Общество с ограниченной ответственностью «Онлайн-Гимназия Адель»
(ООО «Онлайн-Гимназия Адель»)
ИНН 5022076651 ОГРН 1235000132344
140410, Московская область, г Коломна, ул. Зеленая, д. 31А

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ПРОХОЖДЕНИЮ ТЕКУЩЕГО КОНТРОЛЯ ПО ИНФОРМАТИКЕ ПРИ РЕАЛИЗАЦИИ ПРОГРАММ ДОПОЛНИТЕЛЬНОГО ОБЩЕОБРАЗОВАТЕЛЬНОГО ОБЩЕРАЗВИВАЮЩЕГО ОБРАЗОВАНИЯ

"Домашняя Гимназия. Подготовка к аттестации 10-11 класс"

для дистанционного семейного обучения

Коломна

2025

ВВЕДЕНИЕ

Настоящие методические рекомендации разработаны с целью систематизации работы по формированию и развитию ключевых компетенций в области информатики у обучающихся 10–11 модулей. На данном уровне обучения особое значение приобретают углублённое изучение основ программирования, работа с базами данных, сетевые технологии, а также развитие навыков проектной и исследовательской деятельности в сфере информационных технологий.

Цель документа

Основная цель данного документа — создание единого методического пространства, обеспечивающего эффективное освоение обучающимися учебных дисциплин в области информатики на профильном уровне. Рекомендации ориентированы на подготовку выпускников к ЕГЭ по информатике, олимпиадам и продолжению образования в IT-направлениях.

Рекомендации направлены на достижение следующих задач:

1. Унификация подходов к организации углублённой учебной деятельности по информатике
2. Стандартизация критериев оценивания проектных, исследовательских и программистских работ
3. Повышение качества образовательного процесса через чёткое определение целей и планируемых результатов профильного уровня
4. Создание условий для дифференцированного и индивидуального подхода в обучении
5. Обеспечение преемственности между основной школой и профильным обучением

Структура документа

Методические рекомендации состоят из трёх основных разделов:

1. Тестирование — контроль знаний по теоретическим основам информатики, включая вопросы уровня ЕГЭ
2. Проектная и исследовательская деятельность — создание информационных систем, веб-приложений, баз данных и IT-продуктов
3. Решение задач по программированию и анализу данных — написание и оптимизация программ, работа с большими наборами данных

Каждый раздел содержит:

1. Рекомендации для преподавателя с указанием целей и планируемых результатов
2. Рекомендации для обучающихся с конкретными алгоритмами действий
3. Примеры заданий с образцами выполнения
4. Критерии оценивания с чёткими параметрами и шкалами

Методологические основы

Рекомендации построены на следующих принципах:

1. Принцип профильности — материалы соответствуют углублённому уровню освоения информатики в 10–11 модулях
2. Принцип системности — виды работ выстроены в логике от воспроизведения знаний до создания оригинальных IT-продуктов
3. Принцип практической направленности — акцент на реальные задачи разработки ПО, анализа данных и проектирования систем
4. Принцип вариативности — предусмотрены задания разного уровня сложности, включая олимпиадные
5. Принцип объективности — критерии оценивания прозрачны и соотносятся со стандартами ЕГЭ

Целевая аудитория

Данные рекомендации предназначены для:

1. Преподавателей информатики, работающих с обучающимися 10–11 класса на базовом и профильном уровнях
2. Обучающихся, планирующих сдавать ЕГЭ по информатике и поступать на IT-специальности

Ожидаемые результаты реализации

Внедрение данных рекомендаций в образовательную практику позволит:

Для преподавателей:

- Структурировать и унифицировать подходы к профильному обучению информатике
- Обеспечить объективность оценивания проектных и программистских работ
- Сформировать банк заданий для подготовки к ЕГЭ
- Создать условия для выявления и развития технически одарённых обучающихся

Для обучающихся:

- Получить чёткие ориентиры для самостоятельной подготовки к ЕГЭ
- Освоить профессиональные стандарты разработки и документирования программного обеспечения
- Сформировать портфолио учебных IT-проектов
- Развить навыки командной работы над технологическими проектами

Особенности применения

Рекомендации носят рекомендательный характер и могут быть адаптированы с учётом:

- Профиля обучения (информационно-технологический, естественно-научный, универсальный)
- Используемого языка программирования (Python, C++, Java и др.)
- Технических возможностей и имеющегося программного обеспечения
- Ориентации на конкурсы, олимпиады или профессиональные пробы

Составитель: Педагогическое объединение преподавателей информатики
Дата составления: 2025 год
Уровень: 10–11 модуль

1.ТЕСТИРОВАНИЕ

1.1. Рекомендации для преподавателя

Цели проведения тестирования:

1. Проверить глубину усвоения теоретических основ информатики на уровне, соответствующем требованиям ЕГЭ
2. Оценить умение применять знания о сетевых технологиях, базах данных, логических основах ЭВМ и основах безопасности
3. Выявить пробелы в понимании сложных тем: булева алгебра, сложность алгоритмов, реляционные базы данных
4. Развивать навыки самопроверки и подготовки к итоговой аттестации

Планируемые результаты:

1. Обучающийся уверенно оперирует понятиями: булева алгебра, граф, сложность алгоритма, реляционная модель БД, протоколы TCP/IP
2. Умеет переводить числа между системами счисления (двоичная, восьмеричная, шестнадцатеричная) и выполнять арифметику в них
3. Применяет знания об устройстве компьютерных сетей и принципах работы интернета
4. Демонстрирует понимание основ кибербезопасности: шифрование, вирусы, защита данных

Методические рекомендации:

1. Разрабатывайте тесты, ориентируясь на структуру ЕГЭ по информатике: задания с кратким ответом и развёрнутые задания
2. Включайте разнообразные типы заданий:

- Задания с выбором одного или нескольких правильных ответов
- Задания на установление соответствия (алгоритм — результат выполнения)
- Задания с кратким числовым ответом (перевод чисел, вычисление количества бит)
- Задания на анализ фрагмента кода или блок-схемы

3. Включайте в тест задания, имитирующие реальные ситуации: работа с интернет-ресурсами, безопасность, хранение данных
4. Предусматривайте время выполнения: 20–30 минут для текущего контроля, 45–60 минут для рубежного
5. После тестирования разбирайте типичные ошибки в классе

Пример распределения заданий по уровням:

- 50% — базовый уровень (знание определений и понятий)
- 35% — повышенный уровень (применение знаний, анализ ситуаций)
- 15% — высокий уровень (задания олимпиадного характера, синтез знаний)

1.2. Рекомендации для обучающихся

Как подготовиться к тесту:

1. Повторите теоретический материал по всем разделам: системы счисления, логика, алгоритмы, сети, БД, безопасность
2. Решите тренировочные задания из открытого банка ЕГЭ по информатике
3. Составьте шпаргалку (для самоподготовки): формулы перевода систем счисления, таблицы истинности, ключевые определения
4. Потренируйтесь в анализе простых программ и блок-схем — определяйте результат выполнения

Как выполнять тест:

1. Внимательно прочитайте инструкцию к тесту — обратите внимание, сколько ответов нужно выбрать
2. Просмотрите все задания перед началом работы, оцените их сложность
3. Начинайте с заданий, в которых уверены
4. Для числовых заданий записывайте промежуточные вычисления — это поможет найти ошибку при проверке
5. Для задания с анализом кода выполняйте его вручную шаг за шагом, фиксируя значения переменных
6. Обязательно проверьте работу перед сдачей

Полезные советы:

- При переводе чисел используйте метод последовательного деления или таблицу степеней двойки
- При анализе логических выражений составляйте таблицу истинности
- Для задач на граф — рисуйте граф вручную, это снижает вероятность ошибки
- Следите за единицами измерения: бит, байт, Кбайт, Мбайт — частый источник ошибок

1.3. Пример задания и образец выполнения

Тема: «Информация. Системы счисления. Логические основы»

Тестовые задания:

1. Переведите число 10110101_2 в шестнадцатеричную систему счисления.

- а) $A5_{16}$
- б) $B5_{16}$
- в) $C3_{16}$
- г) $D4_{16}$

2. Логическое выражение ¬A ∨ (A ∧ B) равно...

- а) A

- б) B
- в) ¬A ∨ B
- г) A ∧ B

3. Для хранения растрового изображения размером 1024 × 768 пикселей с палитрой 256 цветов требуется:

- а) 786 432 бит
- б) 786 432 байт
- в) 6 291 456 байт
- г) 3 145 728 байт

4. Какой протокол обеспечивает надёжную передачу данных с установлением соединения?

- а) UDP
- б) IP
- в) TCP
- г) HTTP

Образец выполнения:

1. а) $A5_{16}$ (10110101_2 → группируем по 4: 1011 0101 → B 5 → $B5_{16}$, ответ б)
2. в) ¬A ∨ B (упрощение по законам булевой алгебры: ¬A ∨ (A ∧ B) = (¬A ∨ A) ∧ (¬A ∨ B) = 1 ∧ (¬A ∨ B) = ¬A ∨ B)
3. б) 786 432 байт (256 цветов = 8 бит = 1 байт на пиксель; 1024 × 768 × 1 = 786 432 байт)
4. в) TCP

1.4. Критерии оценивания

Оценка	Количество правильных ответов	Характеристика выполнения
5	90–100%	Все задания выполнены правильно или допущена 1 негрубая ошибка в вычислениях
4	75–89%	Допущено 2–3 ошибки, базовые понятия и методы решения усвоены
3	60–74%	Выполнено более половины заданий, имеются пробелы в ключевых темах
2	менее 60%	Большая часть заданий выполнена неправильно, требуется повторение материала

Примечания:

1. Для заданий с числовым ответом засчитывается только точный результат
2. Частично правильный ответ в заданиях с выбором нескольких вариантов оценивается частично (по усмотрению преподавателя)
3. Рекомендуется проводить разбор ошибок сразу после тестирования

2.ПРОЕКТНАЯ И ИССЛЕДОВАТЕЛЬСКАЯ ДЕЯТЕЛЬНОСТЬ

2.1. Рекомендации для преподавателя

Проектная и исследовательская деятельность — это форма учебной работы, в которой обучающийся самостоятельно или в команде решает реальную задачу в области информационных технологий: разрабатывает веб-сайт, программный продукт, базу данных, информационную систему или проводит исследование в области IT. Данный вид деятельности является приоритетным для профильного уровня обучения информатике.

Цели проектной и исследовательской деятельности:

- Формирование навыков проектирования и реализации IT-продуктов
- Развитие умений работать с технической документацией и составлять её самостоятельно
- Освоение полного цикла разработки: анализ → проектирование → реализация → тестирование → презентация
- Формирование навыков командной работы и распределения задач
- Приобщение к профессиональным стандартам IT-отрасли

Планируемые результаты:

- Умеет формулировать техническое задание и ставить цели проекта
- Разрабатывает и реализует проект с применением изученных технологий
- Грамотно оформляет документацию: пояснительная записка, UML-диаграммы, руководство пользователя
- Проводит тестирование и анализирует результаты
- Представляет и защищает проект перед аудиторией

Методические рекомендации:

1. Обеспечьте свободу выбора темы в рамках разделов курса: веб-разработка, базы данных, анализ данных, ИИ, кибербезопасность
2. Организуйте работу поэтапно с промежуточными точками контроля:

- Этап 1 — постановка задачи и технического задания
- Этап 2 — проектирование (архитектура, схемы, макеты)
- Этап 3 — реализация
- Этап 4 — тестирование
- Этап 5 — оформление документации и презентация

3. Применяйте методы наставничества: проводите регулярные консультации, давайте развивающую обратную связь
4. Поощряйте использование систем контроля версий (Git) для отслеживания прогресса
5. Организуйте публичную защиту проектов — это формирует навыки презентации и аргументации

2.2. Рекомендации для обучающихся

Выбор темы проекта

При выборе темы проекта руководствуйтесь следующими критериями:

- Актуальность — тема решает реальную задачу или проблему
- Реализуемость — вы обладаете достаточными знаниями и ресурсами для выполнения
- Измеримость результата — итог работы можно продемонстрировать и оценить
- Личный интерес — вам интересна данная область IT

Структура проекта

Проект должен включать следующие обязательные части:

1. Постановка задачи и цели проекта
2. Анализ предметной области и существующих решений
3. Техническое задание (ТЗ)
4. Проектирование: схема БД / архитектура приложения / макеты интерфейса
5. Реализация с описанием ключевых технических решений
6. Тестирование: сценарии тестирования и результаты
7. Выводы и перспективы развития
8. Список использованных источников и технологий

Оформление пояснительной записки

Пояснительная записка оформляется в соответствии со следующими требованиями:

- Шрифт: Times New Roman или Arial, 12–14 pt
- Поля: верхнее и нижнее — 2 см, левое — 3 см, правое — 1,5 см
- Межстрочный интервал: 1,5
- Нумерация страниц: снизу по центру
- Код программы оформляется моноширинным шрифтом (Courier New, 10–12 pt)
- Рисунки и схемы имеют подписи

Подготовка к защите

Для успешной защиты проекта необходимо:

1. Подготовить презентацию (8–12 слайдов): цель, постановка задачи, реализация, демонстрация, выводы
2. Подготовить демонстрацию работающего продукта (на компьютере или в виде видеозаписи)
3. Сформулировать чёткие ответы на возможные вопросы: почему выбрали данную технологию, с какими трудностями столкнулись, что можно улучшить
4. Уложиться в регламент: 5–7 минут на презентацию, 3–5 минут на вопросы

2.3. Пример задания и образец выполнения

Тема проекта: «Разработка базы данных для учёта книжного фонда библиотеки»

Постановка задачи: Разработать реляционную базу данных для автоматизации учёта книг в школьной библиотеке с возможностью поиска, добавления и выдачи книг читателям.

Схема базы данных (основные таблицы):

Таблица «Книги»	Таблица «Читатели»	Таблица «Выдачи»
id_книги (PK)	id_читателя (PK)	id_выдачи (PK)
название	фамилия	id_книги (FK)
автор	имя	id_читателя (FK)
год_издания	класс	дата_выдачи
жанр	телефон	дата_возврата

Пример SQL-запроса (поиск просроченных выдач):

```
-- Список книг, не возвращённых вовремя
SELECT к.название, к.автор, ч.фамилия, ч.класс,
       в.дата_выдачи, в.дата_возврата
FROM Выдачи в
JOIN Книги к ON в.id_книги = к.id_книги
JOIN Читатели ч ON в.id_читателя = ч.id_читателя
WHERE в.дата_возврата < CURRENT_DATE
  AND в.факт_возврата IS NULL
ORDER BY в.дата_возврата;
```

Состав проекта:

- Пояснительная записка (15–20 страниц)
- Файл базы данных (.sql или .db)
- Презентация (10 слайдов)
- Демонстрация работающего интерфейса

2.4. Критерии оценивания

Оценка	Критерии	Характеристика выполнения
5	Отлично	Проект полностью реализован, документация

Оценка	Критерии	Характеристика выполнения
		оформлена корректно, защита уверенная с аргументированными ответами на вопросы, продукт работает без ошибок
4	Хорошо	Проект реализован с незначительными недочётами в функционале или документации, на большинство вопросов даны верные ответы
3	Удовл.	Проект частично реализован (не менее 50% функционала), документация содержит ошибки, при защите возникли затруднения с ответами на вопросы
2	Неудовл.	Проект не реализован или не демонстрирует заявленный функционал, документация отсутствует или выполнена формально

3.РЕШЕНИЕ ЗАДАЧ ПО ПРОГРАММИРОВАНИЮ И АНАЛИЗУ ДАННЫХ

3.1. Рекомендации для преподавателя

На уровне 10–11 модулей решение задач по программированию выходит на качественно новый уровень: от написания простых линейных программ — к разработке алгоритмов с использованием рекурсии, структур данных (списки, стеки, очереди, деревья), работы с файлами и базами данных, а также к начальным навыкам анализа данных с применением библиотек.

Цели решения задач по программированию:

- Формирование навыков написания эффективных алгоритмов с оценкой их сложности
- Освоение парадигм программирования: процедурное, объектно-ориентированное, функциональное
- Развитие навыков работы с файлами, потоками данных и внешними библиотеками
- Подготовка к решению задач ЕГЭ по информатике (задания 23–27)
- Введение в анализ данных: обработка CSV, статистика, визуализация

Планируемые результаты:

- Умеет анализировать условие задачи, выделять структуры данных и алгоритмические конструкции
- Реализует решения с использованием функций, классов, модулей
- Оценивает эффективность алгоритма по времени (O-нотация)
- Работает с файлами, базами данных, внешними API
- Применяет библиотеки для анализа данных: pandas, matplotlib (для Python)

Методические рекомендации:

Структурированный подход к обучению решению задач на профильном уровне включает:

- Изучение классических алгоритмов: сортировка, поиск, работа с графами
- Разбор задач с разбором решения «от задачи к коду»
- Использование систем автоматической проверки (Codeforces, Яндекс.Контест, Stepik)
- Анализ типичных ошибок и антипаттернов

Ключевые этапы работы с задачей:

1. Анализ условия — тип задачи, входные/выходные данные, ограничения
2. Выбор алгоритма и структур данных
3. Оценка сложности предполагаемого решения
4. Кодирование с соблюдением стиля и комментированием
5. Тестирование: граничные случаи, нагрузочное тестирование

3.2. Рекомендации для обучающихся

На профильном уровне важно не просто написать работающую программу, но и обеспечить её эффективность, читаемость и надёжность. Профессиональный подход к решению задач — это навык, который формируется через систематическую практику.

1. Анализ условия

Прежде чем писать код, чётко определите:

- Что является входными данными? Каков их формат и диапазон значений?
- Что является результатом? Какой формат вывода ожидается?
- Каковы ограничения по времени и памяти?
- Есть ли граничные случаи (пустой ввод, одно значение, максимальный диапазон)?

2. Выбор алгоритма

Исходя из ограничений задачи выберите подходящий алгоритм:

- $N \le 10^6$ — допустим $O(N \log N)$
- $N \le 10^4$ — допустим $O(N^2)$
- $N \le 10^2$ — допустим $O(N^3)$ или $O(2^N)$
- Если задача про граф — рассмотрите BFS, DFS, Дейкстру
- Если задача на оптимизацию — рассмотрите динамическое программирование

3. Написание кода

При реализации алгоритма соблюдайте следующие правила:

- Используйте осмысленные имена переменных и функций
- Разбивайте сложную логику на функции
- Добавляйте комментарии к нетривиальным участкам кода
- Соблюдайте стиль кодирования (PEP 8 для Python)
- Не дублируйте код — выносите повторяющиеся фрагменты в функции

4. Тестирование

Систематически проверяйте программу:

- Типичный ввод — ожидаемый результат должен совпасть
- Граничные значения — пустой массив, один элемент, максимальное значение
- Специальные случаи — отрицательные числа, нулевые значения, строки с пробелами
- Стресс-тест — генерируйте большие входные данные и проверяйте скорость работы

5. Анализ данных (для задач с большими наборами данных)

При работе с табличными данными используйте следующий подход:

- Загрузка данных: pandas.read_csv() или встроенный модуль csv

- Предобработка: проверка на пропущенные значения, удаление дубликатов
- Анализ: describe(), groupby(), merge()
- Визуализация: matplotlib.pyplot или seaborn
- Выводы: формулируйте заключения на основе полученных данных

3.3. Пример задания и образец выполнения

Условие задачи:

Дан список целых чисел. Напишите программу на Python, которая находит все пары чисел в списке, сумма которых равна заданному числу K. Вывести пары без повторений. Оптимизируйте решение до O(N).

Шаг 1. Анализ условия:

- Дано: список целых чисел nums, целое число K
- Найти: все уникальные пары (a, b), где a + b = K
- Ограничение: решение должно работать за O(N)

Шаг 2. Выбор алгоритма:

Наивный подход $O(N^2)$ — два вложенных цикла — не удовлетворяет ограничению. Используем хэш-таблицу (множество/словарь): для каждого элемента a проверяем, есть ли в множестве уже встреченных элементов число K - a.

Шаг 3. Программа на Python:

```
# Поиск пар с заданной суммой за O(N)
def find_pairs(nums, k):
    seen = set()       # множество уже просмотренных чисел
    pairs = set()      # множество найденных пар

    for num in nums:
        complement = k - num
        if complement in seen:
            # сохраняем пару в отсортированном виде
            pair = (min(num, complement), max(num, complement))
            pairs.add(pair)
        seen.add(num)

    return sorted(pairs)

# Пример использования
nums = [1, 5, 3, 7, 2, 8, 4, 6]
```

```
k = 9
result = find_pairs(nums, k)
for a, b in result:
    print(f'{a} + {b} = {k}')
```

Шаг 4. Тестирование:

Тест	Входные данные	Ожидаемый результат	Статус
Типичный	[1,5,3,7,2,8,4,6], k=9	(1,8),(2,7),(3,6),(4,5)	✓ Пройден
Граничный	[], k=5	[]	✓ Пройден
Дубликаты	[3,3,6], k=6	(3,3)	✓ Пройден
Нет пар	[1,2,3], k=10	[]	✓ Пройден

Шаг 5. Оценка сложности:

- Временная сложность: O(N) — один проход по массиву, операции с хэш-таблицей O(1)
- Пространственная сложность: O(N) — для хранения множеств seen и pairs

3.4. Критерии оценивания

Итоговая оценка выводится следующим образом:

Отлично (5):

- Задача решена полностью, программа проходит все тесты включая граничные
- Алгоритм оптимален или обоснован выбор алгоритма
- Код написан в соответствии со стандартами стиля, снабжён комментариями
- Приведена оценка временной и пространственной сложности

Хорошо (4):

- Задача решена, программа проходит основные тесты
- Алгоритм корректен, но неоптимален (например, $O(N^2)$ вместо O(N))
- Код оформлен, но отдельные части не прокомментированы
- Оценка сложности не приведена или содержит неточность

Удовлетворительно (3):

Оценка «удовлетворительно» ставится, когда программа решает задачу лишь для части случаев, логика в целом верна, но содержит существенные ошибки в обработке граничных случаев или некорректное использование структур данных.

Неудовлетворительно (2):

Оценка «неудовлетворительно» ставится, когда программа не компилируется или выдаёт неверный результат на большинстве тестов, алгоритм принципиально неверен или задача не решена.

ЗАКЛЮЧЕНИЕ

Общие рекомендации для преподавателя:

1. Профессиональная ориентация: формируйте у обучающихся понимание реальных требований IT-отрасли — код-ревью, документирование, командная работа, системы контроля версий.

2. Актуальность технологий: обновляйте задания в соответствии с современным стеком технологий — Python, SQL, Git, облачные сервисы.

3. Соответствие ЕГЭ: регулярно включайте задания формата ЕГЭ (задания 23–27) в текущий контроль для планомерной подготовки к аттестации.

4. Мотивация через реальные задачи: связывайте учебные задания с реальными проблемами — это повышает вовлечённость и понимание ценности обучения.

5. Обратная связь: комментируйте не только ошибки, но и удачные решения — поощряйте элегантный код и нестандартные подходы.

6. Развитие талантов: выявляйте и направляйте технически одарённых обучающихся на олимпиады (ВсОШ, ICPC, олимпиады университетов).

Общие рекомендации для обучающихся:

1. Практика ежедневно: решайте хотя бы одну задачу по программированию каждый день — навык формируется только через регулярную практику

2. Читайте чужой код: изучение решений других программистов расширяет горизонты и знакомит с новыми приёмами.

3. Используйте систему контроля версий: сохраняйте все учебные проекты в Git — это формирует портфолио и профессиональную культуру.

4. Готовьтесь к ЕГЭ целенаправленно: проходите пробные варианты полностью, анализируйте ошибки, работайте над слабыми темами.

5. Участвуйте в командных проектах: умение работать в команде — ключевая компетенция в IT-профессии.

6. Следите за индустрией: читайте технические блоги, смотрите выступления конференций — это помогает понять, зачем нужны изучаемые технологии.

7. Не бойтесь ошибок: ошибка в коде — это не провал, а часть процесса разработки. Умение находить и исправлять ошибки ценится не меньше, чем умение писать код.

8. Задавайте вопросы: если что-то непонятно — спрашивайте преподавателя или ищите ответ в документации. Самостоятельный поиск информации — важнейший навык разработчика.