单元二

储备 C#知识和技术

单元要点

- 常量和变量
- 数据类型之间的转换
- 运算符与表达式
- Console 类

技能目标

- 能编写正确的运算符和表达式
- 会声明和使用常量与变量
- 会在程序代码中进行数据类型转换

项目载体

- 工作场景项目：简单个人简历
- 工作训练营项目：简单计算器

2.1 工作场景导入

【工作场景】

小张才从某高校毕业，去一软件公司面试，公司主管要求做一个“简单个人简历”来介绍自己，设计效果如图 2.1 所示，说明个人姓名、年龄、学历、专业等信息，在线输入后显示信息。

图 2.1　工作场景示例

【引导问题】

（1）控制台如何接受用户的输入信息？

（2）如何进行数据类型之间的转换？

（3）如何实现控制台输入和输出？

2.2　技术与知识准备

2.2.1　基本概念

2.2.1.1　基本数据类型

数据类型是对各种数据形态的描述，如整型、浮点型等。在计算机中，数据类型的不同决定所占内存存储空间的大小，使用什么样的数据类型要根据实际情况而定，基本原则是既不要浪费存储空间又不要丢失数据。

C#提供了大量的内置数据类型，称为基本数据类型，与之对应的是用户自定义的复合数据类型。表 2.1 列举了 C#语言中的基本数据类型，以及与它们对应的.NET 框架的数据类型、大小和说明。

表 2.1　C#常用的基本数据类型

C#数据类型	大小（位）	说明
int	32	带符号的 32 位整型
float	32	单精度的浮点类型
double	64	双精度的浮点类型
string	取决于实现平台	指向字符串对象的引用
bool	8	逻辑值，true 或者 false

2.2.1.2　标识符

标识符是计算机语言里常用的一种术语，在 Visual C# 2010 中，变量、常量、函数、类等的命名必须遵循一定的规则，人们把符合这些规则的名称称为合法标识符，这些规则是：

（1）标识符必须由字母、数字、下划线组成，且只能以字母、下划线开头。

（2）如果以下划线开头，必须包括至少一个其他字符。

（3）不能是 Visual C# 2010 的关键字。Visual C# 2010 部分常用关键字如表 2.2 所示。

（4）Visual C# 2010 的标识符区分大小写。

表 2.2　部分常用关键字

abstract	return	new	struct	true	long
private	null	switch	base	case	ref
class	this	bool	false	for	using
throw	break	finally	try	else	interface
byte	virtual	override	catch	static	while
float	params	typeof	extern	if	void
enum	where	in	public	sealed	goto
namespace	protected	do	continue	double	readonly
string	const	foreach	int	char	decimal

2.2.1.3　常量

常量是指在程序运行的过程中，其值保持不变的量。Visual C# 2010 的常量包括符号常量、数值常量、字符常量、字符串常量、布尔常量等。采用 const 语句来声明常量，其语法格式为：

const <数据类型> <变量名>=<表达式>

例如 const float PI=3.14;

const int A1=4,PRICE=5;

说明：

（1）常量名遵循标识符的命名规则，一般采用大写字母。

（2）表达式由数值、字符、字符串常量及运算符组成，也可以包括前面定义过的常量，但是不能使用函数调用。

（3）如果多个常量的数据类型是相同的，可在同一行中声明这些常量，声明时应用逗号将它们隔开。

2.2.1.4　变量

变量是在程序运行过程中，其值可以改变的量，它表示数据在内存中的存储位置，每个变量都有一个数据类型，以确定哪些数据类型的数据能够存储在该变量中。在 Visual C# 2010 中，声明变量的语法格式为：

<数据类型><变量名>=<表达式>

例如 int a,b=2,c;

说明：

（1）<变量名>遵循 C#合法标识符的命名规则；

（2）“=<表达式>”为可选项，可以在声明变量时给其赋初值；

（3）一行可以声明多个相同类型的变量，且只需指定一次数据类型，变量和变量之间用逗号隔开。

2.2.1.5　数据类型之间的转换

在程序中，常常需要把一种数据类型的数据转化成另一种数据类型的数据，这个转换过程称为

类型转换。类型转换按照不同的转换方法可以分为隐式转换和显式转换。

1. 隐式转换

隐式转换的规则很简单：对于数值类型，任何类型 A，只要其取值范围完全包含在类型 B 的取值范围内，就可以隐式转换为类型 B。也就是说，int 类型可以隐式转换为 float、double 类型，float 类型隐式转换为 double 类型。

2. 显式转换

与隐式转换相反，当要把取值范围大的类型转换为取值范围小的类型时，就需要显式转换。

（1）字符串转换为数值型。

当要把字符串转换为数值类型时，可以使用 Parse()方法。不同的数值类型都有自己的 Parse()方法。

字符串转换为整型（string—int）：int.Parse(string);

字符串转换为单精度浮点型（string—float）：float.Parse(string);

字符串转换为双精度浮点型（string—double）：double.Parse(string);

注意：要转换的字符串必须是数字的有效表示形式。简单地讲就是表面上看起来是对应的数字，比如“32”转换为整数就是表面上看到的 32，但是不能把“age”转换成整数，因为它不是整数的有效表示形式。

（2）数值型转换为字符串。

很简单，只要使用 ToString()方法就行了。

例如 int age=18;

```
string myage=age.ToString();
```

即 myage="18"。

（3）使用 Convert 类进行类型转换。

前面讲的 Parse()方法是将字符串转换为数值型的，可以使用 Convert 类来实现各个基本类型之间的相互转换。Convert 类为每种类型转换都提供了一个静态方法，常见的方法见表 2.3。

表 2.3 常见的类型转换方法

方法	说明
Convert.ToInt32()	转换为整型（int 型）
Convert.ToSingle()	转换为单精度浮点型（float 型）
Convert.ToDouble()	转换为双精度浮点型（double 型）
Convert.ToString()	转换为字符串类型（string 型）

例如 float a=3.2;

```
int n=Convert.ToInt32(a);
```

即 n=3。

2.2.2 运算符与表达式

运算符是对一个或者多个返回值的代码元素执行运算的代码单元，Visual C# 2010 的运算符包括算术运算符、逻辑运算符、字符串连接运算符、关系运算符、赋值运算符、位运算符和自增自减

运算符等。

2.2.2.1 算术运算符

算术运算符用于完成算术运算，所涉及的操作对象有文本、常量、变量、表达式、函数调用以及属性调用等。算术运算符及相应的表达式示例见表 2.4。

表 2.4 算术运算符

运算	运算符	算术表达式示例	结果及说明
取负	-	-2	对数字 2 进行取负运算
加法	+	2+3	结果 5
减法	-	3-2	结果 1
乘法	*	2*3	结果 6
除法	/	6/2	结果 3
取余	%	5%2	结果 1

说明：

（1）“%”为求余运算符，求两个数相除后的余数。

（2）进行除法运算时，如果两个数为整数，则得到的结果也是整数，无条件的去掉小数部分，即不是采用四舍五入方法。如：9/2=4。

（3）算术运算符的优先级顺序由高到低依次为：-（取负）、*、/、+、-（减法）。

2.2.2.2 字符串连接运算符

字符串连接字符串的运算符只有一个，即“+”，它一般用于连接两个字符串，字符串连接表达式的结果为字符串型数据。例如：

"12"+"34"="1234"

"太平南路"+"26"+"号"="太平南路 26 号"

说明：

当“+”连接的对象中既有字符串又有数字时，则可以省略数字的字符串定界符（""），例如：

"太平南路"+"26"+"号"可以写成"太平南路"+26+"号"

这样写不会影响计算的结果，但一般并不提倡。

但注意省略数字的字符串定界符必须是在有其他字符串参与到该表达式的前提下，例如：

"2"+"3"不能写成 2+3

这样容易把“+”当成是算术运算符的加号，相加结果为 5。

2.2.2.3 关系运算符

关系运算符用于比较两个表达式之间的关系，比较的对象通常有数值、字符串和对象，关系运算的结果是一个 bool 值，即 true 或者 false。关系运算符及相应的表达式示例如表 2.5 所示。

表 2.5 关系运算符

运算符	测试关系	关系表达式示例	结果（假定 a 为 5）
==	等于	a==6	false
!=	不等于	a!=6	true

续表

运算符	测试关系	关系表达式示例	结果（假定 a 为 5）
>	大于	a>6	false
>=	大于等于	a>=6	false
<	小于	a<6	true
<=	小于等于	a<=6	true

2.2.2.4 逻辑运算符

逻辑运算符（也称布尔运算符）用于判断操作数之间的逻辑关系。逻辑表达式的值也是一个 bool 值，即 true 或者 false。逻辑运算符有：!（非）、&&（与）、||（或），如表 2.6 所示。其中，只有!（非）为一元运算符，其他均为二元运算符。

表 2.6 逻辑表达式真值表

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

说明：

（1）Visual C# 2010 无法自动将数值转换成逻辑值，例如!8，是非法的逻辑表达式；

（2）关系运算符不能用于比较逻辑型数据，例如：bool y=false;y<=true;这样的写法是错误的；

（3）优先级由高到低依次为：逻辑非、算术运算符、关系运算符、逻辑运算符（&&和||）。例如：2+3>5&&5<8 结果为 false。

2.2.2.5 赋值运算符

在 Visual C#中，赋值运算符有基本赋值运算符和复合赋值运算符两种，通常用于将表达式的值赋给一个变量。赋值运算符见表 2.7。

表 2.7 赋值运算符

运算符	赋值表达式示例	结果
=	a=5	a=5
+=	a+=5	a=a+5
-=	a-=5	a=a-5
*=	a*=5	a=a*5
/=	a/=5	a=a/5
%=	a%=5	a=a%5

2.2.2.6 自增自减运算符

Visual C#与 C/C++相同，保留了自增运算符（++）和自减运算符（--），它们是一元运算符，且操作对象只能是变量。

1. 自增运算

自增运算符（++）的作用是对变量的值加 1。

自增运算符（++）可以放在被操作变量的前面（称为前自增），也可以放在被操作变量的后面（称为后自增）。例如：

```
int x;
++x;                //前自增
x++;                //后自增
```

前自增与后自增有很大的区别，如果带有自增运算符（++）的变量出现在表达式中，前自增的执行过程是：先使变量的值加 1，再执行其他运算；而后自增是先执行其他运算，再使变量的值加 1。例如：

```
int x=2,y=3,z;
z=++x*y;
```

因为“++x”为前自增，即变量 x 先自加 1，变成 3 后再与 y 相乘，即表达式“++x*y”的值为 9，因此变量 z 的值为 9。

若将上述代码改为：

```
int x=2,y=3,z;
z=x++*y;
```

因为“x++”为后自增，即先进行乘法运算后，变量 x 的值才自加 1，因此变量 z 的值应该为 2*3，即 6。

2. 自减运算

自减运算符（--）的作用是对变量的值减 1。

自减运算符（--）可以放在被操作变量的前面（称为前自减），也可以放在被操作变量的后面（称为后自减）。

例如：

```
int x;
--x;                //前自减
x--;                //后自减
```

同样的，前自减与后自减也有很大的区别，如果带有自减运算符（--）的变量出现在表达式中，前自减的执行过程是：先使变量的值减 1，再执行其他运算；而后自减是先执行其他运算，再使变量的值减 1。例如：

```
int x=2,y=3,z;
z=--x*y;
```

因为“--x”为前自减，即变量 x 先自减 1，变成 1 后再与 y 相乘，即表达式“--x*y”的值为 3，因此变量 z 的值为 3。

若将上述代码改为：

```
int x=2,y=3,z;
z=x--*y;
```

此时“x--”为后自减，即先进行乘法运算后，变量 x 的值才自减为 1，因此变量 z 的值应该为 2*3，即 6。

2.2.2.7 运算符的优先级和结合顺序

前面介绍了 C#的一些常用的运算符，接下来看一下运算符的优先级。

1. 运算符的优先级

根据运算符所执行运算的特点和它们的优先级，可将它们归为一元运算符和括号、算术运算符、比较运算符、逻辑运算符、赋值运算符、后自增和后自减运算符。这些运算符的优先级顺序如表 2.8 所示。

表 2.8 运算符的优先级

级别	运算符
第一级	++、--（作为前缀）、!、()、+、-（取负）
第二级	*、/、%、+、-
第三级	<、>、<=、>=、==、!=
第四级	&&、||
第五级	=、*=、/=、%=、+=、-=
第六级	++、--（作为后缀）

2. 运算符的结合顺序

运算符的结合顺序分为左结合和右结合两种，在 C#中，所有的一元运算符（++、--作为后缀时除外）都是右结合的。而对于二元运算符，除了赋值运算符外，其他的都是左结合的。运算符的优先级和结合顺序可以通过小括号来控制。

2.2.3 Console 类

2.2.3.1 向控制台输出

利用 Console.WriteLine()方法输出有两种方式。

方式一：Console.WriteLine(要输出的值);

方式二：Console.WriteLine("格式字符串",变量列表);

方式一举例：

```
Console.WriteLine("hello");
Console.WriteLine("hello "+"jack");     //+ 字符串的连接符号
```

方式二举例：

```
string city="北京";
Console.WriteLine("欢迎你， {0}", city);
```

上面两句话的输出结果是什么呢？

该运行结果如图 2.2 所示。

图 2.2 运行结果

从运行结果已经想到了第二种方式是怎样输出的。在这种方式中，WriteLine()的参数由两部分组成：“格式字符串”和变量列表。这里面的“欢迎你，{0}”就是格式字符串，{0}叫做占位符，它占的就是后面的 city 变量的位置。在格式字符串中，依次使用{0}，{1}，{2}…代表要输出的变量，然后将变量依次排列在变量列表中，0 对应于变量列表的第 1 个变量，1 对应变量列表的第 2 个变量，2 对应变量列表的第 3 个变量，依次类推，这种方式要比用加号连接方便多了。

2.2.3.2　从控制台输入

与 Console.WriteLine()相对应，从控制台输入可以使用 Console.ReadLine()，Write 是写的意思，Read 是读的意思。语法如下：

string str=Console.ReadLine()

返回一个字符串，把它赋值给一个字符串变量。

如果要输入整型数据，只需要用 int.Parse()方法把字符串转换成整型。

例如：

int age=int.Parse(Console.ReadLine())

2.3　回到工作场景

通过对以上 2.2 节内容的学习和演练，了解了常量与变量的定义、运算符和表达式以及 Console 类的方法，现在回到 2.1 节工作场景的任务。

定义 4 个变量 name、age、education、professional 来接受用户的输入，分别代表姓名、年龄、学历、专业，输出时用字符串的连接符号“+”。

程序代码如下：

```
Console.Write("你的姓名：");
string name = Console.ReadLine();
Console.Write("你的年龄：");
int age =int.Parse( Console.ReadLine());
Console.Write("你的学历：");
string education =Console.ReadLine();
Console.Write("你的专业：");
string professional = Console.ReadLine();
Console.WriteLine("以下是你的个人信息：");
Console.WriteLine("你的姓名："+name);
Console.WriteLine("你的年龄："+age);
Console.WriteLine("你的学历："+education);
Console.WriteLine("你的专业："+professional);
```

程序运行结果如图 2.1 所示。

注意：

（1）输出语句中“+”为字符串的连接符号。

（2）age 要进行类型的转换，Console.ReadLine 输入的是字符串，要转换成整型赋给 age。

2.4　工作训练营

2.4.1　项目训练

1. 训练内容

编写程序。问题描述如下：编写一个“简单计算器”，能够实现加、减、乘、除、取余功能。

2. 训练目的

掌握各类运算符的使用及类型的转换。

3. 训练过程

仔细理清题目要求，选择合适的运算符，设计出程序的答题步骤和具体的代码处理过程，最后编写出完整的 C#语言源程序。

分析：根据题目要求理清处理步骤。

【步骤 1】新建一个 Windows 窗体应用程序，取名为 ch02-1。

【步骤 2】拖动一个 Label、三个 TextBox、一个 Button、一个 ComboBox 到窗体上，界面如图 2.3 所示，属性设置如表 2.9 所示。

图 2.3　工作实训营项目窗体

表 2.9　控件属性设置

控件类型	控件名称	属性	设置结果
Form	Form1	Text	计算器
Label	Label1	Text	简单计算器
TextBox	TextBox1	Name	txtA
TextBox	TextBox2	Name	txtB
TextBox	TextBox3	Name	txtC
ComboBox	ComboBox1	Items	+ − * / %
		Name	cboChar
Button	Button1	Name	btnCal
		Text	=

【步骤 3】双击 Button 按钮，打开代码窗口，编写代码。

```
public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();
    }
```

```
        private void btnCal_Click(object sender, EventArgs e)
        {
            int n1=int.Parse(txtA.Text);//转化成整型
            int n2=int.Parse(txtB.Text);
            int n3 = 0;
            switch (CboChar.Text.Trim())
           {
               case "+": n3 = n1 + n2; break;//加法
               case "-": n3 = n1 - n2; break;//减法
               case "*": n3 = n1 * n2; break;//乘法
               case "/": if (n2 == 0) MessageBox.Show("除数不能为 0");else n3 =n1/n2; break;
               //判断除数不能为 0.
            }
               txtC.Text = n3.ToString();
            }
        }
    }
```

上面的程序用到了 switch 语句，另外，要注意类型的显式转换。

运行结果如图 2.4 所示。

图 2.4　训练内容的运行结果

4. 技术要点

熟练掌握运算符的使用，特别要注意类型的强制转换。

2.4.2　工作实践常见问题解析

【问题 1】隐式转换的适用场合。

【答】低精度向高精度的转换可以是隐式转换，例如 int 转变为 float，float 转变为 double。

【问题 2】不懂常量和变量的区别。

【答】常量就是不变的量，在整个生命期内，不能通过赋值方式来给常量赋值，变量就是可以改变的量，可以通过赋值等方式来给变量赋值。

【问题 3】Write、WriteLine 的区别。

【答】Write 是输出后不换行，WriteLine 是输出后换行。

【问题 4】“/”是除法运算，实数除以整数，结果是整数还是实数。

【答】整数除以整数，结果是整数，而且是无条件地去除小数部分，整数除以实数，或者实数除以整数，结果都是实数。

小结

本单元主要介绍了 C#语法中的一些基础知识，了解简单数据类型、变量和常量、运算符与表达式等，通过“简单个人简历”和工作训练营项目“简单计算器”整体训练了表达式、运算符和 Console 类的使用方法，现在开始已经享受到了设计程序项目的乐趣，下面将进入复杂一点的循环程序项目熟练阶段，再接再厉吧！