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
23	24	25	26	27	28	29
33	34	35	36	37	38	39
63	64	65	66	67	68	69
73	74	75	76	77	78	79
83	84	85	86	87	88	89

0,4	**0**	**0**	**0**	**0**
0	**0**	**0,5**	**0**	**0**
0	**0**	**-1.1**	**0**	**0**
0	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0,6**

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.

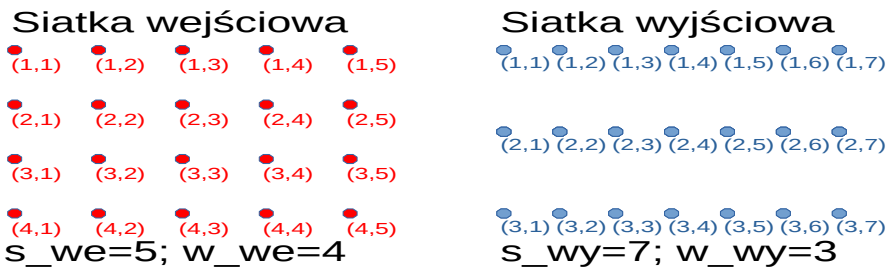
Użyte stałe

- s_wy, w_wy to odpowiednie szerokość i wysokość obrazu wyjściowego,
- s_we, w_we to odpowiednie szerokość i wysokość obrazu wejściowego.

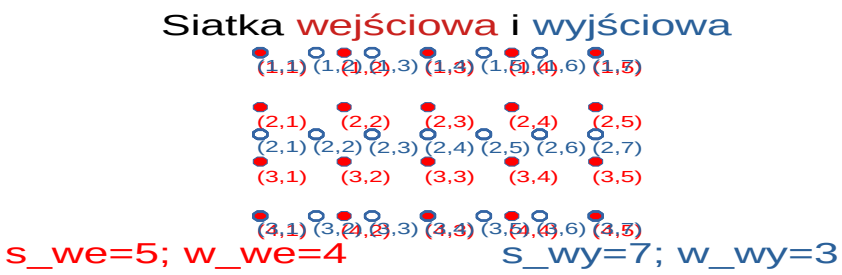
Zakłada się, że $s\_wy, w\_wy, s\_we, w\_we > 1$.

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.



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



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

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

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

gdzie 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 piksela w obrazku wyjściowym obliczyć położenie piksela wejściowego, a następnie go przekopiować.

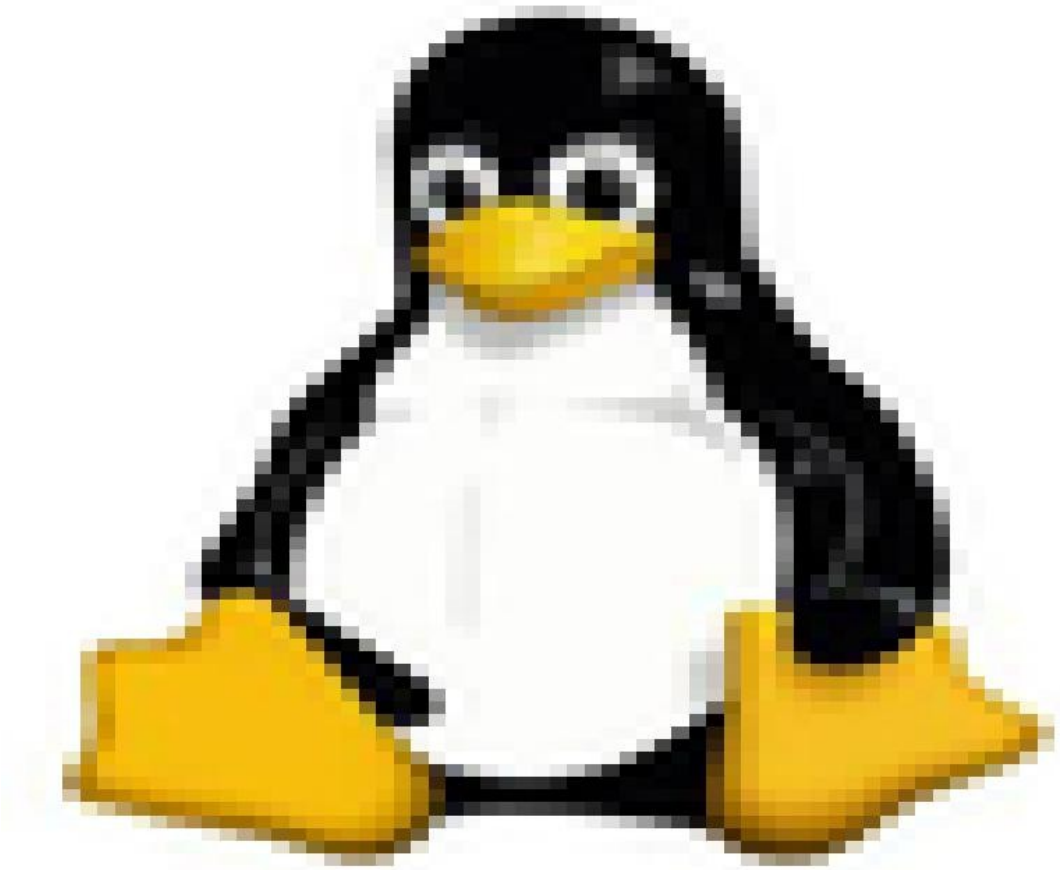
Sugestia

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.

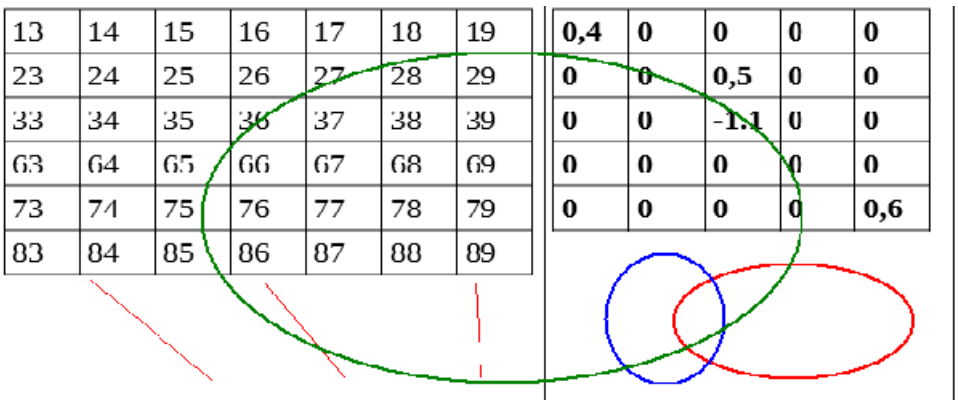


4. Proszę przeskalować algorytmem „najbliższy sąsiad” obrazek „*zrzut_1*” do rozmiaru około 1,5x większego, na przykład 900x432.

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

0,4	**0**	**0**	**0**	**0**
0	**0**	**0,5**	**0**	**0**
0	**0**	**-1.1**	**0**	**0**
0	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0,6**

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.