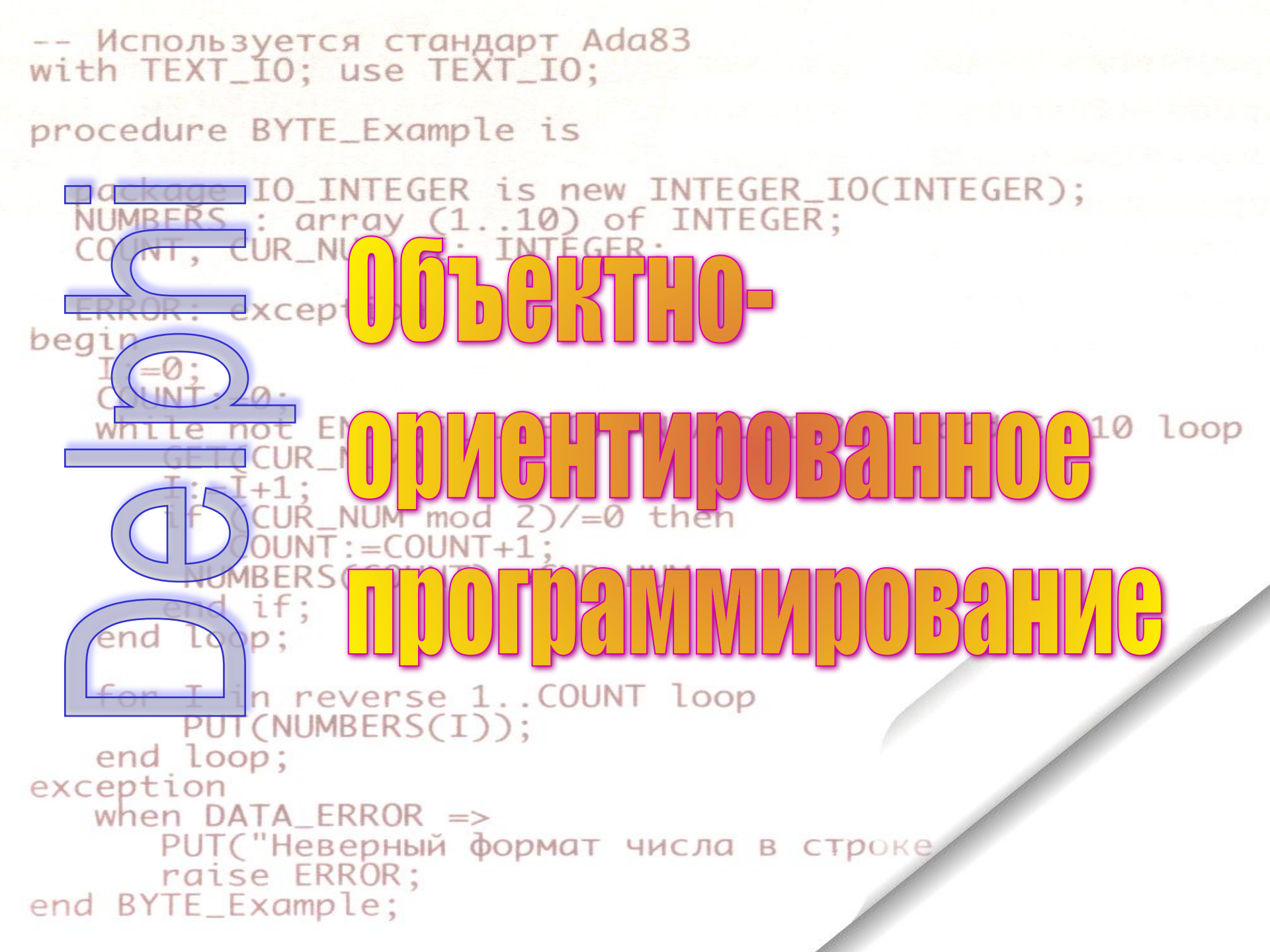

-- Используется стандарт Ada83
with TEXT_IO; use TEXT_IO;
procedure BYTE_Example is
package IO_INTEGER is new INTEGER_IO(INTEGER);
NUMBERS : array (1..10) of INTEGER;
COUNT, CUR_NUM : INTEGER;
ERROR: exception;
begin
I:=0;
COUNT:=0;
while not EN
GET(CUR_N
I:=I+1;
if (CUR_NUM mod 2)/=0 then
COUNT:=COUNT+1;
NUMBERS(
end if;
end loop;
for I in reverse 1..COUNT loop
PUT(NUMBERS(I));
end loop;
exception
when DATA_ERROR =>
PUT("Неверный формат числа в строке
raise ERROR;
end BYTE_Example;
Delphi
Объектно-
ориентированное
программирование

Основные понятия

Среда программирования Delphi

Описание объектов

Проекты

Выход

Объектно-ориентированное программирование (ООП) – это программирование, основанное на составлении программ в виде совокупности объектов, каждый из которых принадлежит определенному классу, а классы составляют иерархию наследования.

*Язык программирования **Delphi** построен на концепции объектно-ориентированного программирования на основе языка Паскаль.*

Delphi оперирует объектными типами данных, которые называют ***классами***, а их экземпляры – ***объектами***. Объекты характеризуются ***полями***.

Объекты имеют определённый набор свойств (атрибутов) и могут выполнять действия. Действия могут выполняться и над ними.

Класс *– это структура, которая, кроме описания данных, содержит описание процедур и функций, которые могут быть выполнены представителем класса –* ***объектом***.

Данные класса называются ***полями***, процедуры и функции – ***методами***.

Поля *класса являются переменными, которые описываются в середине класса. Они предназначены для хранения данных во время работы объекта*.

Методы *– это процедуры и функции, описанные классами и предназначенные для операции над его полями*.

От обыкновенных процедур и функций методы отличаются тем, что им во время вызова передается указатель на объект, который его вызвал, поэтому обрабатываются поля того объекта который вызывал метод.


Класс Белок
Класс Лисиц
зоопарк
Класс Медведей
Класс Попугаев
Белка
Лисица
Медведь
Клетка с птицами
Цвет шерсти - рыжий Хвост – длинный Размер - маленький
Цвет шерсти - рыжий Хвост – длинный Размер - средний
Цвет шерсти - рыжий Хвост – длинный Размер - большой
Попугай
Синица
Канарейка
Класс Синиц
Цвет шерсти – Хвост – Размер -
Классы:
Объекты
Поле (параметр объекта)

Объекты, которые визуализированы на стадии проектирования, называют ***компонентами***.

Работа самой операционной системы Windows и ее приложений основана на понятии ***событие***. Если программный компонент меняет свое состояние, он генерирует соответствующее уведомление, или реагирует на него. Таким образом – ***событие*** *– это уведомление, которое генерируется программой для того, чтобы проинформировать на изменение.* Обработка событий выполняется соответствующими подпрограммами.

Создание программ объектно-ориентированными языками состоит в том, чтобы разработать визуальные объекты и программы, для обработки событий при работе с объектами. Все составляющие данной разработки называют ***проектом***.

Объектно-ориентированное программирование базируется на трех основных ***понятиях***: *инкапсуляция*, *наследственность*, *полиморфизм*.

1. ***Инкапсуляция*** *– объединение данных и методов в объект, она тесно связана со скрытием внутренней структуры объекта*. Инкапсуляция дает возможность делать объекты похожими на программные модули, в которых внутренняя структура спрятана.

2. ***Наследование***. При создании нового класса, который расширяет возможности существующего, нет необходимости в повторном переписывании полей и методов. Достаточно объявить новый класс дочерним относительно существующего (родительского) и добавить к нему новые поля и методы. *Создание новых классов на базе существующих называется наследованием*.

3. ***Полиморфизм***. В дочерних классах можно модифицировать существующие в родительском классе методы. Полиморфизм – возможность использования одинаковых имен для методов которые входят в разные классы.

Основы проектной разработки в среде Delphi

Разработка проекта в среде Delphi состоит из 5 основных *этапов*:

1 этап. Формулировка задания, для которого реализуется проект;

2 этап. Проведение проектного анализа и формирование требований к классам, объектам, компонентам, которые понадобятся в данном проекте;

3 этап. Выбор необходимых компонентов и разработка алгоритмов обработки компонентов;

4 этап. Компиляция программы на основании разработанного проекта;

5 этап. Проверка правильности работы программы (отладка) и соответствие постановки задачи.

Объектно-ориентированная среда

Объектно-ориентированная программная среда предоставляет в распоряжение программиста готовые объекты – элементы интерфейса операционной системы Windows.

Интегрированная среда обеспечивает высокоэффективную работу программиста за счет работы одновременно в нескольких окнах.

В Delphi существует 10 окон различного назначения, но после загрузки среды появляются 5 основных окон:

1. ***главное окно Delphi7*** – ***Progect1*** (имя проекта);
2. ***окно с формой*** для проектирования программных продуктов ***Form1;***
3. ***окно инспектора объектов Object Inspector;***
4. ***окно редактора кода*** (написание программы) ***Unit.pas.***
5. ***Проводника кода Exploring Unit1.pas***

Форма *– это основа окна Windows-приложения, создаваемого в среде Delphi.*

Форма *– основной объект и, как и остальные объекты, имеет свойства, которые можно задавать и менять.*

Увидеть свойства объекта и изменить их можно в окне ***Инспектора объектов****. (Не забудьте предварительно объект выделить!)*

Компоненты *(готовые объекты) – элементы интерфейса операционной системы Windows.*

Компоненты могут быть ***визуальные*** *и* ***невизуальные.***

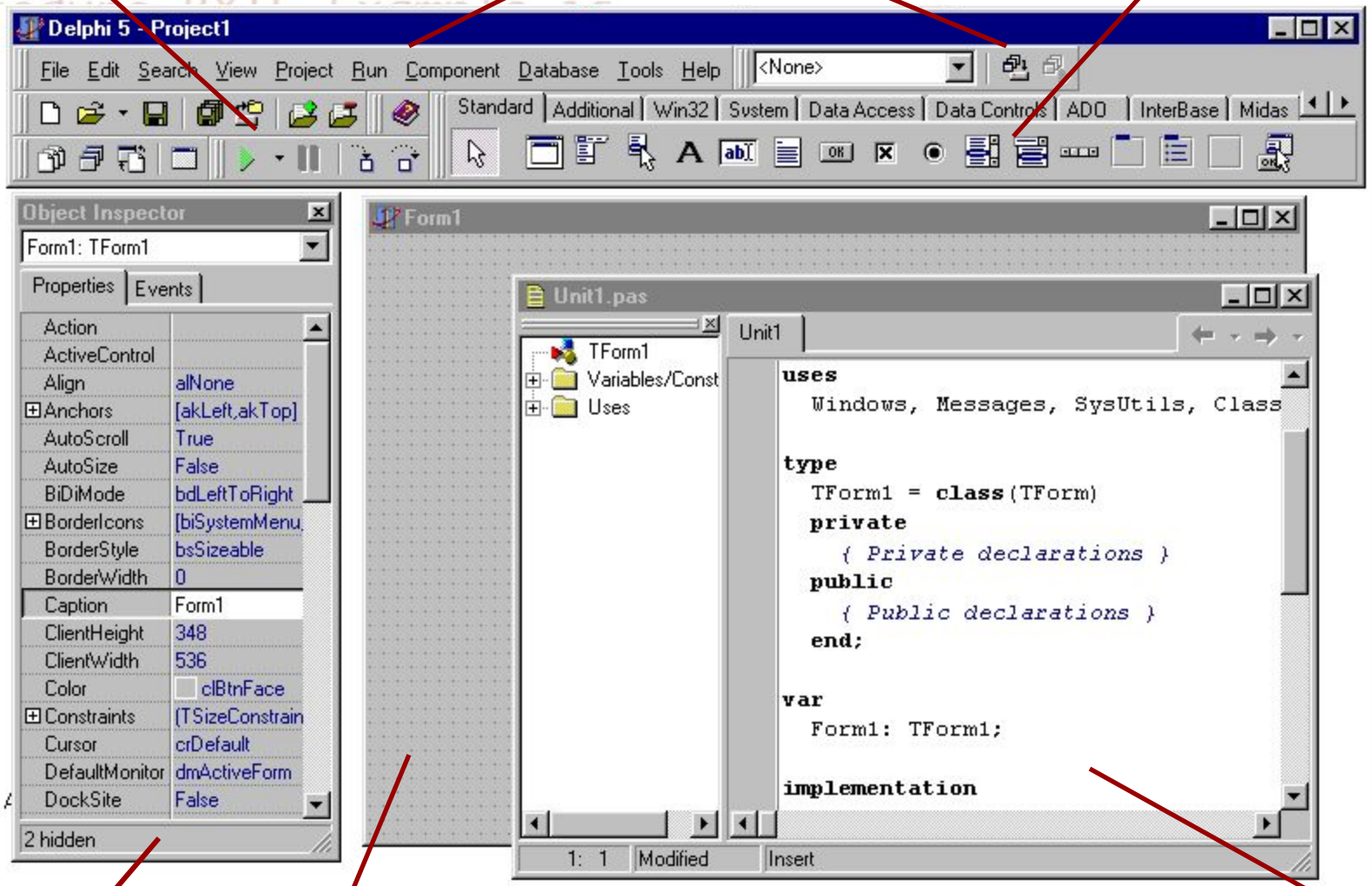
Например кнопка – визуальный компонент.

таймер, компоненты построения диалога – невизуальные компоненты, но дающие видимый результат.

Окно проекта

Панели инструментов («быстрые» кнопки)

Кнопки компонентов



Окно Инспектора объектов

Окно главной формы

Окно Редактора кодов

Кроме этих окон полезно иметь под рукой ***окно дерева объектов Object TreeView*** (открывается горячими клавишами Shift+Alt+F11***)***, с помощью которого можно не только видеть объекты, но и перемещаться между ними.

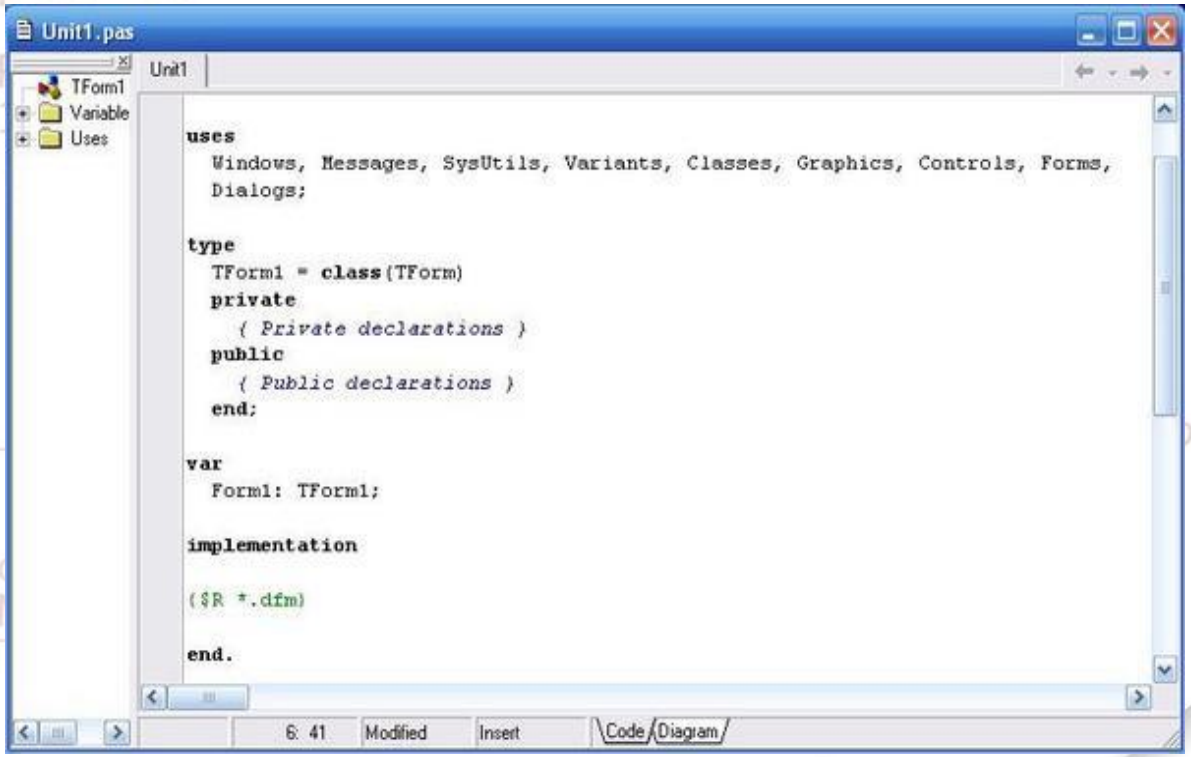
Главное окно содержит:

- строка заголовка;
- строка главного меню;
- панель инструментов;
- палитра компонентов.

Все визуальные компоненты отображаются на палитре компонентов ***Component Palett***, они разделены на группы ***Standart, Win32, System, Data Access*** и т.д. по своему назначению.

Основные элементы управления Windows можно выбрать из панели ***Standart*** (Базовые компоненты).

Все компоненты Delphi хранятся в библиотеке визуальных компонентов ***VCL*** (*Visual Componenet Library*). Каждый из компонентов, а также форма описаны соответствующими классами Паскаля: к названию компонента добавляется буква ***T***.



Например:

форма (***Form***) описывается классом ***TForm***;

компонент «кнопка» (***Button***) описывается классом ***TButton***.

Проект Delphi состоит из нескольких частей, которые размещены в файлах:


Project1.dpr

- *файл проекта* (*.dpr*) – содержит код проекта, для сохранения информации о формах и модулях;


Unit1.pas

- *файл модуля формы* (*.pas*) - содержит код модуля формы (именно файлы модулей доступны для редактирования в Редакторе Кода, остальные файлы проекта с расширением, отличающимся от .pas, редактировать не рекомендуется).


Unit1.dfm

- *файл описания формы* (*.dfm*) – содержит описание формы, включающее данные о ее свойствах, о помещенных в нее компонентах и т.п.;


Project1.dof

- *файл параметров проекта* (*.dof*) – для хранения установок параметров проекта, содержит параметры, устанавливаемые для проекта в пункте главного меню Project => Options;


Project1.res

- *файл ресурсов* (*.res*) – хранит используемую в проекте пиктограмму;
- *файлы резервных копий (.~dp, .~df, .~pa*) - файлы резервных копий файлов проектов, форм и программных модулей.
- *файл настроек проекта* (*.cfg*) - содержит данные о конфигурации проекта ;

Project1.cfg

- *файлы модулей* также имеют расширение *.pas* и являются обычными модулями, написанными на языке Pascal. Обычно они служат в качестве библиотек процедур и функций.


Unit1.~pas

Компилятором создается:


Project1.exe

- *исполняемый файл* (*.exe*) – (автономный, если не используются библиотеки DLL и т. д.);


Unit1.dcu

- *объектный файл модуля* (*.dcu*) – результат компиляции программных модулей (*.pas*) компонуется в исполняемый файл;
- *файл справки* (*.hlp*);


Icon.ico

- *файлы изображений* (*.wmf,.bmp,.ico*).

Адресация к полям объекта

Имя поля объекта является сложно-подчиненной структурой, которая состоит из:

1) *Имени класса;*
2) *Имени объекта;*
3) *Имени поля (или действия).*

например:

1. Имя процедуры:

TForm1.Button1Click

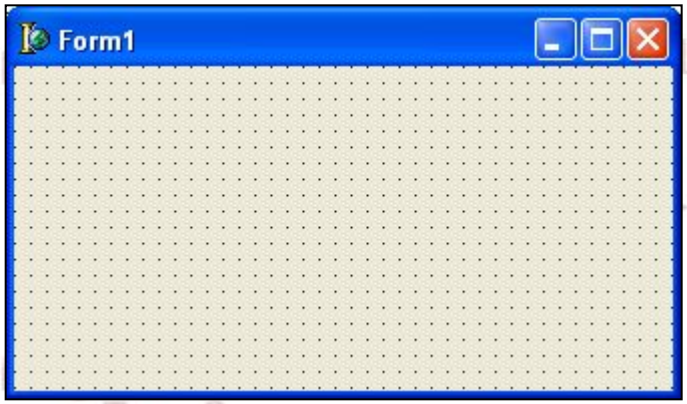
Имя класса *Имя объекта* *действие*

2. Обращение к имени поля заданного объекта:

Label1.Caption ***Button1.Caption***

Имя объекта *Имя поля* *Имя объекта* *Имя поля*

Изучение свойств формы



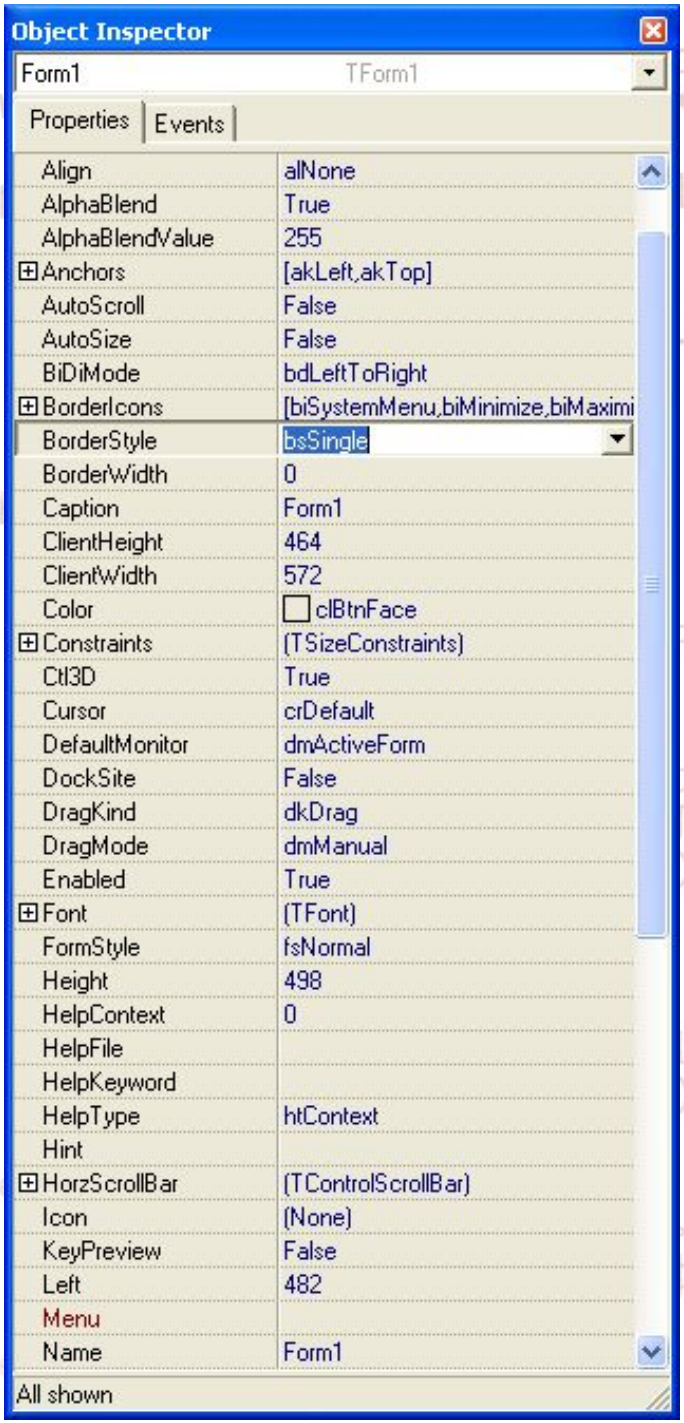
Компонент **Form** (Форма)

Форма *– это основа окна Windows-приложения, создаваемого в среде Delphi.*

На форме располагают все другие компоненты.

Создание нового проекта - команда *File\New\Application,* при этом создается пустая форма, она отображается в окне формы.

Форма – основной объект и, как и остальные объекты, имеет свойства, которые можно задавать и менять. Окно инспектора объектов ***Object Inspector*** отображает свойства этой формы, где описываются все необходимые свойства формы. Свойства могут быть установлены по умолчанию автоматически или настраиваться разработчиком самостоятельно.



Свойства компонента ***Form (Форма)***

Align (*выравнивание*): отвечает за способ выравнивания внутри контейнера. Возможные значения свойства: *alNone* (нет выравнивания), *alTop*, *alBottom*, *alLeft*, *alRight*, *alClient* (*выравнивание по верху, низу, левому, правому краю и во весь контейнер*).

Constraints (*ограничения*): ограничивает минимальный и максимальный размер компонента в случае масштабирования. Для этого необходимо задать *Constraints.MaxHeight*, *Constraints.MinHeight*, *Constraints.MaxWidth*, *Constraints.MinWidth* значениями, отличными от нуля.

AlphaBlend – прозрачность фона (False\True);

AlphaBlendValue – степень прозрачности (0-255);

AutoScroll – установка скроллинга прокрутки (False\True);

AutoSize – установка автоматического подбора размера (False\True);

BorderIcons – Вкл\Выкл системных кнопок;

BorderStyle – стиль оформления рамки;

Caption (заголовок):– строка текста, служащая заголовком или поясняющая назначение компонента.

Color – цвет; определяет цвет компонента. Цвета в Delphi можно задавать с помощью констант *clBlack*, *clNavy, clGreen, clTeal, clMaroon, clPurple, clOlive, clSilver, clGray, clBlue, cLime, clAqua, clRed, clFuchsia, clYellow, clWhite* и других.

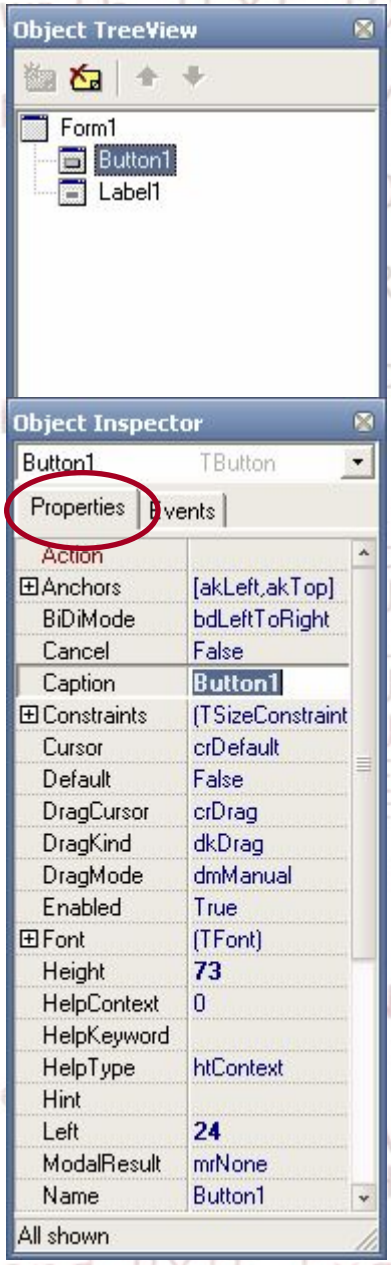
Ctl3D – установка объемного изображения (False\True);

Cursor (*курсор*): задает, как будет выглядеть указатель мыши, когда он проходит над компонентом. Возможные значения *crDeafult, crCross, crHandPoint* и другие.

Enabled (*доступен*): определяет доступность компонента (на недоступную кнопку нельзя нажать, в недоступное поле ввода нельзя ничего ввести и т.д.). Возможные значения: *true* или *false*;

Font – установка параметров шрифта;

Name – имя формы.

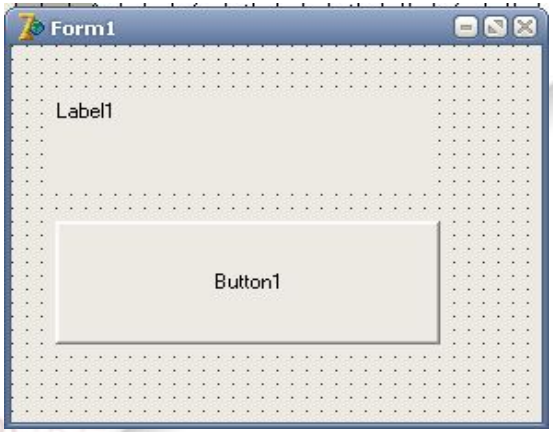


На форме можно располагать различные компоненты, которые размещены на панелях Standard, Additional,Win32, Sistem и др.

Компоненты располагаются методом методом drag and drop (перетащи и оставь) . Информация о размещенных компонентах на форме отображается в окне Object TreeView. С помощью информации этого окна можно активировать (выделять) ту или иную компоненту на форме. Во время активации компоненты меняется содержимое окна Object Inspector в соответствии с выбранным объектом, его свойствами.

В этом окне и можно менять свойства (вкладка Properties) и события (Events) компоненты.

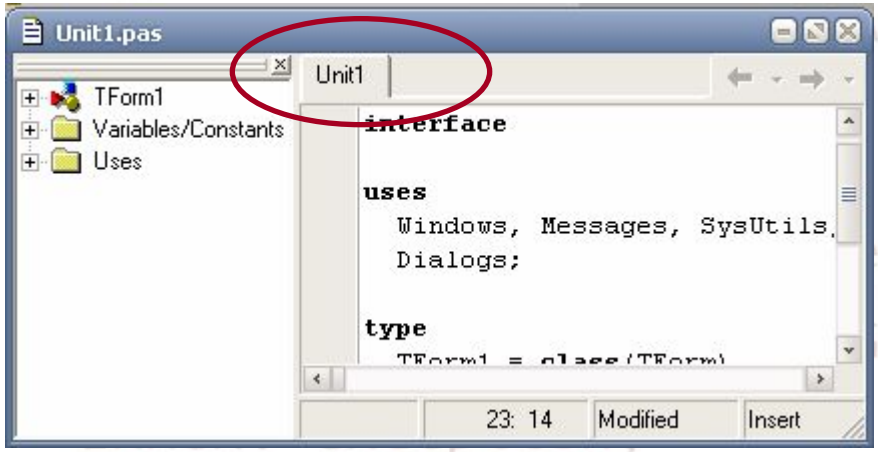
Свойства компоненты описываются указанием их возможных значений и списка.



Например:

Свойство *Caption* описывает заголовок компоненты и стандартно ему присваивается имя компоненты *Button1*, его можно изменить. (например «Нажми на меня!»)

Свойство *Color* определяет цвет компоненты, стандартно имеет значение цвета формы, можно по своему усмотрению из предлагаемого списка.



При размещении компо-ненты на форме и описании его свойств в окне Unit формируется программное описание формы и входящих в нее компо-нентов (объявляется класс Tform1 из стандартного класса TForm, в его состав входят объекты: Button1 из класса TButton, Label1 из класса TLabel1 и пр.)

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Variants,
Classes, Graphics, Controls, Forms,
  Dialogs;

type
  TForm1 = class(TForm)
    Button1: TButton;
    Label1: TLabel;

private
  { Private declarations }
public
  { Public declarations }
end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

end.
```

Каждой компоненте можно описать событие. Событие можно определить по нажатию кнопкой мыши на компоненту (OnClick), по наведению (OnMouseMove), по нажатию кнопки Enter (OnEdit) и др. Это определяется в окне Events (двойным щелком в соответствующем поле). Одновременно с этим в окне Unit появляется описание процедуры в которой и можно записывать программу обработки события.

Процедура для описания события по нажатию (Button1Click) на кнопку

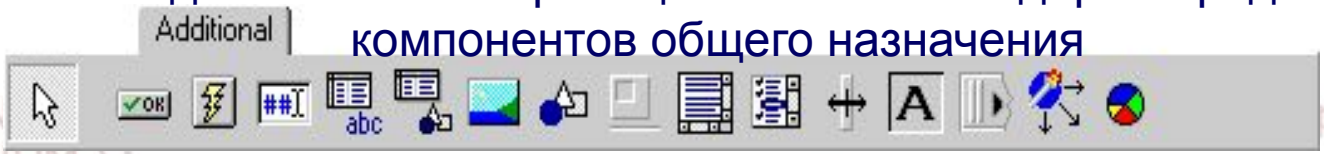
```
procedure TForm1.Button1Click(Sender: TObject);
begin

end;

procedure TForm1.Label1MouseMove(Sender: TObject; Shift: TShiftState);
begin

end;
```

Процедура для описания события по движению мыши (Label1MouseMove) на текстовое поле

Визуальные компоненты среды Delphi можно выбрать из вкладок, которые содержат группы компонентов.

Панель ***Standard*** содержит ряд часто используемых компонентов общего назначения

Панель ***Additional*** является дополнением страницы Standard и содержит ряд часто используемых компонентов общего назначения



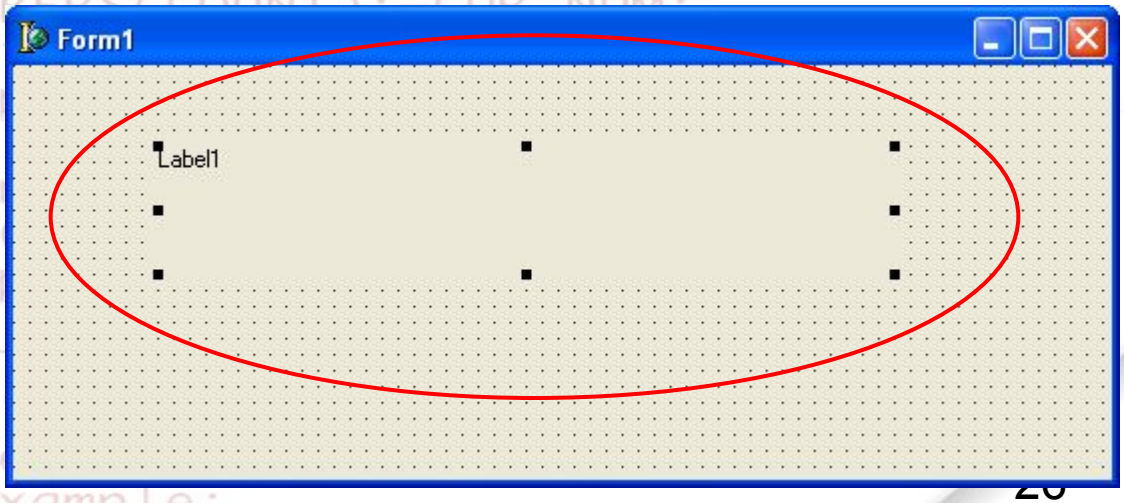
кнопка	***Компонент***	***Тип***	***Описание***
A	*метка*	***Label***	Используется для размещения на формах и других контейнерах текста, который не изменяется пользователем.
ab	*окно редактирования*	***Edit***	Используется для ввода пользователем однострочных текстов. Может использоваться для отображения текста.
	многострочное окно редактирования	***Memo***	Используется для ввода и отображения многострочных текстов.

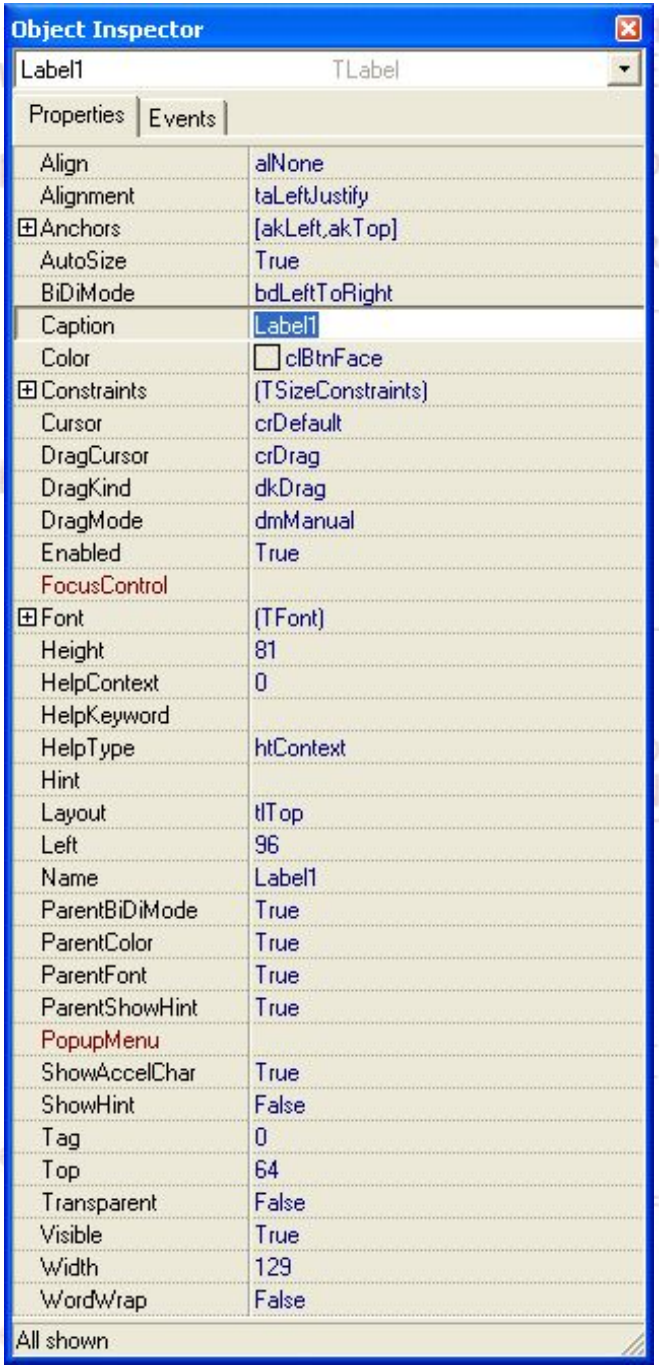
кнопка	Компонент	Тип	Описание
	*командная кнопка*	***Button***	Используется для создания кнопок, которыми пользователь выбирает команды в приложении.
	*изображение*	***Image***	Используется для отображения графики
	*Список строк*	***ListBox***	Компонента используется для формирования списка ввода и выбора строки из нее.
	*Выпадающий список*	***ComboBox***	Компонента используется для формирования выпадающего списка и выбора строки из нее.
	*индикатор с флажком*	***Checkbox***	Позволяет пользователю включать и выключать различные опции.
	*радиокнопка*	***RadioButton***	Предлагают пользователю набор альтернатив, из которых выбирается одна.
	*таблица строк*	***StringGrid***	Используется для отображения текстовой информации в таблице из строк и столбцов.

*Компонент **Label** (Текстовое поле)*

Label (Текстовое поле) – это объект, с помощью которого на форму можно нанести разнообразные надписи, тексты.

Объект ***Label***относится к классу ***TLabel*** . Ее компонента относится к разряду визуальных и расположена на панели ***Standart***.





Свойства компоненты ***Label***

Aligment – выравнивание текста внутри компоненты;

Align – выравнивание;

AutoSize – установка автоматического подбора размера (False\True);

Caption – заголовок;

Color – цвет;

Cursor – установка вида курсора;

Enabled – установка активной надписи (False\True);

Font – установка параметров шрифта;

Name – имя текстового поля;

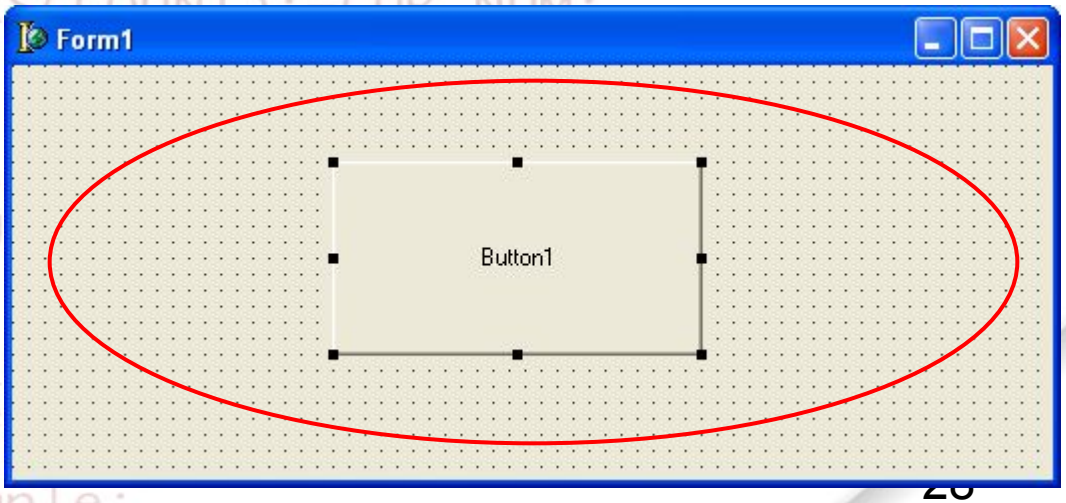
Height – высота текстового поля;

Width– ширина текстового поля.

*Компонент **Button** (Кнопка)*

Объект ***Button*** (Кнопка) предназначен для управления процессами, которые происходят на форме.

Объект ***Button*** относится к классу ***TButton***. Ее компонента относится к разряду ***визуальных*** и расположена на панели ***Standart***.



Свойства компоненты ***Button***

Caption – заголовок;

Cursor – установка вида курсора;

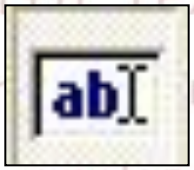
Enabled – установка активной кнопки;

Font – установка параметров шрифта;

Name – имя кнопки;

Height – высота кнопки;

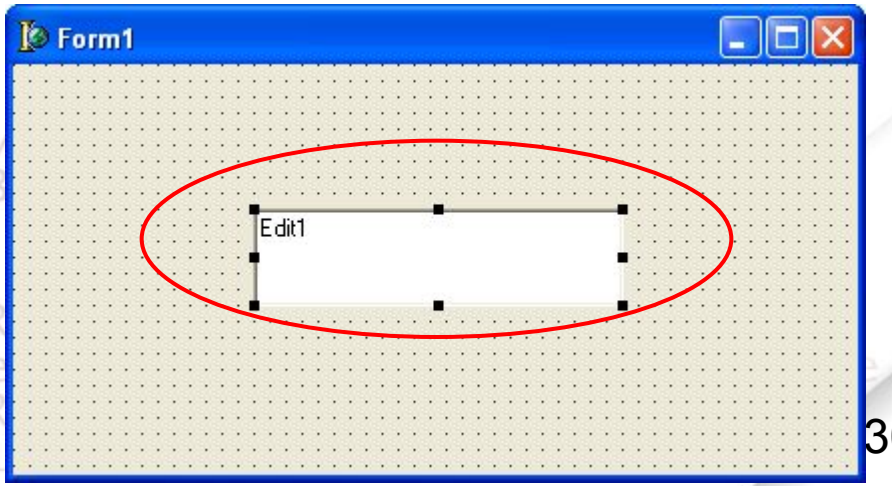
Width– ширина кнопки.

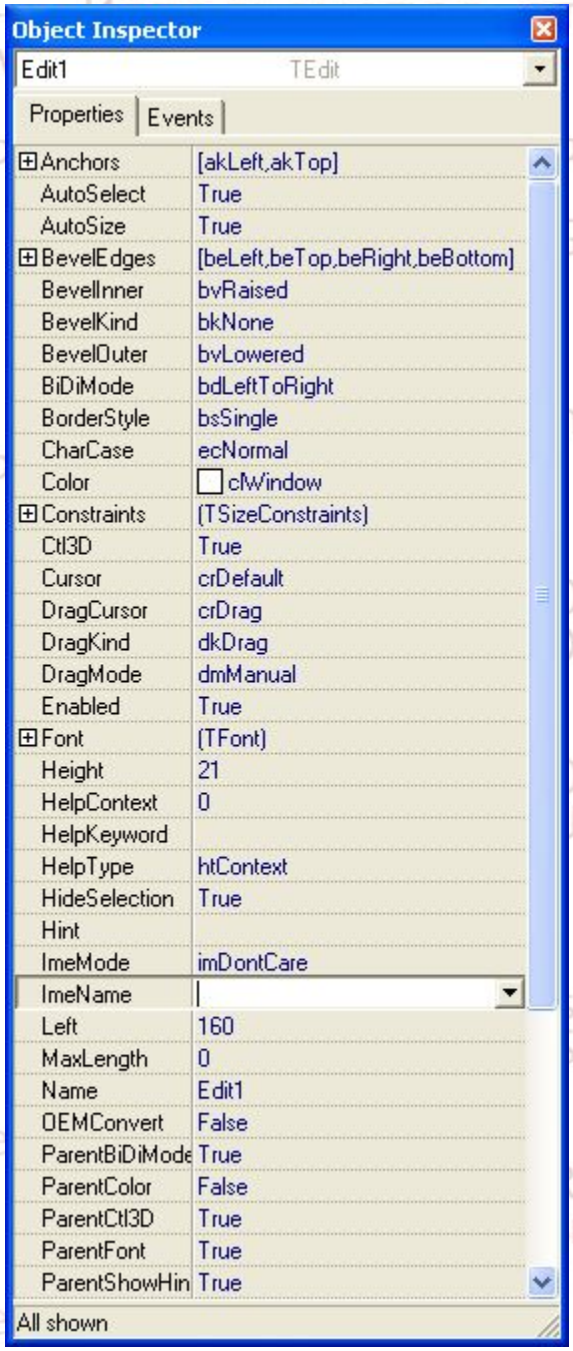


*Компонент **Edit** (поле редактирования, однострочное поле ввода)*

Объект ***Edit*** (поле редактирования) предназначен для введения, отображения и редактирования одной текстовой строки, длинной до *256* символов.

Объект ***Edit*** относится к классу ***TEdit***. Ее компонента относится к разряду ***визуальных*** и расположена на панели ***Standart***.





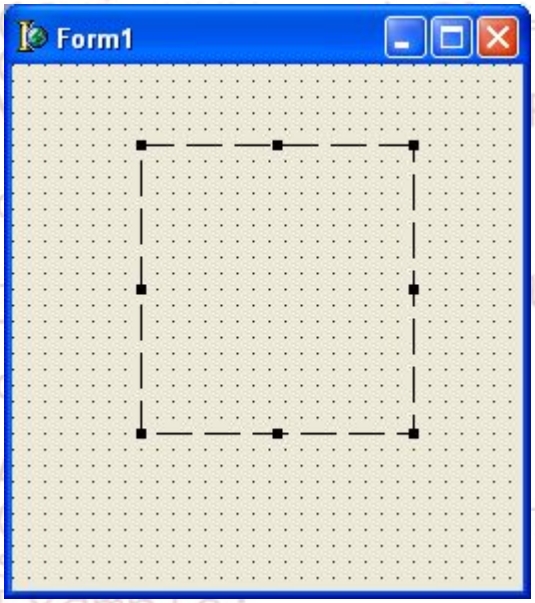
*Свойства компоненты **Edit***

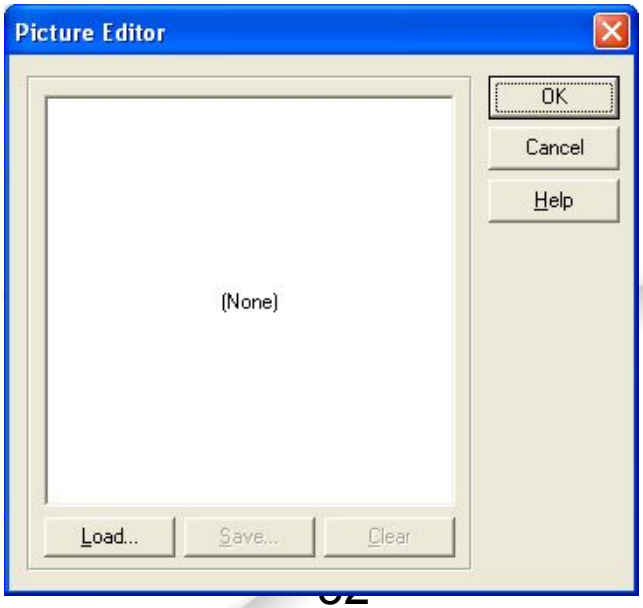
AutoSize – установка автоматического подбора размера (False\True);
Color – цвет;
Ctl3D – установка объемного изображения (False\True);
Cursor – установка вида курсора;
Enabled – установка активной компоненты (False\True);
Font – установка параметров шрифта;
Name – имя компоненты;
Text – текст внутри поля;
Height – высота поля;
Width– ширина поля.

*Компонент **Image** (картинка)*

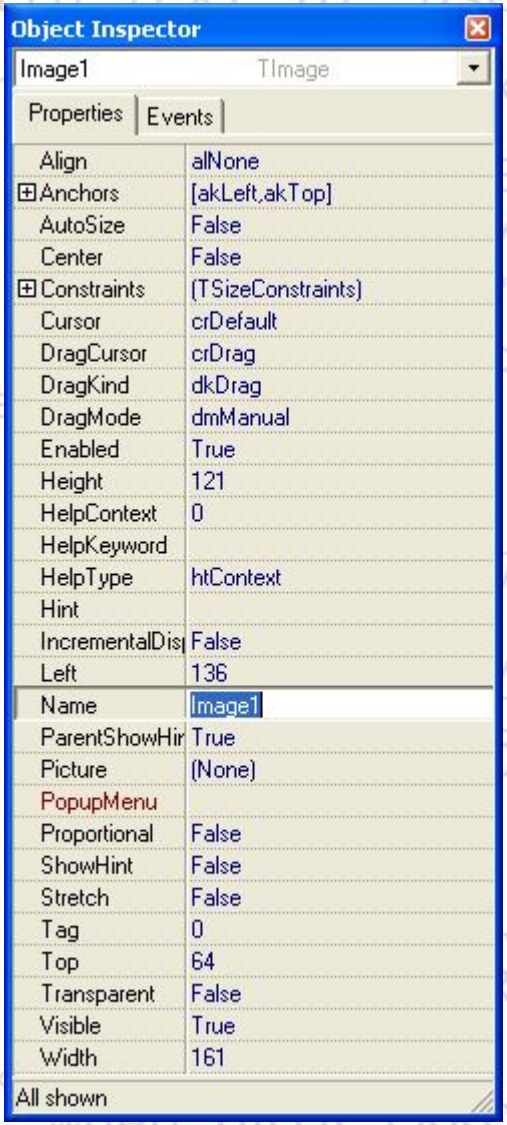
Объект ***Image*** (*Картинка*) предназначен для вставки картинок в форму.

Объект ***Image*** относится к классу ***TImage***. Ее компонента относится к разряду ***визуальных*** и расположена на панели ***Additional***.





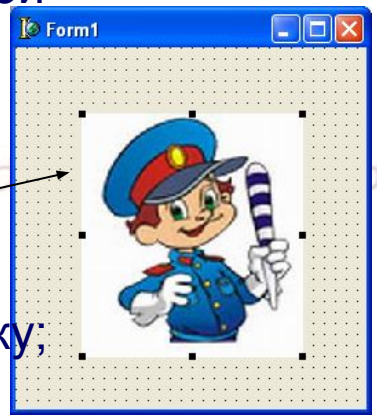
Свойства компоненты ***Image***

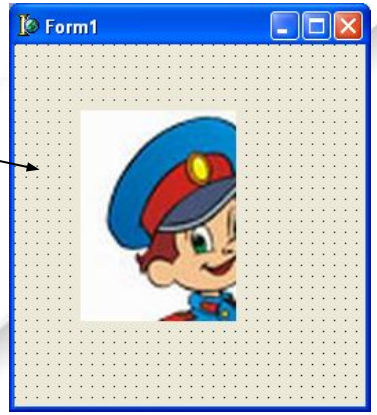


Align – расположение, выравнивание;
AutoSize – установка автоматического подбора размера (False\True);
Cursor – установка вида курсора;
Enabled – установка активной компоненты (False\True);
Name – имя компоненты;
Picture – выбор рисунка;
Stretch – отображение рисунка внутри рамки:

- ***True*** - рисунок вписывается в рамку;
- ***False*** - рисунок сохраняет свои размеры;

Height – высота поля;
Width– ширина поля.

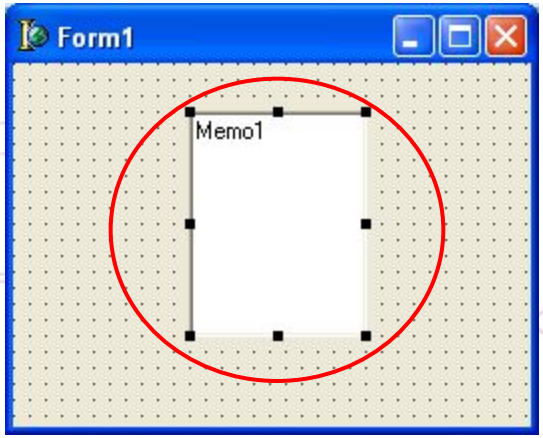


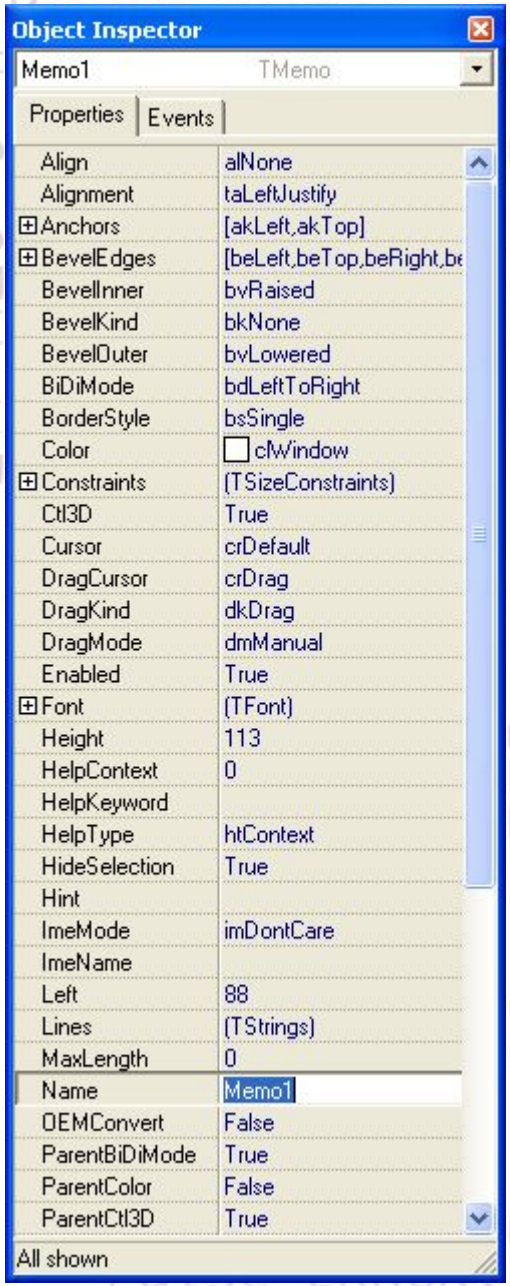


*Компонент **Memo** (поле для ввода/вывода длинного многострочного текста)*

Объект ***Memo*** (*поле многострочного текста*) предназначен для введения, отображения и редактирования текста, длинной до *64 000* символов.

Объект ***Memo*** относится к классу ***TMemo***. Ее компонента относится к разряду ***визуальных*** и расположена на панели ***Standart***.

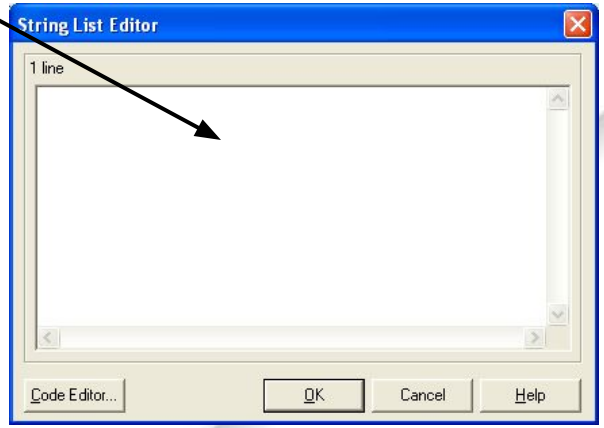




*Свойства компоненты **Memo***

Align – расположение, выравнивание;
Cursor – установка вида курсора;
Enabled – установка активной компоненты (False\True);
Name – имя компоненты;

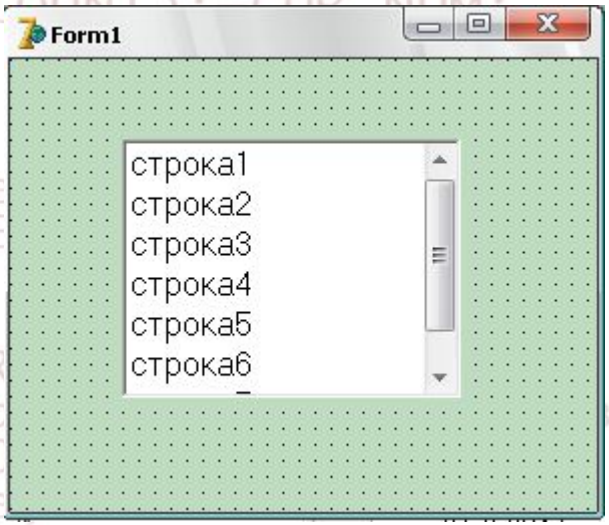
Lines – запуск окна для ввода и редактирования текста;



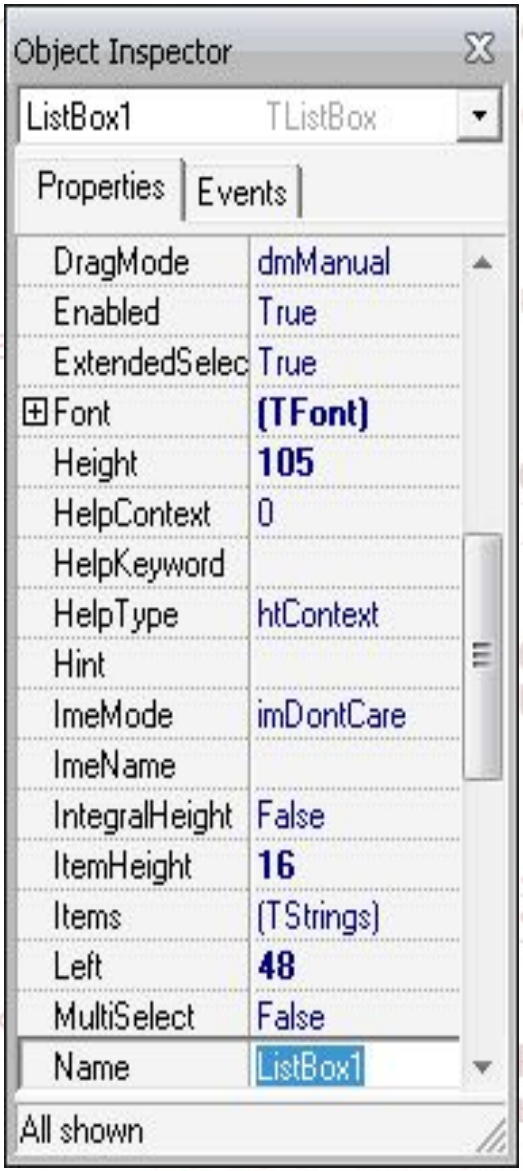
*Компонент **ListBox** (Список строк)*

Объект ***ListBox*** (*список строк*) предназначен для выбора строки из списка.

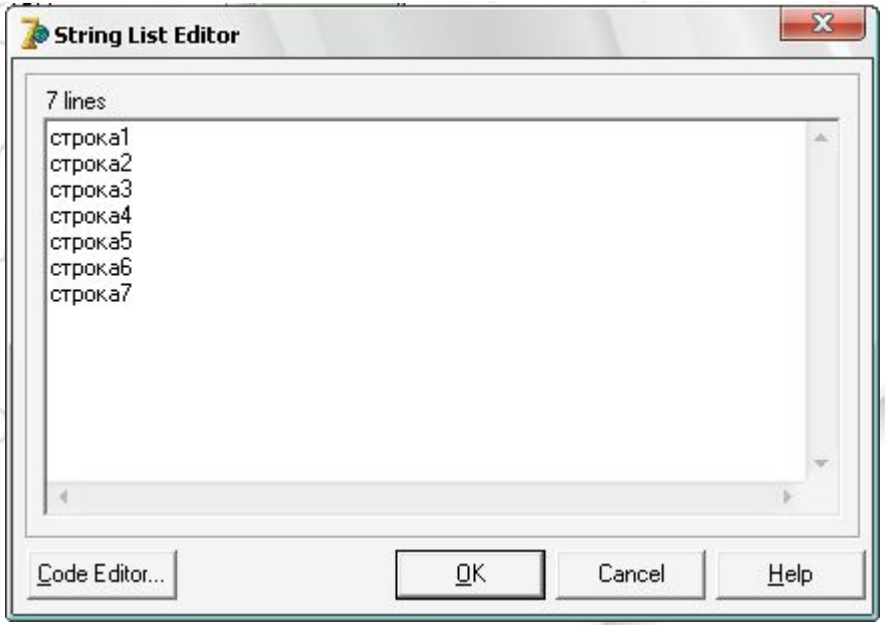
Объект ***ListBox*** относится к классу ***TListBox***. Ее компонента относится к разряду ***визуальных*** и расположена на панели ***Standart***.



Свойства компоненты ***ListBox***



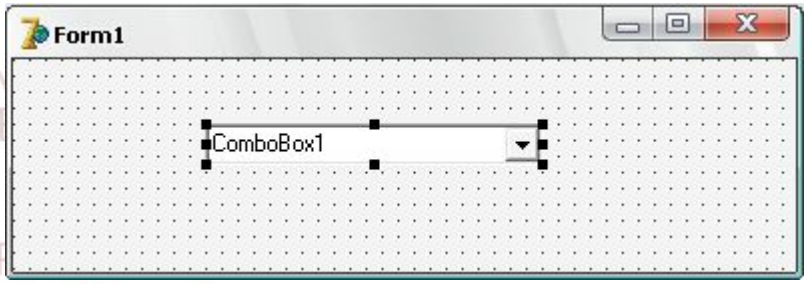
Items – запуск окна для ввода и редактирования списка строк;



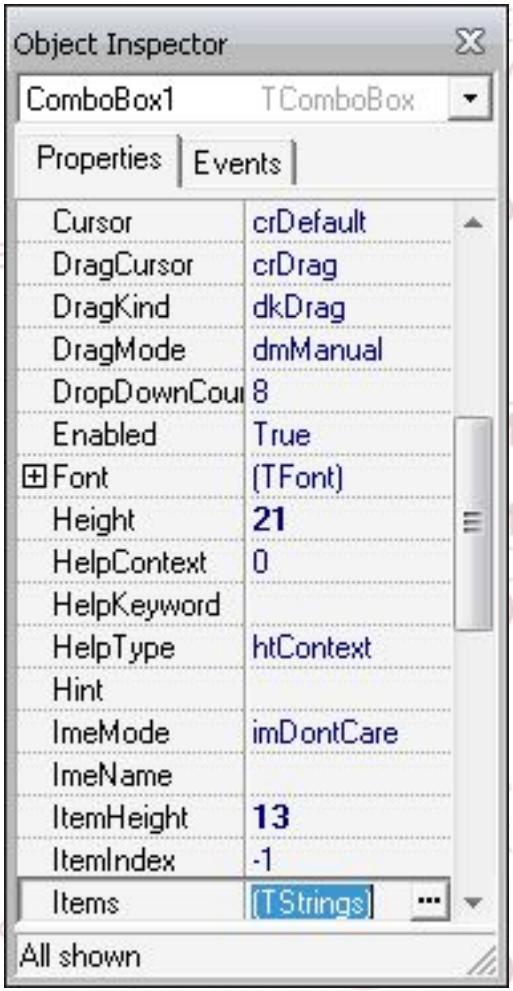
*Компонент **ComboBox** (Выпадающий список)*

Объект ***ComboBox*** (выпадающий *список*) предназначен для выбора строки из выпадающего списка.

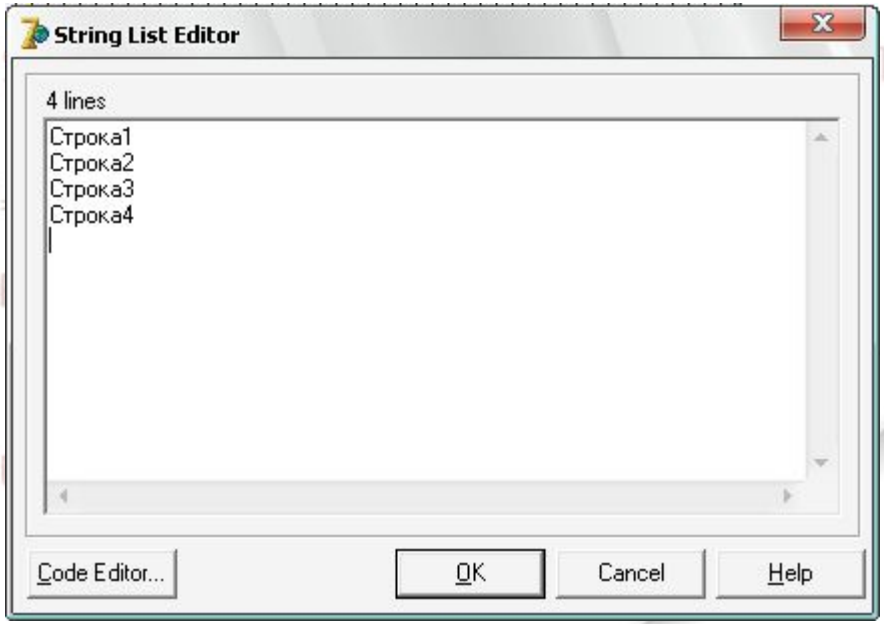
Объект ***ComboBox*** относится к классу ***TComboBox***. Ее компонента относится к разряду ***визуальных*** и расположена на панели ***Standart***.

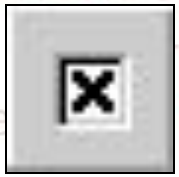


Свойства компоненты ***ListBox***



Items – запуск окна для ввода и редактирования списка;





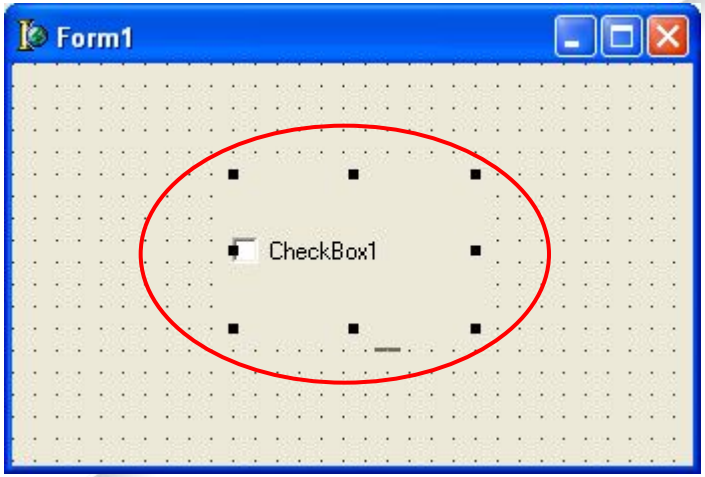
*Компонент **CheckBox** (индикатор с флажком)*

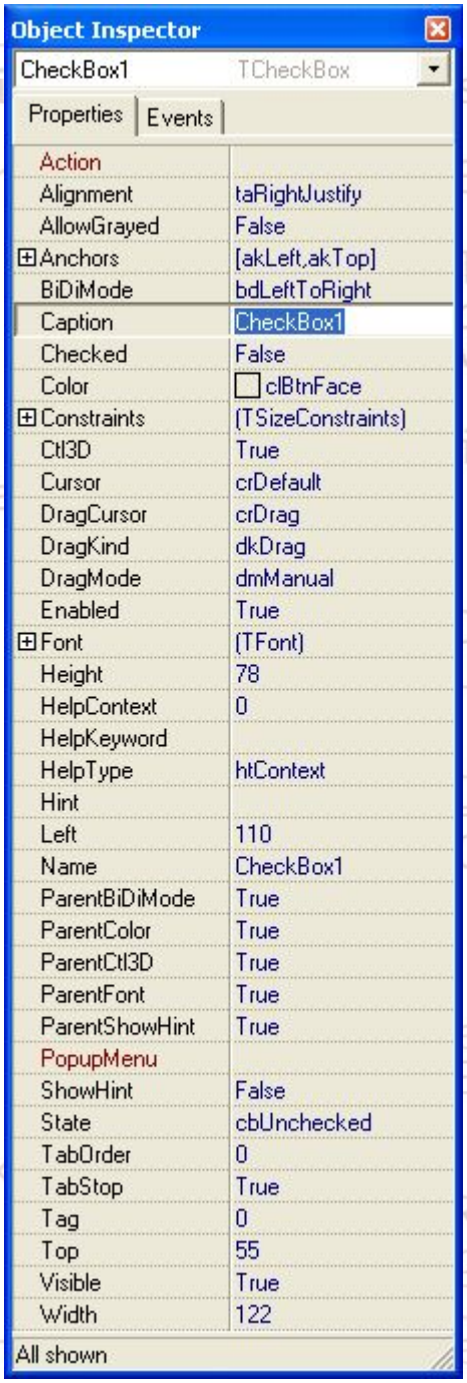
Объект ***CheckBox*** (индикатор с флажком) используются в приложениях для того, чтобы пользователь мог включать и выключать какие-то опции, или для индикации состояния.

При каждом щелчке пользователя на индикаторе его состояние может принимать ***три значения***:

- *выделение* (появление черной галочки),
- *не выделенное* (пустое окно индикатора),
- *промежуточное* (серое окно индикатора и серая галочка).

Объект ***CheckBox*** относится к классу ***TCheckBox***. Ее компонента относится к разряду ***визуальных*** и расположена на панели ***Standart***.





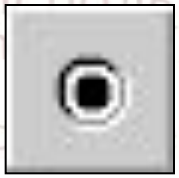
Свойства компоненты ***CheckBox***

Caption – заголовок;
Color – цвет;
Ctl3D – установка объемного изображения (False\True);
Cursor – установка вида курсора;
Enabled – установка активной компоненты (False\True);
Font – установка параметров шрифта;
Name – имя компоненты;
Height – высота компоненты;
Width– ширина компоненты.

State - выбор состояния компонента:

- *cbChecked* - выделенное,
- *cbGrayed* - промежуточное,
- *cbUnchecked* - не выделенное.

Эти три состояния допускаются когда свойство *AllowGrayed = true*. Если же *AllowGrayed = false* (значение по умолчанию), то допускается только два состояния: *выделенное* и не *выделенное*.

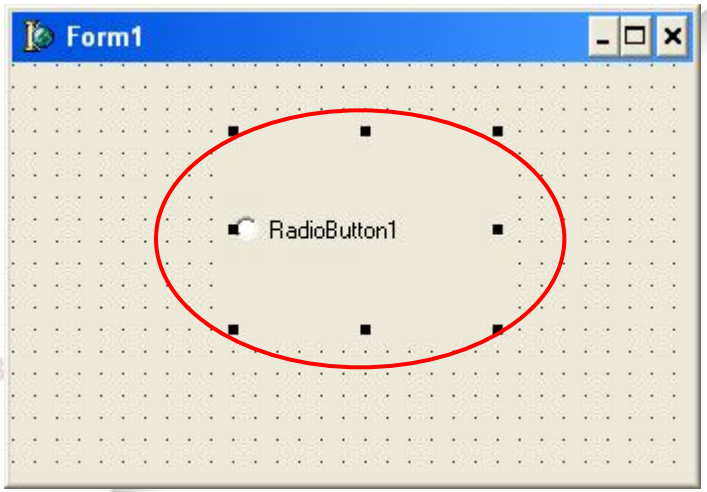


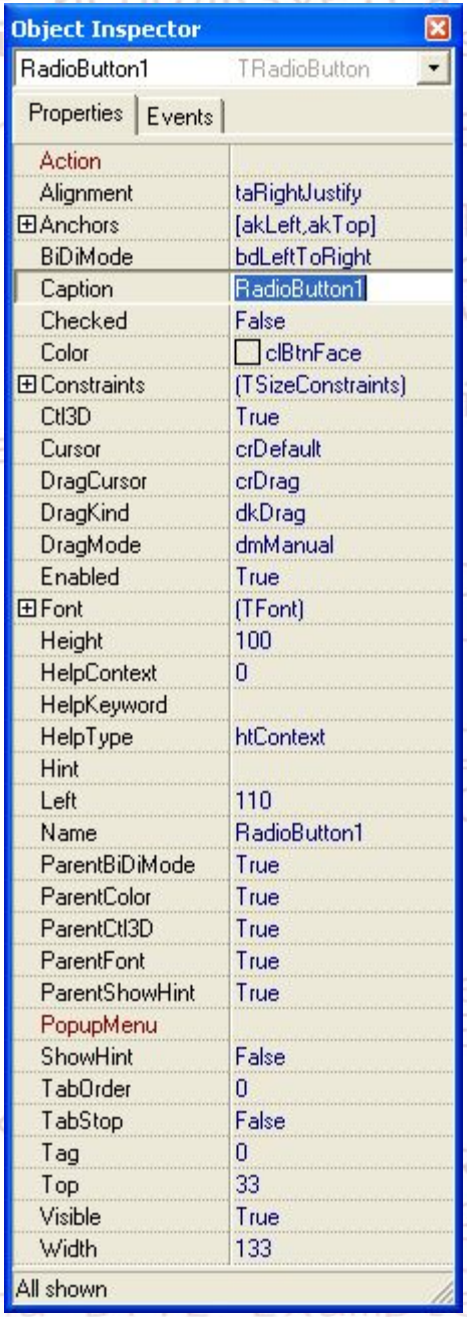
*Компонент **RadioButton** (радиокнопка)*

Объект ***RadioButton*** (радиокнопка) используется для выбора пользователем одной из нескольких взаимоисключающих альтернатив.

Радиокнопки образуют группы взаимосвязанных индикаторов, из которых обычно ***может быть выбран только один***.

Объект ***RadioButton*** относится к классу ***TRadioButton***. Ее компонента относится к разряду ***визуальных*** и расположена на панели ***Standart***.





Свойства компоненты ***RadioButton***

Alignment - определяет, с какой стороны от кнопки появится надпись;

Caption – надпись, появляющаяся около кнопки;

Color – цвет;

Checked - определяет, выбрана (True) ли данная кнопка пользователем, или нет (False); *можно установить* ***true*** *в значение* ***Checked*** *только у одной кнопки из группы!*

Cursor – установка вида курсора;

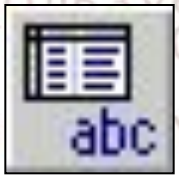
Enabled – установка активной компоненты (False\True);

Font – установка параметров шрифта;

Name – имя компоненты;

Height – высота компоненты ;

Width– ширина компоненты .

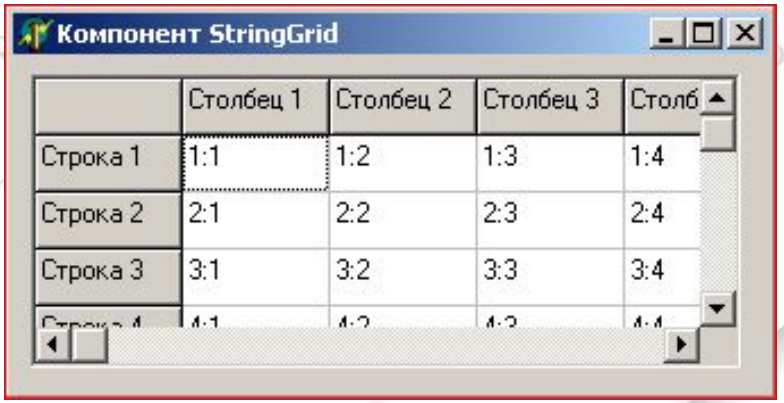


Компонент **StringGrid** (таблица строк)

Объект ***StringGrid*** (таблица строк) используется для отображения текстовой информации в таблице из строк и столбцов. Данные таблицы могут быть *только для чтения* или *редактируемыми*.

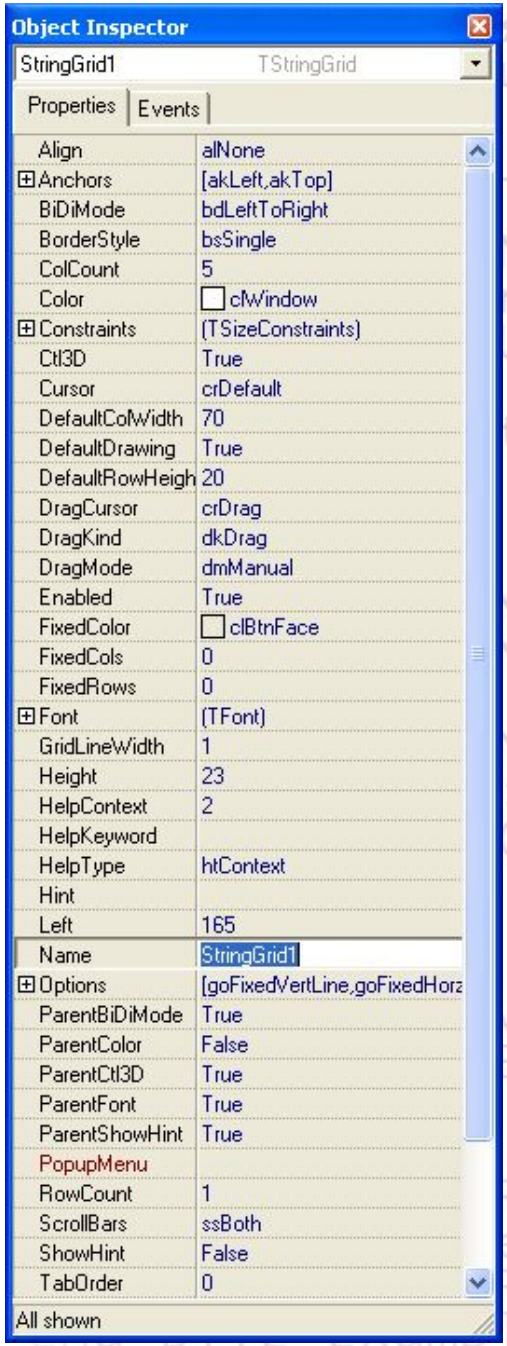
Таблица может иметь полосы прокрутки, причем заданное число первых строк и столбцов может быть фиксированным и не прокручиваться.

Можно задать заголовки столбцов и строк, постоянно присутствующие в окне компонента. Каждой ячейке таблицы может быть поставлен в соответствие некоторый объект.



Объект ***StringGrid*** относится к классу ***TStringGrid***. Ее компонента относится к разряду ***визуальных*** и расположена на панели ***Additional***.

Свойства компоненты **StringGrid**



ColCount - определяет число столбцов;

RowCount - определяет число строк;

Enabled – установка активной компоненты (False\True);

FixedColor - цвет фона фиксированных ячеек;

FixedCols - количество зафиксированных слева столбцов таблицы. Зафиксированные столбцы выделяются цветом и при горизонтальной прокрутке таблицы остаются на месте ;

FixedRows - количество зафиксированных сверху строк таблицы. Зафиксированные строки выделяются цветом и при вертикальной прокрутке таблицы остаются на месте;

ScrollBars - наличие в таблице полос прокрутки;

Options - множество, определяющее свойства таблицы:

goEditing - признак допустимости редактирования содержимого ячеек таблицы. *True* — редактирование разрешено, *False* — запрещено;

goFixedVertLine - наличие разделительных вертикальных линий в фиксированных ячейках;

goFixedHorzLine - наличие разделительных горизонтальных линий в фиксированных ячейках;

goVertLine - наличие разделительных вертикальных линий в не фиксированных ячейках;

goHorzLine - наличие разделительных горизонтальных линий в не фиксированных ячейках;

goColSizing - возможность изменять с помощью мыши размеры столбцов;

goRowSizing - возможность изменять с помощью мыши размеры строк и др.

Раздел 1

Раздел 2

Раздел 3

Раздел 4

Раздел 5

Раздел 6

Раздел 7

Раздел 8

Раздел 9

Раздел 10

Раздел 11

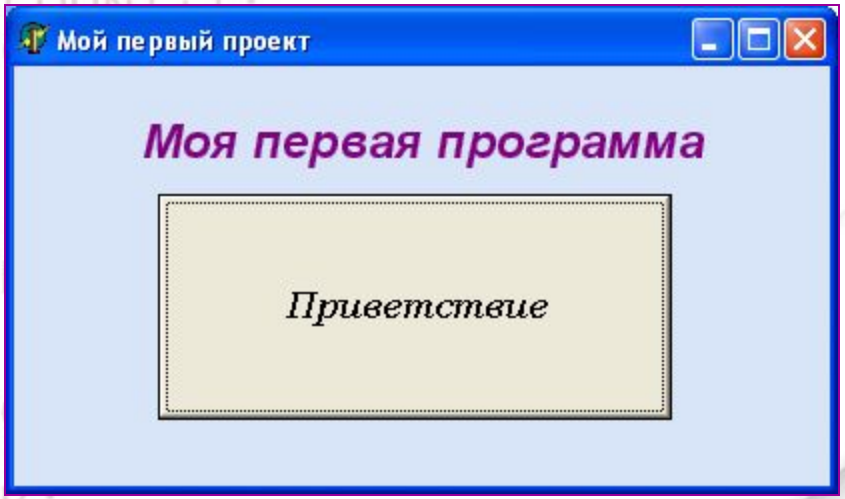
Раздел 12

Выход

Создание проекта №1, используя компоненты вывода. Линейные алгоритмы.

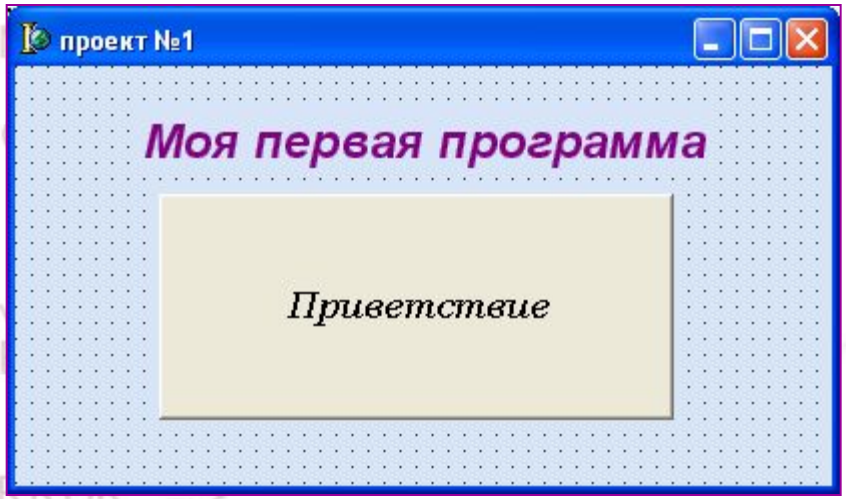
1 этап. Формулировка задания, для которого реализуется проект

Создать форму, в которой по нажатию кнопки должна появиться надпись с приветствием.



2 этап. *Проведение проектного анализа и формирование требований к объектам*

1) Необходимо разработать ***форму*** на которой будет размещен ***текст*** и ***кнопка***.
2) Разработать модуль, который обрабатывает событие: *по нажатию на кнопку на ней появляется текст с приветствием.*



Form (форма)

Label (текст)

Button (кнопка)

3 этап. Выбор необходимых компонентов и разработка алгоритмов обработки компонентов

Установим значения свойств компонент и опишем порядок выполнения событий для проекта

компонент	*Object Inspector*	*Свойство (Properties)\ Событие (Events)*	*Значение свойства\ Обработка события*
Form1	**Properties**	**Name (имя формы)**	***F1***
		Caption (заголовок)	***проект №1***
		Color (цвет формы)	***По выбору***
Label1	**Properties**	**Caption (надпись)**	***Моя первая программа***
		Color (цвет текстового поля)	***По выбору***
		Font (установка параметров шрифта)	***По выбору***
Button1	**Properties**	**Caption (надпись на кнопке)**	***Приветствие***
		Cursor (установка вида курсора)	***crHandPoint***
		Font (установка параметров шрифта)	***По выбору***
	Events	**OnClick**	***Button1.Caption:='Привет!';***

Написание модуля.

1) Активизируйте двойным щелком компоненту ***кнопка***. При этом открывается окно ***редактора кода*** (написание программы) ***Unit.pas*** на котором автоматически записывается процедура с именем компоненты .***Button1*** из класса ***Tf1 - Tf1.Button1Click***, эта процедура пока еще пустая, в ней можно записывать алгоритм обработки ситуации.

```
procedure Tf1.Button1Click(Sender: TObject);
Begin

end;
```

2) Запишем следующее:

```
procedure Tf1.Button1Click(Sender: TObject);
begin
 Button1.Caption:='Привет!';
 Beep;
end;
```

3) Добавьте в процедуру строку ***Button1.Top:=10;*** - это означает указание свойства ***Top*** значение ***10*** (кнопка переместится на позицию 10).
4) Выполните программу (***F9***).

Практическая работа №1_1

Постановка задачи

Создайте форму *Моя визитная карточка*, которая по нажатию кнопки высвечивала бы на ней вашу *Фамилию*. Дополните программу несколькими кнопками: *Имя*, *Отчество*, *Школа*, *Класс*, *Выход*.

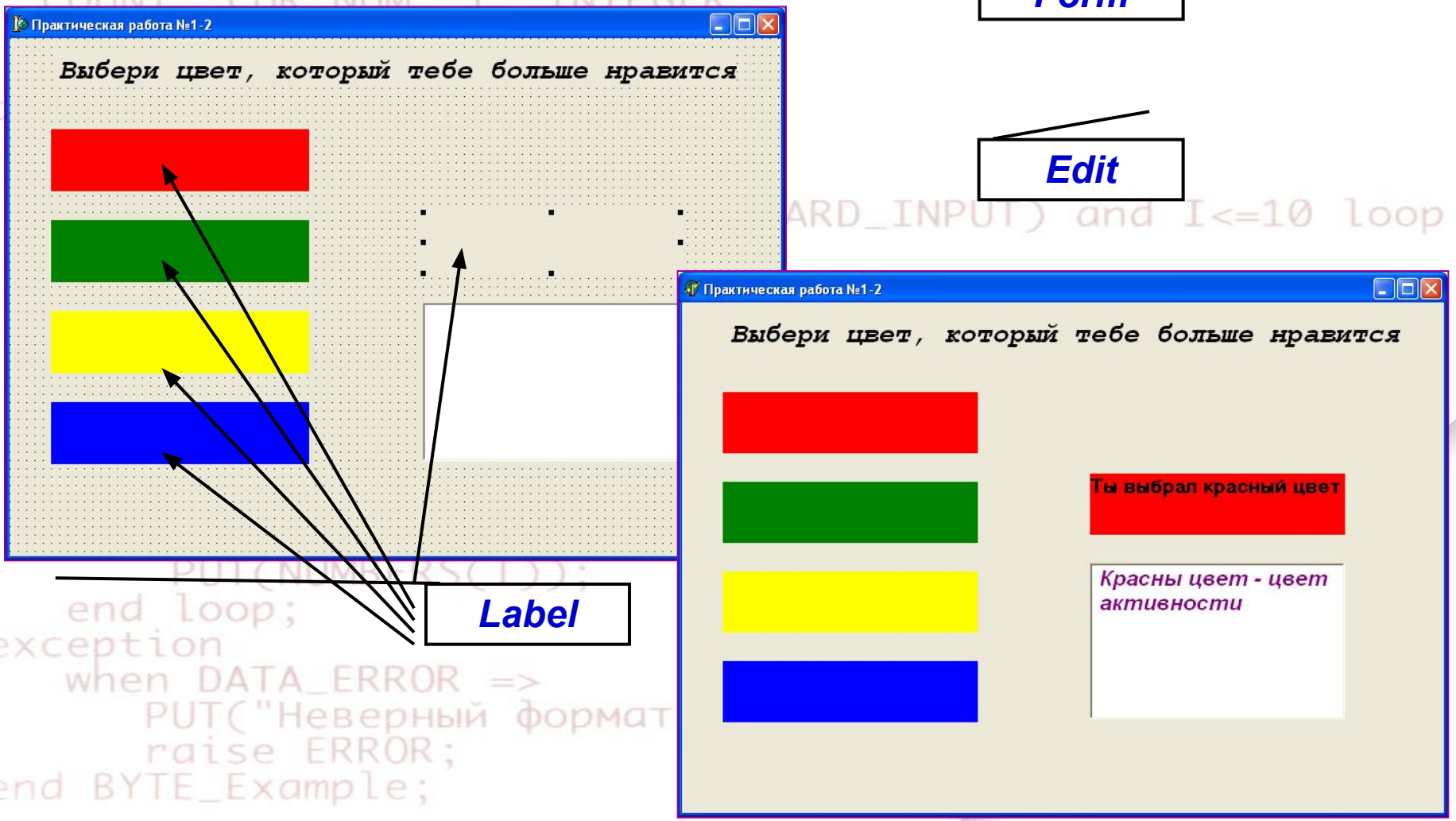
Установим значения свойств компонент и опишем порядок выполнения событий для проекта

компонент	*Object Inspector*	*Свойство (Properties)\ Событие (Events)*	*Значение свойства\ Обработка события*
Form1	**Properties**	**Name**	***F1***
		Caption	***Практическая работа №1_1***
		Color	***clSkyBlue***
Label1	**Properties**	**Caption**	***Моя визитная карточка***
		Color	***clMoneyGreen***
		Font	***Tahoma, жирный курсив, 16 пт, красного цвета***
Button1	**Properties**	**Caption**	***Фамилия***
		Cursor	***crHandPoint***
		Font	***Arial, курсив, 14 пт***
	Events	**OnClick**	***Button1.Caption:='{Ваша фамилия}';***
Button2	**Properties**	**Caption**	***Имя***
		Cursor	***crHourGlass***
		Font	***Arial, курсив, 14 пт***
	Events	**OnClick**	***Button2.Caption:='{Ваше имя}';***

компонент	Object Inspector	Свойство (Properties)\ Событие (Events)	Значение свойства\ Обработка события
Button3	Properties	Caption	Отчество
		Cursor	crHandPoint
		Font	Arial, курсив, 14 пт
	Events	OnClick	Button3.Caption:='{Ваше отчество}';
Button4	Properties	Caption	Школа
		Cursor	crHandPoint
		Font	Courier New, жирный, 16 пт
	Events	OnClick	Button4.Caption:='{номер вашей школы}';
Button5	Properties	Caption	Класс
		Cursor	crHourGlass
		Font	Courier New, жирный, 16 пт
	Events	OnClick	Button5.Caption:='{номер вашего класса}';
Button6	Properties	Caption	Выход
		Cursor	crAppStart
		Font	Tahoma, жирный курсив, 14 пт
	Events	OnClick	Close;

Практическая работа №1-2

Постановка задачи: по выбору цвета вывести характеристику человека.



Установим значения свойств компонент и опишем порядок выполнения событий для проекта

компонент	*Object Inspector*	*Свойство (Properties)\ Событие (Events)*	*Значение свойства\ Обработка события*
Form1	Properties	Name	*F1_2*
		Caption	*Практическая работа №1_2*
		Color	*По выбору*
Label1	Properties	Caption	*Выбери цвет, который тебе больше нравится*
		Font	*Courier New, жирный курсив, 20 пт*
Label2	Properties	Color	*clRed*
		Cursor	*crHandPoint*
	Events	OnClick	*label6.Caption:='Ты выбрал красный цвет';* *label6.Color:=clRed;* *Memo1.Text:='Красны цвет - цвет активности';*
Label3	Properties	Color	*clGreen*
		Cursor	*crHandPoint*
	Events	OnClick	*label6.Caption:='Ты выбрал зеленый цвет';* *label6.Color:=clGreen;* *Memo1.Text:='Зеленый цвет - цвет уравновешенности';*

компонент	Object Inspector	Свойство (Properties)\ Событие (Events)	Значение свойства\ Обработка события
Label4	Properties	Color	clYellow
		Cursor	crHandPoint
	Events	OnClick	label6.Caption:='Ты выбрал желтый цвет'; label6.Color:=clYellow; Memo1.Text:='Желтый цвет - цвет устремленности';
Label5	Properties	Color	clBlue
		Cursor	crHandPoint
	Events	OnClick	Label6.Caption:='Ты выбрал синий цвет'; Label6.Color:=clBlue; Memo1.Text:='Синий цвет - цвет спокойствия';
Label6	Properties	Caption	<пусто>
Memo1	Properties	Enabled	False

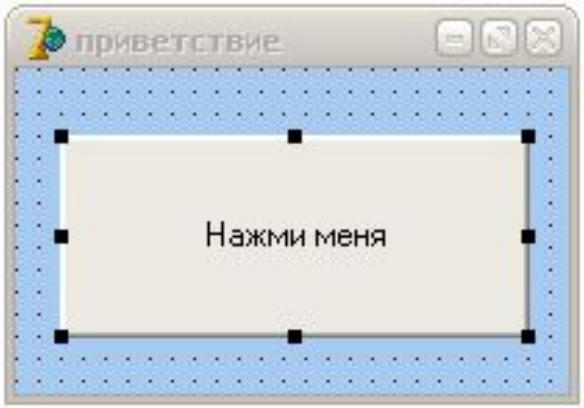
Практическая работа №1-3

Постановка задачи: по нажатию на кнопку формы появляется окно сообщений.

На форме необходимо расположить объект Button.

Для вызова окна сообщений используется команда ShowMessage(‘сообщение’);

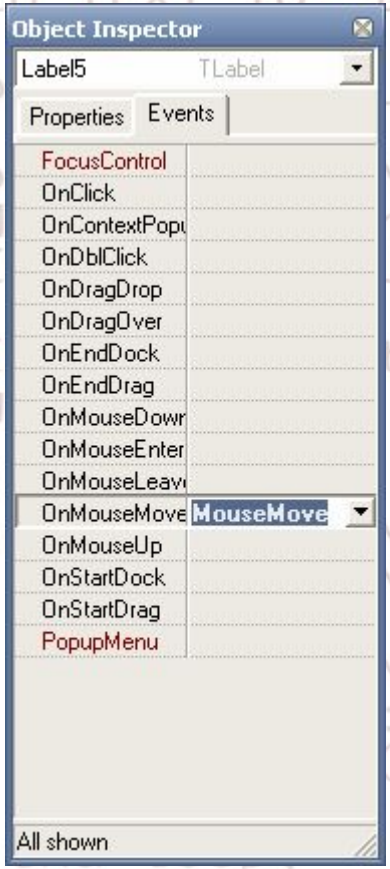
```
procedure TForm1.Button1Click(Sender: TObject);
begin
Showmessage('Здравствуте');
end;

end.
```



Практическая работа №1-4

Постановка задачи: на форме разместить несколько названий цветов. По наведению на название цветка, рядом отображать картинку с его изображением.

На форме необходимо расположить объекты Label1-4 с названием цветов, рядом расположить объект Image1-4 с соответствующими рисунками. Чтобы картинки первоначально не отображать, объектам Image1-4 необходимо установить параметру Visible (отображать) значение False.



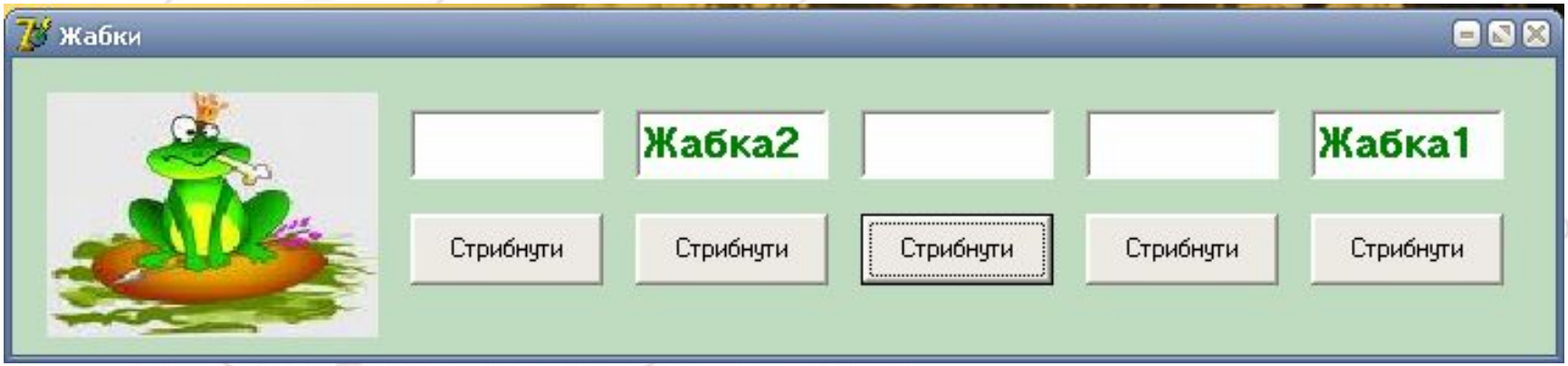
Каждой компоненте Label необходимо описать событие по наведению мыши, для этого в окне Events выполнить двойной щелк мышью в поле OnMouseMove.

В окне Unit нужно менять значения свойства Visible (True, False) соответствующих объектов Image.

```
procedure TForm1.Label2MouseMove(Sender:
TObject; Shift: TShiftState; X,
  Y: Integer);
begin
   Image1.Visible:=true;
   Image2.Visible:=false;
   Image3.Visible:=false;
   Image4.Visible:=false;
end;
```

Практическая работа №1-5

Постановка задачи: Имитировать прыжки двух жабок. Считая, что они прыгают через одну клеточку.



Для решения этой задачи необходимо присваивать компоненте Edit значение параметра Text той, компоненты с которой жабка делает прыжок.

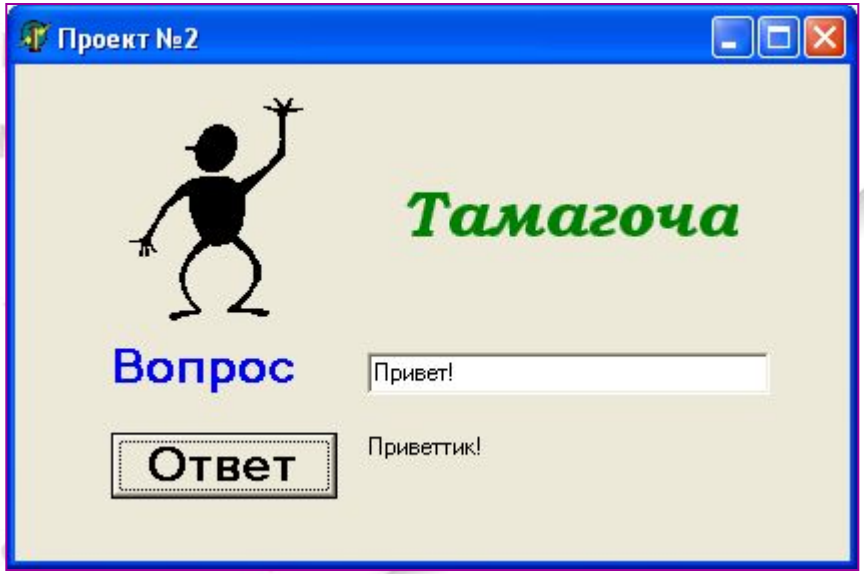
```
procedure TForm1.Button1Click(Sender:
TObject);
begin
Edit3.Text:=Edit1.Text;
Edit1.Text:='';
end;
```

Создание проекта № 2, используя компоненты ввода и вывода. Алгоритмы ветвления.

1 этап. Формулировка задания, для которого реализуется проект

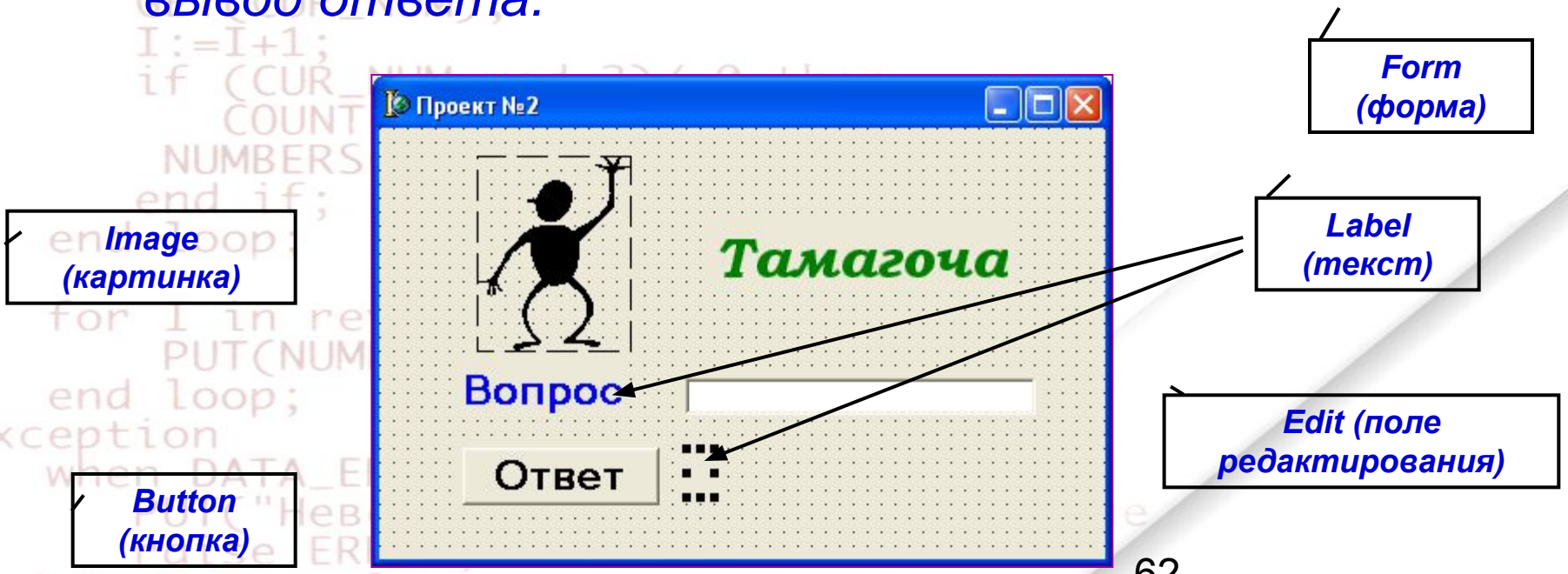
Организуем диалог между пользователем и проектом.

Вариант диалога:
Пользователь: «*Привет*» -
Программа: «*Прриветиик!!!*»;
Пользователь: «*Как тебя зовут?*» -
Программа: «*Тамгоча*»;
Пользователь: «*Как дела?*» -
Программа: «*Суппер!*»;
Пользователь: «*Программировать любишь?*» - Программа:
«*Конечно!!!*»



2 этап. *Проведение проектного анализа и формирование требований к объектам*

1) Необходимо разработать ***форму***, на которой будет размещены ***текст***, окно ***ввода*** вопроса, место ***вывода*** ответа и ***кнопка***.
2) Разработать модуль, который обрабатывает событие: *по нажатию на кнопку происходит вывод ответа.*



3 этап. Выбор необходимых компонентов и разработка алгоритмов обработки компонентов

Установим значения свойств компонент и опишем порядок выполнения событий для проекта

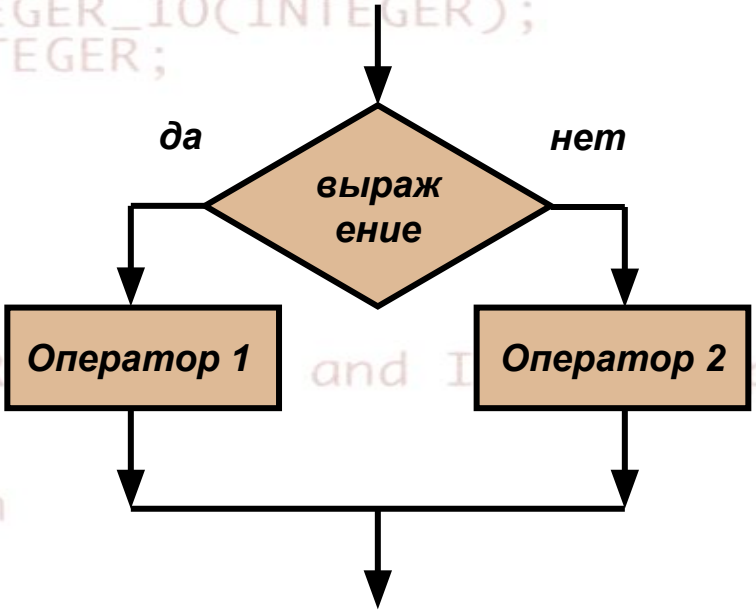
компонент	Object Inspector	Свойство (Properties)\ Событие (Events)	Значение свойства\ Обработка события
Form1	Properties	Name	F1
		Caption	Проект №2
		Color	По выбору
Label1	Properties	Caption	Тамагоча
		Color	По выбору
		Font	По выбору
Label2	Properties	Caption	Вопрос
		Color	По выбору
		Font	По выбору
Label3	Properties	Caption	<пусто>
		Color	По выбору
		Font	По выбору
Image1	Properties	Picture	По выбору
		Stretch	True
Edit1	Properties	Text	<пусто>
		Font	По выбору

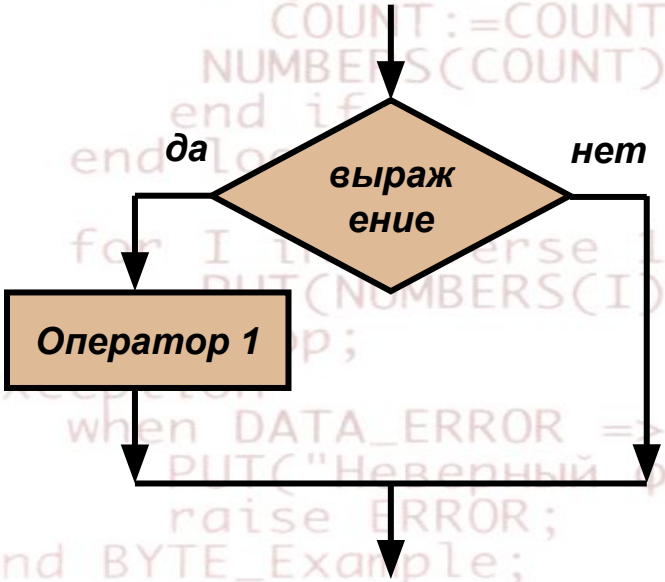
Для реализации диалога необходимо использовать базовый алгоритм ветвления

Полная форма оператора условного перехода

Общий вид оператора:

```
if <логическое выражение>
then <оператор 1>
else <оператор 2>;
```





Сокращенная форма оператора условного перехода

Общий вид оператора:

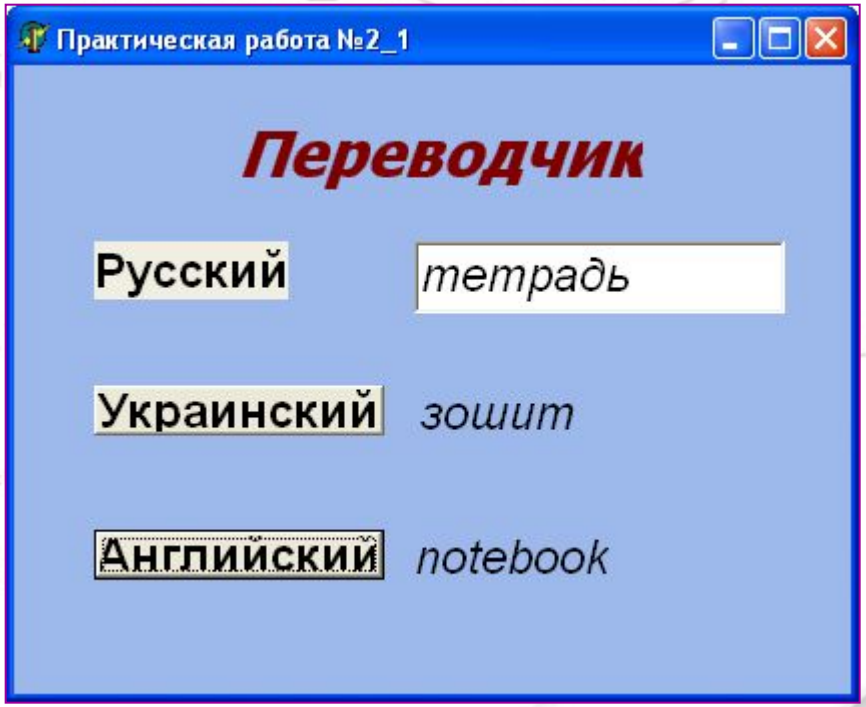
```
If <логическое выражение>
then <оператор 1>;
```

компонент	Object Inspector	Свойство (Properties)\ Событие (Events)	Значение свойства\ Обработка события
Button1	Properties	Caption	Ответ
		Cursor	crHandPoint
		Font	По выбору
	Events	OnClick	if Edit1.Text= 'Привет!' then Label3.Caption:='Приветтик!'; if Edit1.Text= 'Как тебя зовут?' then Label3.Caption:='Тамагоча!'; if Edit1.Text= 'Как дела?' then Label3.Caption:='Суппер!'; if Edit1.Text= 'Программировать любишь?' then Label3.Caption:='Конечно!!!';

Продолжите разработку проекта, дополнив диалог.

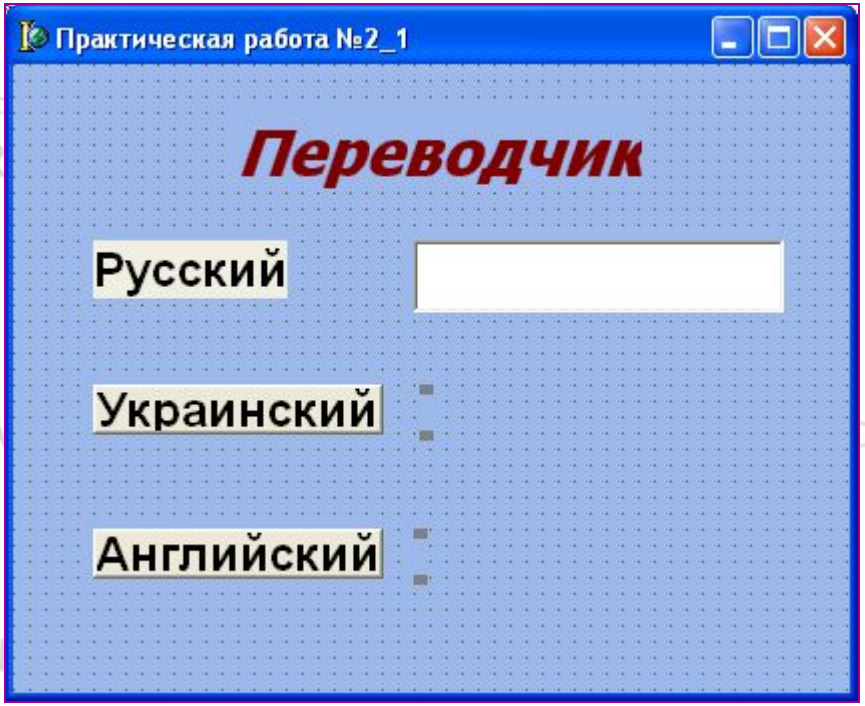
Практическая работа №2_1

Постановка задачи:
Создать проект *Переводчик*, который после ввода слова на *русском* языке, выводит его перевод на *украинском* и *английском* языках.



Самостоятельно:

1. Дополните программу кнопкой *Выход* и рисунком.
2. Разработайте проект, в котором при нажатии на *одну* кнопку, одновременно выводится перевод слова на *двух* языках.

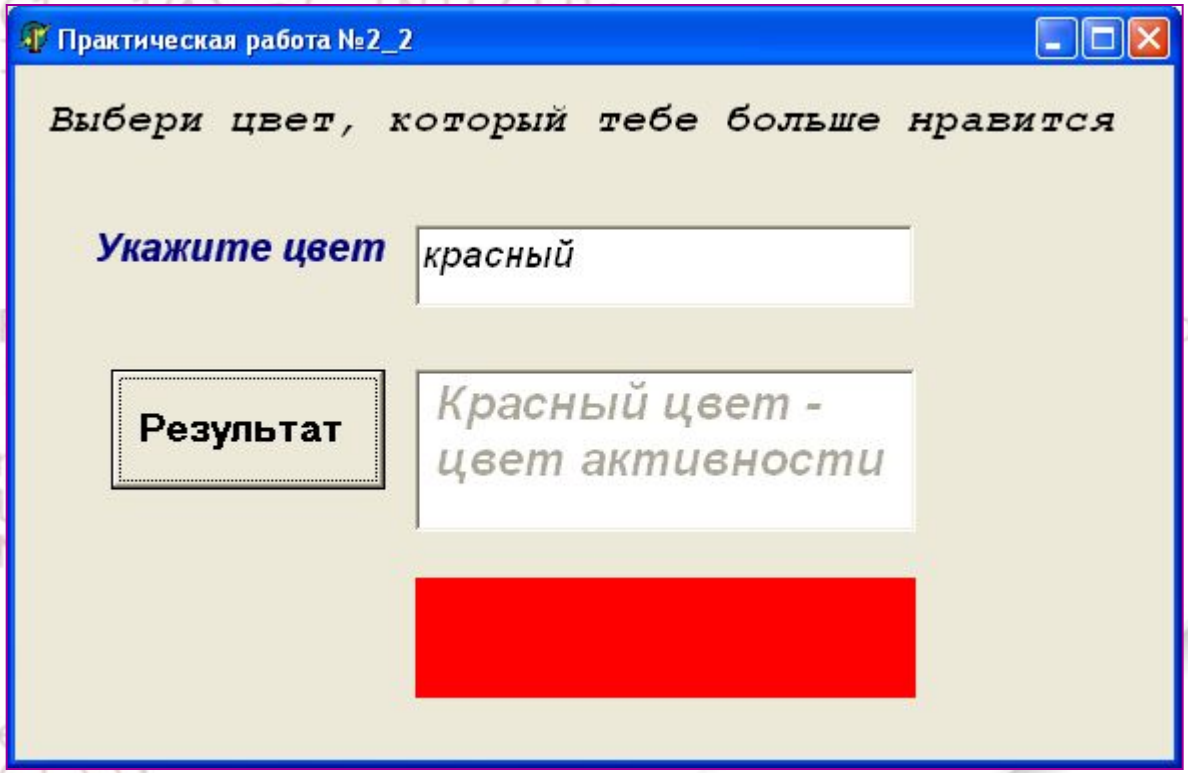


компонент	Object Inspector	Свойство (Properties)\ Событие (Events)	Значение свойства\ Обработка события
Form1	Properties	Name	F2
		Caption	Практическая работа №2_1
		Color	clSkyBlue
Label1	Properties	Caption	Переводчик
		Font	Tahoma, жирный курсив, 24 пт, малинового цвета
Label2	Properties	Caption	Русский
		Font	Arial, курсив, 14 пт
Label3	Properties	Caption	<пусто>
		Font	Tahoma, жирный курсив, 24 пт, малинового цвета
Label4	Properties	Caption	<пусто>
		Font	Arial, курсив, 14 пт
Edit1	Properties	Text	<пусто>
		Font	Arial, курсив, 14 пт

компонент	Object Inspector	Свойство (Properties)\ Событие (Events)	Значение свойства\ Обработка события
Button1	Properties	Caption	Украинский
		Cursor	По выбору
		Font	По выбору
	Events	OnClick	if Edit1.Text='карандаш' then Label3.Caption:='олівець'; if Edit1.Text='тетрадь' then Label3.Caption:='зошит'; if Edit1.Text='доска' then Label3.Caption:='дошка'; if Edit1.Text='расписание' then Label3.Caption:='розклад'; if Edit1.Text='учебник' then Label3.Caption:='підручник';
Button2	Properties	Caption	Английский
		Cursor	По выбору
		Font	По выбору
	Events	OnClick	if Edit1.Text='карандаш' then Label4.Caption:='pencil'; if Edit1.Text='тетрадь' then Label4.Caption:='notebook'; if Edit1.Text='доска' then Label4.Caption:='board'; if Edit1.Text='расписание' then Label4.Caption:='time-table'; if Edit1.Text='учебник' then Label4.Caption:='textbook';

Практическая работа №2-2

Постановка задачи: Создать проект *Цвет - Характер*, который после ввода цвета, отображает его на форме и дает анализ характера.

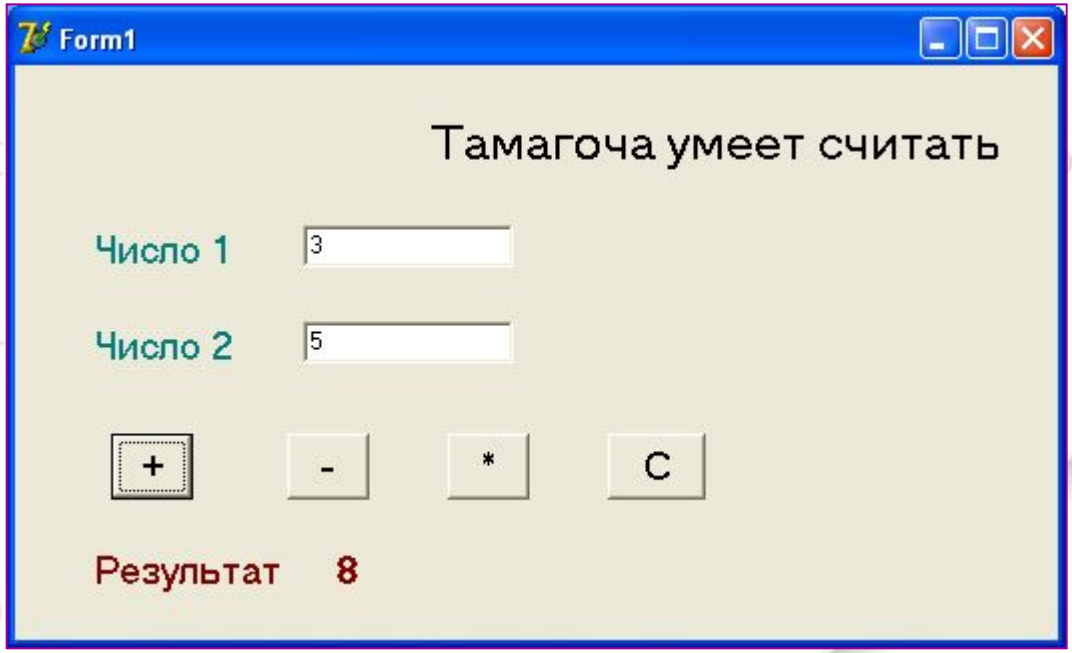


компонент	Object Inspector	Свойство (Properties)\ Событие (Events)	Значение свойства\ Обработка события
Form1	Properties	Name	Form2_2
		Caption	Практическая работа №2_2
Label1	Properties	Caption	Выбери цвет, который тебе больше нравится
Label2	Properties	Caption	Укажите цвет
Label3	Properties	Caption	<пусто>
Edit1	Properties	Text	<пусто>
Memo1	Properties	Lines	<пусто>
		Font	<пусто>
		Enabled	False
Button1	Properties	Caption	Результат
	Events	OnClick	if Edit1.Text=красный' then begin Label3.Color:=clRed; Memo1.Text:='красный цвет – цвет активности'; end; if Edit1.Text=зеленый' then begin Label3.Color:=clGreen; Memo1.Text:=зеленый цвет – цвет уравновешенности'; end;

Создание проекта №3, используя компоненты ввода, вывода и операции вычисления.

Постановка задачи:

Создать проект *Калькулятор*, который позволит выполнять сложение, вычитание и умножение двух целых чисел.



1 этап. *Формулировка задания, для которого реализуется проект*

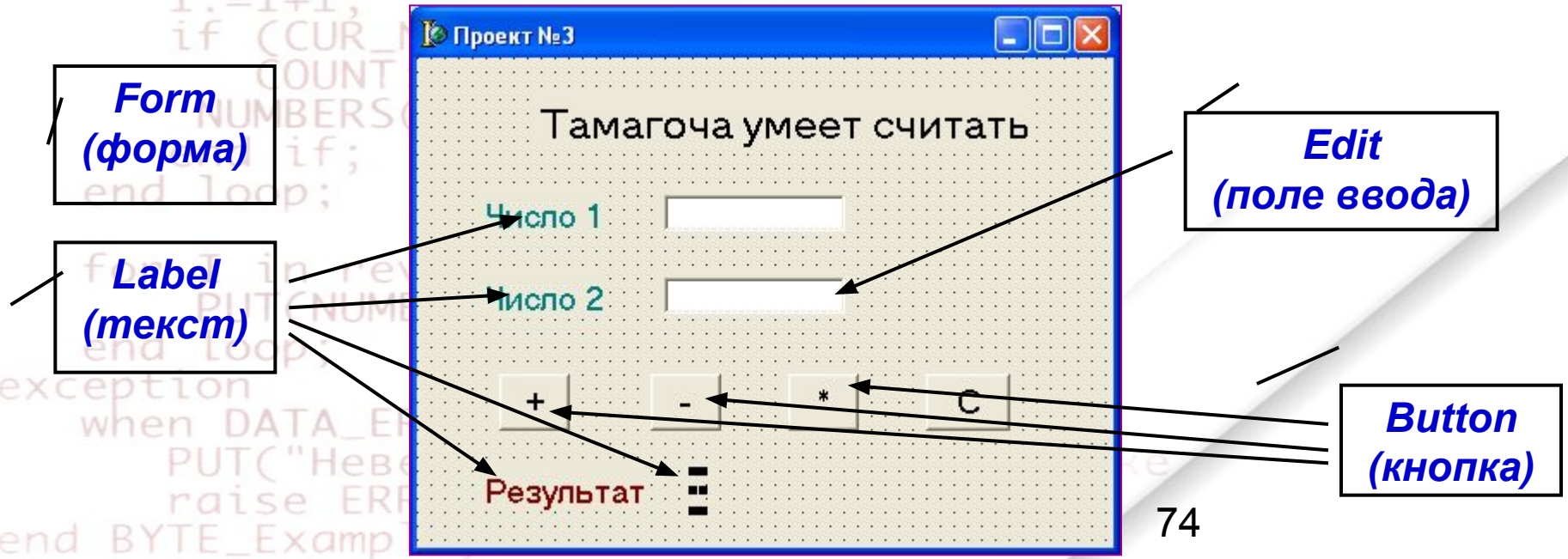
Проект предполагает *ввод* двух чисел и *вывод результата* арифметических действий: *сложения*, *вычитания*, *умножения*. Числа будем вводить целыми, поэтому операцию деления выполнять пока не будем.

Анализ задачи:

Необходимо разработать форму, на которой будет размещен: текст, окна ввода чисел, место вывода ответа и кнопки, по нажатию на которые происходит вывод результата вычисления.

2 этап. Проведение проектного анализа и формирование требований к объектам

1) Создайте чистую *форму*.
2) На форме разместите и настройте компоненты: *Label1* - название проекта; *Label2*, *Label3* – комментарий «*Число1*», «*Число2*»; *Label4*, *Label5* – комментарий и место вывода результата вычислений.
3) Необходимо организовать окна ввода чисел – *Edit1* и *Edit2*.
4) Разместите кнопки *Button1*, *Button2*, *Button3* – для выполнения арифметических операций; предусмотрим кнопку *Button4* предназначенную для очищения окон ввода чисел.
5) Все компоненты оформите по своему усмотрению: цвет, шрифт, размер.



3 этап. Выбор необходимых компонентов и разработка алгоритмов обработки компонентов

Для написания модуля нужно знать, что компоненты *Edit* вводят *текст* а не *числа*, поэтому *необходимо предусмотреть перевод текстовой информации в числовую*. Для этого воспользуемся встроенными функциями

- ***StrToInt* – перевод строки в целое число,** или
- ***StrToFloat* — преобразование строки в значение с плавающей запятой**.

Результат вычисления в нашем случае числовой, для отображения его в текстовом поле *Label* нужно перевести числовое значение в текстовую строку. Для этого воспользуемся встроенными функциями

- ***IntToStr* - перевод целого числа в строку,**
- ***FloatToStr* - перевод числа с плавающей запятой в строку.**

В процедурах обработки кнопки опишем целочисленные переменные, присвоим им введенные значения компоненты *Edit*, выполним арифметическую операцию и компоненте *Label* присвоим полученное значение.

Установим значения свойств компонент и опишем порядок выполнения событий для проекта

компонент	*Object Inspector*	*Свойство (Properties)\ Событие (Events)*	*Значение свойства\ Обработка события*
Form1	**Properties**	**Name**	***Form1***
		Caption	***Проект №3***
Label1	**Properties**	**Caption**	***Тамагоча умеет считать***
		Color	***По выбору***
		Font	***По выбору***
Label2	**Properties**	**Caption**	***Число 1***
		Color	***По выбору***
		Font	***По выбору***
Label3	**Properties**	**Caption**	***Число 3***
		Color	***По выбору***
		Font	***По выбору***
Label4	**Properties**	**Caption**	***Результат***
		Color	***По выбору***
		Font	***По выбору***
Label5	**Properties**	**Caption**	***<пусто>***
		Color	***По выбору***
		Font	***По выбору***
Edit1	**Properties**	**Text**	***<пусто>***

компонент	Object Inspector	Свойство (Properties)\ Событие (Events)	Значение свойства\ Обработка события
Button1	Properties	Caption	+

	Events	OnClick	a:=StrToInt(Edit1.Text);
			b:=StrToInt(Edit2.Text);
			c:=a+b;
			Label5.Caption:=IntToStr(c);

Button2	Properties	Caption	-

	Events	OnClick	a:=StrToInt(Edit1.Text);
			b:=StrToInt(Edit2.Text);
			c:=a-b;
			Label5.Caption:=IntToStr(c);

Button3	Properties	Caption	*

	Events	OnClick	a:=StrToInt(Edit1.Text);
			b:=StrToInt(Edit2.Text);
			c:=a*b;
			Label5.Caption:=IntToStr(c);

Button4	Properties	Caption	C
	Events	OnClick	Edit1.Text:=''; Edit2.Text:='';

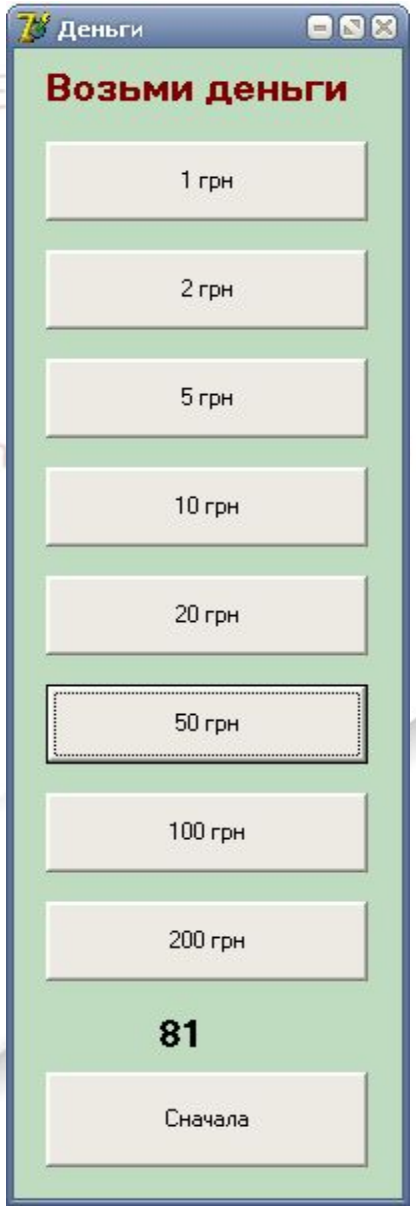
Задание

Продолжите разработку проекта, дополнив возведения первого числа в квадрат.

Глобальные и локальные переменные.

Постановка задачи:

Создать проект *Деньги*, который позволит накапливать сумму денег, при нажатии на соответствующие кнопки.



1 этап. *Формулировка задания, для которого реализуется проект*

Проект предполагает *нажатие пользователем на кнопки на которых написаны номинации купюр: 1 грн, 2 грн, 5 грн и т.д. На окне отображается сумма набранных купюр.*

Анализ задачи:

Необходимо разработать форму, на которой будут размещены кнопки с номинациями купюр, место для вывода суммы и кнопка очищения.

2 этап. Проведение проектного анализа и формирование требований к объектам

1) Создайте чистую *форму*.
2) На форме разместите кнопки *Button1*, *Button2*, *Button3 и т.д.* – для отображения купюр; предусмотрим кнопку *Button* предназначенную для очищения.
3) На форме разместите объект *Label* для отображения результата набора купюр.
4) Все компоненты оформите по своему усмотрению: цвет, шрифт, размер.

3 этап. Выбор необходимых компонентов и разработка алгоритмов обработки компонентов

В процедурах обработки каждой кнопки необходимо наращивать переменную, например: *s:=s+10* и отображать ее значение присваиванием компоненте *Label* полученное значение.

Понятие локальных и глобальных переменных

Однако, если переменную описать внутри процедуры, то на момент активации кнопки резервируется память компьютера под эту переменную и она обнуляется. Такая переменная называется Локальной. Доступ к ней возможен только в той процедуре, где она описана.

Локальная переменная

```
procedure TForm1.Button4Click(Sender: TObject);
s: integer;
begin
s:=s+10;
Label2.Caption:=IntToStr(s);
end;
```

```
unit Unit1;
uses  Windows, ...
type
  TForm1 = class(TForm)
    Button1: TButton;
    ...
    procedure Button1
    ...
var
  Form1: TForm1;
  s:integer;

procedure TForm1.Button1Click(Sender: TObject);
begin  ……  end;…
```

Глобальная переменная

Чтобы переменная сохраняла свое значение на протяжении всей работы программы и передавала значение из одной процедуры в другую, ее необходимо объявить Глобальной. Для этого ее описывают до описания всех процедур.

Такая переменная будет видна и доступна в любой процедуре.

свойства компонентов Button и их события

компонент	*Object Inspector*	*Свойство (Properties)\ Событие (Events)*	*Значение свойства\ Обработка события*
Button1	Properties	Caption	*1 грн*
	Events	OnClick	*s:=s+1;* *Label2.Caption:=IntToStr(s);*
…..			
Button7	Properties	Caption	*100 грн*
	Events	OnClick	*s:=s+100;* *Label2.Caption:=IntToStr(s);*
Button8	Properties	Caption	*Сначала*
	Events	OnClick	*s:=0;* *Label2.Caption:=‘’;*

?1 ?2 ?3 ?4

Практическая работа №3_1

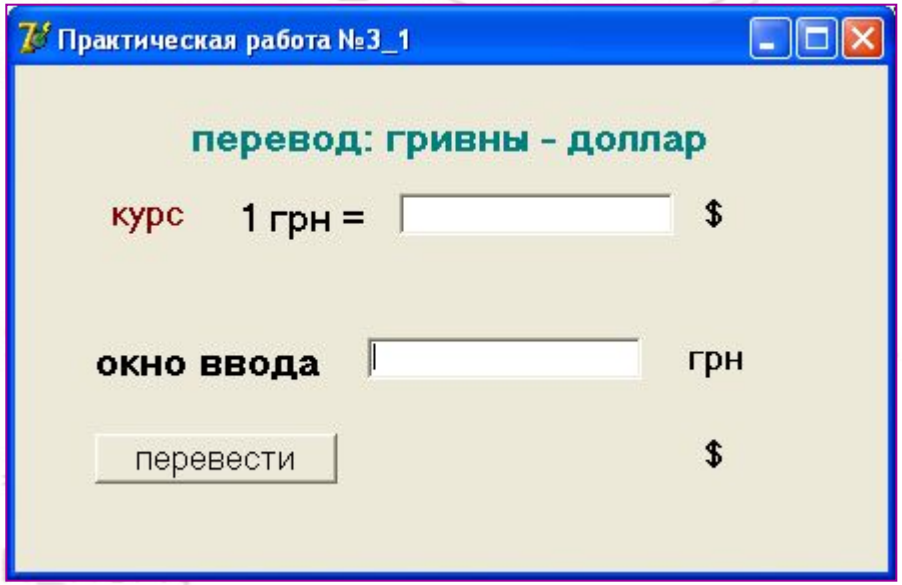
Постановка задачи: Разработайте проект, который вычисляет процентное содержание рыбок в аквариуме.

Необходимо вводить количество рыбок и вычислять общее количество и процент.

Дополнительно: Предусмотреть очищение данных и анализ деления на 0.

Практическая работа №3_2

Постановка задачи:
Разработайте проект по вычислению задачи: *перевести гривны в доллары*. Предположить окно *ввода курса валюты* и *исходное количество денег*.

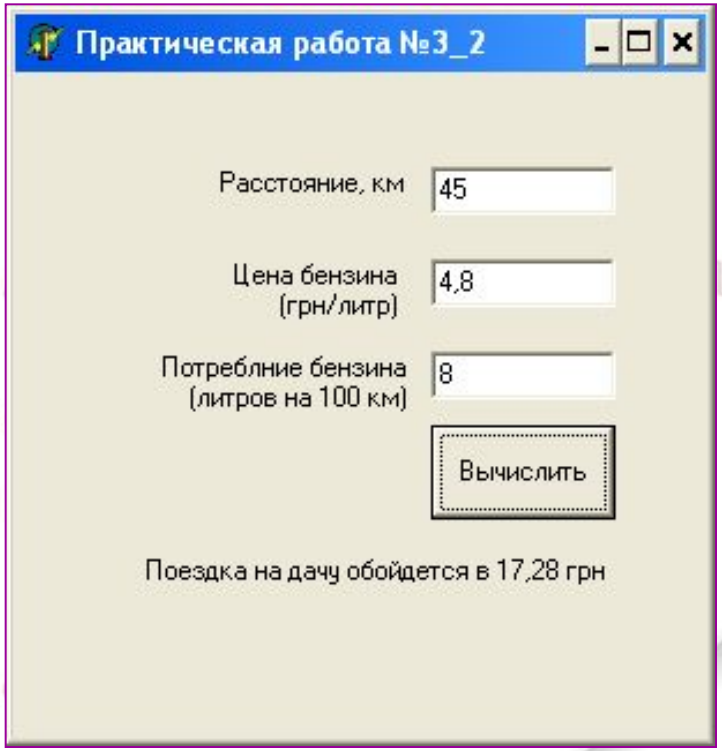


Практическая работа №3_3

Постановка задачи:

Разработайте проект по вычислению стоимости поездки на дачу.

Предусмотреть поле *ввода расстояния до дачи, цены бензина* и *потребление бензина*.

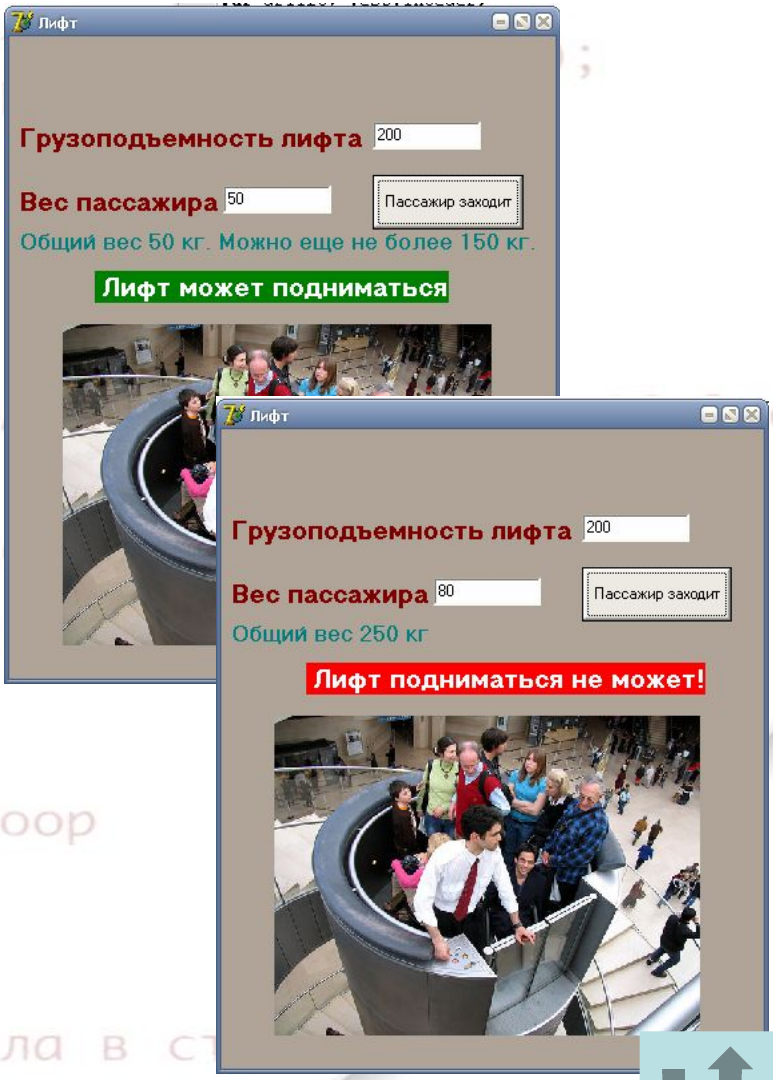


Практическая работа №3_4

Постановка задачи:

Разработайте проект по вычислению суммарного веса пассажирского лифта.

Предусмотреть поле для ввода веса каждого пассажира, грузоподъемность лифта. Программа должна анализировать возможность подъема лифта и сообщать суммарный вес пассажиров.



Создание проекта №4, используя компоненты ввода, вывода.

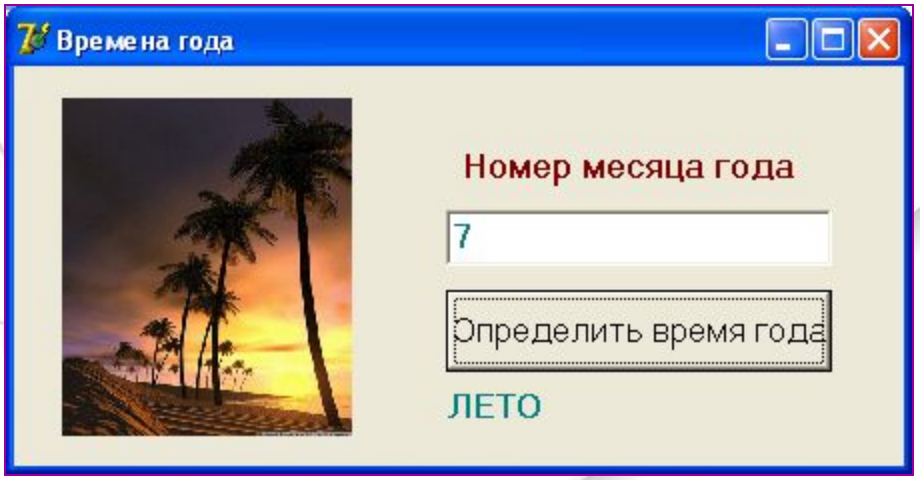
Алгоритмы выбора варианта.

1 этап. Формулировка задания, для которого реализуется проект

Проект предполагает *по номеру месяца в году вывод названия времени года и вывод соответствующего рисунка.*

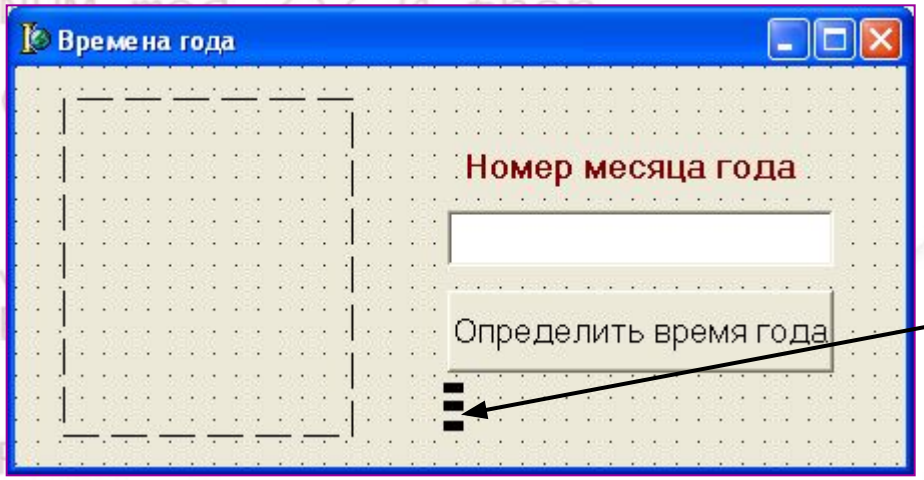
Анализ задачи:

Необходимо разработать форму, на которой будет размещен: текст, окна ввода числа, место вывода ответа и рисунка, кнопки, по нажатию на которые происходит вывод результата.



2 этап. Проведение проектного анализа и формирование требований к объектам

1) Создайте чистую *форму*.
2) На форме разместите и настройте компоненты: *Label1* – «Номер месяца года»; *Label2* –место вывода времени года.
3) Необходимо организовать окна ввода номер месяца года– *Edit1*.
4) Разместите кнопку *Button1* – для выполнения обработки числа.
5) Разместите картину *Image1* – для вывода рисунка с изображением соответствующего времени года.
6) Все компоненты оформите по своему усмотрению: цвет, шрифт, размер.



Form (форма)

Image (картинка)

Button (кнопка)

Edit (поле ввода)

Label (текст)

3 этап. Выбор необходимых компонентов и разработка алгоритмов обработки компонентов

Для написания модуля нужно знать, что компоненты *Edit* вводят *текст* а не *числа*, поэтому *необходимо предусмотреть перевод текстовой информации в числовую*. Для этого воспользуемся встроенными функциями

***StrToInt* – перевод строки в целое число.**

Результат вычисления в нашем случае числовой, для отображения его в текстовом поле *Label* нужно перевести числовое значение в текстовую строку. Для этого воспользуемся встроенными функциями

***IntToStr* - перевод целого числа в строку.**

В процедурах обработки кнопки опишем целочисленные переменные, присвоим им введенные значения компоненты *Edit*, выполним арифметическую операцию и компоненте *Label* присвоим полученное значение.

Если произошла ошибка ввода, то обратимся к процедуре ***ShowMessage***, которая и *вызовет окно комментария с текстом предупреждения*:

ShowMessage('Введите целое число от 1 до 12');

Для реализации проектов данного раздела необходимо использовать базовый алгоритм выбора варианта

```
case переменная of
  метка1: команда1;
  метка2: команда2;
  метка3: команда3;
  …
  else командаК;
end;
```

Где:

переменная - переменная перечисляемого типа (integer, Car);

метка1 – предполагаемые значения переменной;

команда1– оператор языка программирования;

Меткой может быть диапазон значений, например 1..5, и перечислением 1,2,3.

Если по метке необходимо выполнить несколько операторов, то они берутся в операторные скобки Begin … End.

Порядок выполнения оператора:

1. *В списке меток осуществляется поиск значения переменной.*
2. *Если метка найдена, то выполняется оператор, стоящий после :*
3. *Если в списке меток нет значения, то выполняется команда, стоящая по ветке Else.*

Установим значения свойств компонент и опишем порядок выполнения событий для проекта

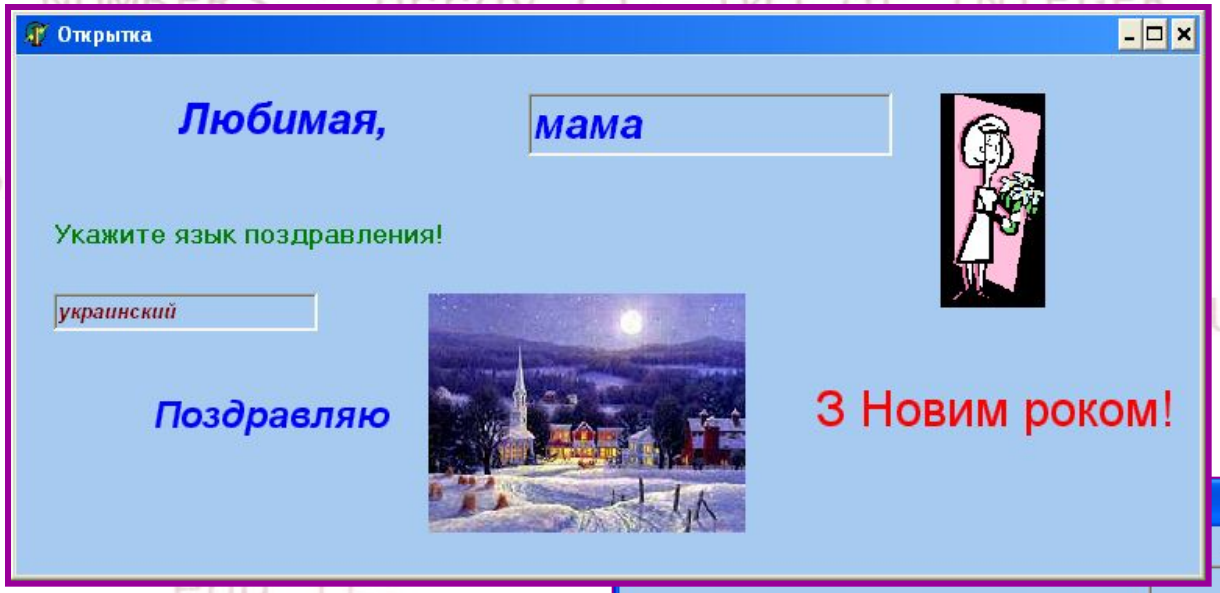
компонент	*Object Inspector*	*Свойство (Properties)\ Событие (Events)*	*Значение свойства\ Обработка события*
Form1	Properties	Name	*Form1*
		Caption	*Времена года*
Label1	Properties	Caption	*Номер месяца года*
Label2	Properties	Caption	*<пусто>*
Edit1	Properties	Text	*<пусто>*
Image1	Properties	Picture	*<пусто>*
		Stretch	*True*
Button1	Properties	Caption	*Определить время года*
	Events	OnClick	*i:=StrToInt(Edit1.Text);* *case i of* *1..2,12: begin* *Label2.Caption:='ЗИМА';* *Image1.Picture.LoadFromFile('zima.bmp') ; end;* *3..5: begin* *Label2.Caption:='ВЕСНА';* *Image1.Picture.LoadFromFile('vesna.bmp'); end;*

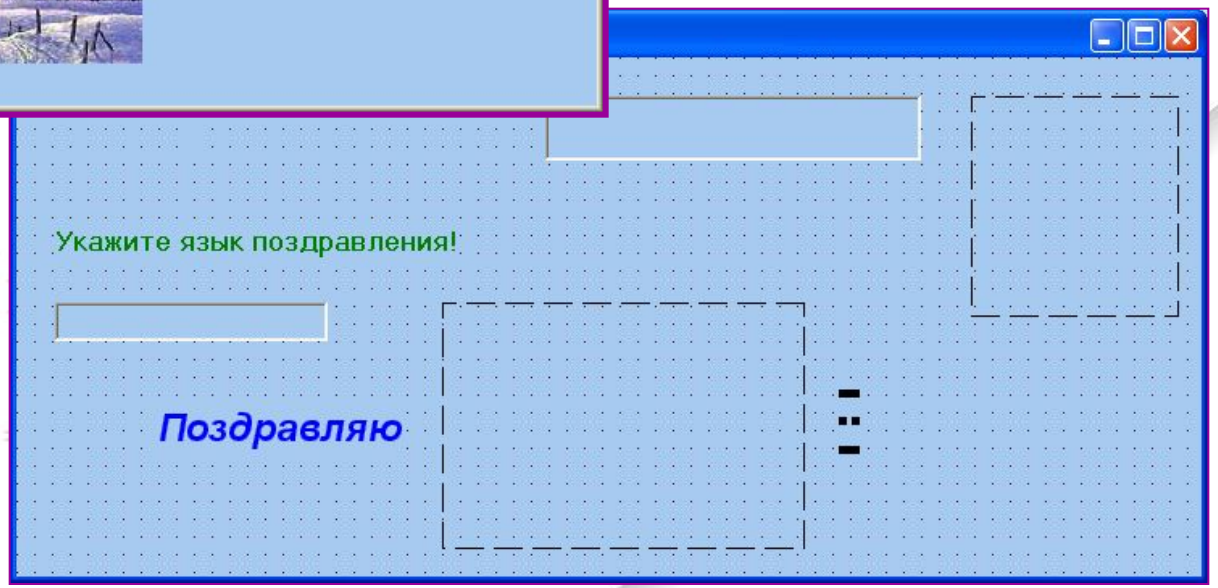
Задание. Продолжите разработку проекта для остальных месяцев.

Практическая работа №4_1

Постановка задачи:

Разработайте проект поздравительной открытки с Новым годом. Предусмотреть *поле ввода для обращения* и поле *ввода языка поздравления*.





Практическая работа №4_2

Постановка задачи:

Разработайте проект определения знака зодиака по номеру года.

Предусмотреть поле *ввода для номера года*.

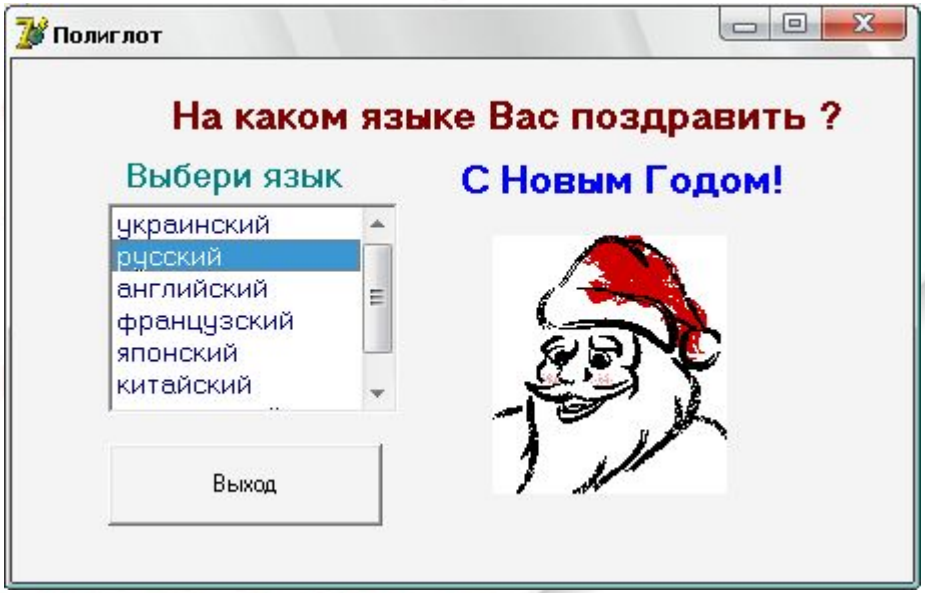
Создание проекта №5, используя компоненты выбора строки из списка. Алгоритм ветвления.

1 этап. Формулировка задания, для которого реализуется проект

Проект предполагает *выбор строки из списка.*

Анализ задачи:

Необходимо разработать форму, на которой будет размещен: текст, окно списка, место вывода текста и рисунка, кнопки, по нажатию на которые происходит вывод из программы.



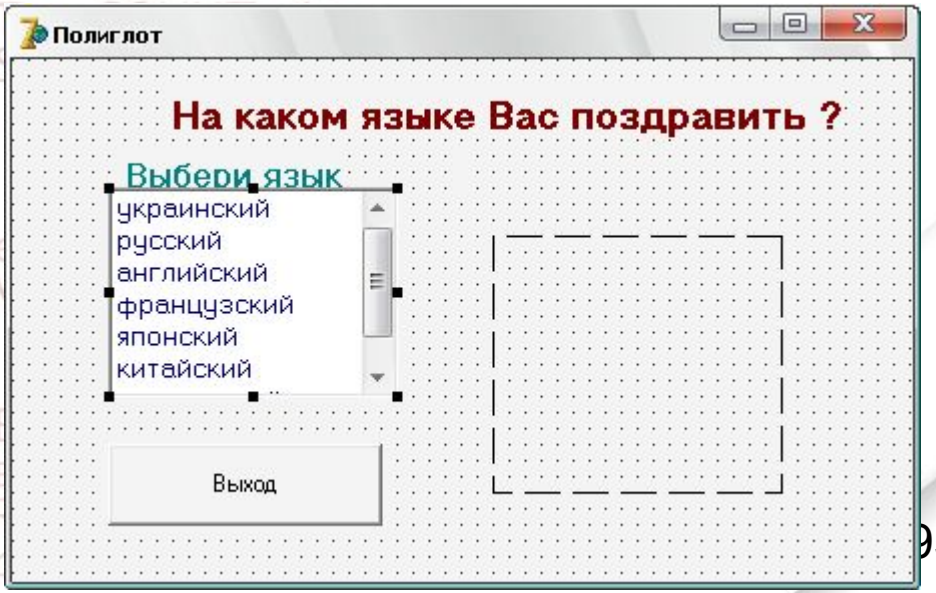
2 этап. Проведение проектного анализа и формирование требований к объектам

1) Создайте чистую *форму*.
2) На форме разместите и настройте компоненты: *Label1* – «На каком языке Вас поздравить?»; *Label2* – «Выбери язык»; *Label3* –место вывода текста.
3) Разместите кнопку *Button1* – для выхода из программы.
4) Разместите картину *Image1* – для вывода рисунка.
5) На форме разместите компоненту *ListBox1*, которая содержит строки – разговорные языки.
6) Все компоненты оформите по своему усмотрению: цвет, шрифт, размер.

Form (форма)

ListBox (список)

Button (кнопка)



Label (текст)

Image (картинка)

3 этап. Выбор необходимых компонентов и разработка алгоритмов обработки компонентов

Для реализации проектов данного раздела необходимо использовать базовый алгоритм выбора варианта (Case).

Переменной перечисляемого типа (integer) необходимо присвоить номер выбранной строки t:=ListBox1.ItemIndex.
Первая строка списка имеет индекс 0,
вторая – 1,
третья – 2, и т.д.
Оператор Case позволит реализовать сложное ветвление в форме выбора варианта.

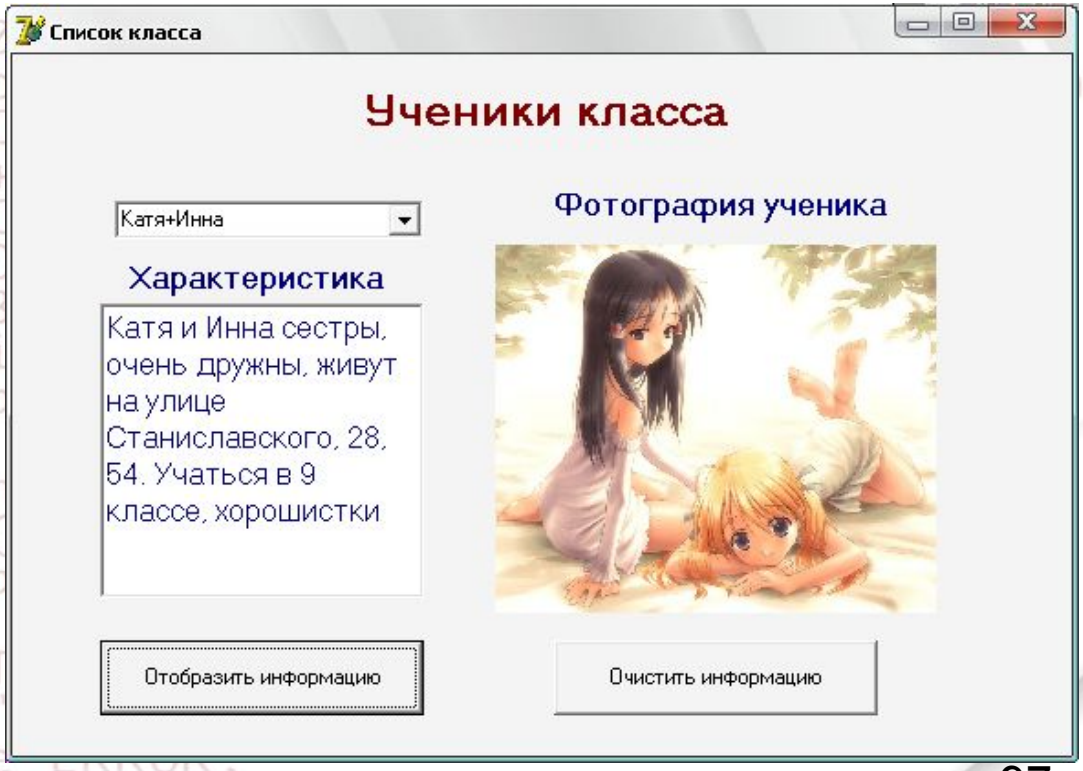
```
case t of
 0: begin Label2.Font.Color:=clpurple;
          Label2.Caption:='З Новим Роком!';
          image1.Picture.LoadFromFile('ukr.bmp'); end;
 1: begin Label2.Font.Color:=clblue;
          Label2.Caption:='С Новым Годом!';
          image1.Picture.LoadFromFile('Rus.bmp'); end;
...
else begin Label2.Font.Color:=clmaroon;
       Label2.Caption:='Язык не известен!';
       image1.Picture.LoadFromFile('not.bmp'); end;
end;
```

Практическая работа №5_1

Постановка задачи:

Разработайте проект вывода информации о ученике класса из списка.

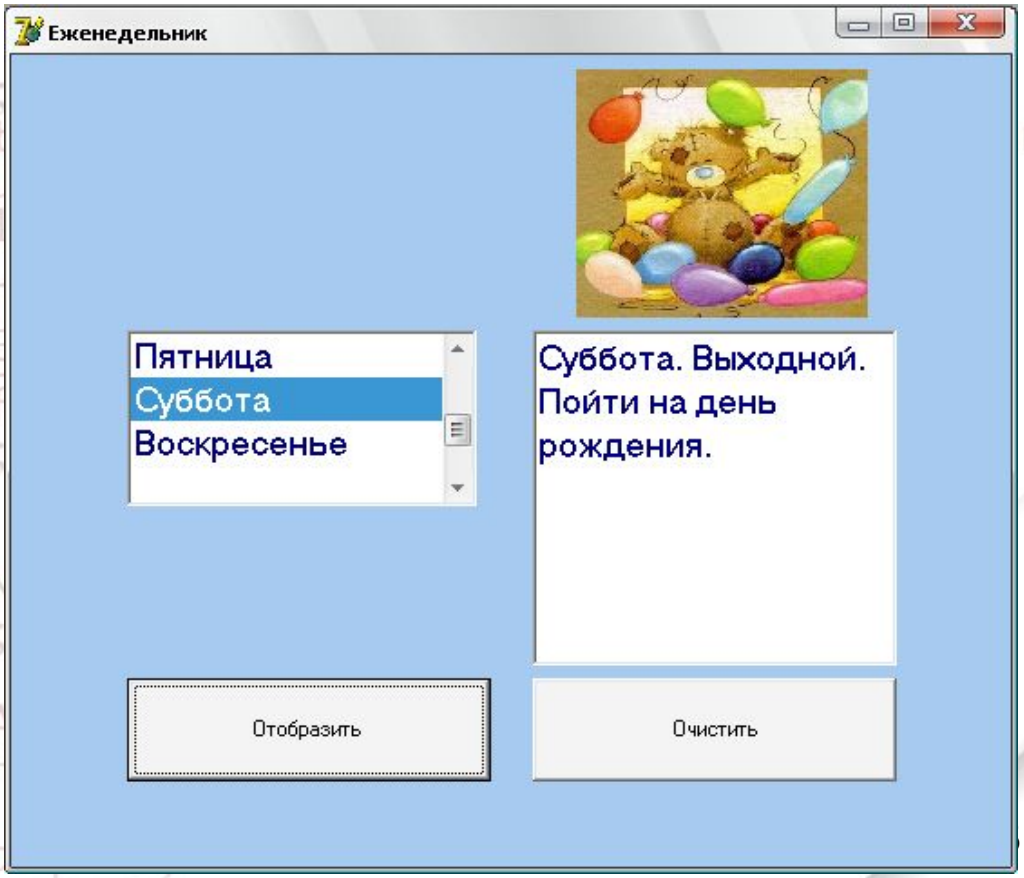
(Проект выполнить используя ComboBox)



Практическая работа №5_2

Постановка задачи:

Разработайте проект вывода ежедневник на неделю. (Проект выполнить используя ListBox)



Создание проекта № 6, используя компоненты ввода, вывода и операции вычисления.

Постановка задачи:

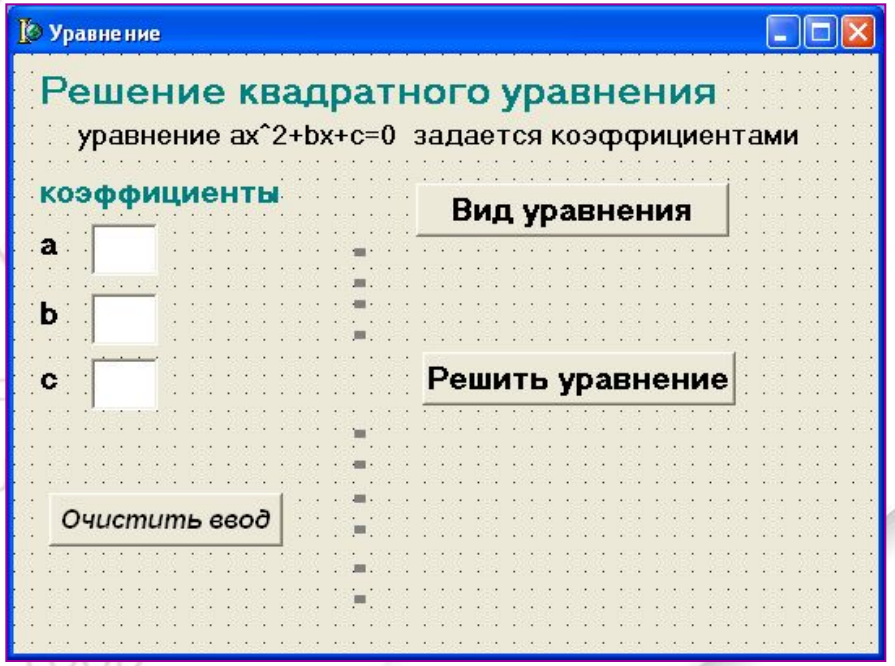
Составить проект, который анализирует вид уравнения и вычисляет его корни. Предусмотреть поля *ввода для коэффициентов*.

1 этап. Формулировка задания, для которого реализуется проект

Проект предполагает *ввод* трех коэффициентов уравнения и *вывод* текстовых сообщений, которые определяют вид уравнения и вывод результатов вычисления его корней.

Анализ задачи:

Необходимо разработать форму, на которой будет размещены: тексты, окна ввода чисел, место вывода ответа, кнопки, по нажатию на которые происходит вывод результата вычисления.



2 этап. Проведение проектного анализа и формирование требований к объектам

1) Создайте чистую форму *(Form)*.
2) Разместите на форме комментарии (*Label*).
3) Разместите на форме окна ввода, для ввода коэффициентов (*Edit*).
4) Кнопки для обработки ситуации определения вида уравнения и вычисления корней уравнения (*Buton*).
5) Зарезервируйте место вывода типа уравнения, его вид, вычисление дискриминанта и корней уравнения (*Label)*.
6) Разместите кнопку «Очистить ввод». (*Buton*)

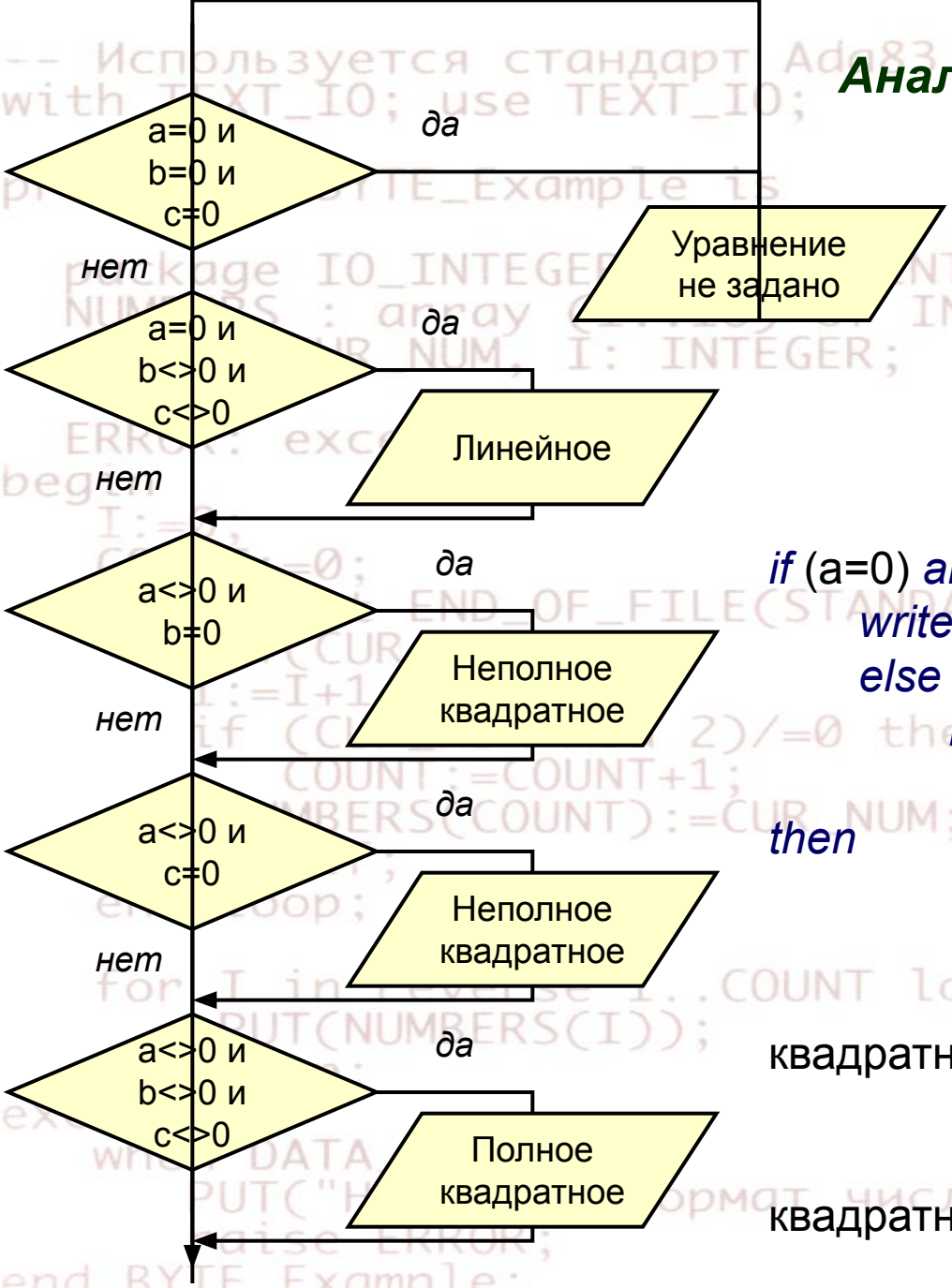
Все компоненты оформите по своему усмотрению: цвет, шрифт, размер.

3 этап. Выбор необходимых компонентов и разработка алгоритмов обработки компонентов

1. Коэффициенты квадратного уравнения будем задавать вещественными числами и научимся выявлять ошибки ввода. Опишем переменные типа *real*:
   var *code1, code2, code3:* ***integer****;*
   a, b, c, d, x1, x2: ***real****;*
2. Коэффициенты квадратного уравнения вводятся в поля *Text* компоненты *Edit*, а значит эти данные текстового типа, их необходимо перевести в вещественный. Для этого воспользуемся стандартной процедурой ***val***, она имеет три параметра ***val(Edit1.Text, a,code1***)
   Edit1.Text - текст, который переводится в число;
   a – переменная числового типа, куда заносится результат перевода;
   code1 - переменная целого типа, где содержится код перевода. Если текст не соответствует числовому представления, то код перевода отличен от нуля.
   Воспользуемся этим параметром и проанализируем код перевода.
3. Если произошла ошибка ввода, то обратимся к процедуре ***ShowMessage***, которая и ***вызовет окно комментария с текстом предупреждения:***

if (code1<>0) or (code2<>0) or (code3<>0) then ShowMessage ('Коэффициенты должны быть числовыми');

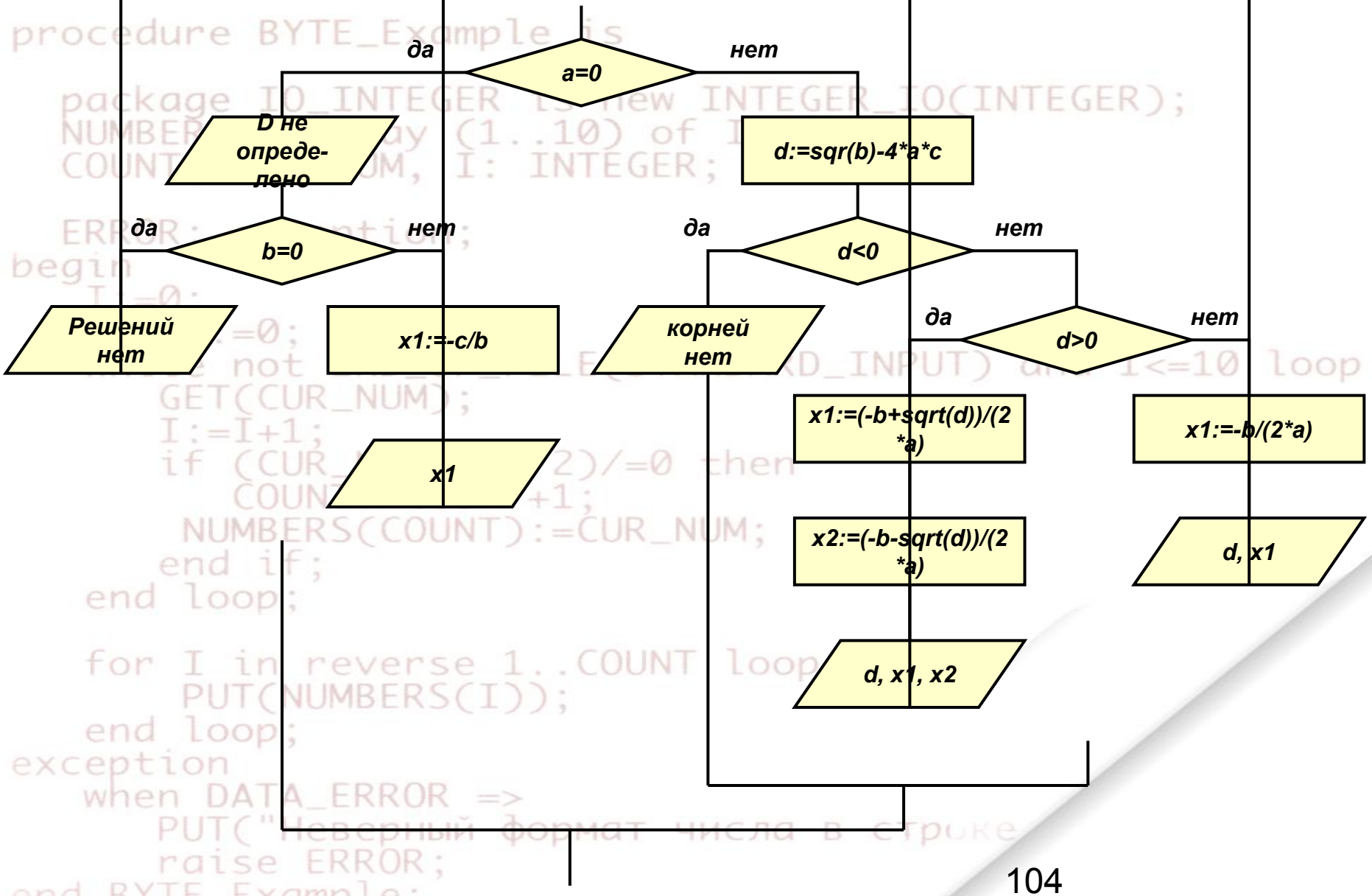
Анализ типа уравнения



Фрагмент программы на Pascal:

```
if (a=0) and (b=0) and (c=0) then
    writeln (’Уравнение не задано‘)
    else
        begin
            if (a=0) and (b<>0) and (c<>0)
then
                writeln    (’ линейное‘);
            If (a<>0) and (b=0) then
                writeln (’ неполное
квадратное’);
            if (a<>0) and (c=0) then
                writeln ('неполное
квадратное ‘);
            if (a<>0) and (b<>0) and (c<>0)
then
```

Вычисление корней уравнения



Фрагмент программы на Pascal:

Задание

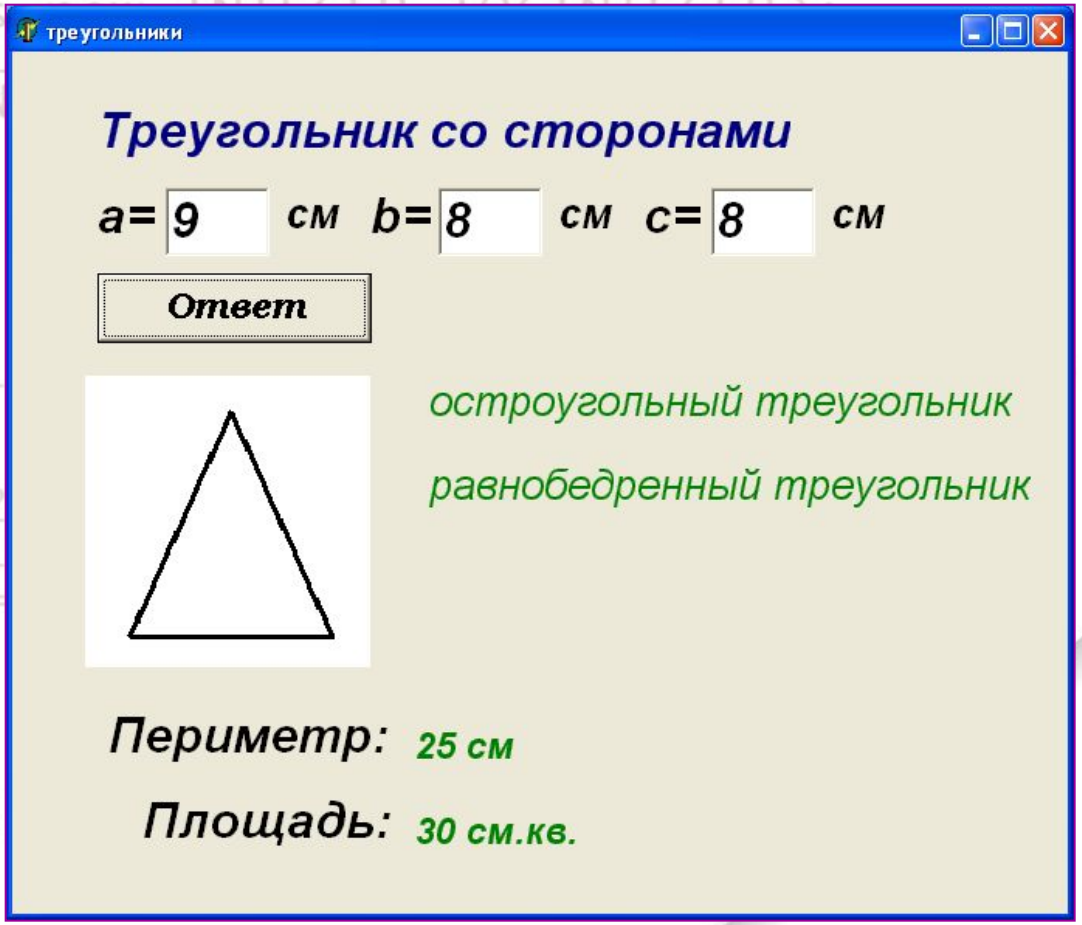
Продолжите разработку проекта, дополнив его выводом уравнения и кнопкой для очистки полей.

```
If (code1<>0) and (code2<>0) and (code3<>0)
 then Writeln('Коэффициенты должны быть числовыми')
else
 if a=0
  then
     begin
        writeln ('не определено');
        if b=0 then writeln ('Решений нет')
        else
           begin
              x1:=-c/b;
              writeln (x1);
           end;
     end
  else
     begin
        d:=sqr(b)-4*a*c;
        if d<0 then
           begin
              writeln (d);
              writeln ('корней нет');
           end
        else
           if d>0 then
              begin
                 x1:=(-b+sqrt(d))/(2*a);
                 x2:=(-b-sqrt(d))/(2*a);
                 writeln (d);
                 writeln (x1);
                 writeln (x2);
              end
           else
              begin
                 x1:=-b/(2*a);
                 writeln (d);
                 writeln (x1);
              end;
     end;
```

Практическая работа №6_1

Постановка задачи:

Составить проект, который анализирует вид треугольника и вычисляет его периметр и сумму. Предусмотреть поля *ввода для значений сторон*.

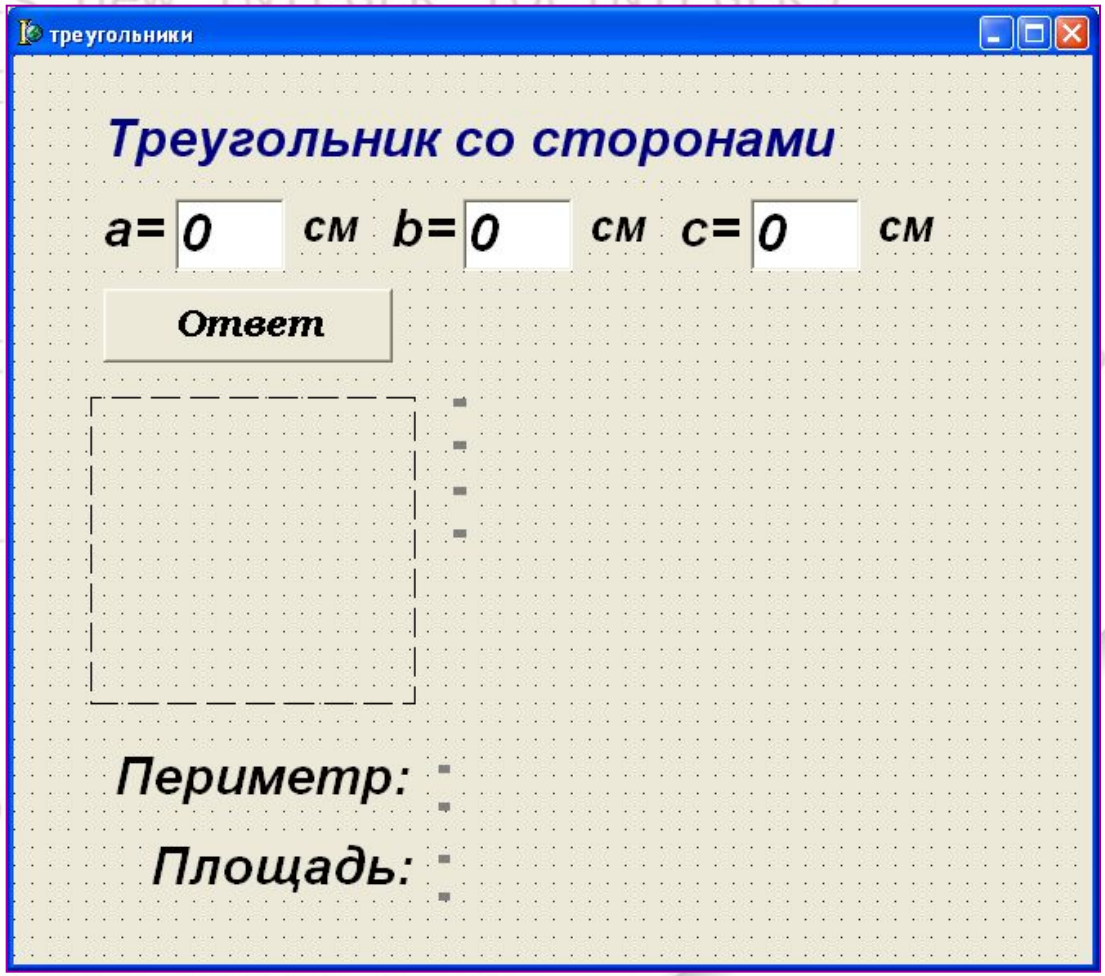


1 этап. Формулировка задания, для которого реализуется проект

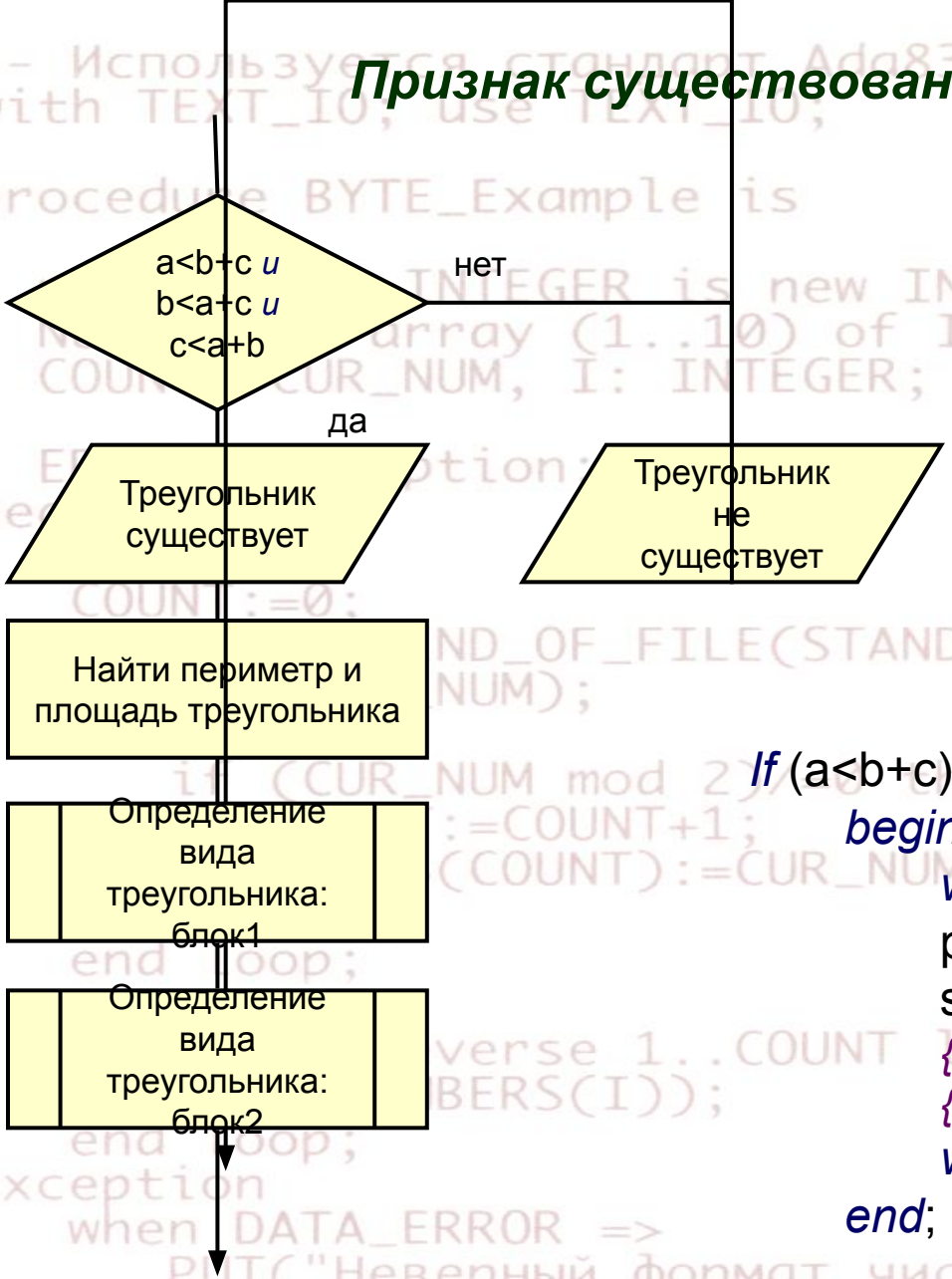
Проект предполагает *ввод* трех сторон треугольника и *вывод* текстовых сообщений, которые определяют вид треугольника и вывод результатов вычисления периметра и площади.

Анализ задачи:

Необходимо разработать форму, на которой будет размещен: тексты, окна ввода коэффициентов, место вывода ответа и рисунка, кнопки, по нажатию на которые происходит вывод результата вычисления.



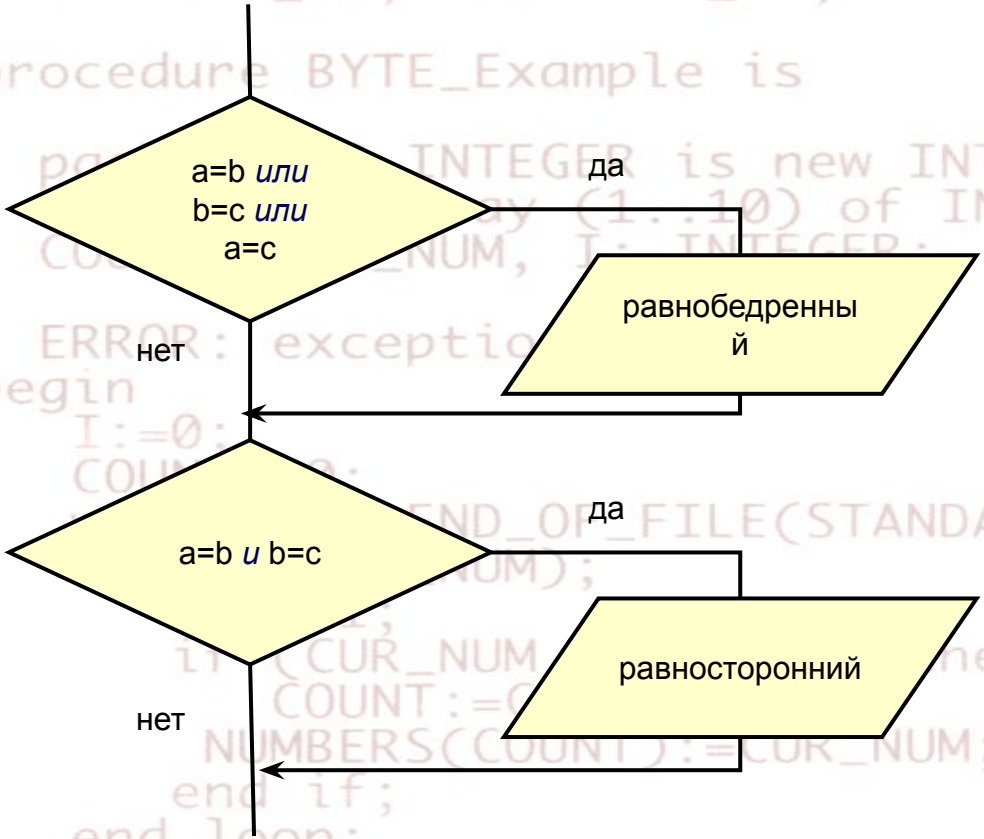
Признак существования треугольника



Фрагмент программы на Pascal:

```
If (a<b+c) and (b<a+c) and (c<a+b)     then
    begin
        writeln('треугольник существует');
        p:=(a+b+c);
        s:=sqrt(p/2*(p/2-a) *(p/2-b) *(p/2-c))
        {определение вида треугольника: блок1};
        {определение вида треугольника: блок2}
        writeln('треугольник существует');
end;
```

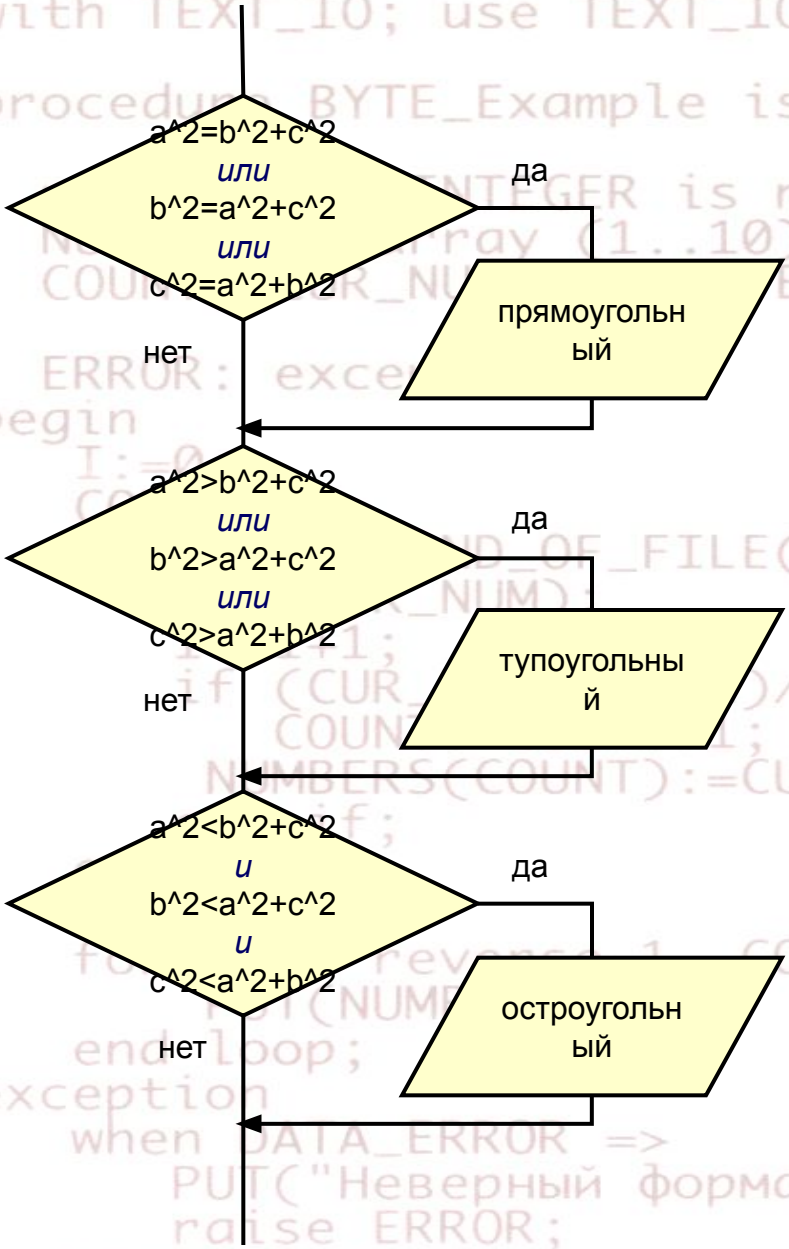
Определение вида треугольника: блок 1



Фрагмент программы на Pascal:

```
If (a=b) or (b=c) or (c=a) then
writeln('треугольник равнобедренный');
If (a=b) and (b=c) then
writeln('треугольник равносторонний');
```

Определение вида треугольника: блок 2



Фрагмент программы на Pascal:

```
If (sqr(a)=sqr(b)+sqr(c)) or
    (sqr(b)=sqr(a)+sqr(c)) or
    (sqr(c)=sqr(b)+sqr(a)) then
    writeln('треугольник прямоугольный');
If (sqr(a)>sqr(b)+sqr(c)) or
    (sqr(b)>sqr(a)+sqr(c)) or
    (sqr(c)>sqr(b)+sqr(a)) then
    writeln('треугольник тупоугольный');
If (sqr(a)<sqr(b)+sqr(c)) and
    (sqr(b)<sqr(a)+sqr(c)) and
    (sqr(c)<sqr(b)+sqr(a)) then
    writeln('треугольник остроугольный');
```

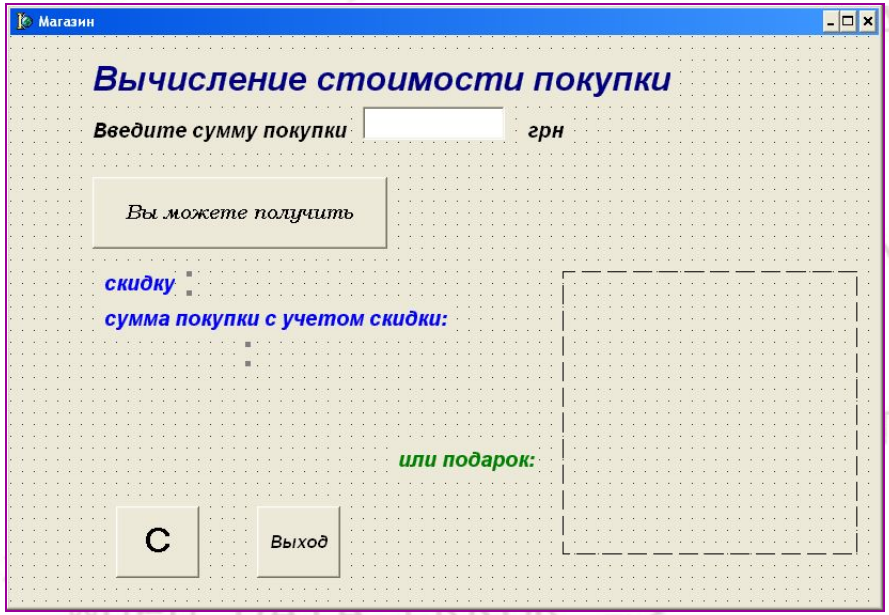
Практическая работа №5_2 (1 вариант)

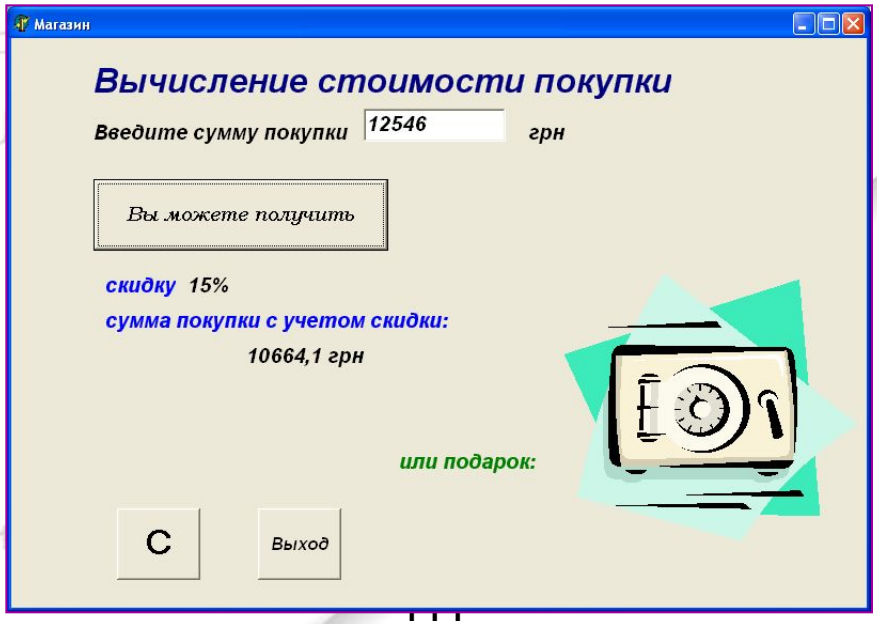
Постановка задачи:

Составить проект, который по значению суммы покупки определяет размер предоставляемой скидки или отображает рисунок с изображением подарка.

Предусмотреть поле *ввода для суммы покупки*.

сумма покупки	скидка	подарок
до 2000	5%	калькулятор
до 5000	8%	принтер
до 8000	10%	телефон
до 20000	15%	фотоаппарат
>20000	20%	компьютер





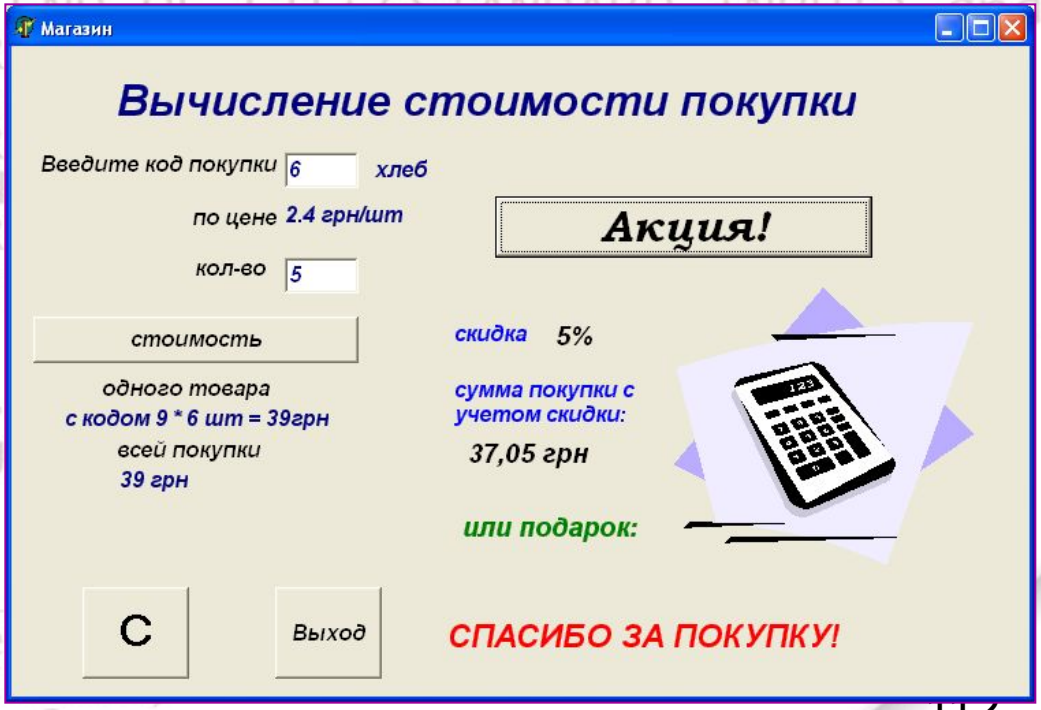
Практическая работа №5_2 (2 вариант)

Постановка задачи:

Составить проект, который по значению суммы покупки определяет размер предоставляемой скидки или отображает рисунок с изображением подарка.

Предусмотреть поле *ввода для суммы покупки*.

сумма покупки	скидка	подарок
до 2000	5%	калькулятор
до 5000	8%	принтер
до 8000	10%	телефон
до 20000	15%	фотоаппарат
>20000	20%	компьютер



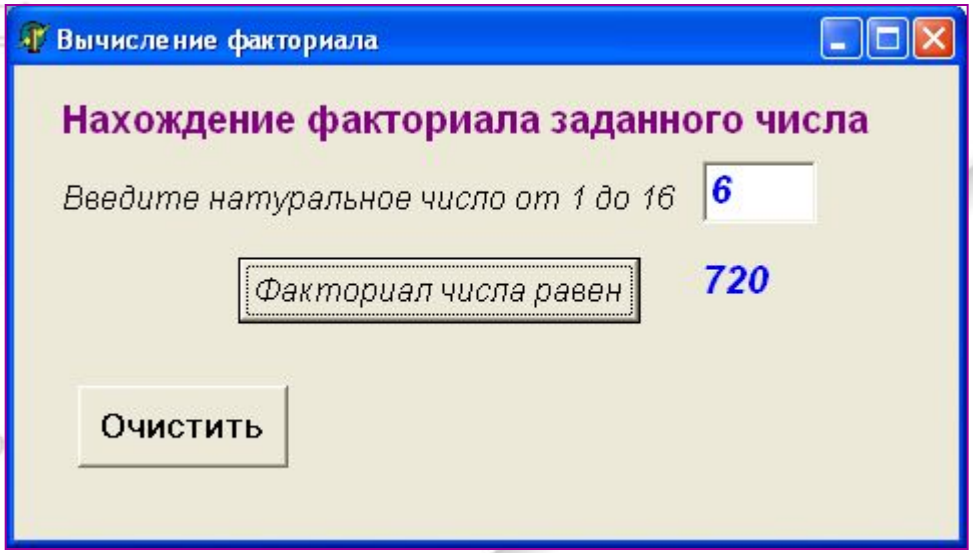
Создание проекта №9, используя циклические алгоритмы.

1 этап. Формулировка задания, для которого реализуется проект

Составим проект, который вычисляет факториал числа.

Анализ задачи:

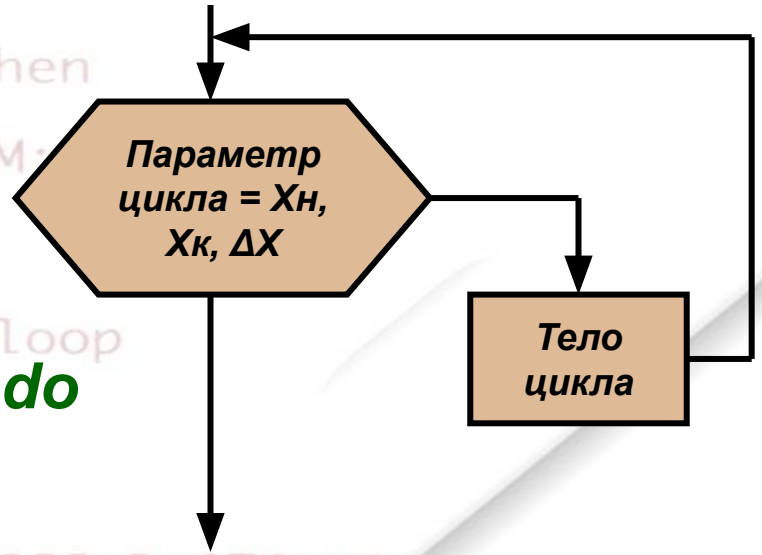
Факториал числа равен произведению всех натуральных чисел от 1 до данного: n!=1*2*3*4*…*n. Факториал числа, большего, чем 16 очень большое число, по разрядам более 5 и не может быть предъявляться типом integer, поэтому ограничим пользователя в воде чисел от 1 до 16.



Для реализации проекта необходимо использовать базовый алгоритм цикла

Перебор всех натуральных чисел от 1 до n организуем стандартным алгоритмом, используя *цикл с параметром (оператор For).*

Цикл с параметром



Общий вид оператора:

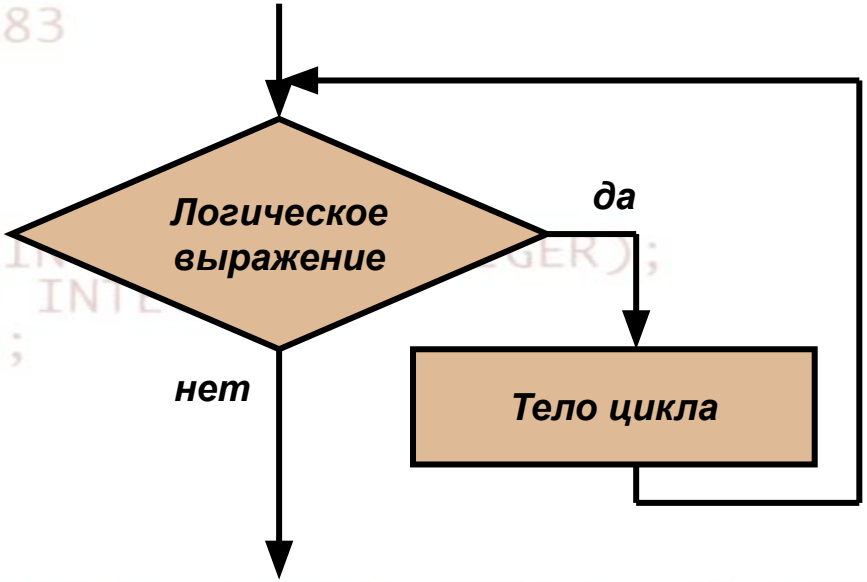
for <параметр цикла> :=Хн to Хк do

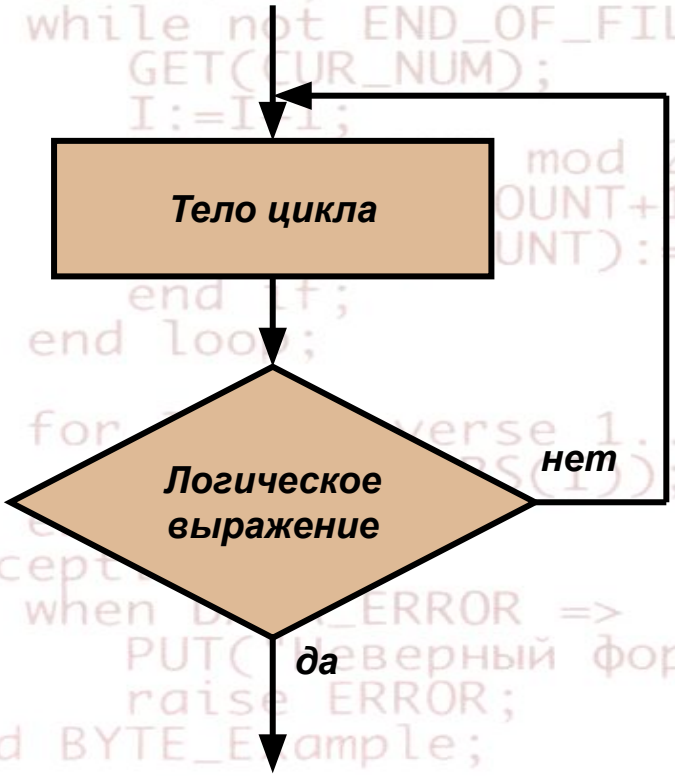
<тело цикла> ;

Цикл-пока

Общий вид оператора:

while <логическое выражение>
do < тело цикла >;





Цикл-до

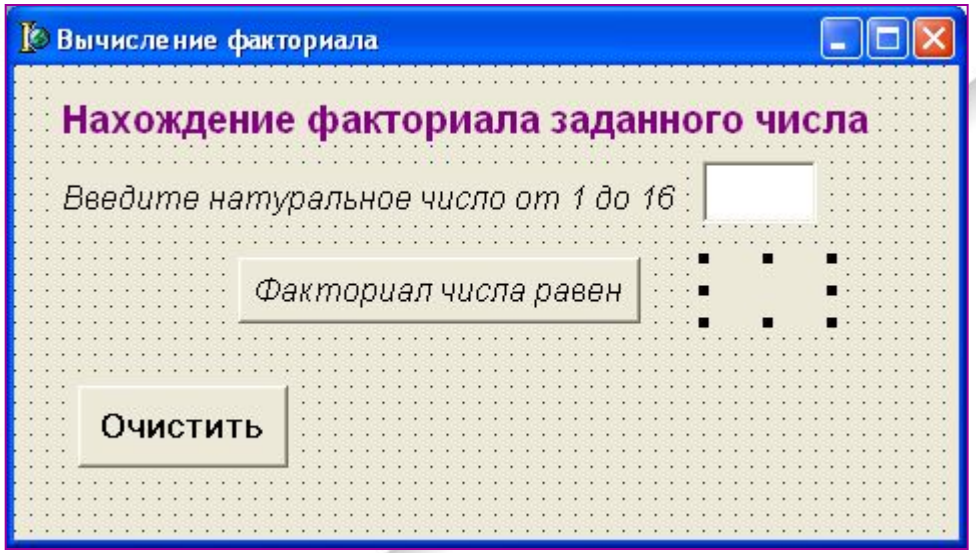
Общий вид оператора:

repeat <тело цикла>
until < логическое выражение>;

2 этап. Проведение проектного анализа и формирование требований к объектам

1) Создайте чистую форму *(Form)*.
2) Разместите на форме комментарии (*Label*).
3) Разместите на форме окно ввода, для ввода числа (*Edit*).
4) Кнопку для вычисления факториала числа (*Buton*).
5) Зарезервируйте место вывода результата вычисление факториала (*Label)*.
6) Разместите кнопку «Очистить» (*Buton*).

Все компоненты оформите по своему усмотрению: цвет, шрифт, размер.



Установим значения свойств компонент и опишем порядок выполнения событий для проекта

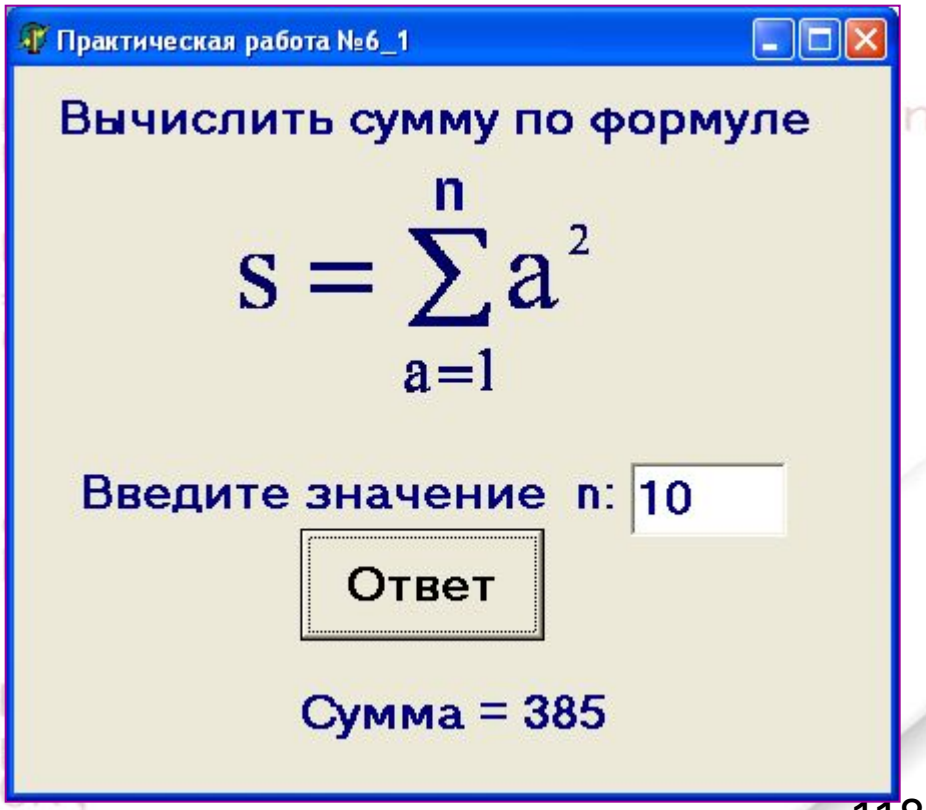
компонент	*Object Inspector*	*Свойство (Properties)\ Событие (Events)*	*Значение свойства\ Обработка события*
Form1	Properties	Caption	*Вычисление факториала*
Label1	Properties	Caption	*Нахождение факториала заданного числа*
Label2	Properties	Caption	*Введите натуральное число от 1 до 16*
Label3	Properties	Caption	*<пусто>*
Edit1	Properties	Text	*<пусто>*
Button1	Properties	Caption	*Факториал числа равен*
	Events	OnClick	*procedure TForm1.Button1Click(Sender: TObject);* *var n, i, code: integer;* *p: integer;* *Begin* *n:=StrToInt(Edit1.Text);* *p:=1;* *for i:=1 to n do* *p:=p*i;* *Label3.Caption:= IntToStr(p);* *end;*
Button2	Properties	Caption	*Факториал числа равен*
	Events	OnClick	*Edit1.Text:='';* *Label3.Caption:='';*

?1 ?2 ?3 ?4 ?5

Практическая работа №9_1

Постановка задачи:

Составим проект, который вычисляет сумму квадратов чисел от 1 до n. Предусмотреть поле *ввода для значения* n.

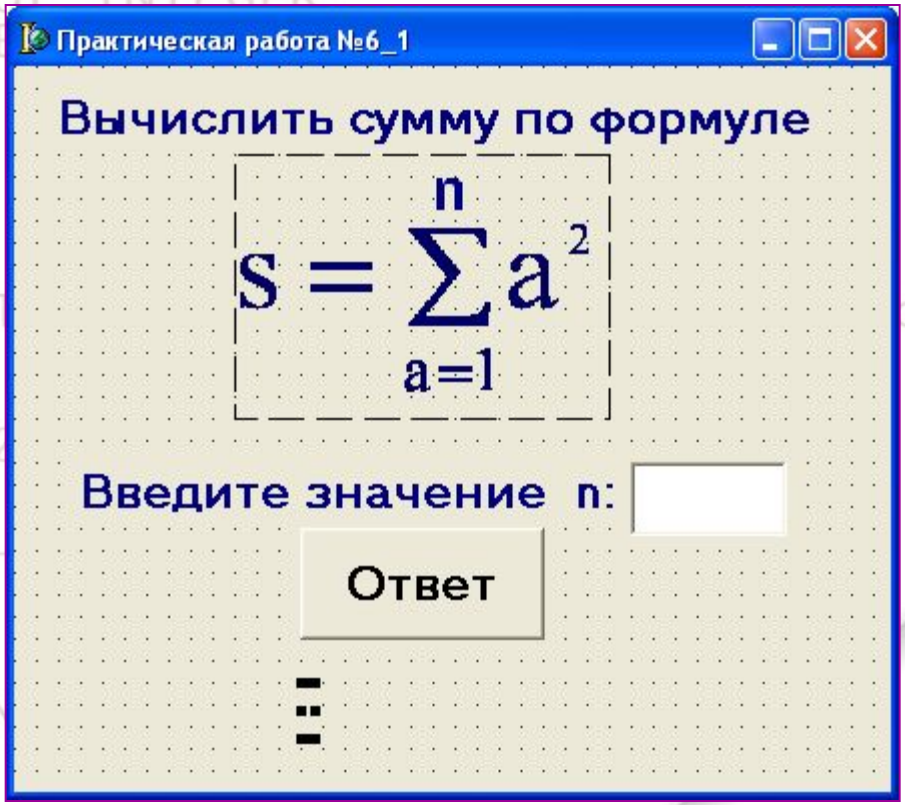


1 этап. Формулировка задания, для которого реализуется проект

Проект предполагает *ввод* количества слагаемых и *вывод* результата вычисления суммы.

Анализ задачи:

Необходимо разработать форму, на которой будет размещены: тексты, рисунок с формулой, окно ввода числа n, место вывода ответа и кнопки, по нажатию на которые происходит вывод результата вычисления.



2 этап. Проведение проектного анализа и формирование требований к объектам

1) Создайте чистую форму *(Form)*.
2) Разместите на форме комментарии (*Label*).
3) Разместите изображение формулы. (*Image*)
4) Разместите на форме окна ввода, для ввода числа n (*Edit*).
5) Кнопки для вычисления суммы чисел (*Buton*).
6) Зарезервируйте место вывода результата вычисления суммы(*Label)*.

Все компоненты оформите по своему усмотрению: цвет, шрифт, размер.

3 этап. Выбор необходимых компонентов и разработка алгоритмов обработки компонентов

Число *n* задаем *целого* типа. Компоненты *Edit* вводят *текст* а не *числа*, поэтому *необходим перевод текстовой информации в числовую*. Для этого воспользуемся встроенными функциями

- ***StrToInt* – перевод строки в целое число.**

Перебор всех натуральных чисел от 1 до n организуем стандартным алгоритмом, используя **цикл со счетчиком (оператор For).**

Результат вычисления числовой, для отображения его в текстовом поле *Label* нужно перевести числовое значение в текстовую строку. Для этого воспользуемся встроенными функциями

- ***IntToStr* - перевод целого числа в строку.**

В процедурах обработки кнопки опишем целочисленные переменные, присвоим им введенные значения компоненты *Edit*, выполним арифметическую операцию и компоненте *Label* присвоим полученное значение.

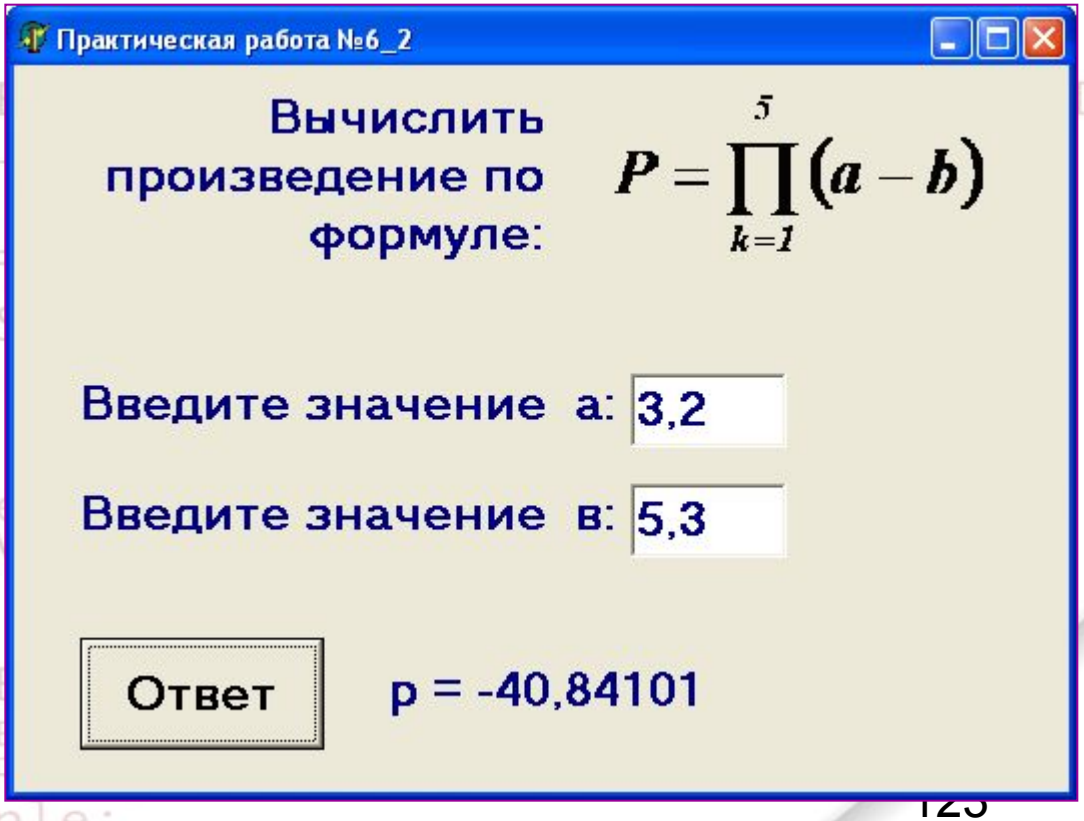
Установим значения свойств компонент и опишем порядок выполнения событий для проекта

компонент	*Object Inspector*	*Свойство (Properties)\ Событие (Events)*	*Значение свойства\ Обработка события*
Form1	Properties	Caption	*Практическая работа №6_1*
Label1	Properties	Caption	*Вычислить сумму по формуле*
Image1	Properties	Picture	*формула.bmp*
		Stretch	*True*
Label2	Properties	Caption	*Введите значение n:*
Label3	Properties	Caption	*<пусто>*
Edit1	Properties	Text	<пусто>
Button1	Properties	Caption	*Ответ*
	Events	OnClick	*procedure TForm1.Button1Click(Sender: TObject);* *var n, s, a: integer;* *Begin* *n:=StrToInt(Edit1.Text);* *s:=0;* *for a:=1 to n do* *s:=s+sqr(a);* *Label3.Caption:='Сумма +IntToStr(s);* *end;*

Практическая работа №9_2

Постановка задачи:

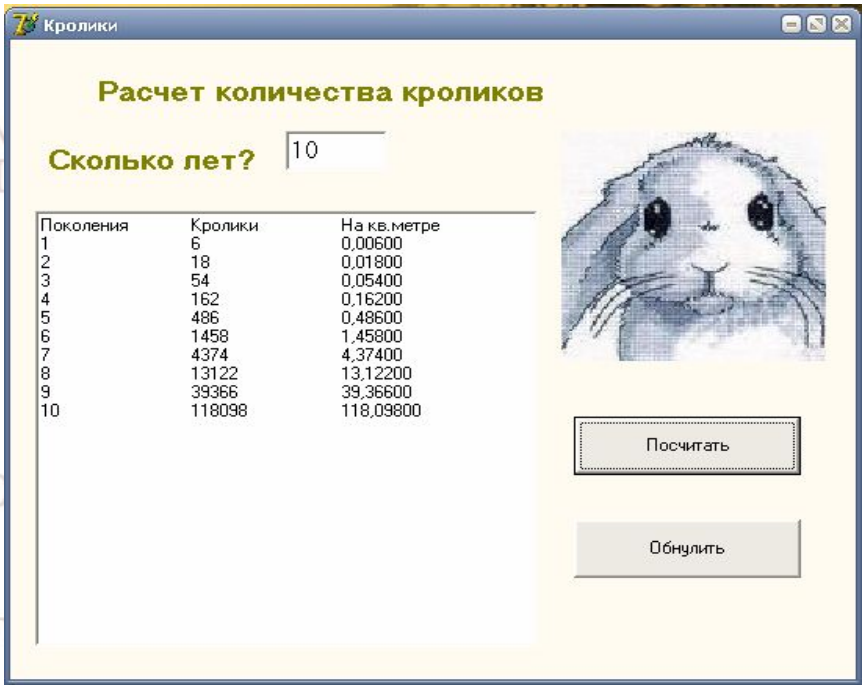
Составить проект, который вычисляет произведение квадратов разности чисел а и в. Предусмотреть поле *ввода для значения* а и в.



Практическая работа №9_3

Постановка задачи:

Составить проект, который вычисляет количество кроликов и плотность их населения на 1 кв.метр, считая, что первоначально их было 6 и каждый год их численность возрастает в 3 раза.



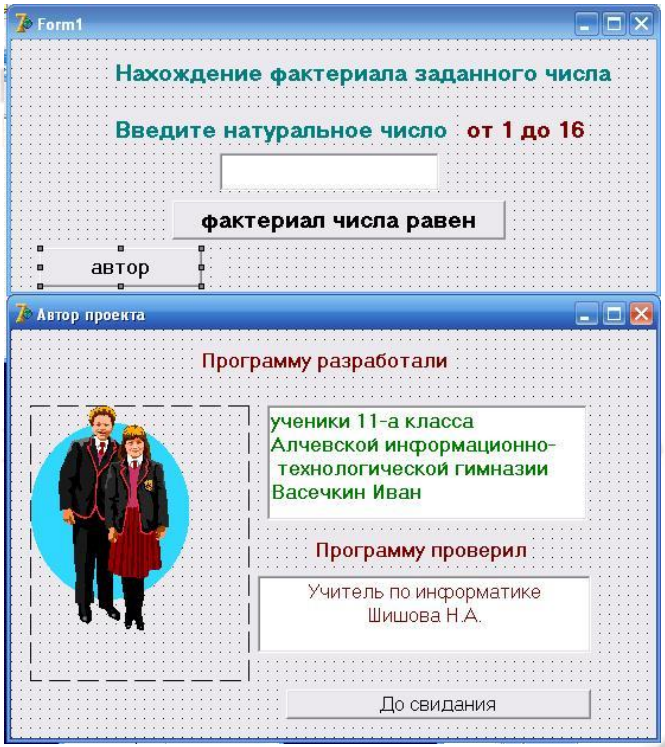
Поколения	Кролики	На кв.метре
1	6	0,00600
2	18	0,01800
3	54	0,05400
4	162	0,16200
5	486	0,48600
6	1458	1,45800
7	4374	4,37400
8	13122	13,12200
9	39366	39,36600
10	118098	118,09800

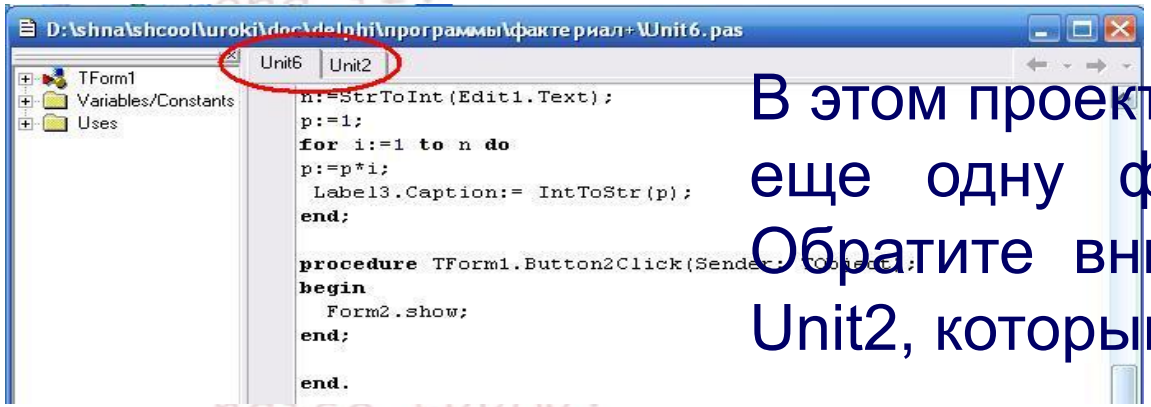
Результат вычисления необходимо отображать в компоненте Memo.
Начальное значение параметра Lines обнулите.
Компонентой Memo можно управлять с помощью внутренних функций: Add-добавить, Clear-очистить и пр.
Для формирования столбцов вывода можно использовать коды управляющих клавиш: Enter - #13, Tab - #9 и др.
Например:

Memo1.Lines.add(IntToStr(i) + #9+#9 + IntToStr(k) + #9+#9+ FloatToStrF(k / 1024,ffFixed,7,5));

Практическая работа №9_4

Постановка задачи:
Составим проект, в котором будет участвовать в работе две формы. На первой форме разместим кнопку вызова вторую. Разместим на второй форме рисунок, комментарии с помощью компоненты Memo.





В этом проекте необходимо создать еще одну форму: File-New-Form. Обратите внимание на появление Unit2, который описывает форму2.

Перейдите на форму1, вызовите *Unit1* для этой формы. Двойным щелчком активизируйте кнопку «Автор». В процедуре, которая обрабатывает эту кнопку нужно написать команду вызова Формы2: *Form2.show.* Так будет выглядеть процедура:

```
procedure TForm1.Button2Click(Sender: TObject);
begin
 Form2.show;
end;
```

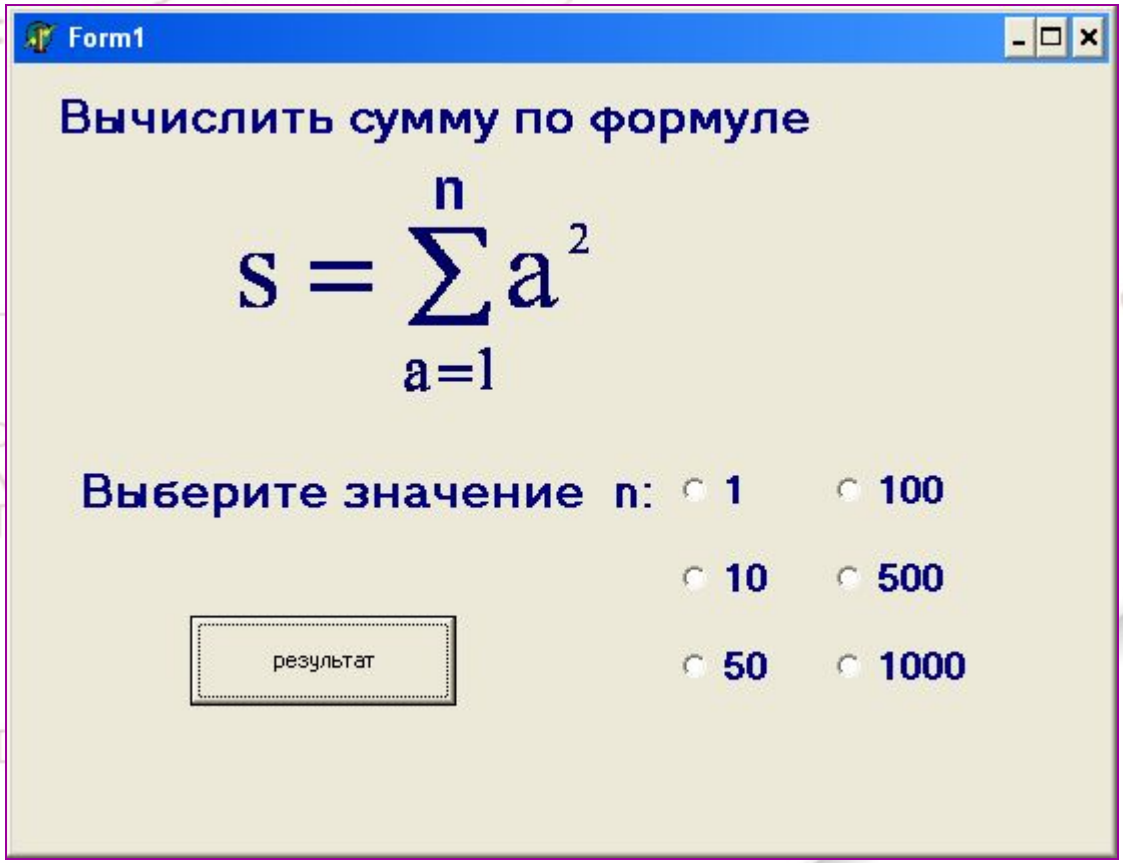
Аналогично на форме2, двойным щелчком активизируйте кнопку «До свидания». В процедуре, которая обрабатывает эту кнопку нужно написать команду закрытия формы: Close; и вызова Формы1: *Form1.show*. Так будет выглядеть процедура:

```
procedure TForm2.Button1Click(Sender: TObject);
begin
 Close;
 Form1.Show;
end;
```

Практическая работа №9_5

Постановка задачи:

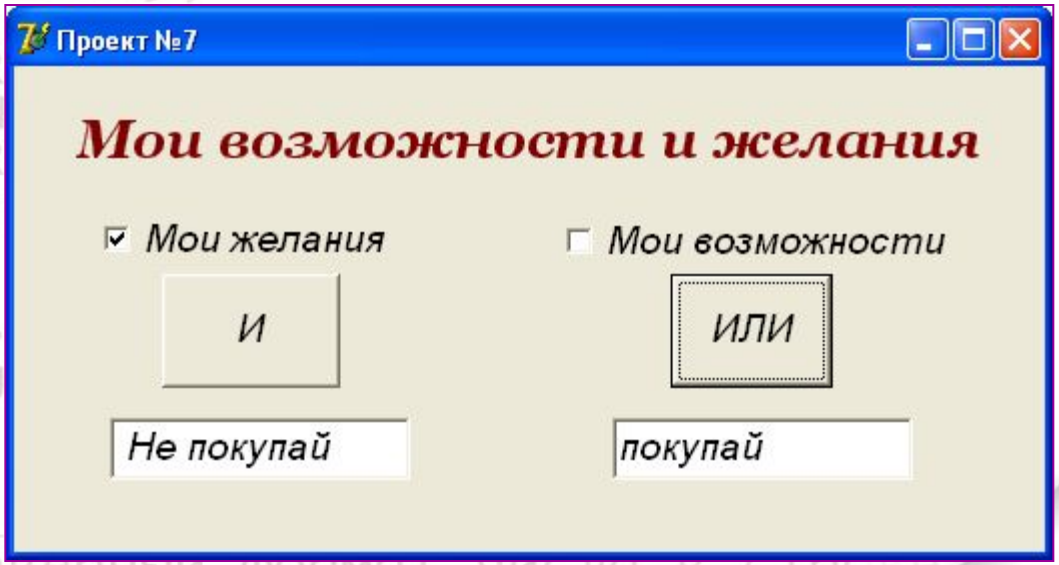
Составить проект, который по указанному числу *n* вычисляет *сумму квадратов* натуральных чисел от *1* до *n*.



Создание проекта №7, используя компоненты индикатор с флажком и радиокнопку.

Постановка задачи:

Указывая на наличие желания и возможностей, определить, можно ли выполнить покупку.



1 этап. Формулировка задания, для которого реализуется проект

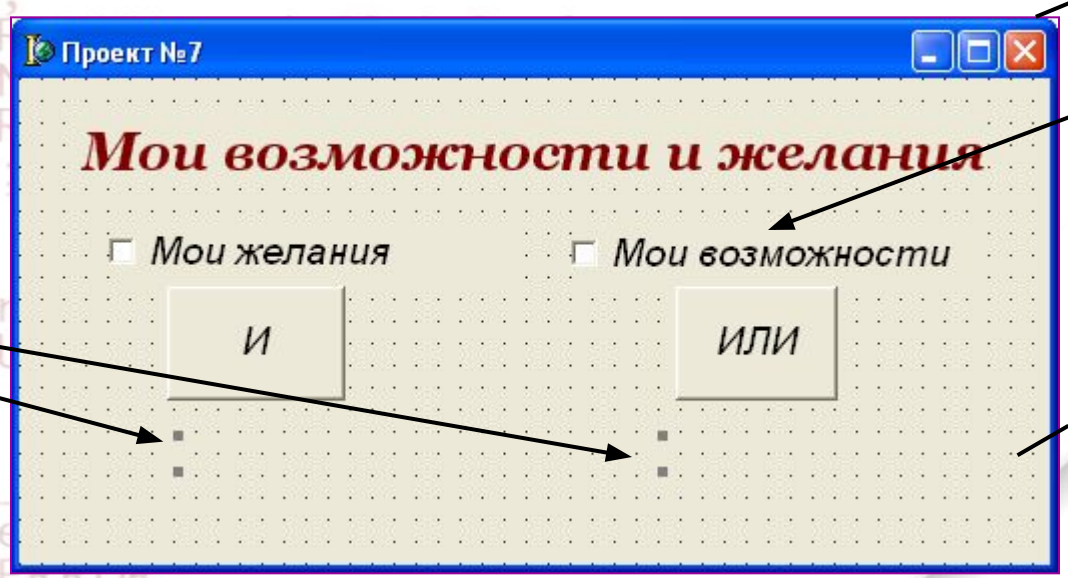
Анализ задачи:

Если два флажка будут включены, то по нажатию на кнопку *«И»* необходимо вывести текст «*Покупай*», в других комбинациях – «*Не покупай*». В программировании это можно реализовать стандартной связкой «*И*».

Если хотя бы один из флажков будет включен, то по нажатию на кнопку «*ИЛИ*» необходимо вывести текст «*Покупай*», в других комбинациях – «*Не покупай*». В программировании это можно реализовать стандартной связкой «*ИЛИ*».

2 этап. Проведение проектного анализа и формирование требований к объектам

1) Создайте чистую *форму*.
2) На форме разместите и настройте компоненты: *Label1* - название проекта; *Label2*, *Label3* – место вывода результата логических вычислений.
3) Разместите на форме переключатели *CheckBox1* и *CheckBox2* - для обработки ситуации.
4) Разместите кнопки *Button1 «И»* , *Button2 «ИЛИ»* – для выполнения логических вычислений.
5) Все компоненты оформите по своему усмотрению: цвет, шрифт, размер.



Form

Label

CheckBox

Button

3 этап. Выбор необходимых компонентов и разработка алгоритмов обработки компонентов

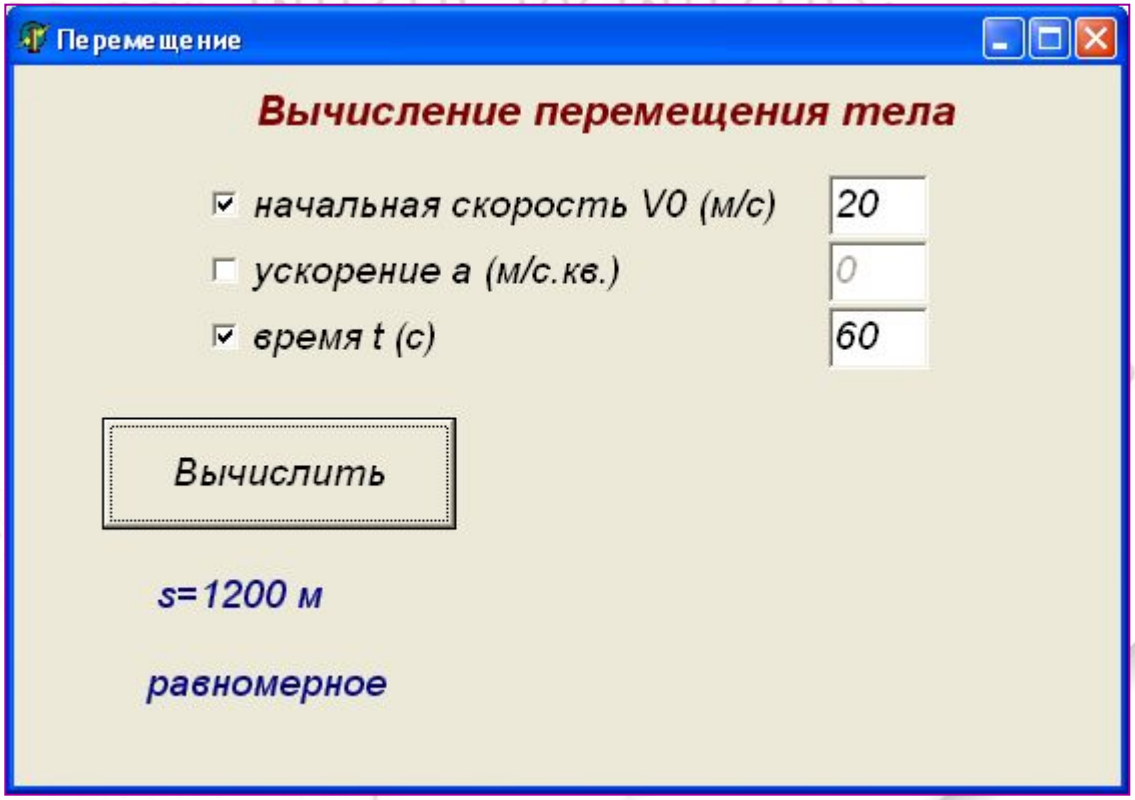
компонент	Object Inspector	Свойство (Properties)\ Событие (Events)	Значение свойства\ Обработка события
Form1	Properties	Name	Form1
		Caption	Проект №7
Label1	Properties	Caption	Мои возможности и желания
Label2	Properties	Caption	<пусто>
Label3	Properties	Caption	<пусто>
CheckBox1	Properties	Caption	Мои желания
	Events	OnClick	procedure TForm1.CheckBox1Click(Sender: TObject); begin end;
CheckBox2	Properties	Caption	Мои возможности
	Events	OnClick	procedure TForm1.CheckBox2Click(Sender: TObject); begin end;

компонент	*Object Inspector*	*Свойство (Properties)\ Событие (Events)*	*Значение свойства\ Обработка события*
Если обе компоненты *CheckBox1* и *CheckBox2* имеют значение *true*, то *Button1 «И»* определяет значение *«Покупай»,* в любой другой комбинации – *«Не покупай!»*			
Button1	**Properties**	**Caption**	***И***
	Events	**OnClick**	***procedure TForm1.Button1Click(Sender: TObject); begin if (CheckBox1.Checked=True) and (CheckBox2.Checked=True) then Label2.Caption:='покупай' else Label2.Caption:='Не покупай'; end;***
Если хоть одна компонента *CheckBox1* или *CheckBox2* имеют значение *true*, то *Button2 «ИЛИ»* определяет значение *«Покупай»,* в любой другой комбинации – *«Не покупай!».*			
Button2	**Properties**	**Caption**	***ИЛИ***
	Events	**OnClick**	***procedure TForm1.Button2Click(Sender: TObject); Begin if (CheckBox1.Checked=True) or (CheckBox2.Checked=True) then Label3.Caption:='покупай' else Label3.Caption:=' Не покупай'; end;***

Практическая работа №7_1

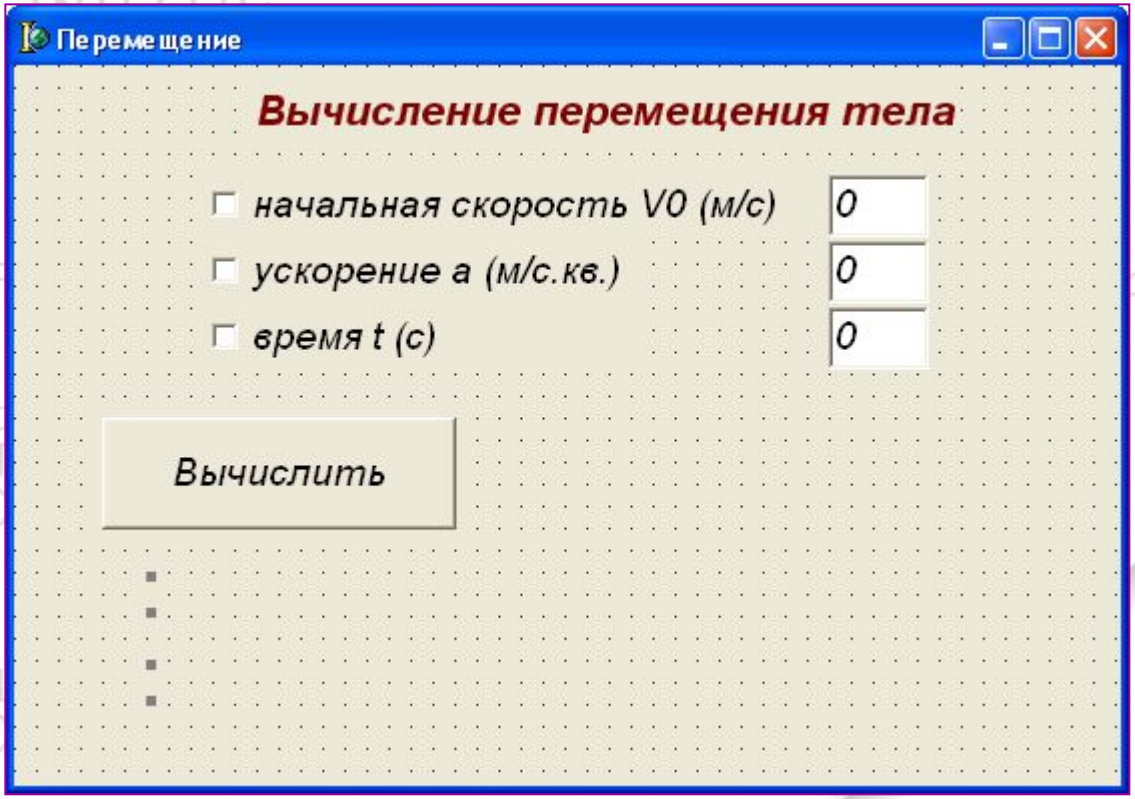
Постановка задачи:

Составим проект, который вычисляет пройденное телом *расстояние* в зависимости от *вида движения* (равномерное или равноускоренное).



1 этап. Формулировка задания, для которого реализуется проект

Проект предполагает ввод начальной скорости тела, его ускорение и время перемещения; вывод результата вычисления перемещения и его вида.



Анализ задачи:

Необходимо разработать форму, на которой будет размещены: тексты, переключатели для выбора исходных значений, окна ввода значений *v, t, a*, место вывода ответа и кнопки, по нажатию на которые происходит вывод результата вычисления *s* и анализа результата.

3 этап. Выбор необходимых компонентов и разработка алгоритмов обработки компонентов

компонент	Object Inspector	Свойство (Properties)\ Событие (Events)	Значение свойства\ Обработка события
Form1	Properties	Caption	Перемещение
Label1	Properties	Caption	Вычисление перемещения тела
Label2	Properties	Caption	<пусто>
Label3	Properties	Caption	<пусто>
Edit1	Properties	Text	0
Edit2	Properties	Text	0
Edit3	Properties	Text	0
Если переключатель CheckBox1 включен (CheckBox1.Checked=True), то компонент Edit1 активен (Edit1.Enabled:=True), в противном случае (Edit1.Enabled:=False) – неактивен, компонент Edit1 принимает значение 0 (Edit1.Text:='0').			
CheckBox1	Properties	Caption	начальная скорость V0 (м/с)

	Events	OnClick	procedure TForm1.CheckBox1Click(Sender:TObject);
			begin
			if CheckBox1.Checked=False then
			begin
			Edit1.Enabled:=False;
			Edit1.Text:='0';
			end
			else Edit1.Enabled:=True;
			end;

компонент	*Object Inspector*	*Свойство (Properties)\ Событие (Events)*	*Значение свойства\ Обработка события*
Если переключатель CheckBox2 включен (CheckBox2.Checked=True), то компонент Edit2 активен (Edit1.Enabled:=True), в противном случае (Edit2.Enabled:=False) – неактивен, компонент Edit2 принимает значение 0 (Edit2.Text:='0').			
CheckBox2	Properties	Caption	*ускорение a (м/с.кв.)*

	Events	OnClick	*procedure TForm1.CheckBox2Click(Sender:TObject);
			begin
			if CheckBox2.Checked=False then
			begin
			Edit2.Enabled:=False;
			Edit2.Text:='0';
			end
			else Edit2.Enabled:=True;
			end;*

Если переключатель CheckBox3 включен (CheckBox3.Checked=True), то компонент Edit3 активен (Edit3.Enabled:=True), в противном случае (Edit3.Enabled:=False) – неактивен, компонент Edit1 принимает значение 0 (Edit3.Text:='0').			
CheckBox3	Properties	Caption	*время t (c)*

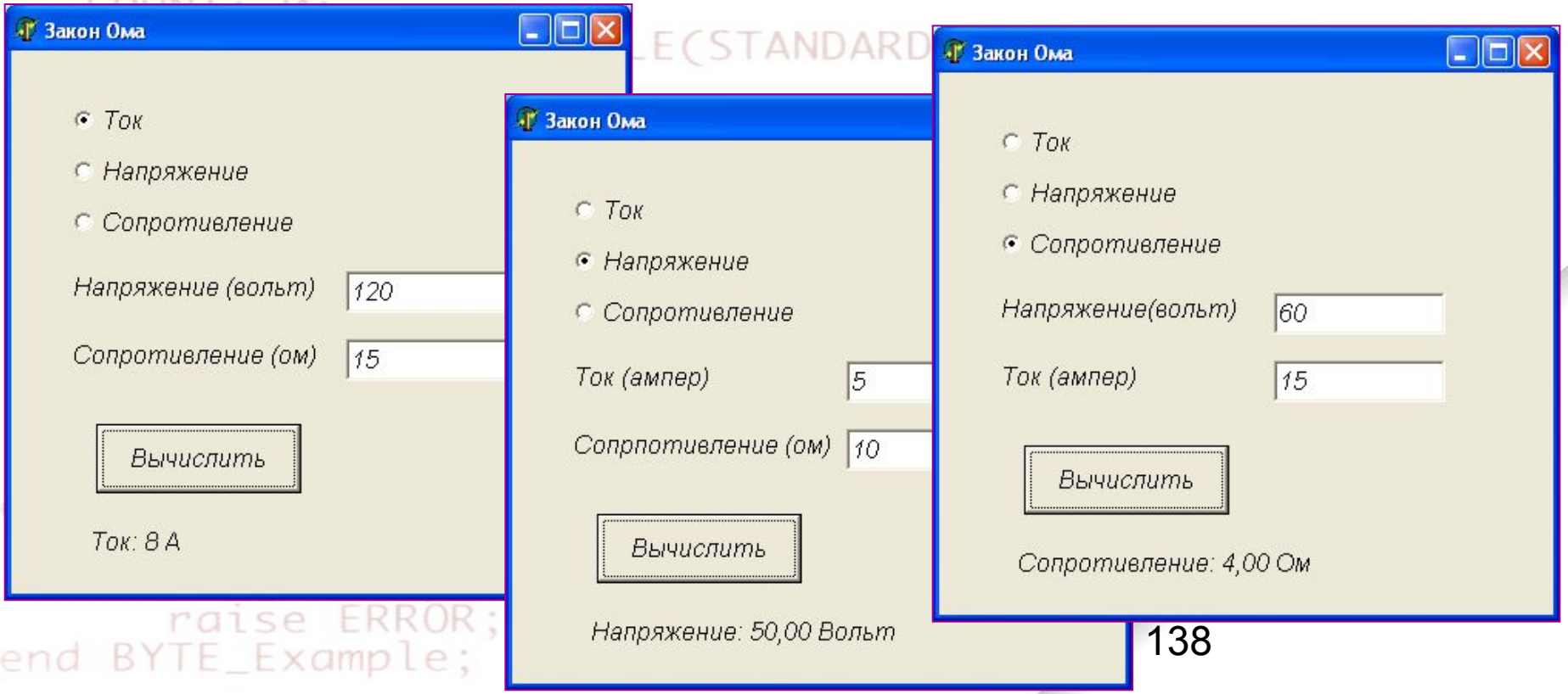
	Events	OnClick	*procedure TForm1.CheckBox3Click(Sender:TObject);
			begin
			if CheckBox3.Checked=False then
			begin
			Edit3.Enabled:=False;
			Edit3.Text:='0';
			end
			else Edit3.Enabled:=True;
			end;*

компонент	*Object Inspector*	*Свойство (Properties)\ Событие (Events)*	*Значение свойства\ Обработка события*
Button1	Properties	Caption	*Вычислить*
	Events	OnClick	*procedure TForm1.Button1Click(Sender: TObject);* *begin* *v0:=StrToFloat(Edit1.Text);* *a:=StrToFloat(Edit2.Text);* *t:=StrToFloat(Edit3.Text);* *s:= v0*t+ a*sqr(t)/2;* *Label2.Caption:='s='+FloatToStr(s)+' м';* *if a=0 then Label3.Caption:='равномерное'* *else* *if a>0 then Label3.Caption:='равноускоренное'* *else Label3.Caption:='равнозамедленное'* *end;*

Практическая работа №7_2

Постановка задачи:

- Вычислять *силу тока* по известному *напряжению* и *сопротивлению*.
- Вычислять *напряжение* тока по известному *сопротивлению* и *силе тока*.
- Вычислять *сопротивление* проводника по известному *напряжению* и *силе тока*.



1 этап. Формулировка задания, для которого реализуется проект

Анализ задачи:

В задаче необходимо использовать *переключатель-радиокнопку*, в зависимости от включения которой, производится необходимое вычисление.

1. Если включена *«Сила тока»,* то запрашивать напряжению и сопротивлению, вычислять силу тока.
2. Если включено «*Напряжение тока*», то запрашивать силу тока и сопротивлению, вычислять напряжение тока.
3. Если включено «*Сопротивление*», то запрашивать силу тока и напряжение, вычислять сопротивление.

3 этап. Выбор необходимых компонентов и разработка алгоритмов обработки компонентов

компонент	Object Inspector	Свойство (Properties)\ Событие (Events)	Значение свойства\ Обработка события
Form1	Properties	Caption	Закон Ома
Label1	Properties	Caption	Напряжение (вольт)
Label2	Properties	Caption	Сопротивление (ом)
Label3	Properties	Caption	<пусто>
Edit1	Properties	Text	<пусто>
Edit2	Properties	Text	<пусто>
Если выбран переключатель CheckBox1 «Ток», то компонент Edit1 и Edit2 очищаются (Edit1.Text:= ' '), компонент Label1 принимает значение 'Напряжение (вольт)', а компонент Label2 принимает значение 'Сопротивление (ом)'.			
CheckBox1	Properties	Caption	Ток
	Events	OnClick	Label1.Caption:='Напряжение (вольт)'; Label2.Caption:='Сопротивление (ом)'; Label3.Caption:=' '; Edit1.Text:=' '; Edit2.Text:=' ';

компонент	Object Inspector	Свойство (Properties)\ Событие (Events)	Значение свойства\ Обработка события
Если выбран переключатель CheckBox2 «Напряжение», то компонент Edit1 и Edit2 очищаются (Edit1.Text:= ' '), компонент Label1 принимает значение Ток(ампер)', а компонент Label2 принимает значение 'Сопротивление (ом)‘.			
CheckBox2	Properties	Caption	Напряжение
	Events	OnClick	Label1.Caption:='Ток(ампер)'; Label2.Caption:= 'Сопротивление (ом)'; Label3.Caption:=' '; Edit1.Text:=' '; Edit2.Text:=' ';
Если выбран переключатель CheckBox3 «Сопротивление», то компонент Edit1 и Edit2 очищаются (Edit1.Text:= ' '), компонент Label1 принимает значение 'Напряжение (вольт)‘, а компонент Label2 принимает значение Ток(ампер)'			
CheckBox3	Properties	Caption	Сопротивление
	Events	OnClick	Label1.Caption:='Напряжение (вольт)'; Label2.Caption:= 'Ток(ампер)'; Label3.Caption:=‘ '; Edit1.Text:=‘ '; Edit2.Text:=‘ ';

<table>
<tr><th>компонент</th><th>Object Inspector</th><th>Свойство (Properties)\ Событие (Events)</th><th>Значение свойства\ Обработка события</th></tr>
<tr><td rowspan="3">Button1</td><td>Properties</td><td>Caption</td><td>Вычислить</td></tr>
<tr><td rowspan="2">Events</td><td colspan="2">Программная обработка выбранной кнопки производится проверкой параметра Checked компоненты RadioButton: Если RadioButton1.Checked=true, то кнопка включена.</td></tr>
<tr><td>OnClick</td><td>if RadioButton1.Checked=true then
begin
U:= StrToFloat(Edit1.Text);
R:= StrToFloat(Edit2.Text);
I:= U/R;
Label3.Caption:= 'Ток: ' + FloatToStrF(I,ffFixed,4,2) + 'A';
end;
if RadioButton2.Checked=true then
begin
I:= StrToFloat(Edit1.Text);
R:= StrToFloat(Edit2.Text);
U:= I*R;
Label3.Caption:= 'Напряжение: ' + FloatToStrF(I,ffFixed,4,2) + ' Вольт';
end;
if RadioButton3.Checked=true then
begin
U:= StrToFloat(Edit1.Text);
i:= StrToFloat(Edit2.Text);
r:= U/i;
Label3.Caption := 'Сопротивление ' + FloatToStrF(r,ffFixed,4,2) + ' ом';
end;</td></tr>
</table>

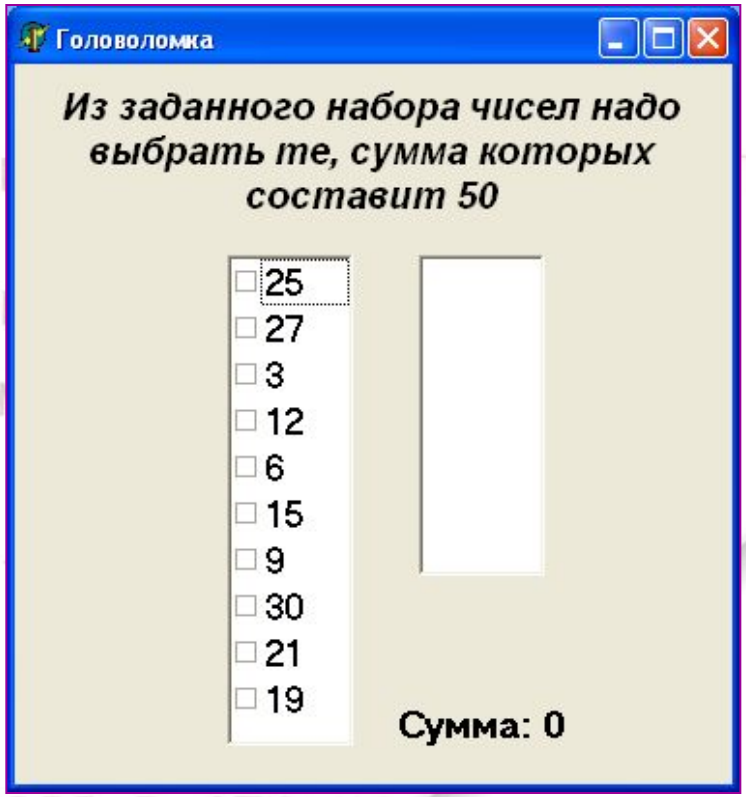
Практическая работа №7_3

Постановка задачи: Составить проект, который позволяет выбрать страну из перечисленных, указать какие сведения о стране необходимы и вывести их.

Создание проекта №8, используя компоненты индикатор с флажком и радиокнопку.

Практическая работа №8_1

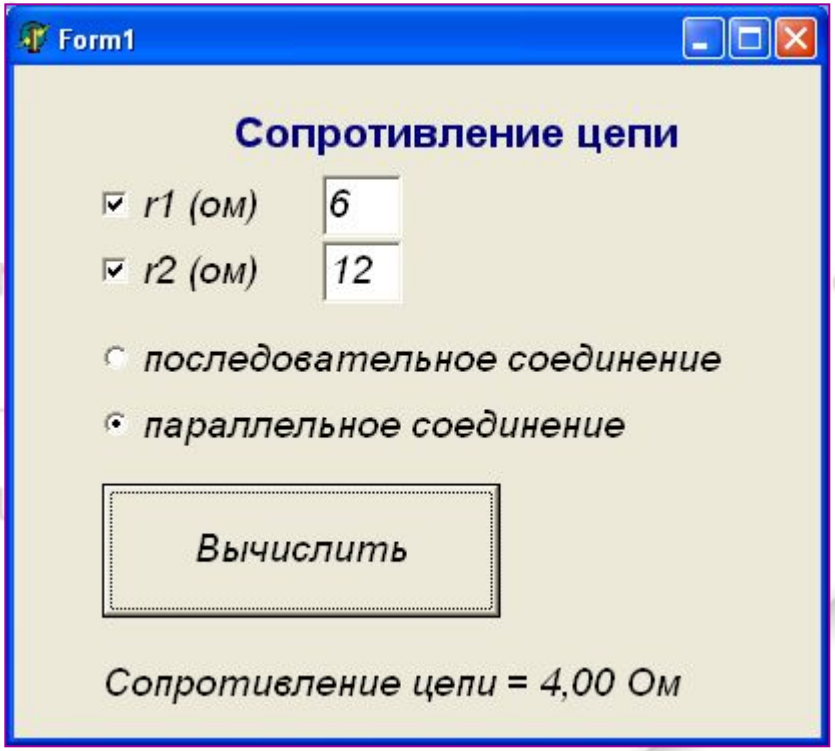
Постановка задачи:
Составим проект, который загадает головоломку:
какой набор чисел даст в сумме 50.
(Ответ: 25, 6, 19)



?2 ?3

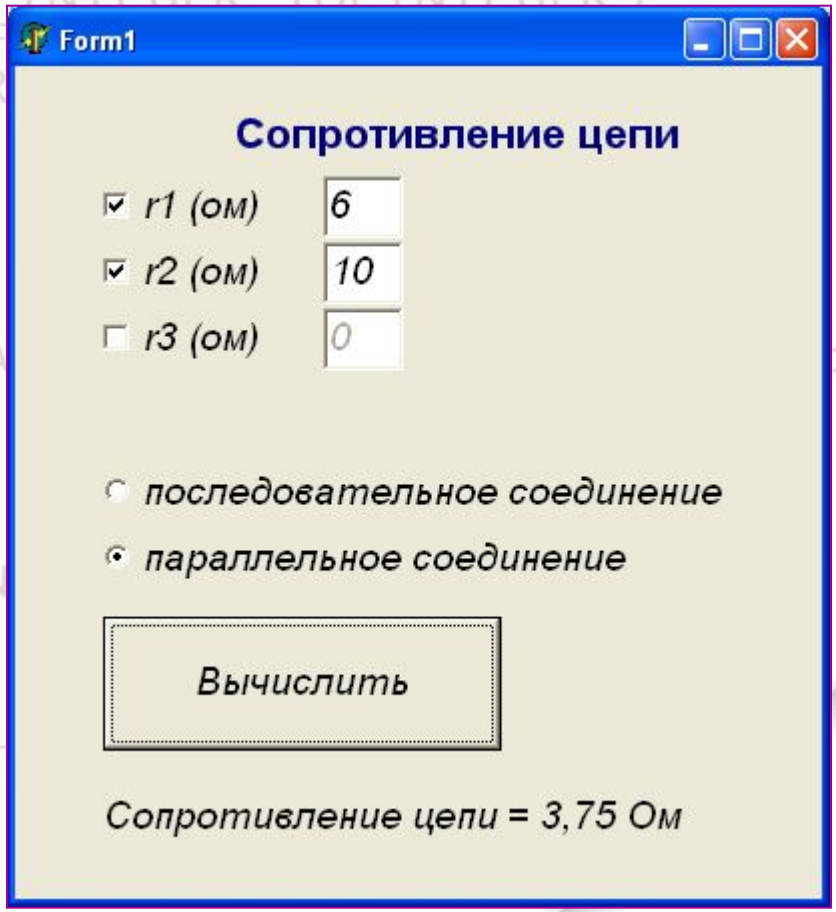
Практическая работа №8_2 (1 вариант)

Постановка задачи:
Разработайте проект по вычислению *общего сопротивления участка электрической цепи.*
Организовать выбор вида соединения сопротивлений и их количество (максимально 2 шт).



Практическая работа №8_2 (2 вариант)

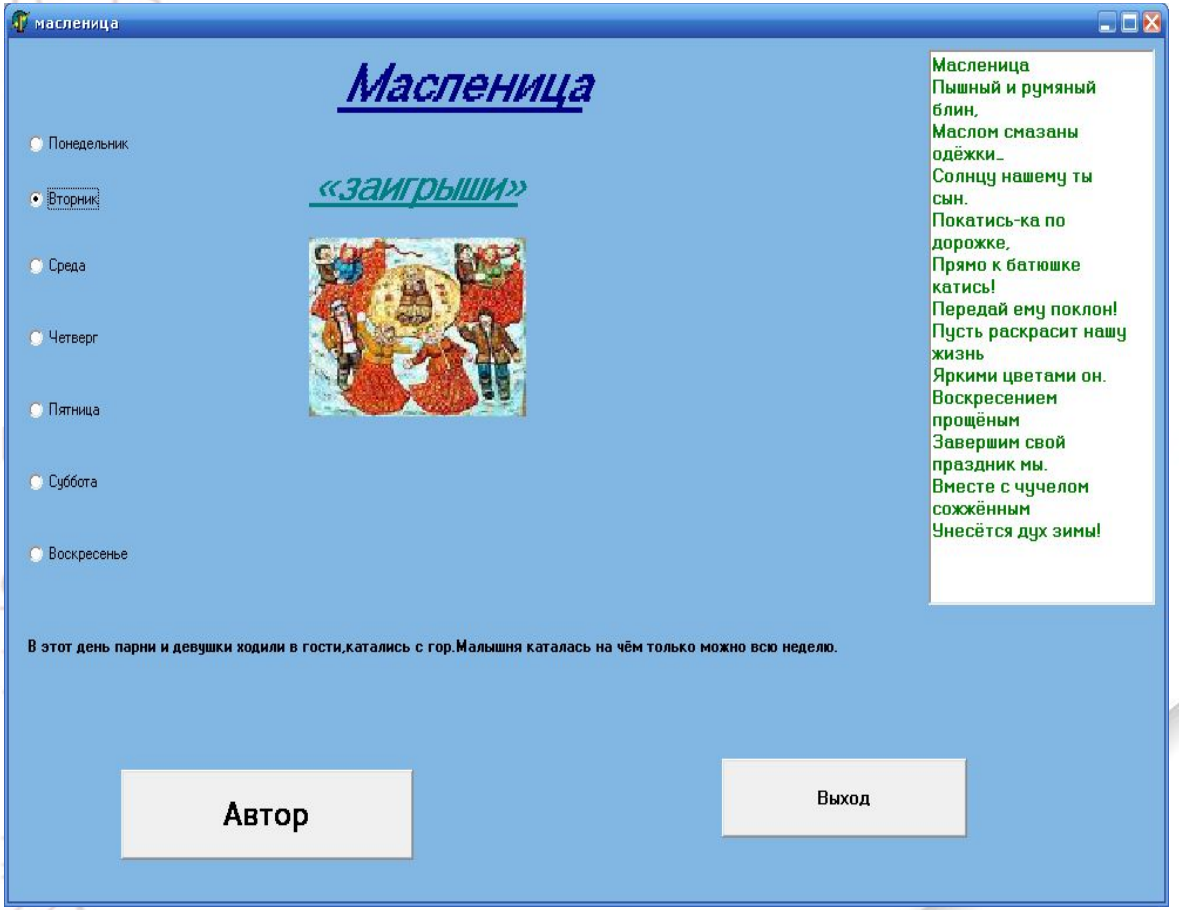
Постановка задачи:
Разработайте проект по вычислению *общего сопротивления участка электрической цепи.*
Организовать выбор вида соединения сопротивлений и их количество (максимально 3 шт).



Практическая работа №8_3

Постановка задачи

Составить проект, который позволяет выводить информацию и рисунки по празднованию русского праздника «Масленица». В зависимости от выбранного дня выводится название дня, рисунок и текст с описанием традиции этого дня праздника.



Создание проекта №10, работа с одномерными массивами.

Информация, объединенная единым смыслом или математическим содержанием (списки фамилий, матрица коэффициентов) относится к структурированному типу данных.

*Для представления таких данных используются **массивы**.*

Массив – совокупность однотипных элементов, с каждым из которых связан упорядоченный набор индексов (номеров).

*Каждый массив имеет свое **имя**.*

*Общий вид имени элемента массива - **A[i],***

где

***A** - имя массива,*

***i** – имя индекса. Индекс может быть **целым, переменным, арифметическим выражением**.*

Объявление массива

Массив перед использованием должен быть объявлен в разделе объявления переменных.

В общем виде инструкция объявления одномерного массива выглядит следующим образом:

имя: array [нижний_индекс. .верхний_индекс] of тип

где:

имя — имя массива;

array — зарезервированное слово языка Delphi, обозначающее, что объявляемое имя является именем массива;

нижний_индекс и верхний_индекс — целые константы, определяющие диапазон изменения индекса элементов массива и количество элементов (размер) массива;

тип — тип элементов массива.

Примеры объявления массивов:

temper: array [1..31] of real;

koef: array [0. .2] of integer;

name: array [1..30] of string [25];

Если массив не является локальным, т. е. объявлен не в процедуре обработки события, а в разделе переменных модуля, то одновременно с объявлением массива можно выполнить его инициализацию, т. е. присвоить начальные значения элементам массива. Инструкция объявления массива с одновременной его инициализацией в общем виде выглядит так:

Имя: array [нижний_индекс..верхний_индекс] of тип = (список);

где

список *— разделенные запятыми значения элементов массива.*

Например:

```
a: array [1..10] of integer = (0,0,0,0,0,0,0,0,0,0);
day: array [1..7] of String [10]=('Пн','Вт','Ср','Чт','Пт','Сб','Вс');
```

Количество элементов списка инициализации должно соответствовать размерности массива. Если это будет не так, то компилятор выведет сообщения об ошибке: ***Number of elements differs from declaration*** *(количество элементов не соответствует указанному в объявлении).*

Вывод массива

Под выводом массива понимается вывод на экран монитора (в диалоговое окно) значений элементов массива.

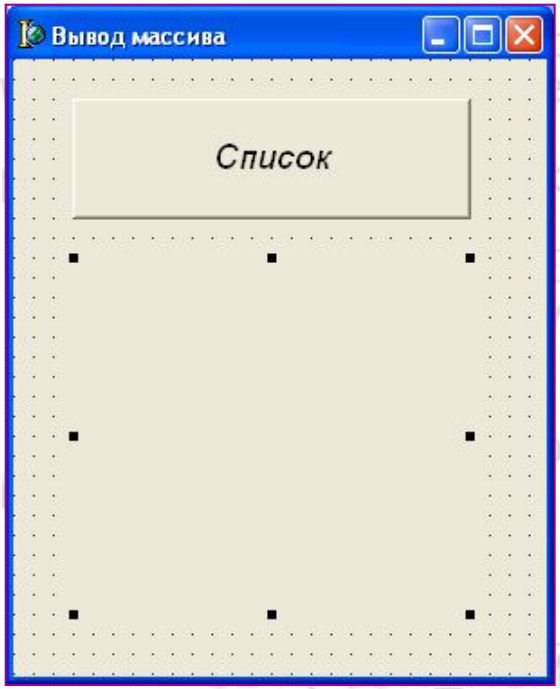
Один из вариантов организации вывода массива - использование компонента ***Label.***

Основной цикл процедуры вывода массива A с использованием компоненты Label может выглядеть так:

- *Для одномерного массива*

```
for i:=1 to n do
Label1.Caption:=Label1.Caption + IntToStr(i)+ ' '+ a[i]+ #13;
```

где #13 – перевод курсора на новую строку

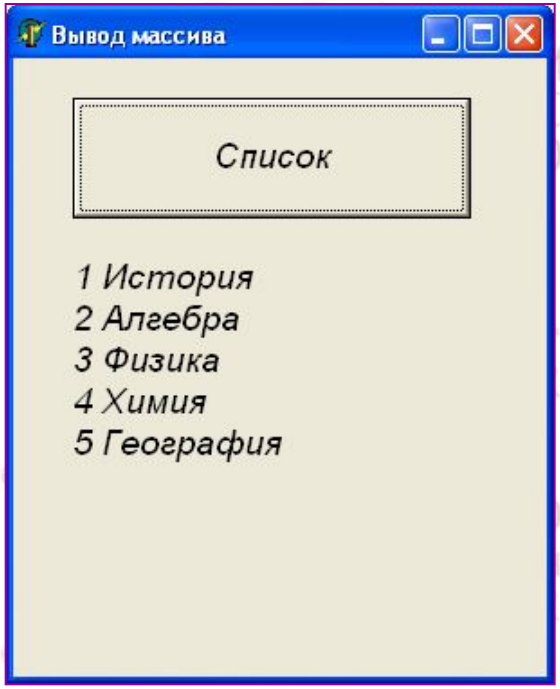


Form

Button

Label

Постановка задачи:
Создать программу вывода пронумерованного списка школьных предметов.



Листинг программы. Инициализация и вывод массива

```
const
n = 5;
var
a: array[1..n] of string[10] =('История', 'Алгебра',
'Физика', 'Химия', 'География');

procedure TForm1.Button1Click(Sender: TObject);
var
i: integer;
begin
for i:=1 to n do
Label1.Caption:=Label1.Caption+IntToStr(i)+' '+a[i]+#13;
end;
```

Ввод массива

Под вводом массива понимается процесс получения от пользователя во время работы программы значений элементов массива. Последовательность чисел удобно вводить в строку таблицы, где каждое число находится в отдельной ячейке.

Два варианта организации ввода массива:

1. *с использованием компонента* ***StringGrid***
2. *с использованием компонента* ***Memo.***

Использование компонента Memo

*Компонент **Memo** позволяет вводить текст, состоящий из достаточно большого количества строк, поэтому его удобно использовать для ввода **символьного** массива.*

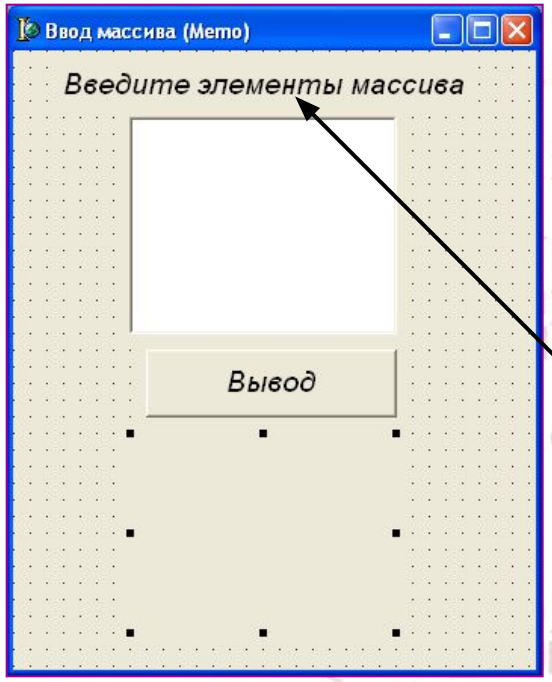
При использовании компонента Memo для ввода массива значение каждого элемента массива следует вводить в отдельной строке и после ввода каждого элемента массива нажимать клавишу <Enter>.

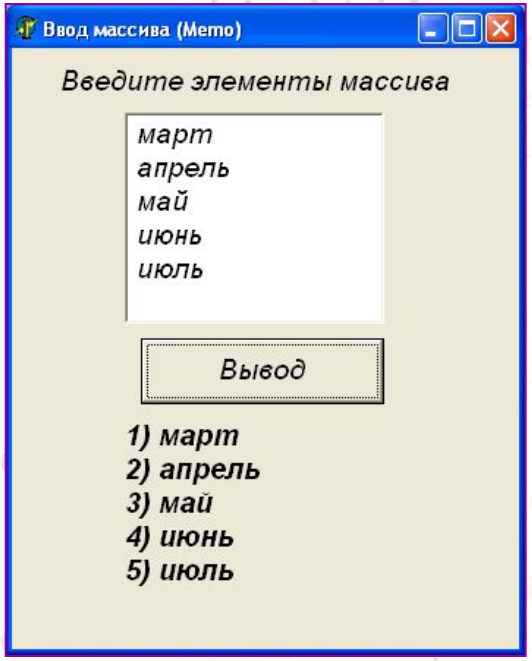
*Получить доступ к находящейся в поле Memo строке текста можно при помощи свойства **Lines**, указав в квадратных скобках номер нужной строки (строки нумеруются с нуля).*

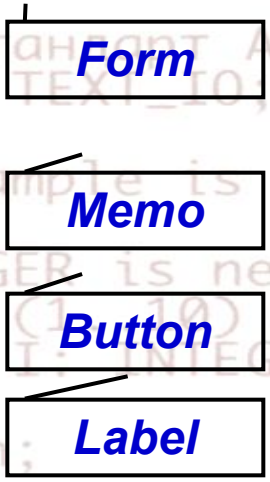
Основной цикл процедуры ввода символьного массива из компонента Memo может выглядеть так:

```
for i:=1 to N do
a[i]:= Memo1.Lines[i];
```

Lines — свойство компонента Memo, представляющее собой массив, каждый элемент которого содержит одну строку находящегося в поле Memo текста.







Постановка задачи:
Создать программу ввода пронумерованного списка названий месяцев.

Листинг программы. Ввод массива строк из компонента Memo

```
procedure TForm1.Button1Click(Sender: TObject);
const
SIZE=5;
var
a: array[1..SIZE]of string [30];
n: integer;
i: integer;
begin
Label2.Caption:='';
n:=Memo1.Lines.Count;
for i:=1 to n do
a[i]:=Form1.Memo1.Lines[i-1];
for i:=1 to n do
Label2.Caption:=Label2.Caption+IntToStr(i)+') '+a[i]+#13;
end;
```

Использование компонента StringGrid

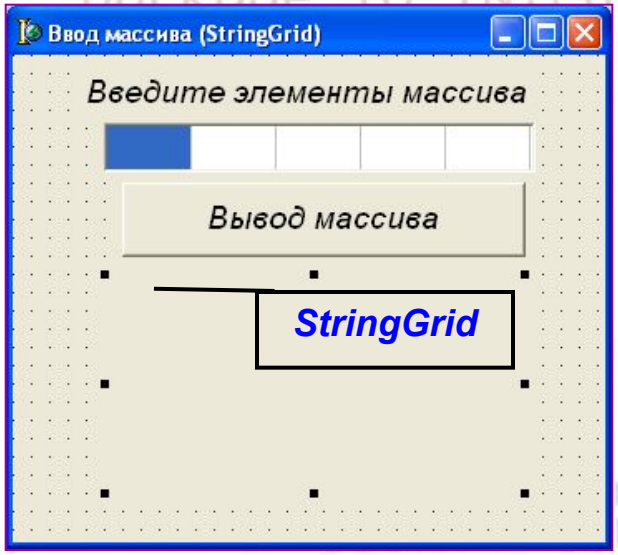
свойство ***Cells*** *- соответствующий таблице двумерный массив. Ячейка таблицы, находящаяся на пересечении столбца номер* ***col*** *и строки номер* ***row*** *определяется элементом* ***cells [col, row]***

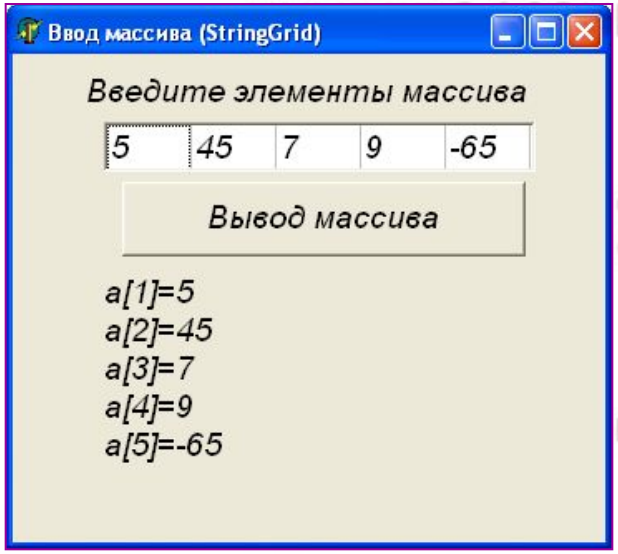
Основной цикл процедуры ввода массива из компонента StringGrid может выглядеть так:

- *Для одномерного массива:*

```
for i:=1 to N do
a[i]:= StrToInt(StringGrid1.Cells[i-1,0]);
```

Постановка задачи:
Создать программу ввода пронумерованного списка целых чисел.





Листинг программы.
Ввод массива чисел из компонента StringGrid

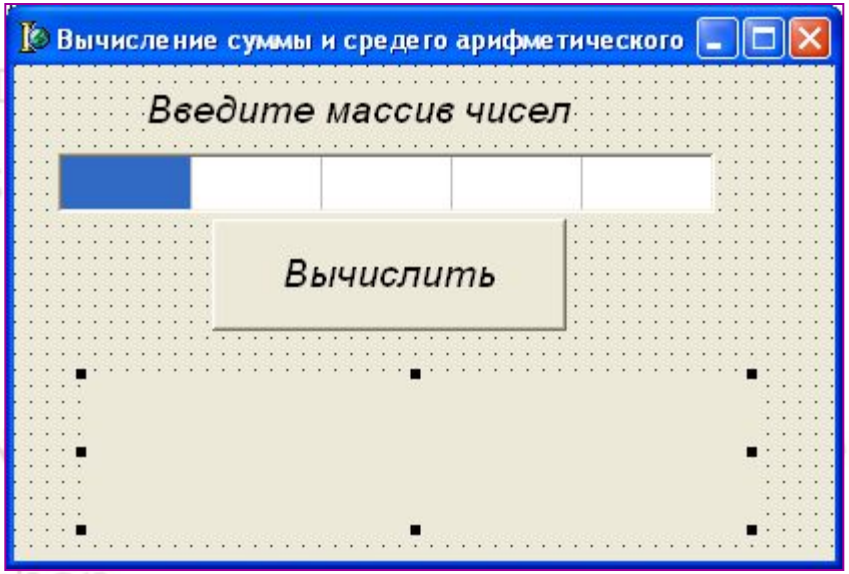
```
procedure TForm1.Button1Click(Sender: TObject);
const
n=5;
var
a: array [1..n] of integer;
i: integer;
begin
for i:=1 to n do
a[i]:=StrToInt(StringGrid1.Cells[i-1,0]);
for i:=1 to n do
Label2.Caption:=Label2.Caption+'a['+IntToStr(i)+']='
+IntToStr(a[i])+#13;
end;
```

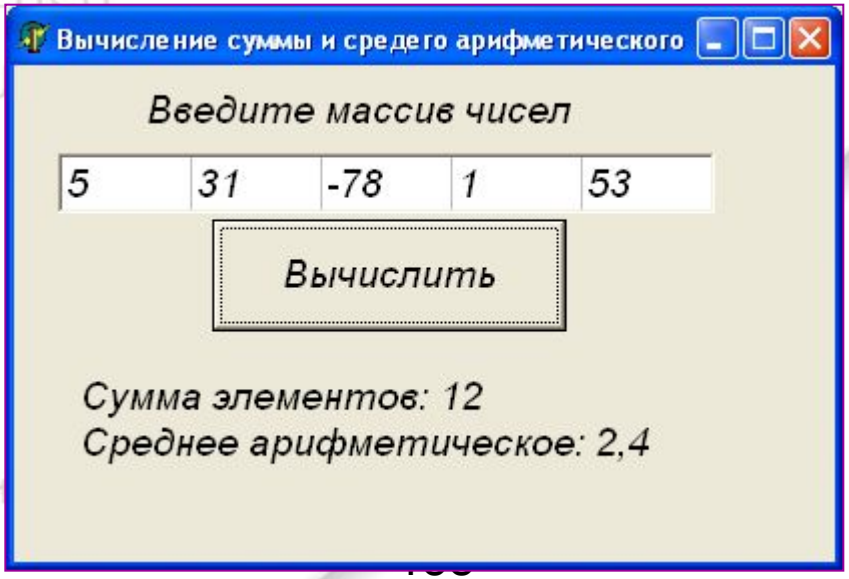
Значения свойств компонента StringGrid1

Свойство	Значение
ColCount	5
FixedCols	0
RowCount	1
DefaultRowHeight	24
DefaultColWidth	64
Options . goEditing	True
Options . AlwaysShowEditing	True
Options .goTabs	

Практическая работа №10_1

Постановка задачи:
Разработайте проект по вычислению *суммы* и *среднего арифметического* значения одномерного массива целых чисел.





Создание проекта №11, работа с двумерными массивами.

Для представления табличных данных используются ***двумерные массивы.***

Двумерный массив – совокупность однотипных элементов, с каждым из которых связан упорядоченный набор индексов, первый из которых определяет порядковый номер строки, второй – номер столбцы, на пересечении которых находится элемент.

I,j	1	2	3	4
1	7	8	6	0
2	8	5	1	0
3	6	7	3	2

Каждый массив имеет свое ***имя.***

Общий вид имени элемента массива - ***A[i,j],***

где

A *- имя массива,*

I, j *– имя индексов.* ***I – номер строки, j – номер столбца.*** *Индексы могут быть* ***целыми числами, переменными, арифметическими выражениями.***

Объявление массива

Массив перед использованием должен быть объявлен в разделе объявления переменных.

В общем виде инструкция объявления одномерного массива выглядит следующим образом:

имя: array [н_инд_строк. .в_инд_ строк, н_инд_столбцов. .в_инд_столбцов] of тип

где:

имя — имя массива;

array — зарезервированное слово языка Delphi, обозначающее, что объявляемое имя является именем массива;

нижний_индекс и верхний_индекс — целые константы, определяющие диапазон изменения индексов элементов массива и количество элементов (размер) массива;

тип — тип элементов массива.

Примеры объявления массивов:

temper: array [1..31, 1..10] of real; - таблица вещественных чисел(31x10)

koef: array [0. .2, 0..5] of integer; - таблица целых чисел(3x6)

name: array [1..30, 1..15] of string [25]; - таблица строк (30x15)

Если массив не является локальным, т. е. объявлен не в процедуре обработки события, а в разделе переменных модуля, то одновременно с объявлением массива можно выполнить его инициализацию, т. е. присвоить начальные значения элементам массива. Инструкция объявления массива с одновременной его инициализацией в общем виде выглядит так:

Имя: array [н_индСтр..в_индСтр, н_индСтолб..в_индСтолб] of тип = (список);

где

список *— разделенные запятыми значения элементов массива.*

Например:

```
a: array [1..2, 1..3] of integer = ((0,0,0),(0,0,0));
day: array [1..3, 1..2] of String [10]=(('Пн','Вт'),('Ср','Чт'),('Пт','Сб'));
```

Количество элементов списка инициализации должно соответствовать размерности массива. Если это будет не так, то компилятор выведет сообщения об ошибке: ***Number of elements differs from declaration*** *(количество элементов не соответствует указанному в объявлении).*

Вывод массива

Под выводом массива понимается вывод на экран монитора (в диалоговое окно) значений элементов массива.

Один из вариантов организации вывода массива - использование компонента ***Label.***

Основной цикл процедуры вывода массива A с использованием компоненты Label может выглядеть так:

Для двумерного массива

```
for j :=1 to 2 do
begin
for i :=1 to 5 do
Label1.Caption := Label1.Caption +  IntToStr(a[i,j])+'    ';
Label1.Caption := Label1.Caption + #13;
end;
```

Использование компонента StringGrid

*свойство **Cells** - соответствующий таблице двумерный массив. Ячейка таблицы, находящаяся на пересечении столбца номер **col** и строки номер **row** определяется элементом **cells [col, row]***

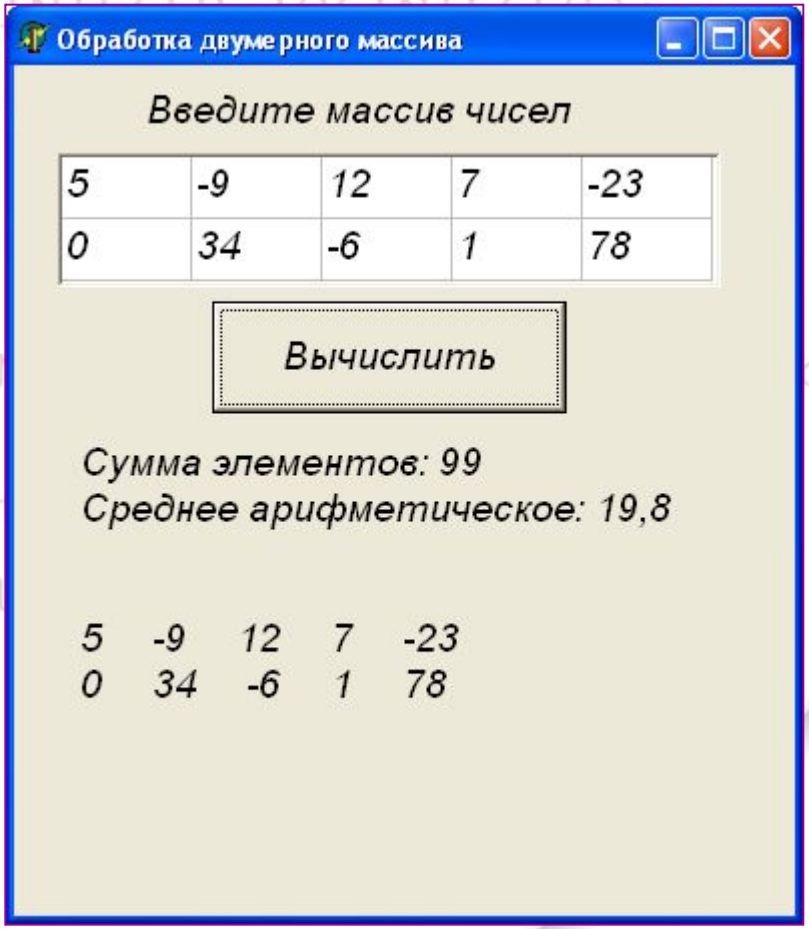
Основной цикл процедуры ввода массива из компонента StringGrid может выглядеть так:

- *Для двумерного массива:*

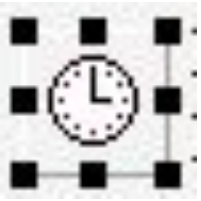
```
for i:=1 to N do
for j:=1 to M do
a[i, j]:= StrToInt(StringGrid1.Cells[i-1,j-1]);
```

Практическая работа №11_1

Постановка задачи:
Разработайте проект по вычислению *суммы* и *среднего арифметического* значения двумерного массива целых чисел.



Создание проекта №12 Игровые ситуации

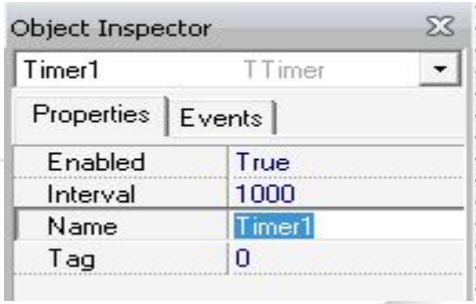


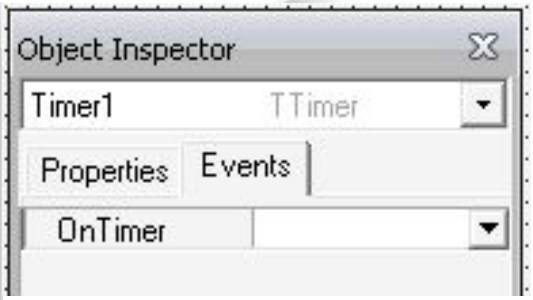
Большинство игр сопровождаются движением объектов. Чтобы заставить двигаться объекты, необходимо менять координаты объекта в определенные моменты времени. Таким образом, в программе необходимо подключить счетчик времени.

Таймер находит многочисленные применения: в мультипликации, закрытие каких-то окон, с которыми пользователь долгое время не работает, включение хранителя экрана или закрытие связей с удаленным сервером при отсутствии действий пользователя, регулярный опрос каких-то источников информации, задание времени на ответ в обучающих программах — все это множество задач, в которых требуется задавать интервалы времени, решается с помощью таймера.

На панели инструментов во вкладке System расположен компонент *Timer.*

Таймер — невизуальный компонент, который может размещаться в любом месте формы. Он имеет два свойства, позволяющие им управлять: *Interval* — интервал времени в миллисекундах и *Enabled* — доступность. Свойство *Interval* задает период срабатывания таймера. Через заданный интервал времени после предыдущего срабатывания, или после программной установки свойства *Interval*, или после запуска приложения, таймер срабатывает, вызывая событие *OnTimer*. В обработчике этого события записываются необходимые операции.





Если задать *Interval = 0* или *Enabled = false*, то таймер перестает работать. Чтобы запустить отсчет времени надо или задать *Enabled = true* и задать положительное значение свойству *Interval* .

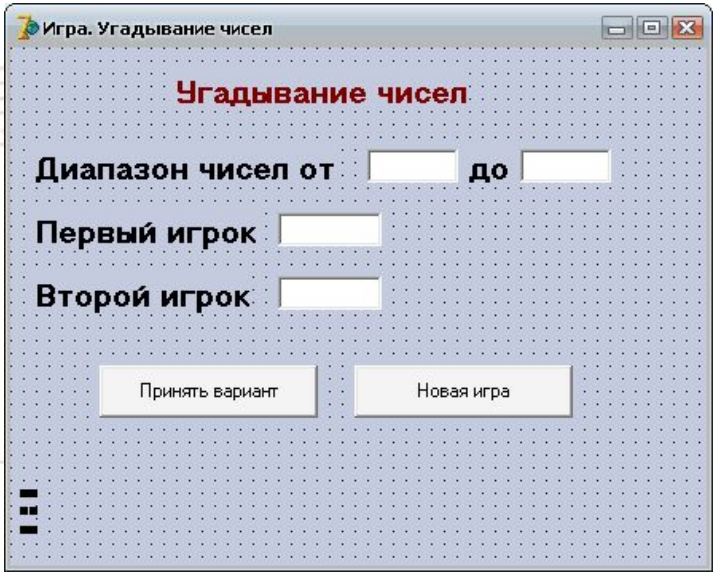
Например, если требуется, чтобы через 5 секунд после запуска приложения закрылась форма — заставка, отображающая логотип приложения, на ней надо разместить таймер, задать в нем интервал *Interval = 5000*, а в обработчик события *OnTimer* вставить оператор *Close*, закрывающий окно формы.

?1 ?2 ?3

Практическая работа №12_1

Постановка задачи:

Разработать игру угадывания чисел из заданного диапазона. Игроки вводят диапазон чисел, компьютер случайным образом определяет число. Игроки вводят предполагаемые числа. Компьютер сравнивает с загаданным и считает количество попыток.



Кнопка «Новая игра» принимает диапазон числа и инициализирует выбирает случайное число из датчика случайных чисел randomize.

```
procedure TForm1.Button1Click(Sender:
TObject);
begin
d1:=strToInt(Edit1.Text);
d2:=strToInt(Edit2.Text);
Edit3.Text:='';
Edit4.Text:='';
Label6.Caption:='';
p:=1;
zagchoslo:=random(d2-d1)+d1;
end;
```

Датчик случайных чисел инициализируется при запуске программы, т.е. в процедуре *TForm1.FormCreate*

```
procedure TForm1.FormCreate(Sender:
TObject);
begin
 p:=1;
 randomize;
end;
```

До всех процедур необходимо описать глобальную переменную (p) для подсчета количества попыток.

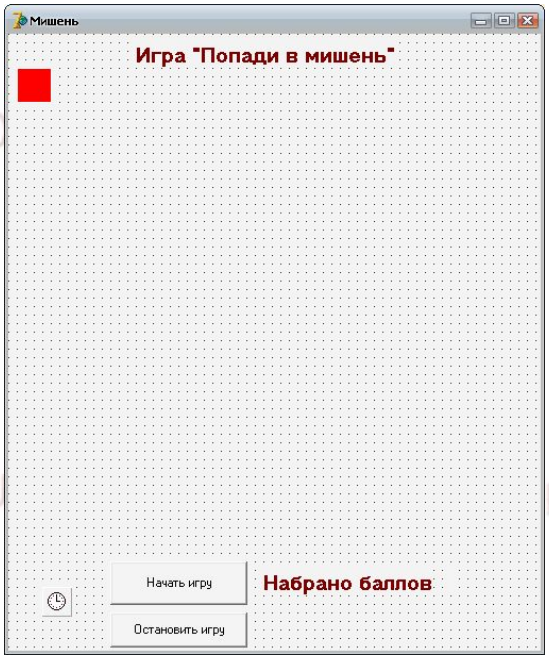
Основная процедура «Принять вариант» проверяет введенные игроками числа и тем, которое загадано и выводит соответствующее сообщение.

```
procedure TForm1.Button2Click(Sender: TObject);
begin
if StrToInt(Edit3.Text)= zagchoslo
   then Label6.Caption:='Первый игрок угадал используя '+IntToStr(p)+' попыток '
   else
    if StrToInt(Edit4.Text)= zagchoslo
    then Label6.Caption:='Второй игрок угадал используя '+IntToStr(p)+' попыток '
    else
      begin
       p:=p+1;
         lABEL6.Caption:='СДЕЛАЙТЕ ЕЩЕ ПОПЫТКУ';
     end;
end;
```

Практическая работа №12_2

Постановка задачи:

Разработать игру «Попади в мишень», которая выводит на экран графический объект. Этот объект движется по экрану, необходимо игроку попадать мышкой на объект. Программа считает количество попаданий.



Для реализации проекта на форме расположите необходимые объекты: *Label1* (заголовок), *Label*2 (для вывода числа попаданий), *Button*1 (начало игры), *Button*2 (остановка игры), *Image*1 (движущийся объект). Все объекты оформляются по усмотрению разработчика.

1. Необходимо описать глобальные переменные для определения координат движущегося объекта и подсчета баллов (попаданий).

```
var
  Form1: TForm1;
  Ball,x,y:integer;
implementation
```

2. По запуску программы необходимо инициализировать датчик случайных чисел randomize

```
procedure TForm1.FormCreate(Sender:
TObject);
begin
  randomize;
end;
```

3. Каждый такт установленного времени должно меняться положение движущегося объекта. Для этого запишем процедуру для компоненты Timer1. В этой процедуре определяются новые координаты движущегося объекта из датчика случайных чисел.

```
procedure TForm1.Timer1Timer(Sender: T
begin
x:=random(400);
y:=random(400);
Image1.Left:=x;
Image1.Top:=y;
end;
end;
```

4. Если игрок сделал клик на объекте, то количество баллов наращивается. Это событие для компоненты Image.

```
procedure TForm1.Image1Click(Sender: TObject);
begin
  Ball:=ball+1;
  Label2.Caption:=IntToStr(Ball);
end;
```

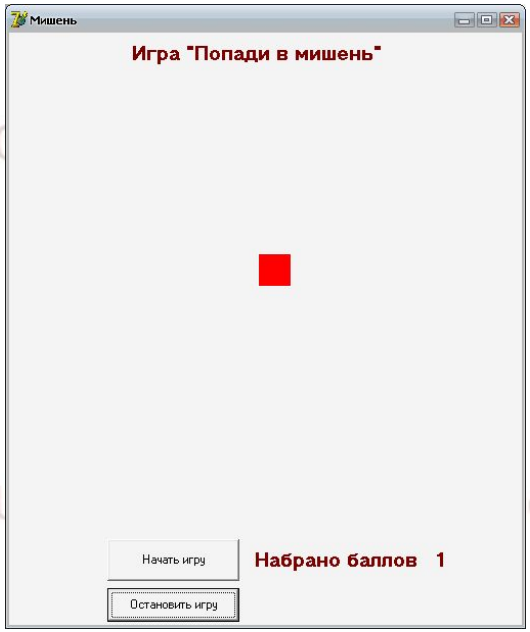
5. Событие для кнопки «Начать игру» - обнуление баллов, запуск таймера

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Ball:=0;
  Timer1.Enabled:=True;
end;
```

6. Событие для кнопки «Конец игры» - отображение результатов игры и остановка таймера.

```
procedure TForm1.Button2Click(Sender: TObject);
begin
Label2.Caption:=IntToStr(Ball);
Timer1.Enabled:=False;
end;
```

Дополнительное задание к игре «Попади в мишень».



Определить выбор уровня игры, для первого уровня оставить задачу в прежнем виде, для второго уровня можно игру усложнить тем, что если игрок попал на объект, то такт времени уменьшается (Timer1.Interval:= Timer1.Interval-100).

В этом случае измениться программный код компоненты Image:

```
procedure TForm1.Image1Click(Sender: TObject);
begin
 Ball:=ball+1;
 if RadioButton2.Checked=True then Timer1.Interval:= Timer1.Interval-100;
 Label2.Caption:=IntToStr(Ball);
end;
```

Практическая работа №12_3

Постановка задачи:

Разработать игру «Кот и мышь». На форме случайно располагаются Кот и Мышь. Они двигаются по форме в любом направлении. Если Кот определяет, что Мышь рядом, то он начинает двигаться за ней.

Игра считается законченной, если:

- Кот догнал Мышь;
- Мышь спряталась в одну из двух
- норок;
- нажата кнопка «Закончить игру».