

## 1. Instrukcje sterujące.

Matlab jest wyposażony w instrukcje sterujące o składni zapożyczonyj z języka C. Daje on także możliwość definiowania funkcji własnych użytkownika. Dostępne są również specyficzne narzędzia, takie jak **feval** lub **eval**, które umożliwiają wykonanie polecenia, funkcji lub wyrażenia zapisanego w postaci łańcucha. Program zawiera instrukcje warunkowe, wyboru, dwa rodzaje instrukcji iteracyjnych oraz instrukcję przerywania wykonywanych instrukcji **break** i **return**.

Tabela 10. Instrukcje sterujące programu Matlab

Nazwa	Opis słów kluczowych
<b>if</b>	warunkowe wykonanie polecenia lub sekwencji poleceń
<b>elseif</b>	używane łącznie z <b>if</b>
<b>else</b>	używane łącznie z <b>if</b>
<b>end</b>	kończy sekwencję poleceń
<b>for</b>	powtarza sekwencje poleceń określoną liczbę razy
<b>while</b>	powtarza sekwencje poleceń nieokreśloną liczbę razy
<b>switch</b>	umożliwia wybór jednej instrukcji (lub sekwencji) z szerszej palety
<b>case</b>	używane łącznie ze <b>switch</b>
<b>otherwise</b>	używane łącznie ze <b>switch</b>
<b>break</b>	przerywa wykonywanie pętli
<b>return</b>	następuje opuszczenie funkcji i powrót do miejsca jej wywołania

Ogólna postać instrukcji warunkowej **if** jest następująca:

```
if wyrażenie_1
    polecenia
elseif wyrażenie_2
    polecenia
else
    polecenia
end
```

Jeśli **wyrażenie\_1** jest liczbą różną od zera, to wykonywane są polecenia po **if**. Jeśli **wyrażenie\_1** jest równe zero, to obliczana jest wartość **wyrażenia\_2**. Jeśli **wyrażenie\_2** jest liczbą różną od zera, to wykonywane są polecenia po **elseif**. Jeśli **wyrażenie\_2** jest równe zero, to wykonywane są polecenia po **else**.

Instrukcja **for** używana jest do wykonania instrukcji lub sekwencji instrukcji określoną (znaną programiście) liczbę razy. Jej składnia jest następująca:

```
for zmienna_sterująca = wartosc_p:krok:wartosc_k
    polecenia
end
```

Jeżeli programista nie wie dokładnie ile razy pętla będzie wykonywana (liczba przebiegów może zależeć od wyników wykonywanych instrukcji), stosowana jest instrukcja **while**. Polecenia są wykonywane tak długo, dopóki **wyrażenie** przyjmuje wartość różną od zera.

Ogólna postać instrukcji **while**:

```
while wyrażenie
    polecenia
end
```

Ogólna postać instrukcji wyboru **switch** jest następująca:

```
switch wyrażenie_sterujące
case wyrażenie_1
    polecenia
case wyrażenie_2
    polecenia
otherwise
    polecenia
end
```


W powyższej instrukcji, **wyrażenie\_sterujące** może być liczbą całkowitą lub łańcuchem, natomiast **wyrażenia** po **case** mogą przyjmować postać stałej lub listy stałych (liczby, łańcuchy).

Instrukcja **switch** działa w sposób następujący:

- obliczona zostaje wartość **wyrażenia\_sterującego** wyborem;
- obliczona wartość zostaje porównana ze wszystkimi stałymi umieszczonymi na listach wyboru po **case**;
- jeżeli podczas realizacji takich porównań, dla jednego z elementów danej listy wyboru wystąpi zgodność, to wykonywana jest sekwencja poleceń umieszczona po tej liście;
- jeżeli wyniki wszystkich porównań dadzą wynik negatywny, to wykonywana jest sekwencja poleceń po słowie **otherwise**.

## 2. Programy Matlaba.

Programy Matlaba zapisywane są w postaci tzw. M-plików, tj. plików tekstowych z rozszerzeniem **\*.m**. Istnieją dwa rodzaje M-plików: skrypty i funkcje.

**Skrypty** – nie wymagają parametrów i działają na zmiennych tworzonych w przestrzeni roboczej. Aby uruchomić program (skrypt) napisany w Matlabie, należy ustawić jako folder bieżący ten folder, w którym został zapisany M-plik (przycisk ) , a następnie wpisać nazwę M-pliku (bez rozszerzenia **\*.m**) i zatwierdzić Enterem. Program można także uruchomić w oknie edytora M-pliku poleceniem

**Debug / Run**, klawiszem **F5** lub klikając przycisk  na pasku narzędziowym.

**Funkcje** – zwracają wartości i zazwyczaj wymagają podania parametrów (argumentów funkcji). Wywołania funkcji są najczęściej używane w wyrażeniach stanowiących fragmenty instrukcji innych M-plików. Zmienne używane w funkcjach są lokalne, tzn. niedostępne poza ciałem funkcji. Nazwa funkcji musi być taka sama, jak nazwa pliku, w którym jest ona zapisana.

Funkcje komunikują się z przestrzenią roboczą poprzez parametry formalne lub zmienne globalne, definiowane poleceniem **global**. Funkcje muszą się rozpoczynać od słowa kluczowego **function**. Pierwsza linia funkcji (nie licząc linii komentarza) powinna być zapisana następująco:

**function [lista\_argumentów\_wyjściowych] = nazwa\_funkcji (lista\_argumentów\_wejściowych)**

Tabela 11. Polecenia wprowadzania danych przez użytkownika

Nazwa	Opis polecenia
<b>menu</b>	generowanie okna w celu realizacji wyboru danych użytkownika
<b>input</b>	funkcja stosowana do wprowadzania danych z klawiatury
<b>pause</b>	wstrzymanie wykonania M-pliku i oczekiwanie na reakcję użytkownika
<b>keyboard</b>	wywołanie klawiatury w trakcie wykonywania M-pliku

W przypadku funkcji **input**, wprowadzane dane są domyślnie zamieniane na **liczbę**. Jeżeli chcemy wprowadzić **znak** lub **łańcuch znaków**, należy zastosować parametr **'s'**.

W tabeli 12 zamieszczono polecenia, które są najczęściej stosowane przy definiowaniu funkcji własnych użytkownika.

Tabela 12. Polecenia wykorzystywane przy definiowaniu funkcji użytkownika

Nazwa	Opis polecenia
<b>mlock</b>	blokuje usuwanie M-plików poleceniem <b>clear</b>
<b>global</b>	definiuje zmienne globalne
<b>persistent</b>	definiuje zmienne lokalne z pamięcią wartości
<b>nargin</b>	liczba argumentów wejściowych funkcji
<b>nargout</b>	liczba argumentów wyjściowych funkcji
<b>function</b>	początek M-pliku funkcyjnego
<b>munlock</b>	pozwala na usuwanie M-plików poleceniem <b>clear</b>
<b>feval</b>	wykonuje funkcje zapisane jako łańcuch
<b>eval, evalin</b>	wykonuje łańcuch jako polecenie Matlaba

### 3. Ćwiczenia.

#### Ćwiczenie 1.

- ♦ Otwórz okno edycji skryptu wybierając **New / Script** i w kolejnych wierszach wpisz polecenia:

```
d = [4 2 -1; 3 -2 5; -1 6 -2]
g = [4 31 -19]
x = g * d
```

- ♦ Zapisz skrypt na dysku pod nazwą **lin.m** używając polecenia **File / Save As**.
- ♦ Wyczyść okno poleceń Matlaba poleceniem **clc**, a następnie wprowadź polecenie:  
**>> lin**
- ♦ Uruchom skrypt różnymi metodami – w oknie edytora wybierz polecenie **Debug / Run lin**, wciśnij

klawisz **F5**, kliknij przycisk  na pasku narzędziowym.

#### Ćwiczenie 2.

- ♦ Napisz skrypt służący do rozwiązania równania kwadratowego.

```
% Program do rozwiązywania równania kwadratowego
clear all, clc;
disp('Podaj współczynniki równania kwadratowego:');
a = input('a = ');
b = input('b = ');
c = input('c = ');
delta = b * b - 4 * a * c;
if delta > 0
    disp('Dwa pierwiastki:')
    x1 = (-b - sqrt(delta)) / (2 * a);
    x2 = (-b + sqrt(delta)) / (2 * a);
    disp(['x1 = ', num2str(x1), ', x2 = ', num2str(x2)]);
elseif (delta == 0)
    disp('Jeden podwójny pierwiastek:')
    x12 = -b / (2 * a);
    disp(['x12 = ', int2str(x12)]);
else
    disp('Brak pierwiastków rzeczywistych');
end
```

Nawias kwadratowy, zastosowany wewnątrz funkcji **disp()**, pozwala na wyświetlenie w jednym wierszu łańcuchów znaków oraz wartości zmiennych.

#### Ćwiczenie 3.

- ♦ Za pomocą skryptu oblicz kwadraty liczb parzystych od **2** do **10** i podstaw je do kolejnych elementów wektora **p**.

```
for i = 1:5
    p(i) = (2*i)^2;
end
disp(p)
```

#### Ćwiczenie 4.

- ♦ Napisz skrypt, w którym obliczysz i wstawiś do wektora **y** ciąg wartości funkcji **sinus**. Przyjmij zakres kąta (w radianach) od **zera** do **pi/2** oraz przyrost równy **0.1**.

```
k = 0;
for x = 0:0.1:pi/2
    k = k + 1;
    y(k) = sin(x);
end
disp(y)
```

### Ćwiczenie 5.

- ♦ Napisz skrypt, w którym za pomocą pętli zagnieżdżonej **for** wygenerujesz macierz o 3 wierszach i 4 kolumnach.

```
for w = 1:3
    for k = 1:4
        M(w, k) = w + k;
    end
end
disp(M)
```

### Ćwiczenie 6.

- ♦ Zapisz poniższy skrypt pod nazwą **liczba.m**.

```
licznik = 1;
a = rand(1, 1);
b = round(a * 10);
disp('Wylosowano liczbę z przedziału [1, 10]')
x = input('Zgadnij, jaka liczba została wylosowana: ');
while (x ~= b)
    x = input('Zgadnij jeszcze raz: ');
    licznik = licznik + 1;
end
disp(['Trafiłeś za ', num2str(licznik), ' razem'])
```

- ♦ Uruchom skrypt i sprawdź jego działanie.
- ♦ Określ przeznaczenie każdej instrukcji zapisanej w skrypcie.

### Ćwiczenie 7.

- ♦ Napisz funkcję służącą do obliczania silni liczby podanej jako jej argument.

```
function [wynik] = silnia(n)
wynik = 1;
for i = 1:n
    wynik = wynik * i;
end
```

- ♦ Zapisz plik pod taką samą nazwą, jak nazwa funkcji, czyli **silnia.m**.
- ♦ Sprawdź działanie funkcji, wywołując ją w oknie poleceń z różnymi argumentami, np.:  
**>> silnia(5)**

## **4. Zadania.**

### Zadanie 1.

Napisz program, który dla trzech liczb rzeczywistych (**a**, **b**, **c**) pobranych od użytkownika, wyświetli je na ekranie w porządku rosnącym.

### Zadanie 2.

Napisz program sprawdzający warunki istnienia trójkąta, przy zadanych wartościach jego boków wprowadzanych do programu po jego uruchomieniu.

### Zadanie 3.

Napisz program, który dla dowolnej liczby **x** wprowadzonej z klawiatury, sprawdzi przy pomocy funkcji **mod(x,2)**, czy jest to liczba parzysta, nieparzysta, niecałkowita. Po sprawdzeniu, powinien zostać wyświetlony odpowiedni komunikat.

#### **Zadanie 4.**

Napisz program, który obliczy pole prostokąta, koła lub trójkąta, w zależności od wybranej opcji menu (1 – prostokąt, 2 – koło, 3 – trójkąt). W programie zastosuj instrukcję **switch**.

#### **Zadanie 5.**

Napisz program wykonujący cztery podstawowe działania arytmetyczne na dwóch wprowadzanych z klawiatury liczbach rzeczywistych. Rodzaj wykonywanego działania powinien zależeć od wybranej opcji menu (+ dodawanie, – odejmowanie, \* mnożenie, / dzielenie). W programie zastosuj instrukcję **switch**. Zabezpiecz także program przed wprowadzeniem zera jako dzielnika.

#### **Zadanie 6.**

Napisz program z pętlą **for...end**, który wyświetli na ekranie pierwiastki kwadratowe z kolejnych liczb nieparzystych od 1 do 9.

#### **Zadanie 7.**

Napisz program z pętlą **for...end**, który dla ciągu wartości  $\varphi$  od 3,6 do 13 co 0,4 obliczy wartości wyrażenia  $\beta = (\varphi - 0,5) / (1,1 + \sin(\varphi))$ .

#### **Zadanie 8.**

Napisz program, który dla pobranego od użytkownika ciągu liczb  $a_1, a_2, \dots, a_n$ , zakończonego liczbą 0 (zero kończy wprowadzanie danych), wyznaczy i wyświetli element minimalny oraz jego indeks (nie uwzględniaj końcowego zera).

#### **Zadanie 9.**

Napisz program, który wczytuje z klawiatury liczby rzeczywiste do momentu, gdy ich suma przekroczy wartość 1000, a następnie oblicza średnią arytmetyczną wprowadzonych liczb i wyświetla stosowną informację.

#### **Zadanie 10.**

Zmodyfikuj skrypt z ćwiczenia nr 2 w ten sposób, aby pierwiastki równania kwadratowego obliczane były za pomocą funkcji. Przykładowe wywołanie funkcji w oknie poleceń powinno mieć postać:

```
>> [x1, x2] = prkw(2, 5, 2)
```

gdzie: **prkw** to przykładowa nazwa funkcji, a liczby w nawiasie okrągłym to wartości współczynników **a**, **b** i **c** przekazywane jako argumenty wywołania funkcji.