第 7 讲外部文件操作

目录

1 概述

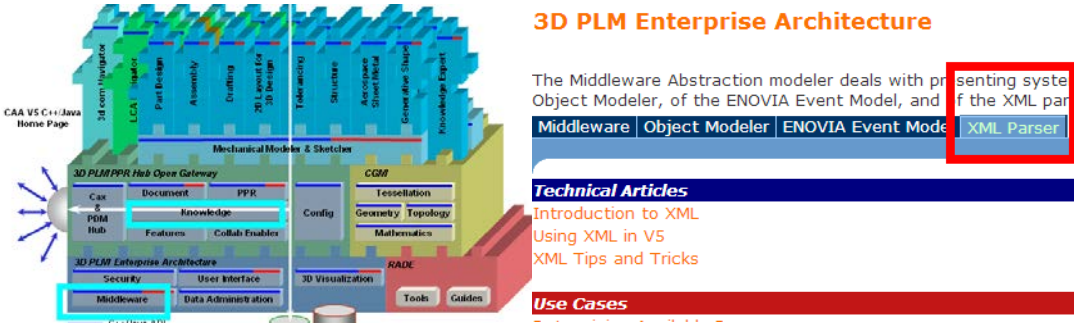
实现 xml，xls，txt 外部文件的读写操作。

从 MultiList 中导出到 D：\Tree.xml 文件。导出.xls 文件，导出 txt 文件

```
<?xml version="1.0" ?>
- <Tree>
  - <Products>
      <Product name="======All Children List====" />
      <Product name="Part1 , Part1.1" />
      <Product name="Part1 , Part1.1" />
      <Product name="Product3 , Product3.1" />
      <Product name="Part1 , Part1.1" />
      <Product name="Part1 , Part2.1" />
      <Product name="Product2 , Product2.1" />
      <Product name="======Direct Children List====" />
      <Product name="Part1 , Part1.1" />
      <Product name="Part1 , Part2.1" />
      <Product name="Product2 , Product2.1" />
      <Product name="======All Children List====" />
      <Product name="Part1 , Part1.1" />
      <Product name="================Part Structure==============" />
      <Product name="---------CATIMechanicalTool---------" />
      <Product name="零件几何体" />
```

2 参考资料

在 CATIA 中 XML 文件作为一种可读写的外部数据能够灵活的处理多种文本编辑任务。关于 xml 文件的叙述主要集中在 3D PLM Enterprise Architecture 中的 MiddleWare 模块，在其中详细讲解了 xml 解析器的构建方法。



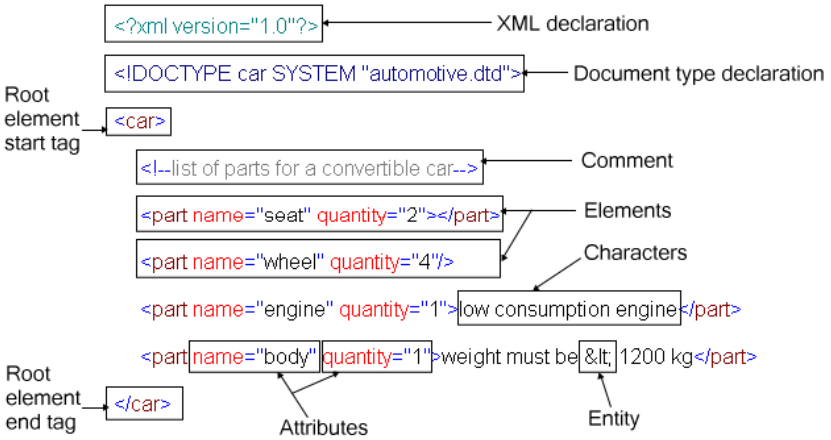
Xls、txt 作为设计表的对象，在 knowledge 模块中有介绍。

3 XML 文件

Xml 是可扩展标记语言，标准通用标记语言的子集，是一种用于标记电子文件使其具有结构性的标记语言。Xml 有诸多优势使得它成为使用广泛的一种标记语言，优势之一是具有简单的定义方式，可以采用文本编辑器读写 xml 文件。优势之二是支持 DTD 或 XSD 结构。优势之三为 w3c 定义的标准的超文本数据传输工具。

3.1 Xml 文件结构

Xml 文件由元素（elements），一般有一个开始标签（<element_name>）和一个结尾标签定义（</element_name>），元素可以包含自由文本和其他的元素。每个 xml 文档都具有一个 root element，它包含了其他的元素。如图所示。



大部分 xml 文档都具有以下的结构：

Xml 声明——声明该文件为 xml 文件，采用的 xml 版本是 1.0，文本的编码方式为 UTF-8。

文件类型声明——说明了采用 DTD 模式，root element 是“car”，这个词存储在 automotive.dtd 文件中，这个文件是 xml 的一个内联文件。

Root element——一个 xml 文件有且只有一个 root element，这儿是“car”。

Elements——一个 element 有一个开始标签和一个结束标签定义。如果 element 下边没有包含其他 element，那么可以只用开始标签，结束标签用“/”代替。比如“wheel”所在的 element。

属性 Attributes——开始标签可以包含属性，比如 name，value。一个 element 中的属性名称唯一，属性的值写在引号中，如第四个 part element 的 name 属性

为 body；

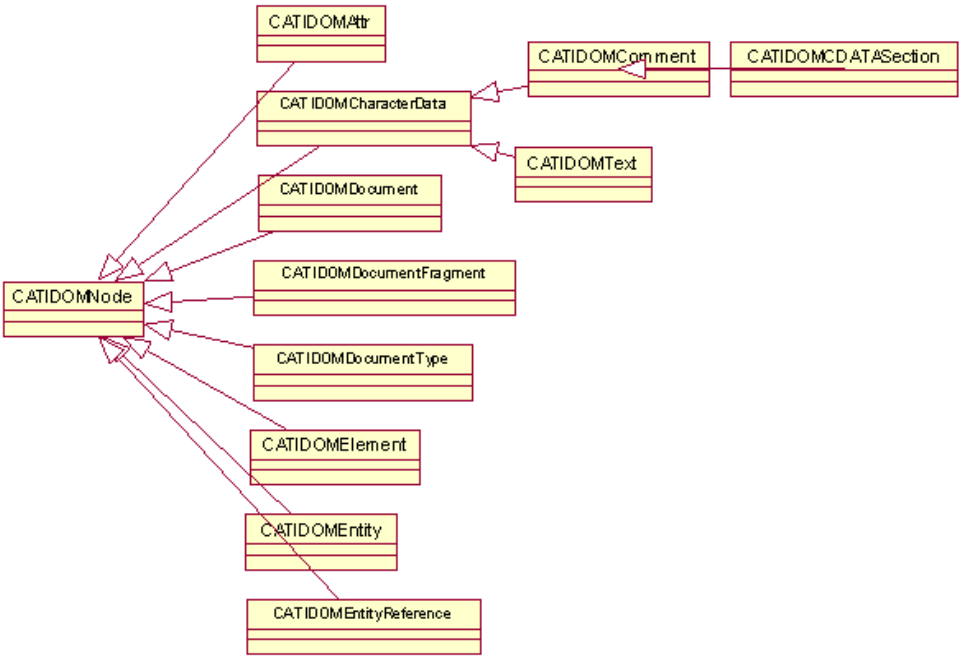
文本 Characters——xml 的 elements 中可以包含自由文本。

注释 comments——注释用”!--”开始，以“--”结束。

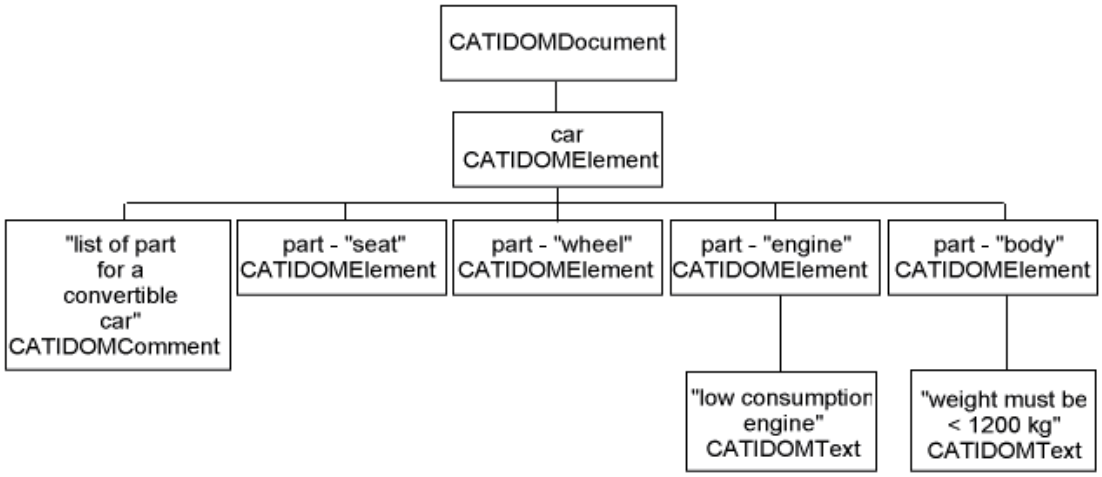
3.2 采用 DOM 应用接口操作 XML 文件

CATIA 中读取 XML 的方式有多种，包括 Well-formedness and Validity、De Jure Standards (W3C)、De Facto Standards: SAX、A Document-oriented API: DOM、An Event-oriented API: SAX。这里采用 DOM 应用接口的方式对 xml 进行读取和写入操作。

DOM(Document Object Model)应用接口采用面向对象的方式描述一个 xml 文档，DOM 定义了 xml 语言中每个结构的接口：elements，attributes，documents，characters，comments 等。这些接口本身具有上下级关系，级别关系定义在了 node 节点类中，关系如下图所示：



DOM 应用接口把 xml 文档看做一个 xml 节点的树，xml 文档的 root element 就是 DOM 节点树的 root node，root element 下的子 elements 就是 root node 的子集（children）。如图所示：



DOM 应用接口定义了多种方法解析 xml 文件（xml 解析成为 DOM 树加载在内存中），操作 xml 文件（插入 elements，编辑属性值，复制、删除子节点等），写入 xml 文档（将从内存中的 DOM 树写入到 xml 文件并保存）。

用 DOM 应用接口处理 xml 的所有接口和类如下：

```
#include"CATUnicodeString.h"//处理字符串
#include"CATIDOMDocument.h"//处理xml文件
#include"CATIXMLDOMDocumentBuilder.h"//DOMDocBuilder
#include"CATIDOMElement.h"//elements
#include"CATIDOMNode.h"//Dom 节点
#include"CATIDOMAttr.h"//节点属性
#include"CATIDOMImplementation.h"//DOM实现
#include"CATIDOMDocumentType.h"//DOM文件类型
#include"CATUnicodeString.h"
#include"CATCollec.h"
#include"CATIDOMNodeList.h"//DOM节点联表
#include"CATIDOMCharacterData.h"//DOM自由字符
#include"CATIDOMText.h"//DOM 文本
#include"CATIDOMNamedNodeMap.h"//DOM节点和名称映射
#include"CATIDOMAttr.h"//DOM属性
```

3.3 数据保存为 xml 文件实现

定义了两个类专门用来处理 xml 文件的读写操作。它分别是 XMLHandler 和它的一个子类 CaseXMLHandler。

CasexmlHandler 的构造函数如下：

```
CaseXMLHandler::CaseXMLHandler(CATUnicodeString iXMLFilePath,
CATUnicodeString iXMFileName)
{
   //创建文件到当前程序目录下
```

```
    m_XMLFilePath = iXMLFilePath + "\\"+ iXMFileName;
    CATLibStatus State=CATFileAccess(m_XMLFilePath,R_OK);//判断是否已
经有m_XMLFilePath存在，
    if (State == CATLibError)//xml文件不存在
    {
        //若找不到文件，则首先创建一个XML文件
        HRESULT hr
= ::CreateCATIXMLDOMDocumentBuilder(m_spDomBuilder);//获得
XMLDOMDocumentBuilder
        if (FAILED(hr)|| NULL_var == m_spDomBuilder )
        {
            return ;
        }
        CATIDOMImplementation_var spImplementation=NULL_var;
        hr = m_spDomBuilder->GetDOMImplementation(spImplementation);//
获得DOMImplement
        if ( FAILED(hr) || NULL_var == spImplementation )
        {
            return ;
        }
        hr = spImplementation->CreateDocument("", "Tree",NULL_var,
m_spDomDocument);//创建根节点声明
        if ( FAILED(hr) || NULL_var == m_spDomDocument )
        {
            return ;
        }
        CATIDOMElement_var spCurDomElement = NULL_var;
        hr = m_spDomDocument->CreateElement("Tree",spCurDomElement);//
创建根节点
        if ( FAILED(hr) || NULL_var == spCurDomElement )
        {
            return ;
        }
        CATIDOMElement_var spCasesNode = NULL_var;
        hr = m_spDomDocument->CreateElement("Products", spCasesNode);//
创建一个products节点
        if ( NULL_var == spCasesNode )
        {
            return ;
        }
        m_spDomDocument->AppendChild(spCurDomElement);//设置根节点节是
文件的子集
        spCurDomElement->AppendChild(spCasesNode);//设置products节点是
根节点的子集
```

```
        if(m_spDomBuilder!=NULL_var)
        {//设置编码方式
            CATListOfCATUnicodeString option,optionValue;
            option.Append("CATEncoding");
            optionValue.Append("UTF-8");
            option.Append("CATSortAttributes");
            optionValue.Append("true");
            HRESULT hr =
m_spDomBuilder->WriteToFile(m_spDomDocument,m_XMLFilePath ,option,opt
ionValue);//保存文件
            option.RemoveAll();optionValue.RemoveAll();
        }
    }
    else//xml文件存在
    {
        HRESULT hr = ::CreateCATIXMLDOMDocumentBuilder(m_spDomBuilder);
        if (FAILED(hr) || NULL_var == m_spDomBuilder )
        {
            return ;
        }
        CATListOfCATUnicodeString readOptions;//设置打开方式
        readOptions.Append("CATDoValidation");
        CATListOfCATUnicodeString readOptionValues;
        readOptionValues.Append("false");
        hr = m_spDomBuilder->Parse(m_XMLFilePath, m_spDomDocument,
readOptions, readOptionValues);//打开xml文件
        if ( FAILED(hr) || NULL_var == m_spDomDocument )
        {
            return;
        }
    }
}
```

将 Multilist 中的元素添加到节点的代码如下：

```
HRESULT CaseXMLHandler::AddCaseIntoXML(  CATUnicodeString iCaseName)
{
    if ( NULL_var == m_spDomDocument )//m_spDomDocument在XMLHandler.h
中声明
    {
        return E_FAIL;
    }
    CATIDOMElement_var spRootElement = NULL_var;
    GetRootDomNode( m_spDomDocument, spRootElement );//获取根节点
    if ( NULL_var == spRootElement )
```

```
    {
        return E_FAIL;
    }
    //1。首先获得Products节点
    CATIDOMElement_var spDOMCasesElement = NULL_var;
    HRESULT rc = GetNodeFromRoot( spRootElement, "Products",
spDOMCasesElement);
    if ( FAILED(rc) || NULL_var == spDOMCasesElement )
    {
        return E_FAIL;
    }
    //2. 查找其中是否有相同的节点
    //添加<Product>节点
    CATIDOMElement_var spNewCaseNode = NULL_var;
    m_spDomDocument->CreateElement("Product", spNewCaseNode);
    if ( NULL_var == spNewCaseNode )
    {
        return E_FAIL;
    }
    spDOMCasesElement->AppendChild(spNewCaseNode);//将新建的节点插入
根节点下
    //设置Product节点的属性name
    spNewCaseNode->SetAttribute("name", iCaseName);
        //写入文件
        m_spDomBuilder->WriteToFile( m_spDomDocument, m_XMLFilePath );
    return S_OK;
}
```

给“输出 xml 文件”按钮添加回调函数的代码：

```
void
BITCreateAssemDlg::OnPushButtonXMLPushBActivateNotification(CATComman
d* cmd, CATNotification* evt, CATCommandClientData data)
{
    CaseXMLHandler* pXMLFile = new CaseXMLHandler("D:\\","Tree.XML");//
给定位置为D盘
    CATUnicodeString  oString;
    for(int i=0;i<_MultiList001->GetLineCount();i++)
    {
        _MultiList001->GetColumnItem( 0,   oString, i)  ;
        pXMLFile->AddCaseIntoXML(oString);//读取一个MultiList的行，输出
一个xml节点
    }
}
```

4 设计表

在 CATIA 中设计表可以用多种文件的方式表达比如：xls，txt，Lotus 文件。此处主要说明 txt 和 xls 形式的设计表。

4.1 xls 文件操作方法

在 CATIA 中 xls 文件主要用于设计表。操作方法主要体现在以下接口和类中：

```
#include"CATICkeParmFactory.h"//参数工厂
#include"CATIDesignTable.h"//设计表
#include"CATICkeParm.h"//参数
#include"CATCkeGlobalFunctions.h"//参数全局函数
#include"CATICkeFunctionFactory.h"//公式
#include"CATICkeSheet.h"//设计表的工作簿
```

Xls 文件的读写主要还是通过 Microsoft office Excel 来实现，也就是说通过启动 excel 来读写 xls 文件，这在一定程度上会降低程序效率，甚至会出错。

Xls 操作的步骤一般为：首先获取结构 container（specification container），然后给 CATICkeParmFactory 赋值，用 CATICkeParmFactory 创建设计表 CATIDesignTable，或者创建工作簿 CATICkeSheet 等。

若为读取 xls 文件，需要获取 Design table 在创建 Deign Table 的时候给定文件路径。然后用 CATIDesignTable::CellAsDouble 方法可以直接获取数值型内容，若获取其他类型的内容，如文本等需要借助 CATICkeSheet。从 Design Table 中获取 CATICkeSheet，在通过 CATICkeSheet::Cell（）等方法获取单元格中的内容。

若写入 xls 文件操作，则必须借助 CATICkeSheet。从 CATICkeParmFactory 创建工作簿 CATICkeSheet，给定保存文件的种类和路径，并用 CATICkeSheet::SetCell()方法即可将数据写入工作簿并保存。

4.2 xls 文件读写代码实例

```
int BITCreateAssemDlg::ReadAndWriteExcel(int ITCode,double Dia)
{
    HRESULT rc;
    CATFrmEditor* pEditor = CATFrmEditor::GetCurrentEditor();  //获得
Editor
    CATDocument *pDoc = pEditor->GetDocument();              //获得
Document
    if (  NULL ==pDoc  )
```

```
    {
        return 2;
    }
    CATInit *pInitOnDoc = NULL ;
    rc = pDoc->QueryInterface(IID_CATInit, (void**)&pInitOnDoc);//从doc
中获取prt文件的初始化
    if ( FAILED(rc) )
    {
        return 3 ;
    }
    pInitOnDoc->Init(TRUE);
    CATIContainerOfDocument_var spNewDocContainer = pDoc;
    CATIContainer *cont = NULL;
    rc = spNewDocContainer->GetSpecContainer(cont);//获取结构container
    if (FAILED(rc) || cont == NULL)
    {
        return 4 ;
    }
    CATICkeParmFactory_var spFact = cont;
    //读取xls
    CATUnicodeString CATpath="D:\\";//设置要读的文件位置
    constchar *path = CATpath.ConvertToChar();
    CATIDesignTable_var spDesign ;
    char file[300];
    strcpy(file, path);
    strcat(file, "/CAALifDesignTableAli.xls");//设置要读的文件名称
    spDesign = spFact->CreateDesignTable("DesignTable.1", "Design table
test", file);//通过路径和文件名创建设计表
    if(spDesign == NULL_var)
    {
        return 6;
    }
    double CurMax;
    double CurMin;
    CurMin = spDesign->CellAsDouble(2,2);//直接读取电子表格
    CurMax = spDesign->CellAsDouble(2,3);//直接读取电子表格
//写入xls
    CATICkeSheet_var spSheet =
spFact->CreateSheet("D:\\ExprotXlsPPP.xls",1);//创建一个要写入空xls文
件
    spSheet->SetOrientation (0);//设置列的方向向下
    spSheet->UpdateLocalCopy();
    spSheet->SetCell(1,1,"表格测试开始",1,0);//在单元格（,2）中写入“表
格测试”
```

```
    for(int i=0;i<_MultiList001->GetLineCount();i++)
    {
        CATUnicodeString oString;
        _MultiList001->GetColumnItem( 0,    oString, i)  ;
        constCATUnicodeString istring =(constCATUnicodeString ) oString;
        //spSheet->SetCell(1,1,(const CATUnicodeString )istring,1,0);
        spSheet->SetCell(1,i+2,istring,0,0);//在单元格中写入数据

    }
    spSheet->SetCell(1,_MultiList001->GetLineCount(),"表格测试结束
",0,1);//在单元格中写入
    spSheet->UpdateLocalCopy();
    return 8;
}
```

4.3 txt 文件操作方法

4.3.1 FILE 结构体操作 txt

CATIA 支持的多种 txt 操作方法。C 语言和 C++ 语言中都定义了结构体 FILE；所以可以直接用来读写 txt 文件。

4.3.2 iostream 读写 txt 的方式

C++中输入输出流可以用来操作 txt 文件，它们定义在头文件： "iostream.h" 和 "fstream.h"中。

4.3.3 以设计表的方式操作 txt

CATIA 中以设计表的方式操作 txt 文件。此操作过程与 xls 文件的操作过程几乎一样。只需要修改几个参数即可。

4.4 txt 文件读写操作代码实例

4.4.1 FILE 结构体写入 txt 文件代码实例

```
    //给指定的文本文件，输出指定的内容
void OutPutTxt(CATUnicodeString ifilePath,CATUnicodeString iTxt)
{
    char* ifile;
    char  LTxt[512];
    FILE*  pFile = NULL;
    ifile = (char *)ifilePath.ConvertToChar();//将CATUnicodeString转化
为char *;
```

```
    iTxt=(char *)iTxt.ConvertToChar();

    if ((pFile = fopen  (ifile,"w")) == NULL) return ;
    sprintf (LTxt,iTxt);
    fwrite ((char *)LTxt,sizeof(char),strlen(LTxt),pFile);
    if (pFile) fclose (pFile);
    pFile = NULL;
    return ;
}
```

4.4.2 用输入输出流的方式读写 txt 代码实例

```
    fileName=fileName+".txt";
    constchar *filepath=fileName.ConvertToChar();
    ofstream outfile(filepath,ios::out|ios::trunc);//用c++的输出文件
    if(!outfile)
    {
        return;
    }
    outfile<<"姓名 "<<"性别 "<<"年龄 "<<"喜欢学科   "<<endl;
    CATUnicodeString   name,gender,age,subject;
    int ColumnNum=_MultiListInformation->GetLineCount();
    for(int i=0;i<ColumnNum;i++)
    {
        //获取MultiListInformation中的信息
        _MultiListInformation->GetColumnItem(0, name,i);
        _MultiListInformation->GetColumnItem(1, gender,i);
        _MultiListInformation->GetColumnItem(3, age,i);
        _MultiListInformation->GetColumnItem(2, subject,i);
        //此处需要调整字符串的输出方式，使得txt格式整齐
        name.Resize(6,' ',0);
        gender.Resize(4,' ',0);
        age.Resize(4,' ',0);
        outfile<<name<<" "<<gender<<" "<<age<<" "<<subject<<" "<<endl;
    }
```

4.4.3 采用设计表的方式读写 txt 代码实例

```
int BITCreateAssemDlg::ReadAndWriteTXT (int ITCode,double Dia)
{
    HRESULT rc;
    CATFrmEditor* pEditor = CATFrmEditor::GetCurrentEditor();  //获得
Editor
    CATDocument *pDoc = pEditor->GetDocument();              //获得
Document
    if (  NULL ==pDoc  )
```

```
    {
        return 2;
    }
    CATInit *pInitOnDoc = NULL ;
    rc = pDoc->QueryInterface(IID_CATInit, (void**)&pInitOnDoc);//从doc
中获取prt文件的初始化
    if ( FAILED(rc) )
    {
        return 3 ;
    }
    pInitOnDoc->Init(TRUE);
    CATIContainerOfDocument_var spNewDocContainer = pDoc;
    CATIContainer *cont = NULL;
    rc = spNewDocContainer->GetSpecContainer(cont);//获取结构Container
    if (FAILED(rc) || cont == NULL)
    {
        return 4 ;
    }
    CATICkeParmFactory_var spFact = cont;
  CATICkeSheet_var spSheet =
spFact->CreateSheet("D:\\ExprotXlsPPP.txt",1);//直接为txt文件
  spSheet->SetOrientation (0);//设置列的方向为向下
CATUnicodeString oString;
    for(int i=0;i<_MultiList001->GetLineCount();i++)
    {
        _MultiList001->GetColumnItem( 0,   oString, i)  ;
        spSheet->SetCell(1,i+1,oString);//写入txt文件并自动保存
    }
    return 8;
}
```