Information technology – lecture 5

Major generic kinds of statements in imperative languages.

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

Imperative programming

- Computation is described in terms of statements that change the program state (the state is expressed via variables)
- Characterised (partially) by the presence of variables and direct assignment statements.

Structured programming

Structured programming is subdiscipline of imperative programming in which program actions are organised into procedures (subroutines, functions) and the program logic is mapped into the procedures calls. Opposite to structured programming is object-oriented programming.

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

Expressions versus statements

- Expressions – The basic building block of statements. An expression evaluates to a value. Expression can serve as a statement on its own.
  - variables, array references, constants, function calls combined with various operators
- Statements – the smallest standalone element of imperative language. Statements do not return results and are used for their side effects;
  - for-loop, conditional statements, etc.

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

Simple statements

- assignment:
  `A= A + 1;`
- call:
  `sin(pi/2);`
- return:
  `return`
- assertion:
  `assert(i!=0, "Variable i cannot be zero");`

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

Compound statements

- if-statement:
  if i $>$ 0 disp("positive"); else disp("not positive"); endif
- switch-statement:
  switch i case 1 disp("Big"); case 2 disp("Small") endswitch
- while-loop:
  while i<10 disp(i); i++; endwhile
- do-until loop:
  do computation(i); until i $<$ 10
- for-loop:
  for i=1:10 disp(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

Octave built-in data types

- real and complex scalars
  1 1.0 1.4e−1 1+2i 1+1j CAUTION: in fact they are matrices of the size 1x1
- real and complex matrices
  [1,2] [1,2;3,1]
- ranges
  1:5 1:0.5:5 −10:1
- character strings
  "Humpty Dumpty sat on a wal"
- data structures
  pA.x=1, pA.y=2
- cell arrays
  {1.0, "Humpty", [1,2,3,4];}

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

Assignment expressions

var = expression

`var` is variable name, element of an array, list of return values. `expression` is any Octave expression.

```
1  v = 1.0
2  z = 1.0 + v
3  A(1) = sin(2.0*pi)
4  [a,b,c] = compute_three_values()
```

The assignment expression produces a value equal to the value of assigned expression:

```
b = 5 + (a=2)
```

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

Arithmetic expressions

+	addition	x + y
-	subtraction	x - y
*	matrix multiplication	x * y
/	division	x
\	left division	inverse(x) * y
^	power operator	x^y
**	power operator	x ** y
-	negation	-x
+	unary plus	+x
'	transpose	x'

There are also versions prefixed with . (dot) (`x .* y`) – their meaning is to perform an operation element by element.

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

Comparison operators

`x < y`	True if `x` is less than `y`.
`x <= y`	True if `x` is less than or equal to `y`.
`x == y`	True if `x` is equal to `y`.
`x >= y`	True if `x` is greater than or equal to `y`.
`x > y`	True if `x` is greater than `y`.
`x != y`	True if `x` is not equal `y`.
`x ~= y`	

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

Boolean expressions

Element by element Boolean operators

`boolean1 & boolean2`	True if both operands are true
`boolean1 | boolean2`	True if either operand is true
`!boolean`	True if the operand is false
`~boolean`	

Short-circuit Boolean operators

```
boolean1 && boolean2
boolean1 || boolean2
```

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

Index expressions

Index expression is used to reference or extract selected elements of matrix or vector.

```
octave:17> A=[8,9,10,11]
A =
     8 9 10 11
octave:18> A(1:2)
ans =
    8 9
octave:19> A(3)
ans = 10
octave:20> B = [8,9;10,11]
B =
     8 9
    10 11
octave:21> B(:,2)
ans =
     9
    11
```

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 **if** statement

```
1  if (condition)
2      then-body
3  else
4      else-body
5  endif
```

```
1  if (condition)
2      then-body
3  else
4      else-body
5  endif
```

```
1  if (condition)
2      then-body
3  elseif (condition)
4      elseif-body
5  else
6      else-body
7  endif
```

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 **switch** statement

```
1  switch expression
2     case label
3        command_list
4     case label
5        command_list
6     ...
7
8     otherwise
9        command_list
10 endswitch
```

```
1  switch (X)
2     case 1
3        do_something ();
4     case 2
5        do_something_else ();
6     otherwise
7        do_something_different ();
8  endswitch
```

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 **for** statement

```
1  for var = expression
2      body
3   endfor
```

```
1  for i = 1:10
2     disp(i);
3  endfor
4
5  for i = [1,23,22,1]
6     disp(i);
7  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

The **while** statement

```
1 while (condition)
2     body
3 endwhile
```

```
1 i=1;
2 while i<= 10
3     disp(i);
4     i++;
5 endwhile
```

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 **do-until** statement

```
1  do
2     body
3  until (condition)
```

```
1  i=1;
2  do
3     disp(i);
4     i++;
5  until i>10
```

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 **break** statement

The break statement jumps out of the innermost for or while loop that encloses it.

```
1 for i=1:10
2   if i > 5
3     break;
4   endif
5   disp(i);
6 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

The **continue** statement

The **continue** statement skips over the rest of the loop body, causing the next cycle around the loop to begin immediately.

```
1 for i=1:10
2   if i == 5
3     continue;
4   endif
5   disp(i);
6 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

Defining functions

```
function ret-var = name (arg-list)
   body
endfunction
```

```
function [x,y] = polar2cartesian(r,fi)
    x = r * cos(fi);
    y = r * sin(fi);
endfunction
```

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

Calling functions

```
1       [x,y] = polar2cartesian(1, pi/4);
```

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

Call by value

From Octave documentation: "In Octave, unlike Fortran, function arguments are passed by value, which means that each argument in a function call is evaluated and assigned to a temporary location in memory before being passed to the function. There is currently no way to specify that a function parameter should be passed by reference instead of by value. This means that it is impossible to directly alter the value of a function parameter in the calling function"

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

Arrays

```
octave:5> A = [88,1;23,2]
A =

   88  1
   23  2

octave:6> det(A)
ans = 153
octave:7> A * [1;1]
ans =

   89
   25
```

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

Data structures

Data structures can be used to organize object of different types.

```
1    shape.center = [1,2];
2    shape.dofill = 1;
3    shape.colorname = "red"
4    shape.vertices = [1,2,23,7,32];
5
6    draw_shape(shape);
```

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

Cell arrays

Cell arrays allow to create arrays of objects of different types (in particular arrays of arrays).

```
octave:9> D = {1, [23,44], "red"}
D =
{
  [1,1] = 1
  [1,2] =

     23 44

  [1,3] = red
}
```

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

Thank you for your attention

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