Федеральное государственное автономное образовательное учреждение высшего образования
"Национальный исследовательский университет "Высшая школа экономики"

Факультет экономики, менеджмента и бизнес-информатики
Кафедра информационных технологий в бизнесе

Рабочая программа дисциплины
Теоретические основы информатики

для образовательной программы «Бизнес-информатика»
направления подготовки 38.03.05 «Бизнес-информатика»
уровень бакалавр

Разработчик программы:
Лядова Л.Н. , к.ф.-м.н., доцент, LLyadova@hse.ru

Одобрена на заседании кафедры информационных технологий в бизнесе
«____»_____2018 г.

И.о. зав. кафедрой
Е.Г. Плотникова_____

Утверждена Академическим советом образовательной программы «Бизнес-информатика» направления подготовки 38.03.05 Бизнес-информатика, образовательной программы «Программная инженерия» направления подготовки 09.03.04 Программная инженерия, образовательной программы «Информационная аналитика в управлении предприятием» направления подготовки 38.04.05 Бизнес-информатика.
«____»_____2018 г., № протокола _____

Академический руководитель образовательной программы
Л.В. Шестакова_____

Пермь, 2018

1. Область применения и нормативные ссылки

Настоящая программа учебной дисциплины устанавливает требования к образовательным результатам и результатам обучения студента и определяет содержание и виды учебных занятий и отчетности.

Программа предназначена для преподавателей, ведущих данную дисциплину, учебных ассистентов и студентов направления подготовки 38.03.05 Бизнес-информатика, изучающих дисциплину «Теоретические основы информатики» на первом курсе.

Программа разработана в соответствии с:

- Образовательным стандартом федерального государственного автономного образовательного учреждения высшего профессионального образования «Национальный исследовательский университет «Высшая школа экономики» по направлению подготовки 38.03.05 Бизнес-информатика (Квалификация: Бакалавр). Утверждён 26.12.2014 (протокол № 10).
- Образовательной программой «Бизнес-информатика» направления подготовки 38.03.05 «Бизнес-информатика» бакалавров.
- Объединенным учебным планом университета по образовательной программе «Бизнес-информатика» направления подготовки 38.03.05 «Бизнес-информатика», утвержденным в 2017 г.

2. Цели освоения дисциплины

Целью освоения дисциплины «Теоретические основы информатики» в соответствии с образовательным стандартом является подготовка в области основ математических и естественнонаучных знаний, обеспечивающих базу для освоения дисциплин высшего профессионально профилированного (на уровне бакалавра) образования, позволяющего выпускнику успешно работать в сфере проектирования архитектуры предприятия, стратегического планирования развития ИС и ИКТ управления предприятием, организации процессов жизненного цикла ИС и ИКТ управления предприятием, аналитической поддержки процессов принятия решений для управления предприятием, обладать универсальными и профессиональными компетенциями, способствующими его социальной мобильности и устойчивости на рынке труда.

Выпускник, освоивший программу дисциплины, должен быть готов решать следующие профессиональные задачи в области научно-исследовательской деятельности:

- Поиск, сбор, обработка, анализ и систематизация информации в экономике, управлении и ИКТ.
- Подготовка обзоров, отчетов и научных публикаций.

Кроме того, изучение дисциплины обеспечивает *базовую подготовку* для следующих видов профессиональной деятельности:

- *аналитическая* (анализ архитектуры предприятия; исследование и анализ рынка ИС и ИКТ; анализ и оценка применения ИС и ИКТ для управления бизнесом; анализ инноваций в экономике, управлении и ИКТ) – через выполнение проектов, предусматривающих необходимость анализа, оценки и выбора оптимальных средств при решении задач;
- *научно-исследовательская* (поиск, сбор, обработка, анализ и систематизация информации в экономике, управлении и ИКТ; подготовка обзоров, отчетов и научных публикаций) – через выполнение проектов, требующих осуществления поиска и анализа информации об использовании ИКТ, оформления отчётов;
- *консалтинговая* (консультирование по рациональному выбору ИС и ИКТ управления бизнесом; обучение и консультирование пользователей в процессе внедрения и эксплуатации ИС и ИКТ) – через взаимодействие с преподавателями и студентами в

процессе выполнения проектов, защиту выполненных проектов, оформление документации, публичные презентации проектов;
- *организационно-управленческая* (подготовка контрактов, оформление документации на разработку, приобретение или поставку ИС и ИКТ; взаимодействие со специалистами заказчика/исполнителя в процессе решения задач управления жизненным циклом ИТ-инфраструктуры предприятия; планирование и организация работы малых проектно-внедренческих групп) – через взаимодействие с преподавателями и студентами в процессе выполнения групповых проектов, их планирование, оформление документации и защиту выполненных проектов.

Успешное освоение дисциплины обеспечивает

- в *теоретической части*: фундаментальную подготовку по теоретическим основам информатики и программирования, необходимую для успешного освоения дисциплин профессионального цикла (базовых и вариативных частей), изучение которых связано с применением средств информационно-коммуникационных технологий, созданием эффективных алгоритмов решения задач, разработкой программного обеспечения для различных предметных областей;
- в *практической части*: изучение основ информатики, которые позволяют студентам приобрести навыки разработки и оценки алгоритмов и структур данных, получить представление о методах разработки программного обеспечения; формирование у студентов компетенций, связанных с базовыми понятиями жизненного цикла ПО и информационных технологий, позволяющих сделать процесс разработки программ более чётким и эффективным; получение навыков использования современных средств разработки, тестирования и отладки программ, их документирования.

3. Компетенции обучающегося, формируемые в результате освоения дисциплины

В соответствии с планируемыми результатами обучения по программе бакалавриата направления подготовки 38.03.05 «Бизнес-информатика», определяемыми образовательным стандартом НИУ ВШЭ, выпускник должен

Знать:

- основные методы, способы и средства получения, хранения, переработки информации.

Уметь:

- выявлять требования к информационной системе;
- анализировать требования к информационной системе;
- разрабатывать архитектуры информационной системы;
- организовывать взаимодействие с клиентами и партнерами в процессе решения задач управления жизненным циклом ИТ-инфраструктуры предприятия;
- готовить научно-технические отчеты, презентации, научные публикации по результатам выполненных исследований.

Владеть:

- навыками работы с компьютером как средством управления информацией;

Иметь опыт:

- оценки необходимых ресурсов для выполнения работ;
- разработки отчетов и документации;
- анализа результатов проведения работ;
- работы с различными видами исходных данных об информационных системах;
- использования терминологии, понятийного аппарата, базовых идей, методов и процессов предметной области заказчика;

- разработки документов в соответствии с требованиями государственных; отраслевых и корпоративных стандартов;
- владения офисными и общесистемными программными средствами;
- владения инструментарием обработки данных на персональном компьютере;
- анализа требований заказчика к информационным системам;
- применения формализованных языков и нотаций для построения моделей процессов, данных, объектов;
- применения специализированных программных средств для построения моделей процессов, данных, объектов;
- разработки технических заданий на выполнение работ;
- согласования технической документации.

Изучение дисциплины «Теоретические основы информатики» формирует базу для успешного формирования перечисленных результатов обучения. Основы необходимых компетенций закладываются при изучении данной дисциплины. Дисциплина ориентирована на формирование у студентов навыков логического и алгоритмического мышления при реализации решений поставленных задач, связанных с разработкой программ; понимания организации данных в информационных системах; навыков использования формальных языков; навыков работы с программным обеспечением при разработке программ и подготовке документов.

Уровни формирования компетенций:

РБ – ресурсная база, в основном теоретические и предметные основы (знания, умения);
СД – способы деятельности, составляющие практическое ядро данной компетенции;
МЦ – мотивационно-ценностная составляющая, отражает степень осознания ценности компетенции человеком и готовность ее использовать.

В результате освоения дисциплины студент осваивает *следующие компетенции*:

Компетенция	**Код по ОС НИУ ВШЭ**	**Уровень формирования компетенции**	**Дескрипторы – основные признаки освоения (показатели достижения результата)**	**Формы и методы обучения, способствующие формированию и развитию компетенции**	**Форма контроля уровня сформированности компетенции**
Способен решать проблемы в профессиональной деятельности на основе анализа и синтеза	УК-3	СД	Четко формулирует задачи, анализирует условия и обоснованно выбирает методы решения, уверенно интерпретирует результаты	Аудиторные занятия проводятся в форме, предполагающей активное участие студентов в работе, обсуждение проблем и анализ решений, предлагаемых студентами и преподавателем на лекциях и практических занятиях	Выполнение лабораторных работ в соответствии с методическими указаниями. Решение задач с обсуждением и анализом результатов и вариантов решения

Компетенция	Код по ОС НИУ ВШЭ	Уровень формирования компетенции	Дескрипторы – основные признаки освоения (показатели достижения результата)	Формы и методы обучения, способствующие формированию и развитию компетенции	Форма контроля уровня сформированности компетенции
		МЦ	Может выполнить декомпозицию сложных задач на подзадачи, использует это умение для разработки алгоритмов путём пошаговой детализации Может выполнить анализ возникающих при выполнении заданий проблем, искать решения, используя ранее полученный опыт, обобщая имеющиеся решения	Выполнение самостоятельных занятий разбивается на этапы, при выполнении каждого из которых необходимо выбрать и обосновать метод решения, используя материалы аудиторных занятий и материалы, предназначенные для самостоятельного изучения	Самостоятельное выполнение индивидуальных практических заданий с оформлением отчётов, включающих анализ задачи, её декомпозицию, обоснование методов решения, документирование каждого этапа решения
Способен критически оценивать и переосмыслять накопленный опыт (собственный и чужой), рефлексировать профессиональную и социальную деятельность	УК-9	МЦ	Демонстрирует способность самостоятельно определять формирующиеся дефициты знаний, умений и навыков в ходе обучения. Показывает умение сформулировать проблемы, связанные с недостатком знаний и навыков, и выбрать подходы к их решению. Демонстрирует способность применять полученные знания для решения новых задач в различных областях. Владеет навыками самостоятельного поиска, изучения и выбора методов и средств решения поставленных задач. Подготовлен к самостоятельному изучению новых технологий, инструментальных средств разработки программ	Самостоятельное изучение отдельных тем. Выполнение лабораторных работ с анализом и обсуждением результатов. Выполнение заданий с постепенным наращиванием требований к сложности, используемым методам и средствам решения, применение которых требует накапливать и переосмысливать опыт, полученные ранее знания, навыки решения задач	Выполнение письменных проверочных работ с разбором и анализом результатов. Выполнение индивидуальных практических заданий с оформлением отчётов, включающих анализ существующих сторонних решений, а также анализ ошибок, допущенных при выполнении заданий

Компетенция	Код по ОС НИУ ВШЭ	Уровень формирования компетенции	Дескрипторы – основные признаки освоения (показатели достижения результата)	Формы и методы обучения, способствующие формированию и развитию компетенции	Форма контроля уровня сформированности компетенции
Способен работать с информацией: находить, оценивать и использовать информацию из различных источников, необходимую для решения научных и профессиональных задач (в том числе на основе системного подхода)	УК-5	РБ	Показывает навыки уверенного владения средствами поиска информации в Internet, в различных источниках, рекомендованных для самостоятельного изучения	Самостоятельное изучение отдельных тем при подготовке к контрольным мероприятиям	Экспресс-опросы по материалу. Оставленному для самостоятельного изучения. Тестирование по отдельным темам, предполагающим самостоятельное изучение материала
		СД	Демонстрирует умение оценивать и отбирать наиболее важную информацию, максимально полезную для решения поставленных задач при выполнении домашних заданий, при подготовке к контрольным мероприятиям	Выполнение индивидуальных заданий, требующее самостоятельно находить информацию, существующие решения и оценивать их. Изучение возможностей инструментальных средств с использованием справочных систем, электронных библиотек, доступных на сайтах фирм-производителей, анализ и обоснование выбора средств при выполнении заданий и при подготовке отчетов	Выполнение индивидуальных практических заданий с оформлением отчётов, включающих анализ существующих сторонних решений, а также анализ различных вариантов решений, обеспечиваемых применяемыми инструментальными средствами, с указанием использованных источников информации и сравнением результатов
			Владеет навыками самостоятельного поиска, изучения и выбора методов и средств решения поставленных задач		
			Подготовлен к самостоятельному изучению новых технологий, инструментальных средств разработки программ		

Компетенция	Код по ОС НИУ ВШЭ	Уровень формирования компетенции	Дескрипторы – основные признаки освоения (показатели достижения результата)	Формы и методы обучения, способствующие формированию и развитию компетенции	Форма контроля уровня сформированности компетенции
Способен обрабатывать, анализировать и систематизировать информацию по теме исследования, используя соответствующий математический аппарат и инструментальные средства	ПК-31	РБ	Уверенно использует способы формального описания структур данных и алгоритмов их обработки Владеет различными способами формального описания языков, умеет дать описания языковых конструкций, может прочитать и использовать на практике формальные описания языков с использованием различных нотаций Знает и может использовать на практике математический аппарат, формальные средства, лежащие в основе различных методов разработки алгоритмов и программ Знает основы организации файлов: - владеет средствами выполнения операций над файлами с различной организацией, - умеет выбрать оптимальные способы представления данных и использовать эффективные средства их обработки при решении задач обработки массивов данных, хранящихся во внешней памяти	Использование и сравнение формальных средств при изучении основных методов разработки программ и средств языка C# в MS VS.NET. Получение формальных оценок алгоритмов и сравнение их с результатами, полученными при их практической реализации при выполнении заданий. Выполнение практических заданий с использованием языка C# на аудиторных практических занятиях.	Выполнение письменных проверочных работ. Выборочный опрос и экспресс-тестирование на лекциях и практических занятиях. Тестирование по отдельным темам. Оформление отчётов о выполнении индивидуальных практических заданий с включением в них формальных описаний алгоритмов и структур данных, анализа эффективности решений с использованием математического аппарата

Компетенция	Код по ОС НИУ ВШЭ	Уровень формирования компетенции	Дескрипторы – основные признаки освоения (показатели достижения результата)	Формы и методы обучения, способствующие формированию и развитию компетенции	Форма контроля уровня сформированности компетенции
			Знает основы методов трансляции и умеет их использовать при разработке программ: - использует знания при выборе оптимальных структур данных и управляющих структур; - способен разработать простейший интерпретатор		
Способен готовить научно-технические отчеты, презентации, научные публикации по результатам выполненных исследований	ПК-32	СД	Умеет грамотно оформлять отчеты о выполнении заданий, включающие постановку задач, описание решений и оценки результатов. Владеет навыками грамотного оформления и документирования текстов программ, результатов их тестирования. Показывает навыки оформления презентаций и публичной защиты предлагаемых решений	Выполнение лабораторных работ для изучения правил и средств подготовки отчётов и презентаций с использованием приложений пакета MS Office. Самостоятельное выполнение индивидуальных заданий и групповых проектов	Оформление результатов выполнения заданий с использованием требований к оформлению отчетов, современных технологий подготовки документов Защита результатов выполнения индивидуальных заданий и проектов

В результате освоения дисциплины студент должен:

- *Знать*:
  - основные методы, способы и средства получения, хранения, переработки информации:
    - понятие типа данных, форматы представления данных при решении задач с помощью компьютера, а также средства конструирования новых типов на основе стандартных типов, используемых в языках программирования;
    - основы алгоритмизации: определение, свойства и средства формализации алгоритмов, методы исследования их свойств, оценки эффективности; основные управляющие структуры и способы описания алгоритмов с использованием различных нотаций; основные методы разработки алгоритмов, особенности их реализации;
    - особенности разработки и реализации параллельных и распределённых алгоритмов (проблема взаимного исключения и методы её решения, проблема тупика и подходы к решению);

  - различные методы разработки алгоритмов, конструирования программ для выбора наиболее подходящих алгоритмов и средств их реализации в зависимости от постановки задачи;
- фазы жизненного цикла информационных систем;

- *Уметь*:
  - выявлять и анализировать требования к информационной системе:
    - анализировать постановку задач, требования к их решению и планировать их выполнение;
    - рассматривать и анализировать варианты решений в зависимости от предлагаемых условий, требований к результатам;
  - готовить научно-технические отчеты, презентации, научные публикации по результатам выполненных исследований:
    - оформлять отчеты о выполнении заданий (реализации проектов) в соответствии с требованиями стандартов с помощью MS Word;
    - готовить презентации выполненных проектов с помощью MS Power Point и публично их представлять.

- *Иметь навыки* (*приобрести опыт*):
  - работы с компьютером как средством управления информацией:
    - использования компьютера для организации взаимодействия при выполнении проектов, для получения заданий и передачи отчетов о их выполнении, для текущего контроля;
  - разработки отчетов и документации, разработки документов в соответствии с требованиями государственных, отраслевых и корпоративных стандартов:
    - оформлять отчеты о выполнении заданий в соответствии с заданными требованиями;
    - документировать разрабатываемые программы, основываясь на стандартах;
  - работы с различными видами исходных данных об информационных системах:
    - изучения исходных данных для выполнения заданий, анализ задач с использованием данных из различных источников и в разных форматах;
    - самостоятельного поиска информации, необходимой для выполнения заданий;
    - работы со справочными системами, электронными библиотеками документации по используемым программным продуктам;
  - использования терминологии, понятийного аппарата, базовых идей, методов и процессов предметной области заказчика:
    - анализировать постановку задач, приведённую в терминологии соответствующей предметной области, требования к выполнению заданий;
    - изучать описания предлагаемых методов решения задач с использованием понятийного аппарата, принятого в соответствующей области;
    - применять при разработке программ и оформлении отчетов терминологию, принятую в соответствующей предметной области;
  - владения офисными и общесистемными программными средствами:
    - настройки операционной среды, установки и настройки программных, инструментальных средств в соответствии с потребностями решения поставленных задач;
    - оформления документов с использованием возможностей текстового процессора MS Word в соответствии с заданными требованиями, стандартами;
    - подготовки презентаций с использованием MS Power Point;
  - владения инструментарием обработки данных на персональном компьютере:
    - оформления тестов и результатов тестирования с помощью MS Excel, их анализа;

- оформления результатов оценки алгоритмов, сравнения теоретических оценок и оценок, полученных при программной реализации алгоритмов, их анализ с помощью MS Excel;

– анализа требований заказчика к информационным системам:
  - анализа задач при выполнении домашних заданий, контрольных работ и выполнении проектов;
– применения формализованных языков и нотаций для построения моделей процессов, данных, объектов:
  - использования теории графов в программировании (при разработке алгоритмов и программ и их анализе);
  - использования различных способов формального описания языков программирования (с помощью диаграмм – диаграмм Вирта, металигвистических формул – БНФ, РБНФ);
  - применения диаграмм для описания алгоритмов (машин Тьюринга);
– применения специализированных программных средств для построения моделей процессов, данных, объектов:
  - применения графических редакторов для описания блок-схем алгоритмов, построения диаграмм;
– работы с различными видами исходных данных в предметной области;
  - работа с файлами в различных форматах при выполнении заданий;
– разработки технических заданий на выполнение работ:
  - оформление отчетов о выполнении заданий, включающих описание требований к разрабатываемым программам;
– разработки программ средней сложности с использованием возможностей современных систем программирования, средств тестирования и отладки, документирования.

4. Место дисциплины в структуре образовательной программы

Настоящая дисциплина относится к циклу профессиональных дисциплин (Major – базовая часть).

Изучение данной дисциплины базируется на следующих дисциплинах:
- Базовый школьный курс информатики и/или выравнивающий факультатив по основам программирования.
- Школьные курсы алгебры и геометрии.
- Алгебра и геометрия (изучается параллельно с данным курсом).
- Дискретная математика (изучается параллельно с данным курсом).
- Программирование (изучается параллельно с данным курсом).

Для освоения учебной дисциплины студенты должны владеть следующими знаниями и компетенциями (вопросы и задачи входного контроля – в Приложении 1):
- Знание основ функционирования персональных компьютеров.
- Знание основ организации обработки данных с помощью компьютеров.
- Базовые навыки работы с персональным компьютером в среде Microsoft Windows.
- Базовые знания и навыки работы с офисными приложениями (текстовым процессором и электронными таблицами).
- Знание основных понятий линейной алгебры, операций над векторами и матрицами.

Основные положения дисциплины будут использованы в дальнейшем при изучении следующих дисциплин учебного плана:
- Программирование (изучается параллельно с данным курсом).
- Дискретная математика (изучается параллельно с данным курсом).

- Информационные процессы, системы и сети.
- Управление данными.
- Моделирование процессов и систем.
- Анализ и совершенствование бизнес-процессов.
- Корпоративные информационные системы.
- Архитектура предприятия.
- Технико-экономическое обоснование ИТ-проектов.
- Введение в машинное обучение.
- Автоматизация учета на предприятии.
- Стратегии электронного бизнеса.
- Семантические информационные системы.
- Технологии анализа данных в Internet.
- Web-программирование.
- Интеллектуальные системы.

Кроме того, полученные знания, умения и навыки будут использованы в дальнейшем при прохождении учебной, производственной и преддипломной практик, при выполнении курсовых и выпускных квалификационных работ.

5. Тематический план учебной дисциплины

№	Название раздела	Всего часов	Аудиторные часы			Самостоятельная работа
			Лекции	Семинары	Практические занятия	
	Раздел 1. Информатика и предмет её исследования	**10**	**2**	**0**	**0**	**8**
1	Основные понятия	6	2	0	0	4
2	Информатика как научная дисциплина	4	0	0	0	4
	Раздел 2. Кодирование информации и представление данных в памяти компьютера	**36**	**6**	**0**	**6**	**24**
3	Понятие системы счисления, связь между системами счисления	12	2	0	2	8
4	Понятие типа данных и представление данных в памяти компьютера	12	2	0	2	8
5	Конструирование типов, рекурсивные типы данных	12	2	0	2	8
	Раздел 3. Основы алгоритмизации и программирования	**88**	**10**	**0**	**16**	**62**
6	Понятие и свойства алгоритма. Способы записи алгоритмов	12	2	0	2	8
7	Машины Тьюринга	18	2	0	4	12
8	Нормальные алгорифмы Маркова	16	2	0	2	12
9	Вычислимые функции и методы разработки алгоритмов	18	2	0	4	12
10	Рекурсия и итерация, особенности реализации	12	0	0	2	10
11	Понятие сложности алгоритма, оценка сложности и классы сложности задач	12	2	0	2	8
	Раздел 4. Программы и языки программирования	**52**	**6**	**0**	**10**	**36**
12	Понятие программы и жизненный цикл программ	6	2	0	0	4

№	Название раздела	Всего часов	Аудиторные часы			Самостоятельная работа
			Лекции	Семинары	Практические занятия	
13	Формальные грамматики и определения языка программирования, способы описания языков	18	2	0	4	12
14	Основы трансляции программ	28	2	0	6	20
	Раздел 5. Сортировка и поиск	**64**	**8**	**0**	**12**	**44**
15	Задача сортировки и сортировка массивов	16	2	0	2	12
16	Файлы и файловые системы, внешняя сортировка	16	2	0	2	12
17	Поиск информации, хеширование	18	2	0	4	12
18	Поиск данных во внешней памяти	14	2	0	4	8
	Раздел 6. Теоретические основы разработки распределённых и параллельных систем	**16**	**4**	**0**	**0**	**12**
19	Проблема взаимного исключения	8	2	0	0	6
20	Проблема тупика и её решение	8	2	0	0	6
	Всего:	***266***	***36***	***0***	***44***	***186***

6. Формы контроля знаний студентов

Тип контроля	Форма контроля	1 год				Параметры
		1	**2**	**3**	**4**	
Текущий (неделя)	Лабораторная работа 1	5				Изучение типов данных и способов кодирования информации в языках программирования
	Лабораторная работа 2	7				Изучение способов записи алгоритмов и реализации управляющих конструкций в языках программирования
	Контрольная работа 1	10				Письменная работа (40 минут) по темам 3-6
	Контрольная работа 2		13			Письменная работа (40 минут) по темам 7, 8
	Контрольная работа 3		17			Письменная работа (40 минут) по темам 9-11
	Контрольная работа 4			21		Письменная работа (40 минут) по темам 13-14
	Лабораторная работа 3			22		Изучение алгоритмов сортировки и анализ сложности – разработка приложений на языке C#
	Лабораторная работа 4			23		Работа со структурами данных, сортировка записей – разработка приложений на языке C#
	Лабораторная работа 5			25		Работа файлами – разработка приложений на языке C# с оформлением отчёта о выполнении
	Лабораторная работа 6			27		Изучение методов хеширования – разработка приложений на языке C#
	Лабораторная работа 7			29		Изучение рекурсивных типов данных и работа с динамикой – разработка приложений на языке C#
	Групповой проект			30		Разработка интерпретаторов MT и HAM
	Самостоятельная работа			31		Индивидуальное практическое задание – разработка программ на языке C#
Итоговый	Экзамен			*		Письменный экзамен (90 минут)

7. Критерии оценки знаний, навыков

Текущий контроль предусматривает выполнение *контрольных* (*проверочных*) *работ*, *лабораторных работ* и *индивидуальных практических заданий* (*самостоятельных работ*).

Оценки по всем формам текущего контроля выставляются по 10-балльной шкале.

Письменные контрольные (проверочные) *работы* выполняются по нескольким темам трёх разделов курса:

- Кодирование информации и представление данных в памяти компьютера:
  - Понятие системы счисления, связь между системами счисления.
  - Понятие типа данных и представление данных в памяти компьютера.
- Основы алгоритмизации и программирования:
  - Понятие и свойства алгоритма. Способы записи алгоритмов.
  - Машины Тьюринга.
  - Нормальные алгорифмы Маркова.
  - Вычислимые функции и методы разработки алгоритмов.
  - Рекурсия и итерация, особенности реализации.
  - Понятие сложности алгоритма, оценка сложности и классы сложности задач.
- Основы алгоритмизации и программирования:
  - Формальные грамматики и определения языка программирования, способы описания языков
  - Основы трансляции программ.

Цель проведения контрольных (проверочных) работ – проверка формирования следующих компетенций:

- Способен обрабатывать, анализировать и систематизировать информацию по теме исследования, используя соответствующий математический аппарат и инструментальные средства (ПК-31).

В ходе выполнения контрольной работы студент должен показать, что он

- Уверенно использует способы формального описания структур данных и алгоритмов их обработки:
  - Владеет формальными методами описания алгоритмов и структур данных.
  - Знает способы описания алгоритмов и может читать предложенные алгоритмы, анализировать их и давать описания с использованием других способов.
  - Знает понятие типа данных и характеристики различных типов.
  - Знает алгоритмы перевода чисел из одной системы счисления в другую.
  - Знает форматы представления данных в памяти компьютера.
  - Может записать данные во внутреннем представлении и интерпретировать коды, записанные в памяти компьютера.
- Владеет различными способами формального описания языков, умеет дать описания языковых конструкций, может прочитать и использовать на практике формальные описания языков с использованием различных нотаций.
- Знает и может использовать на практике математический аппарат, формальные средства, лежащие в основе различных методов разработки алгоритмов и программ.

Контрольные (проверочные) работы включают задания по всем перечисленным выше темам курса.

Все задания имеют одинаковый вес.

Примеры заданий для подготовки к контрольной работе приведены в Приложении 2.

Критерии оценки выполнения заданий:

Характеристика решения	Оценка
Приведено полное решение, приведено его объяснение, обоснование выбранного варианта при наличии нескольких вариантов решений. Формулировки точные, показано знание терминологии, основных понятий, изучаемых в рамках рассматриваемых тем, связи между ними	10
Приведено полное решение, приведено его объяснение. Имеются незначительные замечания по формулировкам, не рассматриваются (при их наличии) все возможные варианты решений, нет их анализа и сравнения (или анализ неполон)	9
Приведено полное решение, но отсутствует его объяснение, сравнение различных вариантов (при их существовании)	8
Приведено полное решение, но имеются неточности в формулировках или незначительные ошибки / Решение неполное, но сужает постановку задачи	6-7
Выбран верный подход к решению, но приведено неполное решение, в формулировках имеются недочеты, допущены отдельные существенные ошибки	4-5
Выбран верный метод, но в решении имеются существенные ошибки	3
Выбран неверный метод	2
Решение не соответствует постановке задачи	1
Решение отсутствует	0

Предусматривается *возможность «защиты»* выполненных письменных работ (апелляции), если приведённое решение допускает неоднозначность оценки. Защита предусматривает:

- Объяснение студентом всех использованных в решении методов, конструкций, их оценку и обоснование.
- Рассмотрение и анализ альтернативных решений.
- Внесение предложенных преподавателем изменений в условия и пояснение, какие изменения это вызовет в приведенном студентом решении.

В ходе защиты студент должен продемонстрировать знание профессиональной терминологии в рамках соответствующей темы, продемонстрировать знание теоретического материала и умение на практике применять эти знания при решении задач, а также владение математическим аппаратом для получения формального описания алгоритмов, доказательства их свойств и получения оценки эффективности решений. Кроме того, он должен показать, что владеет культурой мышления, способен к обобщению, анализу, восприятию информации, постановке цели и выбору путей ее достижения, способен логически верно, аргументированно и ясно строить речь.

Вес контрольных (проверочных) работ точки в определении итоговой оценки показан ниже.

Лабораторные работы выполняются на практических занятиях. По результатам выполнения оформляются отчёты (студенты оформляют отчёты в часы, отведённые для самостоятельной работы), результаты выполнения проверяются и оцениваются преподавателем на практических занятиях.

Лабораторные работы охватывают следующие темы:

- Понятие типа данных и представление данных в памяти компьютера.
- Понятие и свойства алгоритма. Способы записи алгоритмов.
- Задача сортировки и сортировка массивов. Понятие сложности алгоритма, оценка сложности.
- Задача сортировки и сортировка массивов. Конструирование типов.
- Задача сортировки. Конструирование типов. Работа с файлами.

- Поиск информации. Хеширование. Использование динамических структур данных.
- Формальные языки и грамматики, основы трансляции. Использование динамических структур данных.

Примерные требования к их выполнению заданий приведены в Приложении 5.

Выполнение *лабораторных работ* предусматривает: разработку алгоритма (пошаговую детализацию), проектирование программы, кодирование и документирование, тестирование и отладку программ.

По результатам выполнения оформляются отчёты, включающие описания алгоритмов с использованием различных нотаций, самодокументированный текст программы, разработанной студентом, а также набор тестов. Кроме того, отчёт включает анализ ошибок, допущенных при выполнении задания (статистику допущенных синтаксических и семантических ошибок, ошибок в алгоритмах и пр.).

Вес работ в определении оценки показан ниже.

Цель выполнения – проверка формирования следующих компетенций:

- Способен обрабатывать, анализировать и систематизировать информацию по теме исследования, используя соответствующий математический аппарат и инструментальные средства (ПК-31): использование терминологии, понятийного аппарата, базовых идей, методов и процессов предметной области задачи; анализ требований, постановки задачи, её формализации; знание основ организации файлов: владеет средствами выполнения операций над файлами с различной организацией, умеет выбрать оптимальные способы представления данных и использовать эффективные средства их обработки при решении задач обработки массивов данных, хранящихся во внешней памяти.
- Способен решать проблемы в профессиональной деятельности на основе анализа и синтеза (УК-3): анализ задач и разбиение их на подзадачи, разработка алгоритма решения задачи путем пошаговой детализации; анализ требований к исходным данным и результатам и конструирование типов данных на основе требований;
- Способен готовить научно-технические отчеты, презентации, научные публикации по результатам выполненных исследований (ПК-32) – подготовка отчета о выполнении домашнего задания и документирование программы.

В ходе выполнения заданий студент должен показать, что он

- Владеет различными методами разработки алгоритмов и программ.
- Умеет формализовать описание поставленных задач и алгоритмов решения с использованием формальных средств и имеющегося программного обеспечения.
- Способен обосновать выбор алгоритмов, структур данных и программных средств для их реализации.
- Демонстрирует знание и умение оценивать и применять основные методы разработки алгоритмов и программ.
- Владеет средствами кодирования и отладки программ.
- Владеет навыками поиска и использования информации, необходимой для выполнения заданий (поиск описаний алгоритмов, методов их оценки и пр.), из различных источников.
- Умеет самостоятельно работать со справочной информацией, руководствами, владеет знаниями, достаточными для самостоятельного изучения и понимания описаний алгоритмов и программ.
- Способен обосновывать предлагаемые решения (не только разрабатывать алгоритмы и программы, реализующие их, но и уметь доказывать свойства алгоритмов и программ, анализировать и оценивать эффективность решений).

- Владеет навыками грамотного оформления и документирования текстов программ, результатов их тестирования.
- Умеет грамотно оформлять отчёты о выполнении домашних заданий, включающие постановку задач, описание решений и оценки результатов.

Предусматривается возможность «защиты» выполненных работ. Защита предусматривает:

- Объяснение студентом использованных методов решения, реализованных в программе алгоритмов.
- Объяснение студентом всех использованных в программном коде конструкций, типов данных и управляющих структур.
- Рассмотрение и анализ возможных альтернативных решений.
- Внесение предложенных преподавателем изменений в код программы и анализ последствий этих изменений, их результатов.

В ходе защиты студент должен продемонстрировать знание профессиональной терминологии в рамках соответствующей темы, продемонстрировать знание теоретического материала по теме, а также умение оценивать эффективность решений. Кроме того, он должен показать, что владеет культурой мышления, способен к обобщению, анализу, восприятию информации, постановке цели и выбору путей ее достижения, способен логически верно, аргументировано и ясно строить речь.

Примерный перечень *тем заданий, выполняемых на практических занятиях*:

- Понятие системы счисления, связь между системами счисления и алгоритмы перевода чисел из одной системы счисления в другую.
- Понятие типа данных и представление данных различных типов в памяти компьютера.
- Свойства алгоритмов и способы записи алгоритмов.
- Программы машин Тьюринга.
- Нормальные алгорифмы Маркова.
- Вычислимые функции и методы разработки алгоритмов.
- Рекурсия и итерация, особенности реализации, сравнение.
- Понятие сложности алгоритма, оценка сложности и классы сложности задач.
- Формальные грамматики и определения языка программирования, способы описания языков.
- Этапы трансляции программ, процедуры синтаксического разбора.
- Методы сортировки, анализ и реализация алгоритмов сортировки массивов.
- Методы внешней сортировки, анализ и реализация алгоритмов сортировки файлов.
- Поиск и хеширование: сравнение и реализация методов.
- Проблема взаимного исключения.
- Проблема тупика и её решение.

При определении *оценки за аудиторную работу* учитываются результаты выполнения лабораторных работ (примерные задания по темам приведены в приложениях), а также результаты «самотестирования» с использованием LMS и полученные на практических занятиях оценки.

Работа на практических занятиях предполагает самостоятельное выполнение заданий (в том числе и групповых проектов с публичной презентацией результатов (Приложение 6)).

Предполагается *самостоятельное выполнение заданий (решение задач) по темам*:

- Сортировка и поиск.
- Машины Тьюринга.
- Нормальные алгорифмы Маркова.

Выполнение заданий способствует формированию компетенций:

- Способность работать в команде (УК-7).
- Способность готовить научно-технические отчеты, презентации, научные публикации по результатам выполненных исследований (ПК-32).

При разработке программ закрепляются навыки

- разработки алгоритмов и их описания с помощью различных нотаций;
- разработки тестов;
- разработки программ в среде VS.NET (язык C#), работы с различными типами данных (числами, строками), со структурами данных (одно- и двумерными массивами, структурами, динамическими структурами данных – списками и деревьями), со средствами тестирования и отладки программ;
- документирования программ.

Самостоятельная работа (*индивидуальное практическое задание*) предусматривают разработку приложения на языке C#, обобщающего знания и навыки, полученные при выполнении лабораторных работ (это итоговое задание по дисциплине) и оформление отчётов в соответствии с требованиями, приведёнными ниже.

По результатам выполнения задания оформляется отчёт. Требования к выполнению приведены в Приложении 4. Вес контрольной точки в определении оценки показан ниже.

Цель выполнения задания – проверка формирования следующих компетенций:

- Способен обрабатывать, анализировать и систематизировать информацию по теме исследования, используя соответствующий математический аппарат и инструментальные средства (ПК-31): использование терминологии, понятийного аппарата, базовых идей, методов и процессов предметной области задачи; анализ требований, постановки задачи, её формализации; применение формализованных языков и нотаций для построения моделей данных и описания алгоритмов.
- Способен решать проблемы в профессиональной деятельности на основе анализа и синтеза (УК-3): анализ задач и разбиение их на подзадачи, разработка алгоритма решения задачи путем пошаговой детализации; анализ требований к исходным данным и результатам и конструирование типов данных на основе требований;
- Способен готовить научно-технические отчеты, презентации, научные публикации по результатам выполненных исследований (ПК-32) – подготовка отчета о выполнении домашнего задания и документирование программы.

В ходе задания студент должен показать, что он

- Владеет формальными методами описания алгоритмов, умеет формализовать описание поставленных задач и алгоритмов решения с использованием различных формальных средств и имеющегося программного обеспечения.
- Демонстрирует знание и умение оценивать и применять основные методы разработки алгоритмов и программ:
  - знает и может использовать на практике основные управляющие структуры и способы описания алгоритмов с использованием различных нотаций;
  - может применить на практике при разработке программ знания о представлении и кодировании информации.
- Знает понятие типа данных, определения и особенности реализации различных типов данных в разных системах программирования:
  - знает правила описания данных, используемые типы данных, применяет знания на практике;
  - знает и может использовать операции и стандартные функции для работы с данными различных типов.

- Владеет навыками поиска и использования информации, необходимой для выполнения заданий (поиск описаний алгоритмов, методов их оценки и пр.), из различных источников.
- Умеет самостоятельно работать со справочной информацией, руководствами, владеет знаниями, достаточными для самостоятельного изучения и понимания описаний алгоритмов и программ.
- Умеет грамотно оформлять отчёты о выполнении домашних заданий, лабораторных работ, включающие постановку задач, описание решений и оценки результатов.
- Владеет навыками грамотного оформления и документирования текстов программ, результатов их тестирования.

Оценки определяются в соответствии с «Положением об организации промежуточной аттестации и текущего контроля успеваемости студентов НИУ ВШЭ», утвержденным приказом НИУ ВШЭ от 19.08.2014 №6.18.1-01/1908-02.

Формы и сроки проведения контрольных мероприятий определяются учебным планом и графиком учебного процесса.

Итоговый контроль (*экзамен*) проводится в форме письменной работы.

Итоговый экзамен включает теоретические вопросы и решение задач.

Теоретические вопросы включают вопросы по темам всего курса.

При проведении экзамена выполняются практические задания по всем темам курса, которые изучаются в соответствии с программой курса (Приложение 2).

Примерный перечень вопросов и заданий для подготовки к экзамену по дисциплине приведён в приложениях (Приложение 3).

Все вопросы и задания, включённые в экзаменационные билеты, оцениваются по 10-балльной шкале. Вес *оценки за теоретический вопрос* в экзаменационной оценке составляет 30%, вес оценок за выполнение *практических заданий* (решение задач) – 70%.

Критерии оценки заданий письменной работы приведены ниже.

Критерии оценки выполнения практических заданий:

Характеристика решения	**Оценка**
Приведено полное и точное решение, приведено его объяснение, обоснование выбранного варианта при наличии нескольких вариантов решений	10
Приведено полное и точное решение, приведено его объяснение, но отсутствует обоснование	9
Приведено полное и точное решение, но отсутствует его объяснение	8
Приведено полное решение, но имеются неточности в формулировках или незначительные ошибки / Решение неполное, сужает постановку задачи	6-7
Выбран верный подход к решению, но приведено неполное решение, в формулировках имеются недочёты, допущены отдельные существенные ошибки	4-5
Выбран верный метод, но в решении имеются существенные ошибки, решение не доведено до конца (до получения результата)	3
Выбран неверный метод, который не приводит к нужному результату	2
Решение не соответствует постановке задачи	1
Решение отсутствует	0

Критерии оценки ответов на теоретические вопросы:

Характеристика ответа	**Оценка**
Приведён полный и точный ответ на поставленный вопрос, приведены примеры, иллюстрирующие ответ, показано значение вопроса в теории и практике программирования, создания и использования ИКТ, показана связь вопроса с другими темами курса. Использован материал, рекомендованный для самостоятельного изучения	10

Характеристика ответа	Оценка
Приведён достаточно полный ответ на поставленный вопрос, приведены примеры, иллюстрирующие ответ, показано значение вопроса в теории и практике программирования, показана связь вопроса с другими темами курса. Использован материал, рекомендованный для самостоятельного изучения. Имеются незначительные неточности в формулировках	9
Приведён достаточно полный ответ на поставленный вопрос, приведены примеры, иллюстрирующие ответ, показано значение вопроса в теории и практике программирования, показана связь вопроса с другими темами курса. Имеются незначительные неточности в формулировках, ответ недостаточно полный, отсутствуют ссылки материалы, рекомендованные для самостоятельного изучения	8
Приведён ответ на поставленный вопрос, приведены примеры, иллюстрирующие ответ. Имеются неточности в формулировках, ответ недостаточно полный, отсутствуют ссылки материалы, рекомендованные для самостоятельного изучения	6-7
Ответ на поставленный вопрос неполный, отсутствуют примеры, иллюстрирующие его, или в них имеются серьёзные ошибки. Имеются значительные неточности в формулировках, изложение недостаточно чёткое	4-5
Имеются только общие формулировки, не раскрывающие тему, отсутствуют примеры, иллюстрирующие ответ, или в них имеются серьёзные ошибки	3
Имеются только общие формулировки, не раскрывающие тему, имеются ошибки. Полностью отсутствуют примеры, иллюстрирующие ответ	2
Имеются только общие формулировки, не соответствующие поставленному вопросу	1
Ответ отсутствует	0

Для подготовки к занятиям, выполнению лабораторных работ, контрольных (проверочных) работ и для выполнения индивидуальных заданий рекомендуется использовать в первую очередь материалы курса (конспекты лекций, презентации), размещаемые в LMS, а также материалы, рекомендованные преподавателем для самостоятельного изучения, литературу, указанную в списке литературы.

8. Содержание дисциплины

Раздел 1. ИНФОРМАТИКА И ПРЕДМЕТ ЕЁ ИССЛЕДОВАНИЯ

Тема 1. Основные понятия

Определение и свойства информации особенности экономической информации. Понятие информационного ресурса. Понятие информационной технологии, эволюция информационных технологий. Понятие информационной системы и классификация информационных систем.

Количество часов аудиторной работы:
Лекции: 2 часа.
Самостоятельная работа: 4 часа.
Распределение часов, отведённых на самостоятельную работу:

- проработка материала лекций и самостоятельное изучение дополнительного материала, рекомендованного по теме – 2 часа;
- самотестирование по вопросам темы для подготовки к экзамену – 2 часа.

Тема 2. Информатика как научная дисциплина

Информатика и предмет её исследования. Теоретические основы информатики, связь информатики с другими дисциплинами.

Количество часов аудиторной работы: тема изучается самостоятельно.
Самостоятельная работа: 4 часа.

Распределение часов, отведённых на самостоятельную работу:
- самостоятельное изучение материала, рекомендованного по теме – 2 часа;
- самотестирование по вопросам темы для подготовки к экзамену – 2 часа.

Литература по разделу:

Лядова Л.Н. Основы информатики и информационных технологий : учеб. пособие / Л.Н. Лядова, Б.И. Мызникова, Н.В. Фролова. Пермь : Перм. ун-т , 2007.

Лядова Л.Н. Конспекты лекций по темам раздела [электронные ресурсы в форме файлов MS Word и презентаций Power Point].

Формы и методы проведения занятий по разделу, применяемые учебные технологии:

«Проблемно-ориентированное» чтение лекций с обсуждением задач и решений, современных проблем информатики и информационных технологий.
Оперативные опросы по материалам лекций.

Раздел 2. КОДИРОВАНИЕ ИНФОРМАЦИИ И ПРЕДСТАВЛЕНИЕ ДАННЫХ В ПАМЯТИ КОМПЬЮТЕРА

Тема 3. Понятие системы счисления, связь между системами счисления

Понятие системы счисления. Алгоритмы перевода чисел из одной системы счисления в другую. Связь между системами счисления. Примеры.

Количество часов аудиторной работы:
Лекции: 2 часа.
Практические занятия: 2 часа.
Самостоятельная работа: 8 часов.
Распределение часов, отведённых на самостоятельную работу:
- проработка материала лекций и самостоятельное изучение дополнительного материала, рекомендованного по теме, подготовка к практическим занятиям – 4 часа;
- решение задач, предназначенных для самостоятельного решения, и пробное тестирование для самопроверки – 4 часа.

Тема 4. Понятие типа данных и представление данных в памяти компьютера

Понятие типа данных, характеристики и примеры типов. Стандартные типы данных и их аппаратная поддержка. Двоичная система – основа представления данных в памяти компьютера. Представление данных в памяти компьютера и особенности машинной арифметики. Форматы представления чисел. Представление целых чисел в форме с фиксированной точкой, знаковые и беззнаковые числа. Представление вещественных чисел.

Количество часов аудиторной работы:
Лекции: 2 часа.
Практические занятия: 2 часа.
Самостоятельная работа: 8 часов.
Распределение часов, отведённых на самостоятельную работу:
- проработка материала лекций и самостоятельное изучение дополнительного материала, рекомендованного по теме, подготовка к практическим занятиям к и контрольной (проверочной) работе – 2 часа;
- решение задач, предназначенных для самостоятельного решения, и пробное тестирование – 2 часа;
- выполнение лабораторной работы – 4 часа.

Тема 5. Конструирование типов, рекурсивные типы данных

Типы данных как конструктивные объекты. Необходимость разработки типов данных. Понятие рекурсивного типа данных. Понятие конструктора типов. Конструирование массивов, записей. Рекурсивные типы: линейные списки, деревья. Динамическое распределение памяти и конструирование типов. Операции над рекурсивными данными.

Количество часов аудиторной работы:
Лекции: 2 часа.
Практические занятия: 2 часа.
Самостоятельная работа: 8 часов.
Распределение часов, отведённых на самостоятельную работу:

- проработка материала лекций и самостоятельное изучение дополнительного материала, рекомендованного по теме, подготовка к практическим занятиям – 4 часа;
- решение задач, предназначенных для самостоятельного решения, – 4 часа.

Литература по разделу:

Лядова Л.Н. Конспекты лекций по темам раздела [электронные ресурсы в форме файлов MS Word и презентаций Power Point].
Лядова Л.Н. Основы информатики и информационных технологий : учеб. пособие / Л.Н. Лядова, Б.И. Мызникова, Н.В. Фролова. Пермь : Перм. ун-т , 2007.
Канцедал С.А. Алгоритмизация и программирование : учеб. пособие. М.: ФОРУМ-ИНФРА-М , 2010. – 351 с.
Окулов С.М. Программирование в алгоритмах: учеб. пособие. М.: БИНОМ. Лаборатория знаний , 2007. – 383 с.
Королев Л.Н., Миков А.И. Информатика. Введение в компьютерные науки. М.: Высшая школа, 2011. (Часть 1. АЛГОРИТМЫ)

Формы и методы проведения занятий по разделу, применяемые учебные технологии:

«Проблемно-ориентированное» чтение лекций с обсуждением задач и решений.
Оперативные опросы (экспресс-опросы) по материалам лекций.
Пробное тестирование с использованием LMS по темам раздела.
Практические занятия в компьютерном классе:

- индивидуальное выполнение лабораторных работ и практических заданий,
- решение задач с обсуждением и анализом результатов.

Раздел 3. ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ

Тема 6. Понятие и свойства алгоритмов. Способы записи алгоритмов

Интуитивное понятие алгоритма. Свойства алгоритмов. Понятие об исполнителе алгоритма. Уточнение понятия алгоритма. Примеры.

Текстовое и графическое представления алгоритма. Понятие псевдокода, примеры записи алгоритмов на псевдокоде. Представление алгоритмов с помощью блок-схем. Правила описания блок-схем. Принципы структурного программирования и использование структурограмм для описания алгоритмов. Примеры.

Количество часов аудиторной работы:
Лекции: 2 часа.
Практические занятия: 2 часа.
Самостоятельная работа: 8 часов.
Распределение часов, отведённых на самостоятельную работу:

- проработка материала лекций и самостоятельное изучение дополнительного материала, рекомендованного по теме, подготовка к практическим занятиям и контрольной (проверочной) работе – 2 часа;

- решение задач, предназначенных для самостоятельного решения (анализ алгоритмов, представленных с помощью различных нотаций: текстовых (псевдокод), графических (блок-схемы, структурограммы)), подготовка к выполнению контрольной (проверочной) работы и домашнего задания – 2 часа;
- выполнение лабораторной работы – 4 часа.

Тема 7. Машины Тьюринга

Алгоритм как преобразование слов из заданного алфавита. Машина Тьюринга. Формат команды и программа машины Тьюринга. Способы записи программы: таблицы, диаграммы. Примеры. Композиция машин Тьюринга. Примеры. Тезис Тьюринга и его обоснование.

Количество часов аудиторной работы:
Лекции: 2 часа
Практические занятия: 4 часа.
Самостоятельная работа: 12 часов.
Распределение часов, отведённых на самостоятельную работу:

- проработка материала лекций и самостоятельное изучение дополнительного материала, рекомендованного по теме, подготовка к практическим занятиям – 4 часа;
- решение задач, предназначенных для самостоятельного решения (разработка программ машины Тьюринга с использованием табличного представления, диаграмм; анализ программ и трассировка выполнения), подготовка к выполнению контрольной (проверочной) работы – 4 часа;
- подготовка к выполнению практического задания в рамках группового проекта (задание по теме «Разработка программного интерпретатора машины Тьюринга») – 4 часа.

Тема 8. Нормальные алгорифмы Маркова

Нормальные алгоритмы Маркова. Формулы подстановки и схемы. Выполнение алгорифма. Примеры. Принцип нормализации и его обоснование.

Количество часов аудиторной работы:
Лекции: 2 часа
Практические занятия: 2 часа.
Самостоятельная работа: 12 часов.
Распределение часов, отведённых на самостоятельную работу:

- проработка материала лекций и самостоятельное изучение дополнительного материала, рекомендованного по теме, подготовка к практическим занятиям – 4 часа;
- решение задач, предназначенных для самостоятельного решения (разработка нормальных алгорифмов Маркова; анализ и выполнение), подготовка к выполнению контрольной (проверочной) работы – 4 часа;
- подготовка к выполнению задания в рамках группового проекта (задание по теме «Разработка программного интерпретатора нормальных алгорифмов Маркова») – 4 часа.

Тема 9. Вычислимые функции и методы разработки алгоритмов

Понятие вычислимой функции. Суперпозиция, примитивная рекурсия, минимизация. Примеры. Связь с методами разработки алгоритмов. Понятие об алгоритмической неразрешимости. Доказательство существования алгоритмически неразрешимых задач. Примеры. Развитие понятия алгоритма: параллельное программирование и распределённые алгоритмы, объектно-ориентированный подход к разработке программ, методы искусственного интеллекта.

Количество часов аудиторной работы:
Лекции: 2 часа
Практические занятия: 4 часа.
Самостоятельная работа: 12 часов.
Распределение часов, отведённых на самостоятельную работу:
- проработка материала лекций и самостоятельное изучение дополнительного материала, рекомендованного по теме, подготовка к практическим занятиям – 4 часа;
- решение задач, предназначенных для самостоятельного решения, подготовка к контрольной (проверочной) работе – 4 часа;
- подготовка к выполнению лабораторных работ – 4 часа.

Тема 10. Рекурсия и итерация, особенности реализации

Рекурсия и математическая индукция. Реализация механизма рекурсии. Рекурсия и итерация. Модель стека. Использование стека в программировании. Реализация рекурсии. Средства и особенности разработки рекурсивных функций.

Количество часов аудиторной работы:
Практические занятия: 2 часа.
Самостоятельная работа: 10 часов.
Распределение часов, отведённых на самостоятельную работу:
- проработка материала занятий и самостоятельное изучение дополнительного материала, рекомендованного по теме, подготовка к практическим занятиям – 4 часа;
- решение задач, предназначенных для самостоятельного решения, подготовка к проверочной работе – 4 часа;
- подготовка к выполнению лабораторной работы – 2 часа.

Тема 11. Понятие сложности алгоритма и классы сложности задач

Понятие вычислительной сложности (по времени и памяти) алгоритма и его применение для анализа алгоритмов. Основные методы и приёмы анализа сложности. Сложность алгоритмов с ветвлениями, циклами. Сложность рекурсивных алгоритмов. Оптимизация алгоритмов. Сложность задач. Задачи полиномиальной и экспоненциальной сложности (труднорешаемые задачи). Сводимость и другие классы сложности. Примеры.

Количество часов аудиторной работы:
Лекции: 2 часа
Практические занятия: 2 часа.
Самостоятельная работа: 8 часов.
Распределение часов, отведённых на самостоятельную работу:
- проработка материала лекций и самостоятельное изучение дополнительного материала, рекомендованного по теме, подготовка к практическим занятиям – 2 часа;
- решение задач, предназначенных для самостоятельного решения, – 4 часа;
- подготовка к выполнению контрольной работы – 2 часа.

Литература по разделу:

Канцедал С.А. Алгоритмизация и программирование : учеб. пособие. М.: ФОРУМ-ИНФРА-М , 2010. – 351 с.
Окулов С.М. Программирование в алгоритмах: учеб. пособие. М.: БИНОМ. Лаборатория знаний , 2007. – 383 с.
Плаксин М.А. Тестирование и отладка программ – для профессионалов будущих и настоящих. М.: БИНОМ. Лаборатория базовых знаний, 2007.

Королев Л.Н., Миков А.И. Информатика. Введение в компьютерные науки. М.: Высшая школа, 2011. (Часть 1. АЛГОРИТМЫ)
Лядова Л.Н. Конспекты лекций по темам раздела [электронные ресурсы в форме файлов MS Word и презентаций Power Point].

Формы и методы проведения занятий по разделу, применяемые учебные технологии:

«Проблемно-ориентированное» чтение лекций с обсуждением задач и решений.
Оперативные опросы по материалам лекций.
Практические занятия в компьютерном классе: индивидуальное выполнение лабораторных работ и практических заданий, решение задач с обсуждением и анализом результатов.
Выполнение проектов по темам раздела.

Раздел 4. ПРОГРАММЫ И ЯЗЫКИ ПРОГРАММИРОВАНИЯ

Тема 12. Понятие программы и жизненный цикл программ

Понятие программы как способа записи алгоритма. Спецификация программы. Программирование и программное обеспечение. Естественные и искусственные языки. Понятие языка программирования. Функции языков программирования. Классификация. Спецификация и стандартизация языков программирования. Описание языков программирования: лексика, синтаксис, семантика и прагматика.

Понятие жизненного цикла программного обеспечения.

Количество часов аудиторной работы:
Лекции: 2 часа
Самостоятельная работа: 4 часа – проработка материала лекции, подготовка к экзамену по вопросам темы.

Тема 13. Формальные грамматики и определения языка программирования, способы описания языков

Понятие формальной грамматики. Грамматики как способ определения синтаксиса языков программирования. Способы описания синтаксиса языков программирования: металингвистические формулы и диаграммы Вирта. Примеры. Эквивалентность грамматик. Классификация формальных грамматик по Хомскому.

Количество часов аудиторной работы:
Лекции: 2 часа
Практические занятия: 4 часа.
Самостоятельная работа: 12 часов.
Распределение часов, отведённых на самостоятельную работу:

- проработка материала лекций и самостоятельное изучение дополнительного материала, рекомендованного по теме, подготовка к практическим занятиям – 2 часа;
- решение задач, предназначенных для самостоятельного решения (описание грамматик с использованием различных нотаций (БНФ, диаграммы Вирта), сравнение грамматик), подготовка к контрольной работе – 6 часов;
- выполнение лабораторной работы (этапа) – 4 часа.

Тема 15. Основы трансляции программ

Языки программирования и трансляторы. Понятие транслятора, классификация трансляторов. Компиляторы и интерпретаторы. Ассемблеры. Этапы трансляции.

Количество часов аудиторной работы:
Лекции: 2 часа
Практические занятия: 6 часов.

Самостоятельная работа: 20 часов.

Распределение часов, отведённых на самостоятельную работу:

- проработка материала лекций и самостоятельное изучение дополнительного материала, рекомендованного по теме, подготовка к практическим занятиям – 4 часа;
- решение задач, предназначенных для самостоятельного решения, подготовка к контрольной работе – 4 часа;
- выполнение практического задания (лабораторной работы) (этапа реализации синтаксического анализатора и интерпретатора выражений) – 12 часов.

Литература по разделу:

Канцедал С.А. Алгоритмизация и программирование : учеб. пособие. М.: ФОРУМ-ИНФРА-М , 2010. – 351 с.

Окулов С.М. Программирование в алгоритмах: учеб. пособие. М.: БИНОМ. Лаборатория знаний , 2007. – 383 с.

Плаксин М.А. Тестирование и отладка программ – для профессионалов будущих и настоящих. М.: БИНОМ. Лаборатория базовых знаний, 2007.

Королев Л.Н., Миков А.И. Информатика. Введение в компьютерные науки. М.: Высшая школа, 2011. (Часть 1. АЛГОРИТМЫ)

Лядова Л.Н. Конспекты лекций по темам раздела [электронные ресурсы в форме файлов MS Word и презентаций Power Point].

Формы и методы проведения занятий по разделу, применяемые учебные технологии:

«Проблемно-ориентированное» чтение лекций с обсуждением задач и решений.

Оперативные опросы по материалам лекций.

Практические занятия в компьютерном классе: индивидуальное выполнение лабораторных работ и практических заданий, решение задач с обсуждением и анализом результатов.

Выполнение индивидуальных проектов по темам раздела.

Раздел 5. СОРТИРОВКА И ПОИСК

Тема 16. Задача сортировки и сортировка массивов

Задача сортировки и понятие ключа.

Подходы к сортировке данных в оперативной памяти компьютеров: сортировка массивов, сортировка таблиц, сортировка индексов и пр.

Алгоритмы сортировки: сортировка простыми вставками, сортировка выбором, обменные сортировки, сортировка подсчетами.

Количество часов аудиторной работы:

Лекции: 2 часа

Практические занятия: 2 часа.

Самостоятельная работа: 12 часов.

Распределение часов, отведённых на самостоятельную работу:

- проработка материала лекций и самостоятельное изучение дополнительного материала, рекомендованного по теме, подготовка к практическим занятиям и экзамену по вопросам темы – 2 часа;
- решение задач, предназначенных для самостоятельного решения, подготовка к экзамену (решение задач по вопросам темы) – 2 часа;
- самостоятельное выполнение лабораторной работы (программная реализация и сравнение методов сортировки), оформление отчёта – 12 часов.

Тема 16. Файлы и файловые системы, внешняя сортировка

Понятие файла и файловой системы. Файлы и папки (каталоги, директории).

Логическая и физическая организация файлов. Реализация ввода-вывода и буферизация файлов.

Файлы с последовательной организацией. Файлы с прямым доступом. Индексация файлов.

Особенности внешней сортировки. Подходы к решению задачи сортировки файлов. Алгоритмы внешней сортировки.

Количество часов аудиторной работы:
Лекции: 2 часа
Практические занятия: 2 часа.
Самостоятельная работа: 12 часов.
Распределение часов, отведённых на самостоятельную работу:

- проработка материала лекций и самостоятельное изучение дополнительного материала, рекомендованного по теме, подготовка к практическим занятиям и экзамену по вопросам темы – 4 часа;
- решение задач, предназначенных для самостоятельного решения, выполнение лабораторной работы и подготовка к выполнению индивидуального задания – 8 часов.

Тема 17. Поиск информации и хеширование

Задача поиска. Абсолютный и относительный поиск. Поиск и сортировка. Бинарный поиск в массивах и использование бинарных деревьев.

Хеширование данных и понятие хеш-функции. Методы разрешения коллизий (рехеширование, метод цепочек).

Количество часов аудиторной работы:
Лекции: 2 часа
Практические занятия: 4 часа.
Самостоятельная работа: 12 часов.
Распределение часов, отведённых на самостоятельную работу:

- проработка материала лекций и самостоятельное изучение дополнительного материала, рекомендованного по теме, подготовка к практическим занятиям и экзамену по вопросам темы – 4 часа;
- решение задач, предназначенных для самостоятельного решения, подготовка к выполнению лабораторной работы и индивидуального задания и к экзамену (решение задач по вопросам темы) – 4 часов;
- выполнение лабораторной работы – 4 часа.

Тема 18. Поиск данных во внешней памяти

Особенности задачи поиска данных в файлах.

Задача поиска и индексация файлов. Понятие *B*-дерева и использование *B*-деревьев для поиска данных во внешней памяти.

Количество часов аудиторной работы:
Лекции: 2 часа
Практические занятия: 4 часа.
Самостоятельная работа: 8 часов.
Распределение часов, отведённых на самостоятельную работу:

- проработка материала лекций и самостоятельное изучение дополнительного материала, рекомендованного по теме, подготовка к практическим занятиям и экзамену по вопросам темы – 4 часа;
- решение задач, предназначенных для самостоятельного решения, подготовка к экзамену (решение задач по вопросам темы) – 4 часа.

Литература по разделу:

Канцедал С.А. Алгоритмизация и программирование : учеб. пособие. М.: ФОРУМ-ИНФРА-М , 2010. – 351 с.
Окулов С.М. Программирование в алгоритмах: учеб. пособие. М.: БИНОМ. Лаборатория знаний , 2007. – 383 с.
Плаксин М.А. Тестирование и отладка программ – для профессионалов будущих и настоящих. М.: БИНОМ. Лаборатория базовых знаний, 2007.
Королев Л.Н., Миков А.И. Информатика. Введение в компьютерные науки. М.: Высшая школа, 2011. (Часть 1. АЛГОРИТМЫ)
Лядова Л.Н. Конспекты лекций по темам раздела [электронные ресурсы в форме файлов MS Word и презентаций Power Point].

Формы и методы проведения занятий по разделу, применяемые учебные технологии:

«Проблемно-ориентированное» чтение лекций с обсуждением задач и решений.
Оперативные опросы по материалам лекций.
Пробное тестирование с использованием LMS по темам раздела.
Решение задач на практических занятиях с анализом и обсуждением результатов.

Раздел 6. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ РАЗРАБОТКИ РАСПРЕДЕЛЁННЫХ И ПАРАЛЛЕЛЬНЫХ СИСТЕМ

Тема 19. Проблема взаимного исключения

Проблема взаимного исключения. Понятие критической секции, свойства, условия реализации. Программные методы решения. Семафорная техника реализации взаимного исключения. Примеры.

Количество часов аудиторной работы:
Лекции: 2 часа.
Самостоятельная работа: 6 часов.
Распределение часов, отведённых на самостоятельную работу:
- проработка материала лекций и самостоятельное изучение дополнительного материала, рекомендованного по теме, – 2 часа;
- решение задач, предназначенных для самостоятельного решения, подготовка к практическим занятиям и экзамену (решение задач по вопросам темы) – 6 часов.

Тема 20. Проблема тупика и её решение

Проблема тупика. Задачи предотвращения, распознавания, обхода тупиков, вывод системы из тупика. Математические модели, лежащие в основе решения.

Количество часов аудиторной работы:
Лекции: 2 часа.
Самостоятельная работа: 6 часов.
Распределение часов, отведённых на самостоятельную работу:
- проработка материала лекций и самостоятельное изучение дополнительного материала, рекомендованного по теме, – 2 часа;
- решение задач, предназначенных для самостоятельного решения, подготовка к экзамену (решение задач по вопросам темы) – 4 часа.

Литература по разделу:

Лядова Л.Н. Конспекты лекций по темам раздела [электронные ресурсы в форме файлов MS Word и презентаций Power Point].

Формы и методы проведения занятий по разделу, применяемые учебные технологии:

«Проблемно-ориентированное» чтение лекций с обсуждением задач и решений.
Оперативные опросы по материалам лекций.

9. Образовательные технологии

Используется «проблемное» чтение лекций по дисциплине с использованием компьютерного мультимедийного оборудования, предусматривающее разбор практических задач, моделирование типовых ситуаций, возникающих при разработке программного обеспечения различного назначения с использованием универсальных языков программирования высокого уровня, а также особенностей архитектуры персональных компьютеров, иллюстрирующих изучаемый материал, показывающих его практическую значимость и связь с другими дисциплинами. Лекции должны носить интерактивный характер, активизировать и мотивировать самостоятельную работу студентов, позволять им выявлять недостаток знаний и навыков, необходимых для решения поставленных задач.

На практике используются инструментальные средства (системы программирования) учебного назначения, позволяющие получить базовые навыки работы со средствами разработки программного обеспечения, подготовить студентов к использованию для разработки промышленных технологий. Проводятся также занятия в форме деловой игры («Легко ли быть программистом»).

В качестве рекомендаций для оформления отчётов о выполнении заданий рекомендуются стандарты оформления программной документации (ЕСПД), отчетов о НИР и НИОКР. Отчёты оформляются с применением приложений MS Office (как средств обработки текстов, так и работы с графическим материалом, схемами).

Для оперативного информирования и размещения материалов, для проведения текущего контроля рекомендуется использовать возможности LMS.

9.1. Методические рекомендации преподавателю

Работа должна быть направлена на решение поставленных выше образовательных задач, достижение студентами запланированных результатов (получения знаний, умений, навыков, формирование указанных компетенций). Формы проведения занятий, формы контроля описаны выше.

Материал каждой лекции рекомендуется иллюстрировать примерами, рассматривать нестандартные ситуации, требующие решения с использованием изучаемого материала, показывающие практическую значимость вопросов курса.

При этом студенты должны активно участвовать в обсуждении вопросов, выработке решений. Предлагаемые студентами решения обсуждаются, анализируются и оцениваются в ходе лекции. Предлагается рассматривать не только «верные», оптимальные решения, но и неэффективные решения, решения, приводящие к ошибкам. По каждому рассматриваемому на лекции вопросу следует предложить задачи для самостоятельного решения и вопросы для самостоятельного изучения с использованием материалов, размещенных в LMS, доступных в локальной сети. Предлагается также рекомендовать студентам использовать источники в Интернет для поиска решений, их анализа, сравнения.

На *практических занятиях* используются следующие методы обучения и контроля усвоения материала:

1) выполнение практических заданий по теме занятия сопровождается контрольным опросом;
2) выполнение лабораторных работ по теме занятия сопровождается оперативным консультированием (в том числе и с использованием возможностей LMS, электронной почты);
3) обсуждение различных вариантов решения, предложенных студентами, сравнение решений, анализ «предлагаемых обстоятельств» – вариантов изменения исходных данных, условий решения задач;
4) при оценке выполненных работ проводить «защиту» полученных решений, отчётов.

Рекомендуется проведение защит в форме деловой игры.

В учебном процессе рекомендуется использовать возможности LMS для проведения «самоконтроля» (пробного тестирования), выполнения проектов (домашних заданий, лабораторных работ).

Выполнение индивидуальных заданий и групповых проектов рекомендуется организовать как многоэтапные проекты с промежуточными контролями. Графики выполнения заданий рекомендуется согласовать со студентами, развивая у них навыки планирования работы.

9.2. Методические указания студентам

Для решения практических задач и выполнения домашних заданий, для подготовки к контрольным работам *рекомендуется использовать следующие основные источники*:

1. Методические материалы по курсу (тексты лекций, презентации), размещенные в LMS, на сервере в локальной сети НИУ ВШЭ – Пермь.
2. Справочные системы приложений, библиотеку MSDDN.
3. Источники в Интернет (для поиска и анализа решений практических заданий. Их оценки, возможности применения при выполнении проектов).

При разработке программ на языках программирования рекомендуется использовать справочные системы систем программирования, примеры и рекомендации по решению задач, приведенные в электронных пособиях по курсу, указанных в списке дополнительной литературы.

Студентам рекомендуется следующая схема *подготовки к практическому занятию*:

1) проработать конспект лекций;
2) проанализировать основную и дополнительную литературу, рекомендованную по изучаемому разделу;
3) при необходимости найти дополнительную информацию в Internet, на сайтах электронных библиотек;
4) проанализировать варианты решений, предложенные преподавателем, найденные в дополнительных источниках;
5) при затруднениях сформулировать вопросы к преподавателю и обратиться за консультацией.

Студенту рекомендуется следующая схема *подготовки к лекции*:

1) проработать конспект лекций;
2) изучить материал, предложенный для самостоятельного изучения;
3) выполнить предложенные преподавателем задания;
4) при затруднениях задать вопросы к преподавателю при проведении занятий и индивидуальных консультаций.

Рекомендуется при выполнении лабораторных работ, практических заданий и при подготовке к контрольным работам рассмотреть возможность защиты предложенных решений, подготовить документацию и «презентацию» работы.

По результатам выполнения контрольных (проверочных) работ рекомендуется разобрать решения и проанализировать ошибки, недостатки, за которые была снижена оценка.

Для самостоятельного изучения материала курса и подготовки к лекциям и практическим занятиям, проверочным работам предлагается использовать электронные ресурсы, размещаемые на сервере НИУ ВШЭ – Пермь, в LMS.

При выполнении лабораторных работ, индивидуальных заданий рекомендуется разбить проекты на этапы, разработать календарный план, определяющий этапы выполнения работы с привязкой к практическим занятиям и лекциям по соответствующим темам, учитывающий индивидуальные особенности и планы.

10. Оценочные средства для текущего контроля и аттестации студента

10.1. Оценочные средства для оценки качества освоения дисциплины в ходе текущего контроля

Примеры заданий и примерные требования к их выполнению приведены в приложениях (Приложение 2) к настоящей программе, размещаются в LMS.

Вопросы для самопроверки и для подготовки к экзамену приведены в Приложении 3 к настоящей программе, список вопросов размещается в LMS (уточняется перед проведением экзамена).

10.2. Примеры заданий промежуточного/итогового контроля

Примеры задач, заданий для проведения экзамена, примерные задания для проведения текущего контроля приведены в Приложениях 4-6, тексты заданий размещаются в LMS.

11. Порядок формирования оценок по дисциплине

Преподаватель оценивает работу студента на лекциях и практических занятиях (участие в обсуждениях на лекциях, активности на практических занятиях (работа у доски, ответы с места: оценка повышается при выборе обоснованного, эффективного решения, при выполнении анализа вариантов решений, при разборе решения и выявлении ошибок)). Оцениваются также результаты выполнения студентом всех видов заданий: индивидуальных проектов (см. выше), лабораторных работ, контрольных (проверочных) работ, пробных тестов, оперативных опросов.

Оценки $O_{аудиторная}$ за аудиторную работу (работу на практических занятиях) выставляются в рабочую ведомость перед завершающим (итоговым) контролем. Оценка определяется по 10-балльной шкале. Оценка за аудиторную работу включает в себя оценки, полученные за выполнение лабораторных работ, групповых проектов и вычисляется по формуле:

$$O_{аудиторная} = n_1 \cdot O_{лр1} + n_2 \cdot O_{лр2} + n_3 \cdot O_{лр3} + n_4 \cdot O_{лр4} + n_5 \cdot O_{лр5} + n_6 \cdot O_{лр6} + n_7 \cdot O_{лр7} + n_8 \cdot O_{гп},$$

где $O_{лр_i}$ – оценки за лабораторные работы (i = 1, 2, 3, 4, 5, 5, 7), выставляемые по 10-балльной шкале; n_i – коэффициенты, определяющие вес лабораторных работ в оценке за аудиторную работу по дисциплине (i = 1, 2, 3, 4, 5, 5, 7), n_i = 0,1; $O_{гп}$ – оценка за выполнение группового проекта, n_8 = 0,3.

Способ округления оценки за аудиторную работу: арифметический.

На оценку влияет также посещаемость занятий: за каждый пропуск без уважительной причины оценка за аудиторную работу снижается на 1 балл. Активное участие в работе на практических занятиях (решение задач у доски, участие в анализе решений) также оценивается в 1 балл, прибавляемый к оценке за аудиторную работу.

Оценки за самостоятельную работу ($O_{сам.\ работа}$), предусмотренные программой дисциплины (п. 6), определяются по результатам выполнения *индивидуальных практических заданий*, предусматривающих поэтапную разработку приложения на языке C#, охватывающего все основные темы курса.

Оценки за текущий контроль ($O_{текущая}$), предусмотренный программой дисциплины (п. 6), графиком учебного процесса, определяются по результатам выполнения контрольных (проверочных) работ. Критерии оценки работ (заданий) приведены выше. Общая оценка за текущий контроль вычисляется по формуле:

$$O_{текущая} = k_1 \cdot O_{кр1} + k_2 \cdot O_{кр2} + k_3 \cdot O_{кр3} + k_4 \cdot O_{кр4},$$

где $O_{кр_i}$ – оценки за контрольные (проверочные) работы (i = 1, 2, 3, 4), выставляемые по 10-балльной шкале; k_i – коэффициенты, определяющие вес работ в оценке за аудиторную работу по дисциплине (i = 1, 2, 3, 4), k_i = 0,25.

Способ округления оценки текущего контроля: арифметический.

Накопленная оценка учитывает все результаты работы студента по дисциплине: оценку за аудиторную работу студента, оценку по текущему контролю и оценку за самостоятельную работу. Для расчета накопленной оценки знаний студентов предлагается следующая формула:

$$O_{накопленная} = 0{,}4 \cdot O_{текущая} + 0{,}3 \cdot O_{аудиторная} + 0{,}3 \cdot O_{сам.\ работа},$$

где $O_{текущая}$ рассчитывается как взвешенная сумма всех оценок по всем формам текущего контроля (контрольным (проверочным) работам), предусмотренным в программе (формула приведена выше); $O_{сам.\ работа}$ – оценка за самостоятельную работу (выполнение индивидуального практического задания); $O_{аудиторная}$ – оценка за аудиторную работу (формула для расчёта приведена выше).

Способ округления накопленной оценки: арифметический.

Для расчета результирующей оценки знаний студентов предлагается следующая формула:

$$O_{результирующая} = 0{,}6 \cdot O_{накопленная} + 0{,}4 \cdot O_{итог.\ контроль}.$$

Оценка за итоговый контроль определяется по результатам выполнения письменной экзаменационной работы. Все вопросы и задания, включённые в экзаменационные билеты, оцениваются по 10-балльной шкале. Оценка за итоговый контроль вычисляется по формуле:

$$O_{итог.\ контроль} = 0{,}3 \cdot O_{теор.\ вопрос} + 0{,}7 \cdot O_{задачи},$$

где $O_{теор.\ вопрос}$ – оценка за теоретический вопрос; $O_{задачи}$ – оценка за выполнение практических заданий (решение задач).

Способ округления оценки итогового контроля в форме письменного экзамена: арифметический.

На экзамене студент может получить *дополнительный вопрос* (*дополнительную практическую задачу*), ответ на который оценивается в 1 балл, добавляемый к оценке. Студент может также получить дополнительный балл к оценке, выполнив и представив преподавателю *до начала экзамена отчёт о выполнении индивидуального практического задания*, если оно не было сдано (разрешение на пересдачу задания предоставляется в исключительных случаях, при невыполнении его в установленные графиком сроки по уважительной причине, например, при большом числе пропусков занятий по болезни).

На пересдаче экзамена студенту не предоставляется возможность получить дополнительный балл.

В *диплом выставляется* результирующая оценка по учебной дисциплине.

12. Учебно-методическое и информационное обеспечение дисциплины

12.1. Базовый учебник

Канцедал С.А. Алгоритмизация и программирование : учеб. пособие. М.: ФОРУМ-ИНФРА-М , 2010. – 351 с.

12.2. Основная литература

1. *Окулов С.М.* Программирование в алгоритмах: учеб. пособие. М.: БИНОМ. Лаборатория знаний , 2007. – 383 с.
2. *Павловская Т.А.* Паскаль. Программирование на языке высокого уровня: учебник. СПб. : ПИТЕР, 2010. – 460 с.
3. *Плаксин М.А*. Тестирование и отладка программ – для профессионалов будущих и настоящих. М.: БИНОМ. Лаборатория базовых знаний, 2007.

12.3. Дополнительная литература

1. *Андреева Т.А*. Программирование на языке Pascal: Учебное пособие. М.: 2006.-234 с. [Электронный ресурс].

2. *Борисенко В.В*. Основы программирования: Учебное пособие. М.: Интернет-университет информационных технологий; МГУ им. М. В. Ломоносова, 2005.328 с. [Электронный ресурс].
3. *Костюкова Н.И*. Графы и их применение. Комбинаторные алгоритмы для программистов : учеб. пособие. М.: Интернет-Университет Информационных Технологий ; М. : БИНОМ. Лаборатория знаний , 2010. – 311 с.
4. *Лядова Л.Н*. Основы информатики и информационных технологий : учеб. пособие / Л.Н. Лядова, Б.И. Мызникова, Н.В. Фролова. Пермь : Изд-во Перм. ун-та , 2004. – 310 с.
5. Конспект лекций по курсу «Информатика и программирование». М.: ГУ-ВШЭ [Электронный ресурс]

12.4. Справочники, словари, энциклопедии

Интерактивные справочные системы используемых систем программирования и приложений MS Office, библиотека MSDN.

12.5. Программные средства

Для успешного освоения дисциплины, студент использует следующие программные средства:

- Microsoft Visual Studio .NET.
- Microsoft Visio.
- Pascal ABC / Free Pascal.
- Интегрированный пакет Microsoft Office (MS Word и Excel).
- Средства, обеспечивающие возможность доступа к материалам для подготовки к занятиям в различных форматах (документы MS Word, документы в форматах PDF, HTML, презентации MS Power Point), размещенным в LMS, на сервере в локальной сети НИУ ВШЭ – Пермь, материалы, доступные в Internet (на сайте INTUIT и пр.).

12.6. Дистанционная поддержка дисциплины

Система LMS (размещены все материалы курса).

Дистанционная поддержка (средства проверки программ) используется для выполнения домашних заданий. Программное обеспечение устанавливается на сервере.

13. Материально-техническое обеспечение дисциплины

Для проведения лекционных занятий используется компьютер с установленным программным обеспечением для демонстрации презентаций и проектор.

Практические занятия проводятся в компьютерных классах с установленным программным обеспечением, перечисленным выше.

Приложение 1

Входной контроль может быть проведён в форме теста, письменной работы и/или выполнения заданий на компьютере.

Вопросы и задачи, а также варианты теста приведены ниже.

Вопросы и задачи для входного контроля[1]

1. Написать программу, которая выводит на экран максимальное из значений элементов целочисленного массива, содержащего 15 элементов.

2. Написать программу, которая выводит на экран номера строк матрицы размером 5×10, содержащей вещественные числа, сумма элементов которых меньше нуля. Если такой строки нет (по всем строкам суммы элементов неотрицательны), то на экран нужно вывести сообщение – строку «Таких строк в матрице нет!».

3. Написать программу сортировки (упорядочения) элементов массива, содержащего 20 элементов – символов (букв латинского алфавита), в лексико-графическом (алфавитном) порядке.

4. Написать программу, которая выводит на экран номер первого элемента целочисленного массива, содержащего 25 элементов, значение которого является минимальным среди значений всех элементов массива.

5. Написать программу, которая выводит на экран номера строк матрицы размером 10×5, содержащей вещественные числа, и суммы элементов, находящихся в этих строках. Номер и сумма для каждой строки матрицы должны выводиться в отдельных строках на экране.

6. Написать программу сортировки (упорядочения) элементов массива, содержащего 20 элементов – символов (букв латинского алфавита), в лексико-графическом (алфавитном) порядке.

7. В текстовом файле содержится следующая информация (4 строки):

```
Массив (матрица A) содержит следующие значения (по строкам):
1   2   3
4   5   6
7   8   9
```

Необходимо в основной программе организовать ввод данных, заполнив ими двумерный массив (матрицу *A*), транспонировать матрицу с помощью написанной Вами процедуры и в основной программе вывести результат работы процедуры в текстовый файл в формате:

```
Результат транспонирования матрицы A:
    столбцы
     строки  1    2    3
             ------------
        1   | 1    4    7
        2   | 2    5    8
        3   | 3    6    9
```

[1] При подготовке заданий входного контроля допускается рассмотреть варианты как разработки программ на одном из языков программирования, так и разработки алгоритмов (описание может быть выполнено в любой форме: блок-схема, псевдокод)

Напишите программу, решающую описанную выше задачу, организовав ввод данных не из файла, а с клавиатуры, и вывод результатов на экран.

8. Написать программу, которая вводит массив записей содержащих сведения о студентах группы: фамилию, имя и отчество, дату рождения, оценки, полученные в сессию. Сохранить введённые данные в файле. Найти для каждого студента средний балл и вывести результаты на экран.

9. Используя возможности системы программирования, включите в программу, выполняющую суммирование элементов массива, вводимых из текстового файла, средства контроля выхода за границы диапазонов допустимых значений.

10. Напишите программу, которая выводит на экран значение факториала для натурального числа, которое вводится с клавиатуры. Вычисление реализуется с помощью функции.

11. Напишите программу, которая выводит на экран значение скалярного произведения двух векторов, значения элементов которых вводятся с клавиатуры. Для ввода значений элементов массива напишите процедуру. Вычисление скалярного произведения реализуется как функция (векторы передаются как параметры).

12. Организуйте ввод последовательности чисел с клавиатуры и найдите:
    а) сумму всех введённых чисел;
    б) сумму положительных чисел;
    в) максимум среди отрицательных чисел.

    Работа программы завершается, когда с клавиатуры будет ведён символ ‘*’. Используйте возможности Pascal для предотвращения ошибок, связанных с вводом недопустимых значений и выходом за границы допустимых диапазонов значений при вычислениях.

13. Перепишите программу (предыдущее задание), организовав ввод данных из текстового файла.

Примечание: программу можно написать на любом языке, которым Вы владеете (предпочтительнее – Pascal); если Вы не можете написать программу на языке программирования, можно записать алгоритм решения задачи на псевдокоде или представить его в виде блок-схемы.

Варианты проверочного теста по основам программирования

Вариант 1

1. Вычислить выражение

```
(ord(chr(ord('8')+1))-ord('0'))+round(cos(sin(0))+3/1.5)
```

1 10 **2** 11 **3** 12 **4** 13 **5** Выражение содержит ошибку

---

2. Какое из следующих выражений всегда истинно?

а. `(A and B) or (not A and not B)` **б**. `(A and A or not A and A)`

в. `(not A or A and A)`

1 а **2** б **3** в **4** Все **5** Ни одно

---

3. Дана программа:

```
program s;
 Const N=10;
 Var A:array[1..N] of integer;
     i,j:integer;
 procedure Exchange (a,b:integer);
 var c:integer;
 begin c:=a;a:=b;b:=c;end;
begin
 for i:=1 to N-1 do
  for j:=i+1 to N do
   if A[j]<A[j-1] then Exchange(A[j],A[j-1]);
end.
```

Как программа изменяет массив A?

1 Сортирует массив по убыванию
2 Сортирует массив по возрастанию
3 Меняет местами первый и последний элементы массива
4 Меняет местами минимальный и максимальный элементы массива
5 Не изменяет массив

---

4. В программе описан массив A (**`A:array [1..10] of real`**).

Дано описание функции:

```
function f1(i:integer):integer;
begin
  f1:=f1(i-1)*i+A[i];
end;
```

Что будет выведено на экран в результате вызова функции – выполнения оператора ***`write`*****`(f1(3))`** ?

1 Сумма первых 3-х элементов массива
2 Сумма элементов массива с 3-го по 10-ый
3 Сумма всех элементов массива
4 3*A[1]+2*A[2]+A[3]
5 Ничего не будет выведено, т.к. описание функции содержит ошибку

5. Что будет выведено на экран при выполнении программы:

```
program S;
var K: integer; F: text;
function N(A: integer; B: integer; var C: integer):integer;
begin
  if K>0 then
  begin
    K:=B*3-A;C:=C-C div 2; N:=B-N(K,C div 2,C); write(F, C);
  end
  else N:=K+A
end;
begin assign(F, 'FILE_TXT.TXT'); rewrite(F); K:=5; write(N(2*K,K,K))
end.
```

1 1 **2** Программа не закончит работу **3** 3 **4** 4 **5** 5 3 1

6. В программе описаны переменные **Y: real; A: real; B: char; C: integer**. Какое из описаний соответствует вызову **Y:=X(A*2, ord(B), C, B, A+C)?**

1 `function X(a:real;b:integer;var c:real;d:char;e:real):integer;`
2 `function X(a:real;b:real;var c:integer;var d:char;e:real):integer;`
3 `function X(a:real;b:integer;c:integer; d:char;var e:real):char;`
4 `procedure X(a:real;b:real;c:real; var d:char; e:real);`
5 `function X(a:real;b:real;var c:integer; var d:char; var e:real):real;`

Вариант 2

1. Вычислить выражение

```
(ord(chr(ord('8')+1))-ord('0'))+round(cos(sin(0))+3 div 1.5)
```

1 10 **2** 11 **3** 12 **4** 13 **5** Выражение содержит ошибку

---

2. Какое из следующих выражений всегда истинно?

а. `(A and B) or (not A and not B)` **б**. `(A and A or not A and A)`
в. `(not A or A) and (B and not B)`

1 а **2** б **3** в **4** Все **5** Ни одно

---

3. Дана программа:

```
program s;
 Const N=10;
 Var A:array[1..N] of integer;
     i,j:integer;
 procedure Exchange (a,b:integer);
 var c:integer;
 begin c:=a;a:=b;b:=c;end;
begin
 for i:=1 to N-1 do
  for j:=i+1 to N do
   if A[j]<A[j-1] then Exchange(A[j],A[j-1]);
end.
```

Как программа изменяет массив A?

1 Не изменяет массив
2 Сортирует массив по убыванию
3 Сортирует массив по возрастанию
4 Меняет местами первый и последний элементы массива
5 Меняет местами минимальный и максимальный элементы массива

---

4. В программе описан массив A (`A:array [1..10] of real`). Дано описание функции:

```
function F1(I:integer):integer;
begin
  F1:= F1(I-1) * I + A[I]
end;
```

Что будет выведено на экран в результате вызова функции – выполнения оператора ***write*****(F1(5))** ?

1 Сумма первых 3-х элементов массива
2 Сумма элементов массива с 3-го по 10-ый
3 Сумма всех элементов массива
4 5*A[3]+2*A[4]+A[5]
5 Ничего не будет выведено, т.к. описание функции содержит ошибку

5. Что будет выведено на экран при выполнении программы:

```
program S;
var K: integer; F: text;
function N(A: integer; B: integer; var C: integer):integer;
begin
  if K>0 then
  begin K:=B*3-A;C:=C-C div 2; N:=B-N(K,C div 2,C); write(F, C); end
  else N:=K+A
end;
begin assign(F, 'FILE_TXT.TXT'); rewrite(F); K:=5; write(N(2*K,K,K))
end.
```

1 1 **2** Программа не закончит работу **3** 3 **4** 4 **5** 5 3 1

6. В программе описаны переменные **Y: real; A: real; B: char; C: integer**. Какое из описаний соответствует вызову **Y:=X(A*2, ord(B), C, B, A+C)** ?

1 `function X(a:real;b:integer;var c:real;d:char;e:real):integer;`
2 `function X(a:real;b:real;var c:integer;var d:char;e:real):integer;`
3 `procedure X(a:real;b:real;c:real; var d:char; e:real);`
4 `function X(a:real;b:integer;c:integer; d:char;var e:real):char;`
5 `function X(a:real;b:real;var c:integer; var d:char; var e:real):real;`

Приложение 2

Примерный перечень задач для подготовки к контрольным работам и экзамену по соответствующим темам

Приведены примерные задания для подготовки к контрольной. По темам и уровню сложности приведенные задачи соответствуют контрольным заданиям. Для подготовки к контрольной необходимо выполнить все домашние задания и разобрать примеры, приведенные в лекциях. Задания, помеченные звездочками, имеют больший уровень сложности. Их решение необходимо для получения *отличной оценки*.

Контрольная работа включает задачи по следующим темам:

1. Представление и кодирование информации.
2. Способы описания алгоритмов.
3. Способы описания синтаксиса языков программирования.
4. Машины Тьюринга.
5. Нормальные алгорифмы Маркова.
6. Вычислимые функции.
7. Рекурсия.
8. Оценка сложности алгоритмов.
9. Формальные грамматики и определения языка программирования, способы описания языков
10. Основы трансляции программ

Задания по другим темам – в отдельном приложении.

Задачи по теме «Представление и кодирование информации»

1. **Перевести во внутреннее представление в памяти компьютера следующие числа:**
   a) 123 – в формат числа со знаком и числа без знака (использовать формат, наиболее подходящий для представления числа);
   b) 254 – в формат числа со знаком и числа без знака (использовать формат, наиболее подходящий для представления числа);
   c) –129 – в формат числа со знаком (использовать формат, наиболее подходящий для представления числа);
   d) 4589 – в формат двоично-десятичного числа;
   e) –24,125 – в формат числа с плавающей точкой;
   f) 0,03125 – в формат числа с плавающей точкой.
2. **Перевести во внутреннее представление в памяти компьютера следующие числа:**
   a) 123 – в формат числа со знаком и числа без знака (использовать формат, наиболее подходящий для представления числа);
   b) 254 – в формат числа со знаком и числа без знака (использовать формат, наиболее подходящий для представления числа);
   c) –129 – в формат числа со знаком (использовать формат, наиболее подходящий для представления числа);
   d) 4589 – в формат двоично-десятичного числа;
   e) –24,125 – в формат числа с плавающей точкой;
   f) 0,03125 – в формат числа с плавающей точкой.
3. **Как можно интерпретировать следующую строку битов (рассмотрите разные типы данных, поддерживаемые процессором Intel):**
   a) 01001111000011110101010101010100;
   b) 11100111010101011111000000000000;
4. **Описать алгоритмы перевода чисел (данные вводятся как строка символов с клавиатуры и результаты выводятся в символьном виде на экран):**
   c) из десятичной системы в двоичную;

d) из двоичной системы в десятичную;
e) из двоичной системы в шестнадцатеричную;
f) из шестнадцатеричной системы в двоичную;
g) из двоичной системы в восьмеричную;
h) из восьмеричной системы в двоичную;
i) из десятичной системы в двоичнно-десятичную;
j) из двоично-десятичной системы в двоичнную.

Типовые задачи по теме «Способы описания алгоритмов»

Задание 1. Для заданий 1-13 из Приложения 1 разработайте описания алгоритмов
а) на псевдокоде; б) в виде блок-схемы; в) в виде структурограммы.

Задание 2. Даны описания алгоритмов в виде блок-схем. Приведите соответствующие структурограммы.

Задание 3. Даны описания алгоритмов в виде структурограмм. Приведите соответствующие блок-схемы.

Задание 4. Даны описания алгоритмов на псевдокоде. Приведите описания соответствующих алгоритмов в виде блок-схем, структурограмм.

Задание 5. Дано описание алгоритма. Проанализируйте его. Все ли свойства алгоритмов соблюдаются? Поясните ответы.

Задание 6. Даны описания алгоритмов. Проанализируйте их. Решают ли они одну и ту же задачу? При каких условиях результаты выполнения алгоритмов будут совпадать/различаться?

Типовые задачи по теме «Формальное определение алгоритма»

I. ***Построить машину Тьюринга*** (представить программу в *виде таблицы* и в *форме диаграммы*) для решения следующих задач:

1. Прибавить 1 к целому неотрицательному числу (вычислить функцию $F(x) = x + 1$). Рассмотреть задачу для машины Тьюринга с алфавитами
   a) $A = \{0, 1, e\}$ (операции выполняются в двоичной системе);
   b) $A = \{1, e\}$ (строка «…e1e…» соответствует $x = 0$, в записи любого другого целого числа $x > 0$ количество единиц равно $x + 1$).

2. Вычесть 1 из целого неотрицательного числа (вычислить функцию $F(x) = x - 1$). Операции выполняются по следующему правилу:
$$F(x) = \begin{cases} 0, \text{если } x = 0 \\ x - 1, \text{если } x > 0 \end{cases}.$$
   Рассмотреть задачу для машины Тьюринга с алфавитами
   a) $A = \{0, 1, e\}$ (операции выполняются в двоичной системе);
   b) $A = \{1, e\}$ (строка «…e1e…» соответствует записи $x = 0$ на ленте, в записи любого другого целого числа $x > 0$ количество единиц равно $x + 1$).

3. Прибавить 3 к целому неотрицательному числу (вычислить функцию $F(x) = x + 3$). Построить решение в двух вариантах:
   a) машина Тьюринга строится для вычисления указанной функции;
   b) машина Тьюринга строится *как композиция машин Тьюринга*, вычисляющих функции $F(x) = x + 1$ и $F(x) = x + 2$.

4. Умножить на 2 целое положительное число x (вычислить функцию $F(x) = 2 \times x$)). Числа записываются в двоичной системе (используется алфавит $A = \{0, 1, e\}$).

5. Умножить на 2 целое положительное число (вычислить функцию $\boldsymbol{F(x) = 2\times x}$). Для записи чисел используется алфавит $\boldsymbol{A = \{1, e\}}$ (строка «…e1e…» соответствует записи $x = 0$ на ленте, в записи любого другого целого числа $x > 0$ количество единиц равно $x + 1$)*******.
6. Умножить на 4 целое положительное число, записанное в двоичной системе (вычислить функцию $F(x) = 4\times x$).
   a) машина Тьюринга строится для вычисления указанной функции;
   b) машина Тьюринга строится *как композиция машин Тьюринга*, вычисляющих функции $F(x) = 2\times x$.
7. Умножить на степень 2 (на 2^n) целое положительное число x (вычислить функцию $F(x, n) = 2^n\times x$)).
   Числа записываются в двоичной системе (используется алфавит $A = \{0, 1, e, \otimes\}$).
   На ленте числа x и n записываются последовательно и разделяются символом ‘⊗’. В результате выполнения программы машины Тьюринга на ленте должно остаться одно число – результат умножения*******.
8. На ленте записано целое число (используется двоичная система счисления, значение может быть как положительным, так и отрицательным, отрицательное значение записывается в прямом коде со знаком минус, перед положительным числом может стоять знак плюс, т.е. используется алфавит $A = \{0, 1, e, +, –\}$). Написать программу машины Тьюринга для вычисления функций*******
   a) $F(x) = x + 1$ (увеличения на 1 к записанного на ленте числа);
   b) $F(x) = x – 1$ (вычитания 1 из записанного на ленте числа).
9. Построить композиции машин Тьюринга для вычисления функций $f(x) = (x+1)\times 4$, $f(x) = (x+2)\times 2$, $F(x) = 2\times x + 1$.
10. Построить машину Тьюринга, вычисляющую:
    a) обратный код записанного на ленте числа;
    b) дополнительный код записанного на ленте числа.
11. Построить машину Тьюринга для вычисления функции $I_m^n(x_1, x_2,\ldots, x_n) = x_m$. Исходные данные – записанные на ленте значения $m, x_1, x_2,\ldots, x_n$, разделенные запятыми. Результат – значение x_m (все остальные значения должны быть стерты)*******.
12. Построить машину Тьюринга для вычисления функции проверки чётности числа.
13. Построить машину Тьюринга для вычисления логических операций AND, OR, NOT, XOR:
    a) операции выполняются над значениями, представленными одним битом (0 и 1);
    b) операции выполняются поразрядно над значениями, представленными в формате байта.

Проверьте результаты работы созданных машин, протестировав их на различных входных данных (тесты разработайте самостоятельно).

II. ***Какую функцию вычисляет машина Тьюринга*** с программой, представленной таблицей, приведенной ниже, если на ленте записано подряд $x + 1$ единиц, а слева и справа от них – символы e. Маркер находится против левой единицы. Таблица имеет вид:

	q1	**q2**	**q3**	**q4**	**q5**	**q6**
e	*e q*1 *S*	*e q*3 *S*		1 *q*3 *S*	1 *q*3 *S*	1 *q*3 *S*
1	*e q*2 *R*	*e q*6 *R*		1 *q*5 *S*	1 *q*4 *R*	1 *q*3 *S*

Покажите по шагам*, как вычисляется результат выполнения приведенной выше программы* машины для числа **15**, записав число в предложенной системе.

III. ***Какую функцию вычисляет машина Тьюринга*** с программой, представленной таблицей, если на ленте записано подряд $x + 1$ единиц, а слева и справа от них – символы *e*. Маркер находится против левой единицы. Таблица имеет вид:

	***q*1**	***q*2**	***q*3**	***q*4**	***q*5**
e	*e q*1 *S*	*e q*4 *S*	1 *q*4 *R*	1 *q*5 *R*	
1	*e q*2 *R*	*e q*3 *R*	1 *q*3 *R*		

Покажите по шагам*, как вычисляется результат* выполнения программы построенной машины для числа **7**.

IV. ***Определить результат применения алгорифма Маркова*** с заданной схемой к заданной строке (***показать по шагам***, как применяются формулы подстановок).

1. Каков результат действия на слово *P = nnnnnnnnn* нормального алгорифма со схемой
   Z = | nnn → abc | ca → ac | ba → nnnn | ?
2. Каков результат действия на слово *P = hgdfasdfghjgdsadfgg* нормального алгорифма со схемой *F = | fg → sa | df →. hg | j → Λ | gg →. d |*?

V. ***Разработать алгорифм Маркова***, решающий следующую задачу (описать алфавит и схему):

1. В тексте все вхождения знака препинания «точка» (символа '.') в конце предложений заменить на восклицательный знак. В тексте могут использоваться строчные и прописные буквы кириллицы и *все* знаки препинания.
2. Дан текст с некоторыми перечислениями. Перечислению предшествует фраза, заканчивающаяся двоеточием (символом ':'). Один элемент перечисления отделяется от другого точкой с запятой (символом ';'). Необходимо перечисление представить в виде маркированного списка, каждый элемент которого начинается с новой строки, с маркера '-'.

   Пример:

   Исходный текст:

   Светофор имеет три цвета: красный; жёлтый; зелёный.

   Результат преобразования:

   Светофор имеет три цвета: красный, жёлтый, зелёный.

Задачи по теме Вычислимость и рекурсия

VI. Имеется множество вычислимых функций

$$P = \{ z(x), I^n_m(x_1, x_2,\ldots, x_n), Inc(x) \}.$$

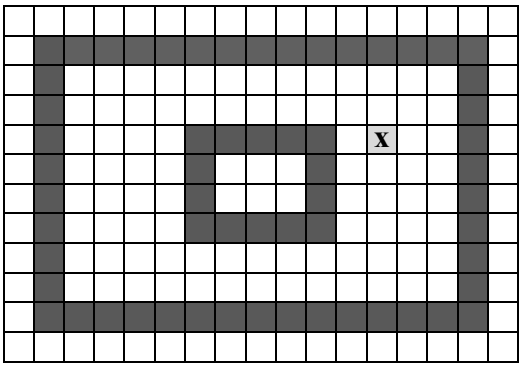
($z(x) = 0$, $I^n_m(x_1, x_2,\ldots, x_n) = x_m$, $Inc(x) = x + 1$). ***Используя операции суперпозиции σ, примитивной рекурсии ρ и минимизации μ, постройте функции***

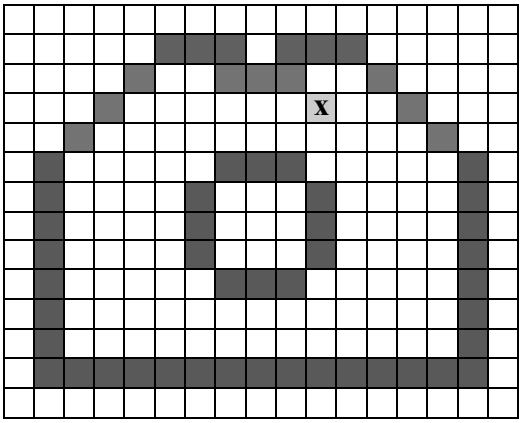
1. $Dec(x) = x - 1$,
2. $Add(x, y) = x + y$,
3. $Sub(x, y) = x - y$,
4. $Mult(x, y) = x * y$,
5. $Power(x, y) = x^y$,
6. $F(x) = x!$, где x – целое неотрицательное число.

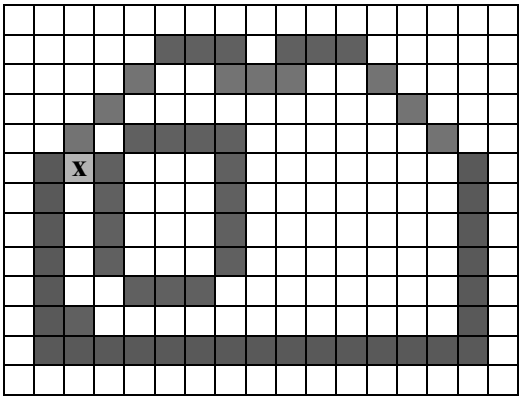
Покажите порядок вывода формулы и пример вычисления функции для 0 и 3, подставляя указанные значения в построенную формулу. При построении функций можно использовать полученные ранее функции, расширяя ими базовый набор.

VII. Напишите на языке программирования высокого уровня функции, формулы для вычисления которых выведены в предыдущем задании. Реализуйте итерационный и рекурсивный варианты.

VIII. ***Напишите рекурсивную процедуру закраски фигуры***. Используется два цвета (цвет границ фигур и цвет закраски). Покажите порядок закраски по шагам (постройте дерево вызова и выполните «заливку» (до 15 вызова процедуры), используя приведенную ниже схему). Начальная точка отмечена буквой "**x**" (первый вызов процедуры выполняется для точки с координатами, соответствующими выделенной на схеме клетке), каждой точке на экране соответствует элемент матрицы на схеме:







IX. ***Напишите рекурсивную функцию***, вычисляющую функцию Маккарти («**Функция 91**») по следующему правилу:

Маккарти(**n**) = ***n*** –10, если ***n***>100,
иначе ***Маккарти***(***Маккарти***(***n***+11)).

Постройте дерево вызовов *и покажите состояние стека при вызове функции со следующими начальными значениями*: *n* = 98.

X. ***Напишите рекурсивную функцию,*** вычисляющую функцию Аккермана по следующему правилу:

Аккерман(***m***, ***n***) = ***n***+1, если ***m*** = 0 или
иначе ***Аккерман***(***m***–1, 1), если ***n*** = 0 или
иначе ***Аккерман***(***m***–1, ***Аккерман***(***m***, ***n***–1)).

Постройте дерево вызовов *при вызове функции со следующими начальными значениями*: *n* = 2, *m* = 1.

XI. Напишите программы, которые позволили бы ввести данные (числа *m* и *n*) с клавиатуры и вычислить и вывести на экран значения функций Маккарти и Аккермана. Проверьте полученный результат (дерево вызова), выполнив программу в пошаговом режиме. Измените входные данные.

Задания по теме «Способы описания синтаксиса языков программирования»

***Задание* 1**. Постройте *диаграммы Вирта* для заданных с помощью БНФ конструкций языка программирования.

***Задание* 2**. Постройте *БНФ* (*формулы Бэкуса-Наура*) для заданных с помощью диаграмм Вирта конструкций языка программирования.

***Задание* 3**. Установите соответствие описаний, выполненных с помощью БНФ и диаграмм Вирта.

***Задание* 4**. Проверьте строки на соответствие синтаксическим правилам, заданным с помощью БНФ и диаграмм Вирта.

***Задание* 5**. Постройте дерево вывода для заданных строк и грамматики.

***Задание* 6**. Постройте дерево вызовов синтаксических процедур, соответствующих конструкциям языка, описанным в грамматике для заданных строк и грамматики.

***Задание* 7**. Опишите с помощью БНФ грамматики:

a. Целых чисел без знака.
b. Целых чисел со знаком.
c. Вещественных чисел со знаком (в форме с фиксированной и плавающей точкой).
d. Выражений.

К каким классам по классификации Хомскогоони относятся?

***Задание* 8**. Опишите с помощью диаграмм Вирта грамматики:

a. Целых чисел без знака.
b. Целых чисел со знаком.
c. Вещественных чисел со знаком (в форме с фиксированной и плавающей точкой).
d. Выражений.

***Задание* 9**. Проверьте, выводимы ли строки для построенных грамматик, постройте дерево разбора:

a. +324
b. -23
c. -0.6778
d. +56.55E+34

e. 1
f. +0
g. 000
h. 00.00
i. 0.
j. .10
k. 0.000E-45667778
l. 233.E+1
m. (((0)))
n. (+577)
o. ((-0.99))
p. (8+00)*45
q. 0/5
r. 0*((77-9))
s. 7*55+0/666
t. (7*7)+(7/7)
u. ((99+8)-(6))
v. ((99)*(6-99))

Задания по теме «Оценка сложности алгоритмов»

***Задание* 1**. Оцените сложность процедуры (в зависимости от исходных данных *V*):

1.
```
procedure F (V: integer);
    var I,J,K: integer;
        S,T: real;
  begin
    T := 0;
    for I:=1 to V do
    begin
      S:=I*123;
      for J:=V downto 2 do
        for K:=1 to I+V*V do T:=(T+S)/5
    end
  end;
```

2.
```
procedure P (V: integer; var B: real);
    var M,K: integer;
        A: real;
  begin
    for M:=1 to V do
    begin
      A:=M+2;
      K:=2;
      while K<V do
      begin
        A:=A-V+M*K;
        B:=2-A;
        K:=K*K
      end
    end
  end;
```

***Задание* 2**. Напишите на языке программирования высокого уровня рекурсивную и итерационную функции вычисления

1. $Add(x, y) = x + y$,
2. $Sub(x, y) = x - y$,
3. $Mult(x, y) = x * y$,
4. $Power(x, y) = x^y$,
5. $F(x) = x!$, где x – целое неотрицательное число.

(задание IV по теме «Вычислимость и рекурсия») и оцените их временную сложность.

***Задание* 3**. Для заданных описаний (предоставленных текстов процедур) выполните оценки и оптимизируйте код. Обоснуйте решение.

Задачи по теме «Взаимное исключение»

1. Задачи, решаемые программой, реализуются с помощью потоков. Каждый поток соответствует одной из описанных ниже на псевдокоде процедур. При выполнении потоков используются общие данные, обработка которых требует взаимного исключения. Для решения задачи взаимного исключения используются программные методы. Определите, решена ли задача взаимного исключения, если в рамках процесса может выполняться один первичный поток, соответствующий процедуре *Main*, и несколько потоков, представленных процедурами *Proc1* и *Proc2*. В описании процедур на псевдокоде использованы следующие обозначения: BEFORE – код, предшествующий критической секции, может содержать любые операторы, не использующие общих данных; AFTER – код, следующий за критической секцией, может содержать любые операторы, не использующие общих данных; CS1 и CS2 – код критических секций, требующий взаимного исключения; глобальные переменные процесса описаны с помощью ключевого слова ***common***; процедура ***start*** порождает поток для процедуры, имя которой передается ей в качестве параметра. Если при выполнении программы могут возникнуть ошибки, объясните их.

```
procedure Main; common boolean C1, C2;
begin C1:=false; C2:=false; BEFORE; start(Proc1); start(Proc2); AFTER; end;
procedure Proc1; common boolean C1, C2;
  Begin
    while true do
        begin  BEFORE1; C1:=true; while C2 do ; CS1;  C1:=false; AFTER1; end
  end;
procedure Proc2; common boolean C1, C2;
  Begin
    while true do
        begin  BEFORE2; while C1 do ; C2:=true; CS2; C2:=false; AFTER2; end
  end;
```

2. Переменная S – переменная, которая может принимать любые целочисленные значения. В описанной ниже программе переменная S является счетчиком ресурсов, которые производятся потоками, соответствующими процедурам *Producer*, и потребляются потоками, представленными процедурами *Consumer*. Для доступа к этим ресурсам потоки-производители используют процедуру *ReleaseResource*, увеличивающую счетчик ресурсов на 1, а потоки-потребители – процедуру *ReserveResource*, уменьшающую счетчик ресурсов на 1, причем потребители могут продолжить выполнение только в том случае, когда они получают в свое распоряжение запрошенную ими единицу ресурса, если же ресурс оказывается недоступным, должно быть организовано его ожидание до получения ресурса от производителя. Для синхронизации доступа к счетчику используются бинарные семафоры. Определите, правильно ли решена проблема синхронизации работы потоков, являющихся производителями и потребителями ресурсов, в описанной ниже на псевдокоде программе, при условии, что в процедуре *Main* может быть запущено произвольное число потоков производителей и потребителей. Если при выполнении программы могут возникнуть ошибки, объясните их. В описании на псевдокоде использованы следующие обозначения: *Main* – процедура, представляющая первичный поток; процедура ***start*** порождает поток для процедуры, имя

которой передается ей в качестве параметра; глобальные переменные процесса описаны с помощью ключевого слова ***common***.

```
procedure Main; common binary semaphore B1, B2; common integer S;
 begin B1:=1; B2:=1; S:=0;
        while true do  begin …; start (Producer); …; start (Consumer); …; end
 end;
procedure ReserveResource; common binary semaphore B1, B2; common integer S;
 Begin P(B1); P(B2); S:=S–1; V(B1) end;
procedure ReleaseResource; common binary semaphore B1, B2; common integer S;
 Begin P(B1); S:=S+1; V(B2); V(B1) end;
procedure Producer;  Begin … ReleaseResource; … end;
procedure Consumer;  Begin … ReserveResource; … end;
```

3**.** Задачи, решаемые программой, реализуются с помощью потоков. Каждый поток соответствует одной из описанных ниже на псевдокоде процедур. При выполнении потоков используются общие данные, обработка которых требует взаимного исключения. Для решения задачи взаимного исключения используются программные методы. Определите, решена ли задача взаимного исключения, если в рамках процесса может выполняться один первичный поток, соответствующий процедуре *Main*, и несколько потоков, представленных процедурами *Proc1* и *Proc2*. В описании процедур на псевдокоде использованы следующие обозначения: BEFORE – код, предшествующий критической секции, может содержать любые операторы, не использующие общих данных; AFTER – код, следующий за критической секцией, может содержать любые операторы, не использующие общих данных; CS1 и CS2 – код критических секций, требующий взаимного исключения; глобальные переменные процесса описаны с помощью ключевого слова ***common***; процедура ***start*** порождает поток для процедуры, имя которой передается ей в качестве параметра. Если при выполнении программы могут возникнуть ошибки, объясните их.

```
procedure Main; common boolean C1, C2;
begin C1:=false; C2:=false; BEFORE; start(Proc1); start(Proc2); AFTER; end;
procedure Proc1; common boolean C1, C2;
   Begin while true do
      begin BEFORE1; while C2 do ;  C1:=true; CS1;  C1:=false; AFTER1; end
   end;
procedure Proc2; common boolean C1, C2;
   Begin while true do
      begin BEFORE2; while C1 do ;  C2:=true; CS2; C2:=false; AFTER2; end
   end;
```

4. Переменная *S* – переменная, которая может принимать любые целые значения. В описанной ниже программе переменная *S* является счетчиком ресурсов, которые производятся потоками, соответствующими процедурам Producer, и потребляются потоками, представленными процедурами Consumer. Для доступа к этим ресурсам потоки-производители используют процедуру *ReleaseResource*, увеличивающую счетчик ресурсов на 1, а потоки-потребители – процедуру *ReserveResource*, уменьшающую счетчик ресурсов на 1, причем потребители могут продолжить выполнение только в том случае, когда они получают в свое распоряжение запрошенную ими единицу ресурса, если же ресурс оказывается недоступным, должно быть организовано его ожидание до получения ресурса от производителя. Для синхронизации доступа к счетчику используются бинарные семафоры. Определить, правильно ли решена проблема синхронизации работы потоков, являющихся производителями и потребителями ресурсов, в описанной ниже на псевдокоде программе, при условии, что в процедуре *Main* может быть запущено произвольное число потоков производителей и потребителей. Если при выполнении программы могут возникнуть ошибки, объясните их. В описании на псевдокоде использованы следующие обозначения: *Main* – процедура, представляющая первичный поток; процедура ***start*** порождает поток для процедуры, имя которой передается ей в качестве параметра; глобальные переменные процесса описаны с помощью ключевого слова ***common***.

```
procedure Main; common binary semaphore B; common integer S;
 begin B:=1; S:=0; while true do begin …; start (Producer); …; start (Consumer); …; end end;
procedure ReserveResource; common binary semaphore B1, B2; common integer S;
    Begin P(B); if  S>0 then S:=S–1; V(B) end;
procedure ReleaseResource; common binary semaphore B1, B2; common integer S;
    Begin P(B); S:=S+1; V(B) end;
procedure Producer;   Begin … ReleaseResource; … end;
procedure Consumer; Begin … ReserveResource; … end;
```

5. Задачи, решаемые программой, реализуются с помощью потоков. Каждый поток соответствует одной из описанных ниже на псевдокоде процедур. При выполнении потоков используются общие данные, обработка которых требует взаимного исключения. Для решения задачи взаимного исключения используются программные методы. Определите, решена ли задача взаимного исключения, если в рамках процесса может выполняться один первичный поток, соответствующий процедуре *Main*, и несколько потоков, представленных процедурами *Proc1* и *Proc2*. В описании процедур на псевдокоде использованы следующие обозначения: BEFORE – код, предшествующий критической секции, может содержать любые операторы, не использующие общих данных; AFTER – код, следующий за критической секцией, может содержать любые операторы, не использующие общих данных; CS1 и CS2 – код критических секций, требующий взаимного исключения; глобальные переменные процесса описаны с помощью ключевого слова ***common***; процедура ***start*** порождает поток для процедуры, имя которой передается ей в качестве параметра. Если при выполнении программы могут возникнуть ошибки, объясните их.

```
procedure Main; common boolean C1, C2;
begin C1:=false; C2:=false; BEFORE; start(Proc1); start(Proc2); AFTER; end;
procedure Proc1; common boolean C1, C2;
  Begin while true do
       begin BEFORE1; C1:=true; while C2 do ;  CS1;  C1:=false; AFTER1; end
  end;
procedure Proc2; common boolean C1, C2;
  Begin while true do
            begin BEFORE2; C2:=true; while C1 do ;  CS2; C2:=false; AFTER2; end
  end;
```

6. Задача “обедающие философы” формулируется следующим образом: “Пять философов садятся обедать за круглый стол, в центре которого стоит одно блюдо со спагетти. На столе имеется пять тарелок и пять вилок между ними. Философ может начать есть, если у него есть тарелка и две вилки, которые он может взять с двух сторон от своей тарелки. Философ может отдать вилки соседям только после того, как он закончит обед”. О соображениях гигиены мы здесь умалчиваем. Может ли описанный ниже на псевдокоде процесс представить алгоритм поведения философа за столом?

```
Инициализация:
{ “Вилки”, которые используются философами – это разделяемые ресурсы,
защищенные бинарными семафорами: }
        common B: array[0..4] of binary semaphore; { глобальный массив семафоров}
        for I:=0 to 4 do B[I]:=1; { все “вилки” свободны }

process Философ(I: Integer);
{ Это процесс, описывающий поведение I-го философа. }
{ “Вилки”, которыми он ест, защищаются двумя бинарными семафорами. }
    common B: array[0..4] of binary semaphore
begin { Пытается получить вилки: } P(B[I]); P(B[(I+1) mod 5]);
   { Ест спагетти, используя вилки, находящиеся слева и справа } …
{ Освобождает вилки: } V(B[(I+1) mod 5]); V(B[I]);
end;
```

7. Задача “обедающие философы” формулируется следующим образом: “Пять философов садятся обедать за круглый стол, в центре которого стоит одно блюдо со спагетти. На столе имеется пять тарелок и пять вилок между ними. Философ может начать есть, если у него есть тарелка и две вилки, которые он может взять с двух сторон от своей тарелки. Философ может отдать вилки соседям только после того, как он закончит обед”. О соображениях гигиены мы здесь умалчиваем. Может ли описанный ниже на псевдокоде процесс представить алгоритм поведения философа за столом?

```
Инициализация:
{ “Вилки”, которые используются философами – это разделяемые ресурсы, защищенные бинарными семафорами: }
    common B: array[0..4] of binary semaphore; { глобальный массив семафоров}
    for I:=0 to 4 do B[I]:=1; { все “вилки” свободны }

process Философ(I: Integer);
{ Это процесс, описывающий поведение I-го философа. }
{ “Вилки”, которыми он ест, защищаются двумя бинарными семафорами. }
    common B: array[0..4] of binary semaphore
begin { Пытается получить вилки: }
  while (P(B[I])=0) or (P(B[(I+1) mod 5])=0) do ;
  { Ест спагетти, используя вилки, находящиеся слева и справа } …
{ Освобождает вилки: }
  V(B[(I+1) mod 5]); V(B[I]);
end;
```

8. Задача: написать процедуры, моделирующие семафорные примитивы для общего семафора (допускаются только неотрицательные значения целочисленной переменной, моделирующей считающий семафор). При написании процедур можно использовать бинарные семафоры. Ниже приведен код процедур, которые моделируют работу семафорных примитивов для общего семафора в соответствии с поставленной задачей. Решена ли задача? Найти в приведенном ниже псевдокоде ошибки, объяснить и исправить их, если они есть.

```
Init: proc (var S: integer; value: integer); common binary semaphore B1, B2;
      begin B1:=1; B2:=1; S:=value end;
PP: procedure (var S: integer); common binary semaphore B1, B2;
      Begin P(B2); P(B1); S:=S–1; if S>0 then V(B2); V(B1) end;
VP: procedure (var S: integer); common binary semaphore B1, B2;
    Begin P(B1); S:=S+1; V(B1); V(B2 ) end;
```

9. Задача: написать процедуры, моделирующие семафорные примитивы для общего семафора (допускаются любые значения целочисленной переменной, моделирующей считающий семафор). При написании процедур можно использовать бинарные семафоры. Ниже приведен код процедур PP и VP, которые моделируют работу семафорных примитивов для общего семафора в соответствии с поставленной задачей. Решена ли задача? Найти в приведенном ниже псевдокоде ошибки, объяснить и исправить их, если они есть.

```
Init: proc (var S: integer; value: integer); begin S:=value end;
PP: procedure (var S: integer); common binary semaphore B1, B2;
   Begin P(B1); S:=S–1; if S<0 then begin V(B1); P(B2) end
                                  else V(B1) end;
VP: procedure (var S: integer); common binary semaphore B1, B2;
   Begin P(B1); S:=S+1; if S<=0 then V(B2); V(B1 ) end;
```

10. Задача: написать процедуры, моделирующие семафорные примитивы для общего семафора (допускаются любые значения целочисленной переменной, моделирующей считающий семафор). При написании процедур можно использовать бинарные семафоры. Ниже приведен код процедур PP и VP, которые моделируют работу семафорных примитивов для общего семафора в соответствии с поставленной задачей. Решена ли задача? Найти в приведенном ниже псевдокоде ошибки, объяснить и исправить их, если они есть.

```
Init: proc (var S: integer; value: integer); begin  S:=value; B1:=1; B2:=1  end;
PP: procedure (var S: integer); common binary semaphore B1, B2;
   Begin P(B1); S:=S–1; if  S<0  then begin V(B1); P(B2) end
                                 else  V(B1) end;
VP: procedure (var S: integer);  common binary semaphore B1, B2;
   Begin P(B1); S:=S+1; if  S > 0  then V(B2); V(B1 ) end;
```

11. Алгоритм Деккера можно сформулировать следующим образом:

Процедура инициализации	**Первый процесс**	**Второй процесс**

procedure INIT;
common boolean C1,C2 ;
common integer N ***;***
begin
C1 := ***false ;***
C2 := ***false ;***
N := 1 ;
start(P_1) ;
start(P_2)
end INIT **.**
	process P_1***;***
	common boolean C1,C2 ;
	begin
	while true do
	begin $BEFORE_1$;
	C1 := ***true ;***
	while C2 ***do***
	begin
	if N<>1 ***then***
	begin
	C1 := ***false ;***
	while N<>1 ***do ;***
	C1 := ***true ;***
	end
	end ;
	CS_1 ;
	C1 := ***false;*** N := 2***;***
	$AFTER_1$;
	end
	end P_1 **.**
		process P_2***;***
		common boolean C1,C2 ;
		begin
		while true do
		begin $BEFORE_2$;
		C2 := ***true ;***
		while C1 ***do***
		begin
		if N<>2 ***then***
		begin
		C2 := ***false ;***
		while N<>2 ***do ;***
		C2 := ***true ;***
		end
		end ;
		CS_2 ;
		C2 := ***false;*** N:=1***;***
		$AFTER_2$;
		end
		end P_2 **.**

Задание: Написать алгоритм Деккера для N процессов.

12. Решить задачу взаимного исключения, используя логические переменные и процедуру перевода процесса в состояние задержки (приостановки) ***Delay***(*T*), где параметр *T* задает время задержки процесса (время, на которое он выводится из конкуренции за время процессора).

13. Решается ли в приведенных ниже программах проблема взаимного исключения (***random*** – процедура, генерирующая случайное число, ***delay***(*T*) – процедура задержки процесса на время *T*, в течение которого процесс не будет конкурировать за время процессора)?

```
process INIT; common boolean C1. C2; common integer N; common real T1, T2;
begin C1:=false; C2:=false; T1:= random; T2:= random; N:=1; start(P1); start(P2);  end;
    process P1;
        Begin  while true do
                begin  BEFORE1;
                       C1:=true;
                       while C2 and (N<>1)  do delay(T1);
                       CS1;
                       C1:=false;
                       N:=2;
                       AFTER1;
                end
        end;
    process P2;
        Begin
                while true do
                begin  BEFORE2;
                       C2:=true;
                       while C1 and (N<>2)  do delay(T2);
                       CS2;
                       C2:=false; N:=1;
                       AFTER2;
                end
        end;
```

Задачи по теме «Тупики»

1. Пусть в системе есть два процесса P1 и P2 и два единичных повторно используемых ресурса R1 и R2, требующих монополизации при использовании. Процессы имеют следующее описание:

a)

Процесс 1	**Процесс 2**
process P1 **;** ***begin*** **...** ***While true do*** ***begin*** **…** *request* **(** R1, 1**)** *;* **...** *request* **(** R2, 1**) ;** **...** *release* **(** R2, 1**) ;** **...** *release* **(** R1, 1**) ;** **...** ***end*** **...** ***end.***	***process*** P2 **;** ***begin*** **...** ***While true do*** ***begin*** **…** *request* **(** R2, 1**) ;** **...** *request* **(** R1, 1**) ;** **...** *release* **(** R2, 1**) ;** **...** *release* **(** R1, 1**) ;** **...** ***end*** **...** ***end.***

b)

Процесс 1	**Процесс 2**
process P1 **;** ***begin*** **...** ***While true do*** ***begin*** **…** *request* **(** R1, 1**)** *;* **...** *request* **(** R2, 1**) ;** **...** *release* **(** R2, 1**) ;** **...** *release* **(** R1, 1**) ;** **...** ***end*** **...** ***end.***	***process*** P2 **;** ***begin*** **...** ***While true do*** ***begin*** **…** *request* **(** R1, 1**) ;** **...** *request* **(** R2, 1**) ;** **...** *release* **(** R2, 1**) ;** **...** *release* **(** R1, 1**) ;** **...** ***end*** **...** ***end.***

Построить граф состояний системы. Возможны ли в системе тупиковые состояния?

2. Пусть в системе есть два процесса P1 и P2 и два потребляемых ресурса R1 и R2, используемых этими процессами. Ресурс R1 производится процессом P1, а R2 – P2. Процессы имеют следующее описание:

Процесс 1	**Процесс 2**
process P1 **;** ***begin*** **...** ***While true do*** ***begin*** **…** *request* **(** R2, 1**)** *;* **...** *release* **(** R1, 1**) ;** **...** ***end*** **...** ***end.***	***process*** P2 **;** ***begin*** **...** ***While true do*** ***begin*** **…** *request* **(** R1, 1**) ;** **...** *release* **(** R2, 1**) ;** **...** ***end*** **...** ***end.***

Построить граф состояний системы (считать, что в начальный момент времени в системе имеется по 1 единице ресурса R1). Возможны ли в системе тупиковые состояния?

Приложение 3

Вопросы для самоконтроля (для подготовки к экзамену)

1. Основные понятия информатики. Информация, данные. Информационный ресурс. Информационная технология. Информационная система.
2. Понятие системы счисления. Связь между системами счисления и алгоритмы перевода.
3. Понятие типа данных. Представление данных в памяти компьютера. Форматы представления числовых данных.
4. Конструирование типов. Рекурсивные типы данных: определение, примеры.
5. Понятие алгоритма и свойства алгоритмов. Примеры.
6. Способы записи алгоритмов. Примеры.
7. Вычислимые функции. Базовый набор функций и операции над функциями: суперпозиция, примитивная рекурсия, минимизация. Примеры.
8. Методы разработки алгоритмов. Суперпозиция, итерация и рекурсия. Примеры.
9. Формализация понятия алгоритма: машина Тьюринга. Представление машин Тьюринга с помощью диаграмм. Табличное представление программ машины Тьюринга. Сравнение. Примеры. Композиция машин Тьюринга. Связь с методами разработки алгоритмов.
10. Формализация понятия алгоритма: нормальные алгорифмы Маркова, определение и выполнение. Примеры.
11. Проблема алгоритмической разрешимости. Примеры алгоритмически неразрешимых задач.
12. Рекурсивные алгоритмы: определение и виды рекурсии. Примеры рекурсивных алгоритмов различных видов. Реализация рекурсии и использование стека. Рекурсия и итерация. Примеры, сравнение.
13. Задача анализа сложности алгоритмов. Понятие сложности. Оценки сложности. Использование управляющего графа для оценки сложности алгоритма. Оценка сложности управляющих структур. Оценка сложности рекурсивных алгоритмов.
14. Понятие сложности задачи и классы сложности задач. Типы задач. Понятие сводимости, полиномиальная сводимость.
15. Понятие формальной грамматики. Описание грамматики с помощью металингвистических формул (БНФ) и диаграмм Вирта. Классификация грамматик.
16. Понятие формального языка. Описание формальных языков.
17. Понятие транслятора. Этапы трансляции.
18. Задача сортировки. Понятие ключа. Отношения предшествования.
19. Методы и алгоритмы сортировки массивов.
20. Файлы и особенности сортировки данных на внешних устройствах. Алгоритмы внешней сортировки.
21. Задача поиска информации, поиск по ключу, относительный и абсолютный.
22. Поиск и хеширование. Понятие хеш-функции. Разрешение коллизий.
23. Использование бинарных деревьев для решения задач поиска.
24. Поиск данных во внешней памяти. B-деревья: определение, операции, использование для поиска данных.
25. Понятие процесса. Классификация процессов. Какие отношения могут возникать между процессами? Какие проблемы необходимо решать для параллельных процессов, между которыми есть отношение конкуренции? Какие задачи решаются для взаимодействующих процессов? Для каких процессов необходимо поддерживать отношение предшествования? Приведите примеры.
26. Понятие ресурса. Классификация ресурсов. Какие проблемы связаны с разделением ресурсов процессами? Примеры.
27. Проблема взаимного исключения. Примеры.

28. Понятие критической секции. Свойства критической секции. Каковы общие условие решения задачи взаимного исключения? Примеры.
29. Программные методы решение проблемы взаимного исключения (использование логических переменных, счётчиков, задержек при выполнении процессов). Примеры.
30. Понятие семафора. Семафорные примитивы для бинарных семафоров и семафоров со счётчиками. Примеры использования семафоров (для решения задачи взаимного исключения, синхронизации процессов, решения задачи поддержания отношения предшествования).
31. Определение тупика. Задачи, связанные с проблемой тупика, подходы к решению. Примеры.
32. Определение заблокированного процесса, процесса, находящегося в тупике. Какое состояние является тупиковым? Какое состояние называется безопасным? Какое состояние называется выгодным? Примеры.
33. Необходимые условия возникновения тупика. Подходы к решению задачи предотвращения тупика, сравнение. Примеры.

По каждому вопросу необходимо приводить примеры. Рекомендуется рассматривать не только те, которые разбирались на лекциях (для получения высоких оценок – обязательно). Для получения хорошей оценки нужно показать, что имеются навыки самостоятельной работы с материалом, использования дополнительных источников, умение связать материал различных вопросов, проиллюстрировать его примерами из практики. Для получения хорошей оценки для приведенных примеров алгоритмов нужно получать оценки сложности, показывая, как сложность вычисляется

Приложение 4

Т Р Е Б О В А Н И Я
к выполнению индивидуального задания (вариант № 1) по дисциплине «Теоретические основы информатики»

Домашнее задание 1. ***Работайте процедуры синтаксического разбора и интерпретации выражений, используя в качестве примера приведённые в методических указаниях по теме описания грамматики языка выражений и тексты процедур разбора и интерпретации***.

Этап 1.

Разработайте (или переработайте) описание грамматики выражений.
Приведите описание разработанной грамматики (включая правила записи операндов (чисел и переменных)) с использованием БНФ (РБНФ), а также диаграмм Вирта.
Приведите примеры правильно записанных выражений (их можно будет использовать как тесты) и постройте для них деревья синтаксического разбора. Приведите примеры записи выражений, содержащие ошибки.
Разработайте набор тестов черного ящика для тестирования процедур синтаксического анализа выражений на описанном языке.

Этап 2.

Разработайте (или переработайте) процедуры синтаксического анализа арифметических выражений, расширив возможности приведённых в презентации к лекциям и в исходном коде на языке Pascal процедур в соответствии с разработанными на этапе 1 правилами. Доработайте процедуры обработки ошибок (в случае обнаружения ошибки разбор должен быть прекращен, на экран должно быть выведено сообщение об ошибке с указанием причины ошибки, рекомендацией по её исправлению (возможна нейтрализация ошибок).
Обратите внимание на фрагмент кода процедуры EXPRESSION, выделенный в тексте лекции зелёным цветом: всегда ли сообщения об ошибке будут выводиться только в случае возникновения ошибки? Приведите пример неверного вывода сообщения об ошибке, если такая ситуация возможна. Исправьте код, если это необходимо. При разработке процедуры NEXTSYMBOL перехода к следующему символу в строке необходимо предусмотреть ситуацию, когда анализируемая строка «неожиданно» заканчивается. Как предотвратить ошибку при выполнении процедур разбора в этом случае (процедуры разбора можно преобразовать в функции, которые в качестве результата возвращают код возврата, который можно анализировать, чтобы распознать необходимость и возможность продолжения разбора)?
Приведите описания алгоритмов синтаксического разбора в виде блок-схем и структурограмм.
Приведите описания тестов для разработанных вами процедур разбора выражений в соответствии с описанными правилами.
Оформите программный код (программа должна быть самодокументированной, текст разрабатывается в соответствии с правилами структурного программирования, комментируется).

Этап 3.

Внесите изменения в тексты процедур синтаксического разбора для построения:
а) дерева, представляющего синтаксически правильное выражение в форме, пригодной для интерпретации (узлы дерева – операции или операнды (листовые вершины представляют переменные или константы));
б) обратной польской записи в форме, пригодной для интерпретации (элементы – записи, представляющие операции или операнды) **.
В случае обнаружения ошибки структуры данных, используемые для построения дерева или ОПЗ, должны быть очищены, интерпретация не должна запускаться.

Этап 4.

Разработайте «Калькулятор» – интерпретатор, вычисляющий значение синтаксически правильного выражения, используя: а) построенное дерево; б) обратную польскую запись**.

Обязательные требования к выполнению задания:

1. Выявление и локализация синтаксических ошибок в выражении, вывод сообщений об ошибке с указанием типа ошибки, ошибочного символа и контекста.
2. Контроль ввода и выполнения арифметических операций и обработка исключений, связанных с неправильным вводом данных, несоответствием типов, ошибками при выполнении операций (переполнение, деление на ноль и т.п.).
3. Возможность повторного вычисления значения выражения с другими исходными данными, вводимыми пользователем, при использовании в выражении переменных.

Таблица 1. ***Критерии оценки программной реализации процедур синтаксического анализа и интерпретации выражений*** **(вес оценки в итоговой оценке за задание 0,7)**

№	Базовые оценки		Дополнительные баллы отмечены*	
	Содержание работы	**Вес**	**Содержание работы**	**Баллы**
1	*Синтаксический анализ* выражения	3	Возможность работы с *целыми числами*, содержащими больше одной цифры	1
			Возможность работы с *разными типами данных*: целыми и вещественными числами*	2
			Возможность использования *в качестве имен переменных идентификаторов*, представляющих собой последовательность букв и цифр*	2
			Возможность использования в записи выражения *разделителей* (пробелов)*	1
			Разбор *выражений, допускающих использование операций сложения и вычитания* (+ и –) и *круглых скобок* для изменения порядка выполнения операций	2
			Разбор *выражений, допускающих использование операций умножения и деления* (* и /)*	2
			Итого баллов за выполнение этапа задания:	**10**
Дополнительно:				
Нейтрализация ошибок при разборе выражения (дополнительное задание для повышения балла): анализ выполняется до конца строки (не до первой ошибки), при этом часть строки, содержащая ошибку, отбрасывается. Возвращается код ошибки, показывающий, можно ли вычислить выражение с нейтрализованными ошибками*				2
*Использование в выражении дополнительных операций, функций**				1-2
2	*Построение внутреннего представления выражения в виде дерева и вывод на экран* построенного дерева, представляющего выражение	4	Построение дерева *для выражений, допускающих использование операций сложения и вычитания* (+ и –) и *круглых скобок* для изменения порядка выполнения операций	3
			Построение дерева *для выражений, допускающих использование операций умножения и деления* (* и /)*	2
			Построение дерева *для выражений, допускающих использование в качестве операндов* чисел разных типов*	1
			Построение дерева *для выражений, допускающих использование в качестве операндов* идентификаторов*	2
			Вывод на экран выражения в формате обратной польской записи (ОПЗ) путём обхода построенного дерева	2
			Итого баллов за выполнение этапа задания:	**10**
Учёт уровня ошибок при построении дерева (дополнительное задание для повышения балла): при наличии ошибок, которые могут быть нейтрализованы, дерево строится для исправленного выражения*				2

3	*Интерпретация* выражения с использованием промежуточного представления в виде дерева	3	Интерпретация выражений, *допускающих использование операций сложения и вычитания* (+ и –) и *круглых скобок* для изменения порядка выполнения операций	2
			Интерпретация выражений, *допускающих использование операций умножения и деления* (* и /)*	1
			Интерпретация выражений, *допускающих использование* чисел с произвольным числом разрядов	2
			Интерпретация выражений, *допускающих использование* идентификаторов*	1
			Создание *таблицы идентификаторов* (бинарного дерева и/или хеш-таблицы), обеспечивающей *однократность* ввода данных (значений переменных)*	2
			Организация вычислений *с использованием таблиц констант*, содержащих поля, в которые записываются значения соответствующих типов, а не строки (исключается повторный перевод значений в виде строк в числовые значения)*	2
			Итого баллов за выполнение третьего этапа задания:	**10**
	Итого:	**10**		

*** Минимальная положительная оценка – 4 балла. Дополнительные баллы можно получить, выполнив соответствующие задачи, отмеченные звёздочкой.**

При успешной защите задания (успешных ответах на теоретические вопросы по теме «Формальные языки и основы трансляции» и на вопросы по реализации программы) соответствующий балл засчитывается как оценка за ответ на вопросы по данной теме на экзамене.

**** Баллы за реализацию интерпретатора с использованием обратной польской записи (ОПЗ) учитываются *в оценке за работу в течение семестра*, а также в экзаменационной оценке по теме.**

Программная реализация оценивается по 10-балльной шкале. За *реализацию процедур каждого этапа* оценка выставляется по 10-баллной шкале (вес в результирующей оценке показан в табл. 1). Вес этой оценки в результирующей оценке за выполнение домашнего задания равен **0,7**.

Отчёт о выполнении задания должен включать

1. ***Постановку задачи*** – ***описание грамматики выражений*** с использованием БНФ (РБНФ) и диаграмм Вирта (с учётом выбранных вариантов решения (описания приведены выше)); *примеры и пояснения к описанной грамматике.*
2. ***Набор тестов***, соответствующий выбранному варианту решения.
3. ***Описание алгоритма*** (пошаговая детализация) синтаксического анализа выражений в виде *блок-схем* и *структурограмм*.
4. *Самодокументированный* ***исходный код программ***.
5. ***Описание сценария работы с программой*** (руководство пользователя).

Оформление отчёта оценивается по 10-балльной шкале. Вес этой оценки в результирующей оценке за выполнение задания равен **0,3**.

Оценка *снижается*, если

- *отсутствует описание грамматики* в форме БНФ (РБНФ) или диаграмм Вирта или описания даны с нарушением правил;
- *отсутствует описание алгоритма* в форме структурограммы, в виде блок-схемы или описания даны с нарушением правил;
- *отсутствуют тесты* или *множество тестов неполно, не соответствует постановке задачи*;
- *нарушены требования к оформлению исходного кода программы*;
- *отсутствует описание порядка работы с программой*;

– *нарушены требования к оформлению отчёта* (правила определяются ГОСТ на оформление отчётов о НИР, правилами технического редактирования и методическими указаниями).

Результирующая оценка вычисляется по формуле:

(Оценка за программную реализацию) × 0,7 +
+ (Оценка за оформление отчёта) × 0,3

Предусмотрена **процедура защиты выполненного задания** (представление работы), в ходе которой необходимо объяснить и обосновать представленное решение, ответить на теоретические вопросы, решить задачи. По результатам защиты выставляется *оценка за представление задания* по 10-балльной шкале. При получении общей оценки от 7 баллов (и выше) эта оценка может быть зачтена как оценка по теме на итоговом контроле.

Оценка *снижается*, если

– *в реализации процедуры имеются ошибки*, не выявленные при тестировании (в зависимости от уровня (серьёзности ошибки));
– *не даётся анализ альтернативных вариантов* решения задачи;
– не даются ответы на вопросы *о внесении изменений в программу при изменениях в условиях задачи*;
– *не даны полные и чёткие ответы* на вопросы по представленной программной реализации.

Приложение 5

Т Р Е Б О В А Н И Я
к выполнению домашнего задания (вариант № 2) по дисциплине
«Теоретические основы информатики»

Задание: Разработать приложение, в котором реализуются *операции над данными, размещенными в индексированных файлах*.

Требования к данным: программа должна работать со структурированными данными – записями; каждая запись содержит данные различных типов (числа, строки и пр.); в каждой записи имеется поле, значение которого уникально для всего набора записей – ключ записи (ключ может быть составным).

Для ускорения поиска записей в файле *файл индексируется* по выбранному ключевому полю; при выполнении операций над данными необходимо соответствующим образом изменять (перестраивать) индекс.

Индексация файла: для ускорения поиска записей при выполнении операций строится индекс, в который включаются записи, содержащие значение ключа и данные о местоположении соответствующей записи в основном файле. Операция индексации может быть реализована как отдельная операция над файлом в меню программы. При этом существующий индекс (если он был построен) уничтожается и строится новый.

Нужно реализовать несколько *вариантов индексации*:

1. Построение *динамического индекса* (строится в памяти при открытии файла в виде *бинарного дерева*; при закрытии файла индекс уничтожается):
   а) для уникального ключа;
   б) для ключевых полей, значения которых могут повторяться в нескольких записях (в этом случае информацию о местоположении соответствующих записей нужно хранить в линейном списке, связанном с соответствующей вершиной в бинарном дереве).

   Реализация только первого варианта выполнения операции – *8 баллов.*
   Реализация второго варианта – *10 баллов*.

Например: номер зачетной книжки – уникальный ключ для записи, содержащей информацию о студенте; номер школы, в которой учился студент до поступления в вуз, – ключевое поле, которое не является уникальным; при построении индекса к каждой вершине, содержащей ключ – номер школы в дереве, подстраивается линейный список студентов – выпускников этой школы.

2. *Построение индекса в отдельном файле*, который связывается с основным файлом и хранится вместе с ним в одной и той же папке.
   Для данного файла-индекса должна выполняться *процедура его сортировки по значению ключевого поля* (*внешняя сортировка* – сортировка файла-индекса).
   Для поиска записи в индексе по ключу используется *метод деления пополам* (бинарный поиск).

   Необходимо рассмотреть варианты:
   а) с уникальными значениями ключей;
   б) с ключами, допускающими повторения.

   Реализация только первого варианта выполнения операции – *8 баллов.*
   Реализация второго варианта – *10 баллов*.

При выполнении операций над файлом, содержащим данные, должен быть соответствующим образом изменен и файл-индекс.

Для проиндексированных файлов должны быть реализованы все перечисленные ниже *операции*: добавление новых записей, просмотр, поиск и редактирование записей, удаление записей.

При *поиске информации по уникальному ключу* на экран выводится информация из одной найденной записи или сообщение, что данные не найдены.

При *поиске информации в файле по ключу, допускающему повторение значений* в нескольких записях, в качестве результата выводится список всех записей, удовлетворяющих условиям поиска, или сообщение о том, что данные не найдены.

Выполнение операций может потребовать перестройки индекса: при добавлении записи в файл добавляется и соответствующая запись в индекс; при удалении записи удаляется и запись о ней в индексе; при редактировании ключевого поля изменится и индекс (запись в индексе со старым значением ключа должна быть удалена, а с новым значением – включена в индекс).

Требования к операциям: при работе с данными должны быть реализованы следующие операции:

1. *Добавление записи*: новая запись добавляется в конец файла данных. В индекс включается соответствующая запись, содержащая ключ и информацию о местоположении новой записи (порядок сортировки данных в индексе при добавлении записи не должен быть нарушен).
2. *Поиск записи в файле по ключу*. Пользователь вводит значение ключевого поля для поиска записи. Для поиска используется индекс, из которого по ключу извлекается информация о местоположении найденной записи и данные из неё выводятся на экран или выводится сообщение о том, что данные не найдены, если запись с заданным ключом в файле отсутствует.
3. *Просмотр списка записей*, размещенных в файле.
   * Баллы, которые можно получить за выполнение данной части задания (реализацию просмотре данных) зависят от варианта выполнения операции:
     – Записи отображаются *в порядке их включения в файл*.
     – Записи выводятся на экран в порядке, который определяется значениями ключевых полей (записи при выводе сортируются с помощью индекса, который задаёт порядок просмотра записей).

   Реализация только первого варианта выполнения операции – *7 баллов.*
   Реализация первого и второго варианта – *10 баллов*.
4. *Удаление записи*: запись для удаления ищется по значению ключевого поля с использованием индекса (операция 2). При удалении записи вносятся соответствующие изменения в индекс.
   * Баллы, которые можно получить за выполнение данной части задания (реализацию операции удаления) зависят от варианта выполнения операции удаления:
     – Запись при выполнении операции *удаляется из файла без возможности восстановления*.
     – Удаляемая запись в файле *помечается для удаления* в специальном служебном поле (это поле может быть включено в индекс). Удаленные записи не отображаются при просмотре файла пользователем, но для работы с ними должны быть реализованы дополнительные процедуры:
       · *просмотра* списка всех помеченных для удаления записей,
       · *восстановления записи*, выбранной пользователем в списке удаленных записей (для выполнения операции пользователю показывается список всех помеченных для удаления записей; с выбранной записи снимается пометка об удалении);
       · *удаления всех записей, помеченных для удаления* (после выполнения этой операции восстановление будет невозможно – пользователь должен получить предупреждение об этом).

Реализация только первого варианта выполнения операции – *7 баллов*
Реализация второго варианта – *10 баллов*.

5. *Редактирование* (изменение) записей: запись для редактирования выбирается по значению ключевого поля. Пользователь при этом не должен заново вводить значения всех полей – ему должна быть предоставлена возможность редактирования ранее введенных значений полей с сохранением значений тех полей, которые не изменялись.
   * Баллы, которые можно получить за выполнение данной части задания (реализацию операции редактирования записи) зависят от варианта выполнения операции:
     – Пользователю разрешается *редактирование только неключевых полей* (соответственно при выполнении операции модификации данных *ключевое поле не изменяется*).
     – Пользователь может *менять значения всех полей записи* (включая ключевые, что требует соответствующей перестройки индекса).

   Реализация только первого варианта выполнения операции – *8 баллов*.
   Реализация второго варианта – *10 баллов*.

Все операции над данными, размещенными в файлах, реализуются с помощью *процедур* (или *функций*).

За реализацию каждой операции ***оценка выставляется по 10-баллной шкале*** (табл. 1).

Оценка за реализацию программы снижается, если
– *в реализации процедуры имеются ошибки* (в зависимости от уровня (серьёзности ошибки);
– *нарушены требования к оформлению программы* (программа должна быть самодокументированной: должна содержать все необходимые для понимания её работы и сопровождения комментарии);
– *отсутствуют тесты* или *множество тестов неполно*.

Результаты выполнения задания представляются в отчете.
Отчет о выполнении задания должен включать два раздела – для приложения, использующего динамическую индексацию с помощью бинарных деревьев, и для приложения, работающего с индексом, хранящемся в файле. Каждый раздел должен содержать следующие пункты:
– *постановку задачи*: описание выбранного варианта решения (предлагаемые варианты см. выше);
– *описание тестовых сценариев и тестов* (тесты для входных данных должны быть подготовлены в отдельных файлах, выбор теста при выполнении программы должен сделать пользователь, введя или выбрав из предлагаемого списка имя соответствующего файла);
– *самодокументированный исходный текст программы* (с комментариями, поясняющими структуры данных, логику программы);
– *руководство пользователя* (пошаговая инструкция для пользователя, поясняющая порядок работы с приложением от запуска программы до выхода из неё, описывающая все возможные варианты (сценарии) использования программы, действия пользователей во всех вариантах реализации);
– *отчет об ошибках* – статистика допущенных ошибок допущенных при разработке программы, в виде таблицы, включающей *общее количество строк* в программе, количество *строк, содержащих комментарии и строк-разделителей*, количество *подпрограмм*, количество *тестов* и количество *ошибок трансляции* (синтаксических и семантических) и *выполнения* (ошибок в алгоритме, ошибок, связанных с выходом за заданные диапазоны значений и пр.).
  Количество допущенных и исправленных ошибок на оценке не отражается!

Предусмотрена ***процедура защиты*** для выполненного задания, в ходе которой необходимо объяснить и обосновать представленное решение. По результатам защиты выставляется *оценка за представление задания* по 10-балльной шкале, которая определяет результирующую оценку.

Оценка за представление задания снижается, если *не даны полные и чёткие ответы* на вопросы по представленной программной реализации.

Результирующая оценка вычисляется по формуле:
(Оценка за программную реализацию) × 0,7 + (Оценка за оформление отчёта) × 0,3

Таблица 1. Оценка за программную реализацию операций при выполнении домашнего задания №2

Задание	**Содержание работы (вариант)**	**Максимальный балл за реализацию выбранного варианта**	**Вес задания в общей оценке по 10- балльной шкале**
Построение *динамического индекса* (строится в памяти при открытии файла в виде *бинарного дерева*; при закрытии файла индекс уничтожается)	Все операции выполняются для уникального ключа	**8**	**5**
	Все операции выполняются для ключевых полей, значения которых могут повторяться в нескольких записях	**10**	
Построение *индекса в отдельном файле,* отсортированном по значению ключа	Значение ключа уникально. Для поиска записи в индексе по ключу используется *метод деления пополам* (бинарный поиск).	**8**	**5**
	Все операции выполняются для ключевых полей, значения которых могут повторяться в нескольких записях	**10**	
Вес отдельных процедур, реализующих операции над данными, в оценке:			
Добавление записи	Новая запись добавляется в конец файла данных. В индекс включается соответствующая запись	**10**	**2**
Поиск записи в файле по ключу	Пользователь вводит значение ключевого поля для поиска записи (записей). Для поиска используется индекс	**10**	**1**
Просмотр списка записей, размещенных в файле	Записи отображаются в порядке их включения в файл	**7**	**1**
	Записи выводятся на экран в порядке, который определяется значениями ключевых полей (записи при выводе сортируются с помощью индекса, который задаёт порядок просмотра записей)	**10**	
Удаление записи	Запись при выполнении операции удаляется из файла без возможности восстановления	**7**	**3**
	Удаляемая запись в файле помечается для удаления	**10**	
Редактирование (изменение) записей	Пользователю разрешается редактирование только неключевых полей	**8**	**3**
	Пользователь может менять значения всех полей записи (включая ключевые)	**10**	
Всего баллов (оценка за программную реализацию операций):			**10**

Приложение 6

Задания группового проекта по теме «Формализация алгоритмов: машины Тьюринга и нормальные алгорифмы Маркова»

Задание проекта включает 2 задачи:

1. **Разработка интерпретатора нормальных алгорифмов Маркова.**
2. **Разработка интерпретатора машин Тьюринга.**

Цель выполнения задания – закрепление теоретических знаний по теме «Основы алгоритмизации» и получение практических навыков решения задач средней сложности с использованием VS.NET (на языке C#).

При выполнении задания студенты должны продемонстрировать:

- понимание различных способов формализации алгоритмов;
- умение анализировать поставленные задачи и разрабатывать алгоритмы их решения, используя пошаговую детализацию решений;
- способность использовать для описания алгоритмов различные способы (псевдокод, графические нотации);
- умение разрабатывать тесты для алгоритмов и программ;
- умение подобрать оптимальные структуры данных и типы, управляющие конструкции для реализации алгоритмов на языке программирования высокого уровня (C#);
- способность работать в среде VS.NET: разрабатывать, тестировать и отлаживать программы с использованием соответствующих средств;
- способность представить, объяснять и обосновывать принятые решения;
- умение работать к команде.

1. Разработка интерпретатора нормальных алгорифмов Маркова (НАМ)

Работайте приложение, реализующее интерпретатор нормальных алгорифмов Маркова. Ограничения на входные данные и порядок их обработки определите *самостоятельно*.

Этап 1.

Разработка общей схемы работы интерпретатора, условий выполнения, требований к входным данным и результатам. Схема работы интерпретатора должна быть описана в виде блок-схем и структурограмм с пошаговой детализацией алгоритмов.

Необходимо выполнить *анализ возможностей C#* (использования различных типов данных и операций) для реализации вербальных алгоритмов (интерпретатора НАМ), обосновать выбор типов данных для представления алфавита и схемы НАМ, а также исходной строки, к которой должна быть применена схема.

Примечание: для ввода данных и разбора строки можно (и нужно) использовать стандартные функции для работы со строками, имеющиеся в C#; в документацию (в отчёт) нужно включить описание используемых средств, обоснование их выбора. Кроме того, *следует рассмотреть и другие возможности работы со строками* (программная реализация с их использованием – дополнительный балл к оценке за программную реализацию и оформление отчета).

Разработка тестов (критерий чёрного ящика).

Оформление соответствующей главы в отчёте о выполнении задания.

Этап 2.

Пошаговая детализация алгоритма ввода исходных данных: ввод алфавита, схемы НАМ.

Разработка программного кода, реализующего ввод данных (алфавита и схемы).

Тестирование и отладка разработанных средств.

Документирование разработанного кода.

Этап 3.

Пошаговая детализация алгоритма интерпретации: ввод входной строки (слова в заданном алфавите) и её пошаговая обработка по заданной схеме НАМ.

Разработка программного кода, реализующего ввод исходного слова, цикл интерпретации и вывод полученного результата.

Тестирование и отладка разработанных средств.

Документирование разработанного кода.

Оценивается программная реализация интерпретатора (50% оценки, см. табл. 1) и оформление отчёта (50% оценки).

Программная реализация оценивается по 10-балльной шкале. За *реализацию процедур каждого этапа* оценка выставляется по 10-баллной шкале (вес в результирующей оценке показан в табл. 1). Вес этой оценки в результирующей оценке за выполнение домашнего задания равен 0,5 (50%). Оценка *снижается*, если *в реализации процедур имеются ошибки*, не выявленные при тестировании (в зависимости от уровня (серьёзности ошибки)), имеются замечания к стилю программирования.

Таблица 1. Критерии оценки программной реализации интерпретатора НАМ

№	Базовые оценки		Дополнительные баллы*	
	Содержание работы	Вес	Содержание работы	Баллы
1	Ввод данных (ввод алфавита, схемы НАМ)	4	Возможность задать алфавит в соответствии с потребностями пользователя и необходимостью контролировать ввод данных для обработки на принадлежность введённых символов алфавиту	2
			Контроль ввода алфавита (исключение повторов и пр.) и возможность внесения изменений	1
			Ввод схемы НАМ (с учётом реализованных для пользователя возможностей ввода с клавиатуры, из файла)	3
			Контроль ввода схемы НАМ (принадлежность левой и правой частей формул подстановок алфавиту и пр.) и возможность правки	2
			Визуализация введённых данных	1
			Возможность исправлений введённых данных	1
			Итого баллов за выполнение первого этапа задания:	**10**
2	Интерпретация НАМ	6	Организация ввода исходных данных (слова в заданном алфавите) с возможностью контроля и исправления ошибок	2
			Пошаговая интерпретация НАМ по заданной схеме с контролем выполнения (зацикливание и пр.)	5
			Возможность повторения интерпретации НАМ для новых входных (слова) данных	1
			Наличие режима пошагового выполнения с визуализацией промежуточных результатов	2
			Итого баллов за выполнение второго этапа задания:	**10**
	Итого:	**10**		

Отчёт о выполнении задания должен включать

1. *Постановку задачи* (с учётом выбранных вариантов решения, условий, требований).
2. *Описание сценариев* выполнения программы и тестов по критериям чёрного ящика.
3. *Описание* алгоритмов (с пошаговой детализацией).
4. Самодокументированный *исходный код программ*.
5. *Набор тестов*, соответствующий выбранному варианту решения.
6. *Описание и анализ средств C#*, которые могут быть использованы для реализации вербальных алгоритмов, обоснование выбора.
7. *Отчет об ошибках* – статистика допущенных ошибок допущенных при разработке программы, в виде таблицы, включающей *общее количество строк* в программе, количество

строк, содержащих комментарии и строк-разделителей, количество *подпрограмм*, количество *тестов* и количество *ошибок трансляции* (синтаксических и семантических) и *выполнения* (ошибок в алгоритме, ошибок, связанных с выходом за заданные диапазоны значений и пр.).

Количество допущенных и исправленных ошибок на оценке не отражается!

Оформление отчёта оценивается по 10-балльной шкале. Вес этой оценки в результирующей оценке за выполнение задания равен 0,5 (50% оценки).

2. Разработка интерпретатора машин Тьюринга (МТ)

Работайте приложение, реализующее интерпретатор программ для машины Тьюринга. Ограничения на входные данные и порядок их обработки определите самостоятельно.

Этап 1.

Разработка общей схемы работы интерпретатора, условий выполнения, требований к входным данным и результатам.

Разработка тестов (критерий чёрного ящика).

Оформление соответствующей главы в отчёте.

Необходимо выполнить *анализ возможностей C#* (использования различных типов данных и операций) для реализации интерпретатора программ машины Тьюринга, обосновать выбор типов данных для представления алфавита и программы, а также данных на ленте, к которым должна быть применена программа.

Примечание: Следует рассмотреть *различные возможности представления данных* в программе, оценить эффективность реализации интерпретатора с их использованием. Сравнительный анализ нужно включить в отчёт.

Этап 2.

Пошаговая детализация алгоритма ввода исходных данных: ввод алфавита, программы (последовательности команд).

Разработка программного кода, реализующего ввод данных.

Тестирование и отладка разработанных средств.

Документирование разработанного кода.

Этап 3.

Пошаговая детализация алгоритма интерпретации: ввод значения (входной строки) на ленту (в заданном алфавите) и её пошаговая обработка по заданной программе.

Разработка программного кода, реализующего ввод исходного значения на ленту, цикл интерпретации по заданной программе и вывод полученного результата.

Тестирование и отладка разработанных средств.

Документирование разработанного кода.

Примечание: для ввода данных и разбора строки можно (и нужно) использовать стандартные функции для работы со строками, имеющиеся в C#; в документацию (в отчёт) можно включить описание используемых средств, обоснование их выбора.

Оценивается программная реализация интерпретатора (50% оценки, см. табл. 2) и оформление отчёта (50% оценки).

Программная реализация оценивается по 10-балльной шкале. За *реализацию процедур каждого этапа* оценка выставляется по 10-баллной шкале (вес в результирующей оценке показан в табл. 2). Оценка *снижается*, если *в реализации процедур имеются ошибки*, не выявленные при тестировании (в зависимости от уровня (серьёзности ошибки)), имеются замечания к стилю программирования.

Отчёт о выполнении задания должен включать

1. *Постановку задачи* (с учётом выбранных вариантов решения, условий, требований).
2. *Описание сценариев* выполнения программы и тестов по критериям чёрного ящика.
3. *Описание* алгоритмов (с пошаговой детализацией).

4. Самодокументированный *исходный код программ*.
5. *Набор тестов*, соответствующий выбранному варианту решения.
6. *Описание и анализ средств C#*, которые могут быть использованы для реализации интерпретатора машины Тьюринга, обоснование выбора.
7. *Отчет об ошибках* – статистика допущенных ошибок допущенных при разработке программы, в виде таблицы, включающей *общее количество строк* в программе, количество *строк, содержащих комментарии и строк-разделителей*, количество *подпрограмм*, количество *тестов* и количество *ошибок трансляции* (синтаксических и семантических) и *выполнения* (ошибок в алгоритме, ошибок, связанных с выходом за заданные диапазоны значений и пр.).
   Количество допущенных и исправленных ошибок на оценке не отражается!

Оформление отчёта оценивается по 10-балльной шкале. Вес этой оценки в результирующей оценке за выполнение задания равен 0,5 (50% оценки).

Таблица 2. Критерии оценки программной реализации интерпретатора МТ

№	Базовые оценки		Дополнительные баллы*	
	Содержание работы	Вес	Содержание работы	Баллы
1	Ввод данных (ввод алфавита, программы МТ)	4	Возможность задать алфавит в соответствии с потребностями пользователя и необходимостью контролировать ввод данных для обработки на принадлежность введённых символов алфавиту	2
			Контроль ввода алфавита (исключение повторов и пр.) и возможность внесения изменений	1
			Ввод программы МТ (с учётом реализованных для пользователя возможностей ввода с клавиатуры, из файла)	3
			Контроль ввода программы МТ (принадлежность символов алфавиту, ограничений на число состояний, направлений сдвига головки и пр.) и возможность правки	2
			Визуализация введённых данных	1
			Возможность исправлений введённых данных	1
			Итого баллов за выполнение первого этапа задания:	**10**
2	Интерпретация МТ	6	Организация ввода исходных данных (значения – строки в заданном алфавите) с возможностью контроля и исправления ошибок	2
			Пошаговая интерпретация программы с контролем выполнения (зацикливание и пр.)	5
			Возможность повторения интерпретации программы для новых входных данных	1
			Наличие режима пошагового выполнения с визуализацией промежуточных результатов	2
			Итого баллов за выполнение второго этапа задания:	**10**
	Итого:	**10**		

Предусмотрена **публичная защита выполненного задания** (*представление работы на специально организованном семинаре на английском языке*), в ходе которой необходимо объяснить и обосновать представленное решение. Для презентации выбирается *одна из программ*, разработанных командой (интерпретатор НАМ или МТ), получившая лучшую оценку.

Для представления проекта готовится *презентация в MS Power Point* на *английском языке*, включающая:

1. Титульный слайд с указанием названия проекта и списка исполнителей.
2. Краткую постановку задачи, требования к исходным данным и результатам, условиям реализации программы и пр.
3. Результаты проведенного анализа средств C#, обоснование выбора типов данных, операций для реализации алгоритма.

4. Описание алгоритма интерпретации, его шагов.
5. Демонстрация работы программы (скриншоты с пояснениями) для различных сценариев.
6. Статистика ошибок, допущенных при разработке программы, в виде таблицы, включающей *общее количество строк* в программе, количество *строк, содержащих комментарии и строк-разделителей*, количество *подпрограмм*, количество *тестов* и количество *ошибок трансляции* (синтаксических и семантических) и *выполнения* (ошибок в алгоритме, ошибок, связанных с выходом за заданные диапазоны значений и пр.). Количество допущенных и исправленных ошибок на оценке не отражается!

По результатам защиты выставляется *оценка за представление задания*, которая будет добавлена к результирующей оценке.

Оценка *снижается*, если

- *не даётся анализ альтернативных вариантов* решения задачи;
- не даются ответы на вопросы *о внесении изменений в программу при изменениях в условиях задачи*;
- *не даны полные и чёткие ответы* на вопросы по представленной программной реализации.

Требования к оформлению отчёта о разработке программы

Отчёт оформляется в формате файла MS Word. Включает текстовое описание и иллюстрации (схемы), а также описание тестов. Отчёт прилагается к файлам проекта (с исходными кодами).

Отчет должен включать следующие главы:

1. ***Постановка задачи*** (формулируется задача, решаемая с помощью программы, и условия её решения: описание входных данных (форматов, ограничений на них), результата (в какой форме должен быть получен)). В данную главу можно включить теоретическую часть, чтобы показать, каким образом от исходной формулировки задачи перейти к использованию программы, можно также сослаться на аналоги, на другие разработки, которые были найдены (проанализировать их, имеющиеся плюсы и минусы, что вы использовали от них в своей программе – со ссылкой на источник).

2. ***Описание сценария использования программы и тестов*** (*критерии чёрного ящика*).

3. ***Описание алгоритма*** (оценивается пошаговая детализация описания алгоритма в соответствии с требованиями методологии нисходящего проектирования алгоритмов и структурного программирования, а также соответствие требованиям оформления):
   a) *текстовое* (псевдокод) – включается в исходный текст программы как комментарий (отдельно можно не описывать);
   b) *графическое*: блок-схема и структурограмма (для одного задания можно представить блок-схему, а для другого – структурограмму).
   c) *Описание тестов* (оценивается полнота тестов и их подготовка для проведения тестирования) – МГТ.

4. ***Описание порядка использования программы*** (действий пользователя для запуска программы и проведения тестирования на разработанном наборе тестов) – «руководство пользователя» – пошаговая инструкция для пользователя программы, описывающая все сценарии использования программы, иллюстрирующая все шаги её выполнения для различных сценариев на приведённых наборах тестов. В инструкцию можно включить скриншоты, иллюстрирующие интерфейс пользователя, выполнение всех операций от запуска программы и ввода данных до получения результатов и завершения работы с программой.

5. ***Исходный текст программы***, где оцениваются
   a) используемые типы данных (работа с массивами, строками, структурами, выбор типов переменных);
   b) реализация интерфейса (организация ввода и вывода (клавиатура/экран, текстовые файлы, обработка ошибок ввода, организация взаимодействия при выполнении алгоритмов, возможность изменения исходных данных и условий);
   c) реализация основного цикла интерпретатора (в том числе и обработка возможных ошибок);
   d) стиль программирования (наличие и качество комментариев, оформление кода, его читабельность).

Все файлы, необходимые для проверки исходного кода и тестирования программы, сохраняются в одну папку (папка именуется с использованием фамилии и инициалов автора программы). В эту же папку помещается файл отчёта.