LABORATORIUM OPROGRAMOWANIA UŻYTKOWEGO

MATLAB

Opracowanie:
dr inż. Jacek Kucharski
dr inż. Piotr Urbanek

ĆWICZENIE 8

Środowisko i podstawowe elementy pakietu MATLAB

WIADOMOŚCI OGÓLNE

MATLAB jest **środowiskiem programowania.** Kiedy uruchamiamy MATLABa zgłasza się *Matlab Command Window* i poprzez ten mechanizm możemy komunikować się z interpreterem.
Po wprowadzeniu każdego polecenia MATLAB sprawdza kolejno:

1. Czy jest to zmienna
2. Jeżeli nie to czy jest to funkcja wbudowana MATLABa
3. Jeżeli nie to czy jest taka funkcja (typu M-pliku, MEX-pliku albo DLL) w bieżącym katalogu
4. Jeżeli nie to czy jest taka funkcja w ścieżce dostępu MATLABa

Ścieżka dostępu MATLABa zapisana jest w pliku *pathdef.m,* Ścieżkę tę można sprawdzać i modyfikować poleceniami path, addpath, rmpath.

>>path - sprawdza ścieżkę
>>path('C:\.....', path) - dodaje nową ścieżkę do listy.

Można wrócić do poprzednio wykonywanych poleceń używając klawiszy ↑↓. Jeżeli wcześniej zostaną wpisane pierwsze litery poszukiwanego polecenia to zostanie odnalezione najbliższe na liście takie polecenie. Polecenie takie można edytować i ponownie wykonać.

MATLAB rozróżnia małe i wielkie litery. Wszystkie funkcje muszą być wprowadzane małymi literami.

Można z poziomu MATLABa wykonywać polecenia systemowe w następujący sposób:

>>!dir - otwarcie nowego okna, wykonanie polecenia i oczekiwanie na jego zakończenie
>>!dir& - otwarcie nowego okna i wykonanie polecenia w tle bez oczekiwania na jego zakończenie

Zakończenie pracy z programem jest równoznaczne z wymazaniem wszystkich zmiennych w przestrzeni roboczej. Całość lub część przestrzeni roboczej można zapisać w pliku dyskowym przed zakończeniem pracy w MATLABie:

>>save - zapisuje wszystkie zmienne w pliku *matlab.mat*
>>save temp - zapisuje wszystkie zmienne w pliku *temp.mat*
>>save temp X Y - zapisuje zmienne X i Y w pliku *temp.mat*

Przywrócenie przestrzeni roboczej po ponownym uruchomieniu programu uzyskuje się poleceniem load, np. load temp.
System pomocy uruchamiany jest następująco:

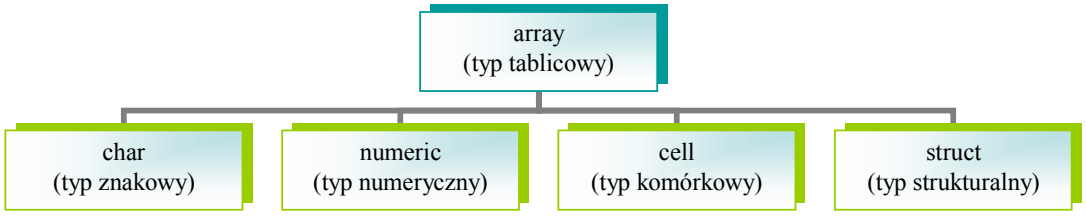
help – wyświetla zestawienie katalogów MATLAB-a wraz z ich opisem
help *nazwa polecenia* – podaje podpowiedź na temat żądanego polecenia, np. help ver
which nazwa_m-pliku – powoduje wyświetlenie ścieżki dostępu do m-pliku lub do m-funkcji

LICZBY, ZMIENNE I WYRAŻENIA

Zapis liczb jest typowy dla notacji matematycznej (dopuszczalne są cyfry, znaki +-, kropka dziesiętna oraz jednostka urojona i lub j dla liczb zespolonych). Należy pamiętać, aby przy pisaniu elementów macierzy unikać spacji w liczbach zespolonych i przed potęgą e (1 + 3i, 1.34 e-5). Jeżeli w danej przestrzeni roboczej używamy zmiennej i lub j do innych celów niż jednostka urojona to trzeba zdefiniować nową jednostkę urojoną, np. ii=sqrt(-1).

Typy danych

W MATLAB-ie zostały zdefiniowane następujące typy danych:



numeric- typ numeryczny podwójnej precyzji. Jest to podstawowy typ danych dla zmiennych MATLAB-a (wszystkie obliczenia dla zmiennych numerycznych są prowadzone w trybie podwójnej precyzji).
char – typ tablicowy znakowy lub łańcuchowy,
cell – typ komórkowy. Elementy tablic komórkowych mogą zawierać inne tablice,
struct – typ strukturalny. Odwołują się do nazw pól, które mogą zawierać inne tablice.

Zatem w środowisku MATLAB dostępne są:

- macierze,
- tablice wielowymiarowe,
- tablice komórkowe,
- tablice strukturalne,

Podstawowym typem zmiennych w MATLAB-ie jest macierz. W ogólnym przypadku jej elementami mogą być liczby zespolone oraz znaki. Jeżeli macierz ma jedną kolumnę lub jeden wiersz to jest to **wektor,** a jeśli wymiar macierzy jest 1x1 to mamy do czynienia ze **skalarem**.

Dwie zmienne mają specjalny charakter permanentny: ans - standardowa zmienna wynikowa i eps - zmienna tolerancji.
Ponadto zdefiniowane są wartości wyrażeń nieokreślonych tj Inf = ∞ jako wynik operacji dzielenia przez zero oraz NaN (*Not a Number*) dla wyrażeń typu 0/0 lub ∞/∞.
Nazwy zmiennych lub funkcji mogą składać się z liter i cyfr, ale muszą się rozpoczynać od litery.
Polecenie who podaje **aktualną listę zmiennych** w przestrzeni roboczej (whos dodatkowo podaje ich rozmiary).
Można usunąć zmienną z przestrzeni roboczej poleceniem clear, np. clear X.

Wyrażenie w MATLABie ma typową formę postaci:

```
variable=expression
```

lub po prostu

```
expression
```

Wyrażenie może składać się z operatorów, nazw funkcji i zmiennych.
Wynikiem wyrażenia jest macierz.
Jeżeli wyrażenie podane jest bez operacji przypisania (drugi z podanych sposobów) to następuje **automatyczne przypisanie** do zmiennej ans.
Jeżeli wyrażenie zakończone jest **średnikiem** (;) to wynik wyrażenia nie jest wyprowadzany na ekran (ale przypisanie wciąż ma miejsce).
Jeżeli wyrażenie jest tak długie, że **nie mieści się w jednej linii** to można je przerwać trzema kropkami (...) i kontynuować w następnej linii (już bez kropek).
W wyrażeniach wykorzystuje się **typowe operatory arytmetyczne**: +, -, *,^,/,\. Dwa ostatnie operatory to dzielenie prawo i lewostronne. Kropka dziesiętna poprzedzająca znak operatora (np. .*, ./) oznacza **operację tablicową** tj. element po elemencie.

Format wyświetlanych wyników może być określony przez użytkownika przez polecenie format.

OPERACJE NA MACIERZACH

Macierze jedno- i dwuwymiarowe.

Macierz może być zdefiniowana na kilka sposobów:

- **przez wyliczenie jej elementów**

W nawiasach kwadratowych podaje się elementy macierzy oddzielając wiersze średnikiem. Wyrazy w wierszu mogą być oddzielone spacją lub przecinkiem:

```
A=[1 2 3; 4,5,6]
```

Można też wyliczyć elementy układając je „graficznie” wewnątrz macierzy, gdyż sekwencja oznaczająca koniec wiersza (ENTER) jest traktowana jak średnik rozdzielający wiersze budowanej macierzy:

```
A=[1 2 3    (tu jest ENTER)
   4 5 6]
```

Możliwe jest też wykorzystanie trzech kropek dziesiętnych w celu kontynuacji wiersza w nowej linii, tak więc polecenie

```
A=[1 2 3 ...
   4 5 6]
```

jest równoważne poleceniu

```
A=[1 2 3 4 5 6]
```

- **przez wygenerowanie elementów**

Wykorzystujemy wyrażenie w ogólnej postaci:

```
min:krok:max
```

które generuje wektor wierszowy o następującej budowie:

```
[min, min+krok, min+2*krok, ..., max]
```

Jeżeli parametr krok jest pominięty to jest on domyślnie przyjmowany jako 1.

Macierze są też generowane jako efekt wyrażeń lub funkcji (np. **ones, zeros, randn)**.

- **przez zbudowanie z innych elementów**

Mając zdefiniowane macierze A, B, C można zbudować macierz D np. jako:

```
D=[A B; C]
```

co daje układ

$$D = \left[\begin{array}{cc} A & B \\ C & \end{array} \right]$$

- **przez mieszanie omówionych wyżej technik**

Przykład: jeżeli A=[1 2 3; 4 5 6] to przez podstawienie

```
D=[A, [1;2]; 1:4]
```

uzyskamy:

$$D = \begin{bmatrix} 1 & 2 & 3 & 1 \\ 4 & 5 & 6 & 2 \\ 1 & 2 & 3 & 4 \end{bmatrix}$$

Dostęp do elementów macierzy uzyskuje się przez podanie współrzędnych podmacierzy w nawiasach okrągłych za nazwą macierzy:

A(2,4) - czwarty element w drugim wierszu,
A(1:2,1:2) - podmacierz czterech lewych górnych elementów,
A(2,1:6) - drugi wiersz elementy od 1 do 6,
A(2,:) - wszystkie elementy drugiego wiersza,
A([1 3], 1:2:5) - elementy na przecięciu 1 i 3 wiersza z 1, 3 i 5 kolumną.

Można też wybierać **elementy spełniające pewne warunki**:
A(A>3) - wybiera wszystkie elementy macierzy większe od 3,
A(:,A(3,:)>2) - wybiera te kolumny, których trzeci wiersz ma element większy niż 2.

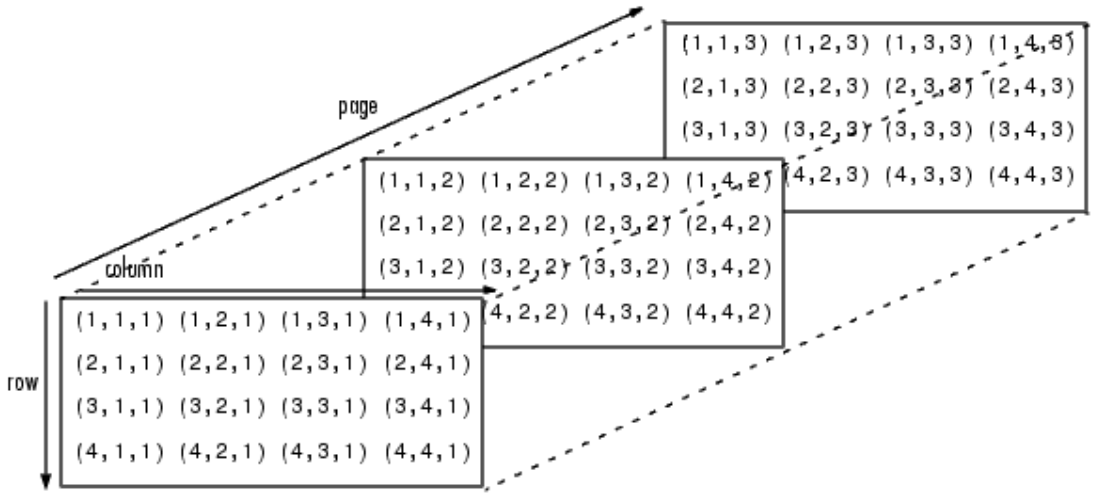
Zawartość macierzy A można wysłać do okna poleceń funkcją disp(A).

Aktualny **rozmiar macierzy** można sprawdzić poleceniem size

[n,m.]=size(A) - zwraca wierszy n i kolumn m macierzy A,
n=size(A,1) - zwraca liczbę wierszy,
m=size(A,2) - zwraca liczbę kolumn.
k=size(A,3) – zwraca liczbę stron.

Tablice wielowymiarowe

Tablice wielowymiarowe w MATLAB-ie są rozszerzeniem tablic dwuwymiarowych.
Przykład macierzy trójwymiarowej przedstawia rysunek 1



Rys. 1 Graficzna interpretacja macierzy trójwymiarowej

Tablice wielowymiarowe można tworzyć następująco:

1. **Poprzez indeksowanie**

   A = [5 7 8; 0 1 9; 4 3 6];
   A(:,:,2) = [1 0 4; 3 5 6; 9 8 7];

   Wynikiem powyższych operacji będzie macierz:
   A(:,:,1) =
   5 7 8
   0 1 9
   4 3 6

   A(:,:,2) =
   1 0 4
   3 5 6
   9 8 7

2. **Poprzez zastosowanie do tworzenia tablicy funkcji ones, zeros, randn, repmat:**

   A=ones(3,2,3);

   A(:,:,1) =

   1 1
   1 1
   1 1

   A(:,:,2) =

   1 1
   1 1
   1 1

```
A(:,:,3) =

     1     1
     1     1
     1     1

B=randn(2,3,2);

B(:,:,1) =

   -0.4326    0.1253   -1.1465
   -1.6656    0.2877    1.1909

B(:,:,2) =

    1.1892    0.3273   -0.1867
   -0.0376    0.1746    0.7258

B = repmat(5,[3 4 2]);
B(:,:,1) =
     5     5     5     5
     5     5     5     5
     5     5     5     5
B(:,:,2) =
     5     5     5     5
     5     5     5     5
     5     5     5     5
```

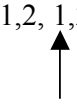
3. Za pomocą konkatenacji (scalania) tablic:

B = cat(dim,A1,A2...), gdzie *dim* jest rozmiarem macierzy wynikowej.

```
B = cat(3,[2 8; 0 5],[1 3; 7 9])
B(:,:,1) =
     2     8
     0     5
B(:,:,2) =
     1     3
     7     9
```

Funkcja *cat* dodaje automatycznie indeks równy 1, jeśli zajdzie taka potrzeba, np.:

```
C = cat(4,[1 2; 4 5],[7 8; 3 2]);
Wynikiem będzie macierz C o rozmiarach:
C(1,2, 1,2)
```

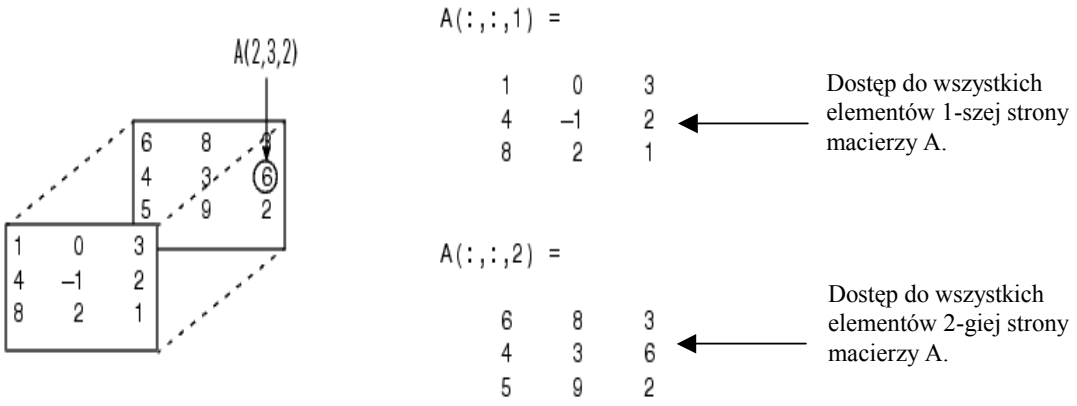


Indeks równy 1

Liczbę indeksów tablicy wielowymiarowej pobieramy za pomocą funkcji **ndims**.

Działania na tablicach wielowymiarowych

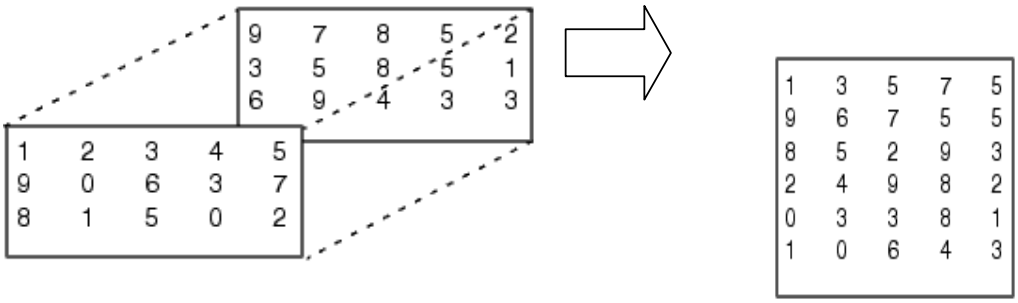
1. Podstawowe sposoby znajdywania elementów macierzy A przedstawia rys.2



Rys. 2 Znajdywanie elementów macierzy A

Zmiana kształtu tablicy za pomocą funkcji reshape:

reshape(x,m,n) – zwraca macierz o nowym wymiarze *m x n*, której elementy są elementami macierzy x



Rys. 3. Działanie funkcji **reshape**

Przykład:

Niech B = cat(3,[2 8; 0 5],[1 3; 7 9]),

Wtedy:
prod(size(B))

ans =

8

```
>> C=reshape(B,[2 4])

C =
    2     8     1     3
    0     5     7     9
```

Dostęp do podtablicy:

Stosując dwukropek : można wycinać fragmenty tablicy lub tablic, np.:
Niech

```
A=[1 9 2
     4 7 3];
B=[5 9 6
     0 8 10];
C=cat(3,A,B)
```

Zatem:

```
C(:,:,1) =
    1     9     2
    4     7     3

C(:,:,2) =
    5     9     6
    0     8    10
```

Liczba elementów w macierzy C wynosi:

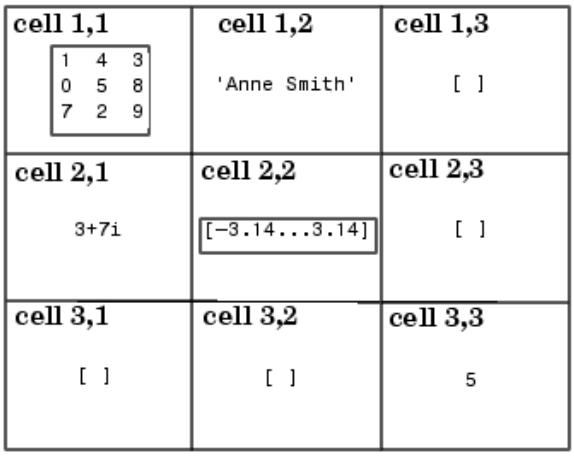
```
prod(size(C))

ans =
   12
```

Wybieramy 3 wiersze, 2 kolumny i 2 strony

```
reshape(C,[3 2 2])
ans(:,:,1) =
    1     7
    4     2
    9     3

ans(:,:,2) =
    5     8
    0     6
    9    10
```

Tablice komórkowe

Tablice komórkowe są specjalną klasa tablic w MATLAB-ie. Elementami tablic komórkowych są komórki (cells), które mogą z kolei zawierać inne tablice.



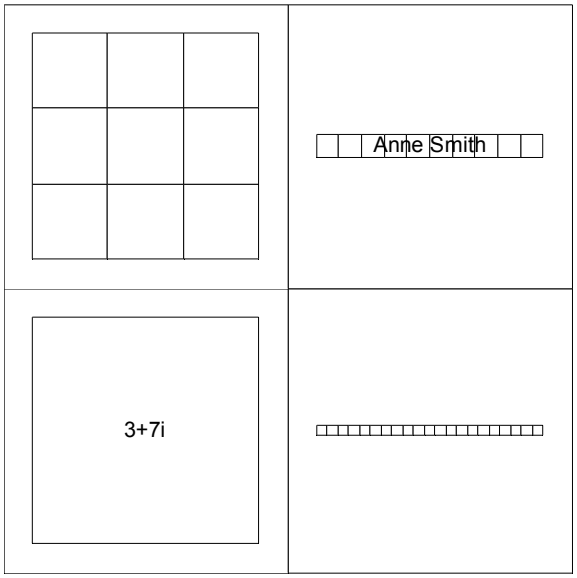
Rys. 4. Tablica komórkowa

Konstruktorem tablicy komórkowej są nawiasy **{ }**. Działają one podobnie jaka nawiasy **[ ]**. Nawiasy **{ }** mogą być zagnieżdżane (mogą tworzyć komórkę w komórce. Jako separator elementów w nawiasach **{ }** jest stosowany przecinek (**,**) lub spacja, która oznacza zmianę kolumny, średnik zaś (**;**) zmianę wiersza.

Tworzenie tablicy komórkowej:
1. Przez indeksowanie

```
A(1,1) = {[1 4 3; 0 5 8; 7 2 9]};
A(1,2) = {'Anne Smith'};
A(2,1) = {3+7i};
A(2,2) = {-pi:pi/10:pi};

celldisp(A)
A{1,1} =
    1    4    3
    0    5    8
    7    2    9

A{2,1} =
  3.0000 + 7.0000i

A{1,2} =
Anne Smith

A{2,2} =
  Columns 1 through 8
   -3.1416   -2.8274   -2.5133   -2.1991   -1.8850   -1.5708   -1.2566   -0.9425
  Columns 9 through 16
   -0.6283   -0.3142         0    0.3142    0.6283    0.9425    1.2566    1.5708
  Columns 17 through 21
    1.8850    2.1991    2.5133    2.8274    3.1416
```

Tablicę komórkową można także przedstawić graficznie za pomocą polecenia **cellplot(A).**



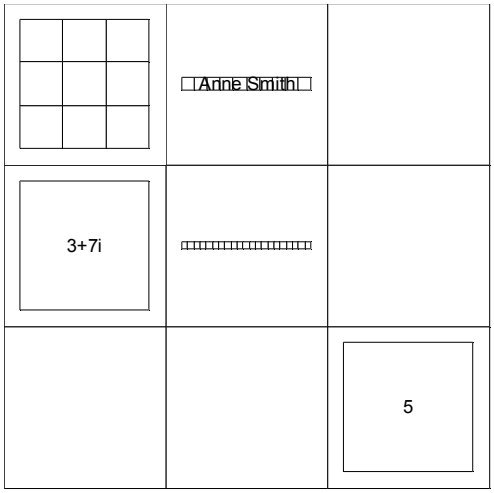
Rys.5. Graficzne prezentacja tablicy komórkowej

2. **Przez przypisanie danych do komórki, których indeks przekracza rozmiar tablicy komórkowej**

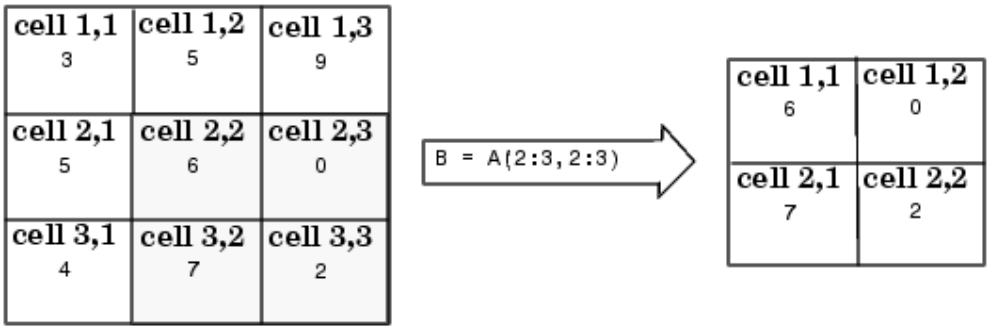
W takim przypadku zachodzi rozszerzenie bieżącej tablicy komórkowej do wymaganego wymiaru.

Przykład: do poprzednio wygenerowanej macierzy A dodajemy:
A(3,3) = {5};

W wyniku otrzymamy tablicę komórkową postaci:



Tworzenie nowej macierzy B poprzez przypisanie jej elementów podmacierzy A, np.:



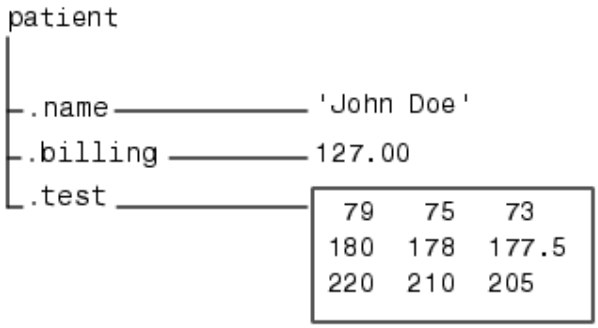
Kasowanie komórek

Odbywa się poprzez przypisanie komórce tablicy pustej, np.:

A{3,3} = [];

Tablice strukturalne

tablice strukturalne są odmianą tablic MATLAB-a, w których dostęp do danych jest możliwy przez podanie nazwy pola. Pola tablicy strukturalnej mogą zawierać różne typy danych, np.:



Sposoby tworzenia tablic strukturalnych

1. Używając instrukcji przypisania, np.:

```
patient.name = 'John Doe';
patient.billing = 127.00;
patient.test = [79 75 73; 180 178 177.5; 220 210 205];
```

Program ćwiczenia

ŚRODOWISKO

1. Sprawdzić istniejące ścieżki dostępu poleceniem path.
2. Zapoznać się z możliwościami wprowadzania zmian ścieżek dostępu:
   - w trakcie bieżącej sesji (addpath, rmpath),
   - na stałe w pliku *pathdef.m*.
3. Wykonać wybrane polecenie systemu operacyjnego z poziomu MATLAB'a wykorzystując operatory !, &, |.
4. Utworzyć kilka dowolnych zmiennych w przestrzeni roboczej, a następnie stworzyć trzy różne pliki przestrzeni roboczych poleceniem save.
5. Sprawdzić możliwość odtwarzania w środowisku dowolnej przestrzeni roboczej poleceniem load.
6. Zapoznać się z możliwymi sposobami formatu wyświetlania liczb korzystając z polecenia format.
7. Zapoznać się z systemem pomocy oraz wyszukiwania (polecenia help, which).

MACIERZE

8. Utworzyć w możliwie rozbudowany sposób macierz liczb rzeczywistych postaci:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

9. Wykorzystując macierz *A* utworzyć macierz liczb zespolonych postaci:

$$B = \begin{bmatrix} 1+2i & 2+3i & 3+4i \\ 4+5i & 5+6i & 6+7i \\ 7+8i & 8+9i & 9+10i \end{bmatrix}$$

10. Wykonać podstawowe działania macierzowe na macierzach *A* i *B* tj.: *A+B, A-B, A*B, A/B, A\B, A', B', A^2, B^2.*
11. Wykonać podstawowe działania tablicowe na macierzach *A* i *B* tj.: *A.*B, A./B, A.\B., A.^2, B.^2*.

12. Na podstawie macierzy *A* i *B* stworzyć macierze *C* i *D* postaci:

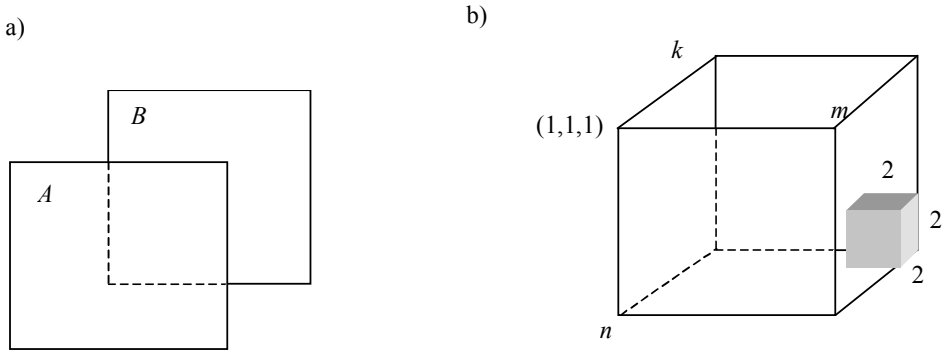
$$C = \begin{bmatrix} -1 & 2 & 3 \\ -4 & 5 & 6 \\ -7 & 8 & 9 \end{bmatrix}; \qquad D = \begin{bmatrix} 1+2i & 2+3i & 3+4i \\ 4+5i & 5+6i & 6+7i \\ -7-8i & -8-9i & -9-10i \end{bmatrix}$$

13. Obliczyć części rzeczywiste i urojone oraz moduły i argumenty elementów macierzy *C* i *D*, korzystając odpowiednio z funkcji real, imag, abs, angle. Porównać i skomentować uzyskane wyniki.

14. Porównać sposoby obliczania macierzy sprzężonej i transponowanej macierzy *D* wykorzystując operatory ‘ i .’.

TABLICE WIELOWYMIAROWE, KOMÓRKOWE I STRUKTURALNE

15. Wykorzystując polecenie cat zbudować z macierzy *A* i *B* trójwymiarową tablicę *AB,* tak jak to pokazano na rys.6a.



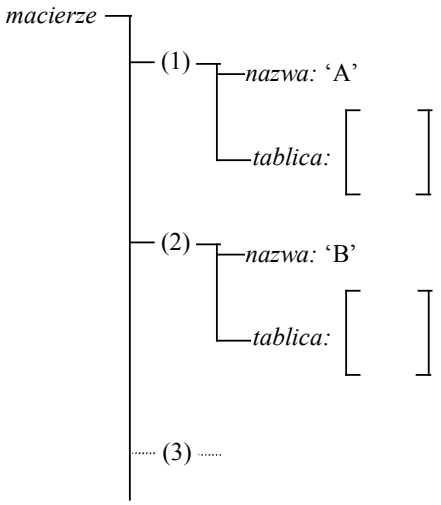
Rys.6.

16. Napisać wyrażenia, w formie niezależnej od faktycznych wymiarów *m,n,k* tablicy trójwymiarowej (patrz rys.6b), realizujące następujące czynności:
    - obliczanie różnicy odpowiadających sobie elementów górnej i dolnej „powierzchni” tablicy trójwymiarowej;
    - utworzenie macierzy o wymiarach *n*x4 składającej się ze wszystkich pionowych „krawędzi” tablicy trójwymiarowej;
    - podwojenie wartości elementów tablicy zwartych w „obszarze” 2x2x2 wskazanym na rys.6b.

    UWAGA: każdy podpunkt zadania należy zrealizować w postaci jednego polecenia, którego poprawność należy sprawdzić w odniesieniu do utworzonej wcześniej tablicy *AB*.

17. Utworzyć tablicę komórkową *c_AB* o wymiarach 3x2, w której elementami kolumn będą: w pierwszej nazwy tablic (*A, B, AB*), a w drugiej – odpowiadające nazwom tablice.

18. Wykorzystując elementy tablicy komórkowej *c_AB* obliczyć sumę macierzy *A* i *B,* umieszczając wynik w tablicy *c_AB* jako nowy element (np. w czwartym wierszu pierwszej kolumny).
19. Utworzyć strukturę o nazwie *macierze*, w której zawarte zostaną macierze *A* i *B* zgodnie ze schematem podanym na rys.7.



Rys.7

20. Wykorzystując pola struktury *macierze* obliczyć sumę macierzy *A* i *B* umieszczając wynik w strukturze *macierze* jako nowy element.
21. Obliczyć sumę wybranego elementu struktury *macierze* i wybranej komórki tablicy *c_AB*.

Opracowanie sprawozdania

Zapisać fragmenty sesji pracy z programem obejmujące ostateczne rozwiązanie poszczególnych punktów instrukcji.