

学习目标：

- 理解文件与流的基本概念；
- 了解文件与目录的基本知识及其常用的类；
- 掌握文件流读写方法。

5.1 概 述

文件是计算机管理数据的基本单位，同时也是应用程序保存和读取数据的一个重要场所。C#中文件处理技术称为 I/O 技术，即输入与输出技术，或称为流处理技术或文件流处理技术。

5.1.1 文件与流

文件和流是既有区别又有联系的两个概念。

文件是指在各种存储介质上永久保存数据的有序集合，并与一个具体的名称对应，它是进行文件读写操作的基本对象。从严格意义上讲，文件指的是放在磁盘上的静态信息，这种信息不是连续的，是随机的。

流是字节序列的抽象概念，流提供一种向后续存储器写入字节或从后续存储器读取字节的方法。流一般指的是连续的字节信息。例如，要对一个文件进行处理，这个文件就会变成连续的字节信息加载到内存中，也就是说文件在处理时必须变成流。流强调的是动态的连续信息，是由文件转换成的。流和文件指的都是一件事物，但是状态不一样。

文件是存储在存储介质上的数据集，是静态的，具有名称和相应路径。当打开一个文件并对其进行读写时，该文件就成为流。

5.1.2 System.IO 命名空间

System.IO 命名空间包含允许读写文件和数据流的类型以及提供基本文件和目录支持的类型，因此在使用这些类时需要引入 System.IO 命名空间。System.IO 命名空间中的常用类如表 5-1 所示。

System.IO 命名空间中的常用类大致分为操作目录的类、操作文件的类、文件读写类等。其中，Directory 类和 DirectoryInfo 类属于操作目录的类；File 类和 FileInfo 类属于操作文件的类，StreamReader 类和 StreamWriter 类属于文本文件读写的类；BinaryReader 类和 BinaryWriter 类属于二进制文件读写的类。

表 5-1 System.IO 命名空间的类

类 名	功能和用途
Directory、DirectoryInfo	创建、删除并移动目录，通过属性获取特定目录的相关信息
File、FileInfo	创建、删除并移动文件，通过属性获取特定文件的相关信息
StreamReader、StreamWriter	读写文本数据信息
BinaryReader、BinaryWriter	读写二进制数据

5.2 目 录 操 作

在程序开发中，有时需要对文件目录进行操作，如创建目录、删除目录等，为此 C# 提供了 Directory 类和 DirectoryInfo 类。

5.2.1 Directory 类

Directory 类是静态类，提供了许多静态方法用于对目录进行操作，如创建、删除和移动目录等。Directory 类的一些常用方法如表 5-2 所示。

表 5-2 Directory 类的常用方法

方 法	说 明
CreateDirectory()	创建指定路径的目录
Exists()	判断目录是否存在
GetDirectoryRoot()	获取指定目录的根目录
GetDirectories()	获取当前目录下的 Directory 对象数组
GetFiles()	获取当前目录下的 File 对象数组
Delete()	删除指定目录及其目录下的所有文件
Move()	将指定目录移动到新的位置

注意：Directory 的 Delete()方法是永久删除，不把目录送到回收站；使用 Move()方法移动目录时，要注意不能跨磁盘移动，如 C 盘的文件不能移到 D 盘下。

【例 5-1】 目录创建程序。

【操作步骤】

- (1) 启动 VS，新建一个 Windows 窗体应用程序 DirectoryApplication。
 - (2) 双击 Form1.cs，切换到设计视图，从工具栏中拖曳 1 个 Label 控件、1 个 TextBox 控件和 1 个 Button 控件到窗体设计区，并调整控件大小进行布局。
 - (3) 在窗体设计区中右击窗体 Form1 和每一个控件，设置窗体和控件的相关属性。
- 表 5-3 列出了窗体及控件属性。

表 5-3 窗体及控件属性设置

窗体和控件	属 性	属 性 值
Form1	Text	创建目录
label1	Name	请输入创建目录名称：
textBox1	Name	txtDirName
button1	Name	btnMake
	Text	创建

(4) 双击“创建”按钮，为其添加单击事件处理程序，程序代码如下：

```
private void btnMake_Click(object sender, EventArgs e)
{
    string path = txtDirName.Text;
    if (Directory.Exists(path))
    {
        MessageBox .Show (path + "目录已经存在");
    }
    else
    {
        Directory.CreateDirectory(path);
        MessageBox.Show(path + "目录创建成功");
    }
}
```

以上代码的功能是，首先定义一个 string 类型的变量 path 保存用户输入的目录，然后调用 Directory 类的 Exists()方法查找该目录是否存在，Exists()方法的参数是用户输入的目录，即变量 path，如果存在，提示“目录已经存在”；如果不存在，则调用 Directory 类的 CreateDirectory()方法创建该目录，并提示“目录创建成功”。

(5) 在解决方案资源管理器中右击 DirectoryApplication 项目，将其设为启动项目。

(6) 编译并运行，运行结果如图 5-1 所示。

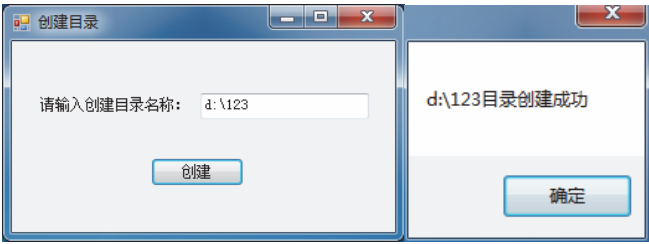


图 5-1 DirectoryApplication 项目运行结果

5.2.2 DirectoryInfo 类

DirectoryInfo 类的功能与 Directory 类相似，不同的是的 DirectoryInfo 类是一个实例类，所有方法都是实例方法。也就是说，要想使用 DirectoryInfo 类提供的方法必须实例化一个属于 DirectoryInfo 类的对象。因此，如果需要对同一个目录进行多次重复操作时，应该考虑使用 DirectoryInfo 类的实例方法。DirectoryInfo 类不仅拥有与 Directory 类功能相似的方法，而且还具有一些特有的属性，如表 5-4 所示。

表 5-4 DirectoryInfo 类的常用属性

属 性	说 明
Name	获取当前 DirectoryInfo 对象的名称
Root	获取路径的根目录
Parent	获取指定子目录的父目录
FullName	获取目录或文件的完整目录
Exists	判断指定目录是否存在

【例 5-2】 目录浏览程序。

【操作步骤】

(1) 启动 VS，新建一个 Windows 窗体应用程序 DirectoryInfoApplication。

(2) 双击 Form1.cs，切换到设计视图，从工具栏中拖曳 2 个 GroupBox 控件、4 个 Label 控件、4 个 TextBox 控件、1 个 ListBox 控件和 1 个 Button 控件到窗体设计区，调整控件大小进行布局。

(3) 在窗体设计区中右击窗体 Form1 和每一个控件，设置窗体和控件的相关属性。表 5-5 列出了窗体及控件属性。

表 5-5 窗体及控件属性设置

窗体和控件	属 性	属 性 值
Form1	Text	浏览目录
label1	Name	请输入目录名称:
textBox1	Name	txtDirName
label2	Text	目录名称:
textBox2	Name	txtName
label3	Text	根目录名称:
textBox3	Name	txtRoot
label4	Text	父目录名称:
textBox4	Name	txtParent
groupBox1	Text	详细信息
groupBox2	Text	目录列表
button1	Name	btnBrowse
	Text	浏览
listBox1	Name	listGetDirectories

(4) 定义一个 dirList()方法，用于在列表框中循环输出指定目录中的所有子目录。

```
public void dirList(DirectoryInfo dir)
{
    listGetDirectories.Items.Clear();
    DirectoryInfo[] dirs = dir.GetDirectories();
    for (int i = 0; i < dirs.Length; i++)
    {
        listGetDirectories.Items.Add(dirs[i]);
    }
}
```

以上代码的功能是，封装一个 dirList()方法，该方法没有返回值，有一个 DirectoryInfo 类型的参数，方法体中首先清空列表框，并通过 DirectoryInfo 对象的 GetDirectories()方法获得目录中的所有子目录，所有子目录构成一个 DirectoryInfo 类型的数组，然后在列表框中循环输出该数组中的每一个元素的值。

(5) 双击“浏览”按钮，为其添加单击事件处理程序，程序代码如下：

```
private void btnBrowse_Click(object sender, EventArgs e)
```

```
{
    string path = txtDirName.Text;
    if (!string.IsNullOrEmpty(path))
    {
        DirectoryInfo dir = new DirectoryInfo(path);
        txtName.Text = dir.Name;
        txtRoot.Text = dir.Root.ToString();
        txtParent.Text = dir.Parent.ToString();
        //输出子目录
        dirList(dir);
    }
    else
    {
        MessageBox.Show("请输入目录名称!");
    }
}
```

以上代码的功能是，首先定义一个 `string` 类型的变量 `path`，用来存储用户输入的目录，然后通过调用 `string` 的 `IsNullOrEmpty()` 方法判断 `path` 值是否为空，即判断用户是否已输入目录名称，如果不为空，则创建该目录的 `DirectoryInfo` 对象，并在相应文本框中输出用户指定目录的名称、根目录名称、父目录名称，通过调用步骤（4）中封装的 `dirList()` 方法输出所有子目录。

（6）在解决方案资源管理器中右击 `DirectoryInfoApplication` 项目，将其设为启动项目。

（7）编译并运行，运行结果如图 5-2 所示。

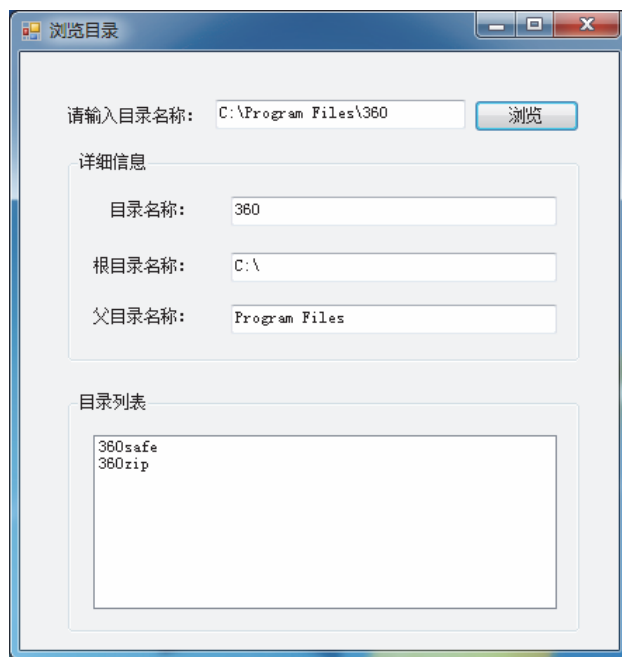


图 5-2 DirectoryInfoApplication 项目运行结果

5.3 文件操作

File 类和 FileInfo 类主要提供与文件有关的各种操作，包括创建、复制、移动、删除文件等。

5.3.1 File 类

File 类是一个静态类，提供了许多静态方法，用于处理文件。File 类的常用方法如表 5-6 所示。

表 5-6 File 类的常用方法

方 法	说 明
Create()	创建文件
Open()	打开指定路径上的文件，返回 FileStream 对象
Copy()	将文件复制到指定位置
Move()	将指定文件移动到新位置
Delete()	删除文件
Exists()	判断指定文件是否存在

注意：Directory 和 File 提供的方法都是共享方法，如果执行一次操作，使用共享方法的效率较高；但如果针对一个目录或文件多次操作，可以考虑使用 DirectoryInfo 和 FileInfo 提供的实例方法。

【例 5-3】 文件删除程序。

【操作步骤】

(1) 启动 VS，新建一个 Windows 窗体应用程序 FileApplication。

(2) 双击 Form1.cs，切换到设计视图，从工具栏中拖曳 1 个 Label 控件、1 个 TextBox 控件和 1 个 Button 控件到窗体设计区，并调整控件大小进行布局。

(3) 在窗体设计区中右击窗体 Form1 和每一个控件，设置窗体和控件的相关属性。

表 5-7 列出了窗体及控件属性。

表 5-7 窗体及控件属性设置

窗体和控件	属 性	属 性 值
Form1	Text	删除文件
label1	Name	请输入删除文件名称：
textBox1	Name	txtDirName
button1	Name	btnDelete
	Text	删除

(4) 双击“删除”按钮，为其添加单击事件处理程序，程序代码如下：

```
private void btnDelete_Click(object sender, EventArgs e)
{
    string path = txtDirName.Text;
```

```
        if (File.Exists(path))
        {
            File.Delete(path);
            MessageBox.Show("文件删除完毕!");
        }
        else
        {
            Console.WriteLine("文件不存在!");
        }
    }
}
```

以上代码功能是,通过 File 类的 Exists()方法判断用户输入的文件是否存在,如果存在,通过调用 File 类的 Delete()方法删除该文件,并提示用户“文件删除完毕!”;如果不存在,提示“文件不存在!”。

(5) 在解决方案资源管理器中右击 FileApplication 项目,将其设为启动项目。

(6) 编译并运行,运行结果如图 5-3 所示。

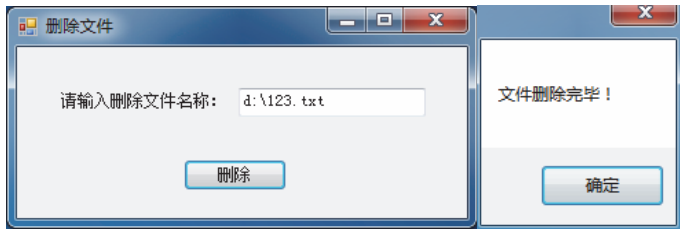


图 5-3 FileApplication 项目运行结果

5.3.2 FileInfo 类

FileInfo 类与 File 类类似,它们都可以对磁盘上的文件进行操作。不同的是 FileInfo 类是实例类,所有的方法必须实例化对象后才能调用。FileInfo 类除了拥有与 File 类相似的方法外,同时也有它特有的属性,如表 5-8 所示。

表 5-8 FileInfo 类的常用属性

属 性	说 明
Directory	获取父目录的实例
DirectoryName	获取表示目录的完整路径的字符串
FullName	获取目录或文件的完整目录
Length	获取当前文件的大小

【例 5-4】 文件浏览程序。

【操作步骤】

(1) 启动 VS,新建一个 Windows 窗体应用程序 FileInfoApplication。

(2) 双击 Form1.cs,切换到设计视图,从工具栏中拖曳 1 个 GroupBox 控件、3 个 Label 控件、3 个 TextBox 控件和 1 个 Button 控件到窗体设计区,调整控件大小进行布局。

(3) 在窗体设计区中右击窗体 Form1 和每一个控件,设置窗体和控件的相关属性。

表 5-9 列出了窗体及控件属性。

表 5-9 窗体及控件属性设置

窗体和控件	属 性	属 性 值
Form1	Text	浏览文件
label1	Name	请输入文件名称:
textBox1	Name	txtFileName
label2	Text	文件当前目录:
textBox2	Name	txtDir
label3	Text	文件大小:
textBox4	Name	txtSize
groupBox1	Text	详细信息
button1	Name	btnBrowse
	Text	浏览

(4) 双击“浏览”按钮，为其添加单击事件处理程序，程序代码如下：

```
private void btnBrowse_Click(object sender, EventArgs e)
{
    FileInfo file = new FileInfo(textBox1.Text);
    if (file.Exists)
    {
        textBox2.Text = file.Directory.ToString();
        textBox4.Text = file.Length.ToString();
    }
    else
    {
        file.Create();
        MessageBox.Show ("文件已经创建成功!");
    }
}
```

以上代码的功能是，首先创建 `FileInfo` 类的对象 `file`，`FileInfo` 类的构造方法中有一个参数，即用户输入的文件名，通过调用对象 `file` 的 `Exists` 属性判断该文件是否存在，如果用户指定的文件存在，则输出文件的当前目录和文件大小；如果文件不存在，则通过调用对象的 `Create()` 方法创建该文件，并提示“文件已经创建成功!”。

(5) 在解决方案资源管理器中右击 `FileInfoApplication` 项目，将其设为启动项目。

(6) 编译并运行，运行结果如图 5-4 所示。



图 5-4 FileInfoApplication 项目运行结果

5.4 文件读写

数据流（也称为流）是一个用于传输数据的对象。数据的传输有两个方向，如果数据是从外部源传输到程序中，就称为读取流；如果数据是从程序传输到外部源，就称为写入流。

在 System.IO 命名空间中，对文件的读写操作是依靠流操作进行的。也就是说.NET 中一个被打开的文件就是一个数据流，对已经打开的文件进行读写操作就是在对内存中的一个数据流进行读写操作。

5.4.1 读写文本文件

文本文件是一种纯文本数据构成的文件。实际上，文本文件只保存了字符的编码。.NET Framework 支持多种编码，包括 ASCII、UTF7、UTF8、Unicode、UTF32 等。在.NET Framework 中，读写文本文件可以使用流读取器 StreamReader 和流写入器 StreamWriter。

1. StreamReader 类

StreamReader 类用于从文件中读取数据，该类是一个通用类，可用于任何流。StreamReader 类以一种特定的编码输入字符，默认的编码为 UTF8，UTF8 可以正确处理 Unicode 字符并在操作系统的本地化版本上提供一致的结果。StreamReader 类的常用方法如表 5-10 所示。

表 5-10 StreamReader 类的常用方法

方 法	说 明
Close()	关闭 StreamReader 对象和基础流
Dispose()	释放所有 StreamReader 对象资源
Peek()	返回下一个可用的字符
Read()	读取输入流中的下一个字符或下一组字符
ReadLine()	从数据流中读取一行数据，并作为字符串返回
ReadToEnd()	从流的当前位置到末尾读取流

2. StreamWriter 类

StreamWriter 类用于将字符和字符串写入文件，它实际上也是先转换成 FileStream 对象，然后向文件中写入数据的，所以在创建对象时可以通过 FileStream 对象来创建 StreamWriter 对象，同时也可以直接创建 StreamWriter 对象。StreamWriter 默认使用 UTF8 编码。StreamWriter 类的常用方法如表 5-11 所示。

表 5-11 StreamWriter 类的常用方法

方 法	说 明
Close()	关闭 StreamWriter 对象和基础流
Dispose()	释放所有 StreamWriter 对象资源
Flush()	清理当前编写器的所有缓冲区，并使所有缓冲数据写入基础流
Write()	写入流
WriteLine()	写入指定的某些数据，后跟行结束符

【例 5-5】 简单日志程序。

【操作步骤】

(1) 启动 VS，新建一个 Windows 窗体应用程序 StreamApplication。

(2) 双击 Form1.cs，切换到设计视图，从工具栏中拖曳 2 个 Label、2 个 TextBox 控件和 2 个 Button 控件到窗体设计区，调整控件大小进行布局。

(3) 在窗体设计区中右击窗体 Form1 和每一个控件，设置窗体和控件的相关属性。

表 5-12 列出了窗体及控件属性。

表 5-12 窗体及控件属性设置

窗体和控件	属 性	属 性 值
Form1	Text	简单日志
label1	Name	请输入日志内容:
textBox1	Name	txtSource
	MultiLine	true
label2	Text	已有的日志内容:
textBox2	Name	txtShow
	MultiLine	true
	ReadOnly	true
button1	Name	btnSave
	Text	保存
button2	Name	btnShow
	Text	显示

(4) 双击“保存”按钮，为其添加单击事件处理程序，程序代码如下：

```
private void btnSave_Click(object sender, EventArgs e)
{
    StreamWriter writer = new StreamWriter(@"D:\示例代码\chapter05\日志.txt",true);
    writer.WriteLine(DateTime .Now .ToString ());
    writer.WriteLine(txtSource .Text );
    writer.Close();
    MessageBox.Show("日志保存成功!");
}
```

以上代码的功能是，在保存日志时，首先利用 StreamWriter 类的构造方法创建流写入器对象，构造方法的第 1 个参数表示文件名的路径；第 2 个参数表示是否添加新内容，如果设置为 false，将覆盖原有内容。然后调用 WriteLine()方法把日志内容写入文件。

(5) 双击“显示”按钮，为其添加单击事件处理程序，程序代码如下：

```
private void btnShow_Click(object sender, EventArgs e)
{
    StreamReader reader = new StreamReader(@"D:\示例代码\chapter05\日志.txt");
    txtShow.Text = reader.ReadToEnd();
    reader.Close();
}
```

以上代码的功能是，在读取日志内容时，首先利用 `StreamReader` 类的构造方法创建读取流的读取器对象，同时打开磁盘文件，接着调用 `ReadToEnd()`方法，把文件内容全部读出，返回的字符串通过文本框输出。

- (6) 在解决方案资源管理器中右击 `StreamApplication` 项目，将其设为启动项目。
- (7) 编译并运行，运行结果如图 5-5 所示。

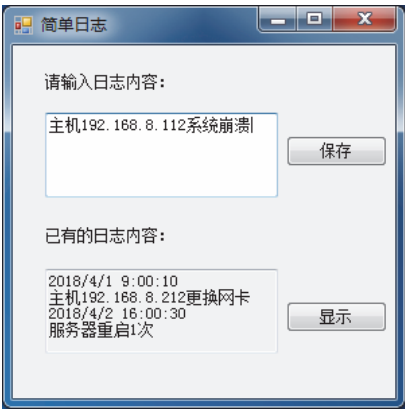


图 5-5 `StreamApplication` 项目运行结果

5.4.2 读写二进制文件

在.NET Framework 中，读写二进制文件可以使用流读取器 `BinaryReader` 和流写入器 `BinaryWriter`。二进制文件是以二进制代码形式存储的文件，数据存储为字节序列。二进制文件可以包含图像、声音、文本或编译之后的程序代码。

C#的 `FileStream` 类提供了最原始的字节级上的文件读写功能，但编程中经常会对字符串操作，于是 `StreamWriter` 类和 `StreamReader` 类增强了 `FileStream` 类，它可以在字符串级别上操作文件。但有时还是需要在字节级上操作文件，却又不是一个字节一个字节地操作，通常是 2 个、4 个或 8 个字节操作，这便有了 `BinaryWriter` 和 `BinaryReader` 类，它们可以将一个字符或数字按指定个数字字节写入，也可以一次读取指定字节转为字符或数字。

1. `BinaryReader` 类

`BinaryReader` 类用特定的编码将基元数据类型读作二进制值，其常用方法如表 5-13 所示。

表 5-13 `BinaryReader` 类的常用方法

方 法	说 明
<code>Close()</code>	关闭 <code>BinaryReader</code> 对象和基础流
<code>Dispose()</code>	释放 <code>BinaryReader</code> 类当前实例所使用的所有资源
<code>PeekChar()</code>	返回下一个可用的字符，并且不提升字节或字符的位置
<code>Read()</code>	从基础流中读取字符，并根据所使用的 <code>Encoding</code> 和从流中读取的特定字符，提升流的当前位置
<code>ReadByte()</code>	从当前流中读取下一个字节，并使流的当前位置提升一个字节
<code>ReadString()</code>	从当前流中读取一个字符串。字符串有长度前缀，一次 7 位地被编码为整数

2. BinaryWriter 类

BinaryWriter 类以二进制形式将基元类型写入流，并支持用特定的编码写入字符串，其常用方法如表 5-14 所示。

表 5-14 BinaryWriter 类的常用方法

方 法	说 明
Close()	关闭当前 BinaryWriter 和基础流
Dispose()	释放 BinaryWriter 类当前实例所使用的所有资源
ToString()	返回表示当前对象的字符串
Write()	将值写入当前流

【例 5-6】 简单学生管理程序。

【操作步骤】

(1) 启动 VS，新建一个 Windows 窗体应用程序 BinaryApplication。

(2) 双击 Form1.cs，切换到设计视图，从工具栏中拖曳 3 个 Label、2 个 TextBox 控件、2 个 RadioButton、1 个 ListBox 和 2 个 Button 控件到窗体设计区，调整控件大小进行布局。

(3) 在窗体设计区中右击窗体 Form1 和每一个控件，设置窗体和控件的相关属性。表 5-15 列出了窗体及控件属性。

表 5-15 窗体及控件属性设置

窗体和控件	属 性	属 性 值
Form1	Text	学生信息管理
label1	Name	学号：
textBox1	Name	txtNum
label2	Text	姓名：
textBox2	Name	txtName
label3	Text	性别：
radioButton1	Name	rdbMale
	Text	男
radioButton2	Name	rdbFemale
	Text	女
listBox1	Name	listShow
button1	Name	btnSave
	Text	保存
button2	Name	btnShow
	Text	显示

(4) 双击“保存”按钮，为其添加单击事件处理程序，程序代码如下：

```
private void btnSave_Click(object sender, EventArgs e)
{
    FileStream fs = new FileStream(@"D:\示例代码\chapter05\student.dat",
    FileMode.Append, FileAccess.Write);
    //通过文件流写文件
    BinaryWriter writer = new BinaryWriter(fs);
```

```
//写入一个整数
writer.Write(Int32.Parse (txtNum.Text ));
//写入一个字符串
writer.Write(txtName.Text);
bool isMale;
if (rdbMale.Checked)
{
    isMale = true;
}
else
{
    isMale = false;
}
//写入一个bool值
writer.Write(isMale);
fs.Close();
writer.Close();
}
```

以上代码的功能是，在保存数据时，首先利用 `FileStream` 类的构造方法创建一个文件流对象，该构造方法有 3 个参数，第 1 个参数表示要操作的文件名；第 2 个参数是文件模式，`FileMode.Append` 表示打开现有文件并查找找到文件尾，如果文件不存在，则创建该文件；第 3 个参数是文件操作模式，`FileAccess.Write` 表示写文件。然后通过文件流对象创建 `BinaryWriter` 写入器对象，并连续调用写入器对象的 `Write()` 方法把数据写入文件流。

(5) 双击“显示”按钮，为其添加单击事件处理程序，程序代码如下：

```
private void btnShow_Click(object sender, EventArgs e)
{
    listShow.Items.Clear();
    listShow.Items.Add("学号\t姓名\t性别");
    FileStream fs = new FileStream(@"D:\示例代码\chapter05\student.dat",
    FileMode.Open, FileAccess.Read );
    //通过文件流读文件
    BinaryReader reader = new BinaryReader(fs);
    fs.Position = 0;
    while (fs.Position !=fs.Length )
    {
        //读出一个整数
        int num = reader.ReadInt32();
        //读出一个字符串
        string name = reader.ReadString();
        string sex = "";
        //读出一个bool值
        if (reader.ReadBoolean())
        {
```

```

        sex = "男";
    }
    else
    {
        sex = "女";
    }
    string result = string.Format("{0}\t{1}\t{2}", num, name, sex);
    listShow.Items.Add(result);
}
reader.Close();
fs.Close();
}

```

以上代码的功能是，在显示数据时，首先创建文件流对象，并指定操作方式为打开（`FileMode.Open`）和读取文件（`FileAccess.Read`），然后通过文件流对象创建 `BinaryReader` 读取器对象，并使用读取器对象从头至尾循环读取文件流，最终把读出来的数据添加到列表框中输出。

（6）在解决方案资源管理器中右击 `BinaryApplication` 项目，将其设为启动项目。

（7）编译并运行，运行结果如图 5-6 所示。

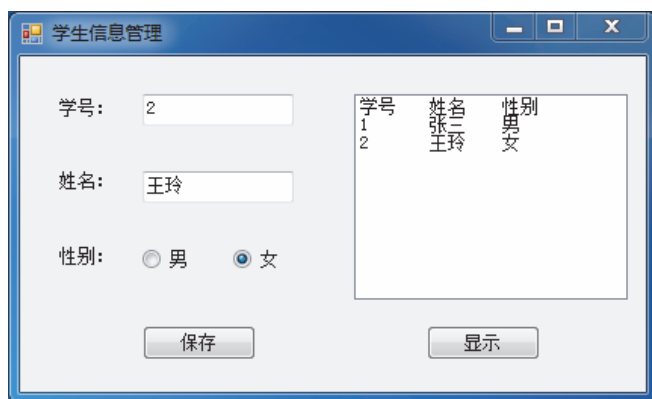


图 5-6 `BinaryApplication` 项目运行结果

5.5 习 题

1. 填空题

- （1）在 C# 中，对文件操作的类都位于_____命名空间中。
- （2）利用 `Directory` 类的_____方法可以获取指定目录中的子目录的名称。
- （3）可以使用 `DirectoryInfo` 类的_____属性获取指定子目录的父目录。
- （4）对文件进行读操作可以使用_____对象。
- （5）数据流是一个用于_____的对象。数据的传输有两个方向，如果是数据从外部源传输到程序中，就称为_____；如果是数据从程序传输到外部源，就称为_____。

2. 选择题

(1) Directory 类 () 方法用于获取目录中所有文件名。

- | | |
|----------------------|------------------|
| A. GetDirectories() | B. GetAllFiles() |
| C. GetAllFileNames() | D. GetFiles() |

(2) FileStream 类在 () 命名空间中。

- | | |
|----------------|------------------|
| A. System.IO | B. System.Data |
| C. System.File | D. System.Stream |

3. 程序设计题

接收 10 个数，保存到二进制文件中，然后读出显示在文本框中。