БРОШЮРА:
«Тренажер компьютерного ЕГЭ»

Автор: Передерев Илья, 11 класс, МБОУ СОШ № 3 г. Салехард

В данной брошюре представлены:

• 13 заданий из демоверсии КИМ ЕГЭ по информатике и ИКТ 2021 года;

• проверяемые знания к каждому заданию;

• элементы содержания, проверяемые на ЕГЭ, по кодификатору;

• решения к заданиям на разных языках программирования;

• пояснения к заданиям;

• ответы ко всем заданиям.

Выполнение данных заданий предоставляет обучающимся возможность самостоятельно подготовиться к государственной итоговой аттестации в форме ЕГЭ, а также объективно оценить уровень своей подготовки к экзамену.

Учителя могут использовать задания для организации контроля результатов освоения школьниками образовательных программ среднего общего образования и интенсивной подготовки обучающихся к ЕГЭ.

Задание 2

(базовый уровень, время – 3 мин)

Тема: Анализ таблиц истинности логических выражений.

Что проверяется:

Умение строить таблицы истинности и логические схемы.

1.5.1. Высказывания, логические операции, кванторы, истинность высказывания

1.1.6. Умение строить модели объектов, систем и процессов в виде таблицы истинности для логического высказывания

Пример.

Миша заполнял таблицу истинности функции (x ⋁ y) ⋀ ¬(y≡ z) ⋀ ¬w, но успел заполнить лишь фрагмент из трёх различных её строк, даже не указав, какому столбцу таблицы соответствует каждая из переменных w, x, y, z.

				$(x \vee y) \wedge \neg(y \equiv z) \wedge \neg w$
				1
				1
				1

Определите, какому столбцу таблицы соответствует каждая из переменных *w*, *x*, *y*, *z*.

В ответе напишите буквы *w*, *x*, *y*, *z* в том порядке, в котором идут соответствующие им столбцы (сначала буква, соответствующая первому столбцу; затем буква, соответствующая второму столбцу, и т.д.). Буквы в ответе пишите подряд, никаких разделителей между буквами ставить не нужно.

Пример. Функция задана выражением ¬*x* ⋁ *y*, зависящим от двух переменных, а фрагмент таблицы имеет следующий вид.

		$\neg x \vee y$
		0

В этом случае первому столбцу соответствует переменная *y*, а второму столбцу – переменная *x*. В ответе следует написать *yx*.

Решение: (на языке Python)

```
print ('x y z w')
for x in [False,True]:
    for y in [False,True]:
        for z in [False,True]:
            for w in [False,True]:
                if (x or y) and not (y==z) and not w:
                    print (int(x), int(y), int(z), int(w))
```

Ответ: zyxw

Пояснение к программе: данная программа перебирает все значения логических переменных x, y, z и w, и подставляет их логическую функцию, определённую в условии задания. Когда логическая функция выдаёт нам нужный «вердикт» (True или False), программа печатает числовые значения этих переменных (0 или 1). Далее следует анализ полученного ответа при выполнении программы, из чего потом можно вывести окончательный ответ.

Задание 12

(повышенный уровень, время – 6 мин)

Тема: Выполнение алгоритмов для исполнителя.

Что проверяется:

Умение анализировать результат исполнения алгоритма.

1.6.2. Вычислимость. Эквивалентность алгоритмических моделей.

1.1.3. Умение строить информационные модели объектов, систем и процессов в виде алгоритмов.

Пример.

Исполнитель Редактор получает на вход строку цифр и преобразовывает её. Редактор может выполнять две команды, в обеих командах v и w обозначают цепочки цифр.

А) заменить (v, w).

Эта команда заменяет в строке первое слева вхождение цепочки v на цепочку w. Например, выполнение команды

заменить (111, 27)

преобразует строку 05111150 в строку 0527150.

Если в строке нет вхождений цепочки v, то выполнение команды

заменить (v, w) не меняет эту строку. Б) нашлось (v).

Эта команда проверяет, встречается ли цепочка v в строке исполнителя Редактор. Если она встречается, то команда возвращает логическое значение

«истина», в противном случае возвращает значение «ложь». Строка исполнителя при этом не изменяется.

Цикл

ПОКА условие

последовательность команд

КОНЕЦ ПОКА

выполняется, пока условие истинно. В конструкции

ЕСЛИ условие

ТО команда1

ИНАЧЕ команда2

КОНЕЦ ЕСЛИ

выполняется команда1 (если условие истинно) или команда2 (если условие ложно).

Какая строка получится в результате применения приведённой ниже программы к строке, состоящей из 70 идущих подряд цифр 8? В ответе запишите полученную строку.

НАЧАЛО
ПОКА нашлось (2222) ИЛИ нашлось (8888)
ЕСЛИ нашлось (2222)
ТО заменить (2222, 88)
ИНАЧЕ заменить (8888, 22)
КОНЕЦ ЕСЛИ КОНЕЦ ПОКА КОНЕЦ

Решение: (на языке Python)

```
s=70*'8'
while ('2222' in s) or ('8888' in s):
    if '2222' in s:
        s=s.replace('2222','88',1)
    else:
        s=s.replace('8888','22',1)  print(s)
```

Ответ: 22

Пояснение к программе: Данная программа получает на вход определённую строку, которая дана нам в условии задания. Далее программа начинает обрабатывать эту строку в соответствии условием, т.е. она берёт подстроку и меняет её на другую подстроку.

Задание 14

(повышенный уровень, время – 5 мин)

Тема: Позиционные системы счисления.

Что проверяется:

Знание позиционных систем счисления.

1.4.1. Позиционные системы счисления.

1.1.3. Умение строить информационные модели объектов, систем и процессов в виде алгоритмов.

Пример.

Значение арифметического выражения: $49^7 + 7^{21} - 7$ – записали в системе счисления с основанием 7. Сколько цифр 6 содержится в этой записи?

Решение: (на языке Python)

```
n=49**7 + 7**21 - 7
count6 = 0
while n>0:
    if n%7==6:
        count6+=1
    n//=7
print (count6)
```

Ответ: 13

Пояснение к программе: Данная программа получает какое-то определённое число на вход (смотреть в условии). Далее происходит перевод числа в систему счисления, которую так же можно найти в условии задания, в нашем случае это 7-ричная система счисления. Далее мы делаем следующее: мы берём число, заводим счётчик и смотрим остаток от деления на основание системы счисления, в которую переводим число, и если остаток совпадает с нужным нам числом, то добавляем к счётчику 1. Далее мы делим число на основание системы счисления. Повторяем процедуру до тех пор, пока число больше 0, затем пишем ответ.

Задание 16

(повышенный уровень, время – 9 мин)

Тема: Рекурсия. Рекурсивные процедуры и функции

Что проверяется:

Вычисление рекуррентных выражений

1.5.3. Индуктивное определение объектов.

1.1.3. Умение строить информационные модели объектов, систем и процессов в виде алгоритмов.

Пример.

Алгоритм вычисления значения функции F(n), где n – натуральное число, задан следующими соотношениями:

F(n) = 1 при n = 1;

F(n) = n + F(n − 1), если n – чётно,

F(n) = 2 × F(n − 2), если n > 1 и при этом n – нечётно.

Чему равно значение функции F(26)?

Решение: (на языке Python)

```
def f(x):
    if x==1:
        return 1
    elif x%2==0:
        return x+f(x-1)
    elif x>1 and x%2==1:
        return 2*f(x-2)
 print(f(26))
```

//

(на языке Pascal)

```
function f(x:integer):integer;
begin
  if x=1 then f:=1;
  if (x>1) and (x mod 2 = 1) then f:=2*f(x-2);
```

```
  if (x>1) and (x mod 2 = 0) then f:=x + f(x-1)
end;
 Begin
  writeln(f(26))
End.
```

Ответ: 4122

Пояснение к программе: Данная программа получает функцию от какого-то аргумента (F(x)), далее начинаются рекурсивные преобразования, указанные в условии задания. Всё это обрабатывается, и мы получаем ответ.

Задание 17

(повышенный уровень, время – 15 мин)

Тема: Перебор целых чисел на заданном отрезке. Проверка делимости

Что проверяется:

Умение создавать собственные программы (20–40 строк) для обработки целочисленной информации.

1.7.2. Основные конструкции языка программирования. Система программирования.

1.1.5. Умение создавать программы на языке программирования по их описанию.

Пример.

Рассматривается множество целых чисел, принадлежащих числовому отрезку [1016; 7937], которые делятся на 3 и не делятся на 7, 17, 19, 27.

Найдите количество таких чисел и максимальное из них.

В ответе запишите два целых числа: сначала количество, затем максимальное число.

Для выполнения этого задания можно написать программу или воспользоваться редактором электронных таблиц.

Решение: (на языке Python)

```
cnt=0
maxx=0
for x in range(1016,7937+1):
    if x%3==0 and x%7!=0 and x%17!=0 and x%19!=0 and x%27!=0:
        cnt+=1
        if x>maxx:
            maxx = x
 print(cnt,maxx)
```

//

(на языке Pascal)

```
Begin
```

```
var count := 0;
var max := -MaxInt;
for var x := 1016 to 7937 do begin
  if (x mod 3 = 0) and (x mod 7 <> 0) and (x mod 17 <> 0) and (x mod 19 <> 0) and (x mod 27 <> 0) then begin
    count += 1;
    if x > max then
      max := x;
  end;
end;
Print(count,max);
End.
```

Ответ: 1568 7935

Пояснение к программе: Данная программа получает диапазон чисел на вход, данных в условии. Заводим счётчик чисел, подходящих нам. Далее происходит проверка числа на делимость на числа, данные в условии. Если все условия выполняются, то мы увеличиваем счётчик. Далее происходит проверка на максимальность, т.е. больше ли число действующего максимума или нет. Если число больше действующего максимума, то максимум становится равен этому числу. Далее проходим по всему диапазону и получаем ответ.

Задания 19-21

(повышенный уровень, время – 6 + 6 + 10 мин)

Тема: Теория игр. Поиск выигрышной стратегии.

Что проверяется:

Умение анализировать алгоритм логической игры. Умение найти выигрышную стратегию игры. Умение построить дерево игры по заданному алгоритму и найти выигрышную стратегию.

1.5.2. Цепочки (конечные последовательности), деревья, списки, графы, матрицы (массивы), псевдослучайные последовательности.

1.1.3. Умение строить информационные модели объектов, систем и процессов в виде алгоритмов.

Пример

Задание 19

Два игрока, Петя и Ваня, играют в следующую игру. Перед игроками лежат две кучи камней. Игроки ходят по очереди, первый ход делает Петя. За один ход игрок может добавить в одну из куч (по своему выбору) один камень или увеличить количество камней в куче в два раза. Например, пусть в одной куче 10 камней, а в другой 5 камней; такую позицию в игре будем

обозначать (10, 5). Тогда за один ход можно получить любую из четырёх позиций: (11, 5), (20, 5), (10, 6), (10, 10). Для того чтобы делать ходы, у каждого игрока есть неограниченное количество камней.

Игра завершается в тот момент, когда суммарное количество камней в кучах становится не менее 77. Победителем считается игрок, сделавший последний ход, т.е. первым получивший такую позицию, при которой в кучах будет 77 или больше камней.

В начальный момент в первой куче было семь камней, во второй куче – S камней; $1 \leqslant S \leqslant 69$.

Будем говорить, что игрок имеет *выигрышную стратегию*, если он может выиграть при любых ходах противника. Описать стратегию игрока – значит описать, какой ход он должен сделать в любой ситуации, которая ему может встретиться при различной игре противника. В описание выигрышной стратегии не следует включать ходы играющего по этой стратегии игрока, не являющиеся для него безусловно выигрышными, т.е. не являющиеся выигрышными независимо от игры противника.

Известно, что Ваня выиграл своим первым ходом после неудачного первого хода Пети. Укажите минимальное значение S, когда такая ситуация возможна.

Задание 20

Для игры, описанной в предыдущем задании, найдите два таких значения S, при которых у Пети есть выигрышная стратегия, причём одновременно выполняются два условия:

- Петя не может выиграть за один ход;
- Петя может выиграть своим вторым ходом независимо от того, как будет ходить Ваня.

Найденные значения запишите в ответе в порядке возрастания.

Задание 21

Для игры, описанной в задании 19, найдите минимальное значение S, при котором одновременно выполняются два условия:

- у Вани есть выигрышная стратегия, позволяющая ему выиграть первым или вторым ходом при любой игре Пети;
- у Вани нет стратегии, которая позволит ему гарантированно выиграть первым ходом.

Решение (ответ сразу на 3 задания, одна программа): (на языке Python)

```
from functools import lru_cache
 def moves(heap):
    a,b = heap
    return (a+1,b),(a*2,b),(a,b+1),(a,b*2)
 @lru_cache(None)
def game(heap):
    if sum(heap)>=77: return 0
    else:
```

```
        steps = [game(x) for x in moves(heap)]
        if any(x%2==0 for x in steps):
            return min(x for x in steps if x%2==0)+1
        else:
            return max(steps)+1
for s in range(69,0,-1):
    print(s,':',game((7,s)),':',[game(x) for x in moves((7,s))])
```

Ответ: 19 задание: 18;
20 задание: 31 34;
21 задание: 30

Пояснение к программе: Для начала мы читаем условия игры, в которую играют игроки. Далее мы смотрим на то, какие значения s может принимать и какие ходы вообще существуют. В данной программе мы работаем с двумя кучами. Мы берём значение s и проверяем, превышает ли сумма двух куч победное значение. Если нет, то число шагов увеличивается на 1 и так далее. Как смотрим ответы: ответ на 19 задание – последнее число, у которого встречается 1 в значениях, что идут после второго двоеточия; ответ на 20 задание – числа, у которых число шагов равно 3 (после первого двоеточия); ответ на 21 задание – числа, у которых число шагов равно 4 (после первого двоеточия).

Задание 23

(повышенный уровень, время – 8 мин)
Тема: динамическое программирование.
Что проверяется:
Умение анализировать результат исполнения алгоритма
1.6.2. Вычислимость. Эквивалентность алгоритмических моделей.
1.1.3. Умение строить информационные модели объектов, систем и процессов в виде алгоритмов.

Пример.
Исполнитель преобразует число на экране.
У исполнителя есть две команды, которым присвоены номера:

1. Прибавить 1
2. Умножить на 2

Первая команда увеличивает число на экране на 1, вторая умножает его на 2. Программа для исполнителя – это последовательность команд.

Сколько существует программ, для которых при исходном числе 1 результатом является число 20, и при этом траектория вычислений содержит число 10?

Траектория вычислений программы – это последовательность результатов выполнения всех команд программы. Например, для программы 121 при исходном числе 7 траектория будет состоять из чисел 8, 16, 17.

Решение: (на языке Python)

```
def f(start,x):
    if x>start:
        if x%2==0:
            return f(start, x-1) + f(start, x//2)
        else:
            return f(start, x-1)
    elif x==start:
        return 1
    elif x<start:
        return 0
print(f(1,10)*f(10,20))
```

//

(на языке Pascal)

```
var start,x: integer;
 function f(start,x:integer):integer;
begin
  if x>start then begin
    if (x mod 2=0) and (x>1) then f:=f(start,x-1)+f(start,x div 2) else f:=f(start,x-1);
  end;
  if x=start then f:=1;
  if x<start then f:=0
end;
 Begin
  writeln(f(1,10)*f(10,20))
end.
```

Ответ: 28

Пояснение к программе: Для начала внимательно вчитываемся в условия задания. Далее наша программа получает на вход два числа: стартовая позиция и число, от которого идём. Эти два значения передаются в функцию и начинают обрабатываться, выполняя действия, противоположные данным, так как мы движемся в обратном направлении. Если в программе есть обязательные пункты, то мы разбиваем наш отрезок чисел на маленькие подотрезки, записываем функции для этих подотрезков, а далее выполняем произведение (смотреть код программы).

Задания 24,26 и 27 выполняются с использованием прилагаемых файлов.

Задание 24

(высокий уровень, время – 18 минут)

Тема: Обработка символьных строк

Что проверяется:

Умение создавать собственные программы (10–20 строк) для обработки символьной информации.

1.5.2. Цепочки (конечные последовательности), деревья, списки, графы, матрицы (массивы), псевдослучайные последовательности.

1.1.3. Строить информационные модели объектов, систем и процессов в виде алгоритмов.

Пример.

Текстовый файл состоит не более чем из 10^6 символов X, Y и Z. Определите максимальное количество идущих подряд символов, среди которых каждые два соседних различны. Для выполнения этого задания следует написать программу.

Решение: (на языке Python)

```
f = open(r'/content/24_demo.txt','r')
data = f.readline()
count = 1
maxlen = 1
for i in range(1,len(data)):
  if data[i]!=data[i-1]:
    count+=1
  else:
    if count>=maxlen:
      maxlen = count
    count=1
print(maxlen)
```

Ответ: 35

Пояснение к программе: Считываем файл, который прилагается к заданию. Далее заводим счётчик на максимальную длину. Начинаем пробегаться по тексту файла. Если выполняется условие того, что предыдущая буква не равна текущей, то увеличиваем длину на 1. В противном же случае смотрим, больше ли длина такой строки текущего максимума. Если это так, то максимальная длина равна длине данной строки. Ставим счётчик длины текущей строки на 1 для следующих строк.

Задание 25

(высокий уровень, время – 20 минут)

Тема: Обработка целых чисел. Проверка делимости

Что проверяется:

Умение создавать собственные программы (10–20 строк) для обработки целочисленной информации.

1.5.2. Цепочки (конечные последовательности), деревья, списки, графы, матрицы (массивы), псевдослучайные последовательности.

1.1.3. Строить информационные модели объектов, систем и процессов в виде алгоритмов.

Пример.

Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [174457; 174505], числа, имеющие ровно два различных натуральных делителя, не считая единицы и самого числа. Для каждого найденного числа запишите эти два делителя в таблицу на экране с новой строки в порядке возрастания произведения этих двух делителей. Делители в строке таблицы также должны следовать в порядке возрастания.

Решение: (на языке Python)

```
for n in range(174457,174505+1):
  d=2
  kd=0
  mas=[]
  while kd<2 and d*d<=n:
  if n%d==0:
  kd+=1
  mas.append(d)
  mas.append(n//d)
  d+=1
  kd*=2
     if kd==2:
   print(*mas)
```

Ответ:

3 58153
7 24923
59 2957
13 13421
149 1171
5 34897
211 827
2 87251

Пояснение к программе: Данная программа работает в диапазоне, указанном в условии. Здесь нужно перебирать до корня из числа для большей эффективности, так как корнем из числа все делители разбиваются пополам. Тогда мы ищем такие числа, у которых есть ровно один делитель до корня из числа. Вообще, нужно проверять и сам корень от числа, так как если он целый, то он тоже является делителем, такое происходит у квадратов чисел. Однако в нашем диапазоне нет ни одного квадрата, поэтому здесь мы можем корень из числа не считать. Далее если делитель один, то выводим этот делитель и число, делённое на этот делитель.

Задание 26

(высокий уровень, время – 35 минут)

Тема: Обработка массива целых чисел из файла. Сортировка.

Что проверяется:

Умение обрабатывать целочисленную информацию с использованием сортировки.

1.6.3. Построение алгоритмов и практические вычисления.

1.1.3. Строить информационные модели объектов, систем и процессов в виде алгоритмов.

Пример.

Системный администратор раз в неделю создаёт архив пользовательских файлов. Однако объём диска, куда он помещает архив, может быть меньше, чем суммарный объём архивируемых файлов.

Известно, какой объём занимает файл каждого пользователя.

По заданной информации об объёме файлов пользователей и свободном объёме на архивном диске определите максимальное число пользователей, чьи файлы можно сохранить в архиве, а также максимальный размер имеющегося файла, который может быть сохранён в архиве, при условии, что сохранены файлы максимально возможного числа пользователей.

Входные данные.

В первой строке входного файла находятся два числа: S – размер свободного места на диске (натуральное число, не превышающее 10 000) и N – количество пользователей (натуральное число, не превышающее 1000). В следующих N строках находятся значения объёмов файлов каждого пользователя (все числа натуральные, не превышающие 100), каждое в отдельной строке.

Запишите в ответе два числа: сначала наибольшее число пользователей, чьи файлы могут быть помещены в архив, затем максимальный размер имеющегося файла, который может быть сохранён в архиве, при условии, что сохранены файлы максимально возможного числа пользователей.

Пример входного файла:

100 4 80

30

50
40

При таких исходных данных можно сохранить файлы максимум двух пользователей. Возможные объёмы этих двух файлов 30 и 40, 30 и 50 или 40 и 50. Наибольший объём файла из перечисленных пар – 50, поэтому ответ для приведённого примера:

2 50

Решение: (на языке Python)

```
f = open(r'/content/26_demo (1).txt','r')
 S,N = map(int, f.readline().split())
 A = [int(f.readline()) for i in range(N)]
A.sort()
 S1 = 0
i=0
 while S1<=S:
  S1+=A[i]
  i+=1
S1=S1-A[i-1]-A[i-2]
i-=1
imax=i
 while S1 + A[i] <= S:
  i+=1
 print(imax,A[i-1])
```

////////////////////////////

(на языке Pascal)

```
var
i, j, t: integer;
a: array [1..1000] of integer;
s: integer;
n: integer;
sum: integer;
maxi: integer;
f: text;
begin
    assign(f,'26_demo.txt');
    reset(f);
    readln(f, s, n);
    for i := 1 to n do readln(f, a[i]);
    for i := 1 to n do
        for j := i + 1 to n do
            if a[i] > a[j] then begin
                t := a[i];
                a[i] := a[j];
```

```
            a[j] := t;
        end;
    sum := 0;
    maxi := 1;
    for i := 1 to n do
        if sum + a[i] <= s then begin
            sum := sum + a[i];
            maxi := i;
        end;
    t := a[maxi];
    for i := maxi to n do
        if ((sum - t) + a[i]) <= s then begin
            sum := sum - t + a[i];
            t := a[i];
        end;
    writeln(maxi, ' ', t);
End.
```

Ответ: 568 50

Пояснение к программе: Сначала считываем файл из задания. Далее считываем все значения, которые есть в этом файле. Записываем это всё в массив и далее отсортировываем его (массив) по возрастанию. Далее пока у нас в получившемся массиве числа вмещаются в размер архива, мы их добавляем в промежуточную сумму. Эта сумма должна быть меньше или равна размера архива. Записываем номер последнего файла и вычитаем из него единицу, это один из ответов. Далее мы вычитаем из промежуточной суммы два последних значения. Далее снова пытаемся вместить файлы, но начинать мы будем с максимального индекса (то, что мы вычитали 1 из номера последнего файла). Записываем ответ.

Задание 27

(высокий уровень, время – 35 мин)

Тема: Обработка данных, вводимых из файла в виде последовательности чисел.

Что проверяется:

Умение создавать собственные программы (20–40 строк) для анализа числовых последовательностей.

1.6.3. Построение алгоритмов и практические вычисления.

1.1.3. Строить информационные модели объектов, систем и процессов в виде алгоритмов.

Пример.

Имеется набор данных, состоящий из пар положительных целых чисел. Необходимо выбрать из каждой пары ровно одно число так, чтобы сумма всех выбранных чисел не делилась на 3 и при этом была максимально возможной. Гарантируется, что искомую сумму получить можно. Программа должна напечатать одно число – максимально возможную сумму, соответствующую условиям задачи.

Входные данные.

Даны два входных файла (файл *A* и файл *B*), каждый из которых содержит в первой строке количество пар *N* (1 ⩽ *N* ⩽ 100000). Каждая из следующих *N* строк содержит два натуральных числа, не превышающих 10 000.

Пример организации исходных данных во входном файле: 6

1 3

5 12

6 9

5 4

3 3

1 1

Для указанных входных данных значением искомой суммы должно быть число 32.

В ответе укажите два числа: сначала значение искомой суммы для файла *A*, затем для файла *B*.

Предупреждение: для обработки файла *B* не следует использовать переборный алгоритм, вычисляющий сумму для всех возможных вариантов, поскольку написанная по такому алгоритму программа будет выполняться слишком долго.

Решение: (на языке Pascal)

```
var x, y: integer;
n: integer;
sum: integer;
mindif: integer;
f: text;
begin
assign(f,'C:\27-A.txt');
reset(f);
readln(f, n);
sum := 0;
mindif := 20001;
while not eof(f) do begin
readln(f, x, y);
if x > y then
sum := sum + x
else
```

```
sum := sum + y;
if (abs(x - y) < mindif) and (abs(x-y) mod 3 <> 0) then mindif := abs(x-y);
end;
if sum mod 3 <> 0 then writeln(sum) else writeln(sum - mindif);
end.
```

Ответ: 127127 399762080

Пояснение к программе: Считываем файл и далее считываем первое значение. Следующим делом мы заводим переменную суммы чисел и переменную минимальной разности, не делящейся в нашем случае на 3, а потом считываем числа парами. Из каждой пары выбираем самое большое число и добавляем его в сумму. Смотрим разность чисел в паре. Если разность меньше текущего минимума и не делится на 3, то минимальная разность равна этой разности. И так до конца. Далее смотрим на сумму: если она делится на 3, то вычитаем эту минимальную разность из суммы, а иначе выводим сумму.