Введение

На курсовое проектирование была поставлена задача, разработать программу на тему: «Знаки дорожного движения».

Цель курсового проекта заключается в знакомстве пользователя со знаками дорожного движения и их разновидностями.

Создаваемая программа будет рассчитана на пользователей, которые интересуются ПДД, так и ученикам автошкол, которые собираются сдавать экзамены на права.

Приведем краткое описание разделов пояснительной записки.

Первый раздел носит название “Анализ задачи”. В нем вы сможете ознакомиться с постановкой задачи, которая включает в себя: исследование предметной области поставленной задачи, определение ее организационно-экономической сущности. Также в этом разделе вы сможете узнать о том, как данная задача решается в настоящее время. Все входные и выходные данные тоже будут описаны в первом разделе. В подразделе “Инструменты разработки” будет рассмотрена среда, в которой создается данный курсовой проект. Здесь также будут установлены минимальные и оптимальные требования к аппаратным характеристикам, обеспечивающим правильное функционирование поставленной задачи.

В разделе “Проектирование задачи” будут рассмотрены основные аспекты разработки программного продукта. Здесь можно будет узнать об организации данных в контексте среды разработки. В данном разделе будет четко описан пользовательский интерфейс, составлены алгоритмы процесса обработки информации, описана разработка системы справочной информации.

“Реализация задачи” – это третий раздел пояснительной записки, в котором описываются все элементы и объекты, которые будут использованы при реализации данного приложения. В этом разделе будут четко описаны функции пользователя и их структура. Здесь можно будет найти таблицу, в которой будет представлена полная аннотация файлов используемых в данном проекте.

Четвертый раздел – “Тестирование”. В нем будет описано полное и функциональное тестирование данной программы, т.е. будет оттестирован каждый пункт меню, каждая операция, которая выполняется приложением. Будут смоделированы все возможные действия пользователя при работе с программой, начиная от запуска до выхода.

В разделе “Применение” будет описано назначение, область применения, среда функционирования курсовой программы. Также в нем будет описано использование справочной системы.

“Заключение” будет содержать краткую формулировку задачи, результаты проделанной работы, описание использованных методов и средств, описание степени автоматизации процессов на различных этапах разработки.

В “Списке используемых источников” будет приведен список используемой литературы, нормативно- техническую и другую документацию.

В приложениях к пояснительной записке будет приведен листинг программы с необходимыми комментариями.

Схема работы системы будет представлена в графической части.

1. Анализ задачи

1.1 Постановка задачи

Наименование задачи: Разработка приложения «Знаки дорожного движения».

Цель разработки: создание приложения «Знаки дорожного движения» заключается в предоставлении пользователям информации о различных дорожных знаках и их значениях.

Назначение: помочь водителям и пешеходам лучше понять и запомнить знаки, используемые на дорогах.

Предметная область: будет изучена предметная область на основе такого приложения, как «Дорожные знаки».

Периодичность использования: зависит от нужд потребителя, может использоваться ежедневно.

Источники и способы получения данных: В разрабатываемой программе будут использоваться три вида данных. К входной информации можно отнести вводимые пользователем значения, например ответы на вопросы при прохождения теста. К выходной – результат прохождения теста. Постоянной информацией в проекте будут являться текстовые файлы, картинки и др.
Программный продукт предоставляет функционал для следующего ряд пользователей: администратор – организовывает работу программой, гость

Разрабатываемый программный продукт позволит выполнить следующие действия:

- просмотреть информацию о знаках;

- пройти тест;

- сыграть в квест.

1.2 Инструменты разработки

Для разработки данного проекта будет выбрана среда Delphi 11, так как
это самое удобная и доступная среда разработки на данный момент. Delphi 11-язык –программирования, относящийся к классу RAD- (Rapid Application Development – «Средство быстрой разработки приложений») средств CASE –технологии. Delphi 11 сделал разработку приложений дляWindows быстрым иприятным процессом. Теперь разрабатывать сложные и интересные проекты можно только одним человеком, использующим Delphi 11

Интерфейс Windows обеспечивает полное перенесение CASE-технологии в интегральную систему поддержки работ по созданию прикладной системы на всех фазах жизненного цикла работы и проектирования системы.

Delphi 11 обладает широким набором возможностей, начиная от проектировщика форм и заканчивая поддержкой всех форматов популярных баз данных. Среда устраняет необходимость программировать такие компоненты Windows общего назначения, как метки, программы и даже диалоговые панели. Работая в Windows, можно видеть одинаковые «объекты» во многих разнообразных приложениях. Диалоговые панели (например, Choose File и Save File) являются примерами многократно-используемых компонентов, встроенных непосредственно в Delphi 11, который позволяет приспособить эти компоненты к имеющийся задаче, чтобы они работали именно так, как требуется создаваемому приложению. Также здесь имеются предварительно-определенные визуальные и не визуальные объекты.

Три основные части разработки интерфейса следующие: проектирование панели, проектирование диалога и представление окон. Для общего пользовательского доступа также должны учитываться условия применения архитектуры прикладных систем.

Сегодня появилась реальная возможность с помощью моделирования на современных многофункциональных средствах обработки и отображения информации таких как Delphi 11 конкретизировать тип и характеристики используемых информационных моделей, выявить основные особенности будущей деятельности операторов, сформулировать требования к параметрам аппаратно-программных средств интерфейса взаимодействия и т.д. Delphi 11 позволяет создать различные виды программ: консольные приложения, оконные приложения, приложения для работы с Интернетом и базами данных. То есть, Delphi 11 является не только средствами для работы с языком программирования Паскаль, но дополнительные инструменты, призванные для максимального упрощения и ускорения создание приложений.

К дополнительным инструментам можно отнести визуальный редактор форм, благодаря которому можно с легкостью создать полноценную программу, и другие визуальные составляющие разработки программного обеспечения. С Delphi вам не нужно вручную просчитывать расположение каждого элемента интерфейса пользователя, поэтому при разработке программы значительно экономится время.

Выгоды от проектирования в среде Windows с помощью Delphi 11:

- устраняется необходимость в повторном вводе данных;
- обеспечивается согласованность проекта и его реализации;

- увеличивается производительность разработки и переносимость программ.

Ни одно серьезное программное обеспечение не обходится без модуля справочной информации и руководства пользователя. Это придает программе законченный вид и показывает заботу о пользователе. DrExplain – легкий в использовании и функциональный инструмент, упрощающий создание справочных файлов Windows, печать справочных руководств и документации в целом. Программа имеет интуитивно понятный интерфейс. Все созданные проекты можно сохранить в различных форматах:HTML Help, Winhelp и MS Help 2.0 / Visual Studio Help, Browser-based Help, PDF и Word RTF, а также печатной документации при использовании одного и того же проекта. В основном окне программы содержатся оглавление (в виде древовидного списка) и текстовый редактор. Это дает возможность легко ориентироваться в оглавлении, редактировать или перемещать разделы справки без каких-либо проблем. Также утилита позволяет конвертировать help-файлы из одного формата в другой. Кроме приложений для работы с текстом в данном продукте содержатся утилиты для создания скриншотов и редактирования графических файлов.

Для создания инсталлятора будет использоваться мощное и удобное средство - Smart Install Maker. Программа обладает удобным и интуитивно понятным интерфейсом, а также полным набором необходимых функций для создания профессиональных инсталляторов с минимальным размером, высокой степенью сжатия файлов и приятным интерфейсом.

Помимо стандартного минимума, Smart Install Maker позволяет редактировать системный реестр и INI-файлы, создавать программные ярлыки, запускать ассоциируемые и исполняемые файлы, регистрировать новые шрифты и ActiveX компоненты, отображать тексты информации и лицензионного соглашения. Также, с помощью этой утилиты, можно создать мультиязыковые инсталляторы с поддержкой более 20-ти популярных языков мира.

Microsoft Word 2016 – редактор текста для написания документации.

Разработка ведется на ноутбуке Lenovo . У данного ноутбука следующие параметры:

- Процессор: Intel(R) Core(TM) i5-6300U 2.50 GHz;
- ОЗУ: 8Gb;
- Память: SSD 256Gb;
- ОС – Windows 11.

Как видно разрабатываемое приложение не очень требовательно к аппаратным ресурсам, что, является большим плюсом.

1.3 Требования к приложению

На этапе исследования предметной области был установлен целый ряд требований, которые предъявляются к разрабатываемой программе.

По этой причине особенно актуальной становится разработка программного электронного продукта, который способствовал бы и давал возможность вспомнить и повторить информацию из теоретического курса. Чтобы эта информация лучше усваивалась необходимо подобрать правильный интерфейс.

Требования к интерфейсу: в связи с частым использованием программы она должна быть с приятной цветовой гаммой и понятной для пользователя. Следовательно, каждое окно должно иметь ясную визуальную иерархию своих элементов. Фрагменты текста должны располагаться на экране так, чтобы пользователя было просто и понятно принимать информацию.

Пользователь не должен испытывать какого-либо дискомфорта в плане восприятия информация, отображённой на экране. Объекты (рисунки и символы) не должны быть слишком мелкие. Все окна приложения по возможности должны помещаться на экран полностью, так как использование в процессе работы полос прокруток достаточно неудобно.

На одной форме нельзя допускать избытка и нагромождения данных. Формы должны быть эффектно оформлены согласно тематике разрабатываемого проекта.

Требования к надежности: специальных требований к надежности не предъявляется. Время восстановления после отказа, вызванного сбоем электропитания технических средств (иными внешними факторами), не фатальным сбое операционной системы, не должно превышать 30-ти минут при условии соблюдения условий эксплуатации технических и программных средств.

Время восстановления после отказа, вызванного неисправностью технических средств, фатальным сбоем операционной системы, не должно превышать времени, требуемого на устранение неисправностей технических средств и переустановки программных средств.

Отказы программы вследствие некорректных действий пользователя при взаимодействии с программой через графический интерфейс не должны влиять на конечный результат.

2. Проектирование задачи

2.1 Организация данных

Проектирование задачи – это очень важный и ответственный этап в разработке любого приложения.

Важным является он вследствие того, что методы, по средствам которых пользователь управляет формами, построены на высокой степени специализации каждого из компонентов.

Необходимым условием при разработке данного приложения является описание организации данных, т.е. логическая и физическая структура данных в контексте среды разработки. В разрабатываемой программе будут использоваться три вида данных.

Первым видом являются данные, которые будут введены разработчиком на этапе реализации задачи. Сюда можно отнести изображения (иконки), описание, исходные коды сортировок.

Вторым видом данных, используемых в программе, является вводимая пользователем информация. Входной информацией в разрабатываемой программе будут являться данные.

Третьим видом данных является результат программы –отсортированные пользовательские данные. Его также относят к отдельному виду, так как ни пользователь, ни разработчик его не вводят, а программа сама получает его в результате выполнения определенных действий.

Таким образом, организация данных является важной задачей при разработке данной и любой программы.

2.2 Процессы

Согласно всем перечисленным требованиям и указаниям, которые были рассмотрены в разделе «Анализ задачи», было определено, чем конкретно должна заниматься разрабатываемая приложение. Главной задачей будет являться: предоставлении пользователям информации о различных знаках и их значении.

Для реализации задач будут использоваться процедуры. С помощью процедуры будет осуществляться переход на текстовый файл в формате .txt.

Тест будет генерироваться на основе выбора пункта в объекте TRadioGroup. Далее после того, как будет прочитан теоретический материал и выполнена практическая часть, можно будет пройти тест для проверки своих знаний и получить оценку своих знаний.

2.3 Описание внешнего пользовательского интерфейса

Важным при выполнении курсового проекта является организация диалога между, пользователем и самой программой. Во многом это зависит от того, как программист разработает данную программу, какие компоненты будут использованы и какие методы будут автоматизированы. Во-первых, особое внимание следует уделить интерфейсу. Разработчик должен так организовать внешний вид своей программы, чтобы пользователь понял, что от него требуется.

Для организации эффективной работы пользователя нужно создать целостное приложение данной предметной области, в которой все компоненты приложения будут сгруппированы по функциональному назначению. При этом необходимо обеспечить удобный графический интерфейс пользователя. Приложение должно позволить пользователю решать задачи, затрачивая значительно меньше усилий, чем при работе с разрозненными объектами.

Ниже на рисунке 1 представлена система меню и организация навигации между окнами программы:

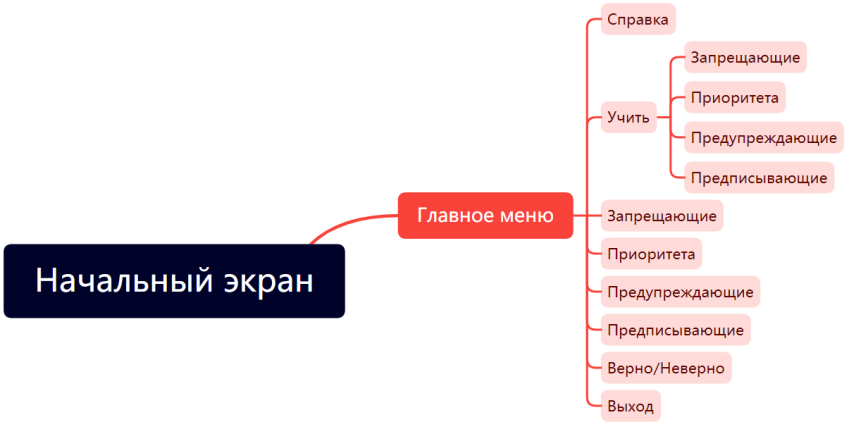

Рисунок 1 – Схема навигация между окнами программы

3. Реализация

3.1 Реализация проекта

Данный курсовой проект содержит 13 модулей. Далее рассмотрим назначение каждого модуля:

Zastavka- является заставкой в курсовом проекте;

Menu- главное меню, содержит теоретический материал, тесты, квест

TestZapr- тестовое задания;

TestPrior- тестовое задания;

TestPredupr- тестовое задания;

TestPredpis- тестовое задания;

InfoZapr- теоретический материал;

InfoPrior- теоретический материал;

InfoPredupr- теоретический материал;

InfoPredpis- теоретический материал;

InfoKategor- содержит выбор категории теоретического материала;

Help – содержит информацию о программе и разработчике.

3.1.1 Структура и описание процедур и функций пользователя

Описание разработанных процедур находятся в таблице 1.

Таблица 1 – Процедуры и функции

Имя процедуры (функции)	В каком модуле находится	За каким компонентом закреплена	Назначение
1	2	3	4
procedure FormCreate(Sender: TObject);	Zastavka	-	Включает таймер
procedure Timer1Timer(Sender: TObject);	Zastavka	Timer1	Отсчитывает время до перехода на форму Меню
procedure Image1Click(Sender: TObject);	Menu	Image1	Переход на форму с теоретическим материалом
procedure Image3Click(Sender: TObject);	Menu	Image3	Переход на форму с тестом
procedure Image4Click(Sender: TObject);	Menu	Image4	Переход на форму с тестом
procedure Image5Click(Sender: TObject);	Menu	Image5	Переход на форму с тестом

Продолжение таблицы 1

procedure Image6Click(Sender: TObject);	Menu	Image6	Переход на форму с тестом
procedure Image7Click(Sender: TObject);	Menu	Image7	Переход на форму с квестом
procedure Image8Click(Sender: TObject);	Menu	Image8	Закрыть проект
procedure FormCreate(Sender: TObject);	TestZapr, TestPrior, TestPredupr, TestPredpis	-	Загружает данные из файла
procedure NextButtonClick(Sender: TObject);	TestZapr, TestPrior, TestPredupr, TestPredpis	NextButton	Переход на следующий вопрос
procedure ExitButtonClick(Sender: TObject);	TestZapr, TestPrior, TestPredupr, TestPredpis	ExitButton	Переход в Главное меню
procedure Image2Click(Sender: TObject);	TestZapr, TestPrior, TestPredupr, TestPredpis	Image2	Переход в Главное меню
procedure DisplayQuestion(QuestionIndex: Integer);	TestZapr, TestPrior, TestPredupr, TestPredpis	-	Отображает вопрос и варианты ответа
function CheckAnswer: Boolean;	TestZapr, TestPrior, TestPredupr, TestPredpis	-	Проверяет правильность ответа на вопрос
procedure FormCreate(Sender: TObject);	InfoZapr, InfoPrior, InfoPredupr, InfoPredpis	-	Загружает информацию из файла
procedure ButtonNextClick(Sender: TObject);	InfoZapr, InfoPrior, InfoPredupr, InfoPredpis	ButtonNext	Переход на следующий теоретический материал
procedure Image1Click(Sender: TObject);	InfoZapr, InfoPrior, InfoPredupr, InfoPredpis	Image1	Переход в меню категории знаков
procedure ButtonMenuClick(Sender: TObject);	InfoZapr, InfoPrior, InfoPredupr, InfoPredpis	ButtonMenu	Переход в Главное меню
procedure Button1Click(Sender: TObject);	InfoKategor	Button1	Переход на форму с теоретическим материалом

Продолжение таблицы 1

procedure Button2Click(Sender: TObject);	InfoKategor	Button2	Переход на форму с теоретическим материалом
procedure Button3Click(Sender: TObject);	InfoKategor	Button3	Переход на форму с теоретическим материалом
procedure Button4Click(Sender: TObject);	InfoKategor	Button4	Переход на форму с теоретическим материалом
procedure FormCreate(Sender: TObject);	Quest	-	Загружает и отображает вопросы из файла
procedure ButtonYesClick(Sender: TObject);	Quest	ButtonYes	Кнопка для ответа на вопрос
procedure ButtonNoClick(Sender: TObject);	Quest	ButtonNo	Кнопка для ответа на вопрос
procedure Image2Click(Sender: TObject);	Quest	Image2	Переход в Главное меню
procedure DisplayQuestion(QuestionIndex: Integer);	Quest	-	Обновляет вопросы для перехода к следующему
procedure ShowResult;	Quest	-	Отображает результат

3.1.2 Описание использованных компонентов

Описание использованных для разработки приложения компонентов приводится в таблице 2.

Таблица 2- Использованные компоненты

Компонент	На какой форме расположен	Назначение
TMainMenu	Menu	Используется для создания главного меню проекта
TImage	Все	Используется как фон, кнопки
TTimer	Zastavka	Используется для определения длительности загрузочного экрана
TLabel	TestZapr, TestPrior, TestPredupr, TestPredpis, Quest	Отображение надписей на форме
TButton	Form2, Form3, Form4, Form5, Form6, Form7	Используется для перехода между формами, ответа и т.д.
TRadioGroup	TestZapr, TestPrior, TestPredupr, TestPredpis	Используется для выбора варианта ответа в тесте
TProgressbar	Zastavka	Используется для индикации загрузки

3.2 Спецификация программы

Точное название проекта и его состав приводится в таблице 3.

Таблица 3 – Спецификация программы

Имя файла	Назначение
1	2
Znaki.exe	Исполняемый файл проекта, используется для запуска программы на выполнение.
Znaki.dproj	Файл проекта, связывает все файлы из которых состоит приложение.
Zastavka.pas	Файл программного модуля заставки
Menu.pas	Файл программного модуля главной формы
TestZapr.pas	Файл программного модуля теста части проекта
TestPrior.pas	Файл программного модуля теста части проекта
TestPredupr.pas	Файл программного модуля теста части проекта
TestPredpis.pas	Файл программного модуля теста части проекта
InfoZapr.pas	Файл программного модуля теоретической части проекта
InfoPrior.pas	Файл программного модуля авторизация части проекта
InfoPredupr.pas	Файл программного модуля описание программы части проекта
InfoPredpis.pas	Файл программного модуля теста части проекта
InfoKategor.pas	Файл программного модуля категорий знаков части проекта
Quest.pas	Файл программного модуля квеста части проекта
Zastavka.dfm	Форма с заставкой
Menu.dfm	Главная форма
TestZapr.pas	Форма с тестом
TestPrior.pas	Форма с тестом
TestPredupr.pas	Форма с тестом
TestPredpis.pas	Форма с тестом
InfoZapr.pas	Форма с теоретической честью
InfoPrior.pas	Форма с теоретической честью
InfoPredupr.pas	Форма с теоретической честью
InfoPredpis.pas	Форма с теоретической честью
InfoKategor.pas	Форма с категориями знаков
Quest.pas	Форма с квестом
zapr.txt	Информация по знакам
prior.txt	Информация по знакам
predpis.txt	Информация по знакам
predupr.txt	Информация по знакам
Запрещающие.txt	Вопросы к тесту
предупреждающие.txt	Вопросы к тесту

Продолжение таблица 3

1	2
приоритета.txt	Вопросы к тесту
quest.txt	Квест по знакам
zaprtext.txt	Информация про знаки
priortext.txt	Информация про знаки
predpistext.txt	Информация про знаки
preduprtext.txt	Информация про знаки
fmenu.jpg	Изображение
ftest.jpg	Изображение
zastavka.jpg	Изображение
fmenu.jpg	Изображение
back.jpg	Изображение
bexit.jpg	Изображение
blean.jpg	Изображение
bprior.jpg	Изображение
bzapr.jpg	Изображение
bpredupr.jpg	Изображение
bpredpis.jpg	Изображение
bquest.jpg	Изображение
exit quest.jpg	Изображение
1.jpg	Изображение
2.jpg	Изображение
3.jpg	Изображение
4.jpg	Изображение
5.jpg	Изображение
6.jpg	Изображение
7.jpg	Изображение
8.jpg	Изображение
9.jpg	Изображение
10.jpg	Изображение
11.jpg	Изображение
12.jpg	Изображение
13.jpg	Изображение
14.jpg	Изображение
15.jpg	Изображение
16.jpg	Изображение

Продолжение таблица 3

1	2
17.jpg	Изображение
18.jpg	Изображение
19.jpg	Изображение
20.jpg	Изображение

4. Тестирование

При разработке данной программы многие возникающие ошибки и недоработки были исправлены на этапе реализации проекта. После завершения этапа написания программы было проведено тщательное функциональное тестирование. Функциональное тестирование должно гарантировать работу всех элементов программы в автономном режиме.

Отчет о результатах тестирования представлен в таблице 4

Таблица 4- Отчет о результатах тестирования

Идентификатор теста	Тест	Ожидаемый результат	Физический результат	Результат тестирование
1	Проверка входа пользователя в приложение	Открытие главного окна	Открытие главного окна	Выполнено
2	Проверка перехода на Главное меню	Открытие формы с Главным меню	Открытие формы с Главным меню	Выполнено
3	Проверка кнопки-«Учить»	Открытие формы с категориями знаков	Открытие формы с категориями знаков	Выполнено
4	Проверка кнопки-«Запрещающие»	Открытие формы с тестом	Открытие формы с тестом	Выполнено
5	Проверка кнопки-«Приоритета»	Открытие формы с тестом	Открытие формы с тестом	Выполнено
6	Проверка кнопки-«Предупреждающие»	Открытие формы с тестом	Открытие формы с тестом	Выполнено
7	Проверка кнопки-«Предписывающие»	Открытие формы с тестом	Открытие формы с тестом	Выполнено
8	Проверка кнопки-«Верно/Неверно»	Открытие формы с викториной	Открытие формы с викториной	Выполнено
9	Проверка главного меня на кнопку «Справка»	Открытие формы с описанием программы	Открытие формы с описанием программы	Выполнено
10	Проверка кнопки «Далее»	Переключение вопроса	Переключение вопроса	Выполнено

Продолжение таблицы 4

1	2	3	4	5
11	Проверка кнопки «На Главную»	Переход в Главное меню	Переход в Главное меню	Выполнено
12	Проверка кнопки «Далее»	Переключение информации со знаками	Переключение информации со знаками	Выполнено
13	Проверка кнопки «Да»	Считывание ответа	Считывание ответа	Выполнено

14	Проверка кнопки «Нет»	Считывание ответа	Считывание ответа	Выполнено
15	Проверка кнопки «Вариант ответа»	Выбор варианта ответа	Выбор варианта ответа	Выполнено
16	Проверка кнопки «Выход»	Закрытие проекта	Закрытие проекта	Выполнено

При разработке программного продукта было решено множество проблем, например, не осуществлялся переход к следующему изображению по нажатию кнопки или открывался доступ к последнему квесту, когда был пройден только первый, поэтому в будущем пользователь не столкнется с данными проблемами.

Элементы программы были проверены, и было установлено, что все они работают правильно и выполняют задачи, указанные в процедурах.

5 Применение

5.1 Общие сведения о программном продукте

Цель данного проекта заключается в предоставлении пользователям информации о различных дорожных знаках и их значениях.

Создаваемое приложение будет рассчитано на помощь водителям и пешеходам лучше понять и запомнить знаки, используемые на дорогах.

Быстродействие любой программы во многом зависит от характеристик выбранного персонального компьютера: рабочей частоты процессора, объема оперативной памяти и т.д. Несмотря на все реализованные в ней задачи, она легко запускается и функционирует на любых машинах.

Тестирование проводилось на разных классах ЭВМ и работать с данной программой было комфортно. Программа разработана на ПК со следующими характеристиками:

- Процессор: Intel(R) Core(TM) i5-6300U 2.50 GHz;
- ОЗУ: 8Gb;
- Память: SSD 256Gb;
- ОС – Windows 11.

5.2 Инсталляция

Для того, чтобы установить программу необходимо запустить файл Установщик.exe. Появится окно установки приложения “ Знаки дорожного движения”.

Затем достаточно следовать приведенной инструкции установки приложения.

5.3 Выполнение программы

5.3.1 Запуск программы

Данную программу можно запустить различными способами. Первым из них является запуск с помощью ярлыка на рабочем столе. Необходимо дважды щелкнуть левой кнопкой мыши на ярлыке с названием “ Электронное средство обучения для учащихся профессиональной подготовки рабочих по профессии «Оператор ЭВМ»

Вторым способом является запуск из каталога, в который устанавливалось приложение (по умолчанию C:\Program Files (x86)\Знаки).

По подготовленным тестам будет осуществляться функциональное и полное тестирование программного продукта. Отчет о результатах тестирования будет представлен в 4 разделе пояснительной записки.

5.3.2 Инструкции по работе с программой

После запуска приложения на экране нас встречает заставка на рисунок 2.

Рисунок 2 – Заставка

Про истечению времени будет осуществлен переход в Главное меню, представлен на рисунке 3, где по нажатию на кнопку «Учить» можно перейти на форму информации по категориям знаков рисунке 4.

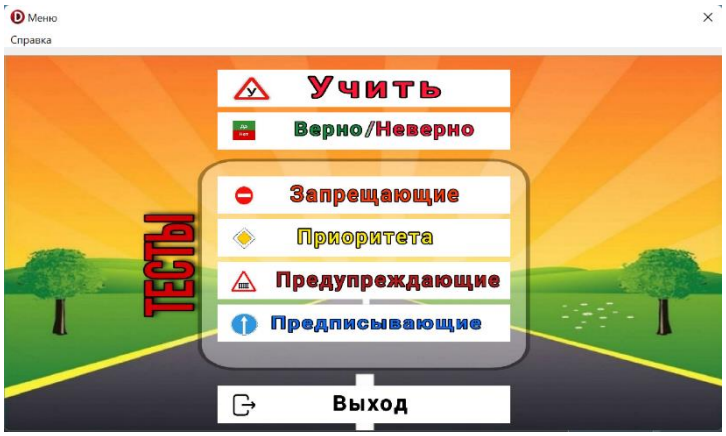

Рисунок 3- Главное меню

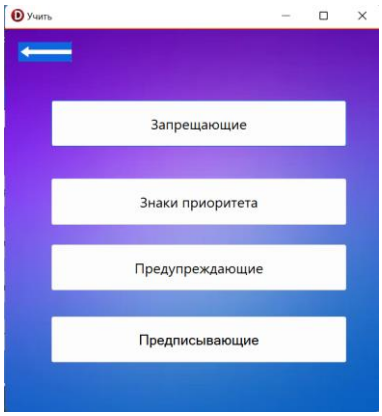

Рисунок 4- меню категорий знаков

По нажатию на кнопку “Справка ” в главном меню можно выбрать описание программы

Рисунок 5-Описание программы

По нажатию кнопки одной из кнопок меню категорий, откроется форма с теоретическией, нажимая кнопку «Далее» меняется знак с информацией.

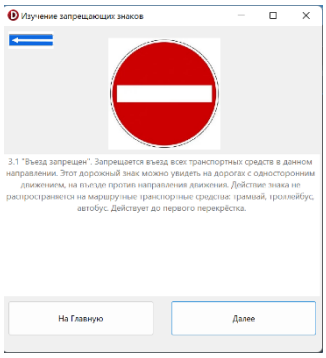

Рисунок 6- Теоретический материал

Нажав на кнопку «На Главную» - переход в Главное меню, где нажав на одну из кнопок в рамке «Тесты», начинается тест по выбранным знакам

Также можем по нажатию кнопки вернуться в Главное меню или нажимая «Далее» - переходить к новому вопросу

После завершение теста перед нами буду выведены результаты.

Рисунок 7- Тест по знакам результат

Нажав на Главном меню на кнопку «Верно/Неверно», вас направит на форму с викториной, где с помощью кнопок «Да» и «Нет» происходит ответ на вопрос, где также по окончанию выдаст резутьтат. Рисунок 8.

Рисунок 8- Результат прохождения викторины

Заключение

Целью данного проекта заключалось предоставлении пользователям информации о различных дорожных знаках и их значениях.

В поставленной задаче был реализован простой и понятный пользовательский интерфейс.

В ходе тестирования все исключительные ситуации были обработаны. Проект работает без сбоев и ошибок.

В процессе разработки программного продукта я научился создавать динамические компоненты на форме, проработал их взаимодействие, закрепил умение создания собственных процедур, научился обрабатывать все исключительные ситуации.

Исходя из этого, можно сделать вывод, что программа реализована успешно.

Список использованных источников

1. Симонович С.В., Евсеев Г.А., Алексеев А.Г. Специальная информатика: Учебное пособие. - М.: АСТ-ПРЕСС: Инфорком-Пресс, 2001. - 480 с.

2. Архангельский, А. Я. Delphi 7. Справочное пособие. - Москва: Бином-Пресс, 2014. - 1024 с.

3. Вощинская, Г.Э. Разработка компонентов в DELPHI. - Воронеж: ИПЦ ВГУ, 2007. - 57 с.

4. Культин, Н. Основы программирования в Delphi 7. - Санкт-Петербург: Питер, 2009. - 640 с.

5. Архангельский, А. Я. Delphi 7. Справочное пособие. - Москва: Бином-Пресс, 2014. - 1024 с.

6. Культин, Н. Delphi 6. Программирование на Object Pascal / Н. Культин. - М.: БХВ-Петербург, 2012. - 528 с.

Приложение А

Листинг программы

```
unit Zastavka;
interface
uses
  Winapi.Windows, Winapi.Messages, System.SysUtils,
System.Variants, System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls,
Vcl.ExtCtrls,
  Vcl.Imaging.jpeg, Vcl.ComCtrls;
type
  TForm1 = class(TForm)
    Image1: TImage;
    ProgressBar1: TProgressBar;
    Timer1: TTimer;
    Label2: TLabel;
    procedure FormCreate(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
implementation
{$R *.dfm}
uses Menu;
procedure TForm1.FormCreate(Sender: TObject);
begin
timer1.Enabled:=true;
end;
procedure TForm1.Timer1Timer(Sender: TObject);
begin
progressbar1.Position:= progressbar1.Position + 1;
if progressbar1.Position = progressbar1.Max then
begin
timer1.Enabled := false;
Form1.Hide;
Form2.Show;
end;
end;
end.close;
end;
end;
end.
```

```
unit Menu;
interface
uses
  Winapi.Windows, Winapi.Messages, System.SysUtils,
System.Variants, System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls,
Vcl.Imaging.jpeg,
  Vcl.ExtCtrls, Vcl.Menus, Vcl.Buttons, Vcl.Imag-
ing.pngimage, ShellAPI;
type
  TForm2 = class(TForm)
    MainMenu1: TMainMenu;
    N1: TMenuItem;
    Image2: TImage;
    Image1: TImage;
    Image3: TImage;
    Image4: TImage;
    Image5: TImage;
    Image6: TImage;
    Image7: TImage;
    Image8: TImage;
    procedure Image1Click(Sender: TObject);
    procedure Image3Click(Sender: TObject);
    procedure Image4Click(Sender: TObject);
    procedure Image5Click(Sender: TObject);
    procedure Image6Click(Sender: TObject);
    procedure Image7Click(Sender: TObject);
    procedure Image8Click(Sender: TObject);
    procedure N1Click(Sender: TObject);
```

```
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form2: TForm2;
implementation
{$R *.dfm}
uses TestZapr, TestPrior, TestPredupr, TestPredpis,
Zastavka, InfoPredupr, InfoKategor, Quest;
procedure TForm2.Image1Click(Sender: TObject);
begin
Form2.Hide;
Form11.Show;
end;

procedure TForm2.Image3Click(Sender: TObject);
begin
Form2.Hide;
Form3.Show;
end;

procedure TForm2.Image4Click(Sender: TObject);
begin
Form2.Hide;
Form4.Show;
end;
procedure TForm2.Image5Click(Sender: TObject);
begin
Form2.Hide;
Form5.Show;
end;
procedure TForm2.Image6Click(Sender: TObject);
begin
Form2.Hide;
Form6.Show;
end;
procedure TForm2.Image7Click(Sender: TObject);
begin
Form2.Hide;
Form12.Show;
end;
procedure TForm2.Image8Click(Sender: TObject);
begin
Close;
end;
procedure TForm2.N1Click(Sender: TObject);
begin
ShellExecute(0, 'Open', 'Справка.chm', nil, nil,
SW_SHOW);
end;
end.

unit TestZapr;
interface
uses
  Winapi.Windows, Winapi.Messages, System.SysUtils,
System.Variants, System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls,
Vcl.ExtCtrls,
  Vcl.Imaging.jpeg;
type
  TQuestion = record
    ImageFileName: string;
    Options: array[0..3] of string;
    CorrectAnswer: Integer;
  end;
  TForm3 = class(TForm)
    Image1: TImage;
    ResultLabel: TLabel;
    QuestionImage: TImage;
    Image2: TImage;
```

```
    OptionGroup: TRadioGroup;
    ExitButton: TButton;
    NextButton: TButton;
    procedure FormCreate(Sender: TObject);
    procedure NextButtonClick(Sender: TObject);
    procedure ExitButtonClick(Sender: TObject);
    procedure Image2Click(Sender: TObject);
  private
    Questions: array of TQuestion;
    CurrentQuestionIndex: Integer;
    CorrectAnswersCount: Integer;
    procedure DisplayQuestion(QuestionIndex: Integer);
    function CheckAnswer: Boolean;
  end;
var
  Form3: TForm3;
implementation
{$R *.dfm}
uses Menu;
procedure TForm3.FormCreate(Sender: TObject);
var
  FileLines: TStringList;
  i: Integer;
begin
  FileLines := TStringList.Create;
  try
    FileLines.LoadFromFile(ExtractFilePath(ParamStr(0))
+'запрещающие.txt', TEncoding.UTF8);
    SetLength(Questions, FileLines.Count div 6);
    for i := 0 to Length(Questions) - 1 do
    begin
      Questions[i].ImageFileName := FileLines[i * 6];
      Questions[i].Options[0] := FileLines[i * 6 + 1];
      Questions[i].Options[1] := FileLines[i * 6 + 2];
      Questions[i].Options[2] := FileLines[i * 6 + 3];
      Questions[i].Options[3] := FileLines[i * 6 + 4];
      Questions[i].CorrectAnswer := StrToInt(FileLines[i *
6 + 5]);
    end;
    CurrentQuestionIndex := 0;
    CorrectAnswersCount := 0;
    DisplayQuestion(CurrentQuestionIndex);
  finally
    FileLines.Free;
  end;
end;
procedure TForm3.Image2Click(Sender: TObject);
begin
Form3.Hide;
Form2.Show;
    CurrentQuestionIndex := 0;
    CorrectAnswersCount := 0;
    DisplayQuestion(CurrentQuestionIndex);
    ResultLabel.Caption := '';
    NextButton.Enabled := True;
end;
procedure TForm3.DisplayQuestion(QuestionIndex: In-
teger);
begin
  QuestionImage.Picture.LoadFromFile(Questions[Ques-
tionIndex].ImageFileName);
  OptionGroup.Items.Clear;
  OptionGroup.Items.Add(Questions[QuestionIndex].Op-
tions[0]);
  OptionGroup.Items.Add(Questions[QuestionIndex].Op-
tions[1]);
  OptionGroup.Items.Add(Questions[QuestionIndex].Op-
tions[2]);
  OptionGroup.Items.Add(Questions[QuestionIndex].Op-
tions[3]);

  OptionGroup.ItemIndex := -1;
end;
procedure TForm3.ExitButtonClick(Sender: TObject);
begin
```

```
Form3.Hide;
Form2.Show;
  CurrentQuestionIndex := 0;
  CorrectAnswersCount := 0;
  DisplayQuestion(CurrentQuestionIndex);
  ResultLabel.Caption := '';
  NextButton.Enabled := True;
end;
function TForm3.CheckAnswer: Boolean;
begin
 Result := OptionGroup.ItemIndex = Questions[Cur-
rentQuestionIndex].CorrectAnswer;
end;

procedure TForm3.NextButtonClick(Sender: TObject);
begin
 if OptionGroup.ItemIndex <> -1 then
 begin
  if CheckAnswer then
   Inc(CorrectAnswersCount);
  Inc(CurrentQuestionIndex);
  if CurrentQuestionIndex < Length(Questions) then
   DisplayQuestion(CurrentQuestionIndex)
  else
  begin
   ResultLabel.Caption := Format('Тест завершен! Ко-
личество правильных ответов: %d из %d',
[CorrectAnswersCount, Length(Questions)]);
    NextButton.Enabled := False;
  end;
 end
 else
  ShowMessage('Выберите вариант ответа!');
end;
end.

unit TestPrior;
interface
uses
 Winapi.Windows, Winapi.Messages, System.SysUtils,
System.Variants, System.Classes, Vcl.Graphics,
 Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls,
Vcl.ExtCtrls,
 Vcl.Imaging.jpeg;
type
 TQuestion = record
  ImageFileName: string;
  Options: array[0..3] of string;
  CorrectAnswer: Integer;
 end;
 TForm4 = class(TForm)
  Image1: TImage;
  ResultLabel: TLabel;
  QuestionImage: TImage;
  Image2: TImage;
  OptionGroup: TRadioGroup;
  ExitButton: TButton;
  NextButton: TButton;
  procedure FormCreate(Sender: TObject);
  procedure NextButtonClick(Sender: TObject);
  procedure ExitButtonClick(Sender: TObject);
  procedure Image2Click(Sender: TObject);
 private
  Questions: array of TQuestion;
  CurrentQuestionIndex: Integer;
  CorrectAnswersCount: Integer;
  procedure DisplayQuestion(QuestionIndex: Integer);
  function CheckAnswer: Boolean;
 end;
var
 Form4: TForm4;
implementation
{$R *.dfm}
uses Menu;
```

```
procedure TForm4.FormCreate(Sender: TObject);
var
  FileLines: TStringList;
  i: Integer;
begin
  FileLines := TStringList.Create;
  try
    FileLines.LoadFromFile(ExtractFilePath(ParamStr(0))
+'приоритета.txt', TEncoding.UTF8);
    SetLength(Questions, FileLines.Count div 6);
    for i := 0 to Length(Questions) - 1 do
    begin
      Questions[i].ImageFileName := FileLines[i * 6];
      Questions[i].Options[0] := FileLines[i * 6 + 1];
      Questions[i].Options[1] := FileLines[i * 6 + 2];
      Questions[i].Options[2] := FileLines[i * 6 + 3];
      Questions[i].Options[3] := FileLines[i * 6 + 4];
      Questions[i].CorrectAnswer := StrToInt(FileLines[i *
6 + 5]);
    end;
    CurrentQuestionIndex := 0;
    CorrectAnswersCount := 0;
    DisplayQuestion(CurrentQuestionIndex);
  finally
    FileLines.Free;
  end;
end;
procedure TForm4.Image2Click(Sender: TObject);
begin
Form4.Hide;
Form2.Show;
    CurrentQuestionIndex := 0;
    CorrectAnswersCount := 0;
    DisplayQuestion(CurrentQuestionIndex);
    ResultLabel.Caption := '';
    NextButton.Enabled := True;
end;
procedure TForm4.DisplayQuestion(QuestionIndex: In-
teger);
begin
  QuestionImage.Picture.LoadFromFile(Questions[Ques-
tionIndex].ImageFileName);
  OptionGroup.Items.Clear;
  OptionGroup.Items.Add(Questions[QuestionIndex].Op-
tions[0]);
  OptionGroup.Items.Add(Questions[QuestionIndex].Op-
tions[1]);
  OptionGroup.Items.Add(Questions[QuestionIndex].Op-
tions[2]);
  OptionGroup.Items.Add(Questions[QuestionIndex].Op-
tions[3]);


  OptionGroup.ItemIndex := -1;
end;
procedure TForm4.ExitButtonClick(Sender: TObject);
begin
Form4.Hide;
Form2.Show;
    CurrentQuestionIndex := 0;
    CorrectAnswersCount := 0;
    DisplayQuestion(CurrentQuestionIndex);
    ResultLabel.Caption := '';
    NextButton.Enabled := True;
end;
function TForm4.CheckAnswer: Boolean;
begin
  Result := OptionGroup.ItemIndex = Questions[Cur-
rentQuestionIndex].CorrectAnswer;
end;
procedure TForm4.NextButtonClick(Sender: TObject);
begin
  if OptionGroup.ItemIndex <> -1 then
  begin
    if CheckAnswer then
      Inc(CorrectAnswersCount);
    Inc(CurrentQuestionIndex);
```

```
    if CurrentQuestionIndex < Length(Questions) then
      DisplayQuestion(CurrentQuestionIndex)
    else
    begin
      ResultLabel.Caption := Format('Тест завершен! Количество правильных ответов: %d из %d',
[CorrectAnswersCount, Length(Questions)]);
      NextButton.Enabled := False;
    end;
  end
  else
    ShowMessage('Выберите вариант ответа!');
end;
end.
unit TestPredupr;
interface
uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Vcl.ExtCtrls,
  Vcl.Imaging.jpeg;
type
  TQuestion = record
    ImageFileName: string;
    Options: array[0..3] of string;
    CorrectAnswer: Integer;
  end;
  TForm5 = class(TForm)
    Image1: TImage;
    ResultLabel: TLabel;
    QuestionImage: TImage;
    Image2: TImage;
    OptionGroup: TRadioGroup;
    ExitButton: TButton;
    NextButton: TButton;
    procedure FormCreate(Sender: TObject);
    procedure NextButtonClick(Sender: TObject);
    procedure ExitButtonClick(Sender: TObject);
    procedure Image2Click(Sender: TObject);
  private
    Questions: array of TQuestion;
    CurrentQuestionIndex: Integer;
    CorrectAnswersCount: Integer;
    procedure DisplayQuestion(QuestionIndex: Integer);
    function CheckAnswer: Boolean;
  end;
var
  Form5: TForm5;
implementation
{$R *.dfm}
uses Menu;
procedure TForm5.FormCreate(Sender: TObject);
var
  FileLines: TStringList;
  i: Integer;
begin
  FileLines := TStringList.Create;
  try
    FileLines.LoadFromFile(ExtractFilePath(ParamStr(0))+'предупреждающие.txt', TEncoding.UTF8);
    SetLength(Questions, FileLines.Count div 6);
    for i := 0 to Length(Questions) - 1 do
    begin
      Questions[i].ImageFileName := FileLines[i * 6];
      Questions[i].Options[0] := FileLines[i * 6 + 1];
      Questions[i].Options[1] := FileLines[i * 6 + 2];
      Questions[i].Options[2] := FileLines[i * 6 + 3];
      Questions[i].Options[3] := FileLines[i * 6 + 4];
      Questions[i].CorrectAnswer := StrToInt(FileLines[i * 6 + 5]);
    end;
    CurrentQuestionIndex := 0;
    CorrectAnswersCount := 0;
```

```
    DisplayQuestion(CurrentQuestionIndex);
  finally
    FileLines.Free;
  end;
end;
procedure TForm5.Image2Click(Sender: TObject);
begin
Form5.Hide;
Form2.Show;
    CurrentQuestionIndex := 0;
    CorrectAnswersCount := 0;
    DisplayQuestion(CurrentQuestionIndex);
    ResultLabel.Caption := '';
    NextButton.Enabled := True;
end;
procedure TForm5.DisplayQuestion(QuestionIndex: Integer);
begin
  QuestionImage.Picture.LoadFromFile(Questions[QuestionIndex].ImageFileName);
  OptionGroup.Items.Clear;
  OptionGroup.Items.Add(Questions[QuestionIndex].Options[0]);
  OptionGroup.Items.Add(Questions[QuestionIndex].Options[1]);
  OptionGroup.Items.Add(Questions[QuestionIndex].Options[2]);
  OptionGroup.Items.Add(Questions[QuestionIndex].Options[3]);

  OptionGroup.ItemIndex := -1;
end;
procedure TForm5.ExitButtonClick(Sender: TObject);
begin
Form5.Hide;
Form2.Show;
    CurrentQuestionIndex := 0;
    CorrectAnswersCount := 0;
    DisplayQuestion(CurrentQuestionIndex);
    ResultLabel.Caption := '';
    NextButton.Enabled := True;
end;
function TForm5.CheckAnswer: Boolean;
begin
  Result := OptionGroup.ItemIndex = Questions[CurrentQuestionIndex].CorrectAnswer;
end;
procedure TForm5.NextButtonClick(Sender: TObject);
begin
  if OptionGroup.ItemIndex <> -1 then
  begin
    if CheckAnswer then
      Inc(CorrectAnswersCount);
    Inc(CurrentQuestionIndex);
    if CurrentQuestionIndex < Length(Questions) then
      DisplayQuestion(CurrentQuestionIndex)
    else
    begin
      ResultLabel.Caption := Format('Тест завершен! Количество правильных ответов: %d из %d', [CorrectAnswersCount, Length(Questions)]);
      NextButton.Enabled := False;
    end;
  end
  else
    ShowMessage('Выберите вариант ответа!');
end;
end.
unit TestPredpis;

interface
uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Vcl.ExtCtrls,
```

```
  Vcl.Imaging.jpeg;
type
  TQuestion = record
    ImageFileName: string;
    Options: array[0..3] of string;
    CorrectAnswer: Integer;
  end;
  TForm6 = class(TForm)
    Image1: TImage;
    ResultLabel: TLabel;
    QuestionImage: TImage;
    Image2: TImage;
    OptionGroup: TRadioGroup;
    ExitButton: TButton;
    NextButton: TButton;
    procedure FormCreate(Sender: TObject);
    procedure NextButtonClick(Sender: TObject);
    procedure ExitButtonClick(Sender: TObject);
    procedure Image2Click(Sender: TObject);
  private
    Questions: array of TQuestion;
    CurrentQuestionIndex: Integer;
    CorrectAnswersCount: Integer;
    procedure DisplayQuestion(QuestionIndex: Integer);
    function CheckAnswer: Boolean;
  end;
var
  Form6: TForm6;

implementation

{$R *.dfm}
uses Menu;
procedure TForm6.FormCreate(Sender: TObject);
var
  FileLines: TStringList;
  i: Integer;
begin
  FileLines := TStringList.Create;
  try
    FileLines.LoadFromFile(ExtractFilePath(ParamStr(0))
+'предписывающие.txt', TEncoding.UTF8);
    SetLength(Questions, FileLines.Count div 6);
    for i := 0 to Length(Questions) - 1 do
    begin
      Questions[i].ImageFileName := FileLines[i * 6];
      Questions[i].Options[0] := FileLines[i * 6 + 1];
      Questions[i].Options[1] := FileLines[i * 6 + 2];
      Questions[i].Options[2] := FileLines[i * 6 + 3];
      Questions[i].Options[3] := FileLines[i * 6 + 4];
      Questions[i].CorrectAnswer := StrToInt(FileLines[i *
6 + 5]);
    end;
    CurrentQuestionIndex := 0;
    CorrectAnswersCount := 0;
    DisplayQuestion(CurrentQuestionIndex);
  finally
    FileLines.Free;
  end;
end;
procedure TForm6.Image2Click(Sender: TObject);
begin
Form6.Hide;
Form2.Show;
    CurrentQuestionIndex := 0;
    CorrectAnswersCount := 0;
    DisplayQuestion(CurrentQuestionIndex);
    ResultLabel.Caption := '';
    NextButton.Enabled := True;
end;
procedure TForm6.DisplayQuestion(QuestionIndex: In-
teger);
begin
```

```
  QuestionImage.Picture.LoadFromFile(Questions[Ques-
tionIndex].ImageFileName);
  OptionGroup.Items.Clear;
  OptionGroup.Items.Add(Questions[QuestionIndex].Op-
tions[0]);
  OptionGroup.Items.Add(Questions[QuestionIndex].Op-
tions[1]);
  OptionGroup.Items.Add(Questions[QuestionIndex].Op-
tions[2]);
  OptionGroup.Items.Add(Questions[QuestionIndex].Op-
tions[3]);


  OptionGroup.ItemIndex := -1;
end;
procedure TForm6.ExitButtonClick(Sender: TObject);
begin
Form6.Hide;
Form2.Show;
    CurrentQuestionIndex := 0;
    CorrectAnswersCount := 0;
    DisplayQuestion(CurrentQuestionIndex);
    ResultLabel.Caption := '';
    NextButton.Enabled := True;
end;
function TForm6.CheckAnswer: Boolean;
begin
  Result := OptionGroup.ItemIndex = Questions[Cur-
rentQuestionIndex].CorrectAnswer;
end;
procedure TForm6.NextButtonClick(Sender: TObject);
begin
  if OptionGroup.ItemIndex <> -1 then
  begin
    if CheckAnswer then
      Inc(CorrectAnswersCount);
    Inc(CurrentQuestionIndex);
    if CurrentQuestionIndex < Length(Questions) then
      DisplayQuestion(CurrentQuestionIndex)
    else
    begin
      ResultLabel.Caption := Format('Тест завершен! Ко-
личество правильных ответов: %d из %d',
[CorrectAnswersCount, Length(Questions)]);
      NextButton.Enabled := False;
    end;
  end
  else
    ShowMessage('Выберите вариант ответа!');
end;
end.


unit InfoZapr;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes,
Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls, Vcl.Imaging.jpeg;
type
  TForm7 = class(TForm)
    ImagePicture: TImage;
    MemoText: TMemo;
    ButtonNext: TButton;
    Image1: TImage;
    ButtonMenu: TButton;
    procedure FormCreate(Sender: TObject);
    procedure ButtonNextClick(Sender: TObject);
    procedure Image1Click(Sender: TObject);
    procedure ButtonMenuClick(Sender: TObject);
  private
    { Private declarations }
    ImagePaths: TStringList;
    CurrentIndex: Integer;
    Texts: TStringList;
  public
    { Public declarations }
```

```
    destructor Destroy; override;
  end;

var
  Form7: TForm7;
implementation
{$R *.dfm}
uses InfoKategor, Menu;
procedure TForm7.FormCreate(Sender: TObject);
var
  Text: string;
begin
  ImagePaths := TStringList.Create;
  ImagePaths.LoadFromFile(ExtractFilePath(Param-
Str(0)) +'zapr.txt', TEncoding.UTF8);
  CurrentIndex := 0;
  if ImagePaths.Count > 0 then
    ImagePicture.Picture.LoadFromFile(ImagePaths[Cur-
rentIndex]);

  Texts := TStringList.Create;
  Texts.LoadFromFile(ExtractFilePath(ParamStr(0))
+'zaprtext.txt', TEncoding.UTF8);

  if Texts.Count > 0 then
  begin
    Text := StringReplace(Texts[CurrentIndex], '*', sLine-
Break, [rfReplaceAll]);
    MemoText.Text := StringReplace(Text, '*', '',
[rfReplaceAll]);
  end;
end;
procedure TForm7.Image1Click(Sender: TObject);
begin
Form7.Hide;
Form11.Show;
end;
procedure TForm7.ButtonMenuClick(Sender: TObject);
begin
Form7.Hide;
Form2.Show;
end;
procedure TForm7.ButtonNextClick(Sender: TObject);
begin
  Inc(CurrentIndex);
  if CurrentIndex >= ImagePaths.Count then
    CurrentIndex := 0;
  if Texts.Count > CurrentIndex then
  begin
    Text := StringReplace(Texts[CurrentIndex], '*', sLine-
Break, [rfReplaceAll]);
    MemoText.Text := StringReplace(Text, '*', '',
[rfReplaceAll]);
  end;
  ImagePicture.Picture.LoadFromFile(ImagePaths[Cur-
rentIndex]);
end;
destructor TForm7.Destroy;
begin
  ImagePaths.Free;
  Texts.Free;
  inherited Destroy;
end;

end.
unit InfoPrior;

interface
uses
  Windows, Messages, SysUtils, Variants, Classes,
Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls, Vcl.Imaging.jpeg;
type
  TForm8 = class(TForm)
    ImagePicture: TImage;
```

```
    MemoText: TMemo;
    ButtonNext: TButton;
    Image1: TImage;
    ButtonMenu: TButton;
    procedure FormCreate(Sender: TObject);
    procedure ButtonNextClick(Sender: TObject);
    procedure Image1Click(Sender: TObject);
    procedure ButtonMenuClick(Sender: TObject);
  private
    { Private declarations }
    ImagePaths: TStringList;
    CurrentIndex: Integer;
    Texts: TStringList;
  public
    { Public declarations }
    destructor Destroy; override;
  end;
var
  Form8: TForm8;
implementation
{$R *.dfm}
uses InfoKategor, Menu;
procedure TForm8.FormCreate(Sender: TObject);
var
  Text: string;
begin
  ImagePaths := TStringList.Create;
  ImagePaths.LoadFromFile(ExtractFilePath(Param-
Str(0)) +'prior.txt', TEncoding.UTF8);
  CurrentIndex := 0;
  if ImagePaths.Count > 0 then
    ImagePicture.Picture.LoadFromFile(ImagePaths[Cur-
rentIndex]);
  Texts := TStringList.Create;
  Texts.LoadFromFile(ExtractFilePath(ParamStr(0))
+'priortext.txt', TEncoding.UTF8);
  if Texts.Count > 0 then
  begin
    Text := StringReplace(Texts[CurrentIndex], '*', sLine-
Break, [rfReplaceAll]);
    MemoText.Text := StringReplace(Text, '*', '',
[rfReplaceAll]);
  end;
end;
procedure TForm8.Image1Click(Sender: TObject);
begin
Form8.Hide;
Form11.Show;
end;

procedure TForm8.ButtonMenuClick(Sender: TObject);
begin
Form8.Hide;
Form2.Show;
end;

procedure TForm8.ButtonNextClick(Sender: TObject);
begin
  Inc(CurrentIndex);
  if CurrentIndex >= ImagePaths.Count then
    CurrentIndex := 0;
  if Texts.Count > CurrentIndex then
  begin
    Text := StringReplace(Texts[CurrentIndex], '*', sLine-
Break, [rfReplaceAll]);
    MemoText.Text := StringReplace(Text, '*', '',
[rfReplaceAll]);
  end;
  ImagePicture.Picture.LoadFromFile(ImagePaths[Cur-
rentIndex]);
end;
destructor TForm8.Destroy;
begin
  ImagePaths.Free;
  Texts.Free;
```

```
  inherited Destroy;
end;
end.
unit InfoPredupr;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes,
Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls, Vcl.Imaging.jpeg;
type
  TForm9 = class(TForm)
    ImagePicture: TImage;
    MemoText: TMemo;
    ButtonNext: TButton;
    Image1: TImage;
    ButtonMenu: TButton;
    procedure FormCreate(Sender: TObject);
    procedure ButtonNextClick(Sender: TObject);
    procedure Image1Click(Sender: TObject);
    procedure ButtonMenuClick(Sender: TObject);
  private
    { Private declarations }
    ImagePaths: TStringList;
    CurrentIndex: Integer;
    Texts: TStringList;
  public
    { Public declarations }
    destructor Destroy; override;
  end;
var
  Form9: TForm9;
implementation
{$R *.dfm}
uses InfoKategor, Menu;
procedure TForm9.FormCreate(Sender: TObject);
var
  Text: string;
begin
  ImagePaths := TStringList.Create;
  ImagePaths.LoadFromFile(ExtractFilePath(Param-
Str(0)) +'predupr.txt', TEncoding.UTF8);


  CurrentIndex := 0;
  if ImagePaths.Count > 0 then
    ImagePicture.Picture.LoadFromFile(ImagePaths[Cur-
rentIndex]);
  Texts := TStringList.Create;
  Texts.LoadFromFile(ExtractFilePath(ParamStr(0))
+'predupртext.txt', TEncoding.UTF8);
  if Texts.Count > 0 then
  begin
    Text := StringReplace(Texts[CurrentIndex], '*', sLine-
Break, [rfReplaceAll]);
    MemoText.Text := StringReplace(Text, '*', '',
[rfReplaceAll]);
  end;
end;
procedure TForm9.Image1Click(Sender: TObject);
begin
Form9.Hide;
Form11.Show;
end;
procedure TForm9.ButtonMenuClick(Sender: TObject);
begin
Form9.Hide;
Form2.Show;
end;
procedure TForm9.ButtonNextClick(Sender: TObject);
begin
  Inc(CurrentIndex);
  if CurrentIndex >= ImagePaths.Count then
    CurrentIndex := 0;
  if Texts.Count > CurrentIndex then
  begin
```

```
  Text := StringReplace(Texts[CurrentIndex], '*', sLine-
Break, [rfReplaceAll]);
  MemoText.Text := StringReplace(Text, '*', '',
[rfReplaceAll]);
 end;
 ImagePicture.Picture.LoadFromFile(ImagePaths[Cur-
rentIndex]);
end;
destructor TForm9.Destroy;
begin
 ImagePaths.Free;
 Texts.Free;
 inherited Destroy;
end;
end.

unit InfoPredpis;
interface
uses
 Windows, Messages, SysUtils, Variants, Classes,
Graphics, Controls, Forms,
 Dialogs, StdCtrls, ExtCtrls, Vcl.Imaging.jpeg;
type
 TForm10 = class(TForm)
  ImagePicture: TImage;
  MemoText: TMemo;
  ButtonNext: TButton;
  Image1: TImage;
  ButtonMenu: TButton;
  procedure FormCreate(Sender: TObject);
  procedure ButtonNextClick(Sender: TObject);
  procedure Image1Click(Sender: TObject);
  procedure ButtonMenuClick(Sender: TObject);
 private
  { Private declarations }
  ImagePaths: TStringList;
  CurrentIndex: Integer;
  Texts: TStringList;
 public
  { Public declarations }
  destructor Destroy; override;
 end;
var
 Form10: TForm10;
implementation
{$R *.dfm}
uses InfoKategor, Menu;
procedure TForm10.FormCreate(Sender: TObject);
var
 Text: string;
begin
 ImagePaths := TStringList.Create;
 ImagePaths.LoadFromFile(ExtractFilePath(Param-
Str(0)) +'predpis.txt', TEncoding.UTF8);
 CurrentIndex := 0;
 if ImagePaths.Count > 0 then
  ImagePicture.Picture.LoadFromFile(ImagePaths[Cur-
rentIndex]);
 Texts := TStringList.Create;
 Texts.LoadFromFile(ExtractFilePath(ParamStr(0))
+'predpistext.txt', TEncoding.UTF8);
 if Texts.Count > 0 then
 begin
  Text := StringReplace(Texts[CurrentIndex], '*', sLine-
Break, [rfReplaceAll]);
  MemoText.Text := StringReplace(Text, '*', '',
[rfReplaceAll]);
 end;
end;
procedure TForm10.Image1Click(Sender: TObject);
begin
Form10.Hide;
Form11.Show;
end;
procedure TForm10.ButtonMenuClick(Sender: TObject);
```

```
begin
Form10.Hide;
Form2.Show;
end;
procedure TForm10.ButtonNextClick(Sender: TObject);
begin
  Inc(CurrentIndex);
  if CurrentIndex >= ImagePaths.Count then
    CurrentIndex := 0;
  if Texts.Count > CurrentIndex then
  begin
    Text := StringReplace(Texts[CurrentIndex], '*', sLine-
Break, [rfReplaceAll]);
    MemoText.Text := StringReplace(Text, '*', '',
[rfReplaceAll]);
  end;
  ImagePicture.Picture.LoadFromFile(ImagePaths[Cur-
rentIndex]);
end;
destructor TForm10.Destroy;
begin
  ImagePaths.Free;
  Texts.Free;
  inherited Destroy;
end;
end.
unit Quest;
interface
uses
  Winapi.Windows, Winapi.Messages, System.SysUtils,
System.Variants, System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls,
Vcl.ExtCtrls,
  Vcl.Imaging.jpeg, Vcl.Imaging.pngimage;
type
  TQuestion = record
    ImageFileName: string;
    QuestionText: string;
    CorrectAnswer: Boolean;
  end;
  TForm12 = class(TForm)
    ImageQuestion: TImage;
    LabelQuestion: TLabel;
    ButtonYes: TButton;
    ButtonNo: TButton;
    LabelResult: TLabel;
    Image2: TImage;
    procedure FormCreate(Sender: TObject);
    procedure ButtonYesClick(Sender: TObject);
    procedure ButtonNoClick(Sender: TObject);
    procedure Image2Click(Sender: TObject);
  private
    Questions: array of TQuestion;
    CurrentQuestionIndex: Integer;
    CorrectAnswersCount: Integer;
    procedure DisplayQuestion(QuestionIndex: Integer);
    procedure ShowResult;
  end;
var
  Form12: TForm12;
implementation
{$R *.dfm}
uses Menu;
procedure TForm12.FormCreate(Sender: TObject);
var
  FileLines: TStringList;
  i: Integer;
begin
  FileLines := TStringList.Create;
  try
    FileLines.LoadFromFile(ExtractFilePath(ParamStr(0))
+'quest.txt', TEncoding.UTF8);
    SetLength(Questions, FileLines.Count div 3);
    for i := 0 to Length(Questions) - 1 do
```

```
    begin
      Questions[i].ImageFileName := FileLines[i * 3];
      Questions[i].QuestionText := FileLines[i * 3 + 1];
      Questions[i].CorrectAnswer := StrTo-
Bool(FileLines[i * 3 + 2]);
    end;
    CurrentQuestionIndex := 0;
    CorrectAnswersCount := 0;
    DisplayQuestion(CurrentQuestionIndex);
  finally
    FileLines.Free;
  end;
end;
procedure TForm12.Image2Click(Sender: TObject);
begin
Form12.Close;
Form2.Show;
    CurrentQuestionIndex := 0;
    CorrectAnswersCount := 0;
    DisplayQuestion(CurrentQuestionIndex);
    LabelResult.Caption :='';
    ButtonYes.Enabled := True;
    ButtonNo.Enabled := True;


end;


procedure TForm12.DisplayQuestion(QuestionIndex: In-
teger);
begin
  ImageQuestion.Picture.LoadFromFile(Questions[Ques-
tionIndex].ImageFileName);
  LabelQuestion.Caption := Questions[QuestionIn-
dex].QuestionText;
end;
procedure TForm12.ButtonYesClick(Sender: TObject);
begin
  if Questions[CurrentQuestionIndex].CorrectAnswer
then
    Inc(CorrectAnswersCount);
  Inc(CurrentQuestionIndex);
  if CurrentQuestionIndex < Length(Questions) then
    DisplayQuestion(CurrentQuestionIndex)
  else
    ShowResult;
end;
procedure TForm12.ButtonNoClick(Sender: TObject);
begin
  if not Questions[CurrentQuestionIndex].CorrectAnswer
then
    Inc(CorrectAnswersCount);
  Inc(CurrentQuestionIndex);
  if CurrentQuestionIndex < Length(Questions) then
    DisplayQuestion(CurrentQuestionIndex)
  else
    ShowResult;
end;
procedure TForm12.ShowResult;
var
  Score: Double;
begin
  Score := (CorrectAnswersCount / Length(Questions)) *
100;
  LabelResult.Caption := Format('Тест завершен!
Правильных ответов: %d из %d (%.2f%%)', [Cor-
rectAnswersCount, Length(Questions), Score]);
  ButtonYes.Enabled := False;
  ButtonNo.Enabled := False;
end;
end.
```