Кировское областное государственное автономное образовательное учреждение дополнительного образования «Центр дополнительного образования одаренных школьников»

ИНФОРМАТИКА, 2016. ЧАСТЬ 2

ЗАДАНИЯ, РЕШЕНИЯ И

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по проведению
II (муниципального) этапа
всероссийской олимпиады школьников
по информатике

в Кировской области
в 2016/2017 учебном году

Часть II

Киров
2016

Печатается по решению методической комиссии III (регионального) этапа всероссийской олимпиады школьников по информатике в Кировской области

Задания, решения и методические указания по проведению II (муниципального) этапа всероссийской олимпиады школьников по информатике в Кировской области в 2016/2017 учебном году. Часть II / Сост. И.С. Кайсин, А.С. Латышев, О. А. Пестов // Под ред. О. А. Пестова. – Киров: Изд-во ЦДООШ, 2016. – 12 с.

Авторы, составители:
Кайсин Илья Сергеевич,
Латышев Алексей Сергеевич,
Пестов Олег Александрович

Научное рецензирование:
кандидат педагогических наук,
Иванов С. Ю.

Компьютерный набор О. Пестова
Подписано в печать 16.09.2016
Формат 60x84 1/16. Бумага типографская. Усл. печ. л. 0,85

Тираж 700 экз.

ОБЩИЕ ПРАВИЛА

1. Олимпиада проводится в компьютерной форме с использованием централизованной тестирующей системы, доступ в которую осуществляется со страницы http://o8v.github.io/43. Длительность олимпиады – 3 часа 30 минут.

2. Учащимся 7–8 классов будет предложено пять заданий: три теоретического характера и две задачи по программированию.

3. Учащимся 9–11 классов будет предложено четыре задачи по программированию.

4. Ответом на теоретическое задание является число, строка или текст. Некоторые из таких заданий предполагают, что участники могут использовать компьютер для получения ответа. Например, им может понадобиться текстовый редактор или приложение «Калькулятор». Также разрешается написание вспомогательных программ.

5. Ответы на теоретические задания отправляются в тестирующую систему. Проверка теоретических заданий осуществляется *во время тура* и участнику сообщается результат проверки. Сдавать ответ можно только *один раз*.

6. Решением задачи по программированию является программа на одном из допустимых языков, которые поддерживаются тестирующей системой. Это Pascal, C/C++, Python, Java, C#. Программа должна быть консольным приложением, не использующим какие-либо графические возможности (диалоговые окна, формы ввода, средства рисования и т.д.). Программа должна читать данные со стандартного ввода (клавиатуры) и выводить результат на стандартный вывод (экран).

7. Решения задач по программированию сдаются в тестирующую систему. Проверка решений выполняется *во время тура* и участнику сообщается результат проверки. Сдавать решения можно *несколько раз*. Количество посылок не влияет на итоговый результат. Из всех решений в зачёт идёт набравшее наибольшее число баллов.

8. Максимальный балл за выполнение одного задания равен ста. Для этого программа должна вывести правильный ответ на всех заранее подготовленных тестах. Если задание теоретическое, то ответ должен быть полностью верным. Иначе участник получает частичный балл в зависимости от пройденных тестов или ответа.

9. Разрешается использовать литературу: книги, документацию, личные записи. Запрещается использовать ресурсы сети интернет (кроме сайтов олимпиады и тестирующей системы), средства связи, электронное оборудование (кроме рабочего компьютера) и носители информации.

10. Все вопросы по условиям задач участники задают через тестирующую систему (ссылка «Отправить вопрос»). Ответ можно увидеть в тестирующей системе на странице «Сообщения».

11. Материалы олимпиады (решения жюри, тесты) будут доступны на странице олимпиады вечером в день проведения. Там же можно будет ознакомиться с полными протоколами проверки решений, используя логин и пароль, которые рекомендуется сохранить. В случае несогласия с результатом, участники имеют право подать обоснованную апелляцию.

ИНСТРУКЦИЯ ДЛЯ ЖЮРИ

1. Продолжительность основного тура — 3 часа 30 минут. Рекомендуемое время начала олимпиады – 10 часов утра.

2. До олимпиады необходимо подготовить рабочие места участников. На компьютере должны быть установлены такие среды разработки, чтобы участник мог писать программы, используя следующие языки программирования: Pascal (PascalABC.Net, Free Pascal), C/C++ (Code::Blocks, Microsoft Visual C++ Express Edition), C# (MonoDevelop, Xamarin Studio, Microsoft Visual C# Express Edition), Java (JDK, Eclipse), Python (Python 2, Python 3, WingIDE 101, PyCharm). Не обязательно устанавливать все программы – достаточно только необходимые участникам.

3. Все компьютеры участников должны быть подключены к интернету. Во время тура необходимо контролировать работу учащихся, чтобы не допустить использование интернет-ресурсов помимо страницы олимпиады и тестирующей системы.

4. Перед проведением олимпиады, членам жюри рекомендуется принять участие в пробном туре, который будет доступен на странице олимпиады http://o8v.github.io/43 с 25 октября по 4 ноября 2016 года. Проверка решений автоматической системой накладывает определённые требования на оформление программ. Эти требования нужно знать, чтобы донести их до участников. В частности, необходимо объявить участникам, что они:

– не должны «разговаривать» с компьютером. Нельзя выводить приглашения перед чтением вида writeln('Введите число:') или возможные «обрамления» ответа вида writeln('Ответ = '). Необходимо строго следовать формату входных и выходных данных;

– не должны проверять вводимые данные на корректность. Если написано, например, что число положительное и не больше ста, то гарантируется, что это так.

5. Перед началом олимпиады жюри получает условия задач и список учётных записей для входа в проверяющую систему. Каждому участнику олимпиады жюри назначает индивидуальный логин и пароль из полученного списка.

6. До начала тура участникам предоставляется возможность проверить работоспособность компьютера, сред разработки, логина в тестирующую систему.

7. Тур начинается после того, как устранены все технические проблемы. Началом тура является выдача условий участникам. Время начала тура записывается на доске.

8. В любых нештатных ситуациях, когда нет возможности сдать задания в тестирующую систему (например, пропало подключение к интернету), необходимо организовать сбор решений участников, архив решений сохраняется у организаторов и передаётся в предметно-методическую комиссию.

9. При наличии проблем в работе тестирующей системы тур не продлевается, учащиеся продолжают решать задачи. Порядок действий аналогичен предыдущему пункту.

10. Несмотря на то, что для проверки олимпиады используется автоматическая тестирующая система, поддерживаемая предметно-методической комиссией, результаты олимпиады подводит жюри муниципального этапа. Жюри олимпиады должно после окончания тура для каждого участника записать его результат из тестирующей системы и составить протокол проведения олимпиады.

11. Победители и призёры муниципального этапа олимпиады определяются отдельно по классам.

12. В случае несогласия с результатом, участники имеют право подать обоснованную апелляцию. Такие ситуации решаются в индивидуальном порядке. Для их разрешения необходимо связаться с предметно-методической комиссией.

Контактные данные для связи с комиссией: Пестов Олег Александрович oleg.pestov@gmail.com, +7(909)143-57-74.

КОМПЛЕКТ ЗАДАНИЙ ДЛЯ 7-8 КЛАССОВ

Ограничения на время работы решений на одном тесте, а также ограничения на объём используемой памяти в задачах на программирование указаны на соответствующих вкладках в тестирующей системе.

Задание 1. Палиндром

Вам даны пять чисел:
1890
83849
129921
7821239
32749193712

Для каждого из них найдите *минимальное* целое число, которое больше данного и является палиндромом (читается одинаково как слева направо, так и справа налево). Например, для числа 898 такой минимальный палиндром это 909.

В ответ нужно записать пять целых чисел. Каждое число должно быть в отдельной строке. Порядок записи менять нельзя. Если вы не можете найти ответ для какого-то числа, то вместо ответа запишите знак «-» (минус без кавычек). Все числа равнозначны и оцениваются в 20 баллов.

Задание 2. Калькулятор

Имеется калькулятор, который выполняет три операции: умножить число на два, умножить число на три и прибавить к числу единицу. Вам даны пять чисел:

13
23
28
40
47

Для каждого из них найдите *минимальное* количество операций, необходимое чтобы получить на калькуляторе данное число из единицы. Например, число 18 можно получить минимум за три операции: умножить на три, умножить на три и умножить на два.

В ответ нужно записать пять целых чисел. Каждое число должно быть в отдельной строке. Порядок записи менять нельзя. Если вы не можете найти ответ для какого-то числа, то вместо ответа запишите знак «-» (минус без кавычек). Все числа равнозначны и оцениваются в 20 баллов.

Задание 3. Последовательность Туэ – Морса

Рассмотрим последовательность, которая начинается с нуля и строится по такому правилу: каждый раз она удваивается, то есть записывается после самой себя, но при этом во второй половине нули заменяются на единицы, а единицы на нули. Получается 0 → 0**1** → 01**10** → 0110**1001** → 01101001**10010110** … Выполним достаточно большое количество «удвоений».

Вам даны пять чисел:

10
152
4090
2128506
998877665544332211

Для каждого из них найдите три цифры, которые идут в последовательности начиная с данной позиции. Например, для числа четыре ответом будет 010, так как на

четвёртой позиции находится ноль, на пятой — единица и на шестой — ноль.

В ответ нужно записать пять последовательностей длины три, состоящих из нулей и единиц. Каждая последовательность должна быть в отдельной строке. Порядок записи менять нельзя. Если вы не можете найти ответ для какого-то числа, то вместо ответа запишите знак «-» (минус без кавычек). Все числа равнозначны и оцениваются в 20 баллов.

Задание 4. Монеты

У Ани есть N монет. Каждая может быть достоинством один, два, пять или десять рублей. Аня знает количество монет, но не знает какие это монеты. Ей интересно сколько рублей у неё может быть. Например, если $N = 2$, то существует десять вариантов: 2, 3, 4, 6, 7, 10, 11, 12, 15, 20.

Напишите программу, которая для заданного числа N определит количество различных сумм, которые могут быть у Ани.

Формат входных данных

В первой строке записано целое число N ($1 \le N \le 25$).

Формат результата

Выведите количество различных сумм.

Примеры

стандартный ввод	стандартный вывод
1	4
2	10

Система оценки

В этой задаче 20 тестов. Тесты оцениваются независимо по пять баллов за тест.

Задание 5. ROT13

Алгоритм ROT13 (англ. rotate; «сдвинуть на 13 позиций») является примером шифра простой заменой. Когда он применяется к строке, то в ней каждая английская буква заменяется на соответствующую со сдвигом на 13 позиций в алфавите. В английском алфавите 26 букв «**a**bcdefghijklm**n**opqrstuvwxyz» и значит a становится n, b становится o и т. д до m, которая становится z, а затем применяются буквы из начала алфавита: n становится a, o становится b и так до z, которая становится m. При этом заменяются только буквы, а другие символы остаются без изменений.

Напишите программу, которая применяет ROT13 к заданной строке.

Формат входных данных

В первой строке записано не более ста символов. Каждый символ это строчная английская буква или цифра или пробел или один из знаков препинания ,.?!-.

Формат результата

Выведите зашифрованный текст.

Примеры

стандартный ввод	стандартный вывод
terra	green
rotate by 13 places!	ebgngr ol 13 cynprf!

Система оценки

Тесты к этой задаче состоят из двух групп. Тесты в первой группе оцениваются независимо по три балла за тест. Тесты во второй группе оцениваются независимо по четыре балла за тест.

Группа	*Тесты*	*Баллы*	*Комментарий*
1	1-20	60	Только буквы
2	21-30	40	Буквы, пробелы и знаки препинания

КОМПЛЕКТ ЗАДАНИЙ ДЛЯ 9-11 КЛАССОВ

Ограничения на время работы решений на одном тесте, а также ограничения на объём используемой памяти указаны на вкладках задач в тестирующей системе.

Задача 1. Монеты

У Ани есть N монет. Каждая может быть достоинством один, два, пять или десять рублей. Аня знает количество монет, но не знает какие это монеты. Ей интересно сколько рублей у неё может быть. Например, если $N = 2$, то существует десять вариантов: 2, 3, 4, 6, 7, 10, 11, 12, 15, 20.

Напишите программу, которая для заданного числа N определит количество различных сумм, которые могут быть у Ани.

Формат входных данных

В первой строке записано целое число N ($1 \le N \le 25$).

Формат результата

Выведите количество различных сумм.

Примеры

стандартный ввод	стандартный вывод
1	4
2	10

Система оценки

В этой задаче 20 тестов. Тесты оцениваются независимо по пять баллов за тест.

Задача 2. ROT13

Алгоритм ROT13 (англ. rotate; «сдвинуть на 13 позиций») является примером шифра простой заменой. Когда он применяется к строке, то в ней каждая английская буква заменяется на соответствующую со сдвигом на 13 позиций в алфавите. В английском алфавите 26 букв «**a**bcdefghijklm**n**opqrstuvwxyz» и значит a становится n, b становится o и т.д до m, которая становится z, а затем применяются буквы из начала алфавита: n становится a, o становится b и так до z, которая становится m. При этом заменяются только буквы, а другие символы остаются без изменений.

Напишите программу, которая применяет ROT13 к заданной строке.

Формат входных данных

В первой строке записано не более ста символов. Каждый символ это строчная английская буква или цифра или пробел или один из знаков препинания ,.?!-.

Формат результата

Выведите зашифрованный текст.

Примеры

стандартный ввод	стандартный вывод
terra	green
rotate by 13 places!	ebgngr ol 13 cynprf!

Система оценки

Тесты к этой задаче состоят из двух групп. Тесты в первой группе оцениваются независимо по три балла за тест. Тесты во второй группе оцениваются независимо по четыре балла за тест.

Группа	*Тесты*	*Баллы*	*Комментарий*
1	1-20	60	Только буквы
2	21-30	40	Буквы, пробелы и знаки препинания

Задача 3. Последовательность Морса — Туэ

Рассмотрим последовательность, которая начинается с нуля и строится по такому правилу: каждый раз она удваивается, то есть записывается после самой себя, но при этом во второй половине нули заменяются на единицы, а единицы на нули. Получается 0 → 0**1** → 01**10** → 0110**1001** → 01101001**10010110** … Выполним достаточно большое количества «удвоений».

Напишите программу, которая находит цифру на позиции N.

Формат входных данных

В первой строке записано целое число N ($1 \le N \le 10^9$).

Формат результата

Выведите цифру 0 или 1.

Примеры

стандартный ввод	стандартный вывод
4	0
12	1

Система оценки

Тесты к этой задаче состоят из четырёх групп. Баллы за группы ставятся только при прохождении всех тестов группы.

Группа	*Тесты*	*Баллы*	*Комментарий*
1	1-10	10	$N \le 32$
2	11-20	30	$N \le 1000$
3	21-30	30	$N \le 10^6$
4	31-40	30	$N \le 10^9$

Задача 4. Сортировка перестановки

Рассмотрим последовательность чисел длины N, в которой встречаются ровно один раз все числа от одного до N. Такая последовательность называется перестановкой. Примеры перестановок для $N = 4$ это 1 2 3 4, 2 4 1 3, 2 1 3 4 и т. д.

Дана перестановка. За один шаг разрешается поменять местами два любых элемента. Требуется узнать минимальное количество операций обмена, необходимых, чтобы упорядочить перестановку по неубыванию. Например, чтобы упорядочить последовательность 4 1 5 2 3 и получить 1 2 3 4 5, необходимо выполнить минимум три обмена.

Формат входных данных

В первой строке задаётся целое число N ($1 \le N \le 100000$). Во второй строке через пробел записаны N целых чисел. Гарантируется, что данная последовательность является перестановкой.

Формат результата

Выведите минимальное количество обменов.

Примеры

стандартный ввод	стандартный вывод
3 1 2 3	0
4 4 3 2 1	2
4 4 1 2 3	3

Система оценки

Тесты к этой задаче состоят из трёх групп. Тесты в первой группе оцениваются независимо по четыре балла за тест. Тесты во второй группе оцениваются независимо по два балла за тест. Тесты в третьей группе оцениваются независимо по два балла за тест.

Группа	*Тесты*	*Баллы*	*Комментарий*
1	1-5	20	$N \le 10$
2	6-25	40	$N \le 1000$
3	26-45	40	$N \le 100000$

УКАЗАНИЯ ПО РЕШЕНИЮ ЗАДАЧ

Палиндром

Поскольку требуется найти минимальный больший палиндром, то необходимо изменить цифру, которая находится как можно ближе к концу числа. Соответственно вариант решения — это перебрать все цифры и для каждой из них проверить, можно ли получить больший палиндром, если слева от этой цифры все останется как есть, а она увеличится на один, два, три и т. д. Если можно, то выбирается вариант, где увеличение минимально. А из всех подходящих цифр выбирается самая правая.

Например, в числе 784321 можно увеличить семь на один и получить 800008, восемь на один и получить 790097, четыре на один и получить 785587 или три на один и получить 784487. Поскольку тройка самая правая из подходящих вариантов, то лучшим является именно данный способ.

Калькулятор

Один из способов решения — найти ответ для небольших чисел (например, меньших 24), перебирая какие числа могут быть получены за одно, два, три и т. д. нажатий. За ноль нажатий можно получить 1, за одно нажатие можно получить 2 и 3. За два нажатия — 4, 6, 9. Переход здесь выполняется так: чтобы узнать числа, которые можно получить за K+1 нажатие, рассмотрим все числа, которые можно получить за K нажатий и к каждому из них применим одну из трёх заданных операций. Если получается новое число, то значит для него найден ответ. Так за три нажатия получаются числа 5, 8, 7, 12, 18 и 10 (для ускорения числа больше 23 игнорируются). За четыре — 15, 16, 14, 21, 13, 19, 11, 20. За пять — 17, 22. За шесть — 23.

Для больших чисел нужно рассмотреть процесс с конца. Например, чтобы получить 40 можно последним шагом умножить 20 на 2. Так, как для 20 ответ 4, то всего понадобится пять нажатий. Другой вариант это к 39 прибавить один, а 39 это 13 умножить на три. Для 13 ответ четыре и этот способ требует шесть шагов и он хуже, чем альтернатива.

Последовательность Туэ – Морса

Для первых трёх чисел ответ можно найти если построить последовательность вручную в текстовом редакторе. Для этого нужно иметь две строки — искомую последовательность и её копию, где нули заменены на единицы, а единицы на нули. Например, если уже получены строки 01101001 и 10010110, то чтобы их удвоить, необходимо к первой дописать вторую, а ко второй первую. Это делается копированием и вставкой.

Для четвёртого числа также можно построить последовательность. Но вручную манипулировать строками длины миллион символов трудно, поэтому легче написать вспомогательную программу.

Последнее число слишком большое, чтобы можно было построить последовательность и здесь необходимо использовать свойства последовательности. А именно то, что на каждом шаге последовательность удваивается и вторая половина это копия первой с некоторой модификацией. Это значит, что если ищется символ на позиции K, в строке длины N, то он находится либо в первой половине и тогда можно отбросить вторую половину и искать символ на позиции K в строке длины $N/2$, либо он находится во второй половине. В последнем случае, искомый символ это символ противоположный соответствующему символу в первой половине. То есть если найти элемент на позиции K-$N/2$ в строке $N/2$, то ответ это противоположный ему.

Таким образом, на каждом шаге в два раза уменьшается длина рассматриваемой строки и очень быстро она станет равна одному, а для этой строки ответ это ноль. И останется учесть изменения элемента на противоположный.

Пример: найти элемент на позиции 27. Чтобы получить этот символ, необходимо выполнить пять удвоений и получить последовательность длины 32. Значит, в строке

длины 32 требуется найти 27-й символ. Он находится во второй половине и является противоположностью 11-го (27-16) элемента в последовательности длины 16. Который в свою очередь находится во второй половине и является противоположностью третьего (11-8) элемента в последовательности длины 8. Он находится в первой половине и поэтому вторая половина может быть отброшена, чтобы найти третий элемент в последовательности длины четыре. Опять вторая половина и искомый элемент это противоположность первого элемента в последовательности длины два. Первый элемент это ноль. Теперь в процессе произошло три раза взятие противоположного элемента и значит 27-й элемент это один.

Следствием из этого алгоритма является то, что для позиции *K* ответ это чётность числа единиц в битовом представлении числа *K*-1. Ответ для всех пяти чисел может быть получен с помощью калькулятора.

Монеты

Покажем, что за исключением двух вариантов, если какую-то сумму *S* можно набрать с помощью *K* монет, то её можно набрать с помощью любого количества монет от *K* до *S*. Тогда, например, из того, что 56 можно набрать с помощью семи монет (10, 10, 10, 10, 10, 5, 1) следует, что 56 можно набрать с помощью восьми, девяти, десяти, ..., 56-ти монет.

Рассмотрим способ увеличения количества монет в наборе, применяя который можно последовательно довести это количество до необходимого:

- если в наборе есть десятка, то заменим её на две пятёрки;
- в наборе нет десяток, но есть двойка — заменим её на две единицы;
- в наборе нет десяток и двоек, но есть пятёрки и единицы — заменим пять и один на три двойки;
- в наборе только единицы. В этом случае получен минимальный возможный набор и нет необходимости его расширять;
- в наборе только пятёрки. Если можно увеличить набор на две монеты, то заменим пятёрку на две двойки и единицу. Иначе, если в наборе есть хотя бы три пятёрки, то заменим три пятёрки на десятку, две двойки и единицу.

Последнее условие говорит, что расширение не работает если необходимо получить пять с помощью двух монет или десять с помощью трёх (тогда в наборе только пятёрки, размер набора необходимо увеличить на один, но пятёрок для этого мало). Эти случаи необходимо рассмотреть отдельно.

А так решение получается простым. Пусть у Ани есть десять монет. Минимальная сумма, которую можно набрать это десять, а максимальная — сто. В диапазоне от десяти до 80 включительно можно набрать любую сумму «жадным» способом: пока получается добавлять в набор десятки, потом пятёрки, потом двойки и в конце единицы. В наборе может быть меньше десяти монет, но его можно расширить до десяти, применяя алгоритм, описанный выше.

В диапазоне от 81 до 90 нельзя набрать 88 и 89, так как это требует минимум одиннадцать монет. В диапазоне от до 91 до 100 нельзя набрать 93, 94, 96, 97, 98, 99 так, как это требует одиннадцать или двенадцать монет. Всего нельзя набрать восемь сумм и значит можно набрать 91-8=83. В общем случае ответ это 10*N*-*N*+1-8 или, что тоже самое, 9*N* – 7. Он работает при *N* > 3. При *N*=3 необходимо учесть, что нельзя получить десять (то есть ответ 9*3-7-1=19), а при *N*=2 нельзя получить 5 (ответ 9*2-7-1=10). Случай, когда только одна монета, разбирается отдельно.

```
n = int(input())
if n == 1:
  print(4)
elif n < 4:
  print(9 * n - 8)
else:
  print(9 * n - 7)
```

ROT13

Техническая задача на умение работать с символами и строками. Один из способов решения — создать константные строки, которые соответствуют алфавиту и преобразованию. Далее для каждой буквы необходимо найти её в первой строке. Это порядковый номер в алфавите и по этому номеру во второй строке находится результат преобразования.

```
a1 = "abcdefghijklmnopqrstuvwxyz"
a2 = "nopqrstuvwxyzabcdefghijklm"

s = input()
for c in s:
  if c in a1:
    i = a1.find(c)
    print(a2[i], end="")
  else:
    print(c, end="")
```

Последовательность Морса — Туэ

Это задача «Последовательность Туэ — Морса», но с одним отличием, которое состоит в том, что алгоритм необходимо реализовать в виде программы, а не применить вручную для нескольких частных случаев.

```
n = int(input())
p = 1
while p < n:
  p  = 2 * p
i = 0
while p > 1:
  q = p // 2
  if n > q:
    i = 1 - i
    n -= q
  p = q
print(i)
```

Сортировка перестановки

Правильное решение — сделать так, чтобы после каждого обмена какое-то число встало на своё место. Например, найти единицу и поменять её с тем числом, которое стоит на первом месте. Потом повторить то же самое с двойкой, тройкой и всеми оставшимися числами.

Если дословно реализовать этот алгоритм, то он будет правильный, но медленный. Правильный и быстрый вариант — это, например, взять число, которое стоит на первом месте и поставить его на своё. После этого обмена на первом месте окажется новое число. Если оно не единица, то операцию нужно повторить и так, пока на первом месте не окажется единица. Потом повторить со вторым местом, с третьим местом и т. д. до последнего.

```
n = int(input())
a = [int(x) - 1 for x in input().split()]
k = 0
for i in range(n):
  while a[i] != i:
    j = a[i]
    a[i] = a[j]
    a[j] = j
    k += 1
print(k)
```

Почему эта стратегия работает за минимальное количество обменов? Для доказательства необходимо рассмотреть перестановку как набор циклов. Что такое цикл легко понять на примере перестановки 6 3 2 4 7 5 1. Встанем на первую позицию, там стоит шесть, поэтому пойдём на шестую позицию, там стоит пять, поэтому пойдём на пятую позиция, там стоит семь, поэтому пойдём на седьмую позицию, там стоит один поэтому пойдём на первую позицию. Это цикл: $1 \to 6 \to 5 \to 7 \to 1$. Другие циклы в этой перестановке это $2 \to 3 \to 2$ и $4 \to 4$. За одну операцию обмена можно увеличить количество циклов на один. Для этого необходимо поменять два элемента из одного цикла. Если поменять элементы из разных циклов, то количество циклов уменьшиться. В отсортированной перестановке N циклов и значит минимальное количество обменов это N минус количество циклов в исходной перестановке. Или, как в примере программы выше, минимальное количество обменов можно получить, если производить эти обмены, постоянно меняя элементы из одного цикла.