Instrukcje sterujące

Operacje relacyjne i logiczne

- Operacje na macierzach o tych samych wymiarach lub pomiędzy macierzą a skalarem – wynikiem jest macierz tych samych wymiarów, której elementami są wartości logiczne (0 lub 1) wyniku operacji na odpowiednich elementach macierzy

- Operatory relacyjne:
  - == równości
  - ~= nierówności
  - < mniejszości
  - <= niewiększości
  - > większości
  - >= niemniejszości

- Operatory logiczne:
  - & logiczne And
  - | logiczne Or
  - ~ logiczne Not

- Elementy niezerowe macierzy traktowane są jak 1 (prawda)

Funkcje logiczne

- Przykładowe funkcje logiczne:
  - ***all(A)*** – sprawdza czy wszystkie elementy wektora A są różne od 0 i zwraca 1 (prawda) lub 0 (fałsz)
  - ***any(A)*** – sprawdza czy którykolwiek z elementów wektora A jest różne od 0 i zwraca 1 (prawda) lub 0 (fałsz)
  - ***isequal(A, B, …)*** – porównuje macierze (rozmiar i zawartość)
  - ***isempty(A)*** – zwraca 1, jeżeli macierz A jest pusta, w przeciwnym wypadku zwraca 0
  - ***ischar, isreal, isnan***

Instrukcja warunkowa „if”

- Składnia:

```
if  wyrażenie warunkowe1
   instrukcje
elseif  wyrażenie warunkowe2
   instrukcje
else
   instrukcji
end
```

```
if  wyrażenie warunkowe
   instrukcje
end
```

- Wyrażenie ***elseif*** oznacza to samo co ***else if*** ale należy pisać je łącznie
- Bloki ***elseif*** oraz ***else*** są opcjonalne

Instrukcja warunkowa „if” – przykład

- Oblicz wartość funkcji f(x) określonej wzorem:

$$f(x) = \begin{cases} x^2 - 6 & dla \quad x > 3 \\ x & dla \quad -1 \leq x \leq 3 \\ x^2 - 2 & dla \quad x < -1 \end{cases}$$

```
if x>3
   f=x^2-6
elseif  x<-1
   f=x^2-2
else
   f=x
end
```

Instrukcja wyboru „switch”

- Składnia:

```
switch wyrażenie
   case wartość1
       blok instrukcji
   case (wartość2,wartość3)    % wspólny blok poleceń dla kilku wartości
       blok instrukcji
   …
   otherwise                   % w pozostałych przypadkach
       blok instrukcji
    end
```

- Wyrażenie może być liczbą lub łańcuchem znakowym

Instrukcja wyboru „switch” – przykład

- Utwórz macierz A o wymiarach nxn. W zależności od wartości c ma to być:
  - gdy c=0 – macierz o wszystkich elementach równych 0
  - gdy c=1 – macierz o wszystkich elementach równych 1
  - gdy c=2 – macierz jednostkowa
  - dla innych wartości c – macierz wypełniona liczbami pseudolosowymi

```
switch c
    case 0
        A=zeros(n)
    case 1
        A=ones(n)
    case 2
        A=eye(n)
    otherwise
        A=rand(n)
end
```

Pętla „for”

- Składnia:

```
for licznik = macierz_wartości
    blok instrukcji
end
```

- ***wektor wartości*** najczęściej podaje się w postaci skróconej: ***początek:krok:koniec***
- Pętla kończy się, gdy ***krok*** minie koniec
- Macierz wartości może mieć postać wyliczeniową: [3 4 8 9]

Pętla „for” – przykład

- Skrypt sprawdza ile liczb spośród trzech wprowadzonych należy do przedziału <5 10>

```
ilosc=0;
for i=1:3
    x=input('Podaj liczbę: ');
    if (x>=5) & (x<=10)
        ilosc=ilosc+1;
    end
end
```

Pętla „for” – przykład

- Funkcja oblicza sumę n liczb naturalnych

```
function suma=suman(n)
%Oblicza sumę liczb od 1 do n
if(n<=0)
    error('Ilosc liczb musi byc wieksza od 0')
else
    suma=0;
    for i=1:n
        suma=suma+i;
    end
end
end
```

Pętla „for” – przykład

- Utwórz macierz $A \in \mathcal{R}^{5\times4}$, $A_{ij} = \mathrm{i} + \mathrm{j}$.

```
for i=1:5
    for j=1:4
            A(i,j)=(i+j);
    end
end
```

```
A =

     2     3     4     5
     3     4     5     6
     4     5     6     7
     5     6     7     8
     6     7     8     9

>> i

i =

     5

>> j

j =

     4
```

Pętla „while”

- Składnia:

```
while wyrażenie
    blok instrukcji
end
```

- ***wyrażenie*** może być wyrażeniem relacyjnym lub kombinacją logiczną wyrażeń relacyjnych
- Pętla działa dopóty, dopóki wyrażenie ma wartość logiczną prawdy

Pętla „while” – przykład

```
a=15; b=0.1;
while a-b>1
    a=a^(1/2)
end
```

```
a =

    3.8730

a =

    1.9680

a =

    1.4029

a =

    1.1844

a =

    1.0883
```

Instrukcje przerywające

- Instrukcja ***continue*** przerywa wykonywanie sekwencji instrukcji w pętli i przenosi sterowanie do pierwszej instrukcji w pętli
- Instrukcja ***break*** przerywa wykonywanie sekwencji instrukcji w pętli i przenosi sterowanie do pierwszej instrukcji po pętli (po słowie end kończącym daną pętlę)
- Instrukcja ***return*** przerywa wykonywanie funkcji i powrót do miejsca, z którego została wywołana