ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ИНКЛЮЗИВНОГО ВЫСШЕГО ОБРАЗОВАНИЯ

**«МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ
ГУМАНИТАРНО-ЭКОНОМИЧЕСКИЙ УНИВЕРСИТЕТ»**
(ВОЛГОГРАДСКИЙ ФИЛИАЛ)

УТВЕРЖДАЮ
директор Волгоградского
филиала МГГЭУ
_____ Рябишин А.П.
«12» февраля 2019 г.

Практикум
«Основы программирования в Delphi»
по дисциплине
«Основы алгоритмизации и программирования»
для специальности 09.02.05 Прикладная информатика (по отраслям)

Волгоград, 2019

ОДОБРЕН
предметно-цикловой
комиссией
Профессиональных модулей
протокол № 6
от «17» 01 2019 г.

Разработан на основе Федерального государственного образовательного стандарта по специальности среднего профессионального образования
09.02.05 Прикладная информатика (по отраслям),
13.08.2014 г.

Председатель предметно-цикловой комиссии
_____О.В. Ермакова

И.о. заместителя директора по учебно-методической работе
_____ Казакова О.И.

Составитель (автор):

Синельник Татьяна Евгеньевна, преподаватель высшей категории Волгоградского филиала ФГБОУ ИВО МГГЭУ,

рецензенты:

_____

_____

Комитет образования, науки и молодежной политики Волгоградской области
Государственное автономное учреждение дополнительного профессионального образования
«Волгоградская государственная академия последипломного образования»
(ГАУ ДПО «ВГАПО»)

400012, г. Волгоград, ул. Новодвинская, 19А
Тел. 8442-606-613, 606-614, 606-609
E-mail: vgapkro@mail.ru (приемная)
E-mail: timnpo@yandex.ru (кафедра ТиМНПО)

ПОРЯДК. №91

ВЫПИСКА
из протокола №1 от «16» сентября 2019 года заседания Экспертного научно-методического совета профессионального образования Волгоградской области

Обсуждали: Предложение УМО по УГС 09.00.00 об использовании в *учебном процессе профессиональных образовательных организаций* ***практикума «Основы программирования в Delphi» по учебной дисциплине Основы алгоритмизации и программирования*** *разработчика* ***Синельник Т.Е.,*** *Московского государственного гуманитарно-экономического университета (Волгоградский филиал).*

Постановили: ***Практикум «Основы программирования в Delphi» по учебной дисциплине Основы алгоритмизации и программирования*** *разработчика* ***Синельник Т.Е.,*** *Московского государственного гуманитарно-экономического университета (Волгоградский филиал),* ***соответствует*** *установленным в системе СПО ВО требованиям в части комплексного учебно-методического обеспечения и* ***рекомендуется*** *в качестве учебного издания по* ***специальности 09.02.05 Прикладная информатика (по отраслям)*** *для использования в учебном процессе профессиональных образовательных организаций, реализующих программы среднего профессионального образования.*

Председатель ЭНМС _____ **/С.В. Куликова, д.п.н., профессор, ректор ГАУ ДПО «ВГАПО»/**

Рецензия

На практикум

«Основы программирования в Delphi»

по дисциплине

«Основы алгоритмизации и программирования»

для специальности 09.02.05 Прикладная информатика (по отраслям)

Автор: Т.Е. Синельник, преподаватель Волгоградского филиала МГГЭУ

Рабочая программа дисциплины «Основы алгоритмизации и программирования» является частью программы подготовки специалистов среднего звена в соответствии с ФГОС по специальности СПО 09.02.05 Прикладная информатика (по отраслям) и направлена на изучение основных принципов алгоритмизации и программирования на языках высокого уровня.

Практикум «Основы программирования в Delphi» по дисциплине «Основы алгоритмизации и программирования» составлен в соответствии с рабочей программы дисциплины по разделу «Среда Delphi».

Пояснительная записка к практикуму описывает цели и задачи дисциплины «Основы алгоритмизации и программирования». В пояснительной записке указываются требования к студенту при подготовке к выполнению практической работы. Также в пояснительной записке приведен перечень требований к студенту при выполнении практической работы и указаны критерии оценки результатов выполнения практических работ.

Практикум «Основы программирования в Delphi» содержит задания к 15-ти практическим работам.

Каждая практическая работа содержит цели работы, краткие теоретические сведения, задание, описание хода работы. Все указания по выполнению практических работ снабжены рисунками и комментариями.

Вывод: практикум «Основы программирования в Delphi» по дисциплине «Основы алгоритмизации и программирования», составленный преподавателем ВФ МГГЭУ Синельник Татьяной Евгеньевной, имеет достаточный профессиональный уровень. Автор хорошо ориентируется в области изуче-

ния требуемого по стандарту круга вопросов. Предлагаемая последовательность выполнения заданий практикума свидетельствуют о достаточной методической подготовке автора. Практикум «Основы программирования в Delphi» может быть рекомендован к использованию на практических занятиях по дисциплине «Основы алгоритмизации и программирования» для специальности 09.02.05 Прикладная информатика (по отраслям).

Рецензент

Преподаватель ГБПОУ ВЭТК

(занимаемая должность)

МП

(подпись)

Д.И. Вертинов

(инициалы, фамилия)

Оглавление

Пояснительная записка

Дисциплина «Основы алгоритмизации и программирования» входит в вариативную часть общепрофессионального цикла дисциплин программы подготовки специалистов среднего звена в соответствии с ФГОС по специальности СПО 09.02.05 Прикладная информатика (по отраслям).

Содержание дисциплины охватывает круг вопросов, связанных с разработкой и реализацией алгоритмов решения прикладных задач программирования.

Практикум «Основы программирования в Delphi» предназначен для обучения основам программирования на языке программирования Delphi.

Цель дисциплины «Основы алгоритмизации и программирования»: изучение основных принципов алгоритмизации и программирования на языках высокого уровня: основные сведения о характеристиках и технологии программирования, средства описания данных, средства описания действий.

Задачи дисциплины «Основы алгоритмизации и программирования»:

- выработать у студентов целостное представление о принципах построения языков программирования;
- изучить методы и средства программирования;
- дать студентам практические навыки по выбору наиболее оптимального языка программирования и/или среды программирования;
- изучить подходы к оценке сложности программирования;
- создать теоретическую базу для последующих дисциплин, связанных с разработкой программных приложений.

В результате освоения учебной дисциплины должны быть сформированы *умения:*

- ставить задачу и разрабатывать алгоритм ее решения;
- выполнять формализованное описание поставленных задач;
- выполнять отладку и тестирование программ;
- использовать современные методы программирования и возможности языка для решения практических задач;
- выбрать из доступных языков или средств программирования наиболее эффективный и надежный для решения поставленной задачи.

знания:

- основные принципы алгоритмизации;
- основные методы обработки данных;
- принципы объектно-ориентированного программирования;

– технологии разработки программ на языках программирования высокого уровня.

В результате выполнения заданий практикума обучающийся должен освоить процесс создания программных приложений на языке Delphi.

Взаимосвязь учебных дисциплин: для понимания заданий практикума студенту необходимо изучить дисциплину «Информатика».

Практикум состоит из 15 практических работ. Для каждой работы сформулированы цели работы, даны краткие теоретические сведения, подробно описан ход работы, представлены необходимые иллюстрации.

При подготовке к выполнению практической работы студент должен повторить теоретический лекционный материал по указанной теме.

В ходе выполнения практических работ студент должен:

– выполнять требования по охране труда;
– соблюдать инструкцию по правилам и мерам безопасности в лаборатории информационных технологий;
– строго выполнять весь объем работы, указанный в задании;
– соблюдать требования эксплуатации компьютерной техники.

Критерии оценки результатов выполнения практических работ:

– оценка «отлично» ставится за полностью выполненную практическую работу;
– оценка «хорошо» ставится, если созданная программа работает правильно, но не выполнены дополнительные задания к работе;
– оценка «удовлетворительно» ставится, если созданная программа работает некорректно;
– оценка «неудовлетворительно» ставится, если созданная программа не работает или практическая работа не выполнена.

Практическая работа № 1
Знакомство со средой разработки

Цели работы:

- закрепление полученных знаний по теме «Характеристика проекта в Delphi»;
- формирование умения работать в среде разработки;
- формирование умения работать с проектом;
- закрепление навыков создания проекта.

Краткие теоретические сведения

Создаваемое в среде Delphi приложение состоит из нескольких элементов, объединенных в *проект*. В состав проекта входят:

1. Главный файл проекта, изначально называемый PROJECT1.DPR.
2. Первый модуль программы /unit/, который автоматически появляется в начале работы. С именем UNIT1.PAS по умолчанию.
3. Файл главной формы, который по умолчанию называется UNIT1.DFM, используется для сохранения информации о внешнем виде главной формы.
4. Файл PROJECT1.RES содержит иконку для проекта, создается автоматически.
5. Файл, который называется PROJECT1.OPT по умолчанию, является текстовым файлом для сохранения установок, связанных с данным проектом. Например, установленные Вами директивы компилятора сохраняются здесь.
6. Файл PROJECT1.DSK содержит информацию о состоянии рабочего пространства.

Работа над новым проектом начинается с создания *стартовой формы*. Неотъемлемыми атрибутами формы являются её заголовок, пиктограмма и три управляющие кнопки, позволяющие быстро свернуть форму на панель задач, развернуть до исходного размера ли на весь экран, а также закрыть её. Все данные атрибуты располагаются в верхней части формы, которая называется *областью заголовка*. Остальное пространство формы называется *клиентской областью.* Она предназначена для размещения всех необходимых визуальных и невизуальных компонентов Delphi, а также для вывода графики.

Компонент *Label* (метка) расположен на странице Standard и используется для отображения текста.

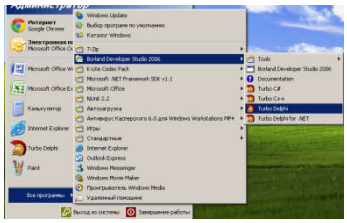
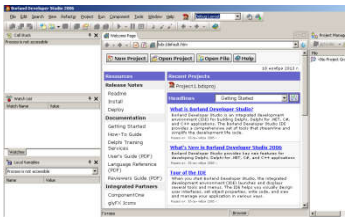
Компонент *Button* (стандартная кнопка) расположен на странице Standard и является оконным элементом управления. Кнопка Button может иметь на поверхности надпись (назначение кнопки или описание действий, выполняемых при её нажатии).

Основным для кнопки является событие *OnClick*, возникающее при её нажатии. При этом кнопка принимает соответствующий вид, подтверждая происходящее действие визуально. Действия, выполняемые в обработчике события OnClick, происходят сразу после отпускания кнопки.

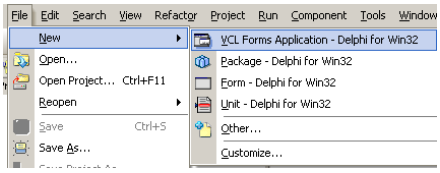
Задание. Создать проект, выполняющий 3 арифметические действия: сложение, вычитание, умножение.

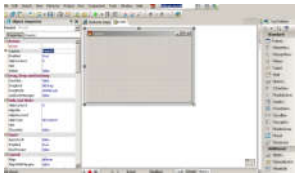
Ход работы:

1. Запустите среду Delphi, откроется стартовое окно:

В меню файл выберите команды: File -> New, откроется вспомогательное меню для выбора типа создаваемого проекта. Выберите первую строчку. Откроется окно для создания нового проекта:



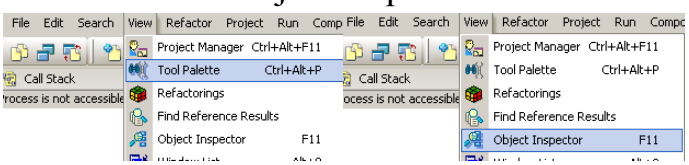


В центре расположено окно формы, справа – палитра компонентов, слева – инспектор объектов.

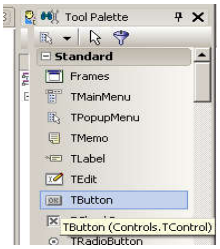
За окном формы находится окно кода программы. Перейти к нему можно, нажав клавишу F12 или выбрать внизу окна вкладку Code. Чтобы вернуться обратно к окну формы, нужно опять нажать клавишу F12 или выбрать внизу окна вкладку Design.

Часть кода программы генерируется автоматически и его удалять нельзя, вы только дополняете свои части программы! Новая программа выглядит так:

Если окна справа и слева не открылись, то нужно в меню View найти и выбрать команды: Tool Palette и Object Inspector:



2. Справа, на палитре компонентов, найдите компоненты TLabel и TButton (буква T означает принадлежность этих компонентов к определенному классу).



3. Расположите (путем перетаскивания) на форме 3 компонента Label, два компонента Button так, как на рис. 1.

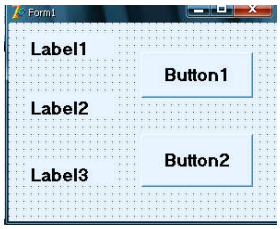
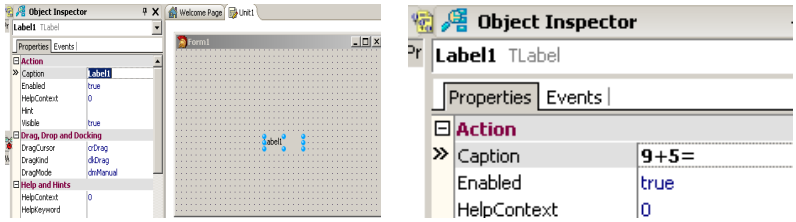


Рис. 1 Начальное расположение компонентов

4. Выделите (т.е. один раз щелкните на компоненте) компонент Label1. Слева, в инспекторе объектов, найдите свойство Caption:



Измените свойство Caption компонента Label1 на «9+5=». Аналогично поочередно выделите остальные компоненты, измените свойство Caption компонента Label2 на «9-5=», свойство Caption компонента Label3 на «9*5=».

5. Измените свойство Caption компонента Button1 на Результат, свойство Caption компонента Button2 на Выход.

6. Щелкните наверху формы, там, где написано Form1, таким образом вы выделите всю форму. Измените свойство Caption формы на Арифметические действия. Конечный вид проекта представлен на рис. 2.

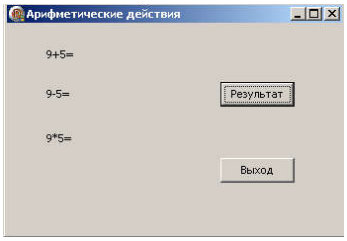
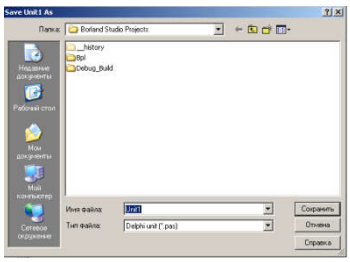
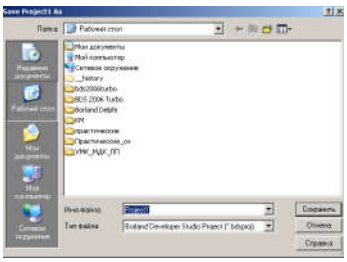


Рис.2 Вид проекта

7. Сохраните проект. Для этого в меню File выберите команду Save All. При первом сохранении файлов проекта обязательно должно появиться два окна:

Правило: каждый проект нужно сохранять в отдельной папке!

8. Дважды нажмите кнопку Button1 (Результат). В результате этого будет создана процедура Button1Click.Откроется код программы, созданная процедура Button1Click. В ней после слова begin запишите:

Label1. Caption:=’9+5=14’;

Label2. Caption:=’9-5=4’;

Label3. Caption:=’9*5=45’;

Процедура должна выглядеть так:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
Label1.Caption:='9+5=4';
Label2.Caption:='9-5=4';
Label3.Caption:='9*5=4';
  end;

end.
```

9. Создайте процедуру Button2Click (нажатие кнопки Выход), аналогично п.8. В открывшемся коде программы запишите: Close;

10. Сохраните проект, запустите его на выполнение, нажав на зеленый треугольник на панели инструментов или клавишу F9.

11. Выполните дополнительные настройки. Выделяя по очереди компоненты, найдите в инспекторе объектов свойство Font, нажмите на три точки слева от свойства. Откроется диалоговое окно для изменения цвета и шрифта:

- установите шрифт надписей 14 пт;
- измените цвет компонентов Label (у кнопок Button цвет изменить нельзя);
- измените цвет формы;

12. Добавьте необходимые компоненты и найдите результат деления числа 9 на число 5.

Рекомендуемая литература:

1. Архангельский, А.Я. Программирование в Delphi для Windows / А.Я. Архангельский. – М.: Бином-Пресс, 2016. – 1248 с.
2. Осипов, Д.Л. Delphi. Профессиональное программирование / Д.Л. Осипов. – СПб.: БХВ-Петербург, 2018. – 1056 с.

3. Эйдлина, Г.М. Delphi. Программирование в примерах и задачах. Практикум. Учебное пособие / Г.М. Эйдлина, К.А. Милорадов. – М.: РИОР, 2018. – 138 с.

Практическая работа № 2
Разработка проекта с условным оператором

Цели работы:

- закрепление полученных знаний по теме «Среда Delphi»;
- формирование умения применять компоненты Button, Label, Edit, Panel;
- формирование умения создавать процедуры обработки событий;
- формирование умения применять условный оператор;
- формирование умения создавать логически правильные и эффективные программы.

Краткие теоретические сведения

Условный оператор обеспечивает выполнение или невыполнение некоторых инструкций в зависимости от соблюдения определенных *условий*. Условный оператор в общем случае предназначен для организации ветвления программы на два направления и имеет формат:

If < условие> Then <инструкция 1> Else [< инструкция2>];

Условие представляет собой выражение логического типа.

Условный оператор работает следующим образом: если *условие* истинно, то выполняется *инструкция 1,* в противном случае - *инструкция 2*. Обе инструкции могут быть составными.

Допускается запись условного оператора в сокращенной форме, когда слово *Else* и *инструкция2* отсутствуют. В этом случае при невыполнении условия управление сразу передается оператору, следующему за условным.

Компонент *Edit* (однострочный редактор) расположен на странице Standard и представляет собой поле ввода текста, в котором возможно отображение и изменение текста.

Компонент *BitBtn* (кнопка с рисунком) расположен на странице Additional. Кнопка с рисунком отличается от стандартной кнопки тем, что помимо заголовка на ней можно отобразить растровое изображение.

Функция IntToStr(I) преобразует целое число I в строку.

Функция StrToInt(S) преобразует строку S в целое число.

Функция FloatToStr(F) преобразует вещественное число F в строку.

Функция StrToFloat(S) преобразует строку S в вещественное число.

Задание. Создать проект *Сравнение чисел*.

Ход работы:

1. Запустите среду Delphi. Создайте новый проект.

2. Расположите компоненты Label, BitBtn, Edit так, как на рис. 1.

Рис. 1 Начальное расположение компонентов

3. Измените свойство Caption компонента Label1 на *Введите первое число* и свойство Caption компонента Label2 на *Введите второе число*. Свойство Caption компонента Label3 оставьте пустым.

4. Уберите надписи в свойстве Text компонентов Edit1 и Edit2. Вид формы представлен на рис. 2.

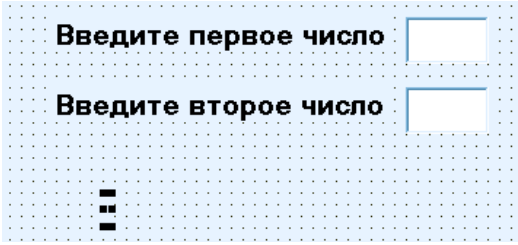


Рис. 2 Вид формы

5. Измените надпись на кнопке BitBtn1. Для этого:

– выделите компонент BitBtn1;

– в Инспекторе объектов в свойстве Kind выберите строку *bkOK*;

– в Инспекторе объектов в свойстве Caption напишите *Результат*;

6. Аналогично создайте надпись для кнопки BitBtn2 (в свойстве Kind выберите строку *bkClose*). Вид кнопок представлен на рис. 3.

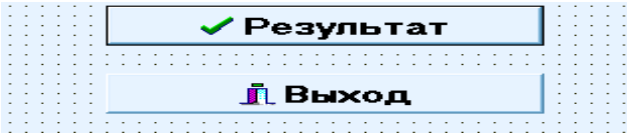


Рис. 3 Вид кнопок

7. Создайте процедуру BitBtn1Click (нажатие кнопки *Результат*). В ней с помощью условного оператора *IF … Then* сравните заданные числа и, в зависимости от результата, выведите соответствующее сообщение (рис. 4).

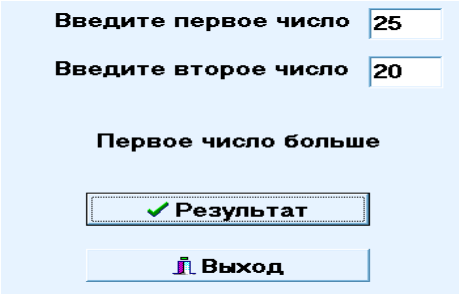


Рис. 4 Вывод сообщения «Первое число больше»

8. Измените название формы на *Сравнение чисел*.
9. Сохраните проект, прокомпилируйте его, результат предъявите преподавателю.
10. Самостоятельно выполните дополнительные настройки: измените цвет надписей и формы, размер и шрифт надписей. Сохраните выполненные изменения, результат предъявите преподавателю.
11. Самостоятельно создайте кнопку *Повтор*, очищающую поле ввода компонентов Edit1 и Edit2 и свойство Caption компонента Label1. Для этого используйте компонент BitBtn, в свойстве Kind этого компонента выберите строку *bkRetry*. Для описания действий этой кнопки создайте процедуру BitBtn3Click. Сохраните выполненные изменения, результат предъявите преподавателю.

Рекомендуемая литература:

1. Архангельский, А.Я. Программирование в Delphi для Windows / А.Я. Архангельский. – М.: Бином-Пресс, 2016. – 1248 с.
2. Осипов, Д.Л. Delphi. Профессиональное программирование / Д.Л. Осипов. – СПб.: БХВ-Петербург, 2018. – 1056 с.
3. Эйдлина, Г.М. Delphi. Программирование в примерах и задачах. Практикум. Учебное пособие / Г.М. Эйдлина, К.А. Милорадов. – М.: РИОР, 2018. – 138 с.

Практическая работа № 3
Вложенные условные операторы

Цели работы:

- закрепление полученных знаний по теме «Среда Delphi»;
- формирование умения применять компоненты Button, BitBtn, Label, Edit;
- формирование умения создавать процедуры обработки событий;
- формирование умения применять вложенные условные операторы;
- формирование умения создавать логически правильные и эффективные программы.

Краткие теоретические сведения

Компонент *Мемо* (многострочный редактор) расположен на странице Standard. Он имеет те же возможности по редактированию текста, что и однострочный редактор. Главное его отличие заключается в том, что он содержит несколько строк текста и доступ к тексту может быть как отдельно по строкам, так и ко всему содержимому сразу.

Функция FloatToStrF имеет формат:

FloatToStrF(Выражение, Формат, Точность, Количество Знаков):String;

Где:

– *Формат* – именованная константа (ffGeneral – общий числовой формат, ffExponent – научный формат, ffFixed – формат всегда с десятичной точкой, ffCurrency – денежный формат);

– *Точность* показывает количество знаков в дробной части, используемых в расчетах (используются значения 7,15,18);

– *КоличествоЗнаков* показывает отображаемое количество знаков в дробной части.

Задание. Создать проект *Площадь кольца*: зная внутренний радиус кольца (20 см), необходимо задать внешний радиус и вычислить площадь кольца.

Ход работы:

1. Запустите среду Delphi. Создайте новый проект.
2. Для создания проекта необходимо использовать компоненты Label, Edit, Memo, Button, BitBtn. Расположите эти компоненты на форме так, как на рис. 1.

Рис. 1 Начальное расположение компонентов

3. Сделайте необходимые надписи для компонентов Label, Button, BitBtn, у компонента Edit надпись уберите (рис. 2).

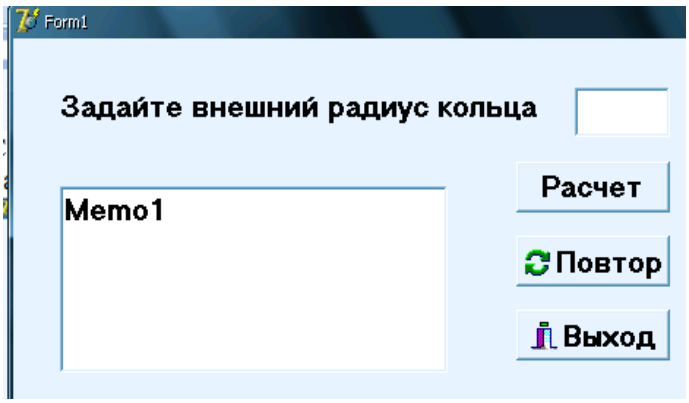


Рис. 2 Вид кнопок и надписи Label1

4. В поле компонента Memo запишите (рис. 3):

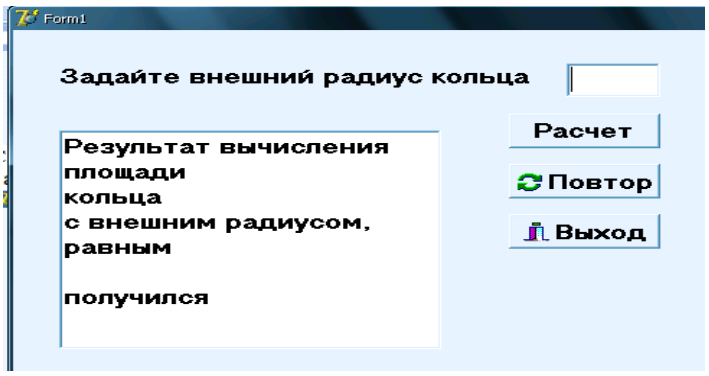


Рис. 3 Текст в окне компонента Memo

5. Создайте процедуру нажатия кнопки *Расчет.* В коде программы необходимо:

– преобразовать текст из поля Edit1 в число;

– проверить, больше ли введенное значение внешнего радиуса 20 см;

– если это условие не выполняется, то необходимо вывести сообщение об этом (используйте еще один компонент Label);

– если введенное значение внешнего радиуса больше 20 см, то выполните вычисление площади кольца.

6. Сохраните проект, прокомпилируйте его, результат предъявите преподавателю.

Рекомендуемая литература:

1. Архангельский, А.Я. Программирование в Delphi для Windows / А.Я. Архангельский. – М.: Бином-Пресс, 2016. – 1248 с.
2. Осипов, Д.Л. Delphi. Профессиональное программирование / Д.Л. Осипов. – СПб.: БХВ-Петербург, 2018. – 1056 с.
3. Эйдлина, Г.М. Delphi. Программирование в примерах и задачах. Практикум. Учебное пособие / Г.М. Эйдлина, К.А. Милорадов. – М.: РИОР, 2018. – 138 с.

Практическая работа №4
Циклы

Цели работы:

– закрепление полученных знаний по темам «Условный оператор», «Циклы»;

– формирование умения организовывать циклы;

– формирование умения создавать обработку событий;

– формирование умения создавать логически правильные и эффективные программы.

Краткие теоретические сведения

Цикл с постусловием целесообразно использовать в случаях, когда тело цикла необходимо выполнить не менее одного раза и заранее неизвестно общее количество повторений цикла.

Цикл с постусловием имеет формат:

Repeat

<инструкция 1>;

<инструкция N>;

Until <условие>;

Условие представляет собой выражение логического типа. Инструкции, заключенные между словами *Repeat* и *Until,* составляют тело цикла и выполняются до тех пор, пока логическое выражение *условие* не примет значение True. Т.к. *условие* проверяется только в конце цикла, тело цикла выполняется минимум один раз.

Один из операторов тела цикла обязательно должен влиять на значение условия, иначе произойдет зацикливание.

Цикл с предусловием целесообразно использовать в случаях, когда число повторений тела цикла заранее неизвестно и тело цикла может ни разу не выполниться.

Цикл с предусловием имеет формат:

While <условие> Do <инструкция>;

Инструкция тела цикла выполняется до тех пор, пока логическое выражение *условие* не примет значение False. Если перед первым выполнением цикла условие не удовлетворяется, то тело цикла не выполнится ни разу и произойдет переход на оператор, следующий за циклом.

Компонент Chart (диаграмма) предназначен для работы со сложными диаграммами различных типов, в том числе объемными. Установка значений свойств этого компонента выполняется при разработке приложения с помощью редактора диаграмм.

Задание. Создать проект *График функции*: пользователю предлагается выбрать функцию, график которой необходимо построить, и задать интервал изменения аргумента X. График строится с помощью компонента Chart, изменение аргумента происходит в цикле.

Ход работы:

1. Запустите среду Delphi. Создайте новый проект.
2. Разместите на форме компоненты как на рис. 1, сделайте им соответствующие подписи.

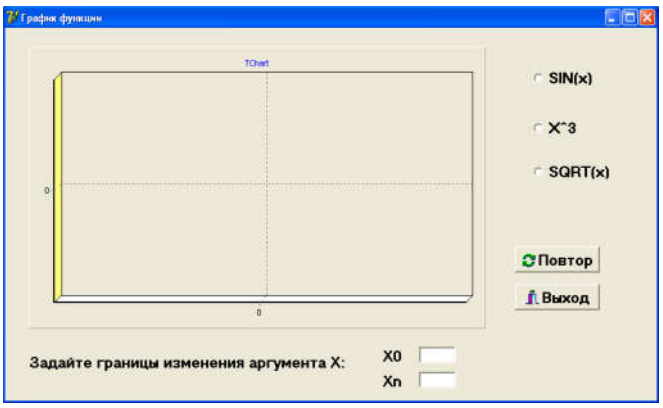


Рис. 1 Вид и расположение компонентов

3. Выделите компонент Chart, нажмите правую кнопку мыши, выберите команду Edit Chart. Откроется встроенный конструктор, с помощью которого осуществляется настройка параметров графика (рис. 2).

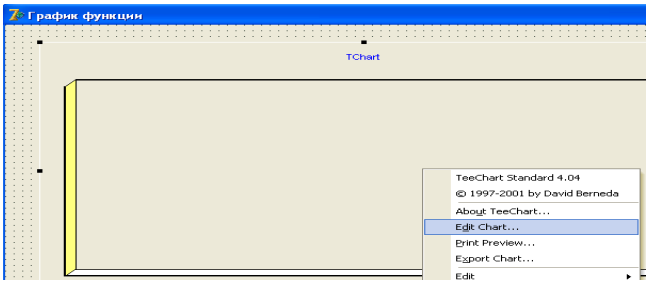


Рис. 2 Настройка параметров графика

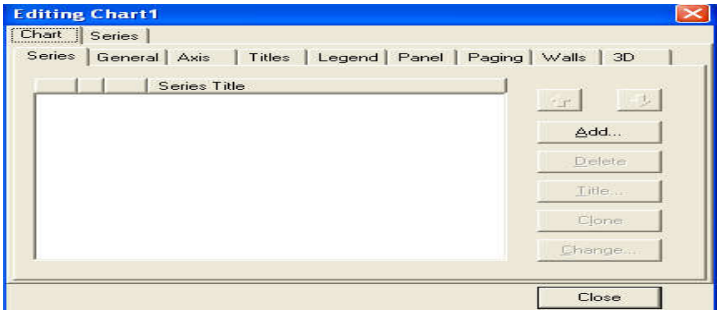


Рис. 3 Кнопка Add

4. Щелкните на кнопке Add (рис. 3), откроется окно выбора типа графика (рис. 4), выберите Line, нажмите OK:

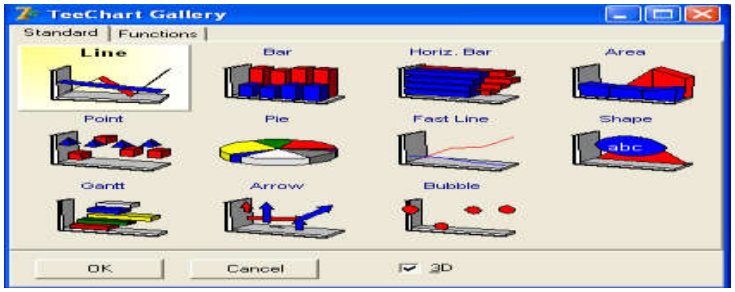


Рис. 4 Выбор типа графика

Опять откроется окно Editing Chart1, в нем перейдите на вкладку Series. Изменение любого параметра в этом окне приведет к изменению внешнего вида графика: нажмите кнопку «Border», выберите стиль Dash Dot, нажмите OK.

5. Перейдите на вкладку Data Source, откройте раскрывающийся список, выберите Function. Это означает, что значения будут формироваться как результат применения функции (рис. 5).

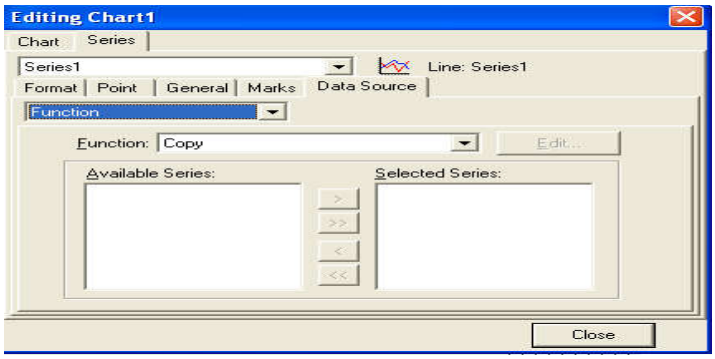


Рис. 5 Вкладка Data Source

6. Далее создайте процедуры построения графиков функций Sin(x), x^3, $\sqrt{x}$.

7. Т.к. квадратный корень можно извлекать только из положительного числа, то необходимо предусмотреть проверку вводимых данных X0 и Xn: если хотя бы одно из этих значений меньше нуля, то с помощью функции ShowMessage выведите сообщение (рис. 6).

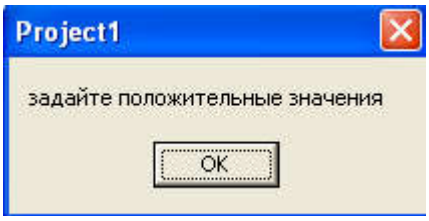


Рис. 6 Вид сообщения

8. Запустите приложение, проверьте построение всех трех графиков на контрольных примерах:

– для функции y=Sin(x) задайте значения X0= -50, Xn=50;
– для функции $y=x^3$ задайте значения X0= -20, Xn=20;
– для функции $y=\sqrt{x}$ задайте значения X0= 1, Xn=60;

9. Сохраните проект. Прокомпилируйте проект.

10. Кнопка *Повтор* должна очищать поверхность построения графика. Создайте процедуру нажатия этой кнопки.

Рекомендуемая литература:

1. Архангельский, А.Я. Программирование в Delphi для Windows / А.Я. Архангельский. – М.: Бином-Пресс, 2016. – 1248 с.
2. Осипов, Д.Л. Delphi. Профессиональное программирование / Д.Л. Осипов. – СПб.: БХВ-Петербург, 2018. – 1056 с.
3. Эйдлина, Г.М. Delphi. Программирование в примерах и задачах. Практикум. Учебное пособие / Г.М. Эйдлина, К.А. Милорадов. – М.: РИОР, 2018. – 138 с.

Практическая работа № 5
Оператор множественного выбора

Цели работы:

- закрепление полученных знаний по теме «Оператор множественного выбора»;
- формирование умения применять оператор множественного выбора;
- формирование умения использовать свойства списков;
- формирование умения создавать логически правильные и эффективные программы.

Краткие теоретические сведения

Компонент ListBox (простой список) расположен на странице Standard и представляет собой прямоугольную область, в которой располагаются его строковые элементы. Если количество строк больше, чем их может поместиться в видимой области списка, то у области отображения появляется полоса прокрутки.

Оператор множественного выбора является обобщением условного оператора и позволяет сделать выбор из произвольного числа имеющихся вариантов, т.е. организовать ветвление на произвольное число направлений. Оператор состоит из выражения, называемого *селектором*, списка вариантов и необязательной ветви Else, имеющий тот же смысл, что и в условном операторе. Формат оператора множественного выбора:

Case <выражение – селектор > Of
<список 1>: <инструкция 1>;

<список N>: <инструкция N>
Else <инструкция >;
End;

Выражение – селектор должно быть порядкового типа. Каждый из вариантов выбора (от *<список 1>* до *<список N>)* представляет собой список констант, отделенных двоеточием от относящейся к данному варианту инструкции, возможно, составной. Список констант выбора состоит из произвольного количества уникальных значений и диапазонов, отделенных друг от друга запятыми. Границы диапазона записываются двумя константами через разделитель «..». Тип констант должен совпадать с типом выражения-селектора.

Задание. Создать проект *Страны - Валюта.* Данный проект должен содержать названия стран и выводить названия соответствующих им валют.

Ход работы:

1. Запустите среду Delphi. Создайте новый проект.

2. Разместите на форме компоненты ListBox, Label, Edit, Button, BitBtn так, как рис. 1.

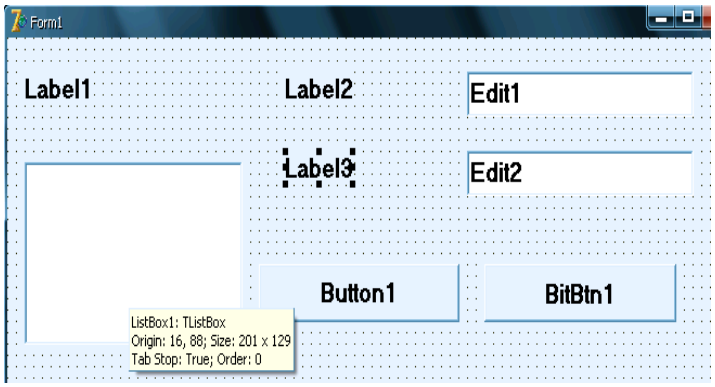


Рис. 1 Начальное расположение компонентов

3. Измените надписи компонентов Label, Button, BitBtn. Поле ввода у компонентов Edit1 и Edit2 очистите (рис. 2).

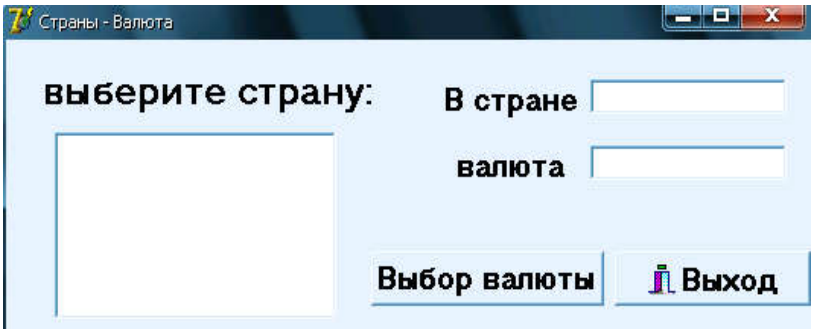


Рис. 2 Вид кнопок и надписей

4. Откройте текстовый редактор компонента ListBox (свойство Items) и введите 6 названий стран (рис. 3).

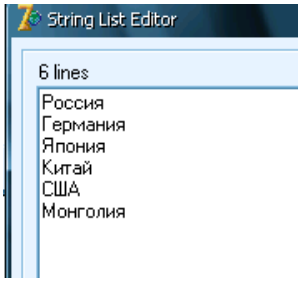


Рис. 3 Заполнение компонента ListBox

5. Создайте процедуру Button1Click (нажатие кнопки *Выбор валюты*). В открывшемся коде программы опишите выбор названия страны и соответствующей ей валюты. Для этого используйте оператор Case. Вместо выражения – селектора используйте свойство ItemIndex компонента ListBox1:

6. Для проверки работы программы необходимо щелкнуть на названии страны, щелкнуть кнопку *Выбор валюты*. В окне *В стране* должно появиться выбранное Вами название страны, в окне *Валюта* - соответствующая ей валюта (рис. 4).

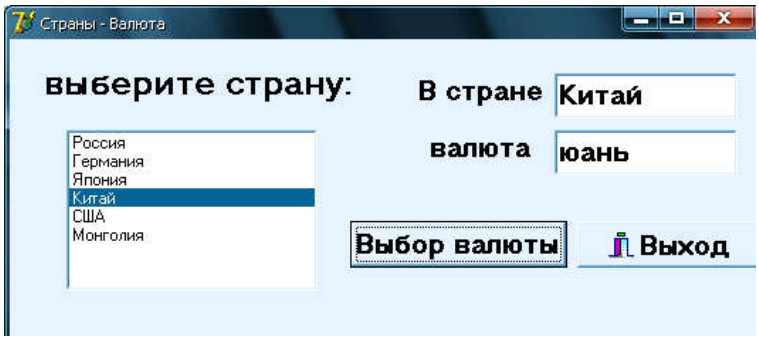


Рис. 4 Результат работы проекта

7. Сохраните проект. Прокомпилируйте проект.
8. Создайте кнопку *Повтор* для очистки полей Edit. После внесенных изменений сохраните проект.

Рекомендуемая литература:

1. Архангельский, А.Я. Программирование в Delphi для Windows / А.Я. Архангельский. – М.: Бином-Пресс, 2016. – 1248 с.
2. Осипов, Д.Л. Delphi. Профессиональное программирование / Д.Л. Осипов. – СПб.: БХВ-Петербург, 2018. – 1056 с.
3. Эйдлина, Г.М. Delphi. Программирование в примерах и задачах. Практикум. Учебное пособие / Г.М. Эйдлина, К.А. Милорадов. – М.: РИОР, 2018. – 138 с.

Практическая работа № 6
Простые списки

Цели работы:

- закрепление полученных знаний по теме «Оператор множественного выбора»;
- формирование умения применять оператор множественного выбора;
- формирование умения использовать свойства списков;
- формирование умения создавать логически правильные и эффективные программы.

Краткие теоретические сведения

В компьютерной графике цвет представляется тремя составляющими: красным, зеленым, голубым (RGB – Red, Green, Blue). В разных пропорциях из этих трех базовых цветов можно получить любой другой. Каждый из цветов представлен в виде одного байта, поэтому для хранения трех цветов достаточно 3 байтов. Только сразу стоит сказать, что на самом деле в Delphi для кодирования цвета отводится не три байта, а четыре. Первый байт используется для обозначения прозрачности, а следующие байты для обозначения цвета.

Один байт может принимать значения от 0 до 255 (в десятичной системе счисления) или от 0 до FF (в шестнадцатеричной системе счисления). В

шестнадцатеричной системе счисления коричневый цвет будет выглядеть $00336699, где 00 – байт прозрачности, 33 – байт для голубого цвета, 66 – байт для зеленого цвета, 99 – байт для красного цвета. Отсюда видно, что на самом деле в памяти цвет хранится как BGR (в обратном порядке). Абсолютно красный цвет – $000000FF, абсолютно зеленый цвет – $0000FF00, абсолютно синий цвет – $00FF0000.

Задание. Создать программу, выполняющую следующие действия.

После запуска программы пользователь выбирает с помощью стрелок на клавиатуре название цвета и нажимает клавишу Enter. На экране появляется название цвета на русском языке и код в формате RGB (рис.1). Программа заканчивает свою работу по нажатию клавиши «Выход».

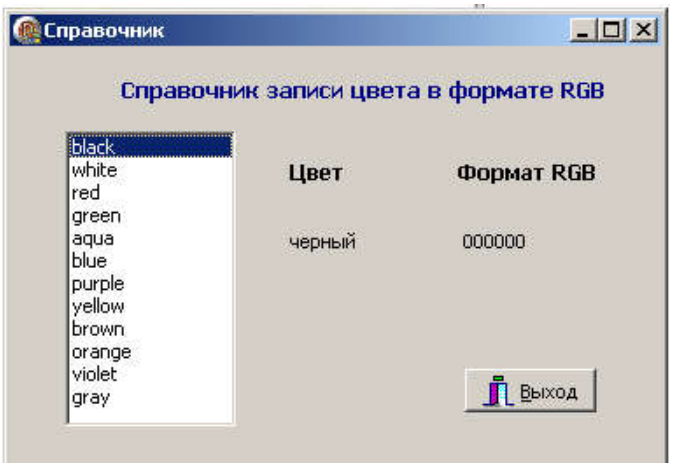


Рис.1

Ход работы:

1. Запустите среду Delphi. Создайте новый проект.
2. Разместите на форме экземпляры компонентов в соответствии с рисунком 2.

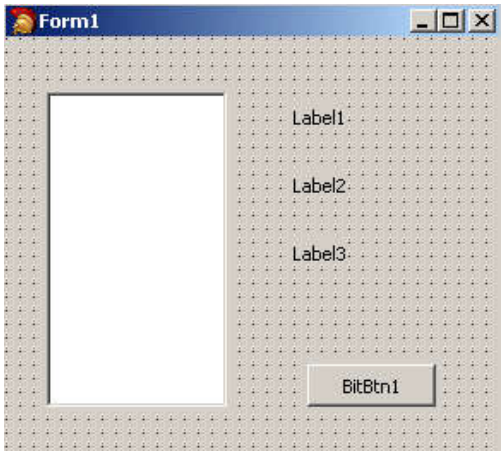


Рис.2

3. Сохраните проект.
4. Выполните следующие действия:

Выделенный объект	Вкладка окна Object Inspector	Имя свойства/ Имя события	Значение/Действие
Form1	Properties	Caption	Справочник
BitBtn1	Properties	Caption	&Выход
		Kind	bkClose
Label1	Properties	Caption	Справочник записи цвета в формате RGB
Label2	Properties	Caption	Цвет в формате RGB
Label3	Properties	Caption	Удалить название объекта

5. Выделите объект ListBox1, найдите свойство Items, щелкните на кнопке с тремя точками, расположенной справа от него. В появившемся окне встроенного редактора String List Editor введите названия цветов, каждый на новой строке.

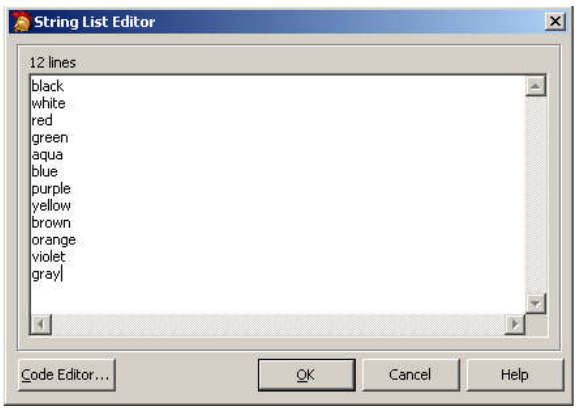


Рис.3

Пояснение:

a) свойство Items содержит элементы списка;

b) список может быть создан при создании формы или во время работы программы;

c) свойство ItemIndex определяет номер элемента, выбранного из списка. Первый элемент имеет номер 0. Если не выбран ни один из элементов, то значение свойства ItemIndex равно – «-1».

6. Сохраните набранный текст в файле под именем Color.txt. Для этого нажмите правую кнопку мыши и выберите режим Save. Для выхода из встроенного редактора щелкните на кнопке OK.

Комментарий. Просмотреть содержимое созданного текстового файла Color.txt, можно с помощью любого текстового редактора, а также внести изменения в тестовый файл, не используя встроенный редактор Delphi.

7. Выполните следующие действия:

Выделенный объект	Вкладка окна Object Inspector	Имя свойства/ Имя события	Значение/Действие
ListBox1	Events	OnKeyPress	if key=#13 then Case Listbox1.ItemIndex of 0: Label3.Caption:=’черный 000000’; 1: Label3.Caption:=’белый FFFFFF’; 2: Label3.Caption:=’красный FF0000’; 3: Label3.Caption:=’зеленый 00FF00’; 4: Label3.Caption:=’бирюзовый 00FFFF’; 5: Label3.Caption:=’синий 0000FF’; 6: Label3.Caption:=’фиолетовый FF00FF’; 7: Label3.Caption:=’желтый FFFF00’; 8: Label3.Caption:=’коричневый 996633’; 9: Label3.Caption:=’оранжевый FF8000’; 10: Label3.Caption:=’лиловый FF0008’; 11: Label3.Caption:=’серый 999999’; end;

Подсказка. Шрифт, который используется для вывода текста, определяется значением свойства Font соответствующего объекта Label.Свойство Font представляет собой объект типа TFont, который имеет свои свойства. Изменить цвет, выводимый на объект Label можно с помощью программы, изменив свойство Color:

Label3.Font.Color:=$FFFFFF; // устанавливается белый цвет

Рекомендуемая литература:

1. Архангельский, А.Я. Программирование в Delphi для Windows / А.Я. Архангельский. – М.: Бином-Пресс, 2016. – 1248 с.
2. Осипов, Д.Л. Delphi. Профессиональное программирование / Д.Л. Осипов. – СПб.: БХВ-Петербург, 2018. – 1056 с.
3. Эйдлина, Г.М. Delphi. Программирование в примерах и задачах. Практикум. Учебное пособие / Г.М. Эйдлина, К.А. Милорадов. – М.: РИОР, 2018. – 138 с.

Практическая работа № 7
Флажки и переключатели

Цели работы:

- закрепление полученных знаний по темам «Элементы языка Delphi», «Условный оператор», «Циклы»;
- формирование умения применять свойства флажков и переключателей;
- формирование умения применять свойства различных комбинированных списков;
- закрепить навыки работы условного оператора;
- формирование умения создавать логически правильные и эффективные программы.

Краткие теоретические сведения

Переключатель (зависимый переключатель) позволяет выбрать единственное значение из определенного множества значений, представленного группой переключателей. Он может находиться в выбранном или невыбранном состоянии. Одновременно можно выбрать только один переключатель в группе.

Флажок (независимый переключатель) отличается от переключателя тем, что в группе флажков одновременно можно установить флажки в любой комбинации (в том числе могут быть установлены или сброшены все флажки). Флажок может находиться в установленном или сброшенном состоянии.

Компонент CheckBox (флажок) находится на странице Standard. Флажок независимо действует от других флажков, несмотря на то, что по функциональному назначению флажки часто объединяют в группы с общим названием. Флажок выглядит как прямоугольник с текстовым заголовком. Если в нем есть галочка, то обозначенная этим флажком опция включена. Если прямоугольник пуст, то флажок снят.

Компонент RadioButton (переключатель) находится на странице Standard и выглядит как кружок с текстовым заголовком. При выбранном состоянии опции в кружке появляется точка.

Переключатели обычно располагаются по группам, визуально выделенным в форме. Выбор переключателя является взаимоисключающим, т.е. при выборе одного переключателя другие становятся невыбранными. Контейнерами для переключателей часто служит компонент GroupBox.

Компонент CheckListBox (комбинированный список) находится на странице Additional и представляет собой комбинацию простого списка и набора флажков.

Компонент RadioGroup (группа переключателей) находится на странице Standard и объединяет в себе переключатели RadioButton. Это упрощает организацию взаимодействия переключателей. Группа переключателей

RadioGroup может содержать другие элементы управления, например, флажок CheckBox или однострочный редактор Edit.

Задание. Создайте проект *Меню*: клиенту кафе предлагается меню, в котором представлены первые, вторые блюда и напитки. Первое блюдо должно выбираться только одно, остальные – по желанию. После сделанного выбора должна подсчитываться итоговая сумма заказа и выводиться на экран вместе со списком выбранных блюд и пожеланием *Приятного аппетита*. Должна быть предусмотрена возможность повторного заказа.

Ход работы:

1. Запустите среду Delphi. Создайте новый проект.
2. Разместите на форме компоненты как на рис. 1, сделайте им соответствующие подписи (рис. 2).

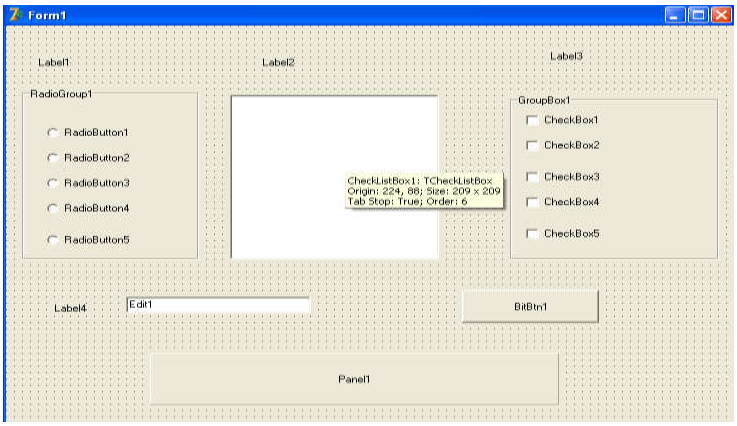


Рис. 1 Начальное расположение компонентов

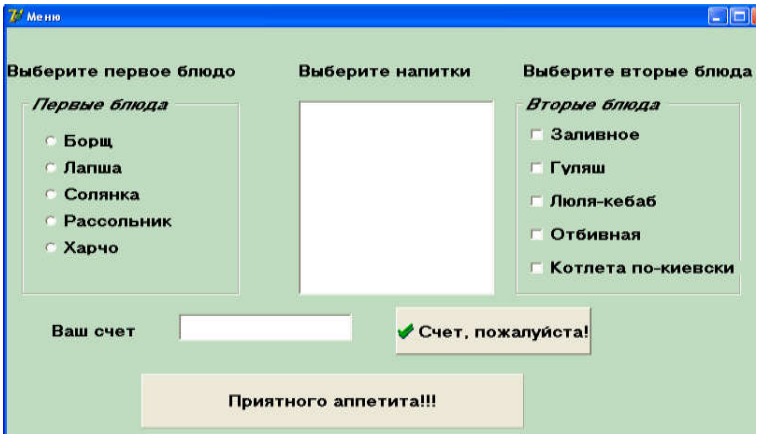


Рис. 2 Вид формы после изменения надписей

3. Сделайте невидимым компонент Panel1.
4. В Инспекторе объектов для компонента CheckListBox найдите свойство Items, нажмите **…** правее свойства. Откроется редактор для заполнения поля данными. Введите данные (рис. 3), нажмите ОК:

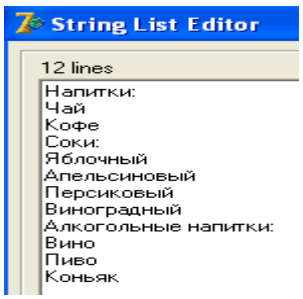


Рис. 3 Заполнение компонента CheckListBox

В итоге список напитков должен выглядеть так, как на рис. 4.

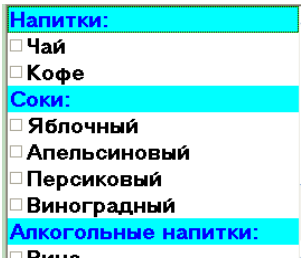


Рис. 4 Вид списка напитков

5. Положите на форму компонент ComboBox1 (рис. 5), сделайте ему подпись:

Рис. 5 Компонент ComboBox1

В список ComboBox1 будут записываться названия и цены выбранных вами блюд.

6. Создайте процедуру BitBtn1Click для кнопки *Счет, пожалуйста!* В данной процедуре необходимо проанализировать состояние всех флажков и переключателей в списках, и, если какие-либо из них нажаты, то включить выбранное блюдо в список ComboBox1. Также необходимо увеличить сумму счета.

7. Сохраните проект. Прокомпилируйте проект.

8. После нажатия кнопки *Счет, пожалуйста!* должен появляться компонент Panel1 с надписью *Приятного аппетита!!!* Выполните это действие.

9. Положите на форму еще одну кнопку BitBtn, сделайте ей подпись, как на рис. 6.

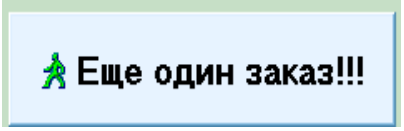


Рис. 6 Вид кнопки BitBtn

После нажатия этой кнопки должны убираться все флажки и отключаться переключатели. Также должно очищаться поле компонентов Edit1, Panel1 и список компонента ComboBox1.

Сохраните проект, запустите на выполнение. Выберите одно первое блюдо, три напитка и два вторых блюда. Нажмите на кнопку *Счет, пожалуйста!* Выбранные вами блюда должны отобразиться в списке, сумма заказа – в окне Edit1. Проверьте правильность счета. Также должна появиться надпись *Приятного аппетита!!!*

В окончательном варианте проект должен выглядеть приблизительно как на рис. 7:

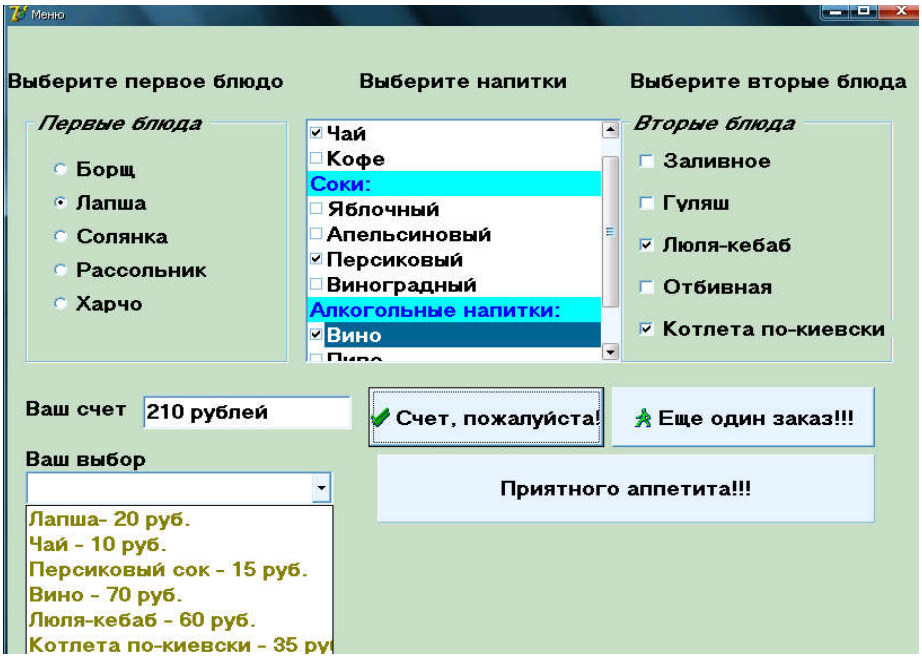


Рис. 7 Примерный вид формы

10. Нажмите на кнопку *Еще один заказ!!!* – заказ должен отмениться, поля очиститься, надпись *Приятного аппетита!!!* должна исчезнуть. После внесенных изменений сохраните проект.

Рекомендуемая литература:

1. Архангельский, А.Я. Программирование в Delphi для Windows / А.Я. Архангельский. – М.: Бином-Пресс, 2016. – 1248 с.
2. Осипов, Д.Л. Delphi. Профессиональное программирование / Д.Л. Осипов. – СПб.: БХВ-Петербург, 2018. – 1056 с.
3. Эйдлина, Г.М. Delphi. Программирование в примерах и задачах. Практикум. Учебное пособие / Г.М. Эйдлина, К.А. Милорадов. – М.: РИОР, 2018. – 138 с.

Практическая работа № 8
Комбинированные списки

Цели работы:

– закрепление полученных знаний по темам «Элементы языка Delphi», «Условный оператор», «Циклы»;

- формирование умения применять свойства флажков;
- формирование умения применять свойства различных комбинированных списков;
- закрепить навыки работы условного оператора;
- формирование умения создавать логически правильные и эффективные программы.

Краткие теоретические сведения

Комбинированный список *TComboBox* представляет собой комбинацию списка TListBox и редактора TEdit, и поэтому большинство его свойств и методов заимствовано у этих компонентов. Существуют пять модификаций компонента, определяемых его свойством style: *csSimple, csDropDown, csDropDownList, csOwnerDrawFixed И csOwner DrawVariabie.* В первом случае список всегда раскрыт, в остальных он раскрывается после нажатия кнопки справа от редактора. В модификации csDropDownList редактор работает в режиме отображения выбора и его нельзя использовать для ввода новой строки (в других модификациях это возможно).

Задание. Разработать программу для составления словаря новых терминов. Должна иметься возможность внесения в словарь изменений, дополнений или сокращения его. При выборе термина на контрольной панели должны появляться его номер и общее количество слов в словаре.

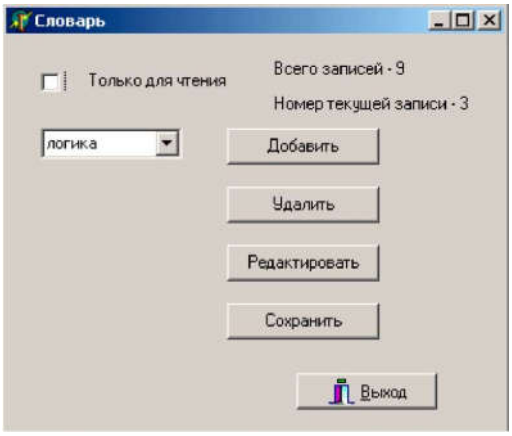


Рис 1 Вид формы

Ход работы:

1. Запустите среду Delphi. Создайте новый проект.
2. Разместите в форме объекты в соответствии с рис.1 и присвойте заголовки меткам и панелям.

Объект	Вкладка окна Object Inspector	Имя свойства/Имя события	Значение/ Действие

Forml	Properties	Caption	Словарь
BitBtnl	Properties	Caption	&Выход
		Kind	bkClose
ComboBoxl	Properties	Text	вставка пробела
		Item	открытие двойным щелчком списка String list editor. Ввод нескольких терминов, сохранение их в файле Glostext.txt, предварительно убедившись, что выбрана нужная папка
Button 1	Properties	Caption	Добавить
	Events	OnClick	ComboBox1.Items.Add (ComboBox1.Text); if ComboBox1.ItemIndex = -1 then ComboBox1.Text :=''; Последний оператор очищает строку ввода после того, как текст из нее попадает в список
Button2	Properties	Caption	Удалить
	Events	OnClick	If MessageDlg('Bbi действительно хотите удалить запись?', mtWarning, [mbYes, mbNo], 0) = mrYes then ComboBox1.Items.Delete(ComboBox1.Item Index);
Button4	Properties	Caption	Сохранить
	Events	OnClick	ComboBox1.Items.SaveToFile ('glostext.txt');
Forml	Events	OnCreate	ComboBox1.Items.LoadFromFile ('glostext.txt');
Button3	Properties	Caption	Редактировать
	Events	OnClick	ComboBox1.Items.Delete (num); ComboBox1.Items.Add (ComboBox1.Text); if ComboBox1.ItemIndex = -1 then ComboBox1.Text:='';
ComboBoxl	Events	OnClick	num:= ComboBox1.ItemIndex; предварительно описать в разделе Var целочисленную переменную num

Комментарии: новая переменная num необходима для сохранения номера выбранной строки. При внесении изменений выбранной строкой становится строка ввода, для которой Itemlndex = -1.

Предусмотреть режим работы со списком, допускающий только чтение.

Выделенный объект	Вкладка окна Object Inspector	Имя свойства/Имя события	Значение/ Действие
CheckBoxl	Properties	Caption	Только чтение
	Events	OnClick	if CheckBoxl .Checked = False then ComboBoxl.Style:= csDropDown else ComboBoxl. Style := csDropDownList; Свойство Checked у выбранного выключателя имеет значение True, а в исходном состоянии - значение False.

Модификации csOwnerDrawFixed и csOwnerDrawVariable используются программой прорисовки элементов списка.

В режиме *Только чтение* сделать недоступной кнопку *Добавить:*

Выделенный объект	Вкладка окна Object Inspector	Имя свойства/ Имя события	Значение/ Действие
CheckBoxl	Events	OnClick	if CheckBoxl. Checked = false then Button 1.Enabled := true else Button 1.Enabled := false;

Аналогично сделать недоступными кнопки Редактировать, Удалить, Сохранить.

Рекомендуемая литература:

1. Архангельский, А.Я. Программирование в Delphi для Windows / А.Я. Архангельский. – М.: Бином-Пресс, 2016. – 1248 с.
2. Осипов, Д.Л. Delphi. Профессиональное программирование / Д.Л. Осипов. – СПб.: БХВ-Петербург, 2018. – 1056 с.
3. Эйдлина, Г.М. Delphi. Программирование в примерах и задачах. Практикум. Учебное пособие / Г.М. Эйдлина, К.А. Милорадов. – М.: РИОР, 2018. – 138 с.

Практическая работа №9
Древовидные списки

Цели работы:

- закрепление полученных знаний по темам «Элементы языка Delphi», «Условный оператор», «Циклы»;
- формирование умения применять свойства древовидных списков;
- закрепить навыки работы оператора множественного выбора;
- формирование умения создавать логически правильные и эффективные программы.

Краткие теоретические сведения

Простые и комбинированные списки позволяют отображать описания только однородных объектов или понятий, которые не зависят друг от друга. Очень часто встречаются объекты, которые объединяют в себе несколько дочерних объектов, а те в свою очередь также являются родителями по отношению к другим. Таким образом, взаимосвязь этих объектов можно представить в виде *иерархической структуры*, иначе называемой *деревом*. Для работы с подобными структурами предназначен компонент TreeView, расположенный на странице Win 32.

Элементами древовидного списка являются объекты класса TTreeNode. Сам список представляет собой прямоугольную область, в которой отображаются заголовки его элементов.

Условно все элементы можно разделить на две категории: узлы и листья. Узловым называется такой элемент списка, который включает дочерние элементы. По умолчанию узел находится в свернутом состоянии.

Листьями называются конечные элементы дерева, которые не содержат вложенных элементов.

Процедура *LoadFromFile (const FileName: String)* заполняет список содержимым указанного текстового файла, при этом предыдущее содержимое списка стирается. Если заданный файл отсутствует на диске, то возникает исключение.

Задание. Создать проект *Видеотека*: пользователю предлагается список фильмов видеотеки в виде дерева объектов. После выбора нужного фильма должна отображаться информация о местонахождении данного фильма (номер полки) и краткое содержание фильма.

Ход работы:

1. Загрузите Delphi. Создайте новый проект.
2. Разместите на форме компоненты как на рис. 1, сделайте им соответствующие подписи (рис. 2), переименуйте форму. Компонент Label3 оставьте без подписи – в него будет выводиться название выбранного фильма:

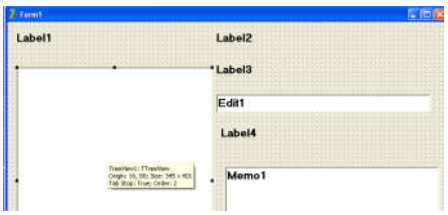


Рис. 1 Начальное расположение компонентов

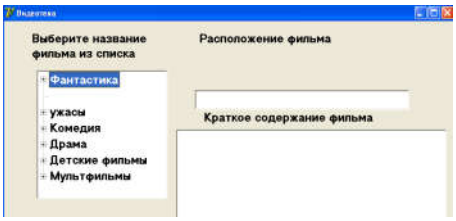


Рис. 2 Итоговый вид формы

3. Вызовите контекстное меню (правая кнопка мыши) компонента TreeView1, выберите команду ItemsEditor. Откроется редактор для наполнения содержимым дерева объектов. Дерево объектов может состоять из узлов (т.е. элементов, содержащих дочерние элементы) и листьев (т.е. элементов, не содержащих дочерних элементов). Формирование дерева начинается с создания его корневого элемента. Для этого щелкните на кнопке New Item. В окне Text введите заголовок первого элемента дерева Фантастика. Далее нажмите на кнопку NewSubItem для ввода названия фильма – введите Необитаемый остров. Для ввода следующего названия фантастического фильма нажмите New Item. Самостоятельно составьте список фантастических фильмов из 3 элементов. Для того чтобы сформировать следующий узел, выделите слово Фантастика, затем нажмите New Item и т.д.

4. По приведенной схеме сформируйте дерево объектов с узловыми элементами по 3 дочерних элемента в каждом:

– Фантастика (уже сделано);
– Ужасы;
– Комедия;
– Драма;
– Детские фильмы;
– Мультфильмы.

Примерный вид дерева объектов представлен на рис. 3.

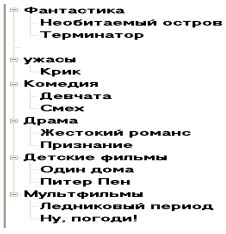


Рис. 3 Заполнение дерева объектов

5. Найдите в коде программы раздел private и опишите в нем переменную класса TTreeNode таким образом:

```
private

s:TTreenode;
```

Эта переменная позволит работать с выбранными элементами дерева объектов.

6. Создайте процедуру TreeView1Click. В ней напишите:

```
s:=TreeView1.Selected;      //происходит сохранение выбранной строки из
                            //дерева объектов
```

7. Далее примените оператор множественного выбора case:

```
case s.AbsoluteIndex of
1: begin
edit1.Text:= 'первая полка';
end;
end;
```

Свойство AbsoluteIndex переменной s содержит номер выбранной вами строчки в дереве объектов.

Продолжите структуру оператора case. Напишите только те номера строк, которые соответствуют фильмам. Строки, содержащие жанры, не используйте!

8. В Блокноте создайте несколько текстовых файлов с описанием каждого фильма, например, как на рис. 4.

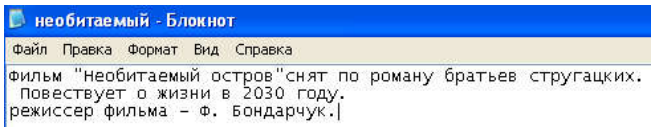


Рис. 4 Текстовый файл с описанием фильма

Сохраните эти файлы в папке с файлами проекта под именами фильмов.

9. Дополните содержимое оператора case выводом текста в поле Memo. Первая секция оператора будет выглядеть так:

```
case s.AbsoluteIndex of
1: begin
edit1.Text:= 'первая полка';
memo1.Lines.LoadFromFile('необитаемый.txt');
end;
end;
```

Здесь метод LoadFromFile загружает содержимое файла 'необитаемый.txt' в поле Memo. Аналогично сделайте для всех секций оператора case.

10. Выделите компонент TreeView1, в инспекторе объектов найдите свойство HotTrack, задайте ему значение True. Таким образом, заголовки элементов будут подчеркиваться при наведении на них указателя мыши.

11. Прокомпилируйте проект.

12. Положите на форму компонент Panel и отобразите в нем общее количество строк в дереве объектов. Для этого примените свойство Items.Count компонента TreeView1.

13. Дополните код программы таким образом, чтобы в компонент Label3 выводилось название выбранного фильма. После внесенных изменений сохраните проект.

Рекомендуемая литература:

1. Архангельский, А.Я. Программирование в Delphi для Windows / А.Я. Архангельский. – М.: Бином-Пресс, 2016. – 1248 с.
2. Осипов, Д.Л. Delphi. Профессиональное программирование / Д.Л. Осипов. – СПб.: БХВ-Петербург, 2018. – 1056 с.
3. Эйдлина, Г.М. Delphi. Программирование в примерах и задачах. Практикум. Учебное пособие / Г.М. Эйдлина, К.А. Милорадов. – М.: РИОР, 2018. – 138 с.

Практическая работа №10
Таблицы

Цели работы:

- закрепление полученных знаний по темам «Элементы языка Delphi», «Условный оператор», «Циклы»;
- формирование умения создавать таблицы;
- формирование умения работать с данными таблицы;
- формирование умения создавать логически правильные и эффективные программы.

Краткие теоретические сведения

Таблица представляет собой элемент, позволяющий отображать данные, разбитые на строки и столбцы.

Компонент StringGrid (таблица) расположен на странице Additional и позволяет хранить и автоматически отображать текстовую информацию. Этот компонент также называют *сеткой* или *сеткой строк*. Автоматически выводится только сетка, а за прорисовку содержимого ячеек отвечает разработчик программы.

Задание. Создать проект *Таблица успеваемости*: вводятся оценки, полученные студентами на экзаменах, находится средний балл. Затем делается вывод о получении студентом стипендии.

Ход работы:

1. Загрузите Delphi. Создайте новый проект.
2. Разместите на форме компоненты так, как на рис. 1.

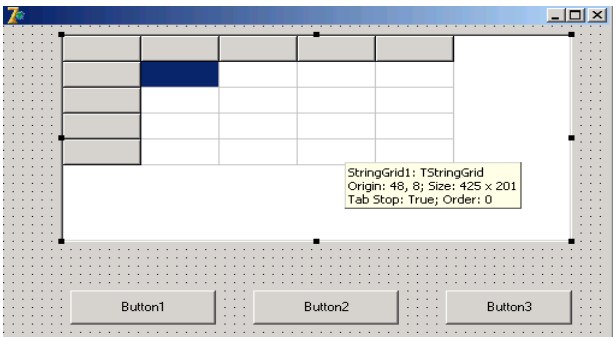


Рис. 1 Расположение компонентов

Измените названия кнопок и формы (рис. 2).

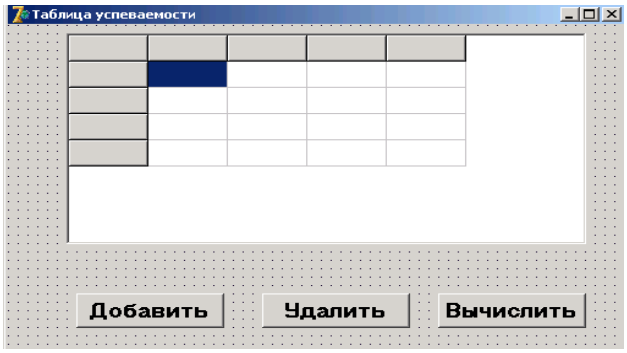


Рис. 2 Вид формы и кнопок

3. Таблица должна иметь следующий вид (рис. 3).

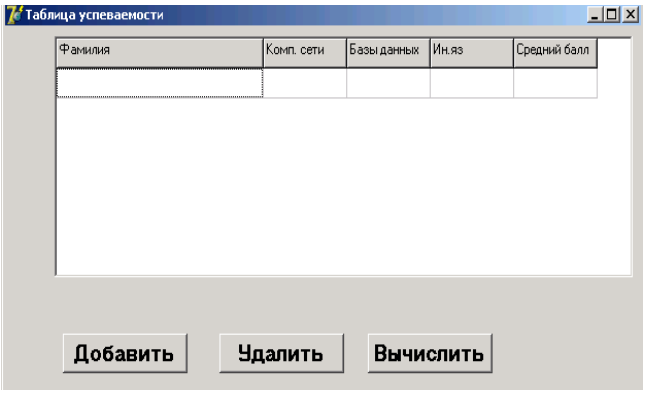


Рис. 3 Итоговый вид таблицы

4. Создайте процедуру обработки события Button1Click – нажатия на кнопку *Добавить*.

5. Аналогично создайте процедуру обработки события Button2Click – нажатия на кнопку *Удалить*.

6. Создайте процедуру обработки события Button3Click – нажатия на кнопку *Вычислить*. Эта процедура должна подсчитывать средний балл для каждого студента.

7. Сохраните проект, проверьте его работу.

8. Добавьте в таблицу еще один столбец, сделайте ему подпись *Стипендия* и выводите в нем слово *Да*, если студент будет получать стипендию, и *Нет* - если не будет.

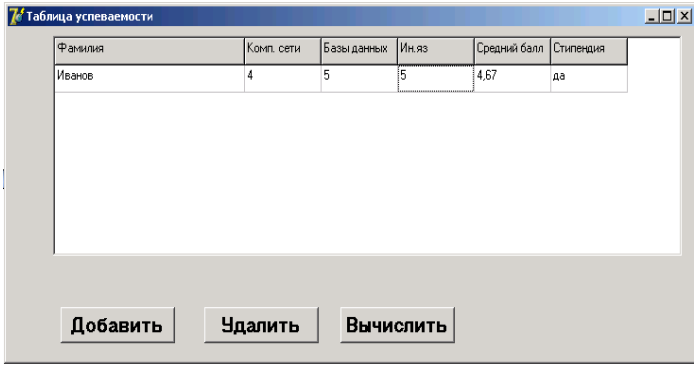


Рис. 4 Заполнение таблицы

9. Измените программу таким образом, чтобы таблица содержала данные о пяти предметах.

Рекомендуемая литература:

1. Архангельский, А.Я. Программирование в Delphi для Windows / А.Я. Архангельский. – М.: Бином-Пресс, 2016. – 1248 с.
2. Осипов, Д.Л. Delphi. Профессиональное программирование / Д.Л. Осипов. – СПб.: БХВ-Петербург, 2018. – 1056 с.
3. Эйдлина, Г.М. Delphi. Программирование в примерах и задачах. Практикум. Учебное пособие / Г.М. Эйдлина, К.А. Милорадов. – М.: РИОР, 2018. – 138 с.

Практическая работа №11
Навигация по файловой системе

Цели работы:

- закрепление полученных знаний по темам «Элементы языка Delphi», «Условный оператор», «Циклы»;
- формирование умения работать с компонентами навигации;
- формирование умения создавать логически правильные и эффективные программы.

Краткие теоретические сведения

Компонент ComboBox (комбинированный список) расположен на странице Standard. Комбинированный список объединяет поле редактирования и список. Пользователь может выбирать значение из списка или вводить его непосредственно в поле. Сам список может быть простым или раскрывающимся.

На странице Win 3.1 расположены компоненты навигации по файловой системе. Компонент DriveComboBox содержит список дисковых устройств системы. Компонент DirectoryListBox включает структуру каталогов текущего диска. Компонент FileListBox содержит список файлов текущего каталога.

Данные компоненты располагаются на форме рядом друг с другом и связываются между собой для грамотной навигации по файловой системе.

Процедура *SaveToFile (const FileName: String)* сохраняет строковые элементы списка в файле *FileName.* Если заданный файл отсутствует на диске, то он создается.

Функция *Add(const S: String): Integer* служит для добавления строк в список. Этот метод добавляет заданную параметром S строку в конец списка, а в качестве результата возвращает положение нового элемента в списке.

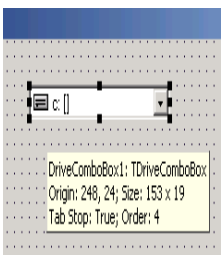
Задание. Создайте проект, в котором можно выбрать необходимый файл, выполняя навигацию по файловой системе, просмотреть его содержимое, выполнить изменения и сохранить их.

Ход работы:

1. Загрузите Delphi. Создайте новый проект.
2. Создайте на своем сетевом диске папку *Список.* Создайте в Блокноте текстовый файл *Список_группы.txt* и сохраните его в папке *Список.* В файл *Список_группы* запишите список студентов своей подгруппы (не по алфавиту). Данный список должен примерно выглядеть следующим образом:

Аникин Петр
Клюкин Валерий
Бобров Иван
Валиулин Руслан
Гладков Олег
Дядченко Мария
Егоров Роман
Журова Анна
Звягинцев Алексей
Иванов Андрей

3. Разместите на форме компоненты ComboBox (страница Standard), DriveComboBox, DirectoryListBox, FileListBox (все 3 компонента со страницы Win 3.1) как на рисунках 1, 2, 3:



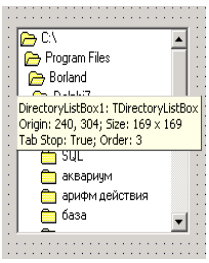


Рис. 1 Компоненты DriveComboBox и DirectoryListBox

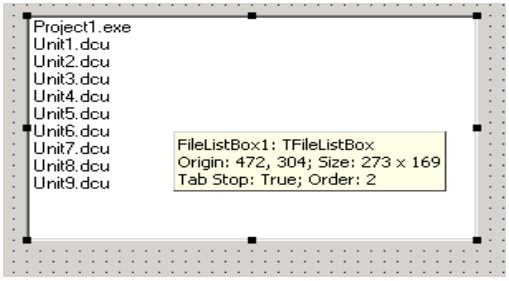


Рис. 2 Компонент FileListBox

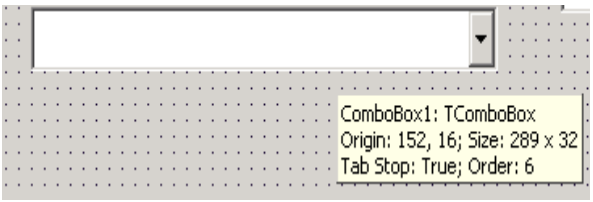


Рис. 3 Компонент ComboBox

4. Положите на форму кнопку Button, измените надпись на кнопке на *Загрузить список*. Сгенерируйте процедуру Button1Click. В этой процедуре с помощью метода LoadFromFile выполните загрузку файла *Список_группы.txt* в поле компонента ComboBox1.

Запустите проект, проверьте работу кнопки – по щелчку на ней в поле компонента ComboBox1 должен появляться список вашей группы (рис.4).

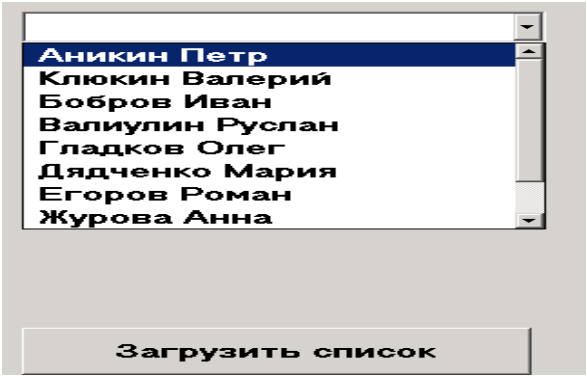


Рис. 4 Работа кнопки «Загрузить список»

5. Компонент ComboBox1 позволяет во время работы проекта менять содержимое списка. Для этого создайте процедуру ComboBox1KeyPress, в ней выполните проверку (*с помощью оператора if ... then*) нажатия клавиши Enter: если эта клавиша нажата, то введенную строку нужно добавить в список (*проверяемое условие после ключевого слова if запишите самостоятельно*). Запишите после ключевого слова then:

6. В этой же процедуре сохраните список в исходном файле с помощью метода SaveToFile. Этот метод работает аналогично методу LoadFromFile.

В итоге проект должен выглядеть как на рис. 5.

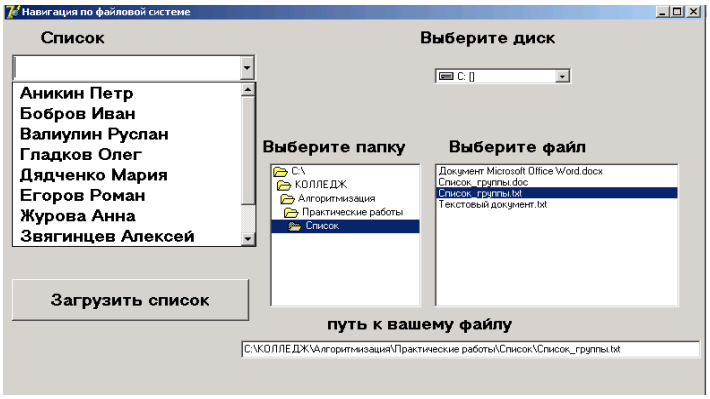


Рис. 5 Итоговый вид проекта

После внесенных изменений сохраните проект.

Рекомендуемая литература:

1. Архангельский, А.Я. Программирование в Delphi для Windows / А.Я. Архангельский. – М.: Бином-Пресс, 2016. – 1248 с.
2. Осипов, Д.Л. Delphi. Профессиональное программирование / Д.Л. Осипов. – СПб.: БХВ-Петербург, 2018. – 1056 с.
3. Эйдлина, Г.М. Delphi. Программирование в примерах и задачах. Практикум. Учебное пособие / Г.М. Эйдлина, К.А. Милорадов. – М.: РИОР, 2018. – 138 с.

Практическая работа № 12
Главное и контекстное меню

Цели работы:

- закрепление полученных знаний по темам «Элементы языка Delphi», «Условный оператор», «Циклы»;
- формирование умения создавать меню на форме;
- формирование умения работать с данными меню;
- формирование умения создавать логически правильные и эффективные программы.

Краткие теоретические сведения

На странице Dialogs представлены компоненты для вызова стандартных диалогов Windows. Объекты, представленные на данной странице, невидимы во время выполнения.

Компонент OpenDialog открывает диалоговое окно, позволяющее открывать (выбирать) файлы.

Компонент SaveDialog открывает диалоговое окно, позволяющее сохранять файлы.

Компонент MainMenu (главное меню) находится на странице Standard. Главное меню формы располагается в верхней части формы под ее заголовком и содержит наиболее общие команды приложения.

Буфер обмена представляет собой область оперативной памяти, которая используется операционной системой для временного хранения данных. Для выполнения операций обмена данными через буфер в Delphi предназначен класс TClipboard. С помощью методов объекта Clipboard можно выполнять стандартные операции:

- метод *CutToClipboard* вырезает данные из источника и помещает в буфер;
- метод *CopyToClipboard* копирует данные из источника и помещает в буфер;
- метод *PasteFromClipboard* вставляет данные из буфера;

Метод *SelectAll* осуществляет выделение всего текста.

Контекстное (всплывающее меню) появляется при размещении указателя в форме или в области некоторого элемента управления и нажатии правой кнопки мыши. Обычно контекстное меню содержит команды, влияющие только на тот объект, для которого вызвано это меню, поэтому такое меню называют локальным.

Компонент PopUpMenu (контекстное меню) находится на странице Standard. Для создания и изменения контекстного меню в процессе разработки приложения используется конструктор меню.

Задание 1. Создайте проект *Текстовый редактор* с возможностью загружать, исправлять, сохранять текстовые файлы.

Ход работы:

1. Загрузите Delphi. Создайте новый проект.
2. Разместите на форме компоненты как на рис. 1, сделайте им соответствующие подписи, как на рис. 2 (слева вверху расположен компонент MainMenu, справа вверху компоненты OpenDialog и SaveDialog):

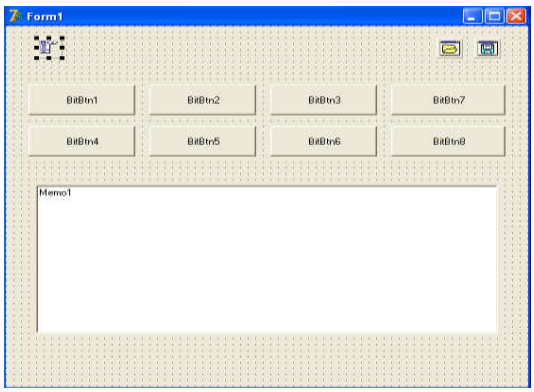


Рис. 1 Начальное расположение компонентов

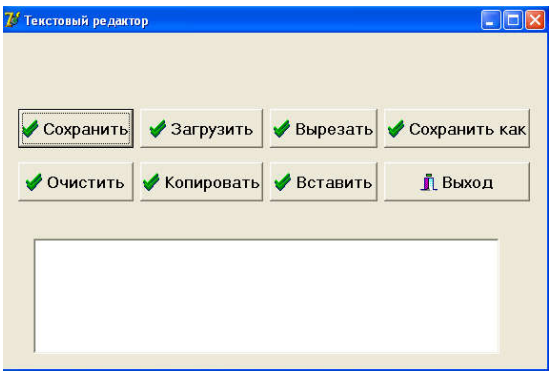


Рис. 2 Итоговый вид формы

3. Установите свойству формы ActiveControl значение Memo1, это позволит при загрузке формы поместить фокус ввода сразу в поле компонента Memo1.

4. Дважды щелкните на компоненте MainMenu, откроется окно редактора (рис. 3)

Рис. 3 Окно редактора главного меню

Сразу активным стал заголовок первого меню. Чтобы создать этот заголовок, в инспекторе объектов в свойстве *Caption* напишите *Файл*. Это слово появится в меню. Дальше для меню *Файл* необходимо создать подпункты. Для этого в редакторе щелкните на слове *Файл,* активируйте первый подпункт меню, в инспекторе объектов в свойстве *Caption* напишите *Новый* (рис. 4*)*.

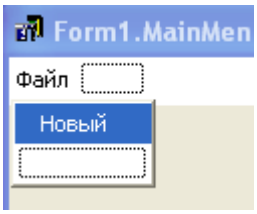


Рис. 4 Пункт меню «Новый»

Аналогично создайте подпункты меню Файл: Открыть, Сохранить, Сохранить как, Выход . Аналогично создайте меню Правка с подпунктами: Вырезать, Скопировать, Вставить, Выделить все (рис. 5).

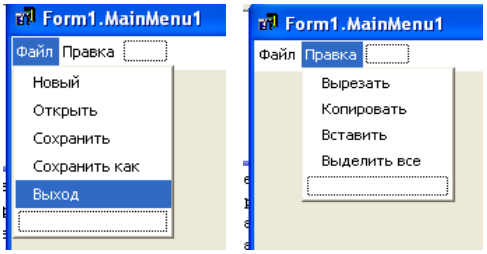


Рис. 5 Пункты меню Файл и Правка

5. Сохраните проект. Запустите проект на выполнение, проверьте структуру меню.

6. Создайте процедуру BitBtn4Click для кнопки Очистить. В ней напишите:
Memo1.Text:=' ';
Создайте процедуру BitBtn2Click для кнопки Загрузить. В ней напишите:
If OpenDialog1.Execute Then
Memo1.Lines.LoadFromFile(OpenDialog1.FileName);
Здесь в поле компонента Memo1 загружается файл, имя которого будет выбрано в открывшемся окне Открыть.

7. Создайте процедуру BitBtn7Click для кнопки Сохранить как. В ней напишите:
If SaveDialog1.Execute Then Memo1.Lines.SaveToFile(SaveDialog1.FileName);
Здесь информация из поля компонента Memo1 сохраняется в файл, имя которого будет выбрано или задано в открывшемся окне Сохранить как.

8. Создайте процедуру BitBtn3Click для кнопки Вырезать. В ней напишите:
Memo1.CutToClipboard;
Здесь метод CutToClipboard помещает данные в буфер обмена, вырезая их из источника данных, в данном случае из поля компонента Memo1.

9. Создайте процедуру BitBtn5Click для кнопки Копировать. В ней напишите:
Memo1.CopyToClipboard;
Здесь метод CopyToClipboard помещает данные в буфер обмена, копируя их из источника данных, в данном случае из поля компонента Memo1.

10. Создайте процедуру BitBtn6Click для кнопки Вставить. В ней напишите:
Memo1.PasteFromClipboard;
Здесь метод PasteFromClipboard вставляет текстовый фрагмент из буфера обмена в поле редактирования компонента Memo1.

11. Далее необходимо продублировать все предыдущие действия в главном меню. Для этого достаточно в обработчиках пунктов меню создать ссылки на обработчики кнопок.

12. Создайте процедуру для пункта меню *Файл Новый*. Для этого откройте редактор создания меню, щелкните на пункте *Новый*. В инспекторе объектов перейдите на вкладку *Events* и раскройте список справа от *OnClick* (рис. 6*)*. Выберите в этом списке строку *BitBtn4Click:*

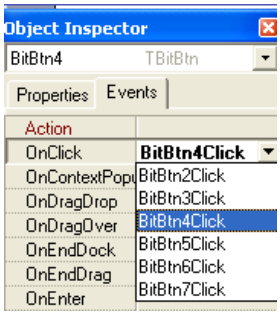


Рис. 6 Список процедур

13. Создайте процедуру для пункта меню *Правка Выделить все*. Для этого в редакторе создания меню дважды щелкните на пункте *Выделить все,* в открывшейся процедуре запишите:

Memo1.SelectAll;

14. Сохраните проект. Запустите проект на выполнение, проверьте работу кнопок и пунктов меню.

15. Самостоятельно создайте процедуры для остальных пунктов меню, кроме пункта *Сохранить*.

16. Самостоятельно создайте процедуру для кнопки *Сохранить* и пункта меню *Сохранить*. Сохраните проект и проверьте его работу.

Задание 2. Создать проект *Работа с меню приложения*.

Ход работы:

1. Создайте новый проект.
2. Разместите на форме компоненты в соответствии с рис. 7.

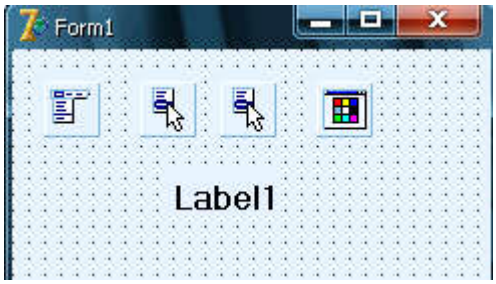


Рис. 7 Начальное расположение компонентов

Для главного меню создайте элементы, как на рис. 8 и 9.

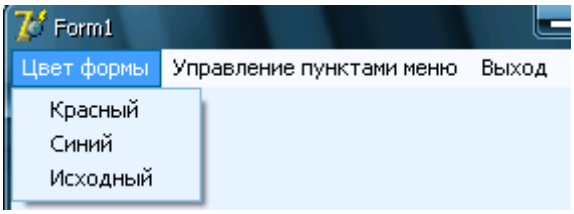


Рис. 8 Пункты меню *Цвет формы*

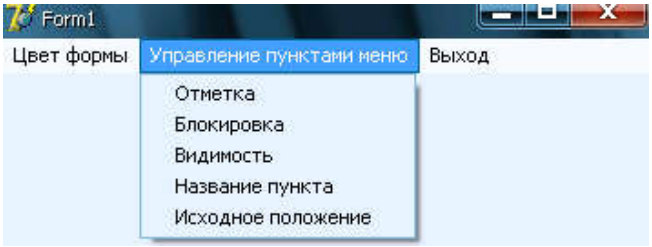


Рис. 9 Пункты меню *Управление пунктами меню*

Первое контекстное меню предназначено для формы, второе – для компонента Label1.

Создайте для этих меню элементы, как на рис. 10, 11, 12.

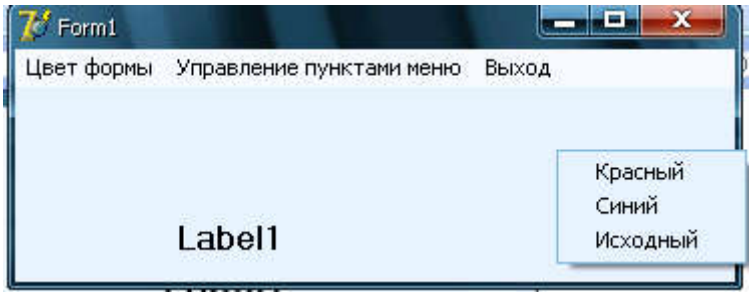


Рис. 10 Пункты первого контекстного меню

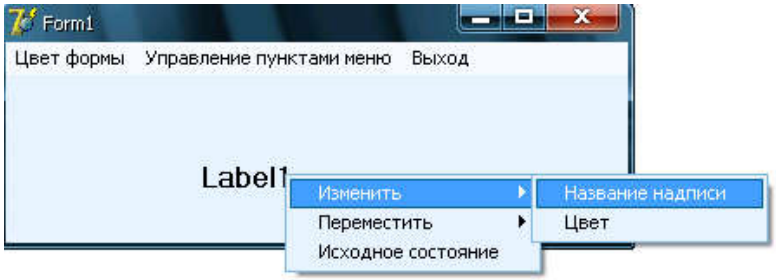


Рис. 11 Пункты второго контекстного меню, элемента «Изменить»

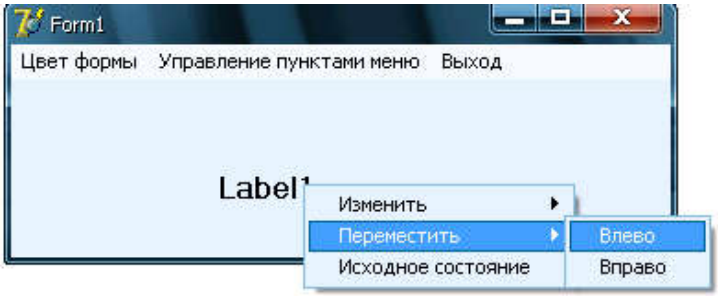


Рис. 12 Пункты второго контекстного меню, элемента «Переместить»

3. В разделе Var кода программы опишите следующие переменные:

FormColorMemory:longint;

LabelColorMemory:longint;

LabelLeftMemory:integer;

4. Создайте процедуру FormCreate. В этой процедуре будут устанавливаться начальные значения параметров и определяться соответствие контекстного меню и компонента. Напишите в процедуре:

form1.PopupMenu:=PopupMenu1;

Label1.PopupMenu:= PopupMenu2;

FormColorMemory:=Form1.Color;

LabelColorMemory:=Label1.Color;

LabelLeftMemory:=Label1.Left;

5. Создайте процедуру N2Click (щелчок по пункту меню Красный), в ней напишите:

Form1.Color:=clRed;

6. Аналогично создайте процедуру для щелчка по пункту меню Синий.

7. Создайте процедуру N4Click (щелчок по пункту меню Исходный), в ней напишите:

Form1.Color:= FormColorMemory;

8. Создайте процедуру N6Click (щелчок по пункту меню Отметка), в ней напишите:

If N6.Checked then N6.Checked:=False
else N6.Checked:=True;

9. Создайте процедуру N7Click (щелчок по пункту меню Блокировка), в ней напишите:

N7.Enabled:=False;

10. Создайте процедуру N8Click (щелчок по пункту меню Видимость), в ней напишите:

N8.Visible:=False;

11. Создайте процедуру N9Click (щелчок по пункту меню Название пункта), в ней напишите:

N9.Caption:='Время'+TimeToStr(Time);

12. Создайте процедуру N10Click (щелчок по пункту меню Исходное положение), в ней напишите:

N6.Checked:=False;
N7.Enabled:=True;
N8.Visible:=True;
N9.Caption:='Время';

13. Процедуру для щелчка по пункту меню Выход создайте самостоятельно.

14. Пункты первого контекстного меню дублируют элементы главного меню, поэтому процедуры нажатия пунктов этого меню будут имитировать команды главного меню. Создайте процедуру N12Click(щелчок по пункту первого контекстного меню Красный), в ней напишите:

N2.Click ;

Аналогично создайте процедуры для щелчка по пунктам первого контекстного меню Синий и Исходный.

15. Создайте процедуру N16Click (щелчок по пункту второго контекстного меню Название надписи), в ней напишите:

Label1.Caption:=InputBox('Изменение заголовка надписи','Новый заголовок',Label1.Caption);

16. Создайте процедуру N17Click (щелчок по пункту второго контекстного меню Цвет), в ней напишите:

If ColorDialog1.Execute then Label1.Color:=ColorDialog1.Color;

17. Создайте процедуру N19Click (щелчок по пункту второго контекстного меню Влево), в ней напишите:

Label1.Left:=Label1.Left-10;
If Label1.Left<10 then N19.Enabled:=False;
If not N20.Enabled then N20.Enabled:=True;

18. Создайте процедуру N20Click (щелчок по пункту второго контекстного меню Вправо), в ней напишите:

Label1.Left:=Label1.Left+10;
If Label1.Left>Form1.ClientWidth-Label1.Width-10
then N20.Enabled:=False;
if not N19.Enabled then N19.Enabled:=True;

19. Создайте процедуру N21Click (щелчок по пункту второго контекстного меню Исходное состояние), в ней напишите:

Label1.Color:= LabelColorMemory;
Label1.Left:= LabelLeftMemory;
N19.Enabled:= True;
N20.Enabled:=True;

20. Самостоятельно создайте новый пункт первого контекстного меню *Зеленый*, меняющий цвет формы на зеленый.
21. Самостоятельно создайте новый пункт второго контекстного меню Шрифт, меняющий размер шрифта компонента Label1.
22. Самостоятельно создайте новый пункт первого контекстного меню *Зеленый,* меняющий цвет формы на зеленый.
23. Самостоятельно создайте новый пункт второго контекстного меню *Шрифт,* меняющий размер шрифта компонента Label1.

Рекомендуемая литература:

1. Архангельский, А.Я. Программирование в Delphi для Windows / А.Я. Архангельский. – М.: Бином-Пресс, 2016. – 1248 с.
2. Осипов, Д.Л. Delphi. Профессиональное программирование / Д.Л. Осипов. – СПб.: БХВ-Петербург, 2018. – 1056 с.
3. Эйдлина, Г.М. Delphi. Программирование в примерах и задачах. Практикум. Учебное пособие / Г.М. Эйдлина, К.А. Милорадов. – М.: РИОР, 2018. – 138 с.

Практическая работа №13
Сложные элементы интерфейса

Цели работы:

- закрепление полученных знаний по темам «Элементы языка Delphi», «Условный оператор», «Циклы»;
- формирование умения создавать сложные интерфейсы;

– формирование умения создавать логически правильные и эффективные программы.

Краткие теоретические сведения

Компонент Tabbednotebook (многостраничный блокнот) находится на странице Win3.1. Он состоит из нескольких страниц (вкладок). Компактное расположение страниц блокнота позволяет удобно размещать и группировать другие элементы управления.

Компонент Calendar находится на странице Samples и представляет собой элемент управления для выбора даты. Внешне он напоминает обычный календарь на один месяц.

На странице Win32 расположены компоненты, аналогичные компонентам Tabbednotebook и Calendar. Это компонент PageControl (многостраничный блокнот), компоненты MonthCalendar (выбор даты) и DteTimePicker (выбор даты и времени).

Задание. Создать проект *Многостраничный блокнот* для работы с календарем и телефонной книгой.

Ход работы:

1. Загрузите Delphi. Создайте новый проект.
2. Разместите на форме компонент Tabbednotebook (многостраничный блокнот) со страницы Win3.1 (рис. 1).

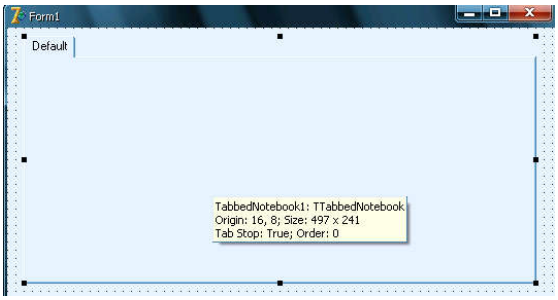


Рис. 1 Компонент Tabbednotebook

Щелкните в Инспекторе объектов по трем точкам, расположенным правее свойства Pages этого компонента. Откроется редактор создания страниц блокнота. Нажмите кнопку Edit и введите название первой страницы Календарь. Нажмите ОК (рис. 2).

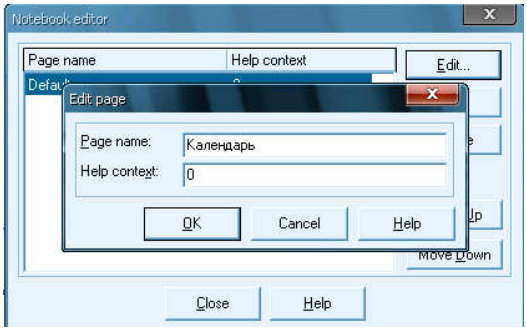


Рис. 2 Создание страницы *Календарь*

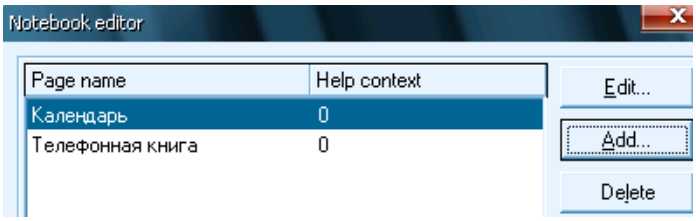


Рис. 3 Страницы календаря

С помощью кнопки Add создайте еще одну страницу *Телефонная книга*. Закройте редактор с помощью кнопки Close.

3. Перейдите на страницу Календарь. Для этого, если у Вас активна страница Телефонная книга, вызовите контекстное меню компонента Tabbednotebook1 и выберите команду PreviousPage. Если будет необходимо перейти обратно, со страницы Календарь на страницу Телефонная книга, то нужно выбрать команду NextPage контекстного меню (рис. 4).

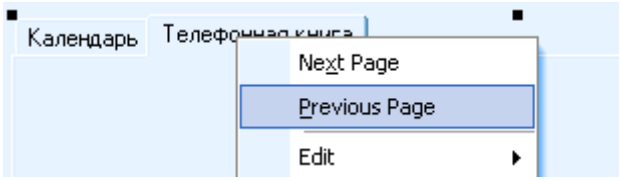


Рис. 4 Переход по страницам

Положите на эту страницу компоненты как на рисунке. Внизу размещен компонент Calendar со страницы Samples (рис. 5).

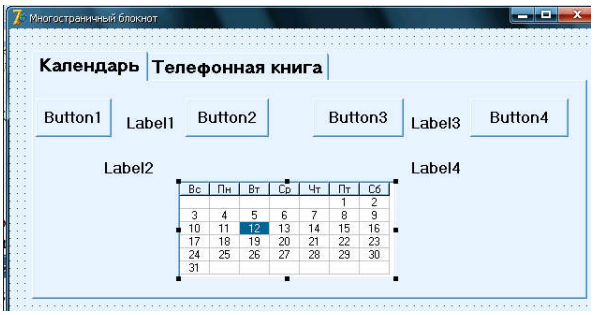


Рис. 5 Размещение компонентов на странице *Календарь*

Измените надписи у компонентов Button1-4, Label1, Label3. Компоненты Label2, Label4 оставьте пустыми (рис. 6) .

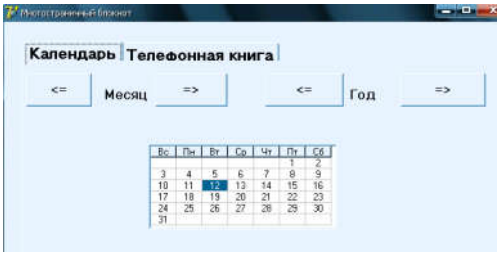


Рис. 6 Итоговый вид страницы *Календарь*

4. Создайте процедуру Button1Click, в ней напишите:
Calendar1.month:=(calendar1.month-1) mod 13;

Label2.caption:=inttostr(calendar1.month);

5. Создайте процедуру Button2Click, в ней напишите:

Calendar1.month:=(calendar1.month+1) mod 13;

Label2.caption:=inttostr(calendar1.month);

6. Создайте процедуру Button3Click, в ней напишите:

Calendar1.year:= Calendar1.year -1;

Label4.caption:= inttostr(calendar1.year);

7. Создайте процедуру Button4Click, в ней напишите:

Calendar1.year:= Calendar1.year +1;

Label4.caption:= inttostr(calendar1.year);

8. Сохраните проект, запустите его на выполнение. Проверьте работу кнопок на странице *Календарь*: кнопки Button1 и Button2 должны «листать» месяцы на календаре, номер выбранного месяца должен отображаться в поле компонента Label2. Кнопки Button3 и Button4 должны «листать» года на календаре, номер выбранного года должен отображаться в поле компонента Label4.

9. Перейдите на страницу *Телефонная книга*, расположите на ней компоненты, как на рис. 7.

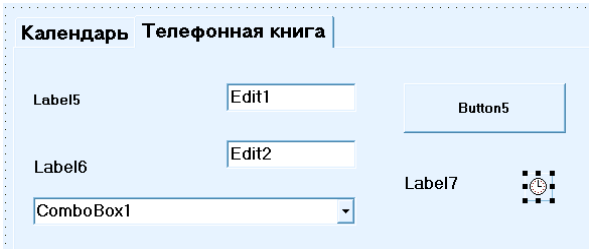


Рис. 7 Расположение компонентов на странице *Телефонная книга*

Измените надписи компонентов как на рис. 8.

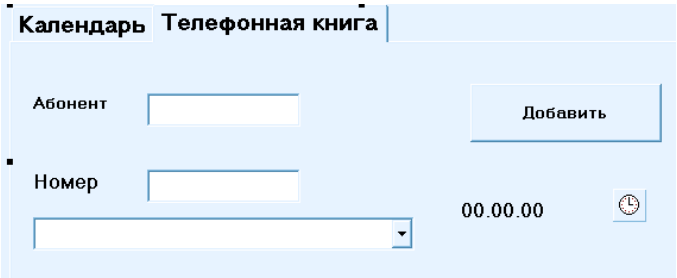


Рис. 8 Итоговый вид страницы *Телефонная книга*

10. Создайте процедуру Button5Click (нажатие кнопки *Добавить*), в ней напишите:

ComboBox1.Items.Add(Edit1.Text+' '+Edit2.Text);

Edit1Ttext:=' ';

Edit2.Text:=' ';

11. Создайте процедуру Timer1Timer, в ней напишите:

Label7.Caption:=TimeToStr(Now);

12. Сохраните проект, запустите его на выполнение. Проверьте работу элементов на странице *Телефонная книга:* сначала необходимо ввести фамилию абонента и номер его телефона, затем нужно нажать кнопку *Добавить*. Эти данные должны появиться в поле компонента ComboBox1 (рис. 9). В поле компонента Label7 должно отображаться текущее время. Сразу после нажатия кнопки *Добавить* поля ввода данных абонента должны очищаться.

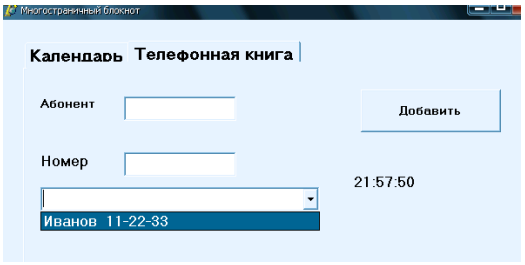


Рис. 9 Добавление абонента

13. Самостоятельно выполните для страницы *Календарь* отображение названия месяца, а не его номера.

14. Самостоятельно выполните сохранение данных телефонной книжки в файл. Для этого добавьте кнопку *Сохранить*.

Рекомендуемая литература:

1. Архангельский, А.Я. Программирование в Delphi для Windows / А.Я. Архангельский. – М.: Бином-Пресс, 2016. – 1248 с.
2. Осипов, Д.Л. Delphi. Профессиональное программирование / Д.Л. Осипов. – СПб.: БХВ-Петербург, 2018. – 1056 с.
3. Эйдлина, Г.М. Delphi. Программирование в примерах и задачах. Практикум. Учебное пособие / Г.М. Эйдлина, К.А. Милорадов. – М.: РИОР, 2018. – 138 с.

Практическая работа № 14
Разработка графики

Цели работы:

- закрепление полученных знаний по темам «Элементы языка Delphi», «Условный оператор», «Циклы»;
- формирование умения применения графических инструментов и примитивов;
- формирование умения применения методов Polygon и Invalidate;
- формирование умения создания проекта с несколькими формами;
- формирование умения создавать логически правильные и эффективные программы.

Краткие теоретические сведения

Pen (карандаш) – графический инструмент, предназначенный для рисования линий и очертаний геометрических фигур. По умолчанию используется карандаш, который рисует сплошные линии черного цвета толщиной в один пиксель. Для изменения данных настроек используются свойства карандаша.

Brush (кисть) – графический инструмент, предназначенный для заполнения внутренней области геометрических фигур различными цветами и узорами. По умолчанию замкнутые геометрические фигуры закрашиваются в белый цвет.

Графическое изображение – это результат объединения или пересечения простейших элементов, которые называются графическими примитивами. Это дуги, линии, эллипс, прямоугольник, сектор.

Вывод данных примитивов осуществляется с помощью специальных методов и свойств объекта Canvas:

– *Rectangle(x1, y1, x2, y2)* – вычерчивает прямоугольник. Параметры (x1, y1) и (x2, y2) задают координаты левого верхнего и правого нижнего углов.

– *Ellipse(x1, y1, x2, y2)* – рисование эллипса с заливкой текущим цветом кисти. Эллипс ограничивается прямоугольником, который определен координатами его левого верхнего угла (x1,y1) и правого нижнего угла (x2,y2).

– *Arc(x1, y1, x2, y2, x3, y3, x4, y4)* – вычерчивает дугу. Параметры x1, y1, x2, y2 определяют эллипс, из которого вырезается дуга, параметры x3, y3, x4, y4 – координаты концов дуги. Дуга вычерчивается против часовой стрелки от точки (x3, y3) к точке (x4, y4).

– *Chord(x1, y1, x2, y2, x3, y3, x4, y4)* – вычерчивает хорду, ограниченную точками пересечения эллипса, вписанного в прямоугольник с координатами левого верхнего угла (x1,y1) и правого нижнего угла (x2,y2), и линии, проходящей через точки (x3,y3) и (x4,y4).

– *RoundRect (x1, y1, x2, y2, x3, y3)* –вычерчивает прямоугольник со скругленными углами. Параметры x1, y1, x2, y2 задают координаты левого верхнего и правого нижнего углов. Углы представляют собой участки границы эллипса, вписанного в прямоугольник шириной x3 пикселя и высотой y3 пикселя.

– *MoveTo(x, y)* – перемещает указатель текущей точки в точку с указанными координатами.

– *LineTo(x, y)* – вычерчивает линию из текущей точки в точку с указанными координатами.

– *TextOut(x, y, s)* – выводит строку S от точки с координатами (x, y).

Компонент Shape расположен на странице Additional и предназначен для вывода простейших геометрических фигур на поверхности формы.

Метод Polygon предназначен для рисования графических объектов, представляющих собой сложные многоугольники.

Метод Invalidate используется для принудительной перерисовки поверхности элемента управления.

Задание. Создайте проект, с помощью которого можно изучить основные дорожные знаки.

Ход работы:

1. Загрузите Delphi. Создайте новый проект.
2. Разместите на форме компоненты в соответствии с рис. 1.

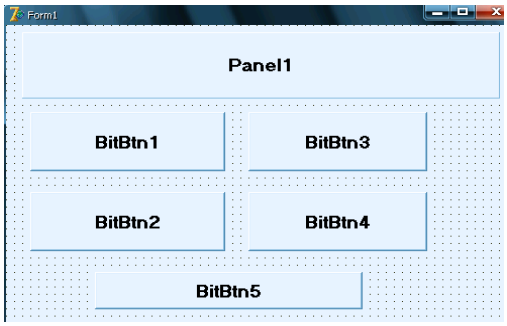


Рис. 1 Начальное расположение компонентов

Измените свойство Caption каждого компонента и формы (рис. 2).

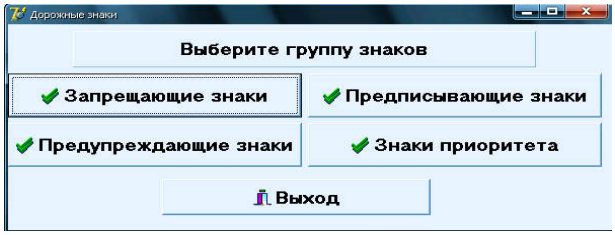


Рис. 2 Итоговый вид формы

3. После нажатия каждой из кнопок должна открываться новая форма, на которой должны быть нарисованы дорожные знаки выбранной категории.

Таблица 1 Команды вызова форм

Имя процедуры	**Команда**
BitBtn1Click	Form2.Show;
BitBtn2Click	Form3.Show;
BitBtn3Click	Form4.Show;
BitBtn4Click	Form5.Show;

4. Сохраните проект. Запустите проект на выполнение, проверьте работу каждой кнопки.

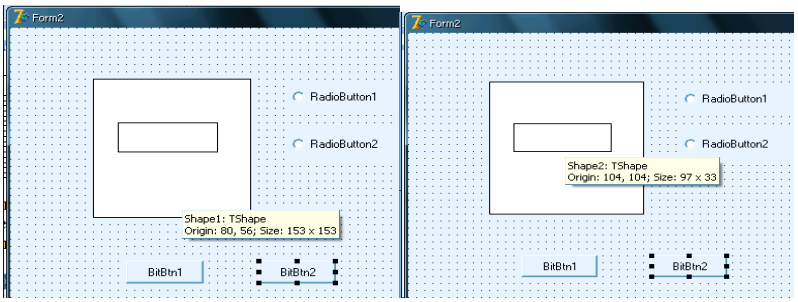


Рис. 3 Расположение компонентов Shape

5. Измените надписи у компонентов и формы как на рис. 4.

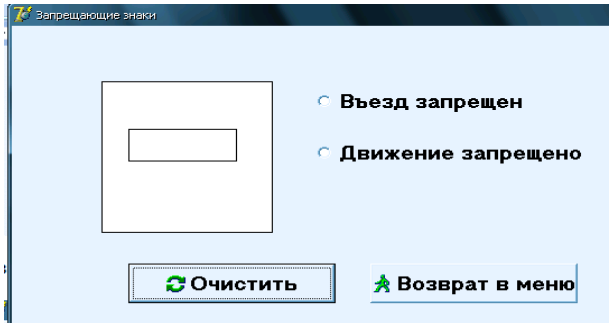


Рис. 4 Вид формы «Запрещающие знаки»

1) В разделе *uses* в конце списка подключаемых модулей допишите *unit1* (рис. 5)

```
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Buttons, ExtCtrls, unit1;
```

Рис. 5 Модуль unit1

Создайте процедуры для создания знаков:

Изображение дорожного знака представлено на рис. 6.

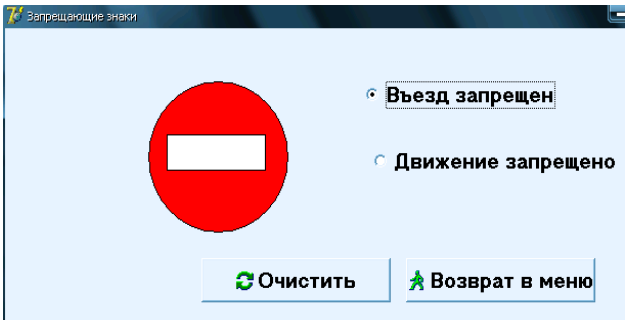


Рис. 6 Знак «Въезд запрещен»

Изображение дорожного знака представлено на рис. 7.

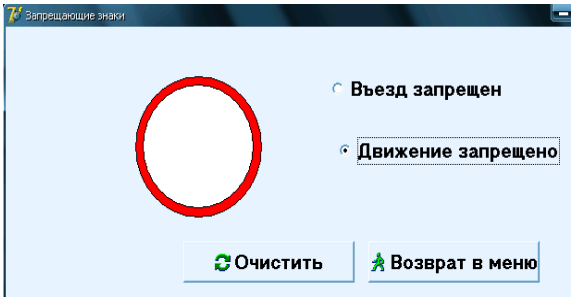


Рис. 7 Знак «Движение запрещено»

6. Перейдите на третью форму, расположите на ней компоненты, как на рис. 8.

Рис. 8 Начальное расположение компонентов на третьей форме

Левее компонентов RadioButton оставьте пустое место для прорисовки знаков. Измените надписи у компонентов и формы (рис. 9).

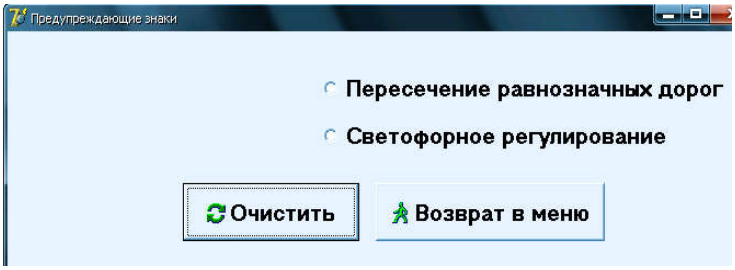


Рис. 9 Итоговый вид третьей формы

7. В разделе *uses* в конце списка подключаемых модулей допишите *unit1*.

8. Самостоятельно создайте процедуру BitBtn2Click для кнопки *Возврат в меню.*

9. Создайте процедуру RadioButton1Click. В ней запишите:

```
var
ris: array[0..2] of TPoint;
begin
ris[0]:=point(150,50);
 ris[1]:=point(280,250);
 ris[2]:=point(20,250);
 canvas.Brush.Color:=clred;
 canvas.Polygon(ris);
```

Создается массив точек *ris*. Метод *Polygon* соединяет эти точки поочередно, последнюю точку соединяет с первой и полученную фигуру закрашивает цветом заливки, в данном примере – красным.

Аналогично постройте белый треугольник. Для этого используйте тот же массив *ris* (описывать его еще раз в разделе var не нужно!!!). Полученные треугольники представлены на рис. 10.

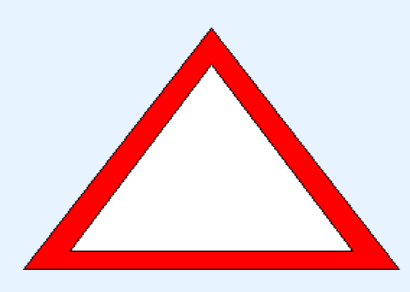


Рис. 10 Красный и белый треугольники

10. Теперь необходимо добавить в центр белого треугольника две перекрещенные черные линии. Постройте их с помощью графического инструмента Pen(карандаш), регулируя его ширину (свойство Width). Для построения первой линии напишите:

canvas.Pen.Color:=clblack;
canvas.Pen.Width:=20;
canvas.MoveTo(100,210);
canvas.LineTo(175,150);

Полученный результат представлен на рис. 11.

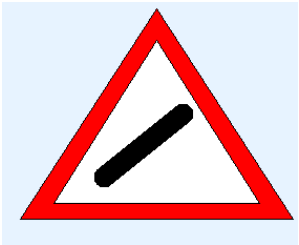


Рис. 11 Первая линия

Самостоятельно постройте вторую линию (рис. 12).

Рис. 12 Вторая линия

11. Создайте процедуру RadioButton2Click. В ней самостоятельно запишите команды построения двух треугольников – красного и белого (рис. 10)

Внутри белого треугольника должны быть три маленькие окружности: красная, желтая, зеленая. Для построения красной окружности напишите:

canvas.Brush.Color:=clred;
canvas.Ellipse(135,105,165,135);

Аналогично постройте две другие окружности. В итоге знак должен иметь вид как на рис. 13.

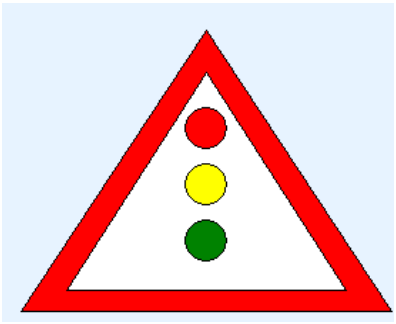


Рис. 13 Знак «Светофорное регулирование»

12. Создайте процедуру BitBtn1Click для кнопки Очистить. В ней напишите:

form3.Invalidate;

Этот метод позволяет очищать поверхность рисования.

13. Сохраните проект. Запустите проект на выполнение, проверьте его работу. Возможно, что при повторном построении знака *Светофорное регулирование* вокруг него будет прорисовываться черная окантовка. Это объясняется тем, что для знака *Пересечение равнозначных дорог* была изменена ширина карандаша. Чтобы исключить эту ситуацию, поставьте в самом начале процедуры RadioButton2Click после слова *begin* команду:

canvas.Pen.Width:=1;

14. Перейдите на четвертую форму, расположите на ней компоненты, как на рис. 14.

Рис. 14 Начальное расположение компонентов на четвертой форме

Измените надписи у компонентов и формы (рис. 15).

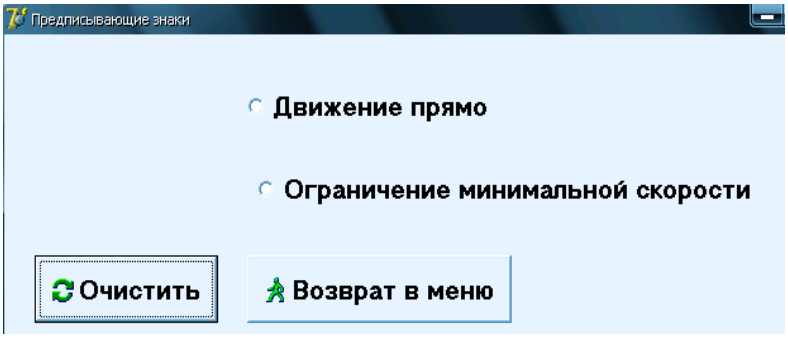


Рис. 15 Итоговый вид четвертой формы

15. В разделе *uses* в конце списка подключаемых модулей допишите *unit1*.

16. Самостоятельно создайте процедуры для кнопок *Очистить* и *Возврат в меню*.

17. Знак *Движение прямо* должен выглядеть как на рис. 16 (белая стрелка на синем круге).

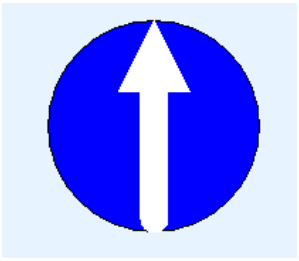


Рис. 16 Знак «Движение прямо»

Самостоятельно постройте окружность, выполните заливку синим цветом. Отрезок (до стрелки) постройте с помощью линии, регулируя её ширину. Стрелку прорисуйте, используя метод *Polygon*.

18. Постройте знак *Ограничение минимальной скорости* (рис. 17).

Рис. 17 Знак «Ограничение минимальной скорости»

Окружность постройте самостоятельно. Для того, чтобы на поверхности рисования отображался текст, нужно использовать графический примитив *TextOut*. Напишите в процедуре RadioButton2Click:

```
form4.Font.Color:= clwhite;
form4.Font.Height:=50;
canvas.TextOut(100,100,'50');
```

Чтобы число *50* было более крупным и выводилось белым цветом, необходимо изменить настройки формы.

19. Сохраните проект. Запустите проект на выполнение, проверьте его работу.

20. Перейдите на пятую форму, расположите на ней компоненты, как на рис. 18.

Рис. 18 Начальное расположение компонентов на пятой форме

Измените надписи у компонентов и формы (рис. 19).

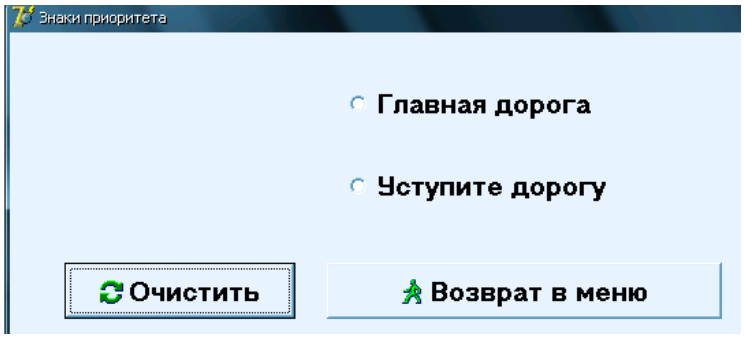


Рис. 19 Итоговый вид пятой формы

21. В разделе *uses* в конце списка подключаемых модулей допишите *unit1*.

22. Самостоятельно создайте процедуры для кнопок *Очистить* и *Возврат в меню.*

23. Знак *Главная дорога* должен выглядеть как на рис. 20 (желтый ромб внутри белого ромба).

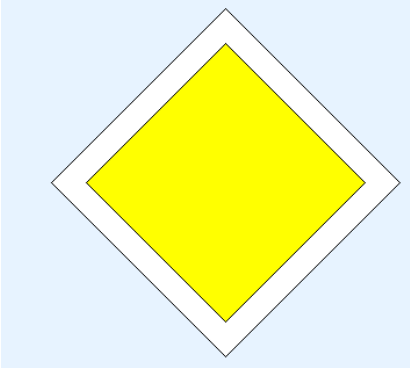


Рис. 20 Знак «Главная дорога»

Постройте этот знак, используя метод *Polygon.*

24. Знак *Уступите дорогу* должен выглядеть как на рис. 21 (белый перевернутый треугольник внутри красного).

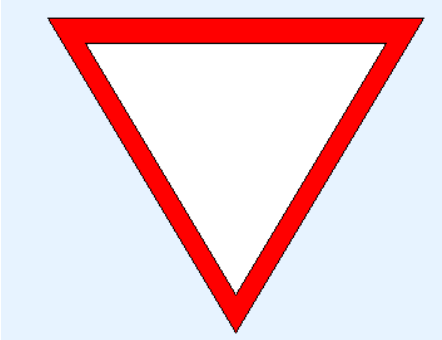


Рис. 21 Знак «Уступите дорогу»

Постройте этот знак, используя метод *Polygon.*

25. Сохраните проект. Запустите проект на выполнение, проверьте его работу.

26. Самостоятельно, на пятой форме создайте знак *Движение без остановки запрещено*, который выглядит как на рис. 22 (красный восьмиугольник, на нем белые буквы STOP).

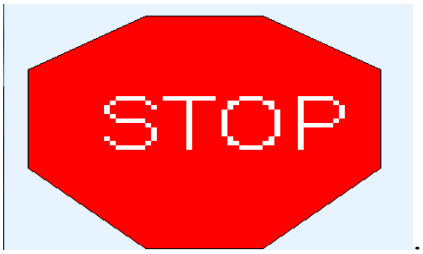


Рис. 22 Знак «Движение без остановки запрещено»

Рекомендуемая литература:

1. Архангельский, А.Я. Программирование в Delphi для Windows / А.Я. Архангельский. – М.: Бином-Пресс, 2016. – 1248 с.
2. Осипов, Д.Л. Delphi. Профессиональное программирование / Д.Л. Осипов. – СПб.: БХВ-Петербург, 2018. – 1056 с.
3. Эйдлина, Г.М. Delphi. Программирование в примерах и задачах. Практикум. Учебное пособие / Г.М. Эйдлина, К.А. Милорадов. – М.: РИОР, 2018. – 138 с.

Практическая работа № 15
Разработка анимации

Цели работы:

- закрепление полученных знаний по темам «Элементы языка Delphi», «Условный оператор», «Циклы»;
- формирование умения использования битовых образов;
- формирование умения применения методов Draw, CopyRect, Bounds, метода базовой точки;
- формирование умения создания проекта с несколькими формами;
- формирование умения создавать логически правильные и эффективные программы.

Краткие теоретические сведения

Битовый образ является объектом типа TBitMap. Битовый образ представляет собой находящуюся в памяти компьютера невидимую графическую поверхность, на которой программа может сформировать изображение.

Содержимое битового образа может быть выведено на поверхность формы или области вывода иллюстрации (Image). Загрузить в битовый образ

нужную картинку можно при помощи метода LoadFromFile, указав в качестве параметра имя BMP-файла, в котором находится нужная иллюстрация.

Вывести содержимое битового образа (картинку) на поверхность формы или области вывода иллюстрации можно путем применения метода Draw к соответствующему свойству поверхности (Canvas).

Сохранение копии фона выполняется при помощи метода CopyRect, который позволяет выполнить копирование прямоугольного фрагмента одного битового образа в другой. Объект, к которому применяется метод CopyRect, является приемником копии битового образа. В качестве параметров методу передаются координаты и размер области, куда должно быть выполнено копирование, поверхность, откуда должно быть выполнено копирование, а также положение и размер копируемой области.

Заполнить поля структуры TRect можно при помощи функции Bounds, инструкция обращения к которой в общем виде выглядит так:

Bounds(x,y,Width,Height)

где:

- x и y — координаты левого верхнего угла области;
- width и Height — ширина и высота области.

Задание. Создайте проект, в котором на фоне города летит самолет.

Ход работы:

1. Загрузите Delphi. Создайте новый проект.
2. В *Paint* нарисуйте самолет (рис. 1), сохраните это изображение как *aplane.bmp.* Создайте еще один рисунок, который будет создавать фон для летящего самолета (рис. 2). Сохраните его как *factory.bmp.*

Рис. 1 Самолет

Рис. 2 Фон

3. Разместите на форме компоненты Timer и Image. Разработайте анимацию.

4. В разделе var опишите следующие переменные:

Back, Bitmap, Buf : TBitMap; // фон, картинка, буфер

BackRct : TRect; // область фона, которая должна быть восстановлена из буфера

BufRet: Trect; // область буфера, которая используется для восстановления фона

5. Создайте процедуру для формы FormActivate. В этой процедуре сначала нужно создать три битовых образа: фон, самолет и буфер. Для этого напишите:

Back := TBitmap.Create; // создается фон

Bitmap := TBitmap.Create; // создается самолет

Buf := TBitmap.Create; // создается буфер

6. Далее необходимо в эти битовые образы загрузить и вывести изображение самолета и фон. Для этого напишите:

Back.LoadFromFile('factory.bmp'); // загрузка фона в битовый образ *Back*

Form1.Image1.canvas.Draw(0,0,Back); //размещение фона в поле компонента *Image1*.

В этой команде применен метод Draw к свойству canvas поверхности компонента *Image*.

Параметры 0,0 определяют положение левого верхнего угла картинки на поверхности компонента.

7. Далее необходимо загрузить картинку, которая будет двигаться. Самостоятельно выполните загрузку картинки *aplane.bmp* в битовый образ *Bitmap. Команду размещения в поле* компонента *Image1 выполнять не нужно!*

8. Т.к. самолет имеет не прямоугольное очертание, то при загрузке картинки на фон вокруг самолета будет виден белый прямоугольник – это тот фон, на котором помещалось изображение в *Paint.* Чтобы этот «лишний» фон был не виден, нужно сделать его прозрачным. Для этого используется свойство *Transparent* . Напишите:

Bitmap.Transparent := True;

bitmap. TransparentColor := bitmap.canvas.pixels[1,1];

9. Для того чтобы после перемещения самолета фон оставался прежним, а не «смазывался» самолетом, прежнее состояние фона необходимо сохранять. Для этого был создан буфер *Buf* для сохранения копии области фон. Сначала необходимо определить высоту и ширину сохраняемой области. Опишите в разделе *var* две целые переменные W и H, а в процедуре FormActivate запишите:

W:= bitmap.Width;

H:= bitmap.Height;

Buf.Width:= W;

Buf.Height:=H;

Buf.Palette:=Back.Palette; //обеспечение соответствия палитр

Buf.Canvas.CopyMode:=cmSrcCopy;

10. Далее необходимо определить область буфера, которая будет использоваться для восстановления фона:

BufRet:=Bounds(0,0,W,H); // начальное положение картинки

x := -W; y := 20;

BackRct:=Bounds(x,y,W,H); // определим сохраняемую область фона

// и сохраним ее

Buf.Canvas.CopyRect(BufRet,Back.Canvas,BackRct);

На этом процедура FormActivate закончена.

11. Создайте процедуру *Timer1Timer* и запишите в ней следующие команды:

// восстановлением фона (из буфера) удалим рисунок

Forml.image1.canvas.Draw(x,y,Buf);

x:=x+2;

if x>form1.Image1.Width then x:=-W;

// определим сохраняемую область фона

BackRct:=Bounds(x,y,W,H);

// сохраним ее копию

Buf.Canvas.CopyRect(BufRet,Back.Canvas,BackRct);

// выведем рисунок

Forml.image1.canvas.Draw(x,y,bitmap);

12. Создайте процедуру *FormClose*, в которой необходимо освободить память, выделенную для хранения битовых образов:

Back.Free;

bitmap.Free;

Buf.Free;

13. Сохраните проект, прокомпилируйте, запустите на выполнение.

14. Самостоятельно дополните проект:

a) В *Paint* нарисуйте машину, организуйте ее движение внизу фоновой картинки параллельно самолету. Для этого измените фон, дорисовав внизу дорогу;

b) Организуйте движение машины навстречу самолету.

Рекомендуемая литература:

1. Архангельский, А.Я. Программирование в Delphi для Windows / А.Я. Архангельский. – М.: Бином-Пресс, 2016. – 1248 с.
2. Осипов, Д.Л. Delphi. Профессиональное программирование / Д.Л. Осипов. – СПб.: БХВ-Петербург, 2018. – 1056 с.
3. Эйдлина, Г.М. Delphi. Программирование в примерах и задачах. Практикум. Учебное пособие / Г.М. Эйдлина, К.А. Милорадов. – М.: РИОР, 2018. – 138 с.