

1 Introduction to programming in Scilab

1.1 Remarks

1. All variables are matrices. Scalar is matrix $a(1,1)$. Vector in first row is $a(1,:)$, in first column $a(:,1)$. The sign $:$ means "all".
2. Semi-column $;$ means: no response. If there is comma or nothing in the end of a command, its value is printed on the screen.
Remark: the command `mode(0)` must be called at the beginning.
3. `help „object“` gives help on "object".
Icon `?` calls the main help.
4. Comment begins with `//`.

1.2 Variables and operations

There are the following main typed of variables:

- **čísła (matice)**

Definition:

- scalar `a=5`;
- row vector `a=[3 5 1]`;
- column vector `a=[3; 5; 1]`, which is the same as `a=[3 5 1]'`
- matrix `a=[2 3 4; 8 7 6]`;
- command `a=5:8` creates the vector `[5 6 7 8]`; `5:2:13 = [5 7 9 11 13]`
- command `a=zeros(2,3)` creates matrix 2×3 from zeros
- command `a=ones(2,3)` creates matrix 2×3 from ones
- transposition is performed by `'` (apostroph)
- matrix `b (3x3)` can be composed like this: `b=[a; 2*a; 5*a]`;

Operations:

- product of matrices `*` division `/` power `^` or `**` square root `sqrt()`
- dot operations `.*` `./` `.^` are performed entry by entry
- in operation `*` the rules of matrix product hold
- operation `a/b` means multiplication of `a` by inversion of `b`
(inversion itself is `inv(b)`)

- **text:** `a='hello'`. It is a vector of letters can be concatenated:
`a='hello '`; `b='boys'` `a c=a+b`, then `c='hello boys'`.

Conversion: `s=string(a)` gives value of variable `a` as a string

- **logical variables** - their values are „true“ (`=1`) a „false“ (`=0`).

Logical operations: `==` `~=` `<` `<=` `>` `>=` `&` (and) `|` (or) `~` (not)

Examples

Set:

```
a=[1 2 3] b=[8; 9] c=[11 12 13; 21 22 23; 31 32 33];
```

Try and justify:

```
x1=a*a'    x2=a'*a    y=[[a;5*a] b]    c(2,:).*a    c(1,2:3)*b
c(3,:).^c(1,:)    c(3,:)**2    d1=c(:)    dd=c'; d2=dd(:)    d2(3:2:7)
```

Set:

```
u='first'    v='attempt'    x=%t (setting of "true")    y=5==5    z=5>5
```

Try and justify:

```
u+' '+v    x & y    x & z    x | y    x | z
```

1.3 Work with variables

- Command `who_user()`; gives information about defined variables.
- `[m,n]=size(a)`, `m=size(a,1)`, `n=size(a,2)` give dimensions of the matrix `a`, resp. number of rows, number of columns. Instead of 1 a 2 one can use 'r' a 'c'.
- `n=length(a)` number of elements of `a`.
- `n=max(size(a))` length of a vector
- `clc` clears screen
- `clear` clears variables
- `xdel(winsid())` clears all graphs (close clears the last one)

1.4 Programming commands

- **Condition if**

```
if b>c,
    a=5;
else
    a=0;
end
```

If `b>c` is true, it is performed `a=5`; otherwise `a=0`;

Example

```
// Determine c as bigger from a, b
a=rand(1,1,'n'); b=rand(1,1,'n');
if a>b, c=a;
else c=b;
end
printf('a = %g, b = %g, c = %g\n',a,b,c)
```

- **Branching of program**

```
select i,
  case 1, prikaz_A;
  case 2, prikaz_B;
  else prikaz_D
end
```

According to `i` the respective command is performed.

- **Example**

```
// According to i perform
// set the vectors
a=[1 3 5]; b=[2 4 6];
// 1 - addition
// 2 - scalar product
// 3 - tenzor product
// set the operation
```

- **cont.**

```
i=2;
select i
case 1, d=a+b;
case 2, d=a*b';
case 3, d=a'*b;
end
disp(d,'result')
```

- **Cycle for**

```
for i=1:5
  a(i)=2*i;
end
```

For $i=1,2,3,4,5$ the command `a(i)=2*i`; is performed. :Result is `a=[2, 4, 6, 8, 10]`.

- **Example 1**

```
// Determine weighted sum
x=[1 2 3 4 5 6]; // numbers
p=[.1 .3 .2 .1 .2 .1]; // weights
n=length(x);
s=0;
for i=1:n
  s=s+x(i)*p(i);
end
disp(s,'the average is')
```

- **Example 2**

```
// Order numbers according to magnitude
n=10; // how many numbers
a=fix(100*rand(1,n,'u')); // čísla
disp(a,'original numbers')
b=[];
for i=1:n
  [x,j]=min(a);
  b=[b x];
  a(j)=NaN;
end
disp(b,'ordered numbers')
end
```

- **Control of the program**

```
pause  stops the program.
resume  resumes the program after pause
abort   stops the program definitely.
```

- **Calling of subprogram**

```
exec('my_program',-1)  runs the program my_program (-1 suppresses response)
```

- **Loading functions to memory**

```
getd('my_address')  loads all subroutines in the address moje_adresa
(Scilab does not have path. It knows only the loaded functions)
```

1.5 Printing

Commands `disp` and `fprintf` .

- `disp(a)` shows value of `a`.
- `disp(a,'text')` gives value and the text
- `fprintf('entry %d of vector a is %g\n',i,a(i));`
gives e.g.: *entry 5 of vector a is 4.12*

1.6 Graphical output

Two-dimensional graph can be constructed by `plot`.

Examples:

- `plot(y)` draws values of `y`.
- `plot(x,y)` draws values of `y` against of `x` (so called xy-graf).
- `plot(a)` draws columns of matrix `a`.

Formatting of a graph:

Line	Points	Color
- (full)	. (point)	r (red)
: (dotted)	+ (plus)	g (green)
-. (dot dashed)	o (ring)	b (blue)
- (dashed)	x (cross)	w (white)

For more details, call: `help plot` or go to Scilab help: `SCILAB HELP >> GRAPHICS > GLOBALPROPERTY`

Examples:

- `plot(x,'or')` draws `x` using red crosses.
- `plot(x,y,'r-+',u,v,'b-x')` draws two curves `(x,y)` a `(u,v)`; the first one is red by full line with pluses, the second one by blue line with crosses.