辑运算符

□ 与 & 或 | 非 ! 异或 ^

□ 条件与 && 条件或 ||



常用运算符(续)

- 赋值

- =

- += -= *= 等等

- 注： $s+=a$ 相当于 $s=s+a$

- 条件运算符

- $z?a:b$

- 如 $m = a>b ? a : b$

- 特殊的 $m = a ?? "hello"$



算术运算符

- + , - , * , / , % , ++ , --
- 有关 / $15/4$ $15/3$ $15/2$ $15.0/2$
- 有关 % $100\%3$ $100\%-3$ $-100\%-3$ $-100\%3$
- 有关%的含义 偶数 $a \%2$, 整除 $a\%7$, 个位 $a\%10$
- 有关++, -- $a=5; a++; b=a^2$
- $a=5; b = ++ a ^2;$
- $a=5; b = a++ ^2;$
- ^不是乘方



字符串连接运算符 +

- "+" 除用于算术加法运算外，还可用于对字符串进行连接操作

```
int i = 300 +5;  
string s = "hello, " + "world!";
```

- "+" 运算符两侧的操作数中只要有一个是字符串 (String) 类型，系统会自动将另一个操作数转换为字符串然后再进行连接

```
string s = "hello, " +300 + 5 + "号";  
//输出：hello, 3005号
```



逻辑运算符(1)

■ 逻辑运算符功能

! -- 逻辑非

& -- 逻辑与

| -- 逻辑或

\wedge -- 逻辑异或 && -- 短路与

|| -- 短路或

■ 逻辑运算符功能说明:

a	b	!a	a&b	a b	$a \wedge b$	a&&b	a b
true	true	false	true	true	false	true	true
true	false	false	false	true	true	false	true
false	true	true	false	true	true	false	true
false	false	true	false	false	false	false	false



逻辑运算符(2)

■ 短路(short-circuit)逻辑运算符应用

`&&` -- 第一个操作数为假则不判断第二个操作数

`||` -- 第一个操作数为真则不判断第二个操作数

```
MyDate d;  
if ((d!=null) && (d.day >31)) {  
    //do something with d  
}  
  
if(i <0 || i >31 ) {  
    //("非法赋值");  
}
```



位运算符

■ 位运算符功能

\sim -- 取反

$\&$ -- 按位与

$|$ -- 按位或

\wedge -- 按位异或

$<<$ 左移

$>>$ 右移

■ 位运算符功能说明:

$$\begin{array}{r} \sim \\ \hline \boxed{0\ 1\ 0\ 0\ 1\ 1\ 1\ 1} \end{array}$$

取反操作：将二进制数的每一位取反。

$$\begin{array}{r} | \\ \hline \boxed{1\ 1\ 0\ 0\ 1\ 0\ 1\ 1} \\ \boxed{0\ 1\ 1\ 0\ 1\ 1\ 0\ 1} \end{array}$$

按位与操作：只有两个位都是1时，结果位才为1。

$$\begin{array}{r} \& \\ \hline \boxed{1\ 1\ 0\ 0\ 1\ 0\ 1\ 1} \\ \boxed{0\ 1\ 1\ 0\ 1\ 1\ 0\ 1} \end{array}$$

按位或操作：只要两个位中有一个是1，结果位就为1。

$$\begin{array}{r} \wedge \\ \hline \boxed{1\ 1\ 0\ 0\ 1\ 0\ 1\ 1} \\ \boxed{0\ 1\ 1\ 0\ 1\ 1\ 0\ 1} \end{array}$$

按位异或操作：如果两个位不同，则结果位为1；如果两个位相同，则结果位为0。



赋值运算符(1)

■ 赋值运算符=

- 当“=”两侧的数据类型不一致时，可以适用默认类型转换或强制类型转换(casting)原则进行处理

```
long l = 100;  
int i = (int)l;
```

- 特例：可以将整型常量直接赋值给byte, short, char等类型变量，而不需要进行强制类型转换，只要不超出其表示范围

```
byte b = 12;          //合法  
byte b = 4096;        //非法
```



赋值运算符(2)

■ 扩展赋值运算符

运算符	用法举例	等效的表达式
<code>+=</code>	<code>a += b</code>	<code>a = a+b</code>
<code>-=</code>	<code>a -= b</code>	<code>a = a-b</code>
<code>*=</code>	<code>a *= b</code>	<code>a = a*b</code>
<code>/=</code>	<code>a /= b</code>	<code>a = a/b</code>
<code>%=</code>	<code>a %= b</code>	<code>a = a%b</code>
<code>&=</code>	<code>a &= b</code>	<code>a = a&b</code>
<code> =</code>	<code>a = b</code>	<code>a = a b</code>
<code>^=</code>	<code>a ^= b</code>	<code>a = a^b</code>
<code><<=</code>	<code>a <<= b</code>	<code>a = a<<b</code>
<code>>>=</code>	<code>a >>= b</code>	<code>a = a>>b</code>



字符串连接运算符 +

- "+" 除用于算术加法运算外，还可用于对字符串进行连接操作

```
int i = 300 +5;  
String s = "hello, " + "world!";
```

- "+"运算符两侧的操作数中只要有一个是字符串(String)类型，系统会自动将另一个操作数转换为字符串然后再进行连接

```
int i = 300 +5;  
String s = "hello, " + i + "号"; //输出：hello, 305号
```

编程提示：

字符串与C语言中的字符串有很大的不同



表达式

- 表达式是符合一定语法规则的运算符和操作数的序列

a

5.0 + a

(a-b)*c-4

i<30 && i%10!=0

- 表达式的类型和值

□ 对表达式中操作数进行运算得到的结果称为表达式的值

□ 表达式的值的数据类型即为表达式的类型

- 表达式的运算顺序

□ 首先应按照运算符的优先级从高到低的顺序进行

□ 优先级相同的运算符按照事先约定的结合方向进行



表达式中的类型转换

- 当有不同种类的混合运算时:
- int→long→float→double

□另外，注意 **整型提升**

□(所有的byte, short, char 等转为int)



运算符优先级与结合性

Separator	. () { } ; ,
Associative	Operators
R to L	++ -- ~ ! (data type)
L to R	* / %
L to R	+ -
L to R	<< >> >>>
L to R	< > <= >= instanceof
L to R	== !=
L to R	&
L to R	^
L to R	
L to R	&&
L to R	
R to L	?:
R to L	= *= /= %= += -= <<= >= >>= &= ^= =



编程提示

- 类型的转换

- 字符串转成数值 :

- double.Parse(s) int.Parse(s)

- 数字转成字符串 :

- 10.ToString()
 - “” + 10

- 使用Convert

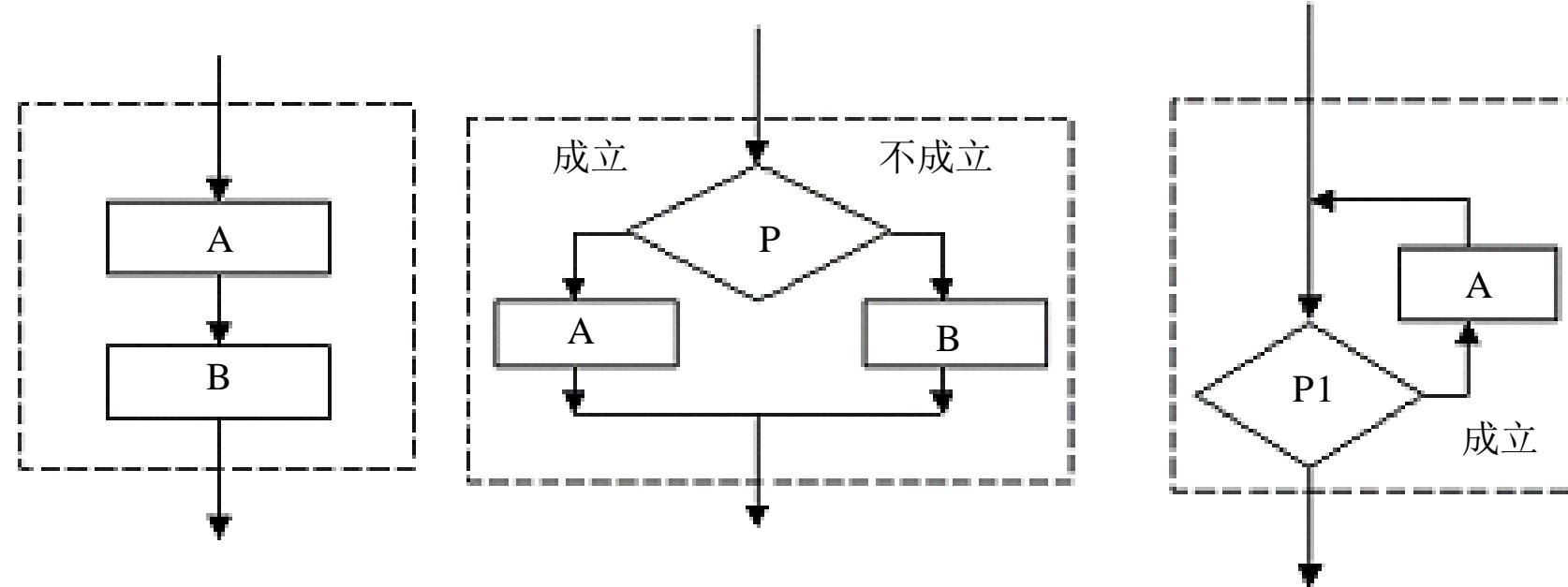
- Convert.ToInt32(textbox1.Text)
 - Convert.ToDouble(“123.45”)
 - Convert.ToDateTime(“2009-10-01 14:00”)



第3节 流程控制语句



结构化程序设计的三种基本流程





简单语句

- 最简单的语句是方法调用语句及赋值语句，是在方法调用或赋值表达式后加一个分号（；）
- 如：
- System.Console.WriteLine("Hello World");
- a = 3+x;
- b = a>0?a:-a;
- s = TextBox1.Text;
- d = int.Parse (s);



分支语句--if

- if(条件表达式)
- 语句块 ; // if分支
- else
- 语句块 ; // else分支
- 例 : **LeapYear.cs**



- 注意：没有表达式语句一说

- 2+3; 不能成为一个语句



分支语句--- switch语句

```
switch(exp){  
    case const1:  
        statement1;  
        break;  
    case const2:  
        statement2;  
        break;  
    ...  
    case constN:  
        statementN;  
        break;  
    [default:  
        statement_default;  
        break;]  
}
```

例：**GradeLevel.cs**

例：**AutoScore.cs**自动出题并判分

编程提示：必须有break；表达式可以是整数或字符串

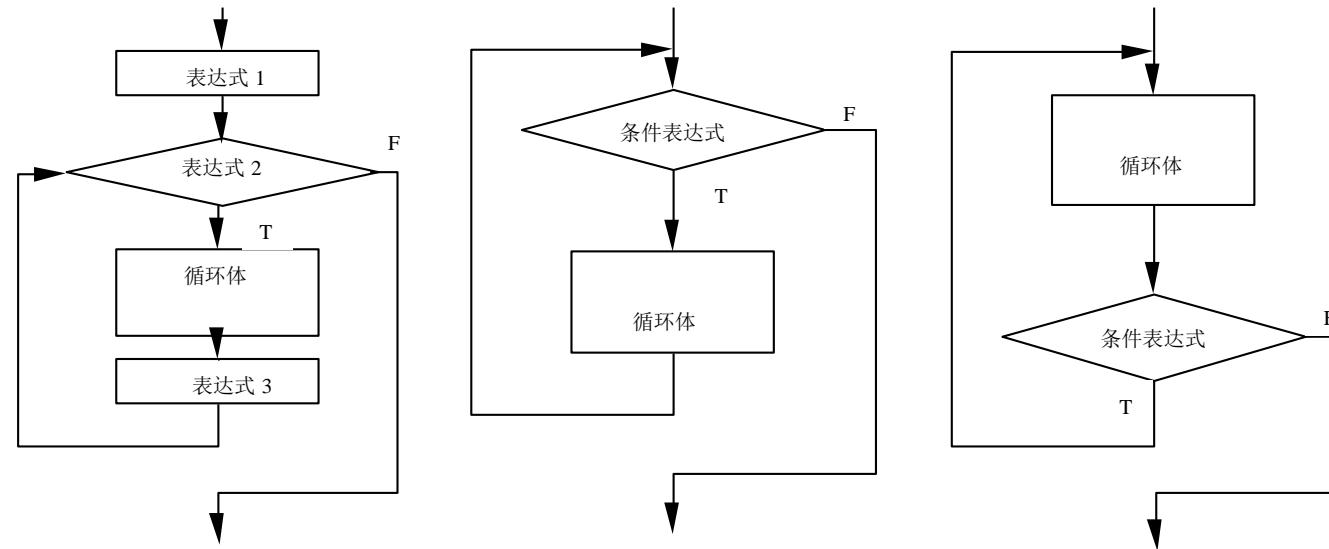


注意

- Switch语句与C++不同之处
 - 变量除了整型、枚举型，还可以用字符串
 - 不能随便贯穿，必须有break;



循环语句





循环语句

- 循环语句功能
 - 句子在循环条件满足的情况下，反复执行特定代码
- 循环五要素
 - 初始化部分 (init_statement)
 - 循环条件部分 (test_exp)
 - 循环体部分 (body_statement)
 - 迭代部分 (alter_statement)
 - "结束后处理"
- 循环语句分类
 - for 循环
 - while 循环
 - do/while 循环



for 循环语句

- 语法格式

```
for (init_statement; test_exp; alter_statement) {  
    body_statement  
}
```

- 应用举例

```
int result = 0;  
for(int i=1; i<=100; i++) {  
    result += i;  
}  
System.Console.WriteLine("result=" + result);
```



while 循环语句

■ 语法格式

```
[init_statement]  
while( test_exp ) {  
    body_statement;  
    [alter_statement];  
}
```

■ 应用举例

```
int result = 0;  
int i=1;  
while(i<=100) {  
    result += i;  
    i++;  
}  
System.Console.WriteLine("result=" + result);
```



do/while 循环语句

- 至少执行一次
- 语法格式

```
[init_statement]  
do {  
    body_statement;  
    [alter_statement];  
} while( test_exp);
```

- 应用举例

```
int result = 0; int i=1;  
do{  
    result += i;  
    i++;  
}while(i<=100);  
System.Console.Write("result=" + result);
```



跳转语句

- 常用的跳转语句：

- ❑break语句

- 结束循环(相当于VB中的Exit Do, Exit For)

- ❑continue语句

- 进入下一次循环

- ❑goto语句

- 跳转到某个语句标号

- ❑try{}catch{}语句

- 异常的捕获



Goto语句及其弊端

- 有关Goto语句的争论
- C#中的goto语句



第4节 数组



数组概述

- 数组是多个相同类型数据的组合，实现对这些数据的统一管理
- 数组属引用类型，数组型数据是对象(object)，数组中的每个元素相当于该对象的成员变量
- 数组中的元素可以是任何数据类型，包括基本类型和引用类型



一维数组声明

- 一维数组的声明方式：

type []var ;

例如：

```
int[] a1;  
double []b  
Mydate []c;
```

- C#语言中**声明数组时不能指定其长度**（数组中元素的个数），例如：

int a[5]; //非法



数组初始化

■ 动态初始化

数组定义与为数组元素分配空间并赋值的操作分开进行。

```
int []a;  
a = new int[3];  
a[0] = 3;  
a[1] = 9;  
a[2] = 8;
```

```
MyDate []dates;  
dates = new MyDate[3];  
dates[0] = new MyDate(22, 7, 1964);  
dates[1] = new MyDate(1, 1, 2000);  
dates[2] = new MyDate(22, 12, 1964);
```



数组初始化

■ 静态初始化：

在定义数组的同时就为数组元素分配空间并赋值。

```
int[] a = { 3, 9, 8};
```

```
MyDate[] dates= {  
    new MyDate(22, 7, 1964),  
    new MyDate(1, 1, 2000),  
    new MyDate(22, 12, 1964)  
};
```

注：最后可以多一个逗号。如
{3,9,8,}



数组元素的默认初始化

- 数组是引用类型，它的元素相当于类的成员变量，因此数组一经分配空间，其中的每个元素也被按照成员变量同样的方式**被隐式初始化**。例如：

```
int []a= new int[5];
//a[3]则是0
```



数组元素的引用

- 数组元素的引用方式 : `arrayName[index]`
 - `index`为数组元素下标 , 可以是整型常量或整型表达式。如`a[3]` , `b[i]` , `c[6*i]`;
 - 数组元素下标从0开始 ; 长度为n的数组合法下标取值范围 : `0 ~ n-1` ;
- 每个数组都有一个属性**Length**指明它的长度 , 例如 : `a.Length` 指明数组`a`的长度(元素个数) ;



多维数组 (1)

- 二维数组举例：

```
int[,] a = {{1,2,5},{3,4,0},{5,6,7}};
```

- 可以用 `a.GetLength(0)` , `a.GetLength(1)` 来获得各个维度的长度



交错数组

- C#中交错数组是数组的数组

```
int [][] t = new int [3][];
```

```
t[0] = new int[2];
```

```
t[1] = new int[4];
```

```
t[2] = new int[3];
```

i \ j	j = 0	j = 1	j = 2	j = 3
i = 0	1	2		
i = 1	3	4	0	9
i = 2	5	6	7	

C#中多维数组的声明和初始化应按从高维到低维的顺序进行

```
int t1[][] = new int [][4]; //非法
```



交错数组

- C#中多维数组不必须是规则矩阵形式

```
int[][] tt = new int[4][];  
tt[0] = new int[2];  
tt[1] = new int[4];  
tt[2] = new int[6];  
tt[3] = new int[8];
```

```
int tt[][] = new int[4][5];
```



复制数组

- **Array.Copy**方法提供了数组元素复制功能：

//源数组

```
int[] source = { 1, 2, 3, 4, 5, 6 };
```

// 目的数组

```
int []dest = { 10, 9, 8, 7, 6, 5, 4, 3, 2, 1 };
```

// 复制源数组中从下标0开始的source.length个元素到
// 目的数组，从下标0的位置开始存储。

```
Array.Copy( source, 0, dest, 0, 5 );
```



foreach语句

- foreach可以方便地处理数组、集合中各元素
- 如：

```
int[] ages = new int[10];  
foreach( int age in ages )  
{  
    //...  
}
```

foreach是只读式的遍历



练习C#语言基础

- 参考LanguageBasic目录下的示例
- 作业请见教学网站



本章小结

- 本章内容

- 数据类型；运算符与表达式；流程控制语句；数组

- 知识要点

- 常见的数据类型；字面常量的书写
 - 值类型与引用类型（重点）
 - 推断类型var, Nullable类型,dynamic类型
 - 常用的算术运算、关系运算、位运算、逻辑运算、连接运算
 - 运算符的优先级
 - if/switch语句
 - for/while/do语句，**循环的五要素**
 - break/continue/goto语句
 - 数组的定义、初始化
 - foreach语句



编程提示

- 变量及函数命名
 - 大小写
 - 匈牙利命名法
 - 动/名词性
 - 二、三个单词
 - 使用特定词而不是通用词
 - 使用“重构”（点右键—重构—重命名）
- 注释
 - 使用中文注释



补充：三种类型的应用程序比较

	WindowsForms程序	WPF程序	WebForm程序
Visual Studio中新建项目的类型	Windows ----Windows窗体应用程序	Windows----WPF应用程序	Web---Visual Studio 2012---ASP.NET web 空应用程序
主要的名称空间	System.Windows.Forms	System.Windows.Controls	System.Web.UI
主窗体继承自	Form	Window	Page
界面文件	自动生成的c#代码	自动生成的 XAML	自动生成的 HTML
程序入口	Program.cs中写的Main 其中有Application.Run(xxxx)	App.xaml中写的 StartupUri="MainWindow.xaml"	文件上点右键，设为起始页 (任何一页都可以)
全局对象	Application	App.Current	
全局变量	可用static变量	可用static变量	Application["变量名"] Session["变量名"]



补充：三种类型的应用程序比较

	WindowsForms程序	WPF程序	WebForm程序
控件命名	有默认名，最好修改一下	控件要自己命名（不然无法编程）	有默认名，最好修改一下
控件布局与位置	总体来说是绝对布局 可用Dock及	默认Grid布局，可用代码控制子控件 <code>btn.SetValue(Grid.RowProperty, 2);</code> 如果在Window上点右键，布局，选 Canvas，这样便于绝对布局 <code>Canvas.SetLeft(this.Button1, 200);</code>	对象的布局用css控制
控件背景及Color	<code>xx.BackColor = Color.Red</code> <code>Color.FromArgb(...)</code>	<code>xx.Background = new</code> <code>SolidColorBrush(Color.FromArgb(120,</code> <code>120, 255));</code> 其中可用 <code>Colors.Red</code>	用css控制，或 <code>CssBackgroundColor</code> 等属性
控件可见性	<code>xx.Visible = true;</code>	<code>xx.Visibility = Visibility.Visible</code> 或 <code>Hidden</code>	<code>xx.Visible = true;</code>
控件文本	<code>xx.Text =;</code>	<code>xx.Content = ...;</code>	<code>xx.Text = ..;</code>
添加子控件	<code>xx.Controls.Add(btn);</code>	<code>this.Grid1.Children.Add(btn);</code> <code>this.Canvas1.Children.Add(btn);</code>	<code>this.Controls.Add(btn);</code>
弹出一个窗体	<code>new Form2().ShowDialog();</code>	<code>new Window1().ShowDialog();</code>	<code>Reponse.Redirect("Fm2.aspx");</code>