МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «ВГУ»)

УТВЕРЖДАЮ

Заведующий кафедрой
цифровых технологий

_____ / Кургалин С.Д.

22.04.2024 г.

РАБОЧАЯ ПРОГРАММА УЧЕБНОЙ ДИСЦИПЛИНЫ

Б1.В.ДВ.01.01 ОПТИМИЗАЦИЯ ПРОГРАММНОГО КОДА НА ЯЗЫКЕ С

1. Код и наименование направления подготовки:
02.04.01 Математика и компьютерные науки

2. Профиль подготовки:
компьютерное моделирование и искусственный интеллект

3. Квалификация выпускника: магистр

4. Форма обучения: очная

5. Кафедра, отвечающая за реализацию дисциплины: цифровых технологий

6. Составители программы:
Романов Александр Викторович, ст. преподаватель

7. Рекомендована:
НМС ФКН (протокол № 5 от 05.03.2024)

8. Учебный год: 2025-2026 **Семестр:** 4

9. **Цели и задачи учебной дисциплины**

Целью освоения учебной дисциплины является овладение студентами методами оптимизации программного кода на языках программирования C/C++, необходимыми для повышения производительности сильно нагруженных математических приложений

Задачи учебной дисциплины:

- дать обзор основных методов оптимизации программного кода на языках C/C++;
- изучить влияние оптимизации на поведение компьютера;
- изучить методы оптимизации с использования строк;
- рассмотреть варианты оптимизации основных алгоритмов;
- изучить способы оптимизации при работе с переменными в динамической памяти;
- рассмотреть способы оптимизации программного кода на уровне инструкций языка;
- дать обзор методов оптимизации сортировки и поиска;
- изучить методы оптимизации структур данных;
- рассмотреть варианты оптимизации ввода-вывода данных;
- дать краткий обзор стратегии оптимизации с использованием параллельности исполнения отдельных участков кода.

10. Место учебной дисциплины в структуре ООП:

Дисциплина относится к вариативной части учебного плана (блок Б1).

Для успешного освоения дисциплины необходимо предварительное изучение математического анализа и программирования.

11. Планируемые результаты обучения по дисциплине/модулю (знания, умения, навыки), соотнесенные с планируемыми результатами освоения образовательной программы (компетенциями) и индикаторами их достижения:

Код	Название компетенции	Код(ы)	Индикатор(ы)	Планируемые результаты обучения
ПК-8	Способен создавать и исследовать новые математические модели в естественных науках, промышленности и бизнесе, с учетом возможностей современных информационных технологий, программирования и компьютерной техники	ПК-8.1	Знает основные методы проектирования и производства программного продукта, принципы построения, структуры и приемы работы с инструментальными средствами, поддерживающими создание программных продуктов и программных комплексов, их сопровождения, администрирования и развития (эволюции)	Знать: методы и средства языков программирования C/C++, оптимизации, документирования и сопровождения программного кода.
		ПК-8.2	Умеет использовать методы проектирования и производства программного продукта, принципы построения, структуры и приемы работы с инструментальными средствами, поддерживающими создание программного продукта	Уметь: использовать средства оптимизации программного кода для разработки и рефакторинга программ, написанных на языках программирования C/C++.
		ПК-8.3	Имеет практический опыт применения указанных выше методов и технологий	Владеть: практическими навыками оптимизации программного кода.
ПК-9	Способен использовать	ПК-9.1	Владеет современными методами разработки и	Знать: методы и средства программ, написанных на

	современные методы разработки и реализации конкретных алгоритмов математических моделей на базе языков программирования и пакетов прикладных программ моделирования.		реализации алгоритмов математических моделей на базе языков и пакетов прикладных программ моделирования	языках С/С++.
		ПК-9.2	Умеет разрабатывать и реализовывать алгоритмы математических моделей на базе языков и пакетов прикладных программ моделирования	Уметь: использовать методы и средства программ, написанных на языках С/С++.
		ПК-9.3	Имеет практический опыт разработки и реализации алгоритмов на базе языков и пакетов прикладных программ моделирования	Владеть: практическими навыками оптимизации программ, написанных на языках С/С++.

12. Объем дисциплины в зачетных единицах/час — 3/108.

Форма промежуточной аттестации: зачет с оценкой

13. Трудоемкость по видам учебной работы

Вид учебной работы		Трудоемкость	
		Всего	По семестрам
			4 семестр
Аудиторные занятия		56	56
в том числе:	лекции	28	28
	практические		
	лабораторные	28	28
Самостоятельная работа		52	52
в том числе: курсовая работа (проект)			
Форма промежуточной аттестации (экзамен)			
Итого:		108	108

13.1. Содержание дисциплины

№ п/п	Наименование раздела дисциплины	Содержание раздела дисциплины	Реализация раздела дисциплины с помощью онлайн-курса, ЭУМК*
		1. Лекции	
1.1	Обзор оптимизации	Оптимизация - часть разработки программного обеспечения. Эффективность оптимизации. Стратегии оптимизации кода на С++. Использование лучших алгоритмов. Использование лучших библиотек. Уменьшение количества выделений памяти и копирований. Устранение вычислений. Использование лучших структур данных. Увеличение параллельности. Оптимизация управления памятью.	
1.2	Оптимизация, влияющая на поведение компьютера	Медленная память. Недоступность байтов. Ограничения на количество памяти. Медленное выполнение команд. Множественные потоки выполнения. Вызовы операционной системы. Порядок выполнения инструкций.	
1.3	Измерение производительности	Закон Амдала. Измерение базовой производительности и постановка целей. Профилирование выполнения программы. Длительно работающий код. Прецизионность,	

		истинность и точность. Измерение времени. Измерение времени с помощью компьютеров. Аппаратная эволюция счетчиков тактов. Циклический возврат. Недетерминистическое поведение. Преодоление проблем измерений. Относительная производительность. Повышение повторяемости с помощью измеряемых модульных тестов. Настройка производительности с использованием метрик. Повышение точности путем усреднения множества итераций. Уменьшение недетерминированности поведения операционной системы путем повышения приоритета. Создание класса-секундомера. Хронометраж функции в тесте. Оценка стоимости кода для поиска узких мест. Оценка стоимости отдельных инструкций C++. Оценка стоимости циклов. Оценка количества повторений вложенных циклов. Оценка циклов с переменным количеством повторений. Распознавание неявных циклов. Распознавание ложных циклов.	
1.4	Оптимизация использования строк	Строки используют динамическое выделение памяти. Строки как значения. Строки выполняют массу копирований. Использование модифицирующих операций. Уменьшение работы с памятью с помощью резервирования для устранения временных значений. Устранение копирования строкового аргумента. Устранение разыменований с помощью итераторов. Устранение копирования возвращаемого значения. Использование массивов символов вместо строк. Устранение преобразования строк. Преобразование между кодировками.	
1.5	Оптимизация алгоритмов	Временная стоимость алгоритмов. Временная стоимость в наилучшем, среднем и наихудшем случаях. Амортизированная временная стоимость. Оптимизации сортировки и поиска. Эффективные алгоритмы поиска. Временная стоимость алгоритмов поиска. Все поиски равноценны при малых n. Эффективные алгоритмы сортировки. Временная стоимость алгоритмов сортировки. Использование информации о входных данных. Шаблоны оптимизации. Предвычисления. Отложенные вычисления. Пакетирование. Кеширование. Оптимизация ожидаемого пути. Хеширование. Двойная проверка.	
1.6	Оптимизация переменных в динамической памяти	Переменные C++. Длительность хранения переменной. Владение переменными. Объекты – значения и объекты — сущности. API динамических переменных C++. Автоматизация владения интеллектуальными указателями. Автоматизация владения динамическими переменными. std::auto_ptr и классы контейнеров. Динамические переменные имеют стоимость времен и выполнения. Статическое создание экземпляров класса. Статическое создание переменных-членов класса. Использование статических структур данных. Применение std::array вместо std::vector. Создание больших буферов в стеке. Статическое создание	

		связанных структур данных. Создание бинарных деревьев в массиве. Использование «главного указателя» для владения динамическими переменными. Уменьшение количества перераспределений динамических переменных. Предварительное выделение памяти для динамических переменных для предотвращения перераспределений. Создание динамических переменных вне циклов. Устранение излишнего копирования. Устранение нежелательного копирования в определении класса. Устранение копирования при вызове из функции. Библиотеки без копирования. Нестандартная семантика копирования. Разделяемое владение сущностями. Перемещение экземпляров в std::vector. Плоские структуры данных.	
1.7	Оптимизация инструкций	Удаление кода из циклов. Кеширование конечного значения цикла. Применение более эффективных инструкций циклов. Изменение направления цикла. Устранение инвариантного кода из циклов. Удаление ненужных вызовов функций из циклов. Удаление скрытых вызовов функций из циклов. Удаление дорогих медленно меняющихся вызовов из циклов. Перемещение циклов в функции для снижения накладных расходов при вызовах. Стоимость вызовов функций. Стоимость указателей на функции. Объявление коротких функций встраиваемыми. Устранение неиспользуемого полиморфизма. Удаление неиспользуемых интерфейсов. Выбор реализации интерфейса во время компоновки. Выбор реализации интерфейса во время компиляции. Исключение применения идиомы PIMPL. Устранение вызовов кода в DLL. Перенесение виртуального деструктора в базовый класс. Оптимизация выражений. Упрощение выражений. Группирование констант. Использование целочисленной арифметики вместо арифметики с плавающей точкой. Замена итеративных вычислений аналитическими выражениями. Идиомы оптимизации потока управления. Применение switch вместо if-elseif-else. Применение виртуальных функций вместо switch или if.	
1.8	Использование лучших библиотек	Оптимизация использования стандартной библиотеки. Философия стандартной библиотеки C++. Вопросы применения стандартной библиотек и C++. Оптимизация существующих библиотек. Проектирование оптимизированных библиотек. Плоские иерархии наследования. Упрощение цепочки вызовов. Упрощение проектирования слоев.	
1.9	Оптимизация сортировки и поиска	Таблицы "ключ/значение" с использованием std::map и std::string. Инструментарий для повышения производительности поиска. Выполнение базовых измерений. Идентификация оптимизируемой деятельности. Разделение оптимизируемой деятельности. Изменение или замена алгоритмов и структур данных. Использование процесса оптимизации для пользовательских абстракций. Оптимизация поиска с	

		использованием std::map. Применение символьных массивов фиксированного размера в качестве ключей std::map. Использование строк в стиле C в качестве ключей std::map. Использование std::set, когда ключ является значением. Оптимизация поиска с использованием заголовочного файла <algorithm>. Таблица «ключ/значение» для поиска в последовательных контейнерах. Бинарный поиск с использованием std::equal_ranqe(). Бинарный поиск с использованием std::lower_bound(). Самостоятельное кодирование бинарного поиска. Самостоятельное кодирование бинарного поиска с использованием strcmp(). Оптимизация поиска в хешированных таблицах «ключ/значение». Хеширование с использованием std::unordered_map. Хеширование с фиксированными символьными массивами в качестве ключей. Хеширование с ключами в виде строк с завершающими нулевым и символами. Хеширование с пользовательской хеш-таблицей. Оптимизация сортировки с использованием стандартной библиотеки C++.	
1.10	Оптимизация структур данных	Последовательные контейнеры. Тип данных элемента. std::vector и std::string. Вставка и удаление в std::vector. Итерирование std::vector. Сортировка std::vector. Поиск в std::vector. Вставка и удаление в std::deque. Итерирование std::deque. Сортировка std::deque. Поиск в std::deque. Вставка и удаление в std::list. Итерирование std::list. Сортировка std::list. Поиск в std::list. Вставка и удаление в std::forward_list. Итерирование std::forward_list. Сортировка std::forward_list. Поиск в std::forward_list. Вставка и удаление в std::map. Итерирование std::map. Сортировка std::map. Поиск в std::map. Вставка и удаление в std::unordered_map. Итерирование std::unordered_map. Поиск в std::unordered_map.	
1.11	Оптимизация ввода-вывода	Создание экономной сигнатуры функции. Сокращение цепочек вызовов. Снижение количества перераспределений. Использование большего входного буфера. Использование построчного чтения. Чтение из std::cin и запись в std::cout.	
1.12	Оптимизация параллельности	Введение в параллельные вычисления. Чередующееся выполнение. Последовательная согласованность. Синхронизация. Атомарность с помощью взаимоисключений. Атомарные аппаратные операции. Возможности параллельности в C++. Потоки. Асинхронные задания. Мьютексы. Блокировки. Условные переменные. Атомарные операции над общими переменными. Барьеры памяти. Оптимизация многопоточных программ C++. Программа без синхронизации. Удаление кода запуска и завершения. Более эффективная синхронизация. Библиотеки для параллельных вычислений.	
1.13	Оптимизация управления	API управления памятью C++. Жизненный	

	памятью	цикл динамических переменных. Функции для выделения и освобождения памяти. Функции управления памятью из библиотеки C. Построение динамических переменных с помощью выражений new. Пользовательский размещающий оператор new. Уничтожение динамических переменных с помощью выражения delete. Высокопроизводительные диспетчеры памяти. Диспетчеры памяти для конкретных классов. Диспетчер памяти для блоков фиксированного размера. Добавление operator new() для конкретного класса. Производительность диспетчера памяти для блоков фиксированного размера. Небезопасные с точки зрения параллельности. Пользовательские аллокаторы стандартной библиотеки. Минимальный аллокатор в C++11. Дополнительные определения для аллокатора C++98. Аллокатор блоков фиксированного размера. Аллокатор блоков фиксированного размера для строк.	
		2. Практические занятия	
2.1	Обзор оптимизации	Оптимизация - часть разработки программного обеспечения. Эффективность оптимизации. Стратегии оптимизации кода на C++. Использование лучших алгоритмов. Использование лучших библиотек. Уменьшение количества выделений памяти и копирований. Устранение вычислений. Использование лучших структур данных. Увеличение параллельности. Оптимизация управления памятью.	
2.2	Оптимизация, влияющая на поведение компьютера	Медленная память. Недоступность байтов. Ограничения на количество памяти. Медленное выполнение команд. Множественные потоки выполнения. Вызовы операционной системы. Порядок выполнения инструкций.	
2.3	Измерение производительности	Закон Амдала. Измерение базовой производительности и постановка целей. Профилирование выполнения программы. Длительно работающий код. Прецизионность, истинность и точность. Измерение времени. Измерение времени с помощью компьютеров. Аппаратная эволюция счетчиков тактов. Циклический возврат. Недетерминистическое поведение. Преодоление проблем измерений. Относительная производительность. Повышение повторяемости с помощью измеряемых модульных тестов. Настройка производительности с использованием метрик. Повышение точности путем усреднения множества итераций. Уменьшение недетерминированности поведения операционной системы путем повышения приоритета. Создание класса-секундомера. Хронометраж функции в тесте. Оценка стоимости кода для поиска узких мест. Оценка стоимости отдельных инструкций C++. Оценка стоимости циклов. Оценка количества повторений вложенных циклов. Оценка циклов с переменным количеством повторений. Распознавание неявных циклов. Распознавание ложных циклов.	

2.4	Оптимизация использования строк	Строки используют динамическое выделение памяти. Строки как значения. Строки выполняют массу копирований. Использование модифицирующих операций. Уменьшение работы с памятью с помощью резервирования для устранения временных значений. Устранение копирования строкового аргумента. Устранение разыменований с помощью итераторов. Устранение копирования возвращаемого значения. Использование массивов символов вместо строк. Устранение преобразования строк. Преобразование между кодировками.	
2.5	Оптимизация алгоритмов	Временная стоимость алгоритмов. Временная стоимость в наилучшем, среднем и наихудшем случаях. Амортизированная временная стоимость. Оптимизации сортировки и поиска. Эффективные алгоритмы поиска. Временная стоимость алгоритмов поиска. Все поиски равноценны при малых n. Эффективные алгоритмы сортировки. Временная стоимость алгоритмов сортировки. Использование информации о входных данных. Шаблоны оптимизации. Предвычисления. Отложенные вычисления. Пакетирование. Кеширование. Оптимизация ожидаемого пути. Хеширование. Двойная проверка.	
2.6	Оптимизация переменных в динамической памяти	Переменные C++. Длительность хранения переменной. Владение переменными. Объекты – значения и объекты — сущности. API динамических переменных C++. Автоматизация владения интеллектуальными указателями. Автоматизация владения динамическими переменными. std::auto_ptr и классы контейнеров. Динамические переменные имеют стоимость времен и выполнения. Статическое создание экземпляров класса. Статическое создание переменных-членов класса. Использование статических структур данных. Применение std::array вместо std::vector. Создание больших буферов в стеке. Статическое создание связанных структур данных. Создание бинарных деревьев в массиве. Использование «главного указателя» для владения динамическими переменными. Уменьшение количества перераспределений динамических переменных. Предварительное выделение памяти для динамических переменных для предотвращения перераспределений. Создание динамических переменных вне циклов. Устранение излишнего копирования. Устранение нежелательного копирования в определении класса. Устранение копирования при вызове из функции. Библиотеки без копирования. Нестандартная семантика копирования. Разделяемое владение сущностями. Перемещение экземпляров в std::vector. Плоские структуры данных.	
2.7	Оптимизация инструкций	Удаление кода из циклов. Кеширование конечного значения цикла. Применение более эффективных инструкций циклов. Изменение направления цикла. Устранение инвариантного кода из циклов. Удаление ненужных вызовов	

		функций из циклов. Удаление скрытых вызовов функций из циклов. Удаление дорогих медленно меняющихся вызовов из циклов. Перемещение циклов в функции для снижения накладных расходов при вызовах. Стоимость вызовов функций. Стоимость указателей на функции. Объявление коротких функций встраиваемыми. Устранение неиспользуемого полиморфизма. Удаление неиспользуемых интерфейсов. Выбор реализации интерфейса во время компоновки. Выбор реализации интерфейса во время компиляции. Исключение применения идиомы PIMPL. Устранение вызовов кода в DLL. Перенесение виртуального деструктора в базовый класс. Оптимизация выражений. Упрощение выражений. Группирование констант. Использование целочисленной арифметики вместо арифметики с плавающей точкой. Замена итеративных вычислений аналитическими выражениями. Идиомы оптимизации потока управления. Применение switch вместо if-elseif-else. Применение виртуальных функций вместо switch или if.	
2.8	Использование лучших библиотек	Оптимизация использования стандартной библиотеки. Философия стандартной библиотеки C++. Вопросы применения стандартной библиотек и C++. Оптимизация существующих библиотек. Проектирование оптимизированных библиотек. Плоские иерархии наследования. Упрощение цепочки вызовов. Упрощение проектирования слоев.	
2.9	Оптимизация сортировки и поиска	Таблицы "ключ/значение" с использованием std::map и std::string. Инструментарий для повышения производительности поиска. Выполнение базовых измерений. Идентификация оптимизируемой деятельности. Разделение оптимизируемой деятельности. Изменение или замена алгоритмов и структур данных. Использование процесса оптимизации для пользовательских абстракций. Оптимизация поиска с использованием std::map. Применение символьных массивов фиксированного размера в качестве ключей std::map. Использование строк в стиле С в качестве ключей std::map. Использование std::set, когда ключ является значением. Оптимизация поиска с использованием заголовочного файла <algorithm>. Таблица «ключ/значение» для поиска в последовательных контейнерах. Бинарный поиск с использованием std::equal_ranqe(). Бинарный поиск с использованием std::lower_bound(). Самостоятельное кодирование бинарного поиска. Самостоятельное кодирование бинарного поиска с использованием strcmp(). Оптимизация поиска в хешированных таблицах «ключ/значение». Хеширование с использованием std::unordered_map. Хеширование с фиксированными символьными массивами в качестве ключей. Хеширование с ключами в виде строк с завершающими нулевым и символами. Хеширование с	

		пользовательской хеш-таблицей. Оптимизация сортировки с использованием стандартной библиотеки C++.	
2.10	Оптимизация структур данных	Последовательные контейнеры. Тип данных элемента. std::vector и std::string. Вставка и удаление в std::vector. Итерирование std::vector. Сортировка std::vector. Поиск в std::vector. Вставка и удаление в std::deque. Итерирование std::deque. Сортировка std::deque. Поиск в std::deque. Вставка и удаление в std::list. Итерирование std::list. Сортировка std::list. Поиск в std::list. Вставка и удаление в std::forward_list. Итерирование std::forward_list. Сортировка std::forward_list. Поиск в std::forward_list. Вставка и удаление в std::map. Итерирование std::map. Сортировка std::map. Поиск в std::map. Вставка и удаление в std::unordered_map. Итерирование std::unordered_map. Поиск в std::unordered_map.	
2.11	Оптимизация ввода-вывода	Создание экономной сигнатуры функции. Сокращение цепочек вызовов. Снижение количества перераспределений. Использование большего входного буфера. Использование построчного чтения. Чтение из std::cin и запись в std::cout.	
2.12	Оптимизация параллельности	Введение в параллельные вычисления. Чередующееся выполнение. Последовательная согласованность. Синхронизация. Атомарность с помощью взаимоисключений. Атомарные аппаратные операции. Возможности параллельности в C++. Потоки. Асинхронные задания. Мьютексы. Блокировки. Условные переменные. Атомарные операции над общими переменными. Барьеры памяти. Оптимизация многопоточных программ C++. Программа без синхронизации. Удаление кода запуска и завершения. Более эффективная синхронизация. Библиотеки для параллельных вычислений.	
2.13	Оптимизация управления памятью	API управления памятью C++. Жизненный цикл динамических переменных. Функции для выделения и освобождения памяти. Функции управления памятью из библиотеки C. Построение динамических переменных с помощью выражений new. Пользовательский размещающий оператор new. Уничтожение динамических переменных с помощью выражения delete. Высокопроизводительные диспетчеры памяти. Диспетчеры памяти для конкретных классов. Диспетчер памяти для блоков фиксированного размера. Добавление operator new() для конкретного класса. Производительность диспетчера памяти для блоков фиксированного размера. Небезопасные с точки зрения параллельности. Пользовательские аллокаторы стандартной библиотеки. Минимальный аллокатор в C++11. Дополнительные определения для аллокатора C++98. Аллокатор блоков фиксированного размера. Аллокатор блоков фиксированного размера для строк.	

13.2. Темы (разделы) дисциплины и виды занятий

№ п/п	Наименование темы (раздела) дисциплины	Виды занятий (количество часов)			
		Лекции	Лабораторные	Самостоятельная работа	Всего
1	Обзор оптимизации	1	1	1	3
2	Оптимизация, влияющая на поведение компьютера	1	1	1	3
3	Измерение производительности	1	1	1	3
4	Оптимизация использования строк	2	2	4	8
5	Оптимизация алгоритмов	2	2	4	8
6	Оптимизация переменных в динамической памяти	3	3	6	12
7	Оптимизация инструкций	3	3	6	12
8	Использование лучших библиотек	2	2	3	7
9	Оптимизация сортировки и поиска	3	3	6	12
10	Оптимизация структур данных	2	2	4	8
11	Оптимизация ввода-вывода	2	2	4	8
12	Оптимизация параллельности	3	3	6	12
13	Оптимизация управления памятью	3	3	6	12
	Итого:	28	28	52	108

14. Методические указания для обучающихся по освоению дисциплины

Освоение дисциплины складывается из аудиторной работы (учебной деятельности, выполняемой под руководством преподавателя) и внеаудиторной работы (учебной деятельности, реализуемой обучающимся самостоятельно).

Аудиторная работа состоит из работы на лекциях и выполнения практических заданий в объёме, предусмотренном учебным планом. Лекция представляет собой последовательное и систематическое изложение учебного материала, направленное на знакомство обучающихся с основными понятиями и теоретическими положениями изучаемой дисциплины.

Лекционные занятия формируют базу для практических занятий, на которых полученные теоретические знания применяются для решения конкретных практических задач. Обучающимся для успешного освоения дисциплины рекомендуется вести конспект лекций и практических занятий.

Самостоятельная работа предполагает углублённое изучение отдельных разделов дисциплины с использованием литературы, рекомендованной преподавателем, а также конспектов лекций, конспектов практических занятий. В качестве плана для самостоятельной работы может быть использован раздел 13.1 настоящей рабочей программы, в котором зафиксированы разделы дисциплины и их содержание. В разделе 13.2 рабочей программы определяется количество часов, отводимое на самостоятельную

работу по каждому разделу дисциплины. Большее количество часов на самостоятельную работу отводится на наиболее трудные разделы дисциплины. Для самостоятельного изучения отдельных разделов дисциплины используется перечень литературы и других ресурсов, перечисленных в пунктах 15 и 16 настоящей рабочей программы. Обязательным элементом самостоятельной работы является выполнение домашнего задания.

Успешность освоения дисциплины определяется систематичностью и глубиной аудиторной и внеаудиторной работы обучающегося.

При использовании дистанционных образовательных технологий и электронного обучения требуется выполнять все указания преподавателей, вовремя подключаться к онлайн-занятиям, ответственно подходить к заданиям для самостоятельной работы.

В рамках дисциплины предусмотрено проведение трёх текущих аттестаций за семестр. Результаты текущей успеваемости учитываются при выставлении оценки по промежуточной аттестации в соответствии с положением П ВГУ 2.1.04.16–2019 «Положение о текущей и промежуточной аттестации знаний, умений и навыков обучающихся на факультете компьютерных наук Воронежского государственного университета с использованием балльно-рейтинговой системы».

Обучение лиц с ограниченными возможностями здоровья осуществляется с учетом их индивидуальных психофизических особенностей и в соответствии с индивидуальной программой реабилитации. Для лиц с нарушением слуха при необходимости допускается присутствие на лекциях и практических занятиях ассистента, а также сурдопереводчиков и тифлосурдопереводчиков. Промежуточная аттестация для лиц с нарушениями слуха проводится в письменной форме, при этом используются общие критерии оценивания. При необходимости время подготовки на зачете может быть увеличено. Для лиц с нарушением зрения допускается аудиальное предоставление информации (например, с использованием программ-синтезаторов речи), а также использование на лекциях звукозаписывающих устройств (диктофонов и т.д.). На лекциях и практических занятиях при необходимости допускается присутствие ассистента. При проведении промежуточной аттестации для лиц с нарушением зрения тестирование может быть заменено на устное собеседование по вопросам. При необходимости время подготовки на экзамене может быть увеличено. Для лиц с нарушениями опорно-двигательного аппарата при необходимости допускается присутствие ассистента на лекциях и практических занятиях. Промежуточная аттестация для лиц с нарушениями опорно-двигательного аппарата проводится на общих основаниях, при необходимости процедура экзамена может быть реализована дистанционно.

15. Перечень основной и дополнительной литературы, ресурсов интернет, необходимых для освоения дисциплины

а) основная литература:

№ п/п	Источник
1	Дейтел, П. С для программистов с введением в С11 : учебное пособие / Дейтел П. ; Дейтел Х. – Москва : ДМК-пресс, 2014. – 544 с. – https://www.studentlibrary.ru/book/ISBN9785970600733.html
2	Лисицин, Д.В. Объектно-ориентированное программирование : практическое пособие / Лисицин Д.В. – Москва : Новосибирский ГТУ, 2010. – 88 с. – https://www.studentlibrary.ru/book/ISBN9785778214545.html
3	Ашарина, И.В. Объектно-ориентированное программирование в С++: лекции и упражнения : учебное пособие / Ашарина И.В. – Москва : Горячая линия - Телеком, 2012. – 320 с. – https://www.studentlibrary.ru/book/ISBN9785991270014.html

б) дополнительная литература:

№ п/п	Источник
1	Малявко, А.А. Параллельное программирование на основе технологий OpenMP, MPI, CUDA : учебное пособие / Малявко А.А. – Москва : Новосибирский ГТУ, 2015. – 116 с. – https://www.studentlibrary.ru/book/ISBN9785778226142.htm
2	Царев, Р.Ю. Программирование на языке Си : учебное пособие / Царев Р.Ю. – Москва : СФУ, 2014. – 108 с. – https://www.studentlibrary.ru/book/ISBN9785763830064.html
3	Введение в объектно-ориентированное программирование (с примерами на C++): учебно-методическое пособие для вузов. Ч. 1 / Воронеж. гос. ун-т; сост. М.К. Чернышов, – Воронеж: ЛОП ВГУ, 2006 – 54 с.
4	Уйманова, Н.А. Основы объектно-ориентированного программирования : практикум / Уйманова Н.А. ; Таспаева М.Г. – Москва : Оренбургский ГУ, 2017. – 768 с. – https://www.studentlibrary.ru/book/ISBN9785741019931.html
5	Иваноа, Г.С. Объектно-ориентированное программирование : учебник / Иваноа Г.С. ; Ничушкина Т.Н. – Москва : МГТУ им. Н.Э. Баумана, 2014. – 455 с. – https://www.studentlibrary.ru/book/ISBN9785703839218.html
6	Мейер, Б. Инструменты, алгоритмы и структуры данных : учебник / Мейер Б. – Москва : ИНТУИТ, 2016. – https://www.studentlibrary.ru/book/intuit_109.html

в) информационные электронно-образовательные ресурсы (официальные ресурсы интернет):

№ п/п	Ресурс
1	ЗНБ ВГУ: https://lib.vsu.ru/
2	Электронно-библиотечная система "Университетская библиотека online": http://biblioclub.ru/
3	Электронно-библиотечная система "Лань": https://e.lanbook.com/
4	Электронно-библиотечная система "Консультант студента": http://www.studmedlib.ru
5	Электронный университет ВГУ: https://edu.vsu.ru/
6	Agner Fog Research – https://www.agner.org

16. Перечень учебно-методического обеспечения для самостоятельной работы

№ п/п	Источник
1	Гергель, В.П. Программирование на современных мультиядерных архитектурах (на примере Intel Xeon Phi) : учебное пособие / Гергель В.П. ; Мееров И.Б. ; Бастраков С.И. – Москва : ИНТУИТ, 2016 – https://www.studentlibrary.ru/book/intuit_278.html
2	Ашарина, И.В. Язык C++ и объектно-ориентированное программирование в C++. Лабораторный практикум : учебное пособие / Ашарина И.В. ; Крупская Ж.Ф. – Москва : Горячая линия - Телеком, 2016. – 232 с. – https://www.studentlibrary.ru/book/ISBN9785991204644.html

17. Образовательные технологии, используемые при реализации учебной дисциплины, включая дистанционные образовательные технологии (ДОТ, электронное обучение (ЭО), смешанное обучение)

При реализации дисциплины могут использоваться технологии электронного обучения и дистанционные образовательные технологии на базе портала edu.vsu.ru, а также другие доступные ресурсы сети Интернет.

18. Материально-техническое обеспечение дисциплины

Аудитория для лекционных занятий: мультимедиа-проектор, экран для проектора, компьютер с выходом в сеть «Интернет». Специализированная мебель (столы ученические, стулья, доска). Программное обеспечение: LibreOffice v.5-7, программа для просмотра файлов формата pdf, браузер.

Аудитория для лабораторных занятий: компьютеры с выходом в сеть «Интернет» и доступом к электронным библиотечным системам, специализированная мебель (столы

ученические, стулья, доска). Программное обеспечение: LibreOffice v.5-7, программа для просмотра файлов формата pdf, браузер.

19. Оценочные средства для проведения текущей и промежуточной аттестаций

Порядок оценки освоения обучающимися учебного материала определяется содержанием следующих разделов дисциплины:

№ п/п	Наименование раздела дисциплины (модуля)	Компетенция	Индикатор(ы) достижения компетенции	Оценочные средства
1	Обзор оптимизации	ПК-8 ПК-9	ПК-8.1 ПК-8.2 ПК-8.3 ПК-9.1 ПК-9.2 ПК-9.3	Лабораторная работа Письменный опрос
2	Оптимизация, влияющая на поведение компьютера	ПК-8 ПК-9	ПК-8.1 ПК-8.2 ПК-8.3 ПК-9.1 ПК-9.2 ПК-9.3	Лабораторная работа Письменный опрос
3	Измерение производительности	ПК-8 ПК-9	ПК-8.1 ПК-8.2 ПК-8.3 ПК-9.1 ПК-9.2 ПК-9.3	Лабораторная работа Письменный опрос
4	Оптимизация использования строк	ПК-8 ПК-9	ПК-8.1 ПК-8.2 ПК-8.3 ПК-9.1 ПК-9.2 ПК-9.3	Лабораторная работа Письменный опрос
5	Оптимизация алгоритмов	ПК-8 ПК-9	ПК-8.1 ПК-8.2 ПК-8.3 ПК-9.1 ПК-9.2 ПК-9.3	Лабораторная работа Письменный опрос
6	Оптимизация переменных в динамической памяти	ПК-8 ПК-9	ПК-8.1 ПК-8.2 ПК-8.3 ПК-9.1 ПК-9.2 ПК-9.3	Лабораторная работа Письменный опрос
7	Оптимизация инструкций	ПК-8 ПК-9	ПК-8.1 ПК-8.2 ПК-8.3 ПК-9.1 ПК-9.2 ПК-9.3	Лабораторная работа Письменный опрос
8	Использование лучших библиотек	ПК-8 ПК-9	ПК-8.1 ПК-8.2 ПК-8.3 ПК-9.1 ПК-9.2 ПК-9.3	Лабораторная работа Письменный опрос
9	Оптимизация сортировки и поиска	ПК-8 ПК-9	ПК-8.1 ПК-8.2 ПК-8.3 ПК-9.1 ПК-9.2 ПК-9.3	Лабораторная работа Письменный опрос
10	Оптимизация структур данных	ПК-8 ПК-9	ПК-8.1 ПК-8.2 ПК-8.3 ПК-9.1 ПК-9.2 ПК-9.3	Лабораторная работа Письменный опрос
11	Оптимизация ввода-вывода	ПК-8 ПК-9	ПК-8.1 ПК-8.2 ПК-8.3 ПК-9.1 ПК-9.2 ПК-9.3	Лабораторная работа Письменный опрос
12	Оптимизация параллельности	ПК-8 ПК-9	ПК-8.1 ПК-8.2 ПК-8.3 ПК-9.1 ПК-9.2 ПК-9.3	Лабораторная работа Письменный опрос
13	Оптимизация управления памятью	ПК-8 ПК-9	ПК-8.1 ПК-8.2 ПК-8.3 ПК-9.1 ПК-9.2 ПК-9.3	Лабораторная работа Письменный опрос
Промежуточная аттестация форма контроля – зачёт с оценкой				Список вопросов к зачёту

20. Типовые оценочные средства и методические материалы, определяющие процедуры оценивания

20.1. Текущий контроль успеваемости

Контроль успеваемости по дисциплине осуществляется с помощью следующих оценочных средств: лабораторная работа, письменный опрос.

Перечень вопросов для письменных опросов

Раздел 1. Обзор оптимизации

1. Основные стратегии оптимизации кода на C++.
2. Эффективность оптимизации.
3. Использование лучших алгоритмов и библиотек.
4. Уменьшение количества выделений памяти и копирований.
5. Устранение вычислений.
6. Использование лучших структур данных.
7. Увеличение параллельности.
8. Оптимизация управления памятью.

Раздел 2. Оптимизация, влияющая на поведение компьютера

1. Влияние скорости и объема памяти на производительность программного кода.
2. Влияние вызовов операционной системы на скорость программ.
3. Влияние на скорость выполнения программ порядка выполнения инструкций.

Раздел 3. Измерение производительности

1. Измерение производительности.
2. Закон Амдала.
3. Оценка стоимости отдельных инструкций C++.
4. Оценка стоимости циклов.
5. Оценка количества повторений вложенных циклов.
6. Оценка циклов с переменным количеством повторений.
7. Распознавание неявных циклов.
8. Распознавание ложных циклов.

Раздел 4. Оптимизация использования строк

1. Использование модифицирующих операций.
2. Уменьшение работы с памятью с помощью резервирования для устранения временных значений.
3. Устранение копирования строкового аргумента.
4. Устранение разыменований с помощью итераторов.
5. Устранение копирования возвращаемого значения.
6. Использование массивов символов вместо строк.
7. Устранение преобразования строк.
8. Преобразование между кодировками.

Раздел 5. Оптимизация алгоритмов

1. Временная стоимость алгоритмов в наилучшем, среднем и наихудшем случаях.

2. Амортизированная временная стоимость.

3. Оптимизации сортировки и поиска.

4. Эффективные алгоритмы поиска. Временная стоимость алгоритмов поиска.

5. Эффективные алгоритмы сортировки. Временная стоимость алгоритмов сортировки.

6. Использование информации о входных данных.

7. Шаблоны оптимизации.Предвычисления. Отложенные вычисления. Пакетирование. Кеширование.

8. Шаблоны оптимизации. Оптимизация ожидаемого пути. Хеширование. Двойная проверка.

Раздел 6. Оптимизация переменных в динамической памяти

1. Длительность хранения переменной. Владение переменными.

2. Объекты – значения и объекты — сущности.

3. API динамических переменных C++.

4. Автоматизация владения интеллектуальными указателями. Автоматизация владения динамическими переменными.

5. Статическое создание экземпляров класса. Статическое создание переменных-членов класса. Использование статических структур данных.

6. Применение std::array вместо std::vector. Создание больших буферов в стеке.

7. Статическое создание связанных структур данных. Создание бинарных деревьев в массиве.

8. Использование «главного указателя» для владения динамическими переменными. Уменьшение количества перераспределений динамических переменных.

9. Предварительное выделение памяти для динамических переменных для предотвращения перераспределений. Создание динамических переменных вне циклов.

10. Устранение излишнего копирования. Устранение нежелательного копирования в определении класса. Устранение копирования при вызове из функции.

11. Библиотеки без копирования. Нестандартная семантика копирования.

12. Разделяемое владение сущностями.

13. Плоские структуры данных.

Раздел 7. Оптимизация инструкций

1. Удаление кода из циклов. Кеширование конечного значения цикла.

2. Применение более эффективных инструкций циклов.

3. Изменение направления цикла.

4. Устранение инвариантного кода из циклов. Удаление ненужных вызовов функций из циклов.

5. Удаление скрытых вызовов функций из циклов. Удаление дорогих медленно меняющихся вызовов из циклов.

6. Перемещение циклов в функции для снижения накладных расходов при вызовах.

7. Стоимость вызовов функций. Стоимость указателей на функции. Объявление коротких функций встраиваемыми.

8. Устранение неиспопьзуемого полиморфизма. Удаление неиспользуемых интерфейсов.

9. Выбор реализации интерфейса во время компоновки. Выбор реализации интерфейса во время компиляции.

10. Оптимизация выражений. Упрощение выражений.

11. Группирование констант.

12. Использование целочисленной арифметики вместо арифметики с ппавающей точкой. Замена итеративных вычислений аналитическими выражениями.

13. Идиомы оптимизации потока управления. Применение switch вместо if-elseif-else. Применение виртуальных функций вместо switch или if.

Раздел 8. Использование лучших библиотек

1. Оптимизация использования стандартной библиотеки.

2. Оптимизация существующих библиотек.

3. Проектирование оптимизированных библиотек.

4. Плоские иерархии наследования.

5. Упрощение цепочки вызовов.

6. Упрощение проектирования слоев.

Раздел 9. Оптимизация сортировки и поиска

1. Таблицы "ключ/значение" с использованием std::map и std::string.

2. Инструментарий для повышения производительности поиска.

3. Идентификация оптимизируемой деятельности. Разделение оптимизируемой деятепьности.

4. Изменение или замена алгоритмов и структур данных.

5. Использование процесса оптимизации для пользовательских абстракций.

6. Оптимизация поиска с использованием std::map.

7. Применение символьных массивов фиксированного размера в качестве ключей std::map. Использование строк в стиле С в качестве ключей std::map. Использование std::set, когда ключ является значением.

8. Оптимизация поиска с использованием заголовочного файла <algorithm>.

9. Таблица «ключ/значение» для поиска в последовательных контейнерах.

10. Бинарный поиск с использованием std::equal_ranqe(). Бинарный поиск с использованием std::lower_bound().

11. Оптимизация поиска в хешированных таблицах «ключ/значение». Хеширование с использованием std::unordered_map.

12. Хеширование с фиксированными символьными массивами в качестве ключей. Хеширование с ключами в виде строк с завершающими нулевым и символами. Хеширование с пользовательской хеш – таблицей.

13. Оптимизация сортировки с использованием стандартной библиотеки C++.

Раздел 10. Оптимизация структур данных

1. Последовательные контейнеры.

2. Тип данных элемента.

3. Вставка и удаление в std::vector. Итерирование std::vector. Сортировка std::vector. Поиск в std::vector.

4. Вставка и удаление в std::deque. Итерирование std::deque. Сортировка std::deque. Поиск в std::deque.

5. Вставка и удаление в std::list. Итерирование std::list. Сортировка std::list. Поиск в std::list.

6. Вставка и удаление в std::forward_list. Итерирование std::forward_list. Сортировка std::forward_list. Поиск в std::forward_list.

7. Вставка и удаление в std::map. Итерирование std::map. Сортировка std::map. Поиск в std::map.

8.Вставка и удаление в std::unordered_map. Итерирование std::unordered_map. Поиск в std::unordered_map.

Раздел 11. Оптимизация ввода-вывода

1. Создание экономной сигнатуры функции.

2. Сокращение цепочек вызовов.

3. Снижение количества перераспределений.

4. Испоnьэование большего входного буфера.

5. Использование построчного чтения.

6. Чтение из std::cin и запись в std::cout.

Раздел 12. Оптимизация параллельности

1. Чередующееся выполнение.

2. Последовательная согласованность.

3. Синхронизация. Атомарность с помощью взаимоисключений. Атомарные аппаратные операции.

4. Возможности параллельности в C++. Потоки.

5. Асинхронные задания.

6. Мьютексы. Блокировки. Условные переменные.

7. Атомарные операции над общими переменными. Барьеры памяти.

8. Оптимизация многопоточных программ C++.

9. Программа без синхронизации.

10. Удаление кода запуска и завершения.

11.Библиотеки для параллельных вычислений.

Раздел 13. Оптимизация управления памятью

1. API управления памятью C++.

2. Жизненный цикл динамических переменных.

3. Функции для выделения и освобождения памяти.

4. Функции управления памятью из библиотеки C.

5. Построение динамических переменных с помощью выражений new. Пользовательский размещающий оператор new.

6. Уничтожение динамических переменных с помощью выражения delete.

7. Высокопроизводитеьные диспетчеры памяти. Диспетчеры памяти для конкретных классов.

8. Диспетчер памяти для блоков фиксированного размера. Добавление operator new() для конкретного класса.

9. Производительность диспетчера памяти для блоков фиксированного размера.

10. Пользовательские аллокаторы стандартной библиотеки.

11. Минимальный аллокатор в C++11. Дополнительные определения для аллокатора C++98.

12. Аллокатор блоков фиксированного размера. Аллокатор блоков фиксированного размера для строк.

Требования к письменным опросам по разделам 1-13

Письменные опросы выполняются студентами в соответствии с полученными вопросами. Ответы на вопросы должны быть правильными, достаточно подробными и выполняться самостоятельно.

Критерии и шкалы оценивания при проведении письменного опроса

Критерии оценивания компетенций	Уровень сформированности компетенций	Шкала оценок
Полное соответствие ответа обучающегося всем перечисленным критериям. Обучающийся демонстрирует высокий уровень владения материалом, ориентируется в предметной области, верно отвечает на все дополнительные вопросы.	Повышенный уровень	Отлично
Ответ на контрольно-измерительный материал не соответствует одному или двум из перечисленных показателей, но обучающийся дает правильные ответы на дополнительные вопросы. Допускаются ошибки при воспроизведении части теоретических положений.	Базовый уровень	Хорошо
Ответ на контрольно-измерительный материал не соответствует любым трём из перечисленных показателей, обучающийся дает неполные ответы на дополнительные вопросы. Сформированные знания основных понятий, определений и теорем, изучаемых в курсе, не всегда полное их понимание с затруднениями при воспроизведении.	Пороговый уровень	Удовлетворительно

Ответ на контрольно-измерительный материал не соответствует любым четырём из перечисленных показателей. Обучающийся демонстрирует отрывочные знания (либо их отсутствие) основных понятий, определений и теорем, используемых в курсе.	–	Неудовлетворительно

Перечень лабораторных работ

Типовое задание для лабораторной работы

Лабораторная работа № 1
«Измерение времени работы программы»

Цель работы: рассмотреть на практике работу с таймерами в языках C/C++, овладеть методикой измерения скорости выполнения программного кода.

Требования к выполнению работы: выполнение лабораторной работы предусматривает написание программы, реализующей выполнение циклов, запуска функций и операторов условного перехода. Квремя выполнения каждого блока программы и отдельных операторов должно быть измерено с помощью инструментов языков программирования C/C++. Отчёт о работе проводится в виде собеседования и заключается в демонстрации работы программы, объяснении принципов работы алгоритма и ответов на дополнительные вопросы.

Критерии оценки: для получении оценки «зачтено» необходимо показать высокий уровень владения теоретическим материалом, уметь объяснить принцип работы написанной программы, верно ответить на дополнительные вопросы.

Задание: написать программу для оценка стоимости отдельных инструкций C++, стоимости циклов, количества повторений вложенных циклов, циклов с переменным количеством повторений. Проверить работу программы на контрольном примере. Оценить время испольнения отдельных блоков программы.

Лабораторная работа № 2
«Оптимизация алгоритмов поиска»

Цель работы: провести оценку эффективности различных алгоритмов поиска.

Требования к выполнению работы: выполнение лабораторной работы предусматривает написание программы, реализующей оценку производительности различных алгоритмов поиска. Отчёт о работе проводится в виде собеседования и заключается в демонстрации работы программы, объяснении принципов работы алгоритма и ответов на дополнительные вопросы.

Критерии оценки: для получении оценки «зачтено» необходимо показать высокий уровень владения теоретическим материалом, уметь объяснить принцип работы написанной программы, верно ответить на дополнительные вопросы.

Задание: написать программу, реализующую любых трех алгоритмов поиска. Проверить работу программы на контрольном примере. Оценить скорость выполнения каждого из алгоритмов.

Лабораторная работа № 3

«Ноптимизация алгоритмов сортировки»

Цель работы: провести оценку эффективности различных алгоритмов сортировки.

Требования к выполнению работы: выполнение лабораторной работы предусматривает написание программы, реализующей оценку производительности различных алгоритмов сортировки. Отчёт о работе проводится в виде собеседования и заключается в демонстрации работы программы, объяснении принципов работы алгоритма и ответов на дополнительные вопросы.

Критерии оценки: для получении оценки «зачтено» необходимо показать высокий уровень владения теоретическим материалом, уметь объяснить принцип работы написанной программы, верно ответить на дополнительные вопросы.

Задание: написать программу, реализующую любых трех алгоритмов сортировки. Проверить работу программы на контрольном примере. Оценить скорость выполнения каждого из алгоритмов.

Лабораторная работа № 4

«Работа с динамическими переменными»

Цель работы: оптимизировать программный код с применением динамических переменных.

Требования к выполнению работы: выполнение лабораторной работы предусматривает написание программы, реализующей использование динамичсеких переменных в циклических структурах. Отчёт о работе проводится в виде собеседования и заключается в демонстрации работы программы, объяснении принципов работы алгоритма и ответов на дополнительные вопросы.

Критерии оценки: для получении оценки «зачтено» необходимо показать высокий уровень владения теоретическим материалом, уметь объяснить принцип работы написанной программы, верно ответить на дополнительные вопросы.

Задание: написать программу, реализующую предварительное выделение памяти для динамических переменных для предотвращения перераспределений., создать динамические переменные вне циклов. Проверить работу программы на контрольном примере.

Лабораторная работа № 5

«Оптимизация копирования»

Цель работы: оптимизировать программный код устранением излишнего копирования.

Требования к выполнению работы: выполнение лабораторной работы предусматривает написание программы, реализующей копирование в определении классов и вызовах функций. Оптимизировать программу устранив лишнее копирование. Отчёт о работе проводится в виде собеседования и заключается в демонстрации работы программы, объяснении принципов работы алгоритма и ответов на дополнительные вопросы.

Критерии оценки: для получении оценки «зачтено» необходимо показать высокий уровень владения теоретическим материалом, уметь объяснить принцип работы написанной программы, верно ответить на дополнительные вопросы.

Задание: написать программу, реализующую определение классов и вызовы функций с использованием и без использования копирования. Проверить работу программы на контрольном примере. Оценить скорость исполнения программного кода с копированием и без копирования.

Лабораторная работа № 6
«Оптимизация циклов»

Цель работы: рассмотреть методы оптимизации циклов в языках C/C++/

Требования к выполнению работы: выполнение лабораторной работы предусматривает написание программы, реализующей различные механизмы оптимизации циклических структур. Отчёт о работе проводится в виде собеседования и заключается в демонстрации работы программы, объяснении принципов работы алгоритма и ответов на дополнительные вопросы.

Критерии оценки: для получении оценки «зачтено» необходимо показать высокий уровень владения теоретическим материалом, уметь объяснить принцип работы написанной программы, верно ответить на дополнительные вопросы.

Задание: написать программу, реализующую кеширование конечного значения цикла, применение более эффективных инструкций циклов, изменение направления цикла, удаление скрытых вызовов функций из циклов. Проверить работу программы на контрольном примере. Оценить время выполнения каждого цикла в программе.

Лабораторная работа № 7
«Оптимизация функций в программе»

Цель работы: рассмотреть механизмы оптимизации функций в языках C/C++/

Требования к выполнению работы: выполнение лабораторной работы предусматривает написание программы, реализующей наиболее распространненое применение функций. Оптимизировать программный код для увеличения скорости исполнения. Отчёт о работе проводится в виде собеседования и заключается в демонстрации работы программы, объяснении принципов работы алгоритма и ответов на дополнительные вопросы.

Критерии оценки: для получении оценки «зачтено» необходимо показать высокий уровень владения теоретическим материалом, уметь объяснить принцип работы написанной программы, верно ответить на дополнительные вопросы.

Задание: написать программу, реализующую перемещение циклов в функции для снижения накладных расходов при вызовах. Оценить стоимость вызовов функций и указателей на функции. Проверить работу программы на контрольном примере.

Лабораторная работа № 8
«Оптимизация выражений»

Цель работы: рассмотреть на практике способы оптимизации выражений в программном коде.

Требования к выполнению работы: выполнение лабораторной работы предусматривает написание программы, реализующей различные механизмы оптимизации выражений в языках C/C++. Отчёт о работе проводится в виде собеседования и заключается в демонстрации работы программы, объяснении принципов работы алгоритма и ответов на дополнительные вопросы.

Критерии оценки: для получении оценки «зачтено» необходимо показать высокий уровень владения теоретическим материалом, уметь объяснить принцип работы написанной программы, верно ответить на дополнительные вопросы.

Задание: написать программу, реализующую упрощение выражений, группирование констант, использование целочисленной арифметики вместо арифметики с плавающей точкой, замену итеративных вычислений аналитическими выражениями. Проверить работу программы на контрольном примере.

Лабораторная работа № 9
«Оптимизация потока управления»

Цель работы: рассмотреть на практике способы оптимизации потока управления.

Требования к выполнению работы: выполнение лабораторной работы предусматривает написание программы, реализующей различные механизмы оптимизации потока управления в программе. Отчёт о работе проводится в виде собеседования и заключается в демонстрации работы программы, объяснении принципов работы алгоритма и ответов на дополнительные вопросы.

Критерии оценки: для получении оценки «зачтено» необходимо показать высокий уровень владения теоретическим материалом, уметь объяснить принцип работы написанной программы, верно ответить на дополнительные вопросы.

Задание: написать программу, реализующую применение switch вместо if-elseif-else и виртуальных функций вместо switch или if. Проверить работу программы на контрольном примере

Лабораторная работа № 10
«Оптимизация поиска»

Цель работы: рассмотреть на практике механизмы оптимизации поиска в программах на языках C/C++

Требования к выполнению работы: выполнение лабораторной работы предусматривает написание программы, реализующей различные варианты оптимизации поиска в программном коде. Отчёт о работе проводится в виде собеседования и заключается в демонстрации работы программы, объяснении принципов работы алгоритма и ответов на дополнительные вопросы.

Критерии оценки: для получении оценки «зачтено» необходимо показать высокий уровень владения теоретическим материалом, уметь объяснить принцип работы написанной программы, верно ответить на дополнительные вопросы.

Задание: написать программу, реализующую, оптимизация поиска в хешированных таблицах «ключ/значение», хеширование с фиксированными символьными массивами в качестве ключей, хеширование с ключами в виде строк с завершающими нулевым и символами, хеширование с пользовательской хеш – таблицей. Проверить работу программы на контрольном примере

Лабораторная работа № 11
«Оптимизация многопоточных программ C++.»

Цель работы: рассмотреть медоды оптимизации с учетом многопоточности в программах.

Требования к выполнению работы: выполнение лабораторной работы предусматривает написание программы, реализующей оптимизацию с учетом нескольких потоков исполнения. Отчёт о работе проводится в виде собеседования и заключается в демонстрации работы программы, объяснении принципов работы алгоритма и ответов на дополнительные вопросы.

Критерии оценки: для получении оценки «зачтено» необходимо показать высокий уровень владения теоретическим материалом, уметь объяснить принцип работы написанной программы, верно ответить на дополнительные вопросы.

Задание: написать программу, реализующую распределение вычислений в программе с использованием нескольких потокв openMP или процессов MPI. Проверить работу программы на контрольном примере

Лабораторная работа № 12

«Управление памятью»

Цель работы: оптимизировать программный код с помощью функций управления памятью.

Требования к выполнению работы: выполнение лабораторной работы предусматривает написание программы, реализующей различные варианты использования функций управления памятью. Оценить скорость выполнения отдельных блоков программы. Отчёт о работе проводится в виде собеседования и заключается в демонстрации работы программы, объяснении принципов работы алгоритма и ответов на дополнительные вопросы.

Критерии оценки: для получении оценки «зачтено» необходимо показать высокий уровень владения теоретическим материалом, уметь объяснить принцип работы написанной программы, верно ответить на дополнительные вопросы.

Задание: написать программу, реализующую функции для выделения и освобождения памяти и управления памятью из библиотеки C. Проверить работу программы на контрольном примере

20.2. Промежуточная аттестация

Промежуточная аттестация по дисциплине осуществляется с помощью следующих оценочных средств: собеседование по экзаменационным билетам, содержащим три теоретических вопроса.

Перечень вопросов к зачету

1. Основные стратегии оптимизации кода на C++. Эффективность оптимизации.
2. Влияние скорости и объема памяти на производительность программного кода.
3. Влияние вызовов операционной системы на скорость программ.
4. Влияние на скорость выполнения программ порядка выполнения инструкций.
5. Измерение производительности. Закон Амдала.
6. Оценка стоимости отдельных инструкций C++.
7. Оценка стоимости циклов. Оценка количества повторений вложенных циклов. Оценка циклов с переменным количеством повторений.
8. Распознавание неявных циклов. Распознавание ложных циклов.
9. Использование модифицирующих операций.
10. Уменьшение работы с памятью с помощью резервирования для устранения временных значений.
11. Устранение копирования строкового аргумента. Устранение разыменований с помощью итераторов. Устранение копирования возвращаемого значения.
12. Использование массивов символов вместо строк.
13. Устранение преобразования строк. Преобразование между кодировками.

14. Временная стоимость алгоритмов в наилучшем, среднем и наихудшем случаях. Амортизированная временная стоимость.

15. Оптимизации сортировки и поиска. Эффективные алгоритмы поиска. Временная стоимость алгоритмов поиска.

16. Эффективные алгоритмы сортировки. Временная стоимость алгоритмов сортировки.

17. Использование информации о входных данных.

18. Шаблоны оптимизации.Предвычисления. Отложенные вычисления. Пакетирование. Кеширование. Оптимизация ожидаемого пути. Хеширование. Двойная проверка.

19. Длительность хранения переменной. Владение переменными. Объекты – значения и объекты — сущности.

20. API динамических переменных C++.

21. Автоматизация владения интеллектуальными указателями. Автоматизация владения динамическими переменными.

22. Статическое создание экземпляров класса. Статическое создание переменных-членов класса. Использование статических структур данных.

23. ПСоздание больших буферов в стеке. Статическое создание связанных структур данных. Создание бинарных деревьев в массиве.

24. Использование «главного указателя» для владения динамическими переменными. Уменьшение количества перераспределений динамических переменных.

25. Предварительное выделение памяти для динамических переменных для предотвращения перераспределений. Создание динамических переменных вне циклов.

26. Устранение излишнего копирования. Устранение нежелательного копирования в определении класса. Устранение копирования при вызове из функции. Библиотеки без копирования. Нестандартная семантика копирования.

27. Разделяемое владение сущностями. Плоские структуры данных.

28. Удаление кода из циклов. Кеширование конечного значения цикла. Применение более эффективных инструкций циклов. Изменение направления цикла.

29. Устранение инвариантного кода из циклов. Удаление ненужных вызовов функций из циклов. Удаление скрытых вызовов функций из циклов. Удаление дорогих медленно меняющихся вызовов из циклов.

30. Перемещение циклов в функции для снижения накладных расходов при вызовах. Стоимость вызовов функций. Стоимость указателей на функции. Объявление коротких функций встраиваемыми.

31. Устранение неиспoльзуемого полиморфизма. Удаление неиспользуемых интерфейсов. Выбор реализации интерфейса во время компоновки. Выбор реализации интерфейса во время компиляции.

32. Оптимизация выражений. Упрощение выражений. Группирование констант.

33. Использование целочисленной арифметики вместо арифметики с плавающей точкой. Замена итеративных вычислений аналитическими выражениями.

34. Идиомы оптимизации потока управления. Применение switch вместо if-elseif-else. Применение виртуальных функций вместо switch или if.

35. Оптимизация использования стандартной библиотеки. Оптимизация существующих библиотек. Проектирование оптимизированных библиотек.

36. Плоские иерархии наследования. Упрощение цепочки вызовов. Упрощение проектирования слоев.

37. Таблицы "ключ/значение" с использованием std::map и std::string.

38. Инструментарий для повышения производительности поиска.

39. Идентификация оптимизируемой деятельности. Разделение оптимизируемой деятельности.

40. Изменение или замена алгоритмов и структур данных.

41. Использование процесса оптимизации для пользовательских абстракций.

42. Оптимизация поиска с использованием std::map.

43. Применение символьных массивов фиксированного размера в качестве ключей std::map. Использование строк в стиле C в качестве ключей std::map. Использование std::set, когда ключ является значением.

44. Таблица «ключ/значение» для поиска в последовательных контейнерах.

45. Бинарный поиск с использованием std::equal_ranqe(). Бинарный поиск с использованием std::lower_bound().

46. Оптимизация поиска в хешированных таблицах «ключ/значение». Хеширование с использованием std::unordered_map.

47. Хеширование с фиксированными символьными массивами в качестве ключей. Хеширование с ключами в виде строк с завершающими нулевым и символами. Хеширование с пользовательской хеш – таблицей.

48. Оптимизация сортировки с использованием стандартной библиотеки C++.

49. Последовательные контейнеры. Тип данных элемента.

50. Создание экономной сигнатуры функции. Сокращение цепочек вызовов.

51. Снижение количества перераспределений. Испольэование большего входного буфера.

52. Использование построчного чтения. Чтение из std::cin и запись в std::cout.

53. Чередующееся выполнение. Последовательная согласованность.

54. Синхронизация. Атомарность с помощью взаимоисключений. Атомарные аппаратные операции.

55 Возможности параллельности в C++. Потоки. Асинхронные задания.

56. Мьютексы. Блокировки. Условные переменные.

57. Атомарные операции над общими переменными. Барьеры памяти.

58. Оптимизация многопоточных программ C++.

59.Библиотеки для параллельных вычислений.

60. API управления памятью C++.

62. Жизненный цикл динамических переменных.

63. Функции для выделения и освобождения памяти. Функции управления памятью из библиотеки C.

64. Высокопроизводитеьные диспетчеры памяти. Диспетчеры памяти для конкретных классов.

65. Диспетчер памяти для блоков фиксированного размера. Добавление operator new() для конкретного класса.

66. Производительность диспетчера памяти для блоков фиксированного размера.

67. Пользовательские аллокаторы стандартной библиотеки.

68. Минимальный аллокатор в C++11. Дополнительные определения для аллокатора C++98.

69. Аллокатор блоков фиксированного размера. Аллокатор блоков фиксированного размера для строк.

Для оценивания результатов обучения на зачёте с оценкой используется 4-балльная шала: «отлично», «хорошо», «удовлетворительно», «неудовлетворительно».

Критерии оценивания компетенций	Уровень сформированности компетенций	Шкала оценок
Дан полный, развёрнутый ответ на поставленный вопрос (вопросы), обучающийся свободно оперирует основными понятиями дисциплины, ориентируется в предметной области. Изложение материала не содержит ошибок, отличается последовательностью, грамотностью, логической стройностью.	Повышенный уровень	Отлично
Дан развёрнутый ответ на поставленный вопрос (вопросы), обучающийся свободно оперирует основными понятиями дисциплины, ориентируется в предметной области. Материал изложен в целом последовательно и грамотно, отсутствуют грубые ошибки, однако имеются отдельные неточности в определениях, вычислениях, доказательствах, изложениях положений теории.	Базовый уровень	Хорошо

Ответ на поставленный вопрос (вопросы) содержит изложение только базового теоретического материала, имеются ошибки в определениях, вычислениях, доказательствах, формулировках положений теории. Нарушена логическая последовательность в изложении материала.	Пороговый уровень	Удовлетворительно
Ответ на поставленный вопрос (вопросы) отсутствует, либо содержит грубые ошибки в определениях, вычислениях, доказательствах, формулировках положений теории. Обучающийся не владеет основными понятиями дисциплины. Отсутствует логическая последовательность в изложении материала.	–	Неудовлетворительно