ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

УПРАВЛЕНИЕ ДИСТАНЦИОННОГО ОБУЧЕНИЯ И ПОВЫШЕНИЯ КВАЛИФИКАЦИИ

Академия Строительства и архитектуры
Кафедра «Информационные системы в строительстве»

Введение в программирование на Delphi

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к проведению практических занятий
по дисциплине

«История отрасли и введение в специальность»

Автор
Позднышева Е.Е.

Ростов-на-Дону, 2016

Аннотация

Методические указания предназначены для студентов направлений подготовки бакалавров 09.03.01 «Информатика и вычислительная техника», 09.03.02 «Информационные системы и технологии» очной формы обучения.

Методические указания «Введение в программирование на Delphi» содержат необходимый материал для выполнения практических работ по дисциплине «История отрасли и введение в специальность». Изложен теоретический и практический материал по возможностям визуальной среды объектно-ориентированного программирования Delphi. Приведены вводные упражнения для освоения инструментальных возможностей среды, создания и сохранения проекта, использования основных элементов палитры компонентов.

Автор

Ст. преподаватель кафедры «Информационные системы в строительстве»
Позднышева Е.Е.

Оглавление

КРАТКАЯ ТЕОРИЯ

Delphi – это среда быстрой разработки программных приложений, в которой в качестве языка программирования используется Object Pascal. В основе системы быстрой разработки программ (RAD – система, Rapid Application Development – среда быстрой разработки приложений) лежит технология визуального проектирования и событийного программирования, суть которого заключается в том, что среда разработки берет на себя большую часть генерации кода программы, оставляя программисту работу по конструированию диалоговых окон и функций обработки событий, написанных на языке Object Pascal, являющимися расширением языка программирования Паскаль.

Для освоения технологии создания и отладки программных приложений студенты должны изучить следующие минимально необходимые теоретические материалы:

1. **Среда Delphi:**

- структура окна Delphi при запуске системы, а именно: главное окно, окно формы, окно инспектора объектов, окно просмотра списка объектов, окно редактора кода, и назначение всех окон;
- основные компоненты страницы Standard:
  - Label (метка),
  - Edit (строка ввода),
  - MEMO (многострочный текстовый редактор),
  - Button (командная кнопка),
  - Panel (контейнер для размещения других компонентов).
- со страницы Additional: в первую очередь изучить:
  - компонент BitBtn (командная кнопка с надписью и пиктограммой)
  - и компонент StaticText (метка с эффектами выделения поля метки).

Перечисленные компоненты необходимо использовать при создании интерфейсного окна пользователя в окне формы для создания простейших программных приложений.

2. **Язык Object Pascal.**

Необходимо изучить алфавит языка, простейшие конструкции языка: простые типы данных и допустимые операции над ни-

ми, а также применимые к ним стандартные функции.

Основные операторы языка:

- присваивание,
- составной,
- условный,
- выбора,
- цикла.

Структура проекта

Создаваемое в Delphi приложение состоит из нескольких элементов, объединенных в проект, которые называются файлами проекта. Разработка проекта строится на тесном взаимодействии процесса создания формы (в окне формы) и процесса написания кода (т.е. текста программы на языке Object Pascal) в окне кода. Между окнами существует неразрывная связь, которую автоматически отслеживает Delphi. При этом по компонентам, размещенным на форме программистом, автоматически создаются строки кода на языке Object Pascal.

Любые изменения на форме, а именно добавления, корректировка компонента, приводит к автоматическому изменению кода программы. Поэтому сначала строят форму, размещая на ней нужные компоненты, а потом переходят к написанию кода. В любом случае, когда проектируется новая форма, в окне кода создается модуль, структура которого приведена ниже.

Первоначально этот модуль **Unit1** формы **Form1**.

Unit <имя модуля>
Interface
{Содержит списки подключаемых модулей, как стандартных модулей Delphi, так и модулей, созданных пользователем.}
Implementation
{Содержит списки подключаемых модулей, объявления типов, констант, переменных, доступ к которым из других модулей закрыт. Здесь же помещаются тексты кодов всех объявленных в разделе Interface функций и процедур.}
Initialization
{Содержит операторы, выполняемые один раз при обращении к модулю.}
Finalization
{Содержит операторы, которые выполняются при окончании выполнения модуля.}

Реализация проекта в компьютере состоит из следующих

этапов: компиляция, компоновка, исполнение. На каждом из этих этапов создаются файлы, имеющие определенное расширение.

При компиляции каждого проекта Delphi создает файлы со следующими расширениями:

***.PAS** – содержит копию текста программы из окна кода программы;

***.DFM** – имеет описание содержимого окна формы;

***.DCU** – содержит результат преобразования в машинные инструкции текста из обоих файлов: ***.PAS** и ***.DFM**.

Файлы *.**DCU** создаются компилятором и далее обрабатываются компоновщиком, который преобразует его в единый загружаемый файл с расширением ***.EXE**.

Файл всего проекта имеет расширение ***.DPR.** Файл проекта является центральным файлом или главным файлом проекта.

В приведенных идентификаторах файлов проекта символ "*" – это место, где записывается имя файла.

В Delphi имена могут состоять из латинских букв, цифр и знака подчеркивания. Никакие другие символы и буквы национальных алфавитов в именах не допускаются. Длина имени, т.е. количество символов, из которых оно состоит, не ограничена.

Главный файл проекта представляет собой программу, которая создается самой Delphi и имеет следующий вид:

```
Program Project1;
Uses
  Forms,
  Unit in 'Unit.pas' (Form1);
  {$R*.Res}
Begin
  Application. Initialize;
  Application. CreateForm(TForm1, Form1);
  Application. Run;
End.
```

Имя проекта (имеет расширение ***.DPR**) и имя исполняемого файла (имеет расширение ***.EXE**) совпадают. По умолчанию система дает имя **"Project1"**. В главном файле проекта в предложении **Uses** указывается имя подключаемого модуля Forms – это стандартный библиотечный модуль Deplhi, который является обязательным для всех предложений, имеющих в своем составе формы, и перечисляются подключаемые модули всех форм про-

екта (если их несколько).

Главный файл проекта содержит всего три инструкции, выполняющие инициализацию приложения, создание формы Form1 и запуск приложения.

Несмотря на то, что файл проекта формируется в Delphi автоматически на экране он не выводится, его можно посмотреть в окне, для чего надо выполнить команду из пункта меню главного окна формы Delphi, а именно **Project/View Source** (**Проект/Просмотр источника**). Не рекомендуется вносить изменения в файл проекта.

В модуле в разделе реализации Delphi записывает директиву компилятора **{$R*.DFM}**. Она предназначена для связывания модуля с описанием соответствующей ему формы. Описание формы и всех размещенных на ней компонентов система Delphi хранит в файлах с расширением ***.DFM**.

Так как проект состоит из нескольких файлов, рекомендуется создавать отдельный каталог для каждого проекта.

Создание формы приложений

Разработка приложений в Deplhi начинается с создания формы. Непосредственно после вставки в проект новой формы Delphi предлагает разработчику «пустую» форму. Но пустой ее можно назвать условно, т.к. она содержит основные элементы окна Windows: заголовок **Form1**, кнопки минимализации, максимализации и закрытия окна и кнопку вызова системного меню, поэтому изначально для любого разрабатываемого приложения предлагается окно формы, для которой уже созданы два файла с описанием и модулем, т.е. файлы ***.DFM** и ***.PAS** (окно кода).

На этапе проектирования форм программа как бы составляется из готовых компонентов, которые можно добавлять к ней с помощью нескольких щелчков мыши.

Компоненты обладают наборами свойств, характеризующими их отличительные особенности. Свойства компонент в процессе проектирования формы устанавливаются с помощью **Инспектора Объектов**.

Создание обработчика событий

Командные кнопки на форме предназначены для активизации программ решения определенных задач. Для того, чтобы получить в окне формы результат выполнения программы при активизации командной кнопки в результате щелчка левой клавиши мыши (ЛКМ) необходимо написать соответствующий обработчик событий и включить его в мо- дуль, созданный в окне кода, в

разделе **implementation**.

Обработчик событий представляет собой подпрограмму, которая выполняет определенные действия, по решению конкретной задачи и активизируется при щелчке на командной кнопке, расположенной на форме. Эти программы программист пишет на языке **Object Pascal**, представляющий собой либо **Procedure** (Процедура), либо **Function** (Функция). Обработчики событий, оформленные в виде процедуры и функции, вставляются в модуль программного приложения в окне кода в разделе **implementation** после директивы компилятора **{SR*.DFM}**.

Для большинства компонент **Delphi** имеются события, обработчики которых можно подготовить, используя страницу **Events** Инспектора объектов. На ней указаны все события, на которые может реагировать выбранный объект. Каждому событию присвоено имя. Например, щелчок кнопки мыши – это событие **OnClick**, которое для командной кнопки является событием по умолчанию.

Надпись на командной кнопке по умолчанию – **Button**, можно заменить в процессе создания формы конкретного приложения. Целесообразно заменять надписи на кнопке текстом, соответствующим назначению обработчика событий. Например, если надо вычислить сумму чисел, введенных в поля **Edit** и за кнопкой **Button** закрепить действия по сложению чисел, то подходящей надписью на кнопке будет название **«Сложить».**

Для создания обработчика событий необходимо, чтобы Delphi создала шаблон в окне редактора кода. Получить окно кода с шаблоном обработчика событий можно, выполнив щелчок ЛКМ на командной кнопке в окне формы, или в инспекторе объектов активизировать страницу **Event** , выбрать строку с именем события **OnClick** и в правой части строки выполнить двойной щелчок ЛКМ. Шаблон обработчика событий представлен ниже.

```
Procedure TForm1.Button1Click (
Sender: TObject) :
begin

end;
```

В шаблон обработчика событий программист вставляет строки кода на языке Object Pascal, реализующего алгоритм решения задачи обработки данных.

ПРАКТИЧЕСКАЯ РАБОТА №1

Создание простейшего Windows-приложения

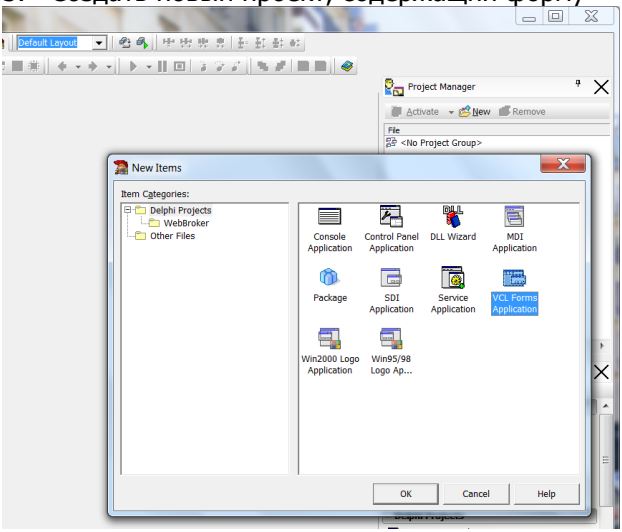
Цели работы:

- Создание простейшего Windows-приложения с заданным заголовком окна и цветом формы
- Создание Windows-приложения, которое содержит текст "Моя первая программа!" и кнопки, позволяющие изменять размер шрифта и двигать текст

Состав задания:

Задание 1

1. Создать папку для сохранения разработанных приложений
2. Запустить Delphi
3. Создать новый проект, содержащий форму



4. Изменить заголовок окна формы с Form1 на Привет: в окне инспектора объектов (Object Inspector) установить для свойства Caption значение **Привет**

Окно инспектора объектов – это окно создаёт и редактирует свойства компонентов, а также является обработчиком событий компонентов. Инспектор объектов – инструмент, который используется для формирования внешнего вида и функциональных возможностей.

Окно инспектора объектов содержит две страницы:

1) свойства Properties
2) события Events

В левой части перечислены все имеющиеся обработчики событий компонентов. События определяют поведение компонента, т.е. будет ли компонент реагировать на щелчок мыши или на нажатие клавиши.

В правых колонках обеих страниц указывается значение свойств или обработчики событий соответствующих компонентов.

Некоторые свойства компонент, имеющие частое применение:

- **Align** – задает тип выравнивания компонента внутри формы (по умолчанию равно **alNone**);
- **Aligment** - определяет выравнивание текста в компоненте. Оно принимает следующие значения: **TaLeftJustify** – выравнивание по левому краю, **TaRightJustify** – выравнивание по правому краю. По умолчанию выравнивание происходит по левому краю,**TaCenter**-выравнивание текста по центру;
- **AutoSize** – при значении свойств **True** будет автоматически изменяться ширина и высота метки в соответствии с размещенным в ней текстом;
- **Caption** – заголовок компонента (надпись на компоненте). Могут использоваться русские буквы;
- **Color** – цвет фона для формы или компонента. Цвет можно задавать при помощи обозначений, например, зеленый – clGreen, или шестнадцатеричных констант (зеленый - $008000);
- **Enable** – если это свойство равно **True**, то компонент реагирует на действия пользователя (сообщения до щелчка мыши и нажатия клавиатуры), иначе эти сообщения игнорируются;
- **Height** – вертикальный размер в пикселях;
- **Hint** – задает текст (подсказку), который будет отображаться при нахождении курсора в области компонента;
- **Left** – горизонтальная координата левого верхнего угла компонента относительно формы в пикселях. Для формы это значение указывается относительно экрана дисплея;
- **Name** – задает имя компонента (идентификатор), которое будет использоваться в программе;

- **ParentColor** – если значение этого свойства равно True, то компонент будет отображаться цветом родительского компонента, иначе используется собственное свойство **Color**;
- **TabOrder** – задает порядок получения компонентами на форме фокуса нажатия клавиши Tab. По умолчанию этот порядок определяется последовательностью размещения компонентов на форме. Для изменения этого порядка необходимо явно изменить значения свойства TabOrder компонентов. Следует отметить, что компонент, значение **TabOrder** которого равно 0, получает фокус при отображении формы. Использование свойства TabOrder зависит от значения свойства **TabStop**;
- **TabStop** – это свойство позволяет указать, может ли компонент получать фокус или нет;
- **Text** – текст, находящийся в поле ввода и редактирования;
- **Top** – вертикальная координата левого верхнего угла интерфейсного элемента относительно формы;
- **Transparent** – при значении свойства **True** фон метки будет прозрачным по отношению к другим компонентам;
- **Visible** – определяет видимость компонента;
- **Width** – горизонтальный размер интерфейсного элемента или формы в пикселях;
- **WordWrap** – при значении **True** после заполнения текущей строки происходит перенос текста на новую строку. По умолчанию имеет значение **False**.

Помимо свойств компоненты содержат **методы** – программный код, обрабатывающий значение свойств (например, устанавливающий переключатель в нужное положение), а также **события** – сообщения, которые компонент принимает от приложения, если во время работы программы выполняется определенное действие (например, щелчок мыши на компоненте). Программист может самостоятельно формировать реакции программ на любые события каждого компонента.

5. Изменить цвет формы со стандартного на другой: в окне инспектора объектов установить для свойства Color значение clAqua.
6. Выполнить приложение:

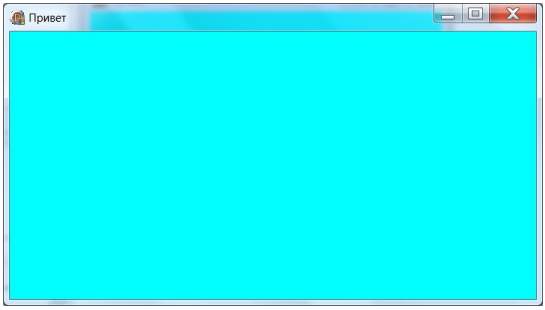
6.1. Запустить приложение - меню Run-Run или F9 или кнопка на панели инструментов.

6.2. Изменить размеры окна.

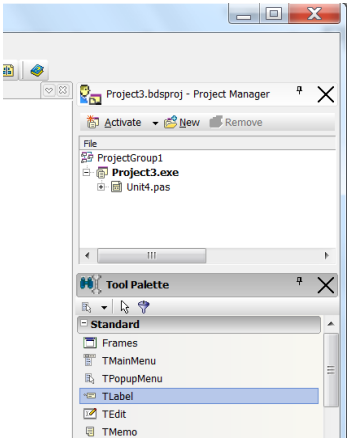
6.3. Поэкспериментировать со стандартными кнопками минимизации и максимизации окна.

7. Сохранить форму и проект на диске:

(Меню File, Save All, установить свою папку, создать новую папку (с именем Лабораторная работа №1), открыть ее, ввести имя проекта).



8. Поместить объект TLabel в окно формы «Привет»:



9. Переместить объект Label1 на желаемое место в форме.

10. Изменить свойства объекта Label1:

В окне инспектора объектов (Object Inspector) установить следующие значения для свойств объекта:

Объект	**Свойство**	**Значение**
Label1	Caption	Моя первая программа!
	Font	12 р., красный
	Alignment	TaCenter

	Color	Желтый (Yellow)
	AutoSize	False

Пояснения

При проектировании формы в Delphi обычно приходится писать код для отклика на некоторые из ее событий. Когда вы нажимаете кнопку мыши в форме или на компоненте, Windows посылает вашему приложению сообщение, информируя его об этом событии. Реакция Delphi состоит в получении сообщения о событии и вызове соответствующего метода отклика на событие. Для каждого вида компонентов Delphi определяет ряд различных событий. Вы можете узнать о событиях, доступных для формы или компонента, рассматривая страницу Events окна Object Inspector в тот момент, когда выбран необходимый элемент.

Вставим в форму компонент – кнопку – и заставим его откликаться на событие, связанное с нажатием левой кнопки мыши. При щелчке по кнопке мышью в работающей программе возникает событие OnClick (По щелчку). Пока это событие никак не обрабатывается программой, и поэтому нажатие кнопки не приведет ни к каким последствиям. Чтобы заставить программу реагировать на нажатие кнопки, необходимо написать на языке Object Pascal фрагмент программы, который называется обработчиком события. Фрагмент оформляется в виде специальной подпрограммы – процедуры.

Чтобы заставить Delphi самостоятельно сделать заготовку для процедуры обработчика события OnClick, дважды щелкните мышью по вновь вставленному компонету. В ответ Delphi активизирует окно кода и вы увидите в нем такой текстовый фрагмент:

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

Слово **procedure** извещает компилятор о начале подпрограммы – процедуры. За ним следует имя процедуры TForm1.Button1Click. Это имя – составное: оно состоит из имени класса TForm1 и собственно имени процедуры Button1Click.

Каждый компонент принадлежит к строго определенному классу, а все конкретные экземпляры компонентов, вставляемые в форму, получают имя класса с добавленным числовым индексом.

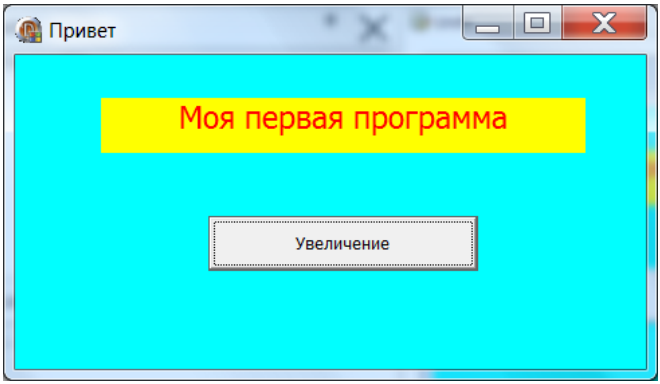
По используемому в Delphi соглашению все имена классов начинаются с буквы T.

Таким образом, имя TForm1 означает имя класса, созданного по образцу стандартного класса Tform.

11. Поместить объект Button (командная кнопка) в окно «Привет». Он по умолчанию получит имя Button1. Изменить его размеры.
12. Установить свойство Caption объекта Button1 в значение "Увеличение".
13. Написать код для события Click на объекте Button1:
    13.1. Два раза щелкнуть по объекту Button1 в форме.
    13.2. Между словами Begin и End написать следующий код:

```
Label1.Font.Size := Label1.Font.Size +2;
```

14. Выполнить программу. Обратить внимание на то, что происходит при нажатии кнопки с надписью "Увеличение".



15. Создать объект "командная кнопка" для уменьшения размера шрифта в тексте.
16. Создать объект "командная кнопка" для того, чтобы двигать текст.

Код:

```
Label1.Left := Label1.Left + 10;
Label1.Top := Label1.Top + 10;
```

17. Создать объект "командная кнопка" для того, чтобы сделать текст невидимым.

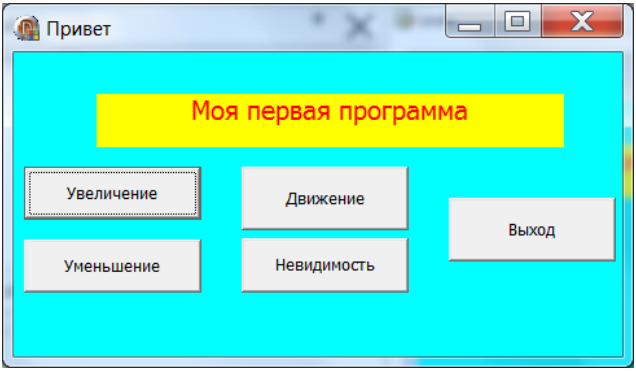
Код:

```
Label1.visible := false;
```

18. Создать объект "командная кнопка" для выхода из работы программы.

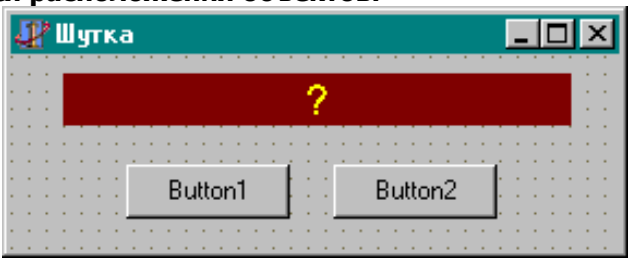
Код:

```
Close;
```



19. Сохранить проект в папке с именем «Практическая работа 1-1»: Меню File, Save All.

Задание 2 **Создание Windows-приложения, в котором изменяется расположения объектов.**



1. Создать новый проект и форму.
2. Поместить компоненты Label и Button в форму в соответствии с рисунком
3. Установить следующие свойства объектов:

Объект	**Свойство**	**Значение**
Form1	Caption	Шутка
Label1	Caption	?
Label1	Color	clMaroon
Label1	Font.Size	18
Label1	Font.Color	Желтый
Label1	Alignment	taCenter

4. Установить свойство объекта Button2: **DragMode dmAutomatic**
5. Записать код для обработки события MouseMove на объекте Button2:

```
procedure TForm1.Button2MouseMove(Sender: TObject; Shift: TShiftState;
X,Y: Integer);
begin
Button2.Left := Button2.Left+10;
Button2.Top := Button2.Top+10;
end;
```

6. Записать код для обработки события Click на объекте Button1:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
Label1.Caption := 'Мы были в этом уверены!'
end;
```

7. Выполнить программу.
8. Сохранить проект в папке с именем «Практическая работа 1-2».
9. Предъявить преподавателю результаты работы.

Вопросы

1. Какие окна присутствуют по умолчании на экране в момент начала работы над новым проектом в Delphi и каковы их функции?
2. Что такое Properties и Events в окне инспектора объектов?
3. В чем разница между свойствами Caption и Name?
4. Что означают значок «+» перед названием свойства в окне инспектора объектов и кнопка с многоточием в строке свойства?
5. Какие файлы создает Delphi при работе с проектом? Каково их назначение? Где они сохраняются?
6. Какой алгоритмический язык используется для программирования в Delphi?
7. Каким образом в Delphi создается стандартная заготовка для обработчиков событий?

ПРАКТИЧЕСКАЯ РАБОТА №2

Простейшая математическая программа

Цели работы:

- практическое освоение методологии и принципов создания базовых стандартных элементов интерфейса Windows-программы в среде визуального проектирования.
- практическое освоение методологии и принципов создания и оформления элементов интерфейса Windows-программы.

Пояснения

В представленном ниже проекте используем следующий минимальный набор компонент.

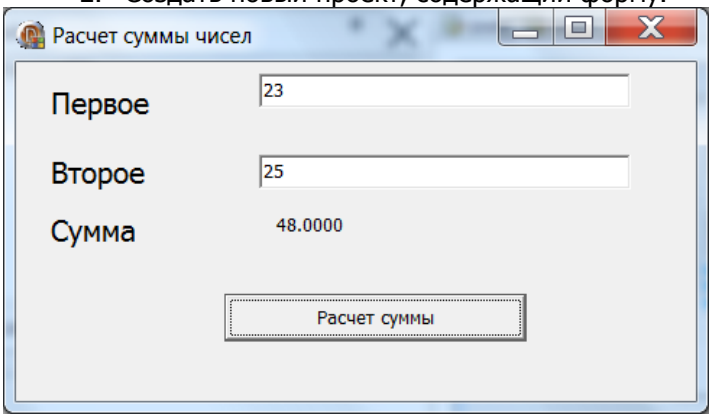
Button – стандартная кнопка, обычно кнопка используется для запуска действия, при этом задействуют только метод OnEvent (реакция на нажатие). Свойство Default=True ассоциирует вводимый компонент с кнопкой Enter, Cancel=True – с кнопкой Esc. Свойства Color для оформления надписи (Caption) у кнопки нет. Амперсант, помещенный в тексте надписи, указывает быструю Alt-клавишу запуска, например, Caption=A&Ppend вызывает срабатывание кнопки при нажатии Alt-P. Свойство ModalResult=true определит обязательность нажатия для закрытия дочернего окна.

Label – метка, используется как надпись или как область вывода информации для чтения. Как и для кнопки, для метки можно определить клавишу быстрого доступа, но она будет запускать связанный с меткой компонент (по FocusControl). Свойство AutoSize=True определит минимизацию размера метки под текст надписи, Aligment – центровку этого текста, WordWrap – возможность расположения текста в несколько строк, Transparent – прозрачность при наложении на другие элементы.

Edit – строка ввода. Заголовка (Caption) у этого компонента нет, но есть свойство Text как содержимое строки. Это свойство можно как считывать, так и присваивать (при необходимости с ограничением длины назначением свойства MaxLength). При вводе конфиденциальной информации указывают отображаемые символы (обычно "*"), при этом нужно переопределить свойство PasswordChar, задав его отличным от #0.

Состав задания

1. Запустить Delphi.
2. Создать новый проект, содержащий форму.



3. Составить проект для суммирования двух чисел, вводимых с клавиатуры.

При этом на форме нужно разместить три надписи (с задаваемыми свойствами Caption) и четвертую надпись с пустой Caption – для отображения суммы. Определить две строки ввода для суммируемых чисел (против меток "первое" и "второе") и одну кнопку "Расчет" для запуска процедуры суммирования после ввода чисел.

После двойного щелчка на кнопке можно заполнить шаблон процедуры реакции на нажатие этой кнопки.

```
procedure TForm11.Button1Click(Sender: TObject);
var a,b,c: real;
    s: string;
    code: integer;
begin
  {ввод данных из полей редактирования}
  val(edit1.text,a,code);
  val(edit2.text,b,code); c:=a+b;
  str(c: 10:4,s); {перевод числа в строку}
  label4.Caption:=s
end;
```

Функция	**Описание**
Concat(sl, s2, s3)	Возвращает последовательное соединение строк. Эквивалентна оператору sl+s2+s3.
Copy(s, pos, len)	Возвращает подстроку длиной len символов, начинающуюся в позиции pos строки s.
Delete(s, pos, len)	Удаляет максимум len символов из строки s, начиная с позиции pos.
Insert(sourse, target, pos)	Вставляет строку source в строковую переменную target, начиная с позиции pos.
Length (s)	Возвращает динамическую длину строки. Подобна функциям LEN в Basic и strlen – в C/C++.
Pos(substring, s)	Возвращает место первого вхождения подстроки substring в строку s. Подобна функциям SUBSTR в Basic и strstr () – в C/C++.
SetLength(s, newlen)	Задает новую динамическую длину newlen строковой переменной s.
SetString	Задает содержимое и длину строки.
Str(x, s)	Преобразует численное значение x в строковую переменную s.
StringOfChars	Возвращает строку с конкретным числом символов.
UniqueString	Делает данную строку уникальной со счетом обращений 1.
Val (s, v, code)	Преобразует строку s в соответствующее численное представление v.

Приведенный выше вариант программы вполне работоспособен. Но в подобных программах обязательное требование в части их оформления – предусмотреть реакции на ввод символов в полях редактирования, например, защиту от ввода букв или второй десятичной точки.

При нажатии Enter естественно переносить курсор в следующее поле редактирования или выполнять другие действия, если ввод данных завершен.

4. Добавить реакции на ввод символов в поля редактирования.

В обработчиках событий (закладка Events инспектора собы-

тий Delphi) для полей ввода определим методы OnKeyPress, задав им имена, например, E1 и E2. Затем после двойного щелчка заполним шаблоны процедур.

```
procedure TForm11.E1(Sender: TObject; var Key: Char);
begin
 {защита поля редактирования на ввод числа }
 case key of
  '0'..'9',chr(8):;
    '.': if pos('.',edit1.text)>0 then key:=chr(0);
    '-': if length( edit1.text)>0 then key:=chr(0);
  chr(13): edit2.SetFocus;
  else key:=chr(0);
end;
 end;
```

Вторая процедура отличается от первой лишь реакцией на нажатие клавиши Enter

```
procedure TForm1.E2(Sender: TObject; var Key: Char);
begin
... edit2.text ...
chr(13): edit2.font.color:=clRed; ...
end;
```

5. Добавить в форму новые кнопки.

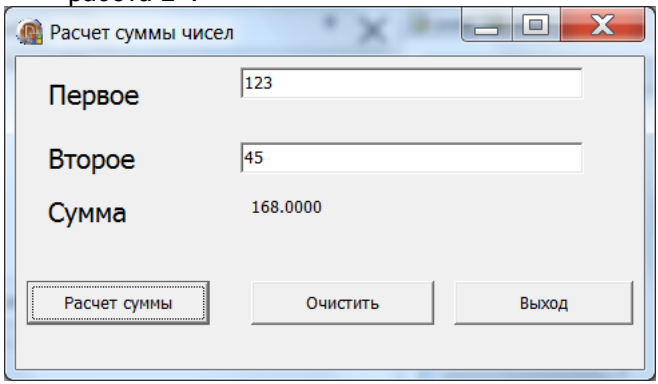
Введем кнопку очистки полей ввода и вывода результата для нового расчета. Заголовок кнопки определим как Caption="Очистить", зададим реакцию OnClick (двойным щелчком на кнопке).

```
procedure TForm11.Button2Click(Sender: TObject);
begin
 {очистка полей ввода}
    edit1.text:='';
    edit2.text:='';
    label4.caption:='';
    edit1.SetFocus
end;
```

Введем кнопку выхода

```
procedure TForm11.Button3Click(Sender: TObject);
begin
  form11.close   { завершение приложения}
end;
```

6. Сохранить проект в папке с именем «Практическая работа 2».



7. Предъявить преподавателю результаты работы

ПРАКТИЧЕСКАЯ РАБОТА №3

Компоненты выбора и настройки параметров

Цели работы: Создание Windows-приложения, в котором:

- при щелчке на радио-кнопке с названием цвета на светофоре загорается соответствующий цвет
- работают цифровые часы с разной скоростью

Пояснения

Выбор и настройка параметров при работе с программным приложением считается стандартной частью работы пользователя с любым серьезным приложением. Это может быть, как настройка самого приложения, так и определение параметров, отображаемых или моделируемых в приложении процессов и явлений. Элементы интерфейса Windows-программы для основных операций такой работы в настоящее время стандартизированы. Рассмотрим создание этих элементов на примере работы с компонентами библиотеки VCL (Visual Component Library) в среде Delphi.

Базовые элементы выбора и настройки параметров расположены на странице Standart палитры компонент Delphi. В представленном ниже проекте используем следующий классический набор компонент:

GroupBox – группа, которая визуально и логически объединяет наборы компонент, определяет порядок перемещения по компонентам на форме (при нажатии клавиши TAB). При помещении в группу новый компонент получает свойства ParentColor, ParentShowHint, ParentFont, ParentCtl3D этой группы. Свойства Left и Top сгруппированных объектов определяются по верхнему углу группы, а не формы;

RadioGroup – группа для объектов RadioButton (см. ниже);

RadioButton – переключатели или радиокнопки, служат для выбора одной возможности из набора взаимоисключающих возможностей. Термин отражает сходство с набором кнопок выбора каналов радиоприемника. Эти кнопки обычно объединяют группой RadioGroup. Выбор кнопки отражает свойство Checked, свойство Alingment определяет положение поясняющей надписи относительно кнопки;

CheckBox – выключатель, выглядит как строка текста с окошком для установки отметки о выборе. Выключатели работают независимо, но их обычно группируют. При определении реакции на выбор можно использовать событие

OnClick, но обычно устанавливают как индикатор свойство State по трем состояниям – cbChecked (есть), cbUnChecked (нет), cbGrayed (неопределенно) внутри программы. При этом для блокировки ручного изменения этого свойства нужно установить DragMode=Automatic.

Состав задания

Задание 1. Пример проекта с выбором параметров «Светофор»

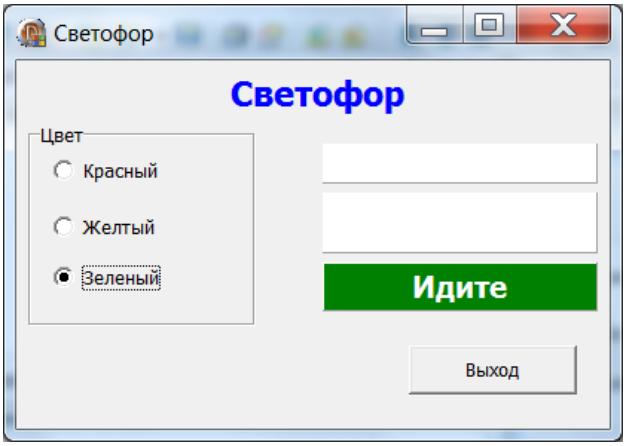
1. В новом проекте создать форму.
2. Поместить компоненты Label, Panel, GroupBox, RadioButton (страница Standard) в форму.
3. Установить следующие свойства объектов, используя Инспектор объектов:

Label1	Caption	Светофор
Panel1,2,3	Caption	
GroupBox1	Caption	Цвет
RadioButton1	Caption	Красный
RadioButton2	Caption	Желтый
RadioButton3	Caption	Зеленый

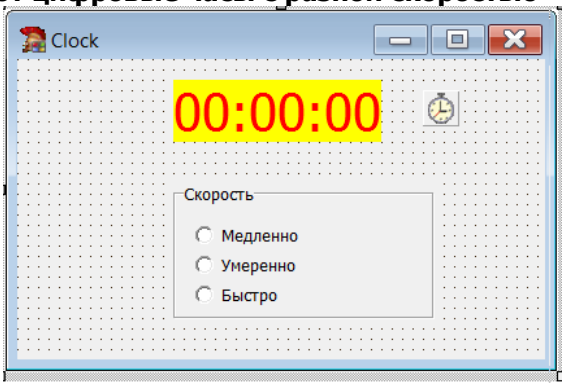
4. Записать код для процедуры обработки события Click (щелчок мыши) на объекте RadioButton1:

```
procedure TForm1.RadioButton1Click(Sender: TObject);
begin
        Panel1.Color := clRed;
        Panel2.Color := clWhite;
        Panel3.Color := clWhite;
end;
```

5. Самостоятельно записать код для процедур: TForm1.RadioButton2Click и TForm1.RadioButton3Click
6. Добавить печать информации "Стойте", "Внимание", "Идите" на панели с соответствующим сигналом белым цветом шрифта жирным начертанием 12п.
7. Сохранить проект в папке с именем «Практическая работа 3-1».
8. Предъявить преподавателю результаты работы.



Задание *2.* Создание Windows-приложения, в котором работают цифровые часы с разной скоростью



1. Создать новый проект и форму.
2. Поместить компоненты TLabel (вкладка Standard) и TTimer (System) в созданную форму.
3. Установить следующие свойства объектов

Объект	**Свойство**	**Значение**
Form1	Name	Clock
Label1	Caption	00:00:00 clYellow
Label1	Color	24
Label1	Font.Size	Красный
Label1	Font.Color	

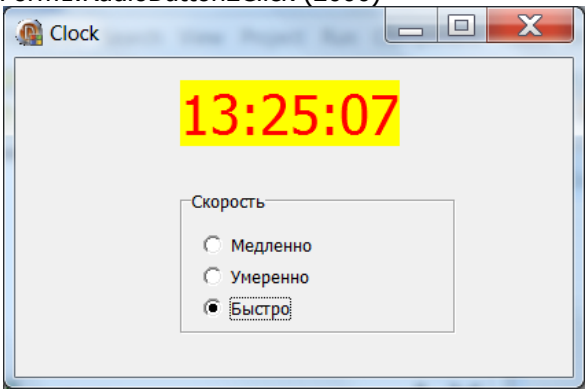
4. Записать код обновления времени для процедуры TClock.Timer1Timer:

```
Label1.Caption:=TimeToStr(Time);
```

5. Добавить в форму компоненты GroupBox и RadioButton:
6. Установить следующие свойства объектов:

GroupBox1	Caption	Скорость
RadioButton1	Caption	Медленно
RadioButton2	Caption	Умеренно
RadioButton3	Caption	Быстро

7. Записать код для процедуры TForm1.RadioButton3Click: Timer1.Interval := 1000;
8. Самостоятельно записать код для процедур: TForm1.RadioButton1Click (3000) и TForm1.RadioButton2Click (2000)



9. Сохранить проект в папке с именем «Практическая работа 3-2».
10. Предъявить преподавателю результаты работы.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Колдаев, В.Д. Структуры и алгоритмы обработки данных: учебное пособие /В.Д. Колдаев. – М.: ИЦ РИОР: Инфра-М, 2014. – 296 с.

2. Эйдлина, Г.М.Delphi: программирование в примерах и задачах: практикум. Учебное пособие /Г.М. Эйдлина, К.А. Милорадов. – М.: ИЦ РИОР: Инфра-М, 2012. – 116 с.

3. Гвоздева, В.А. Введение в специальность программиста. Учебник. /В.А. Гвоздева. – М.: ИД «Форум»: Инфра-М, 2013 – 208 с.

4. Фленов, М.Е. Библия Delphi. Издание 3. / М.Е. Фленов. – СПб.: БХВ-Петербург, 2011. – 686 с.