Piotr Chojnacki:

MATLAB
zajęcia 3:

{Poniżej znajduje się zrzut ekranowy pola gdzie wpisuje się komendy. Nie wszystkie komendy są opisane. Wynik działania każdej z nich widać linijkę niżej.}

Znajdziesz tutaj wstęp do programowania, zasady pisania funkcji i skryptów funkcyjnych – przykłady.

---

zajecia5.mat

Tworzenie skryptu:

Otwieramy kolejno: File -> New -> M-file. Zostaje otworzone okienko, w którym piszemy kolejne linijki kodu. Zapisujemy plik z rozszerzeniem .m. Następnie, aby wywołać skrypt należy w oknie głównym (‘Command window’) wpisać nazwę skryptu np. **f1** i w zależności od kodu z parametrami lub bez np.**f1(2,3)**.

Tworzenie funkcji:

W języku MATLAB istnieje możliwość definiowania własnych funkcji, jako elementów strukturalnych programu. Definicja funkcji ma następującą postać:

function[wartość_funkcji]=nazwa_funkcji(argument1,..,argumentN) ciąg instrukcji

Komentarze:

W skryptach skryptów funkcjach można wstawiać komentarze. Należy je rozpoczynać przez znak %.

Przykłady skryptów i funkcji:

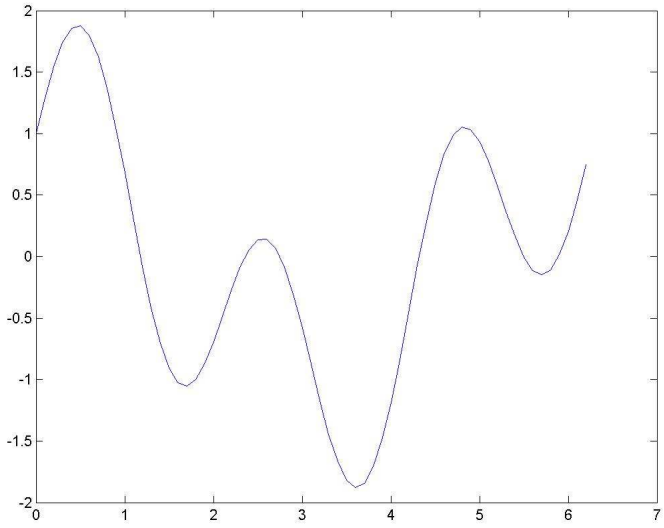
f1.m

{rysowanie wykresu funkcji y=cos(x)+sin(3*x) z parametrami x równymi x=[0:0.1:2*pi]; }

treść skryptu:

```
x=[0:0.1:2*pi];        % definicja x
y=cos(x)+sin(3*x);     % definicja y
plot(x,y)              % rysowanie wykresu
```

Wynikiem działania funkcji jest wykres:



f2.m

{przykład skryptu funkcyjnego z parametrem wejściowym zdefiniowanym dopiero podczas wywoływania funkcji}

treść skryptu funkcyjnego:

```
function y=f1(x)
y=cos(x) + sin(3*x);
plot(x,y);
```

{wywołanie funkcji w 'command window':}

```
>> f2(5)

ans =

    0.9340

>> f2(6)

ans =

    0.2092
```

{Za każdym razem wynikiem działania skryptu funkcyjnego jest punkt na wykresie przy odpowiednim parametrze wejściowym x}

---

f4.m

{przykład skryptu funkcyjnego z parametrem wejściowym zdefiniowanym globalnie w 'command window'}

{wcześniej zdefiniować x globalne}

```
function y=f1(x)
global x
y=cos(x) + sin(3*x);
```

przykład w command window:

```
>> x=[2,4,5,9]

x =

     2     4     5     9

>> global x
>> f1

ans =
       -0.6956   -1.1902    0.9340    0.0452
```

{wyniki funkcji y=cos(x) + sin(3*x) dla poszczególnych, w tym przypadku 4 x}

---

f5.m

{przykład funkcji }

```
function [w,p] = f5(z)
global x
w=z(3)*x.*x+z(2)*x*z(4);
p=2*z(3)*x+z(2);
```

{wcześniej zdefiniowane przykładowe ‘z’ i ‘x’ w programie głównym,}

```
>> x=[4,8,3]

x =

    4     8     3

>> z=[2,5,3,7,1]

z =

    2     5     3     7     1


>> [wx,py]=f5(z)

wx =

    0   250   318


py =

    5    35    41

>> [wx,py]=f5(z)
```

{wywołanie funkcji}

```
wx =

  188   472   132


py =

   29    53    23
```

---

f6.m

{funkcja licząca sumę od N do M}

{przykład funkcji }

```
function suma=f6(N,M)
suma=0;
for i=N:M      {pętla for}
      suma=suma+i;
  end
disp(['suma  =  ',int2str(suma)])      {wyświetlenie sumy i podanie wartości na zewnątrz podstawiając pod ans}

>> f6(2,6) {wywołanie funkcji}
suma = 20

ans =

   20
```

---

f7.m

{Zadaniem funkcji jest wyzerowanie dolnej lub górnej części macierzy w zależności od parametru wejściowego 0 lub 1.}

```
function f=f7(x,y)
n=size(x,1);
m=size(x,2);
if y==0;
  for i=2:n
     for j=1:i-1
       x(i,j)=0;
     end
  end
f=x;
else
  for i=2:n
     for j=1:i-1
       x(j,i)=0;
     end
  end
f=x
end


>> x=magic(6)    {Utworzenie macierzy o wymiarach 6 x 6}

x =

   35    1    6   26   19   24
    3   32    7   21   23   25
   31    9    2   22   27   20
    8   28   33   17   10   15
```

```
   30    5   34   12   14   16
    4   36   29   13   18   11
```

>> f1(x,0) {Wywołanie funkcji z parametrem 0 czyli zerowanie dolnej połówki}

ans =

```
   35    1    6   26   19   24
    0   32    7   21   23   25
    0    0    2   22   27   20
    0    0    0   17   10   15
    0    0    0    0   14   16
    0    0    0    0    0   11
```

>> f1(x,1) {Wywołanie funkcji z parametrem 1 czyli zerowanie górnej połówki.}

ans =

```
   35    0    0    0    0    0
    3   32    0    0    0    0
   31    9    2    0    0    0
    8   28   33   17    0    0
   30    5   34   12   14    0
    4   36   29   13   18   11
```

---

f8.m

{Zadaniem funkcji jest ponowne wyzerowanie dolnej lub górnej części macierzy w zależności od parametru wejściowego 0 lub 1. Funkcja różni się sposobem zapisu.}

```
function f=f8(x,y)
n=size(x,1);

  for i=2:n
     for j=1:i-1
        if y==0
          x(j,i)=0;
        else   x(i,j)=0;
     end
  end
end
f=x;
```

>> f2(x,0) {Wywołanie funkcji z parametrem 0 czyli zerowanie górnej połówki.}

```
ans =

   35    0    0    0    0    0
    3   32    0    0    0    0
   31    9    2    0    0    0
    8   28   33   17    0    0
   30    5   34   12   14    0
    4   36   29   13   18   11
```

>> f2(x,1) {Wywołanie funkcji z parametrem, 1 czyli zerowanie dolnej połówki.}

```
ans =

   35    1    6   26   19   24
    0   32    7   21   23   25
    0    0    2   22   27   20
    0    0    0   17   10   15
    0    0    0    0   14   16
    0    0    0    0    0   11
```

---

ln2.m

{Zadaniem funkcji jest obliczenie ln2 z zadaną dokładnością.}

```
function ln2=ln2(e1)
%obliczanie ln2 z dokladnoscia e1

krok=2;
wartosc=1-1/krok;
zm=1;
p=1;

while abs(wartosc-zm)>e1
  %disp(['zm ' num2str(zm)])
  %disp(['wartosc ' num2str(wartosc)])
  %disp(['krok ' num2str(krok)])
  zm=wartosc;
  krok=(krok+1);
  wartosc=wartosc-(1/(krok*((-1)^p)));
  p=p+1;

end
disp(['ln2 z dokładnością ' num2str(e1) ' wynosi= ' num2str(wartosc)])
```

pi4.m

{Zadaniem funkcji jest obliczenie pi/4 z zadaną dokładnością.}

```
function pi4=pi4(e1)
%obliczanie pi/4 z dokladnoscia e1

krok=3;
wartosc=1-1/krok;
zm=1;
p=1;

while abs(wartosc-zm)>e1
  %disp(['zm ' num2str(zm)])
  %disp(['wartosc ' num2str(wartosc)])
  %disp(['krok ' num2str(krok)])
  zm=wartosc;
  krok=(krok+2);
  wartosc=wartosc-(1/(krok*((-1)^p)));
  p=p+1;

end
disp(['pi/4 z dokładnością ' num2str(e1) ' wynosi= ' num2str(wartosc)])
```

---

porzadek.m

{Zadaniem funkcji jest uporządkowanie wektora od wartości najmniejszych do największych.}

```
function [Wp,krok]=porzadek(W)

licznik=1;
n=length(W);
krok=0;
while (licznik>0)


   licznik=0;
   for i=1:n-1
      if W(i)>W(i+1)
         pom=W(i);
         W(i)=W(i+1);
         W(i+1)=pom;
         licznik=licznik+1;
         krok=krok+1
      end
```

```
   end
end
Wp=W;
```

---

porzadek2.m

{Zadaniem funkcji jest uporządkowanie wektora od wartości najmniejszych do największych. Współpracuje ona z poprzednią, ponieważ najpierw wektor W jest dzielony na dwie części, które są osobno sortowane funkcją porządek.m. Na koniec obie połówki są scalane. Dzięki temu porządkowanie trwa znacznie szybciej.}

```
function [wout,krok]=porzadek2(W)

n=length(W);
sr=mean(W);

k1=0; k2=0;

for i=1:n
   if W(i)<sr k1=k1+1;
      w1(k1)=W(i);
   else k2=k2+1;
      w2(k2)=W(i);
   end
end
[w1p,krok1]=porzadek(w1);
[w2p,krok2]=porzadek(w2);
krok=krok1+krok2;
wout=[w1p w2p];
```

---

minnmaxx.m

{Zadaniem funkcji jest znalezienie minimalnego i maksymalnego wyrazu. Podaje ona położenie jak i wartość minimum i maksimum.}

```
function [mx,pmx,mn,pmn] = minnmaxx (x, macierz)

mx = macierz(1,1);
mn = macierz(1,1);

  [li lj] = size (macierz);
```

```
for I = 1 : 1 : li
   for J = 2 : 1 :lj

      if macierz(I,J) >= mx
         mx = macierz(I,J);
         pmx = x(I,1);
      elseif macierz(I,J) <= mn
         mn = macierz(I,J);
         pmn = x(I,1);
      end;
   end;
end;
text(pmx,(mx + 0.1),'MAX');
text(pmn,(mn + 0.1),'MIN');
```