Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Амурский государственный университет»

Кафедра математического анализа и моделирования

УЧЕБНО-МЕТОДИЧЕСКИЙ КОМПЛЕКС ДИСЦИПЛИНЫ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ И СЕТЕЙ

Основной образовательной программы по специальности 010501.65 – Прикладная математика и информатика

Благовещенск 2012 г.

УМКД разработал канд. техн. наук, доцент Рыженко Андрей Викторович

Рассмотрен и рекомендован на заседании кафедры
Протокол заседания кафедры от «___» _____ 201_ г. №___
Зав. кафедрой _____ / В.В.Сельвинский /

УТВЕРЖДЕН
Протокол заседания УМСС 010501 – Прикладная математика и информатика
от «___» _____ 201_ г. №___
Председатель УМСС _____ / В.В.Сельвинский /

СОДЕРЖАНИЕ

1 Рабочая программа дисциплины 4
1.1 Цели и задачи освоения дисциплины 4
1.2 Место дисциплины в структуре ООП ВПО 4
1.3 Требования к уровню освоения содержания дисциплины 4
1.4 Структура и содержание дисциплины 5
1.5 Содержание разделов и тем дисциплины 7
1.6 Самостоятельная работа 13
1.7 Образовательные технологии 13
1.8 Оценочные средства для текущего контроля успеваемости, промежуточной аттестации по итогам освоения дисциплины 13
1.9 Учебно-методическое и информационное обеспечение дисциплины 16
1.10 Материально-техническое обеспечение дисциплины 17
1.11 Рейтинговая оценка знаний студентов по дисциплине 18
2 Краткое изложение программного материала 19
3 Методические указания 27
3.1 Методические указания по изучению дисциплины 27
3.2 Методические указания по проведению практических (семинарских) занятий и лаботаторных работ 27
3.3 Методические указания по самостоятельной работе студентов 37
4 Контроль знаний 40
4.1 Текущий контроль знаний 40
4.2 Итоговый контроль 40
5 Интерактивные технологии и инновационные методы, используемые в образовательном процессе 40

1. РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ

1.1 ЦЕЛИ И ЗАДАЧИ ОСВОЕНИЯ ДИСЦИПЛИНЫ

Цели изучения дисциплины:

Дисциплина «Программное обеспечение вычислительных сетей и систем» является специальной дисциплиной в рамках подготовки специалистов в области системного и прикладного программирования по специальности 010501 «Прикладная математика и информатика».

Основными целями дисциплины «Программное обеспечение вычислительных сетей и систем» являются:

– изучение теоретических проблем, возникающих при разработке математического обеспечения вычислительных систем и сетей, а именно теория потоков в сетях, анализ сложности алгоритмов и сложности дискретных задач;

– изучение методов решения переборных задач;

– рассмотрение алгоритмов решения некоторых задач на параллельной архитектуре с произвольным доступом;

– изучение теоретических основ и приобретение практических навыков работы с программными комплексами вычислительных систем и сетей, ориентированными на решение различного рода задач;

– овладение основными приемами и методами программного управления средствами вычислительной техники;

– ознакомление с методами и стандартами разработки программного обеспечения.

Достижение указанных целей требует решения ряда задач.

Задачи изучения дисциплины:

– рассмотрение параллельных алгоритмов и методов решения различных классов задач на многопроцессорных вычислительных системах; изучение факторов, влияющих на время выполнения параллельных программ и на эффективность использования вычислительной мощности многопроцессорных вычислительных систем;

– овладение средствами описания и методами построения параллельных алгоритмов на примере задач сортировки данных, генерации псевдослучайных чисел и др.;

– рассмотрение динамической балансировки загрузки процессоров на примере параллельных алгоритмов интегрирования функций и решения систем линейных уравнений специального вида;

– освоение основных приемов решения прикладных задач по темам дисциплины;

– подготовка к поиску и анализу специальной научно-технической информации, необходимой для решения конкретных научно-исследовательских и прикладных задач, в том числе при выполнении междисциплинарных проектов;

– формирование социально-личностных качеств студентов: целеустремленности, организованности, трудолюбия, коммуникативности, готовности к деятельности в профессиональной среде, ответственности за принятие профессиональных решений.

1.2 МЕСТО ДИСЦИПЛИНЫ В СТРУКТУРЕ ООП ВПО

Дисциплина «Программное обеспечение вычислительных сетей и систем» является базовой дисциплиной цикла специальных дисциплин государственного образовательного стандарта высшего профессионального образования (ГОС ВПО) по специальности 010501.65 «Прикладная математика и информатика» (специалист), шифр дисциплины СД.Ф.3. Дисциплина базируется на знаниях, полученных в рамках курсов «Информатика», «Языки программирования и методы трансляции», «Практикум на ЭВМ», «Системное и прикладное программное обеспечение», «Проектирование и реализация языков программирования», «Операционные системы и сетевые технологии».

Знания, полученные при изучении дисциплины «Программное обеспечение вычислительных сетей и систем», могут быть использованы при подготовке к дипломной работе.

1.3 ТРЕБОВАНИЯ К УРОВНЮ ОСВОЕНИЯ СОДЕРЖАНИЯ ДИСЦИПЛИНЫ

В результате освоения дисциплины обучающийся должен демонстрировать следующие результаты образования:

знать: методы и стандарты разработки программного обеспечения, современные методы и средства программирования, архитектуру современных ЭВМ и базовые принципы построения средств вычислительной техники, базовые устройства современных вычислительных систем, параллельные алгоритмы решения некоторых задач, связанных с работой со списками и деревьями, понятия рекурсивных и рекурсивно перечислимых языков, сложностных классов языков и задач и их взаимосвязь между собой, приемы и методы разработки сетевых приложений, методы решения переборных задач (метод «ветвей и границ», рандомизированные алгоритмы, приближенные алгоритмы и др.);

уметь: решать задачи, возникающие в процессе сопровождения и эксплуатации программных средств, работать с пакетами программ и системами программирования, использовать встроенные функции, вести разработку, отладку, тестирование и документирование программного обеспечения, находить требуемую информацию в сети, решать в среде данных пакетов различного рода задачи и представлять результаты исследований, обосновать и определить вычислительную сложность алгоритма, применять современные методы и средства программирования, применять на практике, существующие на сегодняшний день, пакетах прикладных программ для решения различных прикладных и системных задач, использовать в работе пакеты прикладных программ для решения задач пользователя;

владеть: алгоритмами решения потоковых задач о максимальном потоке, алгоритмами планирования вычислений в многопроцессорных вычислительных системах; алгоритмами сортировки, средой программирования для создания программного обеспечения, инструментальными средствами создания серверной и клиентской часть сетевых приложений, методами сопровождения сетевых приложений.

1.4 СТРУКТУРА И СОДЕРЖАНИЕ ДИСЦИПЛИНЫ (МОДУЛЯ)

Общая трудоемкость дисциплины _136_ часов.

№ п/п	Раздел дисциплины (модуль)	Семестр	Неделя семестра	Виды учебной работы, включая самостоятельную работу студентов и трудоемкость (в часах)			Формы текущего контроля успеваемости *(по неделям семестра)* Форма промежуточной аттестации *(по семестрам)*
				Лекц.	Прак.	Самост.	
1	Теория потоков в сетях	7	1-2	3	2	0,5 0,5	Письменный опрос Конспект
2	Приложения потоковых алгоритмов	7	2-3	2	3	1 1	Реферат Отчет по индивид. заданию
3	Алгоритмы сортировки	7	3	1	2		Устный опрос
4	Распознающие алгоритмы. Полиномиально распознаваемые	7	4-5	4	1	0,5 0,5	Письменный опрос Конспект

	языки и класс *P*						
5	Полиномиальные проверяющие алгоритмы. Классы *NP* и *NPC*	7	6-7	4	1	0,5	Письменный опрос
6	Семь основных *NP*-полных задач. Класс *co-NP*	7	8-9	4	1	0,5 0,5	Письменный опрос Конспект
7	Задачи с числовыми параметрами. Псев-дополиномиальная сводимость. Сильная *NP*-полнота	7	10-11	4	1	1	Отчет по индивид. заданию
8	Сводимость по Тьюрингу. *NP*-трудные и *NP*-легкие задачи. Приближенные алгоритмы	7	12-13	4	1	0,5 0,5	Письменный опрос Конспект
9	Метод «ветвей и границ». Рандомизированные алгоритмы	7	14-15	3	2	0,5	Устный опрос Конспект
10	Алгоритмы параллельных вычислений. Параллельная машина с произвольным доступом	7	15-16	3	2	0,5 0,5 1	Письменный опрос Конспект Реферат
11	История развития индустрии программного обеспечения	8	1	2	–		Устный опрос
12	Оформление текста программ, основы законодательства о программном обеспечении	8	2	2	2	2 0,5	Типовое задание Конспект
13	Основные структуры данных и их работа в параллельном окружении	8	3	2	3	4 0,5 1	Типовое задание Письменный опрос Отчет по индивид. заданию
14	Безопасность программного обеспечения. Техника кодирования и основы криптографии. Моделирование угроз	8	4-5	4	1	3 0,5 1 0,5	Типовое задание Письменный опрос Реферат Конспект
15	Эволюция аппаратного обеспечения	8	6	1	–	0,5	Письменный опрос
16	Основы	8	6-7	3	1	2	Типовое задание

	параллельного программирования					0,5	Реферат
17	Технические аспекты написания параллельных программ	8	8	2	1	3 0,5 0,5	Типовое задание Реферат Письменный опрос
18	Объектно-ориентированное программирование и технические особенности ее реализации в языках программирования. Поддержка ООП в различных системах и средах	8	9-10	4	2	0,5 0,5 1	Письменный опрос Конспект Реферат
19	Технология аппаратной виртуализации и виртуальные машины	8	11	2	–		Устный опрос
20	Параллельные алгоритмы для многопроцессорных вычислительных систем	8	12	2	1	2	Типовое задание
21	Методы построения параллельных алгоритмов	8	13	2	2	1 1	Типовое задание Отчет по индивид. заданию
22	Сортировка данных на многопроцессорных вычислительных системах	8	14	2	1	1	Типовое задание
23	Методы параллельной генерации псевдослучайных чисел	8	15	2	1	1 0,5	Типовое задание Конспект
24	Параллельные алгоритмы интегрирования функций и решения систем линейных уравнений	8	16	2	1	1 0,5	Типовое задание Конспект

1.5 СОДЕРЖАНИЕ РАЗДЕЛОВ И ТЕМ ДИСЦИПЛИНЫ

1.5.1 Лекции.

1. Теория потоков в сетях

Основные понятия (сеть, поток, величина потока, разрез, величина разреза, увеличивающий путь, остаточная сеть). Алгоритм Форда-Фалкерсона решения задачи о максимальном потоке. Асимптотические обозначения (O, Ω, θ, o, ω) и их свойства (транзитивность, рефлексивность, симметричность, обращение). Теорема о максимальном

потоке и минимальном разрезе. Алгоритм Карзанова решения задачи о максимальном потоке. Задача о потоке минимальной стоимости. Алгоритм дефекта.

2. Приложения потоковых алгоритмов

Сведение задачи построения многопроцессорного расписания с прерываниями при заданных длительностях работ и директивных интервалах к задаче о максимальном потоке (алгоритм Танаева). Алгоритм упаковки для случая одинаковых директивных интервалов. Алгоритм Коффмана для случая одного процессора. Транспортная задача. Задача о назначениях. Задачи о кратчайшем и самом длинном путях, составление расписания при жестких директивных интервалах. Задача о паросочетаниях.

3. Алгоритмы сортировки

Понятие кучи, ее свойства. Преобразование массива в кучу. Сортировка с помощью кучи. Извлечение из массива максимального элемента и добавление элемента к отсортированному массиву с помощью кучи.

4. Распознающие алгоритмы. Полиномиально распознаваемые языки и класс P

Задачи распознавания свойств и их языки. Детерминированные машины Тьюринга. Детерминированная одноленточная машина Тьюринга. Временная сложность детерминированной машины Тьюринга. Рекурсивные и рекурсивно перечислимые языки и соотношение между ними. Полиномиально распознаваемые языки и класс *P*.

5. Полиномиальные проверяющие алгоритмы. Классы NP и NPC

Класс *NP*. Соотношение между классами *P* и *NP*. Существование экспоненциального проверяющего алгоритма для языков из *NP*. Полиномиальная сводимость. *NP*-полные языки, класс *NPC*. Теорема Кука. Методы доказательства *NP*-полноты.

6. Семь основных NP-полных задач. Класс co-NP

Доказательство *NP*-полноты задач выполнимость, 3-выполнимость, трехмерное сочетание, вершинное покрытие, клика, гамильтонов цикл, разбиение, расписание без прерываний для многопроцессорной системы. Класс *co-NP*. Структура классов *NP* и *co-NP*.

7. Задачи с числовыми параметрами. Псевдополиномиальная сводимость. Сильная NP-полнота

Задачи с числовыми параметрами. Псевдополиномиальные алгоритмы (задача о разбиение, задача рюкзак, задача многопроцессорное расписание без прерываний при фиксированном числе процессоров, задача об упаковке в контейнеры при фиксированном числе контейнеров). Сильная *NP*-полнота и методы ее доказательства. Псевдополиномиальный алгоритм решения задачи о разбиении. Сильная *NP*-полнота задачи расписание без прерываний для многопроцессорной системы, задачи коммивояжера и упаковка в контейнеры.

8. Сводимость по Тьюрингу. NP-трудные и NP-легкие задачи. Приближенные алгоритмы

Сводимость по Тьюрингу. Доказательство *NP*-трудности и *NP*-легкости некоторых задач. Приближенные алгоритмы (решения задач упаковка в контейнеры и расписание без прерываний для многопроцессорной системы) и оценки их погрешности. Невозможность существования полиномиального приближенного алгоритма с фиксированной погрешностью для некоторых *NP*-трудных задач. Приближенный полиномиальный алгоритм решения задачи коммивояжера с неравенством треугольника.

NP-трудные задачи (задача K-е по порядку множество). *NP*-эквивалентные задачи (оптимизационные варианты семи основных *NP*-полных задач, оптимизационная задача коммивояжера).

Приближенные полиномиальные алгоритмы решения *NP*-трудных задач (упаковка в контейнеры, рюкзак, коммивояжер (при выполнении неравенства треугольника), многопроцессорное расписание без прерываний); оценки их погрешности. Применение теории *NP*-полноты к отысканию приближенных решений.

9. Метод «ветвей и границ». Рандомизированные алгоритмы

Общее описание метода «ветвей и границ». Реализация метода «ветвей и границ» для решения задачи расписание без прерываний для многопроцессорной системы для случая различных процессоров. Рандомизированные алгоритмы для решения дискретных оптимизационных задач. Лемма Шварца. Задача проверки идентичности полиномов, задача о паросочетаниях в двудольном графе.

10. Алгоритмы параллельных вычислений. Параллельная машина с произвольным доступом

EREW-алгоритмы решения задач определения номера элемента в списке, параллельная обработка префиксов списка, вычисление глубины вершины в двоичном дереве. CRCW-алгоритм решения задачи определения корня для вершины двоичного леса. CRCW-алгоритм нахождения максимального элемента в массиве. Эффективная параллельная обработка префиксов. Моделирование CRCW-машины с помощью EREW-машины.

11. История развития индустрии программного обеспечения

Проблемы, возникающие при программировании и их решения. Основные требования к индивидуальным заданиям. Примеры индивидуальных и коллективных проектов. Техника написания программ, комментарии, сопровождение, жизненный цикл программ.

12. Оформление текста программ, основы законодательства о программном обеспечении

Основные требования к оформлению текстов программ. Оформление текста программ – написание комментариев, их типы и разновидности, содержание. Копирайты, откуда берутся права, отличия в законодательстве России и США. Хорошая и плохая практика написания текста программ – выбор имен для переменных, использование препроцессора и т.д.

13. Основные структуры данных и их работа в параллельном окружении

Понятие «структуры данных». Выбор структур данных для решения конкретных задач. Типовые элементы структур данных и связанных с ними алгоритмов, итераторы. Односвязный список и его использование в куче (heap) рантайма (библиотеке периода исполнения) языка Си, в функциях malloc и free. Проблемы использования структур данных в параллельном окружении на примере односвязного списка. Проблемы создания программ, использующих односвязные списки, в том числе в параллельном окружении. Двусвязные списки и очереди на их основе. Стек, его использование в программах на языке Си, поддержка рекурсии на его основе. Расположение данных в памяти. Использование структур памяти в параллельном окружении, операции синхронизации. Графы, деревья, использование их для компрессии данных. Хеш-таблицы. Разрешение коллизий с помощью цепочек. Хеш-функции (деление с остатком, умножение, универсальное хеширование). Хеш-таблица в параллельном окружении. Рекомендации по написанию параллельных программ.

14. Безопасность программного обеспечения. Техника кодирования и основы криптографии. Моделирование угроз

Безопасность программного обеспечение. Типовые проблемы, откуда они берутся. Написание программ без «дырок». Техника кодирования защищенных программ и типичные ошибки. Переполнение буфера, определение уровня доступа, работа с минимально возможными привилегиями, криптография и ее корректное применение, предохранение секретных данных, работа с входными данными, проблемы разных путей доступа к одним и тем же данным, запросов к базам данных и веб-страницам, а также проблемы поддержки интернационального ПО. Основные понятия криптографии, история. Симметричная и ассиметричная криптография, использование ее для безопасной почты. Безопасные и небезопасные алгоритмы (является ли секретный алгоритм безопасным?), генераторы случайных чисел. Устройство цифровой подписи. Как правильно применять криптографию для обеспечения безопасности. Способы взлома программ.

Моделирование угроз. Классификация опасностей STRIDE – Spoofing (подмена данных), Tampering (подделка и изменение содержания данных), Repudiation (незаконный отказ от проведенной операции), Information disclosure (разглашение информации), Denial of

service (отказ в обслуживании), Elevation of privilege (незаконное поднятие привилегий). Методика оценки риска DREAD – Damage potential (что может быть сломано), Reproducibility (повторяемость), Exploitability (пригодность угрозы для использования), Affected users (на каких пользователей повлияет), Discoverability (возможно ли детектировать факт использования).

15. Эволюция аппаратного обеспечения

Эволюция современного аппаратного обеспечения и ее влияние на программное обеспечение. Гипертредовые (многопоточные) и многоядерные процессоры, универсальные графические (GPU) процессоры и новые возможности по их использованию.

16. Основы параллельного программирования

Параллельное программирование. Параллельные программы – от работы с разделяемой памятью, использования массивнопараллельных компьютеров и до распределенных расчетов на многих физических компьютерах. Декомпозиция задач на параллельные куски. Закон Амдала. Геометрическая декомпозиция. Параллелизм данных, параллелизм кода. Паттерны параллельного программирования: параллелизм на уровне задач – декомпозиция задачи, «разделяй и властвуй» - декомпозиция задач и данных, геометрическая декомпозиция – декомпозиция данных, конвейерное исполнение – декомпозиция потока данных, «фронт волны» – декомпозиция данных с «многомерными» зависимостями. Пример типового шаблона программирования – пул нитей.

17. Технические аспекты написания параллельных программ

Технические аспекты написания параллельных программ на разделяемой памяти. Многопоточные программы. Работа с разделяемой памятью. Атомарность операций. Синхронизационные примитивы (низкоуровневые атомарные команды, критические секции, взаимоисключающая блокировка – mutex, рекурсивная блокировка – lock, блокировка чтения-записи – read/write lock, многопроцессорная блокировка – spin-lock, семафоры, барьеры). Принятие во внимание особенностей аппаратной платформы, например, последовательности операций записи в память и т.д. Тупики и взаимные блокировки. Неблокирующая синхронизация. Реализация одних примитивов через другие, относительная «мощность» примитивов. Задача о консенсусе как способ оценки примитивов.

18. Объектно-ориентированное программирование и технические особенности ее реализации в языках программирования. Поддержка ООП в различных системах и средах

Объектно-ориентированное программирование с понятийной точки зрения. История появления и возникновения понятий. Объекты как способ моделирования «параллельного внешнего мира» и формализации понятий. Объект как совокупность методов и данных, классы объектов, мета объекты. Объектная модель. Жизненный цикл объектов. Статические и динамические объекты. Раннее и позднее связывание, постоянно существующие объекты. Инкапсуляция и полиморфизм как базовые понятия ООП. Наследование и агрегирование как способы создания новых, более сложных, объектов, их достоинства и недостатки (на примере C++ и COM). Технические способы создания объектов в языках типа C++, представление их в компьютере.

Техническая реализация ООП в языке программирования C++ – конструкторы и деструкторы, представление методов через функции, расположение данных в памяти. Реализация понятия полиморфизма через таблицы виртуальных функций. Наследование и его реализация, связь с виртуальными функциями, абстрактные типы данных. Исключения, их техническая реализация и особенности применения в языке и ОС Windows.

Поддержка ООП в языке и системе программирования ObjectiveC, категории, сообщения и методы. Поддержка ООП в ОС Windows, отличия от поддержки ООП в языке программирования. Реализация основных понятий ООП в этих системах. Использование подходов ООП в разных условиях.

19. Технология аппаратной виртуализации и виртуальные машины

Технологии виртуализации и причины ее широкого распространения в последнее время. Типы виртуализации – виртуализация ОС и виртуализация оборудования. Преимущества и недостатки разных виртуализационных подходов с точки зрения потребителя. Технические основы виртуализации уровня ОС, контейнеры, уровни изоляции. Технология аппаратной виртуализации. Особенности работы кода прикладных программ внутри виртуальных машин. Новые возможности, предоставляемые виртуальными машинами и контейнерами – миграция, универсальность конфигурации, быстрота развертывания и обновления, удобства управления. Технические принципы организации виртуальных машин. Монитор виртуальных машин, аппаратная поддержка виртуализации, бинарная трансляция и другие подходы. Накладные расходы в реализации поддержки системы виртуальных машин. Паравиртуализация. Выбор оптимального типа виртуализации в разных условиях.

20. Параллельные алгоритмы для многопроцессорных вычислительных систем

Примеры задач, решаемые на многопроцессорных системах: вычислительная газовая динамика, молекулярная динамика, символьные вычисления и др. Основные требования к алгоритмам для суперкомпьютеров. Эффективность и ускорение. Кризис эффективности. Закон Амдала. Средства описания параллельных алгоритмов.

Многопроцессорные вычислительные системы с общей и с распределенной памятью; канал передачи данных и его свойства; синхронная и асинхронная передача данных; семафоры и операции над ними. Модели программ на общей и распределенной памяти. Ускорение и эффективности параллельных алгоритмов, свойство внутреннего параллелизма алгоритма. Пример алгоритма, обладающего низкой эффективностью, но высоким быстродействием (нахождение суммы конечного ряда).

21. Методы построения параллельных алгоритмов

Простейшие методы построения параллельных алгоритмов: метод геометрического и конвейерного параллелизма, метод коллективного решения. Их свойства. Проблемы статической и динамической балансировки загрузки процессоров. Алгоритм диффузной балансировки загрузки. Пример задачи, для сокращения времени решения которой необходимо создание нового алгоритма (сложение длинных чисел).

22. Сортировка данных на многопроцессорных вычислительных системах

Построение эталонного последовательного алгоритма сортировки. Пример «плохого» последовательного, но быстрого параллельного метода сортировки. Оценка времени выполнения алгоритма в наилучшем, среднем и наихудшем случае. Зависимость времени сортировки от вида исходной последовательности.

Минимальные сетей сортировки, сети четно-нечетное слияния. Построение на основе сетей сортировки параллельного алгоритма сортировки данных. Определение его характеристик при реализации на системах с общей и с распределенной памятью. Оценка времени выполнения сортировки. Оценка необходимого объема оперативной памяти. Обсуждение возможности сокращения объема передаваемых данных и требуемой оперативной памяти.

23. Методы параллельной генерации псевдослучайных чисел

Задачи, решаемые с использованием последовательностей псевдослучайных чисел (ПСЧ). Параллельные алгоритмы решения таких задач. Требования к генераторам ПСЧ для многопроцессорных систем. Параллельные алгоритмы генерации ПСЧ. Тестирование полученных последовательностей.

24. Параллельные алгоритмы интегрирования функций и решения систем линейных уравнений

Параллельные алгоритмы интегрирования функции одной переменной, построенные на основе методов геометрического параллелизма и коллективного решения. Адаптивный последовательный алгоритм интегрирования. Параллельный алгоритм интегрирования с динамической балансировкой загрузки для многопроцессорных систем с общей памятью.

Задачи, приводящие к появлению систем линейных уравнений с трехдиагональной матрицей коэффициентов. Параллельные алгоритмы решения трехдиагональных систем линейных алгебраических уравнений. Итерационные методы.

1.5.2 Семинарские (практические) занятия.

7 семестр

При выполнении практических работ по данному курсу студенты должны ознакомиться с основными принципами построения, функционирования и использования современных программных комплексов, овладеть основными приемами и методами программного управления средствами вычислительной техники.

Практическое занятие (семинар) 1. Виртуальные машины. Основы работы с Virtual PC, Virtual Box. Опции, настройка, возможности. Установка и настройка гостевых операционных систем Linux, Windows. Проверка сетевого взаимодействия. Добавление роли «контроллер домена».

Практическое занятие (семинар) 2. Инструментальные средства развертывания операционных систем. Применение утилиты Sysprep для создания образа.

Практическое занятие (семинар) 3. Удаленная установка операционной системы. Установка и настройка службы RIS. Настройка сервера удаленной установки.

Практическое занятие (семинар) 4. Групповая политика и настройка брандмауэра. Служба Telnet. Настройка порта. Настройка исключений.

Практическое занятие (семинар) 5. Безопасность операционной системы, настройка защитника Windows. Сканирование компьютера и обнаружение подозрительных действий.

Практическое занятие (семинар) 6. Средства анализа защищенности MBSA 2.0. Работа в автономном режиме. Сканирование и сгенерировать отчета.

Практическое занятие (семинар) 7. Сканирование уязвимостей операционной системы. Создание профиля. Просмотр и исправление обнаруженных уязвимостей. Сканирование удаленного хоста. Создание отчетов и расписаний сканирования уязвимостей.

Практическое занятие (семинар) 8. Установка агента PowerChute Business Edition, настройка параметров конфигурационного профиля, проверка получения уведомлений по электронной почте. Защита рефератов, доклады, выступления. Подготовка к зачету.

8 семестр

Практическое занятие (семинар) 1-2. Реализация структур данных на языке Си. Итератор.

Практическое занятие (семинар) 3-4. Абстрактные типы данных. Библиотеки функций Win32 API, POSIX Threads. Параллельный суппорт.

Практическое (семинарское) занятие 5-6. Семейство протоколов TCP/IP. Сокеты (sockets) в UNIX и основы работы с ними. История семейства протоколов TCP/IP. Архитектуре семейства протоколов TCP/IP. Уровень сетевого интерфейса. Уровень Internet. Протоколы IP, ICMP, ARP, RARP. Internet-адреса. Транспортный уровень. Протоколы TCP и UDP. UDP и TCP сокеты (sockets). Адресные пространства портов. Понятие encapsulation. Уровень приложений/программ. Использование модели клиент-сервер при изучении сетевого программирования. Организация связи между удаленными процессами с помощью датаграмм. Сетевой порядок байт. Функции преобразования IP-адресов. Организация связи между процессами с помощью установки логического соединения. Использование интерфейса сокетов для других семейств протоколов. Файлы типа «сокет».

Практическое занятие 7. Сетевое оборудование и программное обеспечение. Мониторинг и анализ сетей.

Практическое занятие 8. Итоговое занятие. Защита индивидуального типового задания, доклады.

1.6 САМОСТОЯТЕЛЬНАЯ РАБОТА

Самостоятельная работа студентов заключается в проработке лекционного материала, выполнении заданий по каждой теме практического семинарского занятия, выполнении индивидуальных домашних заданий, подготовки рефератов и докладов. На углубленное самостоятельное изучение выносятся следующие темы:

1) Алгоритм Коффмана.

2) Задачи о назначениях, о кратчайшем и самом длинном путях, составление расписания при жестких директивных интервалах.

3) Задача о паросочетаниях.

4) Функции управления программной инженерией.

5) Интегрированные среды разработки программного обеспечения. CASE-технологии.

6) Этапы жизненного цикла разработки программного обеспечения.

7) История развития инструментальных средств поддержки разработки программного обеспечения.

8) Многопроцессорное планирование.

9) Аппаратно-программные средства поддержки мультипрограммирования.

10) Временная сложность детерминированной машины Тьюринга.

11) Моделирование CRCW-машины с помощью EREW-машины.

12) Архитектура SNA.

13) Многопроцессорные блокировки.

14) Задача о консенсусе.

15) Поддержка ООП в языке и системе программирования ObjectiveC.

16) Монитор виртуальных машин, аппаратная поддержка виртуализации.

17) Безопасность удаленного доступа.

18) Технология паравиртуализации.

19) Виртуальные частные сети, принципы действия. Безопасность VPN.

20) Сети с тонкими клиентами. Оборудование тонкого клиента.

№ п/п	№ раздела (модуля)	Форма (вид) самостоятельной работы	Трудоёмкость в часах
1	1-24	Подготовка к защите заданий семинаров	5
2	1-24	Подготовка к письменным опросам	5
3	1-24	Подготовка реферата	5
4	1-24	Подготовка конспекта лекций	5
5	1-24	Выполнение типового индивидуального задания	20
6		***ИТОГО САМОСТОЯТЕЛЬНАЯ РАБОТА***	***40***

1.7 ОБРАЗОВАТЕЛЬНЫЕ ТЕХНОЛОГИИ

Занятия по дисциплине «Программное обеспечение вычислительных сетей и систем» проводятся в форме лекций, семинаров – практических занятий, консультаций; предусмотрена также самостоятельная работа студентов, состоящая из написания конспектов по предложенным темам, подготовки к зачету и экзамену, выполнении практических заданий на ПК, выполнении типового индивидуального задания, написании реферата и доклада.

1.8 ОЦЕНОЧНЫЕ СРЕДСТВА ДЛЯ ТЕКУЩЕГО КОНТРОЛЯ УСПЕВАЕМОСТИ, ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ ПО ИТОГАМ ОСВОЕНИЯ ДИСЦИПЛИНЫ

Оценочные средства состоят из вопросов к экзамену, итоговых семестровых тестов, вариантов индивидуальных самостоятельных домашних работ, проверочных опросов по пройденному теоретическому материалу, самостоятельных практических заданий.

На семинарах студенты разбирают и решают задачи, выполняют практические задания, указанные преподавателем, повторяют основные понятия, рассмотренные на лекциях. В каждом семестре предусмотрены контрольные и самостоятельные работы, индивидуальные практические задания, итоговый экзамен или зачет.

Теоретические вопросы к экзамену

1. Основные понятия потоков в сетях (сеть, поток и его величина, разрез и его величина, увеличивающий путь). Алгоритм Форда-Фалкерсона решения задачи о максимальном потоке.
2. Вычислительная сложность и оценка алгоритма Форда-Фалкерсона. Асимптотические обозначения. Понятие остаточной сети. Другой алгоритм вариант построения алгоритма Форда-Фалкерсона.
3. Разрез и его пропускная способность. Теорема о максимальном потоке и минимальном разрезе.
4. Слоистая сеть. Алгоритм Карзанова решения задачи о максимальном потоке.
5. Обоснование и анализ сложности алгоритма Карзанова.
6. Задача планирования вычислений в многопроцессорных системах. Сведение задачи построения многопроцессорного расписания с прерываниями при заданных длительностях работ и директивных интервалах к задаче о максимальном потоке.
7. Теорема Танаева.
8. Понятие кучи. Сортировка массива с помощью кучи. Оценка сложности алгоритма.
9. Задачи распознавания свойств и языки. Детерминированная одноленточная машина Тьюринга.
10. Рекурсивные и рекурсивно перечислимые языки и их свойства.
11. Полиномиально распознаваемые языки и класс P.
12. Класс NP.
13. Соотношение между классами P и NP. Существование экспоненциального проверяющего алгоритма для языков из класса NP.
14. Полиномиальная сводимость. Полиномиально эквивалентные языки.
15. Класс NPC. Способы доказательства NP-полноты.
16. Доказательство NP-полноты задачи выполнимость.
17. Доказательство NP-полноты задач 3-выполнимость и вершинное покрытие.
18. Доказательство NP-полноты задач клика, расписание для многопроцессорной системы без прерываний и переключений.
19. Класс co-NP. Структура классов NP и co-NP.
20. Задачи с числовыми параметрами. Псевдополиномиальные алгоритмы.
21. Сильная NP-полнота и методы ее доказательства. Псевдополиномиальный алгоритм решения задачи о разбиении.
22. Сильная NP-полнота задачи расписание для многопроцессорной системы без прерываний и переключений.
23. Сводимость по Тьюрингу. Доказательство NP-трудности и NP-легкости некоторых задач.
24. Приближенные алгоритмы (решения задач упаковка в контейнеры и расписание без прерываний для многопроцессорной системы) и оценки их погрешности.
25. Невозможность существования полиномиального приближенного алгоритма с фиксированной погрешностью для некоторых NP-трудных задач.
26. Приближенный полиномиальный алгоритм решения задачи коммивояжера с неравенством треугольника.
27. Метод «ветвей и границ» и его реализация для решения задачи расписание для многопроцессорной системы без прерываний и переключений.
28. Рандомизированный алгоритм решения задачи об идентичности полиномов и задачи о паросочетаниях.

29. EREW-алгоритмы решения задач определения номера элемента в списке, параллельная обработка префиксов.

30. EREW-алгоритмы решения задач вычисление глубины вершины в двоичном дереве.

31. CRCW-алгоритм решения задач определения корня для каждой вершины двоичного леса и нахождения максимального элемента в массиве.

32. Моделирование CRCW-машины с помощью EREW-машины. Эффективная параллельная обработка префиксов.

33. Классификация параллельных архитектур и средства параллельного программирования.

34. Оформление текста программного обеспечения, основные требования.

35. Структуры данных. Типовые элементы структур данных, итераторы. Проблемы использования в параллельном контексте.

36. Односвязные списки. Проблемы использования в параллельном окружении.

37. Структуры данных массив, стек, очередь. Применение. Проблемы использования в параллельном окружении.

38. Ошибки в программном обеспечении, классификация, способы разрешения. Множество. Проблемы использования структуры данных в параллельном окружении.

39. Деревья. Проблемы использования структуры данных в параллельном окружении.

40. Хэш-таблицы. Проблемы использования в параллельном окружении.

41. Безопасное программное обеспечение. Типичные ошибки, кодирование защищенных программ.

42. Использование криптографии при разработке программного обеспечения.

43. Моделирование угроз. Классификация опасностей. Методы оценки риска программного обеспечения.

44. Влияние развития аппаратного обеспечения на программное обеспечение

45. Многопоточные программы. Работа с разделяемой памятью. Атомарность операций. Синхронизационные примитивы. Тупики и взаимные блокировки.

46. Задачи решаемые на многопроцессорных системах. Кризис эффективности. Закон Амдала. Требования к параллельным алгоритмам для суперкомпьютеров. Оценка их эффективности и ускорения на примере задачи суммы ряда.

47. Многопроцессорные системы с общей и с распределенной памятью. Модели программ с общей и распределенной памятью. Методы передачи данных. Семафоры.

48. Методы построения параллельных алгоритмов. Накладные расходы. Методы статической балансировки загрузки процессоров и их свойства.

49. Методы построения параллельных алгоритмов. Методы динамической балансировки загрузки процессоров и их свойства.

50. Сортировка данных с точки зрения многопроцессорных вычислительных систем. Алгоритмы, оценка времени выполнения.

51. Сортировка данных с точки зрения многопроцессорных вычислительных систем. Сети сортировки, четно-нечетное слияние. Параллельные алгоритмы. Оценка времени и объема памяти.

52. Параллельная генерация псевдослучайных чисел. Примеры задач. Генераторы псевдослучайных чисел, требования для многопроцессорных систем. Параллельные алгоритмы.

53. Параллельные алгоритмы интегрирования функций.

54. Параллельные алгоритмы решения систем линейных уравнений.

Примерная тематика типового индивидуального домашнего задания по курсу:

1. Задача о максимальном потоке.
2. Задача о потоке минимальной стоимости.
3. Задача планирования вычислений в многопроцессорных системах.
4. Транспортная задача.

5. Задача о назначениях.
6. Задачи о кратчайшем и самом длинном пути.
7. Задача о паросочетаниях в двудольном графе.
8. Задачи сортировки.
9. Задача проверки идентичности полиномов.
10. Задача выполнимость.
11 Задача 3-выполнимость.
12. Задача трехмерное сочетание.
12 Задача вершинное покрытие.
13. Задача клика.
14. Задача гамильтонов цикл.
15. Задача о разбиение.
16. Задача коммивояжера.
17. Задача упаковка в контейнеры.
18. Задача о рюкзаке.

1.9 УЧЕБНО-МЕТОДИЧЕСКОЕ И ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ (МОДУЛЯ)

а) Основная литература:

1. Абрамов, С.А. Лекции о сложности алгоритмов: учебное пособие / С.А. Абрамов. – М.: МЦНМО, 2009. – 256 с. (ЭБС Университетская библиотека online)

2. Лацис, А.О. Параллельная обработка данных [Текст] : учебн. пособие для студ. вузов / А.О. Лацис. – М. : Издательский центр «Академия», 2010. – 336 с. – (Университетский учебник. Сер. Прикладная математика и информатика)

3. Чекмарев, Ю.С. Вычислительные системы, сети и коммуникации [Электронный ресурс] : учеб. пособие / Ю.В. Чекмарев. – М.: ДМК Пресс, 2009. – 184 с. (ЭБС Университетская библиотека online)

б) Дополнительная литература:

1. Брауде, Эрик Дж. Технология разработки программного обеспечения [Текст] / Э. Д. Брауде. - СПб. : Питер, 2004. - 655 с. : ил. - (Классика computer science).

2. Грекул, В.И. Проектирование информационных систем [Текст] : курс лекций: учеб. пособие / В. И. Грекул, Г. Н. Денищенко, Н. Л. Коровкина. - М. : Интернет- Ун-т Информ. Технологий, 2005. - 300 с.

3. Искусство взлома и защиты систем [Текст] / Дж. Козиол [и др.] ; пер. англ. Е. Матвеева. - СПб. : Питер, 2006. - 416 с.

4. Кормен Т. Алгоритмы : построение и анализ / Т. Кормен, Ч. Лейзерсон, Р. Ривест ; пер. с англ., ред. А. Шеня. -2-е изд., стер.. - М.: БИНОМ. Лаб. знаний; М.: МЦНМО, 2004. - 956 с.

5. Орлов, С.А. Технология разработки программного обеспечения [Текст] : разработка сложных систем : Учеб. пособие : Рек. Мин. обр. РФ / С.А. Орлов. - 2-е изд. – СПб.: Питер, 2003, 2004. – 474 с.

6. Пятибратов А.П. Вычислительные системы, сети и телекоммуникации : Учебник: Рек. Мин. обр. РФ/ А.П. Пятибратов, Л.П. Гудыно, А.А. Кириченко; Под ред. А.П. Пятибратова. -2-е изд., перераб. и доп.. -М.: Финансы и статистика, 2003. -510 с.

7. Пятибратов, А.П. Вычислительные машины, сети и телекоммуникационные системы : учебно-методический комплекс / А.П. Пятибратов. – М.: Евразийский открытый институт, 2009. – 292 с. (ЭБС Университетская библиотека online)

8. Танаев, В.С. Теория расписания. Одностадийные системы [Текст] / Танаев В.С., Гордон В.С., Шафранский Я.М. – М. : Наука, 1984. – 384 с.

9. Сигал, И.Х. Введение в прикладное дискретное программирование: модели и вычислительные алгоритмы : учебное пособие / И.Х. Сигал, А.П. Иванова. – М.: Физматлит, 2007. – 305 с. (ЭБС Университетская библиотека online)

10. Ульянов, М.В. Ресурсно-эффективные компьютерные алгоритмы. Разработка и анализ : учебное пособие М.В. Ульянов. – М.: Физматлит, 2008. – 303 с. (ЭБС Университетская библиотека online)

в) Периодические издания:

1. Communications of the ACM.
2. Informations Processing & Management.
3. PC magazine. Персональный компьютер сегодня.
4. Автоматика и вычислительная техника. Сводн. том.
5. Информационные технологии и вычислительные системы.
6. КомпьютерПресс.
7. Мир ПК + DVD.

г) Программное обеспечение и Интернет-ресурсы:

№	Наименование ресурса	Краткая характеристика
1	http://www.iqlib.ru	Интернет-библиотека образовательных изданий, в которой собраны электронные учебники, справочные и учебные пособия. Удобный поиск по ключевым словам, отдельным темам и отраслям знания
2	http://www.citforum.ru/	Электронная библиотека on-line статей по информационным технологиям. Удобный поиск по разделам, отдельным темам
3	http://www.twirpx.com/	Учебно-образовательная литература, содержащая DjVu-файлы, PDF-файлы, DOC-файлы, по различным дисциплинам
4	http://www.elibrary.ru/	Научная электронная библиотека eLIBRARY.RU – это крупнейший российский информационный портал в области науки, технологий, медицины и образования, содержит рефераты и полные тексты более 144 млн. научных статей и публикаций. На платформе eLIBRARY.RU доступны электронные версии более 2200 российских научно-технических журналов, в том числе более 1100 журналов в открытом виде
5	http://www.biblioclub.ru/	Электронная библиотечная система «Университетская библиотека – online»: специализируется на учебных материалов для ВУЗов по научно-гуманитарной тематике, а также содержит материалы по точным и естественным наукам

1.10 МАТЕРИАЛЬНО-ТЕХНИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ (МОДУЛЯ)

Лекции проводятся в стандартной аудитории (ауд.338а), оснащенной в соответствии с требованиями преподавания теоретических дисциплин, включая мультимедиа-проектор.

Практические (семинарские) занятий проводится в лабораториях кафедры (ауд. 327, 329), в которых установлены операционные системы Windows, Linux, Virtual PC и среда разработки Microsoft Visual Studio 2008.

1.11 РЕЙТИНГОВАЯ ОЦЕНКА ЗНАНИЙ СТУДЕНТОВ ПО ДИСЦИПЛИНЕ

Рейтинговая оценка знаний студентов проводится в соответствии с положением о балльно-рейтинговой системе оценки знаний студентов АмГУ и положением кафедры МАиМ по дисциплине.

Зачет сдается в конце семестра. Экзамен проводится в устной форме, экзаменационный билет включает два теоретических вопроса и тестовое задание.

Текущий контроль включает в себя тестовые письменные задания, аудиторные самостоятельные работы, индивидуальные домашние задания, семинарские и домашние практические задания, самостоятельную работу по изучению теоретического материала, промежуточное и итоговое тестирование.

Условия начисления премиальных баллов за внеаудиторную работу
(олимпиады, конференции, доклады, рефераты, научные кружки):

1) Подготовка и проведение доклада – до 5 баллов.

2) Участие в олимпиаде, конференции (с хорошим результатом) – до 10 баллов (до 20 баллов в случае победы в региональных, всероссийских и международных мероприятиях).

Начисление штрафных баллов:

За каждый пропуск занятий без уважительной причины из суммы баллов вычитается 1 балл.

Допуск к экзамену – 30 баллов.

Учебная дисциплина «Программное обеспечение вычислительных систем и сетей» относится к категории дисциплин с экзаменом (восьмой семестр) и оценивается в 100 баллов за семестр. Пересчет рейтинговой оценки дисциплины проводится по шкале:

от 91 до 100 баллов – «отлично»;

от 75 до 90 баллов – «хорошо»;

от 51 до 74 баллов – «удовлетворительно»;

50 баллов и меньше – «неудовлетворительно».

Рейтинговая оценка складывается из баллов, набранных по текущему контролю, баллов, набранных за экзамен, и премиальных баллов. Из итоговой суммы вычитаются штрафные баллы за пропуски занятий без уважительной причины.

Бальная структура оценки дисциплины

№	Виды работ	Максимальное количество баллов
7 семестр		
1	Письменные опросы (тестирование)	15
2	Практические (семинарские) работы	30
3	Активность на занятиях	10
4	Реферат	15
5	Зачет	30
	Итого:	100
8 семестр		
1	Письменные опросы (тестирование)	15
2	Практические (семинарские) работы	20
3	Активность на занятиях	5
4	Индивидуальное задание	20
5	Экзамен	40
	Итого:	100

2. КРАТКОЕ ИЗЛОЖЕНИЕ ПРОГРАММНОГО МАТЕРИАЛА

1. Теория потоков в сетях

Основные понятия (сеть, поток, величина потока, разрез, величина разреза, увеличивающий путь, остаточная сеть). Алгоритм Форда-Фалкерсона решения задачи о максимальном потоке. Асимптотические обозначения (O, Ω, θ, o, ω) и их свойства (транзитивность, рефлексивность, симметричность, обращение). Теорема о максимальном потоке и минимальном разрезе. Алгоритм Карзанова решения задачи о максимальном потоке. Задача о потоке минимальной стоимости. Алгоритм дефекта.

2. Приложения потоковых алгоритмов

Сведение задачи построения многопроцессорного расписания с прерываниями при заданных длительностях работ и директивных интервалах к задаче о максимальном потоке (алгоритм Танаева). Алгоритм упаковки для случая одинаковых директивных интервалов. Алгоритм Коффмана для случая одного процессора. Транспортная задача. Задача о назначениях. Задачи о кратчайшем и самом длинном путях, составление расписания при жестких директивных интервалах. Задача о паросочетаниях.

Ключевые вопросы: 1) Как называется множество дуг и узлов? 2) Что представляет собой поток в сети? 3) Как нызывается разбиение потока на две части? 4) Как нызывается сумма пропускных способностей рёбер разреза? 5) Оценить конечное число операций алгоритма Форда-Фалкерсона? 6) Какие значения могут быть элементами матрицы инцидентности? 7) Как нызывается путь из вершины в сток, содержащий хотя бы одну насыщенную дугу? 8) В каком случае алгоритм Форда-Фалкерсона может работать бесконечно? 9) Какой величиной сверху ограничена величина произвольного потока в сети? 10) Какое количество раз обрабатывается насыщенная дуга при нахождении тупикового потока? 11) Для чего применяется алгоритм Карзанова? 12) Оценить общее количество операций при выполнении алгоритма Карзанова?

3. Алгоритмы сортировки

Понятие кучи, ее свойства. Преобразование массива в кучу. Сортировка с помощью кучи. Извлечение из массива максимального элемента и добавление элемента к отсортированному массиву с помощью кучи.

Ключевые вопросы: 1) Что относится к характеристикам работы в многопроцессорном расписании? 2) Что следует указывать в многопроцессорном расписании для каждой работы? 3) Какое название носит расписание, при котором каждая работа получает в точности определенное время процессора (длительность), и выполняется в директивном интервале? 4) Укажите оценку количество операций алгоритма упаковки? 5) Какое количество памяти требуется для реализации алгоритма упаковки? 6) Какие узлы применяются в сети при использовании алгоритма Танаева? 7) Какой алгоритм необходимо применить к сети в алгоритме Танаева, если все выходные дуги насыщены? 8) Какую величину составляет максимальное количество прерываний и переключений в алгоритме Танаева? 9) Как называются вершины с номерами [n/2]+1, …, n в двоичном дереве с n вершинами? 10) Как называется двоичное дерево, в котором значение в любой вершине больше (меньше), чем значения ее потомков? 11) Оценить количество операций необходимое для создания кучи из неупорядоченного массива входных данных? 12) Укажите достоинства алгоритма пирамидальной сортировки?

4. Распознающие алгоритмы. Полиномиально распознаваемые языки и класс P

Задачи распознавания свойств и их языки. Детерминированные машины Тьюринга. Детерминированная одноленточная машина Тьюринга. Временная сложность детерминированной машины Тьюринга. Рекурсивные и рекурсивно перечислимые языки и соотношение между ними. Полиномиально распознаваемые языки и класс *P*.

5. Полиномиальные проверяющие алгоритмы. Классы NP и NPC

Класс *NP*. Соотношение между классами *P* и *NP*. Существование экспоненциального проверяющего алгоритма для языков из *NP*. Полиномиальная сводимость. *NP*-полные языки, класс *NPC*. Теорема Кука. Методы доказательства *NP*-полноты.

Ключевые вопросы: 1) Чем характеризуется задача распознавания свойств? 2) Каким образом обозначается длина слова x в задаче распознавания свойств? 3) Укажите составляющие части машины Тьюринга? 4) С помощью чего формируются правила перехода? 5) Если классы P и NP равны, то за какое время можно решить любую задачу из класса NP? 6) Как называется рекурсивное подмножество множества всех возможных слов в алфавите формального языка? 7) По каким из операций замкнуты рекурсивные языки? 8) Что представляет собой рекурсивно перечислимое подмножество множества всевозможных слов над алфавитом языка? 9) От чего зависит сложность функции в классе P, вычисляемой некоторой машиной Тьюринга? 10) За какое время можно решить любую задачу, принадлежащую классу NP? 11) Какие задачи относятся к NP-полным задачам? 12) Приведите примеры алгоритмов класса P?

6. Семь основных NP-полных задач. Класс co-NP

Доказательство *NP*-полноты задач выполнимость, 3-выполнимость, трехмерное сочетание, вершинное покрытие, клика, гамильтонов цикл, разбиение, расписание без прерываний для многопроцессорной системы. Класс *co-NP*. Структура классов *NP* и *co-NP*.

7. Задачи с числовыми параметрами. Псевдополиномиальная сводимость. Сильная NP-полнота

Задачи с числовыми параметрами. Псевдополиномиальные алгоритмы (задача о разбиение, задача рюкзак, задача многопроцессорное расписание без прерываний при фиксированном числе процессоров, задача об упаковке в контейнеры при фиксированном числе контейнеров). Сильная *NP*-полнота и методы ее доказательства. Псевдополиномиальный алгоритм решения задачи о разбиении. Сильная *NP*-полнота задачи расписание без прерываний для многопроцессорной системы, задачи коммивояжера и упаковка в контейнеры.

Ключевые вопросы: 1) Приведите примеры NP-полных задач? 2) Какие операции применяются в формулах в задаче выполнимости? 3) Какова вычислительная сложность задачи выполнимости булевых формул во 2-конъюнктивной нормальной форме? 4) Каков размер вершинного покрытия с 10 вершинами? 5) В чем суть задачи о вершинном покрытии? 6) Какой класс лежит на пересечении классов NP и co-NP? 7) Если в графе степени любых двух несмежных вершин не меньше общего числа вершин в графе, то такой граф считается..? 8) Какой класс относиться к подклассам эквивалентности класса NP? 9) Если задача лежит одновременно в классе NP и в классе co-NP, то она лежит..? 10) В каком классе лежит задача линейного программирования? 11) Дайте определение клики. Чем определяется размер клики? 12) Что необходимо найти в графе в оптимизационной задаче о клике?

8. Сводимость по Тьюрингу. NP-трудные и NP-легкие задачи. Приближенные алгоритмы

Сводимость по Тьюрингу. Доказательство *NP*-трудности и *NP*-легкости некоторых задач. Приближенные алгоритмы (решения задач упаковка в контейнеры и расписание без прерываний для многопроцессорной системы) и оценки их погрешности. Невозможность существования полиномиального приближенного алгоритма с фиксированной погрешностью для некоторых *NP*-трудных задач. Приближенный полиномиальный алгоритм решения задачи коммивояжера с неравенством треугольника.

NP-трудные задачи (задача K-е по порядку множество). *NP*-эквивалентные задачи (оптимизационные варианты семи основных *NP*-полных задач, оптимизационная задача коммивояжера).

Приближенные полиномиальные алгоритмы решения *NP*-трудных задач (упаковка в контейнеры, рюкзак, коммивояжер (при выполнении неравенства треугольника), многопроцессорное расписание без прерываний); оценки их погрешности. Применение теории *NP*-полноты к отысканию приближенных решений.

Ключевые вопросы: 1) Чему полагается равной функция максимума, если в индивидуальной задаче нет чисел? 2) Как называется задача, если в ней нет полинома длины, который сверху ограничивал функцию максимума? 3) От чего зависит полином, ограничивающий вычислительную сложность псевдополиномиального алгоритма? 4) Что можно сказать длина результата каждой операции, если в алгоритме присутствуют только операции сложения и вычитания? 5) Какое максимальное значение можно выбрать в качестве функции максимума при решении задачи о максимальном потоке с помощью псевдополиномиального алгоритма? 6) Какие операции используются в алгоритме Форда-Фалкерсона? 7) Дайте определение задаче являющейся NP-полной в сильном смысле, приведите примеры такого класса задач. 8) Что относится к элементам входа для задачи РМПС? 9) Приведите примеры дискретных оптимизационных задач? 10) Что необходимо учитывать при использовании приближенного алгоритма? 11) К какому классу следует отнести задачу П, если она сводится по Тьюрингу к оптимизационной задаче? 12) Какие алгоритмы можно используются для приближенного решения оптимизационной задачи многопроцессорного расписания?

9. Метод «ветвей и границ». Рандомизированные алгоритмы

Общее описание метода «ветвей и границ». Реализация метода «ветвей и границ» для решения задачи расписание без прерываний для многопроцессорной системы для случая различных процессоров. Рандомизированные алгоритмы для решения дискретных оптимизационных задач. Лемма Шварца. Задача проверки идентичности полиномов, задача о паросочетаниях в двудольном графе.

Ключевые вопросы: 1) Что необходимо найти в задаче о вершинном покрытии? 2) Какое название носит цикл в сети, который проходит ровно один раз через каждый узел? 3) Какое количество ребер в дереве с n вершинами? 4) Что называется остовным деревом? 5) Приведите общее описание метода ветвей и границ, к какому классу методов его следует отнести? 6) Укажите основные этапы метода ветвей и границ. 7) Пусть граф имеет 100 вершин. Каким должно быть количество ребер, чтобы граф был деревом? 8) Что необходимо сделать если при решении задачи минимизации методом ветвей и границ нижняя граница для подобласти А дерева поиска больше, чем верхняя граница какой-либо ранее просмотренной подобласти В? 9) Сколько работ может выполнять один процессор при многопроцессорном расписании в фиксированный момент времени? 10) К какому классу относится аналог задачи многопроцессорного расписания в виде задачи распознавания свойств? 11) Какую не должно превышать количество вершин любого уровня дерева поиска при решении задачи многопроцессорного расписания для m процессоров с помощью метода ветвей и границ? 12) Каково количество компонент связности в остовном дереве графа, если в графе их n?

10. Алгоритмы параллельных вычислений. Параллельная машина с произвольным доступом

EREW-алгоритмы решения задач определения номера элемента в списке, параллельная обработка префиксов списка, вычисление глубины вершины в двоичном дереве. CRCW-алгоритм решения задачи определения корня для вершины двоичного леса. CRCW-алгоритм нахождения максимального элемента в массиве. Эффективная параллельная обработка префиксов. Моделирование CRCW-машины с помощью EREW-машины.

Ключевые вопросы: 1) Укажите модели многопроцессорных систем? 2) Какое название носит многопроцессорная модель с исключающим чтением и исключающей записью? 3) Какая многопроцессорная модель является самой легкой с точки зрения аппаратной поддержки? 4) Что представляется собой величина равная произведения времени работы процессора на количество процессоров? 5) За какое время решается задача определения порядковых номеров в списке однопроцессорным алгоритмом? 6) Оцените величину общих затрат в многопроцессорном алгоритме для определения порядковых номеров в списке? 7) За какое время, имея n процессоров, можно сделать двусторонний список из одностороннего? 8) Каким методом работает однопроцессорный алгоритм вычисления глубины вершины в

двоичном дереве? 9) На какой многопроцессорной модели реализовывается алгоритм определения корня для вершины двоичного леса? 10) Как оценить общие затраты многопроцессорного алгоритма определения корня для вершины двоичного леса? 11) Какова вычислительная сложность многопроцессорного алгоритма определения максимального элемента n-мерного массива для n процессоров? 12) Сколько элементов исключается в алгоритме эффективной параллельной обработке префиксов из каждой группы элементов, за которую отвечает процессор?

11. История развития индустрии программного обеспечения

Проблемы, возникающие при программировании и их решения. Основные требования к индивидуальным заданиям. Примеры индивидуальных и коллективных проектов. Техника написания программ, комментарии, сопровождение, жизненный цикл программ.

Современные тенденции развития программного обеспечения ЭВМ и сетей. Перспективы развития технологических средств разработки ПО. Тенденции развития инструментальных средств разработки.

Задача проектирования программных систем. Организация процесса проектирования программного обеспечения (ПО). Цели проектирования ПО.

Принципы проектирования ПО. Использование декомпозиции и абстракции при проектировании ПО. Процедурная абстракция. Абстракция данных. Специфики процедур и данных. Декомпозиция системы.

Роль и принципы инженерии программного обеспечения. Жизненный цикл программного обеспечения. Определение. Проектирование. Реализация. Внедрение. Эксплуатация.

Классификация, требования к качеству в различных прикладных областях. Показатели качества программного обеспечения. Надёжность программного обеспечения. Факторы, снижающие надёжность работы программ. Программные методы повышения надёжности работы программ.

Испытания и сопровождение ПО. Методы защиты программ и данных.

Проектирование и архитектура программного обеспечения. Спецификация. Верификация.

Процесс производства программного обеспечения. Основные методы разработки программ. Нисходящее и восходящее проектирование. Модульное программирование.

Программирование с защитой от ошибок. Краткий обзор основных технологий программирования. HIPO технология. Структурное программирование. R - технология программирования. Методика Джексона. Коллективные методы разработки программ. Метод главного программиста.

Отладка. Типичные источники ошибок. Комплексная отладка программ. Определение пропускной способности программы.

Тестирование. Этапы тестирования. Нисходящее и восходящее тестирование. Автоматизация тестирования.

Документирование и оценка качества программных продуктов. Состав документации на программный продукт. Государственные стандарты документации на программный продукт.

Инструментальные средства поддержки разработки и программные среды.

12. Оформление текста программ, основы законодательства о программном обеспечении

Основные требования к оформлению текстов программ. Оформление текста программ – написание комментариев, их типы и разновидности, содержание. Копирайты, откуда берутся права, отличия в законодательстве России и США. Хорошая и плохая практика написания текста программ – выбор имен для переменных, использование препроцессора и т.д.

Ключевые вопросы: 1) Опишите технологию одновременной мультипоточности в процессорах. 2) Какой язык является основным в операционной системе MacOS? 3) В каком

файле должно хранится описание структуры данных С-программы? 4) Укажите стандартную библиотеку шаблонов языка С++? 5) Укажите вычислительную сложность перебора с помощью хэш-таблицы? 6) Что представляет собой план написания программы? 7) Каким образом аллокируется память? 8) Что относится к элементам жизненного цикла программы? 9) Что относится к сопровождения программного обеспечения? 10) Укажите типы комментариев программы? 11) Для чего предназначен префикс w в венгерской нотации? 12) В какой виртуальной машине транслируется язык C#?

13. Основные структуры данных и их работа в параллельном окружении

Понятие «структуры данных». Выбор структур данных для решения конкретных задач. Типовые элементы структур данных и связанных с ними алгоритмов, итераторы. Односвязный список и его использование в куче (heap) рантайма (библиотеке периода исполнения) языка Си, в функциях malloc и free. Проблемы использования структур данных в параллельном окружении на примере односвязного списка. Проблемы создания программ, использующих односвязные списки, в том числе в параллельном окружении. Двусвязные списки и очереди на их основе. Стек, его использование в программах на языке Си, поддержка рекурсии на его основе. Расположение данных в памяти. Использование структур памяти в параллельном окружении, операции синхронизации. Графы, деревья, использование их для компрессии данных. Хеш-таблицы. Разрешение коллизий с помощью цепочек. Хеш-функции (деление с остатком, умножение, универсальное хеширование). Хеш-таблица в параллельном окружении. Рекомендации по написанию параллельных программ.

Ключевые вопросы: 1) Дайтеопределение структуры данных. Приведите примеры. 2) Чем является итератор, имеющий доступ к элементам класса в С++? 3) Какая функция стандартной библиотеки языка С, предназначенная для освобождения ранее выделенной динамической памяти? 4) Что обеспечивает поддержку функций, предоставляемых системой программирования, во время выполнения программы? 5) Какой класс STL отвечает за очереди? 6) Каким образом организована структура данных динамической памяти в C/C++ RTL "куча"? 7) Какие из операций доступны в стеке? Какой метод доступа реализуется в стеке? 8) Укажите структуры данных с прямой адресацией элементов структуры? 9) Укажите недостатки использования односвязных списков? 10) Какой макрос языка С добавляет к программе процедуру диагностики? 11) Чем определяется размер одного элемента хеш-таблицы? 12) Что относится к способам разрешения коллизий?

14. Безопасность программного обеспечения. Техника кодирования и основы криптографии. Моделирование угроз

Безопасность программного обеспечение. Типовые проблемы, откуда они берутся. Написание программ без «дырок». Техника кодирования защищенных программ и типичные ошибки. Переполнение буфера, определение уровня доступа, работа с минимально возможными привилегиями, криптография и ее корректное применение, предохранение секретных данных, работа с входными данными, проблемы разных путей доступа к одним и тем же данным, запросов к базам данных и веб-страницам, а также проблемы поддержки интернационального ПО. Основные понятия криптографии, история. Симметричная и ассиметричная криптография, использование ее для безопасной почты. Безопасные и небезопасные алгоритмы (является ли секретный алгоритм безопасным?), генераторы случайных чисел. Устройство цифровой подписи. Как правильно применять криптографию для обеспечения безопасности. Способы взлома программ.

Моделирование угроз. Классификация опасностей STRIDE – Spoofing (подмена данных), Tampering (подделка и изменение содержания данных), Repudiation (незаконный отказ от проведенной операции), Information disclosure (разглашение информации), Denial of service (отказ в обслуживании), Elevation of privilege (незаконное поднятие привилегий). Методика оценки риска DREAD – Damage potential (что может быть сломано), Reproducibility (повторяемость), Exploitability (пригодность угрозы для использования), Affected users (на каких пользователей повлияет), Discoverability (возможно ли детектировать факт использования).

Ключевые вопросы: 1) Пусть *n* - длина символа строки, *m* - число символов строки. Чему равна длина строки? 2) Какова связь между типизацией указателя и его границей? 3) Что такое криптография? Чем занимается наука криптография? 4) Что означает термин аутентичность? 5) Укажите типичные проблемой безопасности для большинства языков программирования. 6) Чем определяется длина ключа при синхронной блочной криптографии? 7) Какова длина асимметричного ключа, стойкого ко взлому? 8) Приведите примеры алгоритмов асимметричного шифрования? 9) Какой алгоритм испольлзует составления цифровых подписей с восстановлением документа? 10) Что такое Tampering? 11) Опишите общую методику оценки риска DREAD? 12) Какой тип ключей использует программа PGP?

15. Эволюция аппаратного обеспечения

Эволюция современного аппаратного обеспечения и ее влияние на программное обеспечение. Гипертредовые (многопоточные) и многоядерные процессоры, универсальные графические (GPU) процессоры и новые возможности по их использованию.

16. Основы параллельного программирования

Параллельное программирование. Параллельные программы – от работы с разделяемой памятью, использования массивнопараллельных компьютеров и до распределенных расчетов на многих физических компьютерах. Декомпозиция задач на параллельные куски. Закон Амдала. Геометрическая декомпозиция. Параллелизм данных, параллелизм кода. Паттерны параллельного программирования: параллелизм на уровне задач – декомпозиция задачи, «разделяй и властвуй» - декомпозиция задач и данных, геометрическая декомпозиция – декомпозиция данных, конвейерное исполнение – декомпозиция потока данных, «фронт волны» – декомпозиция данных с «многомерными» зависимостями. Пример типового шаблона программирования – пул нитей.

17. Технические аспекты написания параллельных программ

Технические аспекты написания параллельных программ на разделяемой памяти. Многопоточные программы. Работа с разделяемой памятью. Атомарность операций. Синхронизационные примитивы (низкоуровневые атомарные команды, критические секции, взаимоисключающая блокировка – mutex, рекурсивная блокировка – lock, блокировка чтения-записи – read/write lock, многопроцессорная блокировка – spin-lock, семафоры, барьеры). Принятие во внимание особенностей аппаратной платформы, например, последовательности операций записи в память и т.д. Тупики и взаимные блокировки. Неблокирующая синхронизация. Реализация одних примитивов через другие, относительная «мощность» примитивов. Задача о консенсусе как способ оценки примитивов.

Ключевые вопросы: 1) Как классифицируются кластеры по степени связности? 2) Что следует отнести кнедостаткам Ethernet? 3) Приведите примеры наиболее важным задачам для суперкомпьютеров? 4) Опишите общие положения научного метода. Укажите типы декомпозиции? 5) Выделите типы блокировки потоков. 6) Что относится к достоинствам реализации потока в пространстве пользователя? 7) Дайте определение латентности. 8) Как называется объект синхронизации потоков позволяющий предотвратить одновременное выполнение некоторого набора операций несколькими потоками? 9) Какие проблем могут решать семафоры? 10) Выделите функции для работы с мьютексами? 11) Какому количеству потоков открыта влиянию атомарная операция? 12) В основе какой задачи лежит принцип выбора любого значения, предъявленного любым потоком?

18. Объектно-ориентированное программирование и технические особенности ее реализации в языках программирования. Поддержка ООП в различных системах и средах

Объектно-ориентированное программирование с понятийной точки зрения. История появления и возникновения понятий. Объекты как способ моделирования «параллельного внешнего мира» и формализации понятий. Объект как совокупность методов и данных, классы объектов, мета объекты. Объектная модель. Жизненный цикл объектов. Статические и динамические объекты. Раннее и позднее связывание, постоянно существующие объекты.

Инкапсуляция и полиморфизм как базовые понятия ООП. Наследование и агрегирование как способы создания новых, более сложных, объектов, их достоинства и недостатки (на примере C++ и COM). Технические способы создания объектов в языках типа C++, представление их в компьютере.

Техническая реализация ООП в языке программирования C++ – конструкторы и деструкторы, представление методов через функции, расположение данных в памяти. Реализация понятия полиморфизма через таблицы виртуальных функций. Наследование и его реализация, связь с виртуальными функциями, абстрактные типы данных. Исключения, их техническая реализация и особенности применения в языке и ОС Windows.

Поддержка ООП в языке и системе программирования ObjectiveC, категории, сообщения и методы. Поддержка ООП в ОС Windows, отличия от поддержки ООП в языке программирования. Реализация основных понятий ООП в этих системах. Использование подходов ООП в разных условиях.

Ключевые вопросы: 1) Дайте общее описание парадигмы объектно-ориентированного программирования? 2) Что относится к этапам жизненного цикла объекта? 3) Укажите типы классов языка C++? 4) Дайте определение полиморфизма, наследования, инкапсуляции. 5) Укажите свойства абстрактного класса. 6) Как называется конструктор, не принимающий аргументов? 7) Что определяет запись ~NameOfClass()? 8) Приведите примеры абстрактных типов данных. 9) Что можно отнести к типичным синхронным исключениям? 10) Что является основной задачей class objects в ObjectiveC? 11) Как называется метод ObjectiveC, который может быть переопределен? 12) Какие директивы используются в ObjectiveC для обработки исключений?

19. Технология аппаратной виртуализации и виртуальные машины

Технологии виртуализации и причины ее широкого распространения в последнее время. Типы виртуализации – виртуализация ОС и виртуализация оборудования. Преимущества и недостатки разных виртуализационных подходов с точки зрения потребителя. Технические основы виртуализации уровня ОС, контейнеры, уровни изоляции. Технология аппаратной виртуализации. Особенности работы кода прикладных программ внутри виртуальных машин. Новые возможности, предоставляемые виртуальными машинами и контейнерами – миграция, универсальность конфигурации, быстрота развертывания и обновления, удобства управления. Технические принципы организации виртуальных машин. Монитор виртуальных машин, аппаратная поддержка виртуализации, бинарная трансляция и другие подходы. Накладные расходы в реализации поддержки системы виртуальных машин. Паравиртуализация. Выбор оптимального типа виртуализации в разных условиях.

Ключевые вопросы: 1) Как называется программа, предназначенная для выполнения определенных пользовательских задач и рассчитанная на непосредственное взаимодействие с пользователем? 2) Что является способом связи процесса с ядром? 3) Какие области доступа поддерживаются в классах C++? 4) Что относится к основным элементам эмулятора? 5) Как называется наследование, при котором у класса более одного потомка? 6) Применение директивы const к какому-либо члену класса в C++ делает объект.. 7) Что относится к типам бинарной компиляции? 8) Как называется техника виртуализации, при которой гостевые операционные системы подготавливаются для исполнения в виртуализированной среде, для чего их ядро незначительно модифицируется? 9) Укажите классические примеры исключительных ситуаций? 10) Укажите типичные асинхронные исключения? 11) Что представляет собой Xen? 12) С помощью какого принципа можно запускать изолированные и безопасные виртуальные серверы на одном физическом сервере в операционной системе Windows?

20. Параллельные алгоритмы для многопроцессорных вычислительных систем

Примеры задач, решаемые на многопроцессорных системах: вычислительная газовая динамика, молекулярная динамика, символьные вычисления и др. Основные требования к

алгоритмам для суперкомпьютеров. Эффективность и ускорение. Кризис эффективности. Закон Амдала. Средства описания параллельных алгоритмов.

Многопроцессорные вычислительные системы с общей и с распределенной памятью; канал передачи данных и его свойства; синхронная и асинхронная передача данных; семафоры и операции над ними. Модели программ на общей и распределенной памяти. Ускорение и эффективности параллельных алгоритмов, свойство внутреннего параллелизма алгоритма. Пример алгоритма, обладающего низкой эффективностью, но высоким быстродействием (нахождение суммы конечного ряда).

Ключевые вопросы: 1) Какие вычислительные системы в настоящее время предоставляют большую вычислительную мощность? 2) Классифицируйте и опишите вычислительных систем с распределенной памятью. 3) В каком случае возможно совмещение вычислений и операций передачи данных? 4) Для решения каких задач предназначены суперкомпьютеры? 5) Оценить вычислительную сложность для задачи нахождения сумму элементов массива с помощью алгоритма сдваивания? 6) Чем определяется в кластерной вычислительной системе время передачи данных между процессорами?

21. Методы построения параллельных алгоритмов

Простейшие методы построения параллельных алгоритмов: метод геометрического и конвейерного параллелизма, метод коллективного решения. Их свойства. Проблемы статической и динамической балансировки загрузки процессоров. Алгоритм диффузной балансировки загрузки. Пример задачи, для сокращения времени решения которой необходимо создание нового алгоритма (сложение длинных чисел).

Ключевые вопросы: 1) Чем может быть обусловлено сокращение времени выполнения программы на многопроцессорной вычислительной системе? 2) Что требует динамическая балансировка загрузки? 3) Какой величины не должно превышать максимальное число процессоров, эффективное использование которых возможно при построении стены Фокса методом геометрического параллелизма? 4) Дайте общее описание метода коллективного решения? 5) Дайте общее описание метода конвейерного параллелизма? 6) Какие затраты относятся к накладным расходам, снижающим эффективность параллельных алгоритмов?

22. Сортировка данных на многопроцессорных вычислительных системах

Построение эталонного последовательного алгоритма сортировки. Пример «плохого» последовательного, но быстрого параллельного метода сортировки. Оценка времени выполнения алгоритма в наилучшем, среднем и наихудшем случае. Зависимость времени сортировки от вида исходной последовательности.

Минимальные сетей сортировки, сети четно-нечетное слияния. Построение на основе сетей сортировки параллельного алгоритма сортировки данных. Определение его характеристик при реализации на системах с общей и с распределенной памятью. Оценка времени выполнения сортировки. Оценка необходимого объема оперативной памяти. Обсуждение возможности сокращения объема передаваемых данных и требуемой оперативной памяти.

23. Методы параллельной генерации псевдослучайных чисел

Задачи, решаемые с использованием последовательностей псевдослучайных чисел (ПСЧ). Параллельные алгоритмы решения таких задач. Требования к генераторам ПСЧ для многопроцессорных систем. Параллельные алгоритмы генерации ПСЧ. Тестирование полученных последовательностей.

24. Параллельные алгоритмы интегрирования функций и решения систем линейных уравнений

Параллельные алгоритмы интегрирования функции одной переменной, построенные на основе методов геометрического параллелизма и коллективного решения. Адаптивный последовательный алгоритм интегрирования. Параллельный алгоритм интегрирования с динамической балансировкой загрузки для многопроцессорных систем с общей памятью.

Задачи, приводящие к появлению систем линейных уравнений с трехдиагональной матрицей коэффициентов. Параллельные алгоритмы решения трехдиагональных систем линейных алгебраических уравнений. Итерационные методы.

3. Методические указания

3.1 Методические указания по изучению дисциплины

Для оптимальной организации изучения дисциплины студентам рекомендуется следовать следующим методическим указаниям.

Студенты обязаны присутствовать на занятиях и выполнять все предусмотренные учебно-методическим комплексом дисциплины формы учебной работы; проходить промежуточный и итоговый контроль в виде защит практических (семинарских) работ, аттестации в форме тестового контроля знаний; сдачи зачета и экзамена в предлагаемой преподавателем форме.

Теоретическая часть курса включает указанные в рабочей программе темы. Каждая лекция содержит необходимый объем теоретического материала по каждой из изучаемых тем. В дополнение к лекционному материалу, студентам рекомендуется использовать основную и дополнительную литературу, указанную в рабочей программе.

Студенты в рамках аудиторных занятий должны, в целом, владеть понятийным аппаратом, основанном на ранее изученных дисциплинах, воспринимать теоретический материал основного содержания лекции, видеть причинно-логические связи в лекции, понимать схему решения примеров, приводимых в лекции. Для освоения темы каждой лекции на более глубоком уровне требуется дополнительная работа с теоретическим материалом в форме прочтения и изучения основной и дополнительной литературы, самостоятельной работы с лекцией.

Практические (семинарские) занятия направлены на закрепление теоретического материала на практическом уровне и предусматривают программную реализацию заданий по вариантам индивидуальных заданий. Допускается работа в подгруппах, состоящих из 2 студентов. Отчет оформляется каждым студентом отдельно. Опрос проводится независимо от личного вклада в результат выполнения работы. Для выполнения практической работы необходимо изучить теоретические основы соответствующего раздела, выполнить программную реализацию, провести верификации программы, на тестовых входных данных, оформить отчет по работе. При возникновении проблемных ситуаций в ходе решения практических задач (неясен алгоритм, непонятна ошибка программной среды при реализации метода, появились затруднения, связанные с тестированием алгоритма и пр.) или освоения теоретического материала преподавателем приветствуется любой диалог или дискуссия (возможно, с участием студентов, выполняющих задание по другому варианту), направленные на решение проблемы, при необходимости отведения дополнительного и/или индивидуального времени – в рамках консультаций во внеаудиторное время.

3.2 Методические указания к практическим (семинарским) занятиям

Для выполнения практических (семинарских) занятий студенту необходимо иметь конспект лекций. Студент знакомится с заданием и выполняет его, опираясь на конспект лекций. При выполнении задания связанного с программированием, студент разрабатывает алгоритм решения предложенной задачи, набирает текст программы (реализующей алгоритм), отлаживает и тестирует программу.

Практический курс предусматривает практические занятия по тематике, указанной в рабочей программе. Кроме текстов практических работ, студентам рекомендуется использовать также основную и дополнительную литературу, указанную в рабочей программе.

Индивидуальное задание по теме практической работы выполняется строго в соответствии с выданным преподавателем заданием и вариантом. Оформлять работу следует четко и аккуратно, придерживаясь основных правил оформления отчетных работ: титульный

лист (содержит: ФИО, №группы, курс, дисциплина, тема работы и т.д.), лист задания (содержит перечень предложенных заданий), раздел, содержащий теоретические основы соответствующего раздела курса (включая подробный алгоритм основного метода), раздел, содержащий описание программной реализации (листинг программного блока и описание интерфейса программы, если таковой имеется, может быть вынесен в приложении), раздел, содержащий анализ результатов.

Практическая работа считается выполненной с отметкой «зачтено», если: 1) программная реализация полностью соответствует заданию; 2) студент отвечает на основные теоретические вопросы по соответствующему разделу; 3) работа оформлена в соответствии с указанными требованиями.

Оформлять индивидуальную зачетную работу следует четко и аккуратно, придерживаясь основных правил оформления отчетных работ:

1) Титульный лист (содержит: наименование вуза, кафедра, ФИО исполнителя, №группы, курс, дисциплина, тема лаб. работы, вариант, выполнил, проверил и т. д.), содержание, введение.

2) Лист задания (содержит предложенное задание).

3) Основная часть. Раздел, содержащий теоретические основы соответствующего раздела курса (включая подробный алгоритм основного метода).

4) Раздел, содержащий описание программной реализации. Листинг программного блока и описание интерфейса программы (если таковой имеется) можно вынести в приложение.

5) Заключение, список использованной литературы.

Сроки сдачи работ ограничены отведенным на выполнение практикума аудиторным временем. Рекомендуется выполнять и сдавать на проверку отчеты по практическим работам по мере изложения лекционного материала и выдачи заданий преподавателем. Необходимым условием допуска студента на экзамен является сдача всех практических работ.

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

Общие требования: выполнение трех индивидуальных задания на семестр. Задачи всех трех заданий согласовываются с преподавателем, некоторые примерные варианты заданий указаны ниже. Каждый студент должен сделать по одной задаче из каждого задания. Тематика заданий выдается по усмотрению преподавателя (с учетом пожеланий студентов, которые могут предлагать свои темы), результат реализации должен включать взаимодействие с пользователем (пользовательский интерфейс в виде меню, желательно графического, хотя возможны и текстовые версии при условии использования ООП библиотек). Как правило, работа может выполняться коллективно (обычно 2 чел), должна включать в себя параллельное исполнение компонент (например, включать в себя нити или несколько взаимодействующих процессов) и использовать примитивы синхронизации (обязательно). Объем работы – не менее 600 строк, написанных одним человеком. Также должны использоваться коллекции данных по выбору (АТД – ATL, STL и т.д.).

Требования к оформлению. Текст программы должен быть оформлен профессиональным образом. Отдельно должен быть предоставлен файл (файлы) с текстом решения задачи, тестом на нее и вставляемые файлы заголовков. То есть решение задачи состоит из не менее чем 2 файлов (.h .CPP), чаще 3 (включая тест). Запрещается вставка одного C/CPP файла в другой, поощряется использование make-файла или проектов (не обязательно).

Комментарии в тексте программы: обязаны присутствовать в файлах в начале, должны быть отдельно написаны к каждой функции, и в коде по тексту (в среднем, каждая третья строка должна содержать комментарий); текст должен выглядеть красиво (отступы и т.д. должны быть оформлены нормально, не должно использоваться более 132 символов в строке), программа не должна иметь неиспользуемых или ненужных кусков (закомментированных). Примером оформления являются samples приложенные к

соответствующим дистрибутивам компиляторов (примеры SDK Windows, Linux kernel sources и т.д.).

Обязательно: Необходимо проверять все параметры функций, используя assert() или аналоги. Необходимо проверять все коды возврата функций (особенно malloc и системных вызовов). Желательно: составление Software Development Plan (SDP).

Задание 1 (*срок сдачи* – 6 неделя семестра)

Основная цель – правильное оформление текста задачи, проверка всех требуемых параметров и т.д. Реализация только на языке Си (не C++/C#). Реализовать одну из структур данных, итератор по ней и тест (3 файла: алгоритм, файл заголовков, тест). Тест должен демонстрировать работоспособность структуры. Помните, что требуемых структур может быть много (более одного экземпляра!). ***Примерные варианты***:

1) односвязный список;
2) двусвязный список;
3) очередь;
4) очередь с приоритетами;
5) стек;
6) набор (set);
7) упорядоченный набор с введением функции отношения;
8) массив произвольного размера;
9) упорядоченный массив;
10) бинарное дерево;
11) сбалансированное дерево;
12) хэш-таблица с открытой адресацией (хешировать текстовые строки произвольной длины);
13) хэш-таблица с цепочками (см. Д. Кнута «Искусство программирования. Т.3», хешировать бинарные данные произвольной длины типа BLOB – Binary Large Object);
14) битовый массив (значения принимаются битами, а адресация – по номеру бита);
15) направленный граф (в виде квадратной таблицы отношений $M(i, j)$=1 если есть путь из i в j узел).

Задание 2 (*срок сдачи* – 11 неделя семестра)

Требования к оформлению – те же самые. Программа должна быть выполнена на C++ и в обязательном порядке использовать чужие библиотеки абстрактных типов данных (любую, например, Borland ClassLib, MS MFC/Atl/Stl, GNU libstdc++, STL, и т.д.).

Задачи должны быть выполнены на C++ или C# (для платформы Windows) с использованием более одной нити или процесса, по выбору студента, **согласованному** с преподавателем. Для этого под Windows может быть использована библиотека функций Win32 API (или более высокоуровневые библиотеки). Под Unix (Linux): POSIX Threads. По желанию студента задача может быть дополнена графическим интерфейсом. Единицу исполнения (нить или процесс) далее будем называть *вычислителем*.

Задачи с использованием деревьев.

Задачи этого раздела используют класс *Tree* для создания древесной структуры данных.

1. Поиск ключа узла дерева.

Исходные данные: дерево, каждый элемент которого хранит уникальный, случайно сгенерированный ключ; ключ, узел с которым требуется найти, количество используемых вычислителей.

Цель: написать параллельную программу для поиска данного ключа. Программа должна создать заданное пользователем дерево, считать количество используемых вычислителей и ключ для поиска, найти элемент дерева, содержащий этот ключ и вывести информацию об этом узле.

2. Поиск максимального элемента дерева.

Исходные данные: дерево, каждый элемент которого хранит уникальный, случайно сгенерированный ключ, количество используемых вычислителей.

Цель: написать параллельную программу для поиска максимального ключа. Программа должна создать заданное пользователем дерево, считать количество используемых вычислителей, найти элемент дерева, содержащий максимальный из ключей и вывести информацию об этом узле. Переменная, содержащая максимальный элемент, должна быть общей для всех вычислителей. Доступ к ней – синхронизован.

3. Поиск суммы всех элементов дерева.

Исходные данные: дерево, каждый элемент которого хранит уникальный, случайно сгенерированный ключ, количество используемых вычислителей.

Цель: написать параллельную программу для поиска суммы всех ключей дерева. Программа должна создать заданное пользователем дерево, считать количество используемых вычислителей, найти и вывести сумму всех ключей дерева. Переменная, содержащая сумму ключей, должна быть общей для всех вычислителей. Доступ к ней – синхронизован.

4. Поиск количества узлов дерева.

Исходные данные: дерево, количество используемых вычислителей.

Цель: написать параллельную программу для поиска количества узлов в дереве.

Программа должна создать заданное пользователем дерево, считать количество используемых вычислителей, найти и вывести количество всех узлов в дереве. Переменная, содержащая максимальный элемент, должна быть общей для всех вычислителей. Доступ к ней – синхронизован.

Возможные алгоритмы распараллеливания работы для задач 1-4:

a) Пусть есть N вычислителей. У первого – ссылка на корень дерева. Он берет одно из поддеревьев для поиска себе, остальные раздает свободным вычислителям. Далее вычислители поступают точно также. Когда свободных вычислителей не остается, каждый выполняет обычный поиск по своему поддереву.

b) Пусть есть N вычислителей, и выбран вычислитель-мастер. Он сам проводит поиск ключа по дереву в ширину до тех пор, пока не найдет *N* независимых поддеревьев, после чего раздает их для поиска остальным вычислителям.

Усложнение для задач 1-3: пусть каждый узел дерева содержит дополнительную информацию – количество узлов под ним. К основной цели задачи добавляется равномерное распределение загрузки вычислителей: каждый должен получить поддерево(вья) с примерно одинаковым количеством узлов (отличающимся не более, чем на единицу).

Вычислительные задачи.

5. Вычисление интеграла численными методами.

Исходные данные: функция одной переменной, отрезок интегрирования, количество вычислителей.

Цель: написать параллельную программу для вычисления интеграла от данной функции по данному отрезку.

Программа должна считать функцию, отрезок интегрирования, количество вычислителей, посчитать интеграл в несколько вычислителей, вывести результат.

Предполагаемый алгоритм распараллеливания: использовать параллелизм по отрезку. Каждый вычислитель считает интеграл по своей части отрезка с помощью одного из стандартных методов (трапеции, Симпсона, Ньютона-Лейбница, …). Результаты отдельных вычислителей суммируются.

Численный метод согласуется с преподавателем.

6. Вычисление интеграла методом Монте-Карло.

Исходные данные: функция одной переменной, отрезок интегрирования, количество вычислителей.

Цель: написать параллельную программу для вычисления интеграла от данной функции по данному отрезку методом Монте-Карло. Программа должна считать функцию,

отрезок интегрирования, количество вычислителей, посчитать интеграл в несколько вычислителей, вывести результат.

Предполагаемый алгоритм распараллеливания:

a) Использовать параллелизм по отрезку. Каждый вычислитель считает интеграл по своей части отрезка. Результаты отдельных вычислителей суммируются.

b) Использовать параллелизм по количеству испытаний в методе Монте-Карло. Необходимое количество испытаний делится между N вычислителями. Результаты испытаний обобщаются.

7. Параллельное перемножение матриц.

Исходные данные: две матрицы [$n \times m$] и [$m \times n$], количество вычислителей.

Цель: написать программу для перемножения матриц параллельно несколькими вычислителями. Программа должна считать m, n, две матрицы, количество вычислителей; распараллелить процесс их перемножения.

Предполагаемый алгоритм распараллеливания: поскольку каждый элемент матрицы-результата рассчитывается независимо от других, он может быть определен отдельным вычислителем. Вычислитель-мастер занимается распределением работы между остальными. Он хранит базу свободных вычислителей, куда синхронизовано заносятся не имеющие работы и откуда берутся рабочие для вычисления следующего элемента матрицы.

8. Параллельное вычисление детерминанта матрицы заданного размера (параллелизм по дереву рекурсии).

Исходные данные: матрица [$n \times n$], количество вычислителей.

Цель: написать программу для расчета детерминанта матрицы параллельно несколькими вычислителями.

Программа должна считать n, матрицу, количество вычислителей; распараллелить процесс вычисления детерминанта матрицы.

Предполагаемый алгоритм распараллеливания: предполагается вычислять определитель методом Гаусса - приведением к верхнетреугольной матрице и далее перемножением элементов диагонали. Надо придумать способ распределения подзадач по вычислителям, например, получив дерево вычислений, оно же дерево рекурсии. А примеры распараллеливания (раздачи элементов дерева вычислителям) описаны в задаче 1 раздела *Задачи с использованием деревьев*.

9. Решение уравнения Ax=b методом Крамера: параллелизм по элементам столбца решения.

Исходные данные: невырожденная матрица [$n \times m$], столбец [$n \times 1$], количество вычислителей.

Цель: написать программу для расчета решения уравнения $Ax=b$ параллельно несколькими вычислителями. Программа должна считать m, n, матрицу A, столбец b, количество вычислителей; распараллелить решение уравнения $Ax=b$ методом Крамера.

Предполагаемый алгоритм решения: методом Крамера каждый элемент столбца-результата вычисляется независимо от остальных. Поэтому его можно передать для расчета отдельному вычислителю.

10. Решение дифференциального уравнения $y'+y=f(x)$ методами вычислительной математики.

Исходные данные: функция $f(x)$ (правая часть), отрезок, на котором ищется решение, количество вычислителей.

Цель: написать программу для расчета y решения уравнения $y'+y=f(x)$ параллельно несколькими вычислителями.

Программа должна считать правую часть, отрезок интегрирования; распараллелить процесс решения дифференциального уравнения заданным методом вычислительной математики.

Способ распараллеливания: стандартный, по отрезку. Отрезок делится на количество вычислителей, и каждому отдается своя часть. Используемый метод вычислительной математики согласовывается с преподавателем.

Сортировка.

11. Сортировка слиянием.

Исходные данные: массив объектов, функция для их сравнения, количество вычислителей.

Цель: написать программу для упорядочивания массива параллельно несколькими вычислителями.

Предполагаемый алгоритм распараллеливания: при сортировке слиянием массив делится на две равных по размеру части, которые сортируются отдельно и сливаются в один. Таким образом, снова получаем дерево рекурсии.

Вывод на экран.

12. Контроль за выводом на экран нескольких потоков.

Исходные данные: нет.

Цель: добиться с помощью средств синхронизации, чтобы несколько вычислителей выводили информацию на экран в строго определенном порядке. Программа должна создать 26 потоков, каждый из которых выводит свою букву алфавита в бесконечном цикле. Добиться, чтобы все буквы выводились в алфавитном порядке.

Работа с файлами.

13. Копирование файла в несколько потоков.

Исходные данные: файл, директория его будущего расположения, количество вычислителей.

Цель: написать программу для копирования файла в другое место параллельно несколькими вычислителями.

Предполагаемый алгоритм распараллеливания:

a) Каждый вычислитель открывает файл отдельно. Они делят файл на равные части, каждый копирует свою часть (читая из копируемого файла и записывая в файл результата блоки данных фиксированного размера).

b) (Для нитей одного процесса) Файл открывается один раз. Каждый вычислитель также должен скопировать свою часть файла. Но, поскольку файловый указатель общий – доступ к нему должен быть синхронизован. После использования указателя (т.е. копирования очередного блока данных из своей части) вычислитель должен вернуть его в первоначальное положение и снова включится в борьбу за общий ресурс.

14. Копирование директории вместе со всем ее содержимым (параллелизм по файлам).

Исходные данные: директория для копирования, директория ее будущего расположения.

Цель: написать программу для копирования директории вместе со всем ее содержимым в другое место параллельно несколькими вычислителями.

Предполагаемый алгоритм распараллеливания: снова для операции копирования можно составить дерево рекурсии => распараллеливание по дереву (задача 1 раздела *Задачи с деревьями*).

Моделирование различных ситуаций.

15. «Тренировка футболистов».

Участвующие вычислители: футболисты. *Общий ресурс:* мяч.

Ситуация: команда футболистов отрабатывает индивидуальные действия. Каждый играет за себя. Цель футболиста – завладеть мячом и забить гол. Таким образом. Он может выполнять два действия:

a) Бороться за мяч.

b) Бить по воротам, как только мяч оказался у него. После удара по воротам вратарь выбивает мяч в поле. Футболист забивает гол с определенной вероятностью. Команда играет определенное время, потом определяется победитель – по забитым мячам.

16. «Семейный автомобиль».

Участвующие вычислители: члены семьи. *Общий ресурс:* автомобиль. *Ситуация* семья из 4-х (N) человек: отец, мать, сын и дочь, – имеет в распоряжении автомобиль. У каждого из них периодически появляется желание/необходимость использовать его. Пользование длится случайное время. По истечении определенного времени (известного заранее) бензин автомобиля заканчивается, и тот, кто в этот момент пользуется автомобилем едет его заправлять. Кроме того, сын ответственен за мытье автомобиля. Также по истечению известного заранее времени автомобиль загрязняется. Если сын начинает пользоваться грязным автомобилем, он его моет перед этим. Итого, все члены семьи «умеют» ездить на автомобиле, заправлять его, ожидать его освобождения. Сын, помимо прочего «умеет» мыть автомобиль.

17. «Аэропорт».

Участвующие вычислители: самолет. *Общий ресурс:* взлетно-посадочные полосы.

Ситуация: в течение каждого часа в аэропорту появляется 5 самолетов, готовый в вылету и 5 самолетов, готовых приземлиться (каждый из них в случайное время в пределах этого часа). Чтобы взлететь или приземлиться самолету требуется 3 минуты. В аэропорте 2 взлетно-посадочных полосы.

18. «Воробьи».

Участвующие вычислители: воробей. *Общий ресурс:* хлебные крошки.

Ситуация: бабушка на скамейке периодически (пусть раз в Q минут) бросает воробьям по одной ровно N хлебных крошек. Когда бросают крошку хлеба, первый подлетевший к не воробей хватает крошку, относит ее в сторону и съедает. За счет этого он пропускает «розыгрыш» следующей крошки. Первоначально у скамейки находится M воробьев. Каждую минуту к стае прилетает еще один воробей. Съев P крошек, воробей наедается и улетает. Чтобы количество воробьев не увеличивалось, должно выполняться соотношение $Q = N/P$. Итак, воробей может выполнять действия:

· Прилететь (рождение процесса).

· Ждать раздачи крошек.

· Драться за крошку.

· Отлететь с крошкой в сторону и съесть ее.

· Улететь (завершение работы процесса).

19. «Раздача карт».

Участвующие вычислители: игрок. *Общий ресурс:* колода карт.

Ситуация: N игроков тянут карты по одной из лежащей перед ними колоды.

Необходимо обеспечить:

· Очередность доступа к колоде.

· Равномерное распределение карт в колоде.

Можно после раздачи посчитать количество очков в вытянутых ими картах и определить победителя.

Задание 3 (*срок сдачи* – 15 неделя семестра).

Тема согласовывается с преподавателем. Примерная тематика заданий рассматривается на лекции.

Каждая из работ должна содержать подробный отчет. Для третьего задания кроме отчета прилагается презентацию и выступление о результатах проделанной работы.

Рекомендуемая литература:

1. Буч Г. Объектно-Ориентированный Анализ и Проектирование. 2-е издание.– М.: БИНОМ, 1998.

2. Страуструп Б. Язык программирования С++. 3-е издание, – М.: БИНОМ, 1999.

3. Секунов Н. Visual C++.NET. – СПб.: БХВ-Петербург, 2002.

4. Топп У., Форд У. Структуры данных в С++. – М.: Вильямс, 2000.

5. Майерс С. Эффективное использование STL. – СПб.: ПИТЕР, 2002.

6. Александреску А. Современное проектирование на С++. Обобщенное программирование и прикладные шаблоны проектирования. – М.: Вильямс, 2002.

7. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Приемы объектно-ориентированного проектирования. Паттерны проектирования. – СПб.: ПИТЕР, 2003.

8. Влиссидес Дж. Применение паттернов проектирования. Дополнительные штрихи. – М.: Вильямс, 2003.

9. М. Руссинович, Д. Соломон Внутреннее устройство Microsoft Windows: Windows Server 2003, Windows XP, Windows 2000. Мастер-класс. Серия: Мастер-класс Издательства: Питер, Русская Редакция, 2005 г. 992 с.

10. Защищенный код. Дэвис Лебланк, М. Ховард, 2-е издание. 704 стр., 2005 г. Издательство: Русская редакция. Серия: Фундаментальные знания.

Дополнительные задачи по различным тематикам

Базы данных:

1. Разработать приложение справочной службы аэропорта. Программа должна выдавать справки об авиарейсах (например, Благовещенск-Москва). По требованию сообщать следующую информацию: количество свободных мест на заданное число, проданные места на заданное число, проданные места на весь месяц.
2. Разработать помощник экзаменатора. Экзаменационные вопросы и ответы хранятся в файлах на диске. Каждый вопрос имеет бал сложности. Необходимо подобрать 5 вопросов из разных разделов курса, имеющих в сумме бал сложности N и вывести их на экран. Предусмотреть тренировочный режим работы, когда возможен вывод ответов на предложенные вопросы. Доступ к тренировочному режиму предоставляется по паролю.
3. Разработать справочную систему по стандартным функциям языка Турбо Паскаль.
4. Разработать справочную систему по операторам языка Турбо Паскаль.
5. Разработать справочную систему по визуальным компонентам Delphi.
6. Разработать справочную систему по стандартным процедурам Object Pascal.
7. В альпинистском клубе ведется хроника восхождений. Записываются даты начала и завершения каждого восхождения, имена и адреса участвующих в нем альпинистов, название и высота горы, район или страна, где она расположена. Создать программу ведения БД, которая включает в себя:

- ввод и корректировку исходных данных;
- вывод информации по запросам: а) наименование вершины, потребовавшей самого длительного восхождения и страны, где она расположена; б) ФИО и адрес альпиниста, совершившего наибольшее число восхождений.

Для хранения информации использовать два файла.

8. Разработать программу ведения БД данных риэлторской фирмы. Данные о проданных квартирах хранятся в двух файлах – основном и справочном. Основной файл содержит сведения о сделках и имеет следующие поля: дата сделки, покупатель, идентификационный номер продаваемой квартиры, цена. Файл-справочник содержит информацию о проданных квартирах: общая площадь квартиры, число комнат. Номер записи в файле-справочнике соответствует идентификационному номеру квартиры. Предусмотреть ввод и корректирование исходной информации и вывод ведомости о сумме продаж, совершенных в каком-либо году, месяце. Ведомость имеет вид:

Месяц	Сумма продаж	По числу комнат в квартире		
		1	2	3
Январь				
…				
Итого	x			

9. Создать программу ведения БД сотрудники. Информация хранится в двух файлах. Файл 1 – список сотрудников: ФИО, код должности и подразделения. Файл 2 – справочник: наименование должности, зарплата за 1 час работы. Количество отработанных часов для каждого сотрудника вводится с клавиатуры ПК. Ведение БД включает в себя: а) ввод информации о сотрудниках и запись ее на диск; б) удаление ненужной информации; в) корректировка записей БД; г) вывод расчетной ведомости для каждого подразделения, которая имеет вид:

№	ФИО	Начислено	Подоходный налог	Отчисления в пенсионный фонд	К выплате

Динамические структуры

10. Построить имитационную модель бензоколонки. На бензоколонке *k* стоек (1 стойка обслуживает один автомобиль). Каждый автомобиль обслуживается s секунд, интервал между прибытием на бензоколонку автомобилей является случайной величиной, распределенной по закону P(x). Если все стойки заняты, автомобиль становится в очередь. Для заданных P(x) и s определить возможно меньшее значение k для того, чтобы очередь не удлинялась.
11. Составить программу решения «задачи коммивояжера». Необходимо определить минимальную стоимость проезда коммивояжера по N городам с возвращением в исходную точку. Каждый город входит в маршрут только один раз. Предположить, что стоимость проезда из города I в город j такая же, как и из j в i.

Игры

12. *Шахматы.* Написать программу, которая генерирует или считывает шахматную позицию и определяет не находится ли один из королей под шахом и не является ли шах матом. Предусмотреть два варианта ввода исходных данных: а) шахматная позиция генерируется с помощью датчика случайных чисел; б) шахматная позиция вводится с клавиатуры.
13. *Кроссворд.* Написать программу составления кроссвордов. Исходными данными является конфигурация 6x6 (некоторое расположение пустых и заполненных квадратов) и список слов, состоящих из шести или менее букв. Результатом должно быть расположение этих слов, образующее общепринятый кроссворд или сообщение о том, что такая конфигурация невозможна.
14. *«Кости».* Вводится число от 2 до 12 и ставка. Программа с помощью датчика случайных чисел дважды выбирает числа от 1 до 6 (бросание двух кубиков, на гранях которых должны быть числа от 1 до 6). Если сумма выпавших цифр меньше 7 и играющий задумал число меньше 7, он выигрывает ставку. Если сумма выпавших цифр больше 7 и играющий задумал число больше 7, он тоже выигрывает ставку. Если угадал сумму цифр, то получает в 4 раза больше, чем ставка. Ставка проиграна, если ни одна из ситуаций не учтена. В начальный момент времени у играющего 100 очков. В программе должно присутствовать графическое изображение кубиков при каждом ходе игрока.
15. *Морской бой.* Поле 10x10 позиций. Стоят невидимые вражеские корабли: 4 по 1 клетке, 3 по 2 клетки, 2 по 3 клетки, 1 – 4 клетки. Необходимо поразить каждую из клеток кораблей. Два игрока вводят позиции кораблей в виде цифр (1,2,3,4) – соответствующие позиции матрицы, тем самым определяя конфигурацию положения кораблей. Игроки по очереди наносят удары по кораблям противника. Если позиция корабля указана верно, то она помечается крестиком на поле. Предусмотреть вариант игры, когда один из игроков – ЭВМ.
16. *«Сбей самолет».* По экрану летят вражеские самолеты. Цель играющего сбить самолеты. Пусковая установка находится в нижней части экрана. Установку можно перемещать по строке вперед и назад.

17. *Обучение работе на клавиатуре.* Составить программу обучения работе с клавиатурой. Программа должна выдавать на экран буквы, цифры, слова и фразы, которые следует набрать на клавиатуре и оценивать правильность и скорость набора. В программе предусмотреть 3 уровня подготовленности обучающегося.
18. *«Скачки».* В игре участвуют 10 наездников. За каждый тур игры каждый из них продвигается вперед на расстояние от 1 до 5 км случайным образом. Длина дистанции 50 км. Всего производится 5 заездов. Победителю каждого заезда начисляется 5 очков. Победитель - наездник, набравший максимальное количество очков во всех заездах. Перед началом заездов, участник игры выбирает номер наездника, с которым он будет идентифицироваться во время игры. В каждом туре с вероятностью 0,1 каждый наездник может упасть, т.е. передвинуться на 0 км. Передвижение наездников отображать графически на экране.

Строки и текстовые файлы.

19. Текст программы на Паскале хранится в файле на диске. Составить программу обработки текста программы: а) подсчитать какие ключевые слова Паскаля, и в каком количестве использованы в обрабатываемом тексте; б) составить перечень имен простых переменных; в) представить перечень меток программы в алфавитном порядке.
20. Текст программы на Паскале хранится в файле на диске. Составить программу обработки текста программы: а) определить максимальную степень вложенности циклов в программе; б) определить общее количество строк и символов, отличных от пробела; в) удалить из текста программы все комментарии.
21. Текст программы на Паскале хранится в файле на диске. Составить программу обработки текста программы: а) первые буквы служебных слов сделать заглавными; б) текст комментария заменить на номер комментария по порядку; в) переписать текс программы в новый файл с минимальным количеством пробелов (сохранить пробелы только там, где они необходимы).
22. Создать программу, анализирующую правильность записи арифметического выражения с точки зрения синтаксиса Паскаля. Арифметическое выражение задается строковой переменной и вводится с клавиатуры.
23. Текст программы на Паскале хранится в файле на диске. Распечатать на экран текст программы таким образом, чтобы в каждой строке размещался только один оператор. Организовать смещение операторов относительно операторных скобок как это принято в Паскале.
24. Составить программу, позволяющую сжимать текстовую информацию, а затем преобразовывать сжатую информацию в исходное состояние. В программе предусмотреть два варианта: а) для хранения текста в сжатом виде найти часто повторяющиеся последовательности из двух букв и заменить их кодом. В качестве кода использовать символы, не встречающиеся в тексте. Составить таблицу кодов; б) в заданном тексте найти слова, которые встречаются более 3 раз. Закодировать их и сжать текст, заменив слова кодами. Составить таблицу кодов.
25. Разработать программу «выравнивания». Исходная информация – текст, записанный в текстовом файле. Программа выводит этот текст с выравниванием по краям. Параметры абзаца задаются в диалоговом режиме.
26. Текст программы на Паскале хранится в файле на диске. Считать программу с диска и получить ее распечатку. В распечатке программы отметить операторы, изменяющие значения заданных переменных (их имена вводятся). Степень вложенности операторов цикла пометить добавлением слева соответствующего количества пробелов.
27. Составить программу, определяющую наличие неописанных идентификаторов в тексте программы на Паскале. Текс программы хранится в файле на диске.

3.3 Методические указания по самостоятельной работе студентов

Самостоятельная работа студента предусмотрено в объеме 40 часов.

Схема самостоятельной работы студентов, перечень тем, рекомендации по работе с литературой, рекомендации по подготовке к аттестации приводиться в рабочей программе.

При выполнении самостоятельной работы студенты могут использовать различные источники приобретения информации: конспекты лекций, учебно-методические материалы курса, ссылки на научную литературу в информационном пространстве Интернета и др.

Задачи для самостоятельного решения

1. Дана сеть с дугами (1, 2), (1, 3), (1, 4), (2, 5), (2, 7), (3, 6), (4, 6), (4, 7), (5, 8), (5, 10), (6, 7), (6, 9), (7, 8), (7, 10), (8, 10), (9, 10), пропускные способности которых равны 4, 3, 3, 2, 2, 2, 3, 3, 3, 3, 3, 3, 1, 2, 2, 2 соответственно. Найти максимальный поток из узла 1 в узел 10 с помощью: а) алгоритма Форда – Фалкерсона; б) алгоритма Карзанова. Найти минимальный разрез в этой сети.

2. Определить вычислительную сложность алгоритмов Форда – Фалкерсона и Карзанова. Являются ли эти алгоритмы полиномиальными?

3. Показать, как с помощью алгоритма Форда –Фалкерсона найти: а) минимальный разрез в сети; б) максимальный поток в сети, когда пропускные способности заданы в вершинах сети.

4. Запасом связности неориентированного графа $G = (V, A)$ называется минимальное число ребер, которые необходимо удалить, чтобы сделать граф несвязным. Как найти запас связности графа с помощью программы поиска максимального потока? (Ее следует применять не более чем к $/V/$ сетям, каждая из которых содержит $O(/V/)$ вершин и $O(/A/)$ дуг.)

5. Как найти максимальное паросочетание в двудольном графе с помощью алгоритма нахождения максимального потока в сети?

6. С помощью алгоритма дефекта найти поток минимальной стоимости в сети $G = (V, A)$, $V = \{1, 2, 3, 4\}$, $A = \{(1,2), (1,3), (2,3), (2,4), (3,2), (3,4), (4,1)\}$. Параметры дуг ($L_{ij}$, U_{ij}, c_{ij}) следующие: (2,5,3), (2,4,4), (0,2,3), (2,4,2), (0,2,2), (1,3,5), (4,4,0). (L_{ij} – нижняя граница потока по дуге (i, j), U_{ij} – верхняя граница потока по дуге (i, j), c_{ij} - стоимость единицы потока по дуге (i, j)).

7. Определить, существует ли допустимое расписание (с прерываниями) выполнения четырех заданий на двух идентичных процессорах, и если оно существует, найти его. Директивные интервалы $[b_i, f_i]$ и длительности заданий t_i следующие: $[b_1, f_1] = [0,5]$, $[b_2,f_2] = [1, 5]$, $[b_3, f_3] = [1, 6]$, $[b_4, f_4]= [0, 6]$, $t_1 = 2$, $t_2 = 3$, $t_3 = 2$, $t_4 = 5$. (Решить данную задачу путем сведения ее к задаче о максимальном потоке.)

8. Имеется три прибора p_1, p_2, p_3 с ресурсом эксплуатации соответственно 20, 10 и 30 часов и четыре работы w_1, w_2, w_3, w_4 с запросами соответственно 14, 12, 18 и 16 часов. Прибором p_1 могут выполняться работы w_1, w_2 и w_4, прибором p_2 - работы w_1, w_3 и w_4, прибором p_3 - работы w_3 и w_4. При выполнении работ допускаются прерывания и переключения с одного прибора на другой. Определить, могут ли быть полностью выполнены все четыре работы и, если могут, то каким образом. (Решить данную задачу путем сведения ее к задаче о максимальном потоке.)

9. Является ли кучей массив (23, 17, 14, 6, 13, 10, 1, 5, 7, 12)?

10. Доказать, что куча из n элементов содержит не более $\left\lceil \frac{n}{2^{h+1}} \right\rceil$ вершин высоты h.

11. Показать, как работает процедура построения кучи для массива (5,3,17, 10, 84, 19, 6, 22, 9).

12. Показать, как работает процедура сортировки с помощью кучи для массива (5, 13, 2, 25, 7, 17, 20, 8, 4).

13. Как будет выглядеть хеш-таблица с цепочками после того, как в нее последовательно поместили элементы с ключами 5, 28, 19, 15, 20, 33, 12, 17, 10 (в указанном порядке)? Число позиций в таблице равно 9, хеш-функция имеет вид $h(k) = k \bmod 9$.

14. Пусть h – хеш-функция, ставящая в соответствие каждому из n различных ключей $\{k_1, k_2, \ldots, k_n\}$ одну из m позиций в таблице. Чему равно математическое ожидание числа коллизий (т.е. мощности множества $\{\{k, l\}: k \neq l, h(k) = h(l)\}$) в предположении равномерного хеширования?

15. Показать, что математическое ожидание времени добавления нового элемента в хеш-таблицу с цепочками, имеющую коэффициент заполнения α (в предположении равномерного хеширования), есть O(1 + α), если новый элемент добавляется в конец соответствующей цепочки.

16. Построить детерминированную машину Тьюринга, которая складывает два числа в двоичной системе. Определить зависимость времени ее работы от длины входа.

17. Можно ли установить взаимно-однозначное соответствие между рекурсивными и не рекурсивными языками?

18. Верно ли, что любой рекурсивный язык является рекурсивно перечислимым?

19. Верно ли, что язык L является рекурсивным тогда и только тогда, когда оба языка L и $\overline{L}$ являются рекурсивно перечислимыми?

20. Верно ли, что всякий язык $L \in NP$ является рекурсивно перечислимым? Будет ли он рекурсивным?

21. Верно ли, что в *co-NP* есть полные языки?

22. Верно ли, что класс языков P замкнут относительно операций: (а) объединения; (б) пересечения; (в) дополнения; (г) конкатенации; (д) операции Клини.

23. Верно ли, что класс языков NP замкнут относительно операций: (а) объединения; (б) пересечения; (в) конкатенации; (г) операции Клини.

24. Верно ли, что $P \subseteq co\text{-}NP$?

25. Верно ли, что если $NP \neq co\text{-}NP$, то $P \neq NP$?

26. Верно ли, что алгоритм, который содержит фиксированное (независящее от входа) число вызовов процедуры, работающей полиномиальное время, сам работает полиномиальное время. Может ли общее время работы алгоритма быть экспоненциальным, если он делает полиномиальное число вызовов такой процедуры?

27. Доказать, что любой язык из NP распознается за время $2^{O(n^k)}$, где n – длина входа, k – не зависящая от n константа.

28. Доказать, что язык L полиномиально сводится к языку $\overline{L}$ тогда и только тогда, когда $\overline{L}$ полиномиально сводится к L.

29. Доказать, что в классе P все языки (кроме $\varnothing$ и Σ^*, Σ - алфавит) полны относительно полиномиальной сводимости.

30. Доказать, что язык L полон в NP тогда и только тогда, когда $\overline{L}$ полон в *co-NP*.

В задачах 31–46 *все числовые параметры предполагаются натуральными.*

В задачах 31–37 *определить, принадлежат ли соответствующие языки классам NP или co-NP. Если язык принадлежит NP, определить, является ли он NP-полным.*

31. *Упаковка множеств.* Заданы семейство C конечных множеств и число K, $K \leq |C|$. Верно ли, что в C имеется K непересекающихся множеств?

32. *Наибольший общий подграф.* Заданы два графа $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$ и число K. Существуют ли такие подмножества $E'_1 \subseteq E_1$ и $E'_2 \subseteq E_2$, что $|E'_1| = |E'_2| \geq K$, а подграфы $G'_1 = (V'_1, E'_1)$ и $G'_2 = (V'_2, E'_2)$ изоморфны?

33. *Доминирующее множество.* Заданы граф $G = (V, E)$ и число $K \leq |V|$. Существует ли такое подмножество $V' \subseteq V$, что $|V'| \leq K$ и каждая вершина $v \in V \setminus V'$ соединена ребром, по крайней мере, с одной вершиной из V'?

34. *Минимум суммы квадратов.* Заданы конечное множество N, размер s_i для каждого $i \in N$ и число K. Верно ли, что *заданное* разбиение множества N на K непересекающихся

подмножеств N_1, ..., N_K имеет минимальный вес $\sum_{j=1}^{K}(\sum_{i\in N_j} s_i)^2$ среди разбиений N на K непересекающихся подмножеств?

35. *Паросочетание.* Задан двудольный граф G и число K. Верно ли, что в G существует K непересекающихся по вершинам ребер (паросочетание мощности K)?

36. *Упаковка в контейнеры.* Заданы конечное множество N предметов, размер v_i каждого предмета $i \in N$, вместимость V контейнера и число K. Существует ли такое разбиение множества N на непересекающиеся подмножества N_1, ..., N_K, что $\sum_{i\in N_j} v_i \le V$ для всех $j = 1, ..., K$?

37. *Интеграл от произведения косинусов.* Задана последовательность чисел a_1, a_2, ..., a_n. Верно ли, что $\int_0^{2\pi}\left(\prod_{i=1}^{n}\cos(a_i x)\right)dx = 0$?

Для задач 38, 39 *привести псевдополиномиальные алгоритмы*

38. *Рюкзак.* Заданы конечное множество предметов N, стоимости s_i, размеры v_i для всех $i \in N$, числа V и S. Существует ли подмножество $M \subseteq N$ такое, что $\sum_{i\in M} v_i \le V$, $\sum_{i\in M} s_i \ge S$? Найти это подмножество M (если оно существует). Вычислительная сложность алгоритма $O(|N|\cdot\min\{V, S\})$, требуемая память $O(|N|\cdot\min\{V, S\})$.

39. *Упаковка в контейнеры* при фиксированном числе контейнеров K (см. задачу 36). Вычислительная сложность алгоритма $O(V^K)$.

40. Доказать сильную NP-полноту задачи *Расписание для многопроцессорной системы*. Заданы конечное множество заданий N, общий директивный срок T и для каждого $i \in N$ длительность t_i. Существует ли m-процессорное расписание без прерываний, при котором все задания завершатся не позднее срока T?

41. Доказать, что "жадный" алгоритм построения максимального остовного дерева в графе даёт оптимальное решение.

Для задач 42, 43 *привести полиномиальные приближенные алгоритмы с оценкой* $r_1 \le 2$.
(*Алгоритм имеет оценку* $r_1 \le K$ ($r_2 \le K$)*, если стоимости находимых им решений не более чем в K раз* (*не более чем на K*) *отличаются от стоимостей оптимальных решений.*)

42. Оптимизационная задача *расписание для многопроцессорной системы* (см. задачу 40), в которой требуется минимизировать длину расписания.

43. Оптимизационный вариант задачи *вершинное покрытие* (найти вершинное покрытие в графе, содержащее наименьшее число вершин).

44. Доказать, что если $P \ne NP$, то не существует полиномиальных приближенных алгоритмов решения оптимизационной задачи коммивояжера с оценками $r_1 \le K$, $r_2 \le K$, где K - константа.

45. Доказать, что задачи нахождения (а) оптимального по быстродействию расписания без прерываний в многопроцессорной системе с идентичными процессорами и заданными длительностями выполнения работ, (б) минимальной клики в графе, (в) кратчайшего маршрута коммивояжера являются NP-трудными и NP-лёгкими.

46. Доказать, что если $P = NP$, то для задачи нахождения минимального вершинного покрытия в графе существует полиномиальный алгоритм.

47. Привести CRCW-алгоритм (с временем работы $O(1)$) нахождения максимального элемента в массиве из n чисел с использованием n^2 процессоров.

48. Привести EREW-алгоритм, который для каждого элемента n-элементного списка определяет, является ли он средним ($\lceil n/2 \rceil$-м), за время $O(\log n)$.

49. Привести EREW-алгоритм (с временем работы $O(\log n)$) поиска корней для каждой вершины двоичных деревьев, где n – общее число вершин в деревьях.

50. Привести CRCW-алгоритм (с временем работы $O(1)$) вычисления логического ИЛИ от n аргументов.

Пример варианта индивидуального задания приводится в рабочей программе.

4 Контроль знаний

4.1 Текущий контроль знаний

Оценочные средства для текущего контроля успеваемости и для промежуточной аттестации: зачетная система оценки знаний учащихся.

Текущий контроль за аудиторной и самостоятельной работой обучаемых осуществляется во время проведения практических занятий посредством устного опроса по контрольным вопросам соответствующего раздела, а также проверки отчетов по практическим работам. Промежуточный контроль осуществляется два раза в семестр в виде анализа итоговых отчетов на аттестационные вопросы. Для промежуточной аттестации студентов в семестре проводится контрольное тестирование по вариантам.

4.2 Итоговый контроль знаний

Итоговый контроль осуществляется после успешного прохождения студентами текущего и промежуточного контроля в виде экзамена.

Экзамен сдается в конце семестра. Форма сдачи экзамена – устная. Необходимым условием допуска на экзамен является сдача всех практических работ, индивидуальной зачетной работы, а также достаточный уровень суммарного количества баллов по тестированию (не менее 50%).

Экзаменационный билет содержит два теоретических вопроса. Студент должен дать краткий ответ на первый вопрос и развернутый – на второй. Экзамен проходит в письменной форме с последующей индивидуальной беседой преподавателя с экзаменующимся. На письменную работу над билетом отводится 1 час.

Перечень теоретических вопросов к экзамену приводен в рабочей программе.

Комплекты экзаменационных билетов составляются на основе перечня вопросов, приведенного в рабочей программе, по следующей форме.

ГОУВПО «Амурский государственный университет»

Утверждено на заседании кафедры	Кафедра *математического анализа и моделирования*
«___» _____ 20 г.	
Заведующий кафедрой	Факультет *математики и информатики*
	Курс 4
Утверждаю: _____	Дисциплина "*Программное обеспечение вычислительных систем и сетей*"

Экзаменационный билет 1

1. EREW-алгоритмы решения задач определения номера элемента в списке, параллельная обработка префиксов, вычисление глубины вершины в двоичном дереве.
2. Моделирование угроз. Классификация опасностей. Методы оценки риска ПО.
3. Тестовое задание.

5 Интерактивные технологии и инновационные методы, используемые в образовательном процессе

При преподавании дисциплины используются следующие инновационные технологии и методы: применение мультимедийного проектора при чтении лекций, использование ресурсов сети Internet и электронных учебников при самостоятельной работе студентов, дискуссии в обсуждении проблемных ситуаций при программировании алгоритмов и обсуждении результатов моделирования.