第5章 集　合

数组是用来存储同类型数据的。但是数组有一定的局限，即一旦数组创建了，其容量就限定了。如“string[] names=new String[5];”限制了容量为5个元素，当设置“names[5]="ada";”时就会发生“数组下标越界”异常。而采用集合(collection)相关类就可突破该容量限制，其容量在不够时会自动扩充。

集合类可以看成一种特殊的数组，也用以存储多个数据。C#提供了对栈(stack)、队列(queue)、列表(list)和哈希表(hash table)等不同类型的支持。

C#中集合类又分为非泛型集合类和泛型集合类两种。非泛型集合类可以存储各类对象，常用类为ArrayList和Hashtable；而泛型集合类用以保存指定类型对象，常用类为List<T>和Dictionary<K,V>。泛型集合类在代码上消除了强制类型转换语句，在减少转换出错概率的同时，提高了代码的可读性。因此，在实际项目中，一般使用泛型集合类。

5.1 非泛型集合类

非泛型集合类定义在System.Collections名称空间中，常用的类有ArrayList(数组列表)和Hashtable(哈希表)。

5.1.1 ArrayList

ArrayList常作为传统数组的替代，提供了常用属性(如Count、Capability)和常用方法(如Add()、AddRange()、Insert()、Remove()、RemoveAt()、Clear()、Contains()、IndexOf()、LastIndexOf()、Sort()、Reverse()等)。

1. 添加元素，判断元素个数与容量

在ArrayList中添加元素，可通过Add()、AddRange()、Insert()三个方法实现。判断ArrayList中元素个数用Count属性，判断容量则用Capacity属性。

【例5-1】 在ArrayList中添加元素，判断元素个数与容量。

```
using System;
using System.Collections;
...
ArrayList aryLst = new ArrayList();
aryLst.Add("Ada");                                          //添加 string 元素
aryLst.Add(46);                                             //添加 int 元素
aryLst.AddRange(new ArrayList(){ 119, 143, 132 });          //添加范围元素
aryLst.Insert(1, 'F');                                      //在下标 1 处插入 char 元素
```

```
foreach(object ele in aryLst)
        Console.Write(ele + "\t");                    //Ada F 46 119 143 132
Console.WriteLine(                                    //元素有 6 个,容量为 8
      "元素有{0}个,容量为{1}", aryLst.Count, aryLst.Capacity);
```

2. 删除元素

删除元素可通过 Remove()、RemoveAt()、Clear()三个方法实现。

【例 5-2】 删除 ArrayList 中的元素。

```
using System;
using System.Collections;
…
ArrayList scores = new ArrayList() { 119, 143, 132, 119 };
scores.Remove(119);                                   //删除第一个值为 119 的元素
foreach (object score in scores)
  Console.Write(score + "\t");                        //143 132 119
scores.RemoveAt(0);                                   //删除下标为 0 的元素
foreach (object score in scores)
    Console.Write(score + "\t");                      //132 119
scores.Clear();                                       //清空元素
foreach (object score in scores)
    Console.Write(score + "\t");                      //无显示
```

3. 修改元素

修改元素可通过下标方式进行。

【例 5-3】 修改 ArrayList 中元素的值。

```
using System;
using System.Collections;
…
ArrayList scores = new ArrayList() { "A+", "B+", "A+" };
scores[0] = 70;
scores[1] = 64;
scores[2] = 70;
foreach (object score in scores)
    Console.Write(score + "\t");     //70, 64, 70
```

4. 查询元素

查询元素可通过 Contains()、IndexOf()、LastIndexOf()三个方法实现。

【例 5-4】 查询 ArrayList 中的元素。

```
using System;
using System.Collections;
…
ArrayList scores = new ArrayList() { 70, 64, 70, 61, 64 };
int idx1 = scores.IndexOf(70);          //0,从首部查找元素,返回找到的第一个元素所在下标,
                                        //若找不到则返回 -1
int idx2 = scores.IndexOf(70,1);        //2,从特定下标往后寻找元素,返回找到的第一个元素
                                        //所在下标,若找不到则返回 -1
```

```
int idx3 = scores.LastIndexOf(70);      //2,从尾部查找元素,返回找到的第一个元素所在下标,
                                        //若找不到则返回 - 1
int idx4 = scores.LastIndexOf(70,1);    //0,从特定下标往前寻找元素,返回找到的第一个元素
                                        //所在下标,若找不到则返回 - 1
bool b = scores.Contains(70);           //true,判断是否包含元素
```

5. 元素排序

对元素排序,可通过 Sort()方法实现,倒序则用 Reverse()方法实现。

【例 5-5】 ArrayList 中元素的排序。

```
using System;
using System.Collections;
...
ArrayList scores = new ArrayList() { 70, 64, 70, 61, 64 };
scores.Sort();
foreach (object score in scores)
    Console.Write(score + "\t");                    //61 64 64 70 70
scores.Reverse();
foreach (object score in scores)
    Console.Write(score + "\t");                    //70 70 64 64 61
```

5.1.2 Hashtable

Hashtable 类似字典,以键值对保存元素。其中,键必须是唯一的,而值不需要唯一。通过唯一的键值就可找到对应的元素值。Hashtable 提供了常用属性(如 Count、Keys、Values)和常用方法(如 Add()、Remove()、Clear()、ContainsKey()、ContainsValue()等)。

1. 添加元素、判断元素个数

添加元素可通过 Add()方法,判断元素个数可通过 Count 属性。

【例 5-6】 添加 Hashtable 元素。

```
using System;
using System.Collections;
...
Hashtable ht = new Hashtable();
ht.Add(1, "Ada");                                   //添加元素
ht.Add("two", "Bob");                               //添加元素
foreach (Object key in ht.Keys)
        Console.Write(ht[key] + "\t");              //Bob Ada
int cnt = ht.Count;                                 //2,元素个数
```

2. 修改、删除元素

删除元素可通过 Remove()、Clear()方法,修改元素可通过 key 下标方式。

【例 5-7】 删除 Hashtable 元素。

```
using System;
using System.Collections;
```

```
…
Hashtable ht = new Hashtable();
ht.Add(1, "Ada");
ht.Add("two", "Bob");
ht.Remove("two");                                    //删除 key 对应的元素
int cnt = ht.Count;                                  //1,元素个数
ht[1] = "Amanda";                                    //通过 key 下标修改元素
foreach (object val in ht.Values)
    Console.Write(val + "\t");                       //Bob Amanda,已修改
```

3. 查询元素

查询元素可通过 ContainsKey()、ContainsValue()方法。

【例 5-8】 查询 Hashtable 元素。

```
using System.Collections;
…
Hashtable ht = new Hashtable();
ht.Add(1, "Ada");
ht.Add("two", "Bob");
bool b2 = ht.ContainsKey("two");
bool b3 = ht.ContainsValue("Bob");                   //true,判断 value 是否存在
```

5.2 泛型集合类

泛型集合类定义在 System.Collections.Generic 名称空间中。常用类为 List<T>和 Dictionary<K,V>。项目实践中泛型集合类使用频繁,应该重点掌握。

5.2.1 List<T>

List<T>不仅具有 ArrayList 集合类功能,通过泛型还能指定操作元素的类型。常用的方法有 Add()、AddRange()、Insert()、Remove()、RemoveAt()、Sort()、Reverse()、Clear(),常用的属性有 Count。

【例 5-9】 List<T>元素的添加、插入、删除、排序、遍历和获取元素个数等。

```
using System;
using System.Collections.Generic;
…
List<int> list = new List<int>();
list.Add(70);                                //添加 int 元素,成功
//list.Add("A+");                            //编译出错"无法从 string 转换为 int"
//new List<T>(IEnumerable<T> collection): 以集合参数创建 List<T>,如下
List<String> names                           //泛型指定<String>,所以 list 中只能放置 String 元素
 = new List<String>(new String[]{ "Bob", "Ada","Carl" });
//添加、删除、修改元素
```

```
//names.Add(70);                    //编译出错"无法从 int 转换为 string"
names.Add("Daniel");                //添加元素,names 为 Bob、Ada、Carl、Daniel
names.AddRange(new string[] { "Fanny","Edwin" });   //添加范围元素,names 为 Bob、Ada、Carl
                                                    //Daniel、Fanny、Edwin
names.Insert(3, "Amanda");          //在下标 3 位置插入元素 Amanda,names 为 Bob、Ada、Carl
                                    //Amanda、Daniel、Fanny、Edwin
bool op1 = names.Remove("Edwin"); //true,删除成功则返回 true,否则返回 false。names 为
                                    //Bob、Ada、Carl、Amanda、Daniel、Fanny
names.RemoveAt(2);                  //删除下标 2 位置元素 Ada,names 为 Bob、Ada、Amanda
                                    //Daniel、Fanny
int cnt1 = names.Count;             //5,元素个数
//排序、倒序
names.Sort();                       //排序,names 为 Ada、Amanda、Bob、Daniel、Fann
names.Reverse();                    //倒序,names 为 Fanny、Daniel、Bob、Amanda、Ada
foreach (string name in names)      //遍历元素
    Console.Write(name + "\t");   //Fanny、Daniel、Bob、Amanda、Ada
names.Clear();                      //清空元素
int cnt2 = names.Count;             //0,元素个数
```

5.2.2 Dictionary<K,T>

Dictionary<K,T>是存放与操作键值对的集合类,它不仅具有 Hashtable 集合类功能,通过泛型还能指定键与值的类型。常用的方法有 Add()、Remove()、Clear()、ContainsKey()、ContainsValue(),常用的属性有 Count、Keys、Values。

【例 5-10】 Dictionary<K,T>元素的添加、遍历、删除,获取元素个数、修改元素值、判断元素是否存在等。

```
Dictionary<int,string> dic = new Dictionary<int, string>();
dic.Add(1, "Ada");                              //添加元素成功, 键值对类型正确
//dic.Add("two", "Bob");                        //添加元素失败, 键值对类型错误
dic.Add(2, "Bob");
foreach (int key in dic.Keys)
    Console.Write(dic[key] + "\t");             //Ada Bob
int cnt1 = dic.Count;                           //2,元素个数
dic.Remove(2);                                  //删除 key=2 对应的元素 Bob
int cnt2 = dic.Count;                           //1,元素个数
dic[1] = "Amanda";                              //通过下标 key 修改元素
foreach (string val in dic.Values)
    Console.Write(val + "\t"); ;                //Amanda,修改了
bool b2 = dic.ContainsKey(1)                    //true,判断 key 是否存在
bool b3 = dic.ContainsValue("Bob");             //false,判断 value 是否存在
dic.Clear();                                    //清除元素
int cnt3 = dic.Count;                           //0,元素个数
```

5.2.3 List<T>类型与数组类型的转换

很多场合下,有 List<T>类型和数组类型进行互转的需要。

数组类型转换为 List 类型,使用数组实例的 ToList()方法;List 类型转换为数组类型,使用 List 实例的 ToArray()方法。

【例 5-11】 Emp 数组转换为 List<Emp>。

```
class Emp     //定义一个类
{
    public int id;
    public String name;
    public Emp(int id, string name)
    {
        this.id = id; this.name = name;
    }
}
```

测试:

```
Emp[] empAry = { new Emp(1,"Ada"), new Emp(2,"Bob") };     //创建对象数组
List<Emp> list = empAry.ToList<Emp>();     //转换为 List,注意先用 using System.Linq
```

【例 5-12】 List<Emp>转换为 Emp 数组。

```
List<Emp> list = new List<Emp>();                    //创建 List 对象
list.Add(new Emp(1, "Ada"));
list.Add(new Emp(2, "Bob"));
Emp[] empAry = list.ToArray();                       //转换为数组
```

5.3 项目案例——中国当代著名科学家,华夏真脊梁 2

第 4 章的项目案例中,选择创建了 5 位当代中国科学家,他们都有着科学家崇高的情操、爱国的情怀、执着的追求和忘我的精神。

除了上述 5 位伟大的科学家,中国还有很多很多非常优秀的科学家。他们都是我国的骄傲,值得每一个人发自内心的尊重。此时如果想在原来系统中加入其他科学家,例如“杨振宁”和“竺可桢”,怎么操作呢?

显然,原有的数组方式存放科学家信息,扩充性太差,不再适合,那么可考虑采用本章所学的泛型集合类来存储;另外,可增补一个“添加科学家信息”功能,既可验证泛型集合类的可扩充性,同时也让系统更完善。

请改造原来的“中国当代著名科学家推荐系统”。

具体设计要求和步骤如下。

5.3.1 设计一:优化科学家信息的存储设计

设计说明:科学家信息原来是放置在数组中的,但数组无法直接扩充,操作不当还会造

成下标越界异常。为此，考虑将数组方式存放改为集合类 List < T >方式存放，即采用 List < Scientist >变量存储可变个数的科学家信息。

设计实现步骤：

(1) 在“解决方案资源管理器”窗口中，打开第 4 章案例项目 WinFormScientists，单击 FormScientists.cs 窗体文件，进入代码设计窗体。

(2) 在 Scientist[] scientists 定义的下方，加入数组转换为 List 的代码：

```
public static List < Scientist > listScientist          //"添加窗体"调用，设为 public
   = scientists.ToList < Scientist >();                 //需要用 using System.Linq;
```

(3) 在 FormScientists_Load()方法中，改写数组操作为 List 操作。

将如下代码：

```
int idx = rand.Next(scientists.Length);          //获得随机下标值
Scientist scientist = scientists[idx];
```

修改为：

```
int idx = rand.Next(listScientist.Count);        //获得随机下标值
Scientist scientist = listScientist[idx];
```

(4) 单击“启动”按钮或按 F5 键，启动应用，效果与原系统一致。

5.3.2 设计二：“添加科学家信息”功能

设计说明：为实现“添加科学家信息”功能，应该增加一个窗体。窗体上应含有针对科学家图片文件上传及显示的相关控件；针对科学家姓名、简介文字信息输入所需的控件；针对生日，能输入日期的控件；另外，需加一个“添加”按钮控件，在其 Click 事件处理方法中编写具体的“添加科学家信息”功能代码。

设计实现步骤：

(1) 提示“添加图片”文件的准备及设置。

① 准备一个图片文件，用以提示添加科学家图片，如图 5-1 所示。

② 将图片文件加入项目 Images 目录中，如图 5-2 所示。

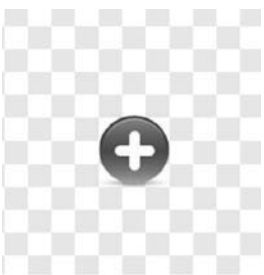

图 5-1 提示“添加图片”文件

图 5-2 将图片文件加入项目

③ 设置该图片文件属性“复制到输出目录”值为“始终复制”,如图 5-3 所示。

图 5-3 图片文件设置为复制到输出目录

(2)“添加”窗体设计。

① 在“解决方案资源管理器”窗口中,右击项目 WinFormScientists,在弹出的快捷菜单中选择“添加”→“窗体”选项,在弹出的对话框中设置名称为 FormAdd.cs,单击“添加”按钮,弹出 FormAdd 窗体。

② 单击左侧“工具箱”,打开“所有 Windows 窗体”选项卡,拖曳 3 个 Label 控件、1 个 PictureBox 控件、2 个 TextBox 控件、1 个 DateTimePicker 控件和 1 个 Button 控件,放置到窗体中。

对相关控件设置属性、调整尺寸,如下:

```
Label3: Name = "LabelDesc";Font = "楷体, 15pt"; AutoSize = False(再用鼠标拖拉调整尺寸值)
Form: Name = "FormAdd";Text = "中国当代著名科学家";设置 Size 值同主窗体尺寸一致
PictureBox: Name = "pictureBoxImg";SizeMode = "Zoom"(图像伸缩以适应)
TextBox: Name = "textBoxName";Font = "楷体, 15pt, style = Bold"
DateTimePicker: Name = "dateTimePickerBirth";Font = "楷体, 10pt"
TextBox: Name = "textBoxDesc";Multiline = True;Font = "楷体, 15pt"(Size 值则可按需自行调整)
button1: Name = "buttonAdd";Text = "添加科学家"
```

“添加”窗体的效果如图 5-4 所示。

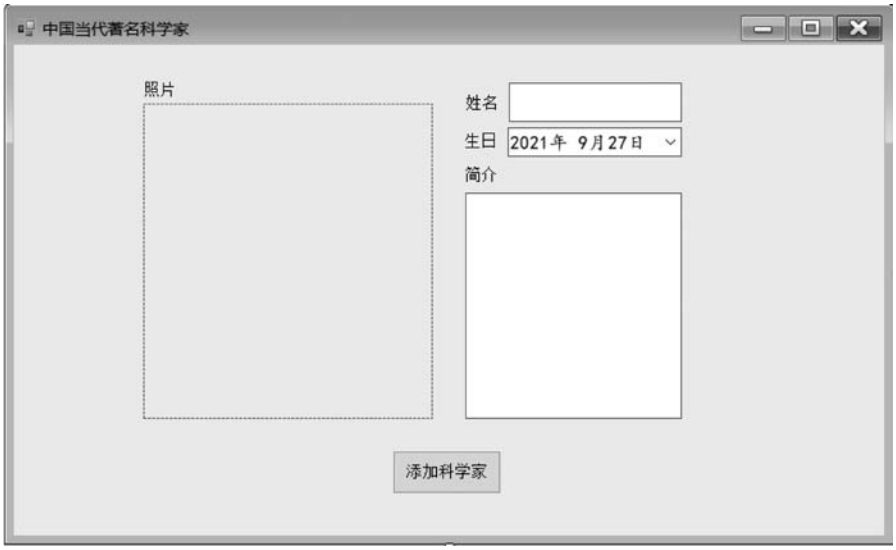

图 5-4 “添加”窗体的效果

(3) 弹出“添加”窗体。

① 在主窗体中加“添加”(+)按钮。

在主窗体 FormScientists 设计界面下,单击左侧“工具箱”,打开“所有 Windows 窗体”选项卡,拖曳 Button 控件到主窗体 FormScientists 右上角,右击按钮控件,在弹出的快捷菜

单中选择“属性”选项，在属性框中，设置 Name 属性值为 buttonAdd、Text 属性值为＋，如图 5-5 所示。

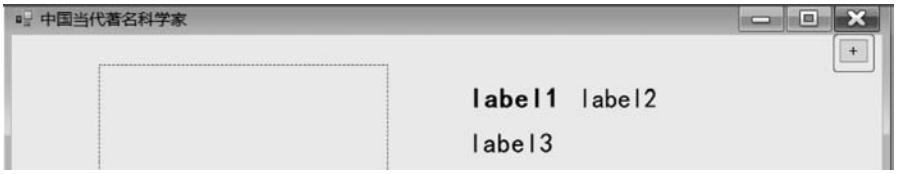

图 5-5　在主窗体中加“添加”(＋)按钮

② 在主窗体中，双击右上角的“添加”(＋)按钮生成按钮 Click 事件处理方法，在方法中编写如下代码，用以弹出“添加”窗体。

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;
using System.Linq;
namespace WinFormScientists
{
    public partial class FormScientists : Form
    {
        …
        private void buttonAdd_Click(object sender, EventArgs e)
        {
              //打开"添加"窗体
              new FormAdd().ShowDialog();
        }
    }
}
```

③ 单击“启动”按钮或按 F5 键，启动应用，弹出主窗体，如图 5-6 所示。

图 5-6　启动应用，弹出主窗体

单击主窗体右上角的“添加”(+)按钮,弹出“添加”窗体,如图 5-7 所示。

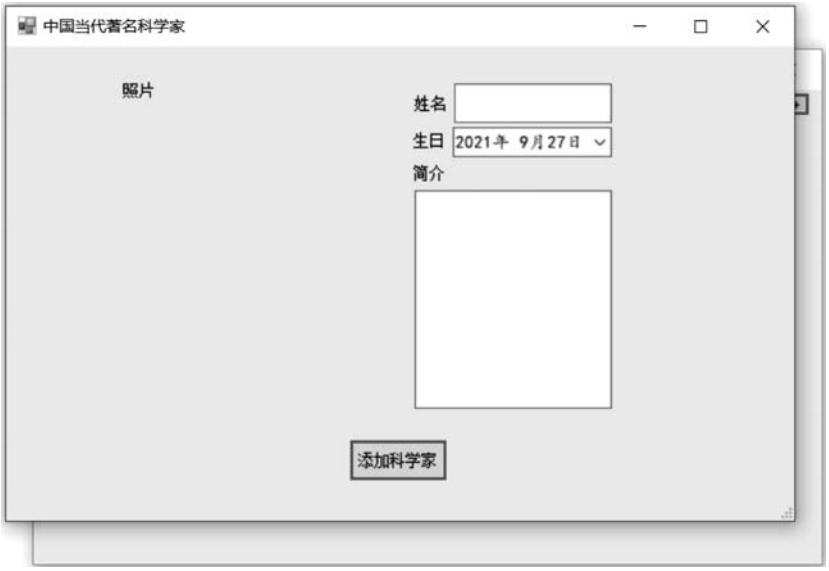

图 5-7　弹出“添加”窗体

(4) 实现“添加”功能。

① 初始化“添加图片”。

双击“添加”窗体空白处,生成窗体 Load 事件处理方法,在方法中编写如下代码,用以显示“添加图片”文件。

```
using System;
using System.Windows.Forms;
namespace WinFormScientists
{
    public partial class FormAdd : Form
    {
        public FormAdd()
        {
            InitializeComponent();
        }
        private void FormAdd_Load(object sender, EventArgs e)
        {
            pictureBoxImg.ImageLocation //初始化显示"添加图片"
                  = Application.StartupPath + @"\Images\plus.png";
        }
    }
}
```

② 单击“启动”按钮或按 F5 键,启动应用。在弹出的主窗体右上角单击“添加”(+)按钮,出现如图 5-8 所示的效果,可看到图片初始化了。

图 5-8 “添加”窗体运行效果

③ 选择图片文件。

在“添加”窗体中，双击 PictureBoxImg 控件，会生成 Click 事件处理方法，在方法中编写如下代码，用于选择、显示和保存科学家图片文件。

```
string imgUrl = null;                    //上传照片位置,设为成员变量,便于 buttonAdd_Click()调用
private void pictureBoxImg_Click(object sender, EventArgs e)
{
    string scientistName = textBoxName.Text;                    //用以为上传图片文件起名
    if (string.IsNullOrWhiteSpace(scientistName))
    {
        MessageBox.Show("姓名不能为空,请先填写");
        textBoxName.Focus();                                    //光标回到姓名输入框,等待输入
        return;                                                 //返回,先修正,方可处理下方代码
    }
    OpenFileDialog fileDialog = new OpenFileDialog();
    fileDialog.Title = "选择要上传的图片";
    fileDialog.Filter = "图片文件|*.bmp;*.jpg;*.jpeg;*.gif;*.png";
    DialogResult dr = fileDialog.ShowDialog();
    if (!File.Exists(fileDialog.FileName))                      //using System.IO;
    {
        MessageBox.Show("照片为空,请选择图片文件");
        return;
    }
    if (dr == DialogResult.OK)
    {
       string image = fileDialog.FileName;
        string ext = Path.GetExtension(fileDialog.FileName); //System.IO.Path
        pictureBoxImg.Image = Image.FromFile(image);        //System.Drawing.Image
        imgUrl = Application.StartupPath + @"\Images\" + scientistName + ext;
        File.Copy(fileDialog.FileName, imgUrl);             //保存图片文件到项目目录中
    }
}
```

注意,Application.StartupPath + @"\Images\" + scientistName + ext 代码是将图片文件的存放路径设置到项目目录下。

④ 实现添加科学家。

双击“添加科学家”按钮生成 Click 事件处理方法,在方法中编写如下代码,用以添加科学家。

```
private void buttonAdd_Click(object sender, EventArgs e)
{
    //获取输入参数: 姓名、生日、简介、(上传)图片文件
    String name = textBoxName.Text;
    if (String.IsNullOrWhiteSpace(name))
    {
        MessageBox.Show("姓名不能为空");
        textBoxName.Focus();                          //光标回到姓名输入框,等待输入
        return;                                       //返回,先修正,再处理下方代码
    }
    DateTime birth = dateTimePickerBirth.Value;
    String desc = textBoxDesc.Text;
    if (String.IsNullOrWhiteSpace(desc))
    {
         MessageBox.Show("简介不能为空");
         textBoxDesc.Focus();
         return;                                      //返回,先修正,再处理下方代码
    }
    if (imgUrl == null)                               //图片文件没选择过
    {
         MessageBox.Show("照片必须选择,请在图片位置单击。");
         return;                                      //返回,先修正,再处理下方代码
    }
    //创建科学家 scientist 对象,放入主窗体科学家集合 listScientist 中
    Scientist scientist = new Scientist();
    scientist.Name = name;
    scientist.Birthday = birth;
    scientist.Description = desc;
    scientist.ImageURL = imgUrl;
    FormScientists.listScientist.Add(scientist);
    MessageBox.Show("添加成功");
    this.Close();
}
```

⑤ 单击“启动”按钮或按 F5 键,启动应用。

启动应用后弹出主窗体,单击其右上角的“添加”(+)按钮,如图 5-9 所示。

⑥ 添加科学家信息。

在弹出的“添加”窗体中,进行添加科学家各项信息的操作:输入姓名→设置生日→加入简介→单击图片弹出文件选择对话框→选择科学家的照片→单击“打开”按钮,如图 5-10 所示。

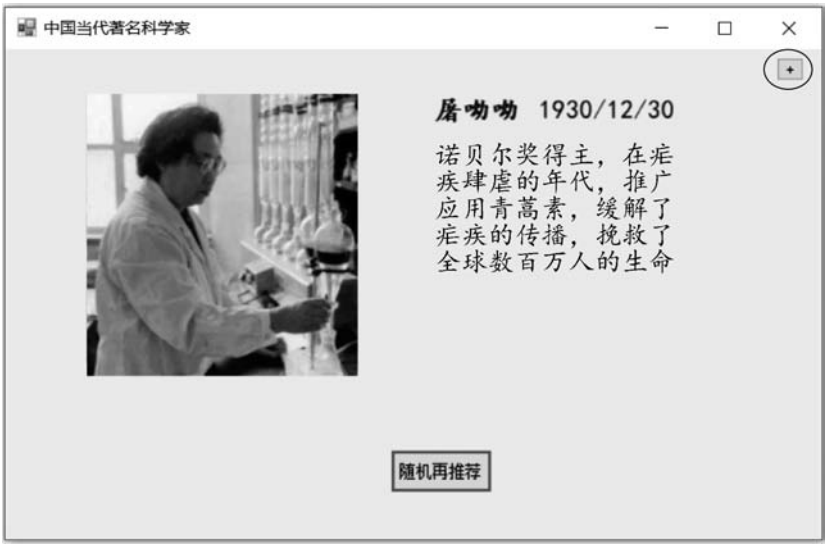

图 5-9　单击主窗体右上角的“添加”(＋)按钮

图 5-10　添加科学家各项信息

照片正常显示后，单击“添加科学家”按钮，在弹出的对话框中单击“确定”按钮，完成添加操作，如图 5-11 所示。

回到主窗体，单击“随机再推荐”按钮，会切换科学家的信息，如图 5-12 所示。

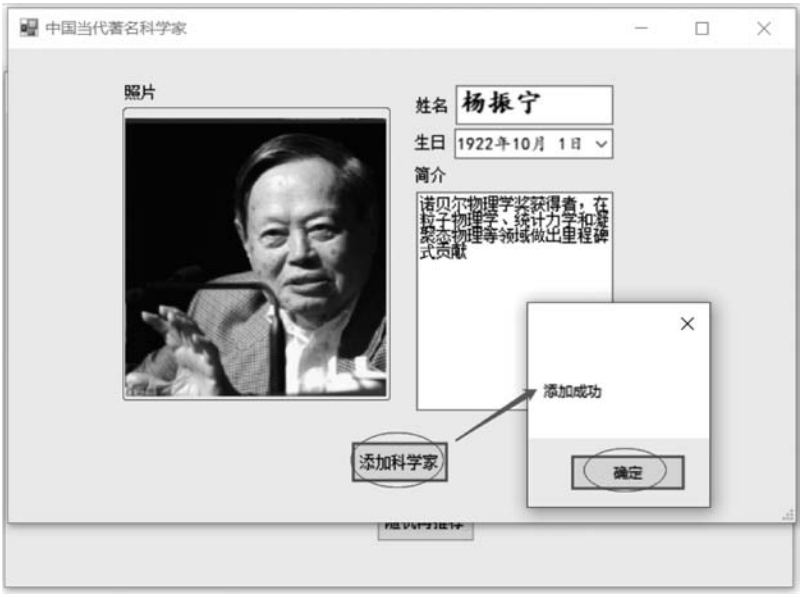

图 5-11　单击“添加科学家”按钮完成添加操作

图 5-12　单击“随机再推荐”按钮切换科学家信息

项目小结：

(1) 用集合类 List < Scientist >变量存放科学家数据。在添加新的科学家信息时，可不用考虑因为采用数组而引起下标越界的问题。因此，在项目实践中一般采用集合类，而不使用数组。

(2) “添加”窗体上，为了添加科学家信息，需要选用各类合适控件。例如，为了便于输入生日，选用了 DateTimePicker 控件。

(3) 图片文件的上传功能：在弹出的文件选择对话框中选择图片文件并确认，将文件保存到适当文件夹中。

核心代码如下：

```
OpenFileDialog fileDialog = new OpenFileDialog();
if (fileDialog.ShowDialog() == DialogResult.OK)
{
    string imgUrl = Application.StartupPath + @"\Images\"
        + scientistName + Path.GetExtension(fileDialog.FileName);
    File.Copy(fileDialog.FileName, imgUrl);
}
```