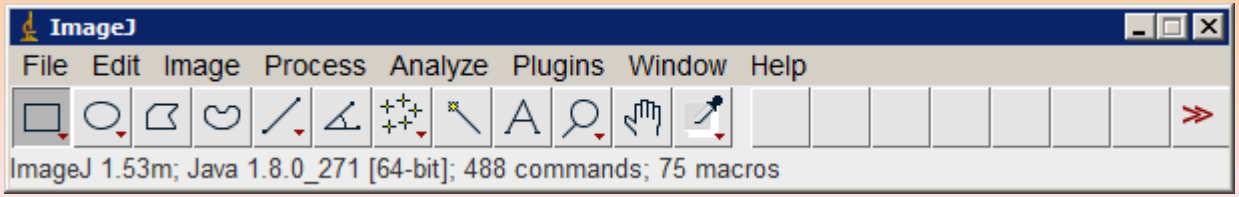
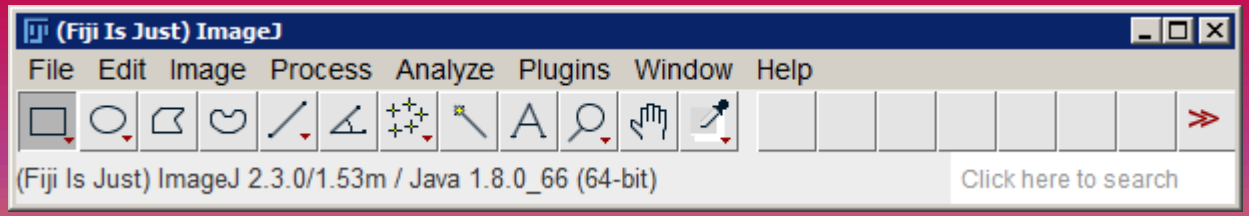
PODSTAWY PRZETWARZANIA OBRAZÓW CYFROWYCH





© 2021,
adam.piorkowski@agh.edu.pl

ImageJ

- sygnowanie: **National Institutes of Health**
- główny autor: Wayne Rasband
- 1997 –
- **Windows / Linux / MacOS / online**
- Java
- Liczne implementacje algorytmów
- Możliwość tworzenia skryptów i wtyczek
- **Program w domenie publicznej**
  (darmowy, można używać komercyjnie)

ImageJ obsługuje

- obrazy: 8 i 16 bit monochrome, float 32, RGB
- odczyt/zapis: liczne formaty,
- pliki tekstowe,
- wizualizacje: wykresy, filmy, montaże, animacje, 3D,
- wszelkie operacje jako pluginy Java,
- wielowątkowość,
- liczne języki skryptowe

Podstawy tworzenia makr

- edytor makr
- nagrywanie czynności (i ich odtwarzanie)
- zmiana parametrów
- obsługa stosów,
- biblioteka dostępnych funkcji
- dynamiczne typowanie
- debugger
- wspomaganie edycji tekstu

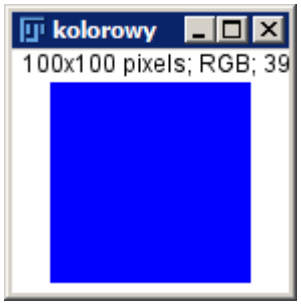
Zasoby

- [100] ImageJ - Image Processing and Analysis in Java https://imagej.nih.gov/ij/
- [101] Fiji: A batteries-included distribution of ImageJ https://fiji.sc/
- [102] Podstawy programowania makr https://imagej.nih.gov/ij/developer/macro/macros.html
- [103] Opisy dostępnych funkcji w makrach https://imagej.nih.gov/ij/developer/macro/functions.html
- [104] Przykłady makr IJM https://imagej.nih.gov/ij/macros/
- [105] Artykuły o ImageJ https://imagej.nih.gov/ij/docs/pdfs/
- [106] Biblioteka MorphoLibJ https://imagej.net/plugins/morpholibj

Gumuła, M., Piórkowski, A.: Wydajna implementacja algorytmów przetwarzania obrazów w postaci wtyczek programowych dla środowiska ImageJ. Studia Informatica, 2017, 38.2: 93--104.
Kopacz, B., Piórkowski, A.: Przegląd możliwości wydajnej implementacji algorytmów przetwarzania obrazów w środowisku MATLAB. Studia Informatica, 2017, 38.3: 177--190

Prosty skrypt – tworzenie obrazka RGB

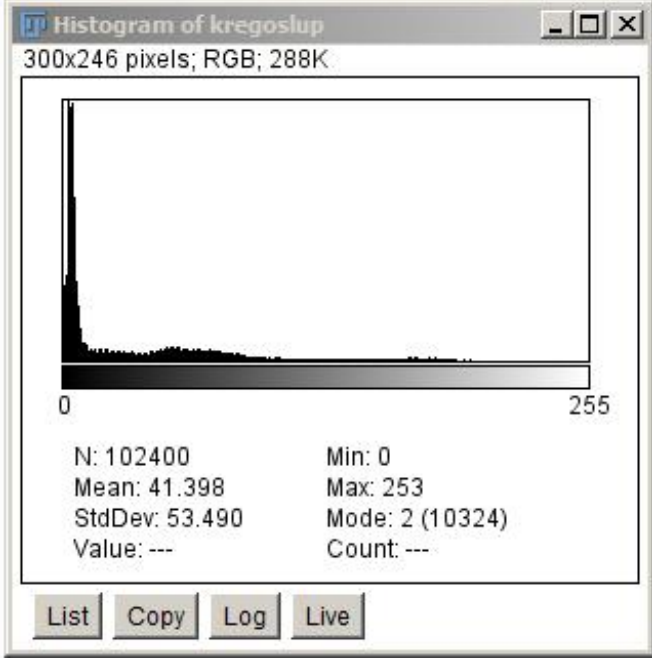
```
// operacje na obrazie kolorowym

close("*");    // zamkniecie wszystkich otwartych okien

w = 100;
h = 100;

newImage("kolorowy", "RGB color", w, h, 1);

for (y=0; y<h; y++)
       for (x=0; x<w; x++)
       {
              kolor = 0;
              kolor += 255;                 // B
              //kolor += 255 * 256;         // G
              //kolor += 255 * 256*256;     // R

              setPixel(x, y, kolor);
       }

              // odczyt wartosci obrazu kolorowego
              // kolor = getPixel(x, y);
              // R = (kolor>>16)&0xFF;
              // G = (kolor>>8)&255;
              // B = (kolor)&255;
```

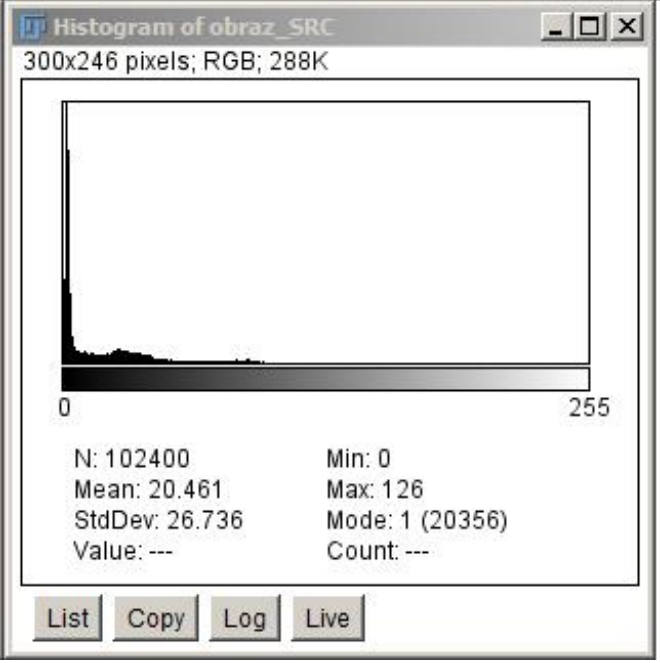


```
// dzialanie na obrazie grayscale 8-bit - kopiowanie do macierzy

open("p.png");
selectWindow("p.png");

w = getWidth();
h = getHeight();
obraz = newArray(w*h);

// zaladowanie obrazkow do tabel

for (y=0; y<h; y++)
        for (x=0; x<w; x++)
                obraz[x + y *w] = getPixel(x, y);

// przetwarzanie danych

for (y=0; y<h; y++)
        for (x=0; x<w; x++)
                obraz[x + y *w] = obraz[x + y *w]/2;

// odtworzenie obrazu

newImage("obraz_nowy", "8-bit", w, h, 1);
for (y=0; y<h; y++)
        for (x=0; x<w; x++)
                setPixel(x, y, obraz[x + y *w] );


// pomocne
newImage("obraz_bialy_na_start", "8-bit white", w, h, 1);
newImage("obraz_czarny_na_start", "8-bit black", w, h, 1);
```


kregoslup.png
320x320 pixels; 8-bit; 100K

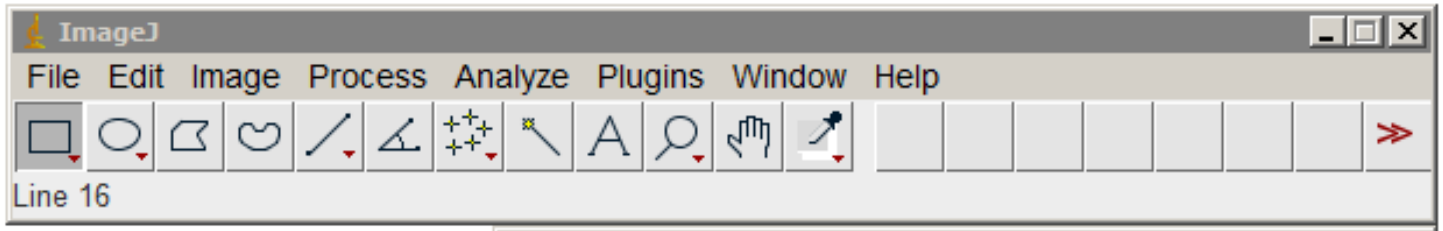

obraz_nowy
320x320 pixels; 8-bit; 100K

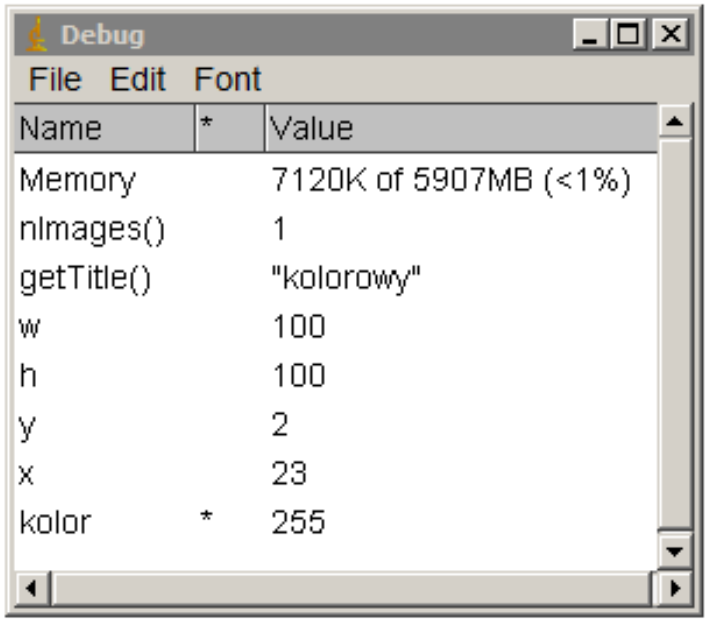

Histogram of kregoslup
300x246 pixels; RGB; 288K
0
255
N: 102400
Mean: 41.398
StdDev: 53.490
Value: ---
Min: 0
Max: 253
Mode: 2 (10324)
Count: ---
List
Copy
Log
Live

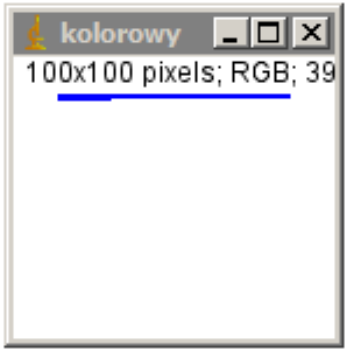

Histogram of obraz_SRC
300x246 pixels; RGB; 288K
0
255
N: 102400
Mean: 20.461
StdDev: 26.736
Value: ---
Min: 0
Max: 126
Mode: 1 (20356)
Count: ---
List
Copy
Log
Live

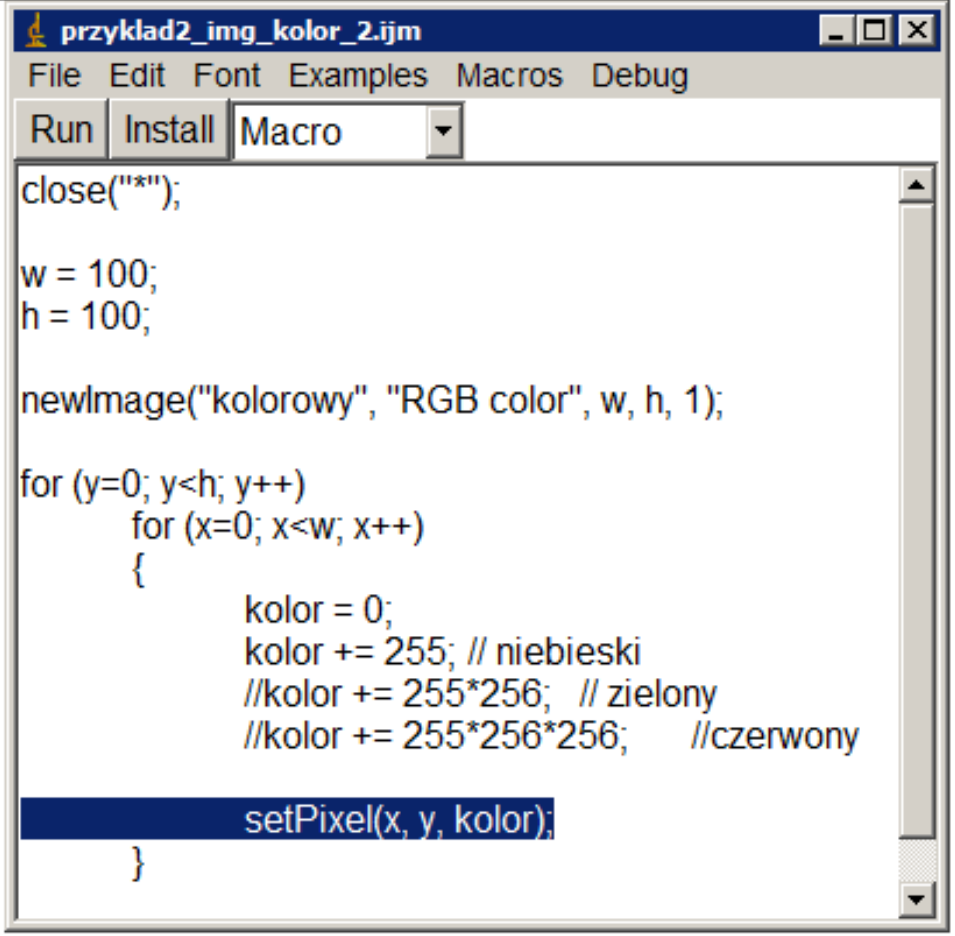
```
// MEAN GRAY VALUE
// wyznaczanie sredniej jasnosci obrazu

print( MGV() );

print( "Srednia jasnosc obrazu glowa.png: " + MGV_okno("glowa.png") );

// MGV dla biezacego obrazu
function MGV()
{
        w = getWidth();
        h = getHeight();

        MGV_wartosc = 0;

        for(i=0;i<w;i++)
                for(j=0;j<h;j++)
                        MGV_wartosc += getPixel(i, j);

        MGV_wartosc /= (w*h);     // oczywiście w*h != 0  !!!  (sprawdzic!)

        return MGV_wartosc;
}

// MGV dla wskazanego obrazu
function MGV_okno(nazwaokna)
{
        selectWindow(nazwaokna);
        return MGV();
}
getStatistics(area, mean, min, max, std, histogram);
```

Podstawy tworzenia makr - **debug**









Tryb debug tylko w ImageJ, w Fiji – brak ☹

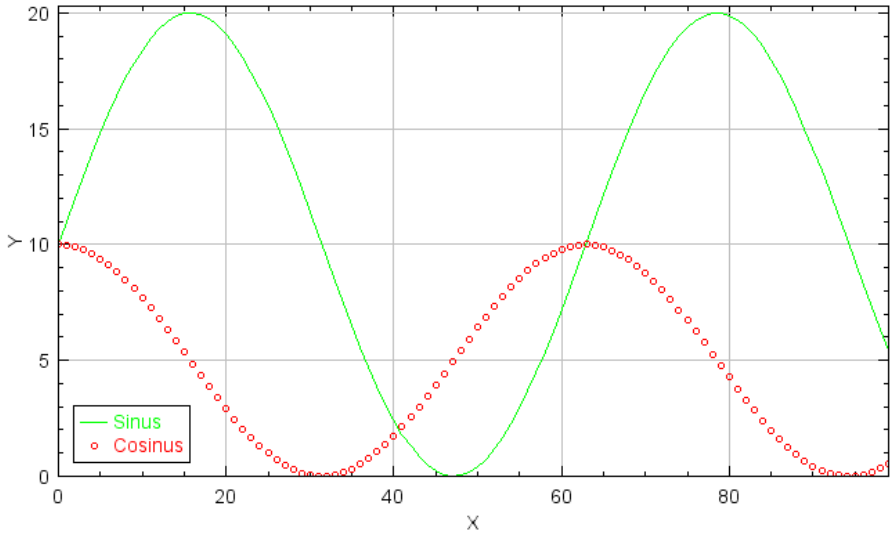
```
// BINARYZACJA GLOBALNA - wtyczka mocno interaktywna ;)
// Image -> Adjust -> Threshold
// Image -> Adjust -> Auto Threshold nie pozwala wprowadzac progu
// UWAGA - okno Adjust->Threshold w nagrywarce dodaje (zakomentowane)
// otworzenie okna Threshold, ale jego zamknięcie 'close();' pozostaje
// odkomentowane !

close("*");

open("D:/PPOC/kregoslup.png");
selectWindow("kregoslup.png");

// binaryzacja z progiem globalnym - ekwiwalent w skrypcie:

setOption("BlackBackground", true);   // (*)
setThreshold(53, 255);                // dolny prog binaryzacji glob.= 53
run("Convert to Mask");               // 'apply', tworzy obraz bin 0/255

// (*) aby tlo bylo 0, a obiekt 255 - czyli biale na czarnym.
// takie podejscie naturalnie wskazuje wartosc tla - kolor 0 (0,0,0)
// bez tego jest uzywany stary tryb - 255=tlo, 0=obiekty + Invert LUT

// pierwotnie w ImageJ obrazy binarne definiowaly: 0 - obiekt, 255 - tlo
// ten tryb jest stosowany nadal w podstawowych funkcjach ImageJ
// a nowsze biblioteki oferuja mozliwosc definiowania czy obiekty sa
// biale czy czarne

// (*) Ustawienie w ImageJ: Process->Binary->Options… 'Black background'
```

```
// METODY PRZEKAZYWANIA PARAMETRÓW DO WYWOŁAŃ SKRYPTÓW I WTYCZEK
// binaryzacja algorytmem lokalnym (promien otoczenia, jeden lub 2 parametry)
// Image -> Adjust -> Auto Local Threshold        mozna wprowadzac promien i 2 parametry

// nagranie "RECORD„
run("Duplicate...", "title=MidGrey_15");
selectWindow("MidGrey_15");
run("Auto Local Threshold", "method=MidGrey radius=15 parameter_1=1 parameter_2=2 white");

// przykladowe wywolania w skrypcie

promien = 30;                  // podanie parametru jako nazwa

selectWindow("kregoslup.png");
run("Duplicate...", "title=MidGrey_30");
selectWindow("MidGrey_30");
run("Auto Local Threshold", "method=MidGrey radius=promien parameter_1=1 parameter_2=2 white");

promien = 50;                  // podanie parametru przez &

selectWindow("kregoslup.png");
run("Duplicate...", "title=MidGrey_50");
selectWindow("MidGrey_50");
run("Auto Local Threshold", "method=MidGrey radius=&promien parameter_1=1 parameter_2=2 white");

promien = 80;                  // konstrukcja tekstu parametrow

selectWindow("kregoslup.png");
run("Duplicate...", "title=MidGrey_80");
selectWindow("MidGrey_80");
run("Auto Local Threshold", "method=MidGrey radius=" + promien + " parameter_1=1 parameter_2=2
white");

// z ciekawosci
print("method=MidGrey radius=" + promien + " parameter_1=1 parameter_2=2 white");
// wyswietla: method=MidGrey radius=80 parameter_1=1 parameter_2=2 white
```

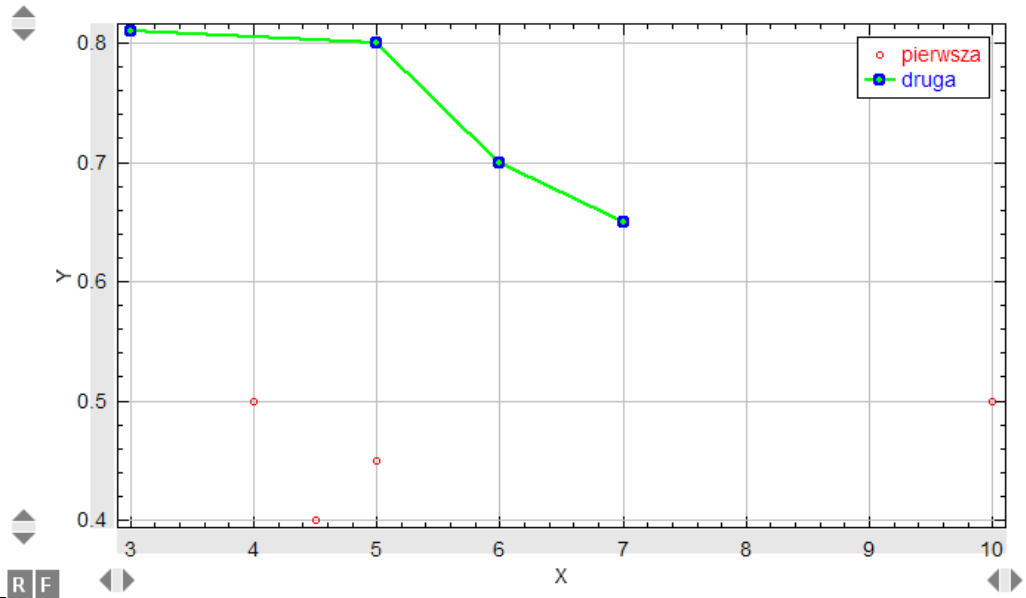
```
                                    // UZYTECZNE FUNKCJE
// kopia obrazka
run("Duplicate...", "title=IMG_SRC");

// zmiana nazwy (np. wyniku po binaryzacji)
rename("IMG_BIN");

// uruchomienie MorphoLibJ -> Analyze -> Label Overlap Measures
run("Label Overlap Measures", "source=IMG_SRC target=IMG_BIN dice");

wspDice = 0;

selectWindow("IMG_BIN-all-labels-overlap-measurements");

if(!isNaN(getResult("DiceCoefficient", 0) )) // sprawdzenie, czy sa wyniki
    wspDice = getResult("DiceCoefficient", 0);       // pobranie wyniku

print(wspDice);                  // wyswietlenie go

close("IMG_SRC*");               // zamkniecie okien o nazwach IMG_SRC*

selectWindow("Log");             // zapis logu do pliku tekstowego

saveAs("Text", "plik.txt");
//print("\\Clear");              // wyczyszczenie logu

saveAs("PNG", "nazwa.png");      // zapis obrazka

// nazwa pliku bez rozszerzenia
nowy = File.getNameWithoutExtension(zmienna_tekstowa_z_adresem_pliku);
saveAs("PNG", nowy + ".png"); // przy uzyciu zmiennej tekstowej
```

```
// uzyteczne funkcje  cd.

close("*");              // sprzatanie okien na poczatek
print("\\Clear");        // wyczyszczenie okna logu

sciezka_pliku = File.openDialog("obraz wzorcowy");
open(sciezka_pliku);

// program wczytujacy liste plikow, otwierajacy te z rozszerzeniem tif
// i zapisujacy do png (w tym samym katalogu)

katalog_plikow = getDirectory("Katalog plikow");

lista_plikow_katalogu = getFileList(katalog_plikow);

rozszerzenie_pliku = ".tif";

// przepisanie tifow do png
for (i=0; i< lista_plikow_katalogu.length; i++)
        if (endsWith(lista_plikow_katalogu[i],  rozszerzenie_pliku))
        {
                showProgress(i+1, lista_plikow_katalogu.length);

                nowy_plik = replace(lista_plikow_katalogu[i],
                                rozszerzenie_pliku, "") + ".png";

                open(katalog_plikow + lista_plikow_katalogu[i]);
                saveAs("PNG", katalog_plikow + nowy_plik);
                close();
        }
```

```
// przyklady wykresow

var N=100;      // var - zmienna globalna

tab_sin = newArray(N);
tab_cos = newArray(N);

for(i=0;i<N;i++)
  {
    tab_sin[i] = 10 + 10*sin(i/10);
    tab_cos[i] = 5 + 5*cos(i/10);
  }

Plot.create("Przyklad", "X", "Y");
Plot.setColor("Green");
Plot.add("line", tab_sin);

Plot.setColor("Red");
Plot.add("dash", tab_cos);

Plot.addLegend("Sinus\nCosinus");

Plot.show();
```



```
// przyklady wykresow

tabA_X = newArray(4, 4.5, 5, 10);
tabA_Y = newArray(0.5, 0.4, 0.45, 0.5);

tabB_X = newArray(3, 5, 6, 7);
tabB_Y = newArray(0.81, 0.8, 0.7, 0.65);

// wykres XY
Plot.create("Wykres_XY", "X", "Y");

Plot.setColor("Red");
Plot.add("dash", tabA_X, tabA_Y);

Plot.add("dash", tabB_X, tabB_Y);
Plot.setStyle(1, "blue,green,2.0,Connected Circles");

Plot.addLegend("pierwsza\ndruga")

Plot.setLimitsToFit();
```



```
// przykład wlasnej funkcji, generujacej losowy kolor
// tekst zawierajacy liczby szesnastkowe, np. #80FF00
// przykladowe zastosowanie: do wykresow

// przykład uzycia
Plot.setColor( randomcolortextint(100) );


// definicja funkcji (najlepiej na koncu pliku)
// parametr s: liczba startowa generatora liczb losowych (ang. seed)
// pozwala to na powtarzalnosc losowanych kolorow, np. w petli

function randomcolortextint(s)
{
      chartab = newArray("0", "1", "2", "3", "4", "5",
            "6", "7", "8", "9", "a", "b", "c", "d", "e", "f");

      kolor = "#";

      for (i=0;i<6;i++)
            kolor += "" +  chartab[ (s*701 + 4969*i)%16 ];

      return kolor;
}
```

```
// przykladowe przetwarzanie tekstu w ImageJ

sciezka = File.openDialog("DANE txt");      // okno dialogowe - wybor pliku
katalog = File.getDirectory(sciezka);
plik = File.getName(sciezka);

nazwa_wynikow = File.getNameWithoutExtension(plik)+ "_przetworzone.txt";

separator_linii = "\n";
separator_kolumn =  " "; //"\t";
minimalna_liczba_kolumn = 3;
kolumna_x = 0;
kolumna_y = 1;
kolumna_v = 2;

filestring=File.openAsString(katalog+plik);    // załadowanie pliku do stringu
rows=split(filestring, separator_linii);       // rozdzielenie na tablicę linii

x = newArray(rows.length);
y = newArray(rows.length);
v = newArray(rows.length);

for(i=0; i<rows.length; i++)                          // zaladowanie danych do tabel
{
        columns=split(rows[i],separator_kolumn);    // rozdzielenie na kolumny
        if(columns.length < minimalna_liczba_kolumn)
                continue;
        //x[i]=parseInt(columns[kolumna_x]);
        x[i]=parseFloat(columns[kolumna_x]);
        y[i]=parseFloat(columns[kolumna_y]);
        v[i]=parseFloat(columns[kolumna_v]);
}
```

. . .

```
// przykladowy program zamieniający format csv w ImageJ
// csv z liczbami z kropkami, rozdzielane przecinkiem
// zamieniane na csv z liczbami z przecinkami, rozdzielane srednikiem


// csv z liczbami z kropkami, rozdzielane przecinkiem


plik_txt = File.openDialog ("PLIK_csv");

danetxt = File.openAsString(plik_txt); // załadowanie pliku do stringu
danetxt = replace(danetxt, ",", ";");  // zamiana przecinkow na sredniki
danetxt = replace(danetxt, ".", ",");  // zamiana kropek na przecinki

File.saveString(danetxt, File.getDirectory(plik_txt) + "//" +
                    File.getNameWithoutExtension(plik_txt) + "_p.csv");
```