МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«Дальневосточный федеральный университет»
(ДВФУ)

ИНЖЕНЕРНАЯ ШКОЛА

«СОГЛАСОВАНО»	«УТВЕРЖДАЮ»
Руководитель ОП «Прикладная механика»	Заведующий кафедрой Механики и математического моделирования (название кафедры)
_____ Озерова Г.П (подпись) (Ф.И.О. рук.ОП)	Бочарова А.А (подпись) (Ф.И.О. зав. каф.)
«22»июня 2016г.	«22» июня 2016г.

РАБОЧАЯ ПРОГРАММА УЧЕБНОЙ ДИСЦИПЛИНЫ (РПУД)

ПРОГРАММИРОВАНИЕ В ИНЖЕНЕРНЫХ ЗАДАЧАХ

Направление подготовки: 15.03.03 Прикладная механика

Профиль подготовки:

«Математическое и компьютерное моделирование механических систем и процессов»

Форма подготовки (очная)

курс 2 семестр 3, 4
лекции 36 час.
практические занятия - час.
лабораторные работы 72 час.
в том числе с использованием МАО лек.8 час. /пр.- час. /лаб.32 час.
всего часов аудиторной нагрузки 108 час.
в том числе с использованием МАО 40 час.
самостоятельная работа 72 час.
в том числе на подготовку к экзамену 45 час.
контрольные работы -
курсовая работа / курсовой проект -
зачет – 3 семестр
экзамен 4 семестр

Рабочая программа составлена в соответствии с требованиями федерального государственного образовательного стандарта высшего образования Дальневосточного федерального университета, принятого решением Ученого совета ДВФУ, протокол от 25.02.2016 № 02-16, введённого в действие приказом ректора ДВФУ от 10.03.2016 № 12-13-391

Рабочая программа обсуждена на заседании кафедры Механики и математического моделирования, протокол № 9 от «21» июня 2016 г.

Заведующий кафедрой:к.ф.-м.н., проф. Бочарова А.А.
Составитель: к.т.н., доцент Озерова Г.П.

Оборотная сторона титульного листа РПУД

I. Рабочая программа пересмотрена на заседании кафедры:

Протокол от «23» _ июня 2017 г. № 11

Заведующий кафедрой _____ Бочарова А.А.

(подпись) (И.О. Фамилия)

В целях повышения качества знаний студентов в РПУД внесены следующие изменения:

- добавлены расчетно-графические работы (РГР)

II. Рабочая программа пересмотрена на заседании кафедры:

Протокол от «_____» _____ 20___ г. № _____

Заведующий кафедрой _____ _____

(подпись) (И.О. Фамилия)

ABSTRACT

Bachelor's degree in 15.03.03 Applied Mechanics

Study profile "Mathematical and Computer Modeling Mechanical Systems and Processes"

Course title: Programming in Eengineering Problems

Variable part of Block 1, 5credits

Instructor: Ozerova Galina Pavlovna

At the beginning of the course a student should be able to:

- should know how to use different information sources: books, textbooks, references, internet;
- should have an algorithmic mind;
- can use computational machines;
- understanding basics of structural and procedural programming.

Learning outcomes:

GPC 9 – the possession of methods in the information technology, the information security basic requirements, including state secrets protection;

GPC 10 – the ability to solve common tasks of professional activities on the basis of bibliographic information and culture using information and communication technologies, taking into account the main requirements for information security;

PC 7 – the ability the ability to search relevant scientific and technical information from electronic resources, including Science Direct, Elsiver Freedom Collection, SCOPUS.

Course description: The content of the course covers issues related to technology design and implementation applications for different purposes. The course topics are Programming technology, Event-driven programming, Object-oriented programming and Visual Development frameworks. Theoretical and practical knowledge of programming will allow students more confident to navigate the complex and diverse phenomena of the modern development of the

information society, as well as some practical skills in the use of computer technology in teaching, research and experimental activities.

Main course literature:

1. Trofimov V.V. Informatics. Textbook for high schools. - M: Yurayt 2010. -911 p. http://lib.dvfu.ru:8080/lib/item?id=chamo:356824&theme=FEFU

2. Pavlovskaya T.A .Pascal. Programming in high-level programming language: textbook for high schools. - St. Petersburg: Peter, 2008. - 393 p. http://lib.dvfu.ru:8080/lib/item?id=chamo:384229&theme=FEFU

3. Alekseev E.R. Free Pascal and the Lazarus: Programming Tutorial [electronic resource]: a textbook / ER Alekseev, OV Chesnokov, TV Coachman. - Moscow: DMK Press, 2010. - 438 p. - Access: http://e.lanbook.com/books/element.php?pl1_id=1267

3. Algorithmic and programming: Textbook / SA Kantsedal. - M .: FORUM: SIC INFRA-M, 2013. - 352 p. http://znanium.com/go.php?id=391351

4. Nemtsov T.I. Programming in high-level programming language. Programming in Object Pascal: Tutorial / TI Nemtsov; Ed. LG Gagarina. - M .: FORUM: INFRA-M, 2015. - 496 p. http://znanium.com/bookread.php?book=472870

Form of final knowledge control: pass-fail exam, exam.

АННОТАЦИЯ

Учебная дисциплина «Программирование в инженерных задачах» предназначена для студентов 2 курса, обучающихся по направлению 15.03.03 «Прикладная механика», профиль «Математическое и компьютерное моделирование механических систем и процессов». Дисциплина входит в вариативную часть блока «Дисциплины», является дисциплиной по выбору. Дисциплина «Программирование в инженерных задачах» логически и содержательно связана с такими курсами как «Численные методы в механике», «Вычислительная механика», «Основы конечно-элементного анализа», «Инженерные web-технологии».

Общая трудоемкость освоения дисциплины составляет 180 часов. Учебным планом предусмотрены лекционные занятия (36 часов), лабораторные работы (72 часа), самостоятельная работа студента (72 часа). Дисциплина реализуется на 2 курсе в 3 и 4 семестрах.

Цель: Изучить общие принципы современных технологий программирования: процедурного, событийного, объектно-ориентированного программирования и получить навыки практического применения этих технологий при создании сложных программных комплексов в инженерных задачах.

Задачи:

1. Сформировать систематизированное представление о концепциях, моделях и принципах организации, положенных в основу технологий программирования.
2. Познакомить с основными понятиями, особенностями и преимуществами объектно-ориентированной технологии программирования.
3. Дать основы событийного программирования.
4. Выработать навыки разработки приложений в визуальных средах программирования.
5. Дать методику, позволяющую свободно изучать и применять новые программные системы.

Для успешного изучения дисциплины «Программирование в инженерных задачах» у обучающихся должны быть сформированы следующие предварительные компетенции:

- владение навыками работы с различными источниками информации: книгами, учебниками, справочниками, Интернет;
- способность к алгоритмическому мышлению;
- владение навыками работы с вычислительной техникой;
- владение основами структурного и процедурного программирования.

В результате изучения данной дисциплины у обучающихся формируются следующие общепрофессиональные и профессиональные компетенции.

Код и формулировка компетенции	**Этапы формирования компетенции**	
ОПК-9 – владением методами информационных технологий, соблюдением основных требований информационной безопасности, в том числе защиты государственной тайны	Знает	особенности современных методологий и технологий использования и создания программных средств
	Умеет	проектировать, конструировать и отлаживать программные продукты в соответствии с заданными критериями качества и стандартами
	Владеет	навыками работы с компьютером как средством управления информацией;
ОПК-10 – способностью решать стандартные задачи профессиональной деятельности на основе информационной и библиографической культуры с применением информационно-коммуникационных технологий и с учетом основных требований информационной безопасности	Знает	основы событийного и объектно-ориентированного программирования
	Умеет	разрабатывать алгоритмы и программы для решения задач профессиональной деятельности средствами высокоуровневых языков программирования и визуальных систем программирования.
	Владеет	способностью использовать для решения аналитических, исследовательских и коммуникативных задач современные средства визуальной разработки программ.
ПК-7– умением извлекать актуальную научно-техническую информацию и наукометрическую информацию из	Знает	тенденции развития средств вычислительной техники и особенности их применение в профессиональной деятельности

электронных ресурсов, в том числе Science Direct, Elsiver Freedom Collection, SCOPUS	Умеет	использовать поисковые системы сети Интернет для отбора информации из наукометрических баз данных
	Владеет	- навыками работы с компьютером как средством управления информацией; - методологией сбора, обработки анализа и обобщения научно-технической информации

Для формирования вышеуказанных компетенций в рамках дисциплины «Программирование в инженерных задачах» применяются следующие методы активного/ интерактивного обучения:

- лекция пресс-конференция;
- лекция «вдвоем»;
- деловая игра;
- «мозговой штурм»;
- групповая консультация.

I. СТРУКТУРА И СОДЕРЖАНИЕ ТЕОРЕТИЧЕСКОЙ ЧАСТИ КУРСА

Раздел I. Основы технологии программирования (14 час.)

Тема 1. Парадигмы программирования (2час.)

Парадигмы программирования: императивное программирование, декларативное программирование, структурное программирование, функциональное программирование, логическое программирование, объектно-ориентированное программирование. Поколения языков программирования. Классификация языков программирования. Системы программирования. Этапы разработки программного продукта.

Тема 2. Алгоритмы (2час.)

Алгоритмы. Понятие алгоритма, его свойства. Способы описания алгоритмов. Структуры алгоритмов. Стандартные приемы алгоритмизации.

Структурный подход к разработке алгоритмов. Краткая характеристика основных структур. Свойства алгоритмов. Пошаговый метод разработки алгоритма. Процедуры и функции. Примеры структурных программ.

Тема 3. Модульное программирование (4час.)

Структура модулей. Заголовок модулей и связь модулей. Интерфейсная часть. Исполняемая часть. Инициирующая и завершающая части. Доступ к объявленным в модуле объектам. Типы модулей. Примеры разработки модулей разного назначения.

Тема4. Визуальная среда разработки программ (2час.)

Структура визуальной среды разработки. Проект и его характеристики. Создание проекта с формой. Организация графического пользовательского интерфейса через формы. Структура формы. Размещение нового компонента. Свойства компонент. Корректировка свойств компонентов. Примеры создания проектов.

Тема 5. Применение визуальных компонент (4 час.)

Общие свойства компонентов. Библиотека визуальных компонентов. Обзор основных компонентов визуальной среды программирования. Использование визуальных компонентов. Отображение текста. Ввод и редактирование информации. Работа со списками. Работа с кнопками. Использование переключателей. Объединение элементов управления.

Раздел II.Технология событийного программирования(6 час.)

Тема 1. Понятие события в системах программирования (4час.)

Свойства и события объектов. Типы событий. Способы реагирования на события. События клавиатуры и мыши. События. Связанные с компонентами. Реализация интерактивных взаимодействий программы с пользователем. Примеры программ реализации интерактивного пользовательского интерфейса.

Тема 2. Организация взаимодействия с пользователем через события (2час.)

Пользовательский интерфейс. Способы его организации. Проектирование интерфейсов. Графический пользовательский интерфейс. Обработка действий пользователя через события. Проектирования приложения пользователя для решения различных задач предметной области.

Раздел III. Технология объектно-ориентированного программирования (16 часов)

Тема 1. Объектно-ориентированные средства языков программирования (4час.)

Объекты, классы. Инкапсуляция данных и методы доступа. Средства ограничения доступа. Интерфейс и реализация. Примеры простых программ с использованием классов.

Переопределение методов. Виртуальные и статические методы. Примеры простых программ с иерархией классов.

Тема 2. Основные концепции объектно-ориентированного программирования. (4час.)

Инкапсуляция, наследование, полиморфизм. Объектно-ориентированный подход к разработке программ. Инициализация объектов класса. Конструкторы. Очистка объектов класса. Деструкторы Примеры программ динамического создания и управления объектами.

Тема 3. Классы обработки исключений и списки. (4час.)

Возбуждение и обработка исключений. Поддержка иерархии классов. Стандартные классы обработки исключительных ситуаций. Примеры простых программ с использованием исключительных ситуаций. Объекты и классы, используемые для создания и обработки динамических списков. Методы классов. Примеры программ со списками.

Тема 4. Графический инструментарий. (4час.)

Модули, объекты и классы, описывающие изображения растровой графики. Преобразование изображений: масштабирование, поворот, перемещение. Реализация мультипликации.

II. СТРУКТУРА И СОДЕРЖАНИЕ ПРАКТИЧЕСКОЙ ЧАСТИ КУРСА

Лабораторные работы (72час.)

Лабораторная работа 1. Разработка проекта «Угадай число» (4 час.)

Лабораторная работа 2. Разработка проекта «Геометрические вычисления на плоскости»(4 час.)

Лабораторная работа 3. Разработка проекта «Калькулятор комплексных чисел»(4 час.)

Лабораторная работа 4. Разработка проекта «Калькулятор смешанных дробей»(4 час.)

Лабораторная работа 5. Обработка событий (4 час.)

Лабораторная работа 6. Реализация проекта «Конструктор рисованных объектов из готовых форм» (4 час.)

Лабораторная работа 7. Реализация проекта «Отображение и корректировка табличных данных» (8 час.)

Лабораторная работа 8. Реализация проекта «Арифметические операции над матрицами и векторами» (4 час.)

Лабораторная работа 9. Реализация проекта «Элементарные матричные операции» (4 час.)

Лабораторная работа 10. Основы объектно-ориентированного программирования(4 час.)

Лабораторная работа 11. Реализация проекта «Создание иерархии классов» (4час.)

Лабораторная работа 12. Реализация проекта «Классы исключений» (4час.)

Лабораторная работа 13. Реализация проекта «Списки» (4час.)

Лабораторная работа 14. Реализация проекта «Программное рисование» (4 час.)

Лабораторная работа 15. Реализация проекта «Движение объекта по заданному закону» (4 час.)

Лабораторная работа 16. Реализация проекта «Физическое и механическое движение» (4 час.)

Лабораторная работа 17. Реализация проекта «Объектно-ориентированный графический редактор» (4 час.)

Лабораторная работа 18. Расширение возможностей проекта «Объектно-ориентированный графический редактор» по вариантам (4 час.)

III. УЧЕБНО-МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ САМОСТОЯТЕЛЬНОЙ РАБОТЫ ОБУЧАЮЩИХСЯ

Учебно-методическое обеспечение самостоятельной работы обучающихся по дисциплине «Программирование в инженерных задачах» представлено в Приложении 1 и включает в себя:

– план-график выполнения самостоятельной работы по дисциплине, в том числе примерные нормы времени на выполнение по каждому заданию;

– характеристика заданий для самостоятельной работы обучающихся и методические рекомендации по их выполнению;

– требования к представлению и оформлению результатов самостоятельной работы;

– критерии оценки выполнения самостоятельной работы.

IV. КОНТРОЛЬ ДОСТИЖЕНИЯ ЦЕЛЕЙ КУРСА

№ п/п	Контролируемые разделы / темы дисциплины	Коды и этапы формирования компетенций		Оценочные средства	
				текущий контроль	промежуточная аттестация
1	Основы технологии программирования	ОПК-9	Знает особенности современных методологий и технологий использования и создания программных средств	Тест (ПР-1)	Вопросы 1-15
			Умеет проектировать, конструировать и отлаживать программные продукты в соответствии с заданными критериями	Лабораторная работа (ПР-6)	Задачи тип I

№ п/п	Контролируемые разделы / темы дисциплины	Коды и этапы формирования компетенций		Оценочные средства	
				текущий контроль	промежуточная аттестация
			качества и стандартами		
			Владеет навыками работы с компьютером как средством управления информацией;	Расчетно-графическая работа (ПР-12)	Задачи тип II
2	Технология событийного программирования	ПК-7	Знает тенденции развития средств вычислительной техники и особенности их применение в профессиональной деятельности	Собеседование (УО-1)	Вопросы 1-7
			Умеет использовать поисковые системы сети Интернет для отбора информации из наукометрических баз данных	Лабораторная работа (ПР-6)	Задачи тип I
			владеет навыками работы с компьютером как средством управления информацией; - методологией сбора, обработки анализа и обобщения научно-технической информации	Расчетно-графическая работа (ПР-12)	Задачи тип II
3	Технология объектно-ориентированного программирования	ОПК-10	Знает основы событийного и объектно-ориентированного программирования	Собеседование (УО-1)	Вопросы 8-33-
			Умеет разрабатывать алгоритмы и программы для решения задач профессиональной деятельности средствами высокоуровневых языков программирования и визуальных систем программирования.	Лабораторная работа (ПР-6) Расчетно-графическая работа (ПР-12)	Задачи тип I
			Владеет способностью использовать для решения аналитических, исследовательских и коммуникативных задач современные средства визуальной разработки	Расчетно-графическая работа (ПР-12)	Задачи тип II

№ п/п	Контролируемые разделы / темы дисциплины	Коды и этапы формирования компетенций	Оценочные средства	
			текущий контроль	промежуточная аттестация
		программ.		

Типовые контрольные задания, методические материалы, определяющие процедуры оценивания знаний, умений и навыков и (или) опыта деятельности, а также критерии и показатели, необходимые для оценки знаний, умений, навыков и характеризующие этапы формирования компетенций в процессе освоения образовательной программы, представлены в Приложении 2.

V. СПИСОК УЧЕБНОЙ ЛИТЕРАТУРЫ И ИНФОРМАЦИОННО-МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

Основная литература

1. Трофимов В. В. Информатика. Учебник для вузов. – М: Юрайт. – 2010. –911 с http://lib.dvfu.ru:8080/lib/item?id=chamo:356824&theme=FEFU

2. Алексеев, Е.Р. Free Pascal и Lazarus: Учебник по программированию [Электронный ресурс] : учебник / Е.Р. Алексеев, О.В. Чеснокова, Т.В. Кучер. — Электрон. дан. — М. : ДМК Пресс, 2010. — 438 с. — Режим доступа: http://e.lanbook.com/books/element.php?pl1_id=1267

3. Озерова, Г.П.Pascal в примерах и задачах [Электронный ресурс] : учебное пособие для вузов / Г. П. Озерова ; Дальневосточный федеральный университет, Инженерная школа. – Владивосток: Изд-во Дальневосточного федерального университета, 2014 , 123с. http://elib.dvfu.ru/vital/access/manager/Repository/fefu:2537

4. Немцова Т. И. Программирование на языке высокого уровня. Программирование на языке Object Pascal: Учебное пособие / Т.И. Немцова; Под ред. Л.Г. Гагариной. - М.: ФОРУМ: ИНФРА-М, 2015. - 496 с. http://znanium.com/bookread.php?book=472870

Дополнительная литература

1. Озерова Г.П. Разработка приложений в среде Delphi : учеб. пособие [для вузов]/ Озерова Г.П. – Владивосток : Изд-во Тихоокеанского экономического университета, 2010. – 200 с. http://lib.dvfu.ru:8080/lib/item?id=chamo:357097&theme=FEFU

2. Эйдлина Г.М. Delphi: программирование в примерах и задачах. Практикум: Учебное пособие / Г.М. Эйдлина, К.А. Милорадов. - М.: ИЦ РИОР: НИЦ Инфра-М, 2012. - 116 с. http://znanium.com/go.php?id=319046

3. Окулов С. Основы программирования (2-е издание). - М: "Бином. Лаборатория знаний", 2008. - 440 стр. http://lib.dvfu.ru:8080/lib/item?id=chamo:274542&theme=FEFU

4. Вирт Н. Алгоритмы и структуры данных. Новая версия для Оберона [Электронный ресурс] : учебное пособие. — Электрон. дан. — М. : ДМК Пресс, 2010. — 272 с. http://e.lanbook.com/books/element.php?pl1_id=1261

Перечень ресурсов информационно-телекоммуникационной сети «Интернет»

1. Сайт свободно распространяемой системы программирования Lazarus. – Режим доступа: http://www.lazarus-ide.org/

2. Информационный портал для разработчиков на Free Pascal и Lazarus. – Режим доступа: http://www.freepascal.ru/

3. Справочник по Паскалю. - http://tpdn.ru/

Перечень информационных технологий и программного обеспечения

Необходимое программное обеспечение: свободно распространяемая система программирования Lazarus (http://www.lazarus-ide.org/).

Материалы курса размещены в LMS BlackBoard, идентификатор: FU50219-151600-PVIZ-0.

VI. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ОСВОЕНИЮ ДИСЦИПЛИНЫ

1. Описание последовательности действий студента («сценарий изучения дисциплины»).

Для успешного изучения дисциплины «Основы программирования в компьютерных системах» студенту необходимо:

1. При подготовке к лекции изучить конспект предыдущей лекции, а также при необходимости теоретический материал, представленный в системе BlackBoard. Далее ознакомиться с материалом следующей лекции (также через систему BlackBoard), вникнуть в суть изучаемой проблемы, подготовить вопросы.

2. На лекционном занятии тщательно конспектировать теоретический материал, участвовать в обсуждении, задавать вопросы.

3. При подготовке к лабораторным занятиям на основе материалов, представленных в системе BlackBoard, сначала понять задание лабораторной работы, найти теоретический материал, необходимый для работы, изучить алгоритм реализации задания, сформулировать вопросы преподавателю.

4. На лабораторном занятии сначала задать вопросы преподавателю по методике выполнения работы, затем выполнить задание «по образцу». Отправить преподавателю через систему BlackBoard выполненное задание на проверку, ответить на вопросы преподавателя. Прежде, чем приступать к выполнению самостоятельных заданий, обдумать алгоритм их реализации, задать вопросы преподавателю по сути заданий, спланировать их выполнение.

5. Самостоятельные задания лабораторной работы можно выполнять как на аудиторном занятии, так и самостоятельно во внеаудиторное время. При этом результат их реализации необходимо проверять через систему BlackBoard и только после этого отправлять преподавателю на проверку.

6. В течение недели выбрать время для работы со специальной литературой в библиотеке и для занятий на компьютере.

7. Самостоятельную работу организовывать в соответствии с графиком выполнения самостоятельной работы, приведенном в приложении 1.

2. Рекомендации по использованию материалов учебно-методического комплекса. Рекомендуется использовать методические указания и материалы по курсу «Основы программирования в компьютерных системах», размещенные в системе BlackBoard, идентификатор курса FU50219-15.03.03-OPvKS-01.

3. Рекомендации по работе с литературой. Теоретический материал курса становится более понятным, когда дополнительно к прослушиванию лекций, использованию учебно-методического комплекса, представленного в системе BlackBoard, изучаются и книги из списка основной и дополнительной литературы. Литературу по курсу можно изучать в библиотеке, брать книгу на дом или читать ее на компьютере (если это электронный ресурс). Полезно использовать несколько учебников, однако желательно придерживаться рекомендация преподавателя по выбору книг по каждому разделу. Не рекомендуется «заучивать» материал, желательно добиться понимания изучаемой темы дисциплины, а затем использовать изученный материал для реализации программ и/или сайтов. Кроме того, очень полезно выявить тенденции развития той или иной технологии разработки, выделить для себя направления дальнейшего изучения материала, для достижения более продвинутого уровня изучения дисциплины.

4. Рекомендации по подготовке к экзамену и зачету. Успешная подготовка к экзамену и зачету включает, с одной стороны, добросовестную работу в течение семестра, выполнение всех заданий преподавателя, а с другой – правильная организация процесса непосредственной подготовки. При подготовке к экзамену необходимо освоить теорию: разобрать определения всех понятий, повторить синтаксис и семантику изучаемых языков программирования, типовые алгоритмы решения задач. Затем рассмотреть примеры и самостоятельно реализовать задания из каждой темы. При этом, если задания формулируются студентом самостоятельно, – достигается более продвинутый уровень изучения дисциплины.

VII. МАТЕРИАЛЬНО-ТЕХНИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

Аудиторные занятия по дисциплине включают лекции, практические занятия и лабораторные работы.

Для проведения лекционных и практических занятий необходима аудитория со следующим оборудованием:

Акустическая система для потолочного монтажа с низким профилем, Extron SI 3CT LP (пара)

Врезной интерфейс с системой автоматического втягивания кабелей TLS TAM 201 Standart III

Документ-камера Avervision CP355AF

Комплект удлинителей DVI по витой паре (передатчик/приёмник), Extron DVI 201 Tx/Rx

Матричный коммутатор DVI 4x4. Extron DXP 44 DVI PRO

Микрофонная петличная радиосистема УВЧ диапазона Sennheiser EW 122 G3 в составе рэкового приёмника EM 100 G3, передатчика SK 100 G3, петличного микрофон ME 4 с ветрозащитой и антенн (2 шт.)

Мультимедийный проектор, Mitsubishi EW330U, 3000 ANSI Lumen, 1280x800

Расширение для контроллера управления Extron IPL T CR48

Сетевая видеокамера Multipix MP-HD718

Сетевой контроллер управления Extron IPL T S4

Усилитель мощности, Extron XPA 2001-100V

Цифровой аудиопроцессор, Extron DMP 44 LC

Шкаф настенный 19" 7U, Abacom VSP-W960SG60

Экран проекционный ScreenLine Trim White Ice, 50 см черная кайма сверху, размер рабочей области 236x147 см

Лабораторные работы проводятся в компьютерном классе, в котором должно быть установлено:

– Моноблок Lenovo C306G-i34164G500UDK (20 шт),

– Акустическая система для потолочного монтажа с низким профилем, Extron SI 3CT LP (пара)

– Акустическая система для потолочного монтажа с низким профилем, Extron SI 3CT LP (пара)

– Акустическая система для потолочного монтажа с низким профилем, Extron SI 3CT LP (пара)

– Врезной интерфейс с системой автоматического втягивания кабелей TLS TAM 201 Standart III

– Документ-камера Avervision CP355AF

– ЖК-панель 47", Full HD, LG M4716CCBA

– Комплект удлинителей DVI по витой паре (передатчик/приёмник), Extron DVI 201 Tx/Rx

– Матричный коммутатор DVI 4x4. Extron DXP 44 DVI PRO

– Микрофонная петличная радиосистема УВЧ диапазона Sennheiser EW 122 G3 в составе рэкового приёмника EM 100 G3, передатчика SK 100 G3, петличного микрофон ME 4 с ветрозащитой и антенн (2 шт.)

– Мультимедийный проектор, Mitsubishi EW330U, 3000 ANSI Lumen, 1280x800

– Расширение для контроллера управления Extron IPL T CR48

– Сетевая видеокамера Multipix MP-HD718

– Сетевой контроллер управления Extron IPL T S4

– Стойка металлическая для ЖК-дисплея У SMS Flatscreen FH T1450

– Усилитель мощности, Extron XPA 2001-100V

– Цифровой аудиопроцессор, Extron DMP 44 LC

– Шкаф настенный 19" 7U, Abacom VSP-W960SG60

– Экран проекционный ScreenLine Trim White Ice, 50 см черная кайма сверху, размер рабочей области 236x147 см

Приложение 1

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Дальневосточный федеральный университет»
(ДВФУ)

ИНЖЕНЕРНАЯ ШКОЛА

УЧЕБНО-МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ САМОСТОЯТЕЛЬНОЙ РАБОТЫ ОБУЧАЮЩИХСЯ

по дисциплине «Программирование в инженерных задачах»

Направление подготовки: 15.03.03 Прикладная механика

Профиль подготовки: «Математическое и компьютерное моделирование механических систем и процессов»

Форма подготовки очная

Владивосток
2016

План-график выполнения самостоятельной работы по дисциплине

№ п/п	Дата/сроки выполнения	Вид самостоятельной работы	Примерные нормы времени на выполнение	Форма контроля
1	1-4 недели семестра	Реализация проекта «Геометрические вычисления на плоскости»	4 час.	ПР-12
2	5-9 недели семестра	Подготовка к тесту по разделу «Основы технологии программирования»	3 час.	ПР-1
3	10-14 недели семестра	Реализация проекта «Применение массивов к решению задач профессиональной деятельности»	4 час.	ПР-12
4	16-17 недели семестра	Подготовка к опросу по разделу «Событийное программирование»	4 час	УО-1
5	18 неделя семестра	Подготовка к зачету	3 час	Зачет
6	1-4 недели семестра	Реализация проекта «Решение задач на создание иерархии классов»	2 час.	ПР-12
7	5-10 недели семестра	Реализация проекта «Использование классов общего назначения»	2 час.	ПР-12
8	11-12 недели семестра	Подготовка к устному опросу по разделу «Технология объектно-ориентированного программирования»	3 час.	УО-1
9	14-17 недели семестра	Реализация проекта «Визуализация и графика»	2 час.	ПР-12
10	Экзаменационная сессия	Подготовка к экзамену	45 часов	Экзамен
		Итого	**72 час.**	

Характеристика заданий для самостоятельной работы обучающихся и методические рекомендации по их выполнению

Расчетно-графические задания

В расчетно-графической работе требуется создать проект в среде Lazarus на языке Object Pascal, предназначенный для организации работы с пользователем и решения некоторой задачи. Перед сдачей преподавателю задания, его необходимо проверить с помощью специального инструмента, реализованного в виде web-приложения и размещенного в курсе

«Программирование в инженерных задачах» LMS BlackBoard. Далее приводится образцы решения каждого расчетно-графического задания.

Расчетно-графическая работа по разделу «Основы технологии программирования»

Реализовать проект в среде Lazarus для решения следующей задачи:

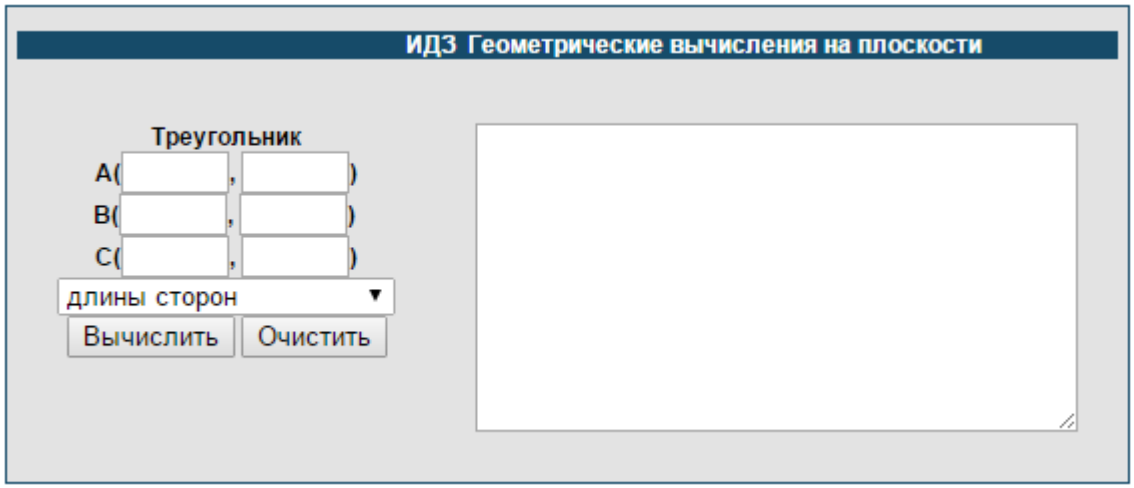
Даны координаты трех вершин некоторого треугольника. Проверить, можно ли по этим вершинам построить треугольник, а затем вычислить:

- длины сторон;
- радиус описанной вокруг треугольник окружности;
- радиус вписанной окружности;
- величины всех его углов;
- длины всех его медиан;
- определить является этот треугольник остроугольным, тупоугольным или прямоугольным.

Последовательность выполнения расчетно-графической работы:

1. Создать проект в среде Lazarus

2. Создать форму следующего вида, используя компоненты Memo, Edit, Combobox, Label,Button:



3. Реализовать программу на языке Object Pascal и разместить ее в процедуру обработки события кнопки «Вычислить»:

```
procedure TForm1.Button_RunClick(Sender: TObject);
type
  tTria = record //вершина
```

```
    x: integer;
    y: integer ;
  end;
var
  Form1: TForm1;
  tria: array[1..3] of tTria; //треугольники
  a: array[1..3] of real;   // углы
  p: array[1..3] of real;   //длины сторон
  m: array[1..3] of real;   //длины медиан
  R1, p2, r2, max, min1, min2: real;
  i, code: Integer;
begin
  val(edit1.Text, tria[1].x, code);
  if code <> 0 then
  begin
    Showmessage('ошибка ' + edit1.Text);
    edit1.SetFocus;
    exit;
  end;
  val(edit2.Text, tria[1].y, code);
  if code <> 0 then
  begin
    Showmessage('ошибка ' + edit2.Text);
    edit2.SetFocus;
    exit;
  end;
  val(edit3.Text, tria[2].x, code);
  if code <> 0 then
  begin
    Showmessage('ошибка ' + edit3.Text);
    edit3.SetFocus;
    exit;
  end;
  val(edit4.Text, tria[2].y, code);
  if code <> 0 then
  begin
    Showmessage('ошибка ' + edit4.Text);
    edit4.SetFocus;
    exit;
  end;
  val(edit5.Text, tria[3].x, code);
  if code <> 0 then
  begin
    Showmessage('ошибка ' + edit5.Text);
    edit5.SetFocus;
    exit;
  end;
  val(edit6.Text, tria[3].y, code);
  if code <> 0 then
  begin
    Showmessage('ошибка ' + edit6.Text);
    edit6.SetFocus;
    exit;
  end;
```

```
  p[1] := sqrt(sqr(tria[1].x - tria[2].x) + sqr(tria[1].y -
sqr(tria[2].y)));
  p[2] := sqrt(sqr(tria[1].x - tria[3].x) + sqr(tria[1].y -
sqr(tria[3].y)));
  p[3] := sqrt(sqr(tria[3].x - tria[2].x) + sqr(tria[3].y -
sqr(tria[2].y)));
  if ((p[2] + p[3] - p[1]) > 0) and ((p[1] + p[3] - p[2]) > 0)
and ((p[1] + p[2] - p[3]) > 0) then
  begin
    for i := 1 to 3 do
      Memo1.Lines.Add(floattostr(tria[i].x) + '      ' +
floattostr(tria[i].y));
    case Combo.ItemIndex of
      0:
        begin

          p[1] := sqrt(sqr(tria[1].x - tria[2].x) +
sqr(tria[1].y - sqr(tria[2].y)));
          p[2] := sqrt(sqr(tria[1].x - tria[3].x) +
sqr(tria[1].y - sqr(tria[3].y)));
          p[3] := sqrt(sqr(tria[3].x - tria[2].x) +
sqr(tria[3].y - sqr(tria[2].y)));

          {углы}
          // по теореме косинусов
          a[1] := arccos((sqr(p[2]) + sqr(p[1]) - sqr(p[3])) /
(2 * p[2] * p[1])) * 180 / pi;
          a[2] := arccos((sqr(p[1]) + sqr(p[3]) - sqr(p[2])) /
(2 * p[1] * p[3])) * 180 / pi;
          a[3] := 180 - a[2] - a[1];

          Memo1.Lines.Add(' угол = ' + floattostr( (a[1])));
          Memo1.Lines.Add(' угол = ' + floattostr( (a[2])));
          Memo1.Lines.Add(' угол = ' + floattostr( (a[3])));
        end;
      1:
        begin
          p2 := (p[1] + p[2] + p[3]) / 2;
          R1 := (p[1] * p[2] * p[3]) / (4 * sqrt(p2 * (p2 -
p[1]) * (p2 - p[2]) * (p2 - p[3])));
          Memo1.Lines.Add(' радиус описанной вокруг треугольник
окружности = ' + floattostr( R1));
        end;
      2:
        begin
          p2 := (p[1] + p[2] + p[3]) / 2;
          r2 := sqrt(((p2 - p[1]) * (p2 - p[2]) * (p2 - p[3])) /
p2);
          Memo1.Lines.Add(' радиус вписанной в треугольник
окружности = ' + floattostr( r2));
        end;
      3:
        begin
```

```
          m[1] := sqrt((2 * p[1] * p[1] + 2 * p[2] * p[2] - p[3]
* p[3]) / 4);
          m[2] := sqrt((2 * p[3] * p[3] + 2 * p[2] * p[2] - p[1]
* p[1]) / 4);
          m[3] := sqrt((2 * p[1] * p[1] + 2 * p[3] * p[3] - p[2]
* p[2]) / 4);
          Memo1.Lines.Add(' MC = ' + floattostr( m[1]));
          Memo1.Lines.Add(' MA = ' + floattostr( m[2]));
          Memo1.Lines.Add(' MB= ' + floattostr( m[3]));
        end;
      4:
        begin
          if (p[1] > p[2]) and (p[1] > p[3]) then
          begin
            max := p[1]; min1 := p[2]; min2 := p[3];
          end
          else
          if (p[2] > p[1]) and (p[2] > p[3]) then
          begin
            max := p[2]; min1 := p[1]; min2 := p[3];
          end
          else
          begin
            max := p[3]; min2 := p[1]; min2 := p[2];
          end;
          if (sqr(max) = sqr(min1) + sqr(min2)) then
            Memo1.Lines.Add('Треугольник прямоугольный')
          else begin
            if (sqr(max) < sqr(min1) + sqr(min2)) then
              Memo1.Lines.Add('Треугольник остроугольный')
            else  Memo1.Lines.Add('Треугольник тупоугольный');
          end;
        end;
    end;
  end else Showmessage('Треугольник не существует');
end;
```

Расчетно-графическая работа по разделу «Технология событийного программирования»

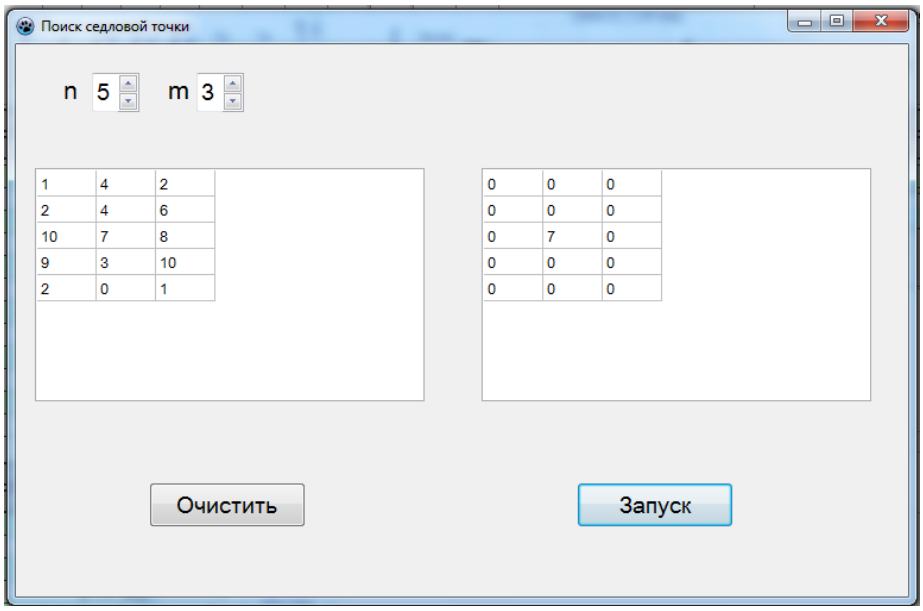
Реализовать проект в среде Lazarus для решения следующей задачи:

Дана матрица A(n..m). Определить все ее «седловые» точки. «Седловой» точкой называется элемент, наименьший в своей строке и одновременно наибольший в своем столбце или наоборот. Для заданной матрицы A(n..m) построить матрицу B(n..m), в которой на местах “седловых” точек выведены их значения, а остальные элементы матрицы – 0.

Последовательность выполнения расчетно-графической работы:

1. Создать проект в среде Lazarus

2. Создать форму следующего вида, используя компоненты Memo, Edit, Combobox, Label,Button, StringGrid



3. Реализовать программу на языке Object Pascal и разместить ее в процедуру обработки события кнопки «Запуск»:

```
procedure TForm1.StartClick(Sender: TObject);
var l,k,min,max:integer;
begin
   for i:=1 to spin1.value do
   for j:=1 to spin2.value do begin
   a[i,j]:=strtoint(input.Cells[j-1,i-1]);
   b[i,j]:=0;
   end;
     for i:=1 to spin1.value do begin
       min:=a[i,1];
       l:=1;
   for j:=2 to spin2.value do
     if a[i,j]<min then begin
       min:=a[i,j];
       l:=j;
     end;
     max:=min;
     for k:=1 to spin1.value do
      if a[k,l]>max then max:=a[k,l];
    if max=min then
       for j:=1 to spin2.value do
         if a[i,j]=min then b[i,j]:=min;
     end;
     for i:= 1 to spin1.value do
       for j:= 1 to spin2.value do
          output.cells[j-1,i-1]:=inttostr(b[i,j]);
end;
```

4. Реализовать процедуры обработки событий: изменение размеров массива; ввод элементов массива; очистка матриц.

Расчетно-графическая работа по разделу «Технология объектно-ориентированного программирования»

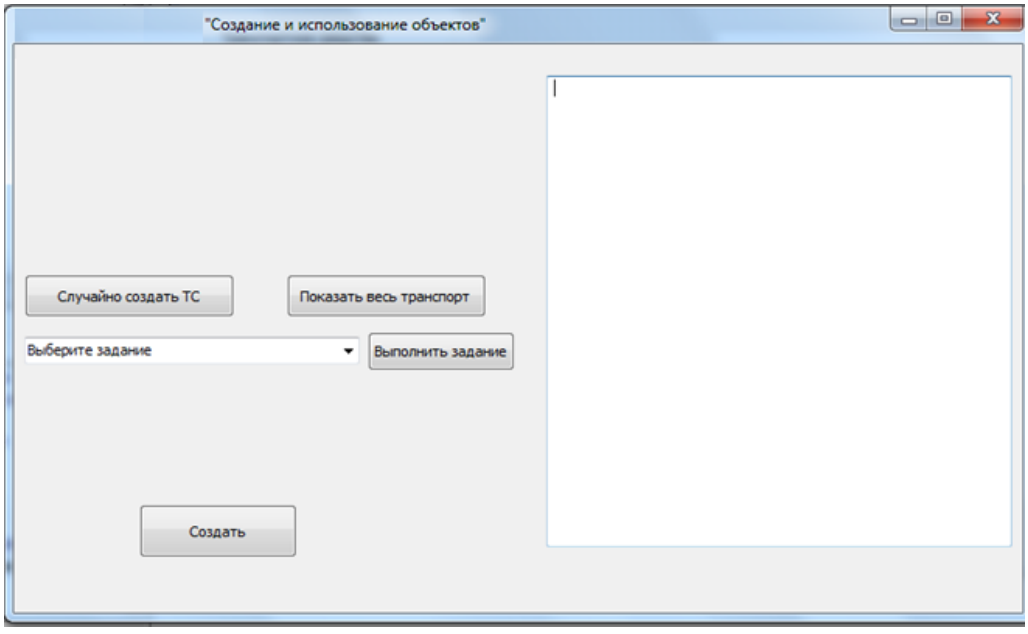
Реализовать проект в среде Lazarus для решения следующей задачи:

Разработать иерархию классов для транспортных средств. Базовый класс ***TCar*** определяет марку и среднюю скорость машины. Класс-потомок ***TBus*** представляет автобусы и определяет максимальное число пассажиров в них. Каждый класс содержит конструктор с параметрами и функцию ***show***( ), которая выводит на экран все характеристики транспортного средства соответствующего типа. Классы включает методы для получения скорости и количества посадочных мест транспортного средства.

Случайным образом создать N транспортных средств разного вида (автобус или легковая машина). Реализовать возможность просмотра характеристик созданных транспортных средств. А затем найти и вывести самое быстрое транспортное средство, самую быструю легковую машину, также отсортировать список транспортных средств по убыванию скорости.

1. Создать проект в среде Lazarus

2. Создать форму следующего вида, используя компоненты Memo, Edit, Combobox, Label,Button:



3. Разработать структуру классов для решения поставленной задачи:

- В разделе **interface type** создать класс для описания Машины и класс-потомок для описания Автобуса:

```
TCar = class
    Private
      speed: Integer; // скорость машины
      mark : string; // ее марка
    public
      constructor Create(vSpeed: Integer; vMark: string);
      function GetSpeed(): Integer; // метод для получения
скорости
     // метод для получения количества мест в машине
      function GetCountPlace(): integer; virtual;
     // показать информацию о машине
      procedure ShowCar(Write:TMemo); virtual;
  end;

  TBus = class(TCar)
    private
      countPlace : integer; // количество мест
    public
      constructor Create(vSpeed: Integer; vMark: string;
                         vCount : integer);
      function GetCountPlace(): integer; override;
      procedure ShowCar(Write:TMemo); override;
  end;
```

- В разделе **implementation** описать реализацию конструкторов и методов классов:

```
constructor TCar.Create(vSpeed: Integer; vMark: string);
begin
  speed := vSpeed;
  mark := vMark;
end;

function TCar.GetSpeed(): Integer;
begin
  Result := speed;
end;

procedure TCar.ShowCar(Write:TMemo);
begin
  Write.Lines.Add('автомобиль марка'+mark+
                  ' скорость'+intToStr(speed));
end;

function TCar.GetCountPlace(): integer;
begin
  result := 5;
```

```
end;

constructor TBus.Create(vSpeed: Integer; vMark: string;
                        vCount : integer);
begin
   inherited Create(vSpeed,vMark);
   countPlace := vCount;
end;

procedure TBus.ShowCar(Write:TMemo);
begin
  Write.Lines.Add('автобус марка '+mark+' скорость='
  +intToStr(speed)+' количество мест'+intToStr(countPlace));
end;

function TBus.GetCountPlace(): integer;
begin
  result := countPlace;
end;
```

- С событием onClick кнопки ButtonCreate связать следующую процедуру:

```
procedure TForm1.ButtonCreateClick(Sender: TObject);
var curCar: TCar;
    ind, k, n, i : integer;
    s : string;
    {массив для описания марок машин}
    arrMark : array [0..5] of string;
 begin
  arrMark[0] := 'Audi';
  arrMark[1] := 'BMW';
  arrMark[2] := 'Citroen';
  arrMark[3] := 'Dodge';
  arrMark[4] := 'Ford';
  arrMark[5] := 'Honda';
  {создание N=10 машин}
  for i := 1 to 10 do
  begin
    ind := random(2);// для выбора типа ТС
    k := random(6); // для определения марки
    s := arrMark[k]; // получение марки машины
    k := 40 + random(70);// случайная скорость, выше 40 км/ч
    n :=  8 + random(50); // случайное количество мест, выше 8
    // создаем либо легковую машину, либо автобус
    case ind of
      0: curCar := TCar.Create(k,s);
      1: curCar := TBus.Create(k,s,n);
    end;
    {сохраняем текущую фигуру в массиве}
    setLength(allCar,i); // задаем новую длину массива
    allCar[i-1] := curCar; // заносим созданную машину в массив
   end;
    memoWrite.Clear;
    memoWrite.Lines.Add('Общее количество ТС:
'+intToStr(length(allCar)));
```

```
end;
```

- С событием onClick кнопки ButtonShow связать следующую процедуру:

```
procedure TForm1.ButtonShowClick(Sender: TObject);
var i : integer;
begin
  {выводим массив машин}
  for i := 0 to High(allCar) do
    allCar[i].ShowCar(memoWrite);
end;
```

- С событием onClick кнопки ButtonTask связать следующую процедуру:

```
procedure TForm1.ButtonTaskClick(Sender: TObject);
var
  max, i, j : integer;
  findCar : TCar;
  find : Boolean;
begin
  memoWrite.Lines.Add('');
  case ComboTask.ItemIndex of
    0:begin //ТС с максимальной скоростью
        findCar := allCar[0];
        for i := 1 to high(allCar) do
          if findCar.GetSpeed() < allCar[i].GetSpeed()
          then  findCar := allCar[i];
        memoWrite.Lines.Add('Самое быстрое транспортное
средство');
        findCar.ShowCar(memoWrite);
      end;

    1:begin //самая быстрая легковая машина
        find := false; // предполагаем, что легковых машин нет
        for i := 0 to high(allCar) do
          if allCar[i].ClassNameIs('TCar') then
            if not find then // пока не было легковых машин
            begin
              findCar := allCar[i]; //запоминаем первую машину
              find := true; //легковые машины точно есть
            end
            else
           {можно находить максимум, так как хотя бы
           одна легковая машина уже была}
              if (findCar.GetSpeed() < allCar[i].GetSpeed()) and
find
              then  findCar := allCar[i];
        if find then // проверка, были ли легковые машины
        begin
          memoWrite.Lines.Add('Самая быстрая легковая машина');
          findCar.ShowCar(memoWrite);
        end
        else memoWrite.Lines.Add('Легковых машин нет');
      end;
```

```
      2: begin // сортировка пузырьком списка машин по скорости
        for i := 0 to High(allCar) -1  do
          for j := 0 to High(allCar)- i - 1 do
             if allCar[j].GetSpeed() < allCar[j+1].GetSpeed()
then
             begin
               findCar := allCar[j];
               allCar[j] := allCar[j+1];
               allCar[j+1] := findCar;
             end;
          {выводим отсортированный массив машин}
        memoWrite.Lines.Add('Отсортированный по скорости список
ТС');
        for i := 0 to High(allCar) do
           allCar[i].ShowCar(memoWrite);
      end;
    else showMessage('Выберите задание');
  end;
end;
```

Устные опросы

Устные опросы проводятся преподавателем по завершению изучения каждого раздела. Вопросы и задания приведены в приложении 2. Для подготовки используется основная и дополнительная литература по дисциплине «Программирование в инженерных задачах», а также информация, размещенная в LMS BlackBoard.

Вопросы, возникающие в процессе подготовки, студент может задать преподавателю любо на консультациях, либо через специальное средство LMS BlackBoard.

Тестирование

Тестирование осуществляется на занятии через систему BlackBoard. Для подготовки тестов используются пробные тесты, размещенные в системе BlackBoard. Эти тесты не оцениваются преподавателем и служат элементом самоконтроля. Оба типа теста формируются на основе одной базы вопросов.

Требования к представлению и оформлению результатов самостоятельной работы

Результатом самостоятельной работы студентов являются:

1. Проекты для решения заданий расчетно-графической работы, реализованные в системе Lazarus и оформленные в соответствии с правилами оформления программ. Созданные проекты обязательно отсылаются преподавателю через систему BlackBoard. Защита расчетно-графического задания осуществляется только после того, как проект проверен и оценен преподавателем в системе BlackBoard.

2. Тесты самоконтроля, выполненные в системе BlackBoard. Все попытки, выполненные пользователем, сохраняются. К основному тесту студент допускается только после прохождения на зачетном уровне данного вида тестов.

Правила оформления программ

I. Блоки

1.1. Каждый оператор языка программирования записывается на отдельной строке.

1.2. Содержимое конструкций begin-end, repeat-until, case-end, record-end выделяется отступом в виде некоторого количества дополнительных пробелов в начале строки или табуляции.

1.3. Содержимое конструкций if-then, if-then-else, while-do, with-do допускается писать в одну строку, если суммарная длина строки остается приемлемой. В случае, если содержимое перенести необходимо, то его можно вынести в отдельную строку, дополнив отступом.

1.4. Конструкцию if-then-else можно писать в две части, выделяя действие в случае невыполнения условия (else) на отдельную строку, но тогда else должен выделяться дополнительным отступом. Также по предыдущему пункту можно писать всю конструкцию в одну строку или в четыре, но никак иначе.

1.5. Если в любую конструкцию (кроме if-then-else) вложен begin-end, то begin-end можно писать на том же уровне вложенности, что и конструкцию, в которую он вложен. Более того, допускается не переносить begin на отдельную строку.

1.6. Содержимое секций uses, type, const, var должно выделяться отступом. Сами названия секций при этом должны идти без отступа в отдельной строке.

1.7. Выбранный вид отступа должен быть единым для всей программы: не допускается менять ширину отступа в пробелах в середине программы, а также менять табуляцию на пробелы и наоборот.

II. Пробелы внутри строк

2.1. Любая бинарная операция (<, >, <=, >=, <>, =, +, -, *, div, mod, /, and, or, xor, и т.д.) должна выделяться пробелом как слева, так и справа от себя. В том числе это относится и к знакуприсваивания :=.

2.2. Однострочный комментарий может идти в той же строке, что и код, но при этом он обязан

отделяться от кода двумя пробелами. Во всех иных случаях комментарий не может идти в той же строке, что и код.

2.3. Перед символами «:», «,» и «;» никогда не должен идти пробел, и всегда должен идти после (если это не конец строки).

2.4. Запрещается использовать в строке более одного пробела подряд, если эти пробелы не составляют отступ и не служат для отделения комментария от кода.

2.5. Ключевые слова array, case, do, downto, else, file, if, in, of, then, to, while, with, until всегда должны содержать по обе стороны от себя хотя бы один пробельный символ (допускается перевод строки).

2.6. Запрещено отделять знак «..» (две точки) от соседних лексем слева и справа при объявлении дипапазонов.

2.7. При обращении к элементу массива пробелы между квадратными скобками и их содержимым не ставятся. Также не ставится пробел между именем переменной и открывающейся квадратной скобкой.

III. Идентификаторы

3.1. В программе запрещается использование кириллицы (даже в комментариях).

3.2. В программе запрещается использование транслита.

3.3. В качестве имен простых переменных/констант, смысл которых достаточно очевиден, разрешается использовать короткие название из одной-двух букв. При этом, по сложившейся традиции, имена счетчиков в циклах обычно называют i и j. Для нетривиальных понятий лучше использовать английские слова.

3.4. Не стоит злоупотреблять длинными именами переменных и констант. 10-15 символов должно быть достаточно для передачи смысла переменной.

3.5. Имена переменных и констант должны содержать только строчные латинские буквы, цифры и нижние подчеркивания для разделения слов в названии.

3.6. Имена функций и процедур именуются с использованием CamelCase, то есть содержат только латинские буквы и цифры, каждое слово, кроме первого, пишется с заглавной буквы.

3.7. Все ключевые слова языка должны записываться только строчными латинскими буквами.

4.2. Каждая процедура и функция должна содержать перед и после себя ровно одну пустую строку.

4.3. В случае, если в функцию или процедуру передается переменная сложного типа (массив, множество и т.п.), она обязательно должна выделяться одним из модификаторов const или var.

4.4. Между списком аргументов и именем функции/процедуры никогда не ставится пробел.

V. Другие правила

5.1. Запрещается использование конструкции goto.

5.2. В программе запрещается использование нетривиальных констант без их явного описания в разделе const. Обычно нетривиальными считаются любые константы, кроме, быть может, -1, 0, 1, 2, 10.

5.3. Каждая переменная перед использованием обязана быть явно инициализирована присваиванием.

5.4. Запрещается сравнивать переменные типа boolean с true или false.

5.5. Внутри цикла for запрещается явное изменение значения счетчика цикла.

5.6. Запрещено использовать строки длиннее 120 символов. Строки можно разбивать на части в длинных выражениях и списках аргументов, при этом если выражение разбивается на бинарной операции, то саму операцию лучше переносить на отдельную строку. Строки 2, 3, 4, ... одного оператора считается вложенными в первую строку и выделяются отступом.

Критерии оценки выполнения самостоятельной работы

Самостоятельная работа студентов включает расчетно-графические задания, подготовку к устным опросам, тестирование. Критерии оценки каждого вида работы приведены в приложении 2.

Приложение 2

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Дальневосточный федеральный университет»
(ДВФУ)

НАЗВАНИЕ ШКОЛЫ (ФИЛИАЛА)

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ

по дисциплине «Программирование в инженерных задачах»
Направление подготовки: 15.03.03 Прикладная механика
Профиль подготовки: «Математическое и компьютерное моделирование механических систем и процессов»
Форма подготовки очная

Владивосток
2016

Паспорт ФОС

Код и формулировка компетенции	Этапы формирования компетенции	
ОПК-9 – владением методами информационных технологий, соблюдением основных требований информационной безопасности, в том числе защиты государственной тайны	Знает	особенности современных методологий и технологий использования и создания программных средств
	Умеет	проектировать, конструировать и отлаживать программные продукты в соответствии с заданными критериями качества и стандартами
	Владеет	навыками работы с компьютером как средством управления информацией;
ОПК-10 – способностью решать стандартные задачи профессиональной деятельности на основе информационной и библиографической культуры с применением информационно-коммуникационных технологий и с учетом основных требований информационной безопасности	Знает	основы событийного и объектно-ориентированного программирования
	Умеет	разрабатывать алгоритмы и программы для решения задач профессиональной деятельности средствами высокоуровневых языков программирования и визуальных систем программирования.
	Владеет	способностью использовать для решения аналитических, исследовательских и коммуникативных задач современные средства визуальной разработки программ.
ПК-7– умением извлекать актуальную научно-техническую информацию и наукометрическую информацию из электронных ресурсов, в том числе Science Direct, Elsiver Freedom Collection, SCOPUS	Знает	тенденции развития средств вычислительной техники и особенности их применение в профессиональной деятельности
	Умеет	использовать поисковые системы сети Интернет для отбора информации из наукометрических баз данных
	Владеет	- навыками работы с компьютером как средством управления информацией; - методологией сбора, обработки анализа и обобщения научно-технической информации

№ п/п	Контролируемые разделы / темы дисциплины	Коды и этапы формирования компетенций		Оценочные средства	
				текущий контроль	промежуточная аттестация
1	Основы технологии программирования	ОПК-9	Знает особенности современных методологий и технологий использования и создания программных средств	Тест (ПР-1)	Вопросы 1-15
			Умеет проектировать, конструировать и отлаживать программные продукты в соответствии с заданными критериями качества и стандартами	Лабораторная работа (ПР-6)	Задачи тип I
			Владеет навыками работы с компьютером как средством управления информацией;	Расчетно-графическая работа (ПР-12)	Задачи тип II
2	Технология событийного программирования	ПК-7	Знает тенденции развития средств вычислительной техники и особенности их применение в профессиональной деятельности	Собеседование (УО-1)	Вопросы 1-7
			Умеет использовать поисковые системы сети Интернет для отбора информации из наукометрических баз данных	Лабораторная работа (ПР-6)	Задачи тип I
			владеет навыками работы с компьютером как средством управления информацией; - методологией сбора, обработки анализа и обобщения научно-технической информации	Расчетно-графическая работа (ПР-12)	Задачи тип II
3	Технология объектно-ориентированного программирования	ОПК-10	Знает основы событийного и объектно-ориентированного программирования	Собеседование (УО-1)	Вопросы 8-33-
			Умеет разрабатывать алгоритмы и программы для решения задач профессиональной деятельности средствами высокоуровневых языков	Лабораторная работа (ПР-6) Расчетно-графическая работа (ПР-12)	Задачи тип I

№ п/п	Контролируемые разделы / темы дисциплины	Коды и этапы формирования компетенций		Оценочные средства	
				текущий контроль	промежуточная аттестация
			программирования и визуальных систем программирования.		
			Владеет способностью использовать для решения аналитических, исследовательских и коммуникативных задач современные средства визуальной разработки программ.	Расчетно-графическая работа (ПР-12)	Задачи тип II

Зачетно-экзаменационные материалы

Перечень типовых вопросов к зачету

1. Конструкции структурного программирования.
2. Технологии разработки программ.
3. Процедурное программирование.
4. Метод пошаговой детализации.
5. Технологии разработки программ.
6. Процедуры и функции.
7. Модули. Структура модулей.
8. Модули. Типы модулей.
9. Реализация модуля для работы с комплексными числами.
10. Понятие о визуальном программировании.
11. Понятие формы. Структура формы.
12. Компоненты для организации взаимодействия с пользователем.
13. Свойства компонент.
14. Компоненты edit, label, memo,button.
15. Ввод, вывод данных разных типов через компоненты формы.

Вопросы к экзамену

1. Событие. Определение и назначение.
2. Типы событий. Обработка событий.
3. Прерывание и событие. Способы реагирования на события.
4. События мыши.
5. События клавиатуры.
6. События компонент.
7. Реализация интерактивного интерфейса.
8. Понятие класса в объектно-ориентированном программировании.
9. Инкапсуляция классов.
10. Наследование классов.
11. Полиморфизм классов.
12. Составляющие класса: поля.
13. Составляющие класса: свойства.
14. Составляющие класса: методы.
15. Объявление классов. Разделы, допустимые при объявлении класса.
16. Конструкторы и деструкторы.
17. Создание объектов. Доступ к данным и вызов методов.
18. Разработка иерархии классов.
19. Директивы, управляющие перекрытием методов классов.
20. Видимость имен элементов. Управление доступом к членам класса.
21. Средства ограничения доступа к компонентам классов.
22. Средства возбуждения и обработки исключительных ситуаций.
23. Обработчики исключений.
24. Класс TList - списки. Основные свойства и методы класса.
25. Класс TStringsList. Свойства и методы класса.
26. Использование списков для решения задач.
27. Графический инструментарий. Класс TPen.
28. Графический инструментарий. Класс TBrush.
29. Графический инструментарий. Класс TFont.

30. Графический инструментарий. Класс КлассTCanvas.

31. Методы класса TCanvas.

32. Управление диалоговыми окнами выбора цвета, шрифта и пр.

33. Примеры задач объектно-ориентированного программирования.

Перечень типовых экзаменационных задач

На экзамен по каждой теме выносится два вида задач: для оценки «продвинутого уровня» студентов предлагается составить иерархию классов, реализовать методы и свойства этих объектов и/или описать алгоритм решения задачи (не обязательно реализовывать полную программу), для проверки «высокого уровня» - предлагается создать проект для реализации некоторой задачи (в среде Lazarus).

Типовые задачи на составление алгоритма (тип I)

Раздел «Основы технологии программирования»

Дано число, записанное в одной из 4-х систем счисления (двоичной, шестеричной, десятичной и шестнадцатеричной). Написать алгоритм перевода числа в десятичную систему счисления.

Раздел «Технология событийного программирования»

Через события описать последовательность выделения и изменения размера некоторой фигуры, размещенной на форме (использовать события указателя мыши).

Раздел «Технология объектно-ориентированного программирования»

Разработайте иерархию классов для банковских счетов. Общий класс *Счет* характеризуется количеством денег на нем. В этом классе должны быть реализованы методы «*Положить деньги», «Снять деньги»* и «*Показать баланс»* (количество денег) на счете. Классы «*Кредитный счет*» и «*Депозит*» сохраняют все свойства класса «*Счет*», дополняя и уточняя его поведение. «*Депозит*» имеет дополнительные поля Срок и процент, в нем переопределяется метод «*Показать баланс*» (текущее количество денег на

счете умножается на процент). С «Депозита» нельзя снять больше денег, чем там находится. «*Кредитный счет*» имеет поле Допустимый лимит, в нем переопределяется метод «*Снять деньги*», можно снимать до допустимого лимита, например, на «*Кредитном счете*» баланс – 0 рублей, Допустимый лимит 1000, значит можно снять до 1000 рублей, имея отрицательный баланс.

Типовые задачи на реализацию проекта в выбранной программной среде (тип II)

Раздел «Основы технологии программирования»

1. Разработать проект, который позволяет ввести целое число, проверить правильность ввода и для введенного числа:

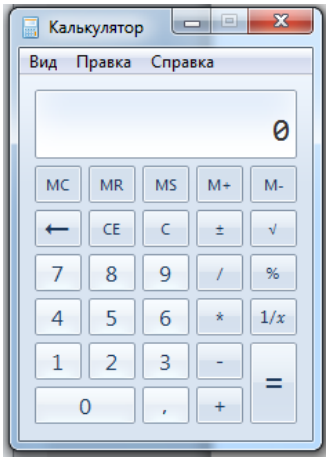
- найти число, которое получится выписыванием в обратном порядке цифр исходного числа;
- определить является ли число палиндромом (палиндром - число, которое читается одинаково справа и слева, например 3443);
- найти разные цифры, которые входят в запись исходного числа, и подсчитать их количество.

1. Используя поисковые системы сети интернет, перечислить сферы применения комплексных чисел. Реализовать модуль для работы с комплексными числами. В модуле описать методы основных операций с комплексными числами.

Раздел «Технология событийного программирования»

1. Даны координаты вершин многоугольника (правильного n-угольника, выпуклого многоугольника). Написать программу для вычисления основных характеристик многоугольника. Используя методы растровой графики, отобразить многоугольник на плоскости.

2. Разработать проект для реализации простого калькулятора следующего вида:



Раздел «Технология объектно-ориентированного программирования»

1. Разработать модель леса, в котором присутствуют классы растений и животных. Растения подразделяются на деревья и траву. Животные подразделяются на хищников и травоядных различного размера. Конструктор с параметрами должен создавать один объект леса. Реализовать у животного метод поиска в лесу пропитания с последующим поеданием (уничтожением). Необходимо учесть, что травоядные питаются только травой, а хищники могут съесть только животных меньше себя. Реализовать проект для демонстрации разработанных классов: создать случайным образом N единиц растений и животных, для животных найти то, что заданное животное сможет съесть.

2. Найти информацию о заданном виде движения материального тела, привести пример использования этого движения в реальной жизни. Реализовать программу для визуализации выбранного примера.

Принцип составления экзаменационного билета заданий к зачету

Первый вопрос являются теоретическим и предназначен для оценивания порогового уровня освоения дисциплины. Второй вопрос, на составление алгоритма, предназначен для оценки продвинутого уровня. Последний вопрос – создание проекта в среде Lazarus – для высокого уровня освоения. Таблица для составления экзаменационных билетов и заданий к экзамену по фонду оценочных средств:

Номер	II семестр

вопроса	
1	вопросы 1 – 35
2	задачи I типа
3	задачи II типа

Образец экзаменационного билета

1. Событие. Определение и назначение.

2. Дано число, записанное в одной из 4-х систем счисления (двоичной, шестеричной, десятичной и шестнадцатеричной). Написать алгоритм перевода числа в десятичную систему счисления

3. Разработать проект для реализации простого калькулятора

Критерии выставления оценки студенту на экзамене по дисциплине «Программирование в инженерных задачах»

Баллы (рейтинговой оценки)	**Оценка экзамена/зачета** (стандартная)	**Требования к сформированным компетенциям**
86-100	«отлично»/ «зачтено»	Оценка «отлично» выставляется студенту, если он глубоко и прочно усвоил материал по различным технологиям программирования, исчерпывающе, последовательно, четко и логически стройно его излагает, умеет составлять алгоритм и программу, используя различные парадигмы программирования, свободно справляется с задачами, вопросами и другими видами применения знаний, причем не затрудняется с ответом при видоизменении заданий, свободно использует компьютер для решения задач профессиональной деятельности, выбирает эффективный алгоритм для реализации программы, правильно обосновывает принятое решение, владеет разносторонними навыками и приемами выполнения практических задач, связанных с созданием приложений в области профессиональной деятельности с использованием различных сред программирования.
76-85	*«хорошо»*/ «зачтено»	Оценка «хорошо» выставляется студенту, если он твердо знает материал по основам событийного и объектно-ориентированного программирования, грамотно и по существу излагает его, не допуская существенных неточностей в ответе на вопрос, правильно применяет теоретические положения при решении практических вопросов и задач, связанных с созданием приложений в области профессиональной деятельности, владеет необходимыми навыками и приемами их реализации с использованием различных сред программирования.
61-75	*«удовлетворительно»*/	Оценка «удовлетворительно» выставляется студенту, если

	«зачтено»	он имеет знания только основного материала в области событийного и объектно-ориентированного программирования, но не усвоил его деталей, допускает неточности, недостаточно правильные формулировки, нарушения логической последовательности в изложении программного материала, испытывает затруднения при выполнении практических работ, связанным с созданием приложений в области своей профессиональной деятельности.
0-60	*«неудовлетворительно»/ «незачтено»*	Оценка «неудовлетворительно» выставляется студенту, который не знает значительной части программного материала по основам технологии программирования, допускает существенные ошибки, неуверенно, с большими затруднениями выполняет практические работы, связанные с созданием приложений. Оценка «неудовлетворительно» ставится студентам, которые не могут продолжить обучение без дополнительных занятий по соответствующей дисциплине

Оценочные средства для текущей аттестации (типовые ОС по текущей аттестации и критерии оценки по каждому виду аттестации по дисциплине «Программирование в инженерных задачах»)

Вопросы для устных собеседований

Раздел «Событийное программирование»

1. Что такое событие? Привести примеры.
2. Для чего используются процедуры обработки события?
3. Какие преимущества дает событийное программирование для разработки приложений пользователя?
4. События мыши. Переменная Shift.
5. События мыши. Переменная Button.
6. мыши. Переменная Sender.
7. События мыши. Обработка нажатия и отпускания клавиши мыши.
8. События мыши. Обработка движения указателя мыши.
9. События мыши. Обработка событий колеса прокрутки мыши.
10. События клавиатуры. Обработка ввода символа.
11. События клавиатуры. Обработка нажатия/отпускания клавиши.
12. События клавиатуры. Переменная Sender.
13. События клавиатуры. Переменная Shift.

14. События, связанные с компонентами визуальной среды.

Раздел «Объектно-ориентированное программирование»

1. Что такое класс?
2. В каких случаях необходимо использовать класс?
3. Как создать класс в коде?
4. Что такое экземпляр класса?
5. Как создать экземпляр класса?
6. Что такое метод класса?
7. Что должен содержать метод create ?
8. Как сделать переменные экземпляра доступными для чтения за пределами класса? Доступными для записи? Доступными для чтения и записи?
9. Может ли класс вызывать свои собственные методы?
10. В чем различие между классом и модулем?
11. Когда следует использовать класс, но не модуль?
12. Как работает наследование?
13. Почему полезно наследование?
14. Как можно расширить класс? В чем смысл этого?
15. Что такое область видимости?
16. В каких частях кода вы можете использовать переменные?
17. Когда объявляется область видимости?
18. Когда методы видимы?
19. Что такое приватный метод?
20. Что такое защищенный метод?
21. В чем различие приватных и защищенных методов?
22. Что значит термин «инкапсуляция»?
23. Что значит термин «наследование»?
24. Что значит термин «полиморфизм»?

Критерии оценки устного ответа на собеседовании:

✓ 100-85 баллов выставляется студенту, если его ответ показывает прочные знания основных положений изучаемого раздела технологии программирования, отличается глубиной и полнотой раскрытия темы; студент показывает свободное владение терминологическим аппаратом; умение объяснять синтаксис и семантику конструкций программирования и структур данных, делать выводы и обобщения, давать аргументированные ответы, приводить примеры структур классов; обучающийся свободно владеет монологической речью, логичностью и последовательностью ответа; умеет алгоритмически описывать проблему и данные из выбранной предметной области.

✓ 85-76 баллов выставляется студенту, если его ответ, обнаруживает прочные знания основных положений изучаемого раздела технологии программирования, отличается глубиной и полнотой раскрытия темы; обучающийся показывает владение терминологическим аппаратом; умение объяснять синтаксис и семантику конструкций программирования и структур данных, делать выводы и обобщения, давать аргументированные ответы, приводить структуры классов; студент демонстрирует свободное владение монологической речью, логичностью и последовательностью ответа; умение алгоритмически описывать проблему и данные из выбранной предметной области. Однако допускается одна - две неточности в ответе.

✓ 75-61 балл выставляется студенту, если его ответ, свидетельствует в основном о знании основных положений изучаемого раздела технологии программирования, отличается недостаточной глубиной и полнотой раскрытия темы; знанием основных вопросов теории; слабо сформированными навыками объяснения синтаксис и семантику основных конструкций программирования и структур данных, недостаточным умением давать аргументированные ответы и приводить примеры структуры классов; студент недостаточно владеет монологической речью, логичностью и последовательностью ответа. Допускается несколько ошибок в содержании

ответа; неумение алгоритмически описывать проблему и структуры данных из выбранной предметной области.

✓ 60-50 баллов выставляется студенту, если его ответ, обнаруживает незнание процессов основных положений изучаемого раздела технологии программирования, отличается неглубоким раскрытием темы; незнанием основных вопросов теории, несформированными навыками анализа синтаксиса и семантики основных конструкций программирования и структур данных; неумением давать аргументированные ответы, слабым владением монологической речью, отсутствием логичности и последовательности. Допускаются серьезные ошибки в содержании ответа; неумение алгоритмически описывать проблему и данные из выбранной предметной области

Тестовые задания по дисциплине «Программирование в инженерных задачах»

Каждому студенту формируется индивидуальный тест, в который входит 20 вопросов, выбранных случайным образом из списка заданий. В тесте реализованы следующее типы тестовых заданий:

- выбор одного варианта ответа;
- выбор нескольких вариантов ответа;
- ввод числового значения.

Типовых заданий теста по разделу «Основы технологии программирования»

1. Для чего используется раздел модуля INTERFACE?

a. В этом разделе перечисляются те модули, которые должны быть включены в проект и располагаются в других файлах;

b. В этом разделе содержится объявление всех глобальных объектов модуля;

c. В этом разделе содержится описание подпрограмм, а также локальные для модуля объекты;

d. В этом разделе размещаются операторы, которые исполняются до передачи управления основной программе;

e. В этом разделе указываются операторы, которые должны выполняться после выполнения основной программы.

2. Для ввода исходных данных служит компонент...

a. Edit

b. Button

c. Label

d. Image

e. Memo

3. Переменная s в результате выполнения команд

```
s:=0;
FOR k:=1 TO 4 DO
  s:=s+k;
```

получит значение…

Критерии оценки теста, состоящего из 20 вопросов

- ✓ 14-20 баллов – считается, что тест пройден.
- ✓ 0-13 баллов – тест не засчитывается

Комплекты заданий для выполнения расчетно-графических работ по дисциплине «Программирование в инженерных задачах»

Расчетно-графическая работа по разделу «Основы технологии программирования»

Расчетно-графическая работа предполагает реализацию приложения в среде программирования Lazarus для решения задачи, включающей несколько заданий. В приложении должен быть реализован ввод данных через компонент *edit*, выбор задания (компонент *combobox*), запуск программы (компонент *button*), вывод результатов (компонент *memo*).

Типовое задание

Дано целое число **n**:

– в точках **1,2,3,...,n** напечатать значения **f(n)** - количество положительных делителей числа n.

– напечатать все простые числа, не превосходящие **n**;

– все полные квадраты, не превосходящие **n**;

– в точках **1,2,3,...,n** напечатать значения функции Эйлера **f(n)** – количество целых чисел от **1** до **n-1**, взаимно простых с числом n.

Расчетно-графическая работа по разделу «Технология событийного программирования»

Расчетно-графическая работа предполагает реализацию приложения в среде программирования Lazarus для решения вычислительной задачи с матрицами и векторами. В приложении должен быть реализован изменение размера матриц (векторов) через компонент *spinedit*, ввод данных (компонент *stringgrid*), при необходимости выбор задания (компонент *combobox*), запуск программы (компонент *button*), вывод результатов (компонент *stringgrid*).

Типовое задание

Дана матрица **A(n..n).** Построить матрицу **B(n..n**), которая является матрицей, полученной поворотом исходной:

– на $\mathbf{90^o}$ против часовой стрелки;

– на $\mathbf{(90 \cdot n)^o}$ где **n** – произвольное целое положительное или отрицательное число (положительное число – поворот по часовой стрелки, отрицательное – против часовой стрелки).

Расчетно-графическая работа по разделу «Технология объектно-ориентированного программирования» «Решение задач на создание иерархии классов»

Расчетно-графическая работа предполагает реализацию приложения в среде программирования Lazarus для решения задачи, в которой описывается структура и обработка классов. В приложении должен быть реализован ввод

данных (компонент *stringgrid*), выбор задания (компонент *combobox*), запуск программы (компонент *button*), вывод результатов (компонент *memo*).

Типовое задание

Напишите программу, которая содержит базовый класс *TCelsius*. Конструктор с параметрами позволяет задавать соответствующую температуру. Также в этом классе определить методы для получения значений температуры в градусах Цельсия (*getC()*), Кельвина (*getK()*) и Фаренгейта (*getF()*) и метод *show( )* для вывода температуры и способа ее измерения (C). Создать два класса наследника *TFarenheit* и *TKelvin*. Переопределить функции *show*( ), которая выводит на экран температуру (в своем представлении *C, K* или *F*), а методы *getK(), getF()* и *getC*(). Реализовать возможность создать *N* объектов определенных классов, вывести в форму их характеристики. Посчитать количество сгенерированных температур каждого представления. Вывести все температуры в каждом из представлений: Цельсия, Кельвина и Фаренгейта. Найти минимальную и максимальную температуру.

Расчетно-графическая работа по разделу «Технология объектно-ориентированного программирования» «Визуализация и графика»

Расчетно-графическая работа предполагает реализацию приложения в среде программирования Lazarus для динамической визуализации объектов растровой графики. Приложение должно включать компонент *image* или любой другой, содержащий *canvas.* В приложении должен быть реализовано вывод изображения, управление его движением средствами компонент визуальной среды программирования.

Типовое задание

Создать приложение для визуализации движения графического объекта заданному закону. Настройку характеристик движения реализовать в интерактивном режиме.

Критерии оценки расчетно-графической работы

✓ 10-8 баллов выставляется студенту, если студент полностью выполнил расчётно-графическое задание. Фактических ошибок, связанных с пониманием проблемы, нет; семантических и синтаксических ошибок в программах нет. При защите студент отвечает на все вопросы преподавателя.

✓ 7-6 баллов – работа выполнена полностью; есть незначительные погрешности в реализации отдельных компонент приложения или в описании данных, или в организации интерфейса с пользователем. При защите студент отвечает на все вопросы преподавателя.

✓ 5-4 балла – работа выполнена полностью, есть погрешности в реализации отдельных компонент приложения или в описании данных, или в организации интерфейса с пользователем, связанные с непонимание постановки задачи. При защите студент не отвечает на 1-2 вопроса преподавателя.

✓ 1-3 балла – работа выполнена не полностью. Допущены ошибки в реализации отдельных компонент приложения или в описании данных, или в организации интерфейса с пользователем, связанные с непонимание постановки задачи или неумением пользоваться средой разработки. При защите студент не отвечает на более, чем на 2 вопроса преподавателя.

Лабораторные работы

Структура каждой лабораторной работы следующая:

1. Реализация приложения-образца в среде Lazarus.
2. Задания для самостоятельной работы.

Все лабораторные работы можно проверить через специальное средство, реализованное в системе BlackBoard. Выполненные лабораторные работы необходимо защитить, но предварительно они должны быть отправлены преподавателю на проверку через систему BlackBoard.

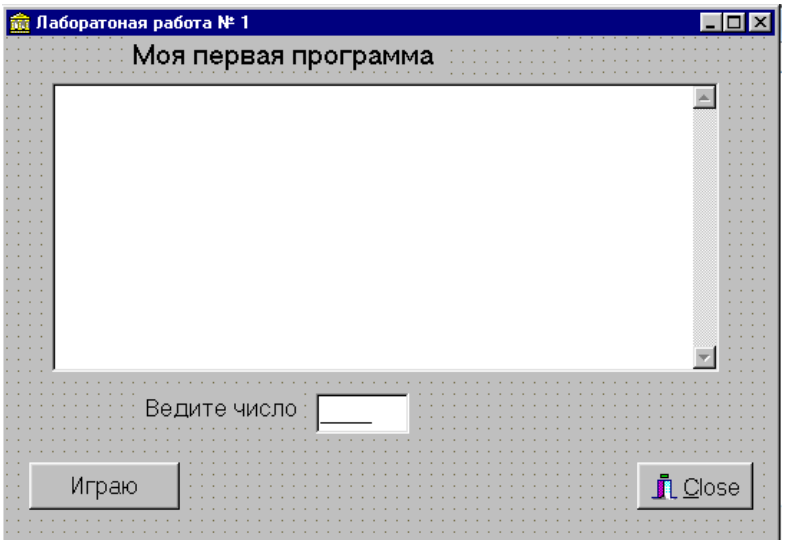
Типовая лабораторная работа

на тему «Создание приложения в среде Lazarus»

Задание: Создать приложение для реализации игры «Угадай число». Программа случайным образом выбирает целое число в диапазоне 0..1000 и

запоминает его. Сравнив ввод с запомненным числом, программа сообщает результат. Ввод продолжается до угадывания. После чего программа предлагает сыграть еще раз

1. Создать форму следующего вида:



В нее включить объекты, приведенные в таблице 1. Все эти компоненты расположены на страницах *Standard* и *Additional* палитры компонентов.

Таблица 1

Класс объекта	Свойства объекта	События, связанные с объектом
TForm	*Name*: **Form1** *Caption*:Лабораторная работа № 1	*OnActive*: 1. Сгенерировать случайное число. 2.Установить курсор в поле ввода.
TLabel (информационная надпись)	*Name*: **Label_Inf** *Caption*: Моя первая программа *Font*: размер шрифта 18	
TBitBtn (для завершения работы)	***Name*:** Bit_close *Kind*: bkClose	
TLabel (надпись «Введите число»)	*Name*: **Label_read** *Caption:* Введите число *Font*: размер шрифта 12	
TMaskEdit (для задания числа)	***Name*:** Mask_num *Font*: размер шрифта 12 *EditMask*: 0999;1;	
TMemo (для вывода результатов)	***Name*:** Memo_write *Font*: размер шрифта 12 *ScrollBars*: ssVertical *Lines*: очистить поле	
TButton	***Name*:** Button_run	*OnClick*:

Класс объекта	Свойства объекта	События, связанные с объектом
(для запуска игры)	*Font*: размер шрифта 12 ***Caption*: Играю**	1. Проверить, введено ли какое-нибудь число в поле ввода. 2. Сравнить введенное число с загаданным. Выдать информационные сообщения. В зависимости от результата либо продолжить игру, либо завершить ее.

Наступление каждого события, указанного в таблице 1, вызывает соответствующие процедуры обработки этих событий:

- Событие *OnActive* при открытии формы:

```
procedure TForm1.FormActivate(Sender: TObject);
begin
  {генерируем случайное число}
  randomize; x:=random(1001);
  label_inf.Caption:='Угадай число от 0 до 1000';
  {устанавливаем курсор в поле ввода}
  mask_num.SetFocus;
  n:=0;
end;
```

- Событие *OnClick* при нажатии на кнопку **Button_run**:

```
procedure TForm1.Button_runClick(Sender: TObject);
var k: integer;
begin
 n:=n+1;
 {изменяем подпись информационной метки}
 Label_inf.Caption:=IntToStr(n)+'-ая попытка';
{в поле ввода числа нет}
if mask_num.text='    ' then {прерываем попытку} exit ;
{преобразуем введенное текстовое значение в числовое}
 k:=StrToInt(Trim(mask_num.text));

{анализируем введенное число}
  If x=k then
  begin {число угадано}
     Label_inf.Caption:=Label_inf.Caption+'  успешная';
     {добавляем строки в поле для вывода}
     memo_write.Lines.Add('  X= '+IntToStr(k));
     memo_write.Lines.Add(' Конец игры');
     {делаем невидимыми все объекты, связанные с вводом}
     mask_num.Visible:=false;
     Label_read.Visible:=false;
     button_run.Visible:=false;
  end
  else
    begin {число не угадано}
```

```
    if k>x then memo_write.Lines.Add('  X< '+IntToStr(k));
    if k<x then memo_write.Lines.Add('  X> '+IntToStr(k));
    {очищаем поле для ввода}
    mask_num.text:='';
   {устанавливаем курсор в поле ввода}
    mask_num.SetFocus;
   end;
end;
```

2. **Задание.** Скорректировать полученный проект так, чтобы можно было начать игру заново. Для этого:

- включить соответствующую кнопку, которая появляется в форме после завершения игры;

- щелчок по этой кнопке должен приводить форму в исходное состояние, на начало игры.

Критерии оценки лабораторной работы

✓ 10-8 баллов выставляется студенту, если студент выполнил все задания лабораторной работы, в том числе и самостоятельные. Фактических ошибок, связанных с пониманием проблемы, нет; семантических и синтаксических ошибок в программах нет; все структуры данных и алгоритмы реализованы верно. При защите студент отвечает на все вопросы преподавателя.

✓ 7-6 баллов – работа выполнена полностью; студент выполнил все предложенные в лабораторной работе задания, одно самостоятельное задание реализована не для всех исходных данных или есть погрешности в реализации структур данных и алгоритмов. При защите студент отвечает на все вопросы преподавателя.

✓ 5-4 балла – работа выполнена полностью. Два самостоятельных задания реализованы не для всех входных данных или есть значительные погрешности в реализации структур данных и алгоритмов. При защите студент не отвечает на 1-2 вопроса преподавателя.

✓ 1-3 балла – работа выполнена не полностью. Более трех самостоятельных задания реализованы неверно или допущены ошибки в

реализации структур данных и алгоритмов или студент демонстрирует слабое владение средой разработки. При защите студент не отвечает более, чем на 2 вопроса преподавателя.