МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ОЛИМПИАДА ШКОЛЬНИКОВ «ГРАНИТ НАУКИ»

ИНФОРМАТИКА

Методические указания для подготовки кзаключительному туру олимпиады в 2024/2025 учебном году для 9 классов

Санкт-Петербург, 2025

СОДЕРЖАНИЕ

Информатика

Характер и уровень сложности олимпиадных задач направлены на достижение целей проведения олимпиады: выявить способных участников, твердо владеющих школьной программой и наиболее подготовленных к освоению образовательных программ технических ВУЗов, обладающих логикой и творческим характером мышления, умеющих алгоритмически описать реальные ситуации из различных предметных областей и применить к ним наиболее подходящие методы информатики.

Задания дифференцированы по сложности и требуют различных временных затрат на верное и полное решение. Они охватывают все разделы школьной программы, но носят, в большинстве, комплексный характер, позволяющий варьировать оценки в зависимости от проявленных в решении творческих подходов и продемонстрированных технических навыков. Участники должны самостоятельно разбить задачу на подзадачи, грамотно выполнить решение каждой подзадачи, синтезировать решение всей задачи из решений отдельных подзадач. Успешное выполнение олимпиадной работы не предполагает знаний, выходящих за пределы школьной программы, но требует творческого подхода, логического мышления, умения увидеть и составить правильный и оптимальный план решения, четкого и технически грамотного выполнения каждой части решения.

Очный тур олимпиады проводится только в письменной форме. Каждый участник олимпиады получает билет, содержащий шесть заданий. При выполнении заданий требуется дать теоретическое обоснование и привести все этапы решения задачи:

– при определении результата перевода заданного числа из одной системы счисления в другую следует указать правило перевода в общем виде, а затем привести схему перевода заданного числа в указанную систему счисления;

– при определении значений ячеек электронной таблицы приводятся результаты, формулы, полученные при копировании, даются пояснения о типах ссылок в формулах, как типы ссылок влияют на результат и др.;

– при ответе на вопрос об определении значения переменной после выполнения программы, алгоритм которой приводится в билете, следует пояснить работу каждого оператора (блока), выписать промежуточные и окончательный результаты;

– в случае решения задач, связанных с таблицами истинности необходимо описывать все промежуточные результаты;

– при решении задачи на программирование необходимо составить блок-схему или описать алгоритм решения задачи, написать текст программы на любом из языков программирования, указав используемый язык программирования и его версию, допускается использовать только стандартную библиотеку языка. Алгоритм должен раскрывать решение на каждом шаге. Запрещено использовать уникальные особенности языка программирования (классы, методы, объекты, специальные функции и библиотеки и др.) для реализации решения. Считывание заранее рассчитанных значений не засчитывается за решение задачи. Код должен содержать достаточное для понимания логики работы количество комментариев;

– программы, составленные участниками, тестируются и всесторонне оцениваются с учетом быстродействия, рациональности, минимального использования памяти компьютера и оригинальности предложенных решений;

– при использовании методов научного перебора необходимо привести доказательства использования этого метода;

При подготовке к олимпиаде следует повторить приведенные ниже темы.

Теоретические основы информатики

Модель. Задачи, решаемые с помощью моделирования. Классификации моделей. Материальные (натурные) и информационные модели. Непрерывные и дискретные модели. Имитационные модели. Игровые модели. Оценка адекватности модели моделируемому объекту и целям моделирования. Табличные модели. Таблица как представление отношения.

Базы данных. Отбор в таблице строк, удовлетворяющих заданному условию.

Граф. Вершина, ребро, путь. Ориентированные и неориентированные графы. Длина (вес) ребра. Весовая матрица графа. Длина пути между вершинами графа. Поиск оптимального пути в графе. Начальная вершина (источник) и конечная вершина (сток) в ориентированном графе. Вычисление количества путей в направленном ациклическом графе.

Дерево. Корень, вершина (узел), лист, ребро (дуга) дерева. Высота дерева. Поддерево. Примеры использования деревьев. Перебор вариантов с помощью дерева.

Понятие математической модели. Задачи, решаемые с помощью математического (компьютерного) моделирования.

Универсальность дискретного представления информации. Двоичное кодирование. Кодирование текстов. Равномерный код. Неравномерный код. Кодировка ASCII. Понятие о кодировках UNICODE. Декодирование сообщений с использованием равномерного и неравномерного кода. Общее представление о цифровом представлении аудио-визуальных и других непрерывных данных. Кодирование цвета. Цветовые модели. Модель RGB. Глубина кодирования. Палитра. Растровое и векторное представление изображений. Пиксель. Оценка информационного объёма графических данных для растрового изображения. Кодирование звука.

Системы счисления. Свойства позиционной записи числа: количество цифр в записи, признак делимости числа на основание системы счисления. Алгоритм перевода целого числа из P-ичной системы счисления в десятичную. Алгоритм перевода целого числа из десятичной системы счисления в P-ичную. Двоичная, восьмеричная и шестнадцатеричная системы счисления; перевод чисел между этими системами. Представление целых и вещественных чисел в памяти компьютера.

Логические высказывания. Логические значения высказываний. Элементарные и составные высказывания. Логические операции: «и» (конъюнкция, логическое умножение), «или» (дизъюнкция, логическое сложение), «не» (логическое отрицание). Приоритет логических операций. Определение истинности составного высказывания, если известны значения истинности входящих в него элементарных высказываний. Логические выражения. Правила записи логических выражений. Построение таблиц истинности логических выражений. Логические элементы. Знакомство с логическими основами компьютера.

Информационные технологии

Понятие об электронных таблицах. Типы данных в ячейках электронной таблицы. Редактирование и форматирование таблиц. Встроенные функции для поиска максимума, минимума, суммы и среднего арифметического. Сортировка данных в выделенном диапазоне. Построение диаграмм (гистограмма, круговая диаграмма, точечная диаграмма). Выбор типа диаграммы.

Преобразование формул при копировании. Относительная, абсолютная и смешанная адресация.

Условные вычисления в электронных таблицах. Суммирование и подсчёт значений,

отвечающих заданному условию. Обработка больших наборов данных. Численное моделирование в электронных таблицах.

Алгоритмы и программирование

Понятие алгоритма. Исполнители алгоритмов. Алгоритм как план управления исполнителем. Свойства алгоритма. Способы записи алгоритма (словесный, в виде блок-схемы, программа). Разбиение задачи на подзадачи. Составление алгоритмов и программ с использованием ветвлений, циклов и вспомогательных алгоритмов для управления исполнителем Робот или другими исполнителями, такими как Черепашка, Чертёжник и др.

Этапы решения задач на компьютере. Язык программирования (Паскаль, Python, Java, C++, C#). Основные конструкции языка программирования. Типы данных: целочисленные, вещественные, символьные, логические. Ветвления. Составные условия. Циклы с условием. Циклы по переменной. Использование таблиц трассировки.

Обработка символьных данных. Встроенные функции языка программирования для обработки символьных строк. Алгоритмы редактирования текстов (замена символа/фрагмента, удаление и вставка символа/фрагмента, поиск вхождения заданного образца).

Табличные величины (массивы). Понятие о двумерных массивах (матрицах). Алгоритмы работы с элементами массива с однократным просмотром массива: суммирование элементов массива; подсчёт количества (суммы) элементов массива, удовлетворяющих заданному условию; нахождение наибольшего (наименьшего) значения элементов массива; нахождение второго по величине наибольшего (наименьшего) значения; линейный поиск элемента; перестановка элементов массива в обратном порядке.

Сортировка одномерного массива. Методы сортировки (метод пузырька, метод выбора, сортировка вставками, сортировка подсчетом), применение встроенных сортировок.

Подпрограммы. Рекурсивные алгоритмы.

Целочисленная арифметика. Арифметические операции (умножение, деление, остатки, сложение, вычитание). Битовые операции и работа с отдельными битами.

Теоретико-числовые алгоритмы. НОД и НОК, системы счисления, длинная арифметика, простые числа и разложение на делители, остатки, быстрое возведение в степень.

Алгоритмы поиска. Линейный поиск, двоичный поиск, поиск подстроки в строке, два указателя.

Динамическое программирование. Рекуррентные последовательности, простое динамическое программирование. Динамическое программирование с несколькими параметрами, по подстрокам, по подмножествам, по профилю, по поддеревьям, на ациклических графах.

Жадный алгоритм. Области применения и стандартные задачи, решаемые жадным алгоритмом. Доказательство применимости.

Алгоритмы на невзвешенных графах. Обход в ширину и глубину и их применение. Топологическая сортировка, компоненты связности, поиск циклов, проверка на двудольность, мосты, точки сочленения, конденсация. Паросочетания. Эйлеров цикл.

Алгоритмы на взвешенных графах. Поиск кратчайших путей: алгоритмы Дейкстры, Беллмана-Форда, Флойда. Минимальные остовные деревья. Потоки.

Вычислительная геометрия. Скалярное и косое произведение. Площади. Взаимное расположение фигур на плоскости и в пространстве. Выпуклые оболочки.

Линейные структуры данных. Стек, дек, очередь. Решение задачи о проверки правильной скобочной последовательности, минимум в окне, обратная польская нотация.

Деревья. Бинарное дерево поиска. Сбалансированность бинарных деревьев поиска. Корневые деревья, система непересекающихся множеств. Дерево отрезков, решение задач RMQ и RSQ. Куча. Дерево Фенвика. Декартово дерево.

Сложность вычисления: количество выполненных операций, размер используемой памяти; зависимость количества операций от размера исходных данных.

Задания для подготовки к олимпиаде

Задача 1. Исследование поведения формул в Excel

С тех пор, как Иван изучил формулы в Excel, он ежедневно оттачивает свое мастерство. Сегодня он записал число 2 в ячейку A1, в соседнюю ячейку B1 он ввел формулу **=ОКРУГЛ(ОСТАТ(8;$A$1)+СТЕПЕНЬ(A1;2)-ЕСЛИ(A1>0;A1^2/3;A1/3);1)** и растянул до ячейки F1. На следующей строке, начиная с ячейки A2 Иван ввел формулу **=ОКРУГЛ(ОСТАТ(B1;A1)-КОРЕНЬ(ABS(A1-B1))+ЕСЛИ(A1+B1<2;B1/A1;A1+3);2)** и скопировал в диапазон B2:F2. На следующей строке он ввел в первую ячейку число 2,5 и повторил действия, как для первой строки. А в четвертую (диапазон A4:F4) скопировал формулу из 2 строки. В пятой строке Иван ввел в первую ячейку число 3 и снова решил повторить действия, как и для строки 1. В ячейку A6 он вставил формулу из A2 и растянул до ячейки F6. После этих действий Иван решил воспользоваться инструментом условного форматирования, выставив следующие условия:

– значения больше $D$1 выделить красным цветом;

– значения меньше $C$3 выделить желтым цветом;

– значения между 10 и 16 выделить зеленым цветом.

Укажите, какое количество ячеек оказалось выделено желтым цветом?

Ответ: 15

Решение:

	A	B	C	D	E	F
1	2	2,7	4,9	16	170,7	19425,7
2	4,86	6,42	5,87	17,26	171,54	19289,32
3	2,5	4,2	11,8	92,8	5741,2	21974252
4	5,9	7,84	16	101,04	3736,34	21969567
5	3	6	24	384	98304	6,44E+09
6	4,27	4,76	8,03	74,08	18042,73	6,44E+09

Фрагмент в режиме отображения формул:

H2

	A	B
1	2	=ОКРУГЛ(ОСТАТ(8;$A$1)+СТЕПЕНЬ(A1;2)-ЕСЛИ(A1>0;A1^2/3;A1/3);1)
2	=ОКРУГЛ(ОСТАТ(B1;A1)-КОРЕНЬ(ABS(A1-B1))+ЕСЛИ(A1+B1<2;B1/A1;A1+3);2)	=ОКРУГЛ(ОСТАТ(C1;B1)-КОРЕНЬ(ABS(B1-C1))+ЕСЛИ(B1+C1<2;C1/B1;B1+3);2)
3	2,5	=ОКРУГЛ(ОСТАТ(8;$A$1)+СТЕПЕНЬ(A3;2)-ЕСЛИ(A3>0;A3^2/3;A3/3);1)
4	=ОКРУГЛ(ОСТАТ(B3;A3)-КОРЕНЬ(ABS(A3-B3))+ЕСЛИ(A3+B3<2;B3/A3;A3+3);2)	=ОКРУГЛ(ОСТАТ(C3;B3)-КОРЕНЬ(ABS(B3-C3))+ЕСЛИ(B3+C3<2;C3/B3;B3+3);2)
5	3	=ОКРУГЛ(ОСТАТ(8;$A$1)+СТЕПЕНЬ(A5;2)-ЕСЛИ(A5>0;A5^2/3;A5/3);1)
6	=ОКРУГЛ(ОСТАТ(B5;A5)-КОРЕНЬ(ABS(A5-B5))+ЕСЛИ(A5+B5<2;B5/A5;A5+3);2)	=ОКРУГЛ(ОСТАТ(C5;B5)-КОРЕНЬ(ABS(B5-C5))+ЕСЛИ(B5+C5<2;C5/B5;B5+3);2)
7		
8		

Задача 2. Вскрытие сейфа

Для получения секретной информации нашим разведчикам необходимо вскрыть сейф, замок которого состоит из 5 ячеек (B2:F2). Задача осложняется тем, что код можно ввести только один раз (метод подбора не поможет), в ячейках может находиться как числовая, так и текстовая информация, и самое главное, формулы с подсказками из ячеек B4:B12 были скопированы в ячейки C5:C13, что привело к смещению ссылок.

Помогите разведчикам найти код и выиграть эту схватку. В Вашем распоряжении значения формул с подсказками в первоначальном состоянии (B4:B12) и формулы в режиме отображений формул после операции копирования (C5:C13).

	A	B	C	D	E	F	G	H	I	J
1										
2	Код:									
3										

	A	B
4	№1	0
5	№2	4
6	№3	3
7	№4	#ДЕЛ/0!
8	№5	#ЗНАЧ!
9	№6	дбор
10	№7	3
11	№8	4
12	№9	17
13		

	C
4	
5	=СЧЁТЕСЛИ(C3:G3;">=10")+СЧЁТЕСЛИ(C3:G3;"<=0")
6	=СЧЁТ(C3:G3)
7	=ДЛСТР(ЕСЛИ(ЕТЕКСТ(D3);D3;ТЕКСТ(D3;"#")))+ДЛСТР(ЕСЛИ(ЕТЕКСТ(E3);E3;ТЕКСТ(E3;"#")))
8	=СУММ(C3:E3)/(F$2-G$2)
9	=$D$2+F$2+G$2
10	=ПСТР("Подбор";$F3;6)
11	=ПОИСК(СЦЕПИТЬ(ЕСЛИ(ЕТЕКСТ(D3);D3;"");ЕСЛИ(ЕТЕКСТ(E3);E3;""));"гипнотизер")
12	=ДЛСТР(СЦЕПИТЬ(ЕСЛИ(ЕТЕКСТ(D3);D3;"");ЕСЛИ(ЕТЕКСТ(E3);E3;"")))+C3
13	=СУММ(C3:G3)

Решение:

1) Находим первоначальное значение формул с учетом операции копирования.

	A	B
4	№1	=СЧЁТЕСЛИ(B2:F2;">=10")+СЧЁТЕСЛИ(B2:F2;"<=0")
5	№2	=СЧЁТ(B2:F2)
6	№3	=ДЛСТР(ЕСЛИ(ЕТЕКСТ(C2);C2;ТЕКСТ(C2;"#")))+ДЛСТР(ЕСЛИ(ЕТЕКСТ(D2);D2;ТЕКСТ(D2;"#")))
7	№4	=СУММ(B2:D2)/(E$2-F$2)
8	№5	=$D$2+E$2+F$2
9	№6	=ПСТР("Подбор";$F2;6)
10	№7	=ПОИСК(СЦЕПИТЬ(ЕСЛИ(ЕТЕКСТ(C2);C2;"");ЕСЛИ(ЕТЕКСТ(D2);D2;""));"гипнотизер")
11	№8	=ДЛСТР(СЦЕПИТЬ(ЕСЛИ(ЕТЕКСТ(C2);C2;"");ЕСЛИ(ЕТЕКСТ(D2);D2;"")))+B2
12	№9	=СУММ(B2:F2)

2) По формуле №1, находящейся в ячейке B4 «=СЧЁТЕСЛИ(B2:F2;">=10")+СЧЁТЕСЛИ(B2:F2;"<=0")» равной 0 определяем, что все числовые значения находится в диапазоне от 1 до 9.

3) По формуле №2,находящейся в ячейке B5 «=СЧЁТ(B2:F2)» равной4 определяем, что количество числовых значений равно 4, следовательно одна из ячеек содержит текстовую информацию.

4) Формула №3, находящаяся в ячейке B6 «=ДЛСТР(ЕСЛИ(ЕТЕКСТ(C2);C2;ТЕКСТ(C2;"#")))+ДЛСТР(ЕСЛИ(ЕТЕКСТ(D2);D2;ТЕКСТ(D2;"#")))» равная 3 подсчитывает количество символов в ячейках C2 и D2, переводя числовые значения в текстовые. В соответствии с тем, что числовые значения находятся в диапазоне от 1 до 9, сумма количества символов, при условии нахождения в этих ячейках числовых значений, равнялась бы 2. Значение 3 говорит о том, что одна из ячеек имеет длину 2 символа и это – ячейка с текстовой информацией.

5) Значение формулы №4, находящейся в ячейке B7 «=СУММ(B2:D2)/(E$2-F$2)» является сообщением об ошибке деления на 0 «#ДЕЛ/0!», это возможно при условии, что значения в ячейках E2 и F2 одинаковы.

6) Значение формулы №5, находящейся в ячейке B8 «=$D$2+E$2+F$2» является сообщением об ошибке «#ЗНАЧ!» означающее, что значение используемое в формуле, имеет неправильный тип данных. Это возможно в том случае, если в этих ячейках находится текстовая информация. В соответствии с пунктом 4 данного решения мы знаем, что текст находится либо в ячейкеC2 либо D2, следовательно ячейка с текстовой информацией– D2.

7) Значение формулы №6, находящейся в ячейке B9 «=ПСТР("Подбор";$F2;6)» равно подстроке «дбор», откуда F2 равно 3, а значения в ячейках E2 и F2 одинаковы, в соответствии с пунктом 5 данного решения. Следовательно, E2=F2=3.

8) Формула №7, находящаяся в ячейке B10 «=ПОИСК(СЦЕПИТЬ(ЕСЛИ(ЕТЕКСТ(C2);C2;"");ЕСЛИ(ЕТЕКСТ(D2);D2;""));»гипнотизер")» показывает, что наше текстовое поле имеется в слове «гипнотизер» начиная с третьей позиции. Зная, что его длина 2 символа, в соответствии с пунктом 4 решения, получаем значение ячейки D2 = «пн».

9) Формула №8, находящаясяв ячейке B11 «=ДЛСТР(СЦЕПИТЬ(ЕСЛИ(ЕТЕКСТ(C2);C2;"");ЕСЛИ(Е ТЕКСТ(D2);D2;"")))+B2»равная 4, позволяет найти значение ячейки B2, как разницы между значением формулы и длиной текстового поляячейки D2. B2=4-2=2.

10) Формула №9, находящаясяв ячейке B12 «=СУММ(B2:F2)» позволяет найти значение последней неизвестной ячейки C2 = 17-2-3-3 = 9.

Ответ:

B2	C2	D2	E2	F2
2	9	пн	3	3

Задача 3. Чертёжник

Исполнитель Чертёжник перемещается на координатной плоскости, оставляя след в виде линии. Чертёжник может выполнять следующие команды:

1) `Сместиться на (a, b)`, где a, b – целые числа. Эта команда перемещает исполнитель из точки с координатами (x, y) в точку с координатами (x + a; y + b). Например, если Чертёжник находится в точке с координатами (1, 2), то команда сместиться на (2, −1) переместит Чертёжника в точку (3, 1).

2) Цикл:

```
ПОКА <условие>
     последовательность команд
КОНЕЦ ПОКА
```

Этот цикл означает, что последовательность команд будет выполнятся до тех пор, пока условие <условие> выполняется.

Исполнителю Чертёжник был дан для исполнения следующий алгоритм (буквами a, b обозначены неизвестные числа):

```
НАЧАЛО
     сместиться на (-2, 1)
ПОКА x + y не равно 4
     сместиться на (a, b)
     сместиться на (3, 1)
КОНЕЦ ПОКА
     сместиться на (-19, -7)
КОНЕЦ
```

После выполнения программы Чертёжник сместился в точку, для которой значения координат x и y изменятся относительно начальных координат на одну и туже величину, Сколько раз выполнилась последовательность команд цикла ПОКА, если известно, что команды цикла ПОКА были выполнены более одного раза и начальные значения координат x и y были одинаковы по модулю и противоположны по знаку.

Решение:

Значение координат Чертёжника в результате выполнения цикла ПОКА:

$$x_n = x_0 + \Delta x = x_0 - 2 + N(a + 3)$$
$$y_n = y_0 + \Delta y = xy_0 + 1 + N(b + 1)$$

В этом уравнении N – количество повторений команд цикла ПОКА, которое является искомым значением.

Из условия завершения цикла $x_n + y_n = 4$, получим:

$$x_0 - 2 + N(a + 3) + y_0 + 1 + N(b + 1) = 4$$

Из условия $x_0 = -y_0$, получим: $N(a + b + 4) = 5$.

Общее изменение координат Чертёжника в результате выполнения этого алгоритма:

$$x - x_0 = -2 + N(a + 3) - 19$$

$$y - y_0 = 1 + N(b + 1) - 7$$

Так как координаты x и y изменятся на одну и ту же величину, получим:

$$-2 + N(a + 3) - 19 = 1 + N(b + 1) - 7 \text{ или } N(a - b - 2) = 15.$$

После преобразований, получим систему уравнений:

$$\begin{cases} N(a + b + 4) = 5 \\ N(a - b - 2) = 15 \end{cases}$$

Нужно найти натуральное N, при котором эта система уравнений разрешима в целых числах относительно a и b. Для этого число N должно быть одновременно делителем чисел 5 и 10. Общими делителями этих чисел являются числа 1 и 5, при этом 1 не подходит по условию, что цикл повторялся более 1 раза.

Ответ: последовательность команд цикла ПОКА выполнилась 5 раз.

Задача 4. День рождения

Петя решил зашифровать день и месяц своего рождения. Для этого он придумал следующее логическое выражение:

$a \vee b \rightarrow \neg b \wedge \neg c \vee a$.

Если построить таблицу истинности этого выражения, то в результирующем столбце отобразится двоичное число. Первые 4 цифры покажут месяц, а последующие – день рождения Пети. Строки в таблице истинности идут последовательно в лексикографическом порядке 000, 001, 010 и т.д.

Найти дату рождения Пети.

Решение

Составим таблицу истинности для нашего выражения:

a	b	c	aVb	¬b	¬c	¬bΛ¬c	¬bΛ¬cVa	aVb→(¬bΛ¬cVa)
0	0	0	0	1	1	1	1	1
0	0	1	0	1	0	0	0	1
0	1	0	1	0	1	0	0	0
0	1	1	1	0	0	0	0	0
1	0	0	1	1	1	1	1	1
1	0	1	1	1	0	0	1	1
1	1	0	1	0	1	0	1	1
1	1	1	1	0	0	0	1	1

Взяв из последнего столбика первые 4 строчки получим $1100_2=12_{10}$, а взяв последние 4 строчки получим $1111_2=15_{10}$. Таким образом, день рождения Пети 15.12.

Ответ: 15.12

Задача 5. Зашифрованное послание

Юный криптограф Скрипт получил секретное послание, зашифрованное с помощью простого шифра Цезаря, модифицированного для использования троичной системы счисления. Послание представляет собой последовательность троичных чисел, каждое из которых соответствует букве русского алфавита. Для каждой буквы используется её порядковый номер в алфавите (А = 0, Б = 1, В = 2, …, Я = 32), который затем преобразуется в троичное представление. Шифр Цезаря применяется к троичному представлению номера буквы. Ключ шифрования – троичное число 12.

Зашифрованный текст, полученный Скриптом, выглядит следующим образом: 20220 12122 101000 110121 2210 102012. Расшифруйте послание, учитывая, что ключ применяется к каждому троичному числу по модулю 33 (количество букв в русском алфавите). Напишите расшифрованный текст в виде последовательности букв русского алфавита.

Решение:

На первом этапе преобразуем ключ шифрования, а также все числа в коде из троичной в десятичную систему счисления:

$12_3=1\times3^1+2\times3^0=5$

$20220_3=2\times3^4+2\times3^2+2\times3^1=186$

$12122_3=1\times3^4+2\times3^3+1\times3^2+2\times3^1+2\times3^0=152$

$101000_3=1\times3^5+1\times3^3=270$

$110121_3=1\times3^5+1\times3^4+1\times3^2+2\times3^1+1\times3^0=340$

$2210_3=2\times3^3+2\times3^2+1\times3^1=75$

$102012_3 = 1\times3^5+2\times3^3+1\times3^1+2\times3^0=302$

На втором этапе для каждой десятичной цифры в зашифрованном тексте согласно принципу работы шифра Цезаря нужно вычесть значение ключа 5 по модулю 33, т.е., если x – десятичное число, представляющее букву, то номер буквы в алфавите – остаток от деления (x – 5) на 33. Полученное десятичное число представляет номер буквы в алфавите. Далее по номеру буквы (от 0 до 32) определяется соответствующая буква русского алфавита.

Код	Дешифровка	Символ
186	(186 – 5) делится на 33 с остатком **16**	**П**
152	(152 – 5) делится на 33 с остатком **15**	**О**
270	(270 – 5) делится на 33 с остатком **1**	**Б**
340	(340 – 5) делится на 33 с остатком **5**	**Е**
75	(75 – 5) делится на 33 с остатком **4**	**Д**
302	(302 – 5) делится на 33 с остатком **0**	**А**

Ответ: ПОБЕДА

Задача 6. Шифрующая перестановка

Перестановка P длины n – это упорядоченный набор, содержащий числа от 1 до n , каждое из которых входит в него ровно один раз. Например, перестановкой длины 13 является набор **(5 11 13 12 6 1 8 4 10 9 7 2 3)**. При помощи перестановки букв можно зашифровать слово. Для примера возьмем приведенную выше перестановку и слово transposition, которое состоит тоже из 13 букв. Далее, следуя перестановке, на первую позицию поставим пятую букву слова, на вторую – одиннадцатую букву, …, на последнюю позицию – третью букву слова. В итоге получим sinoptsntiora. К этому слову снова применим эту же перестановку и получим poartsnoitsin. Повторив эти стадии шифрования k раз, получим зашифрованное сообщение.

t	**r**	**a**	**n**	**s**	**p**	**o**	**s**	**i**	**t**	**i**	**o**	**n**
1	2	3	4	5	6	7	8	9	10	11	12	13
5	11	13	12	6	1	8	4	10	9	7	2	3
s	**i**	**n**	**o**	**p**	**t**	**s**	**n**	**t**	**i**	**o**	**r**	**a**
1	2	3	4	5	6	7	8	9	10	11	12	13
5	11	13	12	6	1	8	4	10	9	7	2	3
p	**o**	**a**	**r**	**t**	**s**	**n**	**o**	**i**	**t**	**s**	**i**	**n**

Вам дано зашифрованное таким образом слово, шифрующая перестановка P и число k . Необходимо восстановить слово.

Формат входных данных

Первая строка содержит два числа n и k – длину перестановки и число стадий шифрования (1 ≤ n, k ≤ 1000). Вторая строка содержит перестановку длины n , числа в ней разделены одним пробелом. Третья строка содержит зашифрованное слово, оно состоит из n прописных букв латиницы.

Формат выходных данных

Вывести одну строку – исходное слово.

Примеры

стандартный ввод	стандартный вывод
13 2 5 11 13 12 6 1 8 4 10 9 7 2 3 poartsnoitsin	transposition

Решение

Инициализация и ввод данных:

Сначала программа считывает количество символов в строке и количество итераций, за которыми следует вектор перестановок и сама строка.

Дешифрование:

Процесс дешифрования заключается в том, что за каждую итерацию создается новая копия строки, основанная на текущем состоянии строки vs. Символы из vs размещаются в строке s по индексам, указанным в векторе p. Обратите внимание, что индексы в векторе p начинаются с 1, поэтому нужно вычитать 1 при доступе к элементам.

Обновление строки:

После завершения всех перестановок на каждой итерации программа обновляет строку vs текущим состоянием s, таким образом, в следующем цикле будет использоваться уже изменённое состояние строки.

Вывод результата:

После того как все итерации завершены, финальная результирующая строка выводится на экран.

Сложность данного алгоритма можно проанализировать, исходя из структуры вложенных циклов. Внешний цикл выполняется k раз, где k – это количество повторений перестановки. Это означает, что алгоритм повторяет процесс перестановки строк k раз.

Внутренний цикл работает n раз, где n – длина строки. На каждой итерации внутреннего цикла происходит переопределение символов в строке с учетом перестановки.

Таким образом, суммарная сложность алгоритма может быть выражена как: O(k×n).

C++

```
	#include <bits/stdc++.h>
	using namespace std;
	// Объявление переменных для хранения длины строки, количества
перестановок, самой строки и временной строки.
	int n, k;
	string s, vs;
```

```
      int main() {
    // Считываем длину строки (n) и количество применений перестановки (k).
          cin >> n >> k;
          // Создаем вектор p для хранения перестановки.
          vector<int> p(n);
          // Считываем саму перестановку.
          for(int i = 0; i < n; i++) {
              cin >> p[i];
          }
          // Считываем строку, которую нужно дешифровать.
          cin >> s;
          // Изначально присваиваем vs значение входной строки s.
          vs = s;
          // Делаем k итераций применения перестановки.
          for(int i = 0; i < k; i++) {
              // На каждой итерации создаем циклом новое зашифрованное слово.
              for(int j = 0; j < n; j++) {
            // Применяем перестановку: p[j] - 1 дает индекс для s.
            /* То есть берем символ из временной строки vs и помещаем его на
нужное место в s.*/
                  s[p[j] - 1] = vs[j];
              }
              /* После завершения внутреннего цикла обновляем vs значением
зашифрованного слова s. */
              vs = s;
          }
          // Выводим окончательную строку после всех применений перестановки.
          cout << s;
          return 0;
      }
```

Python

```
      def encrypt_with_permutation(n, k, permutation, s):
          vs = s  # Сохраняем исходное слово
          for _ in range(k):
              # Создаем новый список символов для зашифрованного слова
              s_list = [''] * n
              for j in range(n):
                 # Применяем перестановку: индексы учитываются с 0 (в Python)
                  s_list[permutation[j] - 1] = vs[j]
              vs = ''.join(s_list)  # Обновляем vs с зашифрованным словом
          return vs

      if __name__ == "__main__":
          n, k = map(int, input().split())
          permutation = list(map(int, input().split()))
          s = input("")
          encrypted_message = encrypt_with_permutation(n, k, permutation, s)
          print(encrypted_message)
```

Задача 7. Очередь

Во время перерыва у дверей буфета выстроилась очередь из **N** студентов. Каждый из вновь подходивших студентов либо вставал в конец очереди, либо умудрялся вклиниться между уже стоящими, либо даже вставал в начало очереди. В итоге, к моменту открытия буфета, номер прихода студента и его номер в очереди могли не совпадать.

Требуется вывести для каждого студента в очереди к моменту открытия буфета, число, соответствующее номеру его прихода.

Входные данные:

В первой строке содержится число **N ($1 \le N \le 10^5$)** – количество человек в очереди. Далее в следующей строке через пробел содержатся N чисел – описание процесса формирования очереди. k-е число, равное s, говорит о том, что студент, пришедший k-м по счету, становится сразу за студентом, пришедшим s-м по счету. Если текущий элемент последовательности равен 0, это говорит о том, что соответствующий студент встал в очереди самым первым. Гарантируется, что число во входной последовательности всегда меньше его номера.

Выходные данные:

В единственной строке выдать через пробел N чисел, соответствующих номерам прихода студентов, номера должны располагаться по порядку, соответствующему расположению студентов в очереди в момент открытия буфета, начиная с ее начала.

Примеры

стандартный ввод	стандартный вывод
10 0 1 0 1 3 5 4 1 2 7	3 5 6 1 8 4 7 10 2 9

Решение:

Эту задачу можно решить неэффективно, создавая сразу искомый список, используя стандартные методы или реализовав связный список на python:

```
n = int(input())
A = list(map(int, input().split()))
B = []
for i in range(len(A)):
    if A[i] == 0:
        B.insert(0,(i + 1))
    else:
        j = 0
        while B[j] != A[i]:
            j += 1
        B.insert(j + 1,(i + 1))
print(" ".join(map(str, B)))
```

На каждом шаге итерации i в цикле выполняются операции вставки и поиска, каждая из которых может занимать O(m), где m — текущая длина списка B. В худшем случае, когда все студенты будут вставляться в начало или в середину списка, количество элементов в B будет увеличиваться до n. Таким образом, в наихудшем случае (когда A содержит много нулей или многократные вхождения), каждая вставка может занимать O(n), следовательно, общая сложность алгоритма будет равна: $O(n^2)$, где n — количество студентов.

Рассмотрим более эффективное решение. Считываем данные:

n = 10

A = [0, 1, 0, 1, 3, 5, 4, 1, 2, 7].

Создаем список next_stud, длиной на 1 элемент больше чем A, заполняем 0, на 0 позицию помещаем n + 1:

next_stud = [11, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

Размещаем в списке B на i позиции номер прихода студента который стоит за i в очереди, на 0 позиции – первый в очереди. Первый пришедший становится первым, 1 записываем в 0 позицию, тот номер, что находился в 0 позиции теперь будет стоять за первым пришедшим, в списке под индексом 1, 11 переписываем в 1 позицию:

[1, 11, 0, 0, 0, 0, 0, 0, 0, 0, 0]

Далее в 1, следовательно, 2 пришедший стоит за 1, в элемент списка с индексом 1 записываем 2, то, что было в ячейке переносим в элемент с индексом 2:

[1, 2, 11, 0, 0, 0, 0, 0, 0, 0, 0]

Далее следует 0, следовательно 3 записываем в 0 ячейку, а, то, что было в ячейке переносим в элемент с индексом 3:

[3, 2, 11, 1, 0, 0, 0, 0, 0, 0, 0]

Продолжая таким же образом:

[3, 4, 11, 1, 2, 0, 0, 0, 0, 0, 0]

[3, 4, 11, 5, 2, 1, 0, 0, 0, 0, 0]

[3, 4, 11, 5, 2, 6, 1, 0, 0, 0, 0]

[3, 4, 11, 5, 7, 6, 1, 2, 0, 0, 0]

[3, 8, 11, 5, 7, 6, 1, 2, 4, 0, 0]

[3, 8, 9, 5, 7, 6, 1, 2, 4, 11, 0]

[3, 8, 9, 5, 7, 6, 1, 10, 4, 11, 2]

В итоге получаем:

0	1	2	3	4	5	6	7	8	9	10
3	8	9	5	7	6	1	10	4	11	2

Теперь нужно вывести результат, на первой позиции студент, пришедший третьим, за третьим (next_stud [3]) стоит 5-ый, за пятым (next_stud [5]) – шестой и т.д.: 3 5 6 1 8 4 7 10 2 9.

Python:

```
n = int(input())
A = list(map(int, input().split()))
next_stud = [0] * (n + 1)
next_stud[0] = n + 1
for i in range(1, n + 1):
    next_stud[A[i - 1]], next_stud[i] = i, next_stud[A[i - 1]]
j = 0
while next_stud[j] < n + 1:
    j = next_stud[j]
    print(j, end = " ")
```

C++:

```
#include<bits/stdc++.h>

using namespace std;
int n, a;

int main(){
   cin >> n;
   vector<int> next(n+2);
   next[0] = n+1;
   next[n+1] = -1;
```

```
        for(int i = 1; i <= n; i++){
             cin >> a;
             next[i] = next[a];
             next[a] = i;
         }
       int y = 0;
       while(next[y] < n+1){
          y = next[y];
          cout<<y<<' ';
       }
      }
```

Общая временная сложность этого алгоритма составляет: O(n), это означает, что алгоритм эффективно работает с линейным временем в зависимости от числа студентов.

Задача 8. Проверка скобочной последовательности

Для заданной последовательности из скобок (,), [,], {,} определить является ли она правильной скобочной последовательностью. Определим правильную скобочную последовательность (ПСП) следующим образом:

– пустая строка является ПСП;

– если строка S является ПСП, то строки (S), [S] и {S} – ПСП;

– если строки S1 и S2 являются ПСП, то строка S1S2 так же ПСП.

Формат входных данных

В единственной строке находится последовательность из символов (,), [,], {,}. Длина этой последовательности не превосходит 10^5.

Формат выходных данных

Вывести «YES», если последовательность является правильной скобочной последовательностью и «NO» в противном случае.

Примеры

стандартный ввод	стандартный вывод
({()[({})[]]}({}[]))	YES

Решение:

Алгоритм проверки правильной скобочной последовательности использует структуру данных, называемую стеком, для управления открывающими и закрывающими скобками.

Инициализация стека:

Создаем пустой стек, который будет хранить открывающие скобки, которые мы встретили в последовательности.

Обход последовательности:

Перебираем каждый символ в строке, используя цикл:

Если текущий символ является одной из открывающих скобок: '(', '[', или '{', мы помещаем его в стек.

Если текущий символ является одной из закрывающих скобок: ')', ']', или '}', выполняем следующие действия. Сначала необходимо проверить, не пуст ли стек. Если стек пуст (то есть, у нас нет открывающих скобок, которые соответствуют текущей закрывающей скобке), это означает, что последовательность неправильная – возвращаем 'NO'.Если стек не пуст, мы берем верхний элемент стека, чтобы посмотреть, какая открывающая скобка находится на вершине. После этого мы проверяем, соответствует ли эта открывающая скобка текущей закрывающей скобке. Для этого мы комбинируем их в строку и проверяем, попадают ли они в множество {'()', '[]', '{}'}.

Если они соответствуют, это означает, что скобка закрыта корректно, и мы убираем верхний элемент стека. Если нет, возвращаем 'NO'.

Проверка на завершение:

После того как мы обработали все символы, нужно удостовериться, что стек пуст. Если стек пуст, это означает, что все открывающие скобки были корректно закрыты, и мы возвращаем 'YES'. Если стек не пуст, это значит, что некоторые открывающие скобки остались незакрытыми, и мы возвращаем 'NO'.

Алгоритм имеет линейную временную сложность, то есть O(n), где n — длина входной строки. Это связано с тем, что каждый символ анализируется только один раз, и операции добавления и удаления из стека имеют постоянную временную сложность O(1). Таким образом, общее время выполнения алгоритма пропорционально количеству символов в строке, что и определяет его эффективность.

C++:

```
#define sz(a) (int)a.size()
#define pb push_back
#define x first
#define y second
using namespace std;
typedef pair<int, int> pii;
string s;
stack<char> st;

int main(){
    cin >> s;
    for(int i = 0; i < s.length(); i++){
        if(s[i] == '(' || s[i] == '[' || s[i] == '{')      st.push(s[i]);
        else
        if(s[i] == ')'){
            if(st.size() == 0 || st.top() != '(') return cout <<"NO", 0;
            st.pop();
        }
        else
        if(s[i] == ']'){
            if(st.size() == 0 || st.top() != '[') return cout <<"NO", 0;
            st.pop();
        }
        else
        if(s[i] == '}'){
            if(st.size() == 0 || st.top() != '{') return cout <<"NO", 0;
            st.pop();
        }
    }
    if(st.size() > 0) return cout <<"NO", 0;
    cout<<"YES";
}
```

Python:

```
stack = []
# Проталкивание
def push(val):
    stack.append(val)
# Выталкивание
def pop():
    return stack.pop()
# Количество элементов в стеке
def size():
    return len(stack)
# Проверка стека на пустоту
def isempty():
    return len(stack) == 0
# Значение последнего элемента стека
def top():
    if not isempty():
        return stack[-1]
# Очистка стека
def clear():
    stack[:] = []
def bracket_sequence(s):
    step = 0
    for i in range(len(s)):
        step += 1
        if s[i] in '({[':
            push(s[i])
        elif isempty():
            return 'NO'
        elif top()+s[i] in {'()','[]','{}'}:
            pop()
        else:
            return 'NO'
    if isempty():
        return 'YES'
    else:
        return 'NO'
print(bracket_sequence(input()))
```

Список рекомендованной литературы и других источников для подготовки

Литературные источники

1. Аникеева Л. А. Лабораторные работы по Excel. – Салон-Пресс, 2022. – 112 с.

2. Долинский М.С. Решение сложных и олимпиадных задач по программированию. – СПб.: Питер, 2006. – 366 с.

3. Кирюхин В.М., Цветкова М.С. Информатика. Программы внеурочной деятельности учащихся по подготовке к Всероссийской олимпиаде школьников: 5–11 классы. – М.: БИНОМ. Лаборатория знаний, 2014. – 224 с.

4. Левитин А.В. Алгоритмы: введение в разработку и анализ. – М.: Издательский дом «Вильямс», 2006.

5. Род Стивенс Алгоритмы: теория и практическое применение. – М.: Издательство «Э», 2016.

6. Антти Лааксонен Олимпиадное программирование. – М.: ДМК Пресс, 2018.

7. Меньшиков, Ф. В. Олимпиадные задачи по программированию. – Москва: Питер, 2006. – 315 с.

8. Московские олимпиады по информатике / Под ред. Е.В. Андреевой, В. М. Гуровица и В. А. Матюхина – М.: МЦНМО, 2006. – 256 с

9. Волченков С.Г., Корнилов П.А., Белов Ю.А. и др. Ярославские олимпиады по информатике. Сборник задач с решениями. – М.: БИНОМ. Лаборатория знаний. 2010. – 405 с.

10. Просветов Г.И. Дискретная математика: задачи и решения: учебное пособие. – М.: БИНОМ. Лаборатория знаний. 2008. – 222 с

11. Задачи по программированию /С.М. Окулов, Т.В. Ашихмина, Н.А. Бушмелева и др.; Под ред. С.М. Окулова. – М.: БИНОМ. Лаборатория знаний, 2006. – 820 с.

Интернет-источники

1. Онлайн-курс «Введение в алгоритмы: реализация на языке C++» [Электронный ресурс] – URL: https://edu.sirius.online/#/course/2155

2. Онлайн-курс «Олимпиадное программирование. Базовый уровень» [Электронный ресурс] – URL: https://stepik.org/course/74336/

3. Тренировки по алгоритмам от Яндекса [Электронный ресурс] – URL: https://yandex.ru/yaintern/algorithm-training_2021

Составители: профессор каф. ИиКТ *Кризский В.Н.,* доцент каф. ИиКТ *Иванов И.В.,* доцент каф. ИиКТ Перязева *Ю.В.,* доцент каф. ИиКТ Иванченко Д.И., доцент каф. ИиКТ Ильн А.Е., ассистент каф. ИиКТ Коротаева А.Э.