

# Zmiana rozmiarów obrazu

Proszę wykonać podane zadania na obrazkach „rzeczka.jpg”, „rzeczka\_mniejsza.jpg”, „logo\_linuxa.png” i „zrzut\_1.png”.



13	14	15	16	17	18	19	0,4	0	0	0	0
23	24	25	26	27	28	29	0	0	0,5	0	0
33	34	35	36	37	38	39	0	0	-1,1	0	0
63	64	65	66	67	68	69	0	0	0	0	0
73	74	75	76	77	78	79	0	0	0	0	0,6
83	84	85	86	87	88	89					



*W celu lepszej obserwacji dokonanych zmian, najlepiej jest zapisywać wynik działania przeskalowania w pliku (polecenie `imwrite`), a następnie porównać oba obrazki w dowolnej przeglądarce (lub edytorze) plików graficznych.*

# Teoria

Proces zmiany rozmiarów obrazu jest zadaniem zaskakująco mocno skomplikowanym, powolnym i przy tym często wykonywanym. Istnieje wiele algorytmów służących do tego celu. Efekt działania większości bardziej skomplikowanych algorytmów jest dość podobny. Najprostszy z nich - „najbliższego sąsiada” jest bardzo prosty w implementacji, niestety efekty jego działania są w większości zastosowań nieakceptowalnie kiepskie.

## Użyte stałe

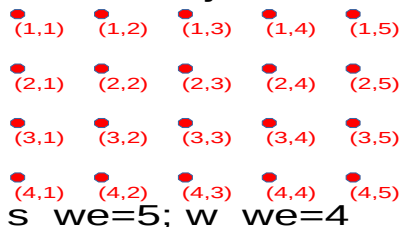
- $s\_wy, w\_wy$  to odpowiednie szerokość i wysokość obrazu wyjściowego,
- $s\_we, w\_we$  to odpowiednie szerokość i wysokość obrazu wejściowego.

Zakłada się, że  $s\_wy, w\_wy, s\_we, w\_we > 1$ .

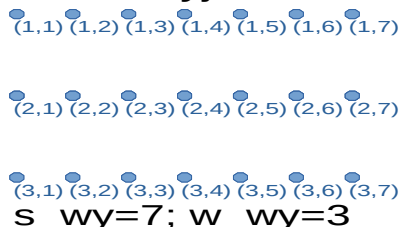
## Algorytm „najbliższego sąsiada”

Aby zrozumieć sposób działania tego algorytmu trzeba wyobrazić sobie 2 siatki punktów, zwane siatką wejściową i wyjściową. Każda z tych siatek zajmuje taką samą szerokość i wysokość, lecz zagęszczenia punktów w obu siatkach w pionie i poziomie są zwykle różne. Obie siatki mają równomiernie rozłożone punkty (o różnych zagęszczeniach). Każdy punkt reprezentuje pojedynczy piksel przed i po przeskalowaniu. Oto przykład.

Siatka wejściowa



Siatka wyjściowa



Po nałożeniu na siebie obu siatek widać zależność między położeniami pikseli **przed** i **po** przeskalowaniu.



W celu znalezienia wartości **pikseła wyjściowego** należy znaleźć najbliższy mu **piksel wejściowy** i przekopiować z niego wartość. Aby ustalić, który to jest piksel, trzeba przyporządkować położeniu pikseła wyjściowego położenie pikseła wejściowego, a następnie dokonać zaokrąglenia,

$$x\_we(x\_wy) = \text{zaokr} \left( 1 + \frac{(x\_wy - 1) \cdot (s\_we - 1)}{s\_wy - 1} \right),$$

$$y\_we(y\_wy) = \text{zaokr} \left( 1 + \frac{(y\_wy - 1) \cdot (w\_we - 1)}{w\_wy - 1} \right),$$

gdzie  $\text{zaokr}()$  to funkcja zaokrąglająca,  $(y\_we, x\_we)$  to położenie punktu wejściowego,  $(y\_wy, x\_wy)$  to położenie punktu wyjściowego;  $s\_wy, w\_wy$  to szerokość i wysokość obrazu wyjściowego;  $s\_we, w\_we$  to szerokość i wysokość obrazu wejściowego. W tych wzorach przyjmuje się, że piksele indeksuje się zaczynając od 1, a nie od 0.

Aby dokonać przeskalowania za pomocą algorytmu „najbliższego sąsiada”, należy dla każdego pikseła w obrazku wyjściowym obliczyć położenie pikseła wejściowego, a następnie go przekopiować.

## Sugestia

W celu przyspieszenia pracy można stworzyć i używać wektor zawierający translację wszystkich możliwych wartości  $x\_wy$  na  $x\_we$  i translację wszystkich możliwych wartości  $y\_wy$  na  $y\_we$ .

1. Proszę przeskalować algorytmem „najbliższy sąsiad” obrazek „rzeczka” do rozmiarów 333x222.



2. Proszę przeskalować algorytmem „najbliższy sąsiad” obrazek „rzeczka\_mniejsza” do rozmiaru 1366x768.





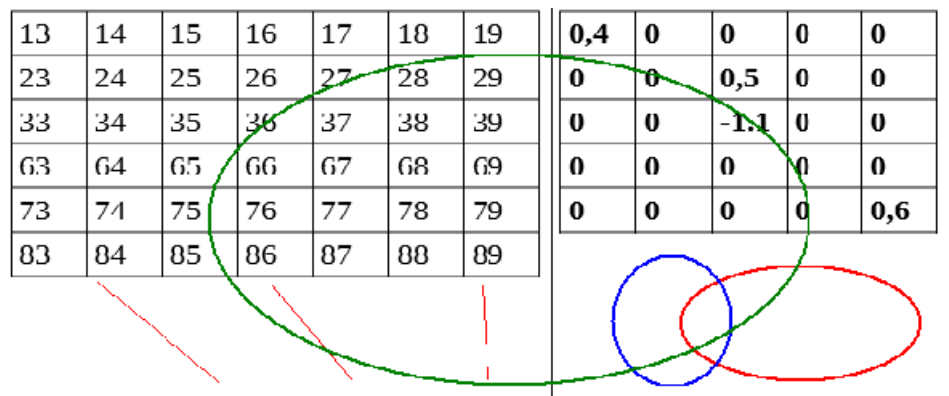
3. Proszę przeskalować algorytmem „najbliższy sąsiad” obrazek „logo\_linuxa” do rozmiaru trochę więcej niż 10 razy większego, na przykład 672x788.



4. Proszę przeskalować algorytmem „najbliższy sąsiad” obrazek „zrzut\_1” do rozmiaru około 1,5x większego, na przykład 900x432.

13	14	15	16	17	18	19	0,4	0	0	0	0
23	24	25	26	27	28	29	0	0	0,5	0	0
33	34	35	36	37	38	39	0	0	-1,1	0	0
63	64	65	66	67	68	69	0	0	0	0	0
73	74	75	76	77	78	79	0	0	0	0	0,6
83	84	85	86	87	88	89					

5. Proszę przeskalować algorytmem „najbliższy sąsiad” obrazek „zrzut\_1” do rozmiaru około 0,75x większego, na przykład 600x246.



6. Proszę przeskalować algorytmem „najbliższy sąsiad” obrazek „zrzut\_1” (bardzo nierównomiernie) do rozmiaru 400x400, a następnie do rozmiaru oryginalnego.

