AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE

Obliczenia w Matlabie

Operatory, instrukcje sterujące, operacje bitowe

Łukasz Sztangret
Katedra Informatyki Stosowanej i Modelowania

Operatory arytmetyczne + - * / \ ^ ‘

```
>> a=2,b=4
a =
     2
b =
     4
>> a+b
ans =
     6
>> a-b
ans =
    -2
>> a*b
ans =
     8
>> a/b
ans =
    0.5000
>> b/a
ans =
     2
>> a\b
ans =
     2
>> b\a
ans =
    0.5000
>> a^b
ans =
    16
>> a'
a =
     2
>> b'
b =
     4
```

Operatory arytmetyczne + - * / \ ^ ‘

```
>> a=5,b=[1 2 3]
a =
     5
b =
     1     2     3
>> a+b
ans =
     6     7     8
>> a-b
ans =
     4     3     2
>> a*b
ans =
     5    10    15
```

```
 >> b/a
ans =
    0.2000    0.4000    0.6000
>> a\b
ans =
    0.2000    0.4000    0.6000
>> a/b
Error using  /
Matrix dimensions must agree.

>> b\a
ans =
         0
         0
    1.6667
```

c=a/b => c*b=a

c=b\a => b*c=a

Operatory arytmetyczne + - * / \ ^ ‘

```
>> a=5,b=[1 2 3]
a =
    5
b =
    1    2    3
>> a^b
Error using  ^
Inputs must be a scalar and a square matrix.
To compute elementwise POWER, use POWER (.^) instead.

>> b^a
Error using  ^
Inputs must be a scalar and a square matrix.
To compute elementwise POWER, use POWER (.^) instead.

>> b*b
Error using  *
Inner matrix dimensions must agree.
```

```
>> b'
ans =
    1
    2
    3
```

Operatory arytmetyczne + - * / \ ^ ‘

```
>> a=5,b=[1 2 3]
a =
    5
b =
    1     2     3
>> a.^b
ans =
    5    25   125
>> b.^a
ans =
    1    32   243
>> b.*b
ans =
    1     4     9
```

```
>> b./a
ans =
    0.2000    0.4000    0.6000
>> a.\b
ans =
    0.2000    0.4000    0.6000
>> a./b
ans =
    5.0000    2.5000    1.6667
>> b.\a
ans =
    5.0000    2.5000    1.6667
```

.* ./ .\ .^ to działania tablicowe!

Operatory arytmetyczne + - * / \ ^ ‘

```
>> a=[1 2 3],b=[1;2;3]
a =
    1    2    3
b =
    1
    2
    3
>> a*b
ans =
   14
>> b*a
ans =
    1    2    3
    2    4    6
    3    6    9
```

```
>> a.*b
Error using  .*
Matrix dimensions must agree.
>> a.^b
Error using  .^
Matrix dimensions must agree.
>> b.^a
Error using  .^
Matrix dimensions must agree.
 >> a^b
Error using  ^
Inputs must be a scalar and a square matrix.
To compute elementwise POWER, use POWER (.^) instead.
 >> b^a
Error using  ^
Inputs must be a scalar and a square matrix.
To compute elementwise POWER, use POWER (.^) instead.
```

Operatory arytmetyczne + - * / \ ^ '

```
>> A=[1 2; 3 4],B=[5 6; 7 8]
A =
     1     2
     3     4
B =
     5     6
     7     8
>> A+B
ans =
     6     8
    10    12
>> A*B
ans =
    19    22
    43    50
>> B*A
ans =
    23    34
    31    46
```

```
>> C=A/B
C =
    3.0000   -2.0000
    2.0000   -1.0000
>> C=B\A
C =
    5.0000    4.0000
   -4.0000   -3.0000
>> C=B/A
C =
   -1.0000    2.0000
   -2.0000    3.0000
>> C=A\B
C =
   -3.0000   -4.0000
    4.0000    5.0000
```

```
>> A_=C*B
A_ =
     1     2
     3     4
>> A_=B*C
A_ =
    1.0000    2.0000
    3.0000    4.0000
>> B_=C*A
B_ =
    5.0000    6.0000
    7.0000    8.0000
>> B_=A*C
B_ =
     5     6
     7     8
```

Liczby zespolone

```
>> a=1+2i,b=-3-4i
a =
   1.0000 + 2.0000i
b =
  -3.0000 - 4.0000i
>> a+b
ans =
  -2.0000 - 2.0000i
>> a-b
ans =
   4.0000 + 6.0000i
>> a*b
ans =
   5.0000 -10.0000i
>> a/b
ans =
  -0.4400 - 0.0800i
```

```
>> a'
ans =
   1.0000 - 2.0000i
>> a.'
ans =
   1.0000 + 2.0000i
>> a=[1 2 3]+i*[4 5 6]
a =
   1.0000 + 4.0000i   2.0000 + 5.0000i   3.0000 + 6.0000i
>> a'
ans =
   1.0000 - 4.0000i
   2.0000 - 5.0000i
   3.0000 - 6.0000i
>> a.'
ans =
   1.0000 + 4.0000i
   2.0000 + 5.0000i
   3.0000 + 6.0000i
```

‘ sprzężenie macierzy

.’ transpozycja macierzy

Operatory relacji > < >= <= == ~=

```
>> a=[1 2 3 4 5]
a =
    1    2    3    4    5
>> a>3
ans =
    0    0    0    1    1
>> a~=2
ans =
    1    0    1    1    1
>> a==2
ans =
    0    1    0    0    0
```

```
>> a=[1 2 3], b=[1 0 4]
a =
    1    2    3
b =
    1    0    4
>> a>b
ans =
    0    1    0
>> a<b
ans =
    0    0    1
```

Operatory logiczne & | ~ && ||

```
>> a=[1 1 0 0], b=[1 0 1 0]
a =
    1    1    0    0
b =
    1    0    1    0
>> a&b
ans =
    1    0    0    0
>> a|b
ans =
    1    1    1    0
>> ~a
ans =
    0    0    1    1
>> xor(a,b)
ans =
    0    1    1    0
```

```
>> a&&b
Operands to the || and && operators must be
    convertible to logical scalar values.

>> a=1,b=0
a =
    1
b =
    0
>> a&&b
ans =
    0
>> a||b
ans =
    1
```

Operator :

```
>> a=1:1:5
a =
    1    2    3    4    5
>> b=5:-1:1
b =
    5    4    3    2    1
>> c=0:0.2:1
c =
         0    0.2000    0.4000    0.6000    0.8000    1.0000
>> d=1:5
d =
    1    2    3    4    5
>> e=5:1
e =
   Empty matrix: 1-by-0
```

Priorytety operatorów

1	()
2	.' .^ ' ^
3	+ - ~
4	.* ./ .\ * / \
5	+ -
6	:
7	< <= > >= == ~=
8	&
9	|
10	&&
11	||

Instrukcje sterujące - if

```
n=input('Podaj liczbe ');
if n>0
    disp('Liczba dodatnia');
elseif n<0
    disp('Liczba ujemna');
else
    disp('Zero');
end
```

Warunek nie musi być w nawiasie

Instrukcje nie są pomiędzy nawiasami **{}**, nawet jeśli jest ich kilka

elseif pisane jest razem

Kończy się słowem **end**

Instrukcje sterujące - switch

```
n=input('Podaj liczbe ');
switch n
    case 1
        disp('Jeden');
    case 2
        disp('Dwa');
    otherwise
        disp('Inna');
end
```

Wartość nie musi być w nawiasie

Poszczególne **case'y** nie są pomiędzy nawiasami **{}**

Po wartości nie ma znaku **:**

Nie używamy instrukcji **break**. Wykona się tylko jeden **case**

Kończy się słowem **end**

Zamiast **default** jest **otherwise**

Instrukcje sterujące - switch

```
n=input('Podaj liczbe ');
switch n
    case {1 2 3}
        disp('Jeden lub dwa lub trzy');
    case 4
        disp('Cztery');
    otherwise
        disp('Inna');
end
```

Możemy porównać do jednej z kilku wartości

Instrukcje sterujące - switch

```
n=0.1;
switch n
    case 0.1
        disp('Jedena dziesiata');
    case 0.2
        disp('Dwie dziesiate');
    otherwise
        disp('Inna');
end
```

Możemy porównać liczby rzeczywiste

Instrukcje sterujące - switch

```
n=0.1;
m=0.1;
k=0.2;
switch n
    case m
        disp('m');
    case k
        disp('k');
    otherwise
        disp('Inna');
end
```

Możemy porównać ze zmiennymi

Pętle - for

```
for i=1:5
    disp(i);
end

x=5:-1:1;
for i=x
    disp(i);
end

for i=1:Inf
    disp(i);
end
```

Licznik pętli przyjmuje kolejne wartości z wektora

Pętla nieskończona

Pętle - while

```
j=0;
while j<5
    disp(j);
    j=j+1;
end
```

Nie ma **j++** ani **j+=1**

```
while 1
    disp(j);
    j=j+1;
end
```

Pętla nieskończona

Instrukcje sterujące - break

```
for i=1:5
    disp(i);
    if i==3
        break;
    end
end

j=0;
while j<5
    disp(j);
    j=j+1;
    if j==3
        break;
    end
end
```

Przerwanie pętli

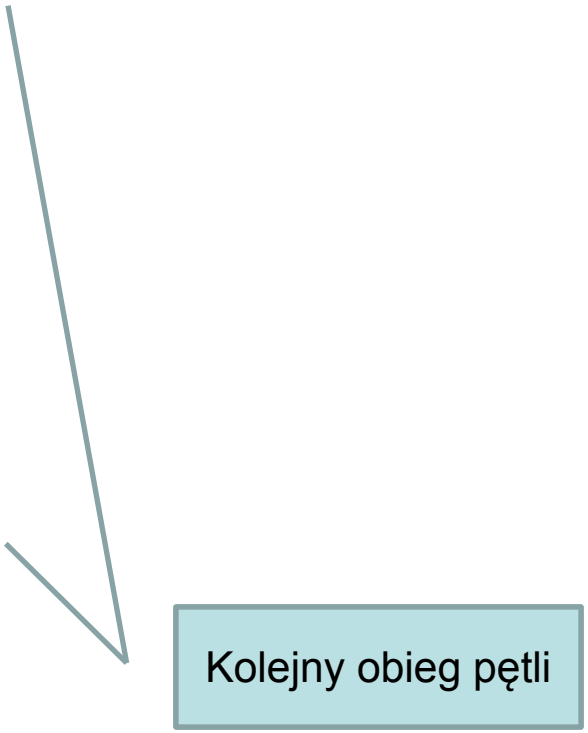
```
disp('Ta linia sie wykona');
break;
disp('Ta juz nie');
```

Przerwanie skryptu

Instrukcje sterujące - continue

```
for i=1:5
    if i==3
        continue;
    end
    disp(i);
end

j=0;
while j<5
    j=j+1;
    if j==3
        continue;
    end
    disp(j);
end
```



Instrukcje sterujące

- **goto** – w Matlabie nie występuje
- **exit**, **quit** – kończy działanie Matlaba
- **error** – przerywa działanie programu i wyświetla informację o błędzie, np.
  error('Z powodu błędu przerywam obliczenia');
- **warning** – wyświetla ostrzeżenie np.
  warning('Wyniki obliczeń mogą być obarczone błędem');

Skrypty:

- **finish.m** – skrypt wykonywany przed zamknięciem Matlaba (musi zostać zapisany w aktualnym katalogu lub na ścieżce)
- **startup.m** – skrypt wykonywany po uruchomieniu Matlaba (musi zostać zapisany w katalogu uruchomieniowym)

Typy numeryczne

Rzeczywiste:

- double – 8B
- single – 4B

Całkowite ze znakiem

- int8, int16, int32, int64

Całkowite bez znaku

- uint8, uint16, uint32, uint64

```
>> a=1
a =
     1
>> class(a)
ans =
double
```

Funkcje wykonujące operacje bitowe

```
>> a=uint8(1), b=uint8(3)
a =
   1
b =
   3
>> bitand(a,b)
ans =
   1
>> bitor(a,b)
ans =
   3
>> bitxor(a,b)
ans =
   2
```

Iloczyn bitowy

Suma bitowa

Bitowa różnica symetryczna

```
>> bitget(a,1)
ans =
   1
>> bitset(a,8)
ans =
  129
>> bitshift(a,2)
ans =
   4
>> bitshift(a,-1)
ans =
   0
>> bitcmp(a)
ans =
  254
>> bitcmp(a,3)
ans =
   6
```

Wartość bitu nr 1

Ustawienie bitu nr 8

Przesunięcie bitowe o 2 pozycje w lewo

Przesunięcie bitowe o -1 pozycje w lewo (o 1 w prawo)

Negacja wszystkich bitów

Traktuje a jako liczbę 3-bitową i ją neguje. W a nie może być wartości 1 na bitach starszych niż 3

Funkcje wykonujące operacje bitowe

```
>> a=1, b=3
a =
     1
b =
     3
>> bitand(a,b)
ans =
     1
>> bitor(a,b)
ans =
     3
>> bitxor(a,b)
ans =
     2
>> bitcmp(a,1)
ans =
     0
```

```
>> bitget(a,1)
ans =
     1
>> bitget(a,2)
ans =
     0
>> bitset(a,2)
ans =
     3
>> bitset(a,3)
ans =
     5
>> bitshift(a,10)
ans =
        1024
```

Operacje bitowe możemy wykonywać tylko gdy wartość zmiennej jest całkowita

Operacje bitowe możemy wykonywać tylko na mantysie (zgodnie ze standardem IEEE 754 mantysa ma 52 bity).

Funkcje wykonujące operacje bitowe

```
>> a=uint8(1), b=uint16(1),
   c=uint32(1), d=uint64(1)
a =
   1
b =
    1
c =
        1
d =
                1
```

```
>> swapbytes(a)
ans =
   1
>> swapbytes(b)
ans =
   256
>> swapbytes(c)
ans =
   16777216
>> swapbytes(d)
ans =
   72057594037927936
```

2^8

2^24

2^56