C#程序设计及应用

唐大仕
dstang2000@263.net
北京大学

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

数据类型、常量与变量

第1节 数据类型、变量与常量

程序中的变量与常量

- 程序的基本任务是：对数据进行处理
- 数据分为变量（variable）与常量（literal）
- int age = 18;
- 变量是值可以改变，本质上是内存的空间，用来存储信息
- 常量的值是固定的，直接写出来的，称字面常量（literal）

变量声明和赋值

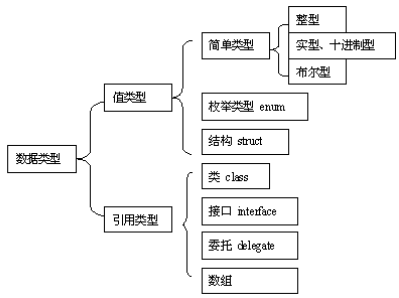
```
public class Test {
    public static void Main () {
        int x, y=8;
        double d = 3.1415;
        bool b = true;
        String s = "Hello";
        Person p = new Person();
    }
}
```


变量命名

- 变量命名(identifier，标识符）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。
- 如：
  - Age, age, personName, book1, book5, _num
- 注：C#是大小写敏感的

数据类型的概念

- int age = 5;
- Label1.Left += 5;
- Timer1.Enabled = true;
- Text1.Text = “Hello”;
- DateTime.Now
- 数据类型：本质上
- 是数据的存储方式及其能参与的运算的抽象



C#数据类型

- C#的数据类型分两大类
  - 值类型 (Value Type)
  - 引用类型 (Reference Type)
- 前者如： int, double, Point, Size, DateTime
- 后者如： Button, Label, Book, Person,

两种类型的区别

- int a = 5;
- int b = a;

- Person p = new Person();
- Person p2 = p;

- DateTime t = new DateTime(2018,1,1);
- DateTime t2 = t;

两种类型又是统一的

- 它们都是类型
  - 任何变量都有类型
    - int a= 5;
    - DateTime t = new DateTime(2020,12,31);
    - Button btn = new Button();
    - double.Parse ( )
- 它们都是Object
  - 它们都能 .ToString()
    - Label1.Text = age.ToString();
    - 都能用于字符串的连接 (+)

类型细分

- 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)

等价类型

- 注意：每种简单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)

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' 表示Unicode '\n' (回车)
- 实数类型
  - float 如 3.14F
  - double 如 3.14 1.5E-3 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) 不能用0或1代替

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
  - 是引用类型，但对字符串常量有特殊处理
  - “abcd1234“
  - @“abcd
  - Pqrst“
  - 字符串前可使用@，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
  - @“c:\windows\system32\aaa.txt”

使用数据类型要注意

- 针对C++程序员
  - 引用类型与值类型是由其类型定义的，而不是由其使用决定的
  - 如C++中 Book b;　Book * b;　Book &b;　Book *& b;
- 针对JavaScript程序员
  - 类型是严格的　TextBox1.Text = a.ToString();
- 针对VB程序员
  - int 为32位长
  - 字符 (char)与字符串(string)不同

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

运算符与表达式

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: +

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

逻辑运算符(1)

- 逻辑运算符功能

  ! -- 逻辑非 & -- 逻辑与 | -- 逻辑或

  ^ -- 逻辑异或 &&-- 短路与 || -- 短路或

- 逻辑运算符功能说明:

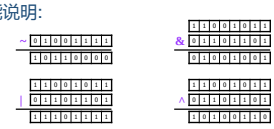
a	b	!a	a&b	a|b	a^b	a&&b	a||b
true	true	false	true	true	false	true	true
true	false	false	false	true	true	false	true
false	true	true	false	true	true	false	true
false	false	true	false	false	false	false	false

逻辑运算符(2)

- 条件逻辑运算符，也叫短路(short-circuit)逻辑运算符
- && 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
  - }

位运算符

- 位运算符功能
- ~ -- 取反 & -- 按位与 | -- 按位或 ^ -- 按位异或
- << 左移 >> 右移
- 位运算符功能说明:



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
```

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

赋值运算符

- 赋值运算符 =
- 扩展赋值运算符

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

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
  - (a-b)*c-4
- i<30 && i%10!=0
- m = a>b?a:b;
- 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

类型转换

- 赋值时
  - 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，只要不超出其表数范围

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


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

单目 > 算术 > 关系 > 逻辑 > 三目 > 赋值

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

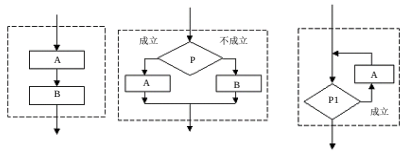
- 类型的转换
  - 字符串转成数值：
    - double.Parse(s) int.Parse(s)
  - 数字转成字符串：
    - 10.ToString()
    - “” + 10
  - 使用Convert
    - Convert.ToInt32(textbox1.Text)
    - Convert.ToDouble(“123.45”)
    - Convert.ToDateTime(“2009-10-01 14:00”)

流程控制语句

第3节 流程控制语句

结构化程序设计的三种基本流程

- 顺序 分支 循环



简单语句

- 最简单的语句：方法调用语句及赋值语句
  - 后面有个分号
- 如：
- System.Console.Write(“Hello World”);
- b = a>0?a:-a;
- s = TextBox1. Text;
- d = int.Parse (s);
- 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--if

- if(条件表达式)
- 语句块
- else
- 语句块

示例

- 例：**LeapYear.cs**

- if((year%4==0 && year%100!=0) || (year%400==0))
- Console.WriteLine(year+" 是闰年.");
- else
- Console.WriteLine(year+" 不是闰年.");

小技巧

- 注意书写格式，特别是缩进
- 将后面的花括号去掉，再重新输一下，就会自动排版
- 也可以按 Ctrl+E+D （格式文档）
- 或 Ctrl+E+F（格式化选中部分）

多分支语句-- switch语句

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

- 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;（除非几个case连起来，中间没有别的语句）

示例

例：**GradeLevel.cs**

例：AutoScore.cs自动出题并判分

例：屏保程序
注意：

示例

- GradeLevel.cs 分数不同的等级
  - 要点：switch

示例

- 简单的屏保程序
  - 要点：Form的BackColor, WindowStatus, FormBorder
  - deltX, deltY
  - 将exe文件改名为 xxxx.scr 并复制到c:\windows\system32

示例

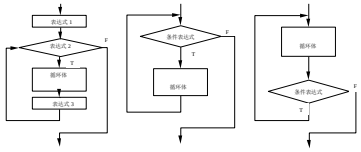
- AutoScore.cs自动出题并判分
  - 要点 if与switch
  - 核心是变量
  - 注意：对象的 “匈牙利命名法”
    - 如 btnNew btnJudge lblOP txtAnswer lstDisplay

循环语句

- 循环语句功能
  - 句的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）我称为 “循环改变”
  - "结束后处理"
- s=0; for(int i=1;i<=100; i++) s+=i;
- Console.Write(s);

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.Write("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.Write("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
  - 可以从内层跳到外层
  - 标号的写法： 标识符加个冒号(:)

示例

- 角谷猜想
- 要点：循环的五要素

示例

- 显示许多圆
- 要点： Graphics g = this.GreateGraphics();
- g.DrawEllipse(...)
- 如：
- g.DrawEllipse(
- new Pen(getRandomColor(),1),
- x0-r,y0-r, r*2, r*2
-);
-

数组

第4节 数组

数组概述

- 数组是多个相同类型数据的组合
- 数组属引用类型

一维数组声明

- 一维数组的声明方式：
- int[] a1; 注意方括号写到变量名的前面
- 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  = 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};   也可写为 int[] a = new int[]{ 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：
  - (数值类型是0， 引用类型是null)
    - 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
  - 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)；

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

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, source.Length );
```


多维数组

- 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];
```

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，这与C++是不一样的
```

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	

示例

- 筛法求素数
- 要点：使用bool数组

示例

- 排块游戏
- 要点
  - 按钮的数组
  - 按钮的生成、加入、事件
  - 按钮的Tag
  - 函数的书写
  - 注释的书写

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

练习C#语言基础

- 参考LanguageBasic目录下的示例
- 作业请见教学网站

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 btn.SetValue(Grid.RowProperty, 2); 如果在Window上点右键，布局，选Canvas，这样便于绝对布局 Canvas.SetLeft(this.Button1, 200);	对象的布局用css控制
控件背景及Color	xx.BackColor = Color.Red Color.FromArgb(...)	xx.Background = new SolidColorBrush(Color.FromRgb(120, 120, 255)); 其中可用Colors.Red	用css控制，或CssBackgroundColor等属性
控件可见性	xx.Visible = true;	xx.Visibility = Visibility.Visible或Hidden	xx.Visible = true;
控件文本	xx.Text = ...;	xx.Content = ...;	xx.Text = ..;
添加子控件	xx.Controls.Add(btn);	this.Grid1.Children.Add(btn); this.Canvas1.Children.Add(btn);	this.Controls.Add(btn);
弹出一个窗体	new Form2().ShowDialog();	new Window1().ShowDialog();	Repsonse.Redirect(“Fm2.aspx”);

关于界面控件的使用简介

- 在Windows设计界面上（Shift+F7)时，按工具箱(Ctrl+W,X)
- 其中有多个工具分类

国庆以后，采用翻转课堂

- 上课地点改到信科机房：理科一号楼1235
- 机房的第一次课我们先过渡
- 视频的网址在10月8日前放到 cf.pku.cn/tds/csharp网站上
- 使用翻转课堂的好处
  - 更加注重编程实践
  - 更好地支持个性化学习
  - 更好地支持合作化学习