

PROGRAMAZIOAREN METODOLOGIA

**KUDEAKETAREN ETA INFORMAZIO SISTEMEN INFORMATIKAREN INGENIARITZAKO
GRADUA**

**BILBOKO INGENIARITZA ESKOLA
UPV / EHU**

LENGOAIA ETA SISTEMA INFORMATIKOAK SAILA

1. MAILA
31 TALDEA

4. GAIA **PROGRAMEN ERATORPEN FORMALA**

2018-19

José Gaintzarain Ibarmia

Bulegoa: P3I 40
Tutoretza-ordutegia: GAUR-en begiratu

AURKIBIDEA

4.1. SARRERA	5
4.2. ADIBIDEAK	6
4.2.1. Bektore bateko elementuen batura kalkulatu (1. ariketa)	6
4.2.2. Bektore bateko elementuen batura kalkulatu (4. ariketa)	14
4.2.3. c aldagaian 0 itzuli $A(1..n)$ taulako posizio denetan 0 badago (7. ariketa)	22
4.2.4. c aldagaian 0 itzuli $A(1..n)$ taulan gutxienez posizio batean 0 badago (11. ariketa)	31
4.2.5. Bi bektore berdinak al diren erabaki (12. ariketa)	40
4.2.6. x elementua $A(1..n)$ bektorean agertzen al den erabaki (18. ariketa)	49

4.1. SARRERA

ϕ eta ψ formulen bidez emandako aurre-ondoetako espezifikazioa, INB inbariantea eta E espresioa kontuan hartuz While bat eraikitzea da helburua.

```

{φ}

Hasieraketak?

while {INB} B? loop
    Aginduak?
end loop;
{ψ}
E

```

"Hasieraketa;", "B" eta "Aginduak" kalkulatu beharko dira.

While-aren hiru osagai horiek kalkulatzeko While-aren erregelako sei puntuak hartu beharko dira kontuan:

- I. $\phi \rightarrow \text{INB}$
- II. $\text{INB} \rightarrow \text{def}(B)$
- III. $\{\text{INB} \wedge B\}$
 Aginduak
 $\{\text{INB}\}$
- IV. $(\text{INB} \wedge \neg B) \rightarrow \psi$
- V. $(\text{INB} \wedge B) \rightarrow E > 0$
- VI. $\{\text{INB} \wedge B \wedge E = v\}$
 Aginduak
 $\{E < v\}$

A) Hasieraketa (While-aren erregelako I puntua erabiliz)

Hasieraketa kalkulatzeko $\phi \rightarrow \text{INB}$ betetzen al den aztertu beharko da.

- $\phi \rightarrow \text{INB}$ betetzen bada ez da ezer hasieratu behar.
- $\phi \rightarrow \text{INB}$ ez bada betetzen, aldagaien bat (gutxienez bat) hasieratu beharko da.

B) While-aren baldintza B kalkulatu (While-aren erregelako II, IV eta V puntuak erabiliz)

- Hasteko $\neg B$ kalkulatu da "while-a noiz geldituko da?" galderari erantzunez.
- $\neg B$ kalkulatu ondoren B bere kontrakoa izango da.
- Bukatzeko $\text{INB} \rightarrow \text{def}(B)$, $(\text{INB} \wedge \neg B) \rightarrow \psi$ eta $(\text{INB} \wedge B) \rightarrow E > 0$ inplikazioak betetzen al diren egiaztatu beharko da, hau da, While-aren II, IV eta V puntuak betetzen al diren egiaztatu beharko da.

C) While-aren barruko aginduak kalkulatu (While-aren erregelako III eta VI puntuak erabiliz)

4.2. ADIBIDEAK

4.2.1. BEKTORE BATEKO ELEMENTUEN BATURA KALKULATU (1. ARIKETA)

A(1..n) bektorea emanda, A(1..n) bektoreko elementuen batura s aldagaian kalkulatzeko duen programa eratorri Hoare-ren kalkuluko Esleipenaren Axioma eta While-aren Erregela erabiliz. Eratorritako programak INB inbariantea, E espresioa eta φ eta ψ formulen bidez emandako aurre-ondoetako espezifikazioarekiko zuzena izan beharko du.

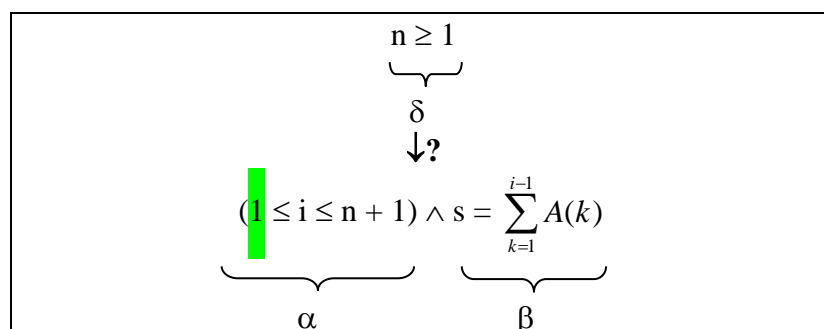
$\{\varphi\} \equiv \{n \geq 1\}$ Hasieraketak? $\{INB\}$ while $\{INB\} \{E\} B?$ loop Aginduak? end loop; $\{\psi\} \equiv \{s = \sum_{k=1}^n A(k)\}$
$\{INB\} \equiv \{(1 \leq i \leq n + 1) \wedge s = \sum_{k=1}^{i-1} A(k)\}$ $E = n + 1 - i$

Hasteko bektorea ezkerretik eskuinera edo eskuinetik ezkerrera zeharkatu behar al den erabaki behar da. E espresioa "goiko muka – indizea" erakoa denez, bektorea ezkerretik eskuinera zeharkatu beharko da.

- **Hasieraketak**

Hasieraketak kalkulatzeko While-aren erregelako lehenengo puntua hartu behar da kontuan, hau da, φ formulak inbariantea inplikatzeko al duen begiratu behar da. Inplikatzeko badu, ez da ezer hasieratu behar. Ez badu inplikatzeko, hasieraketaren bat beharko da. Inbarianteak adieraziko digu zein aldagai eta nola hasieratu. Aldagai baten hasieraketa zein izango den kalkulatu ondoren, Esleipenaren Axioma erabiliz hasieraketa horri dagokion formula kalkulatu da eta φ formulak formula berri hori inplikatzeko al duen aztertuko da. Hasieraketak gehitzeko prozesua φ formulak inplikatzeko duen formula bat lortu arte jarraitu beharko da.

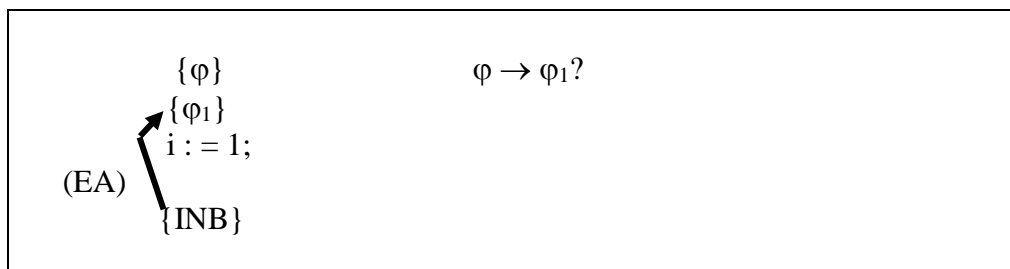
$\varphi \rightarrow INB?$



Inplikazio hori ez da betetzen inplikazioaren lehenengo zatian (δ zatian) ez baitago i eta s aldagaiek α eta β formulek diotena betetzen dutela ziurtatzeko erabili dezakegun informaziorik. Helburua inplikazioa betetzea denez, hau da, α eta β betetzea denez, inplikazio hori betearaziko duten balioekin hasieratu beharko dira i eta s . β formularen i eta s agertzen direnez eta α formularen i bakarrik agertzen denez, α kontuan hartuz i nola hasieratu erabakiko dugu.

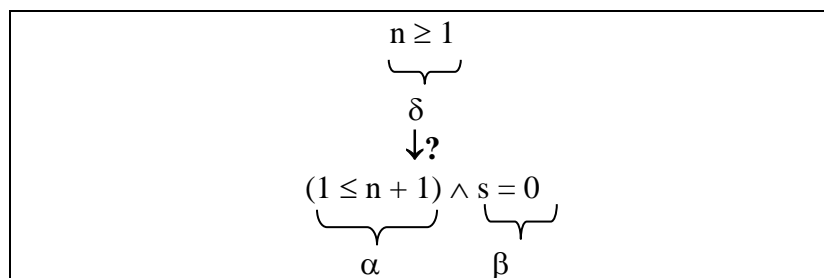
Bektorea ezkerretik eskuinera zeharkatu behar denez, α betetzeko i -ri 1 esleitzea nahikoa da. Beraz α formularen begiratzuz i aldagaiak har dezakeen balio txikiena esleituiko diogu.

Orain Esleipenaren Axiomari jarraituz $\{\varphi_1\}$ formula kalkulatu dugu

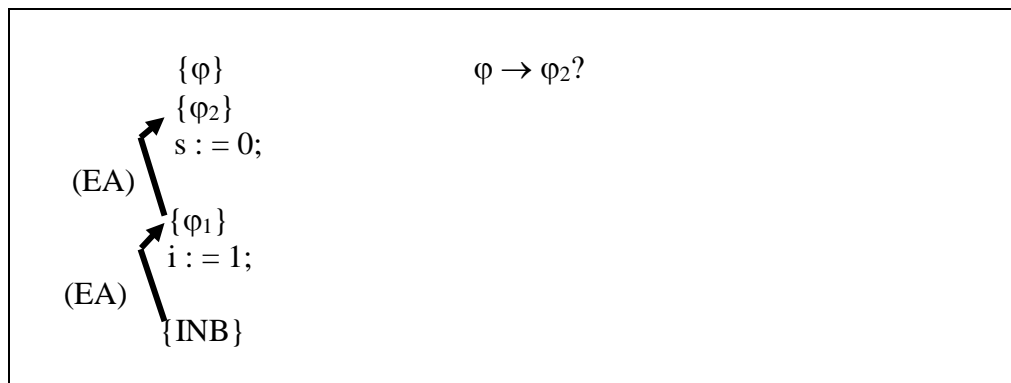


$$\begin{aligned}
 \{\varphi_1\} &\equiv \{\text{def}(1) \wedge (\text{INB})_i^1\} \equiv \\
 &\equiv \{\text{true} \wedge (1 \leq 1 \leq n+1) \wedge s = \sum_{k=1}^{1-1} A(k)\} \equiv \text{sinplifikazioa} \\
 &\equiv \{(1 \leq n+1) \wedge s = \sum_{k=1}^0 A(k)\} \equiv \text{sinplifikazioa} \\
 &\equiv \{(1 \leq n+1) \wedge s = 0\}
 \end{aligned}$$

➤ $\varphi \rightarrow \varphi_1?$



α zatia δ -gatik betetzen da baina β ez da betetzen. β betetzeko s -ri 0 esleitu beharko diogu.



$$\begin{aligned}
 \{\varphi_2\} &\equiv \{\text{def}(0) \wedge (\varphi_1)_s^0\} \equiv \\
 &\equiv \{\text{true} \wedge (1 \leq n + 1) \wedge 0 = 0\} \equiv \\
 &\equiv \text{true} \wedge (1 \leq n + 1) \wedge \text{true} \equiv \\
 &\equiv (1 \leq n + 1)
 \end{aligned}$$

➤ $\varphi \rightarrow \varphi_2?$

$$\begin{array}{c}
 n \geq 1 \\
 \downarrow? \\
 (1 \leq n + 1)
 \end{array}$$

$\varphi \rightarrow \varphi_2$ inplikazioa bete egiten da eta ondorioz hasieraketekin amaitu dugu.

- **While-aren baldintza (B)**

Hasteko $\neg B$ kalkulatu dugu "**while-a noiz geldituko da?**" galderari erantzunez. Bektorea ezkerretik eskuinera zeharkatzen ari garenez eta inbariantearen arabera i aldagaiak hartuko duen balio handiena $n + 1$ izango denez, i -ren balioa $n + 1$ denean gelditu egin beharko dugu:

$$\neg B \equiv i = n + 1$$

$$\text{Eta ondorioz, } B \equiv i \neq n + 1$$

Orain B baldintza zuzena dela egiaztatu beharko da While-aren erregelako II, IV eta V puntuak kontuan hartuz.

$$\text{II. } \text{INB} \rightarrow \text{def}(B)?$$

$$\text{INB} \rightarrow \text{def}(i \neq n + 1)?$$

$$\text{INB} \rightarrow \text{true? Bai, inplikazioaren bigarren zatian true dagoelako}$$

IV. $(INB \wedge \neg B) \rightarrow \psi$?

$$\begin{array}{c}
 (1 \leq i \leq n+1) \wedge s = \sum_{k=1}^{i-1} A(k) \wedge (i = n+1) \\
 \underbrace{\hspace{1.5cm}}_{\alpha} \quad \underbrace{\hspace{1.5cm}}_{\beta} \quad \underbrace{\hspace{1.5cm}}_{\delta} \\
 \downarrow ? \\
 s = \sum_{k=1}^n A(k) \\
 \underbrace{\hspace{1.5cm}}_{\text{bai } \beta \text{ eta } \delta\text{-gatik}}
 \end{array}$$

V. $(INB \wedge B) \rightarrow E > 0$?

$$\begin{array}{c}
 (1 \leq i \leq n+1) \wedge s = \sum_{k=1}^{i-1} A(k) \wedge (i \neq n+1) \\
 \underbrace{\hspace{1.5cm}}_{\alpha} \quad \underbrace{\hspace{1.5cm}}_{\beta} \\
 \downarrow ? \\
 n+1-i > 0 \\
 \underbrace{\hspace{1.5cm}}_{\delta}
 \end{array}$$

 δ formula α eta β -gatik betetzen da.

- Aginduak**

Aginduak kalkulatzeko While-aren erregelako III eta IV puntuetako programak hartu beharko dira kontuan

Prog 1
$\{INB \wedge B\}$
Aginduak
$\{INB\}$

Prog 2
$\{INB \wedge B \wedge E = v\}$
Aginduak
$\{E < v\}$

➤ Honako bi inplikazio hauek betetzen al diren aztertu beharko da:

✓ $(INB \wedge B) \rightarrow INB$?

✓ $(INB \wedge B \wedge E = v) \rightarrow E < v$?

$(INB \wedge B) \rightarrow INV$ inplikazioa beti beteko da, INB eta B egia badira INB ere egia izango delako.

$(INB \wedge B \wedge E = v) \rightarrow E < v$ inplikazioa ez da beteko E eta v berdinak badira E espresioa ez baita v baino txikiagoa izango.

Bigarren inplikazio hori ez denez betetzen, $E < v$ betearazteko helburuarekin agindu bat gehitu beharko dugu Prog 2 programan. Bektorea ezkerretik eskuinera zeharkatzen ari garenez, E -ren balioa txikitu dadin i aldagaiari $i + 1$ balioa eman beharko diogu eta gero esleipenaren axioma erabiliz φ_3' formula kalkulatu da.

	Prog 2
EA ↗	$\{INB \wedge B \wedge E = v\} \equiv \{(1 \leq i \leq n + 1) \wedge s = \sum_{k=1}^{i-1} A(k) \wedge (i \neq n + 1) \wedge n + 1 - i = v\}$ $\{\varphi_3'\} \equiv \{\text{def}(i + 1) \wedge (E < v)_i^{i+1}\} \equiv \{\text{true} \wedge n + 1 - (i + 1) < v\} \equiv \{n - i < v\}$ $i := i + 1;$ $\{n + 1 - i < v\}$

Orain $(INB \wedge B \wedge E = v) \rightarrow \varphi_3'$ inplikazioa betetzen al den egiaztatu beharko dugu

$$\begin{array}{c}
 (1 \leq i \leq n + 1) \wedge s = \sum_{k=1}^{i-1} A(k) \wedge (i \neq n + 1) \wedge \underbrace{n + 1 - i = v}_{\alpha} \\
 \downarrow ? \\
 \underbrace{n - i < v}_{\beta}
 \end{array}$$

Inplikazioa bete egiten da β formula egiazkoa baita α -gatik: $n + 1 - i = v$ bada, orduan $n - i = v - 1$ izango da eta $v - 1$ balioa v baino txikiagoa da.

Orain Prog 2 programa zuzena da baina Prog 1 eta Prog 2 programek agindu berdinak eduki behar dituztenez, $i := i + 1$; esleipena Prog 1 programan ipini beharko da eta gero esleipenaren axioma erabiliz esleipen horri dagokion φ_3 formula kalkulatu da eta $(INB \wedge B) \rightarrow \varphi_3$ inplikazioa betetzen al den aztertuko da.

	Prog 1
EA ↗	$\{INB \wedge B\} \equiv \{(1 \leq i \leq n + 1) \wedge s = \sum_{k=1}^{i-1} A(k) \wedge (i \neq n + 1)\}$ $\{\varphi_3\} \equiv \{\text{def}(i + 1) \wedge (INB)_i^{i+1}\}$ $i := i + 1;$ $\{INB\} \equiv \{(1 \leq i \leq n + 1) \wedge s = \sum_{k=1}^{i-1} A(k)\}$

$$\begin{aligned}
\{\varphi_3\} &\equiv \{\text{def}(i+1) \wedge (\text{INB})_i^{i+1}\} \equiv \\
&\equiv \{\text{true} \wedge (1 \leq i+1 \leq n+1) \wedge s = \sum_{k=1}^{i+1-1} A(k)\} \equiv \\
&\equiv \{(0 \leq i \leq n) \wedge s = \sum_{k=1}^i A(k)\}
\end{aligned}$$

Orain $(\text{INB} \wedge B) \rightarrow \varphi_3$ implikazioa betetzen al den azertu beharko da

$$\begin{array}{ccc}
(1 \leq i \leq n+1) \wedge s = \sum_{k=1}^{i-1} A(k) \wedge (i \neq n+1) & & \\
\begin{array}{ccc} \underbrace{\hspace{1cm}}_{\alpha} & \underbrace{\hspace{1cm}}_{\beta} & \underbrace{\hspace{1cm}}_{\delta} \end{array} & & \\
\downarrow ? & & \\
(0 \leq i \leq n) \wedge s = \sum_{k=1}^i A(k) & & \\
\begin{array}{ccc} \underbrace{\hspace{1cm}}_{\lambda} & \underbrace{\hspace{1cm}}_{\pi} & \end{array} & &
\end{array}$$

- λ zatia α eta δ -gatik betetzen da.
- β -gatik s aldagaian $A(1..i-1)$ zatiko elementuen batura daukagula badakigu baina π formulari s aldagaian $A(1..i)$ zatiko elementuen batura al daukagun galdetzen zaigu eta erantzuna ezezkoa da. Beraz $(\text{INV} \wedge B) \rightarrow \varphi_3$ implikazioa ez da betetzen. Une honetan helburua φ_3 betearaztea denez, hau da heldurua s aldagaian $A(1..i)$ zatiko elementuen batura edukitzea denez, s -ri $A(i)$ balioa gehitu beharko zaio.

$$s := s + A(i);$$

Esleipena ipini ondoren φ_4 kalkulatu da eta $(\text{INB} \wedge B) \rightarrow \varphi_4$ implikazioa betetzen al den aztertuko da

	Prog 1
EA ↗	$\{\text{INB} \wedge B\} \equiv \{(1 \leq i \leq n+1) \wedge s = \sum_{k=1}^{i-1} A(k) \wedge (i \neq n+1)\}$ $\{\varphi_4\} \equiv \{\text{def}(s + A(i)) \wedge (\varphi_3)_s^{s+A(i)}\}$ $s := s + A(i);$ $\{\varphi_3\} \equiv \{(0 \leq i \leq n) \wedge s = \sum_{k=1}^i A(k)\}$ $i := i + 1;$ $\{\text{INB}\} \equiv \{(1 \leq i \leq n+1) \wedge s = \sum_{k=1}^{i-1} A(k)\}$

$$\begin{aligned}
\{\varphi_4\} &\equiv \{\text{def}(s + A(i)) \wedge (\varphi_3)_s^{s + A(i)}\} \equiv \\
&\equiv \{(1 \leq i \leq n) \wedge (0 \leq i \leq n) \wedge s + A(i) = \sum_{k=1}^i A(k)\} \equiv \\
&\equiv \{(1 \leq i \leq n) \wedge s + A(i) = \sum_{k=1}^i A(k)\}
\end{aligned}$$

$(INB \wedge B) \rightarrow \varphi_4$ betezen al da?

$$\begin{array}{ccc}
(1 \leq i \leq n+1) \wedge s = \sum_{k=1}^{i-1} A(k) \wedge (i \neq n+1) & & \\
\begin{array}{ccc} \underbrace{\hspace{1.5cm}}_{\alpha} & \underbrace{\hspace{1.5cm}}_{\beta} & \underbrace{\hspace{1.5cm}}_{\delta} \end{array} & & \\
\downarrow ? & & \\
(1 \leq i \leq n) \wedge s + A(i) = \sum_{k=1}^i A(k) & & \\
\begin{array}{ccc} \underbrace{\hspace{1.5cm}}_{\lambda} & \underbrace{\hspace{1.5cm}}_{\pi} \end{array} & &
\end{array}$$

- λ zatia α eta δ -gatik betetzen da.
- π zatia β -gatik betetzen da.
- Beraz $(INB \wedge B) \rightarrow \varphi_4$ inplikazioa bete egiten da.

Orain Prog 1 programa zuzena da baina Prog 1 eta Prog 2 programek agindu berdinak izan behar dituztenez, Prog 2 programan $s := s + A(i)$; esleipena ipini behar da eta φ_4' formula kalkulatu behar da esleipenaren axioma erabiliz. Bukatzeko $(INB \wedge B \wedge E = v) \rightarrow \varphi_4'$ inplikazioa betetzen al den aztertu beharko da.

	Prog 2
EA ↗	$\{INB \wedge B \wedge E = v\} \equiv \{(1 \leq i \leq n+1) \wedge s = \sum_{k=1}^{i-1} A(k) \wedge (i \neq n+1) \wedge n+1-i = v\}$ $\{\varphi_4'\} \equiv \{\text{def}(s + A(i)) \wedge (\varphi_3')_s^{s + A(i)}\}$ $s := s + A(i);$ $\{\varphi_3'\} \equiv \{n-i < v\}$ $i := i + 1;$ $\{E < v\} \equiv \{n+1-i < v\}$

$$\begin{aligned}
\{\varphi_4'\} &\equiv \{\text{def}(s + A(i)) \wedge (\varphi_3')_s^{s + A(i)}\} \equiv \\
&\equiv \{(1 \leq i \leq n) \wedge n-i < v\}
\end{aligned}$$

$(INB \wedge B \wedge E = v) \rightarrow \varphi_4'$ inplikazioa betetzen al da?

$$\begin{array}{c}
 \underbrace{(1 \leq i \leq n+1)}_{\alpha} \wedge s = \sum_{k=1}^{i-1} A(k) \wedge \underbrace{(i \neq n+1)}_{\beta} \wedge \underbrace{n+1-i=v}_{\delta} \\
 \downarrow ? \\
 \underbrace{(1 \leq i \leq n)}_{\alpha \text{ eta } \beta\text{-gatik}} \wedge \underbrace{n-i < v}_{\delta\text{-gatik, } \delta\text{-gatik } n-i=v-1 \text{ baita eta } v-1 \text{ balioa } v \text{ baino txikiagoa da}}
 \end{array}$$

$(INB \wedge B) \rightarrow \varphi_4$ eta $(INB \wedge B \wedge E = v) \rightarrow \varphi_4'$ inplikazioak bete egin direnez programaren eratorpena bukatu da. Eratorritako programa honako hau da:

```

{φ}
{φ2}
s := 0;
{φ1}
i := 1;

while {INB} i ≠ n + 1 loop
    {φ4} {φ4'}
    s := s + A(i);
    {φ3} {φ3'}
    i := i + 1;
end loop;

{ψ}
E

```

Programa hori zuzena da While-aren erregela jarraituz eraiki dugulako eta gainera metodoaren bidez programa dokumentatzeko balio duten $\{\varphi_1\}$, $\{\varphi_2\}$, $\{\varphi_3\}$, $\{\varphi_3'\}$, $\{\varphi_4\}$ eta $\{\varphi_4'\}$ formulak kalkulatu ditugu.

4.2.2. BEKTORE BATEKO ELEMENTUEN BATURA KALKULATU (4. ARIKETA)

$A(1..n)$ bektorea emanda, $A(1..n)$ bektoreko elementuen batura s aldagaian kalkulatzeko duen programa eratorri Hoare-ren kalkuluko Esleipenaren Axioma eta While-aren Erregela erabiliz. Eratorritako programak INB inbariantea, E espresioa eta φ eta ψ formulen bidez emandako aurre-ondoetako espezifikazioarekiko zuzena izan beharko du.

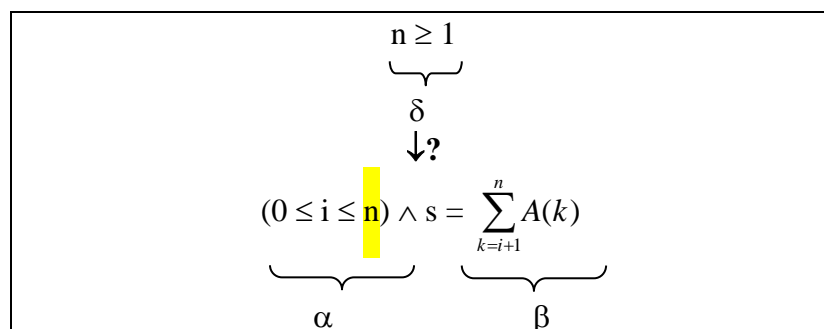
$\{\varphi\} \equiv \{n \geq 1\}$ Hasieraketak? $\{INB\}$ <u>while</u> $\{INB\} \{E\} B?$ <u>loop</u> Aginduak? <u>end loop;</u> $\{\psi\} \equiv \{s = \sum_{k=1}^n A(k)\}$
$\{INB\} \equiv \{(0 \leq i \leq n) \wedge s = \sum_{k=i+1}^n A(k)\}$ $E = i$

Hasteko bektorea ezkerretik eskuinera edo eskuinetik ezkerrera zeharkatu behar al den erabaki behar da. E espresioa "indizea – beheko muga" erakoa denez ($i - 0$), bektorea eskuinetik ezkerrera zeharkatu beharko da.

- **Hasieraketak**

Hasieraketak kalkulatzeko While-aren erregelako lehenengo puntua hartu behar da kontuan, hau da, φ formulak inbariantea inplikatzeko al duen begiratu behar da. Inplikatzeko badu, ez da ezer hasieratu behar. Ez badu inplikatzeko, hasieraketaren bat beharko da. Inbariantea adieraziko digu zein aldagai eta nola hasieratu. Aldagai baten hasieraketa zein izango den kalkulatu ondoren, Esleipenaren Axioma erabiliz hasieraketa horri dagokion formula kalkulatu da eta φ formulak formula berri hori inplikatzeko al duen aztertuko da. Hasieraketak gehitzeko prozesua φ formulak inplikatzeko duen formula bat lortu arte jarraitu beharko da.

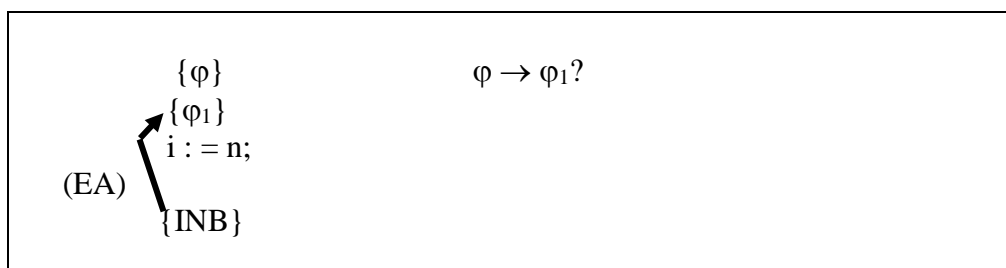
$\varphi \rightarrow INB?$



Inplikazio hori ez da betetzen inplikazioaren lehenengo zatian (δ zatian) ez baitago i eta s aldagaiek α eta β formulek diotena betetzen dutela ziurtatzeko erabili dezakegun informaziorik. Helburua inplikazioa betetzea denez, hau da, α eta β betetzea denez, inplikazio hori betearaziko duten balioekin hasieratu beharko dira i eta s . β formulari i eta s agertzen direnez eta α formulari i bakarrik agertzen denez, α kontuan hartuz i nola hasieratu erabakiko dugu.

Bektorea eskuinetik ezkerrera zeharkatu behar denez, α betetzeko i -ri n esleitzea nahikoa da. Beraz α formulari begiratu i aldagaiak har dezakeen balio txikiena esleituiko diogu.

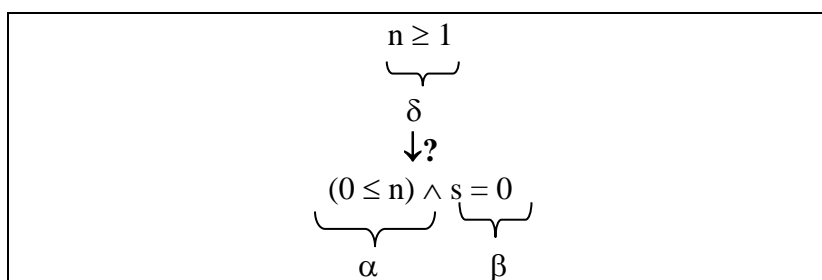
Orain Esleipenaren Axiomari jarraituz $\{\varphi_1\}$ formula kalkulatu dugu



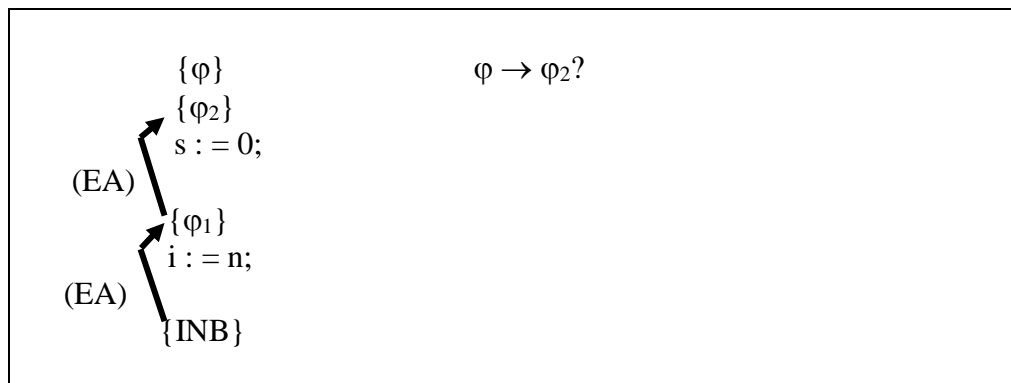
$$\begin{aligned} \{\varphi_1\} &\equiv \{\text{def}(n) \wedge (\text{INB})_i^n\} \equiv \\ &\equiv \{\text{true} \wedge (0 \leq n \leq n) \wedge s = \sum_{k=n+1}^n A(k)\} \equiv \text{simplifikazioa} \\ &\equiv \{(0 \leq n) \wedge s = 0\} \end{aligned}$$

Simplifikatzerakoan batukariko goiko muga behekoa baino txikiagoa denez, batukaria 0 izango da.

➤ $\varphi \rightarrow \varphi_1?$



α zatia δ -gatik betetzen da baina β ez da betetzen. β betetzeko s -ri 0 esleitu beharko diogu.



$$\begin{aligned}
 \{\varphi_2\} &\equiv \{\text{def}(0) \wedge (\varphi_1)_s^0\} \equiv \\
 &\equiv \{\text{true} \wedge (0 \leq n) \wedge 0 = 0\} \equiv \\
 &\equiv \text{true} \wedge (1 \leq n + 1) \wedge \text{true} \equiv \\
 &\equiv (0 \leq n)
 \end{aligned}$$

➤ $\varphi \rightarrow \varphi_2?$

$$\begin{array}{c}
 n \geq 1 \\
 \downarrow? \\
 (0 \leq n)
 \end{array}$$

$\varphi \rightarrow \varphi_2$ inplikazioa bete egiten da eta ondorioz hasieraketekin amaitu dugu.

- **While-aren baldintza (B)**

Hasteko $\neg B$ kalkulatu dugu "**while-a noiz geldituko da?**" galderari erantzunez. Bektorea eskuinetik ezkerre zeharkatzen ari garenez eta inbariantearen arabera i aldagaiak hartuko duen balio txikiena 0 izango denez, i -ren balioa 0 denean gelditu egin beharko dugu:

$$\neg B \equiv i = 0$$

Eta ondorioz, $B \equiv i \neq 0$

Orain B baldintza zuzena dela egiaztatu beharko da While-aren erregelako II, IV eta V puntuak kontuan hartuz.

$$\text{II. } \text{INB} \rightarrow \text{def}(B)?$$

$$\text{INB} \rightarrow \text{def}(i \neq 0)?$$

$\text{INB} \rightarrow \text{true}$? Bai, inplikazioaren bigarren zatian true dagoelako

IV. $(INB \wedge \neg B) \rightarrow \psi?$

$$\begin{array}{c}
 (0 \leq i \leq n) \wedge s = \sum_{k=i+1}^n A(k) \wedge (i = 0) \\
 \underbrace{\hspace{1cm}}_{\alpha} \quad \underbrace{\hspace{1cm}}_{\beta} \quad \underbrace{\hspace{1cm}}_{\delta} \\
 \downarrow? \\
 s = \sum_{k=1}^n A(k) \\
 \underbrace{\hspace{1cm}}_{\text{bai } \beta \text{ eta } \delta\text{-gatik}}
 \end{array}$$

V. $(INB \wedge B) \rightarrow E > 0?$

$$\begin{array}{c}
 \{(0 \leq i \leq n) \wedge s = \sum_{k=i+1}^n A(k)\} \wedge (i \neq 0) \\
 \underbrace{\hspace{1cm}}_{\alpha} \quad \underbrace{\hspace{1cm}}_{\beta} \\
 \downarrow? \\
 \underbrace{\hspace{1cm}}_{i > 0} \\
 \delta
 \end{array}$$

 δ formula α eta β -gatik betetzen da.

- Aginduak**

Aginduak kalkulatzeko While-aren erregelako III eta IV puntuetako programak hartu beharko dira kontuan

Prog 1
$\{INB \wedge B\}$
Aginduak?
$\{INB\}$

Prog 2
$\{INB \wedge B \wedge E = v\}$
Aginduak?
$\{E < v\}$

➤ Honako bi inplikazio hauek betetzen al diren aztertu beharko da:

✓ $(INB \wedge B) \rightarrow INB?$

✓ $(INB \wedge B \wedge E = v) \rightarrow E < v?$

$(INB \wedge B) \rightarrow INV$ inplikazioa beti beteko da, INB eta B egia badira INB ere egia izango delako.

$(INB \wedge B \wedge E = v) \rightarrow E < v$ inplikazioa ez da beteko E eta v berdinak badira E espresioa ez baita v baino txikiagoa izango.

Bigarren inplikazio hori ez denez betetzen, $E < v$ betearazteko helburuarekin agindu bat gehitu beharko dugu Prog 2 programan. Bektorea eskuinetik ezkerrera zeharkatzen ari garenez, E -ren balioa txikitu dadin i aldagaiari $i - 1$ balioa eman beharko diogu eta gero esleipenaren axioma erabiliz φ_3' formula kalkulatu da.

	Prog 2
EA ↗	$\{INB \wedge B \wedge E = v\} \equiv \{(0 \leq i \leq n) \wedge s = \sum_{k=i+1}^n A(k) \wedge (i \neq 0) \wedge i = v\}$ $\{\varphi_3'\} \equiv \{\text{def}(i - 1) \wedge (E < v)_i^{i-1}\} \equiv \{\text{true} \wedge i - 1 < v\} \equiv \{i - 1 < v\}$ $i := i - 1;$ $\{i < v\}$

Orain $(INB \wedge B \wedge E = v) \rightarrow \varphi_3'$ inplikazioa betetzen al den egiaztatu beharko dugu

$$\begin{array}{c}
 (0 \leq i \leq n) \wedge s = \sum_{k=i+1}^n A(k) \wedge (i \neq 0) \wedge \underbrace{i = v}_{\alpha} \\
 \downarrow ? \\
 \underbrace{i - 1 < v}_{\beta}
 \end{array}$$

Inplikazioa bete egiten da β formula egiazkoa baita α -gatik: $i = v$ bada, orduan $i - 1 = v - 1$ izango da eta $v - 1$ balioa v baino txikiagoa da.

Orain Prog 2 programa zuzena da baina Prog 1 eta Prog 2 programek agindu berdinak eduki behar dituztenez, $i := i - 1$; esleipena Prog 1 programan ipini beharko da eta gero esleipenaren axioma erabiliz esleipen horri dagokion φ_3 formula kalkulatu da eta $(INB \wedge B) \rightarrow \varphi_3$ inplikazioa betetzen al den aztertuko da.

	Prog 1
EA ↗	$\{INB \wedge B\} \equiv \{(0 \leq i \leq n) \wedge s = \sum_{k=i+1}^n A(k) \wedge (i \neq 0)\}$ $\{\varphi_3\} \equiv \{\text{def}(i - 1) \wedge (INB)_i^{i-1}\}$ $i := i - 1;$ $\{INB\} \equiv \{(0 \leq i \leq n) \wedge s = \sum_{k=i+1}^n A(k)\}$

$$\begin{aligned}
\{\varphi_3\} &\equiv \{\text{def}(i-1) \wedge (\text{INB})_i^{i-1}\} \equiv \\
&\equiv \{\text{true} \wedge (0 \leq i-1 \leq n) \wedge s = \sum_{k=i-1+1}^n A(k)\} \equiv \\
&\equiv \{(1 \leq i \leq n+1) \wedge s = \sum_{k=i}^n A(k)\}
\end{aligned}$$

Orain $(\text{INB} \wedge B) \rightarrow \varphi_3$ inplikazioa betetzen al den azertu beharko da

$$\begin{array}{ccc}
(0 \leq i \leq n) \wedge s = \sum_{k=i+1}^n A(k) \wedge (i \neq 0) & & \\
\begin{array}{ccc} \underbrace{\hspace{1cm}} & \underbrace{\hspace{1cm}} & \underbrace{\hspace{1cm}} \\ \alpha & \beta & \delta \end{array} & & \\
\downarrow ? & & \\
(1 \leq i \leq n+1) \wedge s = \sum_{k=i}^n A(k) & & \\
\begin{array}{ccc} \underbrace{\hspace{1cm}} & \underbrace{\hspace{1cm}} & \\ \lambda & \pi & \end{array} & &
\end{array}$$

- λ zatia α eta δ -gatik betetzen da.
- β -gatik s aldagaian $A(i+1..n)$ zatiko elementuen batura daukagula badakigu baina π formulari s aldagaian $A(i..n)$ zatiko elementuen batura al daukagun galdetzen zaigu eta erantzuna ezezkoa da. Beraz $(\text{INB} \wedge B) \rightarrow \varphi_3$ inplikazioa ez da betetzen. Une honetan helburua φ_3 betearaztea denez, hau da heldurua s aldagaian $A(i..n)$ zatiko elementuen batura edukitzea denez, s -ri $A(i)$ balioa gehitu beharko zaio.

$$s := s + A(i);$$

Esleipena ipini ondoren φ_4 kalkulatu da eta $(\text{INB} \wedge B) \rightarrow \varphi_4$ inplikazioa betetzen al den aztertuko da

	Prog 1
EA ↗	$\{\text{INB} \wedge B\} \equiv \{(0 \leq i \leq n) \wedge s = \sum_{k=i+1}^n A(k) \wedge (i \neq 0)\}$ $\{\varphi_4\} \equiv \{\text{def}(s + A(i)) \wedge (\varphi_3)_{s+A(i)}\}$ $s := s + A(i);$ $\{\varphi_3\} \equiv \{(1 \leq i \leq n+1) \wedge s = \sum_{k=i}^n A(k)\}$ $i := i - 1;$ $\{\text{INB}\} \equiv \{(0 \leq i \leq n) \wedge s = \sum_{k=i+1}^n A(k)\}$

$$\begin{aligned}
\{\varphi_4\} &\equiv \{\text{def}(s + A(i)) \wedge (\varphi_3)_s^{s+A(i)}\} \equiv \\
&\equiv \{(1 \leq i \leq n) \wedge (1 \leq i \leq n+1) \wedge s + A(i) = \sum_{k=i}^n A(k)\} \equiv \\
&\equiv \{(1 \leq i \leq n) \wedge s + A(i) = \sum_{k=i}^n A(k)\}
\end{aligned}$$

$(INB \wedge B) \rightarrow \varphi_4$ betezen al da?

$$\begin{array}{ccc}
\underbrace{(0 \leq i \leq n)}_{\alpha} \wedge \underbrace{s = \sum_{k=i+1}^n A(k)}_{\beta} \wedge \underbrace{(i \neq 0)}_{\delta} \\
\downarrow ? \\
\underbrace{(1 \leq i \leq n)}_{\lambda} \wedge \underbrace{s + A(i) = \sum_{k=i}^n A(k)}_{\pi}
\end{array}$$

- λ zatia α eta δ -gatik betetzen da.
- π zatia β -gatik betetzen da.
- Beraz $(INB \wedge B) \rightarrow \varphi_4$ inplikazioa bete egiten da.

Orain Prog 1 programa zuzena da baina Prog 1 eta Prog 2 programek agindu berdinak izan behar dituztenez, Prog 2 programan $s := s + A(i)$; esleipena ipini behar da eta φ_4' formula kalkulatu behar da esleipenaren axioma erabiliz. Bukatzeko $(INB \wedge B \wedge E = v) \rightarrow \varphi_4'$ inplikazioa betetzen al den aztertu beharko da.

	Prog 2
EA ↗	$\{INB \wedge B \wedge E = v\} \equiv \{(0 \leq i \leq n) \wedge s = \sum_{k=i+1}^n A(k) \wedge (i \neq 0) \wedge i = v\}$ $\{\varphi_4'\} \equiv \{\text{def}(s + A(i)) \wedge (\varphi_3')_s^{s+A(i)}\}$ $s := s + A(i);$ $\{\varphi_3'\} \equiv \{i - 1 < v\}$ $i := i - 1;$ $\{E < v\} \equiv \{i < v\}$

$$\begin{aligned}
\{\varphi_4'\} &\equiv \{\text{def}(s + A(i)) \wedge (\varphi_3')_s^{s+A(i)}\} \equiv \\
&\equiv \{(1 \leq i \leq n) \wedge i - 1 < v\}
\end{aligned}$$

$(INB \wedge B \wedge E = v) \rightarrow \varphi_4'$ inplikazioa betetzen al da?

$$\begin{array}{c}
 \underbrace{(0 \leq i \leq n)}_{\alpha} \wedge s = \sum_{k=i+1}^n A(k) \wedge \underbrace{(i \neq 0)}_{\beta} \wedge \underbrace{i = v}_{\delta} \\
 \downarrow ? \\
 \underbrace{(1 \leq i \leq n)}_{\alpha \text{ eta } \beta\text{-gatik}} \wedge \underbrace{i - 1 < v}_{\delta\text{-gatik, } \delta\text{-gatik } i - 1 = v - 1 \text{ baita eta } v - 1 \text{ balioa } v \text{ baino txikiagoa da}}
 \end{array}$$

$(INB \wedge B) \rightarrow \varphi_4$ eta $(INB \wedge B \wedge E = v) \rightarrow \varphi_4'$ inplikazioak bete egin direnez programaren eratorpena bukatu da. Eratorritako programa honako hau da:

```

{φ}
{φ2}
s := 0;
{φ1}
i := n;

while {INB} i ≠ 0 loop
    {φ4} {φ4'}
    s := s + A(i);
    {φ3} {φ3'}
    i := i - 1;
end loop;

{ψ}
E

```

Programa hori zuzena da While-aren erregela jarraituz eraiki dugulako eta gainera metodoaren bidez programa dokumentatzeko balio duten $\{\varphi_1\}$, $\{\varphi_2\}$, $\{\varphi_3\}$, $\{\varphi_3'\}$, $\{\varphi_4\}$ eta $\{\varphi_4'\}$ formulak kalkulatu ditugu.

4.2.3. C ALDAGAIAN 0 ITZULI A(1..N) TAULAKO POSIZIO DENETAN 0 BADAGO (7. ARIKETA)

Bakarrik negatiboak ez diren zenbakiak dituen $A(1..n)$ bektorea emanda, $A(1..n)$ taulako posizio denetan 0 baldin badago, osoa den c aldagaian 0 itzuliko duen eta $A(1..n)$ taulako posizio denetan 0 ez badago, c aldagaian 0 itzuliko ez duen programa eratorri Hoare-ren kalkuluko Esleipenaren Axioma eta While-aren Erregela erabiliz. Eratorritako programak INB inbariantea, E espresioa eta ϕ eta ψ formulen bidez emandako aurre-ondoetako espezifikazioarekiko zuzena izan beharko du.

$\{\phi\} \equiv \{n \geq 1 \wedge \text{ezneg}(A(1..n))\}$ Hasieraketak? $\{\text{INB}\}$ while $\{\text{INB}\} \{E\} B?$ loop Aginduak? end loop; $\{\psi\} \equiv \{(c = 0) \leftrightarrow \forall k(1 \leq k \leq n \rightarrow A(k) = 0)\}$
$\{\text{INB}\} \equiv \{\text{ezneg}(A(1..n)) \wedge (0 \leq i \leq n) \wedge c = \sum_{k=1}^i A(k)\}$ $E = n - i$ $\text{ezneg}(A(1..n)) \equiv \{\forall k(1 \leq k \leq n \rightarrow A(k) \geq 0)\}$

Hasteko bektorea ezkerretik eskuinera edo eskuinetik ezkerrera zeharkatu behar al den erabaki behar da. E espresioa "goiko muka – indizea" erakoa denez, bektorea ezkerretik eskuinera zeharkatu beharko da.

- **Hasieraketak**

Hasieraketak kalkulatzeko While-aren erregelako lehenengo puntua hartu behar da kontuan, hau da, ϕ formulak inbariantea inplikatzeko al duen begiratu behar da. Inplikatzeko badu, ez da ezer hasieratu behar. Ez badu inplikatzeko, hasieraketaren bat beharko da. Inbarianteak adieraziko digu zein aldagai eta nola hasieratu. Aldagai baten hasieraketa zein izango den kalkulatu ondoren, Esleipenaren Axioma erabiliz hasieraketa horri dagokion formula kalkulatu da eta ϕ formulak formula berri hori inplikatzeko al duen aztertuko da. Hasieraketak gehitzeko prozesua ϕ formulak inplikatzeko duen formula bat lortu arte jarraitu beharko da.

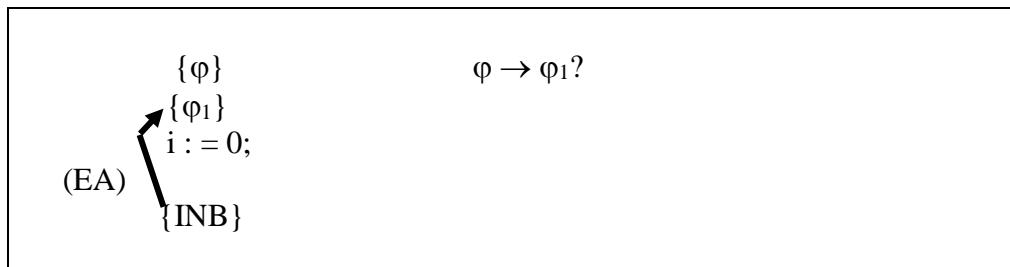
$\varphi \rightarrow \text{INB?}$

$$\begin{array}{c}
 \underbrace{n \geq 1}_{\alpha} \wedge \underbrace{\text{ezneg}(A(1..n))}_{\beta} \\
 \downarrow ? \\
 \underbrace{\text{ezneg}(A(1..n))}_{\delta_1} \wedge \underbrace{(0 \leq i \leq n)}_{\delta_2} \wedge c = \sum_{k=1}^i A(k) \\
 \underbrace{\hspace{10em}}_{\delta} \\
 \text{Bai } \beta\text{-gatik} \qquad \qquad \text{Ez da betetzen}
 \end{array}$$

Inplikazio hori ez da betetzen inplikazioaren lehenengo zatian ($\alpha \wedge \beta$ zatian) ez baitago i eta c aldagaiek δ_1 eta δ_2 formulek diotena betetzen dutela ziurtatzeko erabili dezakegun informaziorik. Helburua inplikazioa betetzea denez, hau da, δ_1 eta δ_2 betetzea denez, inplikazio hori betearaziko duten balioekin hasieratu beharko dira i eta c . δ_2 formulari i eta c agertzen direnez eta δ_1 formulari i bakarrik agertzen denez, δ_1 kontuan hartuz i nola hasieratu erabakiko dugu.

Bektorea ezkerretik eskuinera zeharkatu behar denez, δ_1 betetzeko i -ri 0 esleitzea nahikoa da. Beraz δ_1 formulari begiratzuz i aldagaiak har dezakeen balio txikiena esleituko diogu.

Orain Esleipenaren Axioma jarraituz $\{\varphi_1\}$ formula kalkulatu dugu.

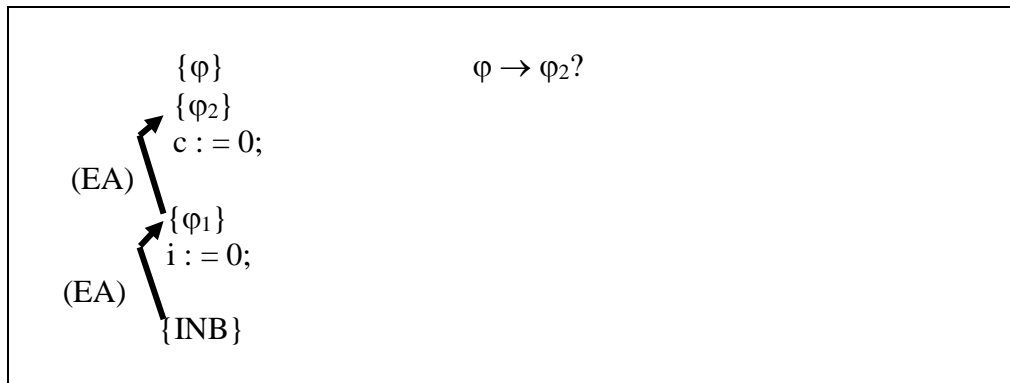


$$\begin{aligned}
 \{\varphi_1\} &\equiv \{\text{def}(0) \wedge (\text{INB})_i^0\} \equiv \\
 &\equiv \{\text{true} \wedge \text{ezneg}(A(1..n)) \wedge (0 \leq 0 \leq n) \wedge c = \sum_{k=1}^0 A(k)\} \equiv \text{sinplifikazioa} \\
 &\equiv \{\text{ezneg}(A(1..n)) \wedge (0 \leq n) \wedge c = \sum_{k=1}^0 A(k)\} \equiv \text{sinplifikazioa} \\
 &\equiv \{\text{ezneg}(A(1..n)) \wedge (0 \leq n) \wedge c = 0\}
 \end{aligned}$$

➤ $\varphi \rightarrow \varphi_1?$

$$\begin{array}{c}
 \underbrace{n \geq 1}_{\alpha} \wedge \underbrace{\text{ezneg}(A(1..n))}_{\beta} \\
 \downarrow? \\
 \underbrace{\text{ezneg}(A(1..n))}_{\text{Bai, } \beta\text{-gatik}} \wedge \underbrace{(0 \leq n)}_{\text{Bai, } \alpha\text{-gatik}} \wedge \underbrace{c = 0}_{\text{Ez da betetzen}}
 \end{array}$$

α eta β betetzeak ez du ziurtatzen $c = 0$ betetzea, eta $c = 0$ bete dadin c -ri 0 esleitu beharko diogu.



$$\begin{aligned}
 \{ \varphi_2 \} &\equiv \{ \text{def}(0) \wedge (\varphi_1)_c^0 \} \equiv \\
 &\equiv \{ \text{true} \wedge \text{ezneg}(A(1..n)) \wedge (0 \leq n) \wedge 0 = 0 \} \equiv \\
 &\equiv \{ \text{true} \wedge \text{ezneg}(A(1..n)) \wedge (0 \leq n) \wedge \text{true} \} \equiv \\
 &\equiv \{ \text{ezneg}(A(1..n)) \wedge (0 \leq n) \}
 \end{aligned}$$

➤ $\varphi \rightarrow \varphi_2?$

$$\begin{array}{c}
 \underbrace{n \geq 1}_{\alpha} \wedge \underbrace{\text{ezneg}(A(1..n))}_{\beta} \\
 \downarrow? \\
 \underbrace{\text{ezneg}(A(1..n))}_{\text{Bai, } \beta\text{-gatik}} \wedge \underbrace{(0 \leq n)}_{\text{Bai, } \alpha\text{-gatik}}
 \end{array}$$

$\varphi \rightarrow \varphi_2$ implikazioa bete egiten da eta ondorioz hasieraketekin bukatu dugu.

- **While-aren baldintza (B)**

Hasteko $\neg B$ kalkulatu dugu "**while-a noiz geldituko da?**" galderari erantzunez. Bektorea ezkerretik eskuinera zeharkatzen ari garenez eta inbariantearen arabera i aldagaiak hartuko duen balio handiena n izango denez, i -ren balioa n denean gelditu egin beharko dugu:

$$\neg B \equiv i = n$$

Eta ondorioz, $B \equiv i \neq n$

Orain B baldintza zuzena dela egiaztatu beharko da While-aren erregelako II, IV eta V puntuak kontuan hartuz.

II. $INB \rightarrow \text{def}(B)?$

$INB \rightarrow \text{def}(i \neq n)?$

$INB \rightarrow \text{true?}$ Bai, inplikazioaren bigarren zatian true dagoelako.

IV. $(INB \wedge \neg B) \rightarrow \psi?$

$$\underbrace{\text{ezneg}(A(1..n))}_{\alpha} \wedge \underbrace{(0 \leq i \leq n)}_{\beta} \wedge c = \underbrace{\sum_{k=1}^i A(k)}_{\delta} \wedge \underbrace{(i = n)}_{\gamma}$$

$\downarrow?$

$$(c = 0) \leftrightarrow \forall k(1 \leq k \leq n \rightarrow A(k) = 0)$$

Bai, α , β , δ eta γ -gatik

δ eta γ -gatik c aldagaian $A(1..n)$ bektoreko elementuen batura daukagu eta α -gatik badakigu $A(1..n)$ bektorean bakarrik zeroak eta batekoak daudela eta ondorioz c -ren balioa 0 izango da $A(1..n)$ -ko posizio denetan 0 baldin badago eta bestela c ez da 0 izango.

V. $INB \wedge B \rightarrow E > 0?$

$$\text{ezneg}(A(1..n)) \wedge \underbrace{(0 \leq i \leq n)}_{\alpha} \wedge c = \sum_{k=1}^i A(k) \wedge \underbrace{(i \neq n)}_{\beta}$$

$\downarrow?$

$$\underbrace{n - i > 0}_{\beta}$$

Bai, α eta β -gatik.

- **Aginduak**

Aginduak kalkulatzeko While-aren erregelako III eta IV puntuetako programak hartu beharko dira kontuan

Prog 1	Prog 2
{INB \wedge B}	{INB \wedge B \wedge E = v}
Aginduak?	Aginduak?
{INB}	{E < v}

- Honako bi inplikazio hauek betetzen al diren aztertu beharko da
- ✓ (INB \wedge B) \rightarrow INB?
 - ✓ (INB \wedge B \wedge E = v) \rightarrow E < v?

(INB \wedge B) \rightarrow INB inplikazioa beti beteko da, INB eta B egia badira INB ere egia izango delako.

(INB \wedge B \wedge E = v) \rightarrow E < v inplikazioa ez da beteko E eta v berdinak badira E espresioa ez baita v baino txikiagoa izango.

Bigarren inplikazio hori ez denez betetzen, E < v betearazteko helburuarekin agindu bat gehitu beharko dugu Prog 2 programan. Bektorea ezkerretik eskuinera zeharkatzen ari garenez, E-ren balioa txikitu dadin i aldagaiari i + 1 balioa eman beharko diogu eta gero esleipenaren axioma erabiliz φ_3' formula kalkulatu da.

	Prog 2
	{INB \wedge B \wedge E = v} \equiv \equiv {ezneg(A(1..n)) \wedge (0 \leq i \leq n) \wedge c = $\sum_{k=1}^i$ A(k) \wedge (i \neq n + 1) \wedge n - i = v}
EA ↗	{ φ_3' } \equiv {def(i + 1) \wedge (E < v) $_{i+1}$ } \equiv {true \wedge n - (i + 1) < v} \equiv {n - i - 1 < v} i := i + 1;
	{n - i < v}

Orain (INB \wedge B \wedge E = v) \rightarrow φ_3' inplikazioa betetzen al den egiaztatu beharko dugu:

$$\begin{aligned}
 & \text{ezneg}(A(1..n)) \wedge (0 \leq i \leq n) \wedge c = \sum_{k=1}^i A(k) \wedge (i \neq n) \wedge \underbrace{n - i = v}_{\alpha} \\
 & \quad \downarrow ? \\
 & \quad \underbrace{n - i - 1 < v}_{\beta}
 \end{aligned}$$

Inplikazioa bete egiten da β formula egiazkoa baita α -gatik: $n - i = v$ bada, orduan $n - i - 1 = v - 1$ izango da eta $v - 1$ balioa v baino txikiagoa da.

Orain Prog 2 programa zuzena da baina Prog 1 eta Prog 2 programek agindu berdinak eduki behar dituztenez, $i := i + 1$; esleipena Prog 1 programan ipini beharko da eta gero esleipenaren axioma erabiliz esleipen horri dagokion φ_3 formula kalkulatu da eta $(INB \wedge B) \rightarrow \varphi_3$ inplikazioa betetzen al den aztertuko da.

	Prog 1
EA ↗	$\{INB \wedge B\} \equiv \{ezneg(A(1..n)) \wedge (0 \leq i \leq n) \wedge c = \sum_{k=1}^i A(k) \wedge (i \neq n)\}$ $\{\varphi_3\} \equiv \{def(i + 1) \wedge (INB)_i^{i+1}\}$ $i := i + 1;$ $\{INB\} \equiv \{ezneg(A(1..n)) \wedge (0 \leq i \leq n) \wedge c = \sum_{k=1}^i A(k)\}$

$$\begin{aligned}
\{\varphi_3\} &\equiv \{def(i + 1) \wedge (INB)_i^{i+1}\} \equiv \\
&\equiv \{\text{true} \wedge ezneg(A(1..n)) \wedge (0 \leq i + 1 \leq n) \wedge c = \sum_{k=1}^{i+1} A(k)\} \equiv \\
&\equiv \{ezneg(A(1..n)) \wedge (0 \leq i + 1 \leq n) \wedge c = \sum_{k=1}^{i+1} A(k)\} \\
&\equiv \{ezneg(A(1..n)) \wedge (-1 \leq i \leq n - 1) \wedge c = \sum_{k=1}^{i+1} A(k)\}
\end{aligned}$$

Orain $(INB \wedge B) \rightarrow \varphi_3$ inplikazioa betetzen al den azertu beharko da:

$$\begin{array}{ccccccc}
\underbrace{ezneg(A(1..n))}_{\alpha} & \underbrace{\wedge (0 \leq i \leq n)}_{\beta} & \underbrace{\wedge c = \sum_{k=1}^i A(k)}_{\delta} & \underbrace{\wedge (i \neq n)}_{\gamma} & & & \\
& & \downarrow ? & & & & \\
\underbrace{ezneg(A(1..n))}_{\alpha\text{-gatik}} & \underbrace{\wedge (-1 \leq i \leq n - 1)}_{\beta \text{ eta } \gamma\text{-gatik}} & \underbrace{\wedge c = \sum_{k=1}^{i+1} A(k)}_{\pi \text{ (Ez da betetzen)}}
\end{array}$$

β -gatik c aldagaian $A(1..i)$ zatiko elementuen batura daukagula badakigu baina π formulari c aldagaian $A(1..i + 1)$ zatiko elementuen batura al daukagun galdetzen zaigu eta erantzuna ezezkoa da. Beraz $(INB \wedge B) \rightarrow \varphi_3$ inplikazioa ez da betetzen. Une honetan helburua φ_3 betearaztea denez, hau da heldurua c aldagaian $A(1..i + 1)$ zatiko elementuen batura edukitzea denez, c -ri $A(i + 1)$ balioa gehitu beharko zaio.

$$c := c + A(i + 1);$$

Esleipena ipini ondoren φ_4 kalkulatu da eta $(INV \wedge B) \rightarrow \varphi_4$ inplikazioa betetzen al den aztertuko da

	Prog 1
	$\{INV \wedge B\} \equiv \{ezneg(A(1..n)) \wedge (0 \leq i \leq n) \wedge c = \sum_{k=1}^i A(k) \wedge (i \neq n)\}$ $\{\varphi_4\} \equiv \{\text{def}(c + A(i + 1)) \wedge (\varphi_3)_c^{c + A(i + 1)}\}$ $c := c + A(i + 1);$ $\{\varphi_3\} \equiv \{ezneg(A(1..n)) \wedge (-1 \leq i \leq n - 1) \wedge c = \sum_{k=1}^{i+1} A(k)\}$ $i := i + 1;$ $\{INV\} \equiv \{ezneg(A(1..n)) \wedge (0 \leq i \leq n) \wedge c = \sum_{k=1}^i A(k)\}$

EA ↗

$$\begin{aligned}
 \{\varphi_4\} &\equiv \{\text{def}(c + A(i + 1)) \wedge (\varphi_3)_c^{c + A(i + 1)}\} \equiv \\
 &\equiv \{(1 \leq i + 1 \leq n) \wedge ezneg(A(1..n)) \wedge (-1 \leq i \leq n - 1) \wedge \\
 &\quad c + A(i + 1) = \sum_{k=1}^{i+1} A(k)\} \equiv \text{sinplifikazioa} \\
 &\equiv \{(0 \leq i \leq n - 1) \wedge ezneg(A(1..n)) \wedge (-1 \leq i \leq n - 1) \wedge \\
 &\quad c + A(i + 1) = \sum_{k=1}^{i+1} A(k)\} \equiv \text{sinplifikazioa} \\
 &\equiv \{(0 \leq i \leq n - 1) \wedge ezneg(A(1..n)) \wedge c + A(i + 1) = \sum_{k=1}^{i+1} A(k)\}
 \end{aligned}$$

$(INV \wedge B) \rightarrow \varphi_4$ betezen al da?

$$\begin{array}{ccccccc}
 \underbrace{ezneg(A(1..n))}_{\alpha} & \underbrace{\wedge (0 \leq i \leq n)}_{\beta} & \underbrace{\wedge c = \sum_{k=1}^i A(k)}_{\delta} & \underbrace{\wedge (i \neq n)}_{\gamma} & & & \\
 & & \downarrow ? & & & & \\
 \underbrace{(0 \leq i \leq n - 1)}_{\beta \text{ eta } \gamma\text{-gatik}} & \underbrace{\wedge ezneg(A(1..n))}_{\alpha\text{-gatik}} & \underbrace{\wedge c + A(i + 1) = \sum_{k=1}^{i+1} A(k)}_{\delta\text{-gatik}} & & & &
 \end{array}$$

Orain Prog 1 programa zuzena da baina Prog 1 eta Prog 2 programek agindu berdinak izan behar dituztenez, Prog 2 programan $c := c + A(i + 1);$ esleipena ipini behar da eta φ_4' formula kalkulatu behar da esleipenaren

axioma erabiliz. Bukatzeko $(INB \wedge B \wedge E = v) \rightarrow \varphi_4'$ inplikazioa betetzen al den aztertu beharko da.

	Prog 2
EA ↗	$\{INB \wedge B \wedge E = v\} \equiv$ $\equiv \{ \text{ezneg}(A(1..n)) \wedge (0 \leq i \leq n) \wedge c = \sum_{k=1}^i A(k) \wedge (i \neq n) \wedge n - i = v \}$ $\{\varphi_4'\} \equiv \{ \text{def}(c + A(i + 1)) \wedge (\varphi_3')_c^{c + A(i + 1)} \}$ $\mathbf{c} := \mathbf{c} + \mathbf{A}(\mathbf{i} + \mathbf{1});$ $\{\varphi_3'\} \equiv \{ n - i - 1 < v \}$ $\mathbf{i} := \mathbf{i} + \mathbf{1};$ $\{E < v\} \equiv \{ n - i < v \}$

$$\begin{aligned} \{\varphi_4'\} &\equiv \{ \text{def}(c + A(i + 1)) \wedge (\varphi_3')_c^{c + A(i + 1)} \} \equiv \\ &\equiv \{ (1 \leq i + 1 \leq n) \wedge n - i - 1 < v \} \\ &\equiv \{ (0 \leq i \leq n - 1) \wedge n - i - 1 < v \} \end{aligned}$$

$(INB \wedge B \wedge E = v) \rightarrow \varphi_4'$ inplikazioa betetzen al da?

$$\begin{array}{c} \underbrace{\text{ezneg}(A(1..n)) \wedge (0 \leq i \leq n)}_{\alpha} \wedge c = \sum_{k=1}^i A(k) \wedge \underbrace{(i \neq n)}_{\beta} \wedge \underbrace{n - i = v}_{\delta} \\ \downarrow ? \\ \underbrace{(0 \leq i \leq n - 1)}_{\alpha \text{ eta } \beta\text{-gatik}} \wedge \underbrace{n - i - 1 < v}_{\delta\text{-gatik, izan ere } n - i = v \text{ bada, } n - i - 1 = v - 1 \text{ izango da eta } v - 1 \text{ balioa } v \text{ baino txikiagoa da}} \end{array}$$

$(INB \wedge B) \rightarrow \varphi_4$ eta $(INB \wedge B \wedge E = v) \rightarrow \varphi_4'$ inplikazioak bete egin direnez programaren eratorpena bukatu da. Eratorritako programa honako hau da:

```
{ $\varphi$ }
{ $\varphi_2$ }
c := 0;
{ $\varphi_1$ }
i := 1;

while {INB} i  $\neq$  n loop
    { $\varphi_4$ } { $\varphi_4'$ }
    c := c + A(i + 1);
    { $\varphi_3$ } { $\varphi_3'$ }
    i := i + 1;
end loop;

{ $\psi$ }
E
```

Programa hori zuzena da While-aren erregela jarraituz eraiki dugulako eta gainera metodoaren bidez programa dokumentatzeko balio duten $\{\varphi_1\}$, $\{\varphi_2\}$, $\{\varphi_3\}$, $\{\varphi_3'\}$, $\{\varphi_4\}$ eta $\{\varphi_4'\}$ formulak kalkulatu ditugu.

4.2.4. C ALDAGAIA 0 ITZULI A(1..N) TAULAN GUTXIENEZ POSIZIO BATEAN 0 BADAGO (11. ARIKETA)

Zenbaki osozko A(1..n) bektorea emanda, A(1..n) taulako posizioen batean 0 baldin badago, osoa den c aldagaian 0 itzuliko duen eta A(1..n) taulan Orik ez badago, c aldagaian 0 itzuliko ez duen programa eratorri Hoare-ren kalkuluko Esleipenaren Axioma eta While-aren Erregela erabiliz. Eratorritako programak INB inbariantea, E espresioa eta ϕ eta ψ formulen bidez emandako aurre-ondoetako espezifikazioarekiko zuzena izan beharko du.

$\{\phi\} \equiv \{n \geq 1\}$ Hasieraketak? $\{INB\}$ while $\{INB\} \{E\} B?$ loop Aginduak? end loop; $\{\psi\} \equiv \{(c = 0) \leftrightarrow \exists k(1 \leq k \leq n \wedge A(k) = 0)\}$
$\{INB\} \equiv \{(1 \leq i \leq n) \wedge c = \prod_{k=1}^i A(k)\}$ $E = n + 1 - i$

Hasteko bektorea ezkerretik eskuinera edo eskuinetik ezkerrera zeharkatu behar al den erabaki behar da. E espresioa "goiko muka – indizea" erakoa denez, bektorea ezkerretik eskuinera zeharkatu beharko da.

- **Hasieraketak**

Hasieraketak kalkulatzeko While-aren erregelako lehenengo puntua hartu behar da kontuan, hau da, ϕ formulak inbariantea inplikatzeko al duen begiratu behar da. Inplikatzeko badu, ez da ezer hasieratu behar. Ez badu inplikatzeko, hasieraketaren bat beharko da. Inbariantea adieraziko digu zein aldagai eta nola hasieratu. Aldagai baten hasieraketa zein izango den kalkulatu ondoren, Esleipenaren Axioma erabiliz hasieraketa horri dagokion formula kalkulatu da eta ϕ formulak formula berri hori inplikatzeko al duen aztertuko da. Hasieraketak gehitzeko prozesua ϕ formulak inplikatzeko duen formula bat lortu arte jarraitu beharko da.

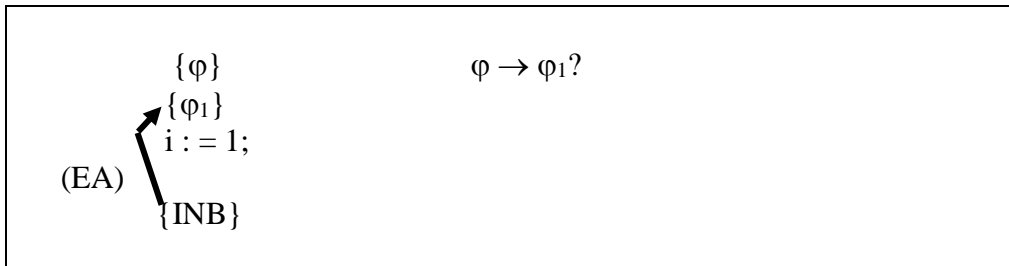
$\varphi \rightarrow \text{INB?}$

$$\begin{array}{c}
 n \geq 1 \\
 \underbrace{} \\
 \alpha \\
 \downarrow? \\
 (1 \leq i \leq n) \wedge c = \prod_{k=1}^i A(k) \\
 \underbrace{} \quad \underbrace{\phantom{\prod_{k=1}^i A(k)}} \\
 \delta_1 \quad \delta_2 \\
 \underbrace{} \\
 \delta \\
 \text{Ez da betetzen}
 \end{array}$$

Inplikazio hori ez da betetzen inplikazioaren lehenengo zatian (α zatian) ez baitago i eta c aldagaiek δ_1 eta δ_2 formulek diotena betetzen dutela ziurtatzeko erabili dezakegun informaziorik. Helburua inplikazioa betetzea denez, hau da, δ_1 eta δ_2 betetzea denez, inplikazio hori betearaziko duten balioekin hasieratu beharko dira i eta c . δ_2 formulari i eta c agertzen direnez eta δ_1 formulari i bakarrik agertzen denez, δ_1 kontuan hartuz i nola hasieratu erabakiko dugu.

Bektorea ezkerretik eskuinera zeharkatu behar denez, δ_1 betetzeko i -ri 0 esleitzea nahikoa da. Beraz δ_1 formulari begiratzuz i aldagaiak har dezakeen balio txikiena esleituiko diogu.

Orain Esleipenaren Axiomari jarraituz $\{\varphi_1\}$ formula kalkulatu dugu.

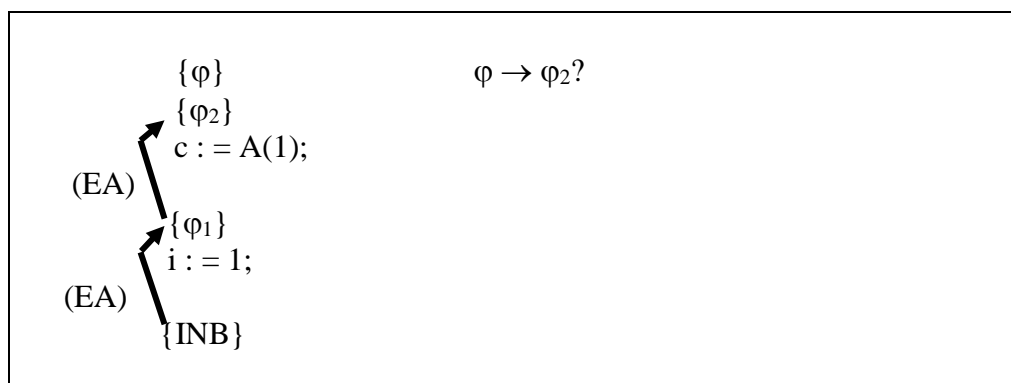


$$\begin{aligned}
 \{\varphi_1\} &\equiv \{\text{def}(1) \wedge (\text{INB})_i^1\} \equiv \\
 &\equiv \{\text{true} \wedge (1 \leq 1 \leq n) \wedge c = \prod_{k=1}^1 A(k)\} \equiv \text{sinplifikazioa} \\
 &\equiv \{(1 \leq n) \wedge c = \prod_{k=1}^1 A(k)\} \equiv \text{sinplifikazioa} \\
 &\equiv \{(1 \leq n) \wedge c = A(1)\}
 \end{aligned}$$

➤ $\varphi \rightarrow \varphi_1?$

$$\begin{array}{c}
 n \geq 1 \\
 \underbrace{} \\
 \alpha \\
 \downarrow? \\
 \underbrace{(1 \leq n)} \wedge \underbrace{c = A(1)} \\
 \text{Bai, } \alpha\text{-gatik} \quad \text{Ez da betetzen}
 \end{array}$$

α betetzeak ez du ziurtatzen $c = A(1)$ betetzea, eta $c = A(1)$ bete dadin c -ri $A(1)$ esleitu beharko diogu.



$$\begin{aligned}
 \{\varphi_2\} &\equiv \{\text{def}(A(1)) \wedge (\varphi_1)_c^{A(1)}\} \equiv \\
 &\equiv \{(1 \leq 1 \leq n) \wedge (1 \leq n) \wedge A(1) = A(1)\} \equiv \\
 &\equiv \{(1 \leq n) \wedge (1 \leq n) \wedge \text{true}\} \equiv \\
 &\equiv \{(1 \leq n)\}
 \end{aligned}$$

➤ $\varphi \rightarrow \varphi_2?$

$$\begin{array}{c}
 n \geq 1 \\
 \underbrace{} \\
 \alpha \\
 \downarrow? \\
 \underbrace{(1 \leq n)} \\
 \text{Bai, } \alpha\text{-gatik}
 \end{array}$$

$\varphi \rightarrow \varphi_2$ implikazioa bete egiten da eta ondorioz hasieraketekin bukatu dugu.

- **While-aren baldintza (B)**

Hasteko $\neg B$ kalkulatu dugu "**while-a noiz geldituko da?**" galderari erantzunez. Bektorea ezkerretik eskuinera zeharkatzen ari garenez eta inbariantearen arabera i aldagaiak hartuko duen balio handiena n izango denez, i -ren balioa n denean gelditu egin beharko dugu:

$$\neg B \equiv i = n$$

Eta ondorioz, $B \equiv i \neq n$

Orain B baldintza zuzena dela egiaztatu beharko da While-aren erregelako II, IV eta V puntuak kontuan hartuz.

$$\text{II. } \text{INB} \rightarrow \text{def}(B)?$$

$$\text{INB} \rightarrow \text{def}(i \neq n)?$$

$\text{INB} \rightarrow \text{true}$? Bai, inplikazioaren bigarren zatian true dagoelako.

$$\text{IV. } (\text{INB} \wedge \neg B) \rightarrow \psi?$$

$$\begin{array}{c} (1 \leq i \leq n) \wedge c = \prod_{k=1}^i A(k) \wedge (i = n) \\ \underbrace{\hspace{1.5cm}}_{\beta} \quad \underbrace{\hspace{1.5cm}}_{\delta} \quad \underbrace{\hspace{1.5cm}}_{\gamma} \\ \downarrow? \\ (c = 0) \leftrightarrow \exists k(1 \leq k \leq n \wedge A(k) = 0) \\ \underbrace{\hspace{10cm}}_{\text{Bai, } \beta, \delta \text{ eta } \gamma\text{-gatik}} \end{array}$$

δ eta γ -gatik c aldagaian $A(1..n)$ bektoreko elementuen biderkadura daukagu eta ondorioz c -ren balioa 0 izango da $A(1..n)$ -ko posizioen batean (gutxienez batean) 0 baldin badago eta bestela c ez da 0 izango.

$$\text{V. } \text{INB} \wedge B \rightarrow E > 0?$$

$$\begin{array}{c} (1 \leq i \leq n) \wedge c = \prod_{k=1}^i A(k) \wedge (i \neq n) \\ \underbrace{\hspace{1.5cm}}_{\alpha} \quad \underbrace{\hspace{1.5cm}}_{\beta} \\ \downarrow? \\ n - i > 0 \\ \underbrace{\hspace{1.5cm}} \\ \text{Bai, } \alpha \text{ eta } \beta\text{-gatik} \end{array}$$

• **Aginduak**

Aginduak kalkulatzeko While-aren erregelako III eta IV puntuetako programak hartu beharko dira kontuan

Prog 1	Prog 2
{INB \wedge B}	{INB \wedge B \wedge E = v}
Aginduak	Aginduak
{INB}	{E < v}


➤ Honako bi inplikazio hauek betetzen al diren aztertu beharko da

- ✓ (INB \wedge B) \rightarrow INV?
- ✓ (INB \wedge B \wedge E = v) \rightarrow E < v?

(INB \wedge B) \rightarrow INV inplikazioa beti beteko da, INB eta B egia badira INV ere egia izango delako.

(INB \wedge B \wedge E = v) \rightarrow E < v inplikazioa ez da beteko E eta v berdinak badira E espresioa ez baita v baino txikiagoa izango.

Bigarren inplikazio hori ez denez betetzen, E < v betearazteko helburuarekin agindu bat gehitu beharko dugu Prog 2 programan. Bektorea ezkerretik eskuinera zeharkatzen ari garenez, E-ren balioa txikitu dadin i aldagaiari i + 1 balioa eman beharko diogu eta gero esleipenaren axioma erabiliz φ_3' formula kalkulatu da.

	Prog 2
<div>EA</div> 	$\{\text{INB} \wedge \text{B} \wedge \text{E} = \text{v}\} \equiv$
	$\equiv \{(1 \leq i \leq n) \wedge c = \prod_{k=1}^i \text{A}(k) \wedge (i \neq n) \wedge n - i = \text{v}\}$
	$\{\varphi_3'\} \equiv \{\text{def}(i + 1) \wedge (\text{E} < \text{v})_i^{i+1}\} \equiv \{\text{true} \wedge n - (i + 1) < \text{v}\} \equiv \{n - i - 1 < \text{v}\}$
	i := i + 1;
	$\{n - i < \text{v}\}$

Orain (INB \wedge B \wedge E = v) $\rightarrow \varphi_3'$ inplikazioa betetzen al den egiaztatu beharko dugu:

$$\begin{array}{c}
 (1 \leq i \leq n) \wedge c = \prod_{k=1}^i A(k) \wedge (i \neq n) \wedge \underbrace{n - i = v}_{\alpha} \\
 \downarrow ? \\
 \underbrace{n - i - 1 < v}_{\beta}
 \end{array}$$

Inplikazioa bete egiten da β formula egiazkoa baita α -gatik: $n - i = v$ bada, orduan $n - i - 1 = v - 1$ izango da eta $v - 1$ balioa v baino txikiagoa da.

Orain Prog 2 programa zuzena da baina Prog 1 eta Prog 2 programek agindu berdinak eduki behar dituztenez, $i := i + 1$; esleipena Prog 1 programan ipini beharko da eta gero esleipenaren axioma erabiliz esleipen horri dagokion φ_3 formula kalkulatu da eta $(INB \wedge B) \rightarrow \varphi_3$ inplikazioa betetzen al den aztertuko da.

	Prog 1
EA ↗	$\{INB \wedge B\} \equiv \{(1 \leq i \leq n) \wedge c = \prod_{k=1}^i A(k) \wedge (i \neq n)\}$ $\{\varphi_3\} \equiv \{\text{def}(i + 1) \wedge (INB)_i^{i+1}\}$ $i := i + 1;$ $\{INB\} \equiv \{(1 \leq i \leq n) \wedge c = \prod_{k=1}^i A(k)\}$

$$\begin{aligned}
\{\varphi_3\} &\equiv \{\text{def}(i + 1) \wedge (INB)_i^{i+1}\} \equiv \\
&\equiv \{\text{true} \wedge (1 \leq i + 1 \leq n) \wedge c = \prod_{k=1}^{i+1} A(k)\} \equiv \\
&\equiv \{(1 \leq i + 1 \leq n) \wedge c = \prod_{k=1}^{i+1} A(k)\} \\
&\equiv \{(0 \leq i \leq n - 1) \wedge c = \prod_{k=1}^{i+1} A(k)\}
\end{aligned}$$

Orain $(INB \wedge B) \rightarrow \varphi_3$ inplikazioa betetzen al den azertu beharko da:

$$\begin{array}{ccc}
\underbrace{(1 \leq i \leq n)}_{\beta} & \underbrace{\wedge c = \prod_{k=1}^i A(k)}_{\delta} & \underbrace{\wedge (i \neq n)}_{\gamma} \\
& \downarrow ? & \\
\underbrace{(0 \leq i \leq n - 1)}_{\text{Bai } \beta \text{ eta } \gamma\text{-gatik}} & \underbrace{\wedge c = \prod_{k=1}^{i+1} A(k)}_{\pi \text{ (Ez da betetzen)}} &
\end{array}$$

β -gatik c aldagaian $A(1..i)$ zatiko elementuen biderkadura daukagula badakigu baina π formulak c aldagaian $A(1..i + 1)$ zatiko elementuen biderkadura al daukagun galdetzen zaigu eta erantzuna ezezkoa da. Beraz $(INB \wedge B) \rightarrow \varphi_3$ inplikazioa ez da betetzen. Une honetan helburua φ_3 betearaztea denez, hau da heldurua c aldagaian $A(1..i + 1)$ zatiko elementuen biderkadura edukitzea denez, $c - ri$ $A(i + 1)$ balioa bidertu beharko zaio.

$$c := c * A(i + 1);$$

Esleipena ipini ondoren φ_4 kalkulatu da eta $(INV \wedge B) \rightarrow \varphi_4$ inplikazioa betetzen al den aztertuko da

	Prog 1
EA ↗	$\{INV \wedge B\} \equiv \{(1 \leq i \leq n) \wedge c = \prod_{k=1}^i A(k) \wedge (i \neq n)\}$ $\{\varphi_4\} \equiv \{\text{def}(c * A(i + 1)) \wedge (\varphi_3)_c^{c * A(i + 1)}\}$ $c := c * A(i + 1);$ $\{\varphi_3\} \equiv \{(0 \leq i \leq n - 1) \wedge c = \prod_{k=1}^{i+1} A(k)\}$ $i := i + 1;$ $\{INV\} \equiv \{(1 \leq i \leq n) \wedge c = \prod_{k=1}^i A(k)\}$

$$\begin{aligned}
\{\varphi_4\} &\equiv \{\text{def}(c * A(i + 1)) \wedge (\varphi_3)_c^{c * A(i + 1)}\} \equiv \\
&\equiv \{(1 \leq i + 1 \leq n) \wedge (0 \leq i \leq n - 1) \wedge \\
&c * A(i + 1) = \prod_{k=1}^{i+1} A(k)\} \equiv \text{sinplifikazioa} \\
&\equiv \{(0 \leq i \leq n - 1) \wedge (0 \leq i \leq n - 1) \wedge \\
&c * A(i + 1) = \prod_{k=1}^{i+1} A(k)\} \equiv \text{sinplifikazioa} \\
&\equiv \{(0 \leq i \leq n - 1) \wedge c * A(i + 1) = \prod_{k=1}^{i+1} A(k)\}
\end{aligned}$$

$(INV \wedge B) \rightarrow \varphi_4$ betezen al da?

$$\begin{array}{ccc}
\underbrace{(1 \leq i \leq n)}_{\beta} & \underbrace{c = \prod_{k=1}^i A(k)}_{\delta} & \underbrace{\wedge (i \neq n)}_{\gamma} \\
& \downarrow ? & \\
\underbrace{(0 \leq i \leq n - 1)}_{\beta \text{ eta } \gamma\text{-gatik}} & \underbrace{\wedge c * A(i + 1) = \prod_{k=1}^{i+1} A(k)}_{\delta\text{-gatik}} &
\end{array}$$

Orain Prog 1 programa zuzena da baina Prog 1 eta Prog 2 programek agindu berdinak izan behar dituztenez, Prog 2 programan $c := c * A(i + 1);$ esleipena ipini behar da eta φ_4' formula kalkulatu behar da esleipenaren

axioma erabiliz. Bukatzeko $(INB \wedge B \wedge E = v) \rightarrow \varphi_4'$ inplikazioa betetzen al den aztertu beharko da.

	Prog 2
EA ↗	$\{INB \wedge B \wedge E = v\} \equiv$ $\equiv \{(1 \leq i \leq n) \wedge c = \prod_{k=1}^i A(k) \wedge (i \neq n) \wedge n - i = v\}$ $\{\varphi_4'\} \equiv \{\text{def}(s * A(i + 1)) \wedge (\varphi_1')_s^{s * A(i + 1)}\}$ $c := c * A(i + 1);$ $\{\varphi_3'\} \equiv \{n - i - 1 < v\}$ $i := i + 1;$ $\{E < v\} \equiv \{n - i < v\}$

$$\begin{aligned}
 \{\varphi_4'\} &\equiv \{\text{def}(c * A(i + 1)) \wedge (\varphi_3')_s^{s * A(i + 1)}\} \equiv \\
 &\equiv \{(1 \leq i + 1 \leq n) \wedge n - i - 1 < v\} \\
 &\equiv \{(0 \leq i \leq n - 1) \wedge n - i - 1 < v\}
 \end{aligned}$$

$(INB \wedge B \wedge E = v) \rightarrow \varphi_4'$ inplikazioa betetzen al da?

$$\underbrace{(1 \leq i \leq n)}_{\alpha} \wedge c = \prod_{k=1}^i A(k) \wedge \underbrace{(i \neq n)}_{\beta} \wedge \underbrace{n - i = v}_{\delta}$$

↓?

$$\underbrace{(0 \leq i \leq n - 1)}_{\alpha \text{ eta } \beta\text{-gatik}} \wedge \underbrace{n - i - 1 < v}_{\delta\text{-gatik, izan ere } n - i = v \text{ bada, } n - i - 1 = v - 1 \text{ izango da eta } v - 1 \text{ balioa } v \text{ baino txikiagoa da}}$$

$(INB \wedge B) \rightarrow \varphi_4$ eta $(INB \wedge B \wedge E = v) \rightarrow \varphi_4'$ inplikazioak bete egin direnez programaren eratorpena bukatu da. Eratorritako programa honako hau da:

```
{ $\varphi$ }
{ $\varphi_2$ }
c := A(1);
{ $\varphi_1$ }
i := 1;

while {INV} i  $\neq$  n loop
    { $\varphi_4$ } { $\varphi_4'$ }
    c := c * A(i + 1);
    { $\varphi_3$ } { $\varphi_3'$ }
    i := i + 1;
end loop;

{ $\psi$ }
E
```

Programa hori zuzena da While-aren erregela jarraituz eraiki dugulako eta gainera metodoaren bidez programa dokumentatzeko balio duten $\{\varphi_1\}$, $\{\varphi_2\}$, $\{\varphi_3\}$, $\{\varphi_3'\}$, $\{\varphi_4\}$ eta $\{\varphi_4'\}$ formulak kalkulatu ditugu.

4.2.5. BI BEKTORE BERDINAK AL DIREN ERABAKI (12. ARIKETA)

$A(1..n)$ eta $B(1..n)$ bektoreak emanda c aldagai boolearrean $A(1..n)$ eta $B(1..n)$ berdinak al diren ala ez erabakitzen duen programa eratorri Hoare-ren kalkuluko Esleipenaren Axioma eta While-aren Erregela erabiliz. Eratorritako programak INB inbariantea, E espresioa eta φ eta ψ formulen bidez emandako aurre-ondoetako espezifikazioarekiko zuzena izan beharko du.

Lortutako programak eraginkorra izan beharko du, hau da, une batean erantzuna ezezkoa izango dela konturatuz gero, programak bukatu egin beharko du gainontzeko posizioak aztertzeke.

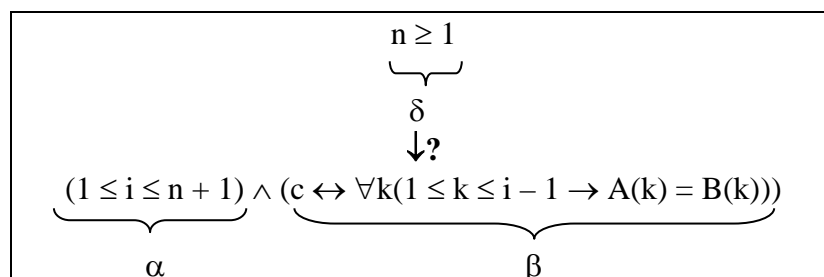
$\{\varphi\} \equiv \{n \geq 1\}$ Hasieraketak? $\{INB\}$ while $\{INB\} \{E\} B?$ loop Aginduak? end loop; $\{\psi\} \equiv \{c \leftrightarrow \forall k(1 \leq k \leq n \rightarrow A(k) = B(k))\}$
$\{INB\} \equiv \{(1 \leq i \leq n + 1) \wedge (c \leftrightarrow \forall k(1 \leq k \leq i - 1 \rightarrow A(k) = B(k)))\}$ $E = n + 1 - i$

Hasteko bektorea ezkerretik eskuinera edo eskuinetik ezkerrera zeharkatu behar al den erabaki behar da. E espresioa "goiko muka – indizea" erakoa denez, bektorea ezkerretik eskuinera zeharkatu beharko da.

- **Hasieraketak**

Hasieraketak kalkulatzeko While-aren erregelako lehenengo puntua hartu behar da kontuan, hau da, φ formulak inbariantea inplikatzeko al duen begiratu behar da. Inplikatzeko badu, ez da ezer hasieratu behar. Ez badu inplikatzeko, hasieraketaren bat beharko da. Inbarianteak adieraziko digu zein aldagai eta nola hasieratu. Aldagai baten hasieraketa zein izango den kalkulatu ondoren, Esleipenaren Axioma erabiliz hasieraketa horri dagokion formula kalkulatu da eta φ formulak formula berri hori inplikatzeko al duen aztertuko da. Hasieraketak gehitzeko prozesua φ formulak inplikatzeko duen formula bat lortu arte jarraitu beharko da.

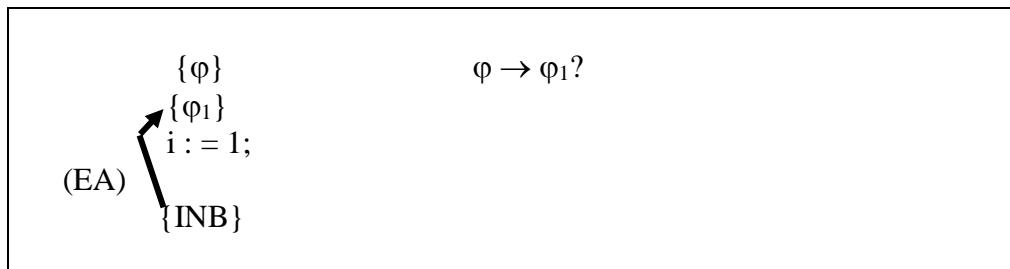
$\varphi \rightarrow INB?$



Inplikazio hori ez da betetzen inplikazioaren lehenengo zatian (δ zatian) ez baitago i eta c aldagaiek α eta β formulek diotena betetzen dutela ziurtatzeko erabili dezakegun informaziorik. Helburua inplikazioa betetzea denez, hau da, α eta β betetzea denez, inplikazio hori betearaziko duten balioekin hasieratu beharko dira i eta c . β formularen i eta c agertzen direnez eta α formularen i bakarrik agertzen denez, α kontuan hartuz i nola hasieratu erabakiko dugu.

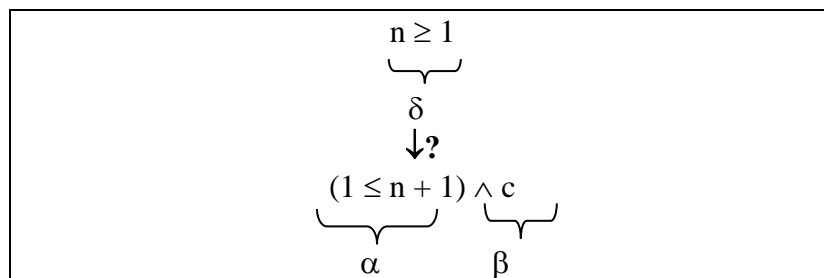
Bektorea ezkerretik eskuinera zeharkatu behar denez, α betetzeko i -ri 1 esleitzea nahikoa da. Beraz α formularen begiratzuz i aldagaiak har dezakeen balio txikiena esleituiko diogu.

Orain Esleipenaren Axioma jarraituz $\{\varphi_1\}$ formula kalkulatu dugu

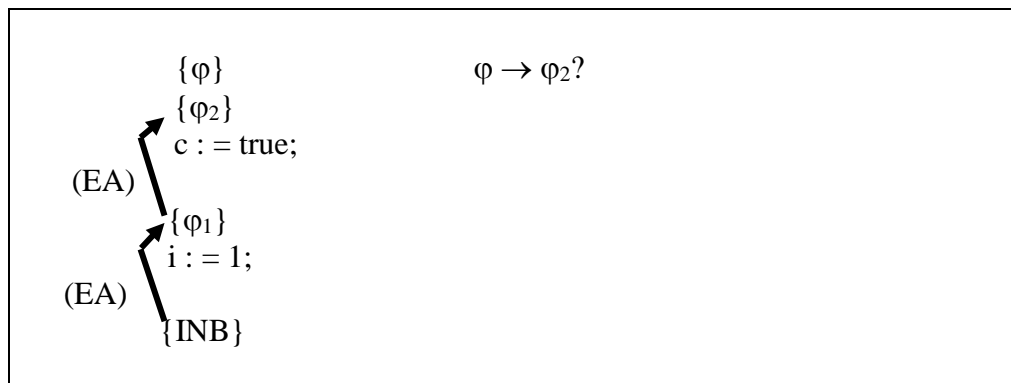


$$\begin{aligned}
 \{\varphi_1\} &\equiv \{\text{def}(1) \wedge (INB)_i^1\} \equiv \\
 &\equiv \{\text{true} \wedge (1 \leq 1 \leq n+1) \wedge (c \leftrightarrow \forall k(1 \leq k \leq 1-1 \rightarrow A(k) = B(k)))\} \equiv \text{sinplifikazioa} \\
 &\equiv \{(1 \leq n+1) \wedge (c \leftrightarrow \forall k(1 \leq k \leq 0 \rightarrow A(k) = B(k)))\} \equiv \text{sinplifikazioa} \\
 &\equiv \{(1 \leq n+1) \wedge c \leftrightarrow \text{true}\} \equiv \text{sinplifikazioa} \\
 &\equiv \{(1 \leq n+1) \wedge c\}
 \end{aligned}$$

➤ $\varphi \rightarrow \varphi_1?$

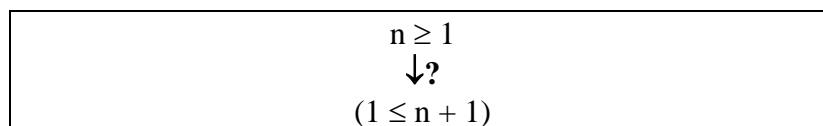


α zatia δ -gatik betetzen da baina β ez da betetzen. β betetzeko c -ri true esleitu beharko diogu.



$$\begin{aligned}
 \{\varphi_2\} &\equiv \{\text{def}(\text{true}) \wedge (\varphi_1)_c^{\text{true}}\} \equiv \\
 &\equiv \{\text{true} \wedge (1 \leq n + 1) \wedge \text{true}\} \equiv \text{simplifikazioa} \\
 &\equiv (1 \leq n + 1)
 \end{aligned}$$

➤ $\varphi \rightarrow \varphi_2?$



$\varphi \rightarrow \varphi_2$ inplikazioa bete egiten da eta ondorioz hasieraketekin amaitu dugu.

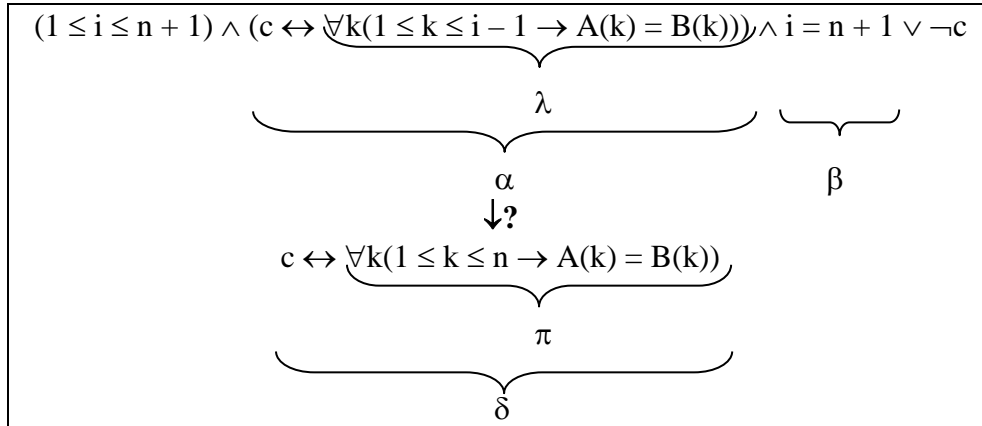
- **While-aren baldintza (B)**

Hasteko $\neg B$ kalkulatu dugu "**while-a noiz geldituko da?**" galderari erantzunez. Bektorea ezkerretik eskuinera zeharkatzen ari garenez eta inbariantearen arabera i aldagaiak hartuko duen balio handiena $n + 1$ izango denez, i -ren balioa $n + 1$ denean gelditu egin beharko dugu, baina programak eraginkorra izan behar duenez, c aldagai boolearrak false balioa hartzen badu ere gelditu egin beharko da. Beraz, i aldagaiak $n + 1$ balioa hartzen badu edo c aldagaiak false balioa hartzen badu, gelditu egin beharko da:

$$\begin{aligned}
 \neg B &\equiv i = n + 1 \vee \neg c \\
 \text{Eta ondorioz, } B &\equiv \neg(i = n + 1 \vee \neg c) \equiv \\
 &\equiv i \neq n + 1 \wedge c
 \end{aligned}$$

Orain B baldintza zuzena dela egiaztatu beharko da While-aren erregelako II, IV eta V puntuak kontuan hartuz.

- II. $INB \rightarrow \text{def}(B)?$
- $INB \rightarrow \text{def}(i \neq n + 1 \wedge c)?$
- $INB \rightarrow \text{true}? \text{ Bai, inplikazioaren bigarren zatian true dagoelako}$

IV. $(INB \wedge \neg B) \rightarrow \psi?$ 

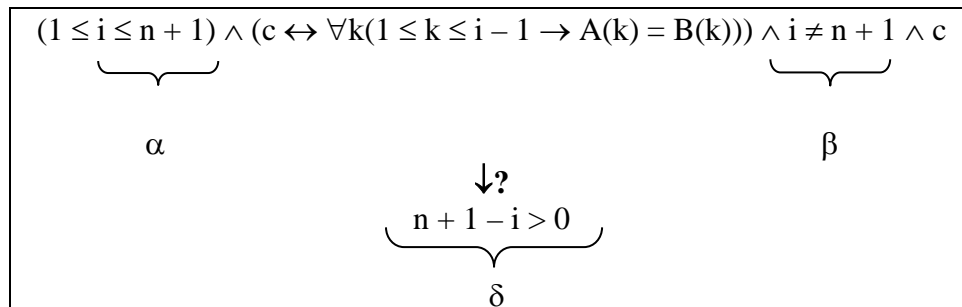
$\neg B$ disjuntzio bat denez, $\neg B$ egia izateko hiru aukera daude, eta hirurak aztertu behar dira:

	$i = n + 1$	$\neg c$
{	True	True
	True	False
	False	True

Lehenengo bi kasuetan, $i = n + 1$ denez, $i - 1$ balioa n da eta δ formula α formularen berdina da eta ondorioz δ bete egiten da.

Hirugarren kasuan $i \neq n + 1$ betetzen da eta $\neg c$ True da, eta ondorioz c False da. α true denez eta c False denez, λ ere false izango da, inplikazio bikoitza True izateko biek False izan beharko dutelako. λ False bada, A eta B bektoreetako $1 \dots i - 1$ tartean posizioen batean A eta B - desberdinak diren elementuak ditugu. Eta orain galdera honako hau da, δ betetzen al da? δ inplikazio bikoitza da eta c false denez π formulak false izan beharko luke δ true izateko. π false al da? Bai, λ False denez badakigu A eta B bektoreetako elementu denak ez direla berdinak eta ondorioz π False da eta δ True da.

Beraz $(INB \wedge \neg B) \rightarrow \psi$ inplikazioa bete egiten da.

V. $(INB \wedge B) \rightarrow E > 0?$ 

δ formula α eta β -gatik betetzen da; izan ere, α eta β -gatik $n + 1 > i$ dela esan dezakegu eta ondorioz, $n + 1 - i > i - i$ betetzen da, hau da, $n + 1 - i > 0$.

- **Aginduak**

Aginduak kalkulatzeko While-aren erregelako III eta IV puntuetako programak hartu beharko dira kontuan

Prog 1
$\{INB \wedge B\}$
Aginduak?
$\{INB\}$

Prog 2
$\{INB \wedge B \wedge E = v\}$
Aginduak?
$\{E < v\}$

➤ Honako bi inplikazio hauek betetzen al diren aztertu beharko da:

✓ $(INB \wedge B) \rightarrow INB$?

✓ $(INB \wedge B \wedge E = v) \rightarrow E < v$?

$(INB \wedge B) \rightarrow INB$ inplikazioa beti beteko da, INB eta B egia badira INB ere egia izango delako.

$(INB \wedge B \wedge E = v) \rightarrow E < v$ inplikazioa ez da beteko E eta v berdinak badira E espresioa ez baita v baino txikiagoa izango.

Bigarren inplikazio hori ez denez betetzen, $E < v$ betearazteko helburuarekin agindu bat gehitu beharko dugu Prog 2 programan. Bektorea ezkerretik eskuinera zeharkatzen ari garenez, E -ren balioa txikitu dadin i aldagaiari $i + 1$ balioa eman beharko diogu eta gero esleipenaren axioma erabiliz φ_3' formula kalkulatu da.

	Prog 2
EA ↗	$\{INB \wedge B \wedge E = v\} \equiv$
	$\{(1 \leq i \leq n + 1) \wedge (c \leftrightarrow \forall k(1 \leq k \leq i - 1 \rightarrow A(k) = B(k))) \wedge i \neq n + 1 \wedge c \wedge n + 1 - i = v\}$
	$\{\varphi_3'\} \equiv \{def(i + 1) \wedge (E < v)_i^{i+1}\} \equiv \{true \wedge n + 1 - (i + 1) < v\} \equiv \{n - i < v\}$
	$i := i + 1;$
	$\{n + 1 - i < v\}$

Orain $(INB \wedge B \wedge E = v) \rightarrow \varphi_3'$ inplikazioa betetzen al den egiaztatu beharko dugu

$$\begin{array}{c}
 (1 \leq i \leq n+1) \wedge (c \leftrightarrow \forall k(1 \leq k \leq i-1 \rightarrow A(k) = B(k))) \wedge i \neq n+1 \wedge c \wedge \underbrace{n+1-i=v}_{\alpha} \\
 \downarrow? \\
 \underbrace{n-i < v}_{\beta}
 \end{array}$$

Inplikazioa bete egiten da β formula egiazkoa baita α -gatik. Izan ere, $n+1-i=v$ bada, orduan $n+1-i-1=v-1$ izango da, hau da, $n-i=v-1$, eta $v-1$ balioa v baino txikiagoa da.

Orain Prog 2 programa zuzena da baina Prog 1 eta Prog 2 programek agindu berdinak eduki behar dituztenez, $i := i+1$; esleipena Prog 1 programan ipini beharko da eta gero esleipenaren axioma erabiliz esleipen horri dagokion φ_3 formula kalkulatu da eta $(INB \wedge B) \rightarrow \varphi_3$ inplikazioa betetzen al den aztertuko da.

	Prog 1
EA ↗	$\{INB \wedge B\} \equiv$ $\equiv \{(1 \leq i \leq n+1) \wedge (c \leftrightarrow \forall k(1 \leq k \leq i-1 \rightarrow A(k) = B(k))) \wedge i \neq n+1 \wedge c\}$ $\{\varphi_3\} \equiv \{\text{def}(i+1) \wedge (INB)_i^{i+1}\}$ $i := i+1;$ $\{INB\} \equiv \{(1 \leq i \leq n+1) \wedge (c \leftrightarrow \forall k(1 \leq k \leq i-1 \rightarrow A(k) = B(k)))\}$

$$\begin{aligned}
 \{\varphi_3\} &\equiv \{\text{def}(i+1) \wedge (INB)_i^{i+1}\} \equiv \\
 &\equiv \{\text{true} \wedge (1 \leq i+1 \leq n+1) \wedge (c \leftrightarrow \forall k(1 \leq k \leq i+1-1 \rightarrow A(k) = B(k)))\} \\
 &\equiv \{(0 \leq i \leq n) \wedge (c \leftrightarrow \forall k(1 \leq k \leq i \rightarrow A(k) = B(k)))\}
 \end{aligned}$$

Orain $(INB \wedge B) \rightarrow \varphi_3$ inplikazioa betetzen al den aztertu beharko da

$$\begin{array}{c}
 \underbrace{(1 \leq i \leq n+1)}_{\alpha} \wedge \underbrace{(c \leftrightarrow \forall k(1 \leq k \leq i-1 \rightarrow A(k) = B(k)))}_{\sigma} \wedge \underbrace{i \neq n+1}_{\delta_1} \wedge \underbrace{c}_{\delta_2} \\
 \downarrow? \\
 \underbrace{(0 \leq i \leq n)}_{\lambda} \wedge \underbrace{(c \leftrightarrow \forall k(1 \leq k \leq i \rightarrow A(k) = B(k)))}_{\gamma} \\
 \underbrace{\hspace{10em}}_{\pi}
 \end{array}$$

- λ zatia α eta δ_1 -egatik betetzen da.

- δ_2 -gatik c true dela badakigu eta β ere true denez, σ ere true izango da (beraz σ true da δ_2 eta β -gatik). σ formulak 1..i – 1 tartean A eta B bektoreak berdinak direla dio. Bestalde, c true denez, π formula true izateko γ formulak ere true izan beharko luke, baina ezin da esan γ true izango denik. Beraz $(INB \wedge B) \rightarrow \varphi_3$ implikazioa ez da betetzen. Une honetan helburua φ_3 betearaztea denez, hau da heldurua π betetzea denez, c aldagaiak true balio beharko luke 1..i tarteko A eta B-ko elementuak berdinak badira eta false balio beharko luke 1..i tarteko A eta B-ko elementuak berdinak ez badira. δ_2 eta β -gatik badakigu c true dela eta 1..i – 1 tarteko A eta B-ko elementuak berdinak direla, eta ondorioz, π betetzeko c aldagaiak true balioarekin jarraitu beharko du A(i) eta B(i) berdinak badira eta c aldagaiak false balioa hartu beharko du A(i) eta B(i) berdinak ez badira. Hori honako esleipenaren bidez lortuko dugu:

$$c := (A(i) = B(i));$$

Esleipena ipini ondoren φ_4 kalkulatu da eta $(INB \wedge B) \rightarrow \varphi_4$ implikazioa betetzen al den aztertuko da

	Prog 1
EA ↗	$\{INB \wedge B\} \equiv$ $\equiv \{(1 \leq i \leq n + 1) \wedge (c \leftrightarrow \forall k(1 \leq k \leq i - 1 \rightarrow A(k) = B(k))) \wedge i \neq n + 1 \wedge c\}$ $\{\varphi_4\} \equiv \{\text{def}(A(i) = B(i)) \wedge (\varphi_3)_c^{(A(i) = B(i))}\}$ $c := (A(i) = B(i));$ $\{\varphi_3\} \equiv \{(0 \leq i \leq n) \wedge (c \leftrightarrow \forall k(1 \leq k \leq i \rightarrow A(k) = B(k)))\}$ $i := i + 1;$ $\{INB\} \equiv \{(1 \leq i \leq n + 1) \wedge (c \leftrightarrow \forall k(1 \leq k \leq i - 1 \rightarrow A(k) = B(k)))\}$

$$\begin{aligned}
\{\varphi_4\} &\equiv \{\text{def}(A(i) = B(i)) \wedge (\varphi_3)_c^{(A(i) = B(i))}\} \equiv \\
&\equiv \{(1 \leq i \leq n) \wedge (0 \leq i \leq n) \wedge ((A(i) = B(i)) \leftrightarrow \forall k(1 \leq k \leq i \rightarrow A(k) = B(k)))\} \equiv \\
&\equiv \{(1 \leq i \leq n) \wedge ((A(i) = B(i)) \leftrightarrow \forall k(1 \leq k \leq i \rightarrow A(k) = B(k)))\}
\end{aligned}$$

$(INB \wedge B) \rightarrow \varphi_4$ betezen al da?

$$\begin{array}{c}
\underbrace{(1 \leq i \leq n+1)}_{\alpha} \wedge \underbrace{(c \leftrightarrow \forall k(1 \leq k \leq i-1 \rightarrow A(k) = B(k)))}_{\sigma} \wedge \underbrace{i \neq n+1}_{\delta_1} \wedge \underbrace{c}_{\delta_2} \\
\downarrow \beta \\
\underbrace{(1 \leq i \leq n)}_{\lambda} \wedge \underbrace{((A(i) = B(i)) \leftrightarrow \forall k(1 \leq k \leq i \rightarrow A(k) = B(k)))}_{\pi}
\end{array}$$

- λ zatia α eta δ_1 -egatik betetzen da.
- π zatia true izateko γ_1 eta γ_2 formulek biek true edo biek false izan beharko lukete. δ_2 -gatik badakigu c true dela. Ondorioz, β true denez, σ formula ere true da, hau da $1..i-1$ tarteko A eta B bektoreetako elementuak berdinak dira. Hori jakinda, γ_1 true bada γ_2 ere true izango da eta π ere true izango da true \leftrightarrow true edukiko dugulako. γ_1 false bada γ_2 ere false izango da ez baita egia izango $1..i$ tarteko A eta B bektoreetako elementuak berdinak direnik eta ondorioz π true izango da false \leftrightarrow false edukiko dugulako.
- Beraz $(INB \wedge B) \rightarrow \varphi_4$ inplikazioa bete egiten da.

Orain Prog 1 programa zuzena da baina Prog 1 eta Prog 2 programek agindu berdinak izan behar dituztenez, Prog 2 programan $\mathbf{c} := (\mathbf{A}(\mathbf{i}) = \mathbf{B}(\mathbf{i}))$; esleipena ipini behar da eta φ_4' formula kalkulatu behar da esleipenaren axioma erabiliz. Bukatzeko $(INB \wedge B \wedge E = v) \rightarrow \varphi_4'$ inplikazioa betetzen al den aztertu beharko da.

	Prog 2
EA ↗	$\{INB \wedge B \wedge E = v\} \equiv$ $\{(1 \leq i \leq n+1) \wedge (c \leftrightarrow \forall k(1 \leq k \leq i-1 \rightarrow A(k) = B(k))) \wedge i \neq n+1 \wedge c \wedge n+1-i = v\}$ $\{\varphi_4'\} \equiv \{\text{def}((A(i) = B(i))) \wedge (\varphi_3')_c^{(A(i) = B(i))}\}$ $\mathbf{c} := (\mathbf{A}(\mathbf{i}) = \mathbf{B}(\mathbf{i}));$ $\{\varphi_3'\} \equiv \{n-i < v\}$ $\mathbf{i} := \mathbf{i} + 1;$ $\{E < v\} \equiv \{n+1-i < v\}$

$$\begin{aligned}
\{\varphi_4'\} &\equiv \{\text{def}((A(i) = B(i))) \wedge (\varphi_3')_c^{(A(i) = B(i))}\} \equiv \\
&\equiv \{(1 \leq i \leq n) \wedge n-i < v\}
\end{aligned}$$

$(INB \wedge B \wedge E = v) \rightarrow \varphi_4'$ inplikazioa betetzen al da?

$$\begin{array}{c}
 \underbrace{(1 \leq i \leq n+1)}_{\alpha} \wedge (c \leftrightarrow \forall k(1 \leq k \leq i-1 \rightarrow A(k) = B(k))) \wedge \underbrace{i \neq n+1}_{\beta} \wedge \underbrace{n+1-i=v}_{\delta} \\
 \downarrow? \\
 \underbrace{(1 \leq i \leq n)}_{\alpha \text{ eta } \beta\text{-gatik}} \wedge \underbrace{n-i < v}_{\delta\text{-gatik, } \delta\text{-gatik } n-i=v-1 \text{ baita eta } v-1 \text{ balioa } v \text{ baino txikiagoa da}}
 \end{array}$$

$(INB \wedge B) \rightarrow \varphi_4$ eta $(INB \wedge B \wedge E = v) \rightarrow \varphi_4'$ inplikazioak bete egin direnez programaren eratorpena bukatu da. Eratorritako programa honako hau da:

```

{φ}
{φ2}
c := true;
{φ1}
i := 1;

while {INB} i ≠ n + 1 and c loop
    {φ4} {φ4'}
    c := (A(i) = B(i));
    {φ3} {φ3'}
    i := i + 1;
end loop;

{ψ}
E

```

Programa hori zuzena da While-aren erregelari jarraituz eraiki dugulako eta gainera metodoaren bidez programa dokumentatzeko balio duten $\{\varphi_1\}$, $\{\varphi_2\}$, $\{\varphi_3\}$, $\{\varphi_3'\}$, $\{\varphi_4\}$ eta $\{\varphi_4'\}$ formulak kalkulatu ditugu.

4.2.6. x ELEMENTUA A(1..N) BEKTOREAN AGERTZEN AL DEN ERABAKI (18. ARIKETA)

x zenbaki bat eta A(1..n) bektorea emanda c aldagai boolearrean x elementua A(1..n) bektorean agertzen al den ala ez erabakitzen duen programa eratorri Hoare-ren kalkuluko Esleipenaren Axioma eta While-aren Erregela erabiliz. Eratorritako programak INB inbariantea, E espresioa eta φ eta ψ formulen bidez emandako aurre-ondoetako espezifikazioarekiko zuzena izan beharko du.

Lortutako programak eraginkorra izan beharko du, hau da, une batean erantzuna baiezkoa izango dela konturatuz gero, programak bukatu egin beharko du gainontzeko posizioak aztertzeke.

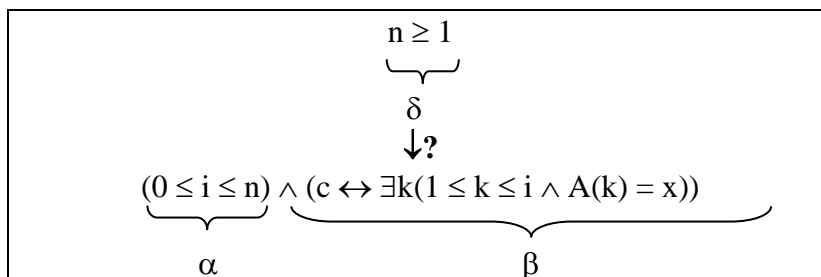
$\{\varphi\} \equiv \{n \geq 1\}$ Hasieraketak? $\{INB\}$ while $\{INB\}$ $\{E\}$ B? loop Aginduak? end loop; $\{\psi\} \equiv \{c \leftrightarrow \exists k(1 \leq k \leq n \wedge A(k) = x)\}$
$\{INB\} \equiv \{(0 \leq i \leq n) \wedge (c \leftrightarrow \exists k(1 \leq k \leq i \wedge A(k) = x))\}$ $E = n - i$

Hasteko bektorea ezkerretik eskuinera edo eskuinetik ezkerreara zeharkatu behar al den erabaki behar da. E espresioa "goiko muka – indizea" erakoa denez, bektorea ezkerretik eskuinera zeharkatu beharko da.

- **Hasieraketak**

Hasieraketak kalkulatzeko While-aren erregelako lehenengo puntua hartu behar da kontuan, hau da, φ formulak inbariantea inplikatzeko al duen begiratu behar da. Inplikatzeko badu, ez da ezer hasieratu behar. Ez badu inplikatzeko, hasieraketaren bat beharko da. Inbarianteak adieraziko digu zein aldagai eta nola hasieratu. Aldagai baten hasieraketa zein izango den kalkulatu ondoren, Esleipenaren Axioma erabiliz hasieraketa horri dagokion formula kalkulatu da eta φ formulak formula berri hori inplikatzeko al duen aztertuko da. Hasieraketak gehitzeko prozesua φ formulak inplikatzeko duen formula bat lortu arte jarraitu beharko da.

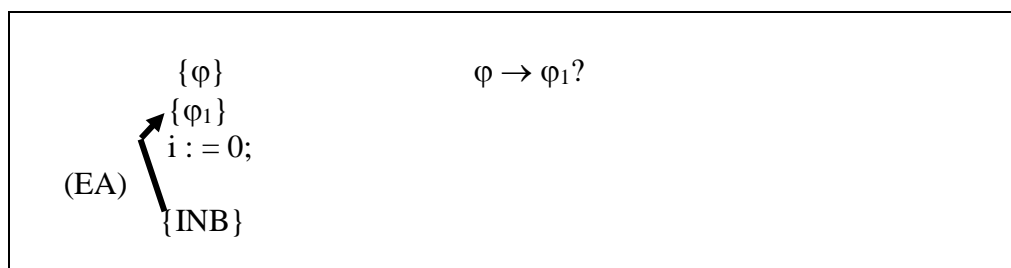
$\varphi \rightarrow INB?$



Inplikazio hori ez da betetzen inplikazioaren lehenengo zatian (δ zatian) ez baitago i eta c aldagaiek α eta β formulek diotena betetzen dutela ziurtatzeko erabili dezakegun informaziorik. Helburua inplikazioa betetzea denez, hau da, α eta β betetzea denez, inplikazio hori betearaziko duten balioekin hasieratu beharko dira i eta c . Horrela, β formulari i eta c agertzen direnez eta α formulari i bakarrik agertzen denez, α kontuan hartuz i nola hasieratu erabakiko dugu.

Bektorea ezkerretik eskuinera zeharkatu behar denez, α betetzeko i -ri 0 esleitzea nahikoa da. Beraz α formulari begiratzuz i aldagaiak har dezakeen balio txikiena esleituko diogu.

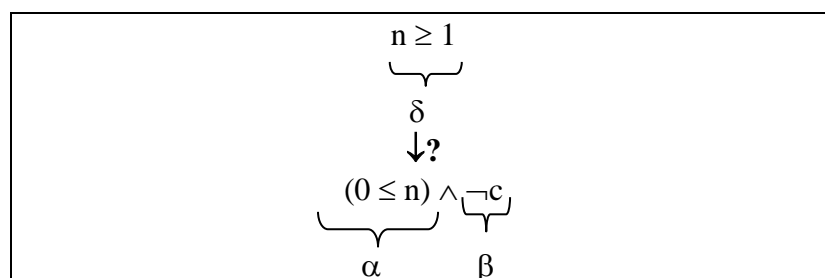
Orain Esleipenaren Axioma jarraituz $\{\varphi_1\}$ formula kalkulatu dugu



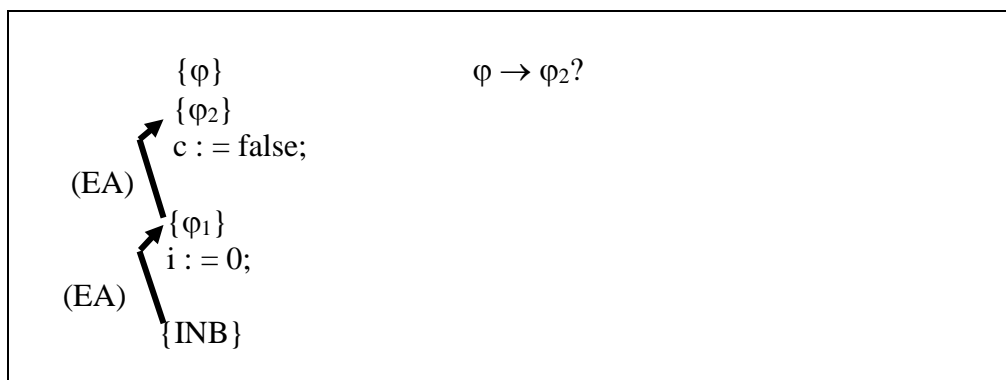
$$\begin{aligned}
 \{\varphi_1\} &\equiv \{\text{def}(0) \wedge (\text{INB})_i^0\} \equiv \\
 &\equiv \{\text{true} \wedge (0 \leq 0 \leq n) \wedge (c \leftrightarrow \exists k(1 \leq k \leq 0 \wedge A(k) = x))\} \equiv \text{sinplifikazioa} \\
 &\equiv \{(0 \leq n) \wedge (c \leftrightarrow \exists k(1 \leq k \leq 0 \wedge A(k) = x))\} \equiv \text{sinplifikazioa} \\
 &\equiv \{(0 \leq n) \wedge c \leftrightarrow \text{false}\} \equiv \text{sinplifikazioa} \\
 &\equiv \{(0 \leq n) \wedge \neg c\}
 \end{aligned}$$

Gogoratu $\exists k(1 \leq k \leq 0 \wedge A(k) = x)$ false dela eremua hutsa delako.

➤ $\varphi \rightarrow \varphi_1?$



α zatia δ -gatik betetzen da baina β ez da betetzen. β betetzeko c -ri false esleitu beharko diogu.



$$\begin{aligned}
 \{\varphi_2\} &\equiv \{\text{def}(\text{false}) \wedge (\varphi_1)_c^{\text{false}}\} \equiv \\
 &\equiv \{\text{true} \wedge (0 \leq n) \wedge \neg \text{false}\} \equiv \text{simplifikazioa} \\
 &\equiv \{\text{true} \wedge (0 \leq n) \wedge \text{true}\} \equiv \text{simplifikazioa} \\
 &\equiv (0 \leq n)
 \end{aligned}$$

➤ $\varphi \rightarrow \varphi_2?$

$$\begin{array}{c}
 n \geq 1 \\
 \downarrow ? \\
 0 \leq n
 \end{array}$$

$\varphi \rightarrow \varphi_2$ inplikazioa bete egiten da eta ondorioz hasieraketekin amaitu dugu.

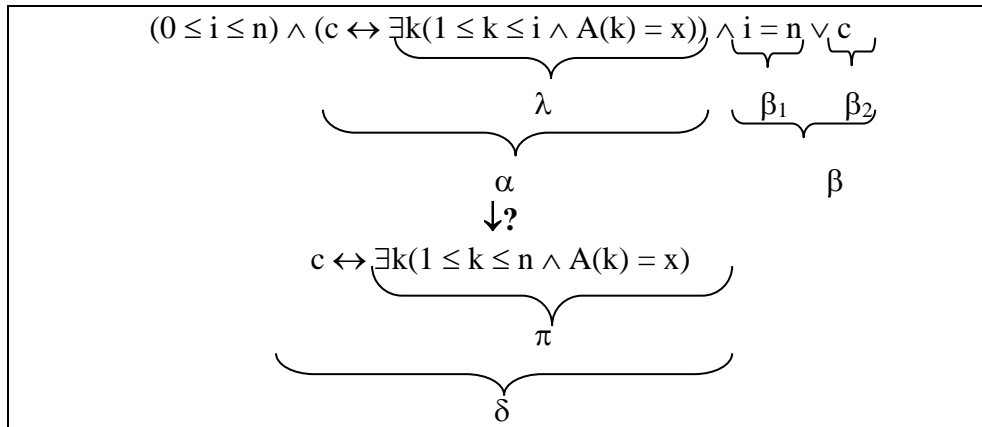
- **While-aren baldintza (B)**

Hasteko $\neg B$ kalkulatu dugu "**while-a noiz geldituko da?**" galderari erantzunez. Bektorea ezkerretik eskuinera zeharkatzen ari garenez eta inbariantearen arabera i aldagaiak hartuko duen balio handiena n izango denez, i -ren balioa n denean gelditu egin beharko dugu, baina programak eraginkorra izan behar duenez, c aldagai boolearrak true balioa hartzen badu ere gelditu egin beharko da. Beraz i aldagaiak n balioa hartzen badu edo c aldagaiak true balioa hartzen badu gelditu egin beharko da:

$$\begin{aligned}
 \neg B &\equiv i = n \vee c \\
 \text{Eta ondorioz, } B &\equiv \neg(i = n \vee c) \equiv \\
 &\equiv i \neq n \wedge \neg c
 \end{aligned}$$

Orain B baldintza zuzena dela egiaztatu beharko da While-aren erregelako II, IV eta V puntuak kontuan hartuz.

$$\begin{aligned}
 \text{II. } \text{INB} &\rightarrow \text{def}(B)? \\
 \text{INB} &\rightarrow \text{def}(i \neq n \wedge \neg c)? \\
 \text{INB} &\rightarrow \text{true?} \text{ Bai, inplikazioaren bigarren zatian true dagoelako}
 \end{aligned}$$

IV. $(INB \wedge \neg B) \rightarrow \psi?$ 

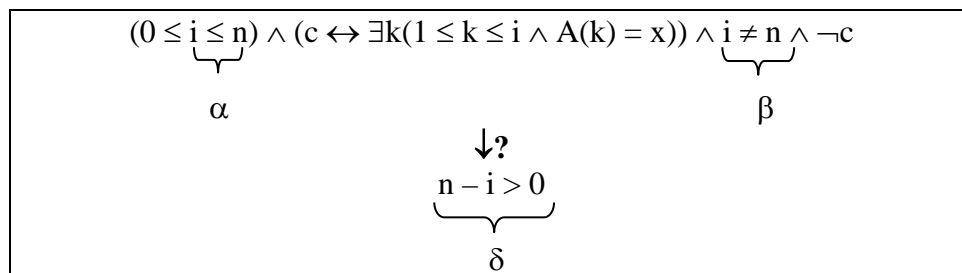
$\neg B$ (edo β) disjuntzio bat denez, $\neg B$ egia izateko hiru aukera daude, eta hirurak aztertu behar dira:

	$i = n$	c
{	True	True
	True	False
	False	True

Lehenengo bi kasuetan, $i = n$ denez, δ formula α formularen berdina da eta ondorioz δ bete egiten da.

Hirugarren kasuan $i \neq n$ betetzen da eta c True da. α true denez eta c true denez, λ ere true izango da, inplikazio bikoitza true izateko biek true izan beharko dutelako. λ true bada, A bektoreko $1 \dots i$ tartean x agertzen da. Eta orain galdera honako hau da, δ betetzen al da? δ inplikazio bikoitza da eta c true denez π formulak true izan beharko luke δ true izateko. π true al da? Bai, λ true denez badakigu A bektoreko $1 \dots i$ tartean x agertzen dela eta ondorioz A bektoreko $1 \dots n$ tartean x agertuko da eta ondorioz π true da eta δ ere true da.

Beraz $(INB \wedge \neg B) \rightarrow \psi$ inplikazioa bete egiten da.

V. $(INB \wedge B) \rightarrow E > 0?$ 

δ formula α eta β -gatik betetzen da, izan ere, α eta β -gatik $n > i$ dela esan dezakegu eta ondorioz, $n - i > i - i$ betetzen da, hau da, $n - i > 0$.

- **Aginduak**

Aginduak kalkulatzeko While-aren erregelako III eta IV puntuetako programak hartu beharko dira kontuan

Prog 1
{INB \wedge B}
Aginduak?
{INB}

Prog 2
{INB \wedge B \wedge E = v}
Aginduak?
{E < v}

➤ Honako bi inplikazio hauek betetzen al diren aztertu beharko da:

✓ (INB \wedge B) \rightarrow INB?

✓ (INB \wedge B \wedge E = v) \rightarrow E < v?

(INB \wedge B) \rightarrow INB inplikazioa beti beteko da, INB eta B egia badira INB ere egia izango delako.

(INB \wedge B \wedge E = v) \rightarrow E < v inplikazioa ez da beteko E eta v berdinak badira E espresioa ez baita v baino txikiagoa izango.

Bigarren inplikazio hori ez denez betetzen, E < v betearazteko helburuarekin agindu bat gehitu beharko dugu Prog 2 programan. Bektorea ezkerretik eskuinera zeharkatzen ari garenez, E-ren balioa txikitu dadin i aldagaiari i + 1 balioa eman beharko diogu eta gero esleipenaren axioma erabiliz φ_3' formula kalkulatu da.

	Prog 2
EA ↗	{INB \wedge B \wedge E = v} \equiv {(0 \leq i \leq n) \wedge (c \leftrightarrow $\exists k(1 \leq k \leq i \wedge A(k) = x)) \wedge i \neq n \wedge \neg c \wedge n - i = v$ }
	{ φ_3' } \equiv {def(i + 1) \wedge (E < v) $^{i+1}$ } \equiv {true \wedge n - (i + 1) < v} \equiv {n - i - 1 < v} i := i + 1; {n - i < v}

Orain (INB \wedge B \wedge E = v) \rightarrow φ_3' inplikazioa betetzen al den egiaztatu beharko dugu

$$\begin{array}{c}
 (0 \leq i \leq n) \wedge (c \leftrightarrow \exists k(1 \leq k \leq i \wedge A(k) = x)) \wedge i \neq n \wedge \underbrace{\neg c \wedge n - i = v}_{\alpha} \\
 \downarrow? \\
 \underbrace{n - i - 1 < v}_{\beta}
 \end{array}$$

Inplikazioa bete egiten da β formula egiazkoa baita α -gatik. Izan ere, $n - i = v$ bada, orduan $n - i - 1 = v - 1$ izango da, hau da, $n - i = v - 1$, eta $v - 1$ balioa v baino txikiagoa da.

Orain Prog 2 programa zuzena da baina Prog 1 eta Prog 2 programek agindu berdinak eduki behar dituztenez, $i := i + 1$; esleipena Prog 1 programan ipini beharko da eta gero esleipenaren axioma erabiliz esleipen horri dagokion φ_3 formula kalkulatu da eta $(INB \wedge B) \rightarrow \varphi_3$ inplikazioa betetzen al den aztertuko da.

	Prog 1
EA ↗	$\{INB \wedge B\} \equiv$ $\equiv \{(0 \leq i \leq n) \wedge (c \leftrightarrow \exists k(1 \leq k \leq i \wedge A(k) = x)) \wedge i \neq n \wedge \neg c\}$ $\{\varphi_3\} \equiv \{\text{def}(i + 1) \wedge (INB)_i^{i+1}\}$ $i := i + 1;$ $\{INB\} \equiv \{(0 \leq i \leq n) \wedge (c \leftrightarrow \exists k(1 \leq k \leq i \wedge A(k) = x))\}$

$$\begin{aligned}
 \{\varphi_3\} &\equiv \{\text{def}(i + 1) \wedge (INB)_i^{i+1}\} \equiv \\
 &\equiv \{\text{true} \wedge (0 \leq i + 1 \leq n) \wedge (c \leftrightarrow \exists k(1 \leq k \leq i + 1 \wedge A(k) = x))\} \\
 &\equiv \{(-1 \leq i \leq n - 1) \wedge (c \leftrightarrow \exists k(1 \leq k \leq i + 1 \wedge A(k) = x))\}
 \end{aligned}$$

Orain $(INB \wedge B) \rightarrow \varphi_3$ inplikazioa betetzen al den aztertu beharko da

$$\begin{array}{c}
 \underbrace{(0 \leq i \leq n)}_{\alpha} \wedge \underbrace{(c \leftrightarrow \exists k(1 \leq k \leq i \wedge A(k) = x))}_{\sigma} \wedge \underbrace{i \neq n}_{\delta_1} \wedge \underbrace{\neg c}_{\delta_2} \\
 \underbrace{\hspace{10em}}_{\beta} \\
 \downarrow? \\
 \underbrace{(-1 \leq i \leq n - 1)}_{\lambda} \wedge \underbrace{(c \leftrightarrow \exists k(1 \leq k \leq i + 1 \wedge A(k) = x))}_{\gamma} \\
 \underbrace{\hspace{10em}}_{\pi}
 \end{array}$$

- λ zatia α eta δ_1 -egatik betetzen da.

- δ_2 -gatik c false dela badakigu eta β true denez, σ ere false izango da (beraz σ flase da δ_2 eta β -gatik). σ false denez, A bektoreko $1..i$ tartean x ez da agertzen. Bestalde, c false denez, π formula true izateko γ formulak ere false izan beharko luke, baina ezin da esan γ false izango denik ez baitakigu A taulako $i + 1$ posizioan x agertzen al den ala ez. Beraz $(INB \wedge B) \rightarrow \varphi_3$ implikazioa ez da betetzen. Une honetan helburua φ_3 betearaztea denez, hau da heldurua π betetzea denez, c aldagaiak true balio beharko luke A taulako $1..i + 1$ tartean x agertzen bada eta false balio beharko luke $1..i + 1$ tartean x ez bada agertzen. δ_2 eta β -gatik badakigu c false dela eta A taulako $1..i$ tartean x ez dela agertzen, eta ondorioz, π betetzeko c aldagaiak false balioarekin jarraitu beharko du $A(i + 1)$ -en x ez bada agertzen eta c aldagaiak true balioa hartu beharko du $A(i + 1)$ -en x agertzen bada. Hori honako esleipenaren bidez lortuko dugu:

$$c := (A(i) = x);$$

Esleipena ipini ondoren φ_4 kalkulatu da eta $(INB \wedge B) \rightarrow \varphi_4$ implikazioa betetzen al den aztertuko da

	Prog 1
EA ↗	$\{INB \wedge B\} \equiv$ $\equiv \{(0 \leq i \leq n) \wedge (c \leftrightarrow \exists k(1 \leq k \leq i \wedge A(k) = x)) \wedge i \neq n \wedge \neg c\}$ $\{\varphi_4\} \equiv \{\text{def}(A(i) = x) \wedge (\varphi_3)_c^{(A(i) = x)}\}$ $c := (A(i) = x);$ $\{\varphi_3\} \equiv \{(-1 \leq i \leq n - 1) \wedge (c \leftrightarrow \exists k(1 \leq k \leq i + 1 \wedge A(k) = x))\}$ $i := i + 1;$ $\{INB\} \equiv \{(0 \leq i \leq n) \wedge (c \leftrightarrow \exists k(1 \leq k \leq i \wedge A(k) = x))\}$

$$\begin{aligned}
\{\varphi_4\} &\equiv \{\text{def}(A(i) = x) \wedge (\varphi_3)_c^{(A(i) = x)}\} \equiv \\
&\equiv \{(1 \leq i \leq n) \wedge (-1 \leq i \leq n - 1) \wedge ((A(i) = x) \leftrightarrow \exists k(1 \leq k \leq i + 1 \wedge A(k) = x))\} \equiv \\
&\quad \{(1 \leq i \leq n) \wedge (-1 \leq i \leq n - 1) \wedge ((A(i) = x) \leftrightarrow \exists k(1 \leq k \leq i + 1 \wedge A(k) = x))\} \\
&\equiv \{(1 \leq i \leq n - 1) \wedge ((A(i) = x) \leftrightarrow \exists k(1 \leq k \leq i + 1 \wedge A(k) = x))\}
\end{aligned}$$

$(INB \wedge B) \rightarrow \varphi_4$ betezen al da?

$$\begin{array}{c}
\underbrace{(0 \leq i \leq n)}_{\alpha} \wedge \underbrace{(c \leftrightarrow \exists k(1 \leq k \leq i \wedge A(k) = x))}_{\sigma} \wedge \underbrace{i \neq n}_{\delta_1} \wedge \underbrace{\neg c}_{\delta_2} \\
\downarrow \beta \\
\underbrace{(1 \leq i \leq n-1)}_{\lambda} \wedge \underbrace{((A(i) = x) \leftrightarrow \exists k(1 \leq k \leq i+1 \wedge A(k) = x))}_{\gamma_1 \wedge \gamma_2} \\
\pi
\end{array}$$

- λ zatia α eta δ_1 -egatik betetzen da.
- π zatia true izateko γ_1 eta γ_2 formulek biek true edo biek false izan beharko lukete. δ_2 -gatik badakigu c false dela. Ondorioz, β true denez, σ formula ere false da, hau da A taulako 1..i tartean x ez da agertzen. Hori jakinda, γ_1 true bada γ_2 ere true izango da eta π ere true izango da true \leftrightarrow true edukiko dugulako. γ_1 false bada γ_2 ere false izango da ez baita egia izango A taulako 1..i + 1 tartean x agertzen dela eta ondorioz π true izango da false \leftrightarrow false edukiko dugulako.
- Beraz $(INB \wedge B) \rightarrow \varphi_4$ inplikazioa bete egiten da.

Orain Prog 1 programa zuzena da baina Prog 1 eta Prog 2 programek agindu berdinak izan behar dituztenez, Prog 2 programan $\mathbf{c} := (\mathbf{A(i) = x})$; esleipena ipini behar da eta φ_4' formula kalkulatu behar da esleipenaren axioma erabiliz. Bukatzeko $(INB \wedge B \wedge E = v) \rightarrow \varphi_4'$ inplikazioa betetzen al den aztertu beharko da.

	Prog 2
EA ↗	$\{INB \wedge B \wedge E = v\} \equiv$ $\{(0 \leq i \leq n) \wedge (c \leftrightarrow \exists k(1 \leq k \leq i \wedge A(k) = x)) \wedge i \neq n \wedge \neg c \wedge n - i = v\}$ $\{\varphi_4'\} \equiv \{\text{def}((A(i) = x)) \wedge (\varphi_3')_c^{(A(i) = x)}\}$ $\mathbf{c} := (\mathbf{A(i) = x});$ $\{\varphi_3'\} \equiv \{n - i < v\}$ $\mathbf{i} := \mathbf{i} + 1;$ $\{E < v\} \equiv \{n + 1 - i < v\}$

$$\begin{aligned}
\{\varphi_4'\} &\equiv \{\text{def}((A(i) = x)) \wedge (\varphi_3')_c^{(A(i) = x)}\} \equiv \\
&\equiv \{(1 \leq i \leq n) \wedge n - i < v\}
\end{aligned}$$

$(INB \wedge B \wedge E = v) \rightarrow \varphi_4'$ inplikazioa betetzen al da?

$$\begin{array}{ccc}
 \underbrace{(0 \leq i \leq n)}_{\alpha} \wedge (c \leftrightarrow \exists k(1 \leq k \leq i \wedge A(k) = x)) \wedge \underbrace{i \neq n}_{\beta} \wedge \neg c \wedge \underbrace{n - i = v}_{\delta} \\
 \downarrow ? \\
 \underbrace{(1 \leq i \leq n)}_{\alpha \text{ eta } \beta\text{-gatik}} \wedge \underbrace{n - i < v}_{\delta\text{-gatik, } \delta\text{-gatik } n - i = v - 1 \text{ baita eta } v - 1 \text{ balioa } v \text{ baino txikiagoa da}}
 \end{array}$$

$(INB \wedge B) \rightarrow \varphi_4$ eta $(INB \wedge B \wedge E = v) \rightarrow \varphi_4'$ inplikazioak bete egin direnez programaren eratorpena bukatu da. Eratorritako programa honako hau da:

```

{φ}
{φ2}
c := false;
{φ1}
i := 0;

while {INB} i ≠ n and not c loop
    {φ4} {φ4'}
    c := (A(i) = x);
    {φ3} {φ3'}
    i := i + 1;
end loop;

{ψ}
E

```

Programa hori zuzena da While-aren erregelari jarraituz eraiki dugulako eta gainera metodoaren bidez programa dokumentatzeko balio duten $\{\varphi_1\}$, $\{\varphi_2\}$, $\{\varphi_3\}$, $\{\varphi_3'\}$, $\{\varphi_4\}$ eta $\{\varphi_4'\}$ formulak kalkulatu ditugu.