Informatyka

Wykład 1.5

Witold Dyrka
witold.dyrka@pwr.wroc.pl

20/2 – 5/3/2012

Program wykładów

0. Informatyka. Wprowadzenie do Matlaba (13.02.12)
1. **Matlab dla programistów C/C++** (20.02.12)
2. Optymalizacja obliczeń. Grafika w Matlabie (05.03.12)
3. Programowanie zorientowane obiektowo (19.03.12)
4. Programowanie zorientowane obiektowe w praktyce (02.04.12)
5. Graficzny interfejs użytkownika (16.04.12)
6. Obliczenia numeryczne (30.04.12)
7. Kolokwium (14.05.12)

Dzisiejszy wykład w oparciu o...

- B. Mrozek, Z. Mrozek. MATLAB i Simulink. Poradnik użytkownika. Wydanie III. Helion 2010. Rozdział 3 i 5.
- MATLAB® Getting Started Guide (R2011b). http://www.mathworks.com/help/pdf_doc/matlab/getstart.pdf
- MATLAB Documentation Center (beta). Programming and Data Types http://www.mathworks.com/help/matlab/
- MATLAB Product Documentation. Programming Fundamentals. Classes (Data Types) http://www.mathworks.com/help/techdoc/matlab_prog/f2-43934.html

Program wykładu

- **Typy danych**
  - dynamiczny system typów
  - typ zmiennoprzecinkowy
    - macierze pełne
    - macierze rzadkie
  - typ całkowitoliczbowy
    - zastosowanie w przetwarzaniu obrazów
  - typ znakowy
  - typ logiczny
    - operatory logiczne

Zmienne

- Domyślną strukturą danych jest tablica 2-wymiarowa

```
>> a = 1
a =
     1
>> size(a)          % zwraca rozmiar tablicy
ans =
     1     1
```

- Nie trzeba deklarować zmiennych
  - zmienna jest tworzona w przestrzeni roboczej w momencie inicjacji
  - typ zmiennej jest określany na podstawie przypisanej wartości – tzw. **dynamiczny system typów**

```
>> class(a)         % zwraca typ zmiennej
ans =
double
```

Dynamiczny system typów

- Zmienne deklarowane bez podania typów
- Typ jest określany w czasie wykonania

- Wydaje się, że to prostsze niż deklarowanie typów
  - ale bardziej błędogenne
    - błędy wykrywane dopiero w czasie wykonania
      - wymaga więcej testowania

- Kiedy tworzymy zmienną Matlab rezerwuje pamięć
  - kiedy zmieniamy typ zmiennej pamięci może brakować
    - Matlab <u>realokuje</u> pamięć (przemieszcza zmienną w pamięci)

Dynamiczny system typów (2)

- Łatwo nadpisać (np. `i, j`) lub przysłonić nazwę:

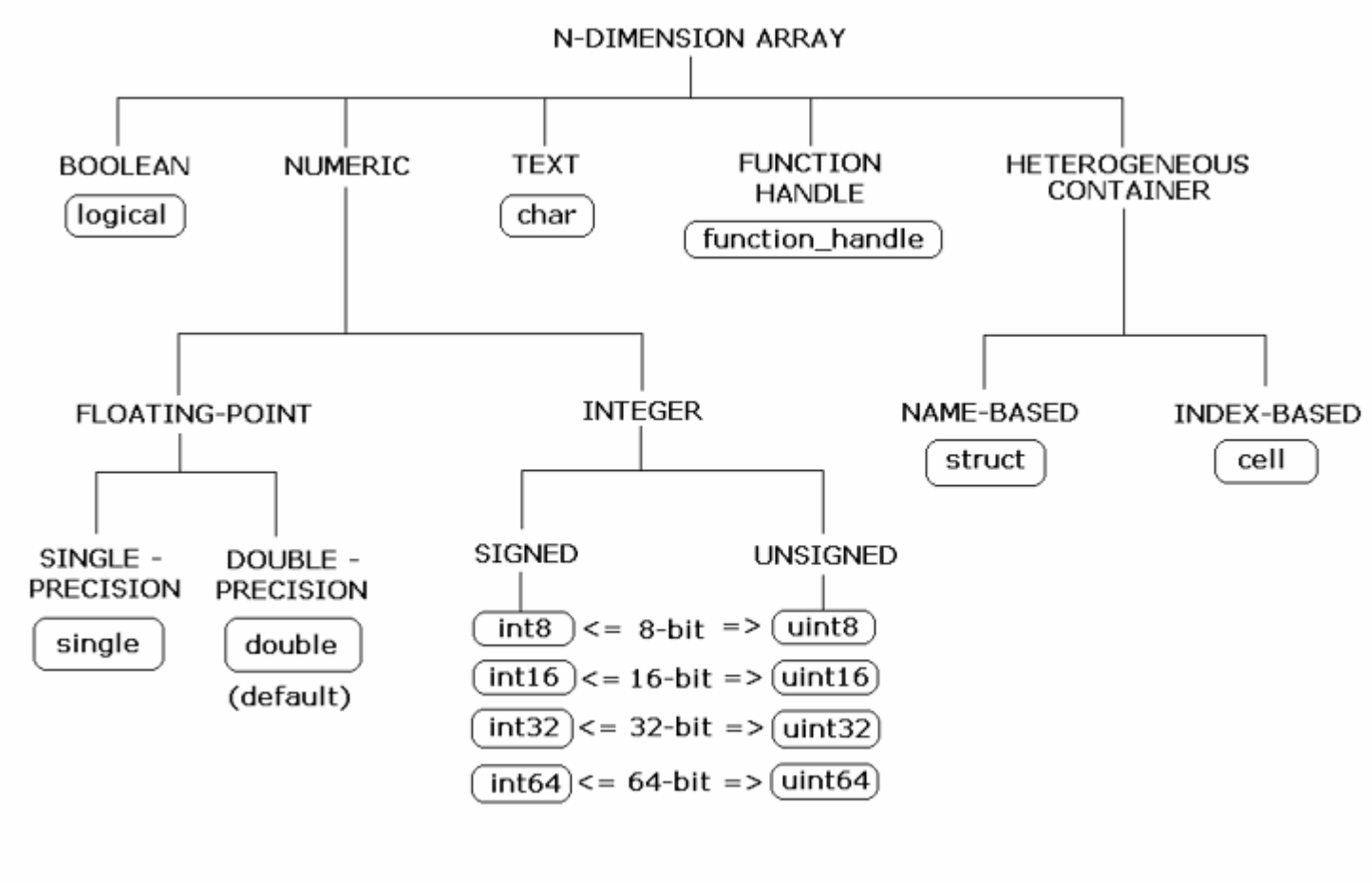
```
>> min(rand(10, 1))
ans =
      0.10563
>> min = 17;
>> min(rand(10, 1))
??? Subscript indices must either be real positive integers or logicals.

>> clear min            % usuwamy zmienną min z przestrzeni roboczej
>> min(rand(10, 1))
ans =
     0.067993
```

- Trudniej sprawdzić poprawność danych:

```
rok = input('Podaj rok');                          % w C++
if (round(rok) ~= rok))                            % int rok;
    error('Rok musi być liczbą całkowitą')         % cin >> rok;
end                                                % if (!cin) throw ...
```

Typy danych

- 15 podstawowych typów danych (klas)



Źródło: http://www.mathworks.com/help/techdoc/matlab_prog/f2-47534.html

Typ zmiennoprzecinkowy

- Tablica n-wymiarowa
  - Typ numeryczny
    - Typ zmiennoprzecinkowy (floating point)
      - pojedyncza precyzja (32bit): `single`
      - podwójna precyzja (64bit): `double` (domyślny)

```
>> a=1;     % to samo co a = double(1);
>> whos a   % wyświetla informacje o zmiennej
  Name      Size            Bytes  Class     Attributes
  a         1x1                 8  double
```

```
>> realmin('double'), realmax('double')   % zakresy
ans =
  2.2251e-308
ans =
  1.7977e+308
```

Precyzja typu zmiennoprzecinkowego

```
>> format long
>> eps(1)                          % precyzja reprezentacji liczby 1:
ans =                              % - następna większa od jedynki liczba
  2.220446049250313e-016           %   to 1.000000000000000222044604925031 3

>> eps(100)                        % precyzja reprezentacji liczby 100:
ans =                              % - następna większa od jedynki liczba
  1.421085471520200e-014           %   to 100.000000000000014210854715202 00

>> sin(pi)                         % sinus(pi) = 0, czyż nie?
ans =
   1.224646799147353e-016          % co???

>> a = 0.0;
>> for i = 1:10, a = a + 0.1; end
>> a
a =
   1.000000000000000               % wygląda dobrze
a == 1                             % czy a równe 1?
ans =
     0                             % nie! bo 0.1 nie jest dokładnie równe 0.1:/

>> b = 1e-16 + 1 - 1e-16;
>> c = 1e-16 - 1e-16 + 1;
>> b == c                          % kolejność działań jest istotna!
ans =
     0
```

http://www.mathworks.com/help/techdoc/matlab_prog/f2-12135.html#f2-98690

Zmiennoprzecinkowe macierze pełne

- <u>Wybrane funkcje</u>

2D	• `eye`	tworzy	macierz jednostkową (1 na przekątnej)
	• `ones`		macierz o elementach równych 1
	• `zeros`		macierz o elementach równych 0
	• `rand`		macierz losową o rozkł. równomiernym [0..1]
	• `randn`		macierz losową o rozkł. normalnym, σ=1
1D	• `linspace`		wektor o wartościach rozłożonych liniowo
1D	• `logspace`		wektor o wartościach rozłożonych logarytm.
2D	• `diag`	tworzy zwraca	macierz diagonalną z wektora główną przekątną macierzy
2D	• `fliplr,flipud`	odbija macierz wertykalnie, odbija macierz horyzontalnie	
	• `reshape`	zmienia wymiary macierzy zachowując elementy	
2D	• `rot90`	obrót macierzy o 90 st.	
2D	• `tril, triu`	tworzy macierz trójkątną dolną, górną	
	• `cat`	konkatenuje macierze (skleja z innych tablic)	
	• `squeeze`	usuwa wymiary jednostkowe macierzy	

```
>> help randn
 RANDN  Normally distributed random numbers.
    R = RANDN(N) returns an N-by-N matrix containing pseudo-random values
    drawn from a normal distribution with mean zero and standard deviation
    one.  RANDN(M,N) or RANDN([M,N]) returns an M-by-N matrix.  RANDN(M,N,P,...)
    or RANDN([M,N,P,...]) returns an M-by-N-by-P-by-... array.  RANDN with
    no arguments returns a scalar.  RANDN(SIZE(A)) returns an array the
    same size as A.

    You can use one of two generator algorithms, as follows:

    RANDN(METHOD,S) causes RANDN to use the generator determined by METHOD, and
     initializes the state of that generator.  S is a scalar integer value from
    0 to 2^32-1, or the output of RANDN(METHOD).  METHOD is one of the following
     strings:

       'state' - Use Marsaglia's Ziggurat algorithm, the default in MATLAB
                 Versions 5 and later.  The period is approximately 2^64.

       'seed'  - Use the polar algorithm, the default in MATLAB Version 4.
                 The period is approximately (2^31-1)*(pi/8).

    RANDN(METHOD) returns the current internal state of the generator
    determined by METHOD.  However, it does not switch generators.

    The sequence of numbers produced by RANDN is determined by the internal
    state of the generator.  Setting the generator to the same fixed state
    allows computations to be repeated.  Setting the generator to different
    states leads to unique computations, however, it does not improve any
    statistical properties.  Since MATLAB resets the state at start-up, RANDN
    will generate the same sequence of numbers in each session unless the
    state is changed.

    Note: The size inputs M, N, and P... should be nonnegative integers.
    Negative integers are treated as 0.
```

```
 Examples:

    Return RANDN to its default initial state.
       randn('state',0)

    Initialize RANDN to a different state each time.
       randn('state',sum(100*clock))

    Save the current state, generate 100 values, reset the state,
    and repeat the sequence.

       s = randn('state');
       r1 = randn(100);
       randn('state',s);
       r2 = randn(100); % contains exactly the same values as r1

     Generate normal values with mean 1 and standard deviation 2.
       r = 1 + 2.*randn(100,1);

See also rand, sprand, sprandn, randperm.

Overloaded methods:
   darray/randn

Reference page in Help browser
   doc randn

>>
```

`randn`

- Generujemy pojedynczą liczbę z rozkładu normalnego N(0,1):

randn

- Generujemy pojedynczą liczbę z rozkładu normalnego N(0,1):

```
>> randn
ans =
   -0.4326
```

- Generujemy macierz 2x2x2 z rozkładu normalnego N(0,1):

randn

- Generujemy pojedynczą liczbę z rozkładu normalnego N(0,1):

```
>> randn
ans =
   -0.4326
```

- Generujemy macierz 2x2x2 z rozkładu normalnego N(0,1):

```
>> randn(2,2,2)
ans(:,:,1) =
   -0.3999    0.8156
    0.6900    0.7119
ans(:,:,2) =
    1.2902    1.1908
    0.6686   -1.2025
```

- Generujemy wektor 7-el. z rozkładu normalnego N(2,3):

randn

- Generujemy pojedynczą liczbę z rozkładu normalnego N(0,1):

```
>> randn
ans =
   -0.4326
```

- Generujemy macierz 2x2x2 z rozkładu normalnego N(0,1):

```
>> randn(2,2,2)
ans(:,:,1) =
   -0.3999    0.8156
    0.6900    0.7119
ans(:,:,2) =
    1.2902    1.1908
    0.6686   -1.2025
```

- Generujemy wektor 7-el. z rozkładu normalnego N(2,3):

```
>> 2+3.*randn(1,8)
ans =
    2.1209    4.0313    3.7067    1.2331    0.8676    1.1123   -2.4254
```

- Resetujemy generator i generujemy skalar z N(0,1):

```
>> randn('state',0); randn
ans =
   -0.4326
```

Zmiennoprzecinkowe macierze rzadkie (2-wym.)

```
>> A = diag([1 2 3])
A =  1     0     0
     0     2     0
     0     0     3

>> B = sparse(A)
B = (1,1)        1
    (2,2)        2
    (3,3)        3
```

- Tylko elementy niezerowe i ich pozycje
  - opłacalne gdy duża liczba zer w macierzy
  - złożoność obliczeniowa zależy od liczby elementów niezerowych
    - macierze pełne: złożoność zależy od wymiarów macierzy m x n

- Tworzenie:

S = sparse(A) % tworzy macierz rzadką **S** z pełnej A
S = sparse(i, j, s, m, n, nzmax) % tworzy macierz rzadką **S** na podst.
% **i** – wektora pozycji w wierszach
% **j** – wektora pozycji w kolumnach
% **s** – wektora wartości
% **m,n** – rozmiary macierzy, domyślnie: **m=max(i), n=max(j)**
% **nzmax** – liczba el. niezerowych, d: **nzmax = length(s)**

speye, sprand, sprandn % dość podobnie jak dla macierzy pełnych

[i, j, s] = find(X) % znajduje el. niezerowe w macierzy **X** – rzadkiej lub pełnej
nz = nnz(X) % liczba elementów niezerowych w macierzy **X**

A = full(S) % konwertuje macierz rzadką **S** na pełną **A**

Macierz rzadka czy pełna?

- Oznaczenia:
  `A` – macierz pełna
  `S` – macierz rzadka

- Jednoargumentowe funkcje zwracające macierz/wektor zachowują typ argumentu

  `~A, diag(A), sum(A), max(A)` – zwracają wektory lub macierze pełne
  `~S, diag(S), sum(S), max(S)` – zwracają wektory lub macierze rzadkie

- Operatory dwuargumentowe zwracają macierze pełne, chyba, że zachowują rzadkość

  `S+A, S*A , A\S, A/S, S/A` – zwracają macierze pełne
  `S.*A, A.*S, S&A, A&S` – zwracają macierze rzadkie
  `A./S` – zwraca macierz pełną, ale `S./A` – zwraca macierz rzadką

- Operator konkatenacji i funkcja `cat` zwracają macierze rzadkie, np. `[A; S]` – m.rzadka

- Indeksowanie zachowuje typ rodzaj macierzy

  `S2 = S(i,j)` – S2 rzadka, także gdy jest skalarem (błąd w dokumentacji Matlaba R2011b)

- Przypisanie zachowuje typ macierzy po lewej stronie

  `A(i,j) = S` - A pozostaje pełna
  `S(i,j) = A` - S pozostaje rzadka

http://www.mathworks.com/help/techdoc/math/f6-8856.html#f6-9622

Typ całkowitoliczbowy

- Tablica n-wymiarowa
  - Typ numeryczny
    - Typ całkowitoliczbowy (integer)
      - ze znakiem 8, 16, 32, 64bit: `int8, int16, int32, int64`
      - bez znaku 8, 16, 32, 64bit: `uint8, uint16, uint32, uint64`

```
>> x=int16(1);

>> y1=uint8(-1), y2=int16(65536)
>> y1=uint8(-1), y2=int16(40000)
y1 =
    0          % liczba poza zakresem [0, 255] jest obcinana
y2 =
  32767        % liczba poza zakresem [-32768, 32767] jest obcinana

>> intmin('int64')
ans =
 -9223372036854775808
>> intmax('uint64')
ans =
 18446744073709551615
```

http://www.mathworks.com/help/techdoc/matlab_prog/f2-12135.html#f2-52790

Typ całkowitoliczbowy (2)

```
>> a = 326.001
a =
  326.0010
>> x = int16(a)
x =
    326
>> b = 325.499;
b =
  325.4990          % typowe zaokrąglanie do połowy
>> y = int16(b)            % na marginesie: do zaokrąglania liczb double
y =                        %   służą funkcje: round, fix, floor, ceil
    325
>> y = y + 0.4      % 325 + 0.4 = 325.4; int16(325.4) = 325
y =
    325
>> y = y + 0.4      % j.w.
y =
    325
>> y = y + 0.5      % 325 + 0.5 = 325.5 int16(325.5) = 326
y =
    326
>> y = y + 0.5      % 326 + 0.5 = 326.5 int16(326.5) = 327
y =
    327                    % na marginesie: nie można mnożyć/dzielić
                           % macierzowo dwóch tablic liczb całkowitych
```

Zastosowanie: obrazy

- wczytanie obrazu RGB (np. JPG, BMP, PNG):

```
>> obrazrgb = imread('DSC00395.JPG'); % ≡ imread('DSC00395','JPG');

>> size(obrazrgb)
ans =
        1536   2048    3

>> class(obrazrgb)
ans =
uint8
```

```
% zdjęcie 2048x1536, 3 kanały RGB

% typ danych:
% - obraz 16.7mln kolorów, tj.
%    24 bit = 3 x 8 bit
```

- wczytanie obrazu indeksowanego (np. GIF):

```
>> [obrazind, mapa] = imread('DSC00395.GIF');    % mapa - mapa indeksów

>> size(obrazind)
ans =
        1536   2048                    % zdjęcie 2048x1536
```

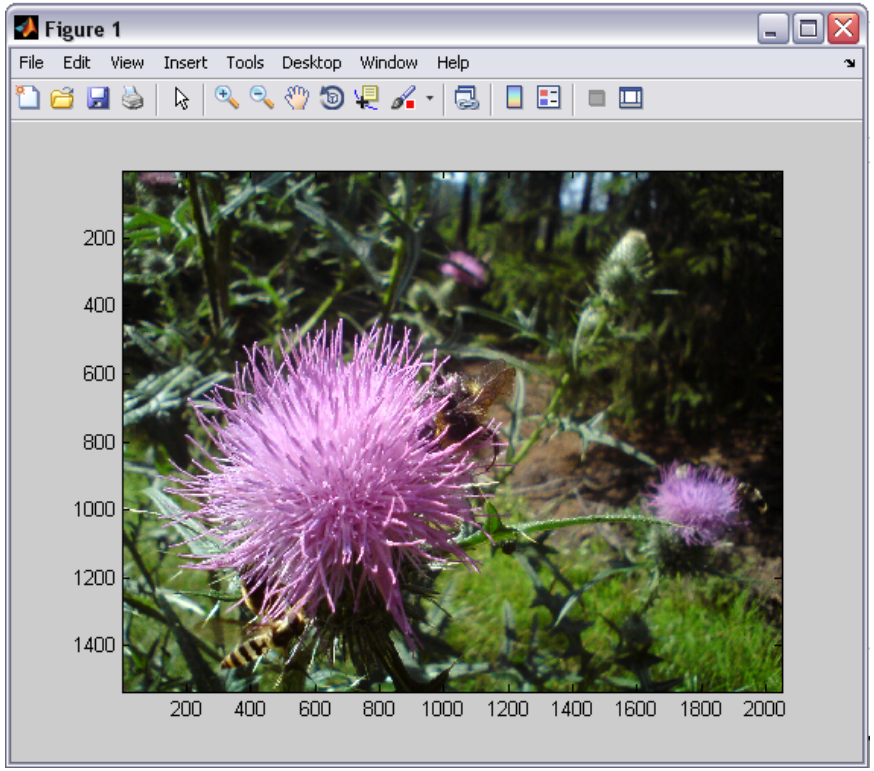
- konwersje:

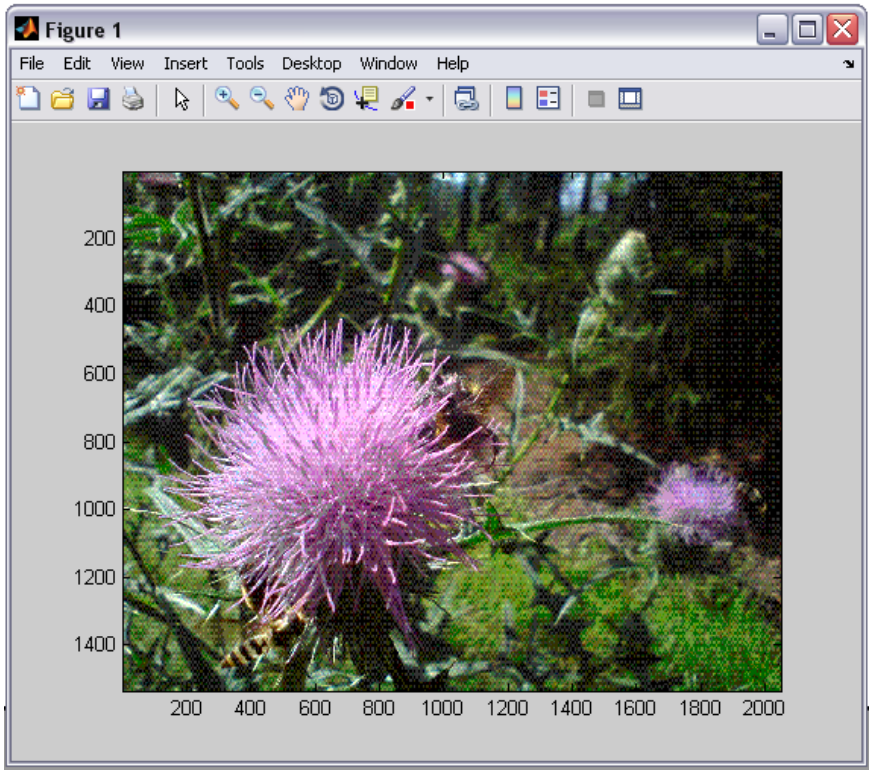
```
obrazind = rgb2ind(obrazrgb, mapa)
obrazind = rgb2ind(obrazrgb, maks_liczba_kolorow)    % są też inne opcje

obrazrgb = ind2rgb(obrazind, mapa)
```

Obrazy (2)

- wyświetlanie:

```
>> image(obrazrgb)
>> image(obrazind), colormap(mapa)
>> axis image                                  % kwadratowe piksele
```






```
% Można także wyświetlić macierz liczb rzeczywistych 2-D
```

Obrazy (3)

- zapis do pliku:

```
>> imwrite(obrazind, colormap('copper'),'moj_obraz.jpg');
```

mapa to tłumaczenie indeksów na kolory RGB, np.

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

- 0 = biały
- 1 = niebieski
- 2 = czerwony

```
colormap([1 1 1; 0 0 1; 1 0 0])
```

```
>> imwrite(obrazrgb,'moj_obraz.png');
```

Typ znakowy

- Tablica n-wymiarowa
  - Typ znakowy (char) zakres kodów znaków: 0 – 65535 (16bit)

- Tworzenie prostych łańcuchów (*string*):

```
>>tekst = 'You''re right!'                          % pojedynczy łańcuch
tekst = You're right!

>> nazwisko = ['Leonard' ' M. ' 'Adleman']          % konkatenacja
nazwisko = Leonard M. Adleman

>> nazwisko = strcat('Leonard', ' M. ', 'Adleman')  % konkatenacja inaczej
nazwisko = Leonard M.Adleman

>> biomolekuly = ['DNA'; 'RNA'; 'Bialka']           % tablica łańcuchów?
Error using vertcat. CAT arguments dimensions are not consistent.
```

- Wszystkie łańcuchy muszą mieć jednakową długość, automatycznie zapewnia to:

```
>> biomolekuly = char('DNA','RNA','Bialka')         % tablica łańcuchów
biomolekuly =                                       % por. strvcat
DNA
RNA
Bialka
```

http://www.mathworks.com/help/matlab/ref/strings.html

Łańcuchy złożone

- Tworzenie łańcuchów złożonych:

```
>> a = 25.4; b=30.6;

>> tekst = ['a ma wartosc ',a,'. b ma wartosc ',b]
tekst = a ma wartosc □. b ma wartosc □                % char(a) = □

>> tekst = ['a ma wartosc ',num2str(a),'. b ma wartosc ',num2str(b)]
tekst = a ma wartosc 25.4. b ma wartosc 30.6          % ok!

>> tekst2 = sprintf('a ma wartosc %g. b ma wartosc %g',a,b)
tekst2 = a ma wartosc ok. 25.4. b ma wartosc 30.6     % to samo
                                                      % %g-liczby rzeczyw.


>> tekst2 = sprintf('a ma wartosc ok.%d. b ma wartosc ok.%d',a,b)
tekst2 = a ma wartosc ok. 2.540000e+001. b ma wartosc 3.060000e+001
                                                      % %d-liczby całkow. ?

>> tekst2 = sprintf('a ma wartosc ok.%d. b ma wartosc ok.%d',round(a),round(b))
tekst2 = a ma wartosc ok.25. b ma wartosc ok.30
                                                      % teraz ok!
```

http://www.mathworks.com/help/matlab/matlab_prog/strings.html

Łańcuchy złożone

- Różne opcje formatowania `sprintf`
  - sprawdź w dokumentacji

```
>> tekst2 = sprintf('a ma wartosc %g. b ma wartosc %g',a,b)
tekst2 = a ma wartosc ok. 25.4. b ma wartosc 30.6

>> tekst2 = sprintf('a ma wartosc ok. %f. b ma wartosc %e',a,b)
tekst2 = a ma wartosc ok. 25.400000. b ma wartosc 3.060000e+001
% %f - format stałoprzecinkowy, %e - format wykładniczy

>> tekst2 = sprintf('a ma wartosc %10.4f. b ma wartosc %5.2e',a,b)
tekst2 = a ma wartosc ok.    25.4000. b ma wartosc 3.06e+001
% %10.4f - format stałoprzecinkowy, szerokość pola >=10, po przecinku 4 cyfry
% %5.2f  - format wykładniczy, szerokość pola >=5, po przecinku 2 cyfry

>> tekst2 = sprintf('a ma wartosc ok. %10.1g. b ma wartosc %5.2g',a,b)
tekst2 = a ma wartosc ok.     3e+001. b ma wartosc    31
% %10.1g - format optymalny, szerokość pola >=10, cyfr znaczące: 1
```

http://www.mathworks.com/help/matlab/ref/sprintf.html

Wybrane funkcje operujące na łańcuchach

- **`blanks`** ..... tworzy łańcuch spacji
- **`ischar`** ..... sprawdza czy zmienna jest typu znakowego
- **`isletter / isspace`** .. sprawdza gdzie tablica zawiera litery / spacje
- **`deblank (strtrim)`** ... usuwa spacje z końca (i początku) łańcucha
- **`lower / upper`** ..... zamienia na małe / duże litery
- **`strfind (strrep)`** .... znajduje (i podmienia) fragment w łańcuchu
- **`regexp (regexprep)`** .. znajduje (i podmienia) wyrażenie regularne
- **`sprintf / sscanf`** .... formatowany zapis / odczyt do / z łańcucha
- **`strtok`** ..... znajduje token w łańcuchu
- **`strcmp (strncmp)`** .... porównuje łańcuchy (na N pozycjach)
- **`eval`** ..... uruchamia tekst jako polecenia Matlaba

Typ logiczny

- Tablica n-wymiarowa
  - Typ logiczny (logical)
    - podobnie jak **bool** w C++
    - dwie wartości: `true` (1) / `false` (0)

- Tworzenie zmiennych typu logicznego:

```
B = logical(-2.8);     % jawna konstrukcja, wartość 1 (bo -2.8 ~= 0)
C = false;             % typ nadany przez użycie literału logicznego
D = 50>40;             % wynik operatora porównania, wartość 1
F = strncmpi('Marcin','marek',3) % wynik porównania tekstu, wartość 1
E = isinteger(4.9);    % wynik funkcji testującej is*, wartość 0
```

http://www.mathworks.com/help/techdoc/matlab_prog/f2-97022.html

Operatory logiczne (1)

- Operatory elementowe (*Element-wise*): `&, |, ~, xor`

```
A =     [0 1 1 0 1],                                A = [0 1 1 0 1],
B =     [1 1 0 0 1],   B = [1 1 0 0 1],             B = [1 1 0 0 1],
A & B = [0 1 0 0 1],  ~B = [0 0 1 1 0],  xor(A,B) = [1 0 1 0 0]
```

  - wymiary operandów (np. `A & B`) muszą być takie same
    - chyba, że jeden jest skalarem
  - automatyczna konwersja operandów do typu logicznego
    - elementy zerowe na `false`, skończone niezerowe na `true`

- Inne funkcje elementowe:

```
any(A)      % true jeśli choć jeden element A niezerowy
all(A)      % true jeśli wszystkie elementy A niezerowe
```

http://www.mathworks.com/help/techdoc/matlab_prog/f0-40063.html#f0-38959

Operatory logiczne (2)

- Operatory skrócone (*short-circuit*): `&&`, `||`
  - jeśli pierwszy operand pozwala określić wynik, pomija drugi, np.

```
A && B            % jeśli A==0 nie trzeba sprawdzać B
A || B            % jeśli A==1 nie trzeba sprawdzać B
```

  - operandy muszą być skalarami logicznymi
  - można stosować w instrukcjach warunkowych `if`

```
if (b ~= 0) && (a/b > 18.5) ...
```

  - lub innych gdy chcemy uniknąć zgłaszania błędu:

```
x = (b ~= 0) && (a/b > 18.5)
```

http://www.mathworks.com/help/techdoc/matlab_prog/f0-40063.html#f0-39129

Uwaga! Operatory elementowe w warunkach `if` i `while` ...

...zachowują się jak operatory skrócone

```
>> A = 1; B = [];

>> A|B        % porównuje skalarne 1 z każdym elementem tablicy B
ans =         % ponieważ B pusta, nie było czego porównywać
     []       % zwraca pustą tablicę typu logicznego

>> B|A
ans = []      % jak wyżej

>> if(A|B), disp('1'), else disp('0'), end
1
% skoro już A jest 1, to Matlab uznał, że cały warunek jest spełniony

>> if(B|A), disp('1'), else disp('0'), end
0
% skoro B nie jest 1, Matlab policzył całe wyrażenie, jak u góry

>> A = [1 1];   B = [2 0 1];
>> A|B
Error using  |. Inputs must have the same size.
>> if(A|B), disp('1'), else disp('0'), end   % ponowne „skrócenie”
1                                            % skoro każdy el A jest 1
```

http://www.mathworks.com/help/techdoc/matlab_prog/f0-40063.html#f0-38959

Operatory logiczne (3)

- Operatory bitowe (*bit-wise*)
  - zdefiniowane na tablicach nieujemnych liczb całkowitych
  - np.

```
A = uint8(28);                % binarne 11100
B = uint8(21;                 % binarne 10101

bitand(A, B)      % wynik: 20, binarne 10100

bitor(A, B)       % wynik: 29, binarne 11101

bitxor(A, B)      % wynik:  9, binarne 01001

bitcmp(A, 5)      % wynik:  5, binarne 00011
                  % dopełnienie na 5 bitach

bitcmp(A)         % wynik:227, bin. 11100011
                  % dopełnienie na 8 bitach
```

http://www.mathworks.com/help/techdoc/matlab_prog/f0-40063.html#f0-39090

Następny wykład

- Optymalizacja obliczeń
- Grafika w Matlabie