MATLAB

Prowadzący: dr hab. inż. Marek Jaszczur
Poziom: początkujący

Laboratorium 8: Programowanie - funkcje

Cel: Zdobycie umiejętności pisania prostych funkcji
Czas: Wprowadzenia 10 minut, ćwiczeń 30 minut, testu 5 minut

Wstęp
Funkcje podobnie jak skrypt są m-plikami, piszemy je tak samo edytujemy je tak samo. Różnią je przede wszystkim 2 rzeczy (i nagłówek). Po pierwsze **funkcje akceptują argumenty** podobnie jak sin(x) (w przeciwieństwie do np. bezargumentowego pi) i po drugie funkcje **operują na własnej przestrzeni roboczej**.
Plik funkcji różni się od pliku skryptu pierwszą linią. Jest nią definicja funkcji w postaci:

function [x,y]=nazwa(a,b,c)

gdzie: function słowo kluczowe (Uwaga! musi być z małej litery); x,y – argument wyjściowe (może być również funkcja bezargumentowa), funkcja- nazwa funkcji (Uwaga! nazwa funkcji musi być zgodna z nazwą pliku); **a,b,c- argumenty wejściowe**. Dla przykładu:

```
function c=pitagoras(a,b);
c=sqrt(a^2+b^2);
```

Po zapisaniu powyższej funkcji możemy zacząć z niej korzystać:

```
>> pitagoras
Error using pitagoras (line 2)
Not enough input arguments.
```

Niestety funkcja nie działa. Co jest powodem komunikatu o błędzie.
Okazuje się, że funkcja jest w porządku. Po prostu została błędnie uruchomiona (wywołana). Wyobraźmy sobie, że wywołamy funkcję sinus tj. sin() nie podając x czyli argumentu. A więc dla jakiego kąta zostanie obliczony sinus ? Nie wiadomo . Dlatego Matlab zgłosi błąd. To samo stało się w przypadku naszej funkcji pitagoras. Potrzebuje ona i to aż 2 argumentów do poprawnego działania:

```
>> pitagoras (2,4)
ans =
   4.4721
```

Teraz działa. Eureka . Proszę to dokładnie przemyśleć i zapamiętać. Jaki będzie wynik działania f-cji pitagoras z jednym argumentem np. pitagoras(2) ?

A teraz zastosujmy funkcje do zadania rozwiązującego układ równań z poprzedniego zestawu 7 (zad.3.) Napisz ~~skrypt~~ funkcję rozwiązującą poniższy układ równań zależny od parametru **r**. Na ekranie powinien być wyświetlony wynik.

$$\begin{bmatrix} 5 & 2r & r \\ 3 & 6 & 2r-1 \\ 2 & r-1 & 3r \end{bmatrix} \begin{Bmatrix} x \\ y \\ z \end{Bmatrix} = \begin{Bmatrix} 2 \\ 3 \\ 5 \end{Bmatrix}$$

```
function [x]= solwer(r)
A=[5 2*r r; 3 6 2*r-1; 2 r-1 3*r]
B=[2 3 5]'
x=A\B
```

Ten prosty przykład pokazuje trochę prymitywny potencjał skryptu. Obliczenia są wykonane, ale wymagają każdorazowej interwencji w treść. co się stanie jeżeli zakomentujemy pierwszą linię skryptu:

```
%r=10                    % promień
Pole=pi*r*r              % pole koła
Obwód=2*pi*r             %obwód koła
```

Wówczas obliczenia zostaną wykonane w oparciu o zmienne z przestrzeni roboczej jeżeli więc zadeklarujemy przed wykonaniem skryptu np. **r=5** wówczas otrzymamy wynik dla zadeklarowanego promienia:

```
>>r=5                   % deklaracja w trybie interaktywnym
>> kolo
r =
    5
Pole =
  78.5398
Obwod =
  31.4159
```

Uwaga. Pliki skryptowe działają na zmiennych dostępnych w przestrzenni roboczej !!!
Proszę sprawdzić czy skrypty modyfikują wartości zmiennych z przestrzeni roboczej.

Zwróć uwagę na komentarze które pełnia funkcję informacyjna i pomocową lecz mogą również wyłączać linie lub fragmenty skryptu.

A gdyby tak skrypt zawierał funkcje która poprosi o podanie promienia **r** a następnie dla wprowadzonej wartości wykona obliczenia. Funkcja taka nosi nazwę **input** i ma przykładowa składnię: **x=input(‘podaj x’);**
W celu wyświetlenia dodatkowego tekstu służy polecenie **disp(‘**tekst lub zmienna do wypisania’**);**

```
disp(‘Ten skrypt oblicza obwód i pole koła dla podanego z klawiatury promienia r’)
r=input(‘podaj promień’);    % promień
Pole=pi*r*r                  % pole koła
Obwód=2*pi*r                 %obwód koła
```

Sprawdź działanie skryptu. Czy średnik w linii 2 ma znaczenie ?.

W oparciu o przedstawiony przykład można konstruować proste rozbudowane interaktywne skrypty umożliwiające wykonanie szeregu obliczeń oraz oszczędzające czas użytkownika który zamiast wpisywać szereg poleceń modyfikuje jedynie dane wejściowe.

Komentarz do nazw

Nigdy nie należy nadawać skryptom nazw identycznych z nazwami liczonych zmiennych. Ponadto nazwa winna zaczynać się od litery. Należy uważać ab nazwy nie były w konflikcie z nazwa funkcji matlaba.
Co się właściwie dzieje gdy użytkownik wpisze polecenie **kolo** (nazwa powyższego skryptu) i naciśnie Enter

1. Matlab szuka w pamięci zmiennej kolo
2. Jeżeli nie ma takiej zmiennej, szuka czy nazwa nie jest związana z wbudowaną funkcją
3. Jeżeli nie ma takiej funkcji, Matlab szuka pliku kolo.m w bieżącym katalogu
4. Jeżeli niema takiego pliku a bieżącym katalogu przeszukiwane są katalogi z *listy* ścieżek
5. jeżeli nie ma takiego pliku wyświetlany jest komunikat :

Undefined function or variable 'kolo’

Sprawdzenia czy dana nazwa jest „wolna” można dokonać poleceniem exist(‘nazwa’). Gdy jest wolna funkcja zwraca wartość zero 0 .