Informatyka

Wykład 3

2016/2017z

Plan

- Efektywne programowanie
  - Prealokacja tablic
  - Wektoryzacja
  - Dobre praktyki w programowaniu
- Złożone typy danych
  - Zmienne typu komórkowego - „cell”
  - Struktury
- Operacje czytania i zapisywania plików
- Przetwarzanie tekstu

Prealokacja tablic (1)

- MATLAB tworząc tablice szuka w pamięci komputera odpowiednio dużego wolnego obszaru
  - Przy zmianie rozmiarów tablicy MATLAB szuka nowego, większego obszaru i kopiuje do niego zawartość modyfikowanej tablicy

```
display('Bez prealokacji')
tic                     %rozpoczęcie pomiaru czasu
for (k=1:10^6)
    tab(k)=k^2;         %każdy krok powiększa…
                        % tablice „tab” o jeden element
end
toc                     %zakończenie pomiaru czasu
```

Elapsed time is 0.248240 seconds.

Prealokacja tablic (2)

- Prealokacja tablic – zarezerwowanie miejsca w pamięci komputera zanim będzie potrzebne.

```
display('Bez prealokacji')
tic
for (k=1:10^6)
    tab(k)=k^2;    %każdy krok powiększa…
                   % tablice „tab” o jeden element
end
toc
```

Bez prealokacji
Elapsed time is 0.248240 seconds.

```
display('Prealokacja')
tic
tab_pre=zeros(1,10^6); %prealokacja tablicy
for (k=1:10^6)
    tab_pre(k)=k^2; %wpisanie wartości do istniejącej tablicy
end
toc
```

Prealokacja
Elapsed time is 0.012111 seconds

20 x szybciej !!

Wektoryzacja (1)

- MATLAB – jest zoptymalizowany do działań na macierzach i wektorach
- Wektoryzacja – zamiana kodu działającego w oparciu o pętle i bazującego na operacjach jedno-elementowych na kod wykorzystujący działania macierzowe, tablicowe
- Korzyści z wektoryzacji w MATABIE:
  - **Szybkość obliczeń !**
  - **Przejrzystość** (formuły bardziej podobne do formuł książkowych)
  - **Większa odporność na błędy** (krótszy kod = mniej okazji do błędów)

Wektoryzacja – przykład 1

- Obliczenie wartości *sinus* dla $x \epsilon [0,10]$

```
%Kod w oparciu o pętlę „for”, operujący
%na pojedynczych elementach

i = 0;
for t = 0:.01:10
   i = i + 1;          %zmiana indeksu
   y(i) = sin(t);      %wyliczenie wartości dla pojedynczego
                       %elementu z wektora „t”
end
```

```
%Kod zwektoryzowany

t=0:.01:10;
y=sin(t);
```

Wektoryzacja – przykład 2

- Obliczenie odchylenia standardowego z próby

```
N=1000;
x=rand(1,N);
```

$$s = \sqrt{\frac{\sum_{i=1}^{N} (x_i - \bar{x})^2}{N-1}}$$

71.5 : 3.5 : 1

```
suma=0;
disp('Petla for')
tic
for (k=1:N)
   s=(x(k)-mean(x))^2; %obliczanie elementów licznika
   suma=suma+s;        %sumowanie licznika
end
s=sqrt(suma/(N-1)) %pierwiastek
toc
```

```
Elapsed time is 0.015773 seconds.
```

```
disp('Wektoryzcja')
tic
s_wek=sqrt(sum((x-mean(x)).^2)/(N-1))
toc
```

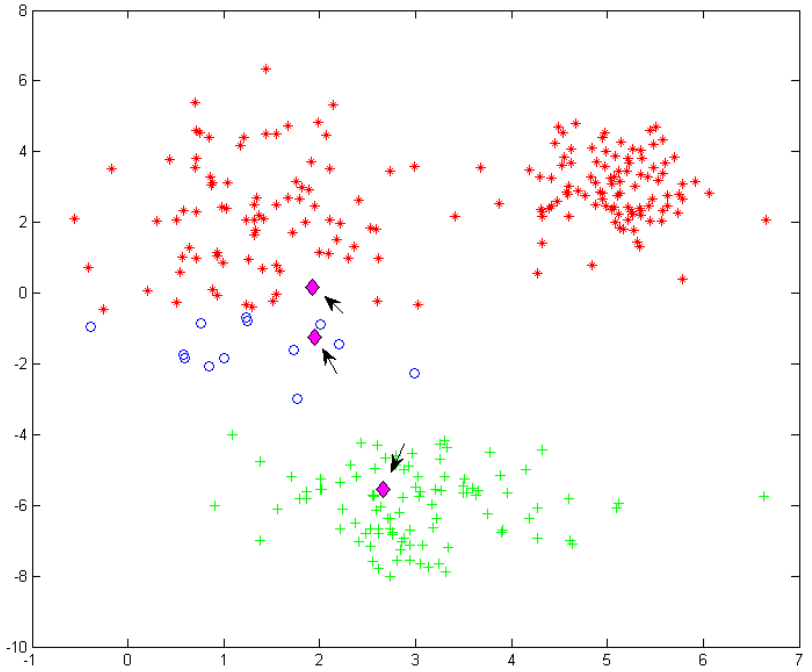
```
Elapsed time is 0.000762 seconds.
```

```
disp('Funkcja_wbudowana')
tic
s_wbd=std(x)
toc
```

```
Elapsed time is 0.000220 seconds.
```

Wektoryzacja – przykład 3 (alg. K-średnich)

- Algorytm K-średnich dzieli zbiór elementów na K podzbiorów (klastrów). Element należy do klastra, którego środek położony najbliżej niego.
  1. Wylosuj K początkowych centroidów
  2. Oblicz odległości każdego punktu do każdego centroidu
  3. Przyporządkuj punkty do ich klastrów
  4. Wylicz współrzędne nowych centroidów
  5. Powtarzaj 2-4, do momentu, gdy centroidy nie zmienią swojego położenia w dwóch kolejnych przebiegach



Początkowy podział punktów w algorytmie K-średnich, K=3

Wektoryzacja – przykład 3 (bez wektoryzacji)

```
function [centroids,label]=Przyklad_KMeans(K,data)
%Przyklad_Kmeans(liczbaKlastrow, macierzDanych)

Ndata=length(data); %dane znajduja sie w tablicy „data” o wymiarach (Ndata x 2)
centrInd=randi([1 Ndata],1,3);      %zainicjowanie centroidow
centroids=data(centrInd,:);         %macierz z wspolrzednymi centroidow
distMat=zeros(Ndata,K);             %prealokacja macierzy z odelglosciami
tic
for (w=1:Ndata)
   for (k=1:K)
   distMat(w,k)=sqrt((data(w,1)-centroids(k,1))^2+... %obliczenie odleglosci punktu „w”
                     (data(w,2)-centroids(k,2))^2);  %od centroidu „k”
   end
end
toc
tic
for (w=1:Ndata)
   label(w)=find(distMat(w,:)==min(distMat(w,:)));   %ktory centroid jest najblizej
end
toc
```

Elapsed time is 0.019733 seconds.

Elapsed time is 0.064164 seconds.

```
%kod do wizualizacji
figure
plot(data(find(label==1),1),data(find(label==1),2),'*r',...
   data(find(label==2),1),data(find(label==2),2),'ob',...
   data(find(label==3),1),data(find(label==3),2),'+g',...
   centroids(:,1),centroids(:,2),'+k');
%figure
%hist(label)
```

Gdyby pominąć prealokację to 0.615368 !!

Testy dla zbioru 30 tys punktów

Wektoryzacja – przykład 3 (wektoryzacja)

```
function [centroids,label]=Przyklad_Kmeans_wek(K,data)
%Przyklad_Kmeans_wek(liczbaKlastrow, macierzDanych)
Ndata=length(data);
centrInd=randi([1 Ndata],1,3); %zainicjowanie centroidow
centroids=data(centrInd,:);
distMat=zeros(Ndata,K);
tic
for (k=1:K) %zwektoryzowany sposób obliczania odleglosci od k-tego centroidu
   distMat(:,k)=sqrt((data(:,1)-centroids(k,1)).^2+(data(:,2)-centroids(k,2)).^2);
end
toc
```

Elapsed time is 0.002022 seconds.

1 : 9.7

```
tic
 [~,label]=min(distMat,[],2); %tablicowe szukanie numeru najbliższego centroidu
 toc
```

Elapsed time is 0.000534 seconds

1: 120 !!!

```
for (w=1:Ndata)
   for (k=1:K)
        distMat(w,k)=sqrt((data(w,1)-centroids(k,1))^2+...
                                (data(w,2)-centroids(k,2))^2);
    end
end
```

Elapsed time is 0.019733 seconds.

```
for (w=1:Ndata)
   label(w)=find(distMat(w,:)==min(distMat(w,:)));
end
```

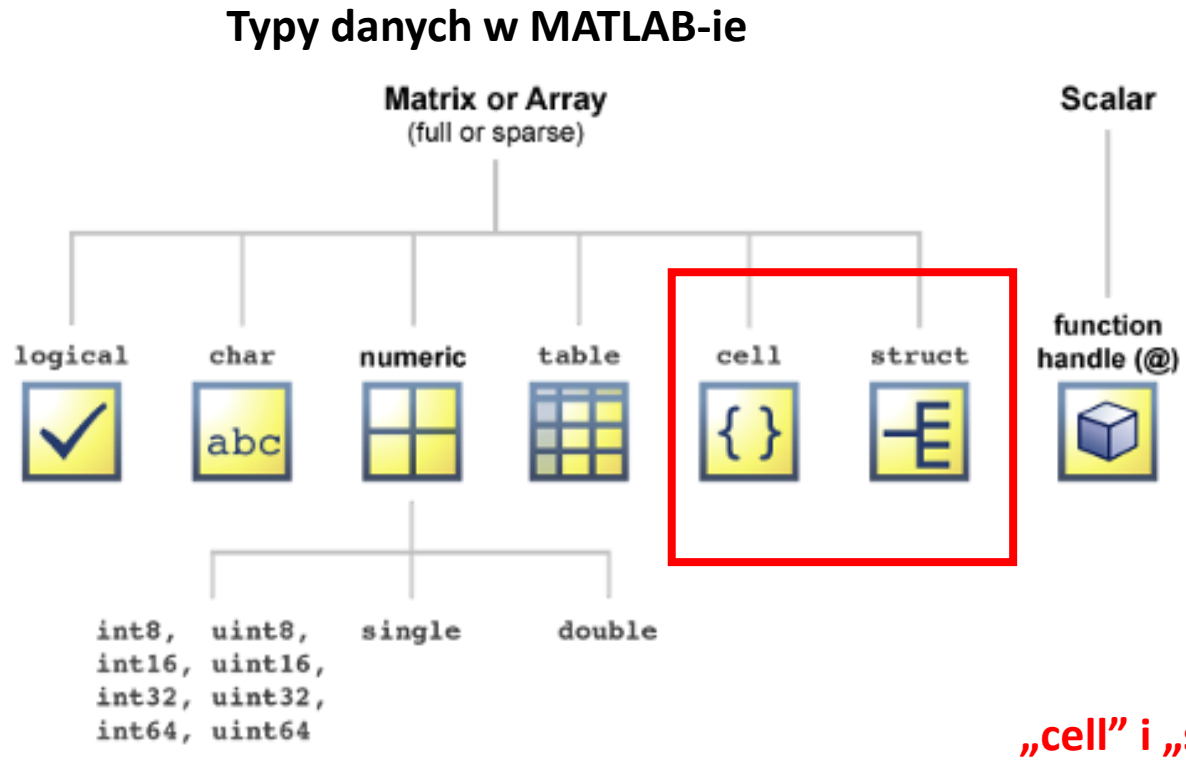
Elapsed time is 0.064164 seconds.

Testy dla zbioru 30 tys punktów

Zasady efektywnego stylu programowania

- Prealokacja pamięci oraz wektoryzacja !!
- Unikanie wielokrotnej realizacji tych samych poleceń (szczególnie wykonywanych przez system operacyjny np. operacje na dysku: zapis, odczyt z pliku)
- Tworzenie własnych M-plików tak, aby działały zarówno na wielkościach skalarnych jak i macierzowych
- Tworzenie M-plików tak, aby mogły być wykorzystywane w innych programach:
  - Ograniczenie funkcji do realizacji konkretnego zadania (np. tylko liczenie pola trójkąta)
  - Korzystanie z funkcji, a nie skryptów
  - Stosowanie komentarzy:
    - bezpośrednio pod nagłówkiem funkcji (zastosowanie, argumenty wej/wyj)
    - wewnątrz (objaśniające kod)
  - Stosowanie znaczących nazw funkcji oraz zmiennych (np. *ObliczParam*, zamiast *Zad3*)

Złożone typy danych – „cell” i „struct”

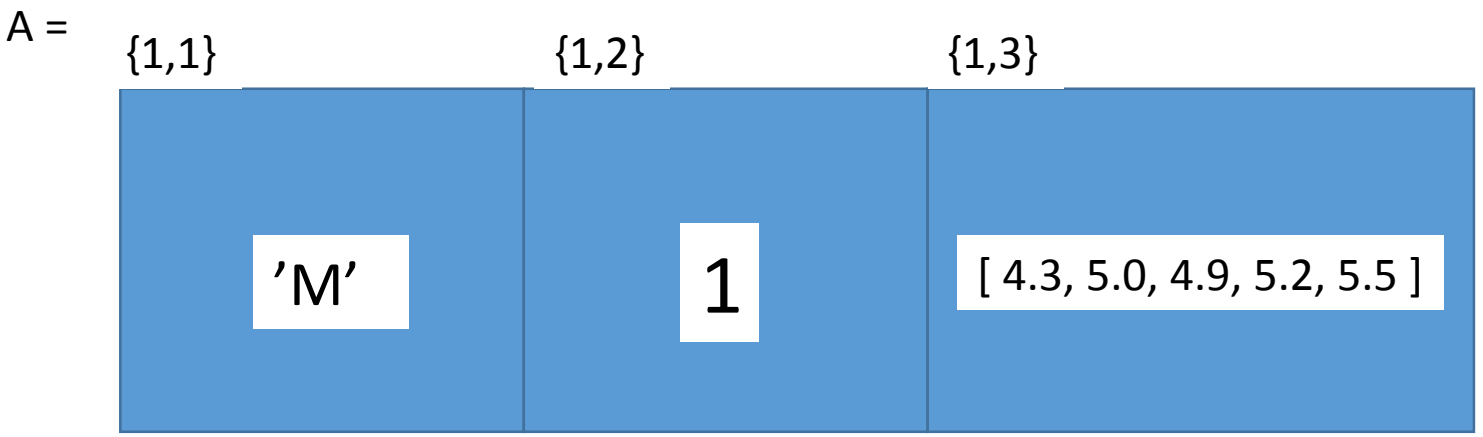


„cell” i „struct” - kontenery niejednorodne (ang. Heterogenous container)

http://www.mathworks.com/help/matlab/matlab_prog/fundamental-matlab-classes.html

Tablice komórkowe (ang. cell arrays)

- Grupują dane dowolnego typu w indeksowanych polach
- Pojedyncze pole to komórka, w której może być umieszczona dowolna zmienna
- Zastosowania:
  - Listy łańcuchów znakowych różnych długości.
  - Przechowywanie danych mieszanych – zmienne znakowe i numeryczne
  - Przechowywanie macierzy różnych rozmiarów



Tablice komórkowe - przykłady

- Definiowanie:

```
>> A = {'M', 1, [4.3, 5.0, 4.9, 5.2, 5.5 ]}  % przecinki oddzielają kolumny, a średniki wiersze
A =
    'M'    [1]    [1x5 double]
```

- Odwoływanie się do obszarów tablicy:

```
>> A(1:2) % wynik to tablica komórkowa
ans =
    'M'    [1]
```

- Odwoływanie się do zawartości komórek (wyłuskanie zawartości):

```
>> A{1,3}(3:end) %wynik to zmienna tego samego typu, co elementu ze sprawdzanej komórki
ans =
    4.9000    5.2000    5.5000
```

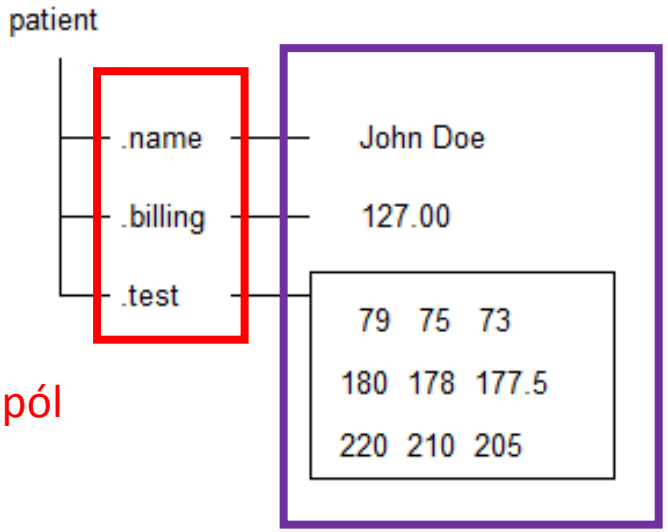
Tablice komórkowe – przykład 2 (varargin)

- varargin – specjalna tablica komórkowa, która może zbierać wszystkie, bądź część argumentów wejściowych funkcji

```
function [ prw ] = PierwWiel( varargin )
%[pierwiastki]=PierwWiel(wspolczynikA,wspolczynnikB,wspolczynnikC,...)
%Funkcja liczy pierwiastki wielomianu o współczynnikach podanych w %argumentach
wejściowych

p=zeros(1,nargin)
for k=1:length(varargin)        %<=> 1:nargin
   p(k)=varagin{k}              %do k-tego elementu wektora „p” zapisywany jest
                                %element wyłuskany z k-tej komórki tablicy „varargin”
end
prw=roots(p)
end
```

Struktury

- Grupują dane dowolnego typu w polach zdefiniowanych przez użytkownika
- Umieszczenie danych w jednej strukturze ma sens, gdy dane są ze sobą związane



Nazwy pól

Zawartość pól

http://www.mathworks.com/help/matlab/matlab_prog/create-a-structure-array.html

Struktury – przykład 1 (definiowanie)

- Definiowanie struktury:
  - Przypisanie wartości do pól struktury

```
patient.name = 'John Doe';          %powstaje struktura o nazwie „patient”, ktora posiada pole „name”
                                    %do ktorej wpisany jest lancuch znakow 'John Doe'
patient.billing = 127.00;           %w strukturze pojawia sie pole „billing” o zadanej zawartosci
patient.test = [79, 75, 73;…        %w strukturze pojawia sie pole „test” przechowujace macierz
                180, 178, 177.5;…
                220, 210, 205];
patient                             %wyswietlnie calosci
```

  - Z wykorzystaniem funkcji ***„struct”***

```
patient = struct('name','John Doe','billing',127, 'test', ... % Kolejno podawane są ’nazwa pola’ i ’wartosc do przypisania’
      [79, 75, 73; 180, 178, 177.5; 220, 210, 205])
```

Struktury – przykład 1 (odwoływanie się do pól)

- Odwołanie się do pól odbywa się przy użyciu operatora „.”
  - *nazwaStruktury.nazwaPola*

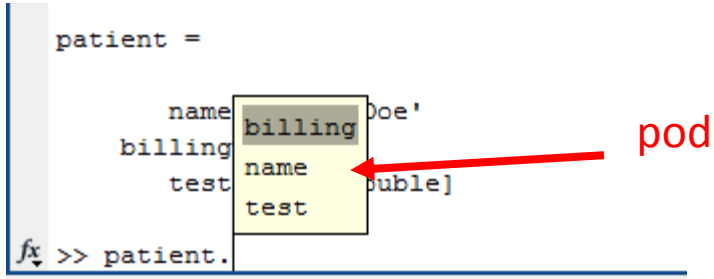
```
>> patient.billing % odniesienie do pola „billing”
ans =
  127
```

Operator „.”

- Częsty błąd – literówka w nazwie pola

```
>> patient.biling
Reference to non-existent field 'biling'.
```

Rozwiązanie: korzystać z tabulatora, gdy kursor jest w linii poleceń – dostajemy podpowiedzi o możliwych opcjach wyboru



podpowiedzi

Struktury – przykład 1 – błędne pola

- Literówka przy podawaniu wartości

```
>> patient.biling = 10 % co się stanie?? Odp za punkt z aktywnosci

 patient =
       name: 'John Doe'
       billing: 127
       test: [3x3 double]
       biling: 10 %pojawilo się nowe, bledne pole; dlaczego?
```

- Usunięcie błędnego pola – funkcja *rmfield*

```
 >> patient=rmfield(patient,'biling') % funkcja operuje na kopii zmiennej, wiec wynik dzialania
 patient =                            % trzeba ponownie przypisac
       name: 'John Doe'
    billing: 127
       test: [3x3 double]
```

Struktury – konstruktor

- Konstruktor – funkcja, która tworzy precyzyjnie zdefiniowaną strukturę (obiekt) korzystając z podanych argumentów wejściowych.

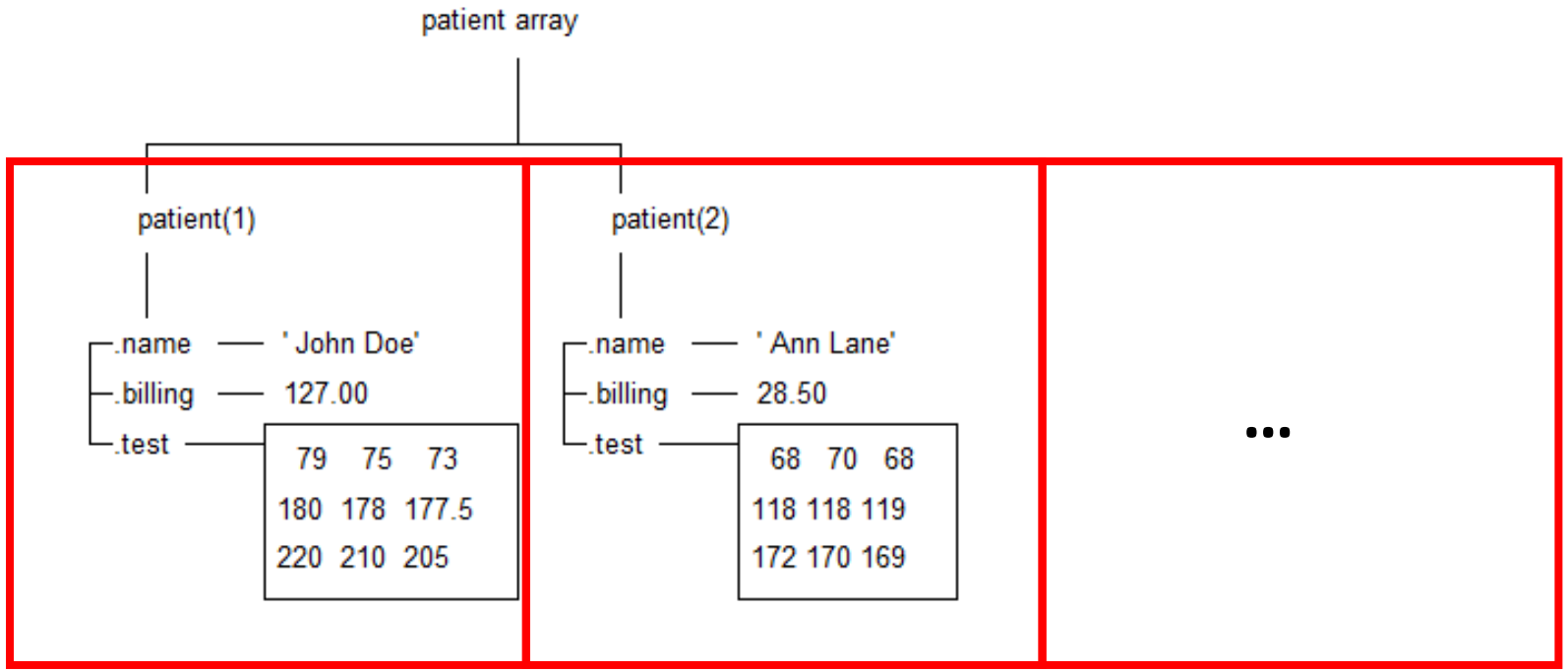
```
function [ patient ] = CreatePatient( nameValue, billingValue, testValue)
%pacjent=CreatePatient('Imie Nazwizko',rachunek, wynikiBadan)
%Funkcja tworzy strukture o polach 'name','billing', 'test'

patient = struct('name',nameValue,'billing',billingValue,'test',testValue);

end
```

```
>> p1=CreatePatient('Jan Kowalski',160, [20 30 28.0])
p1 =
      name: 'Jan Kowalski'
      billing: 160
      test: [20 30 28]
```

Korzystanie z konstruktorów eliminuje problem pomyłek przy tworzeniu struktur

Tablice struktur

- Struktury mogą być zorganizowane w tablicy



http://www.mathworks.com/help/matlab/matlab_prog/create-a-structure-array.html

Tworzenie tablic struktur (1)

- Przez dopisywanie kolejnych struktur do tablicy:

```
>>patientList(1)=struct('name','Adam Adamski', 'billing',220, 'test', [23 15 19]);
>>patientList(2)=CreatePatient('Myszon Kotkowski', 180, [19.1 28 31]);
>>patientList          %wyświetlenie zawartości tablicy patientList
patientList =
1x2 struct array with fields:
    name
    billing
    test
```

%z uzyciem funkcji ***struct***

% z uzyciem ***konstruktora***

Nazwa pola z jednym „l”

```
>> patientList(3)= struct('name','Adam Adamski', 'biling',220, 'test', [23 15 19]);
Subscripted assignment between dissimilar structures.
```

- Pola struktur w tablicy muszą być takie same:
  - Tablicę mogą utworzyć struktury o takich samych polach
  - Do tablicy można dołączyć strukturę, która ma pola spójne z tymi które już są w tablicy

Tworzenie tablic struktur (2) – *cell2struct*

- Do tworzenia struktur i tablic struktur można wykorzystać funkcję ***cell2struct(zawartosc, nazwyPol, wymiar)***

```
>> zawartosc={'Zyraf Lewski', 160, [21 31 45];…
                    'Maciej Mackowski', 155, [100 41 1]}
zawartosc =

   'Zyraf Lewski'       [160]    [1x3 double]
   'Maciej Mackowski'   [155]    [1x3 double]

>> nazwyPol={'name','billing','test'}
nazwyPol =

   'name'    'billing'    'test,
```

Dane 1

Dane 2

Nazwy pól

```
>> newPatients=cell2struct(zawartosc,nazwyPol,2)
newPatients =

2x1 struct array with fields:

   name
   billing
   test
```

2 oznacza, że dane dla jednej struktury są umieszczone w kolejnych kolumnach

Prosty zapis i odczyt danych

- Save / load

```
>> save('Moje_dane')            % zapisuje wszystkie zmienne do plik „Moje_dane.mat”
>> clear all
>> load('Moje_dane')            % wczytuje zmienne z pliku
>>save(’Moja_zmienne’, ’wyniki’, ’daneWej’) % zapisuje tylko wybrane zmienne
```

- Praca z plikami MS Excel

```
[num, txt, raw]=xlsread('ExcelTry.xlsx') % wczytuje liczby z pliku do zmiennej „num”, tekst do zmiennej „txt”
                                         %oraz calosc do tablicy komórkowej „raw”

>> a=[1:10]; xlswrite(’XlsWriteTry',a)    %zapisuje „a” do pliku
>> pacjent=CreatePatient('AA',20,[1 2 3])
>> xlswrite('Write_try',pacjent)          %funkcja xlswrite nie obsługuje struktur
Error using xlswrite (line 166)
Input data must be a numeric, cell, or logical array.
```

Praca z plikami tekstowymi

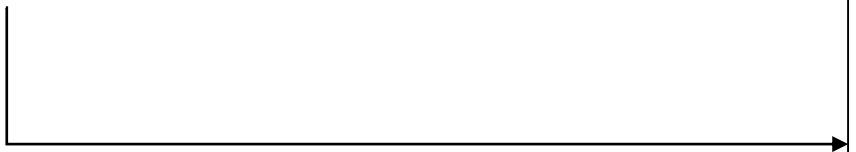
- Dane w plikach mogą być ustrukturyzowane na wiele sposobów:
  - Dane numeryczne o jednorodnej strukturze:

0.4779	0.5018	0.6440	0.1946
0.5096	0.5317	0.6210	0.2401
0.5493	0.4940	0.1692	0.4165

  - Dane alfanumeryczne o jednorodnej strukturze

Janowski	Adam	20	A.Nowak	2012-02-13
Miller	Anna	32	D.Zuch	2011-12-30

  - Dane alfanumeryczne o niejednorodnej strukturze



```
<tbody>
   <tr>
      <td>Kapitalizacja</td><td>26 220 000 000 zł</td>
   </tr><tr>
      <td>Free float</td><td>68,21%</td>
   </tr><tr>
      <td>Liczba akcji</td><td>200 000 000 szt.</td>
   </tr><tr>
      <td>Dywidenda 2013-07-12</td><td>4,90 zł</td>
   </tr><tr>
      <td>Dywidenda (%)</td><td>3,90%</td>
   </tr><tr>
      <td>Cena / Zysk</td><td>11,20</td>
   </tr><tr>
      <td>Cena / WK</td><td>1,13</td>
   </tr>
</tbody>
```

Dane numeryczne, jednorodna struktura

0.4779	0.5018	0.6440	0.1946
0.5096	0.5317	0.6210	0.2401
0.5493	0.4940	0.1692	0.4165

- Wczytywanie:
  - dlmread

```
>> M=dlmread('Sample_numeric_file_tab_separated.txt','\t',[0 0 2 3]) % kolumny rozdzielone znacznikiem
```

Znacznikiem jest tabulator

  - csvread

```
>>CM=csvread('SampleNumericFile_CommaSep.txt')          %kolumny rozdzielone przecinkami
```

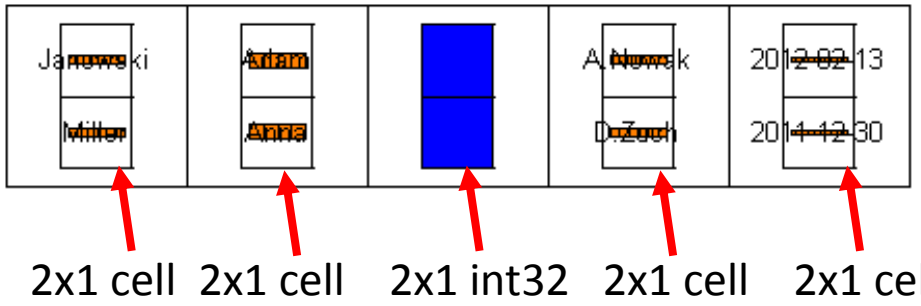
- Zapis:
  - dlmwrite
  - csvwrite

Wczytywanie danych alfanumerycznych z pliku o jednorodnej strukturze

Nazwisko	Imię	Wiek	Lekarz	OstataniaWizyta
Janowski	Adam	20	A.Nowak	2012-02-13
Miller	Anna	32	D.Zuch	2011-12-30

- ***textscan(fid, format, N, 'ParamName1' , ParamVal1, …)***

```
plikID=fopen('BazaPacjentow.txt')                          %otwarcie pliku
nazwyKol=textscan(plikID,'%s %s %s %s %s',1)               %pobranie nazw kolumn
baza=textscan(plikID,'%s %s %d %s %s','HeaderLines',1)     %wczytanie danych
cellplot(baza)
```



Każda kolumna to element tablicy komórek o wymiarach (1xLiczbaKolumn)

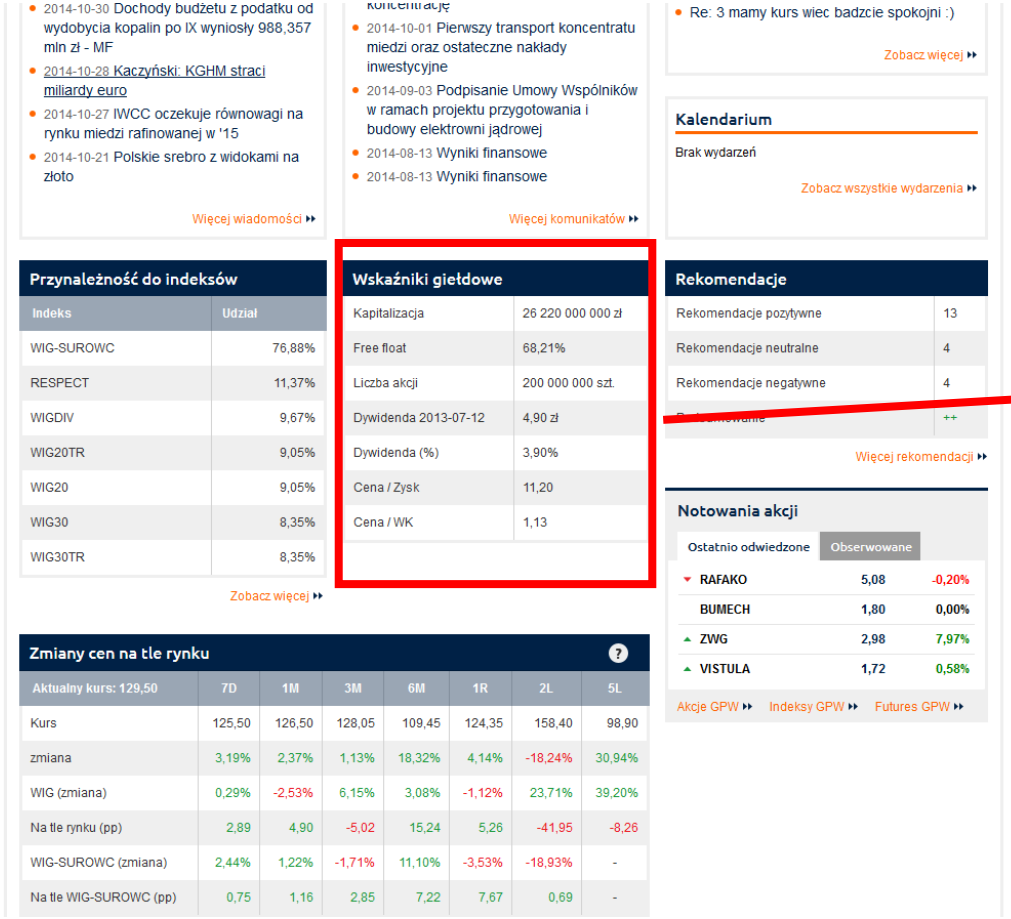
Parametry funkcji ***textscan***

- ***textscan(fid, format, N, ’ParamName1’ , ParamVal1, …)***

baza=textscan(plikID,'%s %s %d %s %s','HeaderLines',1)

  - ***format*** – łańcuch znaków do którego funkcja próbuje dopasować wczytane fragmenty pliku; %X oznacza pole danych
    - pola numeryczne: %d, %d8, … - liczby całkowite, %u, %u8, … - liczby całkowite bez znaku, %f, %f32, %f64, %n – liczby zmiennoprzecinkowe (typ single i double)
    - Pola tekstowe: %s (string), %q (pomija „”), %c (dowolny znak)
      %[…] – czyta znaki tylko zgodne z tymi umieszczonymi w […]
      %[^…] – czyta znaki do momentu pierwszej pasującej do […]
    - Pominięcie: %* - pominięcie pola, %*n – pominięcie n znaków
  - ***N*** – liczba cykli czytania do powtórzenia
  - Inne parametry: ’HeaderLines’, liczba; ,’Delimiter’,’znak’, ’CollectOutput’, 0/1, …

Wczytywanie danych alfanumerycznych z pliku o niejednorodnej strukturze (1)



- 2014-10-30 Dochody budżetu z podatku od wydobycia kopalin po IX wyniosły 988,357 mln zł - MF
- 2014-10-28 Kaczyński: KGHM straci miliardy euro
- 2014-10-27 IWCC oczekuje równowagi na rynku miedzi rafinowanej w '15
- 2014-10-21 Polskie srebro z widokami na złoto

Więcej wiadomości

- 2014-10-01 Pierwszy transport koncentratu miedzi oraz ostateczne nakłady inwestycyjne
- 2014-09-03 Podpisanie Umowy Wspólników w ramach projektu przygotowania i budowy elektrowni jądrowej
- 2014-08-13 Wyniki finansowe
- 2014-08-13 Wyniki finansowe

Więcej komunikatów

- Re: 3 mamy kurs wiec badzcie spokojni :)

Zobacz więcej

Kalendarium

Brak wydarzeń

Zobacz wszystkie wydarzenia

Przynależność do indeksów

Indeks	Udział
WIG-SUROWC	76,88%
RESPECT	11,37%
WIGDIV	9,67%
WIG20TR	9,05%
WIG20	9,05%
WIG30	8,35%
WIG30TR	8,35%

Zobacz więcej

Wskaźniki giełdowe

Kapitalizacja	26 220 000 000 zł
Free float	68,21%
Liczba akcji	200 000 000 szt.
Dywidenda 2013-07-12	4,90 zł
Dywidenda (%)	3,90%
Cena / Zysk	11,20
Cena / WK	1,13

Rekomendacje

Rekomendacje pozytywne	13
Rekomendacje neutralne	4
Rekomendacje negatywne	4

Więcej rekomendacji

Notowania akcji

Ostatnio odwiedzone | Obserwowane

RAFAKO	5,08	-0,20%
BUMECH	1,80	0,00%
ZWG	2,98	7,97%
VISTULA	1,72	0,58%

Akcje GPW | Indeksy GPW | Futures GPW

Zmiany cen na tle rynku

Aktualny kurs: 129,50	7D	1M	3M	6M	1R	2L	5L
Kurs	125,50	126,50	128,05	109,45	124,35	158,40	98,90
zmiana	3,19%	2,37%	1,13%	18,32%	4,14%	-18,24%	30,94%
WIG (zmiana)	0,29%	-2,53%	6,15%	3,08%	-1,12%	23,71%	39,20%
Na tle rynku (pp)	2,89	4,90	-5,02	15,24	5,26	-41,95	-8,26
WIG-SUROWC (zmiana)	2,44%	1,22%	-1,71%	11,10%	-3,53%	-18,93%	-
Na tle WIG-SUROWC (pp)	0,75	1,16	2,85	7,22	7,67	0,69	-

Źródło strony

```
<tbody>
   <tr>
      <td>Kapitalizacja</td><td>26 220 000 000 zł</td>
   </tr><tr>
      <td>Free float</td><td>68,21%</td>
   </tr><tr>
      <td>Liczba akcji</td><td>200 000 000 szt.</td>
   </tr><tr>
      <td>Dywidenda 2013-07-12</td><td>4,90 zł</td>
   </tr><tr>
      <td>Dywidenda (%)</td><td>3,90%</td>
   </tr><tr>
      <td>Cena / Zysk</td><td>11,20</td>
   </tr><tr>
      <td>Cena / WK</td><td>1,13</td>
   </tr>
</tbody>
```

http://www.bankier.pl/inwestowanie/profile/quote.html?symbol=KGHM

Wczytywanie danych alfanmumerycznych z pliku o niejednorodnej strukturze (2)

- Plik musi być analizowany linia po linii

```
plikID=fopen('zrodloStrony.txt');
k=0;                                  %licznik wpisow
expr='<td>(.*)</t.*>(.*)</td>';       %poszukiwane wyrazenie regularne
while ~feof(plikID)                   %do poki nie koniec pliku
   str=fgetl(plikID);                 %wczytuje pojedyncza linie
   fragmenty=regexp(str,expr,'tokens'); %zwraca "zetony" dopasowanego wyrazenia
   if (~isempty(fragmenty))
      k=k+1;
      parametr(k,1)={fragmenty{1}{1}}; %wpisanie nazwy parametru do tablicy
      parametr(k,2)={fragmenty{1}{2}}; %wpisanie wartości parametru do tablicy
   end
end
parametr                  %wyswietlenie wczytanych wartosci
fclose(plikID);
```

```
<tbody>
   <tr>
      <td>Kapitalizacja</td><td>26 220 000 000 zł</td>
   </tr><tr>
      <td>Free float</td><td>68,21%</td>
   </tr><tr>
      <td>Liczba akcji</td><td>200 000 000 szt.</td>
   </tr><tr>
      <td>Dywidenda 2013-07-12</td><td>4,90 zł</td>
   </tr><tr>
      <td>Dywidenda (%)</td><td>3,90%</td>
   </tr><tr>
      <td>Cena / Zysk</td><td>11,20</td>
   </tr><tr>
      <td>Cena / WK</td><td>1,13</td>
   </tr>
</tbody>
```

http://www.bankier.pl/inwestowanie/profile/quote.html?symbol=KGHM

Wczytywanie danych alfanmumerycznych z pliku o niejednorodnej strukturze (3)

- Nazwy i parametry zostaną wczytane do tablicy komórkowej jako łańcuchy znaków

parametr =

'Kapitalizacja'	'26 220 000 000 zł'
'Free float'	'68,21%'
'Liczba akcji'	'200 000 000 szt.'
'Dywidenda 2013-07-12'	'4,90 zł'
'Dywidenda (%)'	'3,90%'
'Cena / Zysk'	'11,20'
'Cena / WK'	'1,13'

```
<tbody>
   <tr>
      <td>Kapitalizacja</td><td>26 220 000 000 zł</td>
   </tr><tr>
      <td>Free float</td><td>68,21%</td>
   </tr><tr>
      <td>Liczba akcji</td><td>200 000 000 szt.</td>
   </tr><tr>
      <td>Dywidenda 2013-07-12</td><td>4,90 zł</td>
   </tr><tr>
      <td>Dywidenda (%)</td><td>3,90%</td>
   </tr><tr>
      <td>Cena / Zysk</td><td>11,20</td>
   </tr><tr>
      <td>Cena / WK</td><td>1,13</td>
   </tr>
</tbody>
```

Wyrażenia regularne – podstawy regexp

- Wyrażenie regularne to krótki łańcuch znaków, który służy wyszukiwania i modyfikacji fragmentów tekstu.

```
str='Ala ma kota.’;   %łańcuch znaków do przeszukania
expr=’a’;             %najprostsze wyrażenie regularne / wzorzec
match=regexp(str,expr,'match') %chcemy otrzymać pełne dopasowanie
match =

    'a'    'a'    'a'
```

'Ala ma kota’

We wzorcu mamy 3 „a”

- Wyrażenia mogą zawierać dowolne znaki z wyjątkiem 12 znaków specjalnych, które powinny być poprzedzone „\” (backslash):
  - [, (,), {, +, *, ., ^, \, ?, $, |

expr=’a\.’ ⟶ match = 'a.'

Złożone wzorce – zakresy i zakotwiczenia

- Znaki specjalne pozwalają tworzyć bardziej złożone wzorce

zakresy

Znak specjalny	Przykład	Dopasowanie
[] – grupy możliwych znaków	'[a-z]a' % dowolna mała litera, następnie „a”	'Ala ma jabłka.'
[^] – grupy wykluczonych znaków	'[^a-l]a' % znak inny niż litery „a” do „l” , następnie „a”	'Ala ma jabłka.'
. – dowolny znak	'... ' % trzy dowolne znaki i spacja	'Ala_ma jabłka.'

Złożone wzorce – zakresy i zakotwiczenia

- Znaki specjalne pozwalają tworzyć bardziej złożone wzorce

	Znak specjalny	Przykład	Dopasowanie
zakresy	[] – grupy możliwych znaków	'[a-z]a' % dowolna mała litera, następnie „a”	'Ala ma jabłka.'
zakresy	[^] – grupy wykluczonych znaków	'[^a-l]a' % znak inny niż litery „a” do „l” , następnie „a”	'Ala ma jabłka.'
zakresy	. – dowolny znak	'... ' % trzy dowolne znaki i spacja	'Ala_ma jabłka.'
zakotwiczenia	^ - znak początku linii	'^[A-Za-z] ' %jedna litera na początku linii	'Ala ma jabłka.'
zakotwiczenia	$ - znak końca linii	'[a-z][a-z]\.$' %dwie małe litery i kropka na końcu linii	'Ala ma jabłka.'
zakotwiczenia	\< - początek słowa	'\<j..' %”j” na początku słowa i dwa dowolne znaki	'Ala ma jabłka.'
zakotwiczenia	\> - koniec słowa	'\<.a\>', % dowolny znak na początku wyrazu, następnie „a” na końcu wyrazu	'Ala ma jabłka.'

Złożone wzorce – klasy znaków i zwielokrotnia

	Znak specjalny	Przykład	Dopasowanie
klasy znaków	\s – biała spacja	'a\s' %”a” i spacja	'Ala ma jabłka.'
	\S – nie biała spacja	'a\S' %”a” i nie spacja	'Ala ma jabłka.'
	\d – cyfra \D – nie cyfra \w – wyraz \W – nie wyraz		
zwielokrotnienia	{N} – dokładnie N znaków	'[a-z]{2}' %dwie małe litery	'Ala ma jabłka.'
	{Nmin,Nmax} – od Nmin do Nmax	'[a-z]{3,5} ' %3 do pięciu małych liter	'Ala ma jabłka.'
	* - dowolna ilość znaków (chciwy)	{'^.* '} %dowolna ilość dowolnych znaków, a na końcu spacja (do ostatniej spacji)	'Ala ma jabłka.'
	*? – dowolna ilość znaków niechciwy	{'^.*? '} %dowolna ilość dowolnych znaków, a na końcu spacja (do pierwszej spacji)	'Ala ma jabłka.'
	+ - przynajmniej 1, ? – 0 lub 1		

Wychwytywanie „żetonów” (ang. tokens)

- Żetony to fragmenty pasujących łańcuchów objęte nawiasami „()”

```
str='<td>Kapitalizacja</td><td>26 220 000 000 zł</td>';
expr='<td>(.*)</t.*>(.*)</td>';
fragmenty=regexp(str,expr,'tokens')
fragmenty{1}
```

Wychwytywanie „żetonów” (ang. tokens)

- Żetony to fragmenty pasujących łańcuchów objęte nawiasami „()”

```
str='<td>Kapitalizacja</td><td>26 220 000 000 zł</td>';
expr='<td>(.*)</t.*>(.*)</td>';
fragmenty=regexp(str,expr,'tokens')
fragmenty{1}

ans =
    'Kapitalizacja'    '26 220 000 000 zł'
```

Korzystanie z funkcji regexp

- Funkcja **regexp** zwraca wyniki w postaci tablic komórkowych
- W zależności od argumentów wejściowych regexp zwraca:

```
startInd=regexp(str,expr,’start’)                    %Indeksy poczatkow lancuchow pasujacych
endInd=regexp(str,expr,’end’)                        %indeksy koncowe
tokenIndRange=regexp(str,expr,'tokensExtens')        %zakres indeksow zetonow
match=regexp(str,expr,'match')                       % lancuchy pasujace
noMatch=regexp(str,expr,’split')                     % lancuchy nie pasujace
token=regexp(str,expr,'tokens')                      %zetony
tokenNames=regexp(str,expr,’names')                  %zetony, które maja nazwy
[ match, token ] =regexp(str,expr,'match’, ’tokens’) % kilka zmiennych wyjsciowych w kolejnosci
                                                     %podanych argumentow wejsciowych
```

Inne funkcje korzystające z wyrażeń regularnych

- regexpi – wersja regexp nierozróżniająca dużych i małych liter
- regexprep – funkcja podmienia dopasowane fragmenty

```
newString = regexprep(str,expr,replace)
str='Ala ma jabłka.'
newStr=regexprep(str,’jabłka’, ’gruszki')
newStr2=regexprep(str,'a\s','A ')
```

```
newStr =
Ala ma gruszki.

newStr2 =
AlA mA jabłka.
```

Inne operacje na łańcuchach znakowych

Nazwa funkcji	Działanie
strcmp	Porównanie dwóch łańcuchów
deblank	Usuwa spacje w łańcuchu
findstr	Znajduje jeden łańcuch w obrębie drugiego
strcat	Łączy ciąg łańcuchów w wiersz
strrep	Wyszukuje łańcuch i podmienia
lower	Zamiana liter na małe
upper	Zamiana liter na duże

Co było dzisiaj ważne?

- Prealokacja i wektoryzacja mogą wielokrotne przyspieszać obliczenia
- Zmienne typu „cell” i struktury pozwalają przechowywać niejednorodne dane
- Metoda odczytu plików powinna być dobrana w zależności od typu wczytywanych danych
- Wyrażenia regularne są uniwersalnym narzędziem do przetwarzania danych tekstowych