Bevezetés a matematikába

Járai Antal

Ezek a programok csak szemléltetésre szolgálnak.

▶ 1. Halmazok

▼ 2. Természetes számok

```
> restart;
```

▼ 2.1. Peano-axiómák

▼ *2.1.1.Peano-axiómák.*

```
> inc:=x->x+1;0;inc(%);inc(%);inc(%);inc(%);dec:=x->x-1;4;dec
(%);dec(%);dec(%);
```

$$inc := x \to x+1$$
$$0$$
$$1$$
$$2$$
$$3$$
$$4$$
$$dec := x \to x-1$$
$$4$$
$$3$$
$$2$$
$$1 \tag{2.1.1.1}$$

▶ **2.1.2. Megjegyzés.*

▶ **2.1.3. Megjegyzés.*

▼ *2.1.4. Rekurziótétel.*

```
> n:=16;twopower:=1;for i to n do twopower:=twopower*2; od;
```

$$n := 16$$
$$twopower := 1$$
$$twopower := 2$$

$$twopower := 4$$
$$twopower := 8$$
$$twopower := 16$$
$$twopower := 32$$
$$twopower := 64$$
$$twopower := 128$$
$$twopower := 256$$
$$twopower := 512$$
$$twopower := 1024$$
$$twopower := 2048$$
$$twopower := 4096$$
$$twopower := 8192$$
$$twopower := 16384$$
$$twopower := 32768$$
$$twopower := 65536 \tag{2.1.4.1}$$

▶ 2.1.5. A természetes számok egyértelműsége.

▶ 2.1.6. A természetes számok létezése.

▼ 2.1.7. Karakterisztikus függvények.

```
> X:={a,b,c,d,e}; Y:={a,c,e};
  chi:=x-> if x in Y then 1 elif x in X then 0 else FAIL fi;
  chi(a); chi(b); chi(1);
```

$$X := \{a, b, c, e, d\}$$
$$Y := \{a, c, e\}$$
$$\chi := x \to \textbf{if } x \in Y \textbf{ then } 1 \textbf{ elif } x \in X \textbf{ then } 0 \textbf{ else } FAIL \textbf{ end if}$$
$$1$$
$$0$$
$$FAIL \tag{2.1.7.1}$$

▶ ->2.1.8. Feladat.

▶ 2.1.9. Feladat.

▶ ->2.1.10. Feladat.

▶ ->2.1.11. Feladat.

▶ ->2.1.12. Feladat.

▶ ->2.1.13. Feladat.

▶ 2.1.14. Feladat.

▶ *->2.1.15. Feladat.*
▶ *2.1.16. Feladat.*
▶ *2.1.17. Feladat.*
▶ *2.1.18. Feladat.*
▶ *->2.1.19. Feladat.*
▶ *2.1.20. Feladat.*
▶ *2.1.21. Feladat.*
▶ *2.1.22. Feladat.*
▶ *2.1.23. Feladat.*
▶ *2.1.24. Feladat.*
▶ *2.1.25. Feladat.*
▶ *2.1.26. Feladat.*
▶ *2.1.27. Feladat.*
▶ *2.1.28. Feladat.*
▶ **2.1.29. Feladat.*
▶ *2.1.30. Feladat.*
▶ *2.1.31. Feladat.*
▶ **2.1.32. Feladat.*
▶ **2.1.33. Feladat.*
▶ **2.1.34. Feladat.*
▶ *->2.1.35. Feladat.*
▶ *->2.1.36. Feladat.*
▶ *2.1.37. További feladatok.*

2.2. Műveletek természetes számokkal

2.2.1. Természetes számok összeadása.

```
> sm:=n->if n=0 then m else inc(sm(dec(n))) fi;
  m:=3;sm(6);sm(10);m:=5;sm(6);sm(10);
```

$$sm := n \rightarrow \textbf{if}\ n = 0\ \textbf{then}\ m\ \textbf{else}\ inc(sm(dec(n)))\ \textbf{end if}$$

$$m := 3$$

$$9$$

$$13$$

$$m := 5$$

$$11$$

(2.2.1.1)

$$15 \qquad (2.2.1.1)$$

2.2.2. Tétel.

2.2.3. Természetes számok szorzása.

```
> pm:=n->if n=0 then 0 else pm(dec(n))+m fi;
  m:=3;pm(6);pm(10);m:=5;pm(6);pm(10);
```

$$pm := n \to \textbf{if } n = 0 \textbf{ then } 0 \textbf{ else } pm(dec(n)) + m \textbf{ end if}$$

$$m := 3$$

$$18$$

$$30$$

$$m := 5$$

$$30$$

$$50 \qquad (2.2.3.1)$$

2.2.4. Tétel.

->2.2.5. Feladat.

->2.2.6. Feladat.

2.2.7. Félcsoport, csoport, kommutativitás.

```
> isgrupoid:=proc(G::set,f::procedure) local x,y;
  for x in G do for y in G do if not f(x,y) in G then return
  false fi;
  od; od; true; end;
```

$isgrupoid := \textbf{proc}(G::set, f::procedure)$ (2.2.7.1)

```
    local x, y;
    for x in G do
        for y in G do
            if not in(f(x, y), G) then
                return false
            end if
        end do
    end do;
    true
end proc
```

```
> rightneutrals:=proc(G::set,f::procedure) local x,y,s,S;
  if not isgrupoid(G,f) then return false fi; S:={};
  for x in G do s:=true; for y in G do
    if f(y,x)<>y then s:=false; break fi; od; if s then S:=S
  union {x} fi; od; S end;
```

```
G:={a,b,c};rightneutrals(G,(x,y)->x);
```

rightneutrals := **proc**(*G*::*set*, *f*::*procedure*)
    local *x*, *y*, *s*, *S*;
    if not *isgrupoid*(*G*, *f*) **then**
        return *false*
    end if;
    S := {};
    for *x* **in** *G* **do**
        s := *true*;
        for *y* **in** *G* **do**
            if *f*(*y*, *x*) <> *y* **then**
                s := *false*;
                break
            end if
        end do;
        if *s* **then**
            S := *union*(*S*, {*x*})
        end if
    end do;
    S
end proc

$$G := \{a, b, c\}$$

$$\{a, b, c\} \tag{2.2.7.2}$$

```
> leftneutrals:=proc(G::set,f::procedure) local x,y,s,S;
  if not isgrupoid(G,f) then return false fi; S:={};
  for x in G do s:=true; for y in G do
    if f(x,y)<>y then s:=false; break fi; od; if s then S:=S
  union {x} fi; od; S end;

  G:={a,b,c};leftneutrals(G,(x,y)->y);rightneutrals(G,(x,y)-
  >y);
```

leftneutrals := **proc**(*G*::*set*, *f*::*procedure*)
    local *x*, *y*, *s*, *S*;
    if not *isgrupoid*(*G*, *f*) **then**
        return *false*
    end if;
    S := {};
    for *x* **in** *G* **do**
        s := *true*;

for y **in** G **do**
if $f(x, y) <> y$ **then**
$s := false$;
break
end if
end do;
if s **then**
$S := union(S, \{x\})$
end if
end do;
S
end proc

$$G := \{a, b, c\}$$
$$\{a, b, c\}$$
$$\{\ \} \qquad (2.2.7.3)$$

```
> neutral:=proc(G::set,f::procedure) local x,y,s,S;
  if not isgrupoid(G,f) then return NULL fi;
  for x in G do s:=true; for y in G do
    if f(x,y)<>y or f(y,x)<>y then s:=false; break fi;
  od; if s then return x fi; od; NULL end;

  G:={a,b,c};neutral(G,(x,y)->y);neutral(G,(x,y)->y);

  neutral({0,1,2},(x,y)->irem(x+y,3));
```

$neutral :=$ **proc**$(G::set, f::procedure)$
local x, y, s, S;
if not $isgrupoid(G, f)$ **then**
return $NULL$
end if;
for x **in** G **do**
$s := true$;
for y **in** G **do**
if $f(x, y) <> y$ **or** $f(y, x) <> y$ **then**
$s := false$;
break
end if
end do;
if s **then**
return x
end if

```
    end do;
    NULL
end proc
```

$$G := \{a, b, c\}$$

$$0 \tag{2.2.7.4}$$

```
> issemigroup:=proc(G::set,f::procedure) local x,y,z;
  if not isgrupoid(G,f) then return false fi;
  for x in G do for y in G do for z in G do
    if f(x,f(y,z))<>f(f(x,y),z) then return false fi;
  od; od; od; true end;

  issemigroup({a,b,c},(x,y)->x);

  issemigroup({true,false},(x,y)-> x implies y);
```

```
issemigroup := proc(G::set, f::procedure)
    local x, y, z;
    if not isgrupoid(G, f) then
        return false
    end if;
    for x in G do
        for y in G do
            for z in G do
                if f(x, f(y, z)) <> f(f(x, y), z) then
                    return false
                end if
            end do
        end do
    end do;
    true
end proc
```

$$true$$

$$false \tag{2.2.7.5}$$

```
> isgroup:=proc(G::set,f::procedure) local x,y,n,i;
  if not isgrupoid(G,f) then return false fi;
  if not issemigroup(G,f) then return false fi;
  n:=neutral(G,f); if n=NULL then return false fi;
  for x in G do i:=false; for y in G do
    if f(x,y)=n and f(y,x)=n then i:=true; break fi;
  od; if i=false then return false fi; od; true; end;
```

```
isgroup({0,1,2},(x,y)->irem(x+y,3));
```

isgroup := **proc**(*G*::*set*, *f*::*procedure*)
    local *x*, *y*, *n*, *i*;
    if not *isgrupoid*(*G*, *f*) **then**
        return *false*
    end if;
    if not *issemigroup*(*G*, *f*) **then**
        return *false*
    end if;
    n := *neutral*(*G*, *f*);
    if *n* = *NULL* **then**
        return *false*
    end if;
    for *x* **in** *G* **do**
        i := *false*;
        for *y* **in** *G* **do**
            if *f*(*x*, *y*) = *n* **and** *f*(*y*,
            x) = *n* **then**
                i := *true*;
                break
            end if
        end do;
        if *i* = *false* **then**
            return *false*
        end if
    end do;
    true
end proc

true (2.2.7.6)

```
> iscommutative:=proc(G::set,f::procedure) local x,y;
  if not isgrupoid(G,f) then return false fi;
  for x in G do for y in G do
    if f(x,y)<>f(y,x) then return false fi;
  od; od; true; end:

  iscommutative({0,1,2},(x,y)->irem(x+y,3));
```

iscommutative := **proc**(*G*::*set*, *f*::*procedure*)
    local *x*, *y*;

```
    if not isgrupoid(G, f) then
        return false
    end if;
    for x in G do
        for y in G do
            if f(x, y) <> f(y, x) then
                return false
            end if
        end do
    end do;
    true
end proc
```

$$true \qquad (2.2.7.7)$$

```
> isabeliangroup:=proc(G::set,f::procedure)
  isgroup(G,f) and iscommutative(G,f) end;

  iscommutative({0,1,2},(x,y)->irem(x+y,3));
```

```
isabeliangroup := proc(G::set, f::procedure)
    isgroup(G,
    f) and iscommutative(G, f)
end proc
```

$$true \qquad (2.2.7.8)$$

2.2.8. Megjegyzés.

2.2.9. Példák.

```
> X:={a,b,c}; with(combinat,powerset);
  P:=powerset(X); isgroup(P,(x,y)->symmdiff(x,y));
```

$$X := \{a, b, c\}$$

$$[powerset]$$

$$P := \{\{\}, \{a, b, c\}, \{a\}, \{b\}, \{a, b\}, \{c\}, \{b, c\}, \{a, c\}\}$$

$$true \qquad (2.2.9.1)$$

2.2.10. Feladat.

2.2.11. Feladat.

2.2.12. Példák.

2.2.13. Példák.

```
> X:={true,false}; `&iff`:=(x,y)->(x implies y) and (y
  implies x);isabeliangroup(X,(x,y)->x &iff y);
```

$$X := \{false, true\}$$

$$\&iff := (x, y) \rightarrow (x \Rightarrow y) \textbf{ and } (y \Rightarrow x)$$

$$true \tag{2.2.13.1}$$

->2.2.14. Feladat.

->2.2.15. Feladat.

->2.2.16. Feladat.

->2.2.17. Feladat.

->2.2.18. Feladat.

->2.2.19. Feladat.

->2.2.20. Feladat.

->2.2.21. Feladat.

->2.2.22. Feladat.

->2.2.23. Feladat.

2.2.24. Feladat.

2.2.25. Feladat.

2.2.26. Feladat.

2.3. A természetes számok rendezése

2.3.1. A természetes számok rendezése.

```
> 2<=5; evalb(%);
```

$$2 \leq 5$$

$$true \tag{2.3.1.1}$$

2.3.2. Tétel.

2.3.3. Tétel.

2.3.4. Tétel.

2.3.5. Sorozatok.

```
> i:='i'; j:='j'; $3..9; i^2$i=2/3..10/3; x[i]$i=3..8;
  {j^i$j=i..8}$i=2..4;
```

$$i := i$$

$$j := j$$

$$3, 4, 5, 6, 7, 8, 9$$

$$\frac{4}{9}, \frac{25}{9}, \frac{64}{9}$$

$$x_3, x_4, x_5, x_6, x_7, x_8$$

$$\{4, 9, 16, 25, 36, 49, 64\}, \{27, 64, 125, 216, 343, 512\}, \{256, 625, 1296, 2401, 4096\} \quad (2.3.5.1)$$

->2.3.6. Feladat.

->2.3.7. Feladat: tranzitív lezárt.

->2.3.8. Feladat.

->2.3.9. Feladat.

2.3.10. Példa.

```
> cat("ab","bcc"); cat("bcc","ab"); evalb(%=%%); "ab"||"bcc";
```

"abbcc"

"bccab"

false

"abbcc" (2.3.10.1)

```
> StringTools[Generate](4,"abc");
```

["aaaa", "aaab", "aaac", "aaba", "aabb", "aabc", "aaca", "aacb", "aacc", "abaa", "abab", "abac", "abba", "abbb", "abbc", "abca", "abcb", "abcc", "acaa", "acab", "acac", "acba", "acbb", "acbc", "acca", "accb", "accc", "baaa", "baab", "baac", "baba", "babb", "babc", "baca", "bacb", "bacc", "bbaa", "bbab", "bbac", "bbba", "bbbb", "bbbc", "bbca", "bbcb", "bbcc", "bcaa", "bcab", "bcac", "bcba", "bcbb", "bcbc", "bcca", "bccb", "bccc", "caaa", "caab", "caac", "caba", "cabb", "cabc", "caca", "cacb", "cacc", "cbaa", "cbab", "cbac", "cbba", "cbbb", "cbbc", "cbca", "cbcb", "cbcc", "ccaa", "ccab", "ccac", "ccba", "ccbb", "ccbc", "ccca", "cccb", "cccc"] (2.3.10.2)

2.3.11. Többváltozós függvények.

2.3.12. Motiváció: további rekurzív definíciók.

2.3.13. Általános rekurziótétel.

2.3.14. Példa: Fibonacci-számok.

```
> Fib:=proc(n::nonnegint) option remember;
  if n<2 then n else Fib(dec(n))+Fib(dec(dec(n))) fi end;
  interface(verboseproc=3):

  Fib(3);print(Fib);

  Fib(7);print(Fib);
```

```
Fib := proc(n::nonnegint)
    option remember;
    if n < 2 then
        n
    else
        Fib(dec(n)) + Fib(dec(dec(n)))
    end if
end proc
```

$$2$$

```
proc(n::nonnegint)
    option remember;
    if n < 2 then
        n
    else
        Fib(dec(n)) + Fib(dec(dec(n)))
    end if
end proc#(0) = 0#(1) = 1#(2) = 1#(3) = 2
```

$$13$$

```
proc(n::nonnegint)                                          (2.3.14.1)
    option remember;
    if n < 2 then
        n
    else
        Fib(dec(n)) + Fib(dec(dec(n)))
    end if
```

end proc$\#(0) = 0$

$\#(1) = 1$

$\#(2) = 1$

$\#(3) = 2$

$\#(5) = 5$

$\#(4) = 3$

$\#(7) = 13$

$\#(6) = 8$

2.3.15. Szorzatok és összegek.

```
> prodfrom1:=proc(x,n)
  if n<1 then 1 elif n=1 then x[1] else prodfrom1(x,dec(n))*x
  [n] fi end;

  prodfrom1(y,5);

  product(y[i],i=0..7);

  sum(x[j],j=4.5..8.7);
```

prodfrom1 := **proc**(x, n)
  if $n < 1$ **then**
    1
  elif $n = 1$ **then**
    $x[1]$
  else
    $prodfrom1(x, dec(n)) * x[n]$
  end if
end proc

$$y_1\, y_2\, y_3\, y_4\, y_5$$

$$y_0\, y_1\, y_2\, y_3\, y_4\, y_5\, y_6\, y_7$$

$$x_5 + x_6 + x_7 + x_8 \qquad (2.3.15.1)$$

->2.3.16. Feladat.

2.3.17. Feladat.

->2.3.18. Feladat.

2.3.19. Feladat.

2.3.20. Feladat.

2.3.21. Feladat.

2.3.22. Feladat.

2.3.23. Feladat.

2.3.24. Feladat.

2.3.25. Feladat.

2.3.26. Feladat.

->2.3.27. Feladat.

->2.3.28. Feladat.

*2.3.29. Feladat.

2.3.30. Feladat.

2.3.31. Feladat.

2.3.32. Feladat.

2.3.33. Feladat.

2.3.34. Feladat: Catalan-számok.

2.3.35. Feladat: Catalan-számok.

2.3.36. Logikai függvények.

2.3.36. Logikai függvények normálalakja.

2.3.38. Normálalakok.

2.3.39. A maradékos osztás tétele.

2.3.40. Hányados és maradék.

```
> irem(17,3); iquo(17,3); 3*%+%%;
```

$$2$$

5

17 (2.3.40.1)

2.3.41. Tétel: számrendszerek.

```
> convert(17,base,3); convert(%,base,3,10); convert(17,
  binary);
  convert(17,octal); convert(17,hex); cat("",%);
```

$[2, 2, 1]$

$[7, 1]$

10001

21

11

"11" (2.3.41.1)

2.3.42. Természetes számok számítógépes ábrázolása.

*2.3.43. Hash-transzformáció.

2.3.44. Feladat.

->2.3.45. Feladat.

->2.3.46. Feladat.

->2.3.47. Feladat.

2.3.48. Feladat.

2.3.49. Feladat.

2.3.50. Feladat.

2.3.51. Feladat.

*2.3.52. Feladat.

2.3.53. Feladat.

2.3.54. Feladat.

2.3.55. Feladat.