A B lehetőségeit bemutató programok és bizonyításaik

Fóthi Ákos

A projekt az Eurpai Unió támogatásával és az Európai Szociális Alap társfinanszírozásával valósul meg, a támogatási szerződés száma TÁMOP 4.2.1./B-09/1/KMR-2010-0003.

EGYSZERŰ PÉLDÁK CIKLUSRA

MACHINE
EXAM1
CONCRETE_VARIABLES
xx
INVARIANT
$xx \in 0 \mathinner{..} 10 \rightarrow \mathbf{NAT}$

INITIALISATION
$xx :\in \{(0 \mathinner{..} 10) \times \{0\}\}$
OPERATIONS
nulla=
BEGIN
$xx := (0 \mathinner{..} 10) \times \{0\}$
END
END

IMPLEMENTATION
EXAM1_i
REFINES
EXAM1
INITIALISATION
xx:=$(0 \mathinner{..} 10) \times \{0\}$
OPERATIONS
nulla=
VAR *ii* **IN**
ii:=-1;
WHILE $ii \neq 10$ **DO**
xx(*ii*+1):=0;
ii:=*ii*+1
INVARIANT
$ii \in \mathcal{Z} \wedge$
$xx \in 0 \mathinner{..} 10 \rightarrow \mathbf{NAT} \wedge$
$ii \in$ -1 $\mathinner{..}$ 10 $\wedge$
$\forall jj.(jj{:}(0 \mathinner{..} ii) \Rightarrow xx(jj){=}0)$
VARIANT
11-*ii*
END
END
END

exam1_i.pmm

```
THEORY User_Pass IS
Operation(nulla) & ff(0) & ss & pp;
Operation(nulla) & ff(0) & dc(jj = ii$0+1) & pr & pr & pp
END
```

MACHINE
EXAM2
CONCRETE_VARIABLES
xx
INVARIANT
$xx \in 0 \mathinner{..} 10 \rightarrow \mathbf{NAT}$

INITIALISATION
$xx := (0 \mathinner{..} 10) \times \{0\}$
OPERATIONS
nulla=
BEGIN
$xx := (0 \mathinner{..} 10) \times \{0\}$
END
END

IMPLEMENTATION
EXAM2_i
REFINES
EXAM2
INITIALISATION
$xx{:=}(0 \mathinner{..} 10) \times \{0\}$
OPERATIONS
nulla=
VAR *ii* **IN**
ii:=-1;
WHILE $ii \neq 10$ **DO**
xx(*ii*+1):=0;
ii:=*ii*+1
INVARIANT
$ii \in \mathcal{Z} \wedge$
$xx \in 0 \mathinner{..} 10 \rightarrow \mathbf{NAT} \wedge$
$ii \in$ -1 .. 10 $\wedge$
$xx{=}xx\$0 \lhd (0 \mathinner{..} ii) \times \{0\}$
VARIANT
11-*ii*
END
END
END

exam2_i.pmm

```
THEORY User_Pass IS
Operation(nulla) & ff(0) & pr;
Operation(nulla) & ff(0) & pp
END
```

MACHINE
EXAM3
VARIABLES
xx
INVARIANT
$xx \in 0 \,..\, 10 \rightarrow \mathbf{NAT}$

INITIALISATION
$xx := (0 \,..\, 10) \times \{0\}$
OPERATIONS
nulla=
BEGIN
$xx := (0 \,..\, 10) \times \{0\}$
END
END

IMPLEMENTATION
EXAM3_i
REFINES
EXAM3
CONCRETE_VARIABLES
xx
INITIALISATION
xx:=$(0 \,..\, 10) \times \{0\}$
OPERATIONS
nulla=
VAR *ii* **IN**
ii:=-1;
WHILE $ii \neq 10$ **DO**
xx(*ii*+1):=0;
ii:=*ii*+1
INVARIANT
$ii \in \mathcal{Z} \wedge$
$xx \in 0 \,..\, 10 \rightarrow \mathbf{NAT} \wedge$
$ii \in$ -1 .. 10 $\wedge$
xx=*xx$0* $\lhd (0 \,..\, ii) \times \{0\}$
VARIANT
11-*ii*
END
END
END

EXAM3_i.pmm

```
THEORY User_Pass IS
Operation(nulla) & ff(0) & pr;
Operation(nulla) & ff(0) & pp
END
```

MACHINE
EXAM5
VARIABLES
xx
INVARIANT
$xx \in 0 \mathinner{..} 10 \rightarrow \mathbf{NAT}$

INITIALISATION
  $xx := (0 \mathinner{..} 10) \times \{0\}$
OPERATIONS
nulla=
BEGIN
  $xx := (0 \mathinner{..} 10) \times \{0\}$
END
END

IMPLEMENTATION
EXAM5_i
REFINES
EXAM5
IMPORTS
 EXAMI
OPERATIONS

nulla=
VAR *ii* **IN**
 ii:=-1;
 WHILE $ii \neq 10$ **DO**
  berak(*ii*+1,0);
  ii:=*ii*+1
 INVARIANT
  $ii \in \mathcal{Z} \wedge$
  $xx \in 0 \mathinner{..} 10 \rightarrow \mathbf{NAT} \wedge$
  $ii \in$ -1 $\mathinner{..}$ 10 $\wedge$
  xx=*xx$0* $\lhd (0 \mathinner{..} ii) \times \{0\}$
 VARIANT
  11-*ii*
 END
END
END

EXAM5_i.pmm

```
 THEORY User_Pass IS
Operation(nulla) & ff(0) & pr;
Operation(nulla) & ff(0) & pp
END
```

MACHINE
EXAMI
VARIABLES
xx
INVARIANT
$xx \in 0 \,..\, 10 \rightarrow \mathbf{NAT}$

INITIALISATION
$xx := (0 \,..\, 10) \times \{0\}$
OPERATIONS
$vv \leftarrow$ **kivesz**(ii)=
PRE
$ii \in \mathbf{NAT} \wedge$
$ii \in 0 \,..\, 10$
THEN
vv:=$xx(ii)$
END;

berak(ii,vv)=
PRE
$ii \in \mathbf{NAT} \wedge$
$vv \in \mathbf{NAT} \wedge$
$ii \in 0 \,..\, 10$
THEN
xx$(ii) := vv$
END
END

A LINEÁRIS KERESÉS VÁLTOZATAI

MACHINE
KER1
CONCRETE_VARIABLES
xx
INVARIANT
$xx \in 0 \,..\, 100 \rightarrow \mathbf{NAT}$

INITIALISATION
$xx :\in \{(0 \,..\, 100) \times \{0\}\}$
OPERATIONS
$ll \leftarrow$ **van_e**(ee)**=**
PRE
$ee \in \mathbf{NAT}$
THEN
ll**:= bool**$(ee \in \mathbf{ran}(xx))$
END
END

IMPLEMENTATION
KER1_i
REFINES
KER1

INITIALISATION
xx:=$(0 \mathinner{..} 100) \times \{0\}$
OPERATIONS
$ll \leftarrow$ **van_e**(ee)=
VAR tt, ii **IN**
tt:= **FALSE**;
ii:=-1;
WHILE $ii < 100 \wedge tt$=**FALSE DO**

VAR ss **IN**
ss:= $xx(ii+1)$;
IF $ee = ss$ **THEN**
tt:=**TRUE**
END;
ii:=ii+1
END
INVARIANT
$ii \in -1 \mathinner{..} 100 \wedge$

(tt = **bool**($ee \in$ **ran**$(0 \mathinner{..} ii \lhd xx)$))

VARIANT
102-ii
END;
ll:=tt
END
END

KER1_i.pmm

```
THEORY User_Pass IS
Operation(van_e) & ff(0) & pr & ar(b1.18,Once);
Operation(van_e) & ff(0) & pp
END
```

MACHINE
KER2
CONCRETE_VARIABLES
xx
INVARIANT
$xx \in 0 \mathinner{..} 100 \rightarrow \mathbf{NAT}$

INITIALISATION
$xx :\in \{(0 \mathinner{..} 100) \times \{0\}\}$
OPERATIONS
$ll \leftarrow$ **van_e**(ee)**=**
PRE
$ee \in \mathbf{NAT}$
THEN
ll**:= bool**$(ee \in \mathbf{ran}(xx))$
END
END

IMPLEMENTATION
KER2_i
REFINES
KER2
INITIALISATION
xx:=$(0 \mathinner{..} 100) \times \{0\}$
OPERATIONS
$ll \leftarrow$ **van_e**(ee)=
VAR tt, ii **IN**
tt:= **FALSE**;
ii:=0;
WHILE $ii \leq 100$ **DO**

VAR ss **IN**
ss:= xx(ii);
IF $ee \neq ss$ **THEN**
ii:=ii+1
ELSE
tt:=**TRUE**;
ii:=101
END
END
INVARIANT
$ii \in 0 \mathinner{..} 101 \wedge$
ii+1 $\in 1 \mathinner{..} 102 \wedge$
$tt \in$ **BOOL** $\wedge$
(tt=**TRUE** $\Rightarrow$ ii=101 $\wedge$ tt = **bool**($ee \in$ **ran**(xx))) $\wedge$
(tt=**FALSE** $\Rightarrow$ tt = **bool**($ee \in$ **ran**($0 \mathinner{..} ii\text{-}1 \lhd xx$)))

VARIANT
102-ii
END;
ll:=tt
END
END

KER2_i.pmm

```
 THEORY User_Pass IS
Operation(van_e) & ff(0) & pr;
Operation(van_e) & ff(0) & pp
END
```

A MAXIMUMKERESÉS VÁLTOZATAI

MACHINE
MAX1
CONCRETE_VARIABLES
xx
INVARIANT
$xx \in 0 \,..\, 10 \rightarrow \mathbf{NAT}$

INITIALISATION
  $xx :\in \{(0 \,..\, 10) \times \{0\}\}$
OPERATIONS
$maxi \leftarrow$ **MaxKer=**
BEGIN
  $maxi :=$ **max**(**ran**(xx))
END
END

IMPLEMENTATION
MAX1_i
REFINES
MAX1
INITIALISATION
xx:=$(0 \mathinner{..} 10) \times \{0\}$
OPERATIONS
$maxi \leftarrow$ **MaxKer=**
VAR ii, ss **IN**
ii:=0;
ss:=$xx(ii)$;
WHILE $ii<10$ **DO**
VAR vv **IN**
ii:=ii+1;
vv:=$xx(ii)$;
IF $(ss \leq vv)$ **THEN**
ss:=vv
END
END
INVARIANT
$ii \in \mathcal{Z} \wedge$
$ii \in 0 \mathinner{..} 10 \wedge$
$ss \in$ **ran**$(xx) \wedge$
$\forall jj.(jj{:}(0 \mathinner{..} ii) \Rightarrow xx(jj) \leq ss)$
VARIANT
10-ii
END;
$maxi$:=ss
END
END

MAX1_i.pmm

```
THEORY User_Pass IS
Operation(MaxKer) & ff(0) & pr;
Operation(MaxKer) & ff(0) & pp;
Operation(MaxKer) & ff(0) & dc(jj = ii+1) & pr & pp
END
```

MACHINE
MAX2
CONCRETE_VARIABLES
xx
INVARIANT
$xx \in 0 \mathinner{..} 10 \rightarrow \mathbf{NAT}$

INITIALISATION
$xx :\in \{(0 \mathinner{..} 10) \times \{0\}\}$
OPERATIONS
$maxi \leftarrow$ **MaxKer=**
BEGIN
$maxi$:= **max**(**ran**(xx))
END
END

REFINEMENT
MAX2_r
REFINES
MAX2

INITIALISATION
$xx :\in \{(0 \mathinner{..} 10) \times \{0\}\}$
OPERATIONS
$maxi \leftarrow$ **MaxKer=**
BEGIN
maxi :($maxi \in$ **ran**(xx) $\wedge$
$\forall\, jj.(jj{:}(0 \mathinner{..} 10) \Rightarrow xx(jj) \leq maxi)$
)
END
END

IMPLEMENTATION
MAX2_i
REFINES
MAX2_r
INITIALISATION
xx:=$(0 \mathinner{..} 10) \times \{0\}$
OPERATIONS
maxi $\leftarrow$ **MaxKer=**
VAR *ii*, *ss* **IN**
ii:=0;
ss:=*xx*(*ii*);
WHILE $ii<10$ **DO**
VAR *vv* **IN**
ii:=*ii*+1;
vv:=*xx*(*ii*);
IF $(ss \leq vv)$ **THEN**
ss:=*vv*
END
END
INVARIANT
$ii \in \mathcal{Z} \wedge$
$ii \in 0 \mathinner{..} 10 \wedge$
$ss \in$ **ran**$(xx) \wedge$
$\forall jj.(jj{:}(0 \mathinner{..} ii) \Rightarrow xx(jj) \leq ss)$
VARIANT
10-*ii*
END;
maxi:=*ss*
END
END

MAX2_i.pmm

```
THEORY User_Pass IS
Operation(MaxKer) & ff(0) & pr;
Operation(MaxKer) & ff(0) & pp
END
```

MACHINE
MAX3
CONCRETE_VARIABLES
xx
INVARIANT
$xx \in 0 \mathinner{..} 10 \rightarrow \mathbf{NAT}$

INITIALISATION
$xx :\in \{(0 \mathinner{..} 10) \times \{0\}\}$
OPERATIONS
$maxi \leftarrow$ **MaxKer=**
BEGIN
$maxi$:= **max**(**ran**(xx))
END
END

REFINEMENT
MAX3_r
REFINES
MAX3

INITIALISATION
$xx :\in \{(0 \mathinner{..} 10) \times \{0\}\}$
OPERATIONS
$maxi \leftarrow$ **MaxKer=**
BEGIN
maxi :($maxi \in$ **ran**(xx) $\wedge$
$\forall\, jj.(jj{:}(0 \mathinner{..} 10) \Rightarrow xx(jj) \leq maxi)$
)
END
END

IMPLEMENTATION
MAX3_i
REFINES
MAX3_r
INITIALISATION
$xx := (0 \,..\, 10) \times \{0\}$
OPERATIONS
$maxi \leftarrow$ **MaxKer=**
VAR *ii*, *ss* **IN**
$ii := 0$;
$ss := xx(ii)$;
WHILE $ii < 10$ **DO**
VAR *vv* **IN**
$ii := ii + 1$;
$vv := xx(ii)$;
IF $(ss \leq vv)$ **THEN**
$ss := vv$
END
END
INVARIANT
$ii \in \mathcal{Z} \wedge$
$ii \in 0 \,..\, 10 \wedge$
$ss =$ **max**(**ran**$((0 \,..\, ii) \lhd xx)$)
VARIANT
$10 - ii$
END;
$maxi := ss$
END
END

csg_i.pmm

```
THEORY User_Pass IS
Operation(MaxKer) & ff(0) & pr;
Operation(MaxKer) & ff(0) & pp
END
```

BONYLULTABB BIZONYÍTÁS

MACHINE
hprim
OPERATIONS
hivas=skip
END

IMPLEMENTATION
hprim_i
REFINES
hprim
IMPORTS
prim, *BASIC_IO*

OPERATIONS
hivas=
BEGIN
VAR *nn*,*xx* **IN**
STRING_WRITE("Az adott szám= ");
nn ← **INTERVAL_READ**(3,**MAXINT**-1);

xx ← **is_prim** (*nn*);
IF *xx*=**TRUE THEN**
INT_WRITE(*nn*);
STRING_WRITE(" prim\n")
ELSE
INT_WRITE(*nn*);
STRING_WRITE(" nem prim\n")
END
END
END

END

MACHINE
prim

OPERATIONS
pp ← **is_prim** (*nn*) =
PRE
$nn \in \mathbf{NAT} \wedge$
$nn \geq 3 \wedge$
$nn <$**MAXINT**
THEN
pp := **bool** ($\forall$ *ii* . (*ii* : (2 . . *nn* - 1) $\Rightarrow$ (*nn* $\mathbf{mod}$ *ii*) $\neq 0$))
END

END

IMPLEMENTATION
prim_i
REFINES
prim

OPERATIONS
pp ← **is_prim** (*nn*) =
BEGIN
VAR *ll* , *kk* , *kk1* **IN**
ll := **TRUE** ;
kk := *nn* ;
WHILE (2 ≠ *kk* ∧ *ll* = **TRUE**) **DO**

kk1:=*nn* **mod** (*kk*-1);
IF *kk1* = 0 **THEN**
kk := *kk*-1;
ll := **FALSE**
ELSE
kk := *kk*-1
END

INVARIANT
ll ∈ **BOOL** ∧
nn ∈ **NAT** ∧
nn ≥ 3 ∧
kk ∈ 2 . . *nn* ∧
(*ll*=**TRUE** ⇒ (∀ *jj*.(*jj* ∈ *kk* . . *nn*-1 ⇒ *nn* **mod** *jj* ≠ 0))) ∧
(*ll*=**FALSE** ⇒ (*kk* ∈ 2 . . *nn*-1 ∧ *nn* **mod** *kk* = 0))

VARIANT
kk
END ;
pp :=*ll*
END
END

END

prim_i.pmm

```
THEORY User_Pass IS
Operation(is_prim) & ff(0) & pr;
Operation(is_prim) & ff(0) & pr & pp;
Operation(is_prim) & ff(0) & pr & ah(-(1)+kk<=2147483647)
         & pr & pr & ar(b1.46,Once);
Operation(is_prim) & ff(0) & dc(jj = kk-1) & pr & ah(jj: kk..nn-1)
         & ss & pp & ah(!jj.(jj: kk..nn-1 => not(nn mod jj = 0))) & pr;
Operation(is_prim) & ff(0) & dc(ll$7777 = TRUE) & dd & ah(kk$7777 = 2)
         & pr & pp & pr & dd & ah(ll$7777 = FALSE) & pp & dd & pr
         & se(kk$7777) & pr
END
```

A CSOKIAUTOMATA FUTTATHATÓ VÁLTOZATA

A B kód

```
MACHINE
   hivo
OPERATIONS
   hivas= skip
END

IMPLEMENTATION
  hivo_i
REFINES
  hivo
IMPORTS
   csg,BASIC_IO
OPERATIONS
   hivas=
   BEGIN
      init;
      STRING_WRITE("indul\n");
      szerviz;
      STRING_WRITE("szerviz\n");
      bedob10;
      STRING_WRITE("bedob10\n");
      kerkiscsoki;
      STRING_WRITE("kerkiscsoki\n");
      bedob20;
      STRING_WRITE("bedob20\n");
      kernagycsoki;
      STRING_WRITE("kernagycsoki\n");
      bedob20;
      STRING_WRITE("bedob20\n");
      kerkiscsoki;
      STRING_WRITE("kerkiscsoki\n");
      visszaad;
      STRING_WRITE("visszaad\n")
   END

END
```

MACHINE
csg
CONSTANTS
maxkassza10,
maxkassza20,
maxkiscsoki,
maxnagycsoki
PROPERTIES
maxkassza10 ∈ **NAT** ∧ *maxkassza10* = 10
∧ *maxkassza20* ∈ **NAT** ∧ *maxkassza20* = 5
∧ *maxkiscsoki* ∈ **NAT** ∧ *maxkiscsoki* = 5
∧ *maxnagycsoki* ∈ **NAT** ∧ *maxnagycsoki* = 5
VARIABLES
kiscsoki,
nagycsoki,
kassza10,
kassza20,
bedobott
INVARIANT
kassza10 ∈ **NAT** ∧ *kassza10* ≤ *maxkassza10*
∧ *kassza20* ∈ **NAT** ∧ *kassza20* ≤ *maxkassza20*
∧ *kiscsoki* ∈ **NAT** ∧ *kiscsoki* ≤ *maxkiscsoki*
∧ *nagycsoki* ∈ **NAT** ∧ *nagycsoki* ≤ *maxnagycsoki*
∧ *bedobott* ∈ **NAT**
∧ *bedobott* ≤ *kassza10* × 10+*kassza20* × 20-10 × *maxkiscsoki*
∧ (*maxkiscsoki-kiscsoki*) × 10 + (*maxnagycsoki-nagycsoki*) × 20 +*maxkiscsoki* × 10
=
(*kassza10* × 10) + (*kassza20* × 20) - *bedobott*

INITIALISATION
kiscsoki := *maxkiscsoki*
|| *nagycsoki* := *maxnagycsoki*
|| *kassza10* := *maxkiscsoki*
|| *kassza20* := 0
|| *bedobott* := 0
OPERATIONS
init=
BEGIN
kiscsoki := *maxkiscsoki*
|| *nagycsoki* := *maxnagycsoki*
|| *kassza10* := *maxkiscsoki*
|| *kassza20* := 0
|| *bedobott* := 0
END
;
szerviz =
PRE
bedobott = 0
THEN
kiscsoki := *maxkiscsoki*
|| *nagycsoki* := *maxnagycsoki*

|| *kassza10* := *maxkiscsoki*
|| *kassza20* := 0
END
;
bedob10 =
PRE
kassza10 < *maxkassza10*
THEN
bedobott := *bedobott* + 10 ||
kassza10 := *kassza10* + 1
END
;
kerkiscsoki =
PRE
bedobott ≥ 10 ∧ *kiscsoki* > 0
THEN
bedobott := *bedobott* - 10
|| *kiscsoki* := *kiscsoki* - 1
END
;
kernagycsoki =
PRE
bedobott ≥ 20 ∧ *nagycsoki* > 0
THEN
bedobott := *bedobott* - 20
|| *nagycsoki* := *nagycsoki* - 1
END
;
bedob20 =
PRE
kassza20 < *maxkassza20*
THEN
bedobott := *bedobott* + 20 ||
kassza20 := *kassza20* + 1
END
;

visszaad =
ANY
vissza10,*vissza20*
WHERE
vissza10 ∈ 0 . . *kassza10*
∧ *vissza20* ∈ 0 . . *kassza20*
∧ (*vissza10* × 10) + (*vissza20* × 20) = *bedobott*
THEN
bedobott := 0
|| *kassza10* := *kassza10* - *vissza10*
|| *kassza20* := *kassza20* - *vissza20*
END

END

IMPLEMENTATION
csg_i
REFINES
csg
VALUES
maxkassza10 = 10;
maxkassza20 = 5;
maxkiscsoki = 5;
maxnagycsoki = 5

CONCRETE_VARIABLES
kiscsoki,
nagycsoki,
kassza10,
kassza20,
be10, *be20*

INVARIANT
$kassza10 \in \mathbf{NAT} \wedge kassza10 \leq maxkassza10$
$\wedge \quad kassza20 \in \mathbf{NAT} \wedge kassza20 \leq maxkassza20$
$\wedge\ kiscsoki \in \mathbf{NAT} \wedge kiscsoki \leq maxkiscsoki$
$\wedge\ nagycsoki \in \mathbf{NAT} \wedge nagycsoki \leq maxnagycsoki$
$\wedge\ be10 \in \mathbf{NAT} \wedge be20 \in \mathbf{NAT}$
$\wedge\ be10 \leq kassza10\text{-}kiscsoki \wedge be20 \leq kassza20$
$\wedge\ bedobott = 10 \times be10 + 20 \times be20$
$\wedge\ (maxkiscsoki\text{-}kiscsoki) \times 10 + (maxnagycsoki\text{-}nagycsoki) \times 20 + maxkiscsoki \times 10$
=
$(kassza10 \times 10) + (kassza20 \times 20)$ - *bedobott*

INITIALISATION
kiscsoki := *maxkiscsoki*;
nagycsoki := *maxnagycsoki*;
kassza10 := *maxkiscsoki*;
kassza20 := 0;
be10 := 0; *be20* := 0

OPERATIONS
init =
BEGIN
kiscsoki := *maxkiscsoki*;
nagycsoki := *maxnagycsoki*;
kassza10 := *maxkiscsoki*;
kassza20 := 0;
be10 := 0;
be20 := 0
END
;
szerviz =
BEGIN

```
        kiscsoki := maxkiscsoki;
    nagycsoki := maxnagycsoki;
    kassza10 := maxkiscsoki;
    kassza20 := 0
    END
;
bedob10 =
    BEGIN
    be10 := be10 + 1;
    kassza10 := kassza10 + 1
    END
;
kerkiscsoki =
    BEGIN
        IF be10 = 0  THEN
            be10 := 1;
            be20 := be20 -1;
            kiscsoki := kiscsoki - 1
        ELSE
            be10 := be10 - 1;
            kiscsoki := kiscsoki - 1
        END

    END
;
kernagycsoki =
BEGIN
    IF be20 = 0 THEN
        be10 := be10-2
    ELSE
        be20 := be20 - 1
    END;
     nagycsoki := nagycsoki - 1
    END
;
bedob20 =
    BEGIN
    be20 := be20 + 1;
    kassza20 := kassza20 + 1
    END



;
visszaad =
BEGIN

    kassza10 := kassza10 - be10;
    kassza20 := kassza20 - be20;
    be10 := 0; be20 := 0
    END


END
```

A HELYESÉG IGAZOLÁSA

A „pmm” fájlok

csg.pmm

```
THEORY User_Pass IS
Operation(bedob10) & ff(0) & pp(rp.1);
Operation(kerkiscsoki) & ff(0) & pp(rp.1);
Operation(kernagycsoki) & ff(0) & pp(rp.1);
Operation(bedob20) & ff(0) & pp(rp.1);
Operation(visszaad) & ff(0) & pp(rp.1)
END
```

csg_i.pmm

```
THEORY User_Pass IS
Operation(szerviz) & ff(0) & ss & pp;
Operation(bedob10) & ff(0) & ss & pp;
Operation(kerkiscsoki) & ff(0) & ss & pp;
Operation(kernagycsoki) & ff(0) & ss & pp;
Operation(bedob20) & ff(0) & ss & pp;
Operation(visszaad) & ff(0) & ss & pp
END
```

A bizonyítás teljes menete

```
THEORY MethodList IS
 pp(rp.1);
 pp(rp.1);
 pr;
 pr;
 pr;
 pr;
 pr;
 pr;
 pr;
 pr;
 pp(rp.1);
 pr;
 pr;
 pr;
 pr;
 pr;
 pr;
 pr;
 pp(rp.1);
 pr;
 pr;
 pr;
 pr;
 pr;
 pr;
 pr;
 pr;
 pp(rp.1);
 pr;
 pr;
 pr;
 pr;
 pr;
 pr;
 pr;
 pr;
 pp(rp.1);
 pr;
 pr;
 pr;
 pr;
```

```
  pr;
  pr;
  pr;
  pr;
  pr;
  pr;
  pr;
  pr;
  pr;
  pr;
  pr;
  pr;
  pr;
  pr;
  pr;
  pr;
  pr
END
```

```
THEORY MethodList IS
 ss & pp;
 pr;
 pr;
 pr;
 pr;
 pr;
 ss & pp;
 pr;
 pr;
 pr;
 pr;
 pr;
 pr;
 pr;
 pr;
 pr;
 pr;
 ss & pp;
 pr;
 pr;
 ss & pp;
 pr;
 pr;
 pr;
 pr;
 pr;
 pr;
 pr;
 pr;
 ss & pp;
 pr;
 pr;
 pr;
 pr;
 ss & pp;
 ss & pp;
 pr;
 ss & pp;
 pr;
 pr;
 pr;
 pr;
 pr;
 pr;
 pr;
 pr;
```

```
pr;
pr;
pr;
pr;
pr;
pr;
pr;
pr;
pr;
pr;
pr;
ss & pp;
pr;
pr;
pr;
pr;
ss & pp;
ss & pp;
pr;
pr;
ss & pp;
pr;
pr;
pr;
pr;
pr;
pr;
pr;
pr;
pr;
pr;
pr;
pr;
pr;
pr;
pr;
pr;
pr;
pr;
pr;
pr;
ss & pp;
pr;
pr;
pr;
ss & pp;
pr;
pr;
```

```
  pr;
  pr;
  pr;
  pr;
  pr;
  pr;
  pr;
  ss & pp;
  ss & pp;
  pr;
  pr;
  pr;
  pr;
  pr;
  pr;
  pr;
  pr;
  pr;
  pr;
  pr;
  pr;
  pr;
  pr;
  pr;
  pr;
  pr;
  pr;
  pr;
  pr;
  pr
END
```

A generált Ada kód

```
-- /home/atelierb/AB/csoper/lang/ada/csoper.bod
-- Project startup module
-- We need all the specifications in order to solve the SEES links
with hivo ;
with csg ;
with basic_io ;
procedure csoper is
this : hivo.PTR_hivo ;
begin
-- Allocation of main object
this := new hivo.TYPE_hivo ;
-- Create instances
hivo.IMPORTS(this) ;
-- Set SEES links
-- Set links for glued variables
-- Initialise instances
hivo.INITIALISATION(this) ;
-- Call entry point
hivo.#hivo#hivas(this) ;
-- End of project
end csoper ;
```

```
 -- File /home/atelierb/AB/csoper/lang/ada/hivo.ads
-- generated by Atelier-B/blk on Mon Oct  4 12:29:28 2010
-- from input file /home/atelierb/AB/csoper/lang/ada/hivo.str
-- File hivo.str
-- generated by Atelier-B/tbada on Mon Oct  4 12:29:28 2010
-- from input_file hivo.imp, checksum f6891ce8e00b4f02690f6a6768892a21
-- hivo_i
-- * Author: atelierb
-- * Creation date: 2010.10.04.
--
with system ;
use system ;
with Unchecked_Conversion ;
with sets ;
use sets ;

-- IMPORTS clause
with csg ;
with BASIC_IO ;

package hivo is

type TYPE_hivo is record
-- Is instance initialised ?
initialisation : BOOLEAN ;
-- Imported machines
ref_csg : csg.PTR_csg ;
ref_BASIC_IO : BASIC_IO.PTR_BASIC_IO ;
end record ;
type PTR_hivo is access TYPE_hivo ;
--------------------
-- INITIALISATION --
--------------------
procedure IMPORTS(this : in PTR_hivo) ; -- Creates imported instances
procedure INITIALISATION(this : in PTR_hivo) ; -- Initialises instances
----------------
-- OPERATIONS --
----------------
procedure hivas(this : in PTR_hivo) ;

end hivo ;
```

```
 -- File hivo.bod
-- generated by Atelier-B/tbada on Mon Oct  4 12:29:28 2010
-- from input_file hivo.imp, checksum f6891ce8e00b4f02690f6a6768892a21

-- hivo_i
-- * Author: atelierb
-- * Creation date: 2010.10.04.
--
package body hivo is

--------------------
-- INITIALISATION --
--------------------
procedure IMPORTS(this : in PTR_hivo) is
begin
-- IMPORTS Clause
-- IMPORTS csg
this.ref_csg := new csg.TYPE_csg ;
csg.IMPORTS(this.ref_csg) ;
-- IMPORTS BASIC_IO
this.ref_BASIC_IO := new BASIC_IO.TYPE_BASIC_IO ;
BASIC_IO.IMPORTS(this.ref_BASIC_IO) ;
-- Component does not extend any machine
null ;
-- Instance is not initialised
this.initialisation := FALSE ;
end IMPORTS ;
procedure INITIALISATION(this : in PTR_hivo) is
begin
if (this.initialisation = TRUE)
then
return ;
end if ;
this.initialisation := TRUE ;
-- Initialisation of imported machines
csg.INITIALISATION(this.ref_csg) ;
BASIC_IO.INITIALISATION(this.ref_BASIC_IO) ;
end INITIALISATION ;
----------------
-- OPERATIONS --
----------------
procedure #hivo#hivas(this : in PTR_hivo) is
begin
csg.#csg#init(this.ref_csg) ;
BASIC_IO.#basic_io#STRING_WRITE(this.ref_BASIC_IO, "indul\n") ;
csg.#csg#szerviz(this.ref_csg) ;
BASIC_IO.#basic_io#STRING_WRITE(this.ref_BASIC_IO, "szerviz\n") ;
```

```
csg.#csg#bedob10(this.ref_csg) ;
BASIC_IO.#basic_io#STRING_WRITE(this.ref_BASIC_IO, "bedob10\n") ;
csg.#csg#kerkiscsoki(this.ref_csg) ;
BASIC_IO.#basic_io#STRING_WRITE(this.ref_BASIC_IO, "kerkiscsoki\n") ;
csg.#csg#bedob20(this.ref_csg) ;
BASIC_IO.#basic_io#STRING_WRITE(this.ref_BASIC_IO, "bedob20\n") ;
csg.#csg#kernagycsoki(this.ref_csg) ;
BASIC_IO.#basic_io#STRING_WRITE(this.ref_BASIC_IO, "kernagycsoki\n") ;
csg.#csg#bedob20(this.ref_csg) ;
BASIC_IO.#basic_io#STRING_WRITE(this.ref_BASIC_IO, "bedob20\n") ;
csg.#csg#kerkiscsoki(this.ref_csg) ;
BASIC_IO.#basic_io#STRING_WRITE(this.ref_BASIC_IO, "kerkiscsoki\n") ;
csg.#csg#visszaad(this.ref_csg) ;
BASIC_IO.#basic_io#STRING_WRITE(this.ref_BASIC_IO, "visszaad\n") ;
end #hivo#hivas ;

end hivo ;
```

```
 -- File /home/atelierb/AB/csoper/lang/ada/csg.ads
-- generated by Atelier-B/blk on Mon Oct  4 12:29:28 2010
-- from input file /home/atelierb/AB/csoper/lang/ada/csg.str
-- File csg.str
-- generated by Atelier-B/tbada on Mon Oct  4 12:25:15 2010
-- from input_file csg.imp, checksum 854e002c40ce3f6b8eb9c5da0c2c5950
-- csg_i
-- * Author: atelierb
-- * Creation date: 2010.10.04.
--
with system ;
use system ;
with Unchecked_Conversion ;
with sets ;
use sets ;

package csg is

type TYPE_csg is record
-- Is instance initialised ?
initialisation : BOOLEAN ;
-- Machine visible variables
kiscsoki : INTEGER ;
nagycsoki : INTEGER ;
kassza10 : INTEGER ;
kassza20 : INTEGER ;
be10 : INTEGER ;
be20 : INTEGER ;
end record ;
type PTR_csg is access TYPE_csg ;
--------------------
-- INITIALISATION --
--------------------
procedure IMPORTS(this : in PTR_csg) ; -- Creates imported instances
procedure INITIALISATION(this : in PTR_csg) ; -- Initialises instances
----------------
-- OPERATIONS --
----------------
procedure init(this : in PTR_csg) ;
procedure szerviz(this : in PTR_csg) ;
procedure bedob10(this : in PTR_csg) ;
procedure kerkiscsoki(this : in PTR_csg) ;
procedure kernagycsoki(this : in PTR_csg) ;
procedure bedob20(this : in PTR_csg) ;
procedure visszaad(this : in PTR_csg) ;

end csg ;
```

```
 -- File csg.bod
-- generated by Atelier-B/tbada on Mon Oct  4 12:25:15 2010
-- from input_file csg.imp, checksum 854e002c40ce3f6b8eb9c5da0c2c5950

-- csg_i
-- * Author: atelierb
-- * Creation date: 2010.10.04.
--
package body csg is

--------------------
-- INITIALISATION --
--------------------
procedure IMPORTS(this : in PTR_csg) is
begin
-- Component does not import any machine
null ;
-- Component does not extend any machine
null ;
-- Instance is not initialised
this.initialisation := FALSE ;
end IMPORTS ;
procedure INITIALISATION(this : in PTR_csg) is
begin
if (this.initialisation = TRUE)
then
return ;
end if ;
this.initialisation := TRUE ;
-- INITIALISATION clause
this.#csg#kiscsoki := csg_maxkiscsoki ;
this.#csg#nagycsoki := csg_maxnagycsoki ;
this.#csg#kassza10 := csg_maxkiscsoki ;
this.#csg#kassza20 := 0 ;
this.#csg#be10 := 0 ;
this.#csg#be20 := 0 ;
end INITIALISATION ;
----------------
-- OPERATIONS --
----------------
procedure #csg#init(this : in PTR_csg) is
begin
this.#csg#kiscsoki := csg_maxkiscsoki ;
this.#csg#nagycsoki := csg_maxnagycsoki ;
this.#csg#kassza10 := csg_maxkiscsoki ;
this.#csg#kassza20 := 0 ;
this.#csg#be10 := 0 ;
```

```
this.#csg#be20 := 0 ;
end #csg#init ;

procedure #csg#szerviz(this : in PTR_csg) is
begin
this.#csg#kiscsoki := csg_maxkiscsoki ;
this.#csg#nagycsoki := csg_maxnagycsoki ;
this.#csg#kassza10 := csg_maxkiscsoki ;
this.#csg#kassza20 := 0 ;
end #csg#szerviz ;

procedure #csg#bedob10(this : in PTR_csg) is
begin
this.#csg#be10 := this.#csg#be10 + 1 ;
this.#csg#kassza10 := this.#csg#kassza10 + 1 ;
end #csg#bedob10 ;

procedure #csg#kerkiscsoki(this : in PTR_csg) is
begin
if (this.#csg#be10 = 0)
then
this.#csg#be10 := 1 ;
this.#csg#be20 := this.#csg#be20 - 1 ;
this.#csg#kiscsoki := this.#csg#kiscsoki - 1 ;
else
this.#csg#be10 := this.#csg#be10 - 1 ;
this.#csg#kiscsoki := this.#csg#kiscsoki - 1 ;
end if ;
end #csg#kerkiscsoki ;

procedure #csg#kernagycsoki(this : in PTR_csg) is
begin
if (this.#csg#be20 = 0)
then
this.#csg#be10 := this.#csg#be10 - 2 ;
else
this.#csg#be20 := this.#csg#be20 - 1 ;
end if ;
this.#csg#nagycsoki := this.#csg#nagycsoki - 1 ;
end #csg#kernagycsoki ;

procedure #csg#bedob20(this : in PTR_csg) is
begin
this.#csg#be20 := this.#csg#be20 + 1 ;
this.#csg#kassza20 := this.#csg#kassza20 + 1 ;
end #csg#bedob20 ;
```

```
procedure #csg#visszaad(this : in PTR_csg) is
begin
this.#csg#kassza10 := this.#csg#kassza10 - this.#csg#be10 ;
this.#csg#kassza20 := this.#csg#kassza20 - this.#csg#be20 ;
this.#csg#be10 := 0 ;
this.#csg#be20 := 0 ;
end #csg#visszaad ;

-- Access methods
end csg ;
```

```
 -- File csg.bod
-- generated by Atelier-B/tbada on Mon Oct  4 12:25:15 2010
-- from input_file csg.imp, checksum 854e002c40ce3f6b8eb9c5da0c2c5950

-- csg_i
-- * Author: atelierb
-- * Creation date: 2010.10.04.
--
package body csg is

--------------------
-- INITIALISATION --
--------------------
procedure IMPORTS(this : in PTR_csg) is
begin
-- Component does not import any machine
null ;
-- Component does not extend any machine
null ;
-- Instance is not initialised
this.initialisation := FALSE ;
end IMPORTS ;
procedure INITIALISATION(this : in PTR_csg) is
begin
if (this.initialisation = TRUE)
then
return ;
end if ;
this.initialisation := TRUE ;
-- INITIALISATION clause
this.#csg#kiscsoki := csg_maxkiscsoki ;
this.#csg#nagycsoki := csg_maxnagycsoki ;
this.#csg#kassza10 := csg_maxkiscsoki ;
this.#csg#kassza20 := 0 ;
this.#csg#be10 := 0 ;
this.#csg#be20 := 0 ;
end INITIALISATION ;
----------------
-- OPERATIONS --
----------------
procedure #csg#init(this : in PTR_csg) is
begin
this.#csg#kiscsoki := csg_maxkiscsoki ;
this.#csg#nagycsoki := csg_maxnagycsoki ;
this.#csg#kassza10 := csg_maxkiscsoki ;
this.#csg#kassza20 := 0 ;
this.#csg#be10 := 0 ;
```

```
this.#csg#be20 := 0 ;
end #csg#init ;

procedure #csg#szerviz(this : in PTR_csg) is
begin
this.#csg#kiscsoki := csg_maxkiscsoki ;
this.#csg#nagycsoki := csg_maxnagycsoki ;
this.#csg#kassza10 := csg_maxkiscsoki ;
this.#csg#kassza20 := 0 ;
end #csg#szerviz ;

procedure #csg#bedob10(this : in PTR_csg) is
begin
this.#csg#be10 := this.#csg#be10 + 1 ;
this.#csg#kassza10 := this.#csg#kassza10 + 1 ;
end #csg#bedob10 ;

procedure #csg#kerkiscsoki(this : in PTR_csg) is
begin
if (this.#csg#be10 = 0)
then
this.#csg#be10 := 1 ;
this.#csg#be20 := this.#csg#be20 - 1 ;
this.#csg#kiscsoki := this.#csg#kiscsoki - 1 ;
else
this.#csg#be10 := this.#csg#be10 - 1 ;
this.#csg#kiscsoki := this.#csg#kiscsoki - 1 ;
end if ;
end #csg#kerkiscsoki ;

procedure #csg#kernagycsoki(this : in PTR_csg) is
begin
if (this.#csg#be20 = 0)
then
this.#csg#be10 := this.#csg#be10 - 2 ;
else
this.#csg#be20 := this.#csg#be20 - 1 ;
end if ;
this.#csg#nagycsoki := this.#csg#nagycsoki - 1 ;
end #csg#kernagycsoki ;

procedure #csg#bedob20(this : in PTR_csg) is
begin
this.#csg#be20 := this.#csg#be20 + 1 ;
this.#csg#kassza20 := this.#csg#kassza20 + 1 ;
end #csg#bedob20 ;
```

```
procedure #csg#visszaad(this : in PTR_csg) is
begin
this.#csg#kassza10 := this.#csg#kassza10 - this.#csg#be10 ;
this.#csg#kassza20 := this.#csg#kassza20 - this.#csg#be20 ;
this.#csg#be10 := 0 ;
this.#csg#be20 := 0 ;
end #csg#visszaad ;

-- Access methods
end csg ;
```

```
 -- File /home/atelierb/AB/csoper/lang/ada/sets.ads
-- generated by Atelier-B/blk on Mon Oct  4 12:29:28 2010
-- from input file /home/atelierb/AB/csoper/lang/ada/sets.str
package sets is
-- Builtin functions
function succ(x : in INTEGER) return INTEGER ;
function pred(x : in INTEGER) return INTEGER ;
function bool(x : in BOOLEAN) return BOOLEAN ;
-- Builtin constants
MAXINT : constant INTEGER := 2147483647 ;
MININT : constant INTEGER := -2147483647 ;
-- Non-array constants from machine csg
csg_maxkassza10 : constant INTEGER := 10 ;

csg_maxkassza20 : constant INTEGER := 5 ;

csg_maxkiscsoki : constant INTEGER := 5 ;

csg_maxnagycsoki : constant INTEGER := 5 ;

-- Builtin functions are inlined for efficiency
pragma inline(succ, pred, bool) ;
end sets ;


 package body sets is
function succ(x : in INTEGER) return INTEGER is
begin
return x + 1 ;
end succ ;
function pred(x : in INTEGER) return INTEGER is
begin
return x - 1 ;
end pred ;
function bool(x : in BOOLEAN) return BOOLEAN is
begin
return x ;
end bool ;
end sets ;
```

```
 -- File /home/atelierb/AB/csoper/lang/ada/basic_io.ads
-- generated by Atelier-B/blk on Mon Oct  4 12:29:28 2010
-- from input file /home/atelierb/AB/BASIC/trad/ada/basic_io.str
--*************************************************************************
--
--Base machine BASIC_IO for ADA target language
--
--(C) 1996 CLEARSY
--
-- Version @(#) basic_io.str version 1.4 (date : 26 Nov 1996)
--
--*************************************************************************
with system ;
use system ;
with Unchecked_Conversion ;
with sets ;
use sets ;

package BASIC_IO is

SCCS_ID : CONSTANT STRING := "@(#) basic_io.str version 1.4 (date : 26 Nov 1996)

type TYPE_BASIC_IO is record
-- Is instance initialised ?
initialisation : BOOLEAN ;

end record ;

type PTR_BASIC_IO is access TYPE_BASIC_IO ;

--------------------
-- INITIALISATION --
--------------------
procedure IMPORTS(this : in PTR_BASIC_IO) ; -- Creates imported instances
procedure INITIALISATION(this : in PTR_BASIC_IO) ; -- Initialises instances

----------------
-- OPERATIONS --
----------------
procedure INTERVAL_READ(
this : in PTR_BASIC_IO ;
mm : in INTEGER ;
nn : in INTEGER ;
bb : in out INTEGER) ;
procedure INT_WRITE(this : in PTR_BASIC_IO ; vv : in INTEGER) ;
procedure BOOL_READ(this : in PTR_BASIC_IO ; bb : in out BOOLEAN) ;
procedure BOOL_WRITE(this : in PTR_BASIC_IO ; bb : in BOOLEAN) ;
```

```
procedure CHAR_READ(this : in PTR_BASIC_IO ; cc : in out INTEGER) ;
procedure CHAR_WRITE(this : in PTR_BASIC_IO ; cc : in INTEGER) ;
procedure STRING_WRITE(this : in PTR_BASIC_IO ; ss : in STRING) ;

end BASIC_IO ;
```