Структура курса

Конструирование алгоритмов и структур данных

ОСЕННИЙ СЕМЕСТР

Базовые алгоритмы и структуры данных

(25 лек., 34 лаб.)

В рамках курса предусмотрено *25 лекционных занятий*, где подробно разбираются основные концепции и подходы, а также *34 лабораторных занятия*, на которых вы сможете применить полученные знания на практике. Для закрепления материала – *5 лабораторных работ по ООП и основам языка C#, 8 контестов на алгоритмы* и *17 задач на структуры данных*. В середине семестра проводится *коллоквиум* для проверки теоретических и практических основ первой части курса, а в конце – *экзамен*.

Система оценки построена так, чтобы поощрять систематическую работу и глубокое понимание. В этом курсе нельзя «выехать» на чём-то одном. Успешный разработчик должен одинаково хорошо владеть и теорией, и практикой.

В течение курса оцениваются 5 аспектов вашей работы:

1. Владение принципами ООП и основами языка C# (лабораторные работы).
2. Умение решать алгоритмические задачи (контесты).
3. Умение проектировать и реализовывать структуры данных (задачи).
4. Понимание теоретических и практических основ первой половины курса (коллоквиум).
5. Понимание теоретических и практических основ курса (экзамен).

Каждый из этих блоков – обязателен. Если вы полностью игнорируете хотя бы один из них, получить положительную оценку за курс будет невозможно. Это не наказание, а отражение реальности: нельзя быть специалистом по алгоритмам, не зная теории или не умея писать код. Экзамен сдают все, потому что это финальная проверка готовности и целостности ваших знаний. Работа в семестре – это не способ «откосить» от экзамена, а способ подготовиться к нему и обеспечить себе отличную итоговую оценку.

Лекции

1. Введение. Асимптотический анализ (O-нотация). Сортировки.
2. Бинарная куча. Очередь с приоритетом.
3. Бинарный поиск. Порядковые статистики.
4. Амортизированный анализ. Стек, очередь, дек.
5. Система непересекающихся множеств (СНМ).
6. Динамическое программирование (редакционное расстояние, оптимальное расположение текста, наибольшая общая подпоследовательность (НОП) и + восстановление, наибольшая возрастающая подпоследовательность (НВП) и + восстановление ответа).
7. Динамическое программирование (рюкзак, по подмножествам).
8. Динамическое программирование (по профилю).
9. Дерево отрезков. Базовые операции.
10. Дерево Фенвика. Разреженная таблица (Sparse Table).
11. Сбалансированные деревья поиска: AVL, Splay.
12. Декартово дерево по неявному ключу.
13. Наименьший общий предок (LCA), бинарные подъёмы, Фарах Колтон и Бендер.
14. Heavy-Light декомпозиция (HLD), центроидная декомпозиция.
15. Графы: обход в глубину (DFS), поиск циклов, топологическая сортировка.
16. Компоненты связности. Задача 2-выполнимости (2-SAT).
17. Мосты. Точки сочленения. Поиск Эйлерова цикла.
18. Минимальные остовные деревья (MST): Крускал, Прим.
19. Кратчайшие пути: обход в ширину (BFS), Дейкстра, Форд-Беллман, Флойд, Джонсон.
20. A*, для поиска кратчайших путей в графах.
21. Хеш-таблицы, множества, словари.
22. Идеальное хеширование, хеширование кукушки.

Лабораторные работы по ООП и основам языка C#

1. Классы, инкапсуляция, наследование.
2. Полиморфизм, интерфейсы.

3. Файлы, I/O.

4. Интерфейсы, делегаты.

5. События, исключения.

Контесты на алгоритмы

1. Сортировки, куча, бинарный поиск.

2. Стеки, очереди, CHM.

3. Динамическое программирование.

4. Дерево отрезков.

5. Дерево поиска.

6. Запросы на деревьях.

7. Графы: DFS, MST.

8. Кратчайшие пути.

Задачи на структуры данных

1. Длина вектора.

2. Комплексные числа.

3. Сортировки массивов целых чисел.

4. Сортировки массивов различных типов.

5. Куча.

6. Очередь с приоритетами.

7. Заявки в приоритетной очереди.

8. Динамический массив.

9. Считывание из файла в динамический массив.

10. Вектор.

11. Считывание из файла в вектор.

12. Стек.

13. Обратная польская запись.

14. Двунаправленная очередь.

15. Считывание из файла в двунаправленную очередь.

16. Двунаправленный список.

17. Сравнение динамического массива и двунаправленного списка.

Оценка за лабораторные работы $Оценка_{Лабораторные\ работы}$

Лабораторные работы нацелены на проверку вашего владения ключевыми концепциями объектно-ориентированного программирования и основ языка C#.

Как сдаётся лабораторная работа

Реализация. Студент пишет код, демонстрирующий применение заданных концепций (например, полиморфизма, обработки исключений).

Код-ревью. Код заливается на git-репозиторий для проверки преподавателем.

Защита у преподавателя:

Оценка 3 – рабочая демонстрация. Код компилируется, корректно работает и демонстрирует требуемый принцип. Студент может объяснить базовую логику.

Оценка 4 – качественное применение. Код хорошо структурирован, принципы ООП применены осмысленно (например, инкапсуляция не нарушена, интерфейсы выделены логично).

Оценка 5 – глубокое понимание. Студент может объяснить теоретические основы темы, сравнить подходы (например, интерфейс vs. абстрактный класс) и обосновать свой выбор в реализации.

Пример защиты лабораторной работы

Преподаватель:

– Здравствуйте! Давайте посмотрим вашу работу по полиморфизму.

Вопрос 1 (структура). Расскажите про иерархию ваших классов. Какой интерфейс вы использовали и зачем он нужен?

Вопрос 2 (ключевая концепция). Покажите в коде, где именно проявляется полиморфизм. Как работает вызов метода Draw() для объекта типа IShape, если реальный объект – Circle?

Вопрос 3 (теория). В чём принципиальная разница между интерфейсом и абстрактным классом? Когда бы вы использовали одно, а когда другое?

Вопрос 4 (демонстрация). Создайте массив из нескольких разных фигур (например, Circle, Rectangle) и в цикле вызовите для каждой метод GetArea(), выводя результат на экран.

Если студент уверенно отвечает, демонстрируя как практические навыки, так и теоретическое понимание, работа засчитывается.

Если защита не сдана:

– Ваша реализация наследования нарушает принцип подстановки Лисков. Пожалуйста, исправьте и приходите снова.

Важно понимать, что это лишь один из возможных сценариев защиты. ПРЕПОДАВАТЕЛЬ ВПРАВЕ ЗАДАВАТЬ ЛЮБЫЕ ВОПРОСЫ ПО ТЕМЕ, ОТТАЛКИВАЯСЬ ОТ ВАШЕГО КОДА И ОТВЕТОВ, чтобы убедиться в глубине ваших знаний, а не в заучивании конкретных формулировок.

Как формируется $Оценка_{Лабораторные\ работы}$

За каждую из 5 лабораторных работ выставляется отдельная оценка. Итоговая оценка за блок считается как среднее арифметическое этих оценок, с округлением по обычным правилам математики:

$$Оценка_{Лабораторные\ работы} = \frac{\sum_{i=1}^{5} Оценка_{Лабораторная\ работа_i}}{5},$$

где $Оценка_{Лабораторная\ работа_i}$ – оценка за i-ю лабораторную работу.

Оценка за контесты $Оценка_{Контесты}$

Контесты проверяют, умеете ли вы применять алгоритмы на практике. Для получения оценки за контест необходимо решить определённое количество задач и успешно защитить их. Итоговая оценка за каждый контест выставляется по результатам устной защиты.

Как сдаётся контест

1. Решение в системе. Студент решает задачи в системе Codeforces. Решение должно пройти все тесты – это подтверждает корректность и эффективность алгоритма.

2. Защита у преподавателя:

Оценка 3 – понимание кода. Студент без запинок объясняет любую строчку своего кода, рассказывает, почему выбрал ту или иную структуру данных, объясняет логику решения.

Оценка 4 – теоретическая база. Студент отвечает на 1-2 вопроса по теме контеста. Например, для темы «Сортировки, куча, бинпоиск» вопросы могут быть: «Какова сложность быстрой сортировки в худшем случае и почему?», «В чём принципиальное отличие двоичной кучи от двоичного дерева поиска?».

Оценка 5 – задачи «на подумать». Студент решает 1-2 задачи «на подумать». Например: «На базе кучи постройте структуру данных, которая может находить и удалять медиану ($n/2$-й элемент в отсортированном порядке)», «Предложите алгоритм сортировки циклических сдвигов за $O(n \cdot \log n)$. Указание: зная порядок на отрезках длины L, порядок на отрезках длины $2 \cdot L$ можно восстановить за $O(n)$».

Важно понимать, что это лишь один из возможных сценариев защиты. ПРЕПОДАВАТЕЛЬ ВПРАВЕ ЗАДАВАТЬ ЛЮБЫЕ ВОПРОСЫ ПО ТЕМЕ, ОТТАЛКИВАЯСЬ ОТ ВАШЕГО КОДА И ОТВЕТОВ, чтобы убедиться в глубине ваших знаний, а не в заучивании конкретных формулировок.

Пример защиты контеста

Преподаватель:

– Здравствуйте! Вижу, вы решили 6 из 8 задач в контесте по динамическому программированию. Отлично. Давайте обсудим задачу «Наибольшая общая подпоследовательность».

Вопрос 1 (суть алгоритма). Расскажите, как вы решали эту задачу. Какая идея лежит в основе решения? Что хранится в вашей DP-таблице dp[i][j]?

Вопрос 2 (сложность). Оцените асимптотику вашего решения по времени и памяти. Почему именно такая?

Вопрос 3 (код). Откройте ваш код. Объясните, что происходит в строках с 25 по 32. Зачем тут двойной цикл?

Вопрос 4 (связь с теорией). На какой лекции разбиралась эта тема? Какие ещё классические задачи на ДП обсуждались на лекции 6?

Вопрос 5 («на подумать»). Ваш алгоритм ищет длину НОП. Как его изменить, чтобы восстановить саму подпоследовательность? А если бы нужно было найти не наибольшую, а наименьшую общую подпоследовательность – имеет ли такая задача смысл?

Если студент уверенно отвечает на все вопросы – задача засчитана. Преподаватель может проверить 2-3 задачи из контеста. Если студент «плавает», не может объяснить код или идею – контест не засчитывается.

Если контест не сдан:

– Вам нужно лучше разобраться в теме. Повторите лекцию, проанализируйте свой код. Можете прийти на пересдачу на следующей неделе. Без успешной защиты решённые задачи не учитываются.

Важно понимать, что это лишь один из возможных сценариев защиты. ПРЕПОДАВАТЕЛЬ ВПРАВЕ ЗАДАВАТЬ ЛЮБЫЕ ВОПРОСЫ ПО ТЕМЕ, ОТТАЛКИВАЯСЬ ОТ ВАШЕГО КОДА И ОТВЕТОВ, чтобы убедиться в глубине ваших знаний, а не в заучивании конкретных формулировок.

Как формируется $Оценка_{Контесты}$

За каждый из 8 контестов выставляется отдельная оценка по итогам его защиты. Итоговая оценка за блок контестов $Оценка_{Контесты}$ считается как среднее арифметическое этих оценок, с округлением по обычным правилам математики:

$$Оценка_{Контесты} = \frac{\sum_{i=1}^{8} Оценка_{Контест_i}}{8},$$

где $Оценка_{Контест_i}$ – оценка за i-й контест.

Оценка за задачи $Оценка_{Задачи}$

Здесь основной упор делается на качество кода и понимание принципов ООП.

Как сдаётся задача

1. Реализация. Студент пишет код, реализующий структуру данных по заданию.

2. Код-ревью. Код заливается на git-репозиторий. Преподаватель может посмотреть и оставить комментарии.

3. Защита у преподавателя:

Оценка 3 – рабочая реализация. Студент показывает код, который компилируется без ошибок и реализует все методы из задания. Преподаватель проверяет работу на примерах.

Оценка 4 – качество кода. Код структурирован: классы, методы и поля логично организованы. Есть комментарии к сложным местам. Соблюдается единый стиль именования (camelCase для методов, PascalCase для классов).

Оценка 5 – объяснение реализации. Студент может объяснить ключевые методы структуры данных и их временные/пространственные сложности.

Важно понимать, что это лишь один из возможных сценариев защиты. ПРЕПОДАВАТЕЛЬ ВПРАВЕ ЗАДАВАТЬ ЛЮБЫЕ ВОПРОСЫ ПО ТЕМЕ, ОТТАЛКИВАЯСЬ ОТ ВАШЕГО КОДА И ОТВЕТОВ, чтобы убедиться в глубине ваших знаний, а не в заучивании конкретных формулировок.

Пример защиты задачи

Преподаватель:

– Добрый день! Давайте посмотрим вашу реализацию двусвязного списка. Откройте, пожалуйста, код.

Вопрос 1 (структура). Расскажите, как устроен ваш класс. Какие приватные поля вы используете и зачем они нужны? Как устроен узел списка?

Вопрос 2 (ключевые методы). Давайте разберём метод add. Прокомментируйте его по шагам. Какие граничные случаи вы предусмотрели (добавление в начало, в конец, в пустой список)?

Вопрос 3 (сложность). Какая сложность у addLast? А у addFirst? Почему так?

Вопрос 4 (сравнение). Вы делали похожую задачу на динамический массив MyArrayList. В чём плюсы и минусы списка по сравнению с массивом? Приведите пример, где список будет лучше, а где – хуже.

Вопрос 5 (демонстрация). Напишите прямо сейчас небольшой main: создайте список MyLinkedList, добавьте 5 элементов, удалите третий, выведите содержимое на экран.

Если студент уверенно отвечает на вопросы, понимает инварианты структуры и умеет сравнивать с другими решениями – задача засчитана.

Если защита не сдана:

– Ваш remove неправильно работает при удалении последнего элемента. Исправьте и приходите ещё раз. Без этого задачу не зачту.

Как формируется $Оценка_{Задачи}$

За каждую практическую задачу выставляется отдельная оценка, отражающая качество её реализации и защиты. Итоговая оценка за блок задач $Оценка_{Задачи}$ считается как среднее арифметическое этих оценок, с округлением по обычным правилам математики:

$$Оценка_{Задачи} = \frac{\sum_{i=1}^{17} Оценка_{Задача_i}}{17},$$

где $Оценка_{Задача_i}$ – оценка за i-ю задачу.

Оценка за коллоквиум $Оценка_{Коллоквиум}$

Коллоквиум проводится для того, чтобы проверить, как вы усвоили материал первой половины курса. Это помогает выявить, кто отстаёт, а также мотивирует всех повторить и систематизировать пройденное. Кроме того, это тренировка перед экзаменом.

Как проходит коллоквиум

В билете 2 теоретических вопроса и 1 практическая задача «на листочке». Темы охватывают все лекции и лаб. занятий первой половины курса.

Как формируется $Оценка_{Коллоквиум}$

За каждый вопрос и задачу ставится отдельная оценка. Итоговая оценка за коллоквиум $Оценка_{Коллоквиум}$ считается как среднее арифметическое этих оценок, с округлением по обычным правилам математики:

$$Оценка_{Коллоквиум} = \frac{Оценка_{Вопрос_1} + Оценка_{Вопрос_2} + Оценка_{Задача}}{3},$$

где $Оценка_{Вопрос_1}$ – оценка за первый теоретический вопрос, $Оценка_{Вопрос_2}$ – оценка за второй теоретический вопрос, $Оценка_{Задача}$ – оценка за задачу.

Оценка за экзамен $Оценка_{Экзамен}$

Экзамен – это итоговая проверка знаний, которая охватывает весь материал, пройденный за семестр. Его цель – оценить, насколько глубоко и системно вы усвоили курс, и проверить ваше умение связывать разные темы в единое целое.

Как проходит экзамен

Формат похож на коллоквиум, но вопросы уже по всему курсу.

В билете 2 теоретических вопроса (один – по первой половине курса, второй – по второй) и 1 практическая задача.

Преподаватель может дополнительно задавать вопросы по всему курсу.

Как формируется $Оценка_{Экзамен}$

За каждый вопрос и задачу ставится отдельная оценка. Итоговая оценка за экзамен $Оценка_{Экзамен}$ считается как среднее арифметическое этих оценок, с округлением по обычным правилам математики:

$$Оценка_{Экзамен} = \frac{Оценка_{Вопрос_1} + Оценка_{Вопрос_2} + Оценка_{Задача}}{3},$$

где $Оценка_{Вопрос_1}$ – оценка за первый теоретический вопрос, $Оценка_{Вопрос_2}$ – оценка за второй теоретический вопрос, $Оценка_{Задача}$ – оценка за задачу.

Итоговая оценка *Итог*

Компоненты оценки:

1. $Оценка_{Лабораторные\ работы}$ – за все 5 лабораторных работ.
2. $Оценка_{Контесты}$ – за все 8 контестов на алгоритмы.
3. $Оценка_{Задачи}$ – за все 17 задач на структуры данных.
4. $Оценка_{Коллоквиум}$ – за коллоквиум.
5. $Оценка_{Экзамен}$ – за экзамен.

Как формируется $Итог$

1. Стандартный случай (все блоки сданы).

Если у студента есть ненулевые оценки по всем пяти компонентам, итог считается как среднее арифметическое:

$$Итог = (Оценка_{Лабораторные\ работы} + Оценка_{Контесты} + Оценка_{Задачи} + \\ + Оценка_{Коллоквиум} + Оценка_{Экзамен}) / 5.$$

2. Штрафной случай (один из блоков провален).

Если хотя бы по одному из пяти ключевых компонентов (лабораторные работы, контесты, задачи, коллоквиум или экзамен) стоит 0, это значит, что минимальные требования по этому направлению не выполнены.

Что такое 0?

0 ставится, если сдано меньше 60% работ в блоке (то есть меньше 3 лабораторных работ, меньше 5 контестов или меньше 11 задач). Это не оценка 2, а именно индикатор того, что блок не выполнен.

В таком случае итог считается так – добавляется ещё один фиктивный 0:

$$Итог = (Оценка_{Лабораторные\ работы} + Оценка_{Контесты} + Оценка_{Задачи} + \\ + Оценка_{Коллоквиум} + Оценка_{Экзамен} + 0) / 6.$$

Оценка 2 – $Итог < 2,5$

Оценка 3 – $2,5 \le Итог < 3,5$

Оценка 4 – $3,5 \le Итог < 4,5$

Оценка 5 – $Итог \ge 4,5$

Пример

Студент отлично сдал лабораторные работы, задачи, коллоквиум, экзамен, но полностью проигнорировал контесты.

Считаем:

$$Итог = \frac{5+0+5+5+5+0}{6} = \frac{20}{6} = 3,(3) \approx 3.$$

Максимум, на что он может рассчитывать – это оценка 3, и то, если всё остальное сдано на 5.

ЛЮБОЕ СПИСЫВАНИЕ И ПЛАГИАТ, включая копирование кода из интернета, у одногруппников или с помощью генераторов вроде ChatGPT, DeepSeek и других ИИ-инструментов, СТРОГО ЗАПРЕЩЕНЫ. Если такое обнаружится, работа АННУЛИРУЕТСЯ, а студенту могут быть назначены ДИСЦИПЛИНАРНЫЕ МЕРЫ. Обсуждать идеи между собой РАЗРЕШАЕТСЯ, но финальные ответы и код должны быть ВАШИМИ СОБСТВЕННЫМИ.

Учёт посещаемости и конспектов

Посещение занятий и ведение конспектов – важная часть учебного процесса. Хотя основное внимание уделяется вашим результатам, систематическое пренебрежение занятиями не останется без последствий.

Конспекты

Допускаются два формата ведения конспекта: традиционный рукописный в тетради или цифровой, созданный от руки с помощью стилуса на планшете.

Наличие конспекта является обязательным условием допуска к коллоквиуму и экзамену.

Штрафы за пропуски

Посещаемость лекций и лабораторных занятий влияет на итоговую оценку.

За каждый пропуск лекции или лабораторного занятия без уважительной причины (без официального документа) из итогового расчётного показателя $Итог$ вычитается 0,05.

Суммарный штраф за все пропуски в течение семестра не может превышать 1.

Пример

Студент получил расчётный показатель $Итог = 4,6$, но пропустил 4 лекции и 2 лабораторных занятия без уважительной причины (всего 6 пропусков). Его итоговый показатель будет скорректирован:

$$Итог = 4,6 - 6 \cdot 0,05 = 4,6 - 0,3 = 4,3.$$

Итог – оценка 4.

Система бонусов за академические достижения

Мы поощряем студентов, которые применяют знания на практике и выходят за рамки обязательной программы курса. Бонусы позволяют улучшить ваши оценки за работу в семестре.

Выход в полуфинал ICPC

Студент, входящий в состав команды, прошедшей в полуфинал ICPC, автоматически получает оценку 5 за курс. Это достижение полностью покрывает цели курса.

Повышение оценок за достижения

Научные публикации, доклады и победы или призовые места на олимпиадах позволяют повысить оценку за одну из сданных работ (лабораторную работу, контест или задачу на структуры данных).

Научная публикация по теме курса позволяет повысить оценку за одну работу на 2 пункта (например, с 3 до 5).

Выступление с докладом на научной конференции позволяет повысить оценку за одну работу на 1 пункт (например, с 4 до 5).

Победа или призовое место на соревновании по программированию (регионального уровня) позволяют получить оценку 5 за один любой контест.