МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

УТВЕРЖДАЮ

Декан факультета
информационных технологий

_____ Филиппович А.Ю.

"01" сентября 2019 г.

Рабочая программа дисциплины

«Объектно-ориентированное программирование»

Направление подготовки:
09.03.01 Информатика и вычислительная техника

Образовательная программа (профиль):
«Программное обеспечение информационных систем»

Год начала обучения:
2019.

Уровень образования:
бакалавриат.

Квалификация (степень) выпускника:
Бакалавр.

Форма обучения:
заочная.

Москва, 2019

Программа дисциплины «Объектно-ориентированное программирование» составлена в соответствии с Федеральным государственным образовательным стандартом высшего образования по направлению подготовки бакалавров **09.03.01 Информатика и вычислительная техника**.

Программу составил:
ст. преподаватель  /Евтихов В. Г./

Программа утверждена на заседании кафедры «Прикладная информатика» 28 августа 2019 г., протокол № 1.

Заведующий кафедрой
доцент, к.э.н.  /С.В. Суворов/

1. Цели освоения дисциплины

Цель курса – формирование у студентов базовых навыков разработки компонент программных комплексов и баз данных, включая навыки разработки баз данных и информационных систем в кооперации с коллегами, навыки обслуживания баз данных, навыки манипуляций хранящейся информацией, навыки контроля целостности, навыки управления эффективностью работы Программирование. Студенты должны научиться работать с базой данных, как с самостоятельной единицей, так и базой данной в роли элемента более сложных программных приложений.

Задачами дисциплины являются:

- обучение студентов методам построения программ и "программирования в малом", выработка навыков владения современными языками объектно-ориентированного программирования, освоение фундаментальных знаний в области технологии и практики современного программирования;
- формирование у студентов представления о основных этапах решения задач на ЭВМ, постановках задачи и спецификациях программы, об использовании стандартных типов данных;
- получение студентами базовых знаний по стандарту POSIX, стандартам языка C, а также интерфейсам прикладного программирования (API) UNIX-подобных систем;
- выработка у студентов навыков программирования отказоустойчивого и эффективного программного обеспечения, предназначенного для решения простых прикладных задач;
- знакомство студентов с вопросами переносимости программного обеспечения на различные платформы.

2. Место дисциплины в структуре ООП бакалавриата

Дисциплина «Программирование» относится к основной части основной образовательной программы бакалавриата

Логически и содержательно-методически дисциплина взаимосвязана с дисциплинами: «Информатика», «Структуры и алгоритмы компьютерной обработки данных», «ЭВМ и периферийные устройства» и «Базы данных».

Материалы дисциплины востребованы также при подготовке выпускной квалификационной работы.

3. Перечень планируемых результатов обучения по дисциплине, соотнесенные с планируемыми результатами освоения образовательной программы

В результате освоения дисциплины у обучающихся формируются следующие компетенции и должны быть достигнуты следующие результаты обучения как этап формирования соответствующих компетенций:

Код компетенции	**В результате освоения образовательной программы обучающийся должен обладать**	**Перечень планируемых результатов обучения по дисциплине**
ОК-6	способностью работать в коллективе, толерантно воспринимая социальные, этнические, конфессиональные и культурные различия	**знать:** основные технологии поддержки управления версиями для коллективной работы; **уметь:**

		работать в небольших командах над единым программным продуктом; **владеть:** основными командами системы Git.
ПК-2	способностью разрабатывать компоненты программных комплексов и баз данных, используя современные инструментальные средства и технологии программирования	**знать:** понятия алгоритма и алгоритмического языка; способы построения программ и методы доказательства их правильности; абстрактные типы данных и их реализации; базисные методы обработки информации; основные концепции объектно-ориентированного программирования и базовые конструкции языка Ruby; рекурсивные определения и алгоритмы; способы конструирования и верификации программ; основы работы с графической информацией; основные алгоритмы внутренних и внешних сортировок; **уметь:** реализовывать простейшие алгоритмы обработки информации на языке Ruby; создавать программы, используя методы написания рекурсивных программ, конструирование цикла при помощи инварианта, теорию инвариантных и индуктивных функций; использовать различные типы данных и основы объектно-ориентированного программирования; работать с простейшими графическими системами; применять алгоритмы внутренних и внешних сортировок; создавать и модифицировать небольшие программы, что является первым этапом в подготовке программиста-профессионала; ставить задачу и разрабатывать алгоритм ее решения, использовать прикладные системы программирования; работать с современными системами программирования, включая объектно-ориентированные. **владеть:** языками процедурного и объектно-ориентированного программирования; навыками разработки и отладки программ не менее, чем на одном из алгоритмических процедурных языков программирования высокого уровня; навыками проверки корректности и работоспособности программ.
ОПК-4	способностью участвовать в настройке и наладке программно-аппаратных комплексов	**знать:** современные технические и программные средства взаимодействия с ЭВМ; технологию разработки алгоритмов программ, методы отладки и решения задач на ЭВМ в различных режимах. **уметь:** применять системные средства Linux для написания прикладных программ. **владеть:** технологиями низкоуровневого программирования на языке С.

4. Структура и содержание дисциплины

Общая трудоемкость дисциплины составляет **16** зачетных единиц (540 часов: 272 – аудиторных занятий; 268 – самостоятельная работа), из которых: лекции – 17 часов; лебораторные работы – 119 часов; семинары и практические занятия – 136 часов. Распределение по семестрам: 1 семестр – 68 часов, зачет; 2 семестр – 68 часов, зачет; 3 семестр – 68 часов, зачет; 4 семестр – 68 часов, зачет.

Структура и содержание дисциплины «Программирование» по срокам и видам работы отражены в Приложении 1.

Содержание разделов дисциплины

Первый семестр

Введение в Программирование
Предмет наук информатика и программирование. Использование языка Ruby для программирования в директивном стиле. Основные управляющие конструкции языка, исключения, ввод и вывод данных. Написание простейших программ и рассмотрение эталонных программ, решающих различные задачи. Применение различных методов решения простейших задач.

Алгоритмы и их свойства
Алгоритмы и их реализация на объектно-ориентированном языке Ruby. Особенности представления чисел в ЭВМ и связанные с этим проблемы. Оценки временной и емкостной сложностей алгоритмов.

Базисные схемы обработки информации
Рекурсия и итерация, их взаимосвязь. Практическое применение методов построения и доказательства правильности рекурсивных программ. Индуктивные функции на пространстве последовательностей. Критерий индуктивности и схема вычисления индуктивной функции. Существование и единственность минимального индуктивного расширения, критерий минимальности и обобщенная схемы вычисления индуктивной функции. Решение практических задач с применением теории индуктивных функций. Схема проектирования цикла при помощи инварианта. Доказательство правильности программ, написанных с помощью этой схемы. Основные методы построения инвариантов цикла и практическое применение этих методов для решения задач. Схема вычисления инвариантной функции и ее связь со схемой проектирования цикла при помощи инварианта. Решение задач с помощью схемы вычисления инвариантной функции.

Компиляция и интерпретация
Понятие о языках и грамматиках. Компиляция и интерпретация. Различные подходы к написанию компилятора с языка простейших арифметических формул. Проект "Компилятор формул". Решение задач на модификацию.

Второй семестр

Методы ООП
Ruby, как язык ООП. Инкапсуляция, наследование и полиморфизм. Объекты и методы, типы и классы. Контейнерные типы в языке Ruby. Использование методов объектно-ориентированного программирования и теории индуктивных функций в проекте "Выпуклая оболочка". Решение задач на модификацию.

Основы работы с графической информацией
Графическое отображение проекта "Выпуклая оболочка". Решение задач на модификацию. Проект "Изображение проекции полиэдра".

Тестирование программ
Основы тестирование разработанных программ. Работа с классом Test::Unit. Тесты к проектам "Компилятор формул", "Выпуклая оболочка" и "Изображение проекции полиэдра".

Третий семестр

Реализация простейших контейнерных структур данных
Понятие и организация вектора (массива), динамического вектора, стека, очереди, дека, множества, одно- и двусвязных списков. Непрерывная и ссылочная реализация структур данных на базе вектора. Примеры непрерывной реализации ограниченного стека и ограниченной очереди на базе вектора. Хеш-функции.

Сортировки
Внутренние сортировки. Сортировка простыми включениями и анализ ее сложности. Сортировки простым выбором и простым обменом. Простейшие модификации пузырьковой сортировки. Сортировка слиянием. Алгоритм и анализ эффективности пирамидальной сортировки. Быстрая сортировка. Рекурсивная и итерационная версии. Внешние сортировки.

Командная разработка программ
Понятие о системе контроля версий. Система Git. Простейший цикл работы. Ветки и релизы. Разработка проекта в составе команды.

Четвёртый семестр

Введение в системное программирование
Понятие о системном программировании. Сравнение прикладного и системного программирования. Предмет системного программирования. Знакомство с POSIX. Передача параметров через функцию main. Обработка опций командной строки. Обработка ошибок.

Взаимодействие с файловой системой
Понятие о файле. Атрибуты файла. Понятие о файловой системе. Соглашения о записи путей к файлам. Соглашения о правах доступа. "Жесткие" и "символьные" ссылки. Операции над файлами: доступ, создание, удаление, перемещение, смена имени, смена владельца и прав доступа, изменение размера файла, изменение времени доступа, модификации и создания файла, получение информации о файле, позиционирование, чтение и запись. Работа с директориями: создание, удаление, обход.

Процессы, группы, сессии, управляющие терминалы, сигналы и демоны
Понятие о процессе, группе, сессии, управляющем терминале. Атрибуты процесса. Создание процессов, освобождение их ресурсов, ожидание дочерних процессов, запуск исполняемых файлов, изменение рабочей директории, привилегий процесса. Изменение группы процесса, сессии и управляющего терминала. Переключение между фоновыми процессами и процессами переднего плана. Получение информации о процессе. Понятие о сигналах. Назначение стандартных сигналов. Генерация, обработка и блокировка сигналов. Понятие о процессах-демонах.

Методы взаимодействия между процессами

Назначение методов взаимодействия между процессами. Сравнение методов из стандартов System V и POSIX. Каналы: безымянные и именованные. Дублирование файловых дескрипторов. Очереди сообщений: создание, обмен сообщениями, настройка, освобождение ресурсов. Разделяемая память: создание, подключение, настройка, освобождение ресурсов. Семафоры (безымянные и именованные): создание, операции увеличения и уменьшения, освобождение ресурсов. Реализация с помощью семафоров механизмов критических секций и барьеров.

Сокеты

Понятие о сокетах. Типы, семейства и протоколы сокетов. Операции над сокетами: создание, удаление, прослушивание, подключение, подтверждение подключения, обмен данными. Получение адресов хоста и информации о них. Особенности работы с сокетами различных семейств и типов.

Мультиплексированный ввод-вывод

Блокирующие и неблокирующие операции. Проблемы при обработке множества потоков ввода-вывода одним процессом. Методы их разрешения. Метод мультиплексирования. Функции select и poll: назначение, сходства и различия. Трудности при использовании мультиплексирования. Перевод файла в неблокирующий режим ввода-вывода. Применение мультиплексирования для работы с сокетами.

Нити

Понятие о нитях. Атрибуты нити. Сравнение процессов и нитей. Создание и уничтожение нитей. Стек функций освобождения ресурсов нити. Присоединение и отсоединение нитей. Особенности использования сигналов и семафоров при работе с нитями. Синхронизация нитей, применение мьютексов и условных переменных. Персональные данные нитей.

5. Образовательные технологии

Методика преподавания дисциплины «Программирование» и реализация компетентностного подхода в изложении и восприятии материала предусматривает использование следующих активных и интерактивных форм проведения групповых, индивидуальных, аудиторных занятий в сочетании с внеаудиторной работой с целью формирования и развития профессиональных навыков обучающихся:

- подготовка к лекциям и к выполнению практических работ;
- выполнение лабораторных работы;
- модификация эталонного проекта;
- использование интерактивных форм проведения занятий.

Удельный вес занятий, проводимых в интерактивных формах, определен образовательной программой, особенностью контингента обучающихся и содержанием дисциплины «Программирование» и в целом по дисциплине составляет 25% аудиторных занятий. Занятия лекционного типа составляют 6% от объема аудиторных занятий.

6. Оценочные средства для текущего контроля успеваемости, промежуточной аттестации по итогам освоения дисциплины и учебно-методическое обеспечение самостоятельной работы студентов

В процессе обучения используются следующие оценочные формы самостоятельной работы студентов, оценочные средства текущего контроля успеваемости и промежуточных аттестаций:

В первом семестре:

- проверка домашних заданий;
- проверка готовности студентов;
- проверка выполнения лабораторных работ.

Во втором семестре:

- проверка домашних заданий;
- проверка готовности студентов;
- проверка выполнения лабораторных работ.

В третьем семестре:

- проверка домашних заданий;
- проверка готовности студентов;
- проверка выполнения лабораторных работ.

В четвёртом семестре:

- проверка домашних заданий;
- проверка готовности студентов;
- проверка выполнения лабораторных работ.

6.1. Фонд оценочных средств для проведения промежуточной аттестации обучающихся по дисциплине

6.1.1. Перечень компетенций с указанием этапов их формирования в процессе освоения образовательной программы

В результате освоения дисциплины формируются следующие компетенции:

Код компетенции	В результате освоения образовательной программы обучающийся должен обладать
ОК-6	способностью работать в коллективе, толерантно воспринимая социальные, этнические, конфессиональные и культурные различия
ПК-2	способностью разрабатывать компоненты программных комплексов и баз данных, используя современные инструментальные средства и технологии программирования
ОПК-4	способностью участвовать в настройке и наладке программно-аппаратных комплексов

В процессе освоения образовательной программы данные компетенции, в том числе их отдельные компоненты, формируются поэтапно в ходе освоения обучающимися дисциплин, практик в соответствии с учебным планом и календарным графиком учебного процесса.

6.1.2. Описание показателей и критериев оценивания компетенций, формируемых по итогам освоения дисциплины, описание шкал оценивания

Показателем оценивания компетенций на различных этапах их формирования является достижение обучающимися планируемых результатов обучения по дисциплине.

ОК-6 – способностью работать в коллективе, толерантно воспринимая социальные, этнические, конфессиональные и культурные различия				
знать: основные технологии поддержки управления версиями для коллективной работы;	Обучающийся демонстрирует полное отсутствие знаний технологий поддержки управления версиями для коллективной работы.	Обучающийся может назвать наиболее популярные системы контроля версий, знает из преимущества и недостатки.	Обучающийся может назвать наиболее популярные системы контроля версий, знает из преимущества и недостатки. Способен работать с сервером github.com.	Обучающийся может назвать наиболее популярные системы контроля версий, знает из преимущества и недостатки. Способен работать с сервером github.com. Знает основы инсталляции данных пакетов в ОС Linux.
уметь: работать в небольших командах над единым программным продуктом;	Обучающийся не способен работать с несколькими ветками git.	Обучающийся умеет загружать различные ветки проекта и вносить в них простейшие изменения.	Обучающийся умеет загружать различные ветки проекта и вносить в них простейшие изменения. Он также демонстрирует умение создавать новые ветки и выполнять их слияние.	Обучающийся умеет загружать различные ветки проекта и вносить в них простейшие изменения. Он также демонстрирует умение создавать новые ветки и выполнять их слияние. Умеет осуществлять выборочный отбор изменений.
владеть: основными командами системы Git.	Обучающийся не владеет никакими командами git.	Обучающийся способен выполнить цикл внесения изменений на удалённый репозиторий git,	Обучающийся способен выполнить цикл внесения изменений на удалённый репозиторий git, умеет создавать новые проекты и просматривать историю изменений.	Обучающийся способен выполнить цикл внесения изменений на удалённый репозиторий git, умеет создавать новые проекты и просматривать историю изменений. Умеет выборочно отменять изменения.
ПК-2 – способностью разрабатывать компоненты программных комплексов и баз данных, используя современные инструментальные средства и технологии программирования				

знать: понятия алгоритма и алгоритмического языка; способы построения программ и методы доказательства их правильности; абстрактные типы данных и их реализации; базисные методы обработки информации; основные концепции объектно-ориентированного программирования и базовые конструкции языка Ruby; рекурсивные определения и алгоритмы; способы конструирования и верификации программ; основы работы с графической информацией; основные алгоритмы внутренних и внешних сортировок	Обучающийся не способен продемонстрировать знания основных абстрактных типов данных и алгоритмов сортировок.	Обучающийся знает основные абстрактные типы данных, их особенности и преимущества, а также демонстрирует знания основных алгоритмов сортировок, но не способен создать полноценные реализации соответствующих программ.	Обучающийся знает основные абстрактные типы данных, их особенности и преимущества, а также демонстрирует знания основных алгоритмов сортировок, способен создать полноценные реализации соответствующих программ.	Обучающийся знает основные абстрактные типы данных, их особенности и преимущества, а также демонстрирует знания основных алгоритмов сортировок, способен создать полноценные реализации соответствующих программ. Способен выполнить асимптотическую оценку сложности данных алгоритмов.

уметь: реализовывать простейшие алгоритмы обработки информации на языке Ruby; создавать программы, используя методы написания рекурсивных программ, конструирование цикла при помощи инварианта, теорию инвариантных и индуктивных функций; использовать различные типы данных и основы объектно-ориентированного программирования; работать с простейшими графическими системами; применять алгоритмы внутренних и внешних сортировок; создавать и модифицировать небольшие программы, что является первым этапом в подготовке программиста-профессионала; ставить задачу и разрабатывать алгоритм ее решения, использовать прикладные системы программирования;	Обучающийся не способен реализовать простейшие алгоритмы ни на языке Ruby, ни на языке C++.	Обучающийся е способен реализовать простейшие алгоритмы на языке Ruby, но не способен реализовать их на языке C++.	Обучающийся е способен реализовать простейшие алгоритмы как на языке Ruby, так и на языке C++.	Обучающийся е способен реализовать простейшие алгоритмы как на языке Ruby, так и на языке C++. Умеет оперировать ключами оптимизации компилятора g++.

работать с современными системами программирования, включая объектно-ориентированные.				
владеть: языками процедурного и объектно-ориентированного программирования; навыками разработки и отладки программ не менее, чем на одном из алгоритмических процедурных языков программирования высокого уровня; навыками проверки корректности и работоспособности программ.	Обучающийся не способен продемонстрировать знание основных конструкций языков Ruby, C++.	Обучающийся способен продемонстрировать знание основных конструкций языка Ruby, но не владеет языком , C++.	Обучающийся способен продемонстрировать знание основных конструкций языков Ruby и C++.	Обучающийся способен продемонстрировать знание основных конструкций языков Ruby, C++. Способен внести изменения в сложный проект.
ОПК-4 – способностью участвовать в настройке и наладке программно-аппаратных комплексов				

знать: современные технические и программные средства взаимодействия с ЭВМ; технологию разработки алгоритмов программ, методы отладки и решения задач на ЭВМ в различных режимах.	Обучающийся не владеет средствами отладки программ gdb, strace, valgrind, gprof.	Обучающийся знает основы применения средств дебаггинга gdb и профайлинга gprof.	Обучающийся знает основы применения средств дебаггинга gdb и профайлинга gprof. Понимает как выполнять обнаружение утечек памяти с помощью valgrind.	Обучающийся знает основы применения средств дебаггинга gdb и профайлинга gprof. Понимает как выполнять обнаружение утечек памяти с помощью valgrind. Знает как выполнять анализ процесса с помощью strace.
уметь: применять системные средства Linux для написания прикладных программ.	Обучающийся не способен использовать средства System V, нити и другие системные средства Linux.	Обучающийся способен использовать простейшие возможности средств System V.	Обучающийся способен использовать простейшие возможности средств System V. Способен написать простейший пример использования нитей.	Обучающийся полностью владеет возможностями средств System V. Демонстрирует уверенные знания в использовании нитей при написании программ.
владеть: технологиями низкоуровневого программирования на языке С.	Обучающийся не способен написать простейшую программу на языке С,	Обучающийся способен написать простейшую программу на языке С, но демонстрирует плохие знания вопросов явного выделения памяти и работы с указателями.	Обучающийся способен написать простейшую программу на языке С, Демонстрирует хорошие знания вопросов явного выделения памяти и работы с указателями.	Обучающийся способен написать простейшую программу на языке С, Демонстрирует хорошие знания вопросов явного выделения памяти и работы с указателями. Владеет основами асинхронного низкоуровневого ввода/вывода.

Шкалы оценивания результатов промежуточной аттестации и их описание:

Форма промежуточной аттестации: зачет.

Промежуточная аттестация обучающихся в форме зачёта проводится по результатам выполнения всех видов учебной работы, предусмотренных учебным планом по данной дисциплине, при этом учитываются результаты текущего контроля успеваемости в течение семестра. Оценка степени достижения обучающимися планируемых результатов обучения по дисциплине проводится преподавателем, ведущим занятия по дисциплине методом экспертной

оценки. По итогам промежуточной аттестации по дисциплине выставляется оценка «зачтено» или «не зачтено».

К промежуточной аттестации допускаются только студенты, выполнившие все виды учебной работы, предусмотренные рабочей программой по дисциплине «Программирование» (которые прошли промежуточный контроль, выполнили курсовую и лабораторные работы).

Шкала оценивания	**Описание**
Зачтено	Выполнены все виды учебной работы, предусмотренные учебным планом. Студент демонстрирует соответствие знаний, умений, навыков приведенным в таблицах показателей, оперирует приобретенными знаниями, умениями, навыками, применяет их в ситуациях повышенной сложности. При этом могут быть допущены незначительные ошибки, неточности, затруднения при аналитических операциях, переносе знаний и умений на новые, нестандартные ситуации.
Не зачтено	Не выполнен один или более видов учебной работы, предусмотренных учебным планом. Студент демонстрирует неполное соответствие знаний, умений, навыков приведенным в таблицах показателей, допускаются значительные ошибки, проявляется отсутствие знаний, умений, навыков по ряду показателей, студент испытывает значительные затруднения при оперировании знаниями и умениями при их переносе на новые ситуации.

Фонды оценочных средств представлены в приложении 1 к рабочей программе.

7. Учебно-методическое и информационное обеспечение дисциплины.

а) основная литература:

1. Мейер Б. Инструменты, алгоритмы и структуры данных [Электронный ресурс] — Издательство: ИНТУИТ, 2016 г. — 543с. — Режим доступа: http://www.knigafund.ru/books/177698/read
2. Белоцерковская И. Е. Алгоритмизация. Введение в язык программирования C++ / Белоцерковская И. Е., Галина Н. В., Катаева Л. Ю. [Электронный ресурс] — Издательство: ИНТУИТ, 2016 г. — 197с. — Режим доступа: http://www.knigafund.ru/books/177446/read

б) дополнительная литература:

1. Царев Р. Ю. Программирование на языке Си: учебное пособие, [Электронный ресурс] — Издательство: Сибирский федеральный университет, 2014 г. – 108с. — Режим доступа: http://www.knigafund.ru/books/184201/read.

в) программное обеспечение и интернет-ресурсы:

1. Операционная система Linux (свободное ПО)
2. Браузер Mozilla Firefox (свободное ПО)
3. Интерпретатор языка Ruby (свободное ПО)
4. Компилляторы gcc, gcc+ (свободное ПО)
5. Система контроля версий Git (свободное ПО)
6. Компилятор языка Assembler (свободное ПО)
7. Офисные приложения LibreOffice для Linux (свободное ПО)
8. Офисные приложения Microsoft Office 2013(или ниже) - Microsoft Open License. Лицензия № 61984042

9. Microsoft office 2013 prof (для обучения). Госконтракт № 18-09/14 от 22.09.2014 Акт № Тг09950
10. Текстовый редактор Gedit (свободное ПО)
11. Среда разработки программ Netbeans (свободное ПО)

8. Материально-техническое обеспечение дисциплины.

Аудитории общего фонда для лекционных, практических и семинарских занятий ул. Автозаводская, 16. ауд. 1201, 1202, оснащенные: столы, стулья, аудиторная доска. Рабочее место преподавателя: стол, стул.

9. Методические рекомендации для самостоятельной работы студентов

Изучение дисциплины осуществляется в строгом соответствии с целевой установкой в тесной взаимосвязи учебным планом. Основой теоретической подготовки студентов являются лекции. При рассмотрении учебного материалы рекомендуется делать акцент на практические примеры, демонстрировать их реальную работу с помощью проектора.

В процессе самостоятельной работы студенты закрепляют и углубляют знания, полученные во время аудиторных занятий, дорабатывают конспекты лекций, готовятся к экзамену, а также самостоятельно изучают отдельные темы учебной программы.

Важным обстоятельством является привлечение внимания студентов к обсуждаемой проблеме, стимулирование интереса к ней и организация активного обсуждения, как структуры проблемы, так и составляющих ее наиболее актуальных тем. Для повышения эффективности проведения занятия требуется предварительная подготовка всех его участников. В этой связи рекомендуется заблаговременно (не менее, чем за неделю) оповестить студентов о теме занятия, дать перечень литературы по теме.

При проведении практического занятия преподаватель выполняет, в основном, функции ведущего – направляет студентов в правильное русло решения задач, рассматривает оптимальность предложенных решений, корректирует возможные ошибки.

Активная работа студента на практическом занятии учитывается при определении итоговой оценки его знаний по дисциплине на экзамене.

Проведение лабораторных работ осуществляется в виде разработки небольшими студенческими группами прикладных программ или модификации готовых эталонных проектов.

При выполнении модификации готового приложения студентам необходимо внести изменения в эталонный проект так, чтобы его функционал был расширен новыми возможностями без потери старых Для сдачи проекта необходимо ответить на 4 из 5 вопросов по проекту. Проект рекомендуется оценивать по 14 балльной шкале. Каждая неудачная попытка сдачи уменьшает наибольшее возможное количество баллов за выполнение проекта на 1 балл. Результат 13-14 баллов формирует компетенции на продвинутом уровне. Результат 3-12 формирует компетенции на базовом уровне. Компетенции считаются неосвоенными при количестве баллов менее 3.

Самостоятельная работа по дисциплине «Программирование» предполагает: выполнение студентами домашних заданий. Домашние задания являются, как правило, продолжением практических занятий и содействуют овладению практическими навыками по основным разделам дисциплины. Самостоятельная работа студентов предполагает изучение теоретического и практического материала по актуальным вопросам дисциплины. Рекомендуется самостоятельное изучение учебной и научной литературы, использование справочной литературы и др..

При выдаче заданий на самостоятельную работу используется дифференцированный подход к студентам. Перед выполнением студентами самостоятельной внеаудиторной работы преподаватель проводит инструктаж по выполнению задания, который включает: цель здания, его содержание, сроки выполнения, ориентировочный объем работы, основные требования к результатам работы, критерии оценки. В процессе инструктажа преподаватель предупреждает студентов о возможных типичных ошибках, встречающихся при выполнении задания. Инструктаж проводится преподавателем за счет объема времени, отведенного на изучение дисциплины.

Текущий контроль осуществляется на практических занятиях, промежуточный контроль осуществляется на зачете, экзамене в письменной (устной) форме.

Самостоятельная работа осуществляется индивидуально.

Контроль самостоятельной работы организуется в двух формах:

- самоконтроль и самооценка студента;
- контроль со стороны преподавателей (текущий и промежуточный).

Текущий контроль осуществляется на практических занятиях, промежуточный контроль осуществляется на зачете, экзамене в письменной (устной) форме.

Критериями оценки результатов самостоятельной работы студента являются:

- уровень освоения студентом учебного материала;
- умения студента использовать теоретические знания при выполнении практических задач;
- сформированность умений;
- оформление материала в соответствии с требованиями.

10. Методические указания для преподавателя

Проведение занятий по дисциплине «Программирование» осуществляется в строгом соответствии с целевой установкой и в тесной взаимосвязи с учебным планом. Основой теоретической подготовки студентов являются лекции. При рассмотрении учебных материалов рекомендуется делать акцент на практические примеры, демонстрировать их реальную работу с помощью проектора.

В процессе самостоятельной работы студенты закрепляют и углубляют знания, полученные во время аудиторных занятий, дорабатывают конспекты лекций, готовятся к экзамену, а также самостоятельно изучают отдельные темы учебной программы.

Важным обстоятельством является привлечение внимания студентов к обсуждаемой проблеме, стимулирование интереса к ней и организация активного обсуждения, как структуры проблемы, так и составляющих ее наиболее актуальных тем. Для повышения эффективности проведения занятия требуется предварительная подготовка всех его участников. В этой связи рекомендуется заблаговременно (не менее, чем за неделю) оповестить студентов о теме занятия, дать перечень литературы по теме.

При проведении практического занятия преподаватель выполняет, в основном, функции ведущего – направляет студентов в правильное русло решения задач, рассматривает оптимальность предложенных решений, корректирует возможные ошибки.

Активная работа студента на практическом занятии учитывается при определении итоговой оценки его знаний по дисциплине на экзамене.

Самостоятельная работа по дисциплине «Программирование» предполагает: выполнение студентами домашних заданий. Домашние задания являются, как правило, продолжением практических занятий и содействуют овладению практическими навыками по основным разделам

дисциплины. Самостоятельная работа студентов предполагает изучение теоретического и практического материала по актуальным вопросам дисциплины. Рекомендуется самостоятельное изучение учебной и научной литературы, использование справочной литературы и др.

При выдаче заданий на самостоятельную работу используется дифференцированный подход к студентам. Перед выполнением студентами самостоятельной внеаудиторной работы преподаватель проводит инструктаж по выполнению задания, который включает: цель здания, его содержание, сроки выполнения, ориентировочный объем работы, основные требования к результатам работы, критерии оценки. В процессе инструктажа преподаватель предупреждает студентов о возможных типичных ошибках, встречающихся при выполнении задания. Инструктаж проводится преподавателем за счет объема времени, отведенного на изучение дисциплины.

Текущий контроль осуществляется на практических занятиях, промежуточный контроль осуществляется на экзамене в письменной или устной форме.

Самостоятельная работа осуществляется индивидуально. Контроль самостоятельной работы организуется в двух формах:

- самоконтроль и самооценка студента;
- контроль со стороны преподавателей (текущий и промежуточный).

Критериями оценки результатов самостоятельной работы студента являются:

- уровень освоения студентом учебного материала;
- умения студента использовать теоретические знания при выполнении практических задач;
- сформированность умений;
- оформление материала в соответствии с требованиями.

Приложение 1 к
рабочей программе

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
(МОСКОВСКИЙ ПОЛИТЕХ)

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»

Профиль подготовки «Программное обеспечение информационных систем»

Форма обучения: очная
Вид профессиональной деятельности: проектно-технологическая

Кафедра: «Прикладная информатика»

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ

ПО ДИСЦИПЛИНЕ

«Программирование»

Состав: 1. Паспорт фонда оценочных средств

2. Описание оценочных средств:
перечень вопросов к зачету,

перечень вопросов к экзамену,

варианты заданий контрольных работы,

темы индивидуальных проектов,

темы групповых проектов,

темы курсовых работ,

задания для расчётно-графической работы

Составитель:

ст. преподаватель

Евтихов В. Г.

Москва, 2019 год

ПОКАЗАТЕЛЬ УРОВНЯ СФОРМИРОВАННОСТИ КОМПЕТЕНЦИЙ

Программирование					
ФГОС ВО **09.03.01 «Информатика и вычислительная техника»**					
В процессе освоения данной дисциплины студент формирует и демонстрирует следующие **Общекультурные, общепрофессиональные и профессиональные компетенции**:					
КОМПЕТЕНЦИИ		**Перечень компонентов**	**Технология формирования компетенций**	**Форма оценочного средства****	**Степени уровней освоения компетенций**
ИН-ДЕКС	**ФОРМУЛИРОВКА**				
ОК-6	способностью работать в коллективе, толерантно воспринимая социальные, этнические, конфессиональные и культурные различия	**Знать:** основные технологии поддержки управления версиями для коллективной работы. **Уметь:** работать в небольших командах над единым программным продуктом. **Владеть:** основными командами системы Git.	Лекция, Лабораторная работа, Самостоятельная работа, семинарские занятия	РЗЗ, ЗЛР, П, Зач	**Базовый уровень:** работать в коллективе при решении практических задач **Повышенный уровень:** способен быть лидером команды при решении практических задач

ПК-2	способностью разрабатывать компоненты программных комплексов и баз данных, используя современные инструментальные средства и технологии программирования	**Знать:** понятия алгоритма и алгоритмического языка; способы построения программ и методы доказательства их правильности; абстрактные типы данных и их реализации; базисные методы обработки информации; основные концепции объектно-ориентированного программирования и базовые конструкции языка Ruby; рекурсивные определения и алгоритмы; способы конструирования и верификации программ; основы работы с графической информацией; основные алгоритмы внутренних и внешних сортировок. **Уметь:** реализовывать простейшие алгоритмы обработки информации на языке Ruby; создавать программы, используя методы написания рекурсивных программ, конструирование цикла при помощи инварианта, теорию инвариантных и индуктивных функций; использовать различные типы данных и основы объектно-ориентированного программирования; работать с простейшими графическими системами; применять алгоритмы внутренних и внешних сортировок; создавать и модифицировать небольшие программы, что является первым этапом в подготовке программиста-профессионала; ставить задачу и разрабатывать алгоритм ее решения, использовать прикладные системы программирования; работать с современными системами программирования, включая объектно-ориентированные. **Владеть:** языками процедурного и объектно-ориентированного программирования; навыками разработки и отладки программ не менее, чем на одном из алгоритмических процедурных языков программирования высокого уровня; навыками проверки корректности и работоспособности программ.	Лекция, Лабораторная работа, Самостоятельная работа, семинарские занятия	РЗЗ, ЗЛР, П, Зач	**Базовый уровень** владеет способностью разрабатывать небольшие программные компоненты, вносить изменения в готовые программные комплексы, на высоком уровне владеет несколькими языками программирования. **Повышенный уровень** владеет способностью разрабатывать небольшие программные компоненты, вносить изменения в готовые программные комплексы, на высоком уровне владеет несколькими языками программирования; умеет разрабатывать программные компоненты, требующие навыков системного программирования; умеет разрабатывать элементы АПК.

ОПК-4	способностью участвовать в настройке и наладке программно-аппаратных комплексов	**Знать:** современные технические и программные средства взаимодействия с ЭВМ; технологию разработки алгоритмов программ, методы отладки и решения задач на ЭВМ в различных режимах. **Уметь:** применять системные средства Linux для написания прикладных программ. **Владеть:** технологиями низкоуровневого программирования на языке С.	Лекция, Лабораторная работа, Самостоятельная работа, семинарские занятия	РЗЗ, ЗЛР, П, Зач	**Базовый уровень:** владеет базовыми навыками настройки и наладки простейших АПК. **Повышенный уровень:** владеет навыками настройки и наладки простейших АПК в экстремальных ситуациях и для нестандартных наборов программного и аппаратного обеспечения.

Перечень оценочных средств по дисциплине «Программирование»

№ ОС	Наименование оценочного средства	Краткая характеристика оценочного средства	Представление оценочного средства в ФОС
1	Зачет (Зач)	Средство промежуточной аттестации студента, проводится в письменно-устной форме.	Перечень вопросов к зачету
2	Разноуровневые задачи и задания (РЗЗ)	Различают задачи и задания: а) репродуктивного уровня, позволяющи оценивать и диагностировать знание фактического материала (базовые понятия, алгоритмы, факты) и умение правильно использовать специальные термины и понятия, узнавание объектов изучения в рамках определенного раздела дисциплины; б) реконструктивного уровня, позволяющие оценивать и диагностировать умения синтезировать, анализировать, обобщать фактический и теоретический материал с формулированием конкретных выводов, установлением причинно- следственных связей; в) творческого уровня, позволяющие оценивать и диагностировать умения, интегрировать знания различных областей, аргументировать собственную точку зрения.	Комплект разноуровневых задач и заданий
3	Защита лабораторной работы (ЗЛР)	Средство проверки умений применять полученные знания для решения прикладных задач определенного типа по теме или разделу и обоснованно защищать свой разработки перед компетентным специалистом.	Комплект тем лабораторных работ
4	Проект (П)	Конечный продукт, получаемый в результате планирования и выполнения комплекс учебных и исследовательских заданий. Позволяет оценить умения обучающихся самостоятельно конструировать свои знания в процессе решения практических задач и проблем, ориентироваться в информационном пространстве и уровень сформированности аналитических, исследовательских навыков, навыков практического и творческого мышления. Может выполняться в индивидуальном порядке или группой обучающихся.	Темы групповых и/или индивидуальных проектов

Примерный перечень вопросов к зачёту по курсу «Программирование»:

1. Введите с клавиатуры три действительных числа a, b и c и напечатайте мощность множества решений уравнения ax^2 + bx + c = 0.
2. Введите с клавиатуры два натуральных числа i и j и напечатайте значение функции Аккермана A(i,j).
3. Введите с клавиатуры непустую последовательность четвёрок действительных чисел и, считая их координатами противоположных вершин стандартных прямоугольников, напечатайте, имеют ли они непустое пересечение. Стандартным прямоугольником называется прямоугольник со сторонами, параллельными осям координат.
4. Введите с клавиатуры непустую последовательность натуральных чисел и напечатайте наибольший общий делитель всех введённых чисел.
5. Введите с клавиатуры непустую последовательность целых чисел и напечатайте число локальных максимумов в ней. Элемент считается локальным максимумом, если он не имеет соседей больших, чем он сам.
6. Введите с клавиатуры натуральное число, большее единицы, и напечатайте, является ли оно простым.
7. Введите с клавиатуры натуральное число n и напечатайте количество счастливых билетов с 2n-значными номерами. Билет считается счастливым, если сумма первых n цифр его номера равна сумме n последних.
8. Введите целое число и напечатайте его представление в виде суммы четырёх квадратов целых чисел или No, если такого представления нет.
9. Введите положительное действительное число и напечатайте простую дробь, знаменатель которой не превосходит ста, наиболее близкую к корню квадратному из введённого числа.
10. Введите из файла коэффициенты системы линейных уравнений и напечатайте её решение методом Гаусса с выбором наибольшего элемента.
11. Введите последовательность целых коэффициентов алгебраического уравнения и напечатайте все его рациональные корни.
12. Введите последовательность пар действительных чисел и, рассматривая их как координаты точек на плоскости, напечатайте наибольшее количество точек, лежащих на одной прямой.
13. Введите последовательность четвёрок действительных чисел (x, y, z, r) и, рассматривая их как координаты центров сфер и их радиусов, напечатайте, вложены ли эти сферы друг в друга, как матрёшки (в произвольном порядке).
14. Напечатайте число e со ста точными знаками после запятой.
15. Напечатайте число pi со ста точными знаками после запятой.
16. Напечатайте на экране терминала все такие расстановки восьми ферзей на шахматной доске, при которых никакие два из них не бьют друг друга, и общее количество таких расстановок.
17. Калькулятор. Вычисление выражений в постфиксной форме.
18. Реализуйте циклическую очередь.
19. Постройте приоритеную очередь. При добавлении элемента в такую очередь порядковый номер элемента должен определятся его приоритетом.
20. Реализуйте L1 список целых чисел с методами и операцией пересечения.
21. Реализуйте L1 список целых чисел с методами и операцией объединения.
22. Реализуйте L2 список целых чисел с методами и операцией симметрическая разность.
23. Реализуйте бинарное упорядоченное дерево с методами и с восходящим проходом по дереву.

24. Реализуйте бинарное упорядоченное дерево с методами и с последовательным проходом по дереву.
25. Напишите программу красивой печати бинарного дерева (корень должен находится между двумя потомками).
26. Имеется N камней веса A1,A2,...,AN. Необходимо разбить их на две кучи таким образом, чтобы веса куч отличались не более чем в 2 раза. Если этого сделать нельзя, то указать это.
27. Имеется N человек и целые числа A1, ..., AN; человека i необходимо познакомить с Ai людьми. Можно ли это сделать?
28. Даны две целочисленных таблицы A [1:10] и B[1:15]. Разработать алгоритм и написать программу, которая проверяет, являются ли эти таблицы похожими. Две таблицы называются похожими, если совпадают множества чисел, встречающихся в этих таблицах.
29. Задается словарь. Найти в нем все анаграммы (слова, составленные из одних и тех же букв).
30. Вводится строка. Каждый символ заносится в элемент списка (L1). Написать программу которая выводит строку в обратном порядке.
31. Есть два отсортированных в порядке неубывания массива A[1,N] и B[1,M]. Получить отсортированный по неубыванию массив C[1,N+M], состоящий из элементов массивов A и B (слить вместе массивы A и B).
32. Задано семейство множеств букв. Найти такое k, для которого можно построить множество, состоящее из k букв, причем каждая из них принадлежит ровно k множествам заданного семейства.
33. Построить алгоритм, выдающий без повторений все перестановки N чисел.
34. Сгенерировать все подмножества данного n-элементного множества {0,..., n-1}.
35. Данные N косточек домино по правилам игры выкладываются в прямую цепочку, начиная с косточки, выбранной произвольно,в оба конца до тех пор, пока это возможно. Построить алгоритм, позволяющий определить такой вариант выкладывания заданных косточек, при котором к моменту, когда цепочка не может быть продолжена, на руках останется максимальное число очков.
36. Сгенерировать все подмножества данного n-элементного множества {0,..., n-1}.
37. Перечислить все вершины ориентированного графа, доступные из данной, в порядке увеличения длины пути от нее. (Тем самым мы решим задачу о кратчайших путях, когда цены ребер равны 1 или бесконечны.) Придумать алгоритм решения этой задачи с числом действий не более C*(число ребер, выходящих из интересующих нас вершин).
38. Неориентированный граф называется двудольным, если его можно раскрасить в два цвета так, что концы любого ребра - разного цвета. Составить алгоритм проверки, является ли заданный граф двудольным (число действий не провосходит C*(число ребер + число вершин).
39. Пусть имеется n городов, пронумерованных числами от 1 до n. Для каждой пары городов с номерами i, j в таблице a[i][j] хра- нится целое число - цена прямого авиабилета из города i в город j. Считается, что рейсы существуют между любыми городами, a[i,i] = 0 при всех i, a[i][j] может отличаться от a[j,i]. Наименьшей стоимостью проезда из i в j считается минимально возможная сумма цен билетов для маршрутов (в том числе с пересадками), ведущих из i в j. (Она не превосходит a[i][j], но может быть меньше.) Найти наименьшую стоимость проезда из 1-го города во все остальные за время O(n в степени 3).
40. Пусть имеется n городов, пронумерованных числами от 1 до n. Для каждой пары городов с номерами i, j в таблице a[i][j] хра- нится целое число - цена прямого авиабилета из города i в город j. Считается, что рейсы существуют между любыми

городами, a[i,i] = 0 при всех i, a[i][j] может отличаться от a[j,i]. Наименьшей стоимостью проезда из i в j считается минимально возможная сумма цен билетов для маршрутов (в том числе с пересадками), ведущих из i в j. (Она не превосходит a[i][j], но может быть меньше.) Найти наименьшую стоимость проезда из 1-го города во все остальные за время O(n в степени 3).

41. Пусть имеется n городов, пронумерованных числами от 1 до n. Для каждой пары городов с номерами i, j в таблице a[i][j] хранится целое число - цена прямого авиабилета из города i в город j. Считается, что рейсы существуют между любыми городами, a[i,i] = 0 при всех i, a[i][j] может отличаться от a[j,i]. Наименьшей стоимостью проезда из i в j считается минимально возможная сумма цен билетов для маршрутов (в том числе с пересадками), ведущих из i в j. (Она не превосходит a[i][j], но может быть меньше.)Известны, что все цены неотрицательны. Найти наименьшую стоимость проезда 1->i для всех i=1..n за время O(n в степени 2).
42. Перечислить все возрастающие последовательности длины k из чисел 1..n в лексикографическом порядке. Пример: при n=5, k=2 получаем 12 13 14 15 23 24 25 34 35 45.
43. Перечислить все вложения (функции, переводящие разные элементы в разные) множества {1..k} в {1..n} (предполагается, что k <= n). Порождение очередного элемента должно требовать порядка k действий.
44. Перечислить все расстановки скобок в произведении n сомножителей. Порядок сомножителей не меняется, скобки полностью определяют порядок действий. Например, для n = 4 есть 5 расста- новок ((ab)c)d, (a(bc))d, (ab)(cd), a((bc)d), a(b(cd)).
45. Предположим, что ориентированный граф без циклов хранится в такой форме: для каждого i от 1 до n в num[i] хранится число выходящих из i стрелок, в adr[i][1],..., adr[i][num[i]] - номера вершин, куда эти стрелки ведут. Составить (рекурсивный) алгоритм, который производит топологическую сортировку не более чем за C*(n+m) действий, где m - число ребер графа (стрелок). (Топологическая сортировка : требуется расположить вершины графа (точки) в таком порядке, чтобы конец любой стрелки предшествовал ее началу.)
46. Составить нерекурсивный алгоритм топологической сортировки ориентированного графа без циклов.
47. Игра Ханойские башни состоит в следующем. Есть три стержня. На первый из них надета пирамидка из n колец (большие кольца снизу, меньшие сверху). Требуется переместить кольца на другой стержень. Разрешается перекладывать кольца со стержня на стержень, но класть большее кольцо поверх меньшего нельзя. Составить программу, указывающую требуемые действия.
48. Написать программу, которая печатает по одному разу все последовательности длины n, составленные из чисел 1..k (их количество равно k в степени n).
49. Напечатать все возрастающие последовательности длины k, элементами которых являются натуральные числа от 1 до n. (Предполагается, что k не превосходит n - иначе таких последова- тельностей не существует.)
50. Перечислить все последовательности длины n из чисел 1..k в таком порядке, чтобы каждая следующая отличалась от предыдущей в единственной цифре, причем не более, чем на 1.
51. Отличия системного и прикладного программирования.
52. Переносимый интерфейс операционных систем POSIX.
53. Функция main в языке C.
54. Обработка опций командной строки. Пример кода.
55. Различные подходы к обработке ошибок. Пример кода.
56. Понятие файла и файловой системы. Типы файлов. Соглашение о путях к файлам.

57. Система прав доступа к файлам ОС семейства UNIX.
58. Получение доступа к файлам, создание и удаление файлов.
59. Управление файлами.
60. Получение информации и о файле.
61. Позиционирование, чтение и запись в файле
62. Работа с директориями
63. Процессы в ОС семейства UNIX. Группы процессов, сессии. Управляющий терминал.
64. Ресурсы процесса. Дочерние и родительские процессы.
65. Порождение новых процессов.
66. Ожидание процессов. Освобождение ресурсов.
67. Управление процессами.
68. Изменение группы и сессии процесса. Группы переднего плана и фоновые процессы.
69. Получение информации о процессе.
70. Сигналы.
71. Демоны.
72. Различия между средствами взаимодействия между процессами Sytem V и POSIX.
73. Каналы.
74. Перенаправление ввода-вывода.
75. Очереди сообщений.
76. Разделяемая память.
77. Отображение в адресное пространство обычных файлов.
78. Семафоры.
79. Понятие сокета. Семейства и типы сокетов. Связанные протоколы.
80. Создание и удаление сокетов.
81. Получение возможных адресов хоста и информации и них.
82. Связывание сокета с адресом хоста.
83. Слушающий режим сокетов.
84. Подключение к сокетам.
85. Подтверждение подключения.
86. Обмен данными с помощью сокетов.
87. Понятие о мультиплексированном вводе-выводе.
88. Функция select.
89. Функция poll.
90. Особенности мультиплексирования.
91. Понятие о нитях. Отличие нитей от процессов.
92. Создание и уничтожение нитей.
93. Присоединение и отсоединение нитей.
94. Стек функций освобождения ресурсов нитей.
95. Особенности обработки сигналов нитями.
96. Семафоры нитей.
97. Мьютексы.
98. Условные переменные.
99. Реализация барьерной синхронизации.
100. Персональные данные нитей.

Задания для проекта «Компилятор формул»

Задание лабораторной работы состоит в модификации эталонного проекта «Компилятор формул», разработанного кафедрой информационных систем и технологий. Проект представляет собой компилятор простейших алгебраических формул из инфиксной формы записи в постфиксную форму. Проект также содержит интерпретатор простейших алгебраических выражений.

Варианты заданий

Внесите изменения в проект, приводящие к указанным ниже изменениям функциональности.

1. Деление трактуется, как правоассоциативная операция.
2. Приоритеты операций сложения и вычитания выше, чем у операций умножения и деления.
3. В качестве имён переменных допускаются произвольные идентификаторы языка Ruby.
4. Для группировки в формулах можно использовать не только круглые, но также квадратные и фигурные скобки.
5. Допускаются формулы, запись которых содержит пробелы и состоит из нескольких строк.
6. Допускаются формулы, запись которых содержит комментарии двух видов: /* комментарий */ и #.
7. Для коммутативных операций аргументы в программе для стекового компилятора появляются в алфавитном порядке.
8. Перед обработкой каждого символа формулы печатается её откомпилированная часть, содержимое стека отложенных операций и необработанная ещё часть формулы.
9. Формулы, содержащие только записанные в десятичной системе счисления натуральные числа, абсолютная величина которых не превосходит 3999, компилируются в программы для стекового калькулятора, содержащие восьмеричные числа (запись которых обязана начинаться с нуля), и при этом операция вычитания является правоассоциативной.
10. Формулы, содержащие только записанные в десятичной системе счисления натуральные числа, абсолютная величина которых не превосходит 3999, компилируются в программы для стекового калькулятора, содержащие шестнадцатеричные числа (запись которых обязана начинаться с 0x или 0X), и при этом операция деления является правоассоциативной.
11. Формулы, содержащие только записанные в десятичной системе счисления натуральные числа, абсолютная величина которых не превосходит 3999, компилируются в программы для стекового калькулятора, содержащие двоичные числа (запись которых обязана начинаться с 0b или 0B), и при этом приоритет вычитания является максимальным.
12. Формулы, содержащие только записанные в десятичной системе счисления натуральные числа, абсолютная величина которых не превосходит 3999, компилируются в программы для стекового калькулятора, содержащие римские числа (в записи которых используются цифры I, V, X, L, C, D и M).
13. Формулы, содержащие только записанные в восьмеричной системе натуральные числа (обязательно начинающиеся с нуля), абсолютная величина которых не

превосходит 3999, компилируются в программы для стекового калькулятора, содержащие только записанные в десятичной системе счисления числа, и при этом приоритет деления является минимальным.

14. Формулы, содержащие только записанные в шестнадцатеричной системе натуральные числа (обязательно начинающиеся с 0x или 0X), абсолютная величина которых не превосходит 3999, компилируются в программы для стекового калькулятора, содержащие только записанные в десятичной системе счисления числа, и при этом деление трактуется как правоассоциативная операция.
15. Формулы, содержащие только записанные в двоичной системе натуральные числа (обязательно начинающиеся с 0b или 0B), абсолютная величина которых не превосходит 3999, компилируются в программы для стекового калькулятора, содержащие только записанные в десятичной системе счисления числа, и при этом вычитание трактуется как правоассоциативная операция.
16. Формулы, содержащие только записанные в римской системе натуральные числа (с цифрами I, V, X, L, C, D и M), абсолютная величина которых не превосходит 3999, компилируются в программы для стекового калькулятора, содержащие только записанные в десятичной системе счисления числа.
17. Формулы, содержащие только записанные в восьмеричной системе натуральные числа (обязательно начинающиеся с нуля), абсолютная величина которых не превосходит 3999, компилируются в программы для стекового калькулятора, содержащие только записанные в двоичной системе счисления числа (запись которых обязана начинаться с 0b или 0B), и при этом приоритет операции умножения является минимальным.
18. Формулы, содержащие только записанные в восьмеричной системе натуральные числа (обязательно начинающиеся с нуля), абсолютная величина которых не превосходит 3999, компилируются в программы для стекового калькулятора, содержащие только записанные в шестнадцатеричной системе счисления числа (запись которых обязана начинаться с 0x или 0X), и при этом операция деления является правоассоциативной.
19. Формулы, содержащие только записанные в восьмеричной системе натуральные числа (обязательно начинающиеся с нуля), абсолютная величина которых не превосходит 3999, компилируются в программы для стекового калькулятора, содержащие римские числа (в записи которых используются цифры I, V, X, L, C, D и M).
20. Формулы, содержащие только записанные в двоичной системе натуральные числа (запись которых обязана начинаться с 0b или 0B), абсолютная величина которых не превосходит 3999, компилируются в программы для стекового калькулятора, содержащие только записанные в восьмеричной системе счисления числа (запись которых обязана начинаться с нуля), и при этом приоритет операции сложения является максимальным.
21. Формулы, содержащие только записанные в двоичной системе натуральные числа (запись которых обязана начинаться с 0b или 0B), абсолютная величина которых не превосходит 3999, компилируются в программы для стекового калькулятора, содержащие только записанные в шестнадцатеричной системе счисления числа (запись которых обязана начинаться с 0x или 0X), и при этом операция вычитания является правоассоциативной.
22. Формулы, содержащие только записанные в двоичной системе натуральные числа (запись которых обязана начинаться с 0b или 0B), абсолютная величина которых не превосходит 3999, компилируются в программы для стекового калькулятора, , содержащие римские числа (в записи которых используются цифры I, V, X, L, C,

D и M).

23. Формулы, содержащие только записанные в шестнадцатеричной системе натуральные числа (запись которых обязана начинаться с 0x или 0X), абсолютная величина которых не превосходит 3999, компилируются в программы для стекового калькулятора, содержащие только записанные в восьмеричной системе счисления числа (запись которых обязана начинаться с нуля), и при этом приоритет операций сложения и вычитания выше, чем приоритет операций умножения и деления.
24. Формулы, содержащие только записанные в шестнадцатеричной системе натуральные числа (запись которых обязана начинаться с 0x или 0X), абсолютная величина которых не превосходит 3999, компилируются в программы для стекового калькулятора, содержащие только записанные в двоичной системе счисления числа (запись которых обязана начинаться с нуля), и при этом все операции являются правоассоциативными.
25. Формулы, содержащие только записанные в шестнадцатеричной системе натуральные числа (запись которых обязана начинаться с 0x или 0X), абсолютная величина которых не превосходит 3999, компилируются в программы для стекового калькулятора, , содержащие римские числа (в записи которых используются цифры I, V, X, L, C, D и M).
26. Формулы, содержащие только записанные в римской системе натуральные числа (с цифрами I, V, X, L, C, D и M), абсолютная величина которых не превосходит 3999, компилируются в программы для стекового калькулятора, содержащие только записанные в восьмеричной системе счисления числа (обязательно начинающиеся с нуля), и при этом приоритет вычитания больше приоритета сложения, но меньше приоритета умножения и деления.
27. Формулы, содержащие только записанные в римской системе натуральные числа (с цифрами I, V, X, L, C, D и M), абсолютная величина которых не превосходит 3999, компилируются в программы для стекового калькулятора, содержащие только записанные в двоичной системе счисления числа (обязательно начинающиеся с 0b или 0B), и при этом все операции трактуются как правоассоциативные.
28. Формулы, содержащие только записанные в римской системе натуральные числа (с цифрами I, V, X, L, C, D и M), абсолютная величина которых не превосходит 3999, компилируются в программы для стекового калькулятора, содержащие только записанные в шестнадцатеричной системе счисления числа (обязательно начинающиеся с 0x или 0X), и при этом приоритеты всех операций равны.
29. В предположении, что язык стекового калькулятора расширен операцией F, которая заменяет верхний элемент стека x на величину 5*x-1, компилировать формулы, содержащие «угловые скобки» <>, обозначающие замену находящейся в них величины x на величину 5*x-1.
30. В предположении, что язык стекового калькулятора расширен битовыми операциями | и &, компилировать формулы, содержащие эти операции.
31. В предположении, что язык стекового калькулятора расширен операциями L (left) и R (right), реализующими побитовый сдвиг влево и вправо соответственно, компилировать формулы, содержащие операции << и >>.
32. В предположении, что язык стекового калькулятора расширен унарной операцией @, изменяющей знак у верхнего элемента стека, компилировать формулы, содержащие унарный минус.
33. В предположении, что язык стекового калькулятора расширен правоассоциативной и имеющей максимальный приоритет операцией возведения в степень ^, компилировать формулы, содержащие эту операцию.

34. В предположении, что язык стекового калькулятора расширен правоассоциативной и имеющей максимальный приоритет операцией возведения в степень ^, компилировать формулы, содержащие эту операцию, обозначаемую символами **.
35. В предположении, что язык стекового калькулятора расширен операцией C (compare), которая извлекает два верхних операнда из стека и помещает обратно в стек результат сравнения извлечённых чисел (как <=> в языке Ruby), компилировать формулы, содержащие операцию сравнения <=>, имеющую минимальный приоритет.
36. В предположении, что язык стекового калькулятора расширен операцией D (duplicate), которая извлекает верхний элемент из стека и записывает его обратно в стек дважды, компилировать формулы, содержащие квадратные скобки [], обозначающие удвоение стоящего в них выражения.
37. В предположении, что язык стекового калькулятора расширен операцией D (duplicate), которая извлекает верхний элемент из стека и записывает его обратно в стек дважды, компилировать формулы, содержащие фигурные скобки {}, обозначающие возведение в квадрат стоящего в них выражения.
38. В предположении, что язык стекового калькулятора расширен унарной операцией A (abs), которая заменяет верхний элемент стека на его абсолютную величину, компилировать формулы, содержащие обозначаемую двумя вертикальными палочками операцию вычисления модуля (например, |a|).
39. Формулы, содержащие переменную a, значение которой следует считать равным нулю, компилируются в оптимизированные формулы, не содержащие лишних сложений и вычитаний.
40. Формулы, содержащие переменную b, значение которой следует считать равным единице, компилируются в оптимизированные формулы, не содержащие лишних умножений и делений.
41. Формулы, содержащие переменные a и b, значение которых следует считать равным двойке, компилируются в оптимизированные формулы, в которых умножение заменено сложением.
42. Результат работы компилятора — формула на входном языке, в которой каждое из выполняемых действий заключено в скобки.
43. Результат работы компилятора — дерево вывода исходной формулы.
44. При компиляции неправильной формулы выдаётся диагностика об ошибке и корректная часть исходной формулы.

Критерии оценки:

На выполнение лабораторной работы отводится 5 недель. Результаты выполнения лабораторной работы оцениваются по шкале от 0 до 10 баллов в зависимости от соответствия полученного решения поставленному заданию, качества алгоритмических подходов и программного кода, а также на основе краткой (не более 5 минут) защиты. Освоение компетенций зависит от результата написания проекта: 7-10 баллов - компетенции считаются освоенными на продвинутом уровне; 4-6 баллов - компетенции считаются освоенными на базовом уровне; 0-3 баллов - компетенции считаются не освоенными.

Задания для лабораторной работы по основам языка Ruby

Лабораторная работа состоит из решения трёх задач из указанного ниже списка.

Варианты заданий

1. Напишите программу, вычисляющую по введённому значению радиуса площадь круга
2. Напишите программу, находящую площадь поверхности и объём шара.
3. Напишите программу, находящую расстояние между двумя точками в пространстве.
4. Напишите программу, находящую корень линейного уравнения вида ax + b = 0
5. Напишите программу, вычисляющую максимальное и минимальное из трёх введённых целых разных чисел.
6. Напишите программу, вычисляющую среднее из трёх введённых целых разных чисел. Например, для 5 8 21 ответ — 8.
7. Напишите программу, решающую уравнение вида ax2 + bx + c = 0, для любых возможных значений a, b и c, введённых пользователем.

Кроме указанных задач оценочный фонд содержит ещё 43 различных задания.

Критерии оценки:

На выполнение лабораторной работы отводится 3 недели. Каждая задача оценивается по шкале от 0 до 5 баллов в зависимости от соответствия полученного решения поставленному заданию, качества алгоритмических подходов и программного кода, а также на основе краткой (не более 5 минут) защиты. Освоение компетенций зависит от результата написания работы: 11-15 баллов - компетенции считаются освоенными на продвинутом уровне; 7-10 баллов - компетенции считаются освоенными на базовом уровне; 0-6 баллов - компетенции считаются не освоенными.

Задания для лабораторной работы по созданию функций и методов

Лабораторная работа состоит из решения трёх задач из указанного ниже списка.

Варианты заданий

1. Напишите программу, вычисляющую факториал целого положительного числа.
2. Напишите программу, находящую минимальную и максимальную из цифр введённого целого неотрицательного числа.
3. Пользователь вводит n целых чисел (n известно заранее и его значение не менее 1). Напишите программу, находящую второе по величине среди данных чисел.
4. Напишите программу, вычисляющую для введённого целого положительного числа n сумму: $2^1 + 2^2 + 2^3 ... + 2^{n-1} + 2^n$. Операцию возведения в степень использовать нельзя.
5. Тернарная проблема Гольдбаха гласит, что любое нечётное число, начиная с 7, можно представить в виде суммы трёх простых чисел. Напишите программу, находящую такое разложение для введённого целого нечётного числа, большего или равного 7.
6. Напишите программу, отвечающую для двух введённых целых чисел на вопрос, являются ли они взаимно простыми (нет общего делителя за исключением единицы) или нет.
7. Напишите функцию, определяющую является ли данное шестизначное целое число счастливым или нет. Число будем называть счастливым, если среди его цифр можно так расставить знаки +-*/, что полученное арифметическое выражение давало результат 100. Пример 100000 — несчастливое, 254545 => 2*5+4*5+4*5=100 — счастливое.

Кроме указанных задач оценочный фонд содержит ещё 43 различных задания.

Критерии оценки:

На выполнение лабораторной работы отводится 3 недели. Каждая задача оценивается по шкале от 0 до 5 баллов в зависимости от соответствия полученного решения поставленному заданию, качества алгоритмических подходов и программного кода, а также на основе краткой (не более 5 минут) защиты. Освоение компетенций зависит от результата написания работы: 11-15 баллов - компетенции считаются освоенными на продвинутом уровне; 7-10 баллов - компетенции считаются освоенными на базовом уровне; 0-6 баллов - компетенции считаются не освоенными.

Задания для проекта «Выпуклая оболочка»

Задание лабораторной работы состоит в модификации эталонного проекта «Выпуклая оболочка», разработанного кафедрой информационных систем и технологий. Проект представляет собой программу индуктивно вычисляющую и визуализирующую выпуклую оболочку вводимого множества точек на плоскости.

Варианты заданий

Внесите изменения в проект, приводящие к указанным ниже изменениям функциональности.

1. Вычисляется среднее арифметическое значений функции sin(xy) в вершинах выпуклой оболочки.
2. Вычисляется максимальное значение функции sin(xy) в серединах рёбер выпуклой оболочки.
3. Выясняется, лежит ли заданная точка плоскости внутри выпуклой оболочки.
4. Вычисляется количество рёбер выпуклой оболочки, параллельных осям координат.
5. Вычисляется количество рёбер выпуклой оболочки, параллельных сторонам заданного треугольника.
6. Вычисляется минимальный из углов между любым из максимальных и произвольным из минимальных рёбер выпуклой оболочки.
7. Вычисляется среднее арифметическое расстояний от заданной точки до вершин выпуклой оболочки. Вычисляется сумма квадратов расстояний от заданной точки до вершин выпуклой оболочки.
8. Вычисляется сумма квадратов расстояний от начала координат до середин рёбер выпуклой оболочки.
9. Вычисляется количество пар вершин выпуклой оболочки, расстояние между которыми не превосходит единицу.
10. Вычисляется количество вершин выпуклой оболочки, расположенных внутри кольца $1<x^2+y^2<4$.
11. Находится минимальный стандартный прямоугольник, содержащий выпуклую оболочку (ограничивающий прямоугольник).
12. Находится максимальный стандартный прямоугольник, содержащийся в выпуклой оболочке.
13. Вычисляется радиус минимального круга с центром в заданной точке, содержащего выпуклую оболочку.
14. Вычисляется радиус максимального круга с центром в заданной точке, содержащегося в выпуклой оболочке.
15. Вычисляется длина минимальной диагонали выпуклой оболочки.
16. Вычисляется диаметр выпуклой оболочки; диаметром d(M) множества M называется точная верхняя граница расстояний между всевозможными точками множества.
17. Вычисляется мощность множества точек пересечения границы выпуклой оболочки с заданной прямой.
18. Вычисляется мощность множества точек пересечения границы выпуклой оболочки с заданным отрезком.
19. Вычисляется мощность множества точек пересечения границы выпуклой оболочки со сторонами заданного стандартного прямоугольника.
20. Вычисляется мощность множества точек пересечения границы выпуклой

оболочки со сторонами заданного треугольника.

21. Вычисляется площадь части выпуклой оболочки, расположенной в верхней полуплоскости.
22. Вычисляется периметр части выпуклой оболочки, расположенной в верхней полуплоскости.
23. Вычисляется площадь части выпуклой оболочки, расположенной в первом квадранте.
24. Вычисляется периметр части выпуклой оболочки, расположенной в первом квадранте.
25. Вычисляется площадь части выпуклой оболочки, расположенной внутри заданного стандартного прямоугольника.
26. Вычисляется периметр части выпуклой оболочки, расположенной внутри заданного стандартного прямоугольника.
27. Вычисляется площадь части выпуклой оболочки, расположенной внутри заданного треугольника.
28. Вычисляется периметр части выпуклой оболочки, расположенной внутри заданного треугольника.
29. Вычисляется количество вершин выпуклой оболочки, лежащих внутри заданного треугольника.
30. Вычисляется количество вершин выпуклой оболочки, лежащих вне заданного треугольника.
31. Вычисляется количество вершин выпуклой оболочки, лежащих в 1-окрестности заданной прямой.
32. Вычисляется количество вершин выпуклой оболочки, лежащих вне 1-окрестности заданной прямой.
33. Вычисляется количество рёбер выпуклой оболочки, целиком лежащих внутри заданного треугольника.
34. Вычисляется количество рёбер выпуклой оболочки, целиком лежащих вне заданного треугольника.
35. Вычисляется количество рёбер выпуклой оболочки, целиком лежащих в 1-окрестности заданной прямой.
36. Вычисляется количество рёбер выпуклой оболочки, целиком лежащих вне 1-окрестности заданной прямой.
37. Вычисляется угол, под которым выпуклая оболочка видна из начала координат.
38. Вычисляется угол, под которым видно из начала координат самое длинное ребро выпуклой оболочки.
39. Вычисляется количество всех острых внутренних углов выпуклой оболочки.
40. Вычисляется количество внутренних острых углов выпуклой оболочки, больших $\pi/4$.
41. Вычисляется сумма внутренних углов выпуклой оболочки, величина которых не превосходит $\pi/4$.
42. Вычисляется сумма углов, под которыми рёбра выпуклой оболочки пересекают заданную прямую.
43. Вычисляется количество рёбер выпуклой оболочки, целиком расположенных внутри квадрата с вершинами (0,0), (0,3), (3,0) и (3,3).
44. Вычисляется мощность множества пересечения границы выпуклой оболочки с полосой $-1<y<1$.
45. Вычисляется площадь части выпуклой оболочки, расположенной внутри кольца $1<x2+y2<4$.
46. Вычисляется периметр части выпуклой оболочки, расположенной внутри кольца $1<x2+y2<4$.

47. Вычисляется расстояние от выпуклой оболочки до заданной точки.
48. Вычисляется расстояние от выпуклой оболочки до заданной прямой.
49. Вычисляется расстояние от выпуклой оболочки до заданного отрезка.
50. Вычисляется расстояние от выпуклой оболочки до заданного треугольника.
51. Вычисляется расстояние от выпуклой оболочки до заданного стандартного прямоугольника.
52. Вычисляется количество пар рёбер выпуклой оболочки, расстояние между которыми не превосходит единицу.
53. Выясняется, лежит ли единичная окружность с центром в начале координат строго внутри выпуклой оболочки.
54. Вычисляется количество вершин выпуклой оболочки, расстояние от которых до квадрата с вершинами (0,0), (0,3), (3,0) и (3,3) не превосходит единицу.
55. Вычисляется мощность множества точек пересечения границы выпуклой оболочки с единичной окружностью с центром в начале координат.
56. Вычисляется мощность множества точек пересечения границы выпуклой оболочки с замкнутым единичным кругом с центром в начале координат.
57. Вычисляется площадь части выпуклой оболочки, расположенной внутри заданного круга.
58. Вычисляется периметр части выпуклой оболочки, расположенной внутри заданного круга.
59. Вычисляется количество вершин выпуклой оболочки, лежащих в 1-окрестности заданного отрезка.
60. Вычисляется количество вершин выпуклой оболочки, лежащих вне 1-окрестности заданного отрезка.
61. Вычисляется количество вершин выпуклой оболочки, лежащих в 1-окрестности заданного заполненного треугольника.
62. Вычисляется количество вершин выпуклой оболочки, лежащих вне 1-окрестности заданного заполненного треугольника.
63. Вычисляется количество рёбер выпуклой оболочки, целиком лежащих в 1-окрестности заданного отрезка.
64. Вычисляется количество рёбер выпуклой оболочки, целиком лежащих вне 1-окрестности заданного отрезка.
65. Вычисляется количество рёбер выпуклой оболочки, целиком лежащих в 1-окрестности заданного заполненного треугольника.
66. Вычисляется количество рёбер выпуклой оболочки, целиком лежащих вне 1-окрестности заданного заполненного треугольника.
67. Вычисляется сумма углов, под которыми рёбра выпуклой оболочки пересекают заданный отрезок.
68. Вычисляется сумма углов, под которыми рёбра выпуклой оболочки пересекают стороны заданного стандартного прямоугольника.
69. Вычисляется сумма углов, под которыми рёбра выпуклой оболочки пересекают стороны заданного треугольника.

Критерии оценки:

На выполнение лабораторной работы отводится 4 недели. Результаты выполнения лабораторной работы оцениваются по шкале от 0 до 10 баллов в зависимости от соответствия полученного решения поставленному заданию, качества алгоритмических подходов и программного кода, а также на основе краткой (не более 5 минут) защиты. Освоение компетенций зависит от результата написания проекта: 7-10 баллов - компетенции считаются освоенными на продвинутом уровне; 4-6 баллов - компетенции

считаются освоенными на базовом уровне; 0-3 баллов - компетенции считаются не освоенными.

Задания для лабораторной работы по L1 и L2-спискам

Лабораторная работа состоит из решения трёх задач из указанного ниже списка.

Варианты заданий

1. Дополните класс L1List методами поиска, добавления и удаления элемента по аналогии с тем, как это сделано выше для класса L2List. Приведите тесты, иллюстрирующие правильность построенной реализации.
2. Реализуйте множество на базе односвязного списка. Приведите тесты, иллюстрирующие правильность построенной реализации.
3. Реализуйте множество на базе двусвязного списка. Приведите тесты, иллюстрирующие правильность построенной реализации.
4. Дополните класс Tree методом draw изображения дерева, который должен рисовать дерево, используя графический интерфейс Tk.
5. Реализуйте массив на базе односвязного списка. Приведите тесты, иллюстрирующие правильность построенной реализации.
6. Реализуйте массив на базе двусвязного списка. Приведите тесты, иллюстрирующие правильность построенной реализации.
7. Реализуйте на базе класса L2List ассоциативный массив.
8. Реализуйте на базе класс L1List мультимножество.

Кроме указанных задач оценочный фонд содержит ещё 42 различных задания.

Критерии оценки:

На выполнение лабораторной работы отводится 6 недель. Каждая задача оценивается по шкале от 0 до 5 баллов в зависимости от соответствия полученного решения поставленному заданию, качества алгоритмических подходов и программного кода, а также на основе краткой (не более 5 минут) защиты. Освоение компетенций зависит от результата написания работы: 11-15 баллов - компетенции считаются освоенными на продвинутом уровне; 7-10 баллов - компетенции считаются освоенными на базовом уровне; 0-6 баллов - компетенции считаются не освоенными.

Комплект разноуровневых задач 4

по дисциплине «Программирование»

Уровень 1

Задача 1. Добавьте в реализацию стека метод для его печати на экран в произвольном формате.

Задача 2. Добавьте в реализацию множества метод для его печати на экран в произвольном формате.

Задача 3. Добавьте в реализацию очереди метод для поиска первого элемента, большего заданного.

Помимо указанных задач оценочный фонд содержит ещё 27 различных задания.

Уровень 2

Задача 1. Добавьте в реализацию стека метод для его сортировки любым доступным методом.

Задача 2. Добавьте в реализацию очереди метод для её сортировки любым доступным методом.

Задача 3. Добавьте в реализацию очереди метод для её перестановки в обратном порядке без использования каких-либо дополнительных структур данных.

Помимо указанных задач оценочный фонд содержит ещё 27 различных задания.

Уровень 3

Задача 1. Реализуйте n-стеков на базе вектора.

Задача 2. Реализуйте n-очередей на базе вектора.

Задача 3. Реализуйте стек и очередь на базе одного вектора.

Помимо указанных задач оценочный фонд содержит ещё 27 различных задания.

Критерии оценки:

- оценка «отлично» выставляется студенту, если решено не менее одной задачи каждого уровня;
- оценка «хорошо» выставляется студенту, если решено не менее одной задачи двух разных уровней;
- оценка «удовлетворительно» выставляется студенту, если решено не менее двух задач первого уровня или не менее одной задачи второго или третьего уровней;
- оценка «неудовлетворительно» выставляется студенту, если не решено ни одной задачи второго и третьего уровней и меньше двух задач первого уровня.

Приложение 2

Структура и содержание дисциплины «Программирование» по направлению подготовки 09.03.01 «Информатика и вычислительная техника».
(бакалавриат)

n/n	Раздел	Семестр	Неделя семестра	Виды учебной работы, включая самостоятельную работу студентов и трудоемкость в часах					Виды самостоятельной работы студентов					Формы аттестации	
				Л	П/С	Лаб	СРС	КСР	К.Р.	К.П.	РГР	Реферат	К/р	Э	З
	Первый семестр														
1.	Введение в Программирование.	1	1	1	2	1	3	+					+		+
2.	Язык Ruby. Синтаксис, основные типы данных, управляющие конструкции.	1	2	1	2	1	3	+					+		+
3.	Использование функции при написании программ. Встроенные библиотеки функций.	1	3	1	2	1	3	+					+		+
4.	Работа с коллекциями, математическими библиотеками и операциями ввода/вывода.	1	4	1	2	1	3	+					+		+
5.	Алгоритмы и их свойства. Реализации на языке Ruby.	1	5	1	2	1	3	+					+		+
6.	Особенности представления чисел в ЭВМ и связанные с этим проблемы.	1	6	1	2	1	3	+					+		+
7.	Оценки временной и емкостной сложностей алгоритмов.	1	7	1	2	1	4	+					+		+
8.	Базисные схемы обработки информации. Рекурсия.	1	8	1	2	1	4	+					+		+
9.	Итерация, взаимосвязь с рекурсией. Практическое применение методов построения и доказательства правильности рекурсивных программ.	1	9	1	2	1	3	+					+		+
10.	Индуктивные функции на пространстве последовательностей. Критерий индуктивности и схема вычисления индуктивной функции.	1	10	1	2	1	4	+					+		+
11.	Существование и единственность минимального индуктивного расширения, критерий минимальности и обобщенная схемы	1	11	1	2	1	4	+					+		+

	вычисления индуктивной функции. Решение практических задач с применением теории индуктивных функций.														
12.	Схема проектирования цикла при помощи инварианта. Доказательство правильности программ, написанных с помощью этой схемы.	1	12	1	2	1	3	+					+		+
13.	Основные методы построения инвариантов цикла и практическое применение этих методов для решения задач.	1	13	1	2	1	4	+					+		+
14.	Схема вычисления инвариантной функции и ее связь со схемой проектирования цикла при помощи инварианта. Решение задач с помощью схемы вычисления инвариантной функции.	1	14	1	2	1	3	+					+		+
15.	Компиляция и интерпретация Понятие о языках и грамматиках. Компиляция и интерпретация.	1	15	1	2	1	3	+					+		+
16.	Проект "Компилятор формул". Решение задач на модификацию.	1	16	1	2	1	4	+					+		+
17.	Регулярные выражения.	1	17	1	2	1	3	+					+		+
	Форма аттестации		**18-21**												+
	Всего часов по дисциплине в первом семестре			17	34	17	57						+		+
	Второй семестр												+		+
18.	Введение в ООП. Объекты и методы, типы и классы.	2	1		2	2	5	+					+		+
19.	Ruby, как язык ООП.	2	2		2	2	4	+					+		+
20.	Инкапсуляция.	2	3		2	2	4	+					+		+
21.	Наследование.	2	4		2	2	4	+					+		+
22.	Полиморфизм.	2	5		2	2	5	+					+		+
23.	Контейнерные типы в языке Ruby.	2	6		2	2	4	+					+		+
24.	Использование методов объектно-ориентированного программирования и теории индуктивных функций в проекте "Выпуклая оболочка". Решение задач на модификацию.	2	7		2	2	5	+					+		+
25.	Основы работы с графической информацией.	2	8		2	2	4	+					+		+
26.	Графическое библиотека GTK. Основы.	2	9		2	2	5	+					+		+
27.	Графическое отображение проекта "Выпуклая	2	10		2	2	5	+					+		+

	оболочка". Решение задач на модификацию.														
28.	Проект "Изображение проекции полиэдра". Решение задач на модификацию.	2	11		2	2	5	+					+		+
29.	Основы тестирование разработанных программ. Классификация тестирований по способам и назначению.	2	12		2	2	4	+					+		+
30.	Работа с классом Test::Unit в языке Ruby.	2	13		2	2	5	+					+		+
31.	Тестирование проекта "Компилятор формул".	2	14		2	2	4	+					+		+
32.	Тестирование проекта "Выпуклая оболочка".	2	15		2	2	5	+					+		+
33.	Тестирование проекта "Изображение проекции полиэдра".	2	16		2	2	4	+					+		+
34.	Модули в языке Ruby. Использование готовых и написание собственных.	2	17		2	2	4	+					+		+
	Форма аттестации		**18-21**												+
	Всего часов по дисциплине во втором семестре				34	34	76						+		+
	Третий семестр												+		+
35.	Контейнерные структуры данных. Понятие, основные виды, особенности. Понятие и организация вектора (массива), динамического вектора, стека, очереди, дека, множества, одно- и двусвязных списков.	3	1		2	2	4	+					+		+
36.	Непрерывная реализация структур данных на базе вектора. Ссылочная реализация структур данных на базе вектора.	3	2		2	2	4	+					+		+
37.	Язык С++, основы синтаксиса, простейшие программы.	3	3		2	2	4	+					+		+
38.	Реализации ограниченного стека на базе вектора. Реализации ограниченной очереди на базе вектора. Реализация ограниченного множества на базе вектора.	3	4		2	2	4	+					+		+
39.	Язык С++ - ввод/вывод. Управление памятью.	3	5		2	2	4	+					+		+
40.	Реализация двунаправленной очереди на базе вектора. Реализация списков.	3	6		2	2	4	+					+		+
41.	Язык С++ - математическая библиотека.	3	7		2	2	4	+					+		+
42.	Язык С++. Класс Vector.	3	8		2	2	4	+					+		+
43.	Сортировки. Основные понятия. Внутренние и внешние сортировки.	3	9		2	2	4	+					+		+

44.	Сортировка простыми включениями и анализ ее сложности.	3	10		2	2	4	+					+		+
45.	Сортировки простым выбором и простым обменом. Простейшие модификации пузырьковой сортировки. Сортировка слиянием.	3	11		2	2	5	+					+		+
46.	Язык С++. Создание собственных классов. Наследование.	3	12		2	2	4	+					+		+
47.	Алгоритм и анализ эффективности пирамидальной сортировки. Быстрая сортировка. Рекурсивная и итерационная версии.	3	13		2	2	4	+					+		+
48.	Язык С++. Шаблоны (Templates).	3	14		2	2	4	+					+		+
49.	Алгоритмы внешних сортировок.	3	15		2	2	6	+					+		+
50.	Система контроля версий Git. Коллективная работа. Простейший цикл внесения изменений.	3	16		2	2	6	+					+		+
51.	Продвинутая коллективная работа с Git. Создание веток, подготовка релиза, слияние веток, откат изменений.	3	17		2	2	6	+					+		+
	Форма аттестации		**18-21**												+
	Всего часов по дисциплине в третьем семестре				34	34	75						+		+
	Четвёртый семестр												+		+
52.	Введение в системное программирование Понятие о системном программировании. Сравнение прикладного и системного программирования. Предмет системного программирования.	4	1		2	2	4	+					+		+
53.	Язык С. Основы синтаксиса. Отличия от С++. Стандарт ANSI.	4	2		2	2	3	+					+		+
54.	Управление памятью в языке С. Malloc и free. Контроль утечек памяти.	4	3		2	2	3	+					+		+
55.	Дебаггинг программ. Методы профайлинга. Работа с strace.	4	4		2	2	3	+					+		+
56.	Знакомство с POSIX. Передача параметров через функцию main. Обработка опций командной строки. Обработка ошибок.	4	5		2	2	4	+					+		+
57.	Взаимодействие с файловой системой	4	6		2	2	3	+					+		+

	Понятие о файле. Атрибуты файла. Понятие о файловой системе. Соглашения о записи путей к файлам. Соглашения о правах доступа. "Жесткие" и "символьные" ссылки.														
58.	Операции над файлами: доступ, создание, удаление, перемещение, смена имени, смена владельца и прав доступа, изменение размера файла, изменение времени доступа, модификации и создания файла, получение информации о файле, позиционирование, чтение и запись. Работа с директориями: создание, удаление, обход.	4	7		2	2	4	+					+		+
59.	Процессы, группы, сессии, управляющие терминалы, сигналы и демоны Понятие о процессе, группе, сессии, управляющем терминале. Атрибуты процесса. Создание процессов, освобождение их ресурсов, ожидание дочерних процессов, запуск исполняемых файлов, изменение рабочей директории, привилегий процесса. Изменение группы процесса, сессии и управляющего терминала.	4	8		2	2	3	+					+		+
60.	Переключение между фоновыми процессами и процессами переднего плана. Получение информации о процессе. Понятие о сигналах. Назначение стандартных сигналов. Генерация, обработка и блокировка сигналов. Понятие о процессах-демонах.	4	9		2	2	4	+					+		+
61.	Методы взаимодействия между процессами Назначение методов взаимодействия между процессами. Сравнение методов из стандартов System V и POSIX. Каналы: безымянные и именованные. Дублирование файловых дескрипторов. Очереди сообщений: создание, обмен сообщениями, настройка, освобождение ресурсов.	4	10		2	2	4	+					+		+
62.	Разделяемая память: создание, подключение, настройка, освобождение ресурсов. Семафоры (безымянные и именованные): создание,	4	11		2	2	4	+					+		+

	операции увеличения и уменьшения, освобождение ресурсов. Реализация с помощью семафоров механизмов критических секций и барьеров.														
63.	Сокеты Понятие о сокетах. Типы, семейства и протоколы сокетов.	4	12		2	2	3	+					+		+
64.	Операции над сокетами: создание, удаление, прослушивание, подключение, подтверждение подключения, обмен данными. Получение адресов хоста и информации о них. Особенности работы с сокетами различных семейств и типов.	4	13		2	2	4	+					+		+
65.	Мультиплексированный ввод-вывод Блокирующие и неблокирующие операции. Проблемы при обработке множества потоков ввода-вывода одним процессом. Методы их разрешения. Метод мультиплексирования.	4	14		2	2	3	+					+		+
66.	Функции select и poll: назначение, сходства и различия. Трудности при использовании мультиплексирования. Перевод файла в неблокирующий режим ввода-вывода. Применение мультиплексирования для работы с сокетами.	4	15		2	2	4	+					+		+
67.	Нити Понятие о нитях. Атрибуты нити. Сравнение процессов и нитей. Создание и уничтожение нитей. Стек функций освобождения ресурсов нити. Присоединение и отсоединение нитей.	4	16		2	2	3	+					+		+
68.	Особенности использования сигналов и семафоров при работе с нитями. Синхронизация нитей, применение мьютексов и условных переменных. Персональные данные нитей.	4	17		2	2	4	+					+		+
	Форма аттестации		**18-21**												+
	Всего часов по дисциплине в четвёртом семестре				34	34	60								