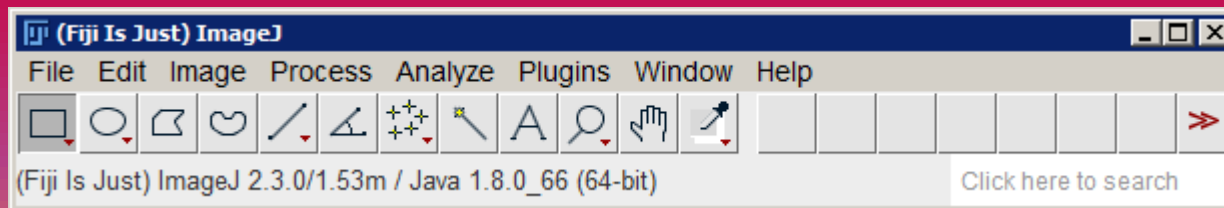
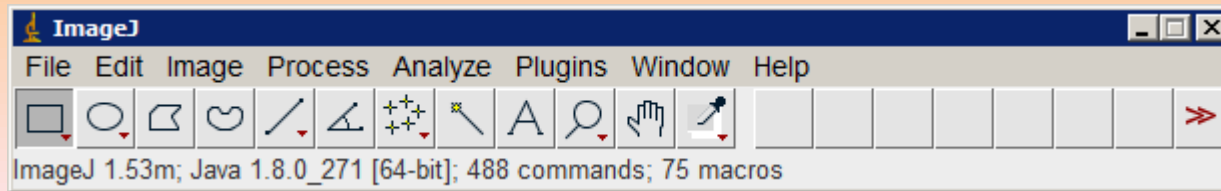


PODSTAWY PRZETWARZANIA OBRAZÓW CYFROWYCH



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, montaż, 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>

Prosty skrypt – tworzenie obrazka RGB

```
// operacje na obrazie kolorowym

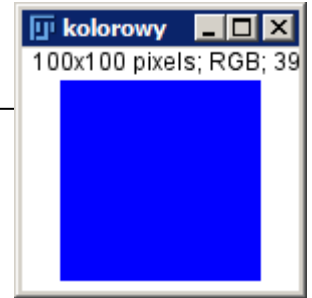
close("*"); // zamkniecie wszystkich otwartych okien

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);
    }
```



```
// 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
```

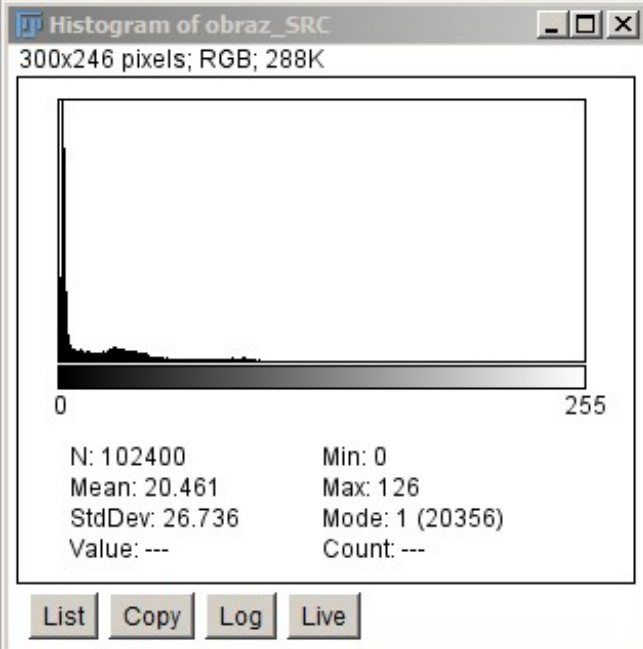
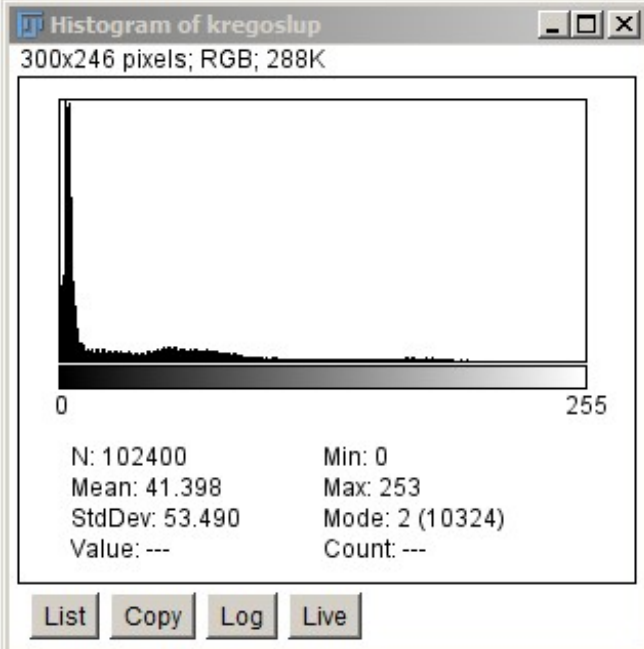
```
for (y=0; y<h; y++)  
    for (x=0; x<w; x++)  
        obraz[x + y *w] = getPixel(x, y);
```

```
// 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] );
```




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

print( MGV() );

print( "Srednia jasnoc 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();
}
```

Podstawy tworzenia makr - debug

The screenshot displays the ImageJ interface during macro execution. The main window 'ImageJ' has a menu bar (File, Edit, Image, Process, Analyze, Plugins, Window, Help) and a toolbar. The 'Debug' window is open, showing a table of variables:

Name	*	Value
Memory		7120K of 5907MB (<1%)
nImages()		1
getTitle()		"kolorowy"
w		100
h		100
y		2
x		23
kolor	*	255

The macro editor window 'przyklad2_img_kolor_2.ijm' shows the following code:

```
close("");  
  
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; // niebieski  
    //kolor += 255*256; // zielony  
    //kolor += 255*256*256; //czerwony  
  
    setPixel(x, y, kolor);  
  }
```

The 'kolorowy' window shows the image dimensions: 100x100 pixels; RGB; 39.

Tryb debug tylko w ImageJ, w Fiji – brak ☹

```
// BINARYZACJA GLOBALNA - wtyczka mocno interaktywna ;)
// Image -> Adjust -> Threshold
// Image -> Adjust -> Auto Threshold nie pozwala wprowadzac prog
// UWAGA - okno Adjust->Threshold w nagrywarcie 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 (promień otoczenia, jeden lub 2 parametry)
// Image -> Adjust -> Auto Local Threshold          można wprowadzać promień 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");

// przykładowe wywołania 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 -> Analize -> 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();
    }
}
```

```

// przykłady 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);
}

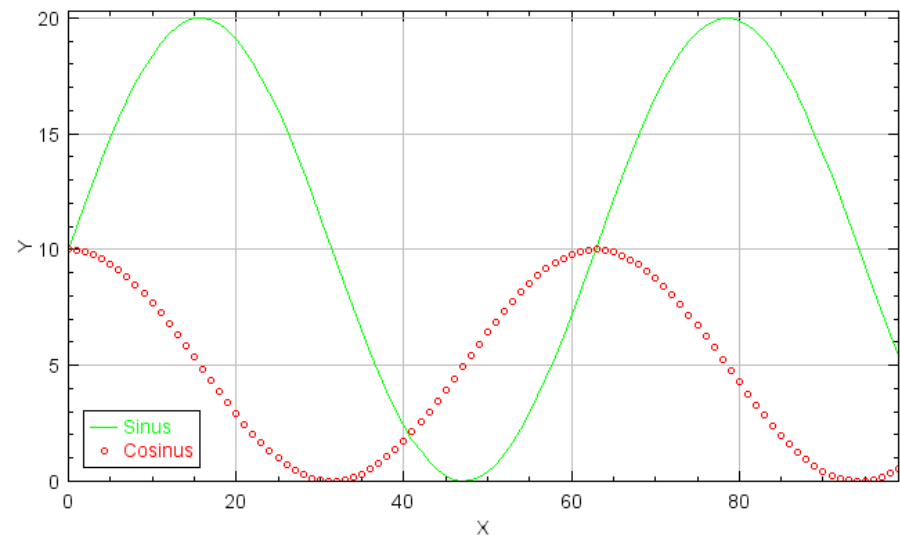
Plot.create("Przyklad", "X", "Y");
Plot.setColor("Green");
Plot.add("line", tab_sin);

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

Plot.addLegend("Sinus\nCosinus");

Plot.show();

```



```
// przykład własnej funkcji, generującej losowy kolor
// tekst zawierający liczby szesnastkowe, np. #80FF00
// przykładowe zastosowanie: do wykresów
```

```
// przykład użycia
```

```
Plot.setColor( randomcolortextint(100) );
```

```
// definicja funkcji (najlepiej na końcu pliku)
```

```
// parametr s: liczba startowa generatora liczb losowych (ang. seed)
```

```
// pozwala to na powtarzalność losowanych kolorów, 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;
```

```
}
```


// przykładowe 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++)                    // załadowanie 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]);
}
```