Information technology – lecture 11

Basic algorithms and data structures. Scalars, arrays and lists.

Roman Putanowicz
R.Putanowicz@L5.pk.edu.pl

Project "The development of the didactic potential of Cracow University of Technology in the range of modern construction" is co-financed by the European Union within the confines of the European Social Fund and realized under surveillance of Ministry of Science and Higher Education

The use of global variables

```
function setval()
  global X
  X = 23;
end

global X=111;

setval();

X
```

Project "The development of the didactic potential of Cracow University of Technology in the range of modern construction" is co-financed by the European Union within the confines of the European Social Fund and realized under surveillance of Ministry of Science and Higher Education

Calculating sum of vector elements

```
function [s] = vecsum(v)
  s = 0;
  for i = 1 : length(v)
    s = s + v(i);
  endfor
endfunction

x = [ 1 2 23 22 ]

xs = vecsum(x)
```

Project "The development of the didactic potential of Cracow University of Technology in the range of modern construction" is co-financed by the European Union within the confines of the European Social Fund and realized under surveillance of Ministry of Science and Higher Education

Searching for vector maximum element

```
function [mx] = vecmax(x)
  mx = x(1);
  for i=2:length(x)
    if mx < x(i)
      mx = x(i);
    endif
  endfor
endfunction

z = [1,4,5,2,-1,28,3]

mz = vecmax(z)
```

Project "The development of the didactic potential of Cracow University of Technology in the range of modern construction" is co-financed by the European Union within the confines of the European Social Fund and realized under surveillance of Ministry of Science and Higher Education

Reversing order of vector elements

```
v = [1, 3, 4, 24]
n = length(v);
for i=1:n/2
  swp = v(i);
  v(i) = v(n-i+1);
  v(n-i+1) = swp;
endfor

v
```

Project "The development of the didactic potential of Cracow University of Technology in the range of modern construction" is co-financed by the European Union within the confines of the European Social Fund and realized under surveillance of Ministry of Science and Higher Education

Reversing order of vector elements

```
v = [1,3,4,24]

v(end:-1,1) = v
```

Project "The development of the didactic potential of Cracow University of Technology in the range of modern construction" is co-financed by the European Union within the confines of the European Social Fund and realized under surveillance of Ministry of Science and Higher Education

Recursive algorithms – calculating factorial

```
function [s] = vecsum(v)
  s = 0;
  for i = 1 : length(v)
    s = s + v(i);
  endfor
endfunction

x = [ 1 2 23 22 ]

xs = vecsum(x)
```

Project "The development of the didactic potential of Cracow University of Technology in the range of modern construction" is co-financed by the European Union within the confines of the European Social Fund and realized under surveillance of Ministry of Science and Higher Education

Fibonacci sequence

$$F_n = F_{n-1} + F_{n-2},$$
$$F_0 = 0 \quad \text{and} \quad F_1 = 1.$$

```
function v = fib(n)
  if n < 3
    v = 1;
  else
    v = fib(n-2) + fib(n-1);
  endif
endfunction

for i=1:10
  fib(i)
endfor
```

Project "The development of the didactic potential of Cracow University of Technology in the range of modern construction" is co-financed by the European Union within the confines of the European Social Fund and realized under surveillance of Ministry of Science and Higher Education

Chebyshev polynomials of the first kind

The Chebyshev polynomials of the first kind are defined by the recurrence relation

$$T_0(x) = 1 \tag{1}$$

$$T_1(x) = x \tag{2}$$

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x). \tag{3}$$

Project "The development of the didactic potential of Cracow University of Technology in the range of modern construction" is co-financed by the European Union within the confines of the European Social Fund and realized under surveillance of Ministry of Science and Higher Education

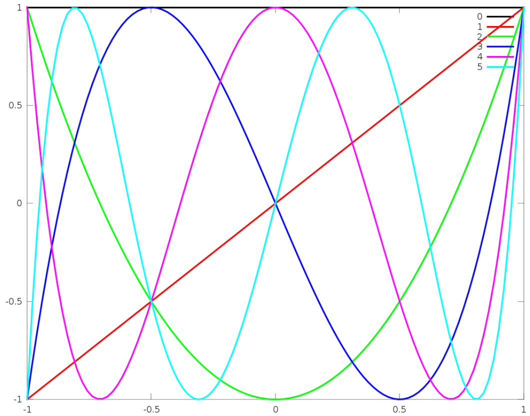
chebyschev polynomials of the first kind

```
function y = chebychev(n, x)
   if n == 0
     y = 1;
   elseif n == 1
     y = x;
   else
     y = 2*x*chebychev(n-1,x)-chebychev(n-2,x);
   endif
endfunction

x = linspace(-1,1);
for n = 0:5
  y = []
  for xk = x
  y = [y, chebychev(n,xk)];
  end
  hold on;
  ft = sprintf("%d;%d;",n,n);
  plot(x,y, ft, "linewidth", 3);
end
pause();
print ("chebyshev.png")
```

Project "The development of the didactic potential of Cracow University of Technology in the range of modern construction" is co-financed by the European Union within the confines of the European Social Fund and realized under surveillance of Ministry of Science and Higher Education

Chebyshev polynomial of the first kind



Project "The development of the didactic potential of Cracow University of Technology in the range of modern construction" is co-financed by the European Union
within the confines of the European Social Fund and realized under surveillance of Ministry of Science and Higher Education

Sorting vector – bubble sort algorithm

```
v = [1,2,4,23,2,23,2,4,56];

n = length(v);
for i=2:n
  for j=n:-1:i+1
    if v(j) < v(j-1)
      v([i,j]) = v([j,i]);
    endif
  endfor
endfor

v
```

Project "The development of the didactic potential of Cracow University of Technology in the range of modern construction" is co-financed by the European Union within the confines of the European Social Fund and realized under surveillance of Ministry of Science and Higher Education

Swapping values

Solution 1: using extra variable:

```
octave:12> a = 22;
octave:13> b = 111;
octave:14> swp = a;
octave:15> a = b;
octave:16> b = swp;
octave:17> a,b
a = 111
b = 22
```

Solution 1: using `swap()` function from `control` package:

```
octave:21> a = 12;
octave:22> b = 111;
octave:23> [a,b] = swap(a,b);
octave:24> a,b
a = 111
b = 12
```

Project "The development of the didactic potential of Cracow University of Technology in the range of modern construction" is co-financed by the European Union
within the confines of the European Social Fund and realized under surveillance of Ministry of Science and Higher Education

Rearranging vectors

Exchanging value of two elements:

```
octave:25> v = [10, 1, 888];
octave:26> v([1,3]) = v([3,1])
v =
   888  1  10
```

Reversing order of vector elements

```
octave:33> v = [1,2,3,4]
v =
   1  2  3  4
octave:34> v = v([end:-1:1])
v =
   4  3  2  1
```

Project "The development of the didactic potential of Cracow University of Technology in the range of modern construction" is co-financed by the European Union
within the confines of the European Social Fund and realized under surveillance of Ministry of Science and Higher Education

Creating lists

```
octave:40> l = list(1,2,3,4);
octave:41> append(l, 22,44)
ans =
(
  [1] = 1
  [2] = 2
  [3] = 3
  [4] = 4
  [5] = 22
  [6] = 44
)
```

The use of list objects is deprecated and instead cell arrays should be used.

Project "The development of the didactic potential of Cracow University of Technology in the range of modern construction" is co-financed by the European Union within the confines of the European Social Fund and realized under surveillance of Ministry of Science and Higher Education

Simulating lists with cell arrays

```
octave:45> lst = {1,2,3};
octave:46> lst = [lst, {111,222}];
octave:47> lst = [{0},lst];
octave:48> lst
lst =
{
  [1,1] = 0
  [1,2] = 1
  [1,3] = 2
  [1,4] = 3
  [1,5] = 111
  [1,6] = 222
}
```

The lists created as cell arrays can hold elements of any type, however some functions do not work on them, e.g. sort, min, max.

Project "The development of the didactic potential of Cracow University of Technology in the range of modern construction" is co-financed by the European Union within the confines of the European Social Fund and realized under surveillance of Ministry of Science and Higher Education

Simulating lists with vectors

```
octave:60> v = [1,2,3];
octave:61> v = [v, 100];
octave:62> v = [333, v];
octave:63> v
v =
   333 1 2 3 100
octave:64> at = 3
at = 3
octave:65> v = [v(1:at-1), 777, v(at:end)]
v =
   333 1 777 2 3 100
```

Project "The development of the didactic potential of Cracow University of Technology in the range of modern construction" is co-financed by the European Union within the confines of the European Social Fund and realized under surveillance of Ministry of Science and Higher Education

Sorting vectors

```
octave:66> v
v =
   333 1 777 2 3 100
octave:67> v = sort(v)
v =
     1 2 3 100 333 777
octave:69> v = sort(v, "descend")
v =
   777 333 100 3 2 1
```

Project "The development of the didactic potential of Cracow University of Technology in the range of modern construction" is co-financed by the European Union within the confines of the European Social Fund and realized under surveillance of Ministry of Science and Higher Education

Finding vector's extreme values

```
octave:70> v
v =
   777 333 100 3 2 1

octave:71> max(v)
ans = 777
octave:72> min(v)
ans = 1
```

Project "The development of the didactic potential of Cracow University of Technology in the range of modern construction" is co-financed by the European Union within the confines of the European Social Fund and realized under surveillance of Ministry of Science and Higher Education

Looking-up vector's values

```
octave:56> v = [10,22,300];
octave:57> lookup(v,22)
ans = 2
octave:58> lookup(v,[22,10])
ans =
   2 1
```

CAUTION: The vector should be strictly monotonic.

Project "The development of the didactic potential of Cracow University of Technology in the range of modern construction" is co-financed by the European Union within the confines of the European Social Fund and realized under surveillance of Ministry of Science and Higher Education

Calculating sums and products

$$sum = \sum_{i=1}^{N} v_i = v_1 + v_2 + \ldots + v_N$$

$$prod = \prod_{i=1}^{N} v_i = v_1 \cdot v_2 \cdot \ldots \cdot v_N$$

```
octave:76> v = [1,2,3,4]
v =
   1 2 3 4
octave:77> sum(v)
ans = 10
octave:78> prod(v)
ans = 24
```

Project "The development of the didactic potential of Cracow University of Technology in the range of modern construction" is co-financed by the European Union within the confines of the European Social Fund and realized under surveillance of Ministry of Science and Higher Education

Simulating associative arrays

```
octave:79> stud.name = "Jan";
octave:80> stud.surname = "Kowalski";
octave:81> stud.id = 978272;
octave:82> stud.note = 5.0;
stud =
{
  name = Jan
  surname = Kowalski
  id = 978272
  note = 5
}
octave:83> getfield(stud, "name")
ans = Jan
```

Project "The development of the didactic potential of Cracow University of Technology in the range of modern construction" is co-financed by the European Union within the confines of the European Social Fund and realized under surveillance of Ministry of Science and Higher Education

Looping over structure fields

```
for [key, val] = stud
  key,val
endfor

key = Jan
val = name
key = Kowalski
val = surname
key = 978272
val = id
key = 5
val = note
```

Project "The development of the didactic potential of Cracow University of Technology in the range of modern construction" is co-financed by the European Union within the confines of the European Social Fund and realized under surveillance of Ministry of Science and Higher Education

More complex data structures

In the absence of pointers more complex data structures like trees and graphs can be implemented either by composing structures or by using cell arrays. The alternative is to code those structures in C++ and expose it to Octave as extension packages.

Project "The development of the didactic potential of Cracow University of Technology in the range of modern construction" is co-financed by the European Union within the confines of the European Social Fund and realized under surveillance of Ministry of Science and Higher Education