КАЗАНСКИЙ КООПЕРАТИВНЫЙ ИНСТИТУТ (ФИЛИАЛ)
АВТОНОМНОЙ НЕКОММЕРЧЕСКОЙ ОБРАЗОВАТЕЛЬНОЙ
ОРГАНИЗАЦИИ ВЫСШЕГО ОБРАЗОВАНИЯ
ЦЕНТРОСОЮЗА РОССИЙСКОЙ ФЕДЕРАЦИИ
«РОССИЙСКИЙ УНИВЕРСИТЕТ КООПЕРАЦИИ»

ОЦЕНОЧНЫЕ МАТЕРИАЛЫ

ПАРАЛЛЕЛЬНОЕ И НИЗКОУРОВНЕВОЕ ПРОГРАММИРОВАНИЕ

Специальность: 10.05.01 Компьютерная безопасность
Специализация: «Разработка защищенного программного обеспечения»
Формы обучения: очная
Объем дисциплины:
в зачетных единицах: 3 з.е.
в академических часах: 108 ак.ч.

Форма промежуточной аттестации: зачет

2025

Оценочные материалы по дисциплине Параллельное и низкоуровневое программирование

обсуждены и рекомендованы к утверждению решением кафедры гуманитарных дисциплин и иностранных языков от 21 марта 2025 г., протокол № 7

одобрены Научно-методическим советом Казанского кооперативного института (филиала) от «2» апреля 2025 г., протокол № 3.

СОДЕРЖАНИЕ

Целью оценочных материалов по дисциплине (модулю) (далее –ОМ) является комплексная оценка результатов обучения по дисциплине Параллельное и низкоуровневое программирование и установление уровня сформированности компетенций обучающегося.

ОМ предназначены для проведения текущего контроля успеваемости и промежуточной аттестации.

ОМ включают оценочные средства для проведения текущего контроля успеваемости и промежуточной аттестации.

Оценочные материалы разработаны в соответствии с рабочей программой дисциплины Параллельное и низкоуровневое программирование.

1. Паспорт оценочных материалов по дисциплине «Параллельное и низкоуровневое программирование»

Формируемые компетенции (код и наименование компетенции)	Индикаторы достижения компетенций	Планируемые результаты обучения	Контролируемые разделы, темы дисциплины	Наименование оценочного средства[1]	
				Текущий контроль	Промежуточная аттестация
1	2	3	4	5	6
ПК-3.1 Способен определять угрозы безопасности, возможные источники и каналы утечки информации, а также принимать решения по обеспечению защиты информации, в том числе с использованием современных методов и программного инструментария искусственного интеллекта	ПК-3.1.1 Способен разрабатывать комплекс правил, процедур, приемов и методов, средств обеспечения защиты информации, в том числе с использованием современных методов и программного инструментария искусственного интеллекта	Знать: основные принципы и стандарты обеспечения информационной безопасности, классификацию угроз информационной безопасности и методы их выявления, методы и средства защиты информации, включая криптографические методы, средства контроля доступа, системы обнаружения вторжений, архитектуры и принципы работы систем защиты информации. Уметь: разрабатывать правила, процедуры и политики информационной безопасности, выбирать и применять адекватные методы и средства защиты информации в зависимости от угроз и рисков, проводить анализ рисков информационной безопасности и оценивать эффективность существующих мер защиты. Владеть: навыками разработки и внедрения комплексных систем защиты информации, навыками разработки и ведения документации по информационной безопасности, навыками адаптации и применения нормативно-	Тема 1. Принципы построения параллельных вычислительных систем Тема 2. Параллельное программирование в системах с общей памятью Тема 3. Параллельное программирование на основе MPI Тема 4. Многопоточное программирование в Java Тема 5. Основы программирования на Ассемблере	Реферат (Тема 1-5) Тест (тема 1-5) Домашние задания по практическим занятиям (тема 1-5)	Тест (П 1-10 тестового задания)

		правовой базы в области защиты информации.			
	ПК-3.1.2 Способен осуществлять оценивание последствий от реализации угроз безопасности информации в КС и автоматизированных системах	Знать: классификацию угроз безопасности информации в КС и автоматизированных системах, основные типы активов, подлежащих защите в КС и автоматизированных системах, методы и модели оценки рисков информационной безопасности (качественные и количественные). Уметь: идентифицировать и классифицировать угрозы безопасности информации в КС и автоматизированных системах, определять активы, подверженные угрозам, и оценивать их ценность, выбирать и применять подходящие методы и модели для оценки рисков информационной безопасности. Владеть: навыками проведения полного цикла оценки рисков информационной безопасности в КС и автоматизированных системах, навыками применения различных методов и моделей оценки рисков на практике, навыками анализа и интерпретации результатов оценки рисков, навыками оценки последствий от реализации угроз безопасности информации.	Тема 1. Принципы построения параллельных вычислительных систем Тема 2. Параллельное программирование в системах с общей памятью Тема 3. Параллельное программирование на основе MPI Тема 4. Многопоточное программирование в Java Тема 5. Основы программирования на Ассемблере	Реферат (Тема 1-5) Тест (тема 1-5) Домашние задания по практическим занятиям (тема 1-5)	Тест (П 11-20 тестового задания)
	ПК-3.1.3 Способен разрабатывать подходы к классификации и прогнозированию	Знать: основные типы угроз безопасности информации (внутренние, внешние, программные, аппаратные и т.д.), методы классификации угроз по различным критериям (источник, объект воздействия,	Тема 1. Принципы построения параллельных вычислительных систем	Реферат (Тема 1-5) Тест (тема 1-5) Домашние задания по	Тест (П 21-30 тестового задания)

	угроз безопасности информации	последствия и т.д.), индикаторы компрометации (IOC) и методы их выявления, источники информации об угрозах (открытые базы данных, отчеты об инцидентах, данные разведывательных служб и т.д.). Уметь: собирать и анализировать информацию об угрозах безопасности информации из различных источников., классифицировать угрозы по различным критериям, выявлять индикаторы компрометации, применять различные методы прогнозирования угроз (статистические, машинное обучение, экспертные оценки). Владеть: навыками работы с информацией об угрозах безопасности информации, навыками применения различных методов классификации и прогнозирования угроз, навыками построения моделей угроз.	Тема 2. Параллельное программирование в системах с общей памятью Тема 3. Параллельное программирование на основе MPI Тема 4. Многопоточное программирование в Java Тема 5. Основы программирования на Ассемблере	практическим занятиям (тема 1-5)	
	ПК-3.1.4 Способен оценивать информационные риски в КС и автоматизированных системах	Знать: иосновные понятия и определения, связанные с информационными рисками, методы идентификации активов, угроз и уязвимостей в КС и автоматизированных системах, различные виды ущерба от реализации информационных рисков (финансовый, репутационный, операционный, юридический). Уметь: идентифицировать активы, угрозы и уязвимости в КС и автоматизированных системах, оценивать стоимость активов, применять различные методы оценки рисков (качественные и количественные),	Тема 1. Принципы построения параллельных вычислительных систем Тема 2. Параллельное программирование в системах с общей памятью Тема 3. Параллельное	Реферат (Тема 1-5) Тест (тема 1-5) Домашние задания по практическим занятиям (тема 1-5)	Тест (П 31-40 тестового задания)

		определять уровень риска и его соответствие критериям приемлемости. Владеть: навыками проведения полного цикла оценки информационных рисков в КС и автоматизированных системах, навыками применения различных методологий и методов оценки рисков, навыками анализа и интерпретации результатов оценки рисков, навыками коммуникации с заинтересованными сторонами по вопросам управления информационными рисками.	программировани е на основе MPI Тема 4. Многопоточное программировани е в Java Тема 5. Основы программировани я на Ассемблере		

2. Оценочные материалы по дисциплине «Параллельное и низкоуровневое программирование»

2.1 Оценочные материалы для проведения текущего контроля успеваемости

Тематика рефератов

Тема 1. Принципы построения параллельных вычислительных систем

1. **Уровни и модели параллелизма в современных вычислительных системах.** (Исследование инструкционного, данных, задачнопараллелизма; модели Flynn).
2. **Процессы и потоки (нити) в современных операционных системах: сравнительный анализ.** (Реализация в Windows, Linux; планирование, контекст).
3. **Проблемы параллелизма: состояние гонки, взаимная блокировка (deadlock), голодание (starvation).** Причины возникновения, классификация, методы обнаружения и предотвращения.
4. **Механизмы синхронизации в ядре ОС.** Реализация мьютексов, семафоров, условных переменных на низком уровне.
5. **Классические задачи синхронизации: "Читатели-писатели", "Обедающие философы", "Спящий парикмахер".** Анализ проблем и обзор алгоритмов решения.
6. **Семафоры Дейкстры: теория и практика применения.** История, классификация (двоичные, счетные), операции P и V, примеры использования.
7. **Мониторы Хоара как высокоуровневая абстракция для синхронизации.** Принципы работы, сравнение с семафорами, реализация в современных языках.
8. **Аппаратная поддержка параллелизма: атомарные операции, барьеры памяти.** Исследование инструкций (CAS, TAS), моделей памяти (sequential consistency, x86-TSO).
9. **Влияние архитектуры кэш-памяти на производительность параллельных систем.** Явление "false sharing" и методы борьбы с ним.
10. **Оценка эффективности параллельных алгоритмов: Закон Амдала, Закон Густавсона-Барсиса.** Математический аппарат, границы применимости, практическая интерпретация.
11. **Суперскалярные и VLIW-архитектуры как пример параллелизма на уровне инструкций.** (Конвейеризация, несколько исполнительных устройств).

Тема 2. Параллельное программирование в системах с общей памятью (OpenMP)

1. **Архитектурная модель OpenMP: история, стандарты, область применения.** Обзор версий (1.0-5.x), поддержка разными компиляторами.

2. **Модель исполнения "Fork-Join" в OpenMP.** Принципы создания и уничтожения групп потоков, overhead, способы оптимизации.

3. **Директивы распределения работы в OpenMP:** for**,** sections**,** single**,** task**.** Сравнительный анализ, правила применения, клаузы (schedule, nowait).

4. **Синхронизация в OpenMP: директивы** critical**,** atomic**,** barrier**,** ordered**.** Глубокий анализ, производительность, особенности реализации.

5. **Управление данными в OpenMP: клаузы** shared**,** private**,** firstprivate**,** lastprivate**,** reduction**.** Модель памяти, проблема разделяемых данных, способы ее решения.

6. **Директива** task **и асинхронное выполнение в OpenMP.** Модель задач, зависимость между задачами (depend), применение для рекурсивных и нерегулярных алгоритмов.

7. **Планирование итераций цикла: клауза** schedule **(**static**,** dynamic**,** guided**,** auto**,** runtime**).** Анализ нагрузки, балансировка, влияние на производительность.

8. **Библиотечные функции OpenMP (**omp_get_*, omp_set_***).** Управление количеством потоков, получение идентификаторов, функции времени.

9. **OpenMP для GPU: директива** target **и управление ускорителями.** Модель исполнения, отображение данных, ограничения.

10. **Отладка и профилирование OpenMP-программ.** Типичные ошибки (data races, deadlocks), инструменты (Intel Inspector, ThreadSanitizer).

Тема 3. Параллельное программирование на основе MPI

1. **Архитектура программ с передачей сообщений (MPI): модель SPMD, коммуникаторы, группы процессов.**

2. **Точечные операции в MPI: блокирующие и неблокирующие функции передачи (**MPI_Send**,** MPI_Recv**,** MPI_Isend**,** MPI_Irecv**).** Сравнение, буферизация, режимы передачи.

3. **Коллективные операции в MPI: широковещательная рассылка (**MPI_Bcast**), распределение данных (**MPI_Scatter**), сбор данных (**MPI_Gather**).** Алгоритмы реализации, оптимизация.

4. **Операции редукции (**MPI_Reduce**,** MPI_Allreduce**) и их роль в параллельных вычислениях.** Пользовательские операции, эффективность на разных топологиях.

5. **Виртуальные топологии в MPI (**MPI_Cart_create**,** MPI_Graph_create**).** Отображение процессов на сетку графа, упрощение коммуникаций для задач вычислительной математики.

6. **Односторонние коммуникации (RMA) в MPI.** Создание окон (MPI_Win_create), операции Put/Get/Accumulate, синхронизация.

7. **Динамическое управление процессами в MPI: функции** MPI_Spawn**.** Запуск дочерних процессов, построение гетерогенных приложений.

8. **Смешанная модель программирования: MPI + OpenMP.** Преимущества и недостатки, способы организации ("MPI внутри OpenMP",

"OpenMP внутри MPI"), примеры для многопроцессорных систем.

9. **Отладка и профилирование MPI-программ.** Использование MPI_Wtime, инструментов (Intel Trace Analyzer, Vampir), логирование.

10. **Оценка эффективности MPI-программ: ускорение, эффективность, масштабируемость (сильная и слабая).** Методика проведения экспериментов, анализ результатов.

Тема 4. Многопоточное программирование в Java

1. **Модель памяти Java (JMM): принцип "Happens-Before", видимость изменений, переупорядочивание операций.** Проблемы и гарантии.

2. **Жизненный цикл потока в Java: состояния (**NEW**,** RUNNABLE**,** BLOCKED**,** WAITING**,** TIMED_WAITING**,** TERMINATED**) и переходы между ними.**

3. **Синхронизация в Java: ключевое слово** synchronized **(методы, блоки).** Взаимодействие с JMM, механизм мониторов.

4. **Пакет** java.util.concurrent.locks**: интерфейсы** Lock**,** ReadWriteLock **и их реализации.** Сравнение с synchronized (гибкость, производительность).

5. **Механизмы координации потоков:** wait()**,** notify()**,** notifyAll()**.** Правила использования, типичные ошибки, альтернативы из java.util.concurrent.

6. **Коллекции из пакета** java.util.concurrent **(**ConcurrentHashMap**,** CopyOnWriteArrayList**, блокирующие очереди).** Внутреннее устройство, гарантии производительности и потокобезопасности.

7. **Пулы потоков (**ExecutorService**,** ThreadPoolExecutor**): архитектура, стратегии управления задачами и потоками.** Выбор и настройка под разные типы нагрузок.

8. **Классы синхронизаторы (**CountDownLatch**,** CyclicBarrier**,** Phaser**,** Semaphore**).** Принципы работы, сценарии применения, сравнительный анализ.

9. **Атомарные переменные (**AtomicInteger**,** AtomicReference **и др.) и неблокирующие алгоритмы.** Реализация на основе CAS (Compare-And-Swap).

10. CompletableFuture **и асинхронное программирование в Java.** Композиция асинхронных операций, комбинирование результатов, обработка исключений.

11. **Потоки-демоны (**Daemon Threads**): назначение, особенности жизненного цикла, практическое применение.**

Тема 5. Основы программирования на Ассемблере

1. **Архитектура x86-64: регистры общего назначения, сегментные регистры, регистры флагов.** Назначение, правила использования.

2. **Организация памяти в архитектуре x86: сегментация, страничная адресация, виртуальная память.** Роль ОС в управлении памятью.

3. **Система команд x86-64: форматы инструкций, операнды, основные группы команд (пересылки, арифметические, логические, управления).**

4. **Арифметические операции в ассемблере: сложение, вычитание,**

умножение, деление. Работа с знаковыми и беззнаковыми числами, обработка переполнения.

5. **Организация условных переходов и циклов на ассемблере.** Использование флагов, инструкции сравнения (CMP) и переходов (Jxx).

6. **Стек и процедуры вызова в ассемблере x86-64.** Соглашения о вызовах (cdecl, stdcall, System V AMD64 ABI), управление кадром стека.

7. **Организация подпрограмм в ассемблере:** CALL**,** RET**, передача параметров, сохранение регистров.**

8. **Работа с массивами и структурами данных в ассемблере.** Адресация элементов, циклы обработки.

9. **Взаимодействие ассемблера и языков высокого уровня (на примере C/C++).** Inline-ассемблер, вызов функций, совместная компоновка.

10. **Оптимизация кода на уровне ассемблера.** Анализ сгенерированного компилятором кода, использование векторных инструкций (SSE, AVX).

11. **Особенности программирования под разные операционные системы (Windows, Linux) на ассемблере x86-64.** Различия в соглашениях о вызовах и системных вызовах.

Критерии оценки выполнения рефератов по учебной дисциплине

Оценка «отлично» выставляется студенту, если тема реферата раскрыта в полной мере и все требования по написанию реферата соблюдены.

Оценка «хорошо» выставляется студенту, если недостаточно раскрыта тема реферата, но все требования по написанию реферата соблюдены.

Оценка «удовлетворительно» выставляется студенту, если недостаточно раскрыта тема реферата, не все требования по написанию реферата соблюдены, присутствуют ошибки и неточности в работе.

Оценка «неудовлетворительно» выставляется студенту, если отсутствует реферат.

Рекомендации по написанию реферата:

Реферат — письменная работа, выполняемая обучающимся в течение определенного срока.

Реферат — краткое точное изложение сущности какого-либо вопроса, темы на основе одной или нескольких книг, монографий или других первоисточников. Реферат должен содержать основные фактические сведения и выводы по рассматриваемому вопросу.

Реферат отвечает на вопрос — что содержится в данной публикации (публикациях). Реферат — не механический пересказ работы, а изложение ее существа.

Кроме реферирования прочитанной литературы, от обучающегося требуется аргументированное изложение собственных мыслей по рассматриваемому вопросу. Тему реферата предлагает преподаватель или сам обучающийся, в последнем случае она должна быть согласованна с

преподавателем.

В реферате нужны развернутые аргументы, рассуждения, сравнения. Материал подается не столько в развитии, сколько в форме констатации или описания.

Содержание реферируемого произведения излагается объективно от имени автора. Если в первичном документе главная мысль сформулирована недостаточно четко, в реферате она должна быть конкретизирована и выделена.

Структура реферата:

1. Титульный лист.

2. После титульного листа на отдельной странице следует оглавление (план, содержание), в котором указаны названия всех разделов (пунктов плана) реферата и номера страниц, указывающие начало этих разделов в тексте реферата.

3. После оглавления следует введение. Объем введения составляет 1,5-2 страницы.

4. Основная часть реферата может иметь одну или несколько глав, состоящих из 2-3 параграфов (подпунктов, разделов) и предполагает осмысленное и логичное изложение главных положений и идей, содержащихся в изученной литературе. В тексте обязательны ссылки на первоисточники. В том случае если цитируется или используется чья-либо неординарная мысль, идея, вывод, приводится какой-либо цифрой материал, таблицу - обязательно сделайте ссылку на того автора у кого вы взяли данный материал.

5. Заключение содержит главные выводы, и итоги из текста основной части, в нем отмечается, как выполнены задачи и достигнуты ли цели, сформулированные во введении.

6. Приложение может включать графики, таблицы, расчеты.

7. Библиография (список литературы) здесь указывается реально использованная для написания реферата литература. Список составляется согласно правилам библиографического описания.

Этапы работы над рефератом.

Работу над рефератом можно условно подразделить на три этапа:

1. Подготовительный этап, включающий изучение предмета исследования;

2. Изложение результатов изучения в виде связного текста;

3. Устное сообщение по теме реферата.

1. Подготовительный этап работы.

Формулировка темы. Подготовительная работа над рефератом начинается с формулировки темы. Тема в концентрированном виде выражает содержание будущего текста, фиксируя как предмет исследования, так и его ожидаемый результат. Для того чтобы работа над рефератом была успешной, необходимо, чтобы тема заключала в себе проблему, скрытый вопрос (даже если наука уже давно дала ответ на этот вопрос, студент, только знакомящийся с соответствующей областью знаний, будет вынужден искать ответ заново, что даст толчок к развитию проблемного, исследовательского мышления).

Поиск источников. Грамотно сформулированная тема зафиксировала

предмет изучения; задача обучающегося — найти информацию, относящуюся к данному предмету и разрешить поставленную проблему. Выполнение этой задачи начинается с поиска источников. На этом этапе необходимо вспомнить, как работать с энциклопедиями и энциклопедическими словарями.

Работа с источниками. Работу с источниками надо начинать с ознакомительного чтения, т. е. просмотреть текст, выделяя его структурные единицы. При ознакомительном чтении закладками отмечаются те страницы, которые требуют более внимательного изучения. В зависимости от результатов ознакомительного чтения выбирается дальнейший способ работы с источником. Если для разрешения поставленной задачи требуется изучение некоторых фрагментов текста, то используется метод выборочного чтения. Если в книге нет подробного оглавления, следует обратить внимание на предметные и именные указатели.

Избранные фрагменты или весь текст (если он целиком имеет отношение к теме) требуют вдумчивого, неторопливого чтения с «мысленной проработкой» материала. Такое чтение предполагает выделение: 1) главного в тексте; 2) основных аргументов; 3) выводов. Особое внимание следует обратить на то, вытекает тезис из аргументов или нет. Необходимо также проанализировать, какие из утверждений автора носят проблематичный, гипотетический характер и уловить скрытые вопросы.

Понятно, что умение таким образом работать с текстом приходит далеко не сразу. Наилучший способ научиться выделять главное в тексте, улавливать проблематичный характер утверждений, давать оценку авторской позиции — это сравнительное чтение, в ходе которого студент знакомится с различными мнениями по одному и тому же вопросу, сравнивает весомость и доказательность аргументов сторон и делает вывод о наибольшей убедительности той или иной позиции.

Создание конспектов для написания реферата.

Подготовительный этап работы завершается созданием конспектов, фиксирующих основные тезисы и аргументы. Здесь важно вспомнить, что конспекты пишутся на одной стороне листа, с полями и достаточным для исправления и ремарок межстрочным расстоянием (эти правила соблюдаются для удобства редактирования). Если в конспектах приводятся цитаты, то непременно должно быть дано указание на источник (автор, название, выходные данные, № страницы).

По завершении предварительного этапа можно переходить непосредственно к созданию текста реферата.

2. Создание текста.

Общие требования к тексту.

Текст реферата должен подчиняться определенным требованиям: он должен раскрывать тему, обладать связностью и цельностью.

Раскрытие темы предполагает, что в тексте реферата излагается относящийся к теме материал и предлагаются пути решения содержащейся в теме проблемы; связность текста предполагает смысловую соотносительность отдельных компонентов, а цельность - смысловую законченность текста.

С точки зрения связности все тексты делятся на тексты-констатации и тексты-рассуждения. Тексты-констатации содержат результаты ознакомления с предметом и фиксируют устойчивые и несомненные суждения. В текстах-рассуждениях одни мысли извлекаются из других, некоторые ставятся под сомнение, дается им оценка, выдвигаются различные предположения.

План реферата.

Универсальный план реферата – введение, основной текст и заключение.

Требования к введению.

Во введении аргументируется актуальность исследования, - т. е. выявляется практическое и теоретическое значение данного исследования. Далее констатируется, что сделано в данной области предшественниками; перечисляются положения, которые должны быть обоснованы. Введение может также содержать обзор источников или экспериментальных данных, уточнение исходных понятий и терминов, сведения о методах исследования. Во введении обязательно формулируются цель и задачи реферата.

Объем введения - в среднем около 10% от общего объема реферата.

Основная часть реферата.

Основная часть реферата раскрывает содержание темы. Она наиболее значительна по объему, наиболее значима и ответственна. В ней обосновываются основные тезисы реферата, приводятся развернутые аргументы, предполагаются гипотезы, касающиеся существа обсуждаемого вопроса. Важно проследить, чтобы основная часть не имела форму монолога. Аргументируя собственную позицию, можно и должно анализировать, и оценивать позиции различных исследователей, с чем-то соглашаться, чему-то возражать, кого-то опровергать. Текст основной части делится на главы, параграфы, пункты. План основной части может быть составлен с использованием различных методов группировки материала: классификации (эмпирические исследования), типологии (теоретические исследования), периодизации (исторические исследования).

Заключение.

Заключение — последняя часть научного текста. В ней краткой и сжатой форме излагаются полученные результаты, представляющие собой ответ на главный вопрос исследования. Здесь же могут намечаться и дальнейшие перспективы развития темы. Небольшое по объему сообщение также не может обойтись без заключительной части - пусть это будут две-три фразы. Но в них должен подводиться итог проделанной работы.

Список использованной литературы.

Реферат любого уровня сложности обязательно сопровождается списком используемой литературы. Названия книг в списке располагают по алфавиту с указанием выходных данных использованных книг.

Требования, предъявляемыек оформлению реферата

Объемы рефератов – от 10-18 машинописных страниц. Работа выполняется на одной стороне листа стандартного формата. По обеим сторонам листа оставляются поля размером 35 мм. слева и 15 мм. справа, рекомендуется шрифт 12-14, интервал - 1,5. Все листы реферата должны быть пронумерованы.

Каждый вопрос в тексте должен иметь заголовок в точном соответствии с наименованием в плане-оглавлении. При написании и оформлении реферата следует избегать типичных ошибок, например, таких:

– поверхностное изложение основных теоретических вопросов выбранной темы, когда автор не понимает, какие проблемы в тексте являются главными, а какие второстепенными,

– в некоторых случаях проблемы, рассматриваемые в разделах, не раскрывают основных аспектов выбранной для реферата темы,

– дословное переписывание книг, статей, заимствования рефератов из интернета и т. д.

При проверке реферата преподавателем оцениваются:

1. Знания и умения на уровне требований стандарта конкретной дисциплины: знание фактического материала, усвоение общих представлений, понятий, идей.

2. Характеристика реализации цели и задач исследования (новизна и актуальность поставленных в реферате проблем, правильность формулирования цели, определения задач исследования, правильность выбора методов решения задач и реализации цели; соответствие выводов решаемым задачам, поставленной цели, убедительность выводов).

3. Степень обоснованности аргументов и обобщений (полнота, глубина, всесторонность раскрытия темы, логичность и последовательность изложения материала, корректность аргументации и системы доказательств, характер и достоверность примеров, иллюстративного материала, широта кругозора автора, наличие знаний интегрированного характера, способность к обобщению).

4. Качество и ценность полученных результатов (степень завершенности реферативного исследования, спорность или однозначность выводов).

5. Использование литературных источников.

6. Культура письменного изложения материала.

7. Культура оформления материалов работы.

Комплект типовых домашних заданий по практическим занятиям

Тема 1. Принципы построения параллельных вычислительных систем

Задание 1.1: Анализ параллелизма в реальных системах

Цель: Научиться идентифицировать виды параллелизма в существующих компьютерных системах.

- **Задача:**
    1. Выберите 3 различных вычислительных устройства (например: смартфон, игровая консоль, сервер, ноутбук).
    2. Для каждого устройства составьте таблицу с анализом:
        - Уровни параллелизма (инструкционный, данных, задач)
        - Количество ядер процессора
        - Наличие GPU и его возможности параллельной обработки

        - Примеры приложений, использующих параллелизм на этом устройстве
- **Требования к отчету:**
    1. Таблица сравнения характеристик
    2. Объяснение, как каждый вид параллелизма проявляется в работе устройств
    3. Выводы о тенденциях развития параллельных архитектур

Задание 1.2: Сравнительный анализ процессов и потоков

Цель: Понимание различий между процессами и потоками на практических примерах.

- **Задача:**
    1. Запустите Диспетчер задач (Windows) или htop (Linux)
    2. Создайте простое приложение, которое создает:
        - 2 независимых процесса
        - 3 потока в одном процессе
    3. Проанализируйте в системных мониторах:
        - Потребление памяти
        - Загрузку CPU
        - Время создания и уничтожения
- **Требования к отчету:**
    1. Скриншоты системных мониторов
    2. Таблица сравнения характеристик
    3. Объяснение различий в поведении

Задание 1.3: Моделирование "Состояния гонки" (Race Condition)

Цель: Понять природу и опасность состояний гонки.

- **Задача:**
    1. Напишите программу на C++/Java/Python, где несколько потоков:
        - Читают общую переменную-счетчик
        - Увеличивают ее значение на 1
        - Записывают обратно
    2. Запустите программу с 10 потоками, каждый выполняет 1000 инкрементов
    3. Зафиксируйте конечное значение счетчика
    4. Повторите эксперимент 10 раз
- **Требования к отчету:**
    1. Исходный код программы
    2. Таблица результатов 10 запусков
    3. Объяснение причин разных результатов
    4. Предложения по устранению проблемы

Задание 1.4: Реализация алгоритма "Обедающие философы"

Цель: Освоить принципы синхронизации и понять проблему взаимной блокировки.

- **Задача:**
    1. Реализуйте классическую задачу "Обедающие философов" с 5 философами
    2. Используйте семафоры для вилок
    3. Продемонстрируйте:
        - Возникновение deadlock'a
        - Решение проблемы (через асимметричный подход или таймауты)
- **Требования к отчету:**
    1. Исходный код двух версий (с deadlock и без)
    2. Визуализация работы алгоритма
    3. Анализ эффективности решения

Задание 1.5: Сравнение механизмов синхронизации
Цель: Сравнить производительность разных механизмов синхронизации.

- **Задача:**
    1. Реализуйте потокобезопасный счетчик с использованием:
        - Мьютексов
        - Атомарных операций
        - Семафоров
    2. Проведите нагрузочное тестирование каждого подхода
    3. Измерьте время выполнения при 1, 2, 4, 8 потоках
- **Требования к отчету:**
    1. Исходный код трех реализаций
    2. Графики производительности
    3. Выводы о применимости каждого подхода

Задание 1.6: Проектирование системы с использованием семафоров
Цель: Научиться применять семафоры для решения практических задач.

- **Задача:**
    1. Смоделируйте работу call-центра:
        - 3 оператора (потоки)
        - Очередь из 10 звонков
        - Семафоры для ограничения доступа к операторам
    2. Реализуйте:
        - Поступление звонков
        - Обслуживание операторами
        - Статистику работы
- **Требования к отчету:**
    1. Блок-схема алгоритма
    2. Исходный код
    3. Пример вывода работы системы

Задание 1.7: Анализ производительности параллельных алгоритмов

Цель: Освоить методы оценки эффективности параллельных вычислений.

- **Задача:**
  1. Реализуйте последовательное и параллельное умножение матриц
  2. Измерьте время выполнения для матриц разного размера (100x100, 500x500, 1000x1000)
  3. Рассчитайте:
     - Ускорение (Speedup)
     - Эффективность (Efficiency)
     - По закону Амдала
- **Требования к отчету:**
  1. Исходный код программ
  2. Таблица с результатами измерений
  3. Графики зависимости ускорения от размера задачи и числа потоков
  4. Анализ соответствия закону Амдала

Задание 1.8: Исследование влияния "False Sharing"

Цель: Понять влияние архитектуры памяти на производительность.

- **Задача:**
  1. Создайте массив структур, где несколько потоков модифицируют разные элементы
  2. Реализуйте два варианта:
     - Элементы расположены близко в памяти (проблема false sharing)
     - Элементы выровнены по размеру кэш-линии
  3. Сравните производительность
- **Требования к отчету:**
  1. Исходный код обоих вариантов
  2. Результаты профилирования
  3. Объяснение наблюдаемой разницы в производительности

Задание 1.9: Моделирование работы планировщика ОС

Цель: Понять принципы планирования процессов и потоков.

- **Задача:**
  1. Реализуйте упрощенный планировщик с алгоритмами:
     - Round Robin
     - Приоритетное планирование
  2. Сгенерируйте набор задач с разными:
     - Временем выполнения
     - Приоритетами
     - I/O операциями
- **Требования к отчету:**
  1. Описание алгоритмов планирования
  2. Исходный код симулятора
  3. Сравнительный анализ эффективности алгоритмов

Задание 1.10: Проектирование параллельной системы обработки данных

Цель: Применить полученные знания для проектирования сложной системы.

- **Задача:**
  1. Спроектируйте систему для обработки потока финансовых транзакций:
     - Чтение транзакций из нескольких источников
     - Валидация данных
     - Агрегация статистики
     - Запись результатов
  2. Определите:
     - Процессы и потоки
     - Механизмы синхронизации
     - Схемы взаимодействия
- **Требования к отчету:**
  1. Архитектурная диаграмма системы
  2. Описание потоков данных
  3. Обоснование выбранных механизмов синхронизации
  4. Оценка потенциальных узких мест

Тема 2. Параллельное программирование в системах с общей памятью

Задание 2.1: Основы модели Fork-Join

Цель: Освоить базовые принципы создания параллельных областей в OpenMP.

- **Задача:**
  1. Напишите программу, которая выводит приветствие от каждого потока
  2. Используйте директиву #pragma omp parallel
  3. Внутри параллельной области выводите:
     - ID потока
     - Общее количество потоков
     - Сообщение "Hello from thread X of Y"
- **Требования:**
  1. Использовать функции omp_get_thread_num() и omp_get_num_threads()
  2. Запустить программу с разным количеством потоков (1, 2, 4, 8)
  3. Проанализировать порядок вывода сообщений
- **Отчет:** исходный код, скриншоты выполнения, анализ поведения программы

Задание 2.2: Параллелизация циклов с различными схемами планирования

Цель: Изучить работу директивы for и клаузы schedule.

- **Задача:**
  1. Создайте программу вычисления интеграла функции ∫sin(x)dx от 0 до π
  2. Реализуйте последовательную и параллельную версии метода прямоугольников
  3. Сравните схемы планирования:
     - schedule(static)
     - schedule(dynamic)
     - schedule(guided)
- **Требования:**
  1. Количество разбиений: 1,000,000
  2. Измерить время выполнения для каждой схемы
  3. Использовать 2, 4, 8 потоков
- **Отчет:** код, таблица времени выполнения, графики ускорения, выводы

Задание 2.3: Работа с разделяемыми и приватными данными

Цель: Освоить клаузы разделения данных в OpenMP.

- **Задача:**
  1. Реализуйте параллельное вычисление числа π методом Монте-Карло
  2. Создайте потокобезопасный счетчик с использованием:
     - reduction
     - critical
     - atomic
  3. Сравните производительность подходов
- **Требования:**
  1. 10,000,000 случайных точек
  2. Измерить время выполнения каждого варианта
  3. Сравнить точность результатов
- **Отчет:** код трех реализаций, сравнительная таблица производительности, анализ

Задание 2.4: Синхронизация потоков

Цель: Изучить механизмы синхронизации в OpenMP.

- **Задача:**
  1. Создайте программу имитации банковских транзакций
  2. Реализуйте:
     - Параллельное пополнение счетов
     - Параллельное снятие средств
     - Проверку баланса
  3. Используйте директивы:
     - critical

    - atomic
    - barrier
- **Требования:**
    1. 100,000 операций
    2. Гарантировать корректность итогового баланса
    3. Измерить накладные расходы синхронизации
- **Отчет:** код, результаты тестирования, анализ эффективности синхронизации

Задание 2.5: Распараллеливание вложенных циклов
Цель: Научиться работать с вложенным параллелизмом.
- **Задача:**
    1. Реализуйте параллельное умножение матриц
    2. Используйте:
        - collapse
        - Вложенные параллельные регионы
        - Распределение по внешнему и внутреннему циклам
- **Требования:**
    1. Матрицы 1000×1000
    2. Сравнить производительность разных подходов
    3. Проанализировать использование кэша
- **Отчет:** код, таблица производительности, рекомендации по оптимизации

Задание 2.6: Работа с секциями
Цель: Освоить распределение независимых задач между потоками.
- **Задача:**
    1. Создайте программу обработки изображения:
        - Размытие
        - Повышение резкости
        - Изменение яркости
    2. Используйте директиву sections
    3. Каждую операцию выполняйте в отдельной секции
- **Требования:**
    1. Изображение не менее 1024×768 пикселей
    2. Синхронизация между операциями
    3. Визуализация результатов
- **Отчет:** код, исходное и обработанное изображение, временные характеристики

Задание 2.7: Оптимизация работы с памятью
Цель: Исследовать влияние выравнивания данных на производительность.
- **Задача:**
    1. Создайте программу обработки большого массива структур
    2. Реализуйте два варианта структур:

        - Без выравнивания (packed)
        - С выравниванием по границе кэш-линии
    3. Сравните производительность операций:
        - Суммирование
        - Поиск
        - Сортировка
- **Требования:**
    1. Массив из 10,000,000 элементов
    2. Использовать aligned и aligned_data
- **Отчет:** код, сравнительный анализ производительности, выводы

Задание 2.8: Работа с задачами (Tasks)
Цель: Освоить асинхронное выполнение задач в OpenMP.
- **Задача:**
    1. Реализуйте параллельный обход дерева каталогов
    2. Используйте директиву task
    3. Для каждого файла выполняйте:
        - Подсчет хэш-суммы
        - Анализ размера
        - Проверку прав доступа
- **Требования:**
    1. Обработать не менее 1000 файлов
    2. Использовать depend для управления зависимостями
    3. Ограничить количество одновременно выполняемых задач
- **Отчет:** код, статистика обработки, анализ эффективности

Задание 2.9: Обработка ошибок в параллельных программах
Цель: Научиться обрабатывать исключения в многопоточной среде.
- **Задача:**
    1. Создайте программу численного решения системы уравнений
    2. Реализуйте обработку:
        - Деления на ноль
        - Переполнения
        - Нечисловых результатов
    3. Используйте механизмы:
        - try-catch внутри параллельной области
        - Отмена задач
- **Требования:**
    1. Гарантировать завершение всех потоков при ошибке
    2. Сохранять частичные результаты
- **Отчет:** код, примеры обработки ошибок, рекомендации

Задание 2.10: Комплексная оптимизация параллельного приложения
Цель: Применить комплексный подход к оптимизации производительности.

- **Задача:**
    1. Возьмите последовательную программу обработки данных (на выбор)
    2. Проведите поэтапную оптимизацию:
        - Профилирование
        - Векторизация
        - Параллелизация
        - Оптимизация работы с памятью
- **Требования:**
    1. Достичь ускорения не менее 3×
    2. Сохранить корректность результатов
    3. Проанализировать каждый этап оптимизации
- **Отчет:** исходный и оптимизированный код, детальный отчет об оптимизации

Тема 3. Параллельное программирование на основе MPI

Задание 3.1: Основы MPI-программирования - Hello World

Цель: Освоить создание базовой MPI-программы и работу с коммуникаторами.

- **Задача:**
    1. Напишите программу, которая выводит информацию о каждом процессе
    2. Используйте функции:
        - MPI_Init()
        - MPI_Comm_rank()
        - MPI_Comm_size()
        - MPI_Finalize()
    3. Организуйте вывод в формате: "Process X of Y on node Z"
- **Требования:**
    1. Запустить на 2, 4, 8 процессах
    2. Добавить задержку для демонстрации асинхронности
    3. Реализовать сбор статистики по времени выполнения
- **Отчет:** исходный код, вывод программы, анализ поведения при разном количестве процессов

Задание 3.2: Блокирующие операции передачи данных

Цель: Изучить двухточечные блокирующие операции.

- **Задача:**
    1. Реализуйте кольцевую передачу данных между процессами
    2. Каждый процесс должен:
        - Отправить данные следующему процессу
        - Принять данные от предыдущего процесса
        - Модифицировать данные (например, добавить свой rank)
- **Требования:**

1. Использовать MPI_Send() и MPI_Recv()
2. Обработать граничные условия (кольцевая топология)
3. Передать массив из 1000 целых чисел

- **Отчет:** код программы, схема коммуникаций, анализ времени передачи

Задание 3.3: Неблокирующие операции и проверка завершения
Цель: Освоить асинхронные операции передачи данных.

- **Задача:**
  1. Реализуйте параллельное вычисление числа π методом прямоугольников
  2. Используйте неблокирующие операции:
     - MPI_Isend()
     - MPI_Irecv()
     - MPI_Wait() или MPI_Test()
  3. Организуйте перекрытие вычислений и коммуникаций
- **Требования:**
  1. 10,000,000 интервалов
  2. Сравнить производительность с блокирующим вариантом
  3. Измерить эффективность перекрытия
- **Отчет:** код двух реализаций, сравнительный анализ производительности

Задание 3.4: Коллективные операции - широковещательная рассылка
Цель: Изучить коллективные операции на примере broadcast.

- **Задача:**
  1. Реализуйте параллельную сортировку слиянием
  2. Процесс 0 генерирует массив для сортировки
  3. Используйте MPI_Bcast() для рассылки данных
  4. Реализуйте распределенную сортировку
- **Требования:**
  1. Массив из 1,000,000 элементов
  2. Сравнить с последовательной версией
  3. Проанализировать масштабируемость
- **Отчет:** код программы, графики производительности, анализ эффективности

Задание 3.5: Операции распределения и сбора данных
Цель: Освоить операции Scatter и Gather.

- **Задача:**
  1. Реализуйте параллельное умножение матрицы на вектор
  2. Используйте:
     - MPI_Scatter() для распределения строк матрицы
     - MPI_Gather() для сбора результатов
  3. Сравните с ручной реализацией через Send/Recv
- **Требования:**

1. Матрица 5000×5000
2. Вектор 5000 элементов
3. Измерить время выполнения для разного числа процессов

- **Отчет:** код, таблица производительности, анализ накладных расходов

Задание 3.6: Операции редукции
Цель: Изучить коллективные операции редукции.

- **Задача:**
  1. Реализуйте параллельное вычисление скалярного произведения векторов
  2. Используйте MPI_Reduce() для суммирования частичных результатов
  3. Создайте пользовательскую операцию для поиска максимума и его индекса
- **Требования:**
  1. Векторы размером 10,000,000 элементов
  2. Реализовать стандартные и пользовательские операции
  3. Сравнить с последовательной версией
- **Отчет:** код, включая пользовательские операции, анализ точности и производительности

Задание 3.7: Работа с виртуальными топологиями
Цель: Освоить создание и использование виртуальных топологий.

- **Задача:**
  1. Реализуйте решение уравнения теплопроводности на сетке
  2. Создайте декартову топологию с помощью MPI_Cart_create()
  3. Используйте MPI_Cart_shift() для обмена граничными условиями
- **Требования:**
  1. Сетка 1000×1000 точек
  2. 1000 итераций
  3. Визуализация результата
- **Отчет:** код, схема топологии, графики сходимости, анализ эффективности

Задание 3.8: Односторонние коммуникации (RMA)
Цель: Изучить механизм удаленного доступа к памяти.

- **Задача:**
  1. Реализуйте параллельный алгоритм гистограммирования
  2. Используйте операции RMA:
     - MPI_Win_create()
     - MPI_Put()
     - MPI_Get()
  3. Сравните производительность с двухточечными операциями
- **Требования:**
  1. 100,000,000 элементов данных

2. 256 корзин гистограммы
3. Гарантировать атомарность операций

- **Отчет:** код двух реализаций, сравнительный анализ, рекомендации по применению

Задание 3.9: Динамическое управление процессами

Цель: Освоить создание и управление динамическими процессами.

- **Задача:**
  1. Реализуйте систему "мастер-воркер" для обработки задач
  2. Используйте MPI_Spawn() для создания рабочих процессов
  3. Организуйте динамическую балансировку нагрузки
- **Требования:**
  1. 1000 независимых задач
  2. Разное время выполнения задач
  3. Мониторинг загрузки процессов
- **Отчет:** код системы, статистика выполнения, анализ эффективности балансировки

Задание 3.10: Комплексное приложение - параллельная СУБД

Цель: Применить различные техники MPI для создания сложного приложения.

- **Задача:**
  1. Реализуйте упрощенную параллельную СУБД с поддержкой:
     - Вставки данных
     - Поиска по ключу
     - Агрегации данных
  2. Используйте комбинацию:
     - Коллективных операций
     - Точечных коммуникаций
     - Односторонних операций
- **Требования:**
  1. 1,000,000 записей
  2. Поддержка транзакционности
  3. Масштабируемость до 64 процессов
- **Отчет:** архитектура системы, код ключевых модулей, тесты производительности

Тема 4. Многопоточное программирование в Java

Задание 4.1: Создание и управление потоками

Цель: Освоить базовые механизмы создания и управления потоками в Java.

- **Задача:**
  1. Создайте программу, которая генерирует 10 потоков двумя способами:

        - Наследование от класса Thread
        - Реализация интерфейса Runnable
    2. Каждый поток должен:
        - Выводить свой ID и имя
        - Выполнять вычисление (например, поиск простых чисел в диапазоне)
        - Измерять время выполнения
- **Требования:**
    1. Использовать `Thread.sleep()` для имитации работы
    2. Реализовать корректное ожидание завершения всех потоков
    3. Собрать статистику по времени выполнения
- **Отчет:** исходный код, вывод программы, сравнительный анализ двух подходов

Задание 4.2: Синхронизация с использованием synchronized

Цель: Изучить механизм синхронизации с помощью ключевого слова synchronized.

- **Задача:**
    1. Реализуйте потокобезопасный банковский счет с методами:
        - `deposit(amount)`
        - `withdraw(amount)`
        - `getBalance()`
    2. Создайте 10 потоков, которые выполняют 1000 операций пополнения и снятия
    3. Сравните производительность:
        - Без синхронизации
        - С synchronized методами
        - С synchronized блоками
- **Требования:**
    1. Гарантировать корректность итогового баланса
    2. Измерить производительность каждого подхода
    3. Обнаружить и продемонстрировать race condition
- **Отчет:** код трех реализаций, результаты тестирования, анализ производительности

Задание 4.3: Механизмы wait(), notify(), notifyAll()

Цель: Освоить механизмы координации потоков.

- **Задача:**
    1. Реализуйте паттерн "Producer-Consumer" с ограниченным буфером
    2. Производители генерируют данные
    3. Потребители обрабатывают данные
    4. Используйте wait/notify для координации
- **Требования:**
    1. Буфер на 10 элементов
    2. 3 производителя и 2 потребителя

3. Обработать 100 элементов
4. Гарантировать, что потребители не берут данные из пустого буфера

- **Отчет:** код программы, диаграмма взаимодействия потоков, анализ deadlock'ов

Задание 4.4: Пакет java.util.concurrent.locks
Цель: Изучить альтернативные механизмы синхронизации.

- **Задача:**
  1. Реализуйте потокобезопасный кэш с использованием:
     - ReentrantLock
     - ReadWriteLock
  2. Сравните производительность при разных сценариях:
     - Много чтений, мало записей
     - Много записей
     - Сбалансированная нагрузка
- **Требования:**
  1. Кэш на 1000 элементов
  2. Поддержка времени жизни записей
  3. Статистика hit/miss ratio
- **Отчет:** код реализации, графики производительности, рекомендации по выбору механизма

Задание 4.5: Пулы потоков ExecutorService
Цель: Освоить работу с пулами потоков.

- **Задача:**
  1. Реализуйте веб-краулер, который параллельно скачивает страницы
  2. Используйте разные реализации ExecutorService:
     - FixedThreadPool
     - CachedThreadPool
     - ScheduledThreadPool
  3. Сравните производительность и использование ресурсов
- **Требования:**
  1. Обработать 100 URL
  2. Ограничить время выполнения задачи
  3. Собрать статистику по времени загрузки
- **Отчет:** код краулера, сравнительный анализ пулов, рекомендации по настройке

Задание 4.6: Коллекции java.util.concurrent
Цель: Изучить потокобезопасные коллекции.

- **Задача:**
  1. Реализуйте многопоточную систему обработки заказов:
     - ConcurrentHashMap для хранения заказов
     - BlockingQueue для очереди обработки

        - CopyOnWriteArrayList для лога операций
    2. Создайте нагрузочный тест с 10 потоками-писателями и 5 потоками-читателями
- **Требования:**
    1. Обработать 10,000 заказов
    2. Гарантировать целостность данных
    3. Измерить производительность
- **Отчет:** код системы, результаты нагрузочного тестирования, анализ поведения коллекций

Задание 4.7: Классы синхронизаторы

Цель: Освоить использование CountDownLatch, CyclicBarrier, Semaphore.

- **Задача:**
    1. Реализуйте систему проведения онлайн-игры:
        - CountDownLatch для ожидания подключения игроков
        - CyclicBarrier для синхронизации ходов
        - Semaphore для ограничения доступа к ресурсам
    2. Смоделируйте игру с 4 игроками и 20 раундами
- **Требования:**
    1. Гарантировать одновременный старт
    2. Синхронизировать выполнение ходов
    3. Ограничить параллельный доступ к игровым ресурсам
- **Отчет:** код игры, схема синхронизации, анализ работы синхронизаторов

Задание 4.8: Атомарные классы и неблокирующие алгоритмы

Цель: Изучить атомарные операции и compare-and-swap.

- **Задача:**
    1. Реализуйте потокобезопасный счетчик тремя способами:
        - synchronized
        - AtomicInteger
        - Custom с использованием CAS (на основе AtomicReference)
    2. Проведите нагрузочное тестирование с 20 потоками
- **Требования:**
    1. 1,000,000 операций инкремента
    2. Измерить производительность и масштабируемость
    3. Сравнить fairness разных подходов
- **Отчет:** код трех реализаций, сравнительные графики производительности, анализ

Задание 4.9: CompletableFuture и асинхронное программирование

Цель: Освоить современные подходы к асинхронному программированию.

- **Задача:**
    1. Реализуйте систему агрегации данных из нескольких внешних API

2. Используйте CompletableFuture для:
    - Параллельного выполнения запросов
    - Композиции результатов
    - Обработки ошибок
    - Таймаутов
- **Требования:**
    1. 5 различных внешних сервисов
    2. Агрегация результатов
    3. Обработка частичных сбоев
- **Отчет:** код системы, схемы асинхронных операций, анализ производительности

Задание 4.10: Комплексная система - многопоточный веб-сервер
Цель: Применить все изученные техники в комплексном проекте.
- **Задача:**
    1. Реализуйте простой веб-сервер с:
        - ThreadPool для обработки запросов
        - Concurrent коллекции для кэширования
        - Синхронизаторы для управления подключениями
        - Atomic переменные для статистики
    2. Поддержка функций:
        - Статический контент
        - Динамические страницы
        - Кэширование
        - Мониторинг
- **Требования:**
    1. Поддержка 100 одновременных подключений
    2. Кэш на 1000 элементов
    3. Статистика запросов в реальном времени
- **Отчет:** архитектура системы, код ключевых компонентов, результаты нагрузочного тестирования

Тема 5. Основы программирования на Ассемблере

Задание 5.1: Знакомство с архитектурой x86-64 и системой команд
Цель: Освоить базовые принципы работы с регистрами и простейшими командами.
- **Задача:**
    1. Напишите программу, которая выполняет арифметические операции с регистрами:
        - Сложение двух чисел (RAX + RBX → RCX)
        - Умножение (RDX * R8 → R9)
        - Битовые операции (AND, OR, XOR)
    2. Организуйте передачу параметров через регистры
    3. Реализуйте вывод результатов через системные вызовы

- **Требования:**
    1. Использовать различные группы регистров (64-bit, 32-bit, 16-bit)
    2. Продемонстрировать работу с флагами
    3. Комментировать каждую инструкцию
- **Отчет:** исходный код с комментариями, схема использования регистров, примеры выполнения

Задание 5.2: Работа с памятью и адресацией
Цель: Изучить механизмы адресации и работу с оперативной памятью.
- **Задача:**
    1. Создайте программу для работы с массивами:
        - Инициализация массива из 10 элементов
        - Поиск минимального и максимального значения
        - Вычисление суммы элементов
    2. Используйте различные режимы адресации:
        - Прямая
        - Косвенная
        - Индексная
- **Требования:**
    1. Размер элементов: 4 байта (double words)
    2. Оптимизировать использование памяти
    3. Проверить корректность на разных наборах данных
- **Отчет:** код программы, таблица используемых адресов, анализ эффективности адресации

Задание 5.3: Организация условных переходов и циклов
Цель: Освоить механизмы управления потоком выполнения.
- **Задача:**
    1. Реализуйте алгоритм пузырьковой сортировки массива
    2. Используйте различные условные переходы:
        - JE/JZ, JNE/JNZ
        - JG/JNLE, JL/JNGE
        - JC, JNC
    3. Оптимизируйте алгоритм с помощью флагов
- **Требования:**
    1. Массив из 15 элементов
    2. Визуализация процесса сортировки
    3. Подсчет количества итераций и сравнений
- **Отчет:** код алгоритма, блок-схема программы, анализ работы флагов

Задание 5.4: Процедуры и соглашения о вызовах
Цель: Изучить организацию подпрограмм и стек фреймов.
- **Задача:**
    1. Создайте библиотеку математических функций:
        - Вычисление факториала

        - Возведение в степень
        - Нахождение НОД (алгоритм Евклида)
    2. Реализуйте соглашение о вызовах CDECL
    3. Организуйте передачу параметров через стек
- **Требования:**
    1. Корректное сохранение/восстановление регистров
    2. Очистка стека после вызовов
    3. Рекурсивная реализация факториала
- **Отчет:** код библиотеки, примеры использования, схема стека для рекурсивных вызовов

Задание 5.5: Работа со строками и символьными данными
Цель: Освоить обработку строковых данных на ассемблере.
- **Задача:**
    1. Реализуйте функции для работы со строками:
        - Вычисление длины строки
        - Копирование строк
        - Конкатенация строк
        - Поиск подстроки
    2. Используйте строковые инструкции:
        - MOVSB, MOVSW, MOVSD
        - CMPSB, SCASB
        - STOSB, LODSB
- **Требования:**
    1. Обработка строк длиной до 255 символов
    2. Оптимизация с помощью префикса REP
    3. Обработка нуль-терминированных строк
- **Отчет:** код функций, тестовые примеры, сравнение производительности разных подходов

Задание 5.6: Взаимодействие с языками высокого уровня
Цель: Научиться интегрировать ассемблерный код с C/C++.
- **Задача:**
    1. Создайте программу на C++, которая вызывает ассемблерные функции:
        - Быстрое преобразование Фурье (упрощенная версия)
        - Оптимизированное умножение матриц
    2. Реализуйте inline-ассемблер и отдельные модули
- **Требования:**
    1. Соблюдение соглашения о вызовах
    2. Передача параметров через стек и регистры
    3. Возврат результатов
- **Отчет:** код на C++ и ассемблере, инструкции по компиляции, анализ производительности

Задание 5.7: Работа с системными вызовами ОС
Цель: Освоить взаимодействие с операционной системой.

- **Задача:**
  1. Создайте программу для работы с файловой системой:
     - Создание и удаление файлов
     - Чтение и запись данных
     - Изменение прав доступа
  2. Используйте системные вызовы Linux:
     - sys_open, sys_close
     - sys_read, sys_write
     - sys_creat, sys_unlink
- **Требования:**
  1. Обработка ошибок системных вызовов
  2. Буферизация ввода-вывода
  3. Работа с разными размерами данных
- **Отчет:** код программы, примеры работы с файлами, таблица системных вызовов

Задание 5.8: Оптимизация критических участков кода
Цель: Научиться оптимизировать код на низком уровне.

- **Задача:**
  1. Возьмите алгоритм на C++ и оптимизируйте его на ассемблере:
     - Вычисление полинома
     - Обработка изображений (яркость/контраст)
     - Криптографические преобразования
  2. Примените техники оптимизации:
     - Векторизация (SSE/AVX)
     - Развертывание циклов
     - Переупорядочивание инструкций
- **Требования:**
  1. Достичь ускорения не менее 2x
  2. Сохранить функциональность
  3. Проанализировать производительность
- **Отчет:** исходный и оптимизированный код, замеры производительности, анализ изменений

Задание 5.9: Обработка прерываний и исключений
Цель: Изучить механизмы обработки исключительных ситуаций.

- **Задача:**
  1. Создайте программу с обработчиками:
     - Деление на ноль
     - Переполнение
     - Неверная операция
  2. Реализуйте простой отладчик с точками останова
- **Требования:**

1. Установка собственных обработчиков прерываний
2. Корректное восстановление выполнения
3. Логирование исключительных ситуаций

- **Отчет:** код обработчиков, примеры срабатывания, схема работы прерываний

Задание 5.10: Комплексный проект - простой дисковой ОС
Цель: Применить все полученные знания в комплексном проекте.

- **Задача:**
  1. Разработайте загрузчик и базовую ОС с функциями:
     - Загрузка с дискеты/USB
     - Работа с памятью
     - Простой командный интерпретатор
     - Базовые системные утилиты
- **Требования:**
  1. Поддержка FAT16 файловой системы
  2. Выполнение простых команд (dir, copy, type)
  3. Управление памятью через BIOS прерывания
- **Отчет:** полный исходный код, документация, скриншоты работы

Критерии оценивания выполнения практического задания:

Оценка «отлично» выставляется, если: работа выполнена в полном объеме с соблюдением необходимой последовательности действий; работа проведена в условиях, обеспечивающих получение правильных результатов и выводов; соблюдены правила техники безопасности; в ответе правильно и аккуратно выполняет все записи, таблицы, рисунки, чертежи, графики, вычисления; правильно выполняет анализ ошибок.

Оценка «хорошо» выставляется, если: работа выполнена правильно с учетом 1-2 мелких погрешностей или 2-3 недочетов, исправленных самостоятельно по требованию преподавателя.

Оценка «удовлетворительно» выставляется, если: работа выполнена правильно не менее чем наполовину, допущены 1-2 погрешности или одна грубая ошибка.

Оценка «неудовлетворительно» выставляется, если: допущены две (и более) грубые ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя или работа не выполнена полностью.

Типовые тесты

Тема 1. Принципы построения параллельных вычислительных систем

П1. Какой вид параллелизма предполагает одновременное выполнение разных операций над разными данными?

а) Инструкционный параллелизм
б) Параллелизм данных
в) Задачный параллелизм ✓
г) Векторный параллелизм

П2. Что такое процесс в операционной системе?

а) Легковесная единица выполнения
б) Экземпляр выполняемой программы с собственным адресным пространством ✓
в) Способ организации вычислительных ресурсов
г) Виртуальная машина

П3. Какое из перечисленных НЕ является общим ресурсом в параллельных системах?

а) Процессорное время
б) Оперативная память
в) Файлы на диске
г) Локальные переменные потока ✓

П4. Что из перечисленного является механизмом синхронизации?

а) Семафоры ✓
б) Потоки
в) Процессы
г) Ресурсы

П5. Какая проблема возникает при неправильной синхронизации доступа к общим данным?

а) Утечка памяти
б) Состояние гонки (race condition) ✓
в) Фрагментация памяти
г) Инверсия приоритетов

П6. Семафоры используются для:

а) Управления памятью
б) Синхронизации доступа к ресурсам ✓
в) Создания процессов
г) Организации файловой системы

П7. Что такое взаимная блокировка (deadlock)?

а) Ускорение выполнения программы
б) Ситуация, когда процессы ожидают ресурсы, занятые друг другом ✓
в) Ошибка выделения памяти
г) Нарушение последовательности выполнения

П8. Какой механизм обеспечивает эксклюзивный доступ к ресурсу?

а) Поток
б) Процесс
в) Мьютекс ✓
г) Память

П9. Что такое поток (thread)?

а) Независимая программа

б) Легковесная единица выполнения внутри процесса ✓

в) Аппаратный компонент

г) Системный вызов

П10. Какая характеристика описывает способность системы работать быстрее при добавлении ресурсов?

а) Надежность

б) Масштабируемость ✓

в) Безопасность

г) Совместимость

Тема 2. Параллельное программирование в системах с общей памятью

П1. Какая директива OpenMP создает параллельную область?

а) #pragma omp for

б) #pragma omp parallel ✓

в) #pragma omp sections

г) #pragma omp single

П2. Что делает директива #pragma omp for**?**

а) Создает новые потоки

б) Распределяет итерации цикла между потоками ✓

в) Синхронизирует потоки

г) Управляет памятью

П3. Какая клауза OpenMP обеспечивает редукцию переменной?

а) private

б) shared

в) reduction ✓

г) copyin

П4. Что такое модель выполнения "fork-join" в OpenMP?

а) Последовательное выполнение

б) Создание и уничтожение потоков для параллельных регионов ✓

в) Обмен сообщениями

г) Распределенная память

П5. Какая директива используется для распределения независимых блоков кода между потоками?

а) #pragma omp for

б) #pragma omp sections ✓

в) #pragma omp parallel

г) #pragma omp atomic

П6. Какая функция возвращает количество потоков в текущей параллельной области?

а) omp_get_thread_num()

б) omp_get_num_threads() ✓
в) omp_get_max_threads()
г) omp_in_parallel()

П7. Какая клауза делает переменную приватной для каждого потока?

а) shared
б) private ✓
в) default
г) nowait

П8. Что обеспечивает директива #pragma omp critical**?**

а) Параллельное выполнение
б) Атомарный доступ к блоку кода ✓
в) Распределение данных
г) Создание задач

П9. Какая клауза schedule **обеспечивает динамическое распределение итераций?**

а) static
б) dynamic ✓
в) guided
г) auto

П10. Что делает директива #pragma omp barrier**?**

а) Создает барьер синхронизации для всех потоков ✓
б) Завершает параллельную область
в) Распределяет работу
г) Управляет памятью

Тема 3. Параллельное программирование на основе MPI

П1. Что означает аббревиатура MPI?

а) Message Passing Interface ✓
б) Multi-Process Integration
в) Memory Parallel Interface
г) Multi-Programming Interface

П2. Какая функция инициализирует среду MPI?

а) MPI_Comm_size()
б) MPI_Comm_rank()
в) MPI_Init() ✓
г) MPI_Finalize()

П3. Какая функция возвращает ранг текущего процесса в коммуникаторе?

а) MPI_Comm_size()
б) MPI_Comm_rank() ✓
в) MPI_Get_processor_name()
г) MPI_Init()

П4. Какая операция MPI используется для рассылки данных от одного процесса всем остальным?

а) MPI_Scatter()

б) MPI_Gather()

в) MPI_Bcast() ✓

г) MPI_Reduce()

П5. Какая функция выполняет блокирующую отправку сообщения?

а) MPI_Isend()

б) MPI_Send() ✓

в) MPI_Recv()

г) MPI_Wait()

П6. Что такое коммуникатор в MPI?

а) Тип данных

б) Группа процессов для взаимодействия ✓

в) Буфер сообщений

г) Процессор

П7. Какая операция собирает данные со всех процессов в одном процессе?

а) MPI_Bcast()

б) MPI_Scatter()

в) MPI_Gather() ✓

г) MPI_Barrier()

П8. Какая функция выполняет неблокирующий прием сообщения?

а) MPI_Recv()

б) MPI_Irecv() ✓

в) MPI_Send()

г) MPI_Isend()

П9. Какая операция выполняет глобальную редукцию данных?

а) MPI_Gather()

б) MPI_Scatter()

в) MPI_Reduce() ✓

г) MPI_Bcast()

П10. Что измеряет ускорение (speedup) в параллельных вычислениях?

а) Энергопотребление

б) Отношение времени последовательной и параллельной версий ✓

в) Количество процессов

г) Объем памяти

Тема 4. Многопоточное программирование в Java

П1. Какой интерфейс должен быть реализован для создания потока в Java?

а) Thread

б) Run

в) Runnable ✓
г) Executable

П2. Какой метод запускает выполнение потока в Java?

а) run()
б) start() ✓
в) execute()
г) begin()

П3. Что такое ключевое слово synchronized **в Java?**

а) Модификатор класса
б) Средство синхронизации доступа к методам или блокам ✓
в) Тип данных
г) Оператор цикла

П4. Какое состояние потока НЕ является валидным в Java?

а) NEW
б) RUNNABLE
в) BLOCKED
г) EXECUTING ✓

П5. Что такое daemon-поток в Java?

а) Поток с высоким приоритетом
б) Поток, который завершается при завершении всех не-daemon потоков ✓
в) Поток, который нельзя остановить
г) Системный поток

П6. Какой метод используется для ожидания завершения потока?

а) sleep()
б) wait()
в) join() ✓
г) pause()

П7. Что такое InterruptedException **в Java?**

а) Ошибка синхронизации
б) Исключение при прерывании потока ✓
в) Ошибка памяти
г) Исключение ввода-вывода

П8. Какой класс используется для атомарных операций с целыми числами?

а) AtomicInt
б) AtomicInteger ✓
в) SyncInteger
г) ThreadSafeInt

П9. Что такое монитор в контексте многопоточности Java?

а) Устройство вывода
б) Механизм синхронизации, связанный с каждым объектом ✓
в) Тип данных
г) Системный процесс

П10. Какой метод позволяет потоку уступить процессорное время?

а) stop()

б) yield() ✓

в) pause()

г) wait()

Тема 5. Основы программирования на Ассемблере

П1. Что такое регистр в архитектуре ЭВМ?

а) Ячейка оперативной памяти

б) Быстрая память небольшого объема внутри процессора ✓

в) Внешнее устройство хранения

г) Системная шина

П2. Какие регистры используются для арифметических операций в x86?

а) Сегментные регистры

б) Регистры общего назначения ✓

в) Регистры управления

г) Регистры отладки

П3. Что такое флаги (flags) в процессоре?

а) Команды ассемблера

б) Специальные биты, отражающие состояние операций ✓

в) Регистры памяти

г) Системные вызовы

П4. Какая команда используется для сложения в ассемблере x86?

а) ADD ✓

б) SUB

в) MOV

г) CMP

П5. Что делает команда MOV **в ассемблере?**

а) Выполняет арифметическую операцию

б) Пересылает данные между регистрами и памятью ✓

в) Сравнивает значения

г) Изменяет флаги

П6. Какая команда используется для условного перехода?

а) JMP

б) CALL

в) Jcc (условный переход) ✓

г) RET

П7. Что такое мультиуровневое программирование?

а) Использование нескольких языков программирования

б) Совмещение высокоуровневого и низкоуровневого кода ✓

в) Программирование для многоядерных систем

г) Создание многоуровневых архитектур

П8. Какая команда используется для вызова подпрограммы?

а) JMP

б) CALL ✓

в) RET

г) INT

П9. Что такое стек в архитектуре x86?

а) Регистр общего назначения

б) Область памяти для хранения временных данных ✓

в) Системная шина

г) Кэш-память

П10. Какая команда используется для возврата из подпрограммы?

а) JMP

б) CALL

в) RET ✓

г) EXIT

Критерии оценивания тестовых заданий для текущего контроля

Максимальное количество баллов за решение одного теста составляет 5 баллов, которые переводятся в академические оценки следующим образом:

- оценка «отлично» (5 баллов) выставляется при условии правильного ответа студента не менее чем на 85% контрольных заданий

- оценка «хорошо» (4 балла) выставляется при условии правильного ответа студента не менее чем на 70% контрольных заданий

- оценка «удовлетворительно» (3 балла) выставляется при условии правильного ответа студента не менее чем на 51% контрольных заданий

- оценка «неудовлетворительно» (1-2 балла) выставляется при условии правильного ответа студента менее чем на 50% контрольных заданий

2.2 Оценочные материалы для проведения промежуточной аттестации

ПК-3.1.1 Способен разрабатывать комплекс правил, процедур, приемов и методов, средств обеспечения защиты информации, в том числе с использованием современных методов и программного инструментария искусственного интеллекта

П1. Какой механизм синхронизации из Тема 1 обеспечивает безопасный доступ к общим ресурсам в системах защиты информации?

а) Несинхронизированные потоки;

б) Глобальные переменные;

в) Семафоры для контроля доступа к критическим секциям; [✓]

г) Условные переходы.

Ответ: в) Семафоры для контроля доступа к критическим секциям

П2. Какая директива OpenMP должна использоваться для защиты общих данных в системах обработки конфиденциальной информации?

а) #pragma omp for;
б) #pragma omp critical для исключения состояний гонки; [✓]
в) #pragma omp parallel;
г) #pragma omp sections.
Ответ: б) #pragma omp critical **для исключения состояний гонки**

П3. Какой механизм многопоточного программирования в Java обеспечивает безопасность при работе с криптографическими ключами?
а) Несинхронизированные методы;
б) Глобальные статические переменные;
в) synchronized методы и блоки для гарантии атомарности операций; [✓]
г) Модификатор volatile.
Ответ: в) synchronized **методы и блоки для гарантии атомарности операций**

П4. Какой подход к распределенному программированию наиболее безопасен для систем защиты информации?
а) Общая память без синхронизации;
б) MPI с шифрованием передаваемых сообщений; [✓]
в) Неблокирующие операции без проверки завершения;
г) Централизованное хранение паролей.
Ответ: б) MPI с шифрованием передаваемых сообщений

П5. Установите соответствие между технологией параллельного программирования и механизмом защиты информации:

Технология	Механизм защиты
1. Семафоры	а) Контроль доступа к общим ресурсам в многопоточных системах
2. MPI	б) Безопасная передача сообщений между процессами
3. Synchronized	в) Гарантия атомарности операций в многопоточной среде

Правильный ответ:
1 – а, 2 – б, 3 – в

П6. Установите соответствие между элементом архитектуры ЭВМ и его ролью в защите информации:

Элемент архитектуры	Роль в защите
1. Привилегированные режимы процессора	а) Изоляция выполнения системного кода от пользовательского
2. Защита памяти	б) Предотвращение несанкционированного доступа к областям памяти
3. Системные вызовы	в) Контролируемый доступ к ресурсам операционной системы

Правильный ответ:
1 – а, 2 – б, 3 – в

П7. Установите правильную последовательность разработки защищенного многопоточного приложения:
а) Анализ общих ресурсов и критических секций.

б) Выбор механизмов синхронизации (семафоры, мьютексы).
в) Проектирование архитектуры потоков и процессов.
г) Реализация и тестирование механизмов защиты.
д) Аудит безопасности и проверка на состояния гонки.
Ответ: ВАБГД

П8. Установите правильную последовательность обеспечения безопасности в распределенной системе:
а) Шифрование передаваемых сообщений.
б) Аутентификация процессов.
в) Анализ угроз передаваемых данных.
г) Реализация безопасных каналов связи.
д) Тестирование устойчивости к перехвату сообщений.
Ответ: БВАГД

П9. Как называется принцип, при котором каждый процесс имеет доступ только к тем ресурсам, которые необходимы для выполнения его функций?
Правильный ответ: Принцип минимальных привилегий (Principle of Least Privilege)

П10. Как называется механизм, который предотвращает одновременный доступ нескольких потоков к общим данным?
Правильный ответ: Синхронизация потоков (Thread Synchronization)

ПК-3.1.2 Способен осуществлять оценивание последствий от реализации угроз безопасности информации в КС и автоматизированных системах

П1. Какое последствие может иметь состояние гонки в системе управления доступом?
а) Увеличение производительности системы;
б) Несанкционированный доступ к конфиденциальным данным; [✓]
в) Улучшение безопасности;
г) Автоматическое шифрование данных.
Ответ: б) Несанкционированный доступ к конфиденциальным данным

П2. Как ошибка синхронизации в OpenMP может повлиять на систему обработки платежей?
а) Ускорение обработки платежей;
б) Финансовые потери из-за некорректных транзакций; [✓]
в) Улучшение масштабируемости системы;
г) Автоматическое резервное копирование.
Ответ: б) Финансовые потери из-за некорректных транзакций

П3. Какое последствие может иметь утечка сообщений в MPI-программе для системы защиты информации?
а) Улучшение производительности системы;
б) Перехват конфиденциальных данных злоумышленником; [✓]
в) Повышение надежности передачи данных;
г) Автоматическое обновление системы защиты.

Ответ: б) Перехват конфиденциальных данных злоумышленником

П4. Как некорректная работа daemon-потоков в Java может повлиять на систему мониторинга безопасности?

а) Ускорение работы системы мониторинга;

б) Прекращение работы критических служб безопасности; [✓]

в) Улучшение обнаружения угроз;

г) Автоматическое восстановление после сбоев.

Ответ: б) Прекращение работы критических служб безопасности

П5. Установите соответствие между ошибкой программирования и последствием для безопасности:

Ошибка программирования	Последствие для безопасности
1. Состояние гонки	а) Несанкционированное изменение общих данных
2. Взаимная блокировка	б) Отказ в обслуживании критических систем
3. Утечка памяти	в) Уязвимость к атакам на доступность

Правильный ответ:

1 – а, 2 – б, 3 – в

П6. Установите соответствие между угрозой и ее последствиями для распределенной системы:

Угроза	Последствия
1. Перехват MPI-сообщений	а) Компрометация конфиденциальных данных
2. Подмена процесса в кластере	б) Неавторизованный доступ к вычислительным ресурсам
3. Отказ в обслуживании	в) Нарушение доступности системы

Правильный ответ:

1 – а, 2 – б, 3 – в

П7. Установите правильную последовательность оценки последствий утечки данных:

а) Идентификация компрометированных данных.

б) Оценка ценности утраченной информации.

в) Анализ возможных способов использования утекших данных.

г) Расчет финансовых и репутационных потерь.

д) Разработка мер по минимизации последствий.

Ответ: АБВГД

П8. Установите правильную последовательность анализа последствий сбоя в многопоточной системе:

а) Определение критичности受影响ших функций.

б) Анализ распространения ошибки между потоками.

в) Оценка времени восстановления работоспособности.

г) Расчет ущерба от простоя системы.

д) Разработка плана аварийного восстановления.

Ответ: БАВГД

П9. Как называется ситуация, когда несколько процессов бесконечно ожидают ресурсы, занятые друг другом?

Правильный ответ: Взаимная блокировка (Deadlock)

П10. Как называется ошибка, при которой результат операции зависит от порядка выполнения потоков?

Правильный ответ: Состояние гонки (Race Condition)

ПК-3.1.3 Способен разрабатывать подходы к классификации и прогнозированию угроз безопасности информации

П1. Как классифицируются угрозы, связанные с неправильной синхронизацией процессов?

а) Угрозы доступности сервисов;

б) Угрозы целостности данных из-за состояний гонки; [✓]

в) Угрозы конфиденциальности через перехват трафика;

г) Физические угрозы оборудованию.

Ответ: б) Угрозы целостности данных из-за состояний гонки

П2. Какие угрозы можно прогнозировать при анализе схем распределения работы в OpenMP?

а) Аппаратные сбои процессора;

б) Утечки данных через разделяемую память; [✓]

в) Сетевые атаки на маршрутизаторы;

г) Социальную инженерию.

Ответ: б) Утечки данных через разделяемую память

П3. Как MPI может помочь в классификации угроз для распределенных систем?

а) Измерением температуры процессоров;

б) Анализом моделей взаимодействия процессов и точек атаки; [✓]

в) Мониторингом потребления энергии;

г) Анализом логических ошибок в алгоритмах.

Ответ: б) Анализом моделей взаимодействия процессов и точек атаки

П4. Какие угрозы можно выявить через анализ многопоточных Java-приложений?

а) Вирусные атаки на файловую систему;

б) Deadlock'и в системах аутентификации; [✓]

в) Фишинг-атаки на пользователей;

г) DDoS-атаки на сетевую инфраструктуру.

Ответ: б) Deadlock'и в системах аутентификации

П5. Установите соответствие между уровнем программирования и типом угроз:

Уровень программирования	Тип угроз
1. Ассемблер	а) Атаки на уровне архитектуры процессора
2. Многопоточность	б) Состояния гонки и взаимные блокировки
3. Распределенные системы	в) Перехват и подмена сообщений

Правильный ответ:

1 – а, 2 – б, 3 – в

П6. Установите соответствие между технологией и методами прогнозирования угроз:

Технология	Метод прогнозирования угроз
1. Модели памяти	а) Анализ возможных состояний гонки
2. Механизмы синхронизации	б) Прогнозирование взаимных блокировок
3. Протоколы передачи	в) Предсказание уязвимостей каналов связи

Правильный ответ:

1 – а, 2 – б, 3 – в

П7. Установите правильную последовательность классификации угроз для параллельной системы:

а) Идентификация общих ресурсов и критических секций.
б) Анализ моделей доступа к разделяемым данным.
в) Классификация угроз по уровням воздействия.
г) Определение вероятности реализации угроз.
д) Ранжирование угроз по степени опасности.

Ответ: АБВГД

П8. Установите правильную последовательность прогнозирования угроз в распределенной системе:

а) Анализ топологии системы и точек взаимодействия.
б) Идентификация потенциальных уязвимостей передачи данных.
в) Прогнозирование возможных атак на каналы связи.
г) Оценка устойчивости к отказам узлов.
д) Разработка сценариев реагирования на угрозы.

Ответ: АБВГД

П9. Как называется классификация угроз по критерию нарушения конфиденциальности, целостности или доступности?

Правильный ответ: Модель CIA (Confidentiality, Integrity, Availability)

П10. Как называется метод прогнозирования угроз через анализ возможных атак на систему?

Правильный ответ: Анализ угроз (Threat Analysis) или моделирование угроз (Threat Modeling)

ПК-3.1.4 Способен оценивать информационные риски в КС и автоматизированных системах

П1. Как оценить риск использования семафоров в системе управления доступом?

а) Измерением скорости работы процессора;
б) Анализом вероятности взаимных блокировок; [✓]
в) Оценкой стоимости оборудования;
г) Анализом пользовательского интерфейса.

Ответ: б) Анализом вероятности взаимных блокировок

П2. Какие риски связаны с использованием общей памяти в OpenMP для систем обработки персональных данных?

а) Риск перегрева процессора;

б) Риск несанкционированного доступа к конфиденциальной информации; [✓]

в) Риск программных ошибок в компиляторе;

г) Риск неправильной настройки сети.

Ответ: б) Риск несанкционированного доступа к конфиденциальной информации

П3. Как оценить риски безопасности при использовании MPI в распределенной системе защиты?

а) Измерением времени задержки в сети;

б) Анализом уязвимостей в механизмах передачи сообщений; [✓]

в) Оценкой квалификации администраторов;

г) Анализом документирования кода.

Ответ: б) Анализом уязвимостей в механизмах передачи сообщений

П4. Какие риски возникают при неправильной синхронизации потоков в Java для финансовых приложений?

а) Риск ухудшения пользовательского опыта;

б) Риск финансовых потерь из-за некорректных транзакций; [✓]

в) Риск увеличения времени разработки;

г) Риск несовместимости с браузерами.

Ответ: б) Риск финансовых потерь из-за некорректных транзакций

П5. Установите соответствие между компонентом системы и методом оценки рисков:

Компонент системы	Метод оценки рисков
1. Многопоточное приложение	а) Анализ состояний гонки и взаимных блокировок
2. Распределенная система	б) Оценка уязвимостей передачи сообщений
3. Система с общей памятью	в) Анализ несанкционированного доступа к данным

Правильный ответ:

1 – а, 2 – б, 3 – в

П6. Установите соответствие между типом риска и методом его оценки:

Тип риска	Метод оценки
1. Риск потери целостности	а) Анализ механизмов синхронизации
2. Риск нарушения конфиденциальности	б) Проверка шифрования данных и управления ключами
3. Риск отказа в обслуживании	в) Оценка устойчивости к взаимным блокировкам

Правильный ответ:

1 – а, 2 – б, 3 – в

П7. Установите правильную последовательность оценки рисков для параллельной системы:

а) Идентификация активов и критических ресурсов.

б) Анализ угроз и уязвимостей.

в) Определение вероятности реализации угроз.
г) Оценка возможного ущерба.
д) Расчет уровня риска и приоритизация.
Ответ: АБВГД

П8. Установите правильную последовательность управления рисками в распределенной системе:
а) Разработка мер по снижению рисков.
б) Реализация механизмов защиты.
в) Мониторинг эффективности мер защиты.
г) Переоценка рисков после внедрения мер.
д) Корректировка мер защиты при необходимости.
Ответ: АБВГД

П9. Как называется произведение вероятности реализации угрозы на величину возможного ущерба?
Правильный ответ: Уровень риска (Risk Level)

П10. Как называется процесс уменьшения уровня риска до приемлемого значения?
Правильный ответ: Снижение риска (Risk Mitigation)

Критерии оценки для проведения зачета по дисциплине (тестирование)

Шкала оценивания результатов промежуточной аттестации и критерии выставления оценок.

Оценка	Уровень сформированности компетенции	Критерии
«Отлично» («зачтено»)	Высокий	Выполнено более 80% заданий предложенного теста
«Хорошо» («зачтено»)	Хороший	Выполнено 60-80% заданий предложенного теста
«Удовлетворительно» («зачтено»)	Достаточный	Выполнено 35-60% заданий предложенного теста
«Неудовлетворительно» («не зачтено»)	Недостаточный	Выполнено менее 35% заданий предложенного теста

Обновление оценочных материалов дисциплины

обсуждено и рекомендовано к утверждению решением кафедры наименование кафедры
от ___ _____ 20_ г., протокол № ___

одобрено Научно-методическим советом _____ от ___ _____ 20__ г., протокол №____