МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное
учреждение высшего профессионального образования
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Ю.К. Атрошенко

РАЗРАБОТКА ПРОГРАММНЫХ ПРИЛОЖЕНИЙ ДЛЯ ВЫПОЛНЕНИЯ ЛАБОРАТОРНЫХ РАБОТ В СРЕДЕ BORLAND DELPHI 7.0

УДК

Атрошенко Ю.К.

Разработка программных приложений для выполнения лабораторных работ в среде Borland Delphi 7.0 : методические указания / Ю.К. Атрошенко; Томский политехнический университет. – Томск: Изд-во Томского политехнического университета, 2014. – 39 с.

В пособии приведены функций и компонентов среды Delphi 7.0, необходимые для разработки приложения для выполнения типовых лабораторных работ по различным дисциплинам.

Пособие предназначено для студентов, обучающихся по направления 140100 «Теплоэнергетика и теплотехника» для разработки творческих проектов, а также для работы в рамках учебно-исследовательских работ.

ОГЛАВЛЕНИЕ

1. СОЗДАНИЕ РАБОЧИХ ФОРМ

Для создания нового проекта выполнить команду:

File → New → Application.

При этом появится новая чистая форма с названием *Form1* и окно кода с заголовком *Unit1.pas*.

После выбора размера формы, необходимо перейти к редактированию её свойств. Параметры и свойства выделенного объекта отображаются в окошке *Object Inspector*, его вид для созданной формы показан на рис. 1.1.

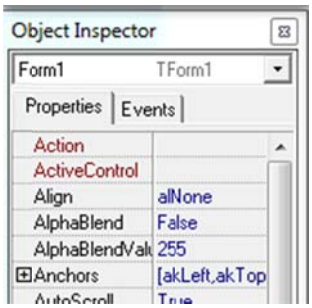

Рис. 1.1. Вид рабочего окна Object Inspector

В строке *Caption* прописывается новое название созданной формы, например ***«Теплотехнические измерения и приборы. Комплекс лабораторных работ»***. Это название будет отображаться в заголовке формы. Ниже в строке *Name* прописывается программное название формы, например ***main***. После этого необходимо сохранить форму под названием ***metod.pas***.

Для возможности дальнейшего расширения программного комплекса за счет добавления новых программ, создается программа с многооконным интерфейсом *MDI (Multiple Document Interface)*. Для этого созданную форму необходимо сделать «материнской»: в окне *Object Inspector* в строке *FormStyle* необходимо выбрать параметр *fsMDIForm* (рис. 1.2).

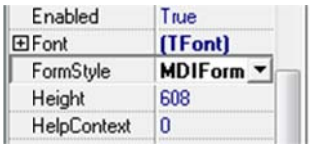

Рис. 1.2. Редактирование свойства FormStyle

Для фиксации изменения размеров запущенной программы, необходимо запретить ее разворачивание (максимизацию) (рис. 1.3).

Рис. 1.3. Запрет максимизации рабочей программы

Для этого в окне *Object Inspector* свойств формы в строке *BorderStyle* необходимо задать значение *bsSingle*, после чего раскрыть свойство *BorderIcons*, нажатием на символ «+», и для появившихся параметров задать значение *False* вместо значений *biMaximize* (рис. 1.4).

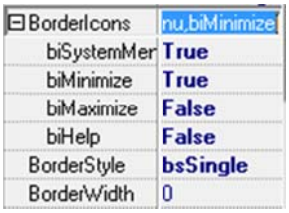

Рис. 1.4. Отключение возможности разворачивания программы на полный экран

Для добавления меню в созданную форму на вкладке *Standard* панели инструментов *Delphi* выбрать значок *MainMenu* (рис. 1.5) и разместить его на произвольном месте формы.

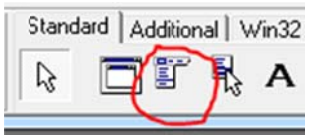

Рис. 1.5. Создание меню

После двойного нажатия на значке *MainMenu* появляется форма редактирования пунктов главного меню. Пункты меню добавляются щелчком на пунктирной рамке. При этом каждого пункта сразу после создания в окне *Object Inspector* в поле *Caption* необходимо прописать отображаемое название пункта и в поле *Name* – программное имя. Поле *Name* можно не редактировать, в этом случае пунктам автоматически будут присвоены имена *N1*, *N2* и т.д. Название первого пункта меню – ***Программа***. Для добавления пунктов в выпадающее меню необходимо нажать на созданный пункт меню правой кнопкой и выбрать *Insert* (рис. 1.6).

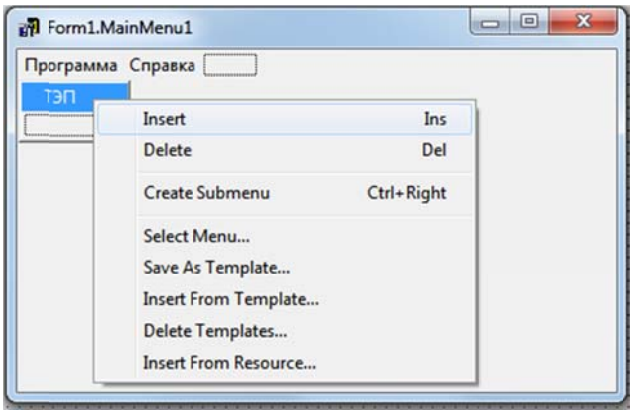

Рис. 1.6. Добавление пунктов в меню

В меню ***Программа*** добавляем название первой подпрограммы – ***ТЭП***. Теперь работа будет осуществляться с дочерней формой, вызываемой при выборе в меню ***Программа*** пункта ***ТЭП***.

Для создания дочерней формы выполнить команду:

File → New → Form.

При этом будет создана новая форма, размер ее должен быть немного меньше размера главной формы. Для редактирования свойств формы ее нужно выделить и в окне *Object Inspector* в поле *FormStyle* задать значение *fsMDIChild* (рис. 1.7), в поле Caption прописать отображаемое имя формы, например ***«Термоэлектрический преобразователь»***, в поле *Name* прописать программное имя *tep1*. После этого сохранить форму под именем *tep.pas*.

Рис. 1.7. Изменение свойства FormStyle

При запуске программы дочерняя форма автоматически появляется внутри главной. Чтобы этого избежать, необходимо выполнить следующее. В меню *Project → Options* в появившемся окне переместить дочернюю форму *tep1* из левой колонки в правую (рис. 1.8).

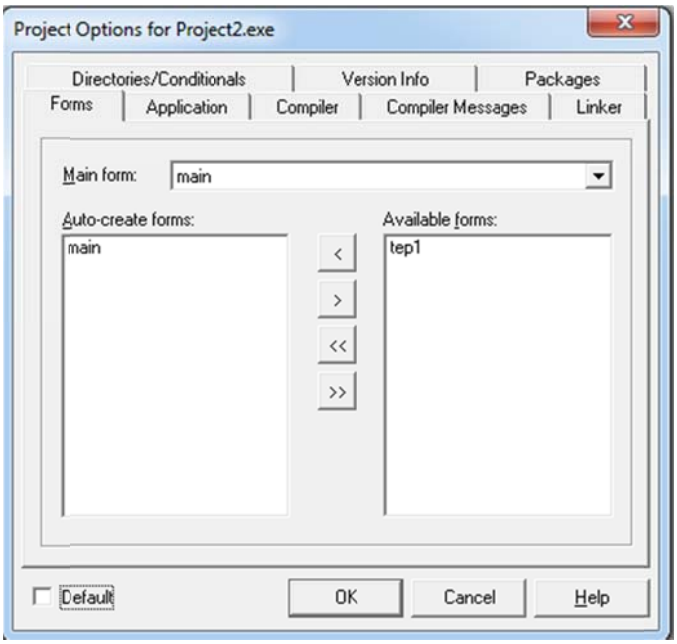

Рис. 1.8. Редактирование порядка запуска дочерней формы

Таким образом, при запуске программы будет появляться только главная форма, а дочерняя должна появляться при выборе пункта меню ***Программа → ТЭП***. Для этого необходимо нажать на пункт меню ***ТЭП*** при этом автоматически появится окно кода, в котором уже будет создана заготовка процедуры, которая будет выполняться при нажатии на пункт ***ТЭП*** в главном меню. В появившемся поле процедуры между *begin* и *end* необходимо добавить строку кода `MDIChildForm := Ttep1.Create(Owner);` (рис. 1.9.).

```
procedure Tmain.menu_tepClick(Sender: TObject);
begin
   MDIChildForm := Ttep1.Create(Owner);
end;
```

Рис. 1.9. Редактирование процедуры дочерней формы

Теперь информацию о дочерней форме необходимо добавить в код главной формы. В противном случае дочерняя форма при запуске программы не будет создана и возникнет ошибка при компиляции кода. В окне кода в верхней части окна после строки uses необходимо добавить в конце tep. Таким образом главной форме «сообщается», что будет использоваться файл tep.pas дочерней формы. (рис. 1.10). Далее после строки var необходимо добавить перепенную `MDIChildForm: Ttep1;` (1.10).

```
interface

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

type
  Tmain = class(TForm)
    MainMenu1: TMainMenu;
    menu_prog: TMenuItem;
    menu_help: TMenuItem;
    menu_tep: TMenuItem;
    menu_about: TMenuItem;
    procedure menu_tepClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  main: Tmain;
  MDIChildForm: Ttep1;

implementation

{$R *.dfm}
```

Рис. 1.10. Задание информации о дочерней форме

Для проверки правильности выполнения всех действий необходимо запустить программу, выполнив команду *Run* или клавишу *F9* (рис. 1.11).

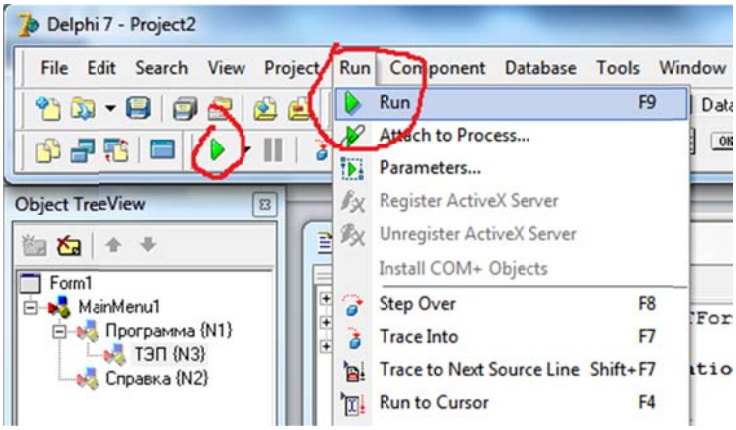

Рис. 1.11. Запуск программы

При запуске программы должна запуститься главная форма, а при выборе пункта *ТЭП* в меню *Программа* должна открываться дочерняя форма. Т.к. при попытке закрытия дочерней формы она не закрывается, а сворачивается, необходимо добавить процедуру обработки нажатия кнопки закрытия дочерней формы. В окне с кодом программы необходимо найти код дочерней вкладки (код на вкладку *tep*) и добавить туда следующую процедуру:

```
procedure Ttep1.FormClose(Sender: TObject; var Action: TCloseAction);
```

```
begin
  Action := caFree;
end;
```
(см. рис. 1.12).

```
metod  tep

var
  tep1: Ttep1;

implementation

{$R *.dfm}


procedure Ttep1.FormClose(Sender: TObject; var Action: TCloseAction);
begin
    Action := caFree;
end;


end.
```

Рис. 1.12. Обработка действия закрытия дочерней формы

При этом, необходимо убедиться, что в окне *Object Inspector* на вкладке *Events* для выбранной формы появилась эта процедура при действии *OnClose* (рис. 1.13).

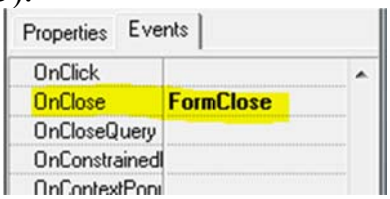

Рис. 1.13. Свойства формы

2. СОЗДАНИЕ ВКЛАДОК

Создание вкладок на внутренней (дочерней) форме осуществляется при помощи компонента *PageControl*, находящегося на вкладке *Win32* (рис. 2.1).

Рис. 2.1. Создание вкладок

После выбора компонента *PageControl*, растянуть его на всю область формы, но так, чтобы избежать появления полос прокрутки. Далее на свободном месте созданного прямоугольника *PageControl* щелчком правой кнопки мыши выбрать пункт *New Page*. Таким образом необходимо создать 4 вкладки (рис. 2.2).

Рис. 2.2. Создание вкладок

Для задания названия для вкладки, ее нужно выделить и ниже на ее свободном месте еще раз щелкнуть мышкой. Выделенной областью должна стать только область под вкладкой (рис. 2.3).

Рис. 2.3. Выделение вкладок

В окне свойств вкладки *Object Inspector* в поле *Caption* прописать название вкладки, в поле *Name* прописать ее имя. Повторить действия для всех созданных вкладок (рис. 2.4).

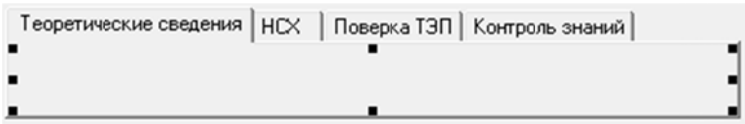

Рис. 2.4. Названия вкладок

3. ВКЛАДКА, СОДЕРЖАЩАЯ ГРАФИЧЕСКУЮ И ТЕКСТОВУЮ ИНФОРМАЦИЮ

Важно! Все процедуры, описывающие действие с компонентом должны быть вызваны из обработчика соответствующего события: Object Inspector, вкладка Events.

Для оформления вкладки, выполняющей только информативные функции можно начать с использования компонента *GroupBox*, расположенного на вкладке *Standard*. Использование этого компонента позволит выделить однородную информацию в группы. После расположения компонента *GroupBox* внутри формы в окне свойств компонента *Object Inspector* удалить текст в поле *Caption* или, при необходимости, изменить его на другое. Таким образом, необходимо разбить свободное поле вкладки на необходимое число полей.

Для добавления изображений используется компонент *Image*, расположенный на вкладке *Additional*. Изображения необходимо заранее подготовить (задать необходимы размер, четкость и другие свойства) с помощью любого графического редактора. В противном случае, редактирование изображения непосредственно в *Delphi* приведет к потере качества изображений и может сделать их нечитаемыми.

Добавление текста осуществляется при помощи компонента *Label*, расположенного на вкладке *Standard* (рис. 3.1).

Рис. 3.1. Добавление текста

После выбора компонента *Label*, его необходимо расположить на нужном месте поля вкладки внутри компонента *GroupBox*. Перед добавлением текста, необходимо обеспечить возможность переноса текста на следующую строку. Для этого в окне *Object Inspector* для созданного объекта *Label* нужно изменить значение в поле *WordWrap* на *True*. После этого необходимо изменить размеры границ *Label* на нужные и вставить заранее подготовленный текст (например, в Блокноте) в поле *Caption*.

Тип, размер и цвет шрифта можно изменить с помощью настроек в поле *Font* окна *Object Inspector* двумя способами: можно развернуть меню *Font*, нажатием на символ «+» слева от *Font* (рис. 3.2) или вызвать

диалоговое меню (рис. 3.3) нажатием на символ «…» справа от *Font* (рис. 3.2).

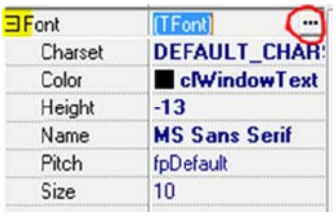

Рис. 3.2. Редактирование свойств шрифта с помощью окна Object Inspector

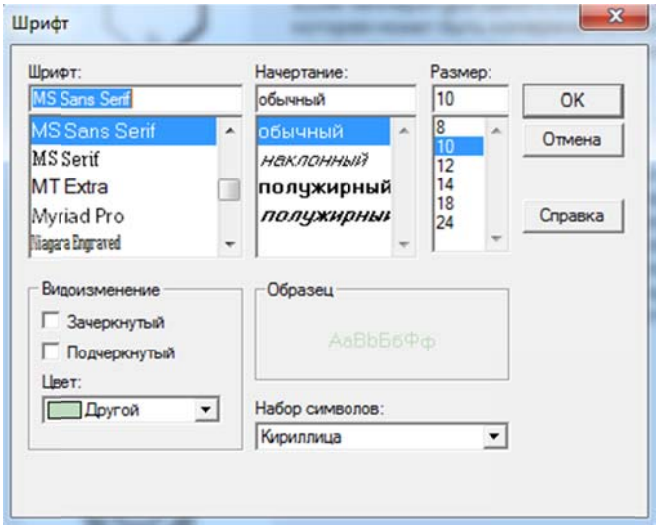

Рис. 3.3. Редактирование свойств шрифта с помощью диалогового окна

Размер шрифта, используемого по умолчанию – 8пт, в работе рекомендуется использовать шрифт размера 10пт или 12пт.

Параметр *Font* можно редактировать не только для объекта *Label*, но и для любого другого компонента, включающего в себя текст (например, названия вкладок, кнопок и т.д.). Пример выполнения описанной вкладки приведен на рис. 3.4.

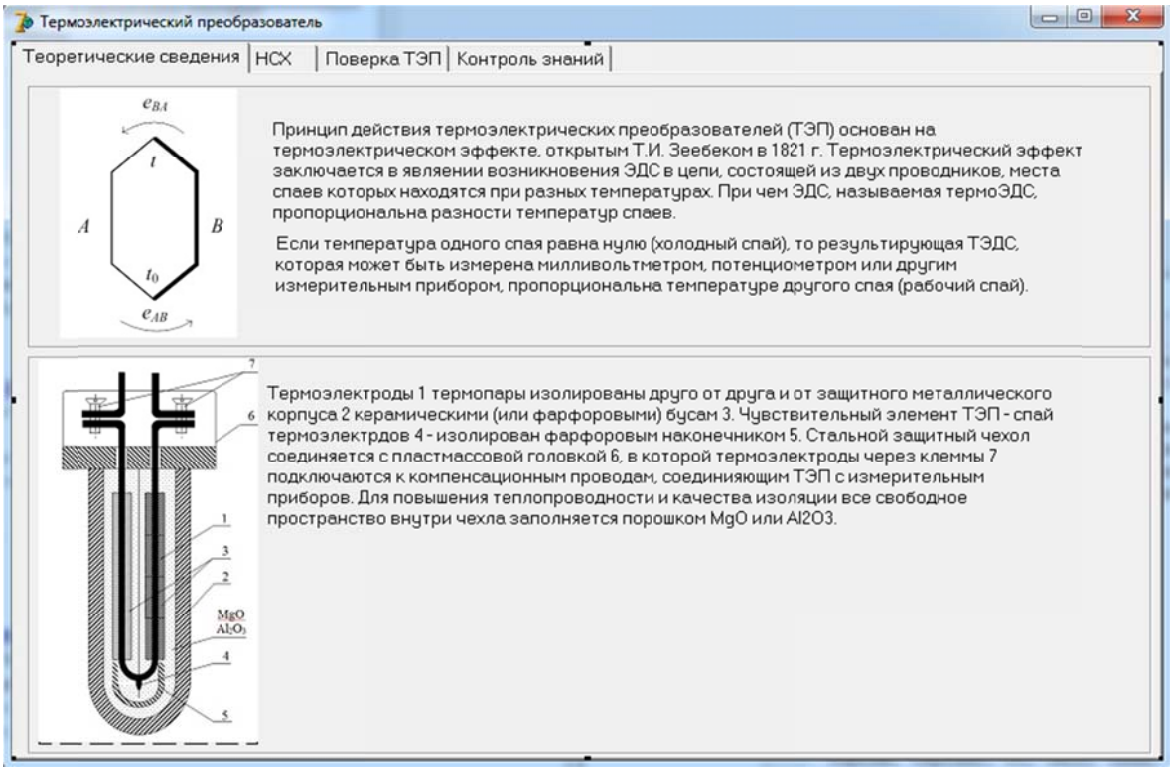

Рис. 3.4. Вид вкладки «Теоретические сведения»

4. ФУНКЦИОНАЛЬНАЯ ВКЛАДКА С РАСЧЕТОМ ТАБЛИЦЫ

4.1. Оформление внешнего вида вкладки

Используя компонент *GroupBox* разбить поле вкладки на необходимое число полей (например, два), при этом удалить текст из поля *Caption* окна *Object Inspector* созданных компонентов *GroupBox.*

Расположить поля друг под другом. Верхнее поле будет содержать текст, нижняя – для выполнения функциональных операций.

Разместить в верхнем поле текст с помощью компонента *Label*, выполнив для него операции по разрешению переноса текста на следующую строку и выравниванию текста по ширине.

Функциональным назначением нижнего поля *GroupBox* является вычисление номинальной статической характеристики термоэлектрического преобразователя в заданном диапазоне температур с заданным шагом по температуре и запись ее в таблицу.

Начинать работу следует с расположения компонентов, предназначенных для ввода исходных данных для расчета. Сначала нужно расположить компоненты *Label* для записи в них названий исходных данных. Расположить компоненты *Label* рекомендуется друг под другом и в поле *Caption* окна *Object Inspector* для каждого компонента прописать названия:

Label 1: «***Выберите тип ТЭП:***»;
Label 2: «***Введите начальную температуру, ˚C***»;
Label 3: «***Введите конечную температуру, ˚C***»;
Label 4: «***Введите шаг по температуре, ˚C***».

Справа от *Label 1* разместить компонент *ComboBox* (выпадающее меню для выбора типа ТЭП), расположенный на вкладке *Standard*. Затем справа от каждого *Label 2, 3, 4* разместить компоненты типа *Edit*, также расположенного на вкладке *Standard*, и предназначенные для ввода данных. При этом необходимо очистить поле *Text* в окне *Object Inspector* свойств компонентов *Edit*.

Ниже под компонентами разместить две кнопки – «***Расчет***» и «***Очистить***».

Схема размещения объектов показана на рис. 4.1.

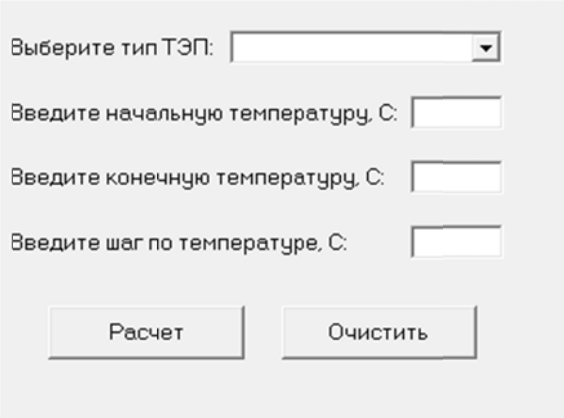

Рис. 4.1. Схема размещения объектов

В правой части вкладки можно поместить таблицу, предназначенную для вывода результатов расчета. Создание таблицы осуществляется с помощью компонента *StringGrig*, расположенного на вкладке *Additional*. Необходимо скорректировать ширину таблицы так, чтобы в ней поместилось два столбца. Для удаления лишних строк и столбцов в таблице в окне *Object Inspector* свойств таблицы в строке *ColCount* (количество столбцов) задать значение 2, в строке *FixedCols* (количество фиксированных столбцов) – 0, в строке *FixedRows* (количество фиксированных строк) – 1, в строке *RowCount* (количество строк) – 2. На практике же параметр *RowCount* будет динамически изменяющимся в зависимости от заданных пользователем исходных данных. Также нужно оставить немного свободного места в правой части таблицы для появления полосы прокрутки. Над таблицей расположить еще один компонент *Label* с названием «***Результаты:***». Вид получившейся вкладки приведен на рис. 4.2.

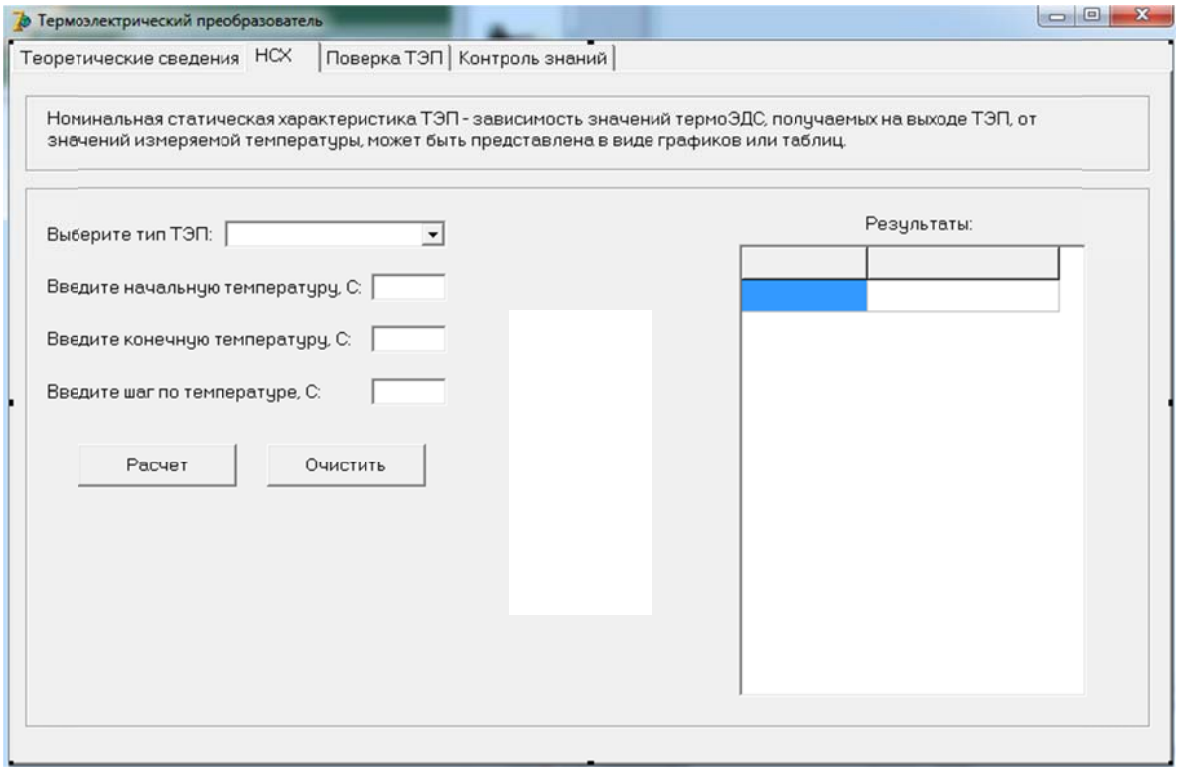

Рис. 4.2. Вид вкладки «Номинальная статическая характеристика»

4.2. Задание функций для объектов вкладки

Первый функциональный объект на разрабатываемой вкладке – *ComboBox*. В меню этого компонента нужно занести 7 видов ТЭП. Для этого в окне *Object Inspector* выделенного объекта нажать на кнопку «…» в поле параметра *Items*. При этом откроется поле *String List Editor*, в который нужно занести все типы ТЭП, при чем каждая новая строка представляет собой новый пункт в списке *ComboBox*. Порядок занесения определяет соответствие параметров указанным типам ТЭП и рекомендуется соблюдать п порядок, показанный на рис. 4.3.

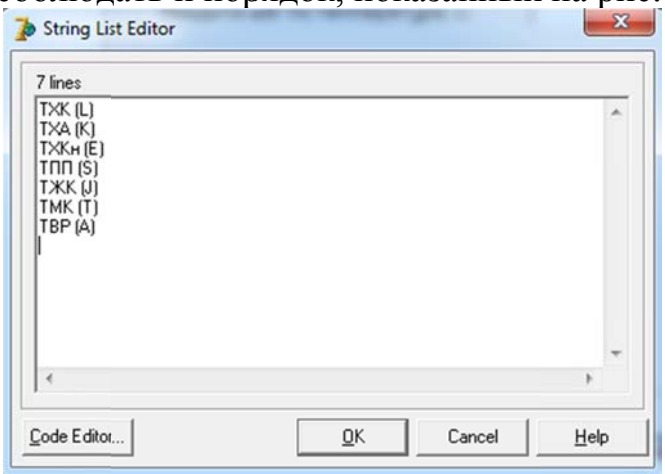

Рис. 4.3.Занесение типов ТЭП в мень компонента ComboBox

НСХ ТЭП в программе будет рассчитываться с использованием полинома вида:

$$E(t) = a \cdot t^6 + b \cdot t^5 + c \cdot t^4 + d \cdot t^3 + e \cdot t^2 + f \cdot t,$$

при чем, значения коэффициентов *a, b, ...f* известны для каждого вида ТЭП.

Все переменные, используемые в программе необходимо описать в окне кода программы. Для этого нужно найти строку *var* в начале кода и дописать значения:

```
tepnum, t1, tmin, tmax, t2, t3, trow: Integer;
 a, b, c, d, e, f, tres, t3f, result: Real;
t1n, t2n, t3n, tb: boolean;
```

После объявления всех переменных можно приступать к написанию процедур программы. Для написания первой процедуры, действующей при активации формы, нужно вызвать из вкладки *Events* окна *Object Inspector* двойным щелчком на поле параметра *OnActivate*. После открытия «заготовки» кода вписать между *begin* и *end* следующий код:

```
StringGrid1.Cells[0, 0] := 't, C';
StringGrid1.Cells[1, 0] := 'E, мВ';
```

Остальные процедуры предназначены для описания полинома для расчета НСХ и определения нижнего и верхнего пределов измерения того или иного типа ТЭП:

```
//ТХК (L)
procedure tep_l;
begin
  a := 3 * (exp((-17)*ln(10)));
  b := (-7) * (exp((-14)*ln(10)));
  c := 7 * (exp((-11)*ln(10)));
  d := (-7) * (exp((-8)*ln(10)));
  e := 6 * (exp((-5)*ln(10)));
  f := 0.0632;
  tmin := (-200);
  tmax := 800;
end;

//ТХА (K)
procedure tep_k;
begin
  a := 2 * (exp((-17)*ln(10)));
  b := (-7) * (exp((-14)*ln(10)));
  c := 1 * (exp((-10)*ln(10)));
  d := (-7) * (exp((-8)*ln(10)));
  e := 2 * (exp((-5)*ln(10)));
  f := 0.0378;
  tmin := (-250);
  tmax := 1370;
end;
```

```
//ТХКн (E)
procedure tep_e;
begin
  a := 7 * (exp((-18)*ln(10)));
  b := (-2) * (exp((-14)*ln(10)));
  c := 4 * (exp((-11)*ln(10)));
  d := (-6) * (exp((-8)*ln(10)));
  e := 6 * (exp((-5)*ln(10)));
  f := 0.0583;
  tmin := (-200);
  tmax := 900;
end;

//ТПП (S)
procedure tep_s;
begin
  a := 1 * (exp((-18)*ln(10)));
  b := (-7) * (exp((-15)*ln(10)));
  c := 2 * (exp((-11)*ln(10)));
  d := (-2) * (exp((-8)*ln(10)));
  e := 1 * (exp((-5)*ln(10)));
  f := 0.0055;
  tmin := (-50);
  tmax := 1660;
end;

//ТЖК (J)
procedure tep_j;
begin
  a := (-3) * (exp((-17)*ln(10)));
  b := 6 * (exp((-15)*ln(10)));
  c := 8 * (exp((-11)*ln(10)));
  d := (-9) * (exp((-8)*ln(10)));
  e := 3 * (exp((-5)*ln(10)));
  f := 0.0503;
  tmin := (-200);
  tmax := 900;
end;

//ТМК (T)
procedure tep_t;
begin
  a := 7 * (exp((-17)*ln(10)));
  b := (-6) * (exp((-14)*ln(10)));
  c := 3 * (exp((-11)*ln(10)));
  d := (-3) * (exp((-8)*ln(10)));
  e := 5 * (exp((-5)*ln(10)));
  f := 0.0386;
  tmin := (-200);
  tmax := 400;
end;

//ТВР (A)
```

```
procedure tep_a;
begin
  a := 2 * (exp((-19)*ln(10)));
  b := (-2) * (exp((-15)*ln(10)));
  c := 7 * (exp((-12)*ln(10)));
  d := (-1) * (exp((-8)*ln(10)));
  e := 1 * (exp((-5)*ln(10)));
  f := 0.0126;
  tmin := 0;
  tmax := 2500;
end;
```

Расчет НСХ будет производиться по нажатию кнопки «Расчет», следовательно, код обрабатывающей программы должен прописываться в теле процедуры, обрабатывающей это событие. С помощью двойного щелчка по кнопке «***Расчет***» откроется «заготовка» для процедуры нажатия кнопки «***Расчет***». Здесь требуется реализовать не только расчет и запись результата в таблицу, но и проверку корректности введенных данных: введены ли цифровые значения, не выходят ли введенные значения за пределы диапазона выбранного типа ТЭП и т.д.

Выбранное значение компонента *ComboBox* считывается с помощью кода ComboBox1.ItemIndex, возвращающего номер выбранного пункта, причем нумерация начинается со значения 0. Это значит, что в соответствии с приведенной на рис. 4.3. последовательностью при выборе пункта ТХК(L) ItemIndex возвращает значение 0, при выборе ТХА (К) – значение 1 и т.д. Если не выбран ни один пункт – значение -1. Код для определения выбранного типа ТЭП и обращения к процедуре, которая соответствует выбранному типу:

```
tepnum := ComboBox1.ItemIndex;
if tepnum = 0 then begin
   tep_l;
 end else if tepnum = 1 then begin
   tep_k;
 end else if tepnum = 2 then begin
   tep_e;
 end else if tepnum = 3 then begin
   tep_s;
 end else if tepnum = 4 then begin
   tep_j;
 end else if tepnum = 5 then begin
   tep_t;
 end else if tepnum = 6 then begin
   tep_a;
 end else begin
    ShowMessage('Выберите тип ТЭП');
 end;
```

Если таблица уже заполнена данными от предыдущего расчета, при нажатии кнопки «Расчет» таблица должна быть возвращена в исходное состояние и переменным, отвечающим за проверку введенных данных, должно быть присвоено значение *False*:

```
StringGrid1.RowCount := 2;
StringGrid1.Cols[0].Clear;
StringGrid1.Cols[1].Clear;
StringGrid1.Cells[0, 0] := 't, C';
StringGrid1.Cells[1, 0] := 'E, мВ';
trow := 1;
t1n := false;
t2n := false;
t3n := false;
```

Следующая часть кода предназначена для проверки корректности введенного значения начальной температуры (не выходит ли оно за пределы диапазона измерения выбранного типа ТЭП, введено ли цифровое значение). В случае успешной проверки переменным будут присвоены соответствующие значения.

```
if (TryStrToInt(Edit1.Text, t1)) and (StrToInt(Edit1.Text) >= tmin) and (StrToInt(Edit1.Text) <=
tmax) and (tepnum > (-1)) then
 begin
  t1 := StrToInt(Edit1.Text);
  t1n := true;
  tres := StrToFloat(Edit1.Text);
 end else
 begin
  ShowMessage ('Введите корректное значение начальной температуры');
 end;
```

Часть кода, выполняющая аналогичную проверку введенного значения конечной температуры:

```
if (TryStrToInt(Edit2.Text, t2)) and (StrToInt(Edit1.Text) >= tmin) and
(StrToInt(Edit1.Text) <= tmax) and (tepnum > (-1)) then
 begin
  t2 := StrToInt(Edit2.Text);
  t2n := true;
 end else
 begin
  ShowMessage ('Введите корректное значение конечной температуры');
 end;
```

Часть кода, выполняющая проверку введенного значения шага по температуре:

```
if (TryStrToInt(Edit3.Text, t3)) and (StrToInt(Edit3.Text)>0) then
begin
   t3 := StrToInt(Edit3.Text);
   t3f := StrToFloat(Edit3.Text);
   t3n := true;
end else
begin
 ShowMessage ('Введите корректное значение шага по температуре');
end;
```

После проведения проверок, запускается часть процедуры, выполняющая расчет значений термо-ЭДС. Т.к. определение расчетных значений производится для каждого значения от начальной до конечной температуры с заданным шагом, рационально использовать цикл *while do*. Запись операции возведения числа в степень в среде *Delphi* выглядит следующим образом: exp(a*ln(b)), где *a* – степень, а *b* – число, возводимое в степень. Т.к. возведение в степень не положительного числа (отрицательных чисел или нуля) при такой записи затруднено и приводит к ошибке. При возведении в степень отрицательных чисел возводиться в степень будет модуль числа, при этом, если степень нечетная, результат будет дополнительно умножен на (-1), т.к. $(-b)^n = b^n$ и $(-b)^{n-1} = -(b^{n-1})$, где *b* – число, возводимое в степень, *n* – четная степень. В случае, если *arg (= t)* = 0, значению термо-ЭДС присваивается значение 0 мВ. Для округления результатов до третьего знака после запятой используется оператор RoundTo(x, -3), где x – округляемое число. Итоговый код (включая проверку значений температуры на положительность) выглядит следующим образом:

```
if (t1n = true) and (t2n = true) and (t3n = true) then
begin
 StringGrid1.RowCount := ((t2 - t1) div t3 + 2);
 while (trow < ((t2 - t1) div t3 + 2)) do
 begin
  StringGrid1.Cells[0, trow] := FloatToStr(tres);
  if tres < 0 then begin
   result := RoundTo((a*exp(6*ln(abs(tres))))+(b*(-
1)*exp(5*ln(abs(tres))))+(c*exp(4*ln(abs(tres))))+(d*(-
1)*exp(3*ln(abs(tres))))+(e*exp(2*ln(abs(tres))))+(f*tres), -3);
   end else
   if tres = 0 then begin
   result := 0;
   end else
   begin
   result :=
RoundTo((exp(6*ln(tres))*a)+(exp(5*ln(tres))*b)+(exp(4*ln(tres))*c)+(exp(3*ln(tres))*d)+(exp(2*ln(tres))*e
)+(f*tres), -3);
```

```
        end;
        StringGrid1.Cells[1, trow] := FloatToStr(result);
       Inc(trow);
       tres := tres + t3f;
      end;

    end else
    begin
      ShowMessage ('Проверьте введённые значения');
    end;
```

Код процедуры копирования данных, выведенных в таблице:

```
procedure SGCopyToCLP(SG: TStringGrid; CopySel: Boolean; CL: integer = -1;
  RT: integer = -1; CR: integer = -1; RB: integer = -1);
var
  i, j: Integer;
  s: string;
begin
  s := '';
  with SG do
  begin
    if CopySel then
    begin
      CL := Selection.Left;
      CR := Selection.Right;
      RT := Selection.Top;
      RB := Selection.Bottom;
    end;
    if (CL < 0) or (CL > CR) or (CL >= ColCount) then
      CL := 0;
    if (CR < 0) or (CL > CR) or (CR >= ColCount) then
      CR := ColCount - 1;
    if (RT < 0) or (RT > RB) or (RT >= RowCount) then
      RT := 0;
    if (RB < 0) or (RT > RB) or (RB >= RowCount) then
      RB := RowCount - 1;
    for i := RT to RB do
    begin
      for j := CL to CR do
      begin
        s := s + Cells[j, i];
        if j < CR then
          s := s + #9;
      end;
      s := s + #13#10;
  end;
  end;
  ClipBoard.AsText := s;
end;
```

Для копирования данных из таблицы в буфер обмена по двойному щелчку мышки на вкладке *Events* окна *Object Inspector* найти параметр *OnDblClick* и два раза кликнуть мышью (рис. 4.4).

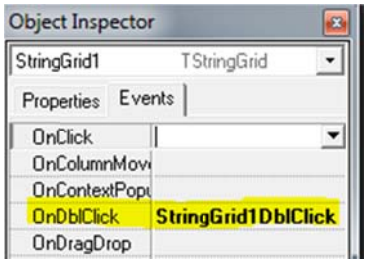

Рис. 4.4. Свойства таблицы

При этом полученная заготовка процедуры должна принять такой вид:

```
procedure Ttep1.StringGrid1DblClick(Sender: TObject);
begin
  SGCopyToCLP(StringGrid1, False);
end;
```

Для задания функции кнопки «***Очистить***» двойным щелчком по кнопке «***Очистить***» вызвать «заготовку» процедуры, код которой должен выглядеть так:

```
procedure Ttep1.Button2Click(Sender: TObject);
begin
   ComboBox1.ItemIndex := (-1);
   Edit1.Text := '';
   Edit2.Text := '';
   Edit3.Text := '';
   StringGrid1.Cols[0].Clear;
   StringGrid1.Cols[1].Clear;
   StringGrid1.RowCount := 2;
   StringGrid1.Cells[0, 0] := 't, C';
   StringGrid1.Cells[1, 0] := 'E, мВ';
   trow := 1;
end;
```

Законченный вид вкладки приведен на рис. 4.5.

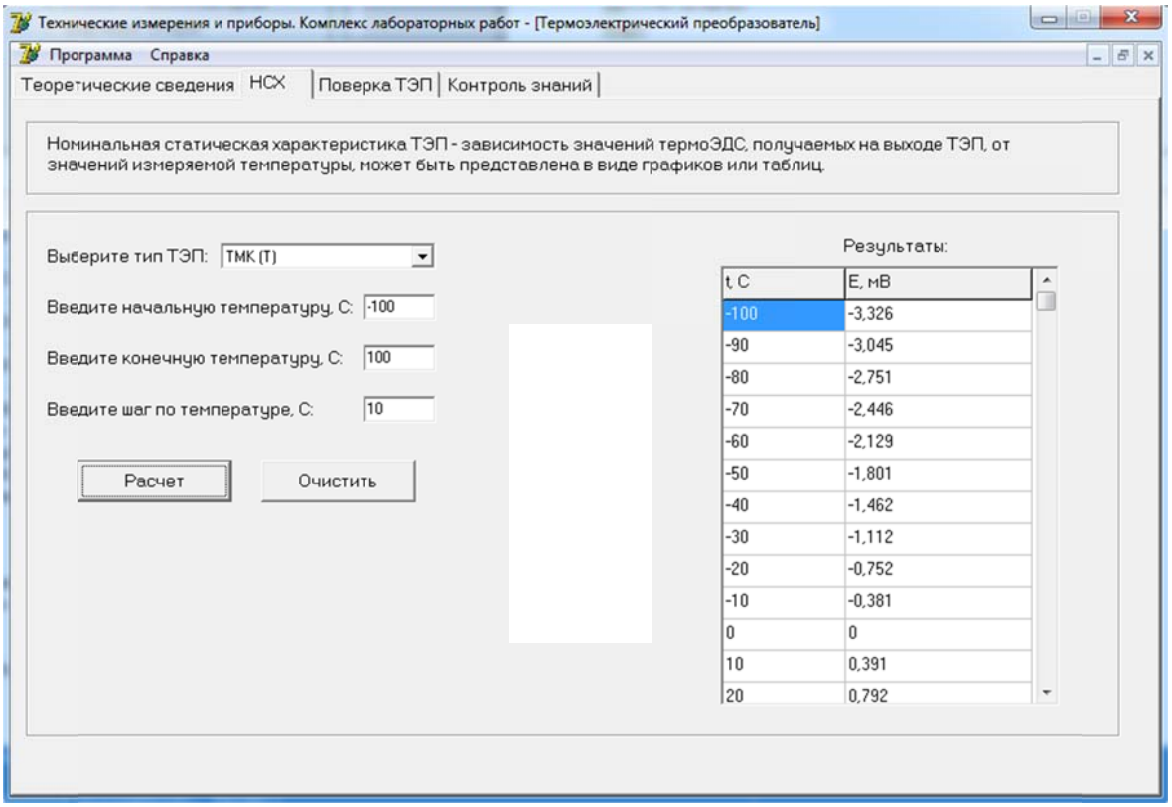

*Рис. 4.5 Вид вкладки «**Номинальная статическая характеристика**»*

5. ФУНКЦИОНАЛЬНАЯ ВКЛАДКА СО СЛУЧАЙНЫМ ЗАДАНИЕМ ВЕЛИЧИН

Вкладка «***Поверка ТЭП***» предназначена для имитации процедуры поверки термоэлектрических преобразователей. Т.к. поверка средств измерения температуры проводится методом сличения показаний поверяемого и эталонного ТЭП, генерировать необходимо показания двух термометров. Во время поверки термопреобразователи находятся в термостате с заданной температурой. Ввиду различных факторов фактическая температура в термостате может отличаться от заданной. Для определения фактической температуры служит эталонный термоэлектрический преобразователь типа *ТПП (S)*.

Разработка вкладки начинается с добавления компонентов *GroupBox*, разделяющих область вкладки на два свободных поля. Верхнее поле будет содержать теоретические сведения, нижнее – графическое изображение (рис. 5.1).

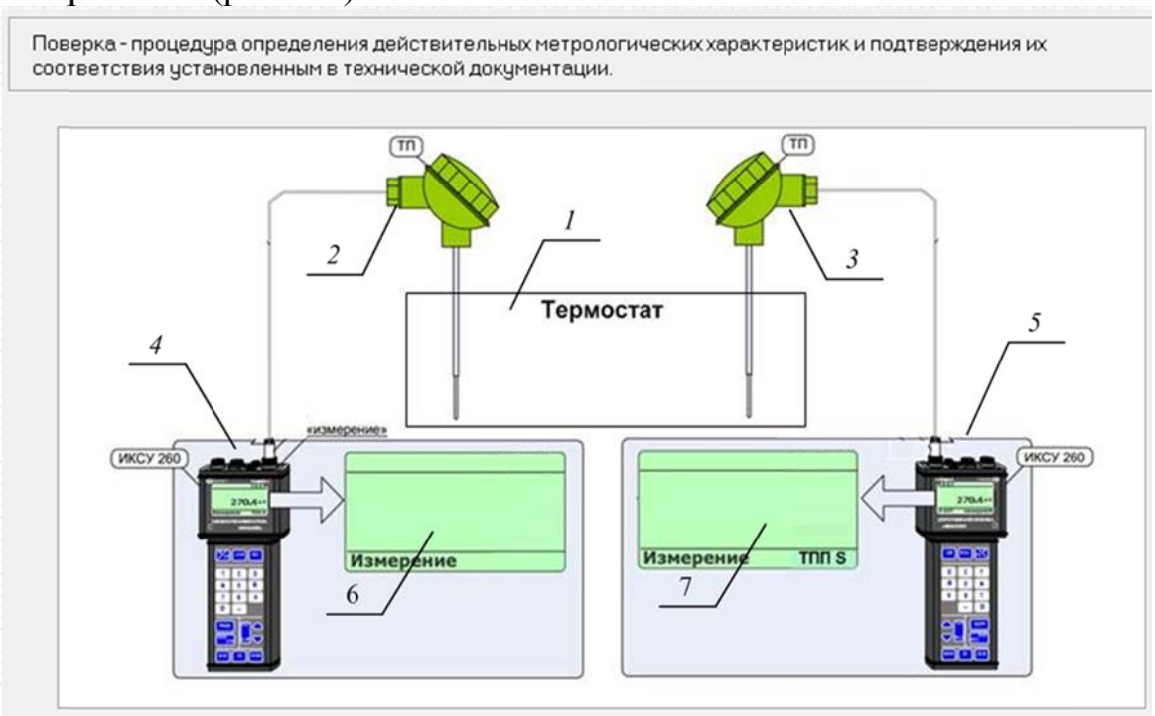

Рис. 5.1. Внешний вид вкладки:
1 – термостат; 2 – поверяемый ТЭП; 3 – эталонный ТЭП; 4 – калибратор, подключенный к поверяемому ТЭП; 5 – калибратор, подключенный к эталонному ТЭП;
6, 7 – поля для вывода показаний соответствующих калибраторов

В рабочем режиме на вкладке будут генерироваться показания калибратора, подключенного к эталонному ТЭП, по которым

пользователем будет определяться фактическая температура в термостате. Здесь нужно учитывать, что эти показания должны незначительно отличаться от заданной температуре в термостате, а значит должны определяться случайным образом, но в заданном диапазоне значений. Аналогичным образом будут генерироваться показания поверяемой температуры, которые также будут отличаться от температуры в термостате, но в более широком диапазоне.

Для визуализации вкладки нужно разместить функциональные компоненты «поверх» рисунка 5.1.

Сначала нужно разместить три компонента *Label* на поле вкладки. Т.к. они будут располагаться поверх рисунка, цвет их фона нужно сделать прозрачным. Для этого в окне *Object Inspector* свойств каждого компонента нужно изменить значение параметра *Transparent* на *True*.

Также на рисунок нужно поместить компонент *ComboBox*, содержащий список тех же ТЭП, что и на второй вкладке (рис. 4.3), три поля *Edit* (один для задания температуре и два для вывода результатов). Образец вкладки показан на рис. 5.2.

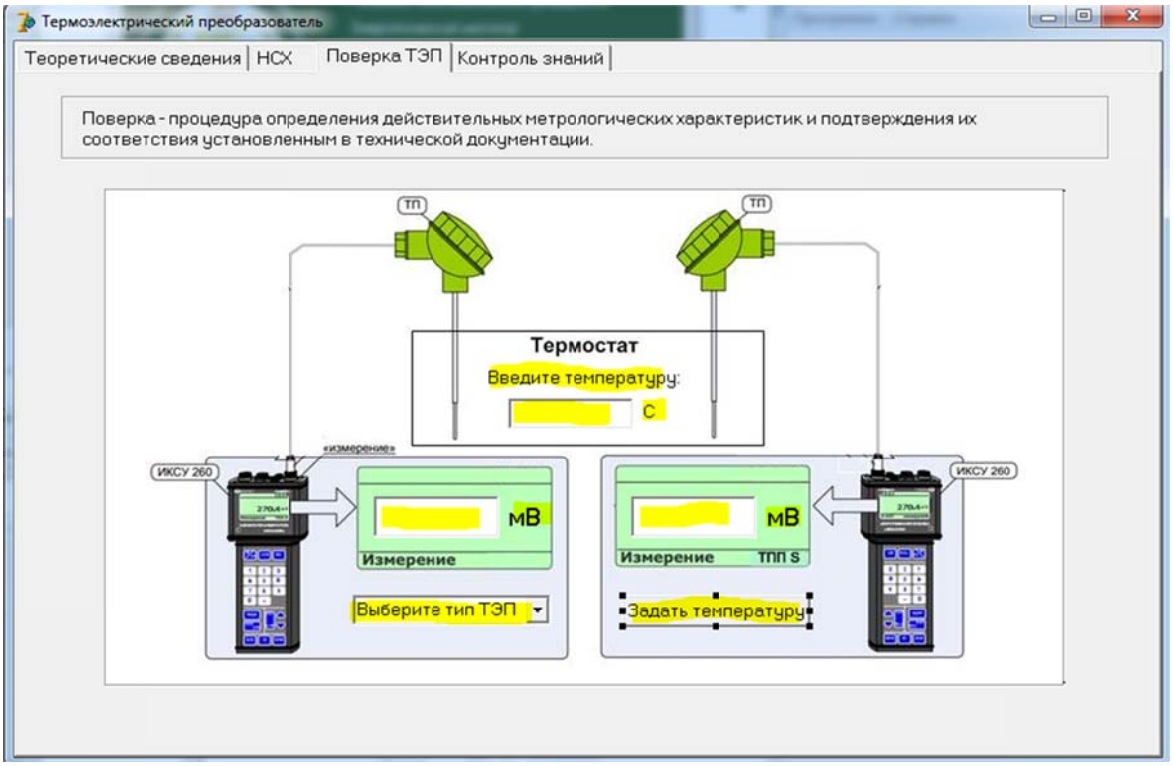

Рис. 5.2. Внешний вид вкладки «Поверка ТЭП»

Т.к. поля, предназначенные для вывода результатов не должны иметь возможность редактирования, для них следует задать параметр «Только чтение». Для этого в окне *Object Inspector* свойств каждого из компонентов нужно изменить значение параметра *ReadOnly* на *True*.

Рис. 5.3. Свойства компонентов Edit

Чтобы избежать связи со второй вкладкой будут введены новые переменные, которые необходимо также описать в начале программы после var, при этом переменные tepnum2, temp1, tmin2, tmax2 типа *Integer*, a2, b2, c2, d2, e2, f2, t3f, result1, result2, temp типа *Real* и t4n типа *Boolean*.

Все операции выполняются по нажатию кнопки «Задать температуру», поэтому следует вызвать «заготовку» процедуры двойным щелчком по этому компоненту. Сначала в теле процедуры необходимо задать коэффициенты полинома для определения термоЭДС, верхний и нижний пределы измерения эталонного ТЭП:

```
a2 := 1 * (exp((-18)*ln(10)));
b2 := (-7) * (exp((-15)*ln(10)));
c2 := 2 * (exp((-11)*ln(10)));
d2 := (-2) * (exp((-8)*ln(10)));
e2 := 1 * (exp((-5)*ln(10)));
f2 := 0.0055;
tmin2 := (-50);
tmax2 := 1660;
t4n := False;
tepnum2 := ComboBox2.ItemIndex;
```

Затем следует условие, определяющие параметры выбранного типа поверяемого ТЭП:

```
if tepnum2 = 0 then
  begin
    tep_l;
  end else if tepnum2 = 1 then
  begin
    tep_k;
  end else if tepnum2 = 2 then
  begin
    tep_e;
  end else if tepnum2 = 3 then
  begin
    tep_s;
  end else if tepnum2 = 4 then
  begin
    tep_j;
  end else if tepnum2 = 5 then
  begin
    tep_t;
  end else if tepnum2 = 6 then
```

```
    begin
      tep_a;
    end else
    begin
      ShowMessage('Выберите тип ТЭП');
    end;
```

После этого производится проверка введенных данных на корректность:

```
   if (TryStrToInt(Edit4.Text, temp1)) and (StrToInt(Edit4.Text) >= tmin) and (StrToInt(Edit4.Text) <= tmax)
and (StrToInt(Edit4.Text) >= tmin2) and (StrToInt(Edit4.Text) <= tmax2) and (tepnum2 > (-1)) then
    begin
      temp := StrToFloat(Edit4.Text);
      t4n := True;
    end else
    begin
      ShowMessage ('Введите корректное значение температуры');
    end;
```

Расчет начинается в случае успешного завершения проверок. Для произвольного задания температуры используется операторы *Randomize* (запускает генератор случайных чисел) и *RandomRange* (непосредственно генерирует случайное число в заданном диапазоне. Пусть показания ТЭП будут отличаться от показаний, соответствующих заданной температуре в термостате не более чем на 0,300. Т.к. оператор *Randomize* совместим только с целыми числами, генерироваться будет число в диапазоне от -300 до 300, которое затем будет разделено на 1000. Код программы тогда будет выглядеть следующим образом:

```
    if t4n = True then begin
      if temp < 0 then begin
         result1 := (a*exp(6*ln(abs(temp)))) + (b*(-1)*exp(5*ln(abs(temp)))) + (c*exp(4*ln(abs(temp)))) +
(d*(-1)*exp(3*ln(abs(temp)))) + (e*exp(2*ln(abs(temp)))) + (f*temp);
         result2 := (a2*exp(6*ln(abs(temp)))) + (b2*(-1)*exp(5*ln(abs(temp)))) + (c2*exp(4*ln(abs(temp)))) +
(d2*(-1)*exp(3*ln(abs(temp)))) + (e2*exp(2*ln(abs(temp)))) + (f2*temp);
      end else
      if temp = 0 then begin
         result1 := 0;
         result2 := 0;
      end else
      begin
         result1 :=
(exp(6*ln(temp))*a)+(exp(5*ln(temp))*b)+(exp(4*ln(temp))*c)+(exp(3*ln(temp))*d)+(exp(2*ln(temp))*e)+
(f*temp);
         result2 :=
(exp(6*ln(temp))*a2)+(exp(5*ln(temp))*b2)+(exp(4*ln(temp))*c2)+(exp(3*ln(temp))*d2)+(exp(2*ln(temp))
*e2)+(f2*temp);
      end;
      Randomize;
      result1 := result1 + ((RandomRange(-300,300)/1000));
```

```
  result2 := result2 + ((RandomRange(-100,100)/1000));
  Edit5.Text := FloatToStr(RoundTo(result1, -3));
  Edit6.Text := FloatToStr(RoundTo(result2, -3));
end;
```

Внешний вид законченной вкладки показан на рис. 5.4.

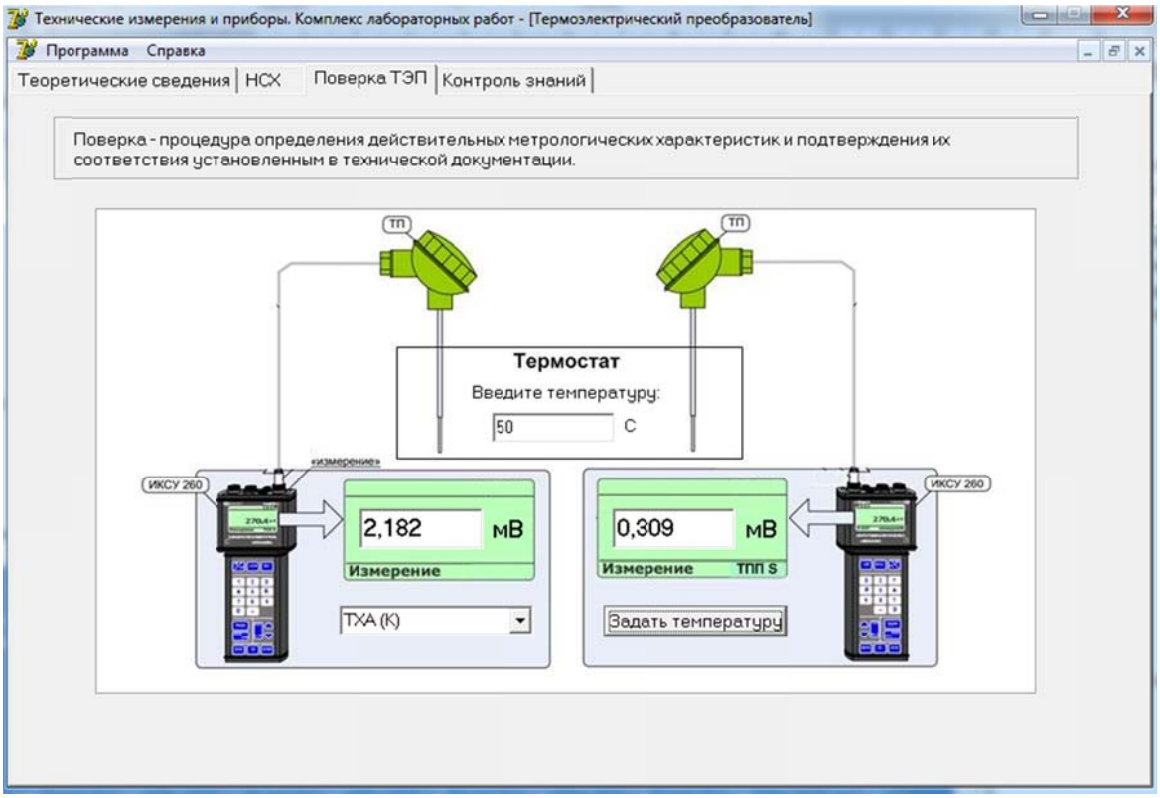

Рис. 5.4. Внешний вид вкладки «Поверка ТЭП»

6. ФУНКЦИОНАЛЬНАЯ ВКЛАДКА С ТЕСТИРОВАНИЕМ

Перед работой с вкладкой расположить один элемент *GroupBox* в верхней части вкладки и подписать название так, как показано на рис. 6.1.

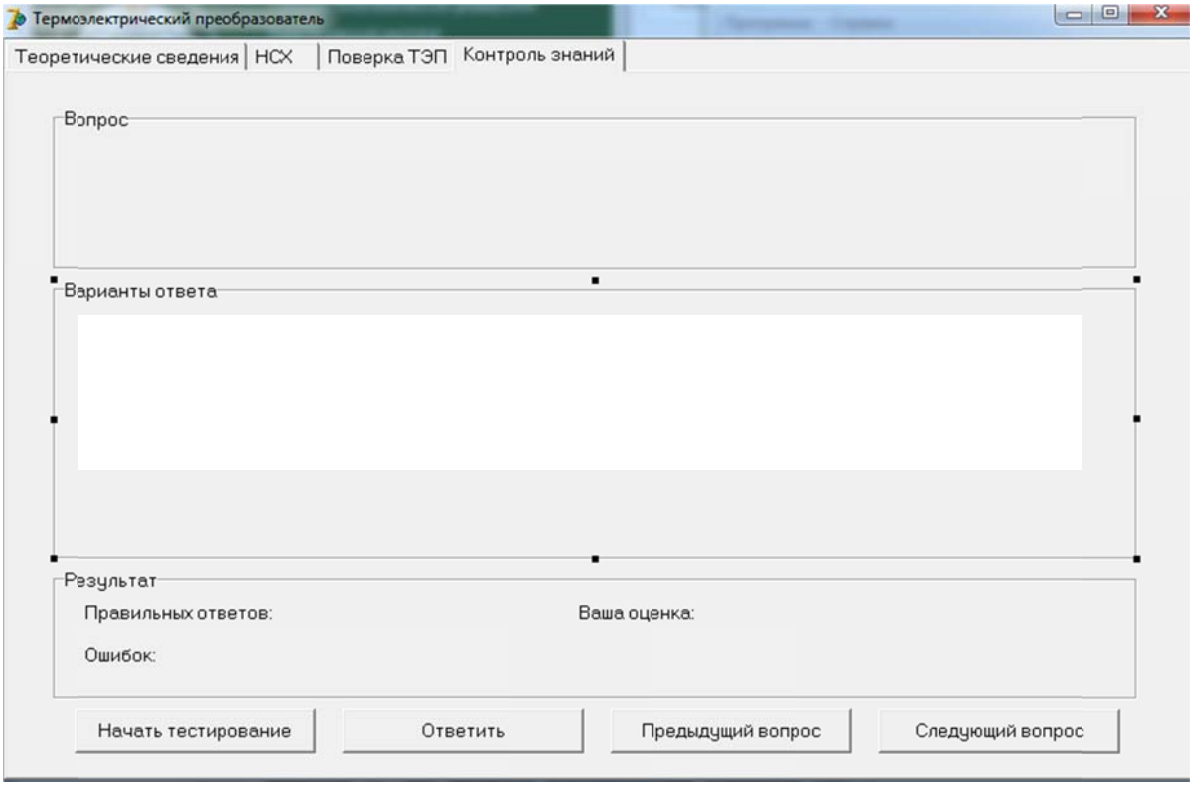

Рис. 6.1. Внешний вид вкладки «Контроль знаний»

Внутри компонента *GroupBox* «Вопрос» поместить компонент *Label* – пока пустой. Сразу задать для него параметры переноса текста (*WordWrap*), ширины (*Width*) и высоты (*Height*).

Под компонентом *GroupBox* расположить компонент *RadioGroup*, расположенный на вкладке *Standard*, назвать его «***Варианты ответа***».

В нижней части вкладки расположить еще один компонент *GroupBox* «***Результат***», внутри которого располагаются три компонента *Label*. В них будут занесены данные о количестве верных ответов и ошибок, оценка тестируемого.

Внизу вкладки разместить четыре компонента *Button*: «***Начать тестирование***» (после нажатия будет меняться на «***Закончить тестирование***», «***Ответить***», «***Предыдущий вопрос***» (для возможности изменения выбранного варианта ответа), «***Следующий вопрос***» (для возможности пропуска вопроса).

В окне свойств *Object Inspector* каждой кнопки в поле *Name* присвоить кнопкам следующие имена:

«***Начать/закончить тестирование***» – *Button_begin*
«***Ответить***» – *Button_ans*
«***Предыдущий вопрос***» – *Button_prev*
«***Следующий вопрос***» – *Button_next*

Для того, чтобы кнопки «***Ответить***», «***Предыдущий вопрос***» и «***Следующий вопрос***» были неактивными до нажатия кнопки «Начать тестирование» в окне свойств *Object Inspector* каждой кнопки следует изменить параметр *Enabled* на значение *False*.

После расположения компонентов, задания их свойств и параметров перейти в окно кода программы и в верхней части после var объявить следующие переменные:

```
quesnum, trueans, ii, ii1, ii2, ii3, ii4, ii5, ii6, prav, oshib: Integer
question, ans1, ans2, ans3, ans4, ocenka: String.
```

После объявления переменных для каждого вопроса пишется процедура:

```
procedure ques1;
begin
 question := 'Вопрос';
 ans1 := 'ответ 1';
 ans2 := 'ответ 2';
 ans3 := 'ответ 3';
 ans4 := 'ответ 4';
 trueans := 1;
end;
```

Такую процедуру следует написать для каждого вопроса, т.е. если вопросов в тесте 10, то процедур тоже будет 10. Названия процедур следует присваивать по порядку вопросов: *ques1, ques2, ques3* и т.д. Переменная *trueans* содержит номер верного ответа.

Следующая процедура предназначена для формирования отчета о результатах тестирования и оценки тестируемого. Вызов процедуры будет производиться по нажатию кнопки «***Завершить тестирование***»:

```
procedure results;
begin
 prav := 0;
 oshib := 0;
 if q1 = true then begin
  Inc(prav);
 end else begin
```

```
  Inc(oshib);
 end;
 if q2 = true then begin
  Inc(prav);
 end else begin
  Inc(oshib);
 end;
 if q3 = true then begin
  Inc(prav);
 end else begin
  Inc(oshib);
 end;
 if q4 = true then begin
  Inc(prav);
 end else begin
  Inc(oshib);
 end;
 if q5 = true then begin
  Inc(prav);
 end else begin
  Inc(oshib);
 end;
 if q6 = true then begin
  Inc(prav);
 end else begin
  Inc(oshib);
 end;
 if prav >= 5 then begin
  ocenka := 'Отлично';
 end else
 if prav = 4 then begin
     ocenka := 'Хорошо';
 end else
 if prav = 3 then begin
    ocenka := 'Удовлетворительно';
 end else
 if prav <= 2 then begin
    ocenka := 'Неудовлетворительно';
 end;
end;
```

Процедура, необходимая для корректного вывода определенного вопроса на экран, а также для выделения уже выбранных пользователем вариантов ответа:

```
procedure testing;
begin
 if quesnum = 1 then
 begin
  ques1;
  ii := ii1;
  end else if quesnum = 2 then
 begin
```

```
    ques2;
    ii := ii2;
  end else if quesnum = 3 then
  begin
    ques3;
    ii := ii3;
  end else if quesnum = 4 then
  begin
    ques4;
    ii := ii4;
  end else if quesnum = 5 then
  begin
    ques5;
    ii := ii5;
  end else if quesnum = 6 then
  begin
    ques6;
    ii := ii6;
  end;
end;
```

Процедура проверки правильности выбранного ответа будет вызываться при нажатии кнопки «***Ответить***»:

```
procedure answer_test;
begin
   if quesnum = 1 then begin
     q1 := True;
   end else if quesnum = 2 then begin
     q2 := True;
   end else if quesnum = 3 then begin
     q3 := True;
   end else if quesnum = 4 then begin
     q4 := True;
   end else if quesnum = 5 then begin
     q5 := True;
   end else if quesnum = 6 then begin
     q6 := True;
   end;
end;
```

Т.к. компиляция кода в *Delphi* производится последовательно, для вызываемые процедуры следует прописывать перед основными процедурами вкладки.

Далее прописываются процедуры, вызываемые при нажатии кнопок.

Первая описываемая кнопка «***Начать тестирование***», при этом после ее нажатия ее название должно измениться на «***Закончить тестирование***». Delphi позволяет изменять свойства объектов в режиме работы. Например, для активации ранее недоступной кнопки

«***Ответить***» может использоваться код `Button_ans.Enabled := True`, а добавить в список вариантов ответа новый пункт можно с помощью кода `RadioGroup1.Items.Add('Вариант ответа')`.

Создать «заготовку» процедуры обработки события «Щелчок по кнопке «Ответить» (см. описание в п. 5 для кнопки «Задать температуру») и записать в нее следующий код:

```
procedure Ttep1.Button_beginClick(Sender: TObject);
begin
 if tb = false then begin
  tb := true;
  Button_begin.Caption := 'Закончить тестирование';
  Button_ans.Enabled := True;
  Button_prev.Enabled := False;
  Button_next.Enabled := True;
  quesnum := 1;
  q1 := false;
  q2 := false;
  q3 := false;
  q4 := false;
  q5 := false;
  q6 := false;
  ii:= -1;
  ii1 := -1;
  ii2 := -1;
  ii3 := -1;
  ii4 := -1;
  ii5 := -1;
  ii6 := -1;
  testing;
  Label16.Width := 780;
  Label16.Caption := question;
  RadioGroup1.Items.Clear;
  RadioGroup1.Items.Add(ans1);
  RadioGroup1.Items.Add(ans2);
  RadioGroup1.Items.Add(ans3);
  RadioGroup1.Items.Add(ans4);
  Label17.Caption := 'Правильных ответов:';
  Label18.Caption := 'Ошибок:';
  Label19.Caption := 'Ваша оценка:';
  GroupBox7.Caption := 'Вопрос №' + IntToStr(quesnum);
  end else
 begin
  tb := false;
  Button_begin.Caption := 'Начать тестирование';
  Button_ans.Enabled := False;
  Button_prev.Enabled := False;
  Button_next.Enabled := False;
  Label16.Caption := '';
  RadioGroup1.Items.Clear;
  results;
  Label17.Caption := 'Правильных ответов: ' + IntToStr(prav);
```

```
    Label18.Caption := 'Ошибок: ' + IntToStr(oshib);
    Label19.Caption := 'Ваша оценка: ' + ocenka;
  end;
end;
```

Очевидно, что первая часть кода проверяет, запущен ли тест, и если он не был запущен, запускает его, выводит на экран вопрос и ответ, активируя при этом все необходимые кнопки. Данные о том, запущен ли тест, содержит переменная *tb*. Для того чтобы изначально она имела значение *False*, нужно вернуться в начало кода, найти процедуру `procedure Ttep1.FormActivate(Sender: TObject);` и прописать в ее теле строку `tb := false;`.

Вторая часть приведенного кода выполняет действия, в случае, если нажата кнопка «***Закончить тест***».

Чтобы тестирование было более эффективным, следует во время выполнения теста запретить обращение к вкладкам, содержащим теоретическую информацию. Выполнить это действие можно с помощью следующей процедуры:

```
procedure Ttep1.PageControl1Changing(Sender: TObject;
  var AllowChange: Boolean);
begin
  If tb = true then
  AllowChange := False
  else
  AllowChange := True;
end;
```

При нажатии кнопки «Ответить» проверяется, выбран ли ответ. После чего производится вызов процедуры проверки правильности ответа (*answer_test*) и проверка номера текущего вопроса. Исходя их этого, определяется, выводить следующий вопрос или вывести сообщение о том, что вопросы кончились и пользователь может нажать кнопку «***Закончить тест***» и посмотреть результат своего тестирования или вернуться к предыдущим вопросам и изменить выбранный вариант ответа.

```
procedure Ttep1.Button_ansClick(Sender: TObject);
begin
  if RadioGroup1.ItemIndex = (trueans - 1) then
  begin
    answer_test;
    testing;
    if quesnum >= 6 then begin
      ShowMessage('Это последний вопрос');
    end else
    begin
```

```
  Button_nextClick(Self);
  end;
  end
 else if RadioGroup1.ItemIndex = (-1) then begin
  ShowMessage('Выберите ответ');
 end
 else if RadioGroup1.ItemIndex <> (trueans - 1) then begin
  if quesnum >= 6 then begin
   ShowMessage('Это последний вопрос');
  end else
  begin
  Button_nextClick(Self);
  end;
 end;
```

Процедура, выполняемая при нажатии кнопки «***Следующий вопрос***» предназначена для вывода следующего по порядку вопроса. В том случае, если в текущий момент на экран выведен последний вопрос, кнопка «***Следующий вопрос***» должна стать неактивной (вернуться в активное состояние она может только в случае нажатия на кнопку «***Предыдущий вопрос***»):

```
procedure Ttep1.Button_nextClick(Sender: TObject);
begin
  quesnum := quesnum + 1;
  testing;
  Label16.Width := 780;
  Label16.Caption := question;
  RadioGroup1.Items.Clear;
  RadioGroup1.Items.Add(ans1);
  RadioGroup1.Items.Add(ans2);
  RadioGroup1.Items.Add(ans3);
  RadioGroup1.Items.Add(ans4);
  RadioGroup1.ItemIndex := ii;
  Button_prev.Enabled := True;
  GroupBox7.Caption := 'Вопрос №' + IntToStr(quesnum);
  if quesnum = 6 then begin
  Button_next.Enabled := False;
  end;
end;
```

Аналогичным образом описывается процедура «Предыдущий вопрос». Отличие состоит в том, что при нажатии кнопки «Предыдущий вопрос», проверяется наличие предыдущего вопроса и, если вопрос есть, он выводится на экран. В противном случае кнопка «Предыдущий вопрос» становится неактивной.

```
procedure Ttep1.Button_prevClick(Sender: TObject);
begin
  quesnum := quesnum - 1;
```

```
   testing;
   Label16.Width := 780;
   Label16.Caption := question;
   RadioGroup1.Items.Clear;
   RadioGroup1.Items.Add(ans1);
   RadioGroup1.Items.Add(ans2);
   RadioGroup1.Items.Add(ans3);
   RadioGroup1.Items.Add(ans4);
   RadioGroup1.ItemIndex := ii;
   Button_next.Enabled := True;
   GroupBox7.Caption := 'Вопрос №' + IntToStr(quesnum);
   if quesnum = 1 then begin
   Button_prev.Enabled := False;
  end;
end;
```

Процедура, предназначенная для «запоминания» выбранного пользователем ответа в каждом вопросе:

```
procedure Ttep1.RadioGroup1Click(Sender: TObject);
begin
   if quesnum = 1 then begin
     ii1 := RadioGroup1.ItemIndex;
   end else if quesnum = 2 then begin
     ii2 := RadioGroup1.ItemIndex;
   end else if quesnum = 3 then begin
     ii3 := RadioGroup1.ItemIndex;
   end else if quesnum = 4 then begin
     ii4 := RadioGroup1.ItemIndex;
   end else if quesnum = 5 then begin
     ii5 := RadioGroup1.ItemIndex;
   end else if quesnum = 6 then begin
     ii6 := RadioGroup1.ItemIndex;
   end;
end;
```

Общий вид вкладки показан на рис. 6.2.

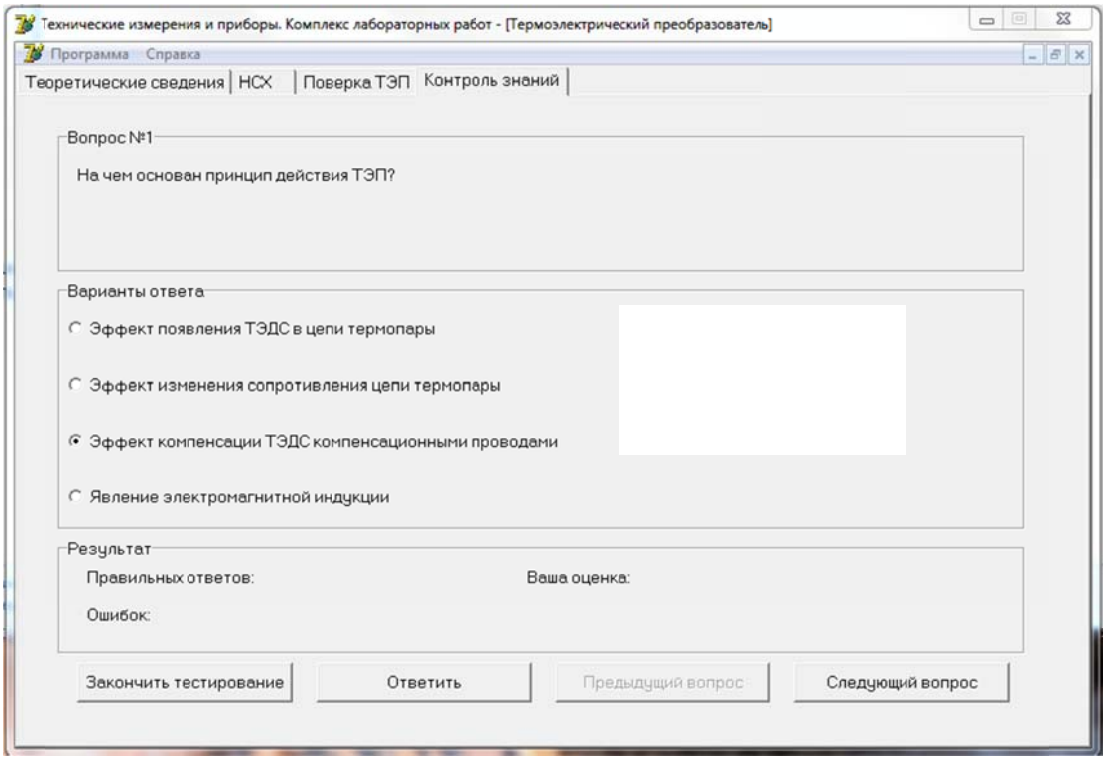

Рис. 6.2. Общий вид вкладки «Контроль знаний»

Учебное издание

АТРОШЕНКО Юлиана Константиновна

РАЗРАБОТКА ПРОГРАММНЫХ ПРИЛОЖЕНИЙ ДЛЯ ВЫПОЛНЕНИЯ ЛАБОРАТОРНЫХ РАБОТ В СРЕДЕ BORLAND DELPHI 7.0

Учебное пособие

Издано в авторской редакции

Подписано к печати. Формат 60x84/16. Бумага «Снегурочка». Печать XEROX. Усл.печ.л. 9,01. Уч.-изд.л. 8,16. Заказ . Тираж 5 экз.		
nqa. ISO 9001 Registered	Национальный исследовательский Томский политехнический университет Система менеджмента качества Издательства Томского политехнического университета сертифицирована NATIONAL QUALITY ASSURANCE по стандарту BS EN ISO 9001:2008	UKAS QUALITY MANAGEMENT 015
ИЗДАТЕЛЬСТВО ТПУ. 634050, г. Томск, пр. Ленина, 30 Тел./факс: 8(3822)56-35-35, www.tpu.ru		