第3章

基本数据类型及运算——求圆的面积和周长


视频讲解

本章将建立一个项目，此项目用于求圆的面积和周长。通过此项目的创建、编写、运行和测试，初步掌握 C# 的数据类型、各种变量的声明方式、运算符的优先级、运算符与表达式的使用方法。

3.1 项目案例功能介绍

创建一个 GetAreaAndCircumference 控制台应用程序项目，此项目用于求圆的面积和周长。在项目中，定义三个 double 类型的变量 dblRadius、dblArea 和 dblCircumference，定义一个 PI 常量；dblArea = PI * dblRadius * dblRadius，dblCircumference = 2 * PI * dblRadius。其中，dblRadius 的数值是在程序运行时输入；把运算结果 dblArea 和 dblCircumference 输出到控制台中。

3.2 项目设计思路

在本项目中，项目设计思路包括以下步骤。

(1) 项目创建。

(2) 程序代码编制。

(3) 系统运行与效果测试。

3.3 关键技术

3.3.1 声明变量

变量是程序运行过程中用于存放数据的存储单元。变量的值在程序的运行过程中是可以改变的。

1. 变量的定义

在定义变量时，首先必须给每一个变量起名，称为变量名，以便区分不同的变量。在计算机中，变量名代表存储地址。变量名必须是合法的标识符。为了保存不同类型的数据，除了变量名之外，在定义变量时，还必须为每个变量指定数据类型，变量的类型决定了存储在变量中的数值的类型。对于一个变量的定义，变量名和变量类型缺一不可。C＃中，采用如下格式定义一个变量。

```
类型标识符 变量名 1, 变量名 2, 变量名 3, …
```

变量定义如下例所示。

```
int i, j, k;              //同时声明多个 int 类型相同的变量,在类型的后面用逗号分隔变量名
float fSum;
string strName, strAddress;
```

注意：任何变量在使用前，必须先定义，后使用。

2. 变量的赋值

变量是一个能保存某种类型的具体数据的内存单元，可以通过变量名来访问这个具体的内存单元。变量的赋值，就是把数据保存到变量中的过程。给一个变量赋值的格式如下。

```
变量名 = 表达式;
```

这里的表达式同数学中的表达式是类似的，如 9＋10、4＋a－c 都是表达式。单个常数或者变量，也可以构成表达式。由单个常数或者变量构成的表达式的值，就是这个常数或者变量本身。变量赋值的意义是：首先计算表达式的值，然后将这个值赋予变量。例如，定义了两个 double 类型的变量 dblTotalScore、dblAverageScore 和一个 int 类型的变量 nStudentCount：

```
double dblTotalScore,dblAverageScore;
int nStudentCount;
```

下面给 dblTotalScore、nStudentCount 赋值，应该写成：

```
dblTotalScore = 2000;
nStudentCount = 20;
```

3. 变量的初始化

在定义变量的同时，也可以对变量赋值，称为变量的初始化。在 C＃中，对变量进行初始化的格式如下。

```
类型标识符 变量名 = 表达式;
```

例如：

```
int nStudentCount = 150;          //定义一个 int 类型变量 nStudentCount,并将其赋予初始值为 150
```

3.3.2 声明常量

常量是指那些基于可读格式的固定数值，在程序的运行过程中其值是不可改变的。通过关键字 const 来声明常量，其格式如下。

```
const 类型标识符 常量名 = 表达式;
```

类型标识符指示所定义常量的数据类型，常量名必须是合法的标识符，在程序中通过常量名来访问该常量，如下例所示。

```
const double PI = 3.14159265;
```

上面的语句定义了一个 double 类型的常量 PI，它的值是 3.141 592 65。

常量具有如下特点。

(1) 程序中，常量只能被赋予初始值。一旦赋予一个常量初始值，这个常量的值在程序运行过程中就不允许改变，即无法对一个常量赋值。

(2) 定义常量时，表达式中的运算符对象只允许出现常量，不能有变量存在。

例如：

```
int a = 20;
const int b = 30;
const int c = b + 25;                    //正确，因为 b 是常量
const int k = a + 45;                    //错误，表达式中不允许出现变量
c = 150;                                 //错误，不能修改常量的值
```

3.3.3　基本数据类型的转换

基本类型转换，是把数据从一种类型转换为另一种类型。在 C# 中，类型转换有两种形式：隐式转换和显式转换。

1. 隐式转换

隐式转换是 C# 默认的以安全方式进行的转换，不会导致数据丢失。例如，从小的整数类型转换为大的整数类型，从派生类转换为基类。

例如，int 转换成 double：

```
int a = 3;
double b = a;
Console.WriteLine(b);                    //b = 3
```

2. 显式转换

显式类型转换，即强制类型转换。显式转换需要强制转换运算符，而且强制转换会造成数据丢失。

1) double 转 int

```
double a = 2.35;
int b = (int)a;
Console.WriteLine(b);                    //b = 2
```

2) string 转 int 或 double (Parse 方法)

```
string a = "123123";
string b = "123.123";
int c = int.Parse(a);
double d = double.Parse(b);
Console.WriteLine(c);                    //c = 123123
Console.WriteLine(d);                    //d = 123.123
```

3）任何类型转 string 类型(.ToString()方法)

```
int a = 123;
double b = 1.23;
string c = a.ToString();
string d = b.ToString();
Console.WriteLine(c);                    //c = 123
Console.WriteLine(d);                    //d = 1.23
```

3.3.4 运算符和表达式

运算符是表示各种不同运算的符号，运算符和运算紧密相关。表达式由变量、常数和运算符组成，是用运算符将运算对象连接起来的运算式，是基本的对数据进行运算和加工的表示形式。表达式的计算结果是表达式的返回值。使用不同的运算符连接运算对象，其返回值的类型是不同的。

1. 运算符

根据运算符所要求的操作数的个数，运算符分为“一元运算符”“二元运算符”和“多元运算符”。一元运算符是指只有一个操作数的运算符，如“＋＋”运算符、“－－”运算符等。二元运算符是指有两个操作数的运算符，如“＋”运算符、“＊”运算符等。在 C＃中，还有一个三元运算符，即“?:”运算符，它有三个操作数。

根据运算的类型，运算符又分为以下几类：算术运算符、赋值运算符、关系运算符、逻辑运算符、条件运算符和其他运算符。

2. 算术运算符

算术运算符用于对操作数进行算术运算，C＃中的算术运算符及其功能如表 3-1 所示。

表 3-1 C＃算术运算符

运算符	意　义	运算对象数目	示　例
＋	取正或加法	1 或 2	＋12、12＋20＋i
－	取负或减法	1 或 2	－3、a－b
＊	乘法	2	i＊j、8＊5
/	除法	2	10/5、i/j
％	模(也可称为取余运算符。如 7％3 的结果等于 1)	2	10％5、i％j
＋＋	自增运算	1	i＋＋、＋＋i
－－	自减运算	1	i－－、－－i

3. 赋值运算符

赋值运算符用于将一个数据赋予一个变量，赋值操作符的左操作数必须是一个变量，赋值结果是将一个新的数值存放在变量所指示的内存空间中。常用的赋值运算符如表 3-2 所示。

表 3-2 C＃的赋值运算符

符号	描　述	举　例
＝	赋值	x＝1
＋＝	加法赋值	x＋＝1 等价于 x＝x＋1

续表

符号	描　　述	举　　例
－＝	减法赋值	x－＝1 等价于 x＝x－1
＊＝	乘法赋值	x＊＝1 等价于 x＝x＊1
/＝	除法赋值	x/＝1 等价于 x＝x/1
%＝	取模赋值	x%＝1 等价于 x＝x%1
&＝	AND 位操作赋值	x&＝1 等价于 x＝x&1
|＝	OR 位操作赋值	x|＝1 等价于 x＝x|1
^＝	XOR 位操作赋值	x^＝1 等价于 x＝x^1
＞＞＝	右移赋值	x＞＞＝1 等价于 x＝x＞＞1
＜＜＝	左移赋值	x＜＜＝1 等价于 x＝x＜＜1

其中，“＝”是简单的赋值运算符，它的作用是将右边的数值赋值给左边的变量，数值可以是常量，也可以是表达式。例如，x＝18 或者 x＝10－x 都是允许的，它们分别执行了一次赋值操作。

除了简单的赋值运算符之外，其他的赋值运算符都是复合的赋值运算符，是在“＝”之前加上其他运算符。复合赋值运算符的运算很简单，例如，x＊＝10 等价于 x＝x＊10，它是对变量进行一次自乘操作。复合赋值运算符的结合方向为自右向左。在 C＃中，可以对变量进行连续赋值，此时，赋值操作符是右关联的，这意味着从右向左运算符被分组。例如，x＝y＝z 等价于 x＝(y＝z)。

4. 表达式

表达式是类似于数学运算中的表达式，是由运算符、操作数和标点符号按照一定的规则连接而成的式子。根据运算符类型的不同，表达式可以分为算术表达式、赋值表达式、关系表达式、逻辑表达式以及条件表达式等。表达式在经过一系列运算后得到一个结果，这就是表达式的结果。结果的类型由参加运算的操作数据的数据类型决定。

在包含多种运算符表达式求值时，如果有括号，先计算括号里面的表达式。在运行时各运算符执行的先后次序由运算符的优先级别和结合性确定。先执行运算优先级别高的运算，然后执行运算优先级别低的。C＃中各个运算符的优先级如表 3-3 所示。

表 3-3　运算符的优先级(从高到低)

类　　别	运　算　符
基本运算符	(x)　x.y　f(x)　a[x]　x＋＋　x－－　new　typeof　sizeof　checked　unchecked
一元运算符	＋　－　！　～　＋＋x　－－x　(T)x
乘/除运算符	＊　/　%
加/减运算符	＋　－
移位运算符	＜＜　＞＞
关系运算符	＜　＞　＜＝　＞＝　is　as
比较运算符	＝＝　！＝
按位与运算符	&
按位异或运算符	^
按位或运算符	|
逻辑与运算符	&&
逻辑或运算符	||
三元运算符	?：
赋值运算符	＝　＊＝　/＝　＋＝　－＝　＜＜＝　＞＞＝　&＝　^＝　|＝

3.3.5 简单数据的输入与输出

控制台(Console)的输入/输出主要通过命名空间 System 中的 Console 类来实现的,它提供了从控制台读写字符的基本功能。控制台输入主要通过 Console 类的 Read()方法和 ReadLine()方法来实现,控制台输出主要通过 Console 类的 Write()方法和 WriteLine()方法来实现。

其中,WriteLine()方法的作用是将信息输出到控制台,同时 WriteLine 方法在输出信息的后面添加一个回车换行符,用来产生一个新行。Write()方法和 WriteLine()方法类似,都是将信息输出到控制台,但是输出到屏幕后并不会产生一个新行。

ReadLine()方法用来从控制台读取一行数据,一次读取一行字符的输入,并且直到用户按下 Enter 键它才会返回。但是,ReadLine()方法并不接收回车键。如果 ReadLine()方法没有接收到任何输入,或者接收了无效的输入,那么 ReadLine()方法将返回 null。Read()方法的作用是从控制台的输入流读取下一个字符,Read()方法一次只能从输入流读取一个字符,并且直到用户按回车键才会返回。

3.4 项目实践

3.4.1 项目创建

项目创建的具体步骤如下。

(1) 启动 VS.NET。

(2) 创建一个 C# 控制台应用程序。首先选择“文件”→“新建”→“项目”命令,打开“新建项目”对话框。

(3) 在弹出的对话框中选择“控制台应用程序”模板,设置相应的项目名称与保存位置,单击“确定”按钮。创建完成的项目界面如图 3-1 所示。

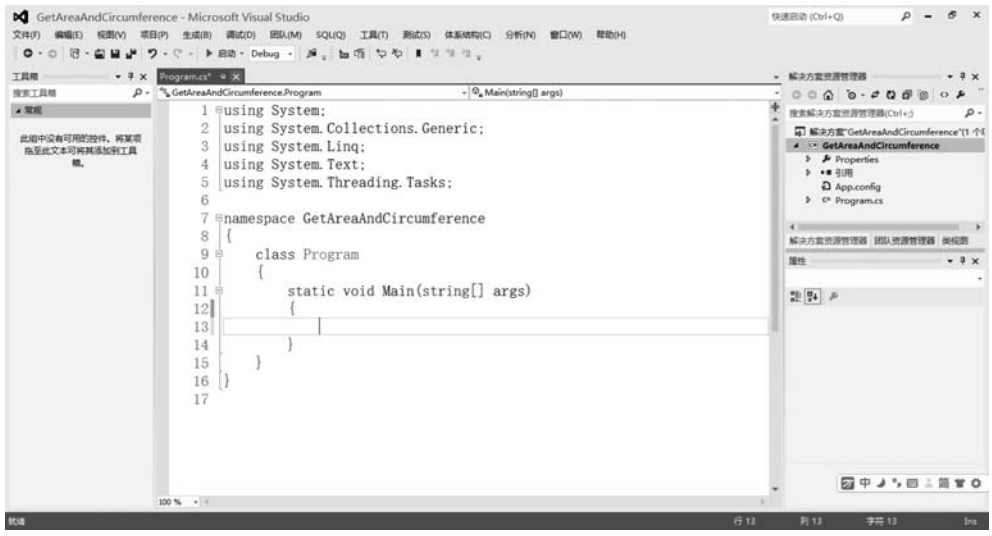

图 3-1 创建完成的项目界面

3.4.2　程序代码设计

程序代码设计，具体实现过程如下。

1. 程序代码设计

根据项目的描述，可在代码编辑器中完成如下代码的添加。

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Linq;
using System.Threading.Tasks;

namespace GetAreaAndCircumference
{
    class Program
    {
        static void Main(string[] args)
        {
            //声明变量
            double dblRadius, dblArea, dblCircumference;
            //声明常量
            const double PI = 3.1415926;
            //定义字符串变量，接收输入数据
            string strInput;

            Console.WriteLine("请输入圆的半径: ");
            //接收用户输入
            strInput = Console.ReadLine();

            //将接收过来的数据转换为浮点数
            dblRadius = double.Parse(strInput);

            //计算圆的面积和周长
            dblArea = PI * dblRadius * dblRadius;
            dblCircumference = 2 * PI * dblRadius;

            Console.Write("圆的面积为: {0}", dblArea + "\n");
            Console.WriteLine("圆的周长为: " + dblCircumference);
        }
    }
}
```

2. 代码分析

(1) 程序中，首先声明三个浮点类型的变量，一个浮点类型的常量以及一个用于接收数据的字符串变量。

(2) 在控制台中，接收用户输入，并通过强制类型转换方法 double.Parse()实现把字符串变量转换为浮点数。

(3) 通过表达式，计算圆的面积和周长。

(4) 输出计算结果。注意代码中包括两个输出语句,这两个输出语句是不同的。前面的一行代码通过转义字符“\n”来实现换行,而后面的一个输出语句则是通过 Console.WriteLine()来实现换行的。

3.4.3 项目运行

项目运行界面如图 3-2 所示。

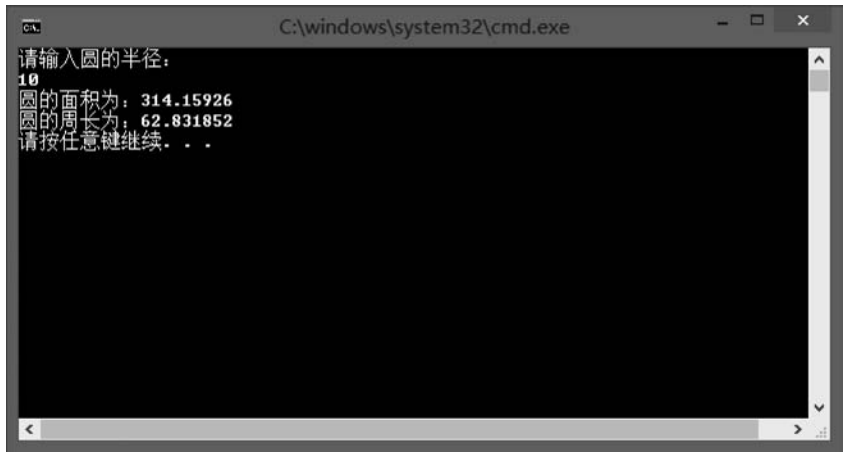

图 3-2 项目运行界面

3.5 小结

此项目主要涉及 C# 的设计基础。其中,标识符是表示某一对象的具体名称,其定义与使用必须符合一定的规范,最好做到见名知义。变量与常量的名称属于标识符范畴,其声明要避免相同作用域内同名现象的出现。变量、常量与一定的数据类型相关联。因此,变量和常量的运算过程涉及精度大小、数据类型转换等相关问题。

计算机在对程序做处理时,涉及运算符和表达式。运算过程中,编译器将根据运算符优先级次序进行先后运算,并返回表达式的最终运算值。

3.6 练一练

(1) 设长方形的高为 1.5,宽为 2.5,求该长方形的周长和面积。

(2) 编写一个控制台程序,输入两个整数,输出这两个整数的和、差、积和商。