О применении библиотеки FastScript в своих проектах. Часть-1 «Расширение функционала»

Оглавление

Термины и определения

В настоящем документе применяются следующие термины с соответствующими определениями (см. таблицу 1).

Таблица 1 − Список терминов, примененных в настоящем документе

Термин	Определение
1	2
FastScript	Библиотека для выполнения скриптов. Правообладатель и разработчик FastScript: компания ООО «Быстрые отчеты»
FS-скрипт	См. «Скрипт»
Интерпретатор	1. Программа, обеспечивающая GUI-интерфейс, ориентированный на формирование и исполнение Скриптов. 2. Программный объект из состава Обвязки (см. ниже). **Важно!** Не путать с компонентом `TfsScript` библиотеки FastScript
Обвязка	Дополнительные программные объекты, позволяющие Программисту упростить и облегчить процесс адаптации библиотеки FastScript для своих проектов **без изменения** исходных кодов библиотеки FastScript
Пользователь	Специалист какой-либо автоматизируемой предметной области (геолог, технолог, конструктор, инженер и т.д.), использующий соответствующее прикладное программное обеспечение
Правообладатель	Компания ООО «Быстрые отчеты» (она же является и разработчиком FastScript)
Скрипт	Исходный текст программного модуля, формируемый Пользователем (или программистом) с применением библиотеки FastScript
Читатель	Разработчик прикладного программного обеспечения (программист)

Принятые сокращения

В настоящем документе применяются следующие сокращения, аббревиатуры и словосочетания (см. таблицу 2).

Таблица 2 − Список сокращений, аббревиатур и словосочетаний, примененных в настоящем документе

Сокращение	Обозначение
1	2
FS	См. термин «FastScript»
FSI	См. термин «Интерпретатор» в части «Обвязка»
UTF-8	Распространенный стандарт кодирования символов, позволяющий более компактно хранить и передавать символы Юникода, используя переменное количество байт (от 1 до 4), и обеспечивающий полную обратную совместимость с 7-битной кодировкой ASCII
БД	База данных
ИБ	Информационная база
ИС	Информационная система
ПО	Программное обеспечение
ППО	Прикладное программное обеспечение
Руководство	Документ «FastScript. Библиотека скриптов. Руководство разработчика»
СУБД	Система управления БД

1 Введение

В данном документе приведены:

используемые при изложении: перечень терминов и сокращений; принятые обозначения;

уровень подготовки Читателя;

краткая информация о библиотеке FastScript и Правообладателе;

примеры, иллюстрирующие применение FastScript;

полезные «приемы», позволяющие Программисту упростить и облегчить процесс адаптации библиотеки FastScript для своих проектов без изменения исходных кодов библиотеки FastScript.

1.1 Соглашения

1. «Сколько геологов, столько и мнений» или «на вкус и цвет товарищей нет».
2. Автор не ставит перед собой цель – удивить этот мир «красотой» и оптимальностью исходного кода или оригинальностью идей, а также применяемых алгоритмов, методик и способов решений задач.
3. Автор, в настоящем документе, лишь излагает свой собственный подход к применению библиотеки FastScript для решения (программирования) различных задач с целью помочь тем, кому это может быть полезным.
4. Иллюстрирующие примеры сформированы в среде Delphi Embarcadero RAD Studio.
5. Читателю доступно «Руководство Разработчика по FastScript» на официальном сайте Правообладателя.
6. В настоящем документе НЕ приводятся исходные коды библиотеки FastScript.
7. Автор предоставляет Читателю право использовать любую информацию, изложенную в настоящем документе, по своему усмотрению (без каких-либо ограничений).
8. Далее, в настоящем документе, считать идентичными (по значению) термины и фразы:
   «Правообладатель», «Разработчик»;
   «Читатель», «Программист».

1.2 Уровень подготовки Читателя

Предполагается, что Читатель умеет свободно разбираться в исходных кодах на языке программирования Pascal (а точнее, Object Pascal) и владеет практическими навыками разработки Приложений в среде Delphi.

2 О библиотеке FastScript

Правообладателем и разработчиком библиотеки FastScript является компания ООО «Быстрые отчеты».

Официальный сайт компании размещен здесь:
https://xn--90aia9aifhdb2cxbdg.xn--p1ai/ru/

Автором библиотеки является Александр Цыганенко (которому принадлежит первоначальная идея ее создания).

Информация о назначении библиотеки FastScript и ее возможностях приведены здесь:
https://xn--90aia9aifhdb2cxbdg.xn--p1ai/ru/product/fast-script/

Раздел сайта Правообладателя, где размещена официальная документация:
https://xn--90aia9aifhdb2cxbdg.xn--p1ai/ru/download/documentation/

Ниже, приведен ряд скринов, сделанных со страницы официального сайта (на момент формирования данного документа):

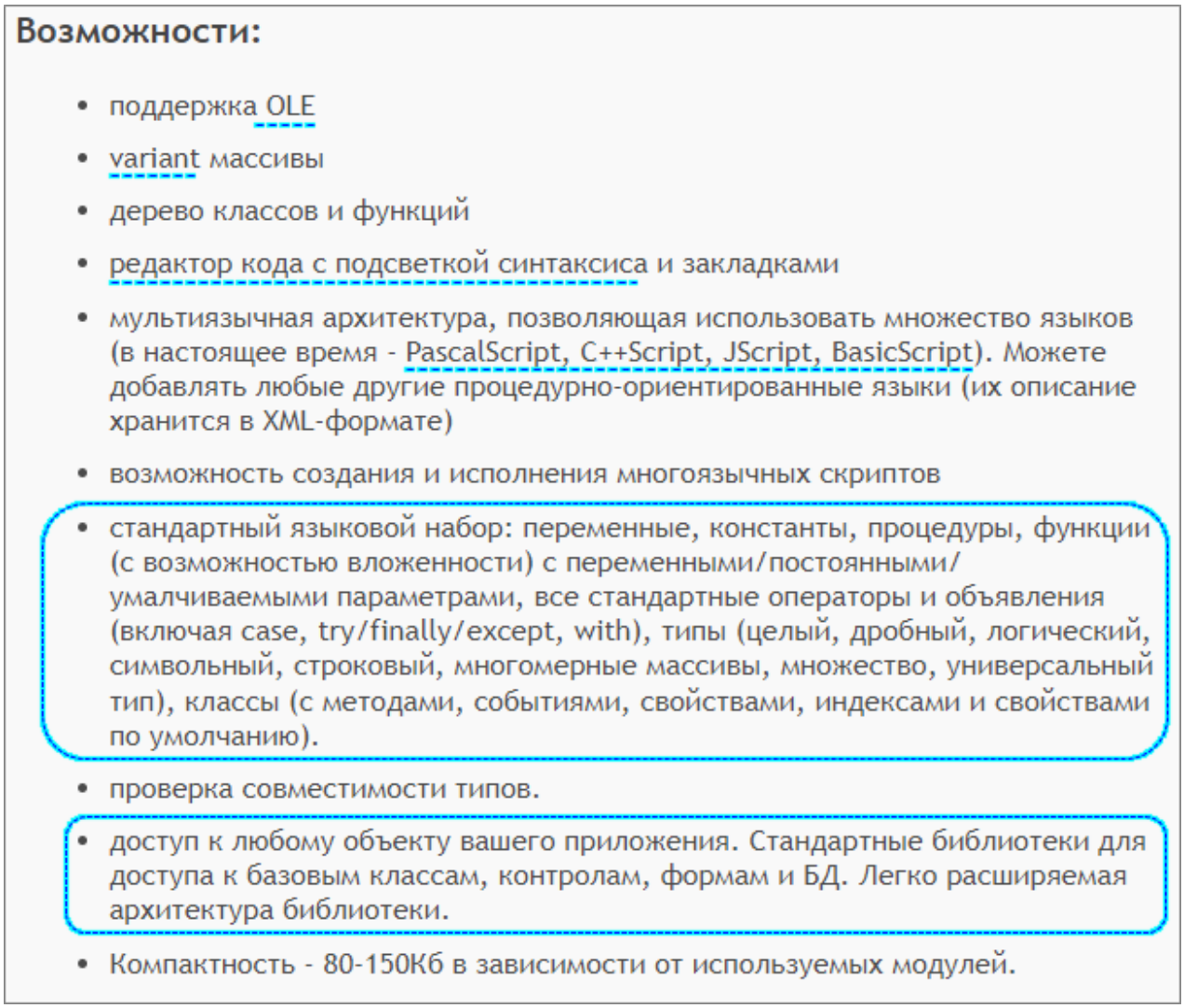
FastScript — библиотека для выполнения скриптов. Она будет полезна разработчикам, желающим добавить возможности исполнения скриптовых программ в свои проекты. Delphi 2010-XE8, C++Builder 2010-XE8, Embarcadero RAD Studio 11 и Lazarus

FastScript написан полностью на 100% Object Pascal и может быть установлен в Delphi 2010-XE8, C++Builder 2010-XE8, Embarcadero RAD Studio 11 и Lazarus.

Максимальная гибкость и мощность

Уникальные возможности FastScript - возможность одновременного использования нескольких языков (в настоящее время - PascalScript, C++Script, JScript и BasicScript), вы можете писать скрипты используя ваш любимый язык программирования. FastScript не использует Microsoft Scripting Host, а потому может использоваться как в Windows, так и в Linux, а также в Mac OS.

FastScript объединяет в себе кросс-платформенность, быстрое выполнение кода, компактность, богатый выбор возможностей и великолепную масштабируемость. Сделайте ваши приложения максимально гибкими и мощными с FastScript!



Возможности:

- поддержка OLE
- variant массивы
- дерево классов и функций
- редактор кода с подсветкой синтаксиса и закладками
- мультиязычная архитектура, позволяющая использовать множество языков (в настоящее время - PascalScript, C++Script, JScript, BasicScript). Можете добавлять любые другие процедурно-ориентированные языки (их описание хранится в XML-формате)
- возможность создания и исполнения многоязычных скриптов
- стандартный языковой набор: переменные, константы, процедуры, функции (с возможностью вложенности) с переменными/постоянными/умалчиваемыми параметрами, все стандартные операторы и объявления (включая case, try/finally/except, with), типы (целый, дробный, логический, символьный, строковый, многомерные массивы, множество, универсальный тип), классы (с методами, событиями, свойствами, индексами и свойствами по умолчанию).
- проверка совместимости типов.
- доступ к любому объекту вашего приложения. Стандартные библиотеки для доступа к базовым классам, контролам, формам и БД. Легко расширяемая архитектура библиотеки.
- Компактность - 80-150Кб в зависимости от используемых модулей.

В тех случаях, когда необходимо сформировать гибкий программный инструментарий, который можно оперативно адаптировать под быстро меняющиеся условия его эксплуатации, применение библиотеки FastScript является наилучшим выбором.

3 Практическое применение библиотеки FastScript

В разделах ниже приведены рекомендации и примеры, которые могут быть полезными (с точки зрения Автора настоящего документа).

Общие замечания:

1. Следует отметить, что существенная часть задач, где может быть применена библиотека FastScript, это автоматизация выполнения различного рода прикладных (специфичных) задач конечного Пользователя − самим же Пользователем (в рамках эксплуатируемой ИС). При этом, Пользователь (даже очень продвинутый), как правило (все-таки) НЕ является программистом со «всеми вытекающими».
   Чем более дружественным (комфортным) для него будет инструментарий, тем эффективнее он будет его применять. Одним из факторов («облегчающих жизнь» конечному Пользователю), является максимально возможная локализация программного инструментария не только в части GUI, но и в части идентификаторов функций, процедур, переменных и констант (что при использовании FastScript легко реализуемо). См. об этом (примеры применения) подробнее в разделах 3.1.1, 3.1.2 и 4.3.
2. В силу того, что возможности библиотеки FastScript существенно велики, а «аппетит растет во время еды», то через довольно-таки небольшой промежуток времени количество «специальных» функций (которые добавляет Программист) может стать знАчимо «большим» (особенно, в «больших» ИС). И в этом случае необходимо сразу предусматривать возможность группировки дополнительного функционала по отдельным модулям (которые будут подключаться по мере необходимости в процессе разработки конкретного скрипта или группы скриптов).

3.1 Расширение функционала FastScript

1. Предполагается, что Читателю доступно «Руководство Разработчика по FastScript» на официальном сайте Правообладателя. Т.е., дублировать изложенную там информацию нет смысла.
2. Автор излагает свой собственный подход к применению библиотеки FastScript, совершенно не претендуя на «истину в последней инстанции».

3.1.1 Добавление новых функций и процедур

3.1.1.1 Функция MyFunc_AddToFS(…)

Добавить в FastScript новую процедуру/функцию на «нескольких языках».

Количество «нескольких языков» концептуально не ограничено.

Конечно, библиотека FastScript предоставляет возможность реализовать «несколько языков» и без использования дополнительных средств, например, так:

```
fsScript1.AddMethod('function INI_Файл_Редактировать(fnINI:string): boolean;', fsCallMethod);
fsScript1.AddMethod('function INI_File_Edit(fnINI:string): boolean;', fsCallMethod);
```

Но в случае изменений перечня параметров функции/процедуры или добавления «нового языка» при локализации, это может привести к ошибкам из-за банальной потери внимания.

Входные параметры функции MyFunc_AddToFS(…):

Параметр	Назначение
1	2
fsScr	Компонент TfsScript (из библиотеки FastScript)
sWhat	Вид: ‘procedure’ или ‘function’
sFuncRes	Тип возвр. значения для ‘function’ или пустая строка для ‘procedure’
sListFuncNames	Перечень (через точку с запятой) идентификаторов (наименований) функции/процедуры
sFuncScript	Текст "объявления" функции/процедуры (без окаймляющих скобок)
fsCallMethod	Функция (метод) формы(TForm), где реализована обработка вызова функции/процедуры

Дополнительные (используемые) функции (см. Приложение):

```
Get_CountWords_In_String(…);
Get_Word_From_String(…).
```

```
function MyFunc_AddToFS(fsScr: TfsScript;
                        sWhat: string;
                        sFuncRes: string;
                        sListFuncNames: string;
                        sFuncScript: string;
                        fsCallMethod: TfsCallMethodEvent
                       ):boolean;
Var
 sNameFunc:string;
 i,c:integer;
 Sx:string;
 s222:string;
begin
  Result:=false;
  if Assigned(fsScr) then begin
     if Assigned(fsCallMethod) then begin
        sWhat:=trim(sWhat);
        if length(sWhat)>0 then begin
           sListFuncNames:=trim(sListFuncNames);
           if length(sListFuncNames)>0 then begin
              s222:='';
              sFuncRes:=trim(sFuncRes);
```

```
                if AnsiUpperCase(sWhat)='PROCEDURE' then sFuncRes:='';
                if sFuncRes<>'' then begin
                   s222:=':';
                   if sFuncRes[length(sFuncRes)]<>';' then begin
                      sFuncRes:=sFuncRes+';';
                   end;
                end
                else begin
                   sFuncRes:=';';
                end;
                sFuncScript:=trim(sFuncScript);
                s222:='';
                if sFuncScript<>'' then begin
                   if sFuncScript[length(sFuncScript)]=';' then begin
                      sFuncScript[length(sFuncScript)]:=' ';
                   end;
                end;
                sFuncScript:=trim(sFuncScript);
                c:=Get_CountWords_In_String(
                                            sListFuncNames,
                                            #32+#9+';'
                                           );
                if c>0 then begin
                   Result:=true;
                   i:=0;
                   while i<c do
                    begin
                     i:=i+1;
                     sNameFunc:=trim(Get_Word_From_String(
                                                          sListFuncNames,
                                                          i,
                                                          #32+#9+';'
                                                         ));
                     if length(sNameFunc)>0 then begin
                        if length(sFuncScript)>0 then begin
                           Sx:=sWhat+' '+sNameFunc+'('+sFuncScript+')'+s222+sFuncRes;
                        end
                        else begin
                           Sx:=sWhat+' '+sNameFunc+s222+sFuncRes;
                        end;
                        //ShowMessage('Объявление:'+#10+Sx); //для тестирования
                        fsScr.AddMethod(Sx, fsCallMethod);
                     end
                     else begin
                        Result:=false;
                        i:=c+1;
                     end;
                   end;
                end;
             end;
          end;
       end;
    end;
 end;
end;
```

Примеры использования.

Пример-1. Добавление процедуры

```
MyFunc_AddToFS(fsScript1,
               'procedure','',
               'Экран_Курсор;Screen_Cursor',
               'sCursor:string='+#39+'crDefault'+#39
               ,
               fsCallMethod);
```

В результате, процедура будет объявлена, как:

```
procedure Экран_Курсор(sCursor:string='crDefault');
procedure Screen_Cursor(sCursor:string='crDefault');
```

Текст «нашей» процедуры, где реализован соответствующий функционал, см. в разделе «4.2.1 Процедуры Экран_Курсор(…); Screen_Cursor».

Пример-2: Добавление функции

```
MyFunc_AddToFS(fsScript1,
      'function', 'integer',
      'Файлы_Список_Получить;Files_List_Get',
      'List: TStrings;'
     +' FullNameMask: string='+#39+'*.*'+#39+';'
     +' YesFileNamesOnly: boolean=true'
      ,
      fsCallMethod);
```

В результате, функция будет объявлена, как:

```
function Файлы_Список_Получить(List: TStrings; FullNameMask:
string='*.*'; YesFileNamesOnly: boolean=true):integer;
function Files_List_to_Get(List: TStrings; FullNameMask: string='*.*';
YesFileNamesOnly: boolean=true):integer;
```

Текст «нашей» функции, где реализован соответствующий функционал, см. в разделе «4.2.2 Функции Файлы_Список_Получить(…); Files_List_Get».

3.1.1.2 Реализация вызовов объявленных процедур и функций

В документации к FastScript приведено детальное и корректное описАние этого процесса.

Здесь – лишь в качестве иллюстрации к примерам, приведенным выше.

ВАЖНО!!! Обработчик типа **`TfsCallMethodEvent`** должен быть обязательно методом формы (**`TForm`**).

В общем-то, это не «такое уж страшное» ограничение (и даже наоборот, это вполне можно рассматривать, как преимущество).

Поскольку, в дальнейшем, можно использовать формы (`TForm`), как «контейнеры» для группировки новых (добавляемых) процедур, функций, глобальных переменных (в случае необходимости) и констант, если в этом возникнет необходимость.

На рисунках ниже приведены скрины исходников, применительно к примерам, приведенным выше.

```
unit fs001_main;

interface

uses

  fs_iinterpreter,

  Winapi.Windows, Winapi.Messages, System.SysUtils, Syste
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.Buttons, Vcl.
  Vcl.ComCtrls, fs_synmemo, fs_ipascal, fs_iclassesrtti;

type
  TForm1 = class(TForm)
    Panel1: TPanel;
    GroupBox7: TGroupBox;
    SpeedButton3: TSpeedButton;
    fsSyntaxMemo1: TfsSyntaxMemo;
    fsScript1: TfsScript;
    fsPascal1: TfsPascal;
    fsClassesRTTI1: TfsClassesRTTI;
    procedure SpeedButton1Click(Sender: TObject);
    procedure SpeedButton2Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure SpeedButton3Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
    function fsCallMethod(Instance: TObject;
                          ClassType: TClass;
                          const MethodName: String;
                          var Params: Variant): Variant;
  end;

var
  Form1: TForm1;
```

Рисунок 1 – Объявление `fsCallMethod`, как метода формы (`TForm`)

```
function TForm1.fsCallMethod(Instance: TObject;
                             ClassType: TClass;
                             const MethodName: String;
                             var Params: Variant): Variant;
Var
  B:boolean;
begin
  Result:=0;
  //--------------------------------
  if POS(AnsiUpperCase(';'+MethodName+';'),
         AnsiUpperCase(';'+'Экран_Курсор;Screen_Cursor'+';')
        )>0 then begin
     Screen_Cursor_Set(trim(Params[0]));
     Application.ProcessMessages;
  end;
  //--------------------------------
  //--------------------------------
  if POS(AnsiUpperCase(';'+MethodName+';'),
         AnsiUpperCase(';'+'Файлы_Список_Получить;Files_List_Get'+';')
        )>0 then begin
     B:=Params[2];
     Result:=Files_ToStrings(
                             POINTER(Integer(Params[0])), //List:TStrings
                             trim(Params[1]),             //FullNameMask
                             B                            //YesFileNamesOnly
                            );
     Application.ProcessMessages;
  end;
  //--------------------------------
end;
```

1

Рисунок 2 – Исходный текст метода формы `fsCallMethod`

Важно! Следует обратить внимание на фрагмент текста, отмеченный выноской-1 на рисунке 2.

Есть 2 варианта «получения» параметра типа TStrings:

1. «По старинке»:
   `POINTER(Integer(Params[0]));`
2. Ныне рекомендуемый Разработчиком:
   `TStrings(frInteger(Params[0]));`

Автор (пока еще) использует вариант-1 (консерватор, однако ☺)…

Исходные тексты используемых функций/процедур приведены в Приложении:
«4.2.1 Процедуры Экран_Курсор(…); Screen_Cursor»;
«4.2.2 Функции Файлы_Список_Получить(…); Files_List_Get».

3.1.1.3 Тестирование того, что выше наработали…

«План – есть план, пока он план», но «лучше один раз увидеть, чем десять раз услышать».

На рисунке ниже приведен исходный текст FS-скрипта для тестирования.

Выполнить скрипт

Скрипт:

```
#language PascalScript

Const
  //FastScript позволяет легко
  //локализовать (применительно к кириллице)
  //имена констант
  МаскаФайлов = 'D:\fs_test\*.*';

Var
  //FastScript позволяет легко
  //локализовать (применительно к кириллице)
  //имена переменных
  Список: TStrings;

BEGIN

  Список := TStringList.Create;
  TRY
    //Новая процедура
    Экран_Курсор('crHourGlass');
    //Новая функция
    Файлы_Список_Получить(
                          Список,
                          МаскаФайлов
                         );
    Экран_Курсор('crDefault');
    ShowMessage(Список.Text);
  FINALLY
    Список.Free;
    Список:=nil;
    Экран_Курсор('crDefault');
  END;

END.
```

Рисунок 3 – Исходный текст FS-скрипта для тестирования

На рисунке ниже приведен исходный текст обработчика события: нажатие на кнопку «Выполнить скрипт» (см. рисунок выше).

```
procedure TForm1.SpeedButton3Click(Sender: TObject);
begin
  Application.ProcessMessages;
  //---------------------
  fsScript1.Clear;
  fsScript1.Lines.Text := fsSyntaxMemo1.Lines.Text;
  fsScript1.Parent := fsGlobalUnit;
  fsScript1.SyntaxType := 'PascalScript';
  //---------------------
  //---------------------
  MyFunc_AddToFS(fsScript1,
                 'procedure','',
                 'Экран_Курсор;Screen_Cursor',
                 'sCursor:string='+#39+'crDefault'+#39,
                 fsCallMethod);
  MyFunc_AddToFS(fsScript1,
                  'function', 'integer',
                  'Файлы_Список_Получить;Files_List_Get',
                  'List: TStrings;'
                 +' FullNameMask: string='+#39+'*.*'+#39+';'
                 +' YesFileNamesOnly: boolean=true'
                  ,
                  fsCallMethod);
  //---------------------
  //---------------------
  if fsScript1.Compile then begin
     fsScript1.Execute;
  end
  else begin
     ShowMessage('Ошибка компиляции скрипта: '+fsScript1.ErrorMsg);
  end;
  //---------------------
end;
```

Рисунок 4 – Исходный текст обработчика события:
нажатие на кнопку «Выполнить скрипт»

На рисунке ниже приведен скрин экрана (результат выполнения **`ShowMessage(Список.Text)`**, см. рисунок 3 и рисунок ниже).

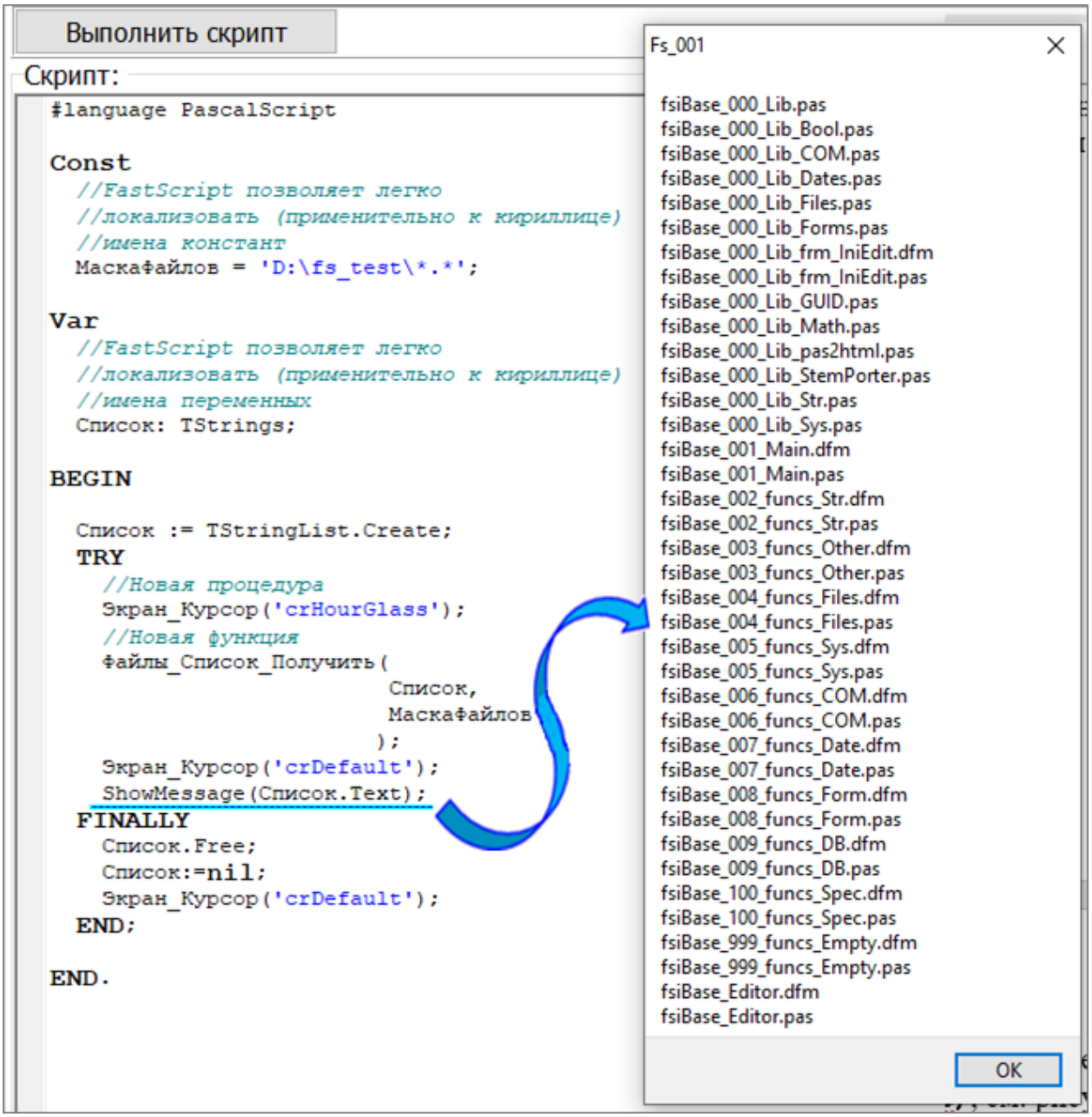


Рисунок 5 – Результат выполнения FS-скрипта

3.1.2 Добавление новых констант

При добавлении новых констант Автор использует тот же подход, который приведен в разделе: «3.1.1.1 Функция MyFunc_AddToFS(…)».

3.1.2.1 Функция MyConst_AddToFS(…)

Добавить в FastScript новую константу на «нескольких языках».

Входные параметры функции MyConst_AddToFS(…):

Параметр	Назначение
1	2
fsScr	Компонент TfsScript (из библиотеки FastScript)
sListConstNames	Перечень (через точку с запятой) идентификаторов (наименований) константы
sConstType	Тип константы (из перечня допустимых, см. Руководство)
sConstVal	Значение константы

Дополнительные (используемые) функции (см. Приложение):

`Get_CountWords_In_String(…);`
`Get_Word_From_String(…).`

```
function MyConst_AddToFS(fsScr: TfsScript;
                         sListConstNames:string;
                         sConstType:string;
                         sConstVal:string
                        ):boolean;
Var
 sNameConst:string;
 i,c:integer;
begin
  Result:=false;
  if Assigned(fsScr) then begin
     sListConstNames:=trim(sListConstNames);
     sConstType:=trim(sConstType);
     sConstVal:=trim(sConstVal);
     if (sListConstNames<>'') and
        (sConstType<>'') and
        (sConstVal<>'') then begin
         c:=Get_CountWords_In_String(sListConstNames, #32+#9+';');
         if c>0 then begin
            Result:=true;
            i:=0;
            while i<c do
             begin
              i:=i+1;
              sNameConst:=trim(Get_Word_From_String(sListConstNames,
                                                    i,
                                                    #32+#9+';'));
              if sNameConst<>'' then begin
                 fsScr.AddConst(sNameConst, sConstType, sConstVal);
              end
              else begin
                 Result:=false;
                 i:=c+1;
              end;
            end;
         end;
     end;
  end;
end;
```

На рисунке ниже приведен исходный текст обработчика события: нажатие на кнопку «Выполнить скрипт» (см. рисунок 3).

```
procedure TForm1.SpeedButton3Click(Sender: TObject);
begin
  Application.ProcessMessages;
  //---------------------
  fsScript1.Clear;
  fsScript1.Lines.Text := fsSyntaxMemo1.Lines.Text;
  fsScript1.Parent := fsGlobalUnit;
  fsScript1.SyntaxType := 'PascalScript';
  //---------------------
  //---------------------
  MyFunc_AddToFS(fsScript1,
                 'procedure','',
                 'Экран_Курсор;Screen_Cursor',
                 'sCursor:string='+#39+'crDefault'+#39,
                 fsCallMethod);
  MyFunc_AddToFS(fsScript1,
                  'function', 'integer',
                  'Файлы_Список_Получить;Files_List_Get',
                  'List: TStrings;'
                 +' FullNameMask: string='+#39+'*.*'+#39+';'
                 +' YesFileNamesOnly: boolean=true'
                  ,
                  fsCallMethod);
  //---------------------
  //---------------------
  MyConst_AddToFS(fsScript1,
                   'Ускорение_свободного_падения',
                   'real',
                   '9.8'
                  );
  MyConst_AddToFS(fsScript1,
                   'Скорость_света;Speed_of_light',
                   'extended',
                   '299792458.0'
                  );
  //---------------------
  //---------------------
  if fsScript1.Compile then begin
     fsScript1.Execute;
  end
  else begin
     ShowMessage('Ошибка компиляции скрипта: '+fsScript1.ErrorMsg);
  end;
  //---------------------
end;
```

Новые константы

Рисунок 6 – Исходный текст обработчика события:
нажатие на кнопку «Выполнить скрипт»
(новые константы)

На рисунке ниже приведен скрин экрана (текст скрипта и результат его выполнения).

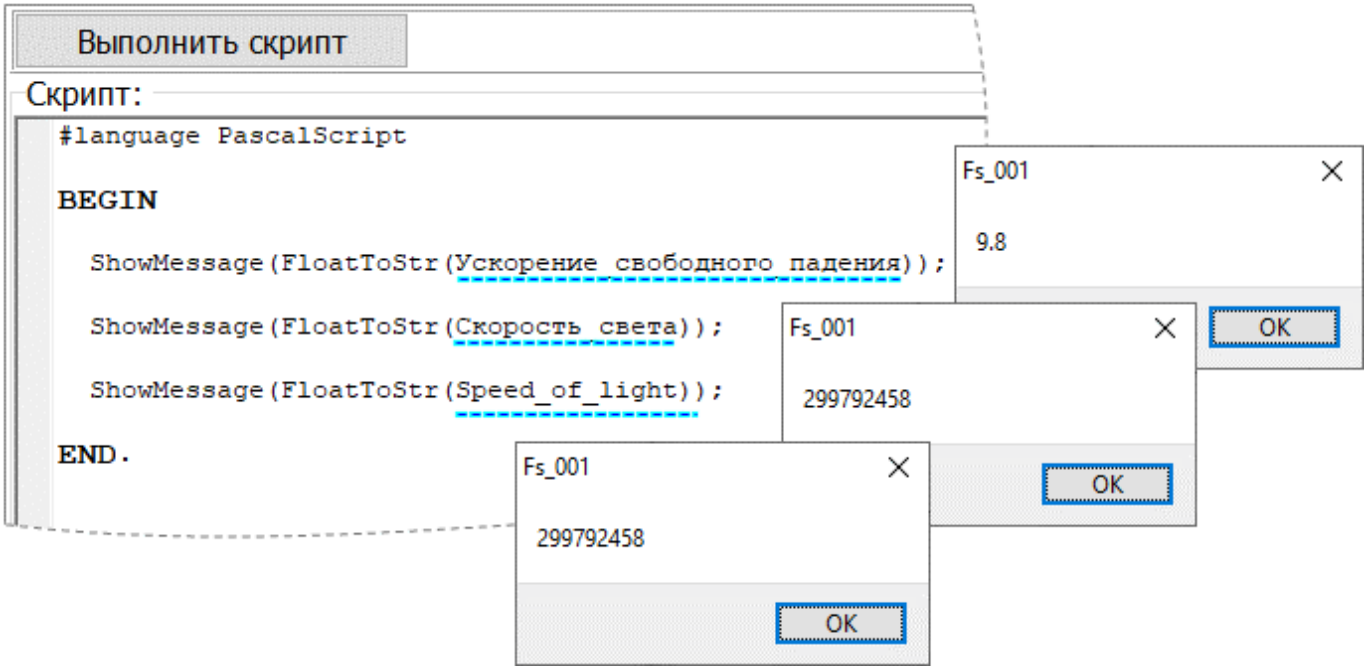


Рисунок 7 – Текст FS-скрипта и результат его выполнения
(значения новых констант)

См., также, раздел 4.3.

Важно! Использование констант представляет, также, практический интерес, как механизм передачи входных параметров для Скрипта (хранящегося, например, в ИБ) от внешних инициаторов информационного обмена. В качестве таких инициаторов могут выступать, например, какие-либо «узлы» в распределенной (в том числе и иерархической) сети ИС. Эту тему предполагается более подробно затронуть в одной из последующих статей.

Впрочем, «Если хочешь рассмешить Бога – расскажи ему о своих планах»… ☺

3.1.3 Добавление новых переменных

Переменные конкретного FS-скрипта актуальны, как правило, именно в контексте этого Скрипта (и/или) другого Скрипта (если используется «Uses»).

Поэтому, Автор не «злоупотребляет» возможностью библиотеки FastScript добавлять переменные вне текста самого Скрипта.

Техническое описание механизма добавления переменных представлено в Руководстве.

Добавление новой переменной:

```
fsScript1.AddVariable(
                       ИдентификаторПеременной, //string
                       ТипПеременной,           //string
                       ЗначениеПеременной
                      );
```

Важно! Использование переменных представляет, также, практический интерес, как механизм передачи вЫходных параметров из Скрипта (хранящегося, например, в ИБ) для внешних инициаторов информационного обмена (см., также, раздел «3.1.2.1 Функция MyConst_AddToFS(…)»). В этом случае, как раз и может быть использован «штатный», для FastScript, механизм создания переменных Скрипта.

См., также, раздел 4.3.

4 Приложения

4.1 Приложение−1. Вспомогательные функции

4.1.1 Функция Get_CountWords_In_String(…)

Определить кол-во слов в строке Sx.

UnChar − множество разделителей.

YesAllTrim_Before − если TRUE, то перед обработкой удалить из строки левые и правые пробелы.

```
function Get_CountWords_In_String(
                                    Sx : String;
                                    UnChar: String=' ';
                                    YesAllTrim_Before: boolean=true
                                   ) : integer;
var
  InWord : byte;
  i : integer;
begin
  Result:=0;
  if YesAllTrim_Before then Sx:=trim(Sx);
  if length(Sx)>0 then begin
     InWord := 0;
     i:=0;
     while i<length(Sx) do
      begin
       i:=i+1;
       if POS(Sx[i],UnChar)<=0 then begin
          if InWord<=0 then Result:=Result+1;
          InWord:=1;
       end
       else begin
          InWord:=0;
       end;
     end;
  end;
end;
```

4.1.2 Функция Get_Word_From_String(…)

Получить слово из строки Sx по номеру NumWord.

UnChar − множество разделителей.

NumWord – номер слова (начиная с единицы), которое нужно получить

YesAllTrim_Before − если TRUE, то перед обработкой удалить из строки левые и правые пробелы.

```
function Get_Word_From_String(
                               Sx : String;
                               NumWord : integer;
                               UnChar: String=' ';
                               YesAllTrim_Before: boolean=true
                              ) : String;
Var
  InWord    : byte;
  NumWordCur, i : integer;
begin
  Result:='';
  if YesAllTrim_Before then Sx:=trim(Sx);
  if length(Sx)>0 then begin
     if NumWord>0 then begin
        InWord := 0;
        NumWordCur:=0;
        i:=0;
        while i<length(Sx) do
         begin
          i:=i+1;
          if POS(Sx[i],UnChar)<=0 then begin
             if InWord<=0 then NumWordCur:=NumWordCur+1;
             InWord:=1; //Мы внутри слова
             if NumWordCur=NumWord then begin
                Result:=Result+Sx[i];
             end;
          end
          else begin
             InWord:=0;
             if NumWordCur=NumWord then begin
                i:=length(Sx)+1;
             end;
          end;
        end;
     end;
  end;
end;
```

4.2 Приложение-2. Новые (добавляемые в FS) процедуры и функции

4.2.1 Процедуры Экран_Курсор(…); Screen_Cursor(…)

```
procedure Экран_Курсор(sCursor:string='crDefault');
procedure Screen_Cursor(sCursor:string='crDefault');
```

Входные параметры:

`sCursor:string` – тип курсора на экране монитора.

Допустимые значения:

`crDefault;`
`crHourGlass;`
`crSQLWait;`
`crAppStart;`
`crHandPoint;`
`crArrow;`
`crNo.`

«Наша» процедура, где реализован соответствующий функционал:

```
procedure Screen_Cursor_Set(sCurs:string='crDefault');
Var
  Yes:boolean;
begin
  sCurs:=trim(sCurs);
  Yes:=false;
  if sCurs='' then sCurs:='crDefault';
  if AnsiUpperCase(sCurs)=AnsiUpperCase('crDefault') then  begin
     Yes:=true;
     Screen.Cursor:=crDefault;
  end;
  if AnsiUpperCase(sCurs)=AnsiUpperCase('crHourGlass') then  begin
     Yes:=true;
     Screen.Cursor:=crHourGlass;
  end;
  if AnsiUpperCase(sCurs)=AnsiUpperCase('crSQLWait') then  begin
     Yes:=true;
     Screen.Cursor:=crSQLWait;
  end;
  if AnsiUpperCase(sCurs)=AnsiUpperCase('crAppStart') then  begin
     Yes:=true;
     Screen.Cursor:=crAppStart;
  end;
  if AnsiUpperCase(sCurs)=AnsiUpperCase('crHandPoint') then  begin
     Yes:=true;
     Screen.Cursor:=crHandPoint;
  end;
  if AnsiUpperCase(sCurs)=AnsiUpperCase('crArrow') then  begin
     Yes:=true;
     Screen.Cursor:=crArrow;
  end;
  if AnsiUpperCase(sCurs)=AnsiUpperCase('crNo') then  begin
     Yes:=true;
     Screen.Cursor:=crNo;
  end;
  if not Yes then begin
     Screen.Cursor:=crDefault;
  end;
end;
```

4.2.2 Функции Файлы_Список_Получить(…); Files_List_Get(…)

```
function Файлы_Список_Получить(List: TStrings; FullNameMask:
string='*.*'; YesFileNamesOnly: boolean=true):integer;
function Files_List_to_Get(List: TStrings; FullNameMask: string='*.*';
YesFileNamesOnly: boolean=true):integer;
```

Входные параметры:

`List: TStrings` – список файлов;

`FullNameMask: string` – маска файлов (включая полный путь к соответствующей папке;

`YesFileNamesOnly: boolean` – флаг, если =TRUE, то в списке будут только имена файлов (в противном случае – полные имена файлов).

«Наша» функция, где реализован соответствующий функционал:

```
function Files_ToStrings(List: TStrings;
                         FullNameMask:string = '*.*';
                         YesFileNamesOnly:boolean=true
                        ):integer;
Var
 sDir,sMask,NameF: string;
 DirInfo: TSearchRec;
 Err: integer;
begin
  Result := 0;
  if Assigned(List) then begin
     List.Clear;
     FullNameMask := trim(FullNameMask);
     if FullNameMask<>'' then begin
        sDir:=trim(ExtractFilePath(FullNameMask));
        if DirectoryExists(sDir) then begin
           sMask := trim(ExtractFileName(FullNameMask));
           if sMask='' then sMask := '*.*';
           Err := 0;
           Err := FindFirst(sDir + sMask, { faArchive } faAnyFile, DirInfo);
           while Err = 0 do
            begin
              NameF := DirInfo.Name;
              if FileExists(sDir + NameF) then begin
                 if YesFileNamesOnly then begin
                    List.Add(NameF);
                 end
                 else begin
                    List.Add(sDir+NameF);
                 end;
              end;
              Err := FindNext(DirInfo);
              Application.ProcessMessages;
           end;
           FindClose(DirInfo);
           Result := List.Count;
        end;
     end;
  end;
end;
```

4.3 Приложение-3. К вопросу о локализации программных объектов на уровне FS-скрипта

Возможности библиотеки FastScript (в контексте локализации программных объектов на уровне FS-скриптов) позволяют (в том числе) использовать и китайские иероглифы.

На рисунке ниже приведен иллюстрирующий пример.

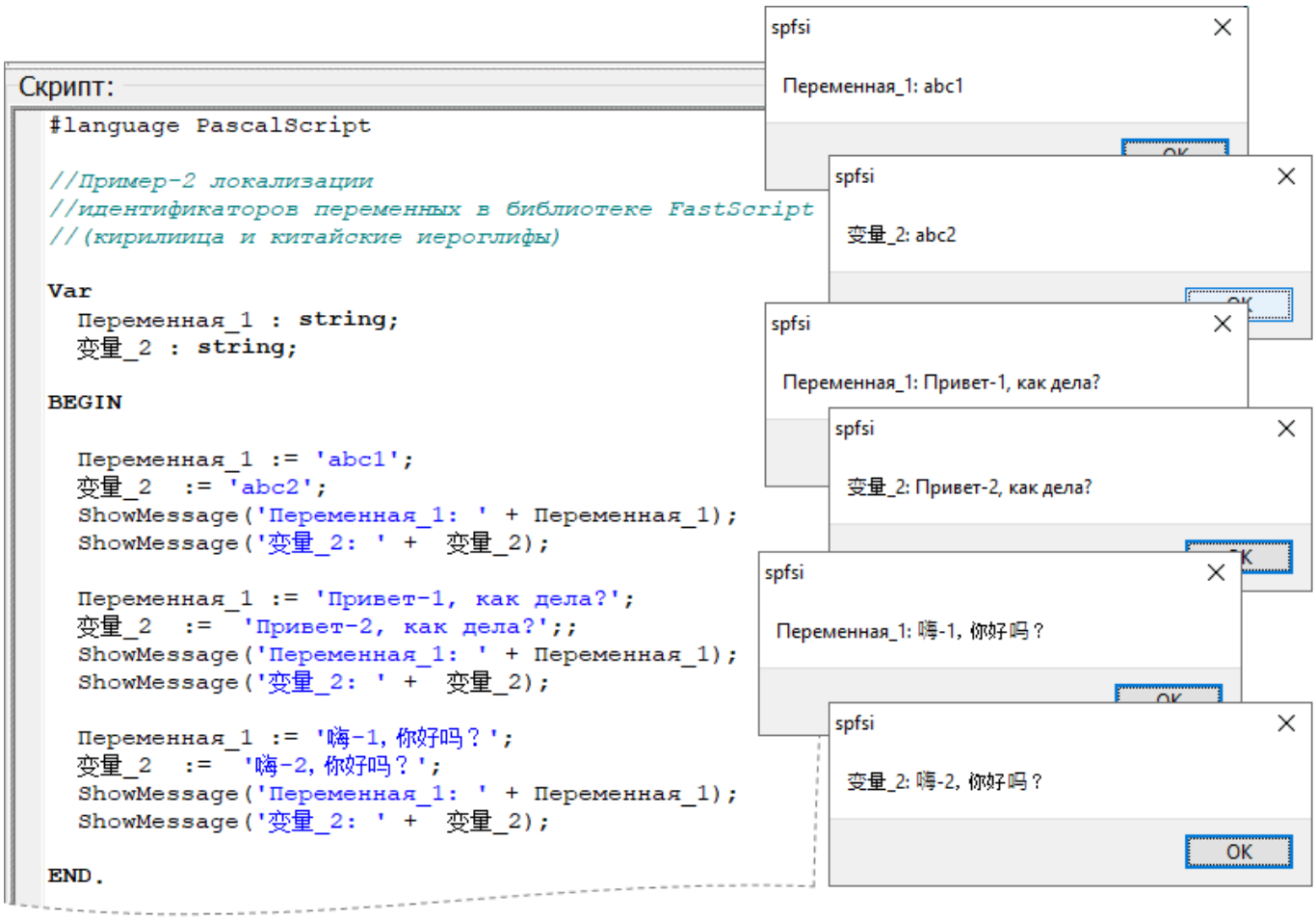


Рисунок 8 – Пример, иллюстрирующий использование кириллицы и китайских иероглифов для идентификации программных объектов на уровне FS-скрипта

Тексты FS-скриптов могут сохраняться как в файлах, так и в таблицах БД.

В обоих вариантах есть нюансы.

В разделах ниже рассмотрены оба этих варианта с практической точки зрения.

Примечание – к этой статье прилагаются исходные тексты соответствующего иллюстрирующего примера.

4.3.1 Хранение текстов FS-скриптов в файлах

Для корректного сохранения FS-скрипта в файл (в кодировке **UTF-8**) и последующего чтения из файла можно использовать следующие функции:

```
//Сохранить FS-скрипт в файл
function FS_SaveToFile(fnScr:string;
                       ListScr: TStrings;
                       ItsUTF8:boolean=false
                       ):boolean;

//Загрузить FS-скрипт из файла
function FS_LoadFromFile(fnScr:string;
                         ListScr: TStrings;
                         ItsUTF8:boolean=false
                         ):boolean;
```

Вызов функций может, например, быть таким:

Сохранить FS-скрипт в файл (UTF-8):

```
FS_SaveToFile('F:\FS\Example1.pas', fsSyntaxMemo1.Lines, true);
```

Загрузить FS-скрипт из файла (UTF-8):

```
FS_LoadFromFile('F:\FS\Example1.pas', fsSyntaxMemo1.Lines, true);
```

Исходные тексты указанных выше функций (а также, тексты вспомогательных функций) и примеры применения приведены в подразделах ниже.

4.3.1.1 Сохранение FS-скрипта в файл

```
function FS_SaveToFile(fnScr:string;
                       ListScr: TStrings;
                       ItsUTF8:boolean=false
                       ):boolean;
//Сохранить FS-скрипт в файл
{
   fnScr - имя файла
   ListScr - текст скрипта
   ItsUTF8 - флаг (если =TRUE, то сохранить в UTF-8)
}
begin
  Result:=false;
  fnScr:=trim(fnScr);
  if fnScr<>'' then begin
     if Assigned(ListScr) then begin
        if ListScr.Count>0 then begin
           Result:=true;
           if not ItsUTF8 then begin
              ListScr.SaveToFile(fnScr);
           end
           else begin
              TStrings_SaveToFile(fnScr, ListScr, 'UTF-8');
           end;
        end;
     end;
  end;
end;
```

4.3.1.2 Загрузка FS-скрипта из файла

```
function FS_LoadFromFile(fnScr:string;
                         ListScr: TStrings;
                         ItsUTF8:boolean=false
                         ):boolean;
//Загрузить FS-скрипт из файла
{
   fnScr - имя файла
   ListScr - текст скрипта
   ItsUTF8 - флаг (если =TRUE, то сохранить в UTF-8)
}
begin
  Result:=false;
  if Assigned(ListScr) then begin
     ListScr.Clear;
     fnScr:=trim(fnScr);
     if FileExists(fnScr) then begin
        Result:=true;
        if not ItsUTF8 then begin
           ListScr.LoadFromFile(fnScr);
        end
        else begin
           TStrings_LoadFromFile(fnScr, ListScr, 'UTF-8');
        end;
     end;
  end;
end;
```

4.3.1.3 Вспомогательные функции

4.3.1.3.1 Функция EncodingName_to_TEncoding

```
function EncodingName_to_TEncoding(sName:string) : TEncoding;
//Конвертация идентификатора "кодовой страницы" из строки в TEncoding
begin
  Result:=TEncoding.GetEncoding(1251);
  sName:=trim(sName);
  if length(sName)>0 then begin
     sName:=UpperCase(sName);
     if sName='WINDOWS-1251' then begin
        Result:=TEncoding.GetEncoding(1251);
     end;
     if sName='UTF-8' then begin
        Result:=TEncoding.UTF8;
     end;
     if sName='DOS-866' then begin
        Result:=TEncoding.GetEncoding(866);
     end;
     if sName='KOI8R' then begin
        Result:=TEncoding.GetEncoding(20866);
     end;
     if sName='UNICODE' then begin
        Result:=TEncoding.Unicode;
     end;
     if sName='UTF-7' then begin
        Result:=TEncoding.UTF7;
     end;
  end;
end;
```

4.3.1.3.2 Функция TStrings_SaveToFile

```
    //Сохранить TStrings в текстовый файл (в заданной кодировке)
    function TStrings_SaveToFile(
                                fn:String;
                                List:TStrings;
                                Encoding:TEncoding
                                ):boolean; overload;

    //Сохранить TStrings в текстовый файл (в заданной кодировке)
    function TStrings_SaveToFile(
                                fn:String;
                                List:TStrings;
                                sEncoding:string='UTF-8'
                                ):boolean; overload;
```

```
function TStrings_SaveToFile(
                             fn:String;
                             List:TStrings;
                             Encoding:TEncoding
                             ):boolean;
//Сохранить TStrings в текстовый файл (в заданной кодировке)
begin
  Result:=false;
  fn:=trim(fn);
  if length(fn)>0 then begin
     if Assigned(List) then begin
        TRY
          List.SaveToFile(fn,Encoding);
          Result:=true;
        FINALLY
        END;
     end;
  end;
end;

function TStrings_SaveToFile(
                             fn:String;
                             List:TStrings;
                             sEncoding:string='UTF-8'
                             ):boolean;
//Сохранить TStrings в текстовый файл (в заданной кодировке)
Var
  E:TEncoding;
begin
  E:=EncodingName_to_TEncoding(sEncoding);
  TRY
     Result:=TStrings_SaveToFile(fn,List,E);
  FINALLY
  END;
end;
```

4.3.1.3.3 Функция TStrings_LoadFromFile

```
    //Загрузить текстовый файл в TStrings (в заданной кодировке)
    function TStrings_LoadFromFile(
                                  fn:String;
                                  List:TStrings;
                                  Encoding:TEncoding
                                  ):boolean; overload;

    //Загрузить текстовый файл в TStrings (в заданной кодировке)
    function TStrings_LoadFromFile(
                                  fn:String;
                                  List:TStrings;
                                  sEncoding:string='UTF-8'
                                  ):boolean; overload;
```

```
function TStrings_LoadFromFile(
                               fn:String;
                               List:TStrings;
                               Encoding:TEncoding
                               ):boolean;
//Загрузить текстовый файл в TStrings (в заданной кодировке)
begin
  Result:=false;
  if Assigned(List) then begin
     List.Clear;
     fn:=trim(fn);
     if FileExists(fn) then begin
        TRY
          List.LoadFromFile(fn,Encoding);
          Result:=true;
        FINALLY
        END;
     end;
  end;
end;

function TStrings_LoadFromFile(
                               fn:String;
                               List:TStrings;
                               sEncoding:string='UTF-8'
                               ):boolean;
//Загрузить текстовый файл в TStrings (в заданной кодировке)
Var
  E:TEncoding;
begin
  E:=EncodingName_to_TEncoding(sEncoding);
  TRY
    Result:=TStrings_LoadFromFile(fn,List,E);
  FINALLY
  END;
end;
```

4.3.2 Хранение текстов FS-скриптов в таблицах БД

При хранении текстов FS-скриптов в таблицах БД есть некоторые нюансы, которые следует учитывать.

Так, например, СУБД SQLite корректно (по умолчанию) поддерживает UTF-8 при хранении текстов в полях типа TEXT.

А вот с PostgreSQL могут быть проблемы.

Например, если БД создана таким образом:

```
CREATE DATABASE mydb
  WITH OWNER = postgres
       ENCODING = 'UTF8'
       TABLESPACE = pg_default
       LC_COLLATE = 'Russian_Russia.1251'
       LC_CTYPE = 'Russian_Russia.1251'
       CONNECTION LIMIT = -1;
```

Рисунок 9 – БД в PostgreSQL

То при сохранении текста в UTF-8 – будут «кракозябры».

В этом случае, для сохранения и считывания FS-скриптов, можно использовать кодировку в Base64.

Тексты функций см. ниже.

```
Uses
  …,
  NetEncoding,

function Str_to_StrBase64(S: string): string;
//Кодирование в Base64
begin
  Result := TNetEncoding.Base64.Encode(S);
end;

function StrBase64_to_Str(S: string): string;
//Декодирование из Base64
begin
  Result := TNetEncoding.Base64.Decode(S);
end;
```

Вызов функций может, например, быть таким:

```
procedure TfMain.btnSaveToDBClick(Sender: TObject);
begin
  qScr.Delete;
  qScr.Append;
  qScr.Edit;
  qScrfs_script.AsString:=fsSyntaxMemo1.Lines.Text;
  qScrfs_script64.AsString:=Str_to_StrBase64(fsSyntaxMemo1.Lines.Text);
  qScr.Post;
end;
```

```
procedure TfMain.btnLoadFromDB64Click(Sender: TObject);
begin
  Application.ProcessMessages;
  fsSyntaxMemo1.Lines.Clear;
  fsSyntaxMemo1.Lines.Text:=StrBase64_to_Str(qScrfs_script64.AsString);
  fsSyntaxMemo1.UpdateView;
end;
```

Рисунок 10 – Пример вызова функций (для Base64)

4.3.3 Пример применения

В иллюстрирующем примере приведены оба варианта хранения текстов FS-скриптов в кодировке UTF-8.

В качестве БД выбрана СУБД SQLite, где создана таблица следующей структуры (см. рисунок ниже).

```
CREATE TABLE fs_scripts -- список FS-скриптов
(
id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT, -- уникальный ID
fs_script TEXT,                                -- текст FS-скрипта
fs_script64 TEXT                               -- текст FS-скрипта в Base64
);
```

Рисунок 11 – Структура таблицы БД

На рисунке ниже приведен скрин главной формы Приложения (в RunTime), где виден результат применения указанных выше функций.

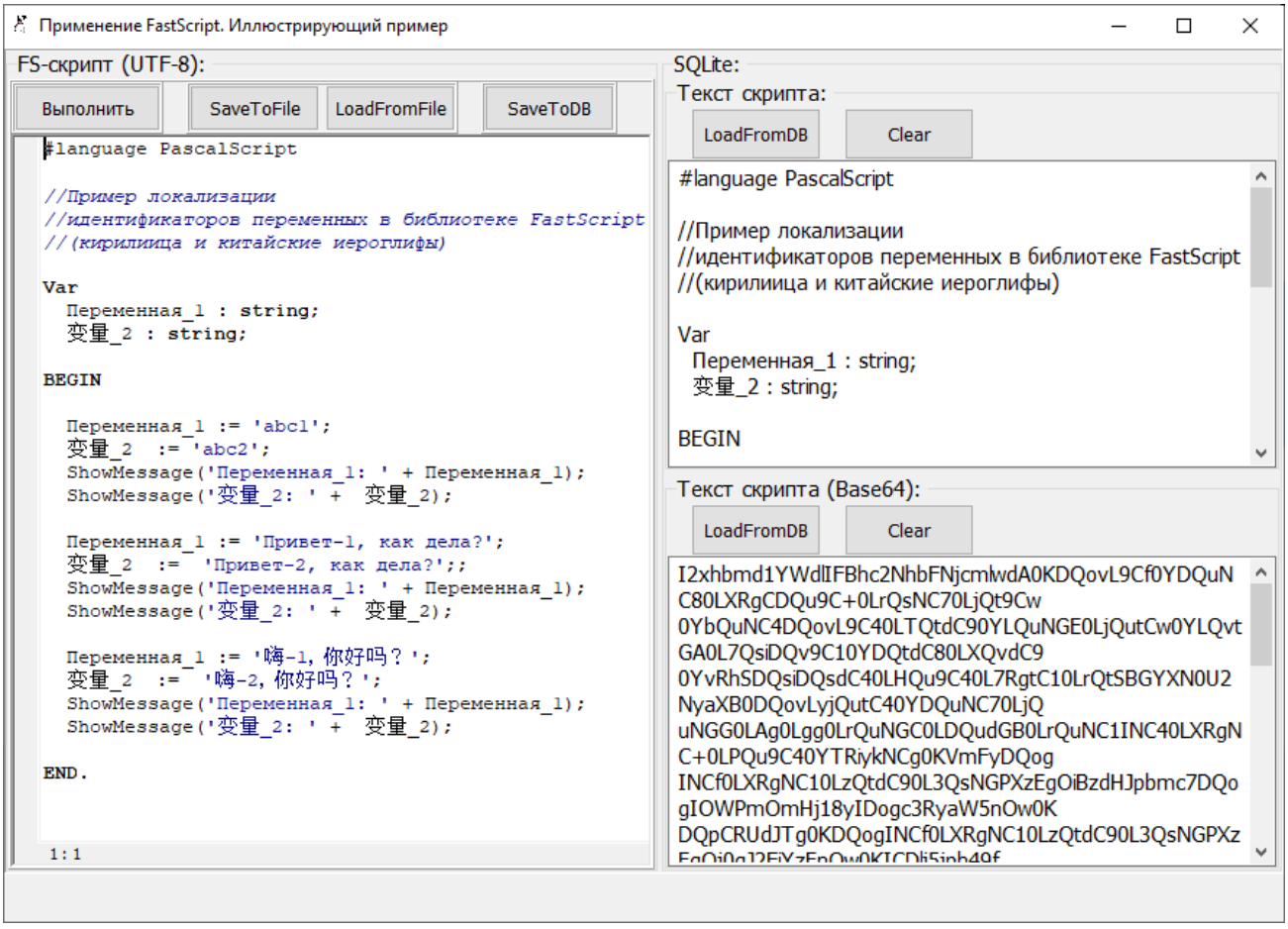


Рисунок 12 – Иллюстрирующий пример. Скрин главной формы Приложения

Здесь все прозрачно, пояснения вряд ли требуются.
Детали – см. в исходниках иллюстрирующего примера.