Smalltalk 2.

Készítette: Szabó Éva

Blokkok

- **Paraméter nélküli blokk**
  - [műveletek] → *[x := 5. 'Hello' print. 2+3]*
  - Kiértékelés: *[művelet]* ***value*** – az értéke az utolsó művelet értéke lesz, de mindet kiírja.

    x := [*művelet****.]***

    x value.
- **Paraméteres blokk**
  - *x := [:a :b :c | (a+b+c) printNl]*
  - Kiértékelés:

    x value:2 value:3 value:4 → 9-et fog kiírni

Vezérlési szerkezetek 1.

- **feltételes (if)**
  - Példák:

    |x| *x := 3.*

    x > 0

        ***ifTrue**: ['x pozitív' printNl]*

        ifFalse**: ['x negatív' printNl]**!
- **kezdőfeltételes ismétléses (while)**
  - Példák:

    |x| *x := 2.*

    *[x < 20] **whileTrue**: [x printNl. x := x + 2]**!***

Vezérlési szerkezetek 2.

- **egyszerű számlálásos ismétléses (loop)**
  - Példák:

    3 **timesRepeat**: ['Hello' printNl]**!**
- **hagyományos számlálásos ismétléses (for)**
  - Példák:

    1 **to:** 5 **by:** 3 **do**: [:x | x printNl]**!**
  - csökkenő számlálásos:

    5 to: 1 by: **-1** do: [:i | i printNl]!

Objektumok összehasonlítása

- **x = y** "igaz, ha x és y egyenlő, tehát a tartalmuk azonos"
- **x ~= y** "igaz, ha x és y nem egyenlő, tehát a tartalmuk különbözik"
- **x == y** "igaz, ha x és y azonos, tehát ugyanaz az objektum"
- **x ~~ y** "igaz, ha x és y különböző, tehát nem ugyanaz az objektum"
- **x *isNil*** – ugyanaz, mintha azt írnánk, hogy x == nil.
- **x *notNil* –** x ~~ y

- Példa:

  Smalltalk at: #x put: 'aaaa' !

  Smalltalk at: #y put: 'aaaa' !

  (x=y) printNl ! ”true”, mert az x és y változók értéke egyenlő.

  (x==y) printNl ! "false", mert az x és y különböző objektumok.

Feladatok:

1. Írassuk ki 1-től N-ig az egész számokat!

2. Adjuk össze az első N egész számot!

3. Másoljuk át egy fájl tartalmát egy másikba úgy, hogy valamennyi karaktert nagybetűsítünk közben!

1. Írassuk ki 1-től N-ig az egész számokat! (st_1.st)

|n| n := 10.

1 to: n do: [:i | i printNl]!

vagy:

i:=1. [:a | a timesRepeat: [i printNl. i := i+1.]] value: 10.

2. Adjuk össze az első N egész számot! (st_2.st és st_2_alt.st)

```
| n  szum | n := 10.
szum :=0.
1 to: n do: [:i | szum := szum + i ].
szum printNl !
```

Vagy:

```
|n szum| n:= 10.
szum:= (n+1)*(n/2).
szum printNl!
```

3. Másoljuk át egy fájl tartalmát egy másikba úgy, hogy valamennyi karaktert nagybetűsítünk közben! (st_8.st)

```
input := FileStream open: 'file.txt' mode: FileStream read.
output :=  FileStream open: 'file2.txt' mode: FileStream write.
input do:
  [ :char |
                output nextPut: (char asUppercase).
  ].
input close.
output close.
```

- Ahhoz, hogy működjön, először hozzatok létre egy file.txt fájlt!

Kollekciók

- Tömb - Array
- Halmaz - Set
- „Táska” - Bag
- Szótár - Dictionary

Tömb (Array)

- 1-től indexeljük őket
- Tömbliterál: ***#(1 $B 'string')***
- Tömbliterálnál nincs különbség print és display között.
- Új tömb létrehozása:

  tömb := Array new: 10.
- ***at: index*** - indexedik elem elérése
- ***at: index put: obj*** – berakjuk *obj* objektumot a tömb indexedik helyére.
- ***size*** - a tömb méretét adja vissza

példa

```
Smalltalk at: #x put: 0 !
x := Array new: 20 !        "20 elemu vektor"
x at: 1 put: 99 !        "megadott helyre adat beirasa"
(x at: 1) printNl !        " adott elem kiiratasa"
"x at: 21 hiba, a vektorunk csak 20 elemu...."
((x at: 1) + 1) printNl !
x at: 2 put: 10 !
(x at:1) + (x at:2) printNl!
x printNl !          "(99 10 nil .... nil)"
```

Halmaz (Set)

- Ismétlés nélküli, rendezetlen.
- Új halmaz:

  Set new.

- 1 elemhozzáadása:

  add:

- Egy kollekció összes elemének hozzáadása:

  add All: **kollekció**

- 2 halmaz uniója:
  - halmaz1 **+** halmaz2

pl.:
x:= Set new.
1 to: 100 do [:i | x add: i].
x printNl.

Bag

- Ismétléses, rendezetlen.
- ***Bag new.***
- ***add:*** obj withOccurrences: 3. ”3-szor adja hozzá”
- pl.:

  Smalltalk at: #x put: 'aaaa'.

  x := Bag new.

  x add: 'aaaa' withOccurrences: 3.

  x printNl !

Szótár(Dictionary)

- Asszociatív tömb, Java-ban: Map.

```
Smalltalk at: #x put: 0 !
x := Dictionary new.
x at: 'One' put: 1 .
x at: 'Two' put: 2 .
x at: 1 put: 'One' .
x at: 2 put: 'Two' .
x printNl .
(x at: 1) printNl .
(x at: 'Two') printNl .
```

Műveletek kollekciókkal

- **Collect**: minden elemre végrehajtja a műveletet

  *(#(1 3 5) **collect:** [:x | x + 2]) printNl!*

- **Select**: válogatás, pl.: páratlan számokat (odd)

  *(#(1 2 3 4 5) **select:** [:x | x odd]) printNl!*

- **Do**: „okosabb for-ciklus” ~C#-ban a foreach

  |sum| sum := 0.

  *#(1 2 3) **do:** [:x | sum := sum + x].*

  sum printNl.

Feladatok:

1. Határozzuk meg egy számokból álló tömb minimális elemét!

2. Írassuk ki egy számokból álló tömb páratlan értékű elemeit!

3. Fordítsuk meg egy tömb tartalmát!

4. Sorfolytonosan írassuk ki egy kétdimenziós tömb elemeit!

1. Határozzuk meg egy számokból álló tömb minimális elemét! (st_3.st és st_3_alt.st)

```
|a| a := #(10 2 3 4 5 3 5 3 4).

|min| min := 100000.

1 to: a size do: [ :m | min > (a at: m)
                ifTrue: [min := (a at: m).]].

'MIN' print. min printNl .
```

vagy:

```
|min| min := 1000.

#(10 2 3 4 5 3 5 3 4) do: [:x | x < min ifTrue:
[min:=x]].

min printNl.
```

2. Írassuk ki egy számokból álló tömb páratlan értékű elemeit! (st_4.st)

```
(#(1 2 3 4 5 6 7 8 9 10) select: [:x | x odd])
printNl.
```

3. Fordítsuk meg egy tömb tartalmát! (st_5.st és st_5_alt.st)

a:=#(1 2 3 4 5 6 7 8 9 10). a ***reverse*** *printNl.*

vagy:

```
|a c| a:=#(1 2 3 4 5 6 7 8 9 10).
c:=a shallowCopy.
for 1 to: (c size/2-1) do:
   [:i |
      b:= (c at:i).
      c  at:i put: (c at: (c size-i+1)).
      c at: (c size-i+1) put:b.
   ].
c printNl.
```

4. Sorfolytonosan írassuk ki egy kétdimenziós tömb elemeit! (st_6.st)

```
|tomb| tomb:=#(#(56 6) #(4 5 5 7) #(3 5 7)).
tomb do: [:x |
  x do: [:i | i print. Transcript show: ' '.].
].
```