

Изучение библиотеки NumPy.

Библиотека NumPy предназначена для работы с многомерными массивами чисел и часто используется при работе с другими библиотеками (в частности `matplotlib`, `scikit`, `torch` и многими другими).

Установка

Нужно запустить Anaconda prompt (через пуск-Anaconda3), там ввести команды:
`conda install numpy pillow scikit-image matplotlib -user.`

Если у вас не установлена Anaconda, то нужно из консоли запустить команду:
`pip install numpy pillow scikit-image matplotlib -user`

Замечания по вводу-выводу

Каждая программа должна начинаться со строк с импортированием NumPy и настройкой вывода:

```
import sys
import numpy as np
np.set_printoptions(precision=4, threshold=sys.maxsize, linewidth=sys.maxsize, suppress=True)
```

Если на вход в задаче даётся один или несколько `numpy`-массивов, то получать все данные необходимо из файла `input.txt` при помощи команды вида:

```
ar = eval(open('input.txt').read())
ar1, ar2 = eval(open('input.txt').read())
```

Во всех остальных случаях используйте стандартные способы чтения.

При выводе массивов всегда используйте функцию `repr`. Если правильный массив хранится в переменной `x`, то используйте для вывода команду

```
print(repr(x))
```

Все задачи должны быть решены без использования циклов. Читайте документацию, экспериментируйте.

Создание массива

```
import numpy as np

# 1-dimension array of zeroes
np.zeros(5)
# array([ 0.,  0.,  0.,  0.,  0.])

# 2-dimension array of zeroes
np.zeros((2, 3))
# array([[ 0.,  0.,  0.], [ 0.,  0.,  0.]])

# 3-dimension array of ones
np.ones((2, 3, 4))
# array([[[ 1.,  1.,  1.,  1.],
# [ 1.,  1.,  1.,  1.],
# [ 1.,  1.,  1.,  1.],
# [ 1.,  1.,  1.,  1.],
# [ 1.,  1.,  1.,  1.]]])

# array of zeroes with type
np.zeros(5, dtype=np.int)
# array([0, 0, 0, 0, 0])

# list-based array
np.array([2, 3, 1, 0])
# array([2, 3, 1, 0])

# 2d list-based array
np.array([[1, 2.0], [0, 0], (1+1j, 3.)])
# array([[ 1.+0.j,  2.+0.j],
# [ 0.+0.j,  0.+0.j],
# [ 1.+1.j,  3.+0.j]])
```

Арифметические прогрессии

```
# arithmetic progression
np.arange(10)
# array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

# arithmetic progression with type
np.arange(2, 10, dtype=np.float)
# array([ 2., 3., 4., 5., 6., 7., 8., 9.])

# non-int difference arithmetic progression
np.arange(2, 3, 0.1)
# array([ 2. , 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.7, 2.8, 2.9])

# arithmetic progression of a given length
np.linspace(1., 4., 6)
# array([ 1. , 1.6, 2.2, 2.8, 3.4, 4. ])
```

Документация

- Оглавление документации: <https://numpy.org/doc/stable/reference/index.html>
- Полный список команд: <https://numpy.org/doc/stable/genindex.html>
- 4 главы на хабре: <https://habr.com/ru/post/352678/>

А. Массив нулей

На вход даётся число N . Выведите массив нулей из N строк и $2N$ столбцов.

Input	Output
3	<code>array([[0., 0., 0., 0., 0., 0.], [0., 0., 0., 0., 0., 0.], [0., 0., 0., 0., 0., 0.]])</code>

Метод `reshape` и обращение по индексам через запятую

```
x = np.arange(12).reshape(3, 4)
print(repr(x))
# array([[ 0,  1,  2,  3],
#        [ 4,  5,  6,  7],
#        [ 8,  9, 10, 11]])

print(x[1, 2])
# 6
```

В. Числа в нулевой строке

На вход даются числа N и M . Выведите массив размера $N \times M$, в котором в первой строчке (строка с нулевым индексом) расположены числа от 0 до $M - 1$, а остальные числа равны 0. Тип элементов массива должен быть `np.int8`.

Input	Output
3 5	<code>array([[0, 1, 2, 3, 4], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0]], dtype=int8)</code>

С. Числа на диагонали

На вход даётся число N . Выведите массив размера $N \times N$, в котором по диагонали расположены числа от 0 до $N - 1$. Тип элементов массива должен быть `np.int64`.

Input	Output
4	<code>array([[0, 0, 0, 0], [0, 1, 0, 0], [0, 0, 2, 0], [0, 0, 0, 3]], dtype=int64)</code>

Срезы

```
x = np.arange(12).reshape(3, 4)
# array([[ 0,  1,  2,  3],
#        [ 4,  5,  6,  7],
#        [ 8,  9, 10, 11]])

# each coordinate can be a slice -- (:, a:b, ::-1, a:b:c)
# row index=1
print(repr(x[1, :]))
# array([4, 5, 6, 7])

# column index=2
print(repr(x[:, 2]))
# array([ 2,  6, 10])

# row in reverse order
print(repr(x[0, ::-1]))
# array([3, 2, 1, 0])
```

Д. Сбитый прицел

На вход даются числа N, R, C . Выведите массив размера $N \times N$, в котором в строке R и столбце C стоят 1, а остальные числа равны 0.

Тип элементов массива должен быть таким, чтобы во-первых, числа выводились как целые, без точек. А во-вторых, чтобы при выводе при помощи `repr` конкретный тип не указывался. В документации можно увидеть список возможных типов.

Input	Output
5 1 3	<code>array([[0, 0, 0, 1, 0], [1, 1, 1, 1, 1], [0, 0, 0, 1, 0], [0, 0, 0, 1, 0], [0, 0, 0, 1, 0]])</code>

Е. Почётные единицы

На вход даётся число N . Выведите массив размера $N \times N$, в котором в строках с чётными индексами стоят 1, а в остальных — нули.

Input	Output
5	<code>array([[1, 1, 1, 1, 1], [0, 0, 0, 0, 0], [1, 1, 1, 1, 1], [0, 0, 0, 0, 0], [1, 1, 1, 1, 1]])</code>

Ф. Решето

На вход даются числа N, M, R, C . Выведите массив размера $N \times M$, в котором в каждой R -ой строчке и в каждом C -ом столбце стоят нули, а остальные элементы равны 1.

Input	Output
5 11 3 4	<code>array([[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1], [0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1]])</code>

Г. Шахматные единицы

На вход даётся число N . Выведите массив размера $N \times N$, имеющий вид шахматной доски. В верхнем левом углу должна стоять единица.

Input	Output
5	<code>array([[1, 0, 1, 0, 1], [0, 1, 0, 1, 0], [1, 0, 1, 0, 1], [0, 1, 0, 1, 0], [1, 0, 1, 0, 1]])</code>

Использование списка индексов в срезах

```
x = np.arange(12).reshape(3, 4)
print(repr(x))
# array([[ 0,  1,  2,  3],
#        [ 4,  5,  6,  7],
#        [ 8,  9, 10, 11]])

# using list of indexes
print(repr(x[:, [1, 3]]))
# array([[ 1,  3],
#        [ 5,  7],
#        [ 9, 11]])

# all lists should be of the same length
print(repr(x[[0, 1], [1, 3]]))
# array([1,  7])

print(repr(x[np.arange(3), np.arange(3)]))
# array([ 0,  5, 10])
```

Н. Разлиновка

На вход даётся число N . В следующей строке записано несколько целых неотрицательных чисел. Выведите массив размера $N \times N$, в котором в строках с перечисленными выше индексами стоят 1.

Input	Output
10 3 4 7 9	array([[0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [1, 1, 1, 1, 1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1, 1, 1, 1, 1], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [1, 1, 1, 1, 1, 1, 1, 1, 1, 1], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [1, 1, 1, 1, 1, 1, 1, 1, 1, 1]])

Операции с массивами и векторами

```
x = np.arange(12).reshape(2, 6)

# can add numbers to all array elements
print(repr(x + 10))
# array([[10, 11, 12, 13, 14, 15],
# [16, 17, 18, 19, 20, 21]])

# can multiply all elements by some number
print(repr(x * 3))
# array([[ 0,  3,  6,  9, 12, 15],
# [18, 21, 24, 27, 30, 33]])

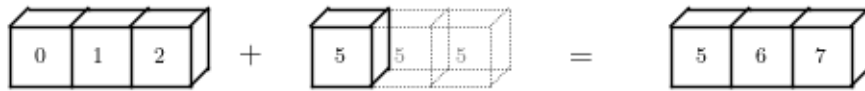
# can add another array of the same shape
x = np.arange(6).reshape(2, 3)
y = np.arange(10, 16).reshape(2, 3)
print(repr(x + y))
# array([[10, 12, 14],
# [16, 18, 20]])

# can add array with 1 row, then broadcasting along rows
print(repr(x + np.array([0, 10, 100, -10, -100, 0])))
# array([[ 0, 11, 102, -7, -96, 5],
# [ 6, 17, 108, -1, -90, 11]])

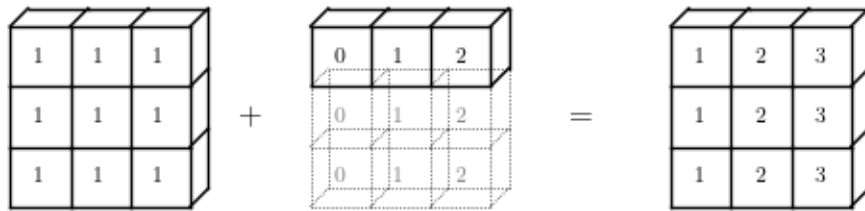
# can add array with 1 column, then broadcasting along columns
print(repr(x + np.array([0], [-100])))
# array([[ 0,  1,  2,  3,  4,  5],
# [-94, -93, -92, -91, -90, -89]])
print(repr(x + np.array([0, -100]).reshape((2, 1))))
# array([[ 0,  1,  2,  3,  4,  5],
# [-94, -93, -92, -91, -90, -89]])

# can apply functions to all array elements (including logical)
print(repr(np.sqrt(np.abs(x))))
array([[ 0. ,  1. ,  1.4142,  1.7321,  2. ,  2.2361],
 [ 2.4495,  2.6458,  2.8284,  3. ,  3.1623,  3.3166]])
print(repr(x > 3))
# array([[False, False, False, False, True, True],
# [ True,  True,  True,  True,  True,  True]], dtype=bool)
print(repr((x > 3) & (x < 8)))
# array([[False, False, False, False, True, True],
# [ True,  True, False, False, False, False]], dtype=bool)
```

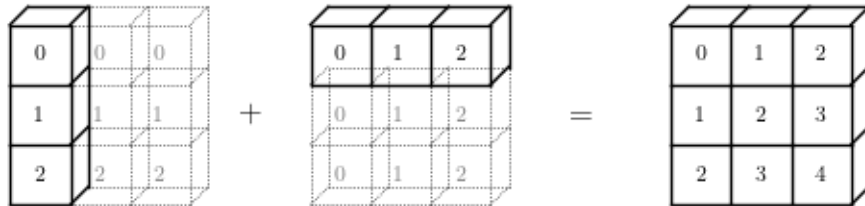
`np.arange(3) + 5`



`np.ones((3, 3)) + np.arange(3)`



`np.arange(3).reshape((3, 1)) + np.arange(3)`



Подробнее про broadcasting: статья в стандартной документации.

I. Скалярное произведение

На вход даются два массива векторов v_i и w_i . Вычислите их скалярные произведения (v_i, w_i) .

Input	Output
<code>(np.array([[9, -3], [2, -10], [7, 8]]), np.array([[-5, 2], [7, -3], [2, 1]]))</code>	<code>array([-51, 44, 22])</code>

J. Нормировка

Дан массив действительных чисел. Сделайте его нормировку $X \rightarrow aX + b$ так, чтобы все числа лежали на отрезке $[0, 1]$ и значения 0 и 1 принимались. Гарантируется, что в массиве есть хотя бы два различных числа.

Input	Output
<code>np.array([[0., -8.], [9., -4.], [1., 4.]])</code>	<code>array([[0.4706, 0.], [1., 0.2353], [0.5294, 0.7059]])</code>

K. Основное тригонометрическое тождество

Дан массив действительных чисел x_i длины N .

Выведите массивы: $\sin x_i, \cos x_i, \sin^2 x_i, \cos^2 x_i, \sin^2 x_i + \cos^2 x_i$.

Input
<code>np.array([5.6594, 14.8314, 11.1459, 5.9925, -9.2026])</code>
Output
<code>array([-0.5841, 0.7686, -0.9887, -0.2866, -0.2204])</code> <code>array([0.8117, -0.6398, 0.1497, 0.9581, -0.9754])</code> <code>array([0.3412, 0.5907, 0.9776, 0.0821, 0.0486])</code> <code>array([0.6588, 0.4093, 0.0224, 0.9179, 0.9514])</code> <code>array([1., 1., 1., 1., 1.])</code>

L. Квадратные уравнения

Даны три массива действительных чисел a_i, b_i, c_i длины N .

Выведите два массива размера N корней квадратных уравнений $a_i x^2 + b_i x + c_i$.

Гарантируется, что у каждого уравнения есть два корня. Сначала должны выводиться корни, в которых вычитается корень из дискриминанта.

Input	Output
<code>(np.array([-5, -1, 4, -2, 1]), np.array([70, 8, -8, 24, -9]), np.array([-225, 9, -192, -54, 14]))</code>	<code>array([9., 9., -6., 9., 2.])</code> <code>array([5., -1., 8., 3., 7.])</code>

Использование массивов bool в срезах

```
x = np.arange(12).reshape(2, 6)
# instead of index's list can apply bool array of the same length: use or not to use
print(repr(x[np.array([True, False]), :]))
# array([[0, 1, 2, 3, 4, 5]])
print(repr(x[:, np.array([True, True, False, False, False, True])]))
# array([[ 0, 1, 5],
#        [ 6, 7, 11]])

# can get these bool arrays automatically
x > 3
# array([[False, False, False, False, True, True],
#        [ True, True, True, True, True, True]], dtype=bool)
print(repr(x[x > 3]))
# array([ 4, 5, 6, 7, 8, 9, 10, 11])

# using and, or, not and not forget about parenthesis
print(repr((x > 3) & (x < 8)))
# array([[False, False, False, False, True, True],
#        [ True, True, False, False, False, False]], dtype=bool)
print(repr(x[(x > 3) & (x < 8)]))
# array([4, 5, 6, 7])
```

М. Большие нуля

Дан массив действительных чисел. Выведите массив его положительных элементов.

Input	Output
np.array([-5, 1, 7, -8, -3, 9, -4, 6, -2, -9])	array([1, 7, 9, 6])

Н. Поменять знак

Дан массив действительных чисел. Поменяйте знак у элементов, значения которых между 3 и 8.

Input
np.array([-8, -6, -3, 8, -4, -10, -9, -7, -3, 6])
Output
array([-8, -6, -3, -8, -4, -10, -9, -7, -3, -6])

Максимумы и минимумы

```
x = np.random.randint(0, 10, 10)
print(repr(x))
# array([8, 2, 8, 3, 4, 8, 9, 9, 3, 1])
print(x.max(), x.min(), x.argmax(), x.argmin())
# 9 1 6 9

# if you have more than 1-dimension array then argmin and argmax
# returns linear index (first occurrence, if many)
x = np.random.randint(0, 10, (2, 6))
print(repr(x))
# array([[1, 4, 9, 6, 6, 7],
#        [9, 8, 9, 5, 2, 7]])
print(x.argmax())
# 2

# but it can be converted into coordinates
print(np.unravel_index(x.argmax(), x.shape))
# (0, 2)

# can find all coordinates based on some rule
print(np.argwhere(x==x.max()))
# array([[0, 2],
#        [1, 0],
#        [1, 2]])
```

О. *Заменить все максимумы*

Дан массив действительных чисел. Замените все значения, равные максимальному, на -1 .

Input
<code>np.array([4, 8, 7, 5, 1, 6, 1, 1, 8, 5])</code>
Output
<code>array([4, -1, 7, 5, 1, 6, 1, 1, -1, 5])</code>

Р. *Ближайшее*

Дан массив действительных чисел и некоторое число. Найдите ближайшее по модулю к этому числу значение в массиве и его индекс.

В решении нужно обойтись без циклов.

Input	Output
<code>(np.array([0.91, 0.55, 0.87, 0.52, 0.34, 0.69, 0.74]), 0.5)</code>	<code>0.52</code> <code>3</code>

Q. *Построчный и постолбцовый максимум*

Дан прямоугольный массив действительных чисел.

Выведите массив максимальных значений в каждой строке, а за ним массив максимальных значений в каждом столбце.

Input	Output
<code>np.array([[80, 85, 61], [57, 41, 22]])</code>	<code>array([85, 57])</code> <code>array([80, 85, 61])</code>

Р. *Среднее, построчное и постолбцовое среднее*

Дан прямоугольный массив действительных чисел X .

Выведите три массива: массив X , у которого из каждого элемента вычли:

- среднее арифметическое всех чисел массива X
- среднее арифметическое всех чисел данной строки массива X
- среднее арифметическое всех чисел данного столбца массива X

Input	Output
<code>np.array([[7, 3, 2], [6, 8, 4]])</code>	<code>array([[2., -2., -3.], [1., 3., -1.]])</code> <code>array([[3., -1., -2.], [0., 2., -2.]])</code> <code>array([[0.5, -2.5, -1.], [-0.5, 2.5, 1.]])</code>

Сортировка

```
a = np.array([[1, 4], [3, 1]])

# sorting (default -- by last axis=1), i.e. from left to right
print(repr(np.sort(a)))
# array([[1, 4],
#        [1, 3]])

# sorting by axis=0, i.e. from top to bottom
print(repr(np.sort(a, axis=0)))
# array([[1, 1],
#        [3, 4]])

print(repr(np.sort(a, axis=None))) # sorting of linearized array
# array([1, 1, 3, 4])

# different methods
x = np.array([3, 1, 2])
print(repr(np.argsort(x)))
# array([1, 2, 0], dtype=int64)

print(repr(x[np.argsort(x)]))
# array([1, 2, 3])

x = np.array([[3, 1, 2], [0, 1, 2], [10, 11, 12]])
print(repr(x))
# array([[ 3,  1,  2],
#        [ 0,  1,  2],
#        [10, 11, 12]])
print(repr(x[:, np.argsort(x[0, :])]))
# array([[ 1,  2,  3],
#        [ 1,  2,  0],
#        [11, 12, 10]])
print(repr(x[np.argsort(x[:, 0]), :]))
# array([[ 0,  1,  2],
#        [ 3,  1,  2],
#        [10, 11, 12]])
```

S. Сортировка

Дан линейный массив действительных чисел X . Отсортируйте его.

Input	Output
<code>np.array([1, 9, 2, 8, 3, 7])</code>	<code>array([1, 2, 3, 7, 8, 9])</code>

T. *argsort*

Дан линейный массив действительных чисел X . Выведите массив индексов, обладающих следующим свойством: в позиции i стоит индекс элемента массива X , который окажется i -ым, если массив отсортировать.

Input	Output
<code>np.array([1, 9, 2, 8, 3, 7])</code>	<code>array([0, 2, 4, 5, 3, 1])</code>

U. *argsort* — 2

Даны три линейных массива одинаковой длины. В первом массиве хранятся стоимости брусков, во втором — длины, а в третьем — массы. Отсортируйте массивы так, чтобы данные по брускам шли в порядке возрастания стоимости.

Input
<code>(np.array([1046, 1267, 1044, 1875, 1620, 1412]), np.array([418, 477, 240, 290, 272, 211]), np.array([3.5002, 1.3224, 1.2677, 3.9586, 1.0951, 1.144]))</code>
Output
<code>array([1044, 1046, 1267, 1412, 1620, 1875]) array([240, 418, 477, 211, 272, 290]) array([1.2677, 3.5002, 1.3224, 1.144 , 1.0951, 3.9586])</code>

V. Порядок индексов

Даны числа W и H — ширина и высота картинки в пикселях.

Создайте подходящий массив, в котором можно хранить трёхцветную картинку данной ширины и высоты. При этом данные в памяти должны храниться так, чтобы сначала шли три цвета одного пикселя, затем цвета следующего пикселя в этой же строчке или первый пиксель из следующей строчки. Каждый элемент должен быть однобайтным беззнаковым целым числом (`np.uint8`).

Input	Output
3 3	<code>array([[0, 0, 0], [0, 0, 0], [0, 0, 0]], [[0, 0, 0], [0, 0, 0], [0, 0, 0]], [[0, 0, 0], [0, 0, 0], [0, 0, 0]]], dtype=uint8)</code>

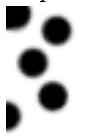
W. Размер картинки

В файле `input.png` сохранена картинка. Прочитайте её при помощи команды вида:

```
from skimage import io  
img = io.imread('input.png')
```

Картинка может быть чёрно-белой (получится двумерный массив, где каждый элемент — число от 0 до 255) или цветной (трёхмерный массив, элементы двумерного массива представляют собой тройки чисел — интенсивности красного, зелёного и синего цветов соответственно).

Выведите её размеры: сначала ширину в пикселях, а затем высоту.

Input	Output
	30 60
	30 60

Ссылки для скачивания: входной файл 1, входной файл 2.

X. Зелёный слой

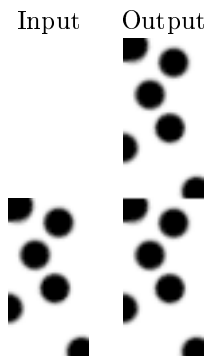
Из файла `input.png` прочитайте цветную картинку. Запишите её зелёный слой в виде ч/б изображения, то есть двумерного массива и сохраните в файл `output.png`.

Input	Output
	

Ссылки для скачивания: входной файл, выходной файл.

Y. Состыковка — 1

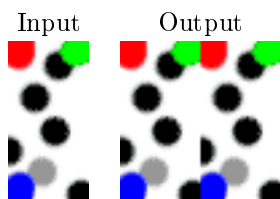
Из файла `input.png` прочитайте ч/б картинку. В файл `output.png` запишите картинку, полученную из двух таких картинок состыковкой «одна над другой».



Ссылки для скачивания: входной файл, выходной файл.

Z. Состыковка — 2

Из файла `input.png` прочитайте ч/б картинку. В файл `output.png` запишите картинку, полученную из двух таких картинок состыковкой «плечо к плечу».

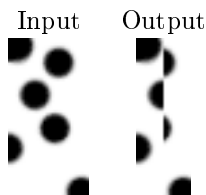


Ссылки для скачивания: входной файл, выходной файл.

ZA. Удаление — 1

Из файла `input.png` прочитайте ч/б картинку. В файл `output.png` запишите картинку, полученную из исходной удалением по центру вертикальной полосы шириной в 10 пикселей.

Гарантируется, что ширина картинки не меньше 12 пикселей.

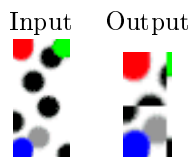


Ссылки для скачивания: входной файл, выходной файл.

ZB. Удаление — 2

Из файла `input.png` прочитайте цветную картинку. В файл `output.png` запишите картинку, полученную из исходной удалением по центру горизонтальной полосы высотой в 20 пикселей, затем удалением вертикальной полосы шириной в 10 пикселей от $\frac{3}{4}W - 5$ до $\frac{3}{4}W + 4$ включительно (W — ширина картинки).

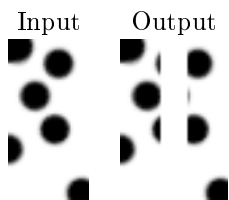
Гарантируется, что высота картинки чётная и не меньше 22 пикселей, ширина не меньше 12 пикселей, а также то, что W делится на 4.



Ссылки для скачивания: входной файл, выходной файл.

ZC. Вставка — 1

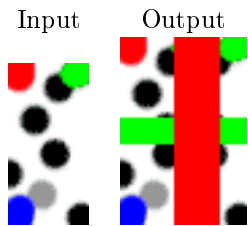
Из файла `input.png` прочитайте ч/б картинку. В файл `output.png` запишите картинку, полученную из исходной вставкой по центру белой вертикальной полосы шириной в 10 пикселей.



Ссылки для скачивания: [входной файл](#), [выходной файл](#).

ZD. Вставка — 2

Из файла `input.png` прочитайте цветную картинку. В файл `output.png` запишите картинку, полученную из исходной вставкой по центру зелёной горизонтальной полосы высотой в 10 пикселей, затем вставкой красной вертикальной полосы шириной в 17 пикселей так, чтобы пространство слева было в два раз шире пространства справа.



Ссылки для скачивания: [входной файл](#), [выходной файл](#).