实 验 三　word 操作

3.1　实验目的

学习和了解 COM 基本原理，掌握使用 VS2013 开发工具采用 C# 编程语言，通过 COM 技术操作 Word 文档的方法。

3.2　COM 技术介绍

微软公司以 Windows 为核心软件产品，有很高的市场占有率，结合 windows 平台它也推出了一套办公自动化产品，这就是鼎鼎大名的 Office 办公软件。为了实现在微软的多种不同软件产品中的通信和功能调用，微软提供了 COM 技术，称为通用组件模型。COM 技术具有程序语言无关性，不同的程序语言通过 COM 技术可以无障碍互相调用，在实现软件功能复用方面意义重大。COM 组件以接口的形式提供给程序开发者，开发者无须了解接口的内部实现，只要在程序中满足接口参数形式，就可实现相应的功能。COM 技术在不同软件产品中扮演“桥”的角色，是一个中间连接器，提供将软件对象与外界的接口，应用程序只需通过调用相应 COM 接口，即可完成软件产品中的各种功能。微软的 COM 技术基于 Windows 操作系统，一般不支持其它操作系统平台。

微软件将 Office2003 产品编号为 OFFICE11，Office2007 产品编号为 OFFICE12，Office2013 编号为 OFFICE15，本书项目基于 OFFICE15 产品。

3.3　Word 中的 COM 对象

程序操作 Word 文档是通过 Office 产品中的相应 COM 对象实现的，这些对象包含大量的的方法属性和常量，下面列出 Word 的主要 COM 对象。

1. Application 对象控制当前 word 进程，该对象的属性和方法控制 Word 环境和运行；
2. Document 对象操作文档；
3. Selection 对象表示当前选择的区域。在 Word 用户界面中执行某项操作（例如，对文本进行加粗）时，应首先选择或突出显示文本，然后应用格式设置。Selection 对象始终存在于文档中，如果未选中任何对象，它表示插入点；
4. Paragraph 对象操作段落；
5. Table 对象操作表格；
6. Section 对象扣作小节；
7. Range 对象操作文档中指定的区域，例如由一个起始字符位置和一个结束字符位置定义，Range 对象的要根据其上级对象来确定所表示的文档内容；

8. Bookmark 对象，书签用于在文档中标记一个位置，或者用作文档中的文本容器。书签对象可以小到只有一个插入点，也可以大到整篇文档。

.NET 平台中 COM 对象的函数参数类型统一为 object 类型，函数参数必须是引用传递，函数参数的缺省值由下面的内置对象定义：

```
object missing=System.Reflection.Missing.Value;
```

3.4 Word 格式文档制作

COM 方式操作 word 文档的流程可划分为下面几个过程：

1. 安装 Word 软件；
2. 准备包含数据的文本文件；
3. 创建应用项目；
4. 项目添加 Word 的 COM 组件；
5. 程序代码添加 Word 对象的命名空间；
6. 程序中创建 COM 对象，调用相应方法操作 Word 文档；
7. 清除 COM 组件后退出程序；

本示例程序采用控制台程序类型将原始的文本内容生成特定格式要求的 Word 文档。文本文件 abstract.txt 包含论文题目和摘要文本内容，文本文件 content.txt 包含正正文本。程序调用适当 COM 方法控制 Word 文档中的小节、页码格式、段落行距、段落首行缩进、文本的字体，并插入目录、表格和图片完成文档格式制作。首先创建名为 CCW 的控制台项目，为方便调试将项目的输出文件设为当前路径，方法是菜单：项目 –> 项目属性，切换到" 生成" 选项页，找到输出路径将其改为"."。使用菜单：项目 --添加引用，在 COM 标签页添加名为"Microsoft Word 15.0 Object Library" 的 Word 对象互操作库，如图 3-1。

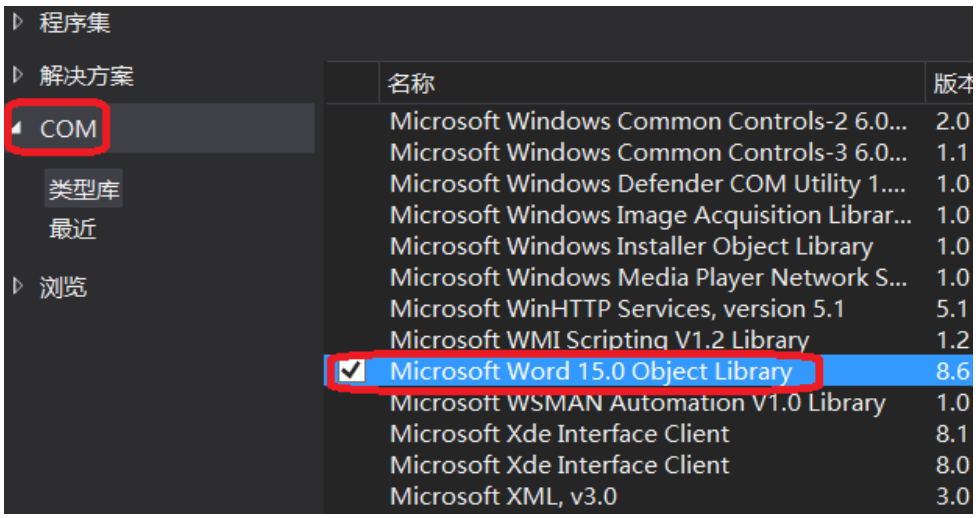

图 3-1 添加 word15 对象操作库

在程序代码源文件中添加命名空间支持：

```
using MsWord=Microsoft.Office.Interop.Word;
```

所有操作 Word 对象的 COM 方法调用代码必须处在在异常处理代码块中，先创建 Word 的

Application 对象，它是 Word 对象操作的最开始，创建此对象时 WINWORD.EXE 进程启动。

```
MsWord.Application oWordApplic;//a reference to Wordapplication
MsWord.Document oDoc;//a reference to thedocument
try
{
}
catch (Exception e2)
{
  MessageBox.Show(e2.Message);
}
```

后续所有操作 Word 对象的代码都要处于 Try 语法块中，下面首先在检测到旧的 word 文档后删除旧文档。

```
string doc_file_name = Directory.GetCurrentDirectory() + @"\content.doc";
if (File.Exists(doc_file_name))
{
  File.Delete(doc_file_name);
}
oWordApplic = new MsWord.Application();
object missing = System.Reflection.Missing.Value;
```

3.4.1 创建 Word 文档的小节

分节符在 Word 文档中是用来生成小节的控制符，每小节的页眉页脚的内容，页码格式等保持一致。本项目生成的 Word 文档要求有不同的页码格式和页眉内容，首先插入 4 个分节符获得 5 个小节，第 1 小节是摘要内容，第 2 小节是目录，第 3 小节是第一章，第 4 小节是表格，第 5 小节是图片。第 1,、2 小节的页码是大写罗马数字，第 3、4、5 小节的页码是阿拉伯数字，且起始页码为 1。

```
MsWord.Range curRange;
object curTxt;
int curSectionNum = 1;
oDoc = oWordApplic.Documents.Add(ref missing, ref missing, ref missing, ref missing);
oDoc.Activate();
Console.WriteLine(" 正在生成文档小节");
object section_nextPage = MsWord.WdBreakType.wdSectionBreakNextPage;
object page_break = MsWord.WdBreakType.wdPageBreak;
//添加三个分节符，共四个小节
for (int si = 0; si < 4; si++)
{
  oDoc.Paragraphs[1].Range.InsertParagraphAfter();
  oDoc.Paragraphs[1].Range.InsertBreak(ref section_nextPage);
```

```
}
```

3.4.2 插入摘要文本并设置文本格式

```
    Console.WriteLine(" 正在插入摘要内容");
#region 摘要部分
curSectionNum = 1;
curRange = oDoc.Sections[curSectionNum].Range.Paragraphs[1].Range;
curRange.Select();
string one_str, key_word;
//摘要的文本来自 abstract.txt 文本文件
StreamReader file_abstract = new StreamReader("abstract.txt");
oWordApplic.Options.Overtype = false;//overtype 改写模式
MsWord.Selection currentSelection = oWordApplic.Selection;
if (currentSelection.Type == MsWord.WdSelectionType.wdSelectionNormal)
{
  one_str = file_abstract.ReadLine();//读入题目
  currentSelection.TypeText(one_str);
  currentSelection.TypeParagraph(); //添加段落标记
  currentSelection.TypeText(" 摘要");//写入" 摘要" 二字
  currentSelection.TypeParagraph(); //添加段落标记
  key_word = file_abstract.ReadLine();//读入题目
  one_str = file_abstract.ReadLine();//读入段落文本
  while (one_str != null)
  {
    currentSelection.TypeText(one_str);
    currentSelection.TypeParagraph(); //添加段落标记
    one_str = file_abstract.ReadLine();
  }
  currentSelection.TypeText(" 关键字：");
  currentSelection.TypeText(key_word);
  currentSelection.TypeParagraph(); //添加段落标记
}
file_abstract.Close();
```

Word 文档的内容都是具有格式的，刚刚写入的文本内容是 Word 的普通正文格式，这里程序使用了段落 Paragraphs 和范围 Range 对象，Paragraph 与文档的段落是一一对应的；Range 对象具有非常灵活的意义，它的代表的范围可大可小，可以是段落中一串字符串片段，也可以跨几个页面内容，包括几个小节都可以。Range 的实际意义由它具体的使用方法来限制。这里 Range 对象所代表是一个自然段，然后通过把段落文字设置为不同的格式将其格式化为摘要的格式，使用宋体 22 号字来表示标题格式。程序通过 Paragraphs 对象的 LineSpacing 值控制文

本的行距为 1.25 倍行距，IndentFirstLineCharWidth 方法则设置了段落的首行缩进 2 个字符。

下面是设置摘要文本的格式：

```
//摘要的标题
curRange = oDoc.Sections[curSectionNum].Range.Paragraphs[1].Range;
curTxt = curRange.Paragraphs[1].Range.Text;
curRange.Font.Name = " 宋体";
curRange.Font.Size = 22;
curRange.Paragraphs[1].Alignment =
MsWord.WdParagraphAlignment.wdAlignParagraphCenter;
//" 摘要" 两个字
curRange = oDoc.Sections[curSectionNum].Range.Paragraphs[2].Range;
curRange.Select();
curRange.Paragraphs[1].Alignment =
MsWord.WdParagraphAlignment.wdAlignParagraphCenter;
curRange.Font.Name = " 黑体";
curRange.Font.Size = 16;
//摘要正文
oDoc.Sections[curSectionNum].Range.Paragraphs[1].Alignment =
MsWord.WdParagraphAlignment.wdAlignParagraphCenter;
for (int i = 3; i < oDoc.Sections[curSectionNum].Range.Paragraphs.Count; i++)
{
  curRange = oDoc.Sections[curSectionNum].Range.Paragraphs[i].Range;
  curTxt = curRange.Paragraphs[1].Range.Text;
  curRange.Select();
  curRange.Font.Name = " 宋体";
  curRange.Font.Size = 12;
  oDoc.Sections[curSectionNum].Range.Paragraphs[i].LineSpacingRule =
    //oDoc.Paragraphs[i].LineSpacingRule =
  MsWord.WdLineSpacing.wdLineSpaceMultiple;
  //多倍行距，1.25 倍，这里的浮点值是以 point 为单位的，不是行距倍数
  oDoc.Sections[curSectionNum].Range.Paragraphs[i].LineSpacing = 15f;
  oDoc.Sections[curSectionNum].Range.Paragraphs[i].IndentFirstLineCharWidth(2);
}
//设置" 关键字：" 为黑体
curRange = curRange.Paragraphs[curRange.Paragraphs.Count].Range;
curTxt = curRange.Paragraphs[1].Range.Text;
object range_start, range_end;
range_start = curRange.Start;
range_end = curRange.Start + 4;
curRange = oDoc.Range(ref range_start, ref range_end);
```

```
curTxt = curRange.Text;
//curRange = curRange.Range(ref range_start, ref range_end);
curRange.Select();
curRange.Font.Bold = 1;
#endregion 摘要部分
```

3.4.3 插入目录并设置目录格式

目录在 Word 文档中是一种特殊的引用，它能从各级标题内容自动生成，下面实现插入目录的操作。

```
Console.WriteLine(" 正在插入目录");
#region 目录
curSectionNum = 2;
curRange = oDoc.Sections[curSectionNum].Range.Paragraphs[1].Range;
curRange.Select();
//插入目录时指定的参数
object useheading_styles = true;//使用内置的目录标题样式
object upperheading_level = 1;//最高的标题级别
object lowerheading_level = 3;//最低标题级别
object usefields = 1;//true 表示创建的是目录
object tableid = 1;
object RightAlignPageNumbers = true;//右边距对齐的页码
object IncludePageNumbers = true;//目录中包含页码
currentSelection = oWordApplic.Selection;
currentSelection.TypeText(" 目录");
currentSelection.TypeParagraph();
currentSelection.Select();
curRange = oDoc.Sections[curSectionNum].Range.Paragraphs[2].Range;
//插入的表格会代替当前 range
//range 为非折叠时，TablesOfContents 会代替 range，引起小节数减少。
curRange.Collapse();
oDoc.TablesOfContents.Add(curRange, ref useheading_styles, ref upperheading_level,
ref lowerheading_level, ref usefields, ref tableid, ref RightAlignPageNumbers,
ref IncludePageNumbers, ref missing, ref missing, ref missing, ref missing);
oDoc.Sections[curSectionNum].Range.Paragraphs[1].Alignment =
MsWord.WdParagraphAlignment.wdAlignParagraphCenter;
oDoc.Sections[curSectionNum].Range.Paragraphs[1].Range.Font.Bold = 1;
oDoc.Sections[curSectionNum].Range.Paragraphs[1].Range.Font.Name = " 黑体";
oDoc.Sections[curSectionNum].Range.Paragraphs[1].Range.Font.Size = 16;
#endregion 目录
```

3.4.4 插入第一章正文并设置格式

Word 文档可以具有文档结构，这是通过设置各级标题来实现的，有了各级标题，就可在文档结构视图中看到文档的组织了，各级标题的样式须事先设置。标题更要的作用是将标题内容自动插入作为目录，并且可自动更新。

```
#region 第一章
curSectionNum = 3;
oDoc.Sections[curSectionNum].Range.Paragraphs[1].Range.Select();
curRange = oDoc.Sections[curSectionNum].Range.Paragraphs[1].Range;
Console.WriteLine(" 正在设置标题样式");
object wdFontSizeIndex;
wdFontSizeIndex = 14;//此序号在 word 中的编号是格式 > 显示格式 > 样式和格式 > 显示所有样式的序号
//14 即是标题一一级标题：三号黑体。
oWordApplic.ActiveDocument.Styles.get_Item(ref wdFontSizeIndex).ParagraphFormat.Alignment=
  MsWord.WdParagraphAlignment.wdAlignParagraphCenter;
oWordApplic.ActiveDocument.Styles.get_Item(ref wdFontSizeIndex).Font.Name = " 黑体";
oWordApplic.ActiveDocument.Styles.get_Item(ref wdFontSizeIndex).Font.Size = 16;//三号
wdFontSizeIndex = 15;//15 即是标题二二级标题：小三号黑体。
oWordApplic.ActiveDocument.Styles.get_Item(ref wdFontSizeIndex).Font.Name = " 黑体";
oWordApplic.ActiveDocument.Styles.get_Item(ref wdFontSizeIndex).Font.Size = 15;//小三
//用指定的标题来设定文本格式
object Style1 = MsWord.WdBuiltinStyle.wdStyleHeading1;//一级标题：三号黑体。
object Style2 = MsWord.WdBuiltinStyle.wdStyleHeading2;//二级标题：小三号黑体。
oDoc.Sections[curSectionNum].Range.Select();
currentSelection = oWordApplic.Selection;
//读入第一章文本信息
StreamReader file_content = new StreamReader("content.txt");
one_str = file_content.ReadLine();//一级标题
currentSelection.TypeText(one_str);
currentSelection.TypeParagraph(); //添加段落标记
one_str = file_content.ReadLine();//二级标题
currentSelection.TypeText(one_str);
currentSelection.TypeParagraph(); //添加段落标记
one_str = file_content.ReadLine();//正文
while (one_str != null)
{
  currentSelection.TypeText(one_str);
  currentSelection.TypeParagraph(); //添加段落标记
  one_str = file_content.ReadLine();//正文
```

```
}
file_content.Close();
//段落的对齐方式
curRange = oDoc.Sections[curSectionNum].Range.Paragraphs[1].Range;
curRange.set_Style(ref Style1);
oDoc.Sections[curSectionNum].Range.Paragraphs[1].Alignment =
MsWord.WdParagraphAlignment.wdAlignParagraphCenter;
curRange = oDoc.Sections[curSectionNum].Range.Paragraphs[2].Range;
curRange.set_Style(ref Style2);
//第一章正文文本格式
for (int i = 3; i < oDoc.Sections[curSectionNum].Range.Paragraphs.Count; i++)
{
  curRange = oDoc.Sections[curSectionNum].Range.Paragraphs[i].Range;
  curRange.Select();
  curRange.Font.Name = " 宋体";
  curRange.Font.Size = 12;
  oDoc.Sections[curSectionNum].Range.Paragraphs[i].LineSpacingRule =
  MsWord.WdLineSpacing.wdLineSpaceMultiple;
  //多倍行距，1.25 倍，这里的浮点值是以 point 为单位的，不是行距的倍数
  oDoc.Sections[curSectionNum].Range.Paragraphs[i].LineSpacing = 15f;
  oDoc.Sections[curSectionNum].Range.Paragraphs[i].IndentFirstLineCharWidth(2);
}
#endregion 第一章
```

3.4.5 插入表格并设置表格格式

Table 对象操作 Word 文档中的表格，Add 方法按指定行列数插入新表格。正面代码通过 Table 对象的 Cell 属性设置其内容，并设置了表格的边框线型等。

```
Console.WriteLine(" 正在插入第二章内容");
#region 第二章表格
curSectionNum = 4;
oDoc.Sections[curSectionNum].Range.Select();
curRange = oDoc.Sections[curSectionNum].Range.Paragraphs[1].Range;
currentSelection = oWordApplic.Selection;
currentSelection.TypeText("2 表格");
currentSelection.TypeParagraph();
currentSelection.TypeText(" 表格示例");
currentSelection.TypeParagraph();
currentSelection.TypeParagraph();
curRange = oDoc.Sections[curSectionNum].Range.Paragraphs[3].Range;
```

```
oDoc.Sections[curSectionNum].Range.Paragraphs[3].Range.Select();
currentSelection = oWordApplic.Selection;
MsWord.Table oTable;
oTable = curRange.Tables.Add(curRange, 5, 3, ref missing, ref missing);
oTable.Range.ParagraphFormat.Alignment =
MsWord.WdParagraphAlignment.wdAlignParagraphCenter;
oTable.Range.Font.Name = " 宋体";
oTable.Range.Font.Size = 16;
oTable.Range.Cells.VerticalAlignment =
MsWord.WdCellVerticalAlignment.wdCellAlignVerticalCenter;
oTable.Range.Rows.Alignment = MsWord.WdRowAlignment.wdAlignRowCenter;
oTable.Columns[1].Width = 80;
oTable.Columns[2].Width = 180;
oTable.Columns[3].Width = 80;
oTable.Cell(1, 1).Range.Text = " 字段";
oTable.Cell(1, 2).Range.Text = " 描述";
oTable.Cell(1, 3).Range.Text = " 数据类型";
oTable.Cell(2, 1).Range.Text = "ProductID";
oTable.Cell(2, 2).Range.Text = " 产品标识";
oTable.Cell(2, 3).Range.Text = " 字符串";
oTable.Borders.InsideLineStyle = MsWord.WdLineStyle.wdLineStyleSingle;
oTable.Borders.OutsideLineStyle = MsWord.WdLineStyle.wdLineStyleSingle;
curRange = oDoc.Sections[curSectionNum].Range.Paragraphs[1].Range;
curRange.set_Style(ref Style1);
curRange.ParagraphFormat.Alignment =
  MsWord.WdParagraphAlignment.wdAlignParagraphCenter;
#endregion 第二章
```

3.4.6 插入图片

图 3-2 在 Word 文档中添加的图片

下面程序在 Word 文档中插入图3-2图片文件。

```
Console.WriteLine(" 正在插入第三章内容");
#region 第三章图片
curSectionNum = 5;
oDoc.Sections[curSectionNum].Range.Paragraphs[1].Range.Select();
```

```
curRange = oDoc.Sections[curSectionNum].Range.Paragraphs[1].Range;
currentSelection = oWordApplic.Selection;
currentSelection.TypeText("3 图片");
currentSelection.TypeParagraph();
currentSelection.TypeText(" 图片示例");
currentSelection.TypeParagraph();
currentSelection.InlineShapes.AddPicture(@"D:\stu\cword\zsc-logo.png",
ref missing, ref missing, ref missing);
curRange = oDoc.Sections[curSectionNum].Range.Paragraphs[1].Range;
curRange.set_Style(ref Style1);
curRange.ParagraphFormat.Alignment =
  MsWord.WdParagraphAlignment.wdAlignParagraphCenter;
#endregion 第三章
```

3.4.7 设置各小节的页眉页脚

在正确划分小节的情况下，文档的页眉页脚能较好反映文档的不同区域，通过改变视图方式可在页眉页脚或正文编辑状态中进行切换。下面对每小节的页码进行了格式设置。

```
Console.WriteLine(" 正在设置第一节摘要页眉内容");
//设置页脚 section 1 摘要
curSectionNum = 1;
oDoc.Sections[curSectionNum].Range.Select();
//进入页脚视图
oWordApplic.ActiveWindow.ActivePane.View.SeekView =
MsWord.WdSeekView.wdSeekCurrentPageFooter;
oDoc.Sections[curSectionNum].
  Headers[MsWord.WdHeaderFooterIndex.wdHeaderFooterPrimary].
  Range.Borders[MsWord.WdBorderType.wdBorderBottom].LineStyle =
MsWord.WdLineStyle.wdLineStyleNone;
oWordApplic.Selection.HeaderFooter.PageNumbers.RestartNumberingAtSection = true;
oWordApplic.Selection.HeaderFooter.PageNumbers.NumberStyle
= MsWord.WdPageNumberStyle.wdPageNumberStyleUppercaseRoman;
oWordApplic.Selection.HeaderFooter.PageNumbers.StartingNumber = 1;
//切换到文档
oWordApplic.ActiveWindow.ActivePane.View.SeekView =
MsWord.WdSeekView.wdSeekMainDocument;
Console.WriteLine(" 正在设置第二节目录页眉内容");
//设置页脚 section 2 目录
curSectionNum = 2;
oDoc.Sections[curSectionNum].Range.Select();
```

```
//进入页脚视图
oWordApplic.ActiveWindow.ActivePane.View.SeekView =
MsWord.WdSeekView.wdSeekCurrentPageFooter;
oDoc.Sections[curSectionNum].
   Headers[MsWord.WdHeaderFooterIndex.wdHeaderFooterPrimary].
   Range.Borders[MsWord.WdBorderType.wdBorderBottom].LineStyle =
MsWord.WdLineStyle.wdLineStyleNone;
oWordApplic.Selection.HeaderFooter.PageNumbers.RestartNumberingAtSection = false;
oWordApplic.Selection.HeaderFooter.PageNumbers.NumberStyle
= MsWord.WdPageNumberStyle.wdPageNumberStyleUppercaseRoman;
//oWordApplic.Selection.HeaderFooter.PageNumbers.StartingNumber = 1;
//切换到文档
oWordApplic.ActiveWindow.ActivePane.View.SeekView =
MsWord.WdSeekView.wdSeekMainDocument;
//第一章页眉页码设置
curSectionNum = 3;
oDoc.Sections[curSectionNum].Range.Select();
//切换入页脚视图
oWordApplic.ActiveWindow.ActivePane.View.SeekView =
   MsWord.WdSeekView.wdSeekCurrentPageFooter;
currentSelection = oWordApplic.Selection;
curRange = currentSelection.Range;
//本节页码不续上节
oWordApplic.Selection.HeaderFooter.PageNumbers.RestartNumberingAtSection = true;
//页码格式为阿拉伯
oWordApplic.Selection.HeaderFooter.PageNumbers.NumberStyle
= MsWord.WdPageNumberStyle.wdPageNumberStyleArabic;
//起如页码为 1
oWordApplic.Selection.HeaderFooter.PageNumbers.StartingNumber = 1;
//添加页码域
object fieldpage = MsWord.WdFieldType.wdFieldPage;
oWordApplic.Selection.Fields.Add(oWordApplic.Selection.Range,
   ref fieldpage, ref missing, ref missing);
//居中对齐
oWordApplic.Selection.ParagraphFormat.Alignment =
MsWord.WdParagraphAlignment.wdAlignParagraphCenter;
//本小节不链接到上一节
oDoc.Sections[curSectionNum].Headers[Microsoft.Office.Interop.
Word.WdHeaderFooterIndex.wdHeaderFooterPrimary].LinkToPrevious = false;
//切换入正文视图
```

```
oWordApplic.ActiveWindow.ActivePane.View.SeekView =
  MsWord.WdSeekView.wdSeekMainDocument;
```

3.4.8 Word 文档保存

完成 Word 文档文本内容输入和格式设置后，还有一项重要的工作是对目录执行更新操作才可获得正确目录内容。最后执行 Document 对象的 SaveAs 方法按指定文件名实现保存任务。

```
//在输入全部内容后由于后面章节内容有变动，在此要更新目录，使页码正确
Console.WriteLine(" 正在更新目录");
oDoc.Fields[1].Update();
#region 保存文档
//保存文档
Console.WriteLine(" 正在保存 Word 文档");
object fileName;
fileName = doc_file_name;
oDoc.SaveAs2(ref fileName);
oDoc.Close();
```

Word 文档任务完成后，需要释放 Document 对象和 Application 对象。 Console.WriteLine(" 正在释放 COM 资源");

```
//释放 COM 资源
System.Runtime.InteropServices.Marshal.ReleaseComObject(oDoc);
oDoc = null;
oWordApplic.Quit();
System.Runtime.InteropServices.Marshal.ReleaseComObject(oWordApplic);
oWordApplic = null;
```

3.4.9 终止 Word 进程

在调试程序时可能会发生异常，这时的 WINWORD 进程将失去控制，可在 finally 语法块中终止 WINWORD 进程。

```
finally
{
  Console.WriteLine(" 正在结束 Word 进程");
  //关闭 word 进程
  System.Diagnostics.Process[] AllProces = System.Diagnostics.Process.GetProcesses();
  for (int j = 0; j < AllProces.Length; j++)
  {
    string theProcName = AllProces[j].ProcessName;
    if (String.Compare(theProcName, "WINWORD") == 0)
    {
      if (AllProces[j].Responding && !AllProces[j].HasExited)
```

```
        {
            AllProces[j].Kill();
        }
      }
    }
    //Close word Process.
}
```

请注意本示例中的代码因为排版问题，过长的代码被截断至下行了，实际应为一行。

3.5 作业

1. 实现在文档中添加艺术字。
2. 在原来的程序基础上添加新的章节内容。
3. 向每小节页眉添加文本信息。