



РЕКОМЕНДАЦИИ ПО СОВЕРШЕНСТВОВАНИЮ ПРЕПОДАВАНИЯ УЧЕБНОГО ПРЕДМЕТА «ИНФОРМАТИКА» ДЛЯ ВСЕХ ОБУЧАЮЩИХСЯ ГОРОДА МОСКВЫ

Учителям

Проведенный анализ выполнения заданий и выявленные типовые ошибки позволили сформировать комплекс методических рекомендаций, направленных на повышение качества преподавания учебного предмета «Информатика». В рамках учебного процесса особое внимание следует уделить следующим ключевым тематическим блокам:

Работа с электронными таблицами (задание №9)

Для предупреждения ошибок учителям рекомендуются следующие методы и приемы работы на уроках информатики:

1. Акцент на отработку ключевых функций электронных таблиц.

Практикумы по базовым функциям:

- проводить мини-практикумы на каждую из ключевых функций (СЧЁТЕСЛИ, МАКС, МИН, СУММ, СТРОКА), чтобы учащиеся четко понимали их назначение и синтаксис;
- разбирать примеры, где комбинируются несколько функций (например, вычисление суммы без учета минимального и максимального значений).

Тренировка логических условий:

- учить составлять сложные условия с помощью логических операторов (И, ИЛИ), особенно в контексте задач на фильтрацию строк.

2. Алгоритмизация решения задач в электронных таблицах.

Пошаговый разбор типовых задач:

- формировать у учащихся четкий алгоритм решения;
- добавить вспомогательные столбцы для проверки условий;
- использовать функции для проверки уникальности чисел (СЧЁТЕСЛИ);
- вычислить требуемые суммы и сравнить их по условию;
- найти строку с наименьшим номером, удовлетворяющую условиям.

3. Сравнение решений в таблицах и программированием.

Показывать преимущества электронных таблиц:

- разбирать, почему в данном задании таблицы эффективнее программирования (не требуется хранить массив, проще проверять условия);
- сравнивать время решения: таблицы позволяют решить задачу за 5–7 минут, а программирование – дольше и сложнее.



- сравнивать время решения: таблицы позволяют решить задачу за 5–7 минут, а программирование – дольше и сложнее.

Контролировать выбор метода решения:

- учить анализировать условие и выбирать оптимальный инструмент (если задача на фильтрацию и анализ данных – таблицы; если на обработку последовательностей – программирование).

Таким образом, на уроках учителям необходимо проводить регулярные практикумы по составлению формул; проводить сравнительный анализ решений в таблицах и средствами программирования; включать практическую работу на модифицированных заданиях с постепенным усложнением.

Обработка текстовых данных (задание № 24)

В ходе преподавания предмета «Информатика» для повышения эффективности работы учащихся с большими объемами данных, учителям рекомендуется:

1. Углубленное изучение обработки строк и работы с файлами

1.1. Работа с большими текстовыми данными:

- чтение файла по частям (буферизация);
- разбирать методы последовательного чтения (readline, read(N)) для оптимизации памяти.

Сравнивать подходы:

- посимвольная обработка (эффективно для задач с окнами, например, «скользящее окно»);
- загрузка всего файла в память (удобно, но требует контроля за объемом данных);
- практика на реальных больших файлах (1+ МБ) для отработки скорости обработки.

1.2. Алгоритмы поиска подстрок.

Метод «скользящего окна» (актуально для задач, подобных 2025 года).

Разбирать шаблон:

- фиксировать начало и конец окна;
- поддерживать счетчики (например, количество «2025» и «W»);
- сдвигать окно, обновляя счетчики.

```
def find_min_window(filename):  
    with open(filename, 'r') as file:  
        data = file.read()
```

ПРИМЕР

```
left = 0  
count_2025 = 0  
count_W = 0  
min_len = float('inf')  
for right in range(len(data)):  
    # Обновляем счетчики для нового символа справа  
    ...  
    # Если условия выполнены, сдвигаем left  
    ...  
return min_len
```

2. Оптимизация кода и работа со сложными условиями.

2.1. Эффективный подсчет подстрок:

Использование префиксных сумм или хешей для быстрого поиска вхождений «2025».

Счетчик букв «W» через массив/словарь:

```
W count = 0
for char in window:
    if char == 'W':
        W count += 1
```

ПРИМЕР

2.2. Логика условий

Разбор комбинированных условий (например, «не менее 110 раз» + «ровно 90 W»):
– учить выделять независимые условия и проверять их отдельно

```
if (count_2025 >= 110) and (count W == 90):
    min len = min(min len, right - left + 1)
```

ПРИМЕР

Также рекомендуется применять дифференцированный подход. Для слабых групп – упрощенные задачи (например, поиск одного вхождения «2025»); для сильных учащихся – задачи с динамическим изменением условий (например, меняющееся число «W»).

Рекомендуется включить в программу не менее 2–3 уроков в месяц на углубленную отработку подобных задач, начиная с 10 класса.

Практическими подходами на уроках будут являться:

- разработка шаблонных решений для типовых задач;
- анализ производительности разных методов;
- работа с реальными большими файлами (1+ МБ).

Алгоритмы сортировки и анализа данных (задание № 26)

Учителям в ходе изучения темы для формирования умений работать с сортировкой и указателями рекомендуется следующее:

1. Проводить разбор математической модели.

Акцент на условии:

- минимальное неиспользованное число определяет, куда встанет продукт (начало или конец);
- если число относится к уже размещённому продукту, его игнорируют;

Визуализация:

- показывать на доске, как двигаются указатели left и right.

2. Практика программирования.

Тренировка на упрощённых примерах – дать задачу с N=3 и разобрать пошагово. Оптимизация кода – учить использовать sort() и множества (set) для ускорения.

3. Работа с файлами.

Чтение данных – автоматизировать обработку входного файла:

ПРИМЕР

```
with open('26.txt') as f:
    N = int(f.readline())
    products = []
    for line in f:
        s, e = map(int, line.split())
        ...
```

Большинство ошибок связано с невнимательностью к условию и недостатком практики. Регулярные тренировки на аналогичных задачах снизят количество недочётов.

Важно помнить, что ключевыми компетенциями по итогам изучения тем являются:

Математическое моделирование:

- формализация текстовых условий;
- перевод задачи в алгоритмическую форму.

Эффективная сортировка:

- применение стандартных методов сортировки;
- оптимизация работы с указателями;
- обработка уже рассмотренных элементов.

Работа с файлами:

- автоматизация чтения и обработки входных данных;

– валидация входных параметров.

Поэтому также важным на уроках является визуализация алгоритмов сортировки, пошаговый разбор на упрощённых примерах, Практика решения задач с ограничениями по времени.

Алгоритмизация и программирование (обработка числовых последовательностей) (задание № 17)

В ходе преподавания предмета учителю важно уделить внимание формированию базовых навыков работы с последовательностями, а именно, проводить регулярные практикумы по чтению числовых данных из файлов, определению характеристик чисел (последние цифры, разряды), работе с отрицательными значениями. Пример упражнения:

```
# Определение чисел, оканчивающихся на 27
def ends_with_27(num):
    return abs(num) % 100 == 27
```

ПРИМЕР

Также будет эффективным ввести систему постепенно усложняющихся задач и использовать систему упражнений для разбиения сложных условий на простые компоненты, пошаговой проверки каждого условия, оптимизации порядка проверок.

Дополнительно на протяжении всего времени преподавания предмета «Информатика», учителям рекомендуется:

- Уделить время решению задач на перестановки букв в слове с ограничениями при большой длине слов и номерам слов по порядку при ограничениях на входящие в него буквы. Научить решать эти задачи аналитически с использованием электронных средств счета.
- При решении задач на формальное исполнение простого алгоритма, записанного на естественном языке уделить внимание особенностям перевода чисел в произвольную позиционную систему счисления. Дать общий алгоритм решения подобных задач.
- В теме алфавитный подход к измерению информации рассмотреть задачи с алфавитами большой мощности или идентификаторами (более 1000 символов), различными ограничениями на дополнительные сведения при кодировании информации. А также разобрать задачи на занимаемый объем памяти при ограничениях сверху или снизу.

- При работе с электронными таблицами обратить внимание на особенности использования стандартных функций, абсолютных и относительных ссылок. Использовать шаблонные файлы с известным результатом для проверки решаемых задач.
- Уделить время на решение задач по учебным исполнителям. Рассмотреть решение этих задач разными методами и с разными вопросами, требующимися в качестве ответа.

***ИПК / ИРО, иным организациям, реализующим программы
профессионального развития учителей***

В рекомендациях необходимо указывать, что они именно по данному предмету и направлены на совершенствование преподавания конкретно этого предмета и должны носить практический характер, а также быть легко реализуемыми.

- При обучении учителей обратить внимание на математические основы информатики: системы счисления, кодирование, подходы к измерению информации. Показать связь этих тем с алгоритмизацией.
- В систему повышения квалификации включить различных учебных исполнителей, их возможную программную реализацию и способы аналитического решения.
- Продумать работу с учителями по вероятным направлениям развития сложности задач с иллюстрацией на примерах.
- Показать для одних классов задач применение только аналитических способов решения, для других – и аналитических, и программных способов решения. А также указать границы применимости определенного способа решения.
- В систему повышения квалификации включить вопросы алгоритмизации и программирования, необходимые для успешного прохождения итоговой аттестации (сортировка, обработка строк, работа с отрицательными числами, оценка эффективности алгоритма и др).