АННОТАЦИЯ

В бакалаврской работе разработано электронное приложение, с помощью которого обеспечено изучение практической части дисциплины.

Ключевые слова: электронное обучающее средство, электронный практикум, приложение на языке Pascal, среда разработки Delphi-7.

Бакалаврская работа содержит: пояснительную записку (листов, рисунков, таблиц); список из 17 литературных источников и 13 Интернет-ресурсов; флэш-накопитель с презентацией бакалаврской работы на слайдах и PDF-файлом текста пояснительной записки.

РАЗРАБОТКА ЭЛЕКТРОННОГО ПРАКТИКУМА ПО ДИСЦИПЛИНЕ "ПРОЦЕДУРНЫЕ ЯЗЫКИ ПРОГРАММИРОВАНИЯ".

В бакалаврской работе изложены следующие вопросы: проанализировано создание электронных обучающих ресурсов, поставлена задача на разработку электронного практикума дисциплины, составлен алгоритм и текст программы обучающего средства, проведено его тестирование и составлена инструкция по его использованию.

					СКФ МТУСИ 230101.65. 09204 ПЗ		
Изм	*Лист*	*№ документа*	*Подп.*	*Дата*			
Разработал						*Лит.*	*Лист*	*Листов*
Проверил							*5*	
Рецензент								
Н. контр								
Утвердил		*Соколов С.В.*				***каф. «СПОИ» гр. ВМ-51 5155555161***		

СОДЕРЖАНИЕ

Введение……..…6

1. Анализ современных реализаций электронных обучающих ресурсов. Постановка задачи ……..……..
2. Разработка схемы алгоритма лабораторного практикума ….……
3. Разработка программы электронного лабораторного практикума ……
4. Разработка программы реализации приложения на выбранном языке программирования……
5. Проведение тестов и отладка лабораторного практикума ……
6. Разработка инструкций по использованию материалов лабораторного практикума ……

Заключение…….....

Список используемых источников…… …….……

Приложения…….……..

ВВЕДЕНИЕ

Современная система образования все активнее использует информационные технологии и компьютерные телекоммуникации. Особенно динамично развивается система дистанционного образования, чему способствует ряд факторов, и прежде всего - оснащение образовательных учреждений мощной компьютерной техникой и развитие сообщества сетей Интернет [1, 2]. Электронные учебники могут использоваться как в целях самообразования, так и в качестве составной части дистанционного образования.

Лекционно-семинарская форма обучения давно потеряла свою эффективность - практика доказала, что почти 50% учебного времени тратится впустую. Изучая актуальные публикации [1-3], можно выделить следующий важный аспект: преподаватель выступает не в роли распространителя информации (как это традиционно принято), а в роли консультанта, советчика, иногда даже коллеги обучаемого.

Это дает некоторые положительные моменты: студенты активно участвуют в процессе обучения, приучаются мыслить самостоятельно, выдвигать свои точки зрения, моделировать реальные ситуации. Основными достоинствами дистанционного обучения, по мнению участников международной конференции, посвященной проблемам образования, являются [3]:

- самостоятельная познавательная деятельность обучаемых;
- выбор обучаемыми места, времени и продолжительности обучения;
- целостное сенсорное восприятие за счет «более естественных для человеческого восприятия трехмерных моделей»;
- исключается стадия «домысливания и достраивания» в восприятии моделей, что значительно ускоряет процессы освоения нового материала;

- возможность приобретения знаний недоступными реальном мире способами;
- возможность визуальной, голосовой, текстовой невербальной
- коммуникации;
- учет индивидуальных особенностей восприятия обработки
- информации человеком;
- реалистичность представления поведения объектов процессов;
- выполнение экспериментов, которые невозможно выполнить
- реальных условиях.

Очевидно, что основой такого вида образования является самостоятельная работа обучаемого с материалами выбранной дисциплины. В этом отношении, конечно, первостепенную роль играют средства обучения, т.е. то с помощью чего он будет осваивать новые знания и навыки.

Как правило, в дистанционной форме обучения применяются электронные учебники.

Достоинствами этих учебников, на мой взгляд, являются: во-первых, их мобильность, во-вторых, доступность связи с развитием компьютерных сетей, в-третьих, адекватность уровню развития современных научных знаний.

С другой стороны, создание электронных учебников способствует также решению и такой проблемы, как постоянное обновление информационного материала. В них также может содержаться большое количество упражнений и примеров, подробно иллюстрироваться в динамике различные виды информации. Кроме того, при помощи электронных учебников осуществляется контроль знаний - компьютерное тестирование.

В настоящее время разработано достаточное количество различных электронных ресурсов: от простого текста в электронном виде до целых сетевых комплексов, включающих несколько направлений подготовки специалистов (и не только в образовании, а и в бизнесе) [2]. Однако, есть то, что остается одинаковым для всех этих представителей электронных средств подготовки и обуче-

ния – это то, что они ориентированы на такой вид работы, как самостоятельная работа с предоставленными материалами [2, 3].

Так, в материалах уже упомянутой конференции [3] сделан конкретный вывод: « …обучающиеся в системе дистанционного обучения должны владеть основами методики техники самостоятельной работы, самостоятельного приобретения, пополнения знаний, обладать навыками работы со средствами современных информационных технологий».

Таким образом, становится чрезвычайно важным иметь при самостоятельной работе пособие электронного типа, которое бы обеспечивало эффективное (с точки зрения затрачиваемого времени и получаемых навыков) изучение дисциплины. Здесь, необходимо отдельно подчеркнуть, что, в конечном итоге, обучаемого всегда интересует то, что он будет уметь делать (т.е. практические навыки) в результате изучения предложенного курса, по сравнению с тем, что он просто будет знать (т.е. теоретические знания). Это очень характерно, особенно, в специальных дисциплинах, связанных с практическими умениями по программированию.

Поэтому, цель бакалаврской работы: разработка электронного практикума по дисциплине "Процедурные языки программирования"- является, безусловно, актуальной.

Развитие информационных технологий предоставило новую, уникальную возможность проведения занятий - внедрение дистанционной формы обучения [2]. Это, во-первых, как подчеркивали выше, позволяет самому обучаемому выбрать и время и место для обучения, во вторых, дает возможность получить образование лицам, лишенным получить традиционное образование в силу тех или иных причин, в третьих, использовать в обучении новые информационные технологии, в четвертых, в определенной степени сокращает расходы на обучение. С другой стороны, дистанционное образование усиливает возможности индивидуализации обучения.

Практика использования электронных учебников показала, что обучаемые самостоятельно качественно усваивают изложенный материал, о чем свидетельствуют результаты проводимого тестирования.

Таким образом, развитие информационных технологий дает широкую возможность для изобретения новых методов, методик в образовании и тем самым повысить его качество, а выбранная тема бакалаврской работы актуальна.

1 Анализ современных реализаций электронных обучающих ресурсов. Постановка задачи

На сегодняшний день, как уже указывалось, разработано и продолжает разрабатываться множество электронных средств обучения. Уже давно есть необходимость в стандартизации требований, предъявляемых к этому виду обучающих ресурсов. В каждой организации, где разрабатываются такие ресурсы, имеются ряд документов, которые регламентируют этот вид деятельности [4-9].

В результате проведенного анализа выяснилось, что сейчас предлагаются к использованию различные электронные ресурсы обучения [3, 9]. Основными из них являются:

- Электронное пособие (ЭП);
- Электронная инструментальная среда (программа);
- Электронная образовательная среда учебного заведения;
- Компьютерные модели, конструкторы и тренажеры;
- Электронный лабораторный практикум.

Каждый из перечисленных ресурсов направлен на решение конкретных задач.

Таким образом, с точки зрения оснащенности и методов взаимодействия обучаемого с пособием существующие учебные пособия можно разделить на 4 группы.

Первая группа. Отсканированные или набранные в электронном виде пособия. Т.е., это фактически, переведенные тексты учебников, практикумов, заданий и т.п. в электронный вид.

Достоинствами такого варианта представления материалов является, прежде всего, простота их изготовления и использования. Недостатки таких ресур-

сов налицо: отсутствие навигации даже в пределах выбранной книги или работы, а также, как правило, большой объем текста не всегда удобный для изучения. Полная же идентичность по содержанию с печатным изданием еще раз говорит о простом представлении материала.

Эта группа ресурсов сейчас активно создается в библиотеках и хранилищах книг [10, 11].

Вторая группа. В основном повторяют по структуре или являются книгами (учебниками, пособиями), которые перевели в электронный вид с минимальным использованием средств автоматизации – ссылок, расставленных по тексту.

Из достоинств можно отметить простоту их производства (обычно это HTML формат файлов с возможностью их просмотра в любом браузере) и в использовании – просмотр текста и переходы по намеченным ссылкам. Благодаря применению гипертекста, возможно, значительно упростить навигацию по учебнику. Так же обладает простотой транспортировки и тиражирования.

Однако, в данном виде электронный ресурс только повторяет обычный учебник, приобретя простоту в использовании.

Третья группа. Мультимедийный учебник может включать в себя дополнительные функции такие, как изображения, аудио-видео записи. Однако, мультимедийные функции играют дополнительную роль, по отношению к содержанию пособия.

Четвёртая группа. Интерактивное учебное пособие имеет ряд преимуществ по сравнению с обычным. Благодаря интерактивности повышается качество усвоения материала и облегчается работа преподавателя. Так же особенностью интерактивного учебного пособия является индивидуальный подход к обучаемому. Для каждого обучаемого может быть сформирован свой учебный план из общего курса.

Классификацию таких средств обучения можно продолжать по различным признакам и их наберется более десятка.

Однако, в настоящее время утвердилась определенная типологическая модель системы учебных изданий для ВУЗов, которая включает четыре группы

изданий, дифференцированных по функциональному признаку, определяющему их значение и место в учебном процессе [11]:

- программно-методические (учебные планы и учебные программы);
- учебно-методические (методические указания, руководства, содержащие материалы по методике преподавания учебной дисциплины, изучения курса, выполнению курсовых и дипломных работ);
- обучающие (учебники, учебные пособия, тексты лекций, конспекты лекций) или коротко – электронные пособия;
- вспомогательные (практикумы, сборники задач и упражнений, хрестоматии, книги для чтения).

Под электронным пособием будем понимать электронное издание частично (полностью) заменяющее или дополняющее учебник [10]. Электронное пособие, в отличие от учебника, кроме трех обязательных частей, присутствующих в учебнике: классического предметного содержания, упражнений для закрепления материала и контроля знаний, предполагает наличие определенной методики, разрабатываемой в рамках педагогического подхода: проблемного, проектного, контекстного, эвристического и др. подходов. Для эксплуатации ЭП достаточно провести экспертизу на уровне учебного заведения. Экспертиза проводится при наличии внешней рецензии, данной данную работодателем или главным специалистом региона. После проведения экспертизы Э.П. утверждается на методическом (ученом) совете образовательного учреждения.

Анализ существующих ЭП говорит о том, что их реализации совершенно различны [11]. Так, по структуре они подразделяются на:

- однотомное электронное издание - электронное издание, выпущенное на одном машиночитаемом носителе;
- многотомное электронное издание - электронное издание, состоящее из двух или более пронумерованных частей, каждая из которых представлена на самостоятельном машиночитаемом носителе, представляющее собой единое целое по содержанию и оформлению;

- электронная серия - серийное электронное издание, включающее совокупность томов, объединенных общностью замысла, тематики, целевым назначением, выходящих в однотипном оформлении.

Среди ЭП по структуре выполненных в виде моноизданий по методу создания выделяются пособия, имеющие управляющую оболочку и без нее.

Пособия без управляющей оболочки, как правило, разрабатываются на основе HTML документов, в которых собрана исходная информация [12]. Такие пособия это фактически электронные издания, снабженные гипертекстовыми ссылками. Открываются они в интернет – браузере. Такого рода пособий сейчас существует великое множество, каждое из которых отличается тем, что уникальным образом снабжено основным их достоинством – гиперссылками. Отсутствие какой-нибудь системы по упорядочению таких ссылок, зачастую путает их и вносит в процесс перехода по ним неразбериху. При этом, основным положительным качеством таких обучающих ресурсов является, конечно их простота структуры и использования. В качестве недостатков можно выделить малую их технлогичность, т.е. процесс создания и модификации является трудоемким, а так же ограниченную информативность.

К такого рода разработкам можно также отнести электронный учебник в формате электронной книги (PDF, DJVU, FB3, ePub, RTF) может воспроизводиться на таких устройствах, как электронные книги (ридеры), планшеты. Пользователь такого электронного учебника может выполнять индивидуальные настройки: шрифта (выбор гарнитуры шрифта, дискретное масштабирование шрифта, выбор цвета шрифта); режимов отображения контента (черный текст на белой подложке, белый на черной и пр.), яркости свечения экрана мобильного устройства.

Для решения задач, пользователю электронного учебника доступны следующие сервисы: поиск информации (полнотекстовый поиск, по гиперссылками, по оглавлению, переход по закладкам, переход по заметкам); работа с текстом (создание закладок, создание заметок в виде текстовых блоков, создание текстовых блоков, выделение текста подчеркиванием, маркирование текста разны-

ми цветами); печать заметок; отправка заметки или выделенного фрагмента содержания по электронной почте.

Используются следующие формы представления информации: текст с возможностью выбора шрифта и изменения его размера и цвета; статическая растровая иллюстрация с возможностью увеличения на весь экран; статическая векторная иллюстрация (для формата PDF).

В отличие от этого пособия, снабженные управляющей оболочкой, в настоящее время создаются в программных средах, предназначенных для разработки электронных пособий, а также документации различного рода. В качестве примера можно привести следующие среды разработки: Dreamweaver, Flash MX, 3D Studio MAX, SunRav [13]. Такие пособия, обычно, снабжаются высококачественными графическими, а также аудио- и видео- материалами, поясняющими излагаемую информацию. Безусловно, такие разработки удобны в использовании, информативны, и как правило, выполняются для совокупности обучающих курсов или направлений в силу своей высокой стоимости. При этом, так же отметим, что исходные обучающие материалы как бы «вшиты» в модуль (том) пособия (курса) и дальнейшая их модификация требует изменения кода всего приложения в целом, что является неудобным.

Такие типовые интерактивные мультимедийные электронные учебники также могут создаваться с помощью стандартного инструментального средства - редактора электронных учебников [14]. Возможности инструментального средства позволяют создавать учебники, превосходящие по своим потребительским свойствам электронные учебники в формате электронной книги.

Мобильными устройствами для воспроизведения таких электронных учебников являются планшеты, обеспечивающие возможность использования для представления информации не только текста и статических, но и динамических иллюстраций (видео, анимация) с возможностью управления их воспроизведением (запустить проигрывание, остановить, перемотать назад, пауза) и увеличением на весь экран. Возможна реализация более широкого спектра тестовых заданий, в том числе на установление соответствий, на установление последовательностей.

Одной из востребованных сред разработки ЭП является XML редактор [15]. Язык XML используется для хранения структурированных данных и обмена информацией между программами. Он представляет собой достаточно удобное средство для создания электронных пособий, т.к. по внешнему оформления среды редактирования очень похож на MS Word привычный для множества пользователей. XML - формирующийся новый стандарт под названием Extensible Markup Language. В отличие от HTML, XML не является приложением SGML. Это набор простых условий для применения SGML. В свою очередь, HTML - HyperText Markup Language - правила (метод) создания гипермедиа-ресурсов для публикации в сети Интернет [16]. Является приложением языка SGML. Большинство HTML-браузеров не поддерживает некоторые конструкции SGML, однако программы для разработки SGML-документов могут производить хорошие HTML-документы. Это способ создания графики без использования компилятора. Это обычный текстовый файл, должен быть в DOS – формате.

Шрифты: Windows, ANSI. Их понимает и обрабатывает любой браузер (Internet Explorer, Netscape). Любой браузер – интерпритатор, исполняюший последовательно файл и исполняет его. HTML быстро развивается и не все браузеры одинаково обрабатывают эти файлы. Сейчас все ориентируются на Internet Explorer 7.0. Язык содержит перечень тегов, один тег – команда интерпритатору, но она пишется исходя из общих правил записи тегов.
Все веб-сайты состоят из нескольких HTML-файлов и нескольких графических файлов.

Таким образом, из проведенного анализа следует, что при создании гипермедиа-средств обучения чаще всего используются следующие языки и инструменты:

- язык разметки гипертекста (HTML)- стандартный язык, используемый в сети Интернет для создания, форматирования и демонстрации информационных гипермедиа-страниц;
- язык Java - специализированный объектно-ориентированный язык программирования, аналогичный языку C++. Данный язык был раз-

работан специально для использования интерактивной графики и анимации в ресурсах сети Интернет. Многие готовые приложения (Java applets) доступны в сети Интернет, и их можно выгрузить на компьютер пользователя для дальнейшего использования при создании собственных информационных сетевых и несетевых электронных средств обучения;

- язык VRML (Virtual Reality Modeling Language) позволяет создавать и размещать в сети объемные трехмерные объекты, создающие иллюзию реального объекта намного сильнее, чем простые анимации. Подобные трехмерные объекты в зависимости от их "объема" принято называть "виртуальными комнатами", "виртуальными галереями" и "мирами";
- CGI (Common Gateway Interface) - по сути является не языком программирования, а спецификацией, описывающей правила сбора информации и создания баз данных.

Становится понятным, что указанные среды разработки таких пособий обладают своими привлекательными сторонами и недостатками. В каждом конкретном случае необходимо подходить творчески к выбору средств создания конкретных обучающих пособий в электронном виде. Такой подход позволяет избежать избыточности созданных приложений и одновременно преодолеть негативные стороны указанных разработок.

В силу того, что электронное пособие, как отмечалось выше, должно обеспечивать:

- расширение и углубление изучаемой предметной области за счет экономии учебного времени;
- расширение сферы самостоятельной деятельности обучаемых за счет возможности организации разнообразных видов учебной деятельности;
- индивидуализацию и дифференциацию процесса обучения за счет реализации возможностей интерактивного диалога, самостоятельно-

го выбора режима учебной деятельности и организационных форм обучения;

- вооружение обучаемого стратегией усвоения учебного материала или решения задач определенного класса за счет реализации множества примеров;
- формирование информационной культуры, компоненты культуры индивида, члена информационного общества, за счет осуществления информационно-учебной деятельности, работы с объектно-ориентированными программными средствами и системами;
- повышение мотивации обучения за счет компьютерной визуализации изучаемых объектов, явлений, управления изучаемыми объектами, ситуацией, возможности самостоятельного выбора форм и методов обучения, вкрапления игровых ситуаций.

Поэтому, в бакалаврской работе ставится задача разработать электронный практикум по дисциплине "Процедурные языки программирования".

Требования, предъявляемые к практикуму:

1. Практикум должен быть выполнен в электронном виде по следующей структуре: один запускающий модуль плюс директории, в которых хранятся обучающие материалы (количество директорий должно соответствовать количеству различных видов обучающих материалов, например, лекции, лабораторные работы и т.п.);
2. Его содержание должно включать материалы лабораторных работ и практических занятий согласно учебной программе по дисциплине, а также справочный лекционный материал;
3. Текстовое наполнение практикума должно быть организовано по принципу гипертекста с переходами по гиперссылкам;
4. Практикум должен быть снабжен понятным для освоения и изучения материала интерфейсом (удобный для пользователя экранный дизайн с организацией простого и понятного диалога, а также управляющие элементы должны быть выделены таким образом, чтобы пользователь мог легко обнаруживать активные зоны экрана);

5. Система навигации между модулями должна соответствовать структуре основных разделов, обеспечивать переходы к любым информационным ресурсам за минимальное число действий пользователя (не более 3-х кликов мыши).
6. Элементы навигации должны, как правило, снабжаться текстовыми подсказками, облегчающими понимание их работы;
7. Должна быть учтена возможность модификации практикума методом внесения изменений и дополнений в тексты учебного материала.

2 Разработка схемы алгоритма лабораторного практикума

Для того, чтобы решить поставленную выше задачу необходимо разработать блок-схему практикума. Согласно указанной структуре он должен иметь управляющую оболочку и три папки – с лекциями, лабораторными работами и с практическими занятиями. Поэтому, логично предположить, что практикум будет реализован в одной из систем разработки, содержащей средства создания интерфейса. Для этого подходят такие среды, как Delphi [17-23], C++ [24] или Java [25]. Одну из них выберем и обоснуем позже.

Таким образом, на главной форме должны содержаться списки лекций, лабораторных работ и практических занятий, а так же -область, где можно было бы открыть тексты каждого из них. Блок схема электронного практикума показана на рисунке 2.1.

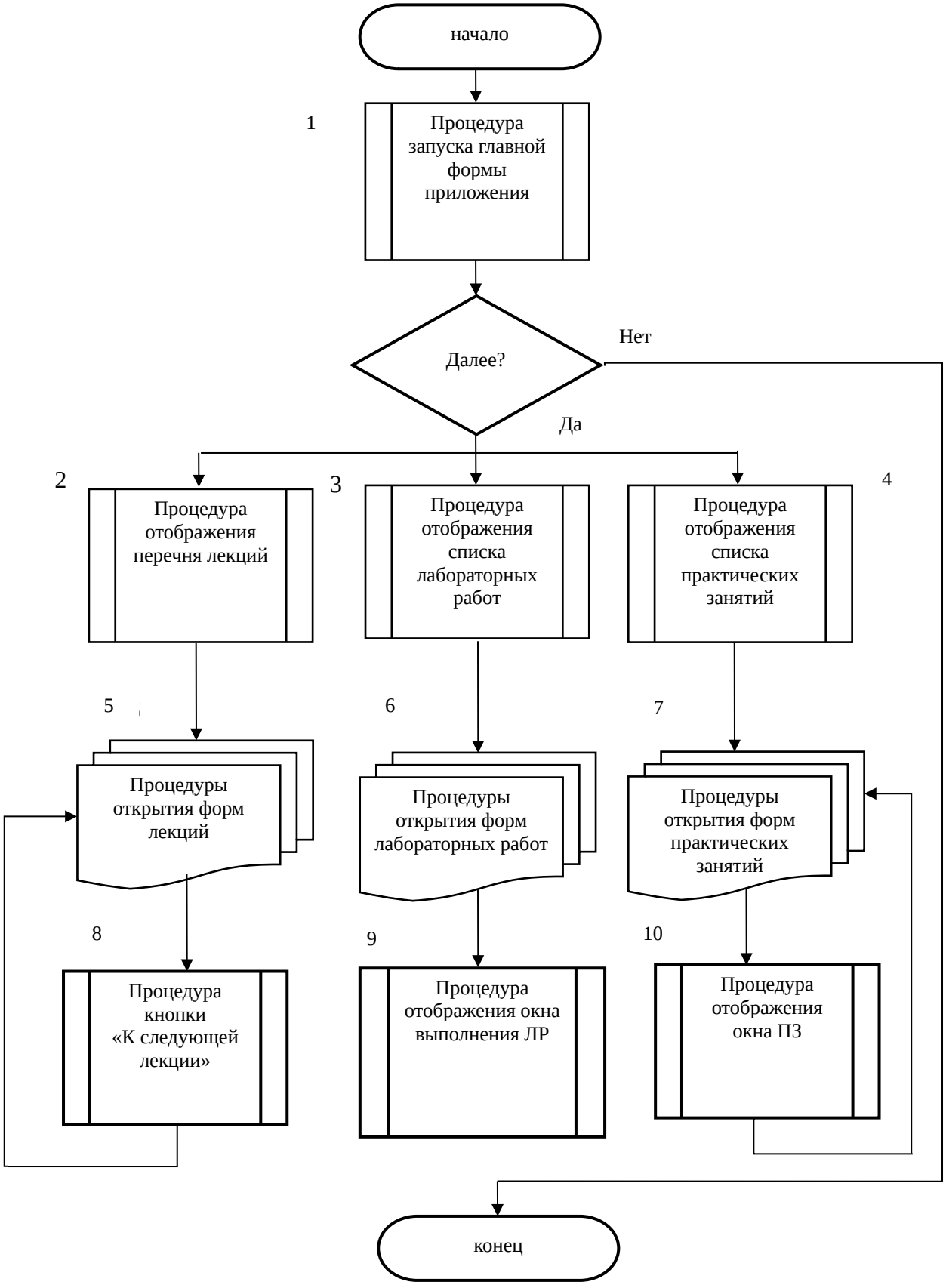

Рисунок 2.1 - Блок-схема алгоритма практикума

Создаваемое приложение работает следующим образом.

Первоначально, передается управление процедуре запускающей основную форму (обозначено 1 на рис. 2.1). Совместно с ней запускаются несколько про-

цедур: открываются списки лекций, лабораторных работ и практических занятий (блоки 2-4 рис. 2.1).

При этом, существует возможность просто закрыть окно приложения и электронный практикум закончит свою работу (ветвь «Нет» выбора по условию продолжения работы рис. 2.1).

Рассмотрим логику работы основных процедур.

Процедура выбора и отображения текста лекции. Ее алгоритм представлен на рис. 2.2.

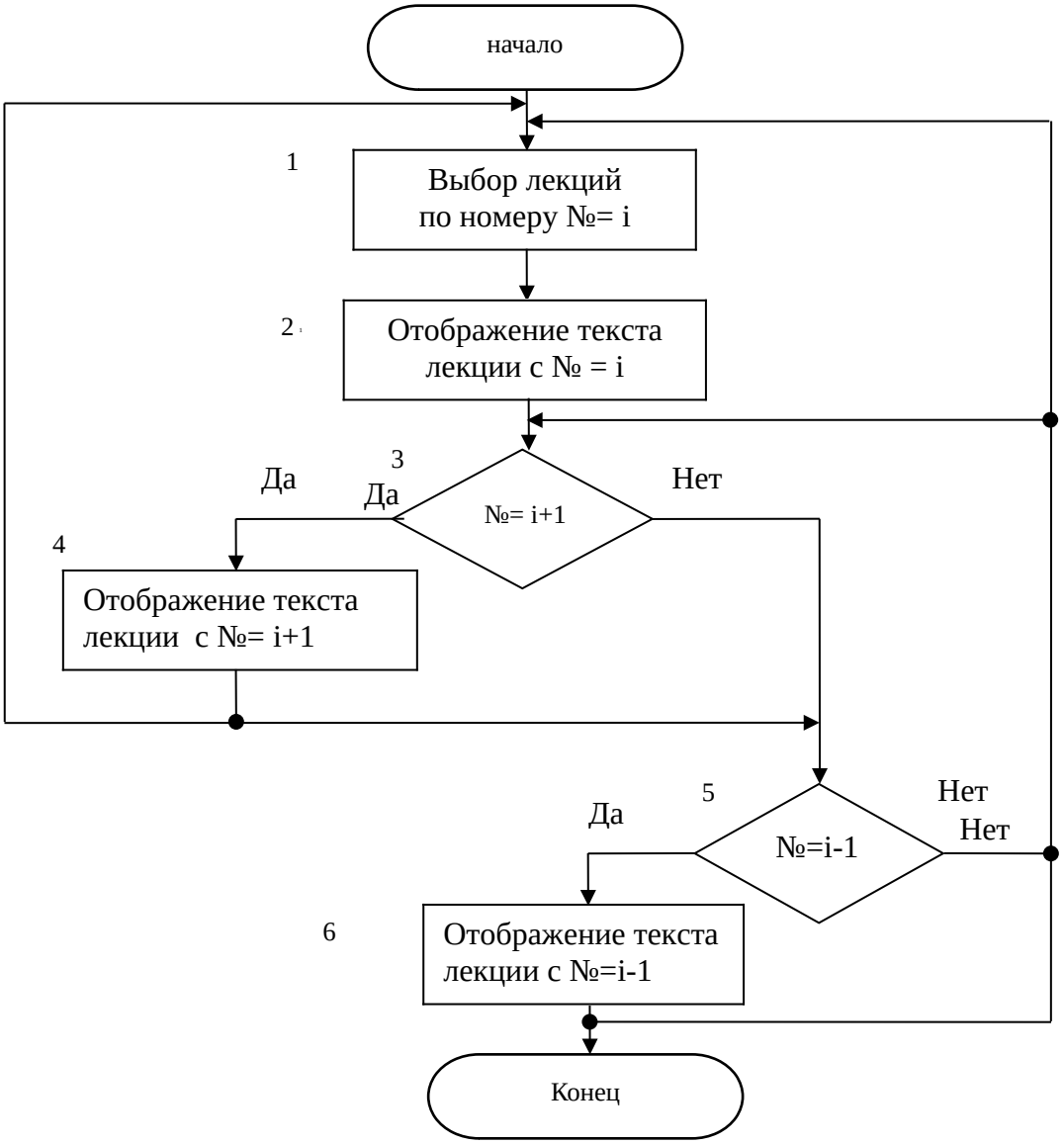

Рисунок 2.2 – Блок-схема алгоритма процедуры выбора лекций

Процедура выполняется следующим образом (рис. 2.2).

В блоке 1 есть возможность выбирать лекцию по ее номеру №=i из списка лекций (с 1-ой по 19-ю), который, как указано выше, отображается на главной форме практикума (рис. 5.1).

Далее, в блоке 2 (рис. 5.2) происходит отображение ее текста и в завершение его появляется кнопка перехода к следующей лекции. Блок 3 обеспечивает выбор по условию перехода. Если эта кнопка нажата, то выполняется переход на следующую лекцию или можно вернуться к предыдущей (блок 5 рис. 5.2). Т.е., если «Да», то идем в следующую лекцию (блок 4), а если «Нет», то идем в предыдущую лекцию (блок 5).

С помощью блока 4 (рис. 2.2) отображается текст уже следующей выбранной лекции и есть возможность вернуться к выбору лекции по ее номеру (блок 1 рис. 2.2). При этом, если есть необходимость, то можно вернуться в предыдущую лекцию (блок 5 рис. 2.2).

Далее, в блоке 5 (рис. 2.2) можно перейти в следующую лекцию и в предыдущую. Логика работает так: если выбрано «Да», то переход к следующей лекции (к блоку 6 рис. 2.2), если - «Нет», то переход к предыдущей лекции (к блоку 3 рис. 2.2), либо – к выбору лекции по ее номеру (к блоку 1 рис. 2.2).

Блок 6, таким образом, отображает текст следующей лекции и после этого можно вернуться в предыдущую лекцию (к блоку 3 рис. 2.2), либо к выбору лекции по ее номеру (к блоку 1 рис. 2.2), или а просто выйти из программы.

Процедура выбора лабораторных работ

Блок-схема процедуры показана на рисунке 2.3.

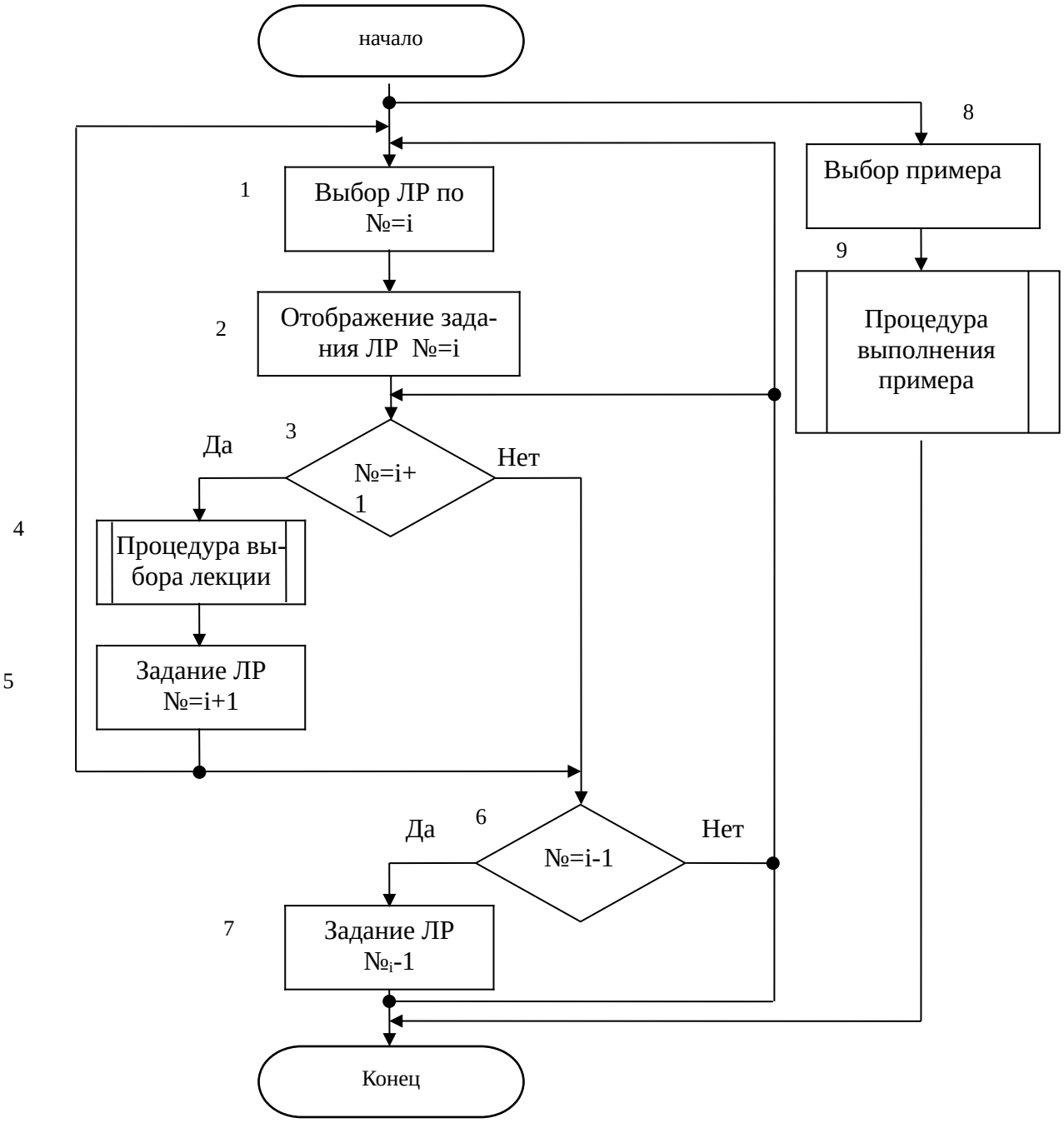

Рисунок 2.3 – Блок-схема алгоритма выбора лабораторных работ

Согласно рисунка 2.3 лабораторные работы выбираются следующим образом.

В блоке 1 выбираем лабораторную работу №=i (с 1-й по 10-ю лабораторную работу).

В блоке 2 выбираем задание по лабораторной работе.

В блоке 3 выполняется переход к следующей лабораторной работе или к - предыдущей, или можно выбрать отображение лекции (процедура 4 рис. 2.3). Если «Да» (блок 3 рис. 2.3), то выбираем лекцию или переходим в следующую

лабораторную работу (блок 5 рис. 2.3). Если - «Нет», то переходим в предыдущую лабораторную работу (блок 6 рис. 2.3).

В блоке 5 выбираем задание по ЛР и можем вернуться в выбор ЛР по номеру (блок 1 рис. 2.3), или - перейти в предыдущую лабораторную работу (блок 6 рис. 2.3).

В блоке 6 можно перейти в следующую, или в предыдущую лабораторную работу. Если выбрано «Да», то - в следующую (блок 7 рис. 2.3), а если - «Нет», то - в предыдущую (блок 3 рис. 2.3), либо к выбору работы по ее номеру (блок 1 рис. 2.3).

В блоке 7 выбираем задание лабораторной работы, или возможно вернуться к предыдущей работе (блок 3 рис. 2.3), либо - к выбору ЛР ее номеру (блок 1 рис. 2.3), или - выйти из программы.

В блоке 8 выбираем пример работы, а процедура 9 запускает пример данной лабораторной работы на выполнение и можно наблюдать результат.

Показанные на этом рисунке процедуры (4 – отображения текста лекции и 9 – решения примера лабораторной работы) написаны по алгоритму, изображенному на рисунке 2.2, а так же, выполненному по конкретному примеру, соответственно.

Процедура выбора практического занятия

Блок-схема этой процедуры показана на рисунке 2.4.

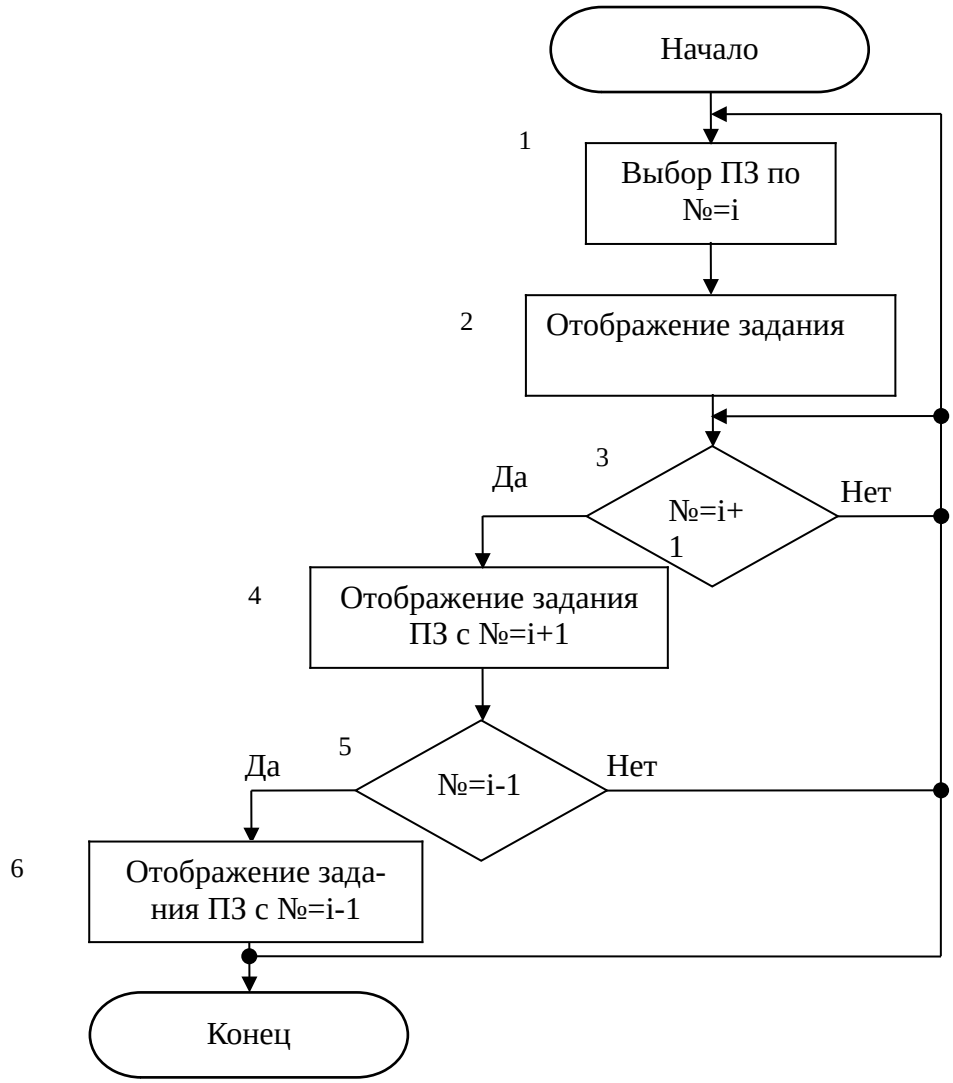

Рисунок 2.4 – Блок-схема алгоритма процедуры выбора практических занятий

В этой процедуре практические занятия (ПЗ) выбираются так.

В блоке 1 выбираем практическое задание по его номеру №=i. В блоке 2 выбираем задание этого ПЗ.

В блоке 3 выполняется переход в следующее практическое занятие или к выбору задания текущего ПЗ. Если «Да», то переход к следующему заданию этого ПЗ (блок 4 рис. 2.3), если «Нет», то выбираем следующее ПЗ (блок 1 рис. 2.3).

Блок 4 отображает текст нового выбранного ПЗ и можем вернуться к выбору еще одного ПЗ по его номеру (блок 1 рис. 2.3), или вернуться в предыдущее ПЗ (блок 5 рис. 2.3).

В блоке 5 можно перейти предыдущее ПЗ. Если «Да», то возврат в предыдущее ПЗ (блок 6), если «Нет», то переход к выбору ПЗ по его номеру (блок 1).

Блок 6 отображает текст выбранного задания предыдущего ПЗ и можно перейти к следующему ПЗ (блок 3), либо к выбору ПЗ по номеру (блок 1) или совсем выйти из программы.

Таким образом, основные алгоритмы практикума выполнены и в следующем разделе работы выберем среду разработки и составим его код.

3 Разработка программы электронного лабораторного практикума

Как правило, разработка любого программного продукта состоит из двух основных этапов – проектирования структуры его составных частей, и непосредственно его программирование, т.е. написание программного кода.

Опишем поэтапную разработку кода практикума.

3.1 Выбор среды программирования

Как было указано ранее, в качестве среды разработки электронного пособия можно выбрать: Delphi, C++ или Java. Дадим краткую характеристику этим языкам программирования и обоснуем выбор одного из них.

Краткая характеристика Borland Delphi-7

Delphi – одна из самых мощных систем, позволяющих на самом современном уровне создавать как отдельные прикладные программы Windows, так и разветвленные комплексы, предназначенные для работы в корпоративных сетях и в Интернет [19-23]. Это продукт, уникальным образом сочетающий высокопроизводительный компилятор, объектно-ориентированные средства визуального программирования и универсальный механизм доступа к базам данных.

Delphi – это система визуального объектно-ориентированного программирования, позволяющая решать множество задач, в частности:

1. Создавать законченные приложения для Windows самой различной направленности, от чисто вычислительных и логических, до графических и мультимедиа.

2. Быстро создавать профессионально выглядящий оконный интерфейс для любых приложений, написанных на любом языке. Такой интерфейс удовлетворяет всем требованиям Windows и автоматически настраивается на ту систему, которая установлена на компьютере пользователя, поскольку использует многие функции, процедуры, библиотеки Windows. Это, пожалуй, одно из самых весомых преимуществ этой системы разработки прикладных программ – интуитивно понятных и просто реализуемый интерфейс.

3. Создавать мощные системы работы с локальными и удаленными базами данных любых типов. При этом, имеются средства автономной отладки приложения с последующим выходом в сеть.

4. Создавать многозвенные распределенные приложения, основанные на различных технологиях.

5. Создавать приложения, которые управляют другими приложения, в частности, такими программами Microsoft Office, как Word, Excel и другими.

6. Создавать кросс-платформенные приложения, которые можно компилировать и эксплуатировать, как и в операционных системах (ОС) семейства Windows, так и в ОС Linux.

7. Создавать приложения различных классов для работы в Интернет.

8. Создавать профессиональные программы установки для приложений Windows, учитывающие всю специфику и все требования Windows.

9. И многое другое, включая создания отчетов, справочных систем, библиотек DLL, компонентов ActiveX.

Delphi 7 прекрасное средство создания приложений для Windows и достаточно быстро превращается в инструмент создания приложений для многозвенных распределенных кросс-платформенных корпоративных информационных систем.

Delphi используется для генерирования и вывода на печать сложных отчетов баз данных.

Delphi предоставляет возможность связываться из своего приложения с такими продуктами Microsoft, как Word, Excel, а также с Access.

Процесс разработки в Delphi предельно упрощен. В первую очередь это относится к созданию интерфейса, на который уходит до 80% времени разработки всей программы. Нужные компоненты помещаются на форму и настраиваются их свойства с помощью специального инструмента (ObjectInspector). С его помощью можно связать события этих компонентов (нажатие на кнопку, выбор мышью элемента в списке и так далее) с кодом его обработки. Причем, разработчик получает в свое распоряжение мощные средства отладки (вплоть до пошагового выполнения команд процессора), удобную контекстную справочную систему (в том числе и по Microsoft API), средства коллективной работы над проектом, всего просто не перечислить [3].

Язык в полной мере поддерживает все требования, предъявляемые к объектно-ориентированному языку программирования.

Таким образом, Delphi используется для создания как самых простых приложений, на разработку которых требуется 2-3 часа, так и серьезных корпоративных проектов, предназначенных для работы десятков и сотен пользователей. Причем, для этого можно использовать самые последние веяния в мире компьютерных технологий с минимальными затратами времени и сил.

После загрузки среды программирования Delphi, на экране появится главное окно.

Верхнюю часть окна занимает меню и панель инструментов. В левой части окна находятся ObjectTreeView и ObjectInspector. В центре находится форма разрабатываемого приложения и форма для внесения кода. Такая компоновка проста и понятна.

ObjectTreeView – служит для отображения всех объектов располо-женных на текущей форме разрабатываемого проекта.

ObjectInspector – служит для отображения и редактирование свойств компонентов расположенных на форме разрабатываемого приложения.

Проект в Delphi – это совокупность файлов, которые используются средой разработки для итоговой генерации программы.

Главное меню позволяет вызывать все инструменты, необходимые для работы с проектом.

Оно включает следующие разделы и функции:

File – содержит набор команд для работы с файлами, позволяет создавать новые проекты, добавлять новые файлы в проект на основе различных шаблонов, переименовывать файлы проекта, а также распечатывать их. Сюда же включена команда закрытия среды разработки;

Edit – здесь, в соответствии с названием, расположены команды, предназначенные для редактирования текста, удаления и перемещения его в буфер обмена, вставки текста из буфера и отмены операций редактировании.

Search – содержит набор команд для работы с текстом, его поиска и замены, причем и то и другое может производится как в одном файле, так и во всех файлах проекта, либо в любом каталоге и его подкаталогах;

View – под этим названием объединены команды вызова инструмента управления проектом, такие как инспектор объектов, конструктор форм, менеджер проектов и т.д.;

Project – предназначен для того чтобы добавлять и удалять модули проекта, сохранять проект, добавлять проекты в группу и убирать их из неё, компилировать как отдельные проекты, так и все проекты в группе, загружать файл самого проекта в редактор кода, а также вызывать диалог настройки свойств проекта;

Run – позволяет запускать проект на выполнение как под отладчиком, так и без него, конфигурировать строку параметров проекта при запуске, производить отладку, задать точки останова, осуществлять пошаговое выполнение кода, просматривать значения переменных и изменять их;

Component – здесь сосредоточены команды, предназначенные для установки новых компонентов и пакетов компонентов и создания новых компонентов и шаблонов компонентов;

Database – здесь сосредоточены команды управления базами дынных;

Tools – позволяет настраивать свойства рабочей среды Delphi и отладчика, произвести настройки репозитория, добавлять и удалять дополнительные утилиты, а так же команды запуска этих самых утилит;

Window – позволяет переключаться между окнами, если вы откроете

какой – либо модуль для редактирования в новом окне;

Help – объединяет команды вызова справочной системы Delphi и её настройки, а так же позволяет обратиться к Web-ресурсам компании Borland для получения дополнительной информации.

Имеется возможность вручную настроить панель инструментов, что является удобным в работе. Для этого нужно вызвать диалоговое окно по ссылке View-Toolbars-Customize.

Процесс компиляции состоит из двух этапов. На первом этапе выполняется проверка текста программы на отсутствие ошибок, на втором — генерируется исполняемая программа (exe-файл).

После ввода текста функции обработки события и сохранения проекта можно из меню Project выбрать команду Compile и выполнить компиляцию. Процесс и результат компиляции отражаются в диалоговом окне Compiling. В это окно компилятор выводит ошибки (Errors), предупреждений (warnings) и подсказок (Hints). Сами сообщения об ошибках, предупреждения и подсказки отображаются в нижней части окна редактора кода.

Очевидно, что среда разработки Delphi выполнена в виде интерактивного средства составления и редактирования приложений с простым созданием интерфейсов. В дополнении ко всему сказанному, ее элементы просто и без ошибок взаимодействуют с различными компонентами и программами разных операционных систем.

Достоинства

Простота, скорость и эффективность Delphi объясняют ее популярность.

Delphi имеет один из самых быстрых компиляторов, обеспечивающий, хороший объектный код.

Есть и другие достоинства: простота изучения ObjectPascal; облегчающие жизнь нововведения - вроде свойств (properties); программы, написанные на Delphi, не требуется снабжать дополнительными библиотеками (в отличие от связки C++/MFC). VCL предоставляет удобный, легко расширяемый объектно-

ориентированный интерфейс к Windows, что ни в коей мере не мешает программисту опускаться в самые глубины Windows API.

Создателям оригинальных компонентов это приходится делать довольно часто, в отличие от "просто программистов". Как было сказано выше, модель программирования в Delphi - компонентная, что позволяет пользоваться компонентами, написанными другими разработчиками, даже не имея их исходного кода и уж подавно не изучая его.

В Интернете есть огромное количество компонентов, значительная часть которых распространяется бесплатно. Применение компонентной модели приводит к тому, что довольно многое в поведении объектов программировать не нужно вообще, и многое, на что в других средах ушли бы недели, можно сделать за часы или даже минуты. Оно и понятно - это ведь RAD-среда.

К достоинствам можно отнести очень быстрый браузер классов и мгновенный вывод подсказки автозавершения кода (codecompletion). Если кратко - может все. Конечно, ObjectPascal накладывает определенные ограничения, но для тех вещей, для которых ее писали, Delphi подходит практически оптимально. Из понравившихся (но, разумеется, нестандартных) "улучшений", внесенных Borland в ObjectPascal, хотелось бы выделить свойства (properties) и перегружаемость процедур и функций (overloading).

Определенные неудобства при работе с низкоуровневыми функциями API может вызвать то, что стандартным языком для API все же является C, и именно на нем пишутся все новые SoftwareDevelopmentKit (SDK) и

заголовочные файлы к ним. Это ни в коем случае не означает, что вы не сможете работать с новыми SDK, просто вам придется написать несложный (но, возможно, объемный) код на Pascal, в котором определить интерфейсы данного SDK.

Недостатки

Главный недостаток (и одновременно достоинство) - статическое присоединение (linking) библиотеки VCL и компонентов к исполняемому файлу. VCL

можно линковать и динамически, но тогда с каждым своим приложением вам придется распространять еще и VCL, а это более 3 Мбайт. Однако если не увлекаться интерфейсными "наворотами" и использовать в программе минимально необходимое число компонентов, то исполняемый файл будет невелик.

Другой недостаток (и опять же достоинство) состоит в том, что в используемой в Delphi парадигме форм (Forms) вся информация о форме, включая свойства, настройки компонентов, значения по умолчанию, хранится в exe-файле, причем не оптимальным образом.

Анализ исходного кода VCL показывает, что при создании формы фактически происходит чуть ли не синтаксический разбор данных инициализации, что не может ее не замедлять.

Третий недостаток, который кто-нибудь тоже может назвать достоинством, - это ObjectPascal. Несмотря на простоту, эффективность и легкость в изучении, ему не хватает очень многих мощных средств C++.

Как следует из проведенного анализа, разрабатывать приложения в этой среде просто и удобно.

Язык программирования на платформе Microsoft C++

Из всех объектно-ориентированных языков C++ является наиболее широко употребительным. Язык Java, представляющий собой последнюю разработку в области объектно-ориентированных языков, лишен таких составляющих, как указатели, шаблоны и множественное наследование, что сделало его менее мощным и гибким по сравнению с C++ (синтаксис языка Java очень похож на синтаксис C++, поэтому знания относительно C++ с успехом могут быть применены при программировании на Java) [24].

Некоторые другие объектно-ориентированные языки, например, C#, также успешно развиваются, однако их распространение в значительной степени уступает C++.

До последнего времени язык C++ развивался вне рамок стандартизации. Это означало, что каждый производитель компиляторов по-своему реализовы-

вал отдельные нюансы языка. Тем не менее комитет по стандартам языка C++ организации ANSI/ISO разработал документ, ныне известный под названием Стандартного C++. (ANSI является сокращением от английского названия Американского Национального Института Стандартов, а ISO — от Международной Организации Стандартов.) Стандартный C++ включает в себя много дополнительных возможностей, например стандартную библиотеку шаблонов (STL).

Наиболее популярной средой разработки для C++ является продукт, совместно разработанный компаниями Microsoft и Borland и предназначенный для работы под управлением операционных систем Microsoft Windows. Это, несомненно, Microsoft Visual C++ и Borland C++ Builder.

В настоящее время, C++ включен в семейство продуктов Visual Studio – мощной интегрированной среды разработки приложений различного уровня сложности и назначения [25]. В этом пакете используется единая интегрированная среда разработки (IDE), состоящая из нескольких элементов: строки меню, панели инструментов, различных закрепленных или автоматически скрываемых окон инструментов в левой, нижней или правой областях, а также области редакторов. Набор доступных окон инструментов, меню и панелей инструментов зависит от типа проекта или файла, в котором выполняется разработка.

Однако, создавать интерфейсы приложений в этой среде программирования не всегда удобно. Приходиться переходить к разработкам с использованием C#. Все это в значительной мере усложняет составление электронных пособий и «утяжеляет», т.е. делает их код более длинным и объемным, а сами разработки – громоздкими.

Язык программирования Java

На сегодняшний день технология Java предоставляет средства для превращения статических Web-страниц в интерактивные динамические документы и для создания распределенных не зависящих от платформы приложений [26].

Можно выделить несколько преимуществ языка Java.

1) Одно из основных преимуществ языка Java — независимость от платформы, на которой выполняются программы: один и тот же код можно запускать под управлением операционных систем Windows, Solaris, Linux, и др. Это действительно необходимо, когда программы загружаются через Интернет для последующего выполнения под управлением разных операционных систем.

2) Другое преимущество заключается в том, что синтаксис языка Java похож на синтаксис языка C++, и программистам, знающим языки C и C++, его изучение не составляет труда.

3) Кроме того, Java — полностью объектно-ориентированный язык, даже в большей степени, чем C++. Все сущности в языке Java являются объектами, за исключением немногих основных типов (primitive types), например чисел (с помощью объектно-ориентированного программирования легко разрабатывать сложные проекты, оно заменило собой структурное программирование).

Принципиально важно, что разрабатывать программы, не содержащие ошибок, на языке Java легче, чем наязыке C++. Разработчики снабдили язык Java средствами, позволяющими исключить саму возможность создавать программы, в которых были бы скрыты наиболее распространенные ошибки. Для этого в языке Java сделано следующее:

- исключена возможность явного выделения и освобождения памяти.
- память в языке Java освобождается автоматически с помощью механизма сборки мусора. Программист гарантирован от ошибок, связанных с неправильным использованием памяти.
- введены истинные массивы и запрещена арифметика указателей. Теперь программисты в принципе не могут стереть данные из памяти вследствие неправильного использования указателей.
- исключена возможность перепутать оператор присваивания с оператором сравнения на равенство.
- исключено множественное наследование. Оно заменено новым понятием — интерфейсом, позаимствованным из языка Objective C.

- интерфейс дает программисту почти все, что тот может получить от множественного наследования, избегая при этом сложностей, возникающих при управлении иерархиями классов.

Характерные особенности языка Java

-Простой;

-Интерпретируемый;

-Распределенный;

-Надежный;

-Безопасный;

-Машино-независимый;

-Объектно-ориентированный;

-Высокопроизводительный;

-Многопоточный;

-Динамичный;

-Не зависящий от архитектуры компьютера.

Однако, при всех положительных качествах интерфейсы, написанные на Java характеризуются довольно трудным взаимодействием с компонентами приложений. Так, для разработки графических интерфейсов необходима отдельная среда программирования, которая для этого только и предназначена, например, Netbeans [27]. Это нарушает целостность разработки программного продукта, как в Delphi и является неудобным.

Таким образом, из приведенного обзора различных инструментов разработки электронных учебных пособий следует, что наиболее подходящей средой разработки, которая удовлетворяет требованиям, предъявляемым к создаваемому пособию (раздел 1) и в которой естественно и просто можно реализовать его алгоритмы (раздел 2) является Delphi.

3.2 Составление кода программы практикума в Delphi -7

Разработка программы, как отмечалось выше, включает следующие этапы:

определение ее назначения и области применения;

задание требований к программному продукту в целом;

описание входных данных для программы и результата их преобразования в выходные данные, а так же их вид и размещение.

Из материала пояснительной записки следует, что программа будет работать на основе файлов, в которых будут храниться данные задачи- материалы лекций, лабораторных работ и практических занятий. Управляющая оболочка, в свою очередь, должна обеспечить пользователю возможность работать с этими материалами, используя удобный графический интерфейс, т.е. чтобы пользователь имел возможность ознакомиться с содержанием этих материалов, а так же переходить по ссылкам между ними и пользоваться интерактивными примерами.

Одним из основных вопросов является выбор формата представления исходной информации по изучаемой дисциплине. Был выбран формат HTML файлов. Такое представление обучающего материала удобно по нескольким причинам. С одной стороны, для пользователя, подготавливающего исходные данные в пособие, не представляет особого труда сохранить исходники в этом формате из MS Word. С другой стороны, в пределах таких документов достаточно просто организовать перекрестные ссылки по материалу всей дисциплины. При этом, подготовительная работа сводится к минимуму и, в итоге, получается хорошо согласованное по дидактическим единицам, цельное пособие.

Поэтому, все материалы по дисциплине были переведены в HTML формат и размещены в трех директориях: Лекции, Лабораторные работы и Практические занятия (рис. 3.1).

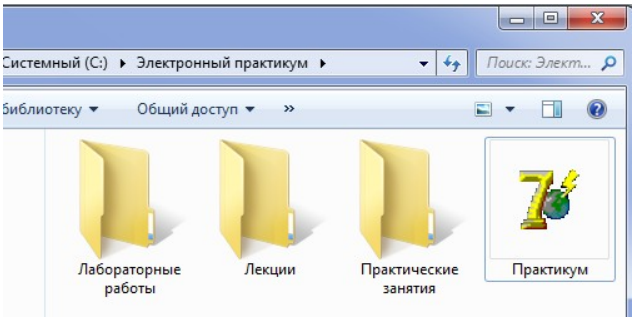

Рисунок 3.1 – Директория электронного практикума

Каждая из показанных директорий содержит текстовые файлы и отдельно вынесенные из них иллюстрации, как того требует выбранных формат данных.

В принципе, каждый документ, переведенный в WEB формат и снабженный гиперссылками, может быть открыт и связан с другими документами посредством WEB браузера. Однако, в этом случае теряется основное преимущество разрабатываемого ресурса – интерактивные примеры. Дело в том, что согласно заданию (раздел 1) в каждой лабораторной работе выполнена в качестве примера одна задача, которая снабжена тестовыми данными, блок-схемой и текстом программы. Вся эта информация должна появляться в отдельном окне приложения. Интерактивность этого окна в том, что в специально отведенных полях можно вносить исходные величины используемых в программе переменных и получать решение в on-line режиме. Все это становится невозможным, если составить обучающий ресурс просто в виде WEB – приложения. Поэтому, в работе используется сочетание управляющей оболочки, выполненной в среде Delphi -7, с исходниками учебно-методических материалов в HTML формате.

Используя схему алгоритма приложения (рис. 2.1) логично разрабатывать программу приложения в виде совокупности отдельных модулей. С другой стороны, такая архитектура следует из заявленных требований к интерфейсу приложения (раздел 1), которые реализуются в том, что должен видеть пользователь на основном окне приложения, а также определяют функционал возможных кнопок.

Основной модуль программы «Практикум» инициирует главную форму приложения (листинг 3.1).

Листинг 3.1 – Код основного модуля Практикум

```
program Praktikum;
uses
  Forms,
  Unit1 in 'Unit1.pas' {Form1},
  Unit2 in '..\1\Unit2.pas' {Form2},
```

```
  Unit3 in '..\1\Unit3.pas' {Form3},
  Unit4 in 'Unit4.pas' {Form4},
  Unit5 in 'Unit5.pas' {Form5},
  Unit6 in 'Unit6.pas' {Form6},
  Unit7 in 'Unit7.pas' {Form7},
  Unit8 in 'Unit8.pas' {Form8},
  Unit9 in 'Unit9.pas' {Form9},
  Unit10 in 'Unit10.pas' {Form10},
  Unit11 in 'Unit11.pas' {Form11};

{$R *.res}
```

Листинг 3.1 – Продолжение

```
begin
  Application.Initialize;
  Application.CreateForm(TForm1, Form1);
  Application.CreateForm(TForm2, Form2);
  Application.CreateForm(TForm3, Form3);
  Application.CreateForm(TForm4, Form4);
  Application.CreateForm(TForm5, Form5);
  Application.CreateForm(TForm6, Form6);
  Application.CreateForm(TForm7, Form7);
  Application.CreateForm(TForm8, Form8);
  Application.CreateForm(TForm9, Form9);
  Application.CreateForm(TForm10, Form10);
  Application.CreateForm(TForm11, Form11);
  Application.Run;
end.
```

Здесь открываются формы приложения, соответствующие его составу. Данный модуль составлен стандартным образом: через использование сервиса,

работающего с WEB файлами организовано считывание текстов структурных единиц учебно-методического комплекса (УМК) указанной дисциплины. В приведенном фрагменте этого модуля видна иллюстрация подключения этого сервиса (листинг 3.2).

Листинг 3.2 – Фрагмент кода модуля Unit 1

```
var
```

Листинг 3.2 – Продолжение

```
Form1: TForm1;

implementation

uses Unit2, Unit3, Unit4, Unit5, Unit6, Unit7, Unit8, Unit11, Unit10,
  Unit9;

{$R *.dfm}
{Процедура подключения Лекции 1}
procedure TForm1.Label3Click(Sender: TObject);
begin
WebBrowser1.Navigate(ExtractFilePath(Application.ExeName)+'лекции/
lek1.html');
Panel2.Visible:=true;
end;

{Процедура подключения Лабораторной работы 1}
procedure TForm1.Label22Click(Sender: TObject);
begin
WebBrowser1.Navigate(ExtractFilePath(Application.ExeName)+'лаборатор-
ные работы/lab1.html');
```

```
Panel2.Visible:=true;
end;
procedure TForm1.Label23Click(Sender: TObject);
begin
```

Листинг 3.2 – Продолжение

```
form2.showmodal;
end;

{Процедура подключения Практического занятия 1}
procedure TForm1.Label43Click(Sender: TObject);
begin
WebBrowser1.Navigate(ExtractFilePath(Application.ExeName)+'практика/
practik1.html');
Panel2.Visible:=true;
end;
```

Аналогично в практикум включены все материалы, а полный текст программы представлен в Приложении А.

Формы, соответствующие примерам использующимся в лабораторных работах, являются стандартными для седы программирования и содержат кнопки и окна для ввода значений (листинг 3.3). Здесь представлены характерные для программы функции и процедуры, оформленные на примере выполнения Лабораторной работы №10.

Листинг 3.3 Текст модуля выполнения примера Лабораторной работы №10

```
unit Unit11;
interface
uses
```

Листинг 3.3 - Продолжение

```
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, StdCtrls, ExtCtrls, jpeg;
type
TForm11 = class(TForm)
Panel1: TPanel;
Panel2: TPanel;
Memo1: TMemo;
Edit1: TEdit;
Edit2: TEdit;
Edit3: TEdit;
Button1: TButton;
Button2: TButton;
Button3: TButton;
Memo2: TMemo;
Image1: TImage;
Label1: TLabel;
Label2: TLabel;
Label3: TLabel;
Label4: TLabel;
Label5: TLabel;
Label6: TLabel;
Label7: TLabel;
procedure Button1Click(Sender: TObject);
procedure Button2Click(Sender: TObject);
```

Листинг 3.3 - Продолжение

```
procedure Button3Click(Sender: TObject);

private
{ Private declarations }
public
```

```
{ Public declarations }
end;
var
Form11: TForm11;
implementation
{$R *.dfm}

procedure TForm11.Button1Click(Sender: TObject);
var
kat1,kat2,gip:real;
function GIPOTENUZA(a:real;b:real):real;
begin
GIPOTENUZA:=sqrt(a*a+b*b);
end;

begin
kat1:=strtofloat(edit1.Text);
        edit1.Clear;
kat2:=strtofloat(edit2.Text);
        edit2.Clear;
```

Листинг 3.3 - Продолжение

```
Edit3.Text:=floattostr(GIPOTENUZA(kat1,kat2));
end;

procedure TForm11.Button2Click(Sender: TObject);
begin
edit1.Clear;

edit2.Clear;
edit3.Clear;
```

```
end;

procedure TForm11.Button3Click(Sender: TObject);
begin
close;
end;
end.
```

Аналогичным образом выстроены и остальные модули, в которых приведены решения интерактивных примеров лабораторных работ (приложение А).

В следующем разделе выполним тестирование разработанного приложения на функционирование.

4 Проведение тестов и отладка лабораторного практикума

В предыдущем разделе мы разработали программу электронного практикума для дисциплины «Процедурные языки программирования». Далее, необходимо протестировать полученный программный продукт для того, чтобы определить насколько он корректно функционирует и отладить его работу, путем устранения возможных ошибок.

Для тестирования используем техники, представленные в [18]. Они базируется на том, что заранее известны структура тестируемого программного обеспечения, его алгоритмы и логика откликов всех модулей программы. При этом, тесты строятся так, чтобы была проверена логика составленной программы. Примем, первичный контроль, т.е. визуальное прочтение программы и синтаксический контроль правильности кода посредством внутреннего отладчика проведены.

Отладка программного средства (ПС) - это деятельность, направленная на обнаружение и исправление ошибок в ПС с использованием процессов выполнения его программ [28, 29].

Тестирование программного средства (ПС) - это процесс выполнения его программ на некотором наборе данных, для которого заранее известен результат применения или известны правила поведения этих программ. Указанный набор данных называется тестовым или просто тестом. Таким образом, отладку можно представить в виде многократного повторения трех процессов: тестирования, в результате которого может быть констатировано наличие в ПС ошибки, поиска места ошибки в программах и документации ПС и редактирования программ и документации с целью устранения обнаруженной ошибки [29]. Другими словами:

Отладка = Тестирование + Поиск ошибок + Редактирование.

Следует отметить, что тестирование используется и как часть процесса аттестации ПС.

В [29] приведены рекомендации по организации отладки:

1. Тестирование считается ключевой задачей разработки ПС.

2. Хорош тот тест, для которого высока вероятность обнаружить ошибку, а не тот, который демонстрирует правильную работу программы.

3. Подготовка тестов как для правильных, так и для неправильных данных.

4. Избегайте невоспроизводимых тестов, документируйте их пропуск через компьютер; детально изучайте результаты каждого теста.

5. Каждый модуль подключайте к программе только один раз, никогда не изменяйте программу, чтобы облегчить ее тестирование.

Пропускайте заново все тесты, связанные с проверкой работы какой-либо программы ПС или ее взаимодействия с другими программами, если в нее были внесены изменения (например, в результате устранения ошибки).

Автономное тестирование модуля целесообразно осуществлять в четыре последовательно выполняемых шага:

Шаг 1. На основании спецификации отлаживаемого модуля подготовьте тест для каждой возможности и каждой ситуации, для каждой границы областей допустимых значений всех входных данных, для каждой области изменения данных, для каждой области недопустимых значений всех входных данных и каждого недопустимого условия.

Шаг 2. Проверьте текст модуля, чтобы убедиться, что каждое направление любого разветвления будет пройдено хотя бы на одном тесте. Добавьте недостающие тесты.

Шаг 3. Убедитесь по тексту модуля, что для каждого цикла существует тест, для которого тело цикла не выполняется, тест, для которого тело цикла выполняется один раз, и тест, для которого тело цикла выполняется максимальное число раз. Добавьте недостающие тесты.

Шаг 4. Проверьте по тексту модуля его чувствительность к отдельным особым значениям входных данных - все такие значения должны входить в тесты. Добавьте недостающие тесты.

Комплексная отладка программного средства

При комплексной отладке тестируется ПС в целом, причем тесты готовятся по каждому из документов. Тестирование этих документов производится, как правило, в порядке, обратном их разработке (исключение составляет лишь тестирование документации по применению, которая разрабатывается по внешнему описанию параллельно с разработкой текстов программ; это тестирование лучше производить после завершения тестирования внешнего описания). Тестирование при комплексной отладке представляет собой применение ПС к конкретным данным, которые в принципе могут возникнуть у пользователя (в частности, все тесты готовятся в форме, рассчитанной на пользователя), но, возможно, в моделируемой (а не в реальной) среде. Например, некоторые недоступные при комплексной отладке устройства ввода и вывода могут быть заменены их программными имитаторами [28].

Тестирование архитектуры ПС.

Целью тестирования является поиск несоответствия между описанием архитектуры и совокупностью программ ПС. К моменту начала тестирования архитектуры ПС должна быть уже закончена автономная отладка каждой подсистемы. Ошибки реализации архитектуры могут быть связаны прежде всего с взаимодействием этих подсистем, в частности, с реализацией архитектурных функций (если они есть). Поэтому хотелось бы проверить все пути взаимодействия между подсистемами ПС. Но так как их может быть слишком много, то желательно бы протестировать хотя бы все цепочки выполнения подсистем без повторного вхождения. Если заданная архитектура представляет ПС в качестве малой системы из выделенных подсистем, то число таких цепочек будет вполне обозримо.

Тестирование внешних функций. Целью тестирования является поиск расхождений между функциональной спецификацией и совокупностью программ ПС. Несмотря на то, что все эти программы автономно уже отлажены, указанные расхождения могут быть, например, из-за несоответствия внутренних спецификаций программ и их модулей (на основании которых производилось авто-

номное тестирование) внешней функциональной спецификации ПС. Как правило, тестирование внешних функций производится так же, как и тестирование модулей на первом шаге, т.е. как черного ящика.

Тестирование качества ПС.

Целью тестирования является поиск нарушений требований качества, сформулированных в спецификации качества ПС. Это наиболее трудный и наименее изученный вид тестирования. Ясно лишь, что далеко не каждый примитив качества ПС может быть испытан тестированием. Завершенность ПС проверяется уже при тестировании внешних функций. На данном этапе тестирование этого примитива качества может быть продолжено, если требуется получить какую-либо вероятностную оценку степени надежности ПС. Однако, методика такого тестирования еще требует своей разработки. Точность, устойчивость, защищенность, временная эффективность, в какой-то мере - эффективность по памяти, эффективность по устройствам, расширяемость и, частично, независимость от устройств могут тестироваться. Каждый из этих видов тестирования имеет свою специфику и заслуживает отдельного рассмотрения. Легкость применения ПС (критерий качества, включающий несколько примитивов качества) оценивается при тестировании документации по применению ПС.

Тестирование документации по применению ПС.

Целью тестирования является поиск несогласованности документации по применению и совокупностью программ ПС, а также неудобств применения ПС. Этот этап непосредственно предшествует подключению пользователя к завершению разработки ПС (тестированию требований к ПС и аттестации ПС), поэтому весьма важно разработчикам сначала самим воспользоваться ПС так, как это будет делать пользователь. Все тесты на этом этапе готовятся исключительно на основании только документации по применению ПС. Прежде всего, должны тестироваться возможности ПС как это делалось при тестировании

внешних функций, но только на основании документации по применению. Должны быть протестированы все неясные места в документации, а также все примеры, использованные в документации. Далее тестируются наиболее трудные случаи применения ПС с целью обнаружить нарушение требований относительности легкости применения ПС.

Тестирование определения требований к ПС. Целью тестирования является выяснение, в какой мере ПС не соответствует предъявленному определению требований к нему. Особенность этого вида тестирования заключается в том, что его осуществляет организация-покупатель или организация-пользователь ПС как один из путей преодоления барьера между разработчиком и пользователем. Обычно это тестирование производится с помощью контрольных задач - типовых задач, для которых известен результат решения. В тех случаях, когда разрабатываемое ПС должно прийти на смену другому варианту ПС, который решает хотя бы часть задач разрабатываемого ПС, тестирование производится путем решения общих задач с помощью как старого, так и нового ПС с последующим сопоставлением полученных результатов. Иногда в качестве формы такого тестирования используют опытную эксплуатацию ПС - ограниченное применение нового ПС с анализом использования результатов в практической деятельности. По существу, этот вид тестирования во многом перекликается с испытанием ПС при его аттестации, но выполняется до аттестации, а иногда и вместо аттестации.

Модульное тестирование

Модульное тестирование (МТ) - это тестирование программы на уровне отдельно взятых модулей, функций или классов. Цель МТ состоит в выявлении локализованных в модуле ошибок в реализации алгоритмов, а также в определении степени готовности системы к переходу на следующий уровень разработки и тестирования. Модульное тестирование проводится по принципу "белого ящика", то есть основывается на знании внутренней структуры программы, и часто включает те или иные методы анализа покрытия кода [29].

В тестирование многомодульных программ можно выделить четыре этапа:

1) тестирование отдельных модулей;

2) совместное тестирование модулей;

3) тестирование спецификации программы;

4) тестирование всего комплекса в целом (т.е. поиск несоответствия созданного программного продукта сформулированным ранее целям проектирования, отраженным в техническом задании).

На первых двух этапах используются прежде всего методы структурного тестирования, т.к. на последующих этапах тестирования эти методы использовать сложнее из-за больших размеров проверяемого программного обеспечения; последующие этапы тестирования ориентированы на обнаружение ошибок различного типа, которые не обязательно связаны с логикой программы.

Известны два подхода к тестированию модулей: монолитное и пошаговое тестирование.

Монолитное тестирование

При монолитном тестировании сначала по отдельности тестируются все модули программного комплекса, а затем все они объединяются в рабочую программу для комплексного тестирования. Для автономного тестирования каждого модуля требуется модуль - драйвер и несколько модулей - заглушек (имитирующих работу модулей, вызываемых из тестируемого).

Пошаговое тестирование

При пошаговом тестировании каждый модуль для своего тестирования подключается к набору уже проверенных модулей. При пошаговом тестировании модули проверяются не изолированно друг от друга, поэтому требуются либо только драйверы, либо только заглушки.

При пошаговом тестировании возможны две стратегии подключения модулей: нисходящая и восходящая.

Нисходящее тестирование

Нисходящее тестирование начинается с главного (или верхнего) модуля программы, а выбор следующего подключаемого модуля происходит из числа модулей, вызываемых из уже протестированных. Одна из основных проблем, возникающих при нисходящем тестировании, - создание заглушек.

Основные достоинства нисходящего тестирования: уже на ранней стадии тестирования есть возможность получить работающий вариант разрабатываемой программы; быстро могут быть выявлены ошибки, связанные с организацией взаимодействие с пользователем.

Восходящее тестирование

При восходящем тестировании проверка программы начинается с терминальных модулей (т.е. тех, которые не вызывают не каких других модулей программы). Эта стратегия во многом противоположна нисходящему тестированию (в частности, преимущества становятся недостатками и наоборот).

Преимущества восходящего тестирования:

- поскольку нет промежуточных модулей, нет проблем, связанных с возможностью или трудностью задания тестов;

- нет трудностей, вызывающих желание перейти к тестированию следующего модуля, не завершив проверки предыдущего.

Недостатком восходящего тестирования является то, что проверка всей структуры разрабатываемого программного комплекса возможна только на завершающей стадии тестирования.

Хотя однозначного вывода о преимущества той или иной стратегии пошагового тестирования сделать нельзя (нужно учитывать конкретные характеристики тестируемой программы), в большинстве случаев более предпочтительным является восходящее тестирование.

Модульное тестирование обычно подразумевает создание вокруг каждого модуля определенной среды, включающей заглушки для всех интерфейсов те-

стируемого модуля. Некоторые из них могут использоваться для подачи входных значений, другие для анализа результатов, присутствие третьих может быть продиктовано требованиями, накладываемыми компилятором и сборщиком.

На уровне модульного тестирования проще всего обнаружить дефекты, связанные с алгоритмическими ошибками и ошибками кодирования алгоритмов, типа работы с условиями и счетчиками циклов, а также с использованием локальных переменных и ресурсов. Ошибки, связанные с неверной трактовкой данных, некорректной реализацией интерфейсов, совместимостью, производительностью и т.п. обычно пропускаются на уровне модульного тестирования и выявляются на более поздних стадиях тестирования [29].

Именно эффективность обнаружения тех или иных типов дефектов должна определять стратегию модульного тестирования, то есть расстановку акцентов при определении набора входных значений. У организации, занимающейся разработкой программного обеспечения, как правило, имеется историческая база данных (Repository) разработок, хранящая конкретные сведения о разработке предыдущих проектов: о версиях и сборках кода (build) зафиксированных в процессе разработки продукта, о принятых решениях, допущенных просчетах, ошибках, успехах и т.п. Проведя анализ характеристик прежних проектов, подобных заказанному организации, можно предохранить новую разработку от старых ошибок, например, определив типы дефектов, поиск которых наиболее эффективен на различных этапах тестирования.

В данном случае анализируется этап модульного тестирования. Если анализ не дал нужной информации, например, в случае проектов, в которых соответствующие данные не собирались, то основным правилом становится поиск локальных дефектов, у которых код, ресурсы и информация, вовлеченные в дефект, характерны именно для данного модуля. В этом случае на модульном уровне ошибки, связанные, например, с неверным порядком или форматом параметров модуля, могут быть пропущены, поскольку они вовлекают информацию, затрагивающую другие модули (а именно, спецификацию интерфейса), в

то время как ошибки в алгоритме обработки параметров довольно легко обнаруживаются [29].

Из приведенного анализа использующихся вариантов тестирования для разработанного электронного практикума более всего подходит прямая проверка на работоспособность, тем более, что он однотомный и в нем нет сложных подключаемых структур.

Для этого запустим программу с помощью файла «Практикум», который выберем из указанной выше директории (рис. 3.1). После запуска появляется стартовое окно программного продукта (рис. 4.1). На нем есть несколько областей: слева перечни всех обучающих материалов, по центру место для отображения выбранного из них.

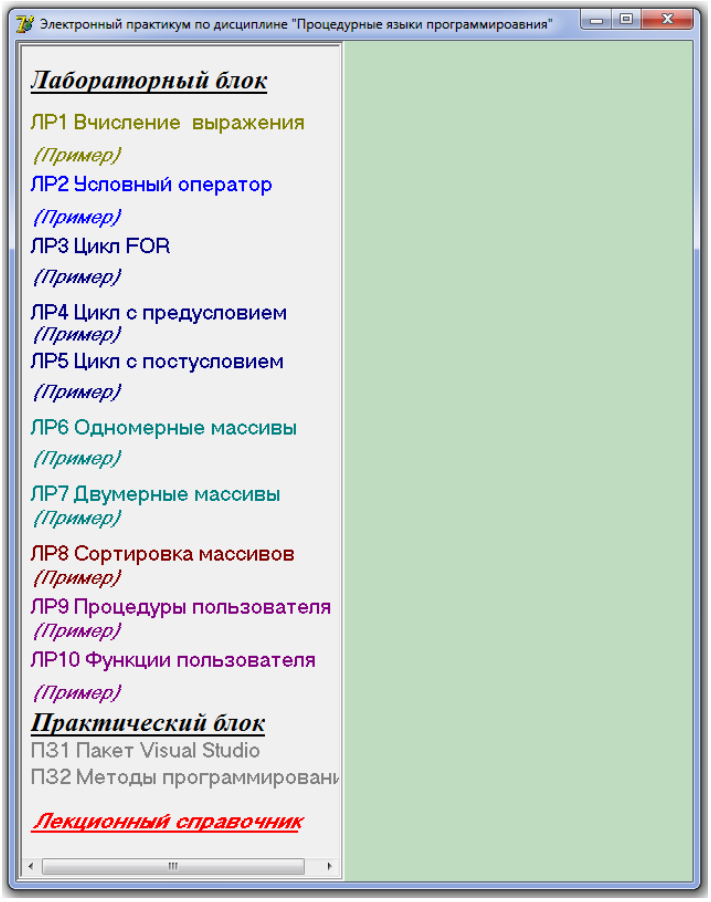

Рисунок 4.1 – Стартовое окно программы

Здесь же можно проверить переходы по гиперссылкам к следующему или к предыдущему обучающему блоку, выбрав соответствующие ссылки в поле основного окна (рис. 4.2).

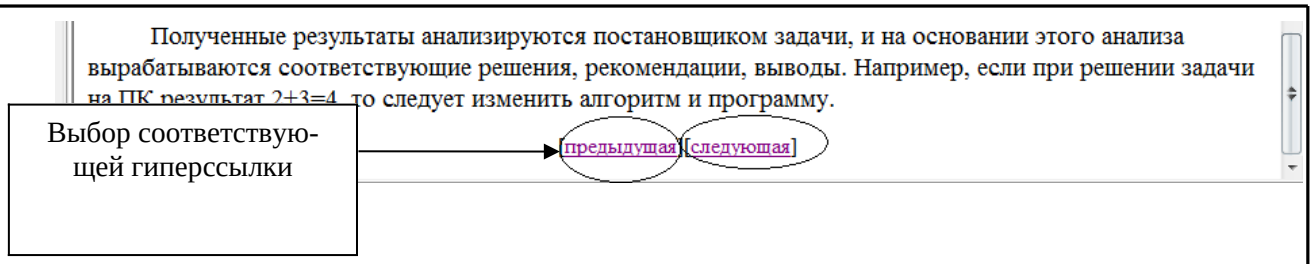

Рисунок 4.2 – Контроль перехода по гиперссылкам

Далее выберем одну из лабораторных работ, ее текст должен отобразиться в основном окне (рис. 4.3).

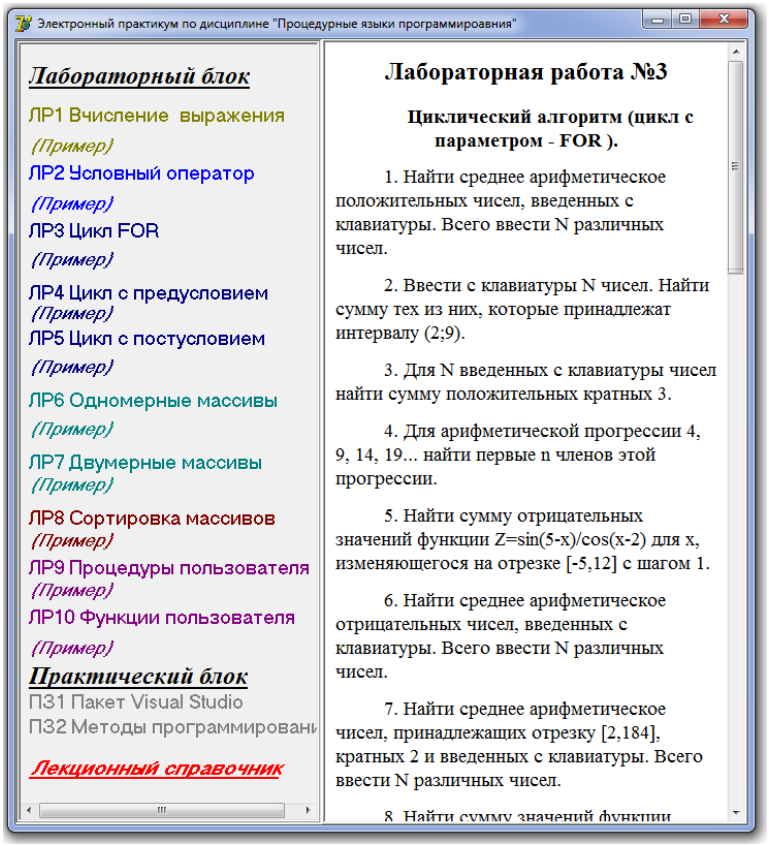

Рисунок 4.3 – Проверка корректного отображения содержания выбранной лабораторной работы

Далее, если выбрать «Пример», то должно появиться окно демонстрации решения тестового примера данной лабораторной работы (рис. 4.4).

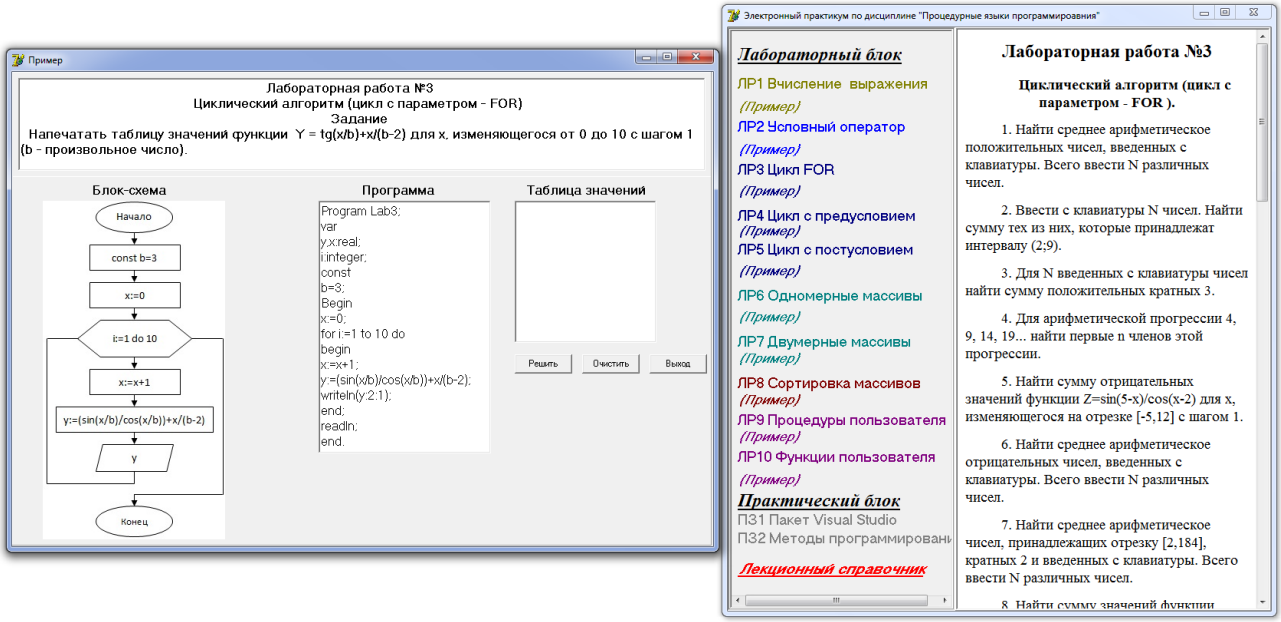

Рисунок 4.4 – Контроль исполнения тестового примера выбранной лабораторной работы

Для того, что бы проконтролировать выполнение тестового примера необходимо нажать кнопку «Решить» в показанной вкладке. После чего, в поле «Вывод положительных значений» можно видеть результат решения задачи (рис. 4.4). Затем после нажатия кнопки «Очистить» это поле должно быть пустым (рис. 4.5).

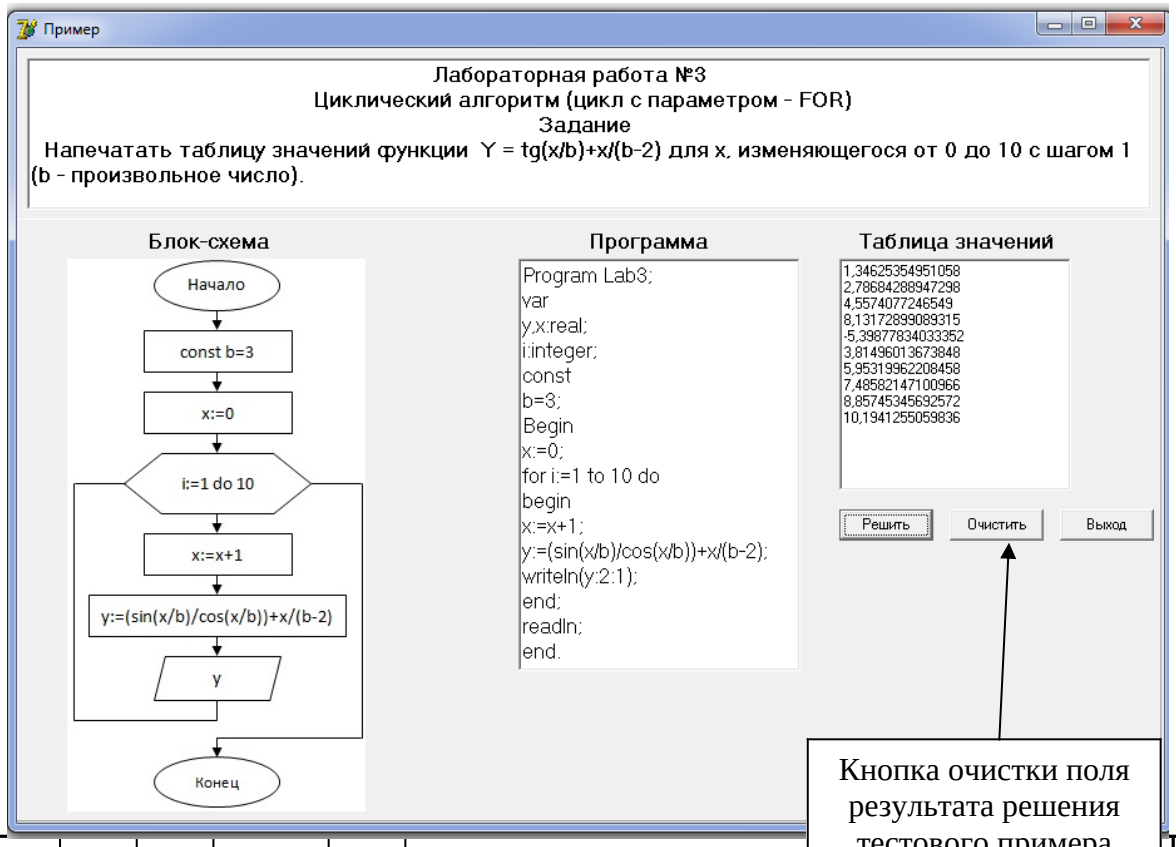

Рисунок 4.5 – Контроль очистки поля решения на вкладке «Пример»

Закрытие этой формы происходит после нажатия на кнопку «Выход» и пособие переходит в режим отображения задания по выбранной лабораторной работе (рис. 4.3).

Аналогично проверяется корректность перехода по гиперссылкам и отображения информации в блоке практических занятий (рис. 4.6).

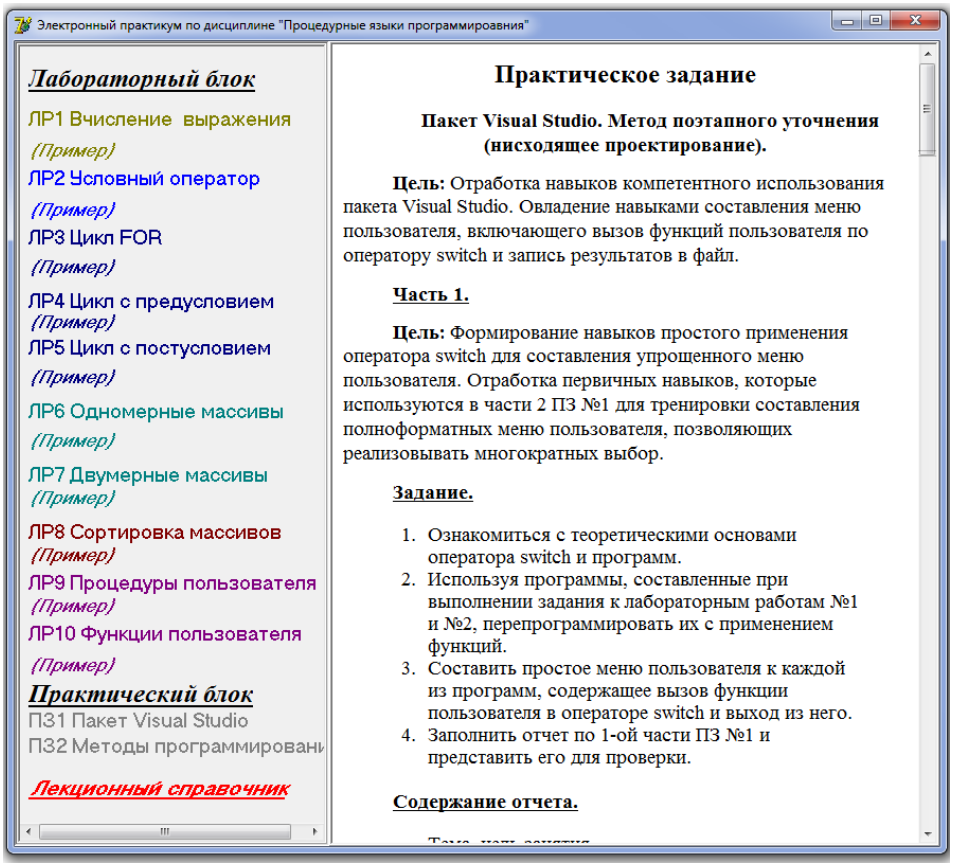

Рисунок 4.6 – Результат контроля корректного отображения материалов практических занятий

Еще одним достоинством разработанного пособия является масштабирование изображений в его окнах. Т.е., при изменении размеров окна текст в нем не теряется а пропорционально занимает область, оставшуюся для экспозиции. Так, изменив размеры окна, показанного на рисунке 4.6, получим новое отображение материала без потери информативности (4.7).

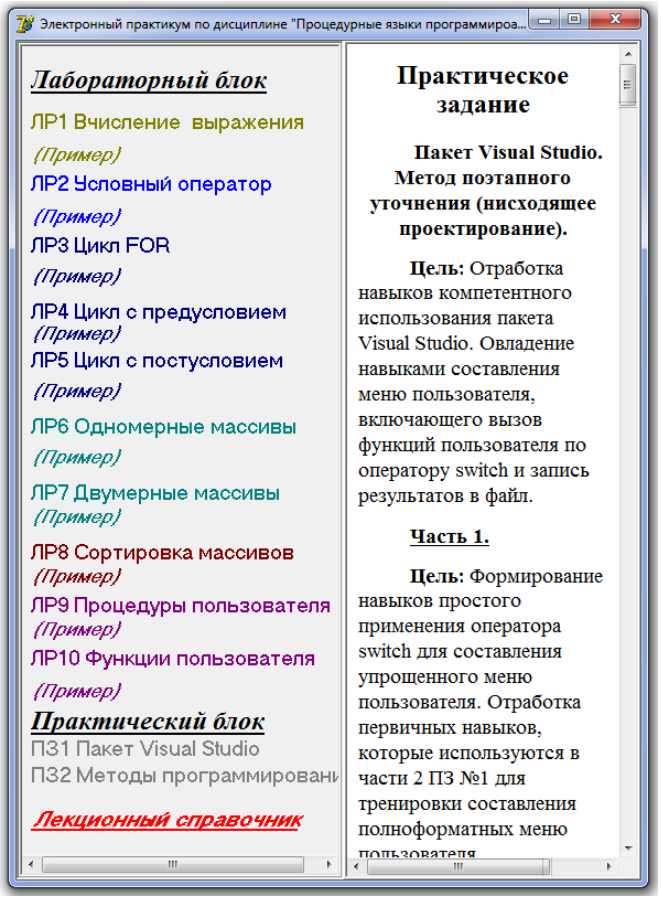

Рисунок 4.7 – Проверка информативности текста при изменении размеров окна

Такое свойство очень полезно при использовании практикума на различных мобильных устройствах (планшеты, ноутбуки, телефоны). Как мы видим, что текст остается доступен полностью для просмотра, а графику можно посмотреть при помощи полос прокрутки.

Таким образом, была проконтролирована логика функционирования разработанного практикума. Вывод – он функционирует корректно.

В следующем разделе работы создадим инструкцию пользователю, в которую включим указания по установке, удалению и изменению разработанного программного продукта.

5 Разработка инструкций по использованию материалов лабораторного практикума

Назначение электронного практикума

Разработанный электронный практикум предназначен для использования при самостоятельном изучении дисциплины «Процедурные языки программирования».

Цель решения задачи в бакалаврской работе – обеспечить одним обучающим ресурсом процесс самостоятельного освоения дисциплины бакалаврами очного и заочного отделений. В силу того, что большую часть времени, отведенного на изучение этой дисциплины обучаемые расходуют согласно программе дисциплины на самостоятельную работу, поэтому настоящий практикум снабжен всеми необходимыми для этого материалами и практическими интерактивными примерами.

Подготовка практикума к работе и его запуск

Перед использованием программного продукта необходимо его перенести на компьютер. Для этого следует выделить отдельную директорию для хранения практикума и его составляющих (рис. 3.1). Никаких специфических предустановок не требуется, для запуска используется управляющий стартовый

файл «Практикум» – он же содержит управляющий фрейм в откомпилированном виде. Однако, необходимо, чтобы в выделенной директории находились указанные папки для хранения контента электронного практикума (рис. 3.1).

Правила пользования практикумом

Использование практикума является простым и понятным: в открывшемся после запуска окне выбираются виды обучающих материалов –лабораторные работы или практические занятия (рис. 5.1).

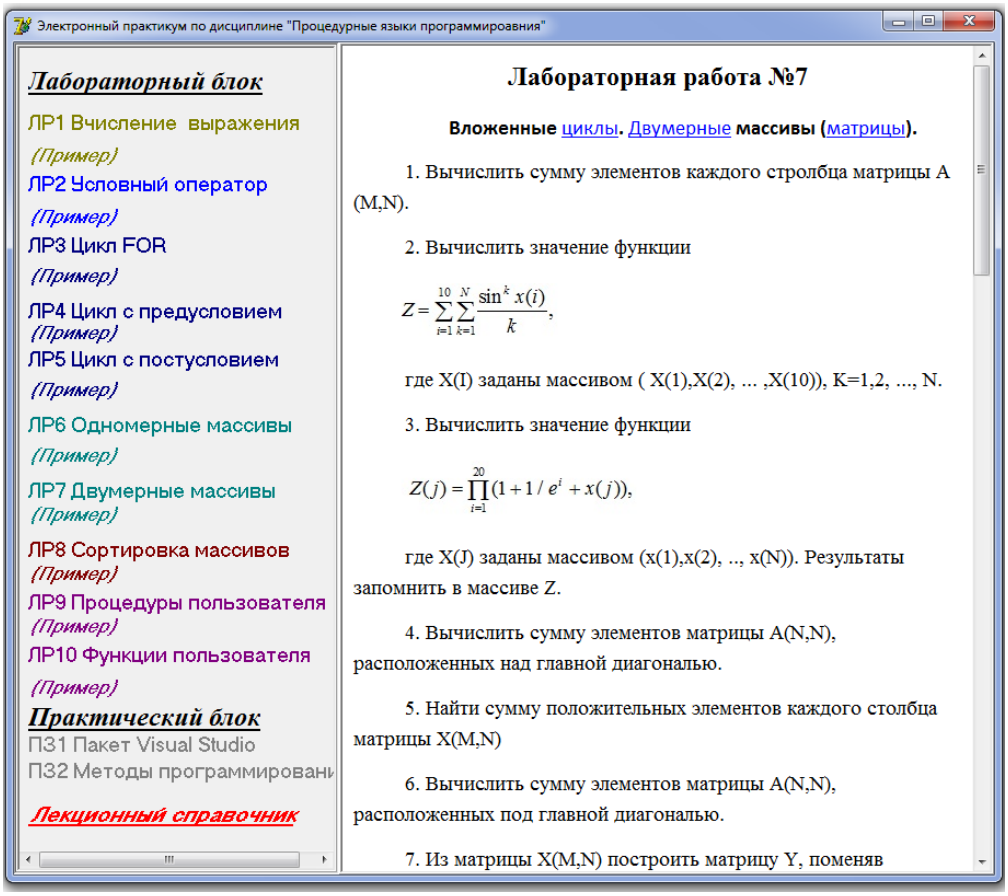

Рисунок 5.1 – Стартовое окно приложения

Из рисунка видно, что для отображения текста изучаемого материала необходимо выбрать слева его наименование и справа в основном окне можно прокручивая его приступить к изучению.

Для того, чтобы перейти по гиперссылке, расположенной в любом из блоков, необходимо кликнуть по ней левой клавишей мыши и произойдет переход в нужное место лекционного материала (рис. 5.2).

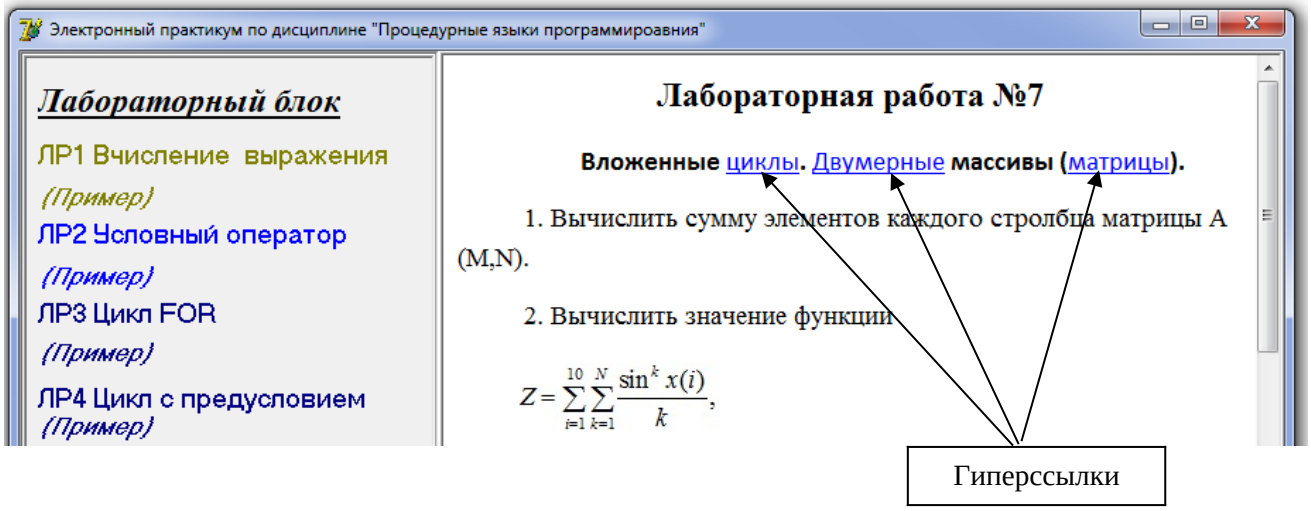

Рисунок 5.2 – Правило использования гиперссылок

После перехода (в данном случае к лекции по циклам) открывается окно с соответствующей лекцией, где продемонстрировано использование теоретических положений, от которых осуществлялся переход (рис. 5.3).

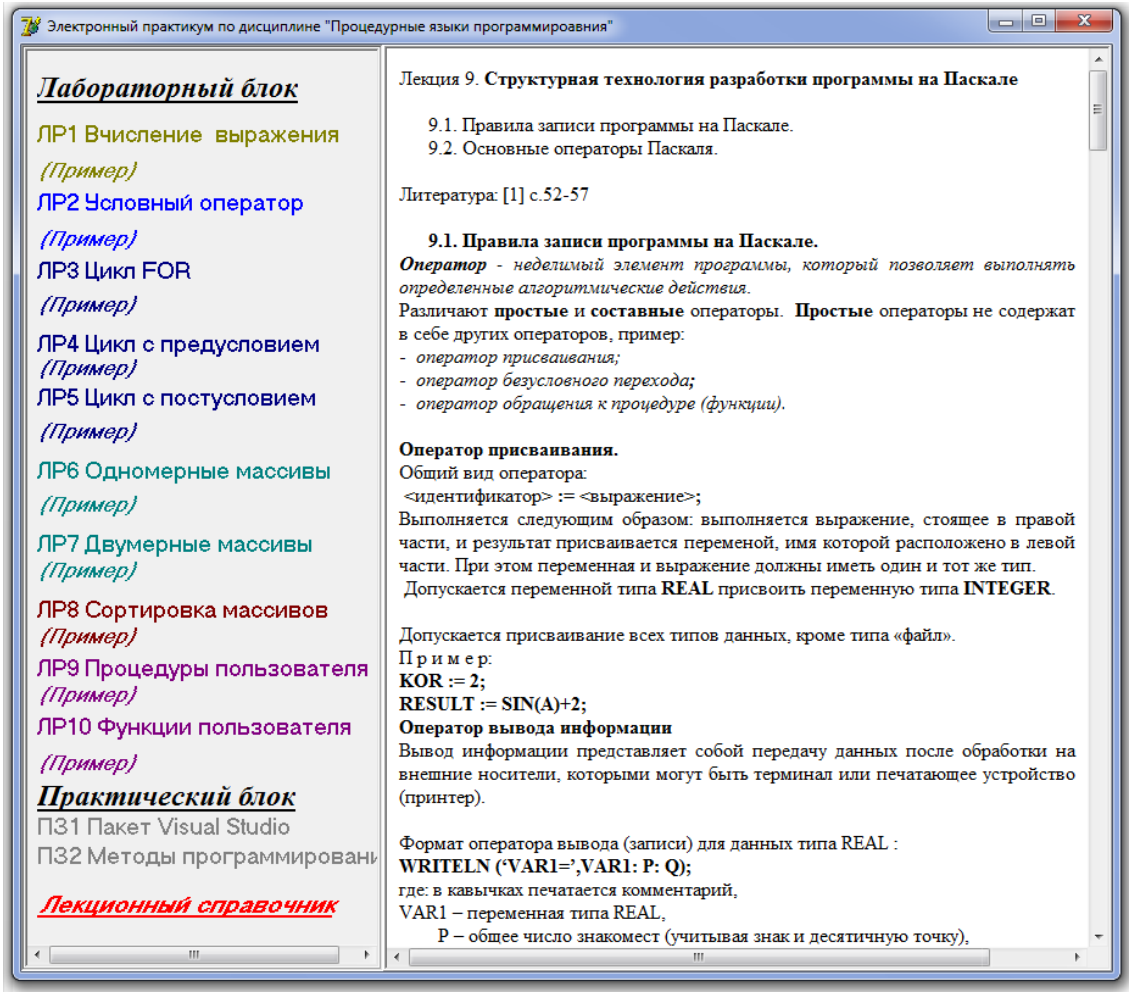

Рисунок 5.3 – Переход по гиперссылке

Далее, здесь же можно открыть пример использования массивов (рис. 5.4).

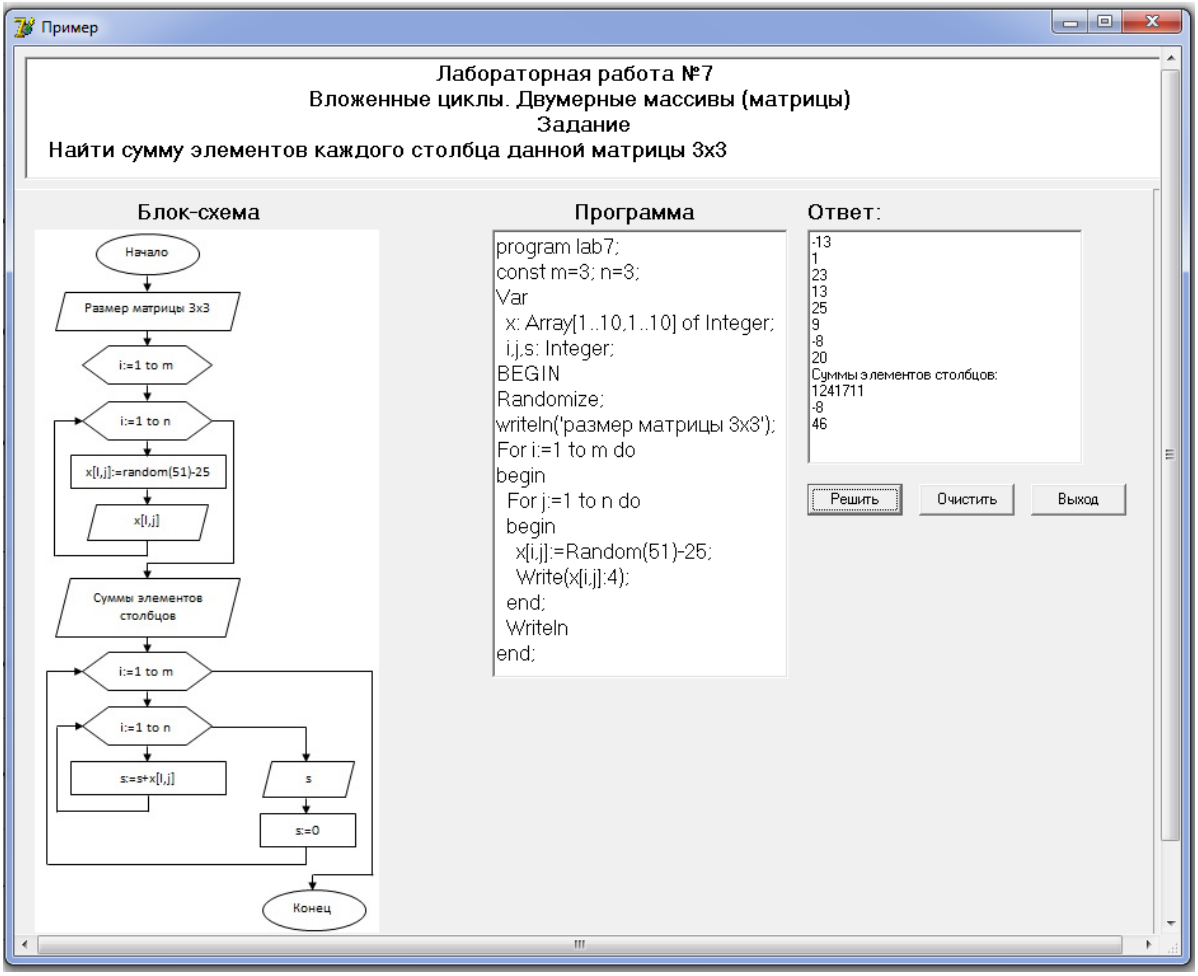

Рисунок 5.4 – Использование расчетной формы «Пример» из раздела лабораторных работ

Управляющие кнопки на форме совершенно понятны, поэтому нажимая «Решить» можно увидеть результаты вычислений, а – «Очистить» все приводится в исходное положение.

Аналогично используются и другие примеры в лабораторных работах. Примечательно то, что в окне примера выведены условия, алгоритм и программа тестовой задачи, что является очень удобным для самостоятельного изучения материала дисциплины.

Видно, что практикум построен таким образом, что необходимые теоретические сведения можно просмотреть по соответствующим гиперссылкам в лабораторных работах и в практических занятиях. Однако, для удобства изучения материала предусмотрен отдельный просмотр всего набора лекций. Для этого нужно выбрать рубрику «Лекционный справочник» и справа мы видим весь перечень лекций (рис. 5.5).

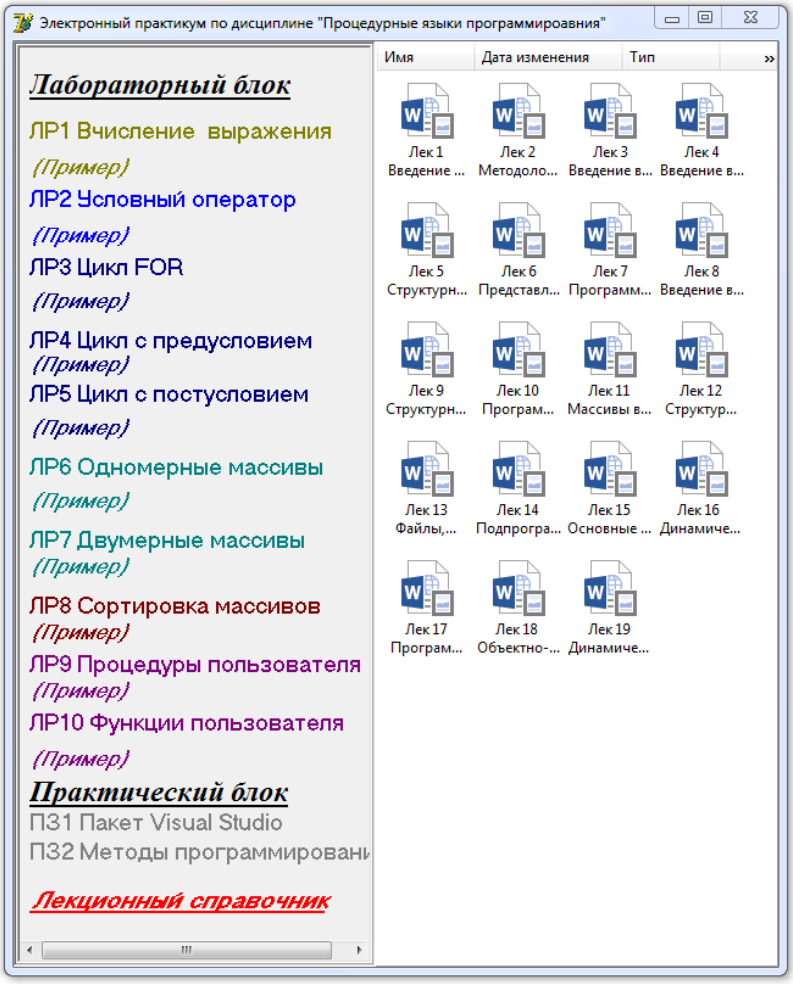

Рисунок 5.5- Использование рубрики «Лекционный справочник»

Далее можно открыть любую из лекций и продолжить изучение (рис. 5.6).

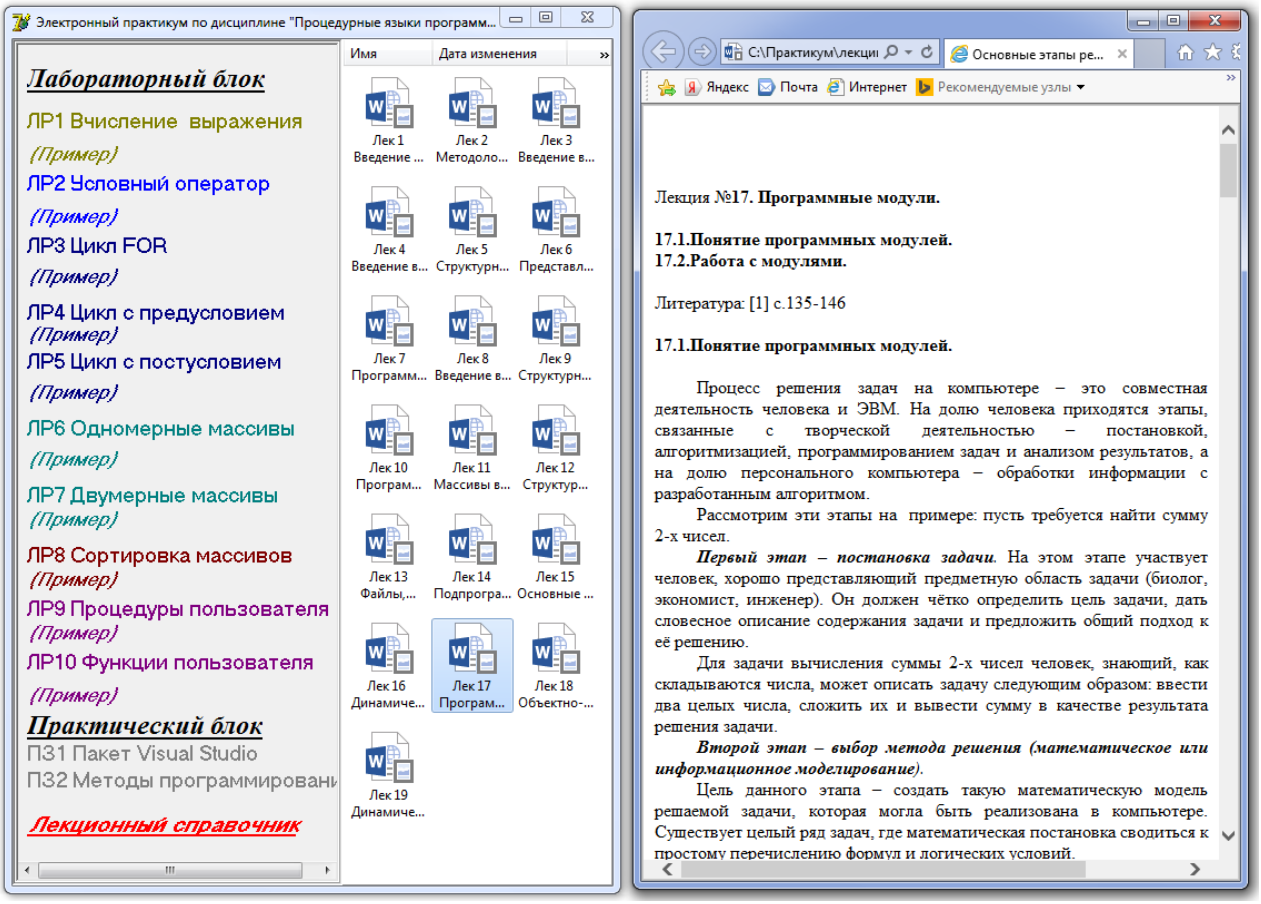

Рисунок 5.6- Формат изучения лекционного материала

Причем, т.к. текст лекции открывается в браузере, то можно в самом практикуме открыть соответствующую лабораторную работу и продолжить изучение совместно, что является удобным и выгодно отличает разработанный ресурс (рис. 5.7).

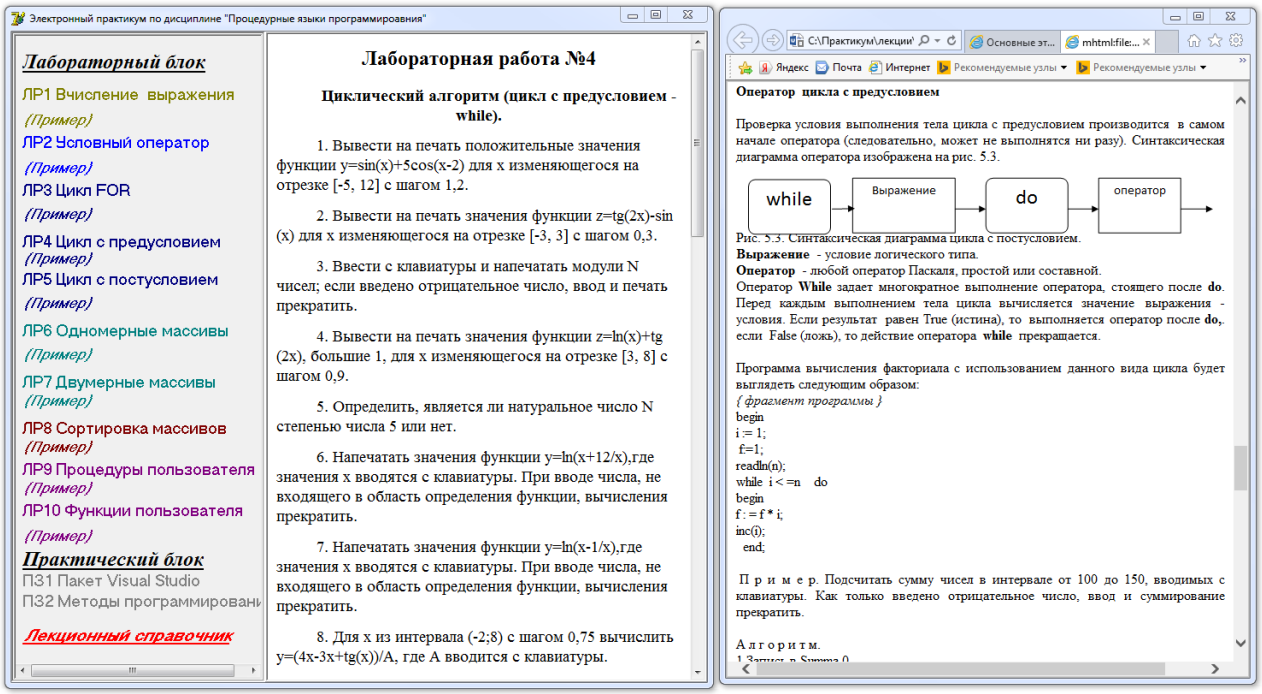

Рисунок 5.7- Совместное открытие лабораторных работ и лекций

В заключение отметим, что для пользования программой необходимо иметь элементарные навыки владения ПК и тщательно ознакомиться с руководством пользователя.

ЗАКЛЮЧЕНИЕ

В соответствии с заданием бакалаврской работы был разработан электронный практикум по дисциплине "Процедурные языки программирования". Полученный программный продукт в виде учебно-методических материалов и управляющей оболочки был внедрен в учебный процесс ВУЗа для обеспечения самостоятельной работы бакалавров по изучению указанной дисциплины.

В процессе выполнения бакалаврской работы были решены следующие задачи:

сформулированы требования к исходному представлению обучающих материалов для использования в практикуме;

рассмотрены технологии и средства создания информационных систем;

сделан анализ программного обеспечения для создания информационной системы;

составлен алгоритм приложения;

на основе алгоритма управляющей оболочки разработан пользовательский интерфейс информационной системы с помощью интегрированной среды программирования Borland Delphi -7;

реализованы интерактивные обучающие примеры в виде выполняемых тестовых заданий.

После создания практикум протестирован и является полностью подготовленным к использованию.

Пояснительная записка решения поставленной задачи оформлена согласно установленных требований [30].

Данный программный продукт можно использовать на базе ОС семейства Windows.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Дистанционное обучение: возможности и ограничения. Кому и когда оно необходимо, когда противопоказано, а когда – нецелесообразно. // [Электронный ресурс]. 2016, *URL: http://ra-kurs.spb.ru*.

2 Модуль "Дистанционное обучение" // [Электронный ресурс]. 2016 *URL: http://www.websoft.ru.*

3 Материалы Международной научной конференции Актуальные проблемы развития образования в России и за рубежом. Россия. Москва, 29-30 апреля 2016г. с. 37-45.

4 Типовое положение о ВУЗе, утвержденное постановлением правительства РФ от 14 февраля 2008 г. № 71.

5 Закон РФ "Об авторском и смежных правах".

6 Закон РФ "О правовой охране программ для ЭВМ и баз данных".

7 ГОСТ 7.60–90 Система стандартов по информации, библиотечному и издательскому делу (СИБИД).

8 Приказ Минобразования РФ N1646 от 19.06.98 о создании Федерального экспертного совета по учебным электронным изданиям.

9 Агеев В. Н. Электронные издания учебного назначения: концепции, создание, использование: учебное пособие. – М., МГУП, 2008. – 235 с.

10 ГОСТ 7.83-2001. Электронные издания. Основные виды и выходные сведения.

11 Вымятнин В.М., Демкин В.П., Можаева Г.В., Руденко Т.В. Мультимедиа-курсы: методология и технология разработки // [Электронный ресурс]. 2013 *URL: http://ido.tsu.ru*.

12 Дубенский В.В. Технология создания электронных обучающих систем. // [Электронный ресурс]. 2013 *URL: http://www.dissercat.com.*

13 Официальный сайт SunRav Software — разработчика программ для образования и бизнеса.// [Электронный ресурс]. 2013 *URL: http://www.sunrav.ru*.

14 Обзор электронных учебных пособий. // [Электронный ресурс]. 2013 *URL:* http://saprr.narod.ru/elektron_uchebnik.htm.

15 XML редактор // [Электронный ресурс]. 2013 *URL:* http://www.bestfree.ru.

16 Язык программирования html// [Электронный ресурс]. 2016 *URL:* http://life-prog.ru.

17 Галисеев Г.В Компоненты в Delphi 7. Профессиональная работа. – М.: Издательский дом «Вильямс», 2008. – 624 с.

18 Иванова Г.С. Технология программирования: Учебник для вузов. - М.: Издательство МГТУ им. Н.Э. Баумана, 2002. - 320 с.

19 Архангельский А.Я. Программирование в Delphi 7. М.: БИНОМ. 2003. 1152с.

20 Культин Н.Б Основы программирования в Delphi 7. – СПб.: БХВ-Петербург, 2009. – 608 с.

21 Delphi программирование [Электронный ресурс]. – *URL:*http://www.delphisources.ru/pages/faq/file_system.html.

22 Гофман В., Хомоненко А. Delphi. Быстрый старт. – СПб.: БХВ-Петербург, 2008. – 289 с .

23 База знаний Delphi. Компоненты и Классы. Списки // [Электронный ресурс]. – *URL: http://www.kansoftware.ru/?did=181/.*

24 Лафоре Р. Объектно-ориентированное программирование в C++ СПб.: БХВ-Петербург, 2009. – 890 с.

25 Краткое описание Visual Studio *// [Электронный ресурс]. – URL: http://habrahabr.ru.*

26 Васильев А. Н. Java. Объектно-ориентированное программирование: Учебное пособие. — СПб.: Питер, 2011. — 400с.

27 Интегрированная среда NetBeans. // [Электронный ресурс]. – *URL:* https://netbeans.org.

28 Ю.М. Безбородов. Индивидуальная отладка программ. - М.: Наука, 2010.

29 В.В. Липаев. Тестирование программ. - М.: Радио и связь, 2010.

Подп. и дата | Инв. № подл. | Взам. Инв. № | Подп. и дата | Инв. №

30 Борисов Б.П., Жуковский А.Г,, Рыбалко И.П., Швидченко С.А. Руко-вод-ство по дипломному проектированию. Учебно-методическое посо-бие.- Ростов-на-Дону: СКФ МТУСИ, 2011.-62с.

Приложение А

(обязательное)

1 Исходные коды программного продукта

```
 unit Unit1;
interface
uses
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, StdCtrls, OleCtrls, SHDocVw, ExtCtrls, Menus, ComCtrls, Grids,
Outline, FileCtrl;
type
TForm1 = class(TForm)
Panel1: TPanel;
Panel2: TPanel;
WebBrowser1: TWebBrowser;
ScrollBox1: TScrollBox;
Label1: TLabel;
Label3: TLabel;
Label4: TLabel;
Label2: TLabel;
Label5: TLabel;
Label6: TLabel;

Label7: TLabel;
Label8: TLabel;
Label9: TLabel;
Label10: TLabel;
Label11: TLabel;
Label12: TLabel;
Label13: TLabel;
```

```
    Label14: TLabel;
    Label15: TLabel;
    Label16: TLabel;
    Label17: TLabel;
    Label18: TLabel;
    Label19: TLabel;
    Label20: TLabel;
    Label21: TLabel;
    Label22: TLabel;
    Label23: TLabel;
    Label24: TLabel;
    Label25: TLabel;
    Label26: TLabel;
    Label27: TLabel;
    Label28: TLabel;
    Label29: TLabel;
    Label30: TLabel;
    Label31: TLabel;
    Label32: TLabel;
    Label33: TLabel;
    Label34: TLabel;
    Label35: TLabel;
    Label36: TLabel;
    Label37: TLabel;
    Label38: TLabel;
    Label39: TLabel;
    Label40: TLabel;
    Label41: TLabel;
    procedure Label3Click(Sender: TObject);
    procedure Label5Click(Sender: TObject);
    procedure Label2Click(Sender: TObject);
```

```
procedure Label4Click(Sender: TObject);
procedure Label6Click(Sender: TObject);
procedure Label7Click(Sender: TObject);
procedure Label8Click(Sender: TObject);
procedure Label9Click(Sender: TObject);
procedure Label10Click(Sender: TObject);
procedure Label11Click(Sender: TObject);
procedure Label12Click(Sender: TObject);
procedure Label13Click(Sender: TObject);
procedure Label14Click(Sender: TObject);
procedure Label15Click(Sender: TObject);
procedure Label16Click(Sender: TObject);
procedure Label17Click(Sender: TObject);
procedure Label18Click(Sender: TObject);
procedure Label19Click(Sender: TObject);
procedure Label20Click(Sender: TObject);
procedure Label23Click(Sender: TObject);
procedure Label25Click(Sender: TObject);
procedure Label27Click(Sender: TObject);
procedure Label29Click(Sender: TObject);
procedure Label31Click(Sender: TObject);

procedure Label33Click(Sender: TObject);
procedure Label35Click(Sender: TObject);
procedure Label41Click(Sender: TObject);
procedure Label39Click(Sender: TObject);
procedure Label37Click(Sender: TObject);
procedure Label22Click(Sender: TObject);
procedure Label24Click(Sender: TObject);
procedure Label26Click(Sender: TObject);
procedure Label28Click(Sender: TObject);
```

```
procedure Label30Click(Sender: TObject);
procedure Label32Click(Sender: TObject);
procedure Label34Click(Sender: TObject);
procedure Label36Click(Sender: TObject);
procedure Label38Click(Sender: TObject);
procedure Label40Click(Sender: TObject);
private
{Private declarations }
public
{ Public declarations }
end;
var
Form1: TForm1;
implementation
uses Unit2, Unit3, Unit4, Unit5, Unit6, Unit7, Unit8, Unit11, Unit10,
  Unit9;
{$R *.dfm}
procedure TForm1.Label3Click(Sender: TObject);
begin



WebBrowser1.Navigate(ExtractFilePath(Application.ExeName) +
+ 'лекции/lek1.html');
Panel2.Visible:=true;
end;

procedure TForm1.Label2Click(Sender: TObject);
begin
WebBrowser1.Navigate(ExtractFilePath(Application.ExeName) +
+ 'лекции/lek2.html');
Panel2.Visible:=true;
```

```
end;

procedure TForm1.Label4Click(Sender: TObject);
begin
WebBrowser1.Navigate(ExtractFilePath(Application.ExeName) +
+ 'лекции/lek3.html');
Panel2.Visible:=true;
end;

procedure TForm1.Label5Click(Sender: TObject);
begin
WebBrowser1.Navigate(ExtractFilePath(Application.ExeName) +
+ 'лекции/lek4.html');
Panel2.Visible:=true;
end;

procedure TForm1.Label6Click(Sender: TObject);
begin
WebBrowser1.Navigate(ExtractFilePath(Application.ExeName) +

+ 'лекции/lek5.html');
Panel2.Visible:=true;
end;

procedure TForm1.Label7Click(Sender: TObject);
begin
WebBrowser1.Navigate(ExtractFilePath(Application.ExeName) +
+ 'лекции/lek6.html');
Panel2.Visible:=true;
end;
```

```
procedure TForm1.Label8Click(Sender: TObject);
begin
WebBrowser1.Navigate(ExtractFilePath(Application.ExeName) +
+ 'лекции/lek7.html');
Panel2.Visible:=true;
end;

procedure TForm1.Label9Click(Sender: TObject);
begin
WebBrowser1.Navigate(ExtractFilePath(Application.ExeName) +
+ 'лекции/lek8.html');
Panel2.Visible:=true;
end;

procedure TForm1.Label10Click(Sender: TObject);
begin
WebBrowser1.Navigate(ExtractFilePath(Application.ExeName) +
+ 'лекции/lek9.html');

Panel2.Visible:=true;
end;

procedure TForm1.Label11Click(Sender: TObject);
begin
WebBrowser1.Navigate(ExtractFilePath(Application.ExeName) +
+ 'лекции/lek10.html');
Panel2.Visible:=true;
end;

procedure TForm1.Label12Click(Sender: TObject);
begin
```

```
WebBrowser1.Navigate(ExtractFilePath(Application.ExeName) +
+ 'лекции/lek11.html');
Panel2.Visible:=true;
end;

procedure TForm1.Label13Click(Sender: TObject);
begin
WebBrowser1.Navigate(ExtractFilePath(Application.ExeName) +
+ 'лекции/lek12.html');
Panel2.Visible:=true;
end;

procedure TForm1.Label14Click(Sender: TObject);
begin
WebBrowser1.Navigate(ExtractFilePath(Application.ExeName) +
+ 'лекции/lek13.html');
Panel2.Visible:=true;

end;

procedure TForm1.Label15Click(Sender: TObject);
begin
WebBrowser1.Navigate(ExtractFilePath(Application.ExeName) +
+ 'лекции/lek14.html');
Panel2.Visible:=true;
end;

procedure TForm1.Label16Click(Sender: TObject);
begin
WebBrowser1.Navigate(ExtractFilePath(Application.ExeName) +
+ 'лекции/lek15.html');
```

```
Panel2.Visible:=true;
end;

procedure TForm1.Label17Click(Sender: TObject);
begin
WebBrowser1.Navigate(ExtractFilePath(Application.ExeName) +
+ 'лекции/lek16.html');
Panel2.Visible:=true;
end;

procedure TForm1.Label18Click(Sender: TObject);
begin
WebBrowser1.Navigate(ExtractFilePath(Application.ExeName) +
+ 'лекции/lek17.html');
Panel2.Visible:=true;
end;

procedure TForm1.Label19Click(Sender: TObject);
begin
WebBrowser1.Navigate(ExtractFilePath(Application.ExeName) +
+ 'лекции/lek18.html');
Panel2.Visible:=true;
end;

procedure TForm1.Label20Click(Sender: TObject);
begin
WebBrowser1.Navigate(ExtractFilePath(Application.ExeName) +
+ 'лекции/lek19.html');
Panel2.Visible:=true;
end;
```

```
procedure TForm1.Label22Click(Sender: TObject);
begin
WebBrowser1.Navigate(ExtractFilePath(Application.ExeName) +
+ 'лабораторные работы/lab1.html');
Panel2.Visible:=true;
end;
procedure TForm1.Label23Click(Sender: TObject);
begin
form2.showmodal;
end;

procedure TForm1.Label24Click(Sender: TObject);
begin
WebBrowser1.Navigate(ExtractFilePath(Application.ExeName) +
+ 'лабораторные работы/lab2.html');

Panel2.Visible:=true;
end;
procedure TForm1.Label25Click(Sender: TObject);
begin
form3.showmodal;
end;

procedure TForm1.Label26Click(Sender: TObject);
begin
WebBrowser1.Navigate(ExtractFilePath(Application.ExeName) +
+ 'лабораторные работы/lab3.html');
Panel2.Visible:=true;
end;
procedure TForm1.Label27Click(Sender: TObject);
begin
```

```
Form4.showmodal;
end;


procedure TForm1.Label28Click(Sender: TObject);
begin
WebBrowser1.Navigate(ExtractFilePath(Application.ExeName) +
+ 'лабораторные работы/lab4.html');
Panel2.Visible:=true;
end;
procedure TForm1.Label29Click(Sender: TObject);
begin
form5.showmodal;
end;

procedure TForm1.Label30Click(Sender: TObject);
begin
WebBrowser1.Navigate(ExtractFilePath(Application.ExeName) +
+ 'лабораторные работы/lab5.html');
Panel2.Visible:=true;
end;
procedure TForm1.Label31Click(Sender: TObject);
begin
 form6.showmodal;
end;

procedure TForm1.Label32Click(Sender: TObject);
begin
WebBrowser1.Navigate(ExtractFilePath(Application.ExeName) +
+ 'лабораторные работы/lab6.html');
Panel2.Visible:=true;
```

```
end;
procedure TForm1.Label33Click(Sender: TObject);
begin
form7.showmodal;
end;

procedure TForm1.Label34Click(Sender: TObject);
begin
WebBrowser1.Navigate(ExtractFilePath(Application.ExeName) +
+ 'лабораторные работы/lab7.html');
Panel2.Visible:=true;
end;
procedure TForm1.Label35Click(Sender: TObject);

begin
form8.showmodal;
end;

procedure TForm1.Label36Click(Sender: TObject);
begin
WebBrowser1.Navigate(ExtractFilePath(Application.ExeName) +
+ 'лабораторные работы/lab8.html');
Panel2.Visible:=true;
end;
procedure TForm1.Label37Click(Sender: TObject);
begin
form9.showmodal;
end;

procedure TForm1.Label38Click(Sender: TObject);
begin
```

```
WebBrowser1.Navigate(ExtractFilePath(Application.ExeName) +
+ 'лабораторные работы/lab9.html');
Panel2.Visible:=true;
end;
procedure TForm1.Label39Click(Sender: TObject);
begin
form10.showmodal;
end;

procedure TForm1.Label40Click(Sender: TObject);
begin
WebBrowser1.Navigate(ExtractFilePath(Application.ExeName) +

+ 'лабораторные работы/lab10.html');
Panel2.Visible:=true;
end;
procedure TForm1.Label41Click(Sender: TObject);
begin
form11.showmodal;
end;
end.
```