A tárgy f?oldala: info2/2008tavasz A tárgy oktatásához felhasznált, http://wiki.math.bme.hu/ -n belüli wikioldalak GNU FDL licenc vagy (választás szerint) CC-BY-SA-2.0 licenc szerint szabadon használhatók és terjeszthet?k.

Tartalomjegyzék

- 1 8. el?adás (2007-04-04)
  - 1.1 Objektum-orientált programozás
  - 1.2 Ruby nyelv
  - 1.3 Feltételes utasítások
    - 1.3.1 if
    - 1.3.2 else
    - 1.3.3 elsif
    - 1.3.4 if kifejezésben
    - 1.3.5 unless
    - 1.3.6 case
    - 1.3.7 ?:
  - 1.4 Ciklus utasítások
    - 1.4.1 while, until
    - 1.4.2 while, until kifejezésekben
    - 1.4.3 for-in ciklus
    - 1.4.4 Iterátorok
    - 1.4.5 break, next
    - 1.4.6 Megszámlálható objektumok
  - 1.5 Blokkok
    - 1.5.1 redo/retry
    - 1.5.2 catch/throw
  - 1.6 Példányváltozó -- setter
  - 1.7 Néhány példaprogram

8. el?adás (2007-04-04)

Objektum-orientált programozás

Néhány új fogalom:

- osztály (a C-beli struktúratípusnak felel meg)
- objektum (a C-beli struktúrának felel meg)
- attribútum (a C-beli struktúramez?nek felel meg)
- metódus (a C-beli függvénynek felel meg)

Osztály *class* (síkidom), osztálypéldány *class instance* = objektum *object* (kör, téglalap), konstruktor *constructor* (új objektum létrehozása), objektum azonosító (object identifier - object ID), példány változó *instance variable* (mely az objektum állapotát/tulajdonságait tartalmazza, pl. középpont), példány metódus *instance method* (),

Ruby nyelv

A Ruby **interpretált**, **szkriptnyelv** (interpretált, és minden rendszer szint? szolgáltatáshoz hozzá lehet férni).

A Ruby osztályhierarchiája hasonló a biológiai törzsfához. Például a gerincesek (osztály) törzsén belül a madarak (osztály) osztályának egy alosztálya a pacsirta (osztály) nev? faj, melynek egy példánya a Csipcsip nev? kismadarunk (objektum), aki nem mellesleg egy pacsirta. ?t jellemezhetjük különböz? tulajdonságai alapján; ezek az attribútumok. Például: él-e még? , mennyire éhes, hogy hívják a párját, stb. Tehát bizonyos tulajdonságait megadjuk, amik csak rá jellemz?ek.

Változók				Konstansok és
Lokális	Globális	Példány	Osztály	osztály nevek
valtozo	$valami	@nev	@@osztvalt	PI
joEjt_2	$_	@XY	@@N	String

Feltételes utasítások

if

El?ször then-nel, majd újsorral elválasztva:

```
if x==2 then x += 1 end

if x==2
  x += 1
end

if x==2; x += 1 end
```

else

```
if x==2 then x += 1 else x += 2 end

if x==2
  x += 1
else
  x += 2
end
```

elsif

```
if x == 1
  "http://wiki.math.bme.huhetfo"http://wiki.math.bme.hu
elsif x == 2
  "http://wiki.math.bme.hukedd"http://wiki.math.bme.hu
elsif x == 3 then "http://wiki.math.bme.huszerda"http://wiki.math.bme.hu
else "http://wiki.math.bme.huhetvege"http://wiki.math.bme.hu
end
```

if kifejezésben

```
x = if y == 1
      3
    else
      4
    end

x = 4
x = 3 if y == 1
```

unless

```
unless x <= 4
  "http://wiki.math.bme.hunagyobb, mint 4"http://wiki.math.bme.hu
end

unless x <= 4
  "http://wiki.math.bme.hunagyobb, mint 4"http://wiki.math.bme.hu
else
  "http://wiki.math.bme.hunem nagyobb"http://wiki.math.bme.hu
end

x = unless x <= 4 then 5 end

x = 5 unless x <= 4
```

case

A fenti if-es példával ekvivalens a következ?vel (a sorvége helyett itt is lehet then vagy pontosvessz?):

```
case x
when 1
  "http://wiki.math.bme.huhetfo"http://wiki.math.bme.hu
when 2
  "http://wiki.math.bme.hukedd"http://wiki.math.bme.hu
when 3; "http://wiki.math.bme.huszerda"http://wiki.math.bme.hu
else "http://wiki.math.bme.huhetvege"http://wiki.math.bme.hu
end
```

Ekkor valójában a következ? hajtódik végre (figyeljük meg a case-egyenl?ség használatát):

```
case
when 1 === x then "http://wiki.math.bme.huhetfo"http://wiki.math.bme.hu
when 2 === x then "http://wiki.math.bme.hukedd"http://wiki.math.bme.hu
when 3 === x then "http://wiki.math.bme.huszerda"http://wiki.math.bme.hu
else "http://wiki.math.bme.huhetvege"http://wiki.math.bme.hu
end
```

Az === néha megegyezik ==-vel, pl. a Fixnum osztályban, más osztályokban külön definiálva van a case számára:

tartomány használata: itt 'a === b' azt jelenti, hogy 'b' benne van-e az 'a' tartományban:

```
case x
when 1..5
  "http://wiki.math.bme.huhetkoznap"http://wiki.math.bme.hu
when 6..7
  "http://wiki.math.bme.huhetvege"http://wiki.math.bme.hu
else "http://wiki.math.bme.hurossz adat"http://wiki.math.bme.hu
end
```

reguláris kifejezés használata: itt 'a === b' azt jelenti, hogy 'b' illeszkedik-e az 'a' mintára:

```
x = "http://wiki.math.bme.huab"http://wiki.math.bme.hu
case x
when /[a-z]+/
  "http://wiki.math.bme.huszoveg"http://wiki.math.bme.hu
when /[0-9]+/
  "http://wiki.math.bme.huszam"http://wiki.math.bme.hu
end
=> "http://wiki.math.bme.huszoveg"http://wiki.math.bme.hu
```

egy összetettebb példa (beolvassa és kiírja a sorokat addig, míg egy 'ennyi' tartalmút nem kap, a megjegyzés sorokat kihagyja):

```
while line=gets.chomp do
  case line
  when /^\s*#/
    next
  when /^ennyi$/i    # i = case insensitive
    break
  else
    puts line
  end
end
```

osztályhoz tartozás: a Class osztályban 'a === b' azt jelenti, hogy a 'b' az 'a' osztály példánya-e:

```
case x
when Numeric then "http://wiki.math.bme.huszam"http://wiki.math.bme.hu
when String then "http://wiki.math.bme.husztring"http://wiki.math.bme.hu
else "http://wiki.math.bme.huvalami mas"http://wiki.math.bme.hu
end
```

A Ruby case utasításában nincs ráfutás, így nem is kell minden ágat lezárni, mint C-beli switch utasításban break-kel vagy return-nel!!!

?:

Ha x == 3, y legyen 4, egyébként 5:

```
x = y == 3 ? 4 : 5
```

Ciklus utasítások

while, until

```
>> x=0
=> 0
>> while x<4 do
?>   x += 1
>>   puts x
>> end
1
2
3
4
=> nil

>> x=0
=> 0
>> until x>=4 do
?>   x += 1
>>   puts x
>> end
1
2
3
4
=> nil
```

while, until kifejezésekben

```
>> x = 0; x += 1 while x<4; x
=> 4

>> x=0; x += 1 until x==4; x
=> 4
```

for-in ciklus

Egy tömbre és egy hash-táblára:

```
>> a = [1,2,3]
=> [1, 2, 3]
>> for i in a
>>   p i
>> end
1
2
3
=> [1, 2, 3]

>> h = {"http://wiki.math.bme.huhetfo"http://wiki.math.bme.hu=>1, "http://wiki.math.bme.hukedd"htt
=> {"http://wiki.math.bme.huhetfo"http://wiki.math.bme.hu=>1, "http://wiki.math.bme.huvasarnap"htt
>> for kulcs, ertek in h
>>   puts "http://wiki.math.bme.hu#{kulcs} a(z) #{ertek}. nap"http://wiki.math.bme.hu
>> end
hetfo a(z) 1. nap
vasarnap a(z) 7. nap
kedd a(z) 2. nap
=> {"http://wiki.math.bme.huhetfo"http://wiki.math.bme.hu=>1, "http://wiki.math.bme.huvasarnap"htt
```

Iterátorok

A times iterátor:

```
>> print "http://wiki.math.bme.huMondja! Maga "http://wiki.math.bme.hu; 3.times {print "http://wik
Mondja! Maga mindent ketszer mond? mindent ketszer mond? mindent ketszer mond?
=> nil

>> a = "http://wiki.math.bme.huMondja! Maga "http://wiki.math.bme.hu
=> "http://wiki.math.bme.huMondja! Maga "http://wiki.math.bme.hu
>> 3.times {a << "http://wiki.math.bme.humindent ketszer mond? "http://wiki.math.bme.hu}
=> 3
>> a
=> "http://wiki.math.bme.huMondja! Maga mindent ketszer mond? mindent ketszer mond? mindent ketsze
```

Ciklusváltozó használatával:

```
>> 5.times {|i| print i}; print "http://wiki.math.bme.hu\n"http://wiki.math.bme.hu
01234
=> nil
```

Az upto, downto iterátor:

```
>> 0.upto(4) {|i| print i}; print "http://wiki.math.bme.hu\n"http://wiki.math.bme.hu
01234
=> nil
>> 5.upto(8) {|i| print i}; print "http://wiki.math.bme.hu\n"http://wiki.math.bme.hu
5678
=> nil
>> 8.downto(5) {|i| print i}; print "http://wiki.math.bme.hu\n"http://wiki.math.bme.hu
8765
=> nil
```

break, next

```
while i<6
  if i == 4 then break end
  print i
  i += 1
end

0.upto(5) do |i|
  if i == 4 then break 17 end  # visszatérési érték is megadható, egyébként nil
  print i
  i += 1
end
```

Megszámlálható objektumok

Megszámlálható objektumok (*enumerable objects*). Amely osztályok objektumaira definiálva van az each metódus. Pl. Array, Hash, Range.

Az each, map és inject iterátorok használata:

```
>> [1,2,3].each {|i| p i}
1
2
3
=> [1, 2, 3]
>> [1,2,3].map {|i| i**2}
=> [1, 4, 9]
>> [1,2,3].inject {|sum, i| sum + i}
=> 6

>> (1..4).each {|i| print i}
1234=> 1..4

>> (1..4).map {|i| i**2}
=> [1, 4, 9, 16]

>> (1..4).inject {|s,i| s+i}
=> 10
```

Dobjuk fel 4-szer a kockát, és nézzük meg, melyik a legnagyobb dobás:

```
>> a = Array.new(4) {rand(6) + 1}
=> [3, 6, 6, 4]
>> a.inject {|m,i| m > i ? m : i}
=> 6
```

s?t!

```
(Array.new(4) {rand(6) + 1}).inject {|m,i| m > i ? m : i}
```

A következ? program egy fájlt kiír az std outputra:

```
File.open(filenev) do |f|
  f.each {|sor| print sor }
end
```

Az each_with_index használata (az el?z? példa sorszámozott sorokkal):

```
>> filenev = "http://wiki.math.bme.hurrrr"http://wiki.math.bme.hu
=> "http://wiki.math.bme.hurrrr"http://wiki.math.bme.hu
>> File.open(filenev) do |f|
?>   f.each_with_index do |sor,i|
?>     print "http://wiki.math.bme.hu#{i+1}: #{sor}"http://wiki.math.bme.hu
>>   end
>> end
1: Ez egy tesztfile
2: Ez a 2. sora
3: Ez meg az utso
=> #<File:rrrr (closed)>
```

Blokkok

Blokkok csak metódushívásokat követhetnek. Vagy kapcsos zárójelek közt vagy do és end közt adhatók meg.

```
>> h = {:piros => 0xff0000, :zold => 0x00ff00, :kek => 0x0000ff}
=> {:piros=>16711680, :zold=>65280, :kek=>255}
>> h.each {|kulcs, ertek| print "http://wiki.math.bme.hu%s kodja "http://wiki.math.bme.hu % kulcs,
```

```
piros kodja ff0000
zold kodja 00ff00
kek kodja 0000ff
=> {:piros=>16711680, :zold=>65280, :kek=>255}
```

redo/retry

A redo csak azt a ciklust ismétli

```
puts "http://wiki.math.bme.huMi jut eszedbe?"http://wiki.math.bme.hu
szavak = %w(alma piros tehen)
valasz = szavak.collect do |szo|
  print szo + "http://wiki.math.bme.hu> "http://wiki.math.bme.hu
  valasz = gets.chop
  if valasz.size == 0
    szo.upcase!
    redo
  end
  valasz
end
```

A retry az egész ciklust ismétli

```
n = 5
n.times do |x|
  print x
  if x == n-1
    n -= 1
    retry
  end
end
```

catch/throw

```
a=[[1,nil,3],[2,3,4],[0,2,5]]
catch :kiugrunk do
  0.upto 2 do |i|
    0.upto 2 do |j|
      throw :kiugrunk unless a[i][j]
      print a[i][j]
    end
    print "http://wiki.math.bme.hu\n"http://wiki.math.bme.hu
  end
end
print "http://wiki.math.bme.huvege\n"http://wiki.math.bme.hu
```

Példányváltozó -- setter

A példányváltozó beállítása az =-vég? metódussal. (Az el?adáson mutatott példában figyelmetlenségb?l nagy bet?vel lett írva a class parancs!) Helyesen:

```
class Nev
  attr_reader :vezetek, :kereszt

  def vezetek=(vezetek)
    if vezetek == nil or vezetek.size == 0
      raise ArgumentError.new('Mindenkinek van vezetekneve')
    end
    vezetek = vezetek.dup
    vezetek[0] = vezetek[0].chr.capitalize
```

```
    @vezetek = vezetek
  end

  def kereszt=(kereszt)
    if kereszt == nil or kereszt.size == 0
      raise ArgumentError.new('Mindenkinek van keresztneve')
    end
    kereszt = kereszt.dup
    kereszt[0] = kereszt[0].chr.capitalize
    @kereszt = kereszt
  end

  def teljes_nev
    "http://wiki.math.bme.hu#{@vezetek} #{@kereszt}"http://wiki.math.bme.hu
  end

  def initialize(vezetek, kereszt)
    self.vezetek=vezetek
    self.kereszt=kereszt
  end
end
```

Néhány példaprogram

Matmul Ruby nyelven:

```
a=[[1,2],                 #megadjuk az 'a' mátrixot
   [3,4],
   [5,6]]

b=[[7],                   #megadjuk a 'b' mátrixot
   [8]]

ab=[]                     #létrehozzuk az üres szorzatmátrixot
i=0; while i<a.size;      # while ciklust nem véletlenül használunk, mivel Rubyban nincs for ciklus
  ujsor=[]
  j=0; while j<b[0].size;
    x=0
    k=0; while k<b.size
      x+=a[i][k]*b[k][j]
      ujsor[j]=x
   end
 ab[i]=ujsor
end
```

Objektum-orientáltan kezdtünk programozni Ruby nyelven, a Sikidom, BBox és Kor osztályok kerültek fel a táblára.

```
class BBox
  attr_accessor :llx, :lly, :urx, :ury
end

class Sikidom
  def bbox()
    fail  # még nem tudjuk megírni, a Sikidom túl absztrakt
  end
  def kerulet()
    fail  # még nem tudjuk megírni, a Sikidom túl absztrakt
  end
  def terulet()
    fail  # még nem tudjuk megírni, a Sikidom túl absztrakt
  end
```

```
end

class Kor <Sikidom
  attr_accessor :cx, :cy, :r
  def bbox()
    b=BBox.new
    b.llx=@cx-@r; b.lly=@cy-@r
    b.urx=@cx+@r; b.ury=@cy+@r
    return b  # a return fölösleges
  end
  def kerulet()
    Math::PI*2*@r
  end
  def terulet()
    Math::PI*@r*@r
  end
end
```

Egy sudoku megoldás (Fischer Richárd megoldása alapján):

```
 # puts "http://wiki.math.bme.huAdd meg a filenevet! "http://wiki.math.bme.hu
 # filenev=gets.chomp!
 filenev = "http://wiki.math.bme.husudoku1"http://wiki.math.bme.hu
 a=Array.new(9) {[]}
 File.open(filenev) do |f|
   f.each_with_index do |sor,i|
     0.upto(8) {|j| a[i][j]= if sor[j]>48 then sor[j].chr.to_i else 0 end}
   end
 end

 vanuj=true
 while vanuj do
   vanuj=false
   0.upto(8) do |i|
     0.upto(8) do |j|
       if a[i][j] == 0
         d=(1..9).to_a
         0.upto(8) do |k|
           d.delete(a[i][k])
           d.delete(a[k][j])
         end
         0.upto(2) do |e|
           0.upto(2) do |f|
             d.delete(a[i/3*3+e][j/3*3+f])
           end
         end
         if d.size==1
           a[i][j]=d[0]
           vanuj=true
         end
       end
     end
   end
 end
 0.upto(8) do |i|
   0.upto(8) do |j|
     print a[i][j]
   end
   print "http://wiki.math.bme.hu\n"http://wiki.math.bme.hu
 end
```

Egy gráfokat kezel? osztály:

```
class Adjacencia < Array
  attr_reader :adj
  def initialize
    @adj = []
  end

  def [](x,y)
    x,y = y,x if x > y
    raise IndexError if x==y
    @adj[ (y*y-y)/2 + x ]
  end

  def []=(x,y,e)
    x,y = y,x if x > y
    raise IndexError if x==y
    @adj[ (y*y-y)/2 + x ] = e
  end
end

class Graf
  attr_reader :adj
  def initialize( *elek )
    @adj = Adjacencia.new
    @csucsok = 0
    for e in elek
      @adj[e[0],e[1]] = 1
      @csucsok = [@csucsok,e[0],e[1]].max
    end
  end

  def [](x,y)
    @adj[x,y]
  end

  def add x,y
    @adj[x,y]=1
    @csucsok = [@csucsok,x,y].max
  end

  def fok(x)
    (0..@csucsok).inject(0) {|s,i| x!=i && @adj[x,i] ? s+1 : s }
  end

  def each_csucs
    (0..@csucsok).each {|v| yield v}
  end

  def each_el
    for i in 0...@csucsok
      for j in i+1..@csucsok
        yield i,j if self[i,j]
      end
    end
  end

  def osszefuggo?
    c = @csucsok
    volt = []
    lesz = [c]
    for i in 0...@csucsok
      lesz << i if self[i,c]
      volt << i if self[i,c]
    end
    while !volt.empty?
```

```
      v = volt.shift
      self.each_el do |x,y|
        if x==v || y==v
          z = x==v ? y : x
          if !lesz.include? z
            lesz << z
            volt << z
          end
        end
      end
    end
    lesz.size <= @csucsok ? false : true
  end
  def euler_kor?
    return false if !osszefuggo?
    paratlan = 0
    each_csucs do |i|
      if fok(i) % 2 == 1
        paratlan += 1
      end
    end
    paratlan == 0
  end

  def euler_ut?
    return false if !osszefuggo?
    paratlan = 0
    each_csucs do |i|
      if fok(i) % 2 == 1
        paratlan += 1
      end
    end
    paratlan == 2
  end

  def euler?
    return false if !osszefuggo?
    paratlan = 0
    each_csucs do |i|
      if fok(i) % 2 == 1
        paratlan += 1
      end
    end
    paratlan <= 2
  end

  def show
    (0...@csucsok).each do |i|
      (0..@csucsok).each do |j|
        print i>=j ? "http://wiki.math.bme.hu "http://wiki.math.bme.hu : (@adj[i,j] ? @adj[i,j] :
      end
      print "http://wiki.math.bme.hu\n"http://wiki.math.bme.hu
    end
  end
end
```