Автономная некоммерческая образовательная организация
высшего образования
«Воронежский экономико-правовой институт»
(АНОО ВО «ВЭПИ»)

АНОО ВО "ВОРОНЕЖСКИЙ ЭКОНОМИКО-ПРАВОВОЙ ИНСТИТУТ", АНОО ВО "ВЭПИ", ВЭПИ

Подписано цифровой подписью: АНОО ВО "ВОРОНЕЖСКИЙ ЭКОНОМИКО-ПРАВОВОЙ ИНСТИТУТ", АНОО ВО "ВЭПИ", ВЭПИ
Дата: 2021.01.01 10:05:31 +03'00'

УТВЕРЖДАЮ
Проректор
по учебно-методической работе
_____ А.Ю. Жильников
«__» _____ 20__ г.

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ ПО ДИСЦИПЛИНЕ (МОДУЛЮ)

Б1.О.21 Высокоуровневые методы информатики и программирования
(наименование дисциплины (модуля))

09.03.03 Прикладная информатика
(код и наименование направления подготовки)

Направленность (профиль) Прикладная информатика в экономике
(наименование направленности (профиля))

Квалификация выпускника Бакалавр
(наименование направленности (профиля))

Форма обучения Очная, заочная
(очная, заочная)

Рекомендованы к использованию Филиалами АНОО ВО «ВЭПИ»

Воронеж 2018

Методические рекомендации по выполнению лабораторных работ по дисциплине (модулю) рассмотрены и одобрены на заседании кафедры прикладной информатики.

Протокол от «13» декабря 2018 г. № 5

Заведующий кафедрой  Г.А. Курина

Разработчики:

Доцент  В.А. Скляров

ЛАБОРАТОРНЫЕ РАБОТЫ

Лабораторная работа № 1
«Законы эволюции программного обеспечения»

Цель работы: знать законы эволюции программного обеспечения.

1. Краткие теоретические сведения

Программирование – это молодая и быстро развивающаяся отрасль науки и техники. Основными этапами развития программирования как науки являются следующие:

- неструктурированное «стихийное программирование»;
- процедурное, модульное программирование;
- объектно-ориентированное программирование;
- компонентное программирование.

До середины 60-ых годов существовала неструктурированная, «стихийная» технология программирования. Несовершенство такой технологии программирования проявилось в отсутствии четких методов проектирования подпрограмм и появлении большого количества ошибок при сборке программного продукта. Лишь некоторые из языков получили тогда широкое применение (FORTRAN, ALGOL, COBOL).

В результате исследовательских работ 60-70-ых годов XX в. была разработана технология процедурного программирования, внесшая ясность в написание программ, простоту тестирования и отладки, легкость модификации. *Процедурное программирование* основано на модели построения программы как иерархии процедур, отсюда и название метода. Известные процедурные языки программирования – это PL1, ALGOL-68, Pascal, C, C++.

При проектировании и реализации информационных систем в экономике и управлении проявились недостатки технологии процедурного программирования:

- программы не всегда объективно отражают объекты реального мира, и поэтому не могут повторно использоваться;
- обнаружились сложности сопровождения и модификации больших программ.

В 80-ых годах Б. Страуструпом был разработан язык C++, обеспечивший возможность объектно - ориентированного подхода к программированию. Технология объектно-ориентированного программирования основывается на модели построения программы как иерархии классов, представлении программы как совокупности объектов – экземпляров определенных классов.

Несмотря на большие преимущества, опыт использования технологии объектно-ориентированного программирования выявил недостатки при компоновке объектов компиляции, полученных разными компиляторами

языка. Отсутствовали стандарты компоновки двоичных результатов компиляции объектов в единое целое даже в пределах одного языка программирования с разными компиляторами. Потребность в разработке технологий, устраняющих указанные недостатки и убыстряющих написание программ, привела к появлению в настоящее время компонентных технологий программирования и CASE-технологий.

2. Порядок выполнения работы и содержание отчета

Порядок выполнения работы:

1. Концепция объектно-ориентированного программирования.
2. Классы объектно-ориентированного программирования.
3. Объекты объектно-ориентированного программирования.
4. Методы объектно-ориентированного программирования.

Содержание отчета:

1) цель работы;
2) задание на лабораторную работу для своего варианта;
3) алгоритм решаемого задания с необходимыми пояснениями;
4) выводы по работе.

3. Контрольные вопросы

1. Концепция объектно-ориентированного программирования.
2. Методы объектно-ориентированного программирования.

Лабораторная работа № 2
«Программирование в средах современных информационных систем: создание модульных программ, элементы теории модульного программирования, объектно-ориентированное проектирование и программирование»

Цель работы: рассмотреть процесс программирования в средах современных информационных систем: создание модульных программ, элементы теории модульного программирования, объектно-ориентированное проектирование и программирование.

1. Краткие теоретические сведения

Программирование представляет собой сферу действий, направленную на создание программ. Программирование может рассматриваться как наука и как искусство. В свою очередь программа - это последовательность команд компьютера, приводящая к решению задачи. Программа является результатом интеллектуального труда, для которого характерно творчество. Программы предназначены для машинной реализации задач. Задача представляет собой часть проблемы, подлежащей решению с помощью технических средств, а приложение (синоним программа) - реализованное на компьютере решение данной задачи. Таким образом, можно заключить, что приложение - это программная реализация решения задачи на компьютере. В свою очередь программное обеспечение (ПО) является совокупностью программных продуктов и технической документации к ним, а программный продукт (ПП) - это комплекс взаимосвязанных программ, предназначенный для реализации определенной задачи массового спроса. Программы являются критерием развития вычислительной техники, они делятся на утилиты (для нужд разработчиков) и программные продукты (для удовлетворения потребностей пользователя).

В настоящее время при создании программных продуктов возникает ряд проблем, основными из которых являются следующее:

1. Быстрая смена вычислительной техники и алгоритмических языков.
2. Не стыковка ЭВМ друг с другом (VAX и IBM).
3. Отсутствие полного взаимопонимания между заказчиком и исполнителем к разработанному программному продукту.

2. Порядок выполнения работы и содержание отчета

Порядок выполнения работы:

1. Состав и назначение окон среды Delphi.
2. Функции строки меню и панели быстрого доступа.
3. Рекомендуемый порядок работы в среде Delphi.
4. Создание, сохранение и открытие проекта.

Содержание отчета:

1) цель работы;

2) задание на лабораторную работу для своего варианта;
3) алгоритм решаемого задания с необходимыми пояснениями;
4) выводы по работе.

3. Контрольные вопросы

1. Состав и назначение окон среды Delphi.
2. Создание, сохранение и открытие проекта.

Лабораторная работа № 3
«Объектно-ориентированный подход к проектированию и разработке программ, сущность объектно-ориентированного подхода; объектный тип данных; переменные объектного типа; инкапсуляция; наследование; полиморфизм; классы и объекты»

Цель работы: рассмотреть объектно-ориентированный подход к проектированию и разработке программ, сущность объектно-ориентированного подхода; объектный тип данных; переменные объектного типа; инкапсуляция; наследование; полиморфизм; классы и объекты.

1. Краткие теоретические сведения

Объектно-ориентированное проектирование - это часть ОО методологии, которая предоставляет возможность программистам оперировать понятием объект, нежели процедур при разработке своего кода. Объекты содержат данные и процедуры, сгруппированные вместе, отображая т.о. сущность объекта. «Интерфейс объекта», описывает взаимодействие с объектом, то, как он определен. Программа, полученная при реализации объектно-ориентированного исходного кода, описывает взаимодействие этих объектов.

Объектно-ориентированное программирование - технология программирования, при которой программа рассматривается как набор дискретных объектов, содержащих, в свою очередь, наборы структур данных и процедур, взаимодействующих с другими объектами.

Характеристики ООП - Вычисления осуществляются путем взаимодействия (обмена данными) между объектами, при котором один объект требует, чтобы другой объект выполнил некое действие. Каждый объект имеет независимую память, которая состоит из других объектов.

Каждый объект является представителем класса, который выражает общие свойства объектов.

В классе задается поведение (функциональность) объекта. Классы организованы в единую древовидную структуру с общим корнем, называемую иерархией наследования. Объектно-ориентированный язык должен обладать свойствами *абстракции*, *инкапсуляции*, *наследования и полиморфизма*. Принципы Объектно-ориентированного подхода.

Действие в объектно-ориентированном программировании инициируется посредством передачи сообщений объекту. Все объекты являются экземплярами, классов. Принцип наследования. Принцип полиморфизма.

2. Порядок выполнения работы и содержание отчета

Порядок выполнения работы:

1. Создание форм, установка и изменение их свойств.
2. Создание в проекте новой формы.

3. Общие сведения о компонентах.
4. Общие свойства управляющих элементов.

Содержание отчета:

1) цель работы;
2) задание на лабораторную работу для своего варианта;
3) алгоритм решаемого задания с необходимыми пояснениями;
4) выводы по работе.

3. Контрольные вопросы

1. Создание форм, установка и изменение их свойств.
2. Общие свойства управляющих элементов.

Лабораторная работа № 4
«Конструкторы и деструкторы»

Цель работы: рассмотреть конструкторы и деструкторы.

1. Краткие теоретические сведения

Класс может содержать любое количество функций - членов самого разнообразного назначения, но два типа функций занимают особое положение. Эти функции называются конструктором и деструктором.

Для многих объектов естественно требовать, чтобы они были инициированы (т.е. имели начальное значение, а не мусор) до начала их использования.

В C++ для упрощения процесса инициализации объектов предусмотрена специальная функция, называемая конструктором.

Конструктор – это компонентная функция, вызываемая автоматически при создании объекта класса и выполняющая необходимые инициализирующие действия.

1. Основное назначение – инициализация объектов.

- инициализация данных класса - задание им начальных значений программно или по умолчанию;
- открытие файлов;
- перевод видеосистемы в графический режим;
- вывод сообщения;
- инициализация объектов вспомогательных классов и. т. д.

2. Имя конструктора должно совпадать с именем класса.

3. Функция-конструктор не может возвращать результат, даже тип void не допустим.

4. Функция автоматически вызывается при определении объекта, или при размещении в памяти объекта с помощью операции new.

5. Формат определения конструктора в теле класса:

< имя класса> (список формальных параметров)

{ операторы тела конструктора }

Конструктор, как и любая компонентная функция, может быть определен и вне тела класса, имея в теле класса прототип.

```
class T {
…
public:
T ( список параметров) ;
…
} ;
T::T( список параметров) { тело конструктора}
```

6. Как правило, конструкторы объявляются в открытой части класса

7. Конструктор может отсутствовать, при создании экземпляров класса компилятор автоматически выделяет под них память, хотя в этом случае данные не инициализируются, и будут содержать мусор.

2. Порядок выполнения работы и содержание отчета

Порядок выполнения работы:

1. Стандарты компрессии / декомпрессии видеоизображения.
2. Виды видеоконференций.
3. Трехмерная графика, технология анимации.

Содержание отчета:

1) цель работы;
2) задание на лабораторную работу для своего варианта;
3) алгоритм решаемого задания с необходимыми пояснениями;
4) выводы по работе.

3. Контрольные вопросы

1. Стандарты компрессии / декомпрессии видеоизображения.
2. Трехмерная графика, технология анимации.

Лабораторная работа № 5
«Отладка и тестирование программ»

Цель работы: рассмотреть процесс отладки и тестирования программ

1. Краткие теоретические сведения

Отладка программы является итеративным процессом обнаружения и исправления ошибок и обычно требует последовательного выполнения четырех этапов:

- выявления ошибки;
- локализации ошибки в тексте программы;
- установления причины ошибки;
- исправления ошибки.

Некоторые ошибки проявляются после первого же запуска программы на выполнение, и для их обнаружения не надо прибегать ни к каким специальным средствам. Некоторые ошибки проявляются в случайные моменты работы программы. С такими ошибками справиться труднее всего – зафиксировать условия возникновения ошибки, понять причину ошибки и устранить ее. С целью обнаружения подобных ошибок осуществляется *тестирование программы*– ее выполнение для специально подобранных представительных контрольных примеров – тестов. *Тест* – это такой набор исходных данных, для которого вручную или другим способом просчитаны промежуточные и конечные результаты и который может быть использован для получения информации о надежности проверяемой программы.

Тестирование программы должно включать в себя прогон трех видов контрольных примеров: нормальных ситуаций, граничных ситуаций и случаев неправильных данных. *Нормальные случаи* – это примеры с правильными входными данными. Если программа не работает в подобных случаях, она требует серьезных переделок. Граничные контрольные примеры помогают установить, способна ли программа нормально реагировать на особые случаи во входных данных. *Граничные примеры* представляют собой данные, которые, будучи математически корректными, приводят программу к необходимости работать особым образом. *Неправильными* являются такие *данные*, которые расположены вне допустимого диапазона. Примеры с неправильными данными должны быть обработаны соответствующим образом, поскольку в повседневной эксплуатации программе придется иметь дело и с неверными входными данными.

После того как ошибка обнаружена, необходимо найти в исходном тексте программы то место, в котором она возникала, – *локализовать ошибку*. Можно использовать ряд различных методов отладки, позволяющих обнаружить расположение ошибки; выбор существенно зависит от особенностей ситуации. Большинство программистов начинают с неформального метода, известного под названием *проверка за*

столом. Используя контрольный пример, который привел к ошибке в программе, программист аналитически трассирует листинг программы в надежде локализовать ошибку. Проверка за столом – это хороший метод, поскольку он заставляет программиста детально понять работу программы. Если применение метода проверки за столом оказалось бесплодным, нужно использовать специальные методы и способы отладки, позволяющие наблюдать за передачей управления в программе и за изменением значений наиболее важных переменных. Полученная отладочная информация позволит локализовать подозрительные ситуации, провести анализ и выявить причину ошибки, устранить ее, а затем продолжить поиск других ошибок.

2. Порядок выполнения работы и содержание отчета

Порядок выполнения работы:

1. Компоненты создания интерфейса пользователя.
2. Компоненты ввода и отображения текста.
3. Пример использования визуальных компонентов в приложении.
4. Основы создания компонентов.
5. Написание компонентов и их установка в среде визуального программирования данных.

Содержание отчета:

1) цель работы;
2) задание на лабораторную работу для своего варианта;
3) алгоритм решаемого задания с необходимыми пояснениями;
4) выводы по работе.

3. Контрольные вопросы

1. Написание компонентов и их установка в среде визуального программирования данных.

Лабораторная работа № 6
«Основы визуального программирования»

Цель работы: знать основы визуального программирования.

1. Краткие теоретические сведения

Программирование в Delphi строится на тесном взаимодействии двух процессов: процесса конструирования визуального проявления программы (т. е. ее Windows – окна) и процесса написания кода, придающего элементам этого окна и программе в целом необходимую функциональность. Для написания кода используется окно кода, для конструирования программы – остальные окна Delphi, и, прежде всего, окно формы.

Между содержимым окон формы и кода существует неразрывная связь, которая строго отслеживается Delphi. Это означает, что размещение на форме компонента приводит к автоматическому изменению кода программы, и наоборот – удаление тех или иных автоматически вставленных фрагментов кода может привести к удалению соответствующих компонентов. Обычно программисты вначале конструируют форму, размещая на ней компоненты, а уже после этого переходят (если это необходимо) к написанию фрагмента кода программы.

Для создания нового проекта нужно воспользоваться пунктом меню *File\New\Application*; для создания новой формы: *File\New\Form* (при этом будет создан модуль, соответствующий новой форме); для создания нового модуля, не привязанного ни к одной форме: *File\New\Unit*.

Нужно помнить, что при запуске Delphi создается новый проект, который содержит одну пустую форму (с соответствующим ей файлом кода программы) и создавать еще один проект *не требуется*.

Модули в Delphi. В первом приближении можно считать модулем самостоятельный раздел программы, в чем-то подобный главе в книге. Модуль создается каждый раз, когда задают новую форму (в программе может быть не одна, а несколько форм и связанных с ними модулей). При компиляции Delphi создает файлы с расширениями PAS, DFM, DCU для каждого модуля:

- PAS – файл содержит копию текста из окна кода программы.
- DFM – хранит описание содержимого окна формы.
- DCU – результат преобразования в машинные инструкции текста из обоих файлов. Файлы DCU создаются компилятором. Эти файлы являются необходимыми для работы компоновщика, который преобразует их в исполняемый EXE – файл.

2. Порядок выполнения работы и содержание отчета

Порядок выполнения работы:

1. Сохранение с потерей и без потери информации.

2. Текстовые файлы.
3. Гипертекст.

Содержание отчета:

1) цель работы;
2) задание на лабораторную работу для своего варианта;
3) алгоритм решаемого задания с необходимыми пояснениями;
4) выводы по работе.

3. Контрольные вопросы

1. Сохранение с потерей и без потери информации.
2. Гипертекст.

Лабораторная работа № 7
«Создание меню и организация стандартного диалога»

Цель работы: научиться создавать меню и организацию стандартного диалога.

1. Краткие теоретические сведения

Для создания главной формы в меню системы Delphi выбирается команда File|New Application. Значение свойства Name формы можно задать как fmMAINFORM, а значение свойства Caption -*Пример интерфейса*.

Чтобы включить меню в главную форму, нужно поместить в нее компонент MainMenu, находящийся на странице Standard палитры компонентов, и вызвать конструктор меню, дважды щелкнув на компоненте MainMenu.

Методика построения меню проста. Разработчику всегда доступен пустой пункт меню. Выбрав его при помощи мыши или клавиш управления курсором, в окне Инспектора объектов нужно задать значения свойств Caption (название пункта меню), Name (имя пункта), Shortcut (комбинация клавиш быстрого выбора). Разновидностью клавиш быстрого выбора является акселератор, который набирается как комбинация клавиш Alt и подчеркнутого символа в названии пункта главного меню или только как подчеркнутый символ в названии пункта дополнительного меню. Акселератор задается знаком амперсанда (&) перед символом в названии пункта меню, являющимся значением свойства Caption.

Пункт-разделитель в меню задается символом «минус» (-) в качестве значения свойства Caption. Выход из конструктора меню задается двойным щелчком на кнопке вызова системного меню.

2. Порядок выполнения работы и содержание отчета

Порядок выполнения работы:

1. Методы логического проектирования баз данных реляционного типа.
2. Нормализация отношений.
3. Правила и методика преобразования концептуальной модели в схему реляционной базы данных.

Содержание отчета:

1) цель работы;
2) задание на лабораторную работу для своего варианта;
3) алгоритм решаемого задания с необходимыми пояснениями;
4) выводы по работе.

3. Контрольные вопросы

1. Методы логического проектирования баз данных реляционного типа.
2. Правила и методика преобразования концептуальной модели в схему реляционной базы данных.

Лабораторная работа № 8
«Формирование и печать изображений»

Цель работы: рассмотреть процесс формирования и печати изображений.

1. Краткие теоретические сведения

В Delphi у формы Form имеется функция Print, которая может выводить на печать форму.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Form1.Print;
end;
```

Также Вы можете использовать свойство PrintScale, которое изменяет масштаб печатаемого объекта. Это свойство содержит три опции:

- poNone - печать будет произведена с настройками принтера.
- poProportional - печать страницы, которая будет иметь те же размеры, что и на экране.
- poPrintToFit - размер изменяется в зависимости от размера страницы.

При этом получается не самое высокое качество печати.

Печать управления TRichEdit

Функция Print имеется также у компонента TRichEdit.

```
procedure TForm1.Button2Click(Sender: TObject);
begin
  RichEdit1.Print('Имя документа');
end;
```

Как видите, здесь функция Print получает один строковой параметр, который будет отображаться в очереди печати как имя документа.

Печать текстовых файлов при помощи ShellExecute

Также можно использовать функцию API ShellExecute для печати текстового документа.

```
uses
 ShellApi
procedure TForm1.Button3Click(Sender: TObject);
begin
  ShellExecute(
     Handle, 'Print', 'C:\Test.txt',
     nil, nil, SW_Hide);
end;
```

При этом открывается приложение, которое ассоциировано с файлом. В моем случае это Блокнот.

Диалоговое окно печати

В Delphi имеется два диалоговых окна для печати: диалоговое окно при помощи компонента TPrintDialog и при помощи диалогового окна установок принтера TPrinterSetupDialog.

Диалоговое окно TPringDialog

Компонент диалогового окна TPrintDialog Вы можете использовать непосредственно перед началом печати. Компонент TPrintDialog имеет свои свойства и метода, которые Вы сможете найти в справке по Delphi. Диалоговое окно вызывается конструкцией:

```
procedure TForm1.Button4Click(Sender: TObject);
begin
  PrintDialog1.Execute;
end;
```

2. Порядок выполнения работы и содержание отчета

Порядок выполнения работы:

1. Перенос глобальной логической модели в среду целевой СУБД.
2. Проектирование физического представления базы данных.
3. Разработка механизмов защиты.
4. Организация мониторинга и настройка функционирования системы.

Содержание отчета:

1) цель работы;
2) задание на лабораторную работу для своего варианта;
3) алгоритм решаемого задания с необходимыми пояснениями;
4) выводы по работе.

3. Контрольные вопросы

1. Перенос глобальной логической модели в среду целевой СУБД.
2. Организация мониторинга и настройка функционирования системы.