

Wprowadzenie do Scilab: podstawy języka Scilab

Magdalena Deckert, Izabela Szczęch,
Barbara Wołyńska, Bartłomiej Prędko

Politechnika Poznańska, Instytut Informatyki

Narzędzia Informatyki



Agenda

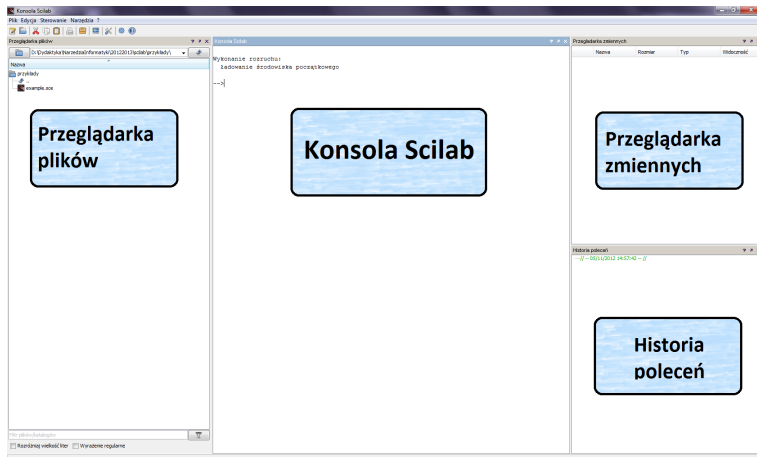
- 1 Pierwsze kroki
 - Scilab
 - Edytor
 - Przeglądarka zmiennych oraz historia poleceń
 - Skrypty
- 2 Podstawy języka Scilab
 - Zmienne
 - Polecenia wieloliniowe i komentarze
 - Zmienne predefiniowane
 - Zmienne logiczne i operatory porównania
 - Zmienna ans
 - Łańcuchy znaków
 - Zmienne liczbowe
 - Dynamiczne typowanie zmiennych
 - Instrukcje warunkowe
 - Pętle
 - Operacje wejścia-wyjścia

Korzystanie ze Scilab

Istnieją 3 główne sposoby korzystania ze Scilab:

- za pomocą konsoli Scilab
- poprzez wykonywanie poleceń zapisanych w plikach za pomocą komendy `exec`
- poprzez uruchamianie z linii poleceń

Widok konsoli Scilab po uruchomieniu



Konsola Scilab umożliwia ciągłe wpisywanie i uruchamianie poleceń oraz analizowanie uzyskanych wyników.

Pierwsze polecenie

Przykład 1

```
-->s="Hello World!"  
s = Hello World!  
-->disp(s)  
Hello World!
```

Znak `-->` jest znakiem zachęty wyświetlanym w konsoli Scilab. W pierwszym kroku tworzymy zmienną `s`, do której przypisujemy określony ciąg znaków. Po wykonaniu polecenia poprzez naciśnięcie *Enter* Scilab wyświetla wykonaną komendę. W następnym kroku za pomocą funkcji `disp()` wyświetlamy wartość zmiennej `s`.

Pierwsze polecenie



```
Konsola Scilab

Wykonanie rozruchu:
  ładowanie środowiska początkowego

-->s="Hello world!"
s =

Hello world!

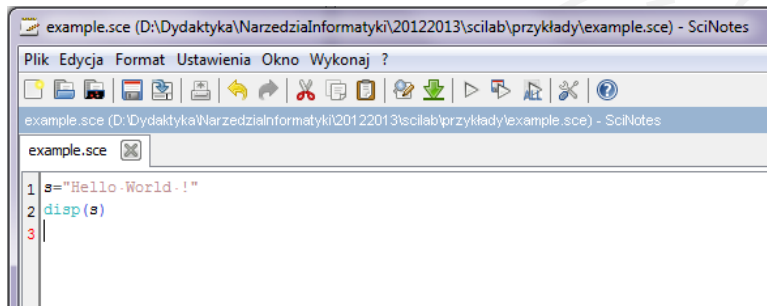
-->disp(s)

Hello world!

-->|
```

Wpisane polecenia można modyfikować jak w zwykłym edytorze przesuwając się po wpisanym poleceniu za pomocą strzałek ← lub →, a następnie kasując błędy. Za pomocą strzałek ↑ lub ↓ mamy dostęp do poprzednio wykonanych poleceń.

Edytor



Scilab udostępnia także edytor do tworzenia i modyfikacji skryptów. Jest on dostępny poprzez menu *Narzędzia* -> *SciNotes* lub z poziomu konsoli wykonując polecenie *editor()*.

Edytor

Najczęściej wykorzystywanymi funkcjami edytora są:

- *wykonaj plik bez echa* – wywołuje operację `exec` na danym skrypcie; na konsoli pojawiają się tylko wyniki powiązane z funkcjami wyświetlającymi
- *wykonaj plik z echem* – kopiuje i wkleja polecenia z pliku do konsoli Scilab; na konsoli wyświetlane są wykonywane operacje (jeśli nie są zakończone znakiem `;`) oraz wyniki powiązane z funkcjami wyświetlającymi
- *wykonaj zaznaczenie z echem* – kopiuje i wkleja polecenia z pliku do konsoli Scilab tylko z zaznaczonego obszaru; na konsoli wyświetlane są wykonywane operacje (jeśli nie są zakończone znakiem `;`) oraz wyniki powiązane z funkcjami wyświetlającymi

Edytor ułatwia analizowanie skryptów poprzez *kolorowanie składni*.

Pierwszy skrypt

Przykład 2

Utwórz skrypt w edytorze zawierający następujące polecenia.

```
s="Hello World!"  
disp(s)
```

Uruchom go na 3 znane sposoby i przyjrzyj się różnicom w wyświetlanych wynikach.

Pierwszy skrypt - wygląd konsoli



```
Konsola Scilab

-->exec('D:\Dydaktyka\NarzedziaInformatyki\20122013\scilab\przyklady\example.sce', -1)

Hello World !

-->s="Hello World !"
s =

Hello World !
-->disp(s)

Hello World !

-->disp(s)

Hello World !

-->
```

1.

2.

3.

- 1.—wynik wykonania pliku bez echa
- 2.—wynik wykonania pliku z echem
- 3.—wynik wykonania zaznaczenia z echem

Przeglądarka zmiennych oraz historia poleceń

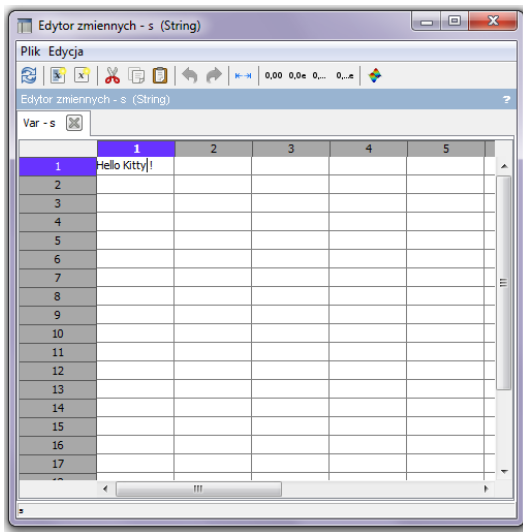
- Dostęp do przeglądarki zmiennych możliwy jest poprzez menu *Narzędzia -> Przeglądarka zmiennych* lub z poziomu konsoli wykonując polecenie *browsevar()*. Klikając dwukrotnie na wybranej zmiennej mamy możliwość zmiany jej wartości w *edytorze zmiennych*. Jeśli w międzyczasie zmienimy wartość zmiennej z poziomu konsoli, to należy pamiętać o odświeżeniu widoku w otwartym edytorze zmiennych.
- Historia poleceń umożliwia przeglądanie wszystkich wykonanych wcześniej komend. Dostęp do historii możliwy jest poprzez menu *Narzędzia -> Historia poleceń*. Umożliwia ona wybranie dowolnego polecenia oraz wykonanie go w konsoli poprzez dwukrotne kliknięcie.

Przeglądarka zmiennych oraz historia poleceń - przykład

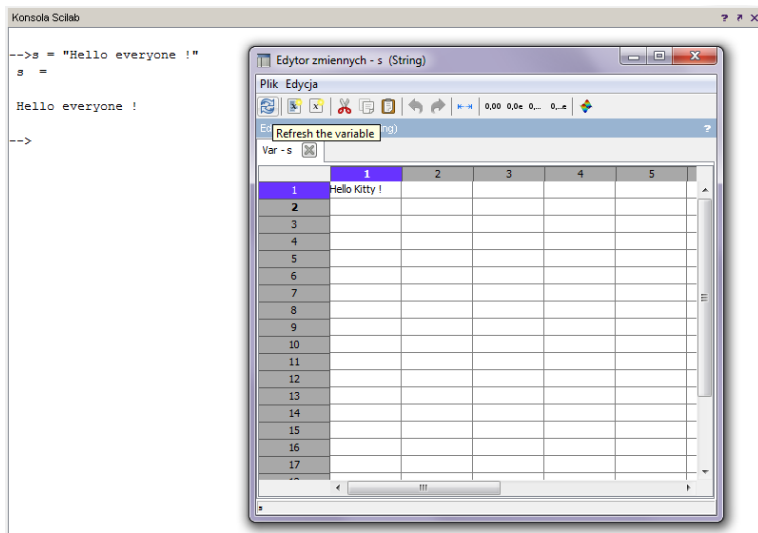
Przykład 3

- Zmodyfikuj wartość zmiennej `s` z `"Hello World!"` na `"Hello Kitty"` za pomocą edytora zmiennej.
- Nie zamykając edytora zmiennej, przejdź do konsoli Scilab i zmień wartość zmiennej `s` za pomocą polecenia `s="Hello everyone!"`
- Przejdź do otwartego wcześniej okna edytora zmiennej. Czy wartość zmiennej `s` uległa zmianie?
- Odśwież widok w edytorze zmiennej. Czy teraz wartość zmiennej `s` zmieniła się?
- Odszukaj w historii poleceń komendę `s="Hello world!"` i wykonaj ją.
- Sprawdź, czy wartość zmiennej `s` wróciła do początkowej wartości.

Modyfikacja wartości zmiennej w edytorze zmiennych



Widok edytora zmiennej przed odświeżeniem



Polecenie *exec*

- Wielokrotnie uruchamiane polecenia warto zapisać do pliku skryptowego.
- Pliki skryptowe mogą mieć dwa rozszerzenia: *.sci* lub *.sce*.
- Pliki *.sci* zawierają tylko definicje funkcji, które są następnie wczytywane do środowiska Scilab, ale nie wykonywane.
- Pliki *.sce* zawierają zarówno funkcje jak i inne polecenia. Wykonanie takiego pliku *.sce* powoduje wczytanie funkcji oraz wykonanie wszystkich zamieszczonych poleceń.
- Pliki skryptowe wykonywane są z poziomu konsoli poprzez wywołanie polecenia `-->exec ("nazwaPliku.(sce|sci)")`.

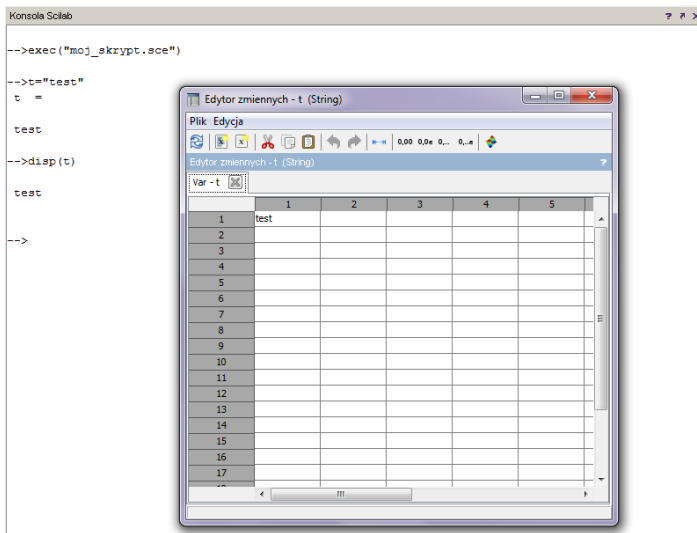
Polecenie `exec` - przykład

Przykład 4

- Otwórz notatnik i stwórz nowy dokument o nazwie `moj_skrypt.sce`. Plik należy umieścić w katalogu bieżącym lub należy zmodyfikować ścieżkę do katalogu domyślnego.
- Zawartość skryptu ma być następująca:

```
t="test"
disp(t)
```
- Zapisz wszelkie zmiany w skrypcie.
- Przejdź do konsoli Scilab.
- Uruchom polecenie `-->exec("moj_skrypt.sce")`.
- Sprawdź w przeglądarce zmiennych, czy została stworzona nowa zmienna `t`. Sprawdź także w edytorze zmiennej jaka jest wartość nowej zmiennej.

Polecenie `exec` - wynik przykładu



Zmienne

- W języku Scilab nie ma konieczności deklarowania zmiennych. Są one tworzone w momencie pierwszego przypisania wartości.
- W Scilabie wartość do zmiennej przypisuje się za pomocą operatora `=`.
- Wartość zmiennej jest wyświetlana po każdym przypisaniu wartości. Jeśli chcemy wyłączyć tę opcję, to należy na końcu polecenia dodać znak `;`.
- Nazwy zmiennych mogą być dowolnej długości, ale tylko początkowe 24 znaki są brane pod uwagę. Nazwy mogą się składać z małych liter, dużych liter, cyft oraz znaków specjalnych takich jak: `%`, `_`, `#`, `!`, `$` oraz `?`.
UWAGA: Zmienne, których nazwa rozpoczyna się od `%` mają specjalne znaczenie! Są to tak zwane zmienne predefiniowane, np. liczba pi to `%pi`.

Podstawowe operatory matematyczne

+	dodawanie
-	odejmowanie
*	mnożenie
/	prawe dzielenie $x/y = xy^{-1}$
\	lewe dzielenie $x \backslash y = x^{-1}y$
^	potęgowanie
**	potęgowanie
'	transpozycja macierzy i sprzężenie liczby zespolonej

Tablica 1 : Podstawowe operatory matematyczne

Zmienne - przykład

Przykład 5

- Przejdź do konsoli Scilab.
- Wykonaj następujące polecenia:

```
-->x=1  
-->x=x*2  
-->y=2;  
-->y=y^2;
```
- Sprawdź w przeglądarce zmiennych, czy zostały stworzone nowe zmienne x i y . Sprawdź ich wartości w edytorze zmiennej.

Zmienne - wynik przykładu

Konsola Scilab

```
-->x=1  
x =  
  
1.  
  
-->x=x*2  
x =  
  
2.  
  
-->y=2;  
  
-->y=y^2;  
  
-->
```

Edytor zmiennych - y (Double)

Plik Edycja

Edytor zmiennych - y (Double)

Var - x Var - y

	1	2	3	4	5
1	4				
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					
16					
17					

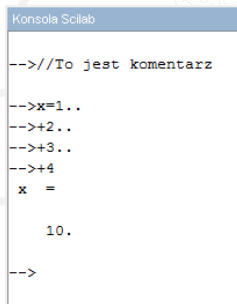
Polecenia wieloliniowe i komentarze

- W przypadku długich poleceń, które nie mieszczą się w jednej linii istnieje możliwość podzielenia ich na wiele linii. Taka linia, która będzie miała swój ciąg dalszy w kolejnej linii musi kończyć się znakiem dwóch kropek ...
- Każda linia, która zaczyna się // traktowana jest jako komentarz.

Przykład 6

- Przejdź do konsoli Scilab.
- Wykonaj następujące polecenia:

```
--> // To jest komentarz  
--> x=1..  
--> +2..  
--> +3..  
--> +4
```



```
Konsola Scilab  
  
--> // To jest komentarz  
  
--> x=1..  
--> +2..  
--> +3..  
--> +4  
x =  
    10.  
  
-->
```

Zmienne predefiniowane

Zmienne, których nazwy rozpoczynają się od symbolu % to *zmienne predefiniowane*. Oto przykładowe zmienne predefiniowane.

%i	jednostka urojona liczby zespolonej
%e	stała Eulera
%pi	stała pi
%t	prawda
%T	prawda
%f	fałsz
%F	fałsz

Tablica 2 : Zmienne predefiniowane

Zmienne logiczne i operatory porównania

Zmienne, które przyjmują wartość *prawda* lub *fałsz* to *zmienne logiczne*.

$a \& b$	logiczne oraz
$a b$	logiczne lub
$\sim a$	logiczne nie
$a == b$	prawda jeśli a jest równe b
$a \sim = b$ lub $a <> b$	prawda jeśli a nie jest równe b
$a < b$	prawda jeśli a jest mniejsze od b
$a > b$	prawda jeśli a jest większe niż b
$a \leq b$	prawda jeśli a jest mniejsze lub równe b
$a \geq b$	prawda jeśli a jest większe lub równe b

Tablica 3 : Operatory porównania

Zmienne logiczne i operatory porównania - przykład

Przykład 7

- Przejdź do konsoli Scilab.
- Wykonaj następujące polecenia:
-->a = %T
-->b = (0==1)
-->a&b
- Co jest wynikiem ostatniej operacji?

```
Konsola Scilab

-->a = %T
a =

T

-->b = (0==1)
b =

F

-->a&b
ans =

F

-->
```

Zmienna *ans*

- Za każdym razem gdy wynik obliczenia nie jest jawnie przypisywany do zmiennej, to jest on przechowywany w zmiennej domyślnej *ans*.
- Gdy wynik został przypisany do zmiennej *ans*, to można korzystać z niej jak z każdej zmiennej.

Przykład 8

- Przejdź do konsoli Scilab.
- Wykonaj następujące polecenia:
-->exp(4)
-->log(ans)

```
Konsola Scilab  
  
-->exp(4)  
ans =  
  
    54.59815  
  
-->log(ans)  
ans =  
  
    4.  
  
-->|
```

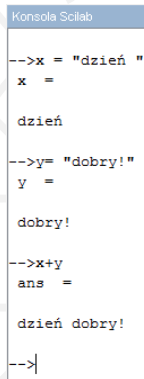
Łańcuchy znaków

Łańcuchy znaków są przechowywane w zmiennych, których wartość ograniczona jest znakami cudzysłowu ".

Przykład 9

- Przejdź do konsoli Scilab.
- Wykonaj następujące polecenia:

```
-->x = "dzień "  
-->y = "dobry!"  
-->x+y
```



```
Konsola Scilab  
  
-->x = "dzień "  
x =  
  
dzień  
  
-->y= "dobry!"  
y =  
  
dobry!  
  
-->x+y  
ans =  
  
dzień dobry!  
  
-->|
```

Liczby całkowite

W Scilab dostępne są następujące typy liczb całkowitych.

int8	int16	int32
uint8	uint16	uint32

Tablica 4 : Typy liczb całkowitych

Zakres liczby całkowitej zależy od użytej liczby bitów.

- Dla liczby całkowitej ze znakiem (*signed*) zakres wartości jest następujący: $[-2^{n-1}, 2^{n-1} - 1]$.
- Dla liczby całkowitej bez znaku (*unsigned*) zakres wartości jest następujący: $[0, 2^n - 1]$.

Liczby całkowite

Do stworzenia liczby całkowitej określonego typu wykorzystuje się następujące funkcje:

$y = \text{int8}(x)$	$y = \text{int16}(x)$	$y = \text{int32}(x)$
$y = \text{uint8}(x)$	$y = \text{uint16}(x)$	$y = \text{uint32}(x)$

Tablica 5 : Funkcje Scilab do tworzenia liczb całkowitych

Aby sprawdzić typ liczby całkowitej wykorzystuje się funkcję *inttype(x)*.

Zwracana wartość	Typ liczby całkowitej
1	8-bitowa ze znakiem
2	16-bitowa ze znakiem
4	32-bitowa ze znakiem
11	8-bitowa bez znaku
12	16-bitowa bez znaku
14	32-bitowa bez znaku

Tablica 6 : Wartości zwracane przez funkcję *inttype(x)*

Liczby całkowite - przykład

Przykład 10

- Przejdź do konsoli Scilab.
- Wykonaj następujące polecenia:

```
-->x = uint32(0)
-->y = x - 1
-->inttype(y)
```

Uzyskana wartość y jest niepoprawna z uwagi na to, że przekroczyliśmy zakres zmiennej typu `uint32`.

```
Konsola Scilab

-->x = uint32(0)
x =

    0

-->y = x - 1
y =

4294967295

-->inttype(y)
ans =

    14.
```

Liczby zmiennoprzecinkowe

Domyślnym typem liczbowym w Scilab jest *double*, czyli 64-bitowy typ liczby zmiennoprzecinkowej.

Przykład 11

- Przejdź do konsoli Scilab.
- Wykonaj następujące polecenia:
-->2^52
-->uint16(2^52)
- Pierwsza liczba jest obliczona poprawnie.
- Druga liczba jest błędna, ponieważ wynik nie mieści się w typie uint16.

```
Konsola Scilab
-->2^52
ans =
    4.504D+15
-->uint16(2^52)
ans =
    0
-->
```

Dynamiczne typowanie zmiennych

Scilab dynamicznie zmienia typ zmiennej w zależności od przypisanej wartości.

Przykład 12

- Przejdź do konsoli Scilab.
- Wykonaj następujące polecenia:

```
-->x = 1  
-->x + 6  
-->x = "dzień "  
-->x + "dobry!"
```

```
Konsola Scilab  
  
-->x = 1  
x =  
  
1.  
  
-->x + 6  
ans =  
  
7.  
  
-->x = "dzień "  
x =  
  
dzień  
  
-->x + "dobry!"  
ans =  
  
dzień dobry!  
  
-->
```


Instrukcje warunkowe - klauzula *if*

- Klauzula *if* wykonuje określony blok operacji, gdy podany warunek logiczny jest spełniony. W przypadku, gdy testowany warunek logiczny nie jest spełniony, to wykonywany jest blok operacji występujący po słowie kluczowym *else*. Jeśli chcemy sprawdzać kolejno kilka warunków logicznych, to pomocny może się okazać blok *elseif*, których można użyć wiele.
- Składnia klauzuli *if* jest następująca:

```
if warunek_logiczny then  
    blok operacji 1  
[ elseif warunek_logiczny_2 then  
    blok operacji 2  
else  
    blok operacji 3]  
end
```



Klauzula *if*

Przykład 13

- Stwórz skrypt zawierający następujące komendy:

```
i=2
if (i == 1) then
disp("Pierwszy warunek if jest prawdziwy.")
elseif (i==2) then
disp("Drugi warunek if jest prawdziwy.")
else
disp("Żaden z powyższych warunków nie jest prawdziwy.")
end
```

- Przejdź do konsoli Scilab i wykonaj powyższy skrypt.

Instrukcje warunkowe - klauzula *select*

- Klauzula *select* sprawdza wartość określonej zmiennej i w zależności od jej wartości wykonuje określony blok operacji *case*. W przypadku, gdy wartość zmiennej nie wystąpiła na liście *case*, to wykonywany jest blok operacji po słowie kluczowym *else*. Jest on opcjonalny, aczkolwiek umieszczanie go jest dobrą praktyką.
- Składnia klauzuli *select* jest następująca:

```
select zmienna  
case wartość_1  
    blok operacji 1  
case wartość_2  
    blok operacji 2  
[ else  
    blok operacji 3]  
end
```

Klauzula *select*

Przykład 14

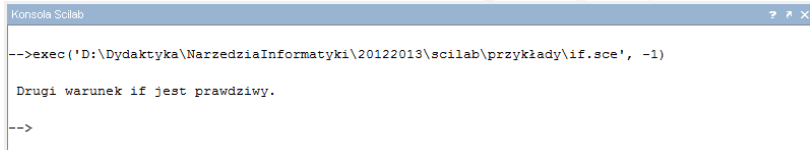
- Stwórz skrypt zawierający następujące komendy:

```
i=2
select i
case 1
disp("Wartość zmiennej jest równa 1.")
case 2
disp("Wartość zmiennej jest równa 2.")
else
disp("Wartość zmiennej nie jest równa 1 ani 2.")
end
```

- Przejdź do konsoli Scilab i wykonaj powyższy skrypt.

Instrukcje warunkowe - wyniki przykładów

- klauzula *if*



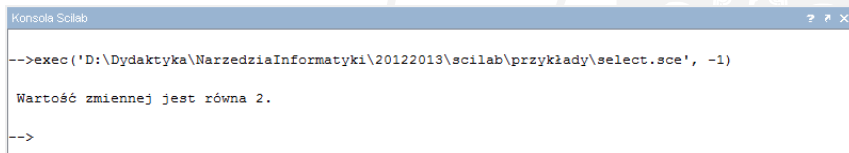
```
Konsola Scilab

-->exec('D:\Dydaktyka\NarzedziaInformatyki\20122013\scilab\przykłady\if.sce', -1)

    Drugi warunek if jest prawdziwy.

-->
```

- klauzula *select*



```
Konsola Scilab

-->exec('D:\Dydaktyka\NarzedziaInformatyki\20122013\scilab\przykłady\select.sce', -1)

    Wartość zmiennej jest równa 2.

-->
```

Pętla *for*

- Pętla *for* wykonuje określony zbiór operacji wielokrotnie. Liczba wykonań zależy od wartości zmiennej sterującej. W większości przypadków iteracja wykonywana jest na wartościach całkowitych zaczynając od początkowej wartości aż do osiągnięcia końcowej wartości z określonym krokiem. Jeśli krok nie jest zdefiniowany inaczej, to wartość zmiennej sterującej jest zwiększana o 1.
- Składnia pętli *for* jest następująca:

```
for start : [krok :] koniec  
    blok operacji  
end
```

Pętla *for*

Przykład 15

- Stwórz skrypt zawierający następujące komendy:

```
for i = 1 : 5  
    disp(i)  
end  
for i = 1 : 2 : 5  
    disp(i)  
end  
for i = 5 : -1 : 1  
    disp(i)  
end
```

- Przejdź do konsoli Scilab i wykonaj powyższy skrypt.

Konsola Scilab

-->exec('D

1.

2.

3.

4.

5.

1.

3.

5.

5.

4.

3.

2.

1.

-->

Pętla *while*

- Pętla *while* wykonuje określony zbiór operacji dopóki warunek testowy jest spełniony. Jeśli warunek testowy przyjmuje wartość *false*, to wykonywanie pętli zostaje zakończone.
- Składnia pętli *while* jest następująca:

```
while warunek_logiczny  
    blok operacji  
end
```


Pętla *while*

Przykład 16

- Stwórz skrypt zawierający następujące komendy:

```
s = 0  
i = 0  
while ( i < 10)  
    i = i + 1  
    s = s + i  
end  
disp(s)
```

- Przejdź do konsoli Scilab i wykonaj powyższy skrypt.

Polecenia *break* i *continue*

Z wykonywaniem operacji w pętlach ściśle powiązane są następujące dwa polecenia:

- *break* służy do przerywania wykonywania całej pętli. Po wywołaniu tego polecenia wykonywanie aktualnej pętli nie jest wznowiane.
- *continue* służy do przerywania wykonywania aktualnej iteracji pętli. Operacje występujące w ciele pętli po poleceniu *continue* nie są wykonywane w danej iteracji. Sterowanie wraca bezpośrednio do klauzuli *while* lub *for* w zależności od rodzaju wykonywanej pętli.

Polecenie *break*

Przykład 17

- Stwórz skrypt zawierający następujące komendy:

```
s = 0
i = 0
while ( %t )
i = i + 1
if ( i > 10) then
break
end
s = s + i
end
disp(s)
```

- Przejdź do konsoli Scilab i wykonaj powyższy skrypt.

Polecenie *continue*

Przykład 18

- Stwórz skrypt zawierający następujące komendy:

```
s = 0
i = 0
while ( i < 10 )
i = i + 1
if ( modulo (i,2) == 0) then
continue
end
s = s + i
end
disp(s)
```

- Przejdź do konsoli Scilab i wykonaj powyższy skrypt.

Pętla *while*, instrukcje *break* i *continue* - wyniki przykładów

- pętla *while*

```
Konsola Scilab
-->exec('D:\Dydaktyka\NarzedziaInformatyki\20122013\scilab\przyklady\while.sce', -1)

55.
-->
```

- instrukcja *break*

```
Konsola Scilab
-->exec('D:\Dydaktyka\NarzedziaInformatyki\20122013\scilab\przyklady\break.sce', -1)

55.
-->
```

- instrukcja *continue*

```
Konsola Scilab
-->exec('D:\Dydaktyka\NarzedziaInformatyki\20122013\scilab\przyklady\continue.sce', -1)

25.
-->
```

Operacje wejścia-wyjścia

- Aby wczytać wartości wprowadzoną z klawiatury należy wykorzystać funkcję *input*.
--> `n = input("Wprowadź wartość zmiennej n: ")`
- Aby wyświetlić wartość zmiennej wraz z tekstem należy skorzystać z funkcji *mprintf*. Podstawowe typy wyświetlanych zmiennych to:
`%i` - liczba całkowita, `%f` - liczba zmiennoprzecinkowa, `%s` - łańcuch znaków.
--> `mprintf("Wartość wprowadzonej zmiennej n wynosi: %i", n)`

Konsola Scilab

```
-->n = input("Wprowadź wartość zmiennej n:")
Wprowadź wartość zmiennej n:7
n =

    7.

-->mprintf("Wartość wprowadzonej zmiennej n wynosi: %i", n)
Wartość wprowadzonej zmiennej n wynosi: 7
-->
```

Literatura

Materiały przygotowane na podstawie "Introduction to Scilab".
<http://www.scilab.org/support/documentation/tutorials>