Министерство образования и науки РФ
ФГБПОУ ВПО Тульский государственный университитет
КАФЕДРА АВТОМАТИКИ И ТЕЛЕМЕХАНИКИ

ИНТЕРФЕЙС ПЕРЕНОСА DRAG-AND-DROP

Лабораторная работа № 7
по курсу «Программирование на ЯВУ»

Вариант № 4

Выполнил: студент группы 220601 _____ Белым А.А.
(подпись)

Проверил: к. ф.-м. н., доцент _____ Сулимова В.В.
(подпись)

Тула 2011

Цель работы

Цель работы заключается в том, чтобы изучить интерфейс переноса Drag&Drop и написать программу с его использованием.

Задание на работу

Реализовать конвертер единиц измерения. На форме расположить список доступных единиц измерения. Из него выбирать нужные единицы измерения путем перетаскивания их мышкой. Результат перевода вместе с исходными данными фиксировать в TMemo, в виде строки, например, 15 сантиметров = 150 миллиметров.

Теоретическая справка

Инструкция пользователю

Программа позволяет конвертировать величины различных единиц измерения между собой.

Выберите класс единиц измерения, после чего перетащите наименования единиц измерения в области, ограниченные рамкой. Введите значение в любое поле ввода и нажмите клавишу <Enter> на клавиатуре или кнопку -"в окне программы.

Инструкция программиста

При создании программы конвертации единиц измерения были объявлены типы:

1. TConvertUnits - запись, описывающая список типов(единиц измерения) величин.

В таблице 1 представлены поля этого типа:

Таблица 1 - Поля типа "список единиц измерения"

имя	тип	предназначение
Category	ShortString	категория
Units	array of ShortString	наименования

2. TUnitsTable=array of TConvertUnits, описывает таблица типов величин.

3. TConvertFunction=function (value:extended):extended; stdcall;

описывает тип функции перевода величин.

4. TConvertLine - запись, которая описывает строку таблицы конвертации. В таблице 2 приведены поля этого типа.

Таблица 2 - Поля типа "строка таблицы конвертации"

имя	тип	предназначение
SrcUnit,DstUnit	ShortString	исходный и требуемый типы
ConvertFunc	TConvertFunction	функция конвертации

5. TConvertTable=array of TConvertLine, описывает таблицу конвертации.

6. UnitsTableExportFunc=function:TUnitsTable; stdcall;

описывает тип функции экспорта таблицы типов из DLL.

7. ConvertTableExportFunc=function:TConvertTable; stdcall;

описывает тип функции экспорта таблицы конвертации из DLL

8. GarbageCollectorFunc=procedure (var units:TUnitsTable;

var convert:TConvertTable); stdcall;

тип функции сборки мусора из DLL

Были объявлены следующие глобальные переменные, описание которых приводится в таблице 3:

Таблица 3 - Глобальные переменные модуля

имя	**тип**	**предназначение**
ConvertTable	TConvertTable	таблица конвертации
UnitsTable	TUnitsTable	таблица единиц измерения
LoadedLibraries	array of Cardina	список подключенных DLL

Далее программа была разбита на следующие подпрограммы:

1. Функция GetConvFunction получает из таблицы ConvTable для единиц измерения SrcUnit(исходный тип) и DstUnit(целевой тип)

функцию конвертации ConvFunc.

Если функция найдена, возвращает True, в противном случае возвращает False.

function GetConvFunction(ConvTable:array of TConvertLine;

SrcUnit,DstUnit:ShortString;

var ConvFunc:TConvertFunction):Boolean;

Параметры функции представлены в таблице 4 :

Таблица 4 - Параметры функции получения функции конвертации

имя	**тип**	**предназначение**
ConvTable	array of TConvertLine	таблица конвертации
SrcUnit,DstUnit	ShortString	исходный тип и целевой тип конвертации
var ConvFunc	TConvertFunction	функция конвертации

Локальные переменные функции представлены в таблице 5 :

Таблица 5 - Локальные переменные функции получения функции конвертации

имя	тип	предназначение
i	LongInt	счетчик для обработки массива

2. Процедура ConnectDLL подключает библиотеку DllName и импортирует таблицы единиц измерений и конвертации.

procedure ConnectDLL(DllName:PChar);

Параметры процедуры представлены в таблице 6 :

Таблица 6 - Параметры процедуры подключения DLL

имя	тип	предназначение
DllName	PChar	путь к библиотеке

Локальные переменные процедуры представлены в таблице 7 :

Таблица 7 - Локальные переменные процедуры подключения DLL

имя	тип	предназначение
Handle	Cardinal	дескриптор библиотеки
NewUnits	TUnitsTable	импортированная таблица типов
NewConvert	TConvertTable	импортированная таблица конвертации
UExport	UnitsTableExportFunc	функция импорта таблицы типов
CExport	ConvertTableExportFunc	функция импорта таблицы конвертации
GCollect	GarbageCollectorFunc	функция сборки мусора
i,j,k,m	LongInt	счетчики для обработки массивов;
l1,l2	LongInt	размеры существующей и импортированной таблиц конвертации
l1u,l2u	LongInt	размер существующего списка типов и импортированного
appended,exist	boolean	флаги наличия импортируемых данных в таблицах

3. Процедура lblFstUnitDragDrop - обработчик "сбрасывания"в поле единиц измерения.

procedure TfrmMain.lblFstUnitDragDrop(Sender, Source: TObject; X, Y: Integer);

Параметры процедуры представлены в таблице 8 :

Таблица 8 - Параметры процедуры-обработчика события"сбрасывания"в поле единиц измерения

имя	тип	предназначение
Sender	TObject	объект-возбудитель события
Sender	TObject	объект-возбудитель события

4. Процедура lblFstUnitDragOver - обработчик "попадания"в поле единиц измерения.

procedure TfrmMain.lblFstUnitDragOver(Sender, Source: TObject; X, Y: Integer;

State: TDragState; var Accept: Boolean);

Параметры процедуры представлены в таблице 9 :

Таблица 9 - Параметры процедуры-обработчика события "попадания"в поле единиц измерения

имя	тип	предназначение
Sender	TObject	объект-приемник и объект-источник DragNDrop
X, Y	Integer	координаты курсора
State	TDragState	состояние операции DragNDrop
Accept	Boolean	принять объект или нет

5. Процедура imgFstTrashDragOver - обработчик "сбрасывания"в мусорную корзину.

procedure TfrmMain.imgFstTrashDragOver(Sender, Source: TObject;

X, Y: Integer;

State: TDragState; var Accept: Boolean);

Параметры процедуры представлены в таблице 10 :

Таблица 10 - Параметры процедуры-обработчика события "сбрасывания"в мусорную корзину

имя	тип	предназначение
Sender	TObject	объект-приемник и объект-источник DragNDrop
X, Y	Integer	координаты курсора
State	TDragState	состояние операции DragNDrop
Accept	Boolean	принять объект или нет

6. Процедура imgFstTrashDragDrop - обработчик "попадания"в мусорную корзину.

procedure TfrmMain.imgFstTrashDragDrop(Sender, Source: TObject;

X, Y: Integer);

Параметры процедуры представлены в таблице 11 :

Таблица 11 - Параметры процедуры-обработчика события "попадания"в мусорную корзину

имя	тип	предназначение
Sender	TObject	объект-приемник и объект-источник DragNDrop
X, Y	Integer	координаты курсора

7. Процедура edtFstValKeyPress - обработчик нажатий клавиатуры в поле ввода величин.

procedure TfrmMain.edtFstValKeyPress(Sender: TObject; var Key: Char);

Параметры процедуры представлены в таблице 12 :

Таблица 12 - Параметры процедуры-обработчика события нажатий клавиатуры в поле ввода величин

имя	тип	предназначение
Sender	TObject	объект-возбудитель события
Key	Char	символ нажатой клавиши

8. Процедура cbxCategorySelect - обработчик выбора категорий типов.

procedure TfrmMain.cbxCategorySelect(Sender: TObject);

Параметры процедуры представлены в таблице 13 :

Таблица 13 - Параметры процедуры-обработчика события нажатия выбора категорий типов

имя	тип	предназначение
Sender	TObject	объект-возбудитель события

9. Процедура FormCreate - обработчик создания формы.

procedure TfrmMain.FormCreate(Sender: TObject);

Параметры процедуры представлены в таблице 14 :

Таблица 14 - Параметры процедуры-обработчика события создания формы

имя	тип	предназначение
Sender	TObject	объект-возбудитель события

10. Процедура FormDestroy - обработчик разрушения формы.

procedure TfrmMain.FormDestroy(Sender: TObject);

Параметры процедуры представлены в таблице 15 :

Таблица 15 - Параметры процедуры-обработчика события разрушения формы

имя	тип	предназначение
Sender	TObject	объект-возбудитель события

11. Процедура btnEquClick - обработчик нажатия кнопки "равно".

procedure TfrmMain.btnEquClick(Sender: TObject);

Параметры процедуры представлены в таблице 16 :

Таблица 16 - Параметры процедуры-обработчика события нажатия кнопки "равно"

имя	тип	предназначение
Sender	TObject	объект-возбудитель события

Кроме того, подключаемые DLL должны экспортировать функции:

1. Процедура CollectGarbage освобождает память из-под таблицы единиц измерения units и таблицы конвертации convert.

procedure CollectGarbage(var units:TUnitsTable;

var convert:TConvertTable); stdcall;

2. Функция ExportUnitsTable возвращает экспортируемую библиотекой таблицу единиц измерения.

function ExportUnitsTable:TUnitsTable; stdcall;

3. Функция ExportConvertTable возвращает экспортируемую библиотекой таблицу единиц конвертации.

function ExportConvertTable:TConvertTable; stdcall;

Текст программы

Далее приведен текст программы конвертации единиц измерения, написанной на языке Delphi 7.

```
unit UnitMain;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ExtCtrls, StdCtrls, ComCtrls;

type
  TConvertUnits=record //список типов величин
    Category:ShortString; //категория
    Units:array of ShortString; //наименования
  end;
  TUnitsTable=array of TConvertUnits;//таблица типов величин
  //тип функции перевода величин
  TConvertFunction=function (value:extended):extended; stdcall;
  //строка таблицы конвертации
  TConvertLine=record
    SrcUnit,DstUnit:ShortString;//исходный и требуемый типы
    ConvertFunc:TConvertFunction; //функция конвертации
  end;
  TConvertTable=array of TConvertLine; //таблица конвертации

  //тип функции экспорта таблицы типов из DLL
  UnitsTableExportFunc=function:TUnitsTable; stdcall;
  //тип функции экспорта таблицы конвертации из DLL
  ConvertTableExportFunc=function:TConvertTable;  stdcall;
  //тип функции сборки мусора из DLL
  GarbageCollectorFunc=procedure (var units:TUnitsTable;
              var convert:TConvertTable); stdcall;

  TfrmMain = class(TForm) //основная форма
    edtFstVal: TEdit;//первая величина
    edtSndVal: TEdit;//вторая величина
    memLog: TMemo;   //вывод результатов
    lblFstUnit: TLabel;//тип первой величины
    lblSndUnit: TLabel;//тип второй величины
    imgFstTrash: TImage;//мусорная корзина 1
    imgSndTrash: TImage;//мусорная корзина 2
    cbxCategory: TComboBox;//выбор категории типов
    sbxUnits: TScrollBox;//единицы измерения
    lblInfo: TLabel;     //информация о программе
    bvlFstBevel: TBevel; //рамка вокруг единицы измерения
    bvlSndBevel: TBevel; //рамка вокруг единицы измерения
    btnEqu: TButton;     //кнопка "равно"

    //обработчик "сбрасывания" в поле единиц измерения
    procedure lblFstUnitDragDrop(Sender, Source: TObject; X, Y: Integer);
    //обработчик "попадания" в поле единиц измерения
    procedure lblFstUnitDragOver(Sender, Source: TObject; X, Y: Integer;
      State: TDragState; var Accept: Boolean);
    //обработчик "сбрасывания" в мусорную корзину
    procedure imgFstTrashDragOver(Sender, Source: TObject; X, Y: Integer;
      State: TDragState; var Accept: Boolean);
    ////обработчик "попадания" в мусорную корзину
    procedure imgFstTrashDragDrop(Sender, Source: TObject; X, Y: Integer);
    //обработчик нажатий клавиатуры в поле ввода величин
    procedure edtFstValKeyPress(Sender: TObject; var Key: Char);
    //обработчик выбора категорий типов
```

```
    procedure cbxCategorySelect(Sender: TObject);
    //обработчик создания формы
    procedure FormCreate(Sender: TObject);
    //обработчик разрушения формы
    procedure FormDestroy(Sender: TObject);
    //обработчик нажатия кнопки "равно"
    procedure btnEquClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  frmMain: TfrmMain; //основная форма
  ConvertTable:tconverttable; //таблица конвертации
  UnitsTable:TUnitsTable; //таблица единиц измерения
  LoadedLibraries:array of Cardinal; //список подключенных DLL
implementation

(*
Функция GetConvFunction получает из таблицы ConvTable для единиц измерения
SrcUnit(исходный тип) и DstUnit(целевой тип) функцию конвертации ConvFunc.
Если функция найдена, возвращает True, в противном случае возвращает False.
Параметры
ConvTable:array of TConvertLine — таблица конвертации
SrcUnit,DstUnit:ShortString — исходный тип и целевой тип конвертации
var ConvFunc:TConvertFunction — функция конвертации
Локальные переменные
i:LongInt — счетчик для обработки массива
*)
function GetConvFunction(ConvTable:array of TConvertLine;
          SrcUnit,DstUnit:ShortString;
            var ConvFunc:TConvertFunction):Boolean;
var i:LongInt;
begin
   Result:=False;
   for i:=low(ConvTable) to HIGH(ConvTable) do
    if (SrcUnit=ConvTable[i].SrcUnit) and (DstUnit=ConvTable[i].DstUnit) then
    begin
      ConvFunc:=ConvTable[i].ConvertFunc;
      Result:=True;
    end;
end;
{$R *.dfm}

procedure TfrmMain.lblFstUnitDragDrop(Sender, Source: TObject; X, Y: Integer);
var DragLabel:TLabel;temp:string;
begin
  try
    if (TLabel(Sender).Name='lblFstUnit') and (TLabel(Source).Name='lblSndUnit')
          or
       (TLabel(Sender).Name='lblSndUnit') and (TLabel(Source).Name='lblFstUnit')
    then begin
      temp:=TLabel(Sender).Caption;
      TLabel(Sender).Caption:=TLabel(Source).Caption;
      TLabel(Source).Caption:=temp;
    end
    else begin
      DragLabel:=TLabel(Source);
      TLabel(Sender).Caption:=DragLabel.Caption;
    end;
  except;
  end;
end;
```

```
procedure TfrmMain.lblFstUnitDragOver(Sender, Source: TObject; X, Y: Integer;
      State: TDragState; var Accept: Boolean);
begin
  Accept:=Source is TLabel;
end;
procedure TfrmMain.imgFstTrashDragOver(Sender, Source: TObject; X, Y: Integer;
  State: TDragState; var Accept: Boolean);
begin
  try
    if (TLabel(Source).Name='lblFstUnit') or (TLabel(Source).Name='lblSndUnit') then
      Accept:=True
    else Accept:=False;
  except else Accept:=False;
  end;
end;

procedure TfrmMain.imgFstTrashDragDrop(Sender, Source: TObject; X, Y: Integer);
begin
  TLabel(Source).Caption:='';
end;

procedure TfrmMain.edtFstValKeyPress(Sender: TObject; var Key: Char);
var Master,Slave:TEdit; MUnit,SUnit:String; ConvFunc:TConvertFunction;
begin
  If Key=#13 then begin
    Master:=TEdit(Sender);
    if Master.Name='edtFstVal' then begin
       Slave:=edtSndVal; MUnit:=lblFstUnit.Caption; SUnit:=lblSndUnit.Caption;
    end else if  Master.Name='edtSndVal' then begin
       Slave:=edtFstVal; MUnit:=lblSndUnit.Caption; SUnit:=lblFstUnit.Caption;
    end else raise EComponentError.Create('Unkown component!');
    try
      strtofloat(Master.Text);
      if (MUnit<>'') and (SUnit<>'') then begin
        if MUnit=SUnit then begin
          slave.Text:=Master.Text;
          memLog.Lines.Add(edtFstVal.Text+lblFstUnit.Caption+' = '+
                                  edtSndVal.Text+lblSndUnit.Caption);
        end else
        if GetConvFunction(ConvertTable,MUnit,SUnit,ConvFunc) then begin
          Slave.Text:=floattostr(ConvFunc(strtofloat(Master.Text)));
          memLog.Lines.Add(edtFstVal.Text+lblFstUnit.Caption+' = '+
                          edtSndVal.Text+lblSndUnit.Caption);
        end else MessageDlg('Неизвестно, как перевести '+MUnit+' в '+SUnit+'.',
                                            mtInformation,[mbok],0);
      end;
    except on EConvertError do
        MessageDlg('Неправильное значение!',
                                            mtInformation,[mbok],0);
  end;
end;

procedure TfrmMain.cbxCategorySelect(Sender: TObject);
var i,j:Cardinal; newlabel:TLabel;
begin
   for i:=0 to length(UnitsTable)-1 do
    if UnitsTable[i].Category=cbxCategory.Text then begin
      while sbxUnits.ControlCount<>0 do
        sbxUnits.Controls[0].Destroy;
      for j:= low(UnitsTable[i].Units) to high(UnitsTable[i].Units) do
  begin
     newlabel:=TLabel.Create(sbxUnits);
     newlabel.AutoSize:=true;
     Newlabel.Align:=alLeft;
     NewLabel.DragMode:=dmAutomatic;
```

```
      newlabel.Font.Size:=14;
      newlabel.Alignment:=taCenter;
      newlabel.Layout:=tlCenter;
      newlabel.Caption:=UnitsTable[i].Units[j];
      newlabel.Parent:=sbxUnits;

      newlabel:=TLabel.Create(sbxUnits);
      newlabel.AutoSize:=true;
      Newlabel.Align:=alLeft;
      newlabel.Font.Size:=14;
      newlabel.Alignment:=taCenter;
      newlabel.Layout:=tlCenter;
      newlabel.Caption:='  ';
      newlabel.Parent:=sbxUnits;
  end;
  end;
end;

(*
Процедура ConnectDLL подключает библиотеку DllName и
импортирует таблицы единиц измерений и конвертации.
Параметры
DllName:PChar — путь к библиотеке
Локальные переменные
Handle:Cardinal — дескриптор библиотеки
NewUnits:TUnitsTable — импортированная таблица типов
NewConvert:TConvertTable — импортированная таблица конвертации
UExport:UnitsTableExportFunc — функция импорта таблицы типов
CExport:ConvertTableExportFunc — функция импорта таблицы конвертации
GCollect:GarbageCollectorFunc — функция сборки мусора
i,j,k,m:LongInt — счетчики для обработки массивов;
l1,l2:LongInt — размеры существующей и импортированной таблиц конвертации;
l1u,l2u:LongInt — размер существующего списка типов и импортированного;
appended,exist:boolean — флаги наличия импортируемых данных в таблицах;
*)
procedure ConnectDLL(DllName:PChar);
var Handle:Cardinal;
var NewUnits:TUnitsTable; NewConvert:TConvertTable;
var UExport:UnitsTableExportFunc;
  CExport:ConvertTableExportFunc;
  GCollect:GarbageCollectorFunc;
var i,j,k,m,l1,l2,l1u,l2u:LongInt;
    appended,exist:boolean;
begin
newUnits:=nil;newConvert:=nil;
Handle:=LoadLibrary(DllName);
if Handle = 0 then
  begin
    raise EAccessViolation.Create('Ошибка подключения библиотеки '+DllName) ;
  end;
@UExport:=GetProcAddress(Handle,'ExportUnitsTable');
if @UExport=nil then
     raise EAccessViolation.Create('Ошибка подключения библиотеки '+DllName) ;
NewUnits:=UExport;

l2:=length(NewUnits);
for i:= 0 to l2-1 do begin
  l1:=length(UnitsTable); appended:=false;
  for j:=0 to l1-1 do
    if UnitsTable[j].Category=NewUnits[i].Category then begin


      l2u:=length(NewUnits[i].Units);
      for k:= 0 to l2u-1 do begin
          l1u:=length(UnitsTable[j].Units);  exist:=false;
```

```
          for m:=0 to l1u-1 do
            if UnitsTable[j].Units[M]=NewUnits[i].Units[k] then
               exist:=true;
          if not exist then begin
            inc(l1u);setlength(UnitsTable[j].Units,l1u);
            UnitsTable[j].Units[l1u-1]:=NewUnits[i].Units[k];
          end;
       end;
       appended:=true;
     end;
  if not appended then begin
    inc(l1);setlength(UnitsTable,l1);
    UnitsTable[l1-1].Category:=NewUnits[i].Category;
    setlength(UnitsTable[l1-1].Units,length(NewUnits[i].Units));
    For j:=0 to length(NewUnits[i].Units)-1 do
    UnitsTable[l1-1].Units[j]:=NewUnits[i].Units[j];
  end;
end;

@CExport:=GetProcAddress(Handle,'ExportConvertTable');
if @CExport=nil  then
     raise EAccessViolation.Create('Ошибка подключения библиотеки '+DllName) ;
NewConvert:=CExport;


l2:=length(NewConvert);
for i:=0 to l2-1 do begin
  l1:=length(ConvertTable); exist:=false;
  for j:=0 to l1-1 do
    if (ConvertTable[j].DstUnit=NewConvert[i].DstUnit)
       and (ConvertTable[j].SrcUnit=NewConvert[i].SrcUnit) then
        exist:=true;
  if not exist then begin
    inc(l1);setlength(ConvertTable,l1);
    ConvertTable[l1-1].SrcUnit:=NewConvert[i].SrcUnit;
    ConvertTable[l1-1].DstUnit:=NewConvert[i].DstUnit;
    ConvertTable[l1-1].ConvertFunc:=NewConvert[i].ConvertFunc;
  end;
end;
@GCollect:=GetProcAddress(Handle,'CollectGarbage');
if @GCollect=nil then
     raise EAccessViolation.Create('Ошибка подключения библиотеки '+DllName) ;
for i:= 0 to length(newunits)-1 do
      newunits[i].Units:=NIL;
GCollect(NewUnits,NewConvert);
SetLength(LoadedLibraries,length(LoadedLibraries)+1);
LoadedLibraries[0]:=Handle;
end;

procedure TfrmMain.FormCreate(Sender: TObject);
var i:Longint;
  search:TSearchRec;
begin
if FindFirst('*.dll',0,search)=0 then begin
  try
    ConnectDLL(PChar(search.name));
  except else MessageDlg('Попытка подключения библиотеки '+
                      search.name+' неудачна!',mtWarning,[mbok],0);
  end;
  While FindNext(search)=0 do begin
    try
      ConnectDLL(PChar(search.name));
    except  else MessageDlg('Попытка подключения библиотеки '+
                             search.name+' неудачна!',mtWarning,[mbok],0);
    end;
```

```
  end;
  FindClose(search);
end;
if length(UnitsTable)<>0 then begin
for i:=0 to length(UnitsTable)-1 do
  cbxCategory.Items.Add(UnitsTable[i].Category);
cbxCategory.ItemIndex:=0;
cbxCategorySelect(cbxCategory);
end;
end;

procedure TfrmMain.FormDestroy(Sender: TObject);
var i:longint;
begin
  for i:=0 to length(UnitsTable)-1 do
    UnitsTable[i].Units:=nil;

  UnitsTable:=nil;
  while sbxUnits.ControlCount<>0 do
        sbxUnits.Controls[0].Destroy;
  SetLength(ConvertTable,0);
  for i:= 0 to length(LoadedLibraries)-1 do
    FreeLibrary(LoadedLibraries[i]);
  LoadedLibraries:=NIL;
end;

procedure TfrmMain.btnEquClick(Sender: TObject);
var PseudoKey:Char;
begin
  PseudoKey:=#13;
  edtFstValKeyPress(edtFstVal,PseudoKey);
end;

end.
```

Ниже приведен текст библиотеки для конвертации единиц температуры.

```
library thermal;

uses
  SysUtils,
  Classes,
  Dialogs;

{$R *.res}
type
  TConvertUnits=record //список типов величин
    Category:ShortString; //категория
    Units:array of ShortString; //наименования
  end;
  TUnitsTable=array of TConvertUnits;//таблица типов величин
  //тип функции перевода величин
  TConvertFunction=function (value:extended):extended; stdcall;
  //строка таблицы конвертации
  TConvertLine=record
    SrcUnit,DstUnit:ShortString;//исходный и требуемый типы
    ConvertFunc:TConvertFunction; //функция конвертации
  end;
  TConvertTable=array of TConvertLine; //таблица конвертации

  //тип функции экспорта таблицы типов из DLL
  UnitsTableExportFunc=function:TUnitsTable; stdcall;
  //тип функции экспорта таблицы конвертации из DLL
  ConvertTableExportFunc=function:TConvertTable;  stdcall;
  //тип функции сборки мусора из DLL
```

```
  GarbageCollectorFunc=procedure (var units:TUnitsTable;
                        var convert:TConvertTable); stdcall;

(*
Процедура CollectGarbage освобождает память из—под таблицы
единиц измерения units и таблицы конвертации convert.
*)
procedure CollectGarbage(var units:TUnitsTable;
                var convert:TConvertTable); stdcall;
begin
   units:=NIL;
   convert:=NIL;
end;

(*
Функция ExportUnitsTable возвращает экспортируемую библиотекой
таблицу единиц измерения.
*)
function ExportUnitsTable:TUnitsTable; stdcall;
begin
  setlength(Result,1);
  Result[0].Category:='Температура';
  setlength(Result[0].Units,3);
  Result[0].Units[0]:='C';
  Result[0].Units[1]:='F';
  Result[0].Units[2]:='K';
end;

function CtoK(value:extended):extended;  stdcall;
begin
  Result:=value+273;
end;
function KtoC(value:extended):extended; stdcall;
begin
  Result:=value—273;
end;

function CtoF(value:extended):extended; stdcall;
begin
  Result:=value*9/5+32;
end;
function FtoC(value:extended):extended; stdcall;
begin
  Result:=(value—32)*5/9;
end;

function FtoK(value:extended):extended; stdcall;
begin
  Result:=CtoK(FToC(Value));
end;
function KtoF(value:extended):extended; stdcall;
begin
  Result:=CtoF(KtoC(value));
end;

(*
Функция ExportConvertTable возвращает экспортируемую библиотекой
таблицу единиц конвертации.
*)
function ExportConvertTable:TConvertTable; stdcall;
begin
  setlength(Result,6);
  Result[0].SrcUnit:='C';
  Result[0].DstUnit:='K';
  Result[0].ConvertFunc:=CtoK;
```

```
  Result[1].SrcUnit:='K';
  Result[1].DstUnit:='C';
  Result[1].ConvertFunc:=KtoC;

  Result[2].SrcUnit:='K';
  Result[2].DstUnit:='F';
  Result[2].ConvertFunc:=KtoF;

  Result[3].SrcUnit:='F';
  Result[3].DstUnit:='C';
  Result[3].ConvertFunc:=FtoC;

  Result[4].SrcUnit:='C';
  Result[4].DstUnit:='F';
  Result[4].ConvertFunc:=CtoF;

  Result[5].SrcUnit:='F';
  Result[5].DstUnit:='K';
  Result[5].ConvertFunc:=FtoK;
end;
exports CollectGarbage,ExportUnitsTable,ExportConvertTable;
begin
end.
```

Ниже приведен текст библиотеки для конвертации единиц давления.

```
library pressure;

uses
  SysUtils,
  Classes,
  Dialogs;

{$R *.res}
type
  TConvertUnits=record //список типов величин
    Category:ShortString; //категория
    Units:array of ShortString; //наименования
  end;
  TUnitsTable=array of TConvertUnits;//таблица типов величин
  //тип функции перевода величин
  TConvertFunction=function (value:extended):extended; stdcall;
  //строка таблицы конвертации
  TConvertLine=record
    SrcUnit,DstUnit:ShortString;//исходный и требуемый типы
    ConvertFunc:TConvertFunction; //функция конвертации
  end;
  TConvertTable=array of TConvertLine; //таблица конвертации

  //тип функции экспорта таблицы типов из DLL
  UnitsTableExportFunc=function:TUnitsTable; stdcall;
  //тип функции экспорта таблицы конвертации из DLL
  ConvertTableExportFunc=function:TConvertTable;  stdcall;
  //тип функции сборки мусора из DLL
  GarbageCollectorFunc=procedure (var units:TUnitsTable;
                       var convert:TConvertTable); stdcall;

(*
Процедура CollectGarbage освобождает память из—под таблицы
единиц измерения units и таблицы конвертации convert.
*)
procedure CollectGarbage(var units:TUnitsTable;
                     var convert:TConvertTable); stdcall;
begin
```

```
   units:=NIL;
   convert:=NIL;
end;

(*
Функция ExportUnitsTable возвращает экспортируемую библиотекой
таблицу единиц измерения.
*)
function ExportUnitsTable:TUnitsTable; stdcall;
begin
  setlength(Result,1);
  Result[0].Category:='Давление';
  setlength(Result[0].Units,2);
  Result[0].Units[0]:='Па';
  Result[0].Units[1]:='мм.рт.ст';
end;

function PatommHg(value:extended):extended;  stdcall;
begin
  Result:=value/133.322;
end;
function mmHgtoPa(value:extended):extended; stdcall;
begin
  Result:=value*133.322;
end;

(*
Функция ExportConvertTable возвращает экспортируемую библиотекой
таблицу единиц конвертации.
*)
function ExportConvertTable:TConvertTable; stdcall;
begin
  setlength(Result,2);
  Result[0].SrcUnit:='Па';
  Result[0].DstUnit:='мм.рт.ст';
  Result[0].ConvertFunc:=PatommHg;

  Result[1].SrcUnit:='мм.рт.ст';
  Result[1].DstUnit:='Па';
  Result[1].ConvertFunc:=mmHgtoPa;
end;
exports CollectGarbage,ExportUnitsTable,ExportConvertTable;
begin
end.
```

Ниже приведен текст библиотеки для конвертации единиц энергии.

```
library energy;

uses
  SysUtils,
  Classes,
  Dialogs;

{$R *.res}
type
  TConvertUnits=record //список типов величин
    Category:ShortString; //категория
    Units:array of ShortString; //наименования
  end;
  TUnitsTable=array of TConvertUnits;//таблица типов величин
  //тип функции перевода величин
  TConvertFunction=function (value:extended):extended; stdcall;
  //строка таблицы конвертации
  TConvertLine=record
```

```
    SrcUnit,DstUnit:ShortString;//исходный и требуемый типы
    ConvertFunc:TConvertFunction; //функция конвертации
  end;
  TConvertTable=array of TConvertLine; //таблица конвертации

  //тип функции экспорта таблицы типов из DLL
  UnitsTableExportFunc=function:TUnitsTable; stdcall;
  //тип функции экспорта таблицы конвертации из DLL
  ConvertTableExportFunc=function:TConvertTable;  stdcall;
  //тип функции сборки мусора из DLL
  GarbageCollectorFunc=procedure (var units:TUnitsTable;
                  var convert:TConvertTable); stdcall;

(*
Процедура CollectGarbage освобождает память из—под таблицы
единиц измерения units и таблицы конвертации convert.
*)
procedure CollectGarbage(var units:TUnitsTable;
                  var convert:TConvertTable); stdcall;
begin
   units:=NIL;
   convert:=NIL;
end;

(*
Функция ExportUnitsTable возвращает экспортируемую библиотекой
таблицу единиц измерения.
*)
function ExportUnitsTable:TUnitsTable; stdcall;
begin
  setlength(Result,1);
  Result[0].Category:='Энергия';
  setlength(Result[0].Units,4);
  Result[0].Units[0]:='Дж';
  Result[0].Units[1]:='кал.';
  Result[0].Units[2]:='эВ';
  Result[0].Units[3]:='Вт*ч';
end;

function JtoWh(value:extended):extended;  stdcall;
begin
  Result:=1/3600*value;
end;
function WhtoJ(value:extended):extended; stdcall;
begin
  Result:=3600*value;
end;

function eVtoJ(value:extended):extended; stdcall;
begin
  Result:=value/6.241e18;
end;
function JtoeV(value:extended):extended; stdcall;
begin
  Result:=6.241e18*value;
end;

function caltoJ(value:extended):extended; stdcall;
begin
  Result:=value/0.239;
end;
function Jtocal(value:extended):extended; stdcall;
begin
  Result:=0.239*value;
end;
```

```
//////////////////
function WhtoeV(value:extended):extended; stdcall;
begin
  Result:=2.247e22*value;
end;
function eVtoWh(value:extended):extended; stdcall;
begin
  Result:=value/2.247e22;
end;

function Whtocal(value:extended):extended; stdcall;
begin
  Result:=859.8*value;
end;
function caltoWh(value:extended):extended; stdcall;
begin
  Result:=value/859.8;
end;
/////////////////
function eVtocal(value:extended):extended; stdcall;
begin
  Result:=value/2.613e19;
end;
function caltoeV(value:extended):extended; stdcall;
begin
  Result:=2.613e19*value;
end;

(*
Функция ExportConvertTable возвращает экспортируемую библиотекой
таблицу единиц конвертации.
*)
function ExportConvertTable:TConvertTable; stdcall;
begin
  setlength(Result,12);
  Result[0].SrcUnit:='Дж';
  Result[0].DstUnit:='Вт*ч';
  Result[0].ConvertFunc:=JtoWh;

  Result[1].SrcUnit:='Дж';
  Result[1].DstUnit:='кал.';
  Result[1].ConvertFunc:=Jtocal;

  Result[2].SrcUnit:='Дж';
  Result[2].DstUnit:='эВ';
  Result[2].ConvertFunc:=JtoeV;

  Result[3].SrcUnit:='Вт*ч';
  Result[3].DstUnit:='Дж';
  Result[3].ConvertFunc:=WhtoJ;

  Result[4].SrcUnit:='кал.';
  Result[4].DstUnit:='Дж';
  Result[4].ConvertFunc:=caltoJ;

  Result[5].SrcUnit:='эВ';
  Result[5].DstUnit:='Дж';
  Result[5].ConvertFunc:=eVtoJ;
  ////////////////////
  Result[6].SrcUnit:='Вт*ч';
  Result[6].DstUnit:='эВ';
  Result[6].ConvertFunc:=WhtoeV;

  Result[7].SrcUnit:='Вт*ч';
  Result[7].DstUnit:='кал.';
```

```
  Result[7].ConvertFunc:=Whtocal;

  Result[8].SrcUnit:='кал.';
  Result[8].DstUnit:='Вт*ч';
  Result[8].ConvertFunc:=caltoWh;

  Result[9].SrcUnit:='эВ';
  Result[9].DstUnit:='Вт*ч';
  Result[9].ConvertFunc:=eVtoWh;
  //////////////////////////
  Result[10].SrcUnit:='эВ';
  Result[10].DstUnit:='кал.';
  Result[10].ConvertFunc:=eVtocal;

  Result[11].SrcUnit:='кал.';
  Result[11].DstUnit:='эВ';
  Result[11].ConvertFunc:=caltoeV;
end;
exports CollectGarbage,ExportUnitsTable,ExportConvertTable;
begin
end.
```

Тестовый пример

Ниже на рисунке 1 представлен пример работы программы конвертации единиц измерения.

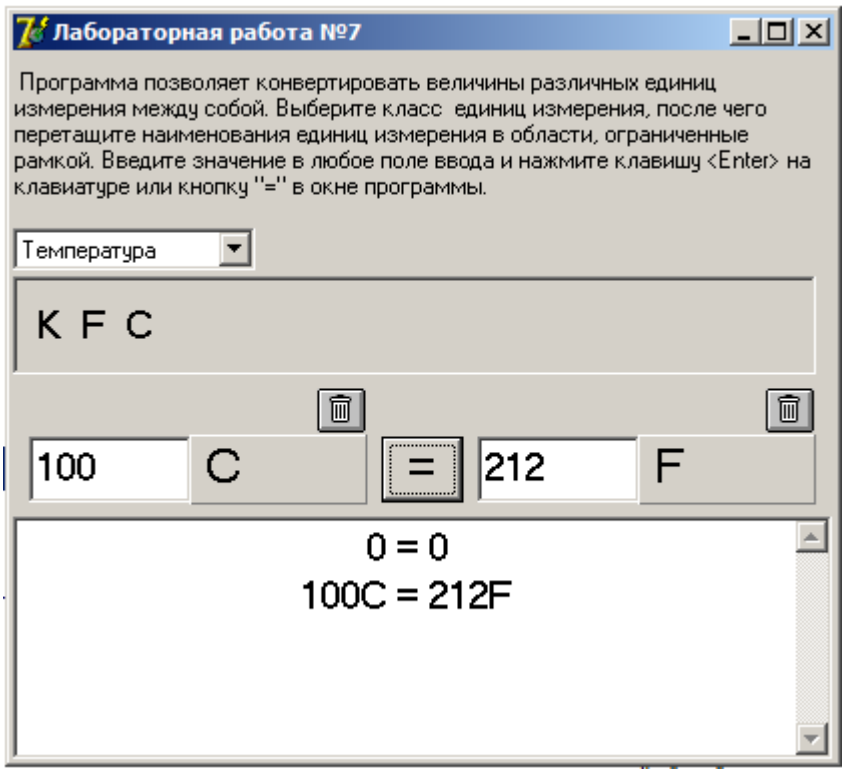

Рисунок 1 - Пример работы программы конвертации единиц измерения

Вывод

В этой лабораторной работе я изучил способы реализации механизма Drag-and-Drop в программах на Delphi. Механизм Drag-and-Drop позволяет создавать удобные и дружелюбные к пользователю приложения, т. к. позволяет сделать интерфейс приложения более интуитивным.