# BÀI 1: - LÀM QUEN VỚI MALTAB
# - ĐỌC VÀ HIỂN THỊ ẢNH TRONG MATLAB

## I. LÀM QUEN VỚI MÔI TRƯỜNG LẬP TRÌNH MATLAB

### 1. Khởi động MATLAB

### 2. Thay đổi Layout hiển thị, làm quen với các thành phần trên giao diện MATLAB.

### 3. Thay đổi thư mục hiện hành (current folder) của MATLAB.

### 4. Làm việc với Command Window

Command Window được dùng cho việc nhập các biến, chạy một hàm và các script M-file. Phím mũi tên lên (↑) được dùng cho việc gọi lại các lệnh đã được nhập trước đó. Sau khi gọi lại chúng ta có thể điều chỉnh lệnh đó và nhấn enter để chạy nó.

#### *4.1 Các thao tác cơ bản trong Command Window*

- Tạo một row vector với các số 1,2,3,4,5 và gán nó vào cho biến ‘x’

```
» x = [1, 2, 3, 4, 5]
 x =

     1   2   3   4   5
```

- Tạo một column vector với các số 6,7,8,9 và gán nó vào cho biến ‘y’

```
» y = [6; 7; 8; 9]
 y =

     6
     7
     8
     9
```

- Tạo một row vector từ 0,1,2,3,4,5,6,7,8 đơn giản bằng cách chỉ định giá trị bắt đầu và kết thúc dãy số:

```
» a = [0:8]
  a =

     0   1   2   3   4   5   6   7   8
```

- Tạo một row vector theo thứ tự tăng và bước tăng bằng 2:

```
» u = [0:2:8]
  u =

     0   2   4   6   8
```

- Tạo một row vector thứ tự giảm và bước giảm bằng 2:

```
» u = [12:-2:0]
  u =

    12   10   8   6   4   2
```

- Thực hiện phép cộng

```
» u = 12+14
 ans =

     26
```

- Thực hiện phép trừ

```
» u = 12-14
 ans =

    -2
```

### *4.2 Các lệnh điều khiển trong MATLAB*

***Lệnh if***

Lệnh **if** được dùng cho việc đánh giá biểu thức luận lý và thực thi một nhóm các lệnh chỉ khi điều kiện cho kết quả *true*; **elseif** và **else** được dùng để thực thi nhóm lệnh còn lại khi điều kiện cho kết quả *false.*

```
» if a > b
      fprintf (‘greater);
  elseif a == b
           fprintf( ‘equal’ );
    elseif a < b
         fprintf (‘less’);
Else
       fprintf(‘ error’);
end
```

***Lệnh switch…case***

Trong lệnh **switch** nhóm lệnh được thực thi dựa trên giá trị của biến hay biểu thức trong **case**

```
» x = input(‘Enter the no: ‘);

switch x
  case 1
    disp('the number is negative')
  case 2
    disp('zero')
  case 3
    disp('the number is positive')
  otherwise
    disp('other value')
end
```

***Vòng lặp for***

Vòng lặp **for** lặp lại nhóm lệnh với số lần lặp xác định.

Cú pháp của vòng lặp for như sau:

```
for index = values
   statements
end
```

```
» for x = [2, 1, 3, 4, 5]
disp (x)
end
     2

     1

     3

     4

     5
```

***Vòng lặp while***

Khi điều kiện chỉ định trong **while** cho kết quả true, **while** sẽ thực thi lặp lại nhóm lệnh trong thân của nó.

Cú pháp của vòng lặp while như sau:

```
while expression
    statements
end
```

```
» x = 2;
while( x< 18 )
fprintf('value of x: %d\n', x);
 x = x + 1;
 end
```

***Lệnh break***

Lệnh **break** chỉ được dùng để thoát khỏi vòng lặp while hoặc for. Khi các vòng lặp lồng nhau, break cho phép chương trình thoát khỏi vòng lặp chứa nó.

```
» x = 2;
 while (x < 12)
    fprintf('value of x: %d\n', x);
   x = x+1;
      if(‘ x > 7’)
        break;
     end
end
```

***Lệnh continue***

Lệnh **continue** được dùng trong các vòng lặp. Điều khiển chương trình nhảy đến đầu vòng lặp cho lần lặp tiếp theo bằng cách bỏ qua việc thực thi các lệnh trong thân vòng lặp trong lần lặp hiện tại.

```
» x = 2;

  while(x<12)
     if x == 7
       x = x+1;
       continue;

   end
  fprintf('value of x: %d\n', x);
   x = x+1;
end
```

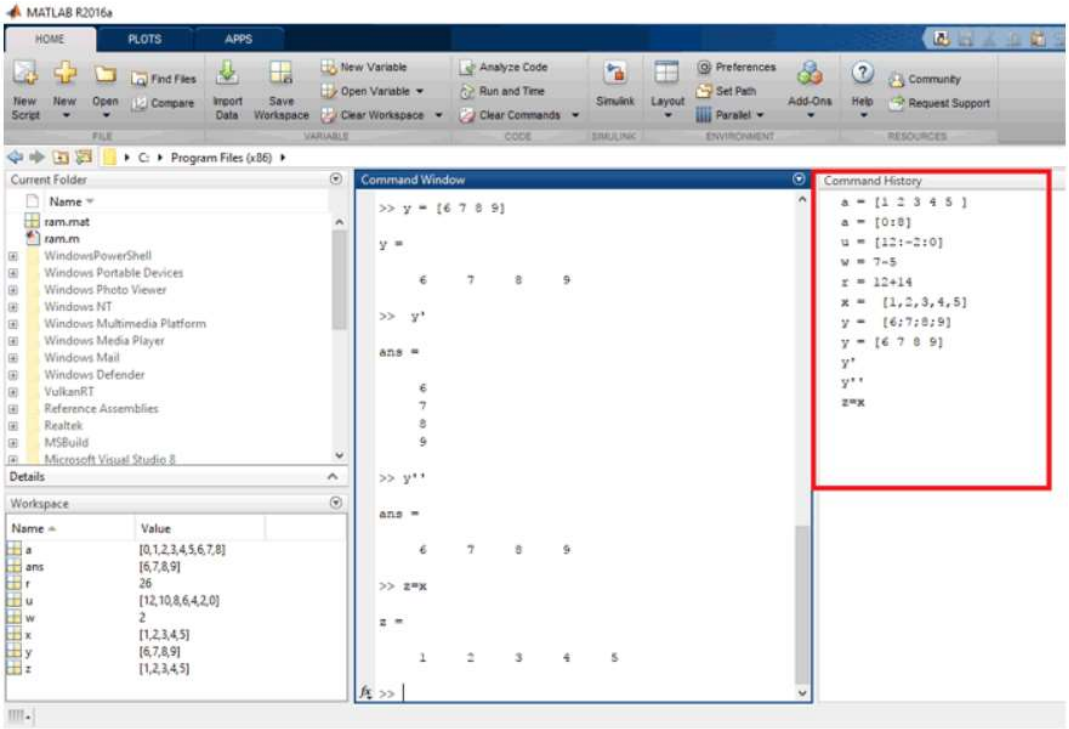
## 5. Thao tác với Command History

Command History nghĩa là lịch sử của cửa sổ lệnh (Command Window).

Các hàm hoặc các dòng được nhập trong Command Window sẽ hiển thị trong Command History.

Chúng ta có thể chọn bất kỳ một hàm hay dòng nào trong Command History để thực thi nó.

Chúng ta cũng có thể tạo một M-file từ các lệnh đã chọn. M-file là chỉ làm một tập tin văn bản có chứa mã nguồn MATLAB.




## 6. Thao tác với Editor

Editor là một bộ xử lý văn bản được thiết kế đặc biệt cho việc tạo và debug các M-file. Một M-file bao gồm một hoặc nhiều câu lệnh để thực thi. Sau khi lưu M-file, chúng ta có thể gọi trực tiếp nó bằng cách đánh tên file vào Command Window.

### *6.1 M-file chứa hàm*

- Một M-file chứa một hàm với các thành phần như sau:

***Dòng định nghĩa hàm (bắt đầu 1 hàm):***

```
function [y1,...,yN] = myfun(x1,...,xM)
```

Khai báo một hàm với tên là myfun chấp nhận các input x1,…, xM và trả về các output y1,…, yN

+ Từ khóa function không thể thiếu

+ Tên hàm:

- có thể chứa chữ cái, số hoặc dấu gạch dưới
- bắt đầu bằng một chữ cái

***Dòng H1:***

+ Dòng ghi chú đơn đầu tiên liền sau dòng định nghĩa hàm

+ Không có bất kỳ dòng trống hoặc khoảng trống ở đầu giữa dòng H1 và dòng định nghĩa hàm

+ Cung cấp những thông tin tổng hợp quan trọng về nội dung và mục đích của M-file, nó nên được mô tả rõ rang nhất có thể

***Help text:*** khối các văn bản theo sau dòng H1, không có khoảng trắng giữa Help text và H1

***Thân hàm:*** chứa tất cả code MATLAB thực hiện tính toán hoặc gán các giá trị cho các tham số output

***Ghi chú*** *trong MATLAB được chỉ định bắt đầu bởi dấu %*


### *6.2 Ví dụ về M-file chứa hàm*

- Tạo tập tin script soạn thảo mã nguồn mới bằng cách chọn:
- Trong tập tin script vừa tạo trong cửa sổ Editor: nhập nội dung như sau:

```
function z = raise_to_power(val, exp)
%RAISE_TO_POWER tính lũy thừa
%z=raise_to_power(val, exp) lấy lũy thừa exp của val và gán vào z
z=val^exp;
```

- Lưu tập tin với tên **raise_to_power.m**
- Trong Command Window, kiểm tra kết quả của raise_to_power:

```
>> x=2
x =
     2
>> y=5
y =
     5
>> raise_to_power(x,y)
ans =
    32
```

## II. ĐỌC VÀ HIỂN THỊ ẢNH BẰNG LỆNH

### 1. Image Processing Toolbox (IPT)

- Image Processing Toolbox (IPT) là một tập các hàm mở rộng khả năng cơ bản của môi trường MATLAB cho phép các hoạt động xử lý tín hiệu và hình ảnh chuyên biệt:

+ Biến đổi không gian
+ Phân tích và tang cường ảnh
+ Các hoạt động khối và vùng lân cận
+ Lọc tuyến tính và thiết kế bộ lọc
+ Các biến đổi toán học

+ Làm nét (deblurring)
+ Xử lý hình thái
+ Xử lý ảnh màu

- Hầu hết các hàm của toolbox đều là MATLAB M-files, code của chúng có thể được thấy được bằng cách mở file sử dụng MATLAB editor hoặc đơn giản là gõ `type function_name` tại dấu nhắc trong Command Window

- Để xem thư mục lưu trữ các M-file, có thể gõ `which function_name` tại dấu nhắc trong Command Window

- Để biết phiên bản của IPT đã cài đặt, gõ `ver` tại dấu nhắc trong Command Window

**2. Tập dữ liệu ảnh**

- Thao tác với tập ảnh B1_images.
- Sao chép tập ảnh B1_images vào thư mục làm việc hiện thời của MATLAB.

**3. Hiển thị thông tin tập tin ảnh với hàm `imfinfo`**

- Cú pháp: `info = imfinfo(filename)`
- Ý nghĩa: trả về một cấu trúc với các trường chứa thông tin về hình ảnh trong tập tin đồ họa được chỉ định (filename), bao gồm định dạng tập tin. Với các định dạng ảnh khác nhau sẽ trả về các trường thông tin khác nhau.
- Có thể truy xuất các trường dữ liệu sau thông qua tên biến (info):

| | | |
|---|---|---|
| + `Filename` | + `Width` | + `NumberOfSamples` |
| + `FileModDate` | + `Height` | + `CodingMethod` |
| + `FileSize` | + `BitDepth` | + `CodingProcess` |
| + `Format` | + `ColorType` | + `Comment` |
| + `FormatVersion` | + `FormatSignature` | |

- Ví dụ:

```
>> info=imfinfo('pout.jpg');
>> info

info =

  struct with fields:

          Filename: 'D:\Giangday\20_21_HK2\Chinhquy\imageprocessing\DemoMATLAB\pout.jpg'
       FileModDate: '08-Apr-2021 14:46:33'
          FileSize: 6548
            Format: 'jpg'
     FormatVersion: ''
             Width: 240
            Height: 291
          BitDepth: 8
         ColorType: 'grayscale'
   FormatSignature: ''
   NumberOfSamples: 1
      CodingMethod: 'Huffman'
     CodingProcess: 'Sequential'
           Comment: {}
```

```
>> info.CodingMethod

ans =

    'Huffman'
```

**Yêu cầu:**

Sử dụng hàm `imfinfo` xem thông tin các tập tin ảnh có trong tập ảnh B1_images. Ghi nhận sự khác biệt về thông tin giữa các định dạng ảnh khác nhau.

### 4. Đọc ảnh với hàm `imread`

- Cú pháp: `img = imread(filename)`
- Ý nghĩa: đọc ảnh từ tập tin được chỉ định bằng `filename`, bao gồm định dạng của tập tin. Nếu `filename` là một tập tin nhiều ảnh, `imread` sẽ đọc ảnh đầu tiên trong tập tin
- Ví dụ:

```
>> img= imread('B1_images\pout.tif')

img =

  291×240 uint8 matrix

  Columns 1 through 17

   107   108   107   106    99   101   102   107   107   103   112   164   200   197   170   182   213
   109   106   108   107   103   102   103   110   113   108   128   182   210   210   182   166   189
   107   106   110   110   106   107   107   120   133   133   125   151   181   189   175   151   159
   106   107   108   108   108   108   108   114   126   146   133   146   142   143   143   133   133
   105   108   109   109   108   106   107   110   113   132   126   128   139   146   157   153   137
   105   108   109   110   108   108   109   109   109   117   121   109   145   174   202   206   189
   108   109   109   109   108   109   110   109   109   110   112   110   138   175   211   215   214
   107   107   108   108   107   110   110   109   110   108   109   110   142   178   214   215   213
```

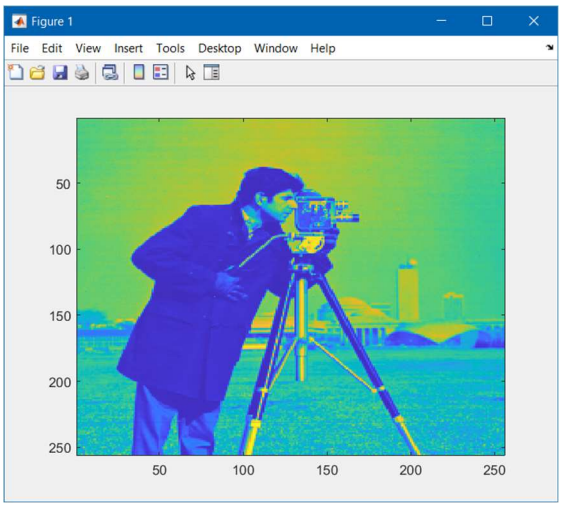
### 5. Hiển thị ảnh với các hàm `image`, `imagesc` và `imshow`

#### *5.1 Hàm* `image`

- Hàm `image` với cú pháp `image(C)` hiển thị dữ liệu trong mảng C như một hình ảnh. Mỗi thành phần của C chỉ định màu cho 1 pixel trong ảnh. Ảnh kết quả của image là một lưới m-by-n các pixel trong đó m là số dòng và n là số cột của C. Chỉ số dòng và chỉ số cột xác định các tâm pixel tương ứng.
- Kết quả thu được có thể xuất hiện hỗn hợp màu sắc kì dị và bị kéo giãn một chút. Các màu sắc kỳ lạ này xuất hiện là do thực tế `image` sử dụng color map hiện tại để gán màu xác cho các thành phần ma trận. Color map mặc định là **jet**, bao gồm 64 màu rất sáng, không phù hợp cho việc hiển thị ảnh đa cấp xám.
- Ví dụ:

```
>> img=imread("B1_images\cameraman.tif");
>> image(img);
```

Kết quả thu được:




- Để hiển thị hình ảnh đúng cách chúng ta cần thêm một số lệnh bổ sung cho dòng lệnh image như sau:

+ **`truesize`**: hiển thị một phần tử ma trận (1 pixel) tương ứng với một pixel màn hình. Chính thức hơn, chúng ta có thể sử dụng truesize([256 256]) với các thành phần vector cho biết số lượng pixel màn hình theo chiều dọc và chiều ngang được sử dụng để hiển thị. Nếu vector không được chỉ định, mặc định số lượng pixel tương ứng với kích thước ảnh.

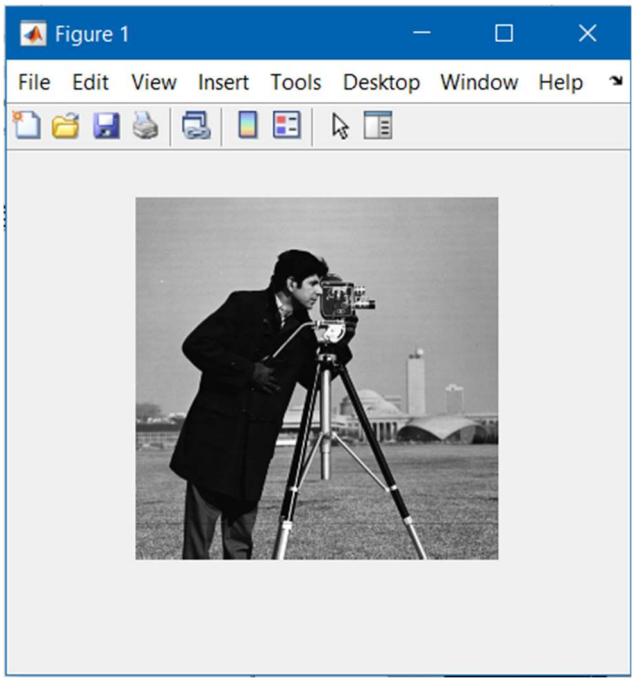
+ **`axis off`**: tắt nhãn các trục.

+ **`colormap(gray(<số lượng mức xám>))`**, điều chỉnh color map để chỉ sử dụng các sắc thái của màu xám. Chúng ta có thể tìm thấy số lượng mức xám đã dùng trong ảnh bằng lệnh `size(unique(img))`. Ví dụ, với ảnh cameraman. tif có 247 mức xám khác nhau được dùng, vì vậy chúng ta chỉ cần điền số mức xám này vào color map.

- Ví dụ: lệnh đầy đủ để hiển thị chính xác ảnh cameraman.tif

```
>> image(img), truesize, axis off, colormap(gray(247));
```

Kết quả thu được:



*- Lưu ý, chúng ta có thể thu được các ảnh tối hoặc sáng hơn bằng cách thay đổi giá trị số lượng mức xám sử dụng để hiển thị ảnh trong color map.*

- Hàm `image` làm việc tốt với các ảnh màu được lập chỉ mục, miễn là chúng ta nhớ sử dụng `imread` để chọn color map (có thể xác định ảnh màu được lập chỉ mục trong trường thông tin `ColorType` của ảnh, dùng hàm `imfinfo`).

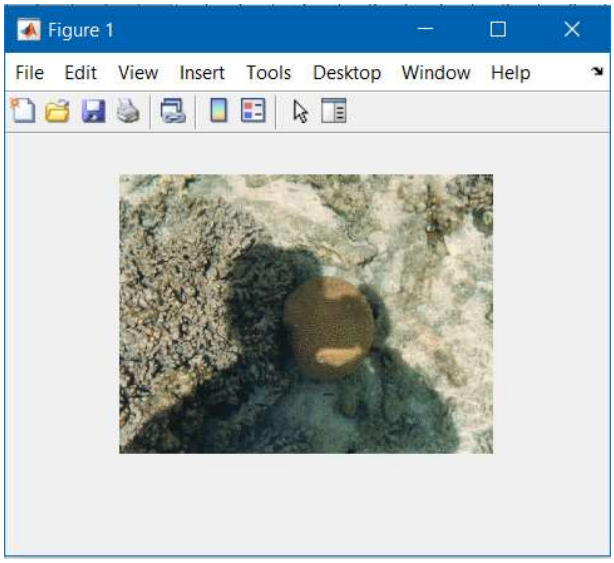
**Yêu cầu:**

+ Sử dụng hàm `imfinfo` kiểm tra trường `ColorType` ảnh shadow.tif trong tập ảnh B1_images.

+ dùng hàm `imread` để đọc ảnh, với `x` chứa ảnh được lập chỉ mục shadow.tif và `map` chứa colormap tương ứng của ảnh. Giá trị colormap trong tập tin ảnh sẽ được biến đổi tỉ lệ về phạm vi [0,1]

```
>> [x,map]=imread("B1_images\shadow.tif");
>> image(x), truesize, axis off, colormap(map);
```

Kết quả thu được:




### *5.2 Hàm* `imagesc`

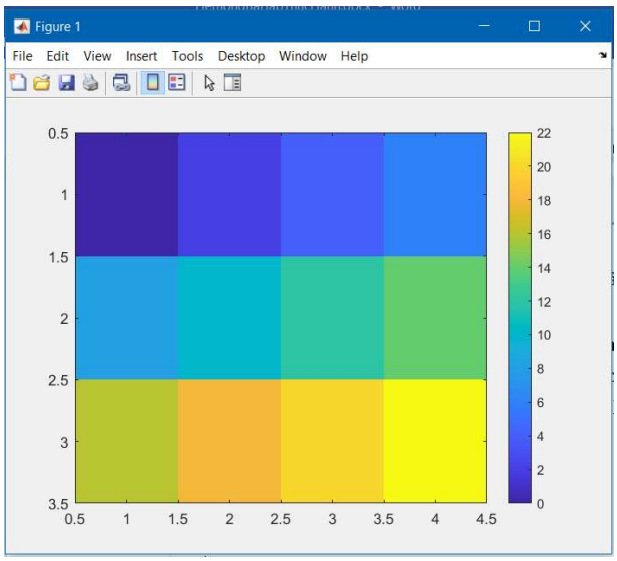
- Với cú pháp `imagesc(C)` hiển thị dữ liệu trong mảng C như một hình ảnh sử dụng đầy đủ màu trong colormap. Hay nói cách khác là chia tỉ lệ dữ liệu ảnh vào một phạm vi đầy đủ của color map hiện thời và hiển thị ảnh (hiển thị ảnh trên các trục đồ thị).

- Mỗi thành phần của C chỉ định màu cho 1 pixel trong ảnh. Ảnh kết quả của image là một lưới m-by-n các pixel trong đó m là số dòng và n là số cột của C. Chỉ số dòng và chỉ số cột xác định các tâm pixel tương ứng.

- Ví dụ:

```
>> C = [0 2 4 6; 8 10 12 14; 16 18 20 22];
imagesc(C)
colorbar
```

Kết quả thu được:




### *5.3 Hàm* `imshow`

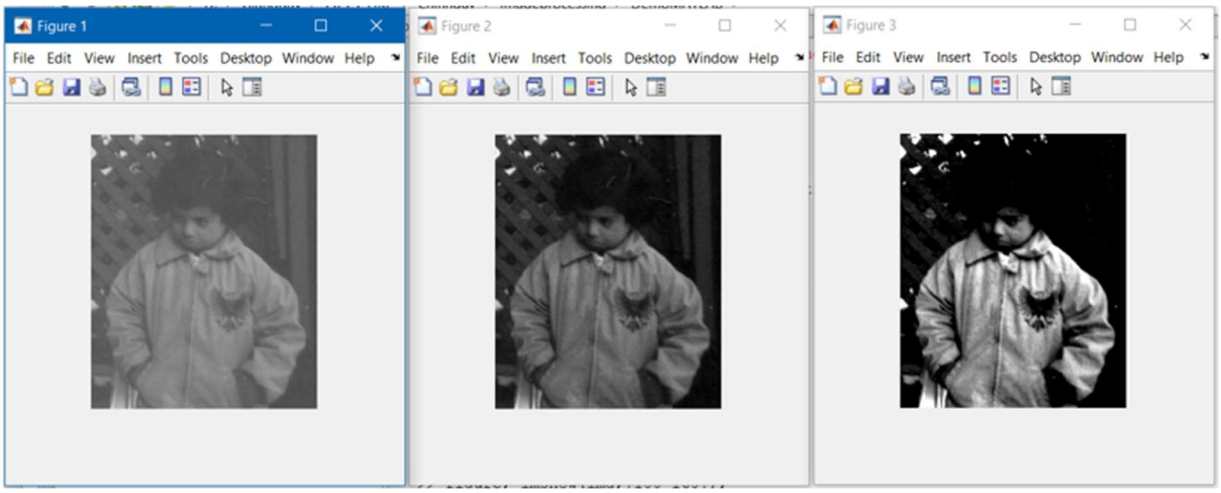
- Hiển thị ảnh đa cấp xám, ảnh RGB (true color), ảnh được lập chỉ mục (indexed) hoặc ảnh nhị phân.

- Có thể chứa một số tham số tối ưu và tùy chọn đối với các thiết lập thuộc tính liên quan đến hiển thị ảnh.

- Ví dụ:

```
>> img= imread('B1_images\pout.tif');
>> imshow(img);
>> imshow(img,[]);
>> imshow(img,[100 160]);
```

Kết quả thu được:




- Hàm `figure` cho phép tạo ra một cửa sổ figure mới với các giá trị thuộc tính mặc định. Figure tạo ra sẽ là figure hiện hành. Figure hiện thời sẽ chứa kết quả của các lệnh đồ họa như `axes`, `colormap`, trong trường hợp này là `imshow`

- Hàm `subplot` cho phép hiển thị nhiều ảnh trong cùng một figure (hiện hành).

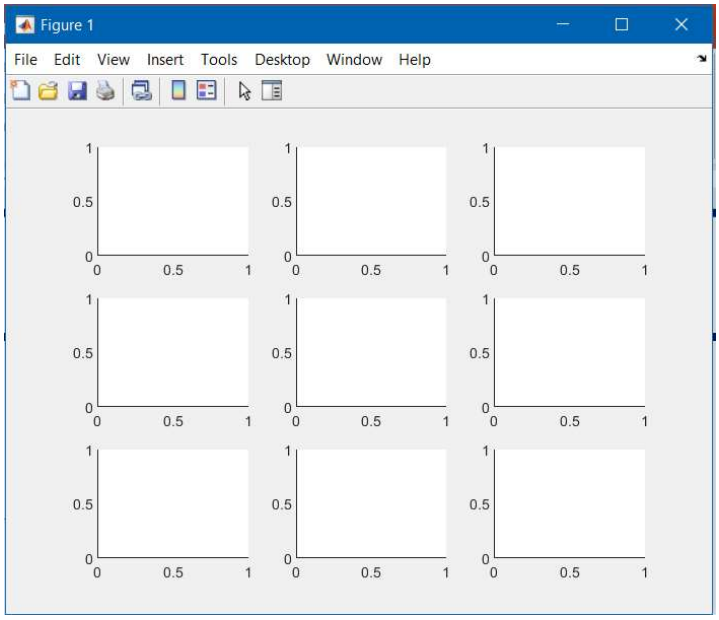
  + Hai tham số đầu cho biết số dòng và cột để chia figure

  + Tham số thứ ba chỉ định subplot được dùng để hiển thị ảnh

  + Ví dụ:

```
>> subplot(331);
>> subplot(332);
>> subplot(333);
>> subplot(334);
>> subplot(335);
>> subplot(336);
>> subplot(337);
>> subplot(338);
>> subplot(339);
```

Kết quả thu được:

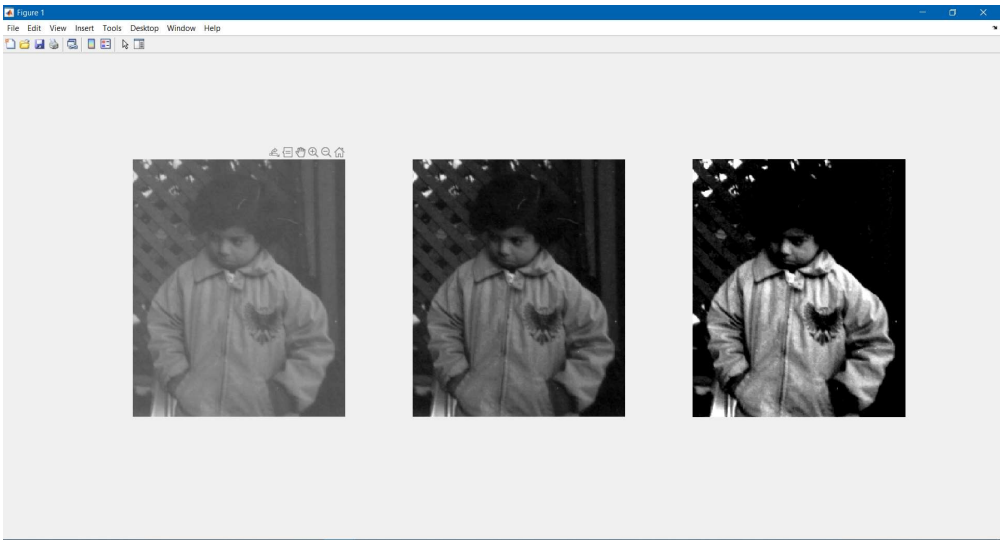



- Để hiển thị ảnh vào một subplot chúng ta chỉ định subplot trước, sau đó mới gọi imshow để đưa ảnh vào subplot.

**Yêu cầu:**

Anh (Chị) hãy viết code kết hợp `subplot` và `imshow` cho phép hiển thị 3 ảnh kết quả của ảnh pout.tif như sau:

+ subplot 1: hiển thị ảnh gốc
+ subplot 2: hiển thị ảnh có cùng tỉ lệ mức xám với ảnh gốc
+ subplot 3: hiển thị ảnh được chỉ định phạm vi mức xám [100 160].



**6. Đọc và hiển thị một ảnh cụ thể từ tập tin TIFF nhiều trang**

- Tập tin ảnh **corn.tif (B1_images)** là một ví dụ về tập tin ảnh có nhiều trang. Tập tin ảnh **corn.tif** có chứa ba ảnh: một ảnh được lập chỉ mục, một ảnh màu thật RGB, và một ảnh đa cấp xám.

- Sử dụng hàm `imfinfo` để kiểm tra thông tin của tập tin **corn.tif**:

Kết quả thu được:

```
>> info = imfinfo('B1_images\corn.tif')

info =

  3×1 struct array with fields:

    Filename
    FileModDate
    FileSize
    Format
    FormatVersion
    Width
    Height
    BitDepth
    ColorType
```

Chúng ta có thể thấy **corn.tif** là cấu trúc mảng 3×1, tức sẽ có 3 ảnh con được lưu trữ trong **corn.tif**. Chúng ta có thể truy xuất từng ảnh này thông qua chỉ mục tương ứng của chúng trong mảng.

- Ví dụ:

```
>> info(1)

ans =

  struct with fields:

                Filename: 'D:\Giangday\20_21_HK2\Chinhquy\imageprocessing\D
            FileModDate: '29-Jul-2020 13:14:12'
               FileSize: 230537
                 Format: 'tif'
          FormatVersion: []
                  Width: 312
                 Height: 415
               BitDepth: 8
```

**Yêu cầu:**

- Lần lượt kiểm tra thông tin của các ảnh con trong corn.tif.
- Dùng hàm `imread` với cú pháp:

```
imread (<tên tập tin>, <chỉ số của ảnh>)
```

để đọc các ảnh con, sau đó hiển thị các ảnh con này vào cùng figure hiện hành. Lưu ý, chỉ mục mảng trong MATLAB bắt đầu bằng 1.

- Hãy cho biết các ảnh con này là loại ảnh gì?

### 7. Ghi/lưu ảnh kết quả với hàm `imwrite`

- Cú pháp cơ bản: `imwrite(A, filename)`

- Ghi dữ liệu ảnh `A` vào một tập tin đồ họa được chỉ định bởi `filename`, bao gồm cả phần định dạng tập tin.

- Hàm imwrite tạo một tập tin mới trong thư mục hiện hành (current folder). Độ sâu bit của ảnh đầu ra phụ thuộc vào dữ liệu của A và định dạng tập tin được chỉ định.

- Đối với hầu hết định dạng:

*+ Nếu A có kiểu dữ liệu uint8, thì imwrite cho ra các giá trị 8-bit.*

*+ Nếu A có kiểu dữ liệu uint16 và định dạng tập tin đầu ra hỗ trợ dữ liệu 16-bit (JPEG, PNG và TIFF) thì đầu ra của imwrite sẽ là các giá trị 16-bit. Nếu định dạng tập tin đầu ra không hỗ trợ dữ liệu 16-bit, thì imwrite sẽ trả về lỗi.*

*+ Nếu A là một ảnh cấp xám hoặc ảnh màu RGB có kiểu dữ liệu double hoặc single, thì imwrite giả định rằng phạm vi động là [0,1] và tự động chia tỉ lệ dữ liệu với 255 trước khi ghi nó ra tập tin dưới dạng các giá trị 8-bit. Nếu dữ liệu trong A là single, chuyển A thành double trước khi ghi ra tập tin GIF hoặc TIFF.*

*+ Nếu A có kiểu dữ liệu luận lý, thì imwrite giả định dữ liệu là một ảnh nhị phân và ghi nó ra tập tin với độ sâu bit là 1 nếu định dạng tập tin cho phép. Các định dạng BMP, PNG và TIFF chấp nhận các ảnh nhị phân như các mảng đầu vào.*

- Nếu A có chứa dữ liệu ảnh được lập chỉ mục, chúng ta nên chỉ định thêm map như một thông số đầu vào. Cú pháp: `imwrite(A,map,filename)`.

- Nếu định dạng tập tin đầu ra sử dụng phương pháp nén không bảo toàn, `imwrite` cho phép chỉ định một thông số `quality`, dùng để cân bằng giữa chất lượng chủ quan của ảnh kết quả và kích thước tập tin

- Ví dụ:

```
>> img=imread('B1_images\onion.png');
>> imshow(img);
>> imwrite(img,'onion75.jpg');
>> imwrite(img,'onion05.jpg', 'quality',5);
>> imwrite(img,'onion95.jpg', 'quality',95);
```

- Xem các ảnh mới tạo trong thư mục hiện hành (ngoài môi trường MATLAB).

**Yêu cầu:**

a/- Ghi ảnh đa cấp xám ra tập tin PNG. Ví dụ, ghi một mảng 50-by-50 các giá trị đa cấp xám ra tập tin PNG trong current folder.

```
>> A=rand(50);
>> imwrite(A, 'myGray.png');
```

b/- Ghi dữ liệu ảnh được lập chỉ mục ra tập tin PNG

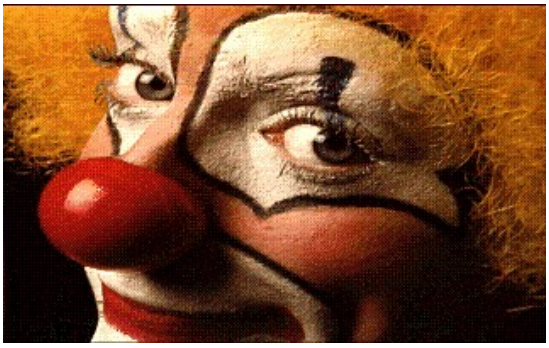
+ Nạp dữ liệu ảnh mẫu từ tập tin clown.mat (có sẵn trong MATLAB). Ảnh x và colormap của nó (map) sẽ được nạp vào trong workspace của MATLAB.

```
>> load clown.mat
```

+ Ghi dữ liệu ra một tập tin PNG mới.

```
>> imwrite(X,map, 'myclown.png');
```

Kết quả thu được (xem ngoài môi trường MATLAB):



c/- Ghi ảnh được lập chỉ mục với MATLAB colormap: ghi dữ liệu ảnh ra tập tin PNG mới với MATLAB colormap dựng sẵn, **copper**.

Định nghĩa một colormap copper-tone với 81 vector RGB. Sau đó, ghi dữ liệu ảnh ra tập tin PNG sử dụng newmap vừa định nghĩa.

```
>> newmap=copper(81);
>> imwrite(X,newmap, 'copperclown.png');
```

Kết quả thu được (xem ngoài môi trường MATLAB):

d/- Ghi ảnh màu RGB ra tập tin JPEG: tạo và ghi dữ liệu ảnh màu thật ra một tập tin JPEG.

+ Tạo một mảng 49-by-49-by-3 với các giá trị RGB ngẫu nhiên.

```
>> A = rand(49,49);
>> A(:,:,2) = rand(49,49);
>> A(:,:,3) = rand(49,49);
```

+ Ghi dữ liệu ảnh ra tập tin JPEG, chỉ định định dạng đầu ra bằng 'jpg'. Thêm một comment vào tập tin bằng một cập thông số tên-giá trị (name-value) 'Comment':

```
>> imwrite(A,'newImage.jpg','jpg','Comment','My JPEG file');
```

+ Xem thông tin tập tin ảnh vừa tạo với hàm `imfinfo`.

## 8. Chuyển ảnh màu thành ảnh đa cấp xám và chuyển ảnh xám thành ảnh nhị phân

### *8.1 Hàm `rgb2gray` chuyển ảnh màu RGB (truecolor) thành ảnh đa cấp xám*

- Chuyển đổi ảnh hoặc colormap RGB thành đa cấp xám (grayscale)
- Cú pháp:

```
I = rgb2gray(RGB)
newmap = rgb2gray(map)
```

- `I = rgb2gray(RGB)` sẽ chuyển một ảnh màu thật RGB thành một ảnh đa cấp xám I. Hàm rgb2gray chuyển một ảnh RGB thành ảnh đa cấp xám bằng cách loại bỏ các thông tin màu sắc (hue) và độ bão hòa (saturation) nhưng vẫn giữ lại độ sáng (luminance) của ảnh.

- `newmap = rgb2gray(map)` trả về một colormap đa cấp xám tương đương với map

**Yêu cầu:**

a/- Chuyển ảnh RGB thành ảnh đa cấp xám:

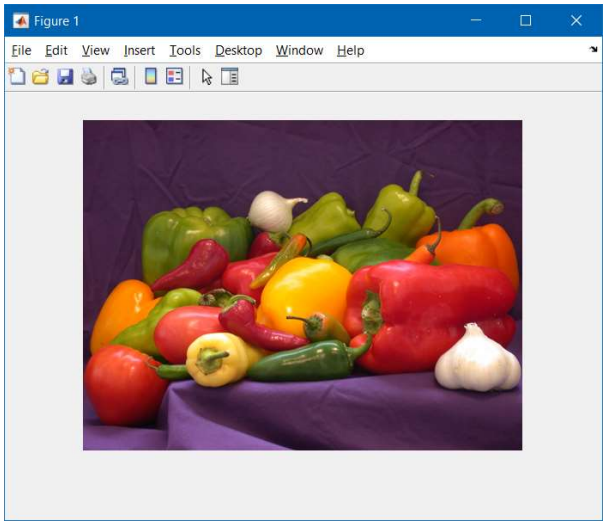
- Đọc và hiển thị một ảnh màu thật RGB bất kỳ từ tập ảnh B1_images và sau đó chuyển nó thành ảnh đa cấp xám.

- Ví dụ:

+ Đọc tập tin ảnh **peppers.png**

```
>> rgb=imread('B1_images\peppers.png');
>> imshow(rgb);
```

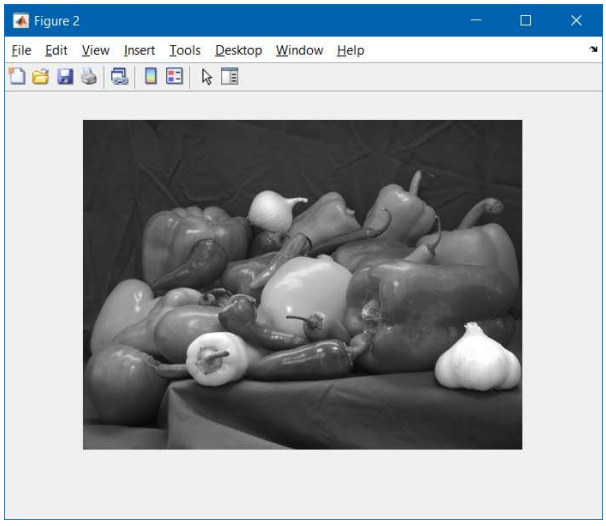
Kết quả thu được:




+ Chuyển ảnh RGB sang ảnh đa cấp xám và hiển thị nó:

```
>> I=rgb2gray(rgb);
>> figure, imshow(I);
```

Kết quả thu được:



b/- Chuyển một colormap RGB thành colormap đa cấp xám

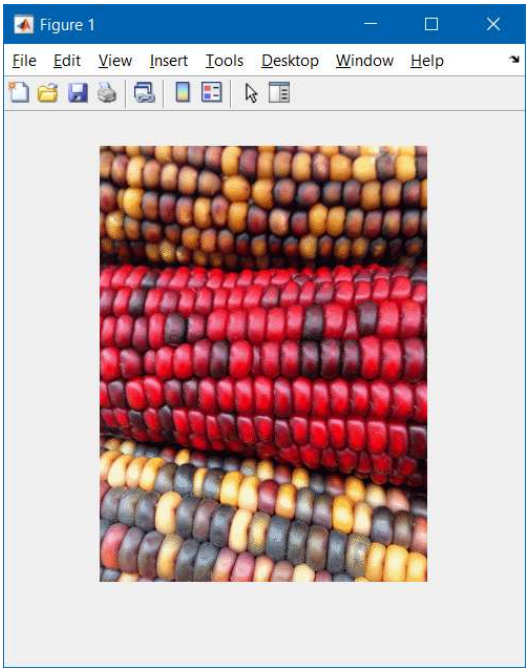
- Đọc một ảnh được lập chỉ mục với colormap RGB. Sau đó chuyển colormap thành đa cấp xám.

- Ví dụ:

+ Đọc tập tin **corn.tif** ảnh đầu tiên là ảnh được lập chỉ mục bởi một colormap RGB.

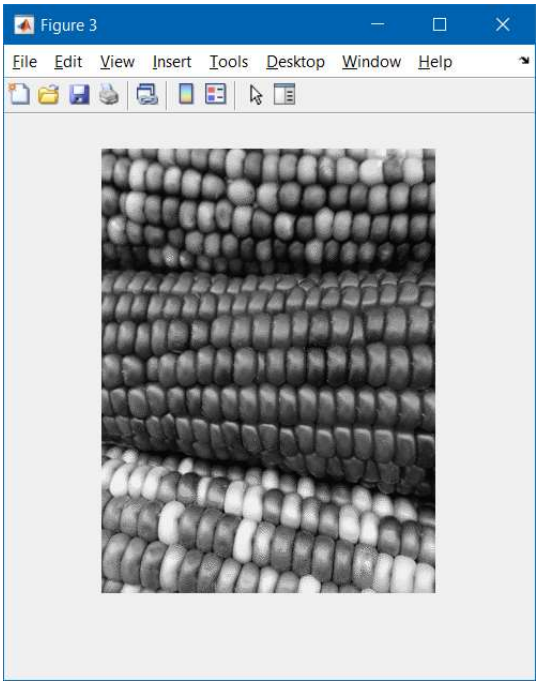
```
>> [X,map]=imread('B1_images\corn.tif');
>> imshow(X,map);
```

Kết quả thu được:




+ Chuyển colormap RGB thành colormap đa cấp xám và hiển thị lại ảnh

```
>> newmap=rgb2gray(map);
>> figure, imshow(X,newmap);
```



### *8.2 Hàm ind2gray chuyển ảnh màu được lập chỉ mục thành ảnh đa cấp xám*

Hàm ind2gray thực hiện chuyển ảnh được lập chỉ mục với một color map thành một ảnh grayscale. Hàm ind2gray loại bỏ các thông tin Hue (sắc thái màu) và Saturation (độ bảo hòa màu) ra khỏi ảnh đầu vào và chỉ giữ lại thông tin Value (độ sáng).

- Cú pháp:

```
I=ind2gray(X, cmap)
```

Trong đó:

`X`: ảnh gốc

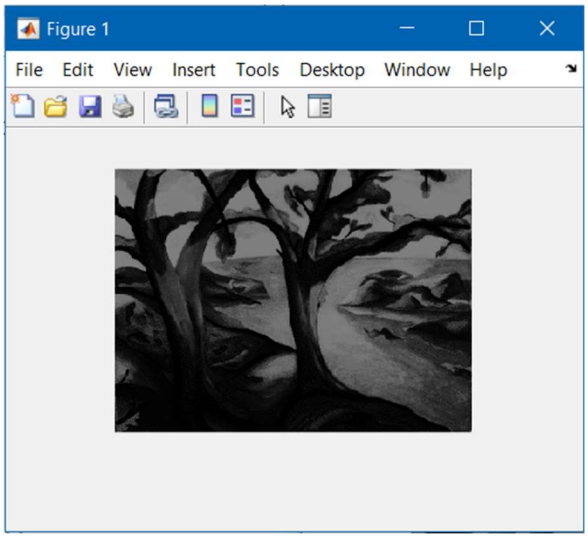
`cmap`: color map của ảnh gốc

`I`: ảnh kết quả

- Ví dụ: tập tin tree.tif là một tập tin ảnh có chứa 8 ảnh bên trong với ảnh đầu tiên là một ảnh màu được lập chỉ mục.

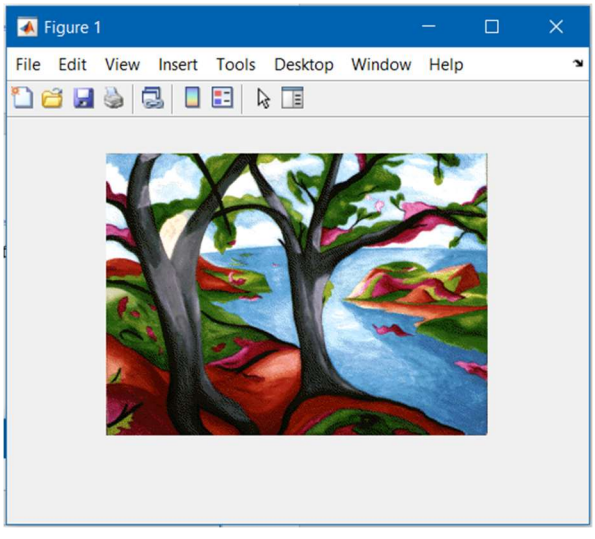
+ Nếu chỉ đọc và hiển thị ảnh như thông thường, kết quả ảnh hiển thị sẽ không chính xác do thiếu đi thành phần color map của ảnh gốc.

```
>> indImage=imread("B2_images\trees.tif");
>> imshow(indImage);
```



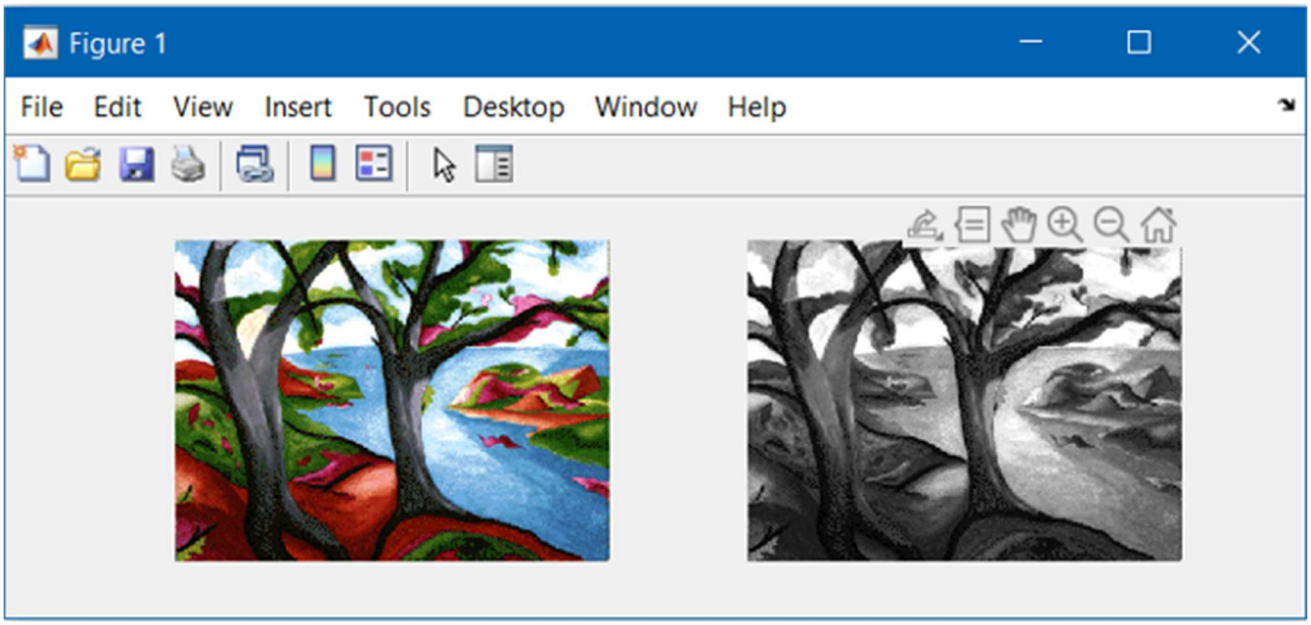

+ Vì vậy, khi đọc và hiển thị ảnh sẽ cần thêm thông số color map như sau:

```
>> [indImage,cmap]=imread("B2_images\trees.tif",1);
>> imshow(indImage,cmap);
```



+ Để chuyển ảnh index trên thành ảnh đa cấp xám chúng ta sẽ thực hiện đoạn lệnh trong command window sau:

```
>> [indImage,cmap]=imread("B2_images\trees.tif",1);
>> subplot(121);
>> imshow(indImage,cmap);
>> gray=ind2gray(indImage,cmap);
>> subplot(122);
>> imshow(gray);
```




### *8.3 Hàm imbinarize chuyển ảnh xám thành ảnh nhị phân*

❖ **Hàm `imbinarize`**

- Nhị phân hóa ảnh đa cấp xám 2-D hoặc ảnh volumn 3-D bằng cách phân ngưỡng.
- Cú pháp:

```
BW = imbinarize(I)
BW = imbinarize(I,method)
BW = imbinarize(I,T)
BW = imbinarize(I,'adaptive',Name,Value)
```

- `BW = imbinarize(I)` tạo một ảnh nhị phân từ ảnh đa cấp xám 2-D hoặc 3-D bằng cách thay thế tất cả các giá trị trên một ngưỡng toàn cục bằng các con số 1, và đặt các giá trị còn lại bằng các con số 0.

- Mặc định `imbinarize` sử dụng phương pháp của Otsu để chọn giá trị ngưỡng nhằm giảm thiểu phương sai bên trong lớp (intraclass variance) của các pixel trắng và đen đã được phân ngưỡng. `imbinarize` sử dụng histogram 256-bin để tính toán ngưỡng Otsu, nếu muốn sử dụng histogram khác có thể tìm hiểu thêm `otsuthresh`

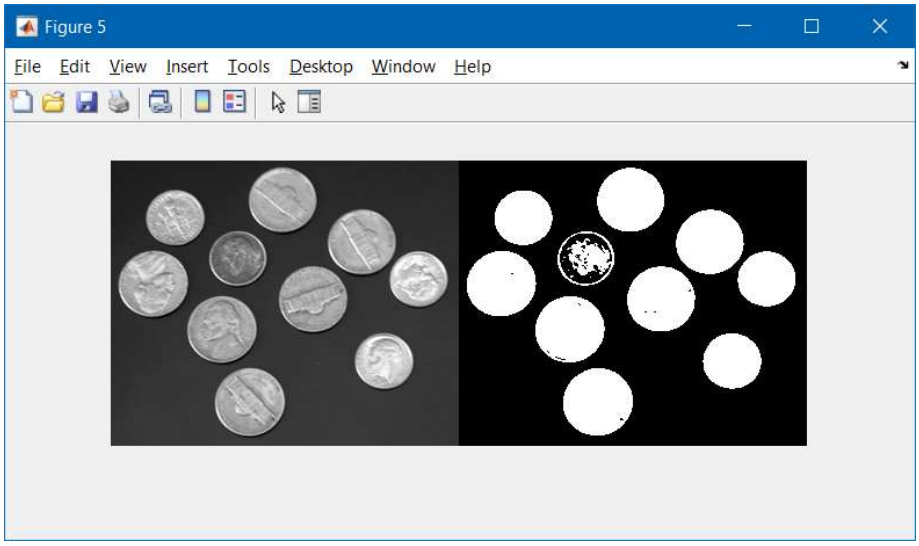
**Yêu cầu:**

- Đọc và hiển thị ảnh bất kỳ trong tập ảnh B1_images (nếu là ảnh màu chuyển sang ảnh đa cấp xám).
- Chuyển ảnh đa cấp xám thành ảnh nhị phân và hiển thị kết quả.

- Ví dụ:

```
>> img=imread('B1_images\coins.png');
>> BW=imbinarize(img);
>> figure, imshowpair(img, BW, 'montage');
```

Kết quả thu được:




***Giới thiệu thêm:***

Bên cạnh đó trong IPT của MATLAB cũng có cung cấp hàm im2bw cho phép chuyển ảnh thành ảnh nhị phân dựa trên ngưỡng như sau:

- Cú pháp:

```
BW = im2bw(I,level)
BW = im2bw(X,cmap,level)
BW = im2bw(RGB,level)
```

với:

+ `BW = im2bw(X,cmap,level)` chuyển một ảnh đa cấp xám I thành ảnh nhị phân BW, bằng cách thay thế các pixel trên ảnh đầu vào với độ sáng lớn hơn level bằng giá trị 1 (màu trắng) và thay thế tất cả các pixel còn lại bằng giá trị 0 (màu đen).

+ `BW = im2bw(X,cmap,level)` chuyển ảnh được lập chỉ mục X với colormap cmap thành ảnh nhị phân.

+ `BW = im2bw(RGB,level)` chuyển ảnh màu thật RGB thành ảnh nhị phân.

- Tuy nhiên **từ phiên bản MATLAB R2016a trở đi, hàm `im2bw` không còn được khuyến nghị sử dụng** bởi vì giá trị ngưỡng mặc định của `im2bw` không tối ưu với hầu hết các ảnh. Nếu muốn sử dụng một ngưỡng thích hợp cho ảnh cần xử lý, chúng ta phải tính toán level bằng `graythresh` trước khi gọi `im2bw`.