Якунина М.В.

ПРАКТИЧЕСКОЕ ИСПОЛЬЗОВАНИЕ LIBREOFFICE BASIC

Оглавление

Введение

Офисные приложения используются для выполнения различных задач широкого спектра: от набора текста до создания многостраничного отчета. Такую работу можно назвать рутинной. Автоматизация такой работы позволит сэкономить рабочее время и облегчить трудоемкие процессы. Одним из инструментов такой автоматизации можно назвать макросы.

Макрос - сохраненная последовательность команд или сочетаний клавиш, которые передаются приложению или нескольким приложениям для последовательного выполнения.

По сути, макрос представляет собой программу или набор инструкций в виде обычного текста, написанном на каком-либо языке программирования.

LibreOffice предоставляет возможность написания макросов на 4-х языках программирования:

- LibreOffice Basic.
- Python.
- BeanShell.
- JavaScript.

Наиболее используемым является LibreOffice Basic, синтаксис которого похож на синтаксис языков Microsoft Visual Basic for Applications и Microsoft Visual Basic.

Язык программирования LibreOffice Basic может быть разделен на четыре компонента:

- Язык LibreOffice Basic: определяет элементарные лингвистические конструкции, например, для определения переменных, циклов и функций.
- Библиотека времени выполнения: обеспечивает стандартные функции: например, функции для редактирования чисел, строк, значений данных, и файлов.
- API (Интерфейс прикладного программирования): обеспечивает доступ к документам LibreOffice и позволяет их создавать, сохранять, изменять, и печатать.
- Редактор Диалогов: позволяет создавать диалоговые окна и обеспечивает возможность добавления элементов управления и обработчиков событий.

Справочная информация по API, доступна в Интернете по адресу: http://api.openoffice.org/docs/common/ref/com/sun/star/module-ix.html.

Создание нового макроса

Для получения доступа к среде разработки макросов нужно из любого приложения в главном меню выбрать **Сервис→Макросы→Управление макросами→LibreOffice Basic…**

Откроется диалоговое окно **Макросы LibreOffice Basic** (Рис. 1).

В левой части окна находятся объекты, содержащие библиотеки. Такие объекты называются контейнерами библиотек. А в правой части - макросы, уже существующие в выделенном модуле выбранной библиотеки.

Каждый вновь созданный документ LibreOffice уже имеет пустую библиотеку **Standard**.

Выберите документ, в котором хотите создать макрос и нажмите кнопку **Создать**. В открывшемся диалоговом окне **Новый модуль** введите имя нового макроса (Рис. 2).

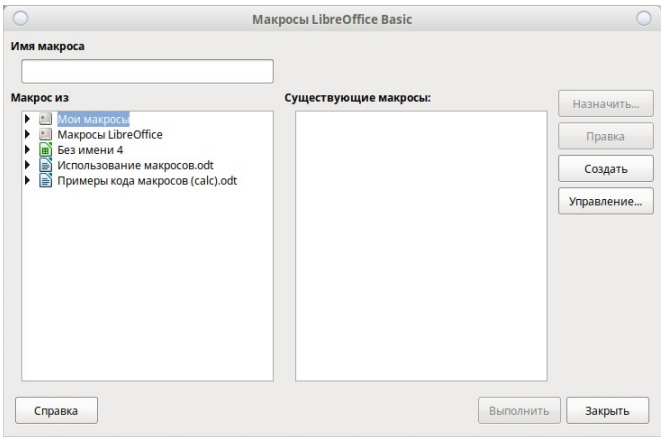


Рис. 1. Диалоговое окно Макросы LibreOffice Basic

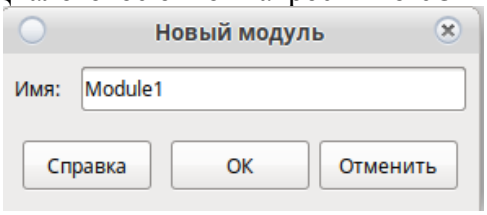


Рис. 2. Диалоговое окно Новый модуль

Нажав на кнопку **ОК** откроется редактор LibreOffice Basic (Рис. 3).

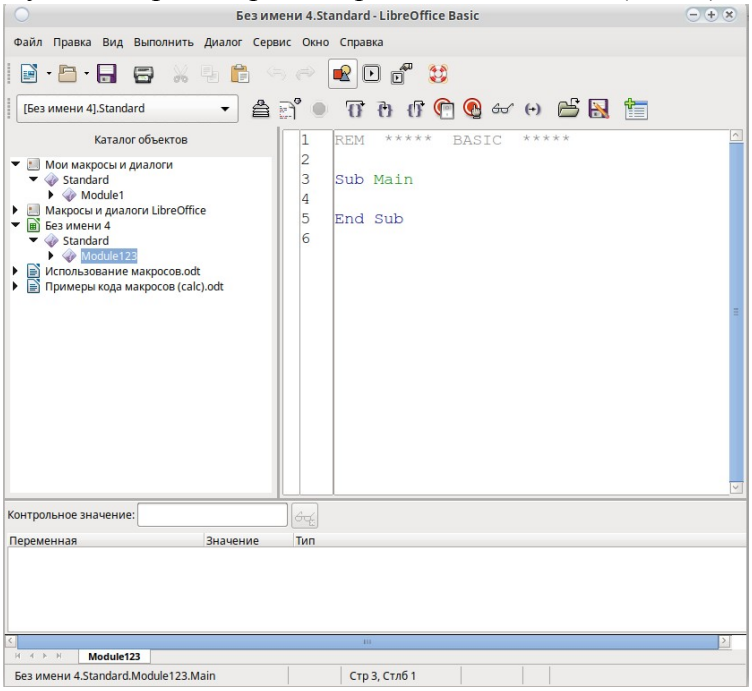


Рис. 3. Окно среды разработки

В окне среды разработки уже будет создана пустая процедура с именем Main (или Macro1, если модуль библиотеки уже содержит процедуру с именем Main).

Любая процедура должна начинаться с ключевого слова *Sub*, после которого через пробел следует ее имя и заканчиваться словами *End Sub*. Весь исполняемый код должен находиться внутри процедуры, т. е. между *Sub Main и End Sub.*

Например, выведем предложение «Мой первый макрос» в диалоговое окно.

Для этого в тело процедуры поместите инструкцию *MsgBox "Мой первый макрос"*. Для выполнения макроса нужно поместить курсор в текст процедуры и нажать кнопку **Выполнить** (или клавишу **F5)**. Результат представлен на рисунке 4.

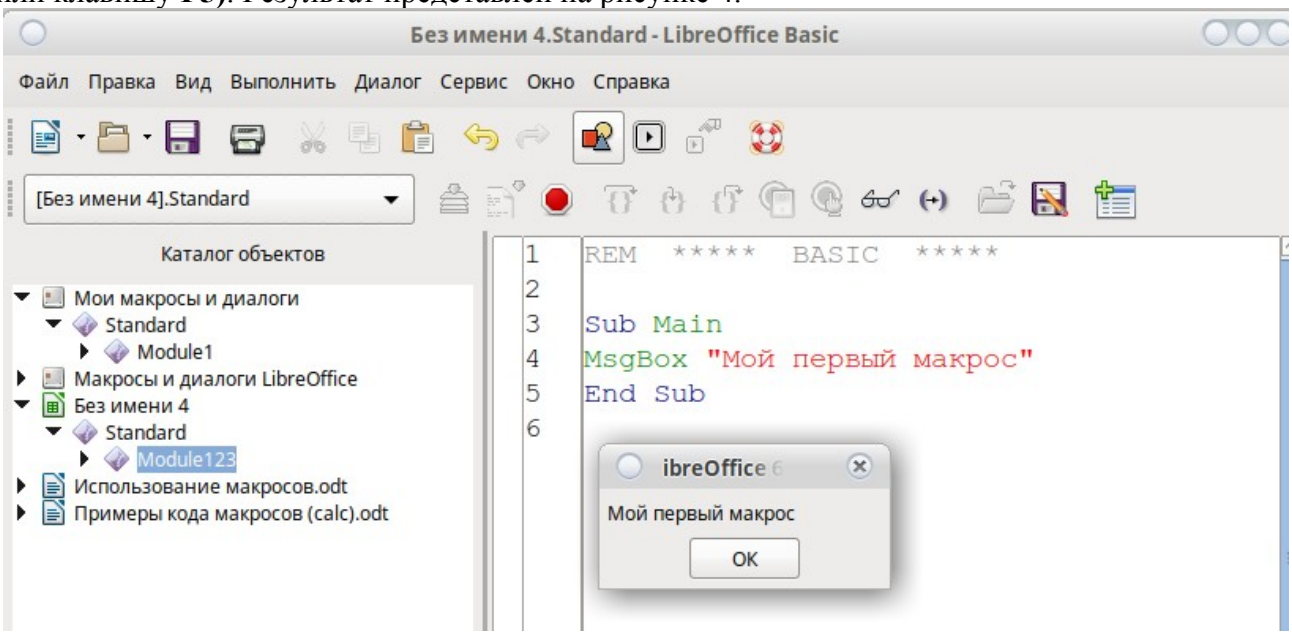


Рис. 4. Окно среды разработки и результат работы макроса

Инструкция *MsgBox* отображает диалоговое окно с сообщением.

Полный синтаксис инструкции:

MsgBox (Текст As String [,Тип As Integer [,Заголовок As String]]).

Где *Текст* - строковое значение (текст, заключенный в кавычки, или переменная типа String), выводимое в окне сообщения, *Тип* - необязательный параметр, представленный целым числом. Он указывает тип окна сообщения и набор его кнопок. Когда данный параметр отсутствует, его значение считается равным нулю. *Заголовок* - также необязательный строковый параметр, содержит текст, отображаемый в заголовке окна сообщения. Указатели типа параметра (*As String*, *As Integer*) писать не нужно, в описании синтаксиса они присутствуют с единственной целью - обозначить тип параметра: *As String* указывает на то, что данный параметр инструкции должен быть строкой, а *As Integer* - на то, что параметр должен быть целым числом.

Возможные значения параметра *Тип.*

Набор кнопок:

- 0. Показать кнопку «ОК»
- 1. Показать кнопки «ОК» и «Отмена»
- 2. Показать кнопки «Прервать», «Повторить» и «Пропустить»
- 3. Показать кнопки «Да», «Нет» и «Отмена»
- 4. Показать кнопки «Да» и «Нет»
- 5. Показать кнопки «Повторить» и «Отмена»

Тип окна:

- 16. Тип «Останов»
- 32. Тип «Вопрос»
- 48. Тип «Предупреждение»
- 64. Тип «Информация»

Кнопка по умолчанию:

- 128. Первая кнопка в диалоговом окне как кнопка по умолчанию
- 256. Вторая кнопка в диалоговом окне как кнопка по умолчанию
- 512. Третья кнопка в диалоговом окне как кнопка по умолчанию

Эти значения являются набором битовых флагов, т. е. для комбинирования достаточно указать их сумму. Например, если необходимо, чтобы сообщение об ошибке с предложением прервать работу макроса сопровождалось соответствующим системным звуком и значком ошибки, а также имело две кнопки, причем по умолчанию была бы выбрана кнопка Отмена, то параметр Тип должен быть равен 273 (1+16+256).

В листинге 1 представлен код инструкций MsgBox с различными параметрами.

Листинг 1. Инструкция MsgBox с различными параметрами

```
Sub Main
    MsgBox "Только кнопка OK", 0, "Заголовок"
    MsgBox ("Только кнопка OK", 0, "Заголовок")
    MsgBox "Кнопки OK и Отмена", 1, "Заголовок"
    MsgBox "Кнопки Да и Нет", 4, "Заголовок"
    MsgBox "Останов", 16, "Заголовок"
    MsgBox "Вопрос", 32, "Заголовок"
    MsgBox "Предупреждение", 48, "Заголовок"
    MsgBox "Информация", 64, "Заголовок"
    MsgBox "Останов и кнопки Да, Нет и Отмена", 16 + 3, "Заголовок"
    MsgBox "Информация и кнопка OK", 64 + 3, "Заголовок"
    MsgBox "Вопрос и кнопки OK и Отмена." & _
    "Кнопка Отмена установлена по умолчанию.", 32 + 1 + 256, "Заголовок"
    MsgBox "То же, что и предыдущее сообщение.", 289, "Заголовок"
End Sub
```

Объявление переменных

LibreOffice Basic не требует явного описания используемых переменных, т. е. можно использовать переменные не объявляя их. Это удобно, но может приводить к возникновению ошибки. Например, если имя переменной было введено неправильно (опечатка), то она становиться новой переменной, вместо того, чтобы вызвать ошибку.

Можно объявить переменную с или без указания типа. Переменная без явного указания типа становится переменной типа *Variant*, который в состоянии принять любой тип. Это означает, что можно использовать тип *Variant*, чтобы содержать числовое значение и затем, в следующей строке кода, перезаписать число текстом.

Для объявления переменных используется оператор *Dim*. Можно объявить несколько переменных в одной строке и каждой переменной задать тип при ее объявлении.

В таблице 1 представлены способы явного объявления переменных.

Таблица 1. Объявление простых переменных

Объявление	Описание
Dim Name	*Name* - типа *Variant*, потому что никакой тип не заявлен
Dim Name As String	*Name* - типа *String*, потому что тип явно заявлен
Dim Name$	*Name$* - типа *String*, потому что *Name$* заканчивается *$*
Dim Name As String, Weight As Single	*Name* - типа *String*, а *Weight* - типа *Single*
Dim Width, Length	*Width* и *Length* - типа *Variant*
Dim Weight, Height As Single	*Weight* - типа *Variant*, а *Height* - типа *Single*

Типы данных

Тип описывает множество значений, которое может принимать та или иная переменная.

Типизация - система организации данных, при которой каждой переменной назначается тип, указывающий практический смысл хранящегося в ней значения. Такой подход позволяет программе всегда знать, как отобразить те или иные данные на экране, какие операции с ними допустимы, а какие нет, и что нужно делать, выполняя каждую из них.

Для указания типа при объявлении переменной служит ключевое слово *As*, после которого следует ключевое слово, указывающее тип.

Например, объявим переменную *cT* как целое число:

Dim cT As Integer

После того как переменная бала объявлена, ей можно присвоить значение:

cT=5

Функции

В LibreOffice Basic кроме процедур можно использовать функции. Основное отличие функции от процедуры - это то, что она может возвращать результат работы в виде значения переменной. Функция должна начинаться ключевым словом *Function*, за которым следует имя, и заканчиваться словами *End Function.*

Обычно при использовании функции ей передаются некоторые параметры, над которыми в ней производятся действия.

Синтаксис функции обычно выглядит так:

Function FunctionName(Val1 As Type1, Val2 As Type2,...) As TypeFunctionResult

Код

FunctionName = Result

End Function

Где *FunctionName* - имя функции (уникальное в модуле); *Val1, Val2* - параметры, передаваемые в функцию; *Type1, Type2* — тип передаваемых значений; *TypeFunctionResult* - тип данных, возвращаемых функцией; *Код* - программный код, вычисляющий результат функции. В строчке *FunctionName = Result* результат, вычисленный в коде, присваивается функции.

Листинг 2. Пользовательская функция

```
Sub test
    MsgBox "Результат работы функции: " & TestFunc
End Sub

Function TestFunc As String
    TestFunc = "Привет"
End Function
```

На рисунке Рис. 5 приведен код и результат работы пользовательской функции.

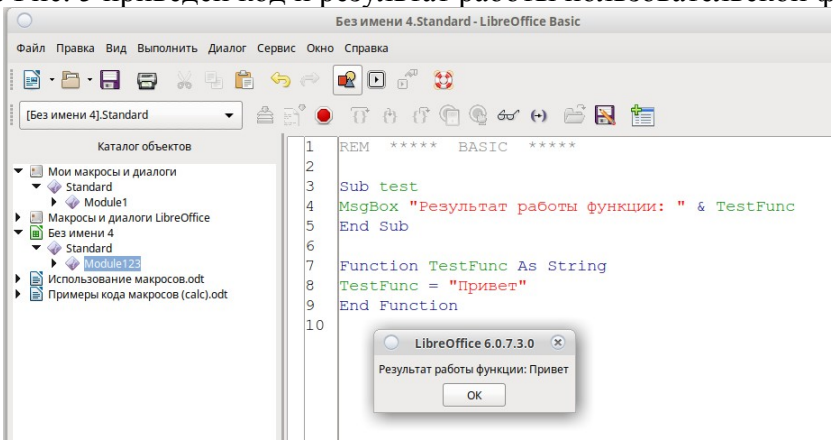


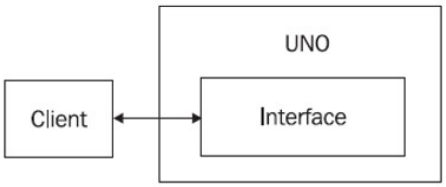
Рис. 5. В окне среды разработки пользовательская функция и результат ее работы

Универсальные сетевые объекты

LibreOffice Basic поддерживает универсальные сетевые объекты UNO (Universal Network Object).

Универсальный сетевой объект (UNO) - компонентная модель, которая предлагает возможность взаимодействия между различными языками программирования, моделями объектов, архитектурами машин и процессами. UNO подобен COM по функциональным возможностям. UNO применяется для управления внутренней работой LibreOffice, используя интерфейс прикладного программирования (API).

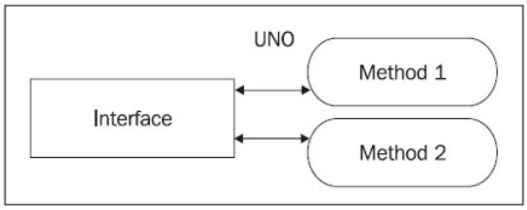
На простейшем уровне есть макрос (клиент), который соединяется с интерфейсом:



Интерфейс

Каждый интерфейс состоит из набора одного или нескольких методов, которые можно использовать:

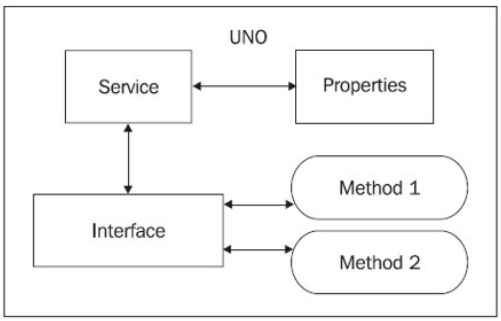
- Получать или устанавливать параметры.
- Управлять выполнением любых функциональных возможностей, которые интерфейс определяет.



Каждый интерфейс содержится в сервисе.

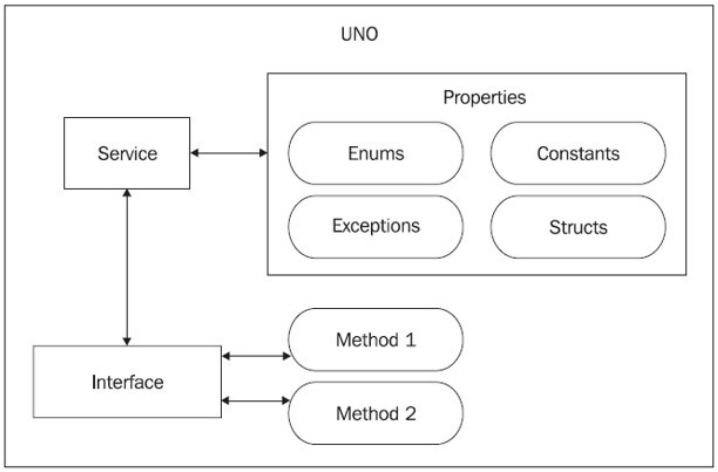
Сервис

Сервис - компонент UNO - компоновочный блок LibreOffice. Каждый сервис состоит из одного или более интерфейсов и имеет ряд типов, связанных с ним (интерфейс также является типом):



Существует 4-е типа, связанных с сервисами:

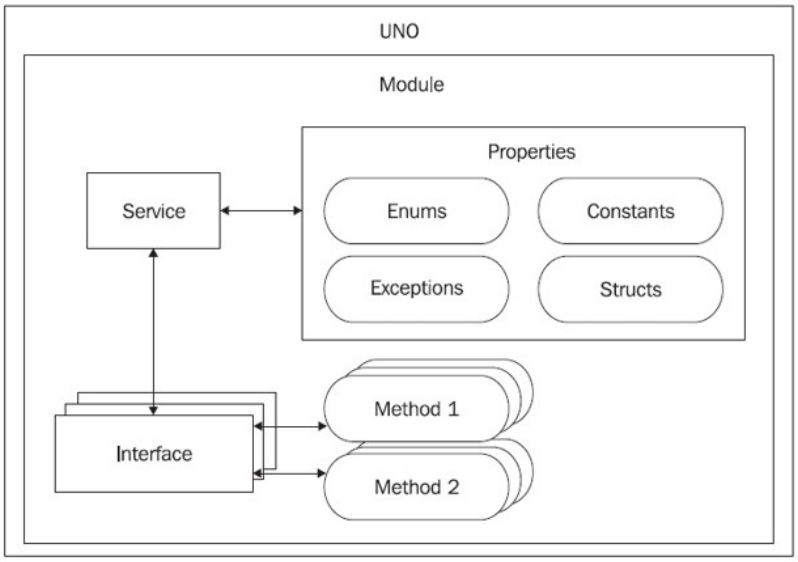
- Константы.
- Перечни (Списки).
- Исключения.
- Структуры.



В основном используются 3-и типа свойств: константы, исключения и структуры. Перечни - набор числовых значений, сгруппированных в зависимости от их использования.

Модуль

Сервисы UNO сгруппированы иерархически в модули:



Модули могут быть вложены в другие модули. Все модули вложены в один основной модуль - *com.sun.star*.

Использование LibreOffice Basic

Управление документами

Все макросы LibreOffice Basic направлены на облегчение работы с документами. Но прежде, чем выполнить какое-либо действие с документом, его необходимо получить как объект.

Чаще всего действие выполняется с текущим документом LibreOffice, т. е. с тем документом, окно которого активно в данный момент. Существуют два простых способа получить текущий документ. У каждого способа есть свои достоинства и недостатки.

Первый способ - обратиться к глобальной переменной *StarDesktop*, ссылающейся на объект рабочего стола, являющийся корневым объектом компонентов LibreOffice Basic, и, используя ее метод *getCurrentComponent*, получить текущий документ:

oDoc = StarDesktop.getCurrentComponent

Активация окна среды разработки делает его текущим документом, что вызывает трудности при отладке кода, т. к. в качестве текущего документа возвращается среда разработки Basic.

Второй способ — использовать глобальную переменную *ThisComponent*, ссылающуюся на текущий компонент:

oDoc = ThisComponent

Ее достоинство в том, что активация окон среды разработки или справочной системы не изменяет ссылку на текущий компонент, а недостаток заключается в том, что, переменная ссылается на этот компонент, а не на текущий. В итоге, если макрос сохранен в контейнере библиотек какого-либо документа, переменная *ThisComponent* будет ссылаться именно на него, несмотря на то, что в данный момент он может быть и неактивен.

Иногда требуется получить все открытые документы LibreOffice. Для этого можно использовать метод *createEnumeration* свойства *Components* объекта *StarDesktop*. Этот метод удобен, когда необходимо перебрать некоторое множество объектов. Следующая процедура создает массив, содержащий все открытые документы, и массив, содержащий описания этих документов (тип). Для определения типа документа используется функция *getDocumentType*, код которой приведен в листинге 3.

Листинг 3. Получение всех открытых документов

```
Sub Main
	Dim oDocs As Object
' Массив объектов
	Dim oDoc() As Object
' Массив с типами документов
	Dim oDocDiscrp() As String
	Dim i As Integer
	Dim sMes As String
' Создаем нумератор открытых компонентов
	oDocs = StarDesktop.Components.createEnumeration
' Пока нумератор содержит элементы
	Do While oDocs.hasMoreElements
' Изменяем размер массива с защитой содержимого
	ReDim Preserve oDoc(i)
	ReDim Preserve oDocDiscrp(i)
' Получаем очередной документ методом нумератора nextElement, который возвращает очередной элемент и исключает его из нумератора
	oDoc(i) = oDocs.nextElement
' Получаем его описание
	oDocDiscrp(i) = getDocumentType(oDoc(i))
	sMes = sMes & oDocDiscrp(i) & Chr(10)
' Увеличиваем счетчик
	i = i + 1
	Loop
' Выводим список открытых документов
	MsgBox "Открыты следующие документы: " & Chr(10) & sMes
End Sub
```

Почти все компоненты, на которые может ссылаться глобальная переменная *ThisComponent*, поддерживают интерфейс *com.sun.star.lang.XServiceInfo*, предоставляющий метод *supportsService*. Этот метод требует в качестве параметра имя сервиса и возвращает *True*, если этот сервис поддерживается данным объектом. Для любого документа LibreOffice существуют уникальные сервисы, поддерживаемые только этим компонентом. Отсюда вытекает и способ определения типа документа. В листинге 4 представлен код функции, позволяющей определить тип документа.

Листинг 4. Определение типа документа

```
Function getDocumentType(ByVal oDoc As Object) As String
```

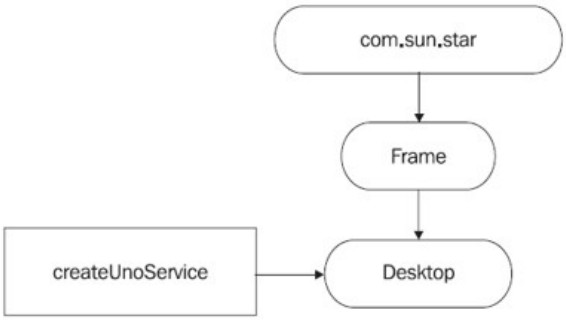
```
    On Error GoTo NotSupSupportsService
    If oDoc.SupportsService("com.sun.star.sheet.SpreadsheetDocument") Then
        getDocumentType = "Calc"
    ElseIf oDoc.SupportsService("com.sun.star.text.TextDocument") Then
        getDocumentType = "Writer"
    ElseIf oDoc.SupportsService("com.sun.star.sdb.DatabaseDocument") Then
        getDocumentType = "Base"
    ElseIf  oDoc.SupportsService("com.sun.star.presentation.PresentationDocument") Then
        getDocumentType = "Impress"
    ElseIf oDoc.SupportsService("com.sun.star.drawing.DrawingDocument") Then
        getDocumentType = "Draw"
    ElseIf oDoc.SupportsService("com.sun.star.formula.FormulaProperties") Then
        getDocumentType = "Math"
    ElseIf oDoc.SupportsService("com.sun.star.script.BasicIDE") Then
        getDocumentType = "BasicIDE"
    ElseIf oDoc.SupportsService("com.sun.star.text.WebDocument") Then
        getDocumentType = "Справка OpenOffice.org"
    Else
        getDocumentType = "Неизвестный документ"
    End If
    Exit Function
    NotSupSupportsService:
    getDocumentType = "Неизвестный документ"
End Function
```

Автоматическое открытие и закрытие документа

Прежде чем использовать сервис, необходимо знать:

- Каждый сервис храниться в модуле.
- Все модули храняться внутри центрального модуля - *com.sun.star*.
- Сервис создается при использовании функции *createUnoService*.

Сервис *Desktop* расположен в модуле *Frame*:



С помощью сервиса *com.sun.star.frame.Desktop* документы могут быть созданы, открыты и импортированы.

Сервис *com.sun.star.frame.Desktop* открывается автоматически, когда LibreOffice запускается. Чтобы сделать это, LibreOffice создает объект, к которому может быть получен доступ посредством глобального имени *StarDesktop*.

Самый важный интерфейс *StarDesktop* - *com.sun.star.frame.XComponentLoader*. Он в основном содержит метод *loadComponentFromURL*, который собственно и является ответственным за создание, импортирование, и открытие документов.

Документы открываются, импортируются и создаются с использованием метода *StarDesktop.loadComponentFromURL(URL, Frame, SearchFlags, FileProperties)*.

Первый параметр *loadComponentFromURL* определяет URL связанного файла.

В качестве второго параметра, *loadComponentFromURL* ожидает имя для фреймового объекта окна, которое LibreOffice создает внутри для его управления. Здесь обычно определяется предопределенное имя *_blank*, и оно гарантирует, что LibreOffice создаст новое окно. Альтернативно, может также быть определено *_hidden*, и оно гарантирует, что соответствующий документ будет загружен, но останется невидимым.

В листинге 5 представлен программный код открытия документа.

Листинг 5. Пример кода открывающего документ, если файл существует, а если нет, то создается новый документ

```
Sub Main
	Dim oDesk as Object
	Dim oDoc as Object
	Dim oFile as String
	Dim oUrl as String
	Dim oUrlTemp as String
' Создаем сервис
	oDesk = createUnoService ("com.sun.star.frame.Desktop")
' В переменную oFile помещаем путь к файлу, который нужно открыть
	oFile="/home/ADM72.LOCAL/yakuninamv/Рабочий стол/Макросы/Примеры кода макросов.odt"
' Преобразуем локальное имя файла в URL
	oUrl = convertToUrl (oFile)
' Проверяем, существует ли файл. Если это не так, используем пустой документ
	If fileExists (oFile) Then
		oUrlTemp = oUrl
	Else
' Создаем пустой документ LO Calc
		oUrlTemp = "private:factory/scalc"
	End If
	oDoc = oDesk.loadComponentFromURL (oUrlTemp, "_blank", 0, Array())
' Сохраняем файл
	oDoc.storeAsUrl (oUrl, Array())
' Закрываем файл, если это необходимо
	' oDoc.close(true)
End Sub
```

Сохранение документа

Унифицированный указатель ресурса (URL) - местоположение, где документ сохранен. URL файла обычно содержит полный путь к файлу. *URL* - общий способ записи местоположения хранения, который может быть удобно расширен для включения широкого диапазона типов местоположения хранения независимый от изготовителя и компьютера.

LibreOffice - платформо-независимое приложение, использует для имен файлов URL нотацию.

Для преобразования локального имени файла в URL, LibreOffice предоставляет функцию *ConvertToUrl*. Для преобразования URL в локальное имя файла, LibreOffice предоставляет функцию *ConvertFromUrl*.

Функция *loadComponentFromURL* возвращает объект документ. Он поддерживает сервис *com.sun.star.document.OfficeDocument*, который предоставляет два центральных интерфейса:

- Интерфейс *com.sun.star.frame.XStorable*, который является ответственным за сохранение документов.
- Интерфейс *com.sun.star.view.XPrintable*, который содержит методы для печати документов.

В свою очередь интерфейс *com.sun.star.frame.XStorable* предоставляет следующие методы:

- *store()* - эквивалентен команде меню **Сохранить**. Сохраняет документ в файле, из которого он прочитан.
- *storeAsURL(sURL As String, lArguments As []com.sun.star.beans.PropertyValue)* - эквивалентен команде меню **Сохранить как...**. Сохраняет документ по указанному пути. Принимает массив аргументов типа *com.sun.star.beans.PropertyValue*.
- *storeToURL(sURL As String, lArguments As []com.sun.star.beans.PropertyValue)* - этот метод напоминает предыдущий (*storeAsURL*), но, в отличие от него, не изменяет значение свойства *Location* документа, таким образом создавая копию документа в другом месте. Удобен при создании шаблонов.

Интерфейс *com.sun.star.frame.XStorable* предоставляет дополнительные методы, которые могут быть полезны при сохранении документа:

- *hasLocation()* - определяет, был ли документу уже назначен URL.
- *isReadonly()* - определяет, имеет ли документ защиту только для чтения.
- *isModified()* - определяет, был ли документ изменен, после того как было выполнено его последнее сохранение.

В листинге 6 представлен пример кода сохранения документа, если документ был фактически изменен.

Листинг 6. Пример кода сохранения документа, если он фактически изменен

```
Sub Main
        Doc = ThisComponent
' Проверяем, если документ был изменен или открыт с опцией Только для чтения
        If (Doc.isModified) Then
        If (Doc.hasLocation And (Not Doc.isReadOnly)) Then
' То сохраняем изменения
                Doc.store()
        Else
' Иначе сохраняем документ под существующим URL
                Doc.storeAsURL(URL, Dummy())
        End If
        End If
        Doc.close(true)
End Sub
```

Сохранение документа по новому местоположению

Для сохранения документа в указанное место используются два метода объекта *storeAsURL()* или *storeToURL()*. Различие между методами в том, что *storeAsURL()* устанавливает текущее местоположение (URL), а *storeToURL()* не делает этого. Последовательность действий в таблице 2 помогает объяснить разницу.

Таблица 2. Различие между storeToURL и storeAsURL

Шаг	Действие	Комментарий
1.	Создание документа	Не возможно использование метода *store()*, потому что документ не имеет местоположения
2.	Использование *storeToURL*	Документ сохраняется, но не возможно использование метода *store()*, потому что он не имеет местоположения
3.	Использование *storeAsURL*	Можно использовать метод *store()*, потому что теперь документ имеет местоположение
4.	Использование *storeToURL*	Документ сохраняется, но местоположение - то же

Шаг	Действие	Комментарий
		самое что установлено на шаге 3

В листинге 7 придставлен пример кода сохранения активного документа по новому адресу.

Листинг 7. Пример кода сохранения активного документа по новому местоположению

```
Sub Main
	Doc = ThisComponent
' Создаем массив от 0 до 0 типа PropertyValue
	Dim Ar(0) As new com.sun.star.beans.PropertyValue
' Задаем свойства массиву
' Свойство Overwrite перезаписывает любой существующий файл при сохранении
	Ar(0).Name="Overwrite"
' Не перезаписывать существующий документ
	Ar(0).Value =False
' Сохраняем документ по новому адресу
	Doc.storeToURL(ConvertToURL("home/ADM72.LOCAL/yakuninamv/Рабочий стол/Макросы/Строка1.ods"), Ar())
End Sub
```

Часто используемые параметры метода *storeAsURL()*:

- *CharacterSet (String)* - определяет набор символов документа.
- *FilterName (String)* - определяет специальный фильтр для функции *loadComponentFromURL* (фильтр, используемый для импорта или сохранения документа).
- *FilterOptions (String)* - определяет дополнительные параметры для фильтров.
- *Overwrite (Boolean)* - позволяет уже существующий файл перезаписать без вопроса.
- *Password (String)* - передает пароль для защищенного файла.
- *Unpacked (Boolean)* - сохраняет документ (без сжатия) в подкаталогах.

Массив значений свойств можно посмотреть в таблице 1.1, представленной в приложении 1.

Экспорт документа в указанный формат

Для экспорта документа в другой формат должен быть определен специальный фильтр экспорта и установлены все необходимые свойства.

Текущие фильтры экспорта и импорта можно посмотреть «Эндрю Питоньяк. OpenOffice.org Объяснение Макросов. - Hentzenwerke Publishing, 2004.» перевод Дмитрий Чернов (с) 2007-2008 Таблицы 92-100 или http://api.openoffice.org/docs/common/ref/com/sun/star/module-ix.html.

В листинге 8 придставлен пример кода экспорта документа LO Calc в формат Microsoft Excel.

Листинг 8. Пример кода экспорта документа в формат Microsoft Excel

```
Sub Main
	Doc = ThisComponent
' Создаем массив от 0 до 0 типа PropertyValue
	Dim Ar(0) As new com.sun.star.beans.PropertyValue
' Задаем свойства массиву
' Определяем специальный фильтр
	Ar(0).Name="FilterName"
	Ar(0).Value ="MS Excel 97"
' Сохраняем документ по новому адресу в формате MS Excel
```

```
    Doc.storeToURL(ConvertToURL("home/ADM72.LOCAL/yakuninamv/Рабочий
стол/Макросы/Строка1.xls"), Ar())
End Sub
```

В листинге 9 придставлен пример кода экспорта документа LO Calc в формат pdf.

Листинг 9. Пример кода экспорта документа в формат pdf

```
Sub Main
    Doc = ThisComponent
' Создаем массив от 0 до 0 типа PropertyValue
    Dim Ar(0) As new com.sun.star.beans.PropertyValue
' Задаем свойства массиву
' Определяем специальный фильтр
    Ar(0).Name="FilterName"
    Ar(0).Value ="calc_pdf_Export"
' Сохраняем документ по новому адресу в формате pdf
    Doc.storeToURL(ConvertToURL("home/ADM72.LOCAL/yakuninamv/Рабочий
стол/Макросы/Строка1.pdf"), Ar())
End Sub
```

Печать документов

Метод *Print* интерфейса *com.sun.star.view.Xprintable* предназначен для печати документов.

Метод *Print* ожидает набор данных *PropertyValue* как параметр, который отражает параметры настройки диалога печати LibreOffice:

- *CopyCount (Integer)* - определяет число копий, которые будут напечатаны;
- *FileName (String)* - посылает вывод в файл, а не на принтер;
- *Collate (Boolean)* - обратный порядок печати;
- *Sort (Boolean)* - осуществляет сортировку страниц при печати нескольких копий (CopyCount > 1);
- *Pages (String)* - содержит список страниц, которые будут напечатаны (синтаксис как определено в диалоге печати).

В листинге 10 представлен код печати одной страницы документа.

Листинг 10. Пример кода печати 1-ой страницы из текущего документа

```
Sub Main
' Создаем массив от 0 до 0 типа PropertyValue
    Dim Props(0) As New com.sun.star.beans.PropertyValue
' Задаем свойства массиву
' Определяем специальный фильтр
    Props(0).Name = "Pages"
    Props(0).Value = "1"
' Выводим на печать
    ThisComponent.print(Props())
End Sub
```

Печать документов Writer

Различные типы документов поддерживают дополнительные параметры для печати. Текстовые документы поддерживают интерфейс *com.sun.star.text.XPagePrintable* (см. Таблицу 3). Интерфейс *XPagePrintable* реализует дополнительный метод печати документа, который предоставляет большие возможности управления результатом. Основное преимущество состоит в том, что можно напечатать несколько страниц из документа на одном листе.

Таблица 3. Методы, определяемые интерфейсом com.sun.star.text.XPagePrintable

Методы объекта	Описание
getPagePrintSettings()	Возвращает массив свойств (см. Таблицу 4)
setPagePrintSettings(properties)	Изменяет установки (см. Таблицу 4)
printPages(properties)	Печать с использованием свойств (см. свойства метода *Print*)

Метод объекта *printPages()* принимает те же самые свойства, что и метод *Print()* (см. Таблицу 4).

Таблица 4. Свойства, используемые интерфейсом com.sun.star.text.XPagePrintable

Свойства	Описание
PageRows	Число рядов страниц на каждой печатаемой странице
PageColumns	Число столбцов страниц на каждой печатаемой странице
LeftMargin	Левое поле
RightMargin	Правое поле
TopMargin	Верхнее поле
BottomMargin	Нижнее поле
HoriMargin	Поле между рядами страниц
VertMargin	Поле между столбцами страниц
IsLandscape	True или False; печать в альбомном формате

В листинге 11 представлен код печати двух копий 1-й страницы документа.

Листинг 11. Пример кода печати двух копий 1-й страницы документа

```
Sub Main
' Создаем массив от 0 до 1 типа PropertyValue
    Dim Props(0 To 1) As New com.sun.star.beans.PropertyValue
' Задаем свойства массиву
' Определяем параметры специального фильтра
    Props(0).Name = "Pages"
    Props(0).Value = "1"
    Props(1).Name = "CopyCount"
    Props(1).Value = 2
' Выводим на печать
    ThisComponent.print(Props())
End Sub
```

Печать документов Calc

Для выполнения функции печати документа Calc нужно изменить свойства документа и свойства стилей страницы, а затем использовать стандартный метод *Print()*.

Одной из проблем печати документа Calc является масштабирование его при печати, например, чтобы он уместился на указанном числе страниц. Для масштабирования листа нужно установить свойство *ScaleToPages*. Чтобы просто смасштабировать страницу в процентах, нужно использовать свойство *PageScale* (Листинг 12).

Листинг 12. Пример кода печати электронной таблицыв масштабе 50%

```
Sub Main
    Dim s$
' Имя стиля
' Текущий стиль страницы
    Dim oStyle
' В документе Calc, текущий контроллер знает, какой лист является активным
```

```
        s = ThisComponent.CurrentController.getActiveSheet().PageStyle
        oStyle = ThisComponent.StyleFamilies.getByName("PageStyles").getByName(s)
' oStyle.PageScale = 100 Значение по умолчанию - 100 (100%)
' oStyle.ScaleToPages = 0 Значение по умолчанию - 0, не масштабировать
' Масштаб документа - 50%
        oStyle.PageScale = 50
' Печать документа
        ThisComponent.Print(Array())
End Sub
```

В таблице 5 представлены методы, которые опредляются интерфейсом *com.sun.star.sheet.XprintAreas*.

Таблица 5. Методы, определяемые интерфейсом com.sun.star.sheet.XPrintAreas

Метод объекта	Описание
getPrintAreas()	Возвращает массив типа *com.sun.star.table.CellRangeAddress*
setPrintAreas(ranges)	Задает область печати для листа с массивом типа *CellRangeAddress*. Печатает все, если ничего не задано
getPrintTitleColumns()	Возвращает *True* если столбцы заголовков повторяются на всех печатаемых страницах справа
setPrintTitleColumns(boolean)	*True* задает повторение заголовков столбцов на всех печатаемых страницах справа
getTitleColumns()	Массив типа *com.sun.star.table.CellRangeAddress*
setTitleColumns(ranges)	Устанавливает столбцы для использования как заголовки. Строки игнорируются; только столбцы имеют значение
getPrintTitleRows()	Возвращает *True* если строки заголовков повторяются на всех печатаемых страницах
setPrintTitleRows(boolean)	*True* задает повторение строк заголовков на всех печатаемых страницах сверху
getTitleRows()	Возвращает массив типа *com.sun.star.table.CellRangeAddress*
setTitleRows(ranges)	Задает строки, используемые в качестве заголовков. Столбцы игнорируются; только строки имеют значение

Методы в таблице 5 основаны на электронных таблицах, а не на документах Calc.

В листинге 13 представлен пример кода установки и печати нескольких диапазонов в документе Calc.

Листинг 13. Пример кода установки и печати нескольких диапазонов в документе Calc

```
Sub Main
        Dim oRanges(1) As New com.sun.star.table.CellRangeAddress
' Устанавливаем лист печати (Лист1)
        oRanges(0).Sheet = 0
' Устанавливаем первую ячейку первого диапазона (A1)
        oRanges(0).StartColumn = 0
        oRanges(0).StartRow = 0
' Устанавливаем последнюю ячейку первого диапазона (D5)
        oRanges(0).EndColumn = 3
        oRanges(0).EndRow = 4
' Устанавливаем лист печати (Лист1)
        oRanges(1).Sheet = 0
' Устанавливаем первую ячейку второго диапазона (A9)
```

```
        oRanges(1).StartColumn = 0
        oRanges(1).StartRow = 8
' Устанавливаем последнюю ячейку второго диапазона (D11)
        oRanges(1).EndColumn = 3
        oRanges(1).EndRow = 10
' Выбираем указанные диапазоны
        ThisComponent.CurrentController.getActiveSheet().setPrintAreas(oRanges())
' Печатаем
        ThisComponent.Print(Array())
End Sub
```

Работа с документами LibreOffice Calc

LibreOffice поддерживает три основных типа таблиц: текстовые таблицы в документах Writer, таблицы базы данных и электронные таблицы в документах Calc.

Концептуально, все типы документов имеют два компонента: данные, которые они содержат и контроллер, который определяет, как данные отображаются. В LibreOffice, набор данных, содержащихся в документе называют моделью. Каждая модель имеет контроллер, который ответственен за визуальное представление данных. Контролер знает местоположение видимого текстового курсора и текущей страницы, и знает то, что в настоящее время выделено.

Каждый документ Calc поддерживает сервис *com.sun.star.sheet.SpreadsheetDocument*.

Доступ к листам

Основная цель документа электронной таблицы состоит в том, чтобы действовать как контейнер для отдельных листов через интерфейс *XSpreadhseetDocument*. Интерфейс *XSpreadsheetDocument* определяет единственный метод *getSheets()*, который возвращает объект *Spreadsheets,* используемый для манипулирования отдельными листами. Например:

```
' Метод, определяемый интерфейсом XSpreadsheetDocument
ThisComponent.getSheets()

' Свойство документа электронная таблица
ThisComponent.Sheets
```

Сервис *Spreadsheet* позволяет возвращать отдельные листы по индексу, перебором и по имени. Методы реализуемые сервисом *com.sun.star.sheet.Spreadsheets* представлены в таблице 6. Сервис *Spreadsheet* также позволяет создавать, перемещать и удалять листы.

Таблица 6. Методы, реализуемые сервисом com.sun.star.sheet.Spreadsheets

Метод	Описание
copyByName(srcName, destName, index)	Копирует лист, имеющий имя *srcNam* по указанному индексу и называет его *destName*
createEnumeration()	Создает объект, который выполняет перебор листов электронной таблицы
getByIndex(index)	Получает лист электронной таблицы на основе индекса листа
getByName(name)	Получает лист электронной таблицы на основе имени листа
getCount()	Возвращает число листов в виде длинного целого числа
hasByName(name)	Возвращает *True* если имя листа электронной таблицы существует
hasElements()	Возвращает *True* если документ содержит по крайней мере один лист электронной таблицы
insertNewByName(name, index)	Создает новый лист электронной таблицы и вставляет его в указанное место с указанным именем
moveByName(name, index)	Перемещает заданный именем лист электронной таблицы по указанному индексу
removeByName(name)	Удаляет заданный именем лист электронной

Метод	Описание
	таблицы

!!! Нумерация листов начинается с 0

В листинге 14 представлен пример кода получения доступа к листу по его номеру.

Листинг 14. Пример кода получения доступа к листу по его номеру

```
Sub Main
        Dim Doc As Object
        Dim Sheet As Object
        Doc = StarDesktop.CurrentComponent
        Sheet = Doc.Sheets(0)
….
End Sub
```

В листинге 15 представлен пример кода получения доступа к листу по его имени.

Листинг 15. Пример кода получения доступа к листу по его имени

```
Sub Main
        Dim Doc As Object
        Dim Sheet As Object
        Doc = StarDesktop.CurrentComponent
' Получаем лист на основе его имени
        Sheet = Doc.Sheets.getByName("Лист1")
….
End Sub
```

Объект *Sheet*, который получен методом *getByName*, поддерживает сервис *com.sun.star.sheet.Spreadsheet*. В дополнение к предоставлению нескольких интерфейсов для редактирования содержимого, этот сервис предоставляет следующие свойства:

- *IsVisible (Boolean)* – видимость электронной таблицы;
- *PageStyle (String)* – имя стиля страницы для электронной таблицы.

Создание, удаление и переименование листов

Список *Sheets* для документа *Spreadsheet* также используется для создания, удаления и переименования отдельных листов.

В листинге 16 представлен пример добавления листа в книгу, удаление листа из книги и переименование листа.

Листинг 16. Пример кода добавления листи в книгу, удаление листа из книги и переименование листа

```
Sub Main
        Dim oDoc as Object
        Dim oSheet as Object
        oDoc = ThisComponent
' Создаем сервис
        oSheet = oDoc.createInstance ("com.sun.star.sheet.Spreadsheet")
' Добавление нового листа в книгу и назначение имени листу
        oDoc.Sheets.insertByName ("Лист2", oSheet)
' Удаление листа
        oDoc.Sheets.removeByName("Лист1")
' Переименование листа
        oSheet.Name="пример"
End Sub
```

Ячейки листа

Электронная таблица состоит из двумерного списка, содержащего ячейки. Ячейка может быть идентифицирована ее именем, которое использует букву столбца и номер строки или ее позицией. Например, верхняя левая ячейка - «A1» имеет позицию (0, 0), а ячейка «B3» имеет позицию (1, 2).

Получить ячейку или диапазон ячеек листа можно, воспользовавшись методами интерфейса *com.sun.star.table.XCellRange*. Например, для получения одной ячейки годится метод *getCellByPosition*:

oCell = oSheet.getCellByPosition(0,0)

Переменная *oCell* содержит ссылку на ячейку «A1».

Для получения диапазона по адресу или по имени служат методы *getCellRangeByPosition* и *getCellRangeByName* интерфейса *com.sun.star.table.XcellRange*.

Получить выделенную ячейку можно с помощью метода *getCurrentSelection* интерфейса *com.sun.star.frame.XModel*:

oCell = ThisComponent.getCurrentSelection

Адрес ячейки

В LibreOffice, адрес ячейки определяется листом, который содержит ячейку, и строкой и столбцом, в которых расположена ячейка. LibreOffice инкапсулирует адрес ячейки в структуре *CellAddress*. Свойства структуры *CellAddress* представлены в таблице 7. Структура *CellAddress* доступна непосредственно из ячейки, и она также используется как аргумент многочисленными методами объекта.

Таблица 7. Свойства структуры com.sun.star.table.CellAddress

Свойство	Описание
Sheet	Короткое целое число, индекс листа, который содержит ячейку
Column	Длинное целое число, индекс столбца, где расположена ячейка
Row	Длинное целое число, индекс строки, где расположена ячейка

Данные ячеки

Ячейки таблицы могут содержать текст, числа, формулы или быть пустыми. Тип ячейки не определяется содержанием, которое сохранено в ячейке, а скорее свойствами объекта, который использовался для ее ввода. Числа могут быть вставлены и получены с использованием свойства *Value*, текст - свойства *String*, а формулы - свойства *Formula*.

В листинге 17 представлен пример вставки числа, текста и формулы.

Листинг 17. Пример кода вставки числа, текста и формулы

```
Sub Main
        Dim Doc As Object
        Dim oSheet As Object
        Dim Cell As Object
        Doc = ThisComponent
' Получаем второй рабочий лист
        oSheet = Doc.Sheets(1)
' В ячейку A1 вставляем значение 100
        Cell =oSheet.getCellByPosition(0, 0)
        Cell.Value = 100
' В ячейку A2 вставляем текст «Тест»
        Cell =oSheet.getCellByPosition(0, 1)
        Cell.String = "Тест"
' В ячейку A3 вставляем результат сложения числа в ячейки A1 и 15
```

```
    Cell =oSheet.getCellByPosition(0, 2)
    Cell.Formula = "=A1+15"
End Sub
```

Для проверки содержимого ячейки используется свойство *Type.* В листинге 18 представлен пример определения содержимого ячейки.

Листинг 18. Пример кода определения содержимого ячейки

```
Sub Main
    Dim Doc As Object
    Dim Sheet As Object
    Dim Cell As Object
    Doc = ThisComponent
' Получаем второй рабочий лист
    Sheet = Doc.Sheets(1)
' В ячейку B2 вставляем результат сложения числа в ячейки A1 и 15
    Cell = Sheet.getCellByPosition(1,1)
    Cell.Formula = "=A1+15"
' Проверяем что находится в ячеки и результат выводим в диалоговое окно
    Select Case Cell.Type
    Case com.sun.star.table.CellContentType.EMPTY
    MsgBox "Содержимое: Пусто"
    Case com.sun.star.table.CellContentType.VALUE
    MsgBox "Содержимое: Число"
    Case com.sun.star.table.CellContentType.TEXT
    MsgBox "Содержимое: Текст"
    Case com.sun.star.table.CellContentType.FORMULA
    MsgBox "Содержимое: Формула"
    End Select
End Sub
```

Результат работы макроса представлен на рисунке 6.

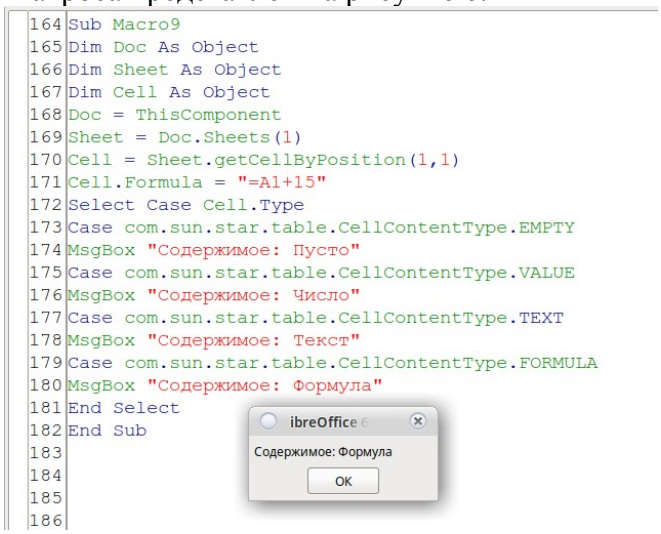


Рис. 6. В окне среды разработки результат работы макроса, представленного в листинге 18

Свойство *Cell.Type* возвращает значение из списка *com.sun.star.table.CellContentType*, которое идентифицирует тип содержимого ячейки. Возможные значения:

- EMPTY - пусто, нет значения;
- VALUE - число;
- TEXT - строка;
- FORMULA - формула.

Вставка, удаление, копирование и перемещение ячеек

LibreOffice Calc предоставляет интерфейс, который позволяет вставлять, удалять, копировать или объединять ячейки. Интерфейс с*om.sun.star.sheet.XRangeMovement*, доступен через объект электронная таблица и обеспечивает четыре метода для изменения содержимого ячейки:

- *insertCell* - вставка ячейки в лист;
- *removeRange* - удаление ячейки из листа;
- *moveRange* - перемещение ячейки;
- *copyRange* - копирование ячейки.

В листинге 19 представлен пример вставки диапазона ячеек в лист.

Листинг 19. Пример кода вставки области ячеек в лист

```
Sub Main
        Dim Doc As Object
        Dim Sheet As Object
        Dim CellRangeAddress As New com.sun.star.table.CellRangeAddress
        Doc = ThisComponent
' Получаем второй рабочий лист
        Sheet = Doc.Sheets(1)
' Указываем адрес листа (Лист2)
        CellRangeAddress.Sheet = 1
' Указываем начало диапазона для вставки (ячейка B2)
        CellRangeAddress.StartColumn = 1
        CellRangeAddress.StartRow = 1
' Указываем конец диапазона для вставки (ячейка C3)
        CellRangeAddress.EndColumn = 2
        CellRangeAddress.EndRow = 2
' Вставляем область ячеек, размером в две строки и два столбца
        Sheet.insertCells(CellRangeAddress, com.sun.star.sheet.CellInsertMode.DOWN)
End Sub
```

Для определения области вставляемых ячеек используется структура *com.sun.star.table.CellRangeAddress*, свойства которой представлены в таблице 8.

Таблица 8. Свойства структуры com.sun.star.table.CellRangeAddress

Свойство	Описание
Sheet	Короткое целое число, индекс листа, который содержит ячейку
StartColumn	Длинное целое число, индекс столбца, где расположен левый край
StartRow	Длинное целое число, индекс строки, где расположен верхний край
EndColumn	Длинное целое число, индекс столбца правого края диапазона
EndRow	Длинное целое число, индекс строки нижнего края диапазона

Заполненную структуру *CellRangeAddress* нужно передать как первый параметр методу *insertCells*. Второй параметр *insertCells* содержит значение из списка *com.sun.star.sheet.CellInsertMode* и определяет то, что должно быть сделано со значениями, которые расположены перед точкой вставки. Список *CellInsertMode* предоставляет следующие значения:

- NONE - текущие значения не остаются в их существующем положении;
- DOWN - ячейки в и под позицией вставки перемещаются вниз;
- RIGHT - ячейки в и справа от позиции вставки перемещаются вправо;
- ROWS - строки после позиции вставки перемещаются вниз;
- COLUMNS - столбцы после позиции вставки перемещаются вправо.

В листинге 20 представлен пример удаления области ячеек, определенных в структуре *CellRangeAddress*.

Листинг 20. Пример кода удаления области ячеек из листа

```
Sub Main
	Dim Doc As Object
	Dim Sheet As Object
	Dim CellRangeAddress As New com.sun.star.table.CellRangeAddress
	Doc = ThisComponent
	Sheet = Doc.Sheets(1)
' Указываем адрес листа (Лист2)
	CellRangeAddress.Sheet = 1
' Указываем начало диапазона для вставки (B2)
	CellRangeAddress.StartColumn = 1
	CellRangeAddress.StartRow = 1
' Указываем конец диапазона для вставки (C3)
	CellRangeAddress.EndColumn = 2
	CellRangeAddress.EndRow = 2
' Удаляем область ячеек (B2:C3) и перемещаем расположенные под ней ячейки вверх на две строки
	Sheet.removeRange(CellRangeAddress, com.sun.star.sheet.CellDeleteMode.UP)
End Sub
```

Тип удаления определяется одним из следующих значений из списка *com.sun.star.sheet.CellDeleteMode*:

- NONE - текущие значения не остаются в их существующем положении;
- UP - ячейки, расположенные под удаляемой областью перемещаются вверх;
- LEFT - ячейки, расположенные справа от удаляемой области, перемещаются влево;
- ROWS - строки, расположенные ниже удаляемой области, перемещаются вверх;
- COLUMNS - столбцы, расположенные справа от удаляемой области, перемещаются влево.

В листинге 21 представлен пример перемещения области ячеек B2:C3 в позицию, которая начинается с ячейки A14.

Листинг 21. Пример кода перемещения области ячеек

```
Sub Main
	Dim Doc As Object
	Dim Sheet As Object
	Dim CellRangeAddress As New com.sun.star.table.CellRangeAddress
	Dim CellAddress As New com.sun.star.table.CellAddress
	Doc = ThisComponent
	Sheet = Doc.Sheets(1)
	CellRangeAddress.Sheet = 1
	CellRangeAddress.StartColumn = 1
	CellRangeAddress.StartRow = 1
	CellRangeAddress.EndColumn = 2
	CellRangeAddress.EndRow = 2
' Указываем лист и начальную ячейку вставки диапазона
	CellAddress.Sheet = 1
	CellAddress.Column = 0
	CellAddress.Row = 13
' Вставляем диапазон ячеек
	Sheet.moveRange(CellAddress, CellRangeAddress)
End Sub
```

Метод *copyRange* функционирует таким же образом, как метод *moveRange*, за исключением того, что *copyRange* вставляет копию области ячеек вместо ее перемещения.

Использование диапазона ячеек

В электронной таблице, ячейки могут быть сгруппированы в прямоугольных областях *SheetCellRange*. Группировка ячеек позволяет управлять несколькими ячейками одновременно. Сервис *SheetCellRange* поддерживает многие из тех интерфейсов и свойств, что и *SheetCell*.

Каждая ячейка имеет адрес ячейки (см. Таблицу 8), который идентифицирует ее положение в документе электронной таблицы. Каждый диапазон ячеек имеет аналогичную структуру, которая идентифицирует его положение в электронной таблице.

Одиночный диапазон ячеек листа существует на одном листе и содержит одну прямоугольную область. Множественные диапазоны инкапсулированны сервисом *SheetCellRanges*, который поддерживает большинство тех же самых свойств и сервисов, что и *SheetCellRange*.

Сервис *SheetCellRanges* обеспечивает доступ к каждому диапазону с помощью интерфейса *XElementAccess* и интерфейса *XIndexAccess*. Методы в таблице 9 также обеспечивают доступ к содержимому диапазонов ячеек.

Таблица 9. Методы, определяемые интерфейсом com.sun.star.table.XSheetCellRanges

Метод	Описание
getCells()	Возвращает коллекцию содержащихся ячеек как *XEnumerationAccess*
getRangeAddressesAsString()	Возвращает строку с адресами всех содержащихся диапазонов. Выходной формат подобен «Лист1.B2:D6;Лист3.C4:D5»
getRangeAddresses()	Возвращает массив сервисов типа *CellRangeAddress* (см. Таблицу 8)

В листинге 22 представлен пример использования диапазона ячеек.

Листинг 22. Пример кода использования дипазона ячеек

```
Sub Main
        Dim oDoc as Object
        Dim oCell1 as Object
        Dim OCell2 as Object
        Dim OCell3 as Object
        oDoc= ThisComponent
        oSheet = oDoc.sheets (0)
' В ячейки A1, A2, A3 вставляем числа 36, 57, 42 соответственно
        oCell1 = oSheet.getCellRangeByName ("A1")
        oCell2 = oSheet.getCellRangeByName ("A2")
        oCell3 = oSheet.getCellRangeByName ("A3")
        oCell1.Value = 36
        oCell2.Value = 57
        oCell3.Value = 42
' Выбираем диапазон A1:A3
        oRange1 = oSheet.getCellRangeByName ("A1:A3")
' Подключаем сервис функций
        oFunction = createUnoService("com.sun.star.sheet.FunctionAccess")
' С помощью функции SUM суммируем числа, находящиеся в диапазоне A1:A3
        oCell4.Value = oFunction.callFunction("SUM", Array(oRange1.getDataArray ()))
End Sub
```

Результат работы макроса представулен на рисунке 7.

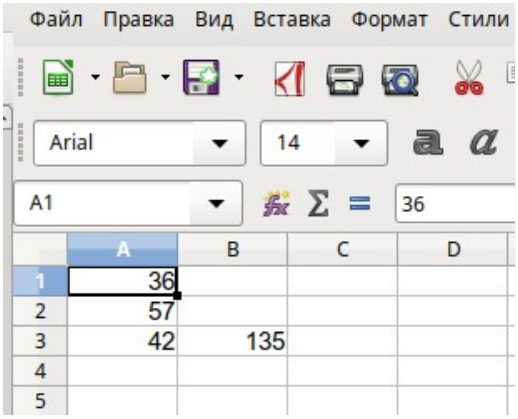


Рис. 7. В окне среды разработки результат работы макроса,
представленного в листинге 22

Объединение ячеек

С помощью метода *merge (Boolean)* - *merge (True)* можно объеденить диапазон ячеек, а с помощью метода *merge (False)* удалить объединение диапазона.

В листинге 23 представлен пример объединения диапазона ячеек B6:D8 и вывод в диалог значения, которое содержиться в ячейке C7.

Листинг 23. Пример кода объединения диапазона ячеек

```
Sub Main
        Dim oCell
        Dim oRange
        Dim oSheet
' Указываем лист, на котором необходимо объеденить ячейки (в данном случае второй лист)
        oSheet = ThisComponent.Sheets(1)
' Указываем диапазон
        oRange = oSheet.getCellRangeByName("B2:D7")
' Объединяем
        oRange.merge(True)
' Выводим содержимое объединенной ячейки в диалоговое окно
        oCell = oSheet.getCellByPosition(2, 8)                ' Ячейка C7
        Print oCell.getValue()
End Sub
```

Форматирование электронных таблиц

Документ электронная таблица предоставляет свойства и методы для форматирования ячеек и страниц.

Основное форматирование связано с размерами столбца и строки. Заполняя ячейки данными, можно обнаружить, что столбец имеет недостаточную ширину. Это можно исправить воспользовавшись одним из свойств, поддерживаемых отдельными столбцами. Все манипуляции, которые можно производить со столбцами, можно производить и со строками, за исключением *getName()*. Различия свойств, поддерживаемых отдельными столбцами и строками представлены в таблице 10.

Таблица 10. Свойства отдельных строк и столбцов

Тип	Свойство	Описание
Столбец	*Width*	Ширина столбца (в 0,01 мм) как длинное целое число
Строка	*Height*	Высота строки (в 0,01 мм) как длинное целое число
Столбец	*OptimalWidth*	Если *True*, столбец всегда поддерживает оптимальную

Тип	Свойство	Описание
		ширину
Строка	*OptimalHeight*	Если *True*, строка всегда поддерживает оптимальную высоту
Оба	*IsVisible*	Если *True*, строка или столбец - видимы
Оба	*IsStartOfNewPage*	Если *True*, горизонтальный (вертикальный) разрыв страницы добавляется к столбцу (строке)

В листинге 24 представлен пример оптимизации ширины столбцов для всего документа.

Листинг 24. Пример кода оптимизации ширины столбцов для всего документа

```
Sub Main
	Dim oDoc as Object
	Dim oSheet as Object
	Dim oCell as Object
	oDoc = ThisComponent
	oSheet = oDoc.sheets (0)
' Заполняем ячейки некоторыми данными
	oCell = oSheet.getCellByPosition (0, 0)
	oCell.String = date
	oCell = oSheet.getCellByPosition (0, 1)
	oCell.String = "Investigator Name"
	oCell = oSheet.getCellByPosition (1, 1)
	oCell.String = "Pygoscelis P.Ellsworthy"
	oCell = oSheet.getCellByPosition (2, 1)
	oCell.String = "Client Email Address"
	oCell = oSheet.getCellByPosition (3, 1)
	oCell.String = "ellworthyp@penguinpi.com"
' Оптимизация ширины столбцов для всего листа
	oSheet.getColumns.OptimalWidth = True
End Sub
```

Свойства ячейки

Для форматирования ячейки используются различные параметры, например, такие как тип шрифта и размера для текста, и многие другие. Каждая ячейка поддерживает сервисы *com.sun.star.style.CharacterProperties* (свойства символа можно посмотреть в таблице 2.1 приложения 2) и *com.sun.star.style.ParagraphProperties* (свойства абзаца можно посмотреть в таблице 2.2 приложения 2).

Есть еще специфические для ячейки свойства, такие как установка границ. Для определения границы ячейки используется структура *BorderLine* (см. Таблицу 11), а структура *TableBorder* определяет границы для всей таблицы (см. Таблицу 12).

Таблица 11. Свойства структуры com.sun.star.table.BorderLine

Свойство	Описание
Color	Цвет линии как длинное целое число
InnerLineWidth	Ширина внутренней части двойной линии (в 0,01 мм) как короткое целое число - ноль для одиночной линии
OuterLineWidth	Ширина одиночной линии или ширина внешней части двойной линии (в 0,01 мм) как короткое целое число
LineDistance	Расстояние между внутренней и внешней частями двойной линии (в 0,01 мм) как короткое целое число

Таблица 12. Свойства структуры com.sun.star.table.TableBorder

Свойство	Описание
TopLine	Стиль линии по верхнему краю (см. Таблицу 11)
IsTopLineValid	Если *True*, *TopLine* используется при задании значений
BottomLine	Стиль линии по нижнему краю (см. Таблицу 11)
IsBottomLineValid	Если *True*, *BottomLine* используется при задании значений
LeftLine	Стиль линии по левому краю (см. Таблицу 11)
IsLeftLineValid	Если *True*, *LeftLine* используется при задании значений
RightLine	Стиль линии по правому краю (см. Таблицу 11)
IsRightLineValid	Если *True*, *RightLine* используется при задании значений
HorizontalLine	Стиль линии для горизонтальных линий между ячейками (см. Таблицу 11)
IsHorizontalLineValid	Если *True*, *HorizontalLine* используется при задании значений
VerticalLine	Стиль линии для вертикальных линий между ячейками (см. Таблицу 11)
IsVerticalLineValid	Если *True*, *VerticalLine* используется при задании значений
Distance	Расстояние между линиями и другим содержимым как короткое целое число
IsDistanceValid	Если *True*, *Distance* используется при задании значений

Специальное форматирование ячеек обрабатывается сервисом *com.sun.star.table.CellProperties*. Главные свойства этого сервиса описаны в таблице 3.1 приложения 3.

В листинге 25 представлен пример форматирования ячейки, макрос записывает число 1000 в ячейку B2, изменяет цвет фона на красный и создает легкую серую тень для ячейки и помещает ее справа и внизу.

Листинг 25. Пример кода форматирования ячейки

```
Sub Main
    Dim Doc As Object
    Dim Sheet As Object
    Dim Cell As Object
' Создаем новую структуру тени
    Dim ShadowFormat As New com.sun.star.table.ShadowFormat
    Doc = ThisComponent
    Sheet = Doc.Sheets(0)
' В ячейку B2 записываем число 1000
    Cell = Sheet.getCellByPosition(1, 1)
    Cell.Value = 1000
' Меняем цвет ячейки на красный
    Cell.CellBackColor = RGB(255, 0, 0)
' Задаем положение тени справа
    ShadowFormat.Location = com.sun.star.table.ShadowLocation.BOTTOM_RIGHT
' Задаем размер тени равный 1 мм
    ShadowFormat.ShadowWidth = 100
' Раскрашиваем тень в светлосерый
    ShadowFormat.Color = RGB(160, 160, 160)
    Cell.ShadowFormat = ShadowFormat
```

```
End Sub
```

В листинге 26 представлен пример форматирования содержимого ячейки.

Листинг 26. Пример кода форматирования содержимого ячейки

```
Sub Main
    Dim Doc As Object
    Dim Sheet As Object
    Dim Cell As Object
    Dim ShadowFormat As New com.sun.star.table.ShadowFormat
    Doc = ThisComponent
    Sheet = Doc.Sheets(0)
    Cell = Sheet.getCellByPosition(2, 2)
    Cell.String = "ПРИВЕТ"
' Горизонтальное выравнивание текста в ячейки по левому краю
    Cell.HoriJustify = com.sun.star.table.CellHoriJustify.LEFT
' Вертикальное выравнивание текста в ячейки по левому краю
    Cell = Sheet.getCellByPosition(3, 3)
    Cell.String = "ПРИВЕТ"
    Cell.VertJustify = com.sun.star.table.CellVertJustify.TOP
' Каждый символ сверху вниз горизонтально
    Cell = Sheet.getCellByPosition(4, 4)
    Cell.String = "ПРИВЕТ"
    Cell.Orientation = com.sun.star.table.CellOrientation.STACKED
End Sub
```

Свойства диапазона ячеек

Диапазоны ячеек листа и ячейки листа поддерживают свойства предоставляемые сервисом *com.sun.star.sheet.SheetCell* (см. Таблицу 13). Свойство *Position* для диапазона ячеек листа предоставляет положение верхней левой ячейки - что эквивалентно получению свойства *Position* для верхней левой ячейки в диапазоне. Свойство *Size* для ячейки возвращает размер одной ячейки, а свойство *Size* для диапазона ячеек листа предоставляет размер для всех ячеек, содержащихся в диапазоне.

Таблица 13. Свойства, поддерживаемые сервисом com.sun.star.sheet.SheetCell

Свойство	Описание
Position	Положение ячейки в листе (в 0,01 мм) как структура *com.sun.star.awt.Point*. Это абсолютное положение во всем листе, не положении в видимой области. • X - x-координата как длинное целое число. • Y - y-координата как длинное целое число.
Size	Размер ячейки (в 0,01 мм) как структура *com.sun.star.awt.Size*: • Width - ширина как длинное целое число. • Height - высота как длинное целое число.
FormulaLocal	Необязательная строка, содержащая формулу, использующую локализованное имя функции
FormulaResultType	Тип результата формулы со значениями из группы констант *com.sun.star.sheet.FormulaResult*: • VALUE = 1 - формула возвращает число с плавающей запятой двойной точности. • STRING = 2 - формула возвращает строковое значение. • ERROR = 4 - формула имеет какую-то ошибку.

Свойство	Описание
ConditionalFormat	Условные параметры настройки форматирования для этой ячейки. Когда условный формат изменяется, он должен быть повторно вставлен в набор свойств
ConditionalFormatLocal	Необязательное свойство, дублирующее *ConditionalFormat* за исключением того, что формула использует локализованные имена
Validation	Параметры настройки подтверждения правильности данных для этой ячейки как *com.sun.star.beans.XPropertySet*
ValidationLocal	Необязательное свойство, дублирующее *Validation* за исключением того, что формула использует локализованные имена

Условное форматирование

Условное форматирование позволяет стилю ячейки изменяться на основании содержимого ячейки. Ячейки листа и диапазоны ячеек листа оба поддерживают свойство *ConditionalFormat*, которое в свою очередь поддерживает интерфейс *XsheetConditionalEntries*. Кроме того, можно получить доступ к условным записям форматирования при использовании элемента доступа, доступа по индексу или методов *addNew(properties())*, *clear()* и *removeByIndex(index)*.

Можно применить несколько записей условного форматирования для одной и той же ячейки. Применяется первая, которая подходит. Каждую запись форматирования представляет массив структур *PropertyValue*. Пример применения условного форматирования представлен в листинге 27. Макрос задает диапазону ячеек использовать стиль «Accent», если ячейка содержит отрицательное число.

Листинг 27. Пример кода условног форматирования диапазона ячеек

```
Sub Main
        Dim oRange as Object
' Используемый диапазон ячеек
        Dim oConFormat as Object
' Объект условного форматирования
        Dim oCondition(2) As New com.sun.star.beans.PropertyValue
        oRange = ThisComponent.Sheets(2).getCellRangeByName("A1:D6")
' Получаем объект проверки
        oConFormat = oRange.ConditionalFormat
' Создаем условие проверки
' Значение должно быть меньше чем указанное значение
        oCondition(0).Name = "Operator"
        oCondition(0).Value = com.sun.star.sheet.ConditionOperator.LESS
' Указываем значение (=0)
        oCondition(1).Name = "Formula1"
        oCondition(1).Value = 0
' Присваевам стиль "Accent"
        oCondition(2).Name = "StyleName"
        oCondition(2).Value = "Accent"
        oConFormat.addNew(oCondition())
        oRange.ConditionalFormat = oConFormat
End Sub
```

Работа с документами LibreOffice Writer

Структура текстовых документов

Текстовый документ может содержать четыре типа информации:

- фактический текст;
- стили для форматирования символов, абзацев, и страниц;
- нетекстовые элементы, такие как таблицы, диаграммы и изображения;
- общие параметры настройки для текстового документа.

Абзацы и части абзацев

Основная часть текстового документа состоит из последовательности абзацев. Они не именуются, не вносятся в указатель и поэтому, возможности прямого обращения к отдельным абзацам нет. Абзацы могут последовательно перебираться с помощью объекта *Enumeration*, который позволяет редактировать абзацы.

Когда к объекту невозможно обратиться по имени или индексу используются интерфейсы *XEnumeration* и *XenumerationAccess*.

Интерфейсы *com.sun.star.container.XEnumeration* и *com.sun.star.container.XenumerationAccess*

Интефейс *XenumerationAccess* содержит только метод *createEnumeration*, который возвращает вспомогательный объект, предоставляемый интерфейсом *Xenumeration*. Интерфейс *Xenumeration* содержит методы *hasMoreElements* и *nextElement*.

В листинге 28 представлен код программы, позволяющий перебрать все абзацы имеющиеся в тексте.

Листинг 28. Пример кода перебора всех абзацев документа

```
Sub Main
	Dim ParagraphEnumeration As Object
	Dim Paragraph As Object
' Создаем вспомагательный элемент
	ParagraphEnumeration = ThisComponent.Text.createEnumeration
' В цикле перебираем все абзацы
' Цикл заканчивается как только метод hasMoreElements возвращает значение False
	While ParagraphEnumeration.hasMoreElements()
		Paragraph = ParagraphElements.nextElement()
	Wend
End Sub
```

При работе с объетом *Enumeration* нужно обратить внимание на то, что он возвращает не только абзацы, но и таблицы.

!!! В LibreOffice Writer таблица это специальный тип абзаца

Перед обращением к возвращенному объекту необходимо проверять поддерживает он сервис *com.sun.star.text.Paragraph* для абзаца или сервис *com.sun.star.text.TextTable* для таблицы.

Абзацы

Сервис *com.sun.star.text.Paragraph* предоставляет доступ к содержимому абзаца.

В листинге 29 представлен код программы, позволяющий перебрать все абзацы в документа и изменить цвет каждого.

Листинг 29. Пример кода изменяющего цвет каждого абзаца

```
Sub Main
	Dim Doc as object
	Dim oEnum as object
```

```
	Dim TextElement as object
	Doc=ThisComponent
' Создаем вспомагательный элемент
	oEnum=Doc.Text.createEnumeration()
' Цикл по всем текстовым элементам
	While oEnum.hasMoreElements()
		TextElement=oEnum.nextElement
' Проверяем является ли текущий блок текста параграфом
	if TextElement.supportsService("com.sun.star.text.Paragraph") Then
' Если да, то меняем цвет
		TextElement.CharColor=RGB(0, 255, 0)
	End if
	Wend
End Sub
```

Части абзаца

Абзац состоит из индивидуальных подобъектов, каждый из которых содержит информацию о форматирование.

Пользователь может получить доступ к частям абзаца с помощью собственных объетов *Enumeration*.

В листинге 30 приведен пример кода двойного цикла перебора в текстовом документе. Внешний цикл обращается к текстовым абзацам. Внутренний цикл обрабатывает части этих абзацев. Код примера изменяет определенное содержание в каждом абзаце.

Листинг 30. Пример кода замены одних слов на другие

```
Sub Main
	Dim oDoc as Object
	Dim oEnum1 as object
	Dim oEnum2 as object
	Dim TextElement as object
	Dim oPors as object
	Doc=ThisComponent
' Создаем вспомагательный элемент
	oEnum1=Doc.Text.createEnumeration()
' Цикл по всем текстовым элементам
	While oEnum1.hasMoreElements()
		TextElement=oEnum1.nextElement
' Проверяем является ли текущий блок текста параграфом
	if TextElement.supportsService("com.sun.star.text.Paragraph") Then
		oEnum2=TextElement.createEnumeration
' Цикл по всем элементам текущего параграфа (текстовым позициям) TextElement, пока есть еще элементы
	While oEnum2.hasMoreElements()
		oPors=oEnum2.nextElement
' Меняем одну последовательность символов на другую
	oPors.String=Replace(oPors.String, "на","НА")
	oPors.String=Replace(oPors.String, "ии","И")
	Wend
	End if
	Wend
' Выводим сообщение в диалоговое окно
	msgBox "Конец"
End Sub
```

Форматирование

Свойства символа

Свойства связанные с символами находятся в сервисе *CharacterProperties* (см. Таблицу 2.1 приложения 2).

В листинге 31 приведен пример кода изменения свойств символа с помощью изменения свойства *FontRelief*.

Листинг 31. Пример кода изменения свойств символа с помощью изменениря свойства *FontRelief*

```
Sub Main
	Dim oEnum
	Dim oPar
' Создаем вспомагательный элемент
	oEnum = ThisComponent.Text.createEnumeration()
' Цикл по всем текстовым элементам
	Do While oEnum.hasMoreElements()
		oPar = oEnum.nextElement()
' Проверяем является ли текущий блок текста параграфом
	If oPar.supportsService("com.sun.star.text.Paragraph") Then
' Если да, то присваиваем значение рельефа =2(символы выгравированны, т. е. утоплены)
		oPar.CharRelief = 2
	End If
	Loop
End Sub
```

Свойства абзаца

Свойства абзаца доступны через сервис com.sun.star.style.ParagraphProperties (см. таблицу 2.2 приложения 2). Все объекты, которые поддерживают сервис *com.sun.star.text.Paragraph* также обеспечивают поддержку для свойств абзаца в *com.sun.star.style.ParagraphProperties*.

В листинге 32 приведен пример кода изменения цвета фона абзацев.

Листинг 32. Пример кода изменения цвета фона абзаца

```
Sub Main
	Dim oEnum
	Dim oPar
	oEnum = ThisComponent.Text.createEnumeration()
	Do While oEnum.hasMoreElements()
		oPar = oEnum.nextElement()
	If oPar.supportsService("com.sun.star.text.Paragraph") Then
' Меняем цвет фона абзаца на зеленый
	oPar.ParaBackColor =RGB(0, 255, 0)
	End If
	Loop
End Sub
```

Редактирование текстовых документов

Курсоры

Одним из методов получения доступа и манипулирование документом является использование курсора. *TextCursor* - *TextRange*, который может перемещаться в пределах объекта *Text*. Т.е. текстовый курсор не только может определять одну точку в тексте, но и

охватывать текстовый диапазон. В LibreOffice доступны два основных типа текстовых курсоров: курсоры, которые являются отображаемыми курсорами (см. Таблицу 14), и курсоры, которые не отображаются (см. Таблицу 16).

Таблица 14. Отображаемые курсоры не связаны с текстовыми диапазонами или XtextCursor

Курсор	Описание
com.sun.star.view.XViewCursor	Простой курсор с основными методами перемещения, которые работают и в тексте и в таблицах
com.sun.star.text.XTextViewCursor	Полученный из *XTextCursor*, он описывает курсор в представлении текстового документа. Он поддерживает только очень простые перемещения
com.sun.star.view.XLineCursor	Определяет методы связанные со строкой; этот интерфейс не получается из текстового диапазона
com.sun.star.text.XPageCursor	Определяет методы связанные со страницей, этот интерфейс не получается из текстового диапазона
com.sun.star.view.XScreenCursor	Определяет методы для перемещения вверх и вниз на один экран за раз

Отображаемые курсоры

Отображаемый курсор имеет дело с видимым курсором. Можно использовать один отображаемый курсор за раз. Отображаемые курсоры поддерживают команды, которые непосредственно связаны с представлением. Чтобы переместить курсор на одну строку или один экран за раз, требуется отображаемый курсора. Отображаемый курсор знает, как отображается текст (см. Таблицу 15).

Таблица 15. Методы объекта, связанные с отображаемыми курсорами

Определяется	Метод	Описание
XViewCursor	*goDown(n, Boolean)*	Перемещает курсор вниз на n строк
XViewCursor	*goUp(n, Boolean)*	Перемещает курсор вверх на n строк
XViewCursor	*goLeft(n, Boolean)*	Перемещает курсор влево на n символов
XViewCursor	*goRight(n, Boolean)*	Перемещает курсор вправо на n символов
XTextViewCursor	*isVisible()*	Возвращает *True* если курсор видим
XTextViewCursor	*getPosition()*	Возвращает структуру *com.sun.star.awt.Point*, определяющую координаты курсора относительно верхней левой позиции первой страницы документа
XLineCursor	*isAtStartOfLine()*	Возвращает *True*, если курсор — в начале строки
XLineCursor	*isAtEndOfLine()*	Возвращает *True*, если курсор — в конце строки
XLineCursor	*gotoEndOfLine(Boolean)*	Перемещает курсор в конец текущей строки
XLineCursor	*gotoStartOfLine(Boolean)*	Перемещает курсор в начало текущей строки
XPageCursor	*jumpToFirstPage()*	Перемещает курсор на первую страницу
XPageCursor	*jumpToLastPage()*	Перемещает курсор на последнюю страницу

Определяется	Метод	Описание
XPageCursor	*jumpToPage(n)*	Перемещает курсор на указанную страницу
XPageCursor	*getPage()*	Возвращает текущую страницу в виде короткого целого числа
XPageCursor	*jumpToNextPage()*	Перемещает курсор на следующую страницу
XPageCursor	*jumpToPreviousPage()*	Перемещает курсор на предыдущую страницу
XPageCursor	*jumpToEndOfPage()*	Перемещает курсор в конец текущей страницы
XPageCursor	*jumpToStartOfPage()*	Перемещает курсор в начало текущей страницы
XScreenCursor	*screenDown()*	Прокручивает представление вперед на одну видимую страницу
XScreenCursor	*screenUp()*	Прокручивает представление назад на одну видимую страницу

В листинге 33 приведен пример кода вставки символа со значением Unicode 257 в текущее положение курсора.

Листинг 33. Пример кода вставки символа со значением Unicode 257 в текущее положение курсора

```
Sub Main
    Dim oViewCursor As Object
' Получаем отображаемый курсор от текущего контроллера
    oViewCursor = ThisComponent.CurrentController.getViewCursor()
' Вставляем специальный символ в текущее положение курсора
    oViewCursor.getText.insertString(oViewCursor.getStart(), CHR$(257), False)
End Sub
```

Текстовые (неотображаемые) курсоры

Отображаемый курсор знает, как данные отображаются, но не знает непосредственно о самих данных. Текстовые курсоры (неотображаемые курсоры) знают много о данных, но очень немного о том, как они отображаются. Например, отображаемые курсоры не знают о словах или абзацах, а текстовые курсоры не знают о строках, экранах или страницах (см. Таблицу 16).

Таблица 16. Текстовый курсор объединяет все реализации интерфейса XTextCursor

Курсор	Описание
com.sun.star.text.XTextCursor	Основной текстовый курсор, который определяет простые методы перемещения
com.sun.star.text.XWordCursor	Предоставляет методы перемещения и проверки связанные со словами
com.sun.star.text.XSentenceCursor	Предоставляет методы перемещения и проверки связанные с предложениями
com.sun.star.text.XParagraphCursor	Предоставляет методы перемещения и проверки связанные с абзацами
com.sun.star.text.XTextViewCursor	Полученный из *XTextCursor*, он описывает курсор в представлении текстового документа

Таблица 17. Методы объекта, связанные с текстовыми курсорами

Определяется	Метод	Описание
XTextCursor	*collapseToStart()*	Устанавливает конечную позицию равной начальной позиции
XTextCursor	*collapseToEnd()*	Устанавливает начальную позицию равной конечной позиции
XTextCursor	*isCollapsed()*	Возвращает *True* если начальная и конечная позиции равны
XTextCursor	*goLeft(n, Boolean)*	Перемещает курсор влево на n символов
XTextCursor	*goRight(n, Boolean)*	Перемещает курсор вправо на n символов
XTextCursor	*gotoStart(Boolean)*	Перемещает курсор в начало текста
XTextCursor	*gotoEnd(Boolean)*	Перемещает курсор в конец текста
XTextCursor	*gotoRange(XTextRange, Boolean)*	Перемещает или расширяет курсор в *TextRange*
XWordCursor	*isStartOfWord()*	Возвращает *True* если в начале слова
XWordCursor	*isEndOfWord()*	Возвращает *True* если в конце слова
XWordCursor	*gotoNextWord(Boolean)*	Перемещает в начало следующего слова
XWordCursor	*gotoPreviousWord(Boolean)*	Перемещает в конец предыдущего слова
XWordCursor	*gotoEndOfWord(Boolean)*	Перемещает в конец текущего слова
XWordCursor	*gotoStartOfWord(Boolean)*	Перемещает в начало текущего слова
XSentenceCursor	*isStartOfSentence()*	Возвращает *True* если в начале предложения
XSentenceCursor	*isEndOfSentence()*	Возвращает *True* если в конце предложения
XSentenceCursor	*gotoNextSentence(Boolean)*	Перемещает в начало следующего предложения
XSentenceCursor	*gotoPreviousSentence(Boolean)*	Перемещает в конец предыдущего предложения
XSentenceCursor	*gotoEndOfSentence(Boolean)*	Перемещает в конец текущего предложения
XSentenceCursor	*gotoStartOfSentence(Boolean)*	Перемещает в начало текущего предложения
XParagraphCursor	*isStartOfParagraph()*	*True* если в начале абзаца
XParagraphCursor	*isEndOfParagraph()*	*True* если в конце абзац
XParagraphCursor	*gotoNextParagraph(Boolean)*	Перемещает в начало следующего абзаца
XParagraphCursor	*gotoPreviousParagraph(Boolean)*	Перемещает в конец предыдущего абзаца
XParagraphCursor	*gotoEndOfParagraph(Boolean)*	Перемещает в конец текущего абзац

Определяется	Метод	Описание
XParagraphCursor	*gotoStartOfParagraph(Boolean)*	Перемещает в начало текущего абзаца

Курсор слова, курсор предложения и курсор абзаца, все определяют чрезвычайно идентичные методы объекта. Интерфейс *XTextViewCursor* упоминается в таблице 15, поэтому он пропущен в таблице 17.

В листинге 34 приведен пример кода вставки одного абзаца в начало документа.

Листинг 34. Пример кода вставки одного абзаца в начало документам

```
Sub Main
	Dim T as Object
	Dim oEn as object
	Dim abz
	T=ThisComponent.Text
' Создаем вспомагательный элемент
	oEn=T.createEnumeration()
' Выбираем первый элемент (абзац)
	abz=oEn.nextElement()
' Вставляем его в начало документа
	T.insertString(T.getStart(),abz.String,False)
End Sub
```

В листинге 35 приведен пример кода перестановки первого и второго абзацев.

Листинг 35. Пример кода перестановки первого и второго абзацев

```
Sub Main
	Dim T as Object
	Dim oEn as object
	dim document as object
	Dim abz, abz1
	document=ThisComponent
	T=document.Text
' Создаем вспомагательный элемент
	oEn=T.createEnumeration()
' В переменные abz и abz1 помещаем первый и второй абзац соответственно
	abz=oEn.nextElement(0)
	abz1=oEn.nextElement(1)
' Вставляем второй абзац в начало документа и вставляем перевод курсора на новую строку
	T.insertString(T.getStart(),abz1.String & CHR$(13), False)
' Удаляем теперь уже третий абзац
	abz1.setString("")
End Sub
```

Использование курсора для перемещения по тексту

В листинге 36 приведен пример кода перебора всех абзацев с использованием курсора.

Листинг 36. Пример кода использования курсора для перебора всех абзацев

```
Sub Main
	Dim oCursor
' Создаем текстовый курсор
	oCursor = ThisComponent.Text.createTextCursor()
' Начинаем пребор абзацев с начала документа
	Do
' Выделяем весь абзац. Перемещение
oCursor.gotoEndOfParagraph(True)
```

```
' Что-то делаем с абзацем
…..
' Дальше оператор цикла перемещает курсор к следующему абзацу и отменяет выделение текста в курсоре
        Loop While oCursor.gotoNextParagraph(False)
End Sub
```

В листинге 37 приведен пример кода раскраски фона всех абзацев в красный цвет.

Листинг 37. Пример кода

```
Sub Main
        Dim oCursor
        oCursor = ThisComponent.Text.createTextCursor()
        Do
' Красим фон абзацев в красный
                oCursor.ParaBackColor=RGB(255,0,0)
        Loop While oCursor.gotoNextParagraph(False)
        msgBox "Конец"
End Sub
```

В листинге 38 приведен пример кода форматирования первого слова каждого предложения жирным шрифтом.

Листинг 38. Пример кода форматирования первого слова каждого предложения жирным шрифтом

```
Sub Main
        Dim Cursor As Object
        Dim Proceed As Boolean
        Cursor = ThisComponent.Text.createTextCursor
        Do
' Выделяем первое слово, перемещая курсор в конец слова
                Cursor.gotoEndOfWord(True)
' Форматируем жирным шрифтом
                Cursor.CharWeight = com.sun.star.awt.FontWeight.BOLD
' Перемещаем курсор в начало следующего предложения
                Proceed = Cursor.gotoNextSentence(False)
        Loop While Proceed
End Sub
```

Выделенный текст

Выделенный текст, по сути, это текстовый диапазон.

При работе с документами часто возникает необходимость поместить часть текста в строковую переменную для последующей обработки. В большинстве случаев такой текст является выделенным. Для получения выделенного текста можно воспользоваться методом *getCurrentSelection()* интерфейса *XtextSectionsSupplier*.

Интерфейс *com.sun.star.text.XtextSectionsSupplier*, поддерживаемый тектовыми документами, предоставляет доступ к содержащимся в документе текстовым разделам.

В листинге 39 приведен пример кода изменения цвета выделенного текста.

Листинг 39. Пример кода изменения цвета выделенного текста

```
Sub Main
        Dim oDoc as Object
        Dim oCurs as Object
        Dim oText as Object
        Dim oTextCurs as Object
```

```
	Dim oTextCurs1 as Object
	oDoc=ThisComponent
' Получаем выдимый курсор
	oCurs=oDoc.CurrentController.getViewCursor
' Получаем объект Текст
	oText=oDoc.getText
' Создаем текстовый курсор в позиции видимого
' Начало выделения
	oTextCurs=oText.createTextCursorByRange(oCurs.getStart())
' Конец выделения
	oTextCurs1=oText.createTextCursorByRange(oCurs.getEnd())
' Выделяем от начала видимого курсора до конца
	oTextCurs.gotoRange(oTextCurs1,True)
' Меняем цвет
	oTextCurs.CharColor=RGB(0,255,0)
End Sub
```

В листинге 40 приведен пример кода выделения, изменения цвета и начертания каждой первой и пятой буквы первого слова каждого абзаца.

Листинг 40. Пример кода

```
Sub Main
	Dim oDoc as Object
	Dim oCurs as Object
	Dim oPos as Boolean
	oDoc=ThisComponent
	oCurs=oDoc.Text.createTextCursor()
' Выделение первого символа в первом слове первого абзаца
	oPos=oCurs.goRight(1,False)
	oPos=oCurs.goLeft(1,True)
' Изменение наклона и цвета шрифта для выделенного символа
	oCurs.CharPosture=com.sun.star.awt.FontSlant.ITALIC
	oCurs.CharColor=RGB(255,0,0)
' Выделение пятого символа в первом слове абзаца
	oPos=oCurs.goRight(5,False)
	oPos=oCurs.goLeft(1,True)
' Изменение наклона и цвета шрифта для выделенного символа
	oCurs.CharPosture=com.sun.star.awt.FontSlant.ITALIC
	oCurs.CharColor=RGB(255,0,0)
' Перебор всех абзацев
	Do
' Переход к концу параграфа
		oCurs.gotoEndOfParagraph(True)
' Выделение первого символа в первом слове абзаца
		oPos=oCurs.goRight(2,False)
		oPos=oCurs.goLeft(1,True)
' Изменение наклона и цвета шрифта для выделенного символа
		oCurs.CharPosture=com.sun.star.awt.FontSlant.ITALIC
		oCurs.CharColor=RGB(255,0,0)
' Выделение пятого символа в первом слове абзаца
		oPos=oCurs.goRight(6,False)
		oPos=oCurs.goLeft(1,True)
' Изменение наклона и цвета шрифта для выделенного символа
```

```
        oCurs.CharPosture=com.sun.star.awt.FontSlant.ITALIC
        oCurs.CharColor=RGB(255,0,0)
' Перейти на следующее предложение без выделения
        oPos=oCurs.gotoNextSentence(False)
' Перейти на начало текущего предложения без выделения
        oCurs.gotoStartOfParagraph(False)
' Условие выполняется пока значение текущей переменной oPos истино
    Loop While oPos
End Sub
```

Поиск и замена

Процесс поиска управляется описателем поиска, который в состоянии осуществлять поиск только в объекте, который его создал. Т.е невозможно использовать один и тот же описатель для поиска в нескольких документах. Описатель поиска определяет текст поиска и то как текст ищется (см. Таблицу 18).

Таблица 18. Свойства сервиса com.sun.star.util.SearchDescriptor

Свойство	Описание
SearchBackwards	Если *True*, поиск выполняется по документу в обратном порядке
SearchCaseSensitive	Если *True*, регистр букв учитывается при поиске
SearchWords	Если *True*, ищутся только полные слова
SearchRegularExpression	Если *True*, строка поиска рассматривается как регулярное выражение
SearchStyles	Если *True*, текст ищется на основе примененных именах стилей - не на содержании текста
SearchSimilarity	Если *True*, выполняется «поиск подобных»
SearchSimilarityRelax	Если True, свойства *SearchSimilarityRelax*, *SearchSimilarityRemove*, *SearchSimilarityAdd* и *SearchSimilarityExchange* все используются
SearchSimilarityRemove	Короткое целое число, определяющее, сколько символов может игнорироваться при поиске соответствия
SearchSimilarityAdd	Короткое целое число, определяющее, сколько символов может быть добавлено при поиске соответствия
SearchSimilarityExchange	Короткое целое число, определяющее, сколько символов может быть заменено при поиске соответствия

Описатель поиска поддерживает строковое свойство *SearchString*, которое представляет строку для поиска. Интерфейс *XsearchDescriptor* определяет методы *getSearchString()* и *setSearchString()* для получения и установки свойства. Интерфейс *XSearchable* определяет методы, используемые для поиска и создания описателя поиска (см. Таблицу 19).

Таблица 19. Методы, определяемые интерфейсом com.sun.star.util.XSearchable

Метод	Описание
createSearchDescriptor()	Создает новый описатель поиска
findAll(XSearchDescriptor)	Возвращает *XindexAccess*, содержащий все найденные элементы
findFirst(XSearchDescriptor)	Начав с начала доступного для поиска объекта, возвращает текстовый диапазон, содержащий первый найденный текст
findNext(XText Range,	Начиная от указанного текстового диапазона, возвращает

Метод	Описание
XSearchDescriptor)	текстовый диапазон, содержащий первый найденный текст

В листинге 41 приведен пример кода установки свойства символа *CharWeight* всех возникновений текста «курсор» в *com.sun.start.awt.FontWeight.BOLD.*

Листинг 41. Пример кода установки для всех появлений слова «курсор» свойства символа полужирный

```
Sub Main
' Описатель поиска
	Dim oDescriptor
' Найденный диапазон
	Dim oFound
' Создаем новый описатель поиска
	oDescriptor=ThisComponent.createSearchDescriptor()
	With oDescriptor
		.SearchString="Курсор"
' Атрибуты по умолчанию False
		.SearchWords=true
' Регистр букв при поиске не учитывается
		.SearchCaseSensitive=False
	End With
' Ищем первое совпадение
	oFound=ThisComponent.findFirst(oDescriptor)
	Do While Not IsNull(oFound)
' Выводим в диалоговое окно найденное слово для потверждения замены
		Print oFound.getString()
' Выделяем найденное слово полужирным
		oFound.CharWeight=com.sun.star.awt.FontWeight.BOLD
' Заканчиваем замену, если текст закончился
		oFound=ThisComponent.findNext(oFound.End, oDescriptor)
	Loop
End Sub
```

Все совпадения текста искать быстрее если использовать метод объекта *findAll()*. *FindAll()* применяется, когда используются все совпадения указанного текста.

В листинге 42 приведен пример кода заменяющего все совпадения (в полном смысле этого слова: не важно целое это слова или только часть) «Курсор» на «КУРСОР»

Листинг 42. Пример кода

```
Sub Main
' Описатель поиска
	Dim oDescriptor
' Найденный диапазон
	Dim oFound
' Найденный диапазон
	Dim oFoundAll
' Общая индексная переменная
	Dim n%
	oDescriptor = ThisComponent.createSearchDescriptor()
' Помещаем в описатель «Курсор»
	oDescriptor.SearchString = "Курсор"
' Ищем все совпадения
	oFoundAll = ThisComponent.findAll(oDescriptor)
' Пока есть совпадения меняем их на «КУРСОР»
```

```
    For n% = 0 to oFoundAll.getCount()-1
        oFound = oFoundAll.getByIndex(n%)
        oFound.setString("КУРСОР")
    Next
End Sub
```

Специальный объект *ReplaceDescriptor* поддерживается сервисом *com.sun.star.util.ReplaceDescriptor*. Все свойства *SearchDescriptor*, описанные в таблице 18 поддерживаются *ReplaceDescriptor*.

В листинге 43 приведен пример кода заменяющего несколько совпадений одновременно.

Листинг 43. Пример кода

```
Sub Main
    Dim I As Long
    Dim Replace As Object
' Создаем два массива, состоящии из 3-х элементов каждый
    Dim OWords(2) As String
    Dim IWords(2) As String
' В первый массив помещаем то, что собираемся искать
    OWords() = Array("курсор", "я", "макрос")
' Во второй массив помещаем то, на что будем менять
    IWords() = Array("КУРСОР", "Я", "МАКРОС")
    oDoc=ThisComponent
' Создаем новый описатель
    Replace = oDoc.createReplaceDescriptor
' Пока есть элементы в массивах
    For I = 0 To 2
        Replace.SearchString = OWords(I)
        Replace.ReplaceString = Iwords(I)
' Находим совпадения и заменяем их
        oDoc.replaceAll(Replace)
    Next I
End Sub
```

Таблицы в текстовом документе

Текстовые документы кроме текстовых абзацев могут содержать и другие объекты такие как, таблицы, рисунки, текстовые поля и справочники. Все они могут размещаться в любом месте в пределах текста. Данные объекты поддерживают общий базовый сервис *com.sun.star.text.TextContent*, основная цель которого привязать объект к окружающему его тексту. В таблице 20 представлены свойства сервиса.

Таблица 20. Свойства, определяемые сервисом com.sun.star.text.TextContent

Свойство	Описание
AnchorType	Перечислимый тип *com.sun.star.text.TextContentAnchorType*, который определяет, каким образом это текстовое содержимое присоединено к окружающему тексту. • AT_PARAGRAPH - Привязка устанавливается относительно верхней левой позиции абзаца. Объект перемещается, если перемещается абзац. • AS_CHARACTER - Объект текстового содержимого привязывается как символ. Размер объекта влияет на высоту текстовой строки, а объект может перемещаться как символ, если перемещается

Свойство	Описание
	окружающий текст. • AT_PAGE - Объект текстового содержимого привязывается к странице. Объект не перемещается, даже если текстовое содержимое вокруг него изменяется. • AT_FRAME - Объект текстового содержимого привязывается к текстовой врезке. • AT_CHARACTER - Объект текстового содержимого привязывается к символу. Объект перемещается, если перемещается символ.
AnchorTypes	Массив *TextContentAnchorType*, который содержит связанные типы текстового содержимого
TextWrap	Перечислимый тип *com.sun.star.text.WrapTextMode*, который определяет, как окружающий текст обтекает этот объект текстового содержимого. • NONE - Текст не обтекает вокруг объекта. • THROUGHT - Поток текста игнорирует объект. (Да, это - *THROUGHT*.) Об этом можно думать как *THROUGHT iT*, т.е., «текст течет через объект». • PARALLEL - Текст обтекает объект слева и справа. • DYNAMIC - Форматирование текста выбирает лучший метод обтекания. • LEFT - Текст обтекает объект слева. • RIGHT - Текст обтекает объект справа.

Объекты *TextContent* реализуют методы создания, вставки и удаления объектов:

- новый объект *TextContent* создается с использованием метода *createInstance* объекта документ;
- объект вставляется с использованием метода *insertTextContent* объекта *text*;
- объекты *TextContent* удаляются с использованием метода *removeTextContent*.

В листинге 44 приведен пример кода создания таблицы.

Листинг 44. Пример кода создания таблицы

```
Sub Main
        Dim Table As Object
        Dim Cursor As Object
        Dim oDoc As Object
        oDoc=ThisComponent
' Создаем текстовый курсор
        Cursor = oDoc.Text.createTextCursor()
' Создаем таблицу
        Table = oDoc.createInstance("com.sun.star.text.TextTable")
' Устанавливает количество строк=5 и столбцов=4
        Table.initialize(5, 4)
' Вставляем таблицу в текстовый документ
        oDoc.Text.insertTextContent(Cursor, Table, False)
End Sub
```

Редактирование таблиц

Методы, поддерживаемые таблицами в текстовом документе подобны методам, которые поддерживают электронные таблицы, т. е. таблицы, содержащиеся в документах LibreOffice Calc. В таблице 21 представлены методы объекта, поддерживаемые таблицами текстового документа.

Таблица 21. Методы объекта, поддерживаемые таблицами текстового документат

Метод	Описание
autoFormat(name)	Примените указанное имя автоформата к таблице
createCursorByCellName(name)	*XTextTableCursor*, помещенный в указанную ячейку
createSortDescriptor()	Массив *PropertyValues*, который определяет критерии сортировки
dispose()	Уничтожает текстовый объект, а также удаляет его из документа
getAnchor()	Возвращает идентифицирующий текстовый диапазон, где привязана таблица. Этот метод позволяет легко добавит текстовое содержимое перед или после текстовой таблицы
getCellByName(name)	Возвратите *XCell* на основе имени ячейки, таком виде, например «B3»
getCellByPosition(col, row)	Нумерация начинается с нуля. Имеет трудности со сложными таблицами
getCellNames()	Массив строк имен ячеек, содержащихся в таблице
getCellRangeByName(name)	*XCellRange*, на основе имен ячеек, таких как A1:B4. Терпит неудачу, если название идентифицирует ячейку, которая была разбита
getCellRangeByPosition(left, top, right, bottom)	*XCellRange*, на основе числового диапазона
getColumnDescriptions()	Массив строк, описывающих столбцы. Терпит неудачу для сложных таблиц
getColumns()	Объект *XTableColumns* для перебора столбцов по индексу. Текже поддерживаются *insertByIndex(idx, count)* и *removeByIndex(idx, count)*
getData()	Получает числовые данные как вложенную последовательность значений (массив в массиве). Терпит неудачу для сложных таблиц
getDataArray()	То же самое, что *getData()*, но может содержать строки или вещественные числа двойной точности
getName()	Получает имя таблицы в виде строки
getRowDescriptions()	Массив строк, описывающих строки. Терпит неудачу для сложных таблиц
getRows()	Объект *XTableRows* для перебора строк по индексу. Текже поддерживаются *insertByIndex(idx, count)* и *removeByIndex(idx, count)*
initialize(rows, cols)	Задает число строк и столбцов. Должен быть выполнен прежде, чем таблица вставлена
setColumnDescriptions(string())	Задает описания столбцов из массива строк
setData(Double())	Задает числовые данные в виде вложенной последовательности значений. Терпит неудачу для сложных таблиц
setDataArray(array())	То же самое, что *setData()*, но может содержать строки или

Метод	Описание
	вещественные числа двойной точности
setName(name)	Задает имя таблицы
setRowDescriptions(string())	Задает описания строк из массива строк
sort(array())	Сортирует таблицу, основываясь на описателе сортировки

Таблицы в текстовых документах поддерживают свойства, которые поддерживаются абзацами (см. таблицу 2.2 приложения 2). Объекты таблицы в текстовых документах поддерживают различные свойства, которые предствлены в таблице 22.

Таблица 22. Свойства, поддерживаемые сервисом com.sun.star.text.TextTable

Свойство	Описание
BreakType	Определяет тип разрыва, который применен в начале таблицы (см. параметр *BreakType* в Таблице 2.2 приложения 2)
LeftMargin	Определяет левый отступ таблицы в 0,01 мм как длинное целое число. Устанавливает свойство *HoriOrient* во что-либо другое чем *FULL*
RightMargin	Определяет правый отступ таблицы в 0,01 мм как длинное целое число. Устанавливает свойство *HoriOrient* во что-либо другое чем *FULL*
HoriOrient	Определяет горизонтальную ориентацию используя константы *com.sun.star.text.HoriOrientation*. Значение по умолчанию - *com.sun.star.text.HoriOrientation.FULL*. • NONE = 0 - Выравнивание не применено. • RIGHT = 1 - Объект выровнен по правой стороне. • CENTER = 2 - Объект выровнен по центру. • LEFT = 3 - Объект выровнен по левой стороне. • INSIDE = 4 - (Еще не поддерживается) • OUTSIDE = 5 - (Еще не поддерживается) • FULL = 6 - Объект использует все пространство (только длятекстовых таблиц). • LEFT_AND_WIDTH = 7 - Определены левое смещение и ширина объекта.
KeepTogether	Если *True*, предотвращает разрыв страницы или колонки между этой таблицей и следующим абзацом или текстовой таблицей
Split	Если *False*, таблица не будет разбиваться между двумя страницами или колонками
PageDescName	Если эта строка установлена, разрыв страницы происходит перед абзацем, и новая страница использует данное имя стиля страницы (см. *PageDescName* в Таблице 2.2 приложения 2)
PageNumberOffset	Если разрыв страницы происходит, определяет новый номер страницы (см. *PageNumberOffset* в Таблице 2.2 приложения 2)
RelativeWidth	Определяет ширину таблицы относительно ее окружения в виде короткого целого числа
IsWidthRelative	Если *True*, относительная ширина действительна
RepeatHeadline	Если *True*, первая строка таблицы повторяется на каждой новой странице

Свойство	Описание
ShadowFormat	Определяет тип, цвет, и размер тени (см. *ParaShadowFormat* в Таблице 2.2 приложения 2)
TopMargin	Определяет верхний отступ таблицы в 0,01 мм как длинное целое число
BottomMargin	Определяет нижний отступ таблицы в 0,01 мм как длинное целое число
BackTransparent	Если *True*, фоновый цвет прозрачный
Width	Определяет абсолютную ширину таблицы как длинное целое число - это свойство только для чтения
ChartRowAsLabel	Если *True*, первая строка рассматривается как надписи осей, если создается диаграмма
ChartColumnAsLabel	Если *True*, первый столбец рассматривается как надписи осей, если создается диаграмма
TableBorder	Определяет границы таблицы в структуре *com.sun.star.table.TableBorder*. Структура содержит много сложных свойств: • Свойства *TopLine*, *BottomLine*, *LeftLine*, *RightLine*, *HorizontalLine* и *VeriticalLine* - все структуры типа *BorderLine* как описано для свойства *LeftBorder* в Таблице 2.2 приложения 2. • Свойство *Distance* содержит расстояние между линиями и другим содержанием. • Можно включить каждое свойство границы или выключить его, устанавливая одно из следующих свойств в *True* или *False*: *IsTopLineValid*, *IsBottomLineValid*, *IsLeftLineValid*, *IsRightLineValid*, *IsHorizontalLineValid*, *IsVerticalLineValid* и *IsDistanceValid*.
TableColumnSeparators	Определяет ширину каждого столбца в зависимости от массива разделителей столбцов таблицы. Каждый разделитель - структура *com.sun.star.text.TableColumnSeparator*. • *Position* - короткое целое число, которое определяет положение разделителя ячеек. • *IsVisible* - определяет, видим ли разделитель. Ширина ячейки определяется положением разделителя между смежными ячейками. Когда две ячейки объединены, разделитель скрыт, но не удален. Значение *Position* свойство *TableColumnRelativeSum* относительно текстовой таблицы. Это свойство действительно для таблицы, только если каждая строка имеет ту же самую структуру. Если это не так, применяется разделитель от отдельного объекта строки
TableColumnRelativeSum	Определяет сумму значений ширины столбцов, используемых в *TableColumnSeparators* как короткое целое число
BackColor	Определяет фоновый цвет абзаца как длинное целое число
BackGraphicURL	Определяет URL фонового изображения абзаца
BackGraphicFilter	Определяет имя графического фильтра для фонового изображения абзаца

Свойство	Описание
BackGraphicLocation	Определяет положение фонового изображения (см. *ParaBackGraphicLocation* в Таблице 2.2 приложения 2)

Строки

Таблица состоит из списка, содержащего строки.

В листинге 45 приведен пример кода извлечения и форматирования строк.

Листинг 45. Пример кода извлечения и форматирования строк

```
Sub Main
	Dim Table As Object
	Dim Cursor As Object
	Dim oDoc As Object
	oDoc=ThisComponent
' Создаем текстовый курсор
	Cursor = oDoc.Text.createTextCursor()
' Создаем таблицу
	Table = oDoc.createInstance("com.sun.star.text.TextTable")
' Устанавливает количество строк=5 и столбцов=4
	Table.initialize(5, 4)
' Вставляем таблицу в текстовый документ
	oDoc.Text.insertTextContent(Cursor, Table, False)
' Создаем список строк для перебора
	Rows = Table.getRows
' Пока есть элементы списка
	For I = 0 To Rows.getCount() - 1
' Красим строку в цвет фуксии
		Row = Rows.getByIndex(I)
		Row.BackColor = &HFF00FF
	Next
End Sub
```

Столбцы

Таблица состоит из отдельных строк. Они в свою очередь содержат отдельные ячейки. По сути в LibreOffice Basic нет столбцов таблицы. Они порождаются неявно при упорядочении строк (один под другим) рядом друг с другом. Для упращения доступа к таблицам предоставляются методы, использующиеся для управления столбцами.

К столбцам можно получить доступ, используя аналогичные методы доступа (*getByIndex*, *getCount*, *insertByIndex*, и *removeByIndex*) к строкам, только для объекта *Column*. Данные методы могут использоваться только для таблиц, которые не содержат объединенные ячейки.

Ячейки

Каждая ячейка таблицы текстового документа имеет уникальное имя. Объект ячейка доступен через метод *getCellByName()* объекта таблица.

В листинге 46 приведен пример кода, в котором в цикле перебираются все ячейки таблицы и вводится соответствующий номер строки и столбца в ячейки.

Листинг 46. Пример кода

```
Sub Main
	Dim Table As Object
	Dim Cursor As Object
	Dim oDoc As Object
```

```
        oDoc=ThisComponent
' Создаем текстовый курсор
        Cursor = oDoc.Text.createTextCursor()
' Создаем таблицу
        Table = oDoc.createInstance("com.sun.star.text.TextTable")
' Устанавливает количество строк=5 и столбцов=4
        Table.initialize(5, 4)
' Вставляем таблицу в текстовый документ
        oDoc.Text.insertTextContent(Cursor, Table, False)
' Создаем список строк для перебора
        Rows = Table.getRows
' Создаем список столбцов для перебора
Cols = Table.getColumns
' Пока есть строки
        For RowIndex = 1 To Rows.getCount()
' Пока есть столбцы
        For ColIndex = 1 To Cols.getCount()
' Вставляем номер строки и номер столбца
                CellName = Chr(64 + ColIndex) & RowIndex
                Cell = Table.getCellByName(CellName)
                Cell.String = "строка: " & CStr(RowIndex) + ", столбец: " & CStr(ColIndex)
        Next
        Next
End Sub
```

Ячейка таблицы сопоставима со стандартным текстом. Она поддерживает интерфейс *createTextCursor* (см. таблицу 23) для создания связанного объекта *TextCursor*.

CellCursor = Cell.createTextCursor()

Поэтому автоматически доступны все варианты форматирования для отдельных символов и абзацев.

Таблица 23. Методы, определяемые интерфейсом com.sun.star.text.XtextTableCursor

Методы	**Описание**
getRangeName()	Возвращает диапазон ячеек, выделенный этим курсором в виде строки. Например, «B3:D5»
gotoCellByName(String, boolean)	Перемещает курсор в ячейку с указанным именем; возвращает логическое значение
goLeft(n, boolean)	Перемещает курсор влево на n ячеек; возвращает логическое значение
goRight(n, boolean)	Перемещает курсор вправо на n ячеек; возвращает логическое значение
goUp(n, boolean)	Перемещает курсор вверх на n ячеек; возвращает логическое значение
goDown(n, boolean)	Перемещает курсор вниз на n ячеек; возвращает логическое значение
gotoStart(boolean)	Перемещает курсор в верхнюю левую ячейку
gotoEnd(boolean)	Перемещает курсор в нижнюю правую ячейку
mergeRange()	Объединяет выделенный диапазон ячеек; в случае успеха возвращает *True*

Методы	Описание
splitRange(n, boolean)	Создает n (целое число) новых ячеек в каждой ячейке выделенной курсором. Для *Boolean*, установленного в *True* выполняет разбиение по горизонтали, *False* - по вертикали. В случае успеха возвращает *True*

В листинге 47 приведен пример кода, в котором перебираются все таблицы текстового документа и раскращиваются ячейки не содержащие числовые значения.

Листинг 47. Пример кода

```
Sub Main
	Dim TextTables As Object
	Dim Table As Object
	Dim CellNames
	Dim Cell As Object
	Dim CellCursor As Object
	Dim I As Integer
	Dim J As Integer
' Создаем список содержащий все таблицы
	TextTables = ThisComponent.getTextTables()
' Пробегаем по стокам
	For I = 0 to TextTables.count-1
		Table = TextTables(I)
		CellNames = Table.getCellNames()
' Пробегаем по столбцам
	For J = 0 to UBound(CellNames)
		Cell = Table.getCellByName(CellNames(J))
' Если ячейка не содержит число, то раскращиваем ячейку
		If IsNumeric(Cell.String)=NONE Then
			CellCursor = Cell.createTextCursor()
			CellCursor.ParaBackColor=RGB(0,255,0)
		End If
	Next
	Next
End Sub
```

В листинге 48 приведен пример кода, в котором в первую таблицу текстового документа после первой строки вставляется две строки и в позицию (0, 3) (первый столбец, четвертая строка, A4) вставляется текст.

Листинг 48. Пример кода

```
Sub Main
	Dim oAllTables as Variant
	Dim oTable1 as Variant
	Dim oCell as Variant
' Создаем список содержащий все таблицы
	oAllTables = ThisComponent.getTextTables()
' Получаем таблицу по индексу, можно по имени .getByName("Таблица1")
	oTable1 = oAllTables.getByIndex(0)
' Добавляем строку
' Вставка строки по индексу 1 - после какой строки вставить, 2 - количество строк для
вставки
	oTable1.Rows.insertByIndex(1,2)
' Получаем доступ к ячейке (столбец, срока)
	oCell = oTable1.getCellByPosition(0,3)
```

```
' Задаем текст для ячейки
        oCell.Text.String = "Новая ячейка!"
End Sub
```

В листинге 49 приведен пример кода, в котором в выделенной таблице текстового документа меняется цвет текста.

Листинг 49. Пример кода

```
Sub Main
        Dim oDoc as Object
        Dim oDisp as Variant
' Выделяем таблицу
        oDoc = ThisComponent.CurrentController
        oDisp = createUnoService("com.sun.star.frame.DispatchHelper")
' Меняем цвет текста
        dim args1(0) as new com.sun.star.beans.PropertyValue
        args1(0).Name = "FontColor"
        args1(0).Value = 65535
        oDisp.executeDispatch(oDoc, ".uno:FontColor", "", 0, args1())
End Sub
```

В листинге 50 приведен пример кода, в котором выделяется таблица по имени, копируется в буфер обмена и, затем, вставляется в конец документа.

Листинг 50. Пример кода

```
Sub Main
        Dim oTable1 as Variant
        Dim oVCursor
        Dim oDispatcher
' Устанавливаем текущий отображаемый курсор
        oVCursor = ThisComponent.CurrentController.getViewCursor()
        oDispatcher = createUnoService("com.sun.star.frame.DispatchHelper")
' Получаем таблицу по имени, можно по индексу .getByIndex(0)
        oTable1 = ThisComponent.getTextTables().getByName("Таблица1")
' Поместим курсор в начало первой ячейки
        ThisComponent.CurrentController.select(oTable1)
' Перемещаем курсор в конец текущей ячейки
        oVCursor.gotoEnd(True)
' Перемещаем курсор в конец таблицы
        oVCursor.gotoEnd(True)
' Копируем таблицу в буфер обмена
        oDispatcher.executeDispatch(ThisComponent.CurrentController.Frame, ".uno:Copy", "", 0,
Array())
' Перемещаем курсор в конец документа и затем вставляем таблицу
        oVCursor.gotoRange(ThisComponent.getText().getEnd(), False)
        oDispatcher.executeDispatch(ThisComponent.CurrentController.Frame, ".uno:Paste", "", 0,
Array())
End Sub
```

Литература

1) Эндрю Питоньяк. OpenOffice.org Объяснение Макросов. — Hentzenwerke Publishing, 2004. Перевод: Дмитрий Чернов (c) 2007-2008
2) Программирование на языке OpenOffice.org BASIC. Перевод: Дмитрий Чернов
3) М.А. Бейн. Изучение программирования макросов для электронных таблиц в OpenOffice.org. OOo Basic и Автоматизация Calc. - 2006 Packt Publishing. Санкт-Петербург 2008. Перевод: Дмитрий Чернов
4) OpenOffice.org 3. Полное руководство пользователя / Р. Ю. Козодаев, А. В. Маджугин / Под ред. Е. В. Ушаковой. — СПб.: БХВ-Петербург, 2010
5) Эндрю Питоньяк (Andrew Pitonyak) OpenOffice.org pro. Автоматизация работы. М: ДМК Пресс, 2008. Перевод: Заимских А. Н.
6) http://api.openoffice.org/docs/common/ref/com/sun/star/module-ix.html.

Приложение 1

Таблица 1.1. Свойства объекта, определяемые сервисом MediaDescriptor

Свойство	Описание
AsTemplate	Документ был загружен как шаблон
Author	Автор этой версии документа; для контроля версий
CharacterSet	Набор символов документа для однобайтовых символов
Comment	Комментарий к текущей версии документа; для контроля версий
DocumentTitle	Если заголовок документа определен, он содержится здесь
FilterName	Фильтр используемый для импорта или сохранения этого документа
FilterOptions	Параметры фильтра используемые для импорта этого документа
FilterData	Дополнительные свойства импорта, если строка *FilterOptions* не достаточна
Hidden	Если аргумент *Hidden* определен во время загрузки, он содержится здесь
InputStream	Если *InputStream* определен во время загрузки, он содержится здесь
InteractionHandler	Обработчик исключений, если ошибка происходит во время импорта
JumpMark	Переход к этой отмеченной позиции после загрузки документа
MediaType	MIME тип этого документа
OpenNewView	Открывает новое представление для уже загруженного документа, вместо того, чтобы только открыть документ во второй раз. Другими словами, запрашивает два представления тех же самых данных
Overwrite	Перезаписывает любой существующий файл при сохранении
Password	Пароль для загрузки или сохранения документа
Preview	Документ, загружается в режиме предварительного просмотра; оптимизирует для использования только для предварительного просмотра
ReadOnly	Документ открывается как только для чтения; контроллер не будет изменять документ
StartPresentation	Немедленно после загрузки документа Impress запустить презентацию
Referer	URL страницы, ссылающейся на документ - например, если открыто, при нажатии на HTTP ссылку
RepairPackage	Открытие документа в режиме восстановления
StatusIndicator	Если индикатор состояния был определен, когда документ был загружен, он содержится здесь
Unpacked	Если *True*, документ LO сохраняется как папка, а не ZIP файл
URL	URL документа
Version	Текущая версия документа, если поддерживается контроль версий
ViewData	Данные представления для использования
ViewId	Идентификатор начального представления

Свойство	Описание
MacroExecutionMode	Определяет, как обрабатываются макросы, когда документ загружаетсяыгвщ куищще

Приложение 2

Таблица 2.1. Свойства, поддерживаемые сервисом com.sun.style.CharacterProperties

Свойство	Описание
CharFontName	Определяет имя шрифта в западном тексте. Это может быть разделенный запятой списком имен
CharFontStyleName	Определяет имя начертания шрифта
CharFontFamily	Определяет имя семейства шрифтов как определено в группе констант *com.sun.star.awt.FontFamily*. • DONTKNOW = 0 - Семейство шрифтов не известно. • DECORATIVE = 1 - Семейство шрифтов использует декоративные шрифты. • MODERN = 2 - Семейство шрифтов - шрифт Modern; это - определенный стиль. • ROMAN = 3 - Семейство шрифтов — шрифт Roman с засечками. • SCRIPT = 4 - Семейство шрифтов - рукописный шрифт. • SWISS = 5 - Семейство шрифтов - шрифт Roman без засечек. • SYSTEM = 6 - Семейство шрифтов - системный шрифт.
CharFontCharSet	Определяет кодировку текста шрифта используя группу констант *com.sun.star.awt.CharSet*. Значения не требуют объяснений: DONTKNOW, ANSI, MAC, IBMPC_437 (набор символов IBM PC номер 437), IBMPC_850, IBMPC_860, IBMPC_86, IBMPC_863, IBMPC_865, SYSTEM и SYMBOL.
CharFontPitch	Определяет расстояние между символами шрифта используя группу констант *com.sun.star.awt.FontPitch*. Значения не требуют объяснений: DONTKNOW, FIXED и VARIABLE
CharColor	Определяет цвет текста в виде длинного целого числа
CharEscapement	Определяет короткое целое число, представляющее процент поднятия или опускания для символов верхнего/нижнего индекса. Отрицательные значения понижают символы
CharHeight	Определяет высоту символа в пунктах в виде десятичного числа
CharUnderline	Определяет тип подчеркивания символа используя группу констант *com.sun.star.awt.FontUnderline*. • NONE = 0 - Нет подчеркивания. • SINGLE = 1 - Одиночная линия. • DOUBLE = 2 - Двойная линия. • DOTTED = 3 - Пунктирная линия. • DONTKNOW = 4 - Неизвестное подчеркивание. • DASH = 5 - Штриховая линия. • LONGDASH = 6 - Длинный штрих. • DASHDOT = 7 - Штрихпунктирная последовательность. • DASHDOTDOT = 8 - Штрихпунктирная последовательность с двумя точками. • SMALLWAVE = 9 - малая волна. • WAVE = 10 - волна. • DOUBLEWAVE = 11 - двойная волна.

Свойство	Описание
	• BOLD = 12 - Жирная линия. • BOLDDOTTED = 13 - Жирный пунктир. • BOLDDASH = 14 - Жирный штрих. • BOLDLONGDASH = 15 - Жирный длинный штрих. • BOLDDASHDOT = 16 - Жирная штрихпунктирная последовательность. • BOLDDASHDOTDOT = 17 - Жирная штрихпунктирная последовательность с двумя точками. • BOLDWAVE = 18 - Жирная волна.
CharWeight	Определяет плотность шрифта используя группу констант *com.sun.star.awt.FontWeight*. • DONTKNOW = 0.000 - Не указанный/неизвестный. • THIN = 50.00 - плотность шрифта 50%. • ULTRALIGHT = 60.00 - плотность шрифта 60%. • LIGHT = 75.00 - плотность шрифта 75%. • SEMILIGHT = 90.00 - плотность шрифта 90%. • NORMAL = 100.00 - нормальная плотность шрифта (100%). • SEMIBOLD = 110.00 - плотность шрифта 110%. • BOLD = 150.00 - плотность шрифта 150%. • ULTRABOLD = 175.00 - плотность шрифта 175%. • BLACK = 200.00 - плотность шрифта 200%.
CharPosture	Определяет положение символа, используя список *com.sun.star.awt.FontSlant* со значениями: • NONE - Без наклона, нормальный текст. • OBLIQUE - Наклонный шрифт (наклон не планируемый в шрифте). • ITALIC - Курсивный шрифт (наклон запланированный в • шрифте). • DONTKNOW - Неизвестный наклон. • REVERSE_OBLIQUE - Обратный наклон (наклон не планируемый в шрифте). • REVERSE_ITALIC - Обратный курсивный шрифт (наклон запланированный в шрифте).
CharAutoKerning	Устанавливается в *True* для использования таблиц кернинговых пар для текущего шрифта. Автоматический кернинг регулирует интервал между определенными парами символов, чтобы улучшить удобочитаемость
CharBackColor	Определяет фоновый цвет текста в виде длинного целого числа
CharBackTransparent	Если *True*, фоновый цвет текста прозрачный
CharCaseMap	Определяет как символы будут отображаться используя группу констант *com.sun.star.style.CaseMap*. Это не изменяет реальный текст - только метод, которым он отображается. • NONE = 0 - Преобразование регистра не выполняется; обычно используемое значение. • UPPERCASE = 1 - Все символы отображаются в верхнем регистре. • LOWERCASE = 2 - Все символы отображаются в нижнем регистре.

Свойство	Описание
	• TITLE = 3 - Первый символ каждого слова отображается как прописная буква (в верхнем регистре). • SMALLCAPS = 4 - Все символы отображаются в верхнем регистре, но шрифтом меньшего размера.
CharCrossedOut	Если *True*, символы содержат линию через них
CharFlash	Если *True*, символы отображаются мигающими
CharStrikeout	Определяет зачёркивание символа используя группу констант *com.sun.star.awt.FontStrikeout*: • NONE = 0 - Нет зачёркивания символов. • SINGLE = 1 - Зачёркивание символов одиночной линией. • DOUBLE = 2 - Зачёркивание символов двойной линией. • DONTKNOW = 3 - Режим зачёркивания не определен. • BOLD = 4 - Зачёркивание символов жирной линией. • SLASH = 5 - Зачёркивание символов при помощи символов '\'. • X = 6 - Зачёркивание символов при помощи символов 'X'.
CharWordMode	Если *True*, белые места (пробелы и табуляции) игнорируют свойства *CharStrikeout* и *CharUnderline*
CharKerning	Определяет значение кернинга символов как короткое целое число
CharLocale	Определяет региональные настройки для символа в виде структуры *com.sun.star.lang.Locale*
CharKeepTogether	Если *True*, LO пробует продолжать последовательность символов на той же самой строке. Если должен произойти разрыв, он происходит перед последовательностью символов
CharNoLineBreak	Если *True*, LO игнорирует разрыв строки в последовательности символов. Если должен произойти разрыв, он происходит после последовательности символов, таким образом возможно, что они пересекут границу
CharShadowed	Если *True*, символы форматируются и отображаются с теневым эффектом
CharFontType	Определяет основную технологию шрифта используя группу констант *com.sun.star.awt.FontType*. • DONTKNOW = 0 - Тип шрифта не известен. • RASTER = 1 - Шрифт - растровый шрифт. • DEVICE = 2 - Шрифт - определяется устройством вывода, например, шрифт принтера. • SCALABLE = 3 - Шрифт масштабируемый.
CharStyleName	Определяет имя начертания шрифта в виде строки
CharContoured	Если *True*, символы форматируются и отображаются с контурным эффектом (3-D контур)
CharCombineIsOn	Если *True*, текст форматируется и отображается с использованием двух строк. Строка *CharCombinePrefix* предшествует тексту в натуральную величину, а строка *CharCombineSuffix* следует за текстом в натуральную величину
CharCombinePrefix	Определяет префикс (обычно круглые скобки) используемый со

Свойство	Описание
	свойством *CharCombineIsOn*
CharCombineSuffix	Определяет суффикс (обычно круглые скобки) используемый со свойством *CharCombineIsOn*
CharEmphasize	Определяет тип и положение знаков ударения в азиатском тексте используя группу констант *com.sun.star.text.FontEmphasis*: • NONE = 0 - Знаки ударения не используются. • DOT_ABOVE = 1 - Точка установленная выше (или справа от вертикального текста) текста. • CIRCLE_ABOVE = 2 - Кружок установленный выше (или справа от вертикального текста) текста. • DISK_ABOVE = 3 - Диск установленный выше (или справа от вертикального текста) текста. • ACCENT_ABOVE = 4 - Знак ударения установленный выше (или справа от вертикального текста) текста. • DOT_BELOW = 11 - Точка установленная ниже (или слева от вертикального текста) текста. • CIRCLE_BELOW = 12 - Кружок установленный ниже (или слева от вертикального текста) текста. • DISK_BELOW = 13 - Диск установленный ниже (или слева от вертикального текста) текста. • ACCENT_BELOW = 14 - Знак ударения установленный ниже (или слева от вертикального текста) текста.
CharRelief	Определяет значение рельефа из группы констант *com.sun.star.text.FontRelief*: • NONE = 0 - Рельеф не используется; обычный текст. • EMBOSSED = 1 - Символы выглядят рельефными (приподнятыми). • ENGRAVED = 2 - Символы выглядят выгравированными (утопленными).
CharRotation	Определяет вращение символа в градусах в виде короткого целого числа. Не все реализации поддерживают все значения
CharRotationIsFitToLine	Если *True*, LO пробует разместить вращаемый текст по высоте окружающей строки
CharScaleWidth	Определяет масштабирование для верхнего и нижнего индексов как процент, используя короткое целое число
HyperLinkURL	Определяет URL гиперссылки (если установлена) в виде строки
HyperLinkTarget	Определяет имя цели для гиперссылки (если установлена) в виде строки
HyperLinkName	Определяет имя гиперссылки (если установлена) в виде строки
VisitedCharStyleName	Определяет стиль символа для посещенной гиперссылки в виде строки
UnvisitedCharStyleName	Определяет имя стиля символа для непосещенной гиперссылки в виде строки
CharEscapementHeight	Определяет дополнительную высоту, используемую для символов верхнего и нижнего индексов как целое число в процентах. Для символов нижнего индекса значение отрицательно

Свойство	Описание
CharNoHyphenation	Если *True*, слово не может быть перенесено на символе
CharUnderlineColor	Определяет цвет подчеркивания как длинное целое число
CharUnderlineHasColor	Если *True*, *CharUnderlineColor* используется для подчеркивания
CharStyleNames	Массив имен стилей символа примененных к тексту. Порядок не важен

Таблица 2.2. Свойства, поддерживаемые сервисом com.sun.style.ParagraphProperties

Свойство	Описание
ParaAdjust	Определяет как абзац выравнивается. Поддерживаются пять значений из списка *com.sun.star.style.ParagraphAdjust*: • LEFT - Выравнивание абзаца по левому краю. • RIGHT - Выравнивание абзаца по правому краю. • CENTER - Выравнивание абзаца по центру. • BLOCK - Выравнивание по ширине каждой строки за исключением последней. • STRETCH - Выравнивание по ширине каждой строки включая последнюю.
ParaLastLineAdjust	Регулирует последнюю строку, если *ParaAdjust* установлен в BLOCK
ParaLineSpacing	Определяет межстрочный интервал абзаца. Свойство - структура типа *com.sun.star.style.LineSpacing*, которая содержит два свойства типа *Short*. Свойство *Height* определяет высоту, а свойство *Mode* определяет, как использовать свойство *Height*. Свойство *Mode* поддерживает значения, определенные в группе констант *com.sun.star.style.LineSpacingMode*. • PROP = 0 - Высота пропорциональна. • MINIMUM = 1 - Высота - минимальная высота строки. • LEADING = 2 - Высота - расстояние до предыдущей строки. • FIX = 3 - Высота фиксированна.
ParaBackColor	Определяет фоновый цвет абзаца в виде длинного целого числа
ParaBackTransparent	Если *True*, устанавливает фоновый цвет абзаца в прозрачный
ParaBackGraphicURL	Определяет URL фонового изображения абзаца
ParaBackGraphicFilter	Определяет имя графического фильтра для фонового изображения абзаца
ParaBackGraphicLocation	Определяет положение фонового изображения используя список *sun.star.style.GraphicLocation*: • NONE - Положение еще не назначено. • LEFT_TOP - Изображение находится в верхнем левом углу. • MIDDLE_TOP - Изображение находится по

Свойство	Описание
	середине верхнего края. • RIGHT_TOP - Изображение находится в верхнем правом углу. • LEFT_MIDDLE - Изображение находится по середине левого края. • MIDDLE_MIDDLE - Изображение находится в центре окружающего объекта. • RIGHT_MIDDLE - Изображение находится по середине правого края. • LEFT_BOTTOM - Изображение находится в нижнем левом углу. • MIDDLE_BOTTOM - Изображение находится по середине нижнего края. • RIGHT_BOTTOM - Изображение находится в нижнем правом углу. • AREA - Изображение масштабируется так, чтобы заполнить все окружающее пространство. • TILED - Изображение повторяется по окружающему объекту подобно плитке.
ParaExpandSingleWord	Если *True*, одиночные слова могут быть растянуты
ParaLeftMargin	Определяет левый отступ абзаца в 0,01 мм в виде длинного целого числа
ParaRightMargin	Определяет правый отступ абзаца в 0,01 мм в виде длинного целого числа
ParaTopMargin	Определяет верхний отступ абзаца в 0,01 мм в виде длинного целого числа. Расстояние между двумя абзацами — максимум нижнего отступа предыдущего параграфа и верхнего отступа текущего абзаца
ParaBottomMargin	Определяет нижний отступ абзаца в 0,01 мм в виде длинного целого числа. Расстояние между двумя абзацами — максимум нижнего отступа текущего абзаца и верхнего отступа следующего абзаца
ParaLineNumberCount	Если *True*, этот абзац включается в нумерацию строк
ParaLineNumberStartValue	Определяет начальное значение для нумерации строк в виде длинного целого числа
PageDescName	Установка этой строки вызывает разрыв страницы перед абзацем. Новая страница использует заданное имя стиля страницы
PageNumberOffset	Задает номер новой страницы если встречается разрыв страницы
ParaRegisterModeActive	Если *True* и если стиль страницы, на которой расположен абзац, также установил регистровый режим в *True*, регистровый режим — активный для этого абзаца. Если регистровый режим активен, каждая строка имеет одну и ту же высоту
ParaTabStops	Определяет позиции табуляции для этого абзаца. Это

Свойство	Описание
	массив структур типа *com.sun.star.style.TabStop*. Каждая структура содержит следующие свойства: • Position - Длинное целое число, положение относительно левого края. • Alignment - Выравнивание текстового диапазона перед табуляцией. Это перечислимый тип *com.sun.star.style.TabAlign*. Разрешенные значения включают LEFT, RIGHT, CENTER, DECIMAL и DEFAULT. • DecimalChar - Определяет, какой символ является десятичным разделителем. • FillChar - Символ, используемый для заполнения пространства между текстом.
ParaStyleName	Определяет имя текущего стиля абзаца
DropCapFormat	Структура, которая определяет, используют ли первые символы абзаца буквицу. *com.sun.star.style.DropCapFormat* содержит следующие свойства: • Lines - Число строк, используемых для буквицы. • Count - Число символов в буквице. • Distance - Расстояние между буквицей и последующим текстом.
DropCapWholeWord	Если *True*, *DropCapFormat* применяется ко всему первому слову
ParaKeepTogether	Если *True*, предотвращает разрыв страницы или столбца после этого абзаца - например, препятствует заголовку стать последней строкой на странице или в колонке
ParaSplit	Если *False*, предотвращает разрыв абзаца на границе двух страниц или столбцов
NumberingLevel	Определяет уровень нумерации абзаца
NumberingRules	Определяет правила нумерации, применяемые для данного абзаца. Этот объект реализует интерфейс *com.sun.star.container.XIndexReplace*
NumberingStartValue	Определяет начальное значение для нумерации если *ParaIsNumberingRestart - True*
ParaIsNumberingRestart	Определяет возобновление нумерации у текущего абзаца (см. *NumberingStartValue*)
NumberingStyleName	Определяет имя стиля нумерации (см. *ParaLineNumberCount*)
ParaOrphans	Определяет минимальное число строк внизу страницы, если абзац размещается более чем на одной странице
ParaWidows	Определяет минимальное число строк наверху страницы, если абзац размещается более чем на одной странице
ParaShadowFormat	Определяет формат тени абзаца как *com.sun.star.table.ShadowFormat*:

Свойство	Описание
	• *Location* - Определяет расположение тени как перечислимый тип *com.sun.star.table.ShadowLocation*. Разрешенные значения включают *NONE*, *TOP_LEFT*, *TOP_RIGHT*, *BOTTOM_LEFT* и *BOTTOM_RIGHT*. • *ShadowWidth* - Определяет размер тени в виде целого числа. • *IsTransparent* - Если *True*, тень прозрачная. • *Color* - Определяет цвет тени в виде длинного целого числа.
LeftBorder	Определяет левую границу как *com.sun.star.table.BorderLine*: • *Color* - Определяет цвет линии. • *InnerLineWidth* - Определяет внутреннюю ширину двойной линии (в 0,01 мм). Если ноль, рисуется одиночная линия. • *OuterLineWidth* - Определяет ширину одиночной линии или внешнюю ширину двойной линии (в 0,01 мм). Если ноль, линия не рисуется. • *LineDistance* - Определяет расстояние между внутренней и внешней частями двойной линии (в 0,01 мм).
RightBorder	Определяет правую границу (см. *LeftBorder*)
TopBorder	Определяет верхнюю границу (см. *LeftBorder*)
BottomBorder	Определяет нижнюю границу (см. *LeftBorder*)
BorderDistance	Определяет расстояние от границы до объекта (в 0,01 мм)
LeftBorderDistance	Определяет расстояние от левой границы до объекта (в 0,01 мм)
RightBorderDistance	Определяет расстояние от правой границы до объекта (в 0,01 мм)
TopBorderDistance	Определяет расстояние от верхней границы до объекта (в 0,01 мм)
BottomBorderDistance	Определяет расстояние от нижней границы до объекта (в 0,01 мм)
BreakType	Определяет тип разрыва, который применен в начале абзаца. Это перечислимый тип *com.sun.star.style.BreakType* со следующими значениями: • NONE - Разрыв столбца или страницы не применяется. • COLUMN_BEFORE - Разрыв столбца применен перед текущим абзацем. Текущий абзац, поэтому, является первым в столбце. • COLUMN_AFTER - Разрыв столбца применен после текущего абзаца. Текущий абзац, поэтому, последнее в столбце. • COLUMN_BOTH - Разрыв столбца применен

Свойство	Описание
	перед и после текущего абзаца. Текущий абзац, поэтому, единственный абзац в столбце. • PAGE_BEFORE - Разрыв страницы применен перед текущим абзацем. Текущий абзац, поэтому, является первым на странице. • PAGE_AFTER - Разрыв страницы применен после текущего абзаца. Текущий абзац, поэтому, последний на странице. • PAGE_BOTH - Разрыв страницы применен перед и после текущего абзаца. Текущий абзац, поэтому, единственный абзац на странице.
DropCapCharStyleName	Определяет имя стиля символа для буквиц
ParaFirstLineIndent	Определяет отступ для первой строки в абзаце
ParalsAutoFirstLinelndent	Если *True*, первая строка с автоматическим отступом
ParalsHyphenation	Если *True*, применяется автоматическая расстановка переносов
ParaHyphenationMaxHyphens	Определяет максимальное количество последовательных переносов для слов, содержащихся в текущем абзаце
ParaHyphenationMaxLeadingChars	Определяет максимальное число символов, которые остаются перед символом переноса
ParaHyphenationMaxTrailingChars	Определяет максимальное число символов, которые остаются после символа переноса
ParaVertAlignment	Определяет вертикальное выравнивание абзаца. Это группа констант типа *com.sun.star.text.ParagraphVertAlign* с разрешенными значениями: • AUTOMATIC = 0 - В автоматическом режиме, горизонтальный текст выравнивается по базовой линии. То же самое относится к тексту, повернутому на 90 градусов. Текст, повернутый на 270 градусов выравнивается по центру. • BASELINE = 1 - Текст выравнивается по базовой линии. • TOP = 2 - Текст выравнивается по по верху. • CENTER = 3 - Текст выравнивается по центру. • BOTTOM = 4 - Текст выравнивается по низу.
ParaUserDefinedAttributes	Хранит XML атрибуты, которые сохраняются и восстанавливаются из автоматических стилей внутри XML файлов. Объект реализует интерфейс *com.sun.star.container.XNameContainer*
NumberingIsNumber	Если *True*, нумерация абзаца - число, не имеющая символа. Это бесполезно, если абзац не является частью абзаца из последовательности нумерации
ParalsConnectBorder	Если *True*, границы абзаца объединяются с предыдущим абзацем, если границы идентичны. Эта свойство может быть бесполезным

Приложение 3

Таблица 3.1. Свойства, поддерживаемые сервисом com.sun.star.table.CellProperties

Свойство	Описание
CellStyle	Необязательное свойство; имя стиля ячейки как строка
CellBackColor	Фоновый цвет ячейки как длинное целое число (см. *IsCellBackgroundTransparent*)
IsCellBackgroundTransparent	Если *True*, фон ячейки прозрачен, и *CellBackColor* игнорируется
HoriJustify	Горизонтальное выравнивание ячейки как набор *com.sun.star.table.CellHoriJustify*: • STANDARD - выравнивание по умолчанию, вправо для чисел и влево для текста. • LEFT - Содержимое выравнивается по левому краю ячейки. • CENTER - Содержимое центрируется по горизонтали. • RIGHT - Содержимое выравнивается по правому краю ячейки. • BLOCK - Содержимое подогнано по ширине ячейки. • REPEAT - Содержимое повторяется, чтобы заполнить ячейку (но это, кажется, не работает).
VertJustify	Вертикальное выравнивание ячейки как набор *com.sun.star.table.CellVertJustify*: • STANDARD - Используется по умолчанию. • TOP - Выравнивание по верхнему краю ячейки. • CENTER - Выравнивание по вертикали по середине ячейки. • BOTTOM - Выравнивание по нижнему краю ячейки.
IsTextWrapped	Если *True*, содержимое ячейки автоматически переносится по словам на правой границе
ParaIndent	Отступ содержимого ячейки (в 0,01 мм) как короткое целое число
Orientation	Если *RotateAngle* - ноль, определяет ориентацию содержания ячейки как набор *com.sun.star.table.CellOrientation*: • STANDARD - содержимое ячейки показано слева направо. • TOPBOTTOM - содержимое ячейки показано сверху вниз. • BOTTOMTOP - содержимое ячейки показано снизу вверх. • STACKED - То же самое, что TOPBOTTOM, но каждый символ горизонтально.
RotateAngle	Определяет, на сколько повернуть содержимое ячейки (в 0,01 градуса) как длинное целое число. Вся строка поворачивается как единый блок, а не как отдельные символы
RotateReference	Определяет край, по которому выравниваются вращаемые ячейки, используя тот же самый набор, что и *VertJustify*
AsianVerticalMode	Если *True*, только азиатские символы используют вертикальную

Свойство	Описание
	ориентацию. Другими словами, в азиатском режиме только азиатские символы печатаются в горизонтальной ориентации, если свойство *Orientation* - STACKED; для других ориентаций это значение не используется
TableBorder	Описание границы ячейки или диапазона ячеек (см. Таблицу 12). Когда используется с одиночной ячейкой, устанавливает значения границы для одиночной ячейки. Когда используется с диапазоном ячеек, границы - для внешних краев диапазона, а не отдельных ячеек
TopBorder	Описание верхней границы ячейки (см. Таблицу 11)
BottomBorder	Описание нижней границы ячейки (см. Таблицу 11)
LeftBorder	Описание левой границы ячейки (см. Таблицу 11)
RightBorder	Описание правой границы ячейки (см. Таблицу 11)
NumberFormat	Индекс числового формата ячейки. Собственное значение может быть определено при использовании интерфейса *com.sun.star.util.XNumberFormatter*, поддерживаемого в соответствии с документом
ShadowFormat	Определяет формат тени, используя структуру *com.sun.star.table.ShadowFormat*: • *Location* - положение тени как набор *com.sun.star.table.ShadowLocation* с допустимыми значениями: NONE, TOP_LEFT, TOP_RIGHT, BOTTOM_LEFT и BOTTOM_RIGHT. • *ShadowWidth* - размер тени как короткое целое число. • *IsTransparent* - Если True, тень прозрачная. • *Color* - цвет тени как длинное целое число.
CellProtection	Определяет защиту ячейки как структура *com.sun.star.util.CellProtection*: • *IsLocked* - Если *True*, ячейка заблокирована от модификаций пользователем. • *IsFormulaHidden* - Если *True*, формула скрыта от пользователя. • *IsHidden* - Если *True*, ячейка скрыта от пользователя. • *IsPrintHidden* - Если *True*, ячейка скрыта при выводе на печать.
UserDefinedAttributes	Это свойство используется для хранения дополнительных атрибутов в интерфейсе *com.sun.star.container.XNameContainer*