Wprowadzenie do Grafiki Maszynowej NS
Marcin Głód
164004
grupa II

Laboratorium 2 – sprawozdanie

Zadanie 1.

Napisałem funkcję *rysuj_ramke()* która przyjmuje dwa argumenty: obraz do którego ma dodać ramkę, oraz grubość ramki w pikselach:

```
def rysuj_ramke(obraz, grub):
    tab_obrazu = np.asarray(obraz)
    h, w = tab_obrazu.shape

   tab1 = np.zeros((h,w), dtype=np.uint8)
    for i in range(h):
        for j in range(w):
            if i in range(0,grub):
                tab1[i,j] = 0
            elif i in range(h-grub, h):
                tab1[i,j] = 0

            elif j in range(0,grub):
                tab1[i,j] = 0
            elif j in range(w-grub, w):
                tab1[i,j] = 0
            else:
                tab1[i,j] = tab_obrazu[i,j]

    tab2 = tab1.astype(np.bool_)
    tab_show = Image.fromarray(tab2)
    return tab_show
```

Zadanie 2.

Wynik zastosowania funkcji *rysuj_ramke()* na obrazie z inicjałami z parametrem *grub* ustawionym odpowiednio na 5 i 10 (pikseli):

Zadanie 3.

Napisałem funkcje tworzące obrazy opisane w zadaniu:
Funkcja *rysuj_same_ramki()* tworzy tablicę o wymiarach (w, h), a następnie rysuje czarną ramkę grubości grub po brzegach obrazu który będzie utworzony z tablicy. Następnie odsuwa się o dwukrotność *gurb* od krawędzi i rysuje kojejną ramkę.

```
def rysuj_same_ramki(h, w, grub):
    tab1 = np.ones((h,w), dtype=np.uint8)
    for x in range(0,h,2*grub):
        for i in range(x,h-x):
            for j in range(x,w-x):
```

```
            if i in range(x,grub+x):
                tab1[i,j] = 0
            elif i in range(h-grub-x, h-x):
                tab1[i,j] = 0

            elif j in range(x,grub+x):
                tab1[i,j] = 0
            elif j in range(w-grub-x, w-x):
                tab1[i,j] = 0

    tab2 = tab1.astype(np.bool_)
    tab_show = Image.fromarray(tab2)
    return tab_show
```

Funkcja *rysuj_pionowe()* rysuje czarne pasy grubości *grub* w obrazie o wymiarach (w, h):

```
def rysuj_pionowe(h,w, grub):
    tab1 = np.ones((h,w), dtype=np.uint8)
    for i in range(h):
        for j in range(w):
            if j%(2*grub) >= grub:
                tab1[i,j] = 0

    tab2 = tab1.astype(np.bool_)
    tab_show = Image.fromarray(tab2)
    return tab_show
```

Kolejna funkcja, *wypelnij_dopunktu()* w utworzonej tablicy przypisuje wartość 0 wszystkim punktom o obu wartościach położenia niższych oraz wyższych niż położenie punktu wskazanego jako parametr. Jeżeli tylko jeden parametr jest wyższy, wartość piksela pozostaje jako 1.

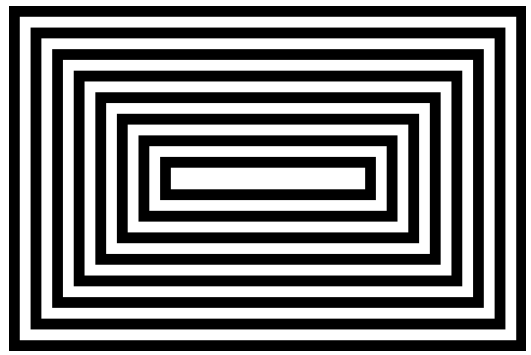
```
def wypelnij_dopunktu(h,w,m,n):
    tab1 = np.ones((h,w), dtype=np.uint8)
    for i in range(h):
        for j in range(w):
            if (i<m and j<n):
                tab1[i,j] = 0
            elif (i>m and j>n):
                tab1[i,j] = 0

    tab2 = tab1.astype(np.bool_)
    tab_show = Image.fromarray(tab2)
    return tab_show
```

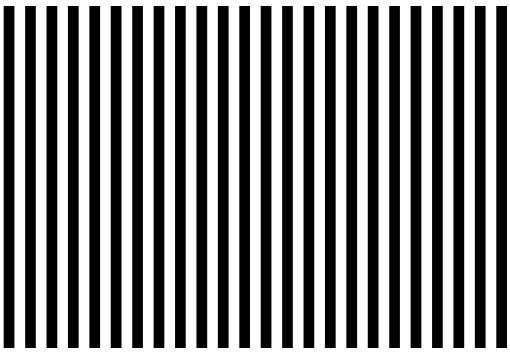
Ostatnia funkcja z tego zadania, rysuj_kola(), korzysta z biblioteki *math* celem policzenia pierwiastka kwadratowego i obliczenia odległości każdego piksela od punktu wskazanego jako parametr. Piksele w odległości równej parametrowi *grubosc* będą w kolorze czarnym, pozostałe piksele przyjmą kolor biały.

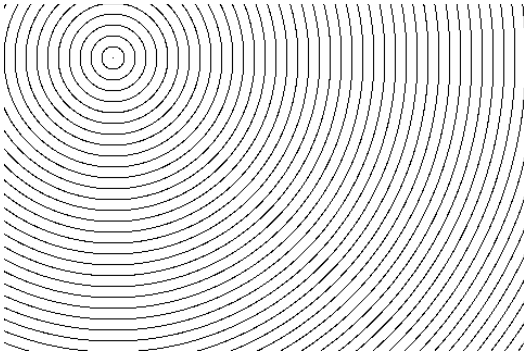
```
def rysuj_kola(h,w,m,n, grubosc):
```

```
    tab1 = np.ones((h,w), dtype=np.uint8)
    for i in range(h):
        for j in range(w):
            tab1[i,j] = math.sqrt((i-n)**2 + (j-m)**2)%grubosc

    tab2 = tab1.astype(np.bool_)
    tab_show = Image.fromarray(tab2)
    return tab_show
```

Zadanie 4.
Wynik działania powyższych funkcji przyjmując parametry w=480, h=320, grub=10, m=100, n=50. Pliki w formacie .bmp dodałem jako załącznik.







Zadanie 5.
Utworzyłem dwie dodatkowe funkcje ukazujące działania możliwe na obrazach w trybie L:

Funkcja *rysuj_kola_l()* tworzy koncentryczne gradienty rozchodzące się od punktu wskazanego jako parametr, ich szerokość możemy regulować parametrem *gestosc:*

```
def rysuj_kola_l(h,w,m,n, gestosc=1):
    tab1 = np.ones((h,w), dtype=np.uint8)
    for i in range(h):
        for j in range(w):
            tab1[i,j] = (math.sqrt((i-n)**2 + (j-m)**2)*gestosc)%256

    tab_show = Image.fromarray(tab1)
    return tab_show
```

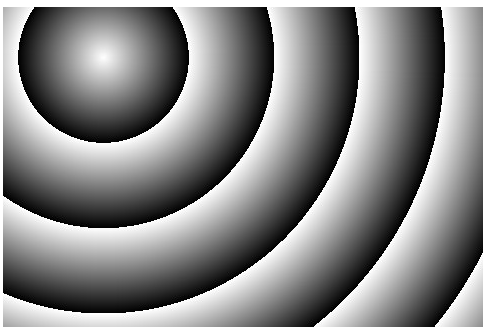
Funkcja *rysuj_pionowe_l()*, analogicznie do funkcji przedstawionej w zadaniu 3. rysuje pionowe pasy grubości *grub*, jednak zamiast jednolitej bieli używa gradientu szarości rozjaśniającego się ku dolnej krawędzi i przeskalowanego tak, by gradient był jednolity przez całą wysokość obrazka.

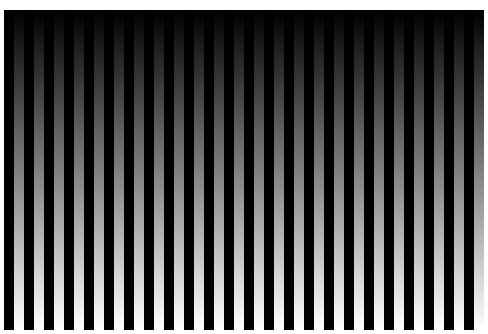
```
def rysuj_pionowe_l(h,w, grub):
    tab1 = np.ones((h,w), dtype=np.uint8)
    for i in range(h):
        for j in range(w):
            if j%(2*grub) >= grub:
                tab1[i,j] = i*256/h

    tab_show = Image.fromarray(tab1)
    return tab_show
```

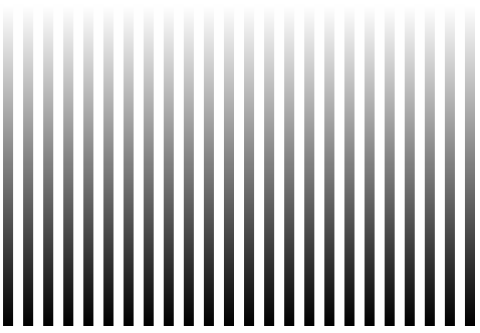
Dodatkowo stworzyłem funkcję *negatyw()* która zamienia wszystkie wartości w tabeli z której złożony zostanie nowopowstały obraz na ich dopełnienia do 255:

```
def negatyw(obrazek):
    tab1 = np.asarray(obrazek)
    h,w = tab1.shape
    tab2 = np.zeros((h,w), dtype=np.uint8)
    for i in range(h):
        for j in range(w):
            tab2[i,j] = 255 - tab1[i,j]

    tab_show = Image.fromarray(tab2)
    return tab_show
```

Nowoutworzone obrazy i ich negatywy:







Zadanie 6.
Funkcje analogiczne do przedstawionych w zadaniu 5, przedstawiające możliwości trybu RGB:

Jako pierwszą utworzyłem funkcję *negatyw_rgb()*, która przekształca przekazany jej obraz na trójwymiarową tablicę, a następnie zamienia wartość każdego kanału (R, G, B) na jego dopełnienie do 255:

```
def negatyw_rgb(obrazek):
    tab1 = np.asarray(obrazek)
    h,w,t = tab1.shape
    tab2 = np.zeros((h,w,t), dtype=np.uint8)
    for i in range(h):
        for j in range(w):
            tab2[i,j,0] = 255 - tab1[i,j,0]
            tab2[i,j,1] = 255 - tab1[i,j,1]
            tab2[i,j,2] = 255 - tab1[i,j,2]

    tab_show = Image.fromarray(tab2)
    return tab_show
```

Następnie utworzyłem funkcję analogiczną do funkcji tworzących pionowe linie w poprzednim zadaniu, jednak tym razem skalując wartości jak następuje:
Kanał czerwony zmienia wartości w pionie od 0 do 255 na całej wysokości obrazu
Kanał zielony zmienia wartości w pionie od 255 do 0 na całej wysokości obrazu
Kanał Niebieski zmienia wartości w poziomie od 0 na lewej krawędzi, do 255 na prawej krawędzi.
Kolor nadawany jest tylko pomiędzy czarnymi pasami grubości *grub.*

```
def rysuj_pionowe_rgb(h,w, grub):
    tab1 = np.ones((h,w,3), dtype=np.uint8)
    for i in range(h):
        for j in range(w):
            if j%(2*grub) >= grub:
                tab1[i,j,0] = i*256/h
                tab1[i,j,1] = 255-(i*256/h)
                tab1[i,j,2] = j*256/w

    tab_show = Image.fromarray(tab1)
    return tab_show
```

Drugą utworzoną funkcją w trybie RGB jest *rysuj_kola_rgb()* która podobnie do wariantu w trybie L rysuje koncentryczny gradient dookoła wskazanego punktu, następnie nadaje całemu obrazowi gradient zielony w pionie (od 0 do 255), oraz niebieski w poziomie (0 do 255).

```
def rysuj_kola_rgb(h,w,m,n, gestosc=1):
    tab1 = np.zeros((h,w,3),dtype=np.uint8)
    for i in range(h):
        for j in range(w):
            tab1[i,j,0] = (math.sqrt((i-n)**2 + (j-m)**2)*gestosc)%256
            tab1[i,j,1] = (i*256)/h
            tab1[i,j,2] = (j*256)/w

    tab_show = Image.fromarray(tab1)
```

```
    return tab_show
```

Kolejną utworzoną funkcją jest funkcja *kolorowe_inicjaly()*, która przyjmuje za argument obraz w trybie 1, a następnie jego czarną część wypełnia kolorowymi pasami, białą część pozostawiając bez zmian.

```
def kolorowe_inicjaly(obraz,grub):
    tab_obrazu = np.asarray(obraz)
    h, w = tab_obrazu.shape

    tab1 = np.zeros((h,w,3), dtype=np.uint8)
    for i in range(h):
        for j in range(w):
            if (tab_obrazu[i,j] == 0):
                if i%(2*grub) <= grub:
                    tab1[i,j,0] = 33
                    tab1[i,j,1] = 255
                    tab1[i,j,2] = 90
                else:
                    tab1[i,j,0] = 200
                    tab1[i,j,1] = 0
                    tab1[i,j,2] = 44
            else:
                tab1[i,j,:] = 255
    tab_show = Image.fromarray(tab1)
    return tab_show
```

Ze względu na różnicę w zapisie pliku, pomiędzy obrazami w formacie .jpg oraz .png występuje różnica: obraz w formacie .jpg jest kompresowany, przez co wartość nasycenia niektórych pikseli może się nieco różnić. Jeżeli przekształcimy oba obrazy do tablic, a następnie odejmiemy od siebie ich wartości będziemy w stanie zauważyć te różnice jako tzw. Artefakty.

Zadanie 7.

Typ uint8, w przypadku gdy przekroczymy wartość 255 przeskoczy z powrotem do wartości 0: bardzo dobrze to widać, gdy zapiszemy sobie tą liczbę w wartości binarnej:
255(dec) = 1111 1111(bin). Gdy dodamy w tym momencie 1, nastąpi przepełnienie i wartość ulegnie zmianie następująco: 256(dec) = 1 0000 0000(bin). Zaznaczona na czerwono jedynka już się nie mieści w przestrzeni przeznaczonej dla wartości uint8, więc zapisana w pamięci pozostanie wartość 0000 0000(bin).
Gdy nadamy wartość 328, zapisana wartość wyniesie 73
Gdy nadamy wartość -1, zapisana wartość wyniesie 255
Gdy nadamy wartość -24, zapisana wartość wyniesie 232

Zadanie 8.

Tak jak opisałem w ostatnim podpunkcie zadania 5. Obrazy w formacie .jpg zostają minimalnie zniekształcone poprzez proces kompresji. Format .jpg poddaje dane kompresji stratnej, „upraszczając” sobie obraz i uzyskując mniej danych potrzebnych do zapisu, przy relatywnie niewielkich stratach z punktu widzenia odbiorcy.