

3. gaia. Ariketak.

- 3.1** Lor ezazu testuingururik gabeko gramatika bat, **anbigua ez dena** eta testuingururik gabeko gramatiken lengoaia definitzen duena. Gramatika horren gainean idatz ezazu ondoko itzulpena egiten duen SZIE: gramatika bat emanda itzuli dezala **a** string bat non

a eskuineko aldean ikur gehien dauzkan erregelaren ezker aldeko ikurraren izena den.

Hurrengo gramatika definitu beharreko lengoaiaren formaren adibide bat da. Ikus daitekeenez ';' ez-bukaerako bati dagozkien erregelen bukaera adierazten du.

S	→	A B C1 ;
A	→	'a' B 'a'
		'o' ;
B	→	'b' B 'c'
		'b' 'c' DD
		'b' 'c' 'd' ;
C1	→	'c' 'e' 'r' 'e' 'z' 'o'
		'konstante' ;
DD	→	'd' 'a' 'm' 'o' 'c' 'l' 'e' 's'
		'd' 'e'
		'a' 'r' 'c' 'o' ;

Gramatika honi dagokion itzulpena **DD** da.

- 3.2** Idatzi **LL(1) baldintzak betetzen duen** testuingururik gabeko gramatika bat adierazpen erregularren lengoaia definitzeko. Adierazpen erregularrek ondoren deskribatuko den forma izango dute:

Edozein karaktere adierazpen erregularra da.

λ (kate hutsa) adierazpen erregularra da

e, e1 eta e2 adierazpen erregularrak izanda:

e+ adierazpen erregularra da

e1 | e2 adierazpen erregularra da

e1 e2 adierazpen erregularra da

(e) adierazpen erregularra da

Gramatika horren gainean SZIE bat definitu, adierazpen erregular batek soilik *hiru* karaktereko kateak sortzen dituen ala ez hasierako ikurrarekin lotutako atributu batean lortuko duena.

3.3 Honako SZIE hau gramatika anbiguo baten gainean eraikita dago. SZIE honetan oinarrituta eraiki ezazu LL(1) baldintzak betetzen dituen gramatika baliokidea eta horri dagokion SZIEa.

```

Adierazpena → integer { Adierazpena.mota := 'osokoa' }
Adierazpena → id { Adierazpena.mota := eman_mota(id.izena) }
Adierazpena → Adierazpena mod Adierazpena
    { Baldin Adierazpena(1).mota = 'osokoa' eta Adierazpena(2).mota = 'osokoa'
      orduan Adierazpena.tipo := 'osokoa'
      bestela Adierazpena.tipo := 'errorea' }
Adierazpena → Adierazpena [ Adierazpena ]
    { Baldin Adierazpena(2).mota = osokoa eta Adierazpena(1).mota = array(n,t)
      orduan Adierazpena.mota := t
      bestela Adierazpena.mota := 'errorea' }
Adierazpena → Adierazpena.all
    { Baldin Adierazpena(1).mota = erakuslea(t) orduan Adierazpena.mota := t
      bestela Adierazpena.mota := 'errorea' }

```

3.4 Ondoren emandako SZIE-ek zer espezifikatzen du? SZIE honetan oinarrituta beharrezko analisi aurreralea egiteko baldintzak betetzen dituen SZIE eraiki ezazu.

```

LE → ( E RE ) { baldin RE.val > E.val orduan LE.a:=RE.st
                  bestela LE.a:=E.st }
RE → , E RE { baldin RE(1).val > E.val orduan
               RE.val:=RE(1).val; RE.st:=RE(1).st
               bestela RE.val:=E.val; RE.st:=E.st; }
RE → ξ { RE.val:=-1; RE.st:=""; }

E → E E + { E.val:=E(1).val + E(2).val; E.st:=E(1).st || '+' || E(2).st; E.op:='+'; }
E → E E *
    { E.val:=E(1).val * E(2).val; E.op:='*';
      baldin E(1).op=E(2).op='+' orduan
          E.st:='(' || E(1).st || ')' * (' || E(2).st || ');'
      bestela baldin E(1).op='+' orduan E.st:='(' || E(1).st || ')' * ' || E(2).st ;
      bestela baldin E(2).op='+' orduan E.st:=E(1).st || '*' (' || E(2).st || ');'
      bestela E.st:=E(1).st || '*' || E(2).st ; }
E → digito { E.val:=digito.val; E.st:=digito.st; E.op:=null }

```

3.5 SZIE bat eraiki adierazpen bat notazio postfixura pasatzeko.

Notazio postfixua:

- E1 + E2 adierazpenaren itzulpena: E1-en itzulpena E2-ren itzulpena +
- E identifikadorea edo zenbakia bada, orduan bere itzulpena E da
- (E) bezalako adierazpenaren itzulpena E-ren itzulpena da (parentesirik ez da behar)

Adibidez:	x * y * z	---->	xy*z*
	3 + x * (a + b)	---->	3xab++
	3 + (a + b) + c	---->	3ab++c+

Analisi-zuhaitzak lortu sarrera hauetarako: 9-5+2 eta 9-5*2

3.6 Gauza bera egin alderantzizko itzulpena lortzeko. Berdin notazio arruntetik notazio prefixura pasatzeko (hots, $x+y$ adierazpenaren itzulpena $+xy$ izango da)

3.7 SZIE bat egin adierazpen bat emanda, parentesi erredundanterik gabe emango duena.

Parentesiak erredundanteak dira kenduz gero adierazpenaren eragiketak ordena berdinean exekutazen badira.

Adibidez:	$(3 + 5) + 2$	Parentesiak erredundanteak dira
	$3 + (5 + 2)$	Parentesiak ez dira erredundanteak
	$(3 + 5) * 2$	Parentesiak ez dira erredundanteak
	$3 + (5 * 2)$	Parentesiak erredundanteak dira

3.8 Hirugarren gaian landu dugun lengoaiari **do** agindu berri bat gehitu nahi diogu. Bere forma ondokoa da:

```
do
    agindua1;
    ...
    aginduan;
while (E) ;
```

Bere semantika aginduaren exekuzioa errepikatzea da, adierazpena faltsua izan arte. Honela, barruko aginduak gutxienez behin exekutatuko dira.

Agindu horren portaera osatzeko, beste agindu bat dago: **continue**. Agindu hori **do** baten barruan ager daiteke, eta uneko iterazioa bukatzeko erabiltzen da. Ondorioa da kontrol-fluxua adierazpenaren ebaluaziora joango dela. Adibidez:

```
do
    i=i*2;
    if (i==10) continue;
    ...
while (i>100) ;
```

agindua ikurrari dagokion atributua *agindua.next* da, eta E-rekin E.izena, E.true eta E.false kalkulatzeko direla suposatuta, eta bere esanahia hirugarren gaian azaldutakoa izanda:

- SZIE bat eraiki, agindu horiek hiru helbideko koderaz itzultzeko prozesua definituko duena.
- **do** aginduaren murriztapen semantikoen artean honako hau gehitu nahiko bagenu: **continue** sententzia bakarrik **do** baten barruan ager daiteke. Egiaztapen estatikoa ala dinamikoa izango genuke? Zure erantzuna arrazoitu. SZIEan murriztapen hori egiaztatze behar diren ekintzak sartu.

3.9 Hirugarren gaian lan egin dugun lengoaiari **new_loop** agindua gehitu nahi diogu. Bere forma honako hau da:

```
new_loop identifikadorea  
    agindua;  
end_new_loop identifikadorea;
```

Bere semantika aginduaren exekuzioa etengabe errepikatzean datza.

Agindu honen portaeraren osagarri bezala bigarren agindu bat ere definituko dugu: **exit_new_loop** *identifikadorea*. Agindu hau **new_loop** barruko aginduen artean azaldu daiteke, eta **new_loop** aginduaren exekuzioa eteten du. Efektu bezala kontrol-fluxua identifikadore horrekin etiketatuta dagoen **new_loop** aginduaren hurrengo agindura pasatzen du.

Demagun *aginduari* lotutako atributua *NEXT* (*hurrengoa*) dela, eta beraren esanahia 2. gaian ikusitakoa dela, eta ebazpena garatzeko ezin dela sinbolo-taulara jo, ezta aldagai globalak erabili ere:

- Agindu hauek hiru helbideko kodera itzultzeko prozesua definitzen duen SZIE eraiki.
- Esan ondoko murriztapen semantikoa estatikoa ala dinamikoa izango zen: **new_loop** bakoitzean gutxienez **exit_new_loop** bat egon beharko da.
- **new_loop** bakoitzean gutxienez **exit_new_loop** bat dagoela ziurtatzeko beharrezkoak diren ekintzak gehitu SZIEri.

3.10 Hirugarren gaian lan egin dugun lengoaiari **for** agindua gehitu nahi diogu. Bere forma honako hau da:

```
newfor id [ascending / descending] from E to E do  
    agindua;  
endnewfor;
```

Bere semantika ohiko *for* aginduaren antzekoa da, eta beraz identifikadorearen balioa aldatzen joango litzateke (banan-banan handitzen edo txikitzen) lehenbiziko adierazpenak emandako baliotik hasi eta bigarren adierazpenak emandako balioraino.

Adibideak: **newfor** I ascending from 1 to n+1 to...endnewfor;
 newfor adibidea descending from n*2 to n do ... endnewfor;

Demagun E-ri lotutako atributua *izena* (*aldagaia*) dela eta *aginduari* lotutako atributua *next* dela, eta beraien esanahia 2. gaian ikusitakoa dela:

- Eraiki agindu berri honentzat hiru helbideko kodera itzultzeko prozesua definitzen duen SZIE.
- Agindu honi ondoko murrizpen semantikoak gehitu nahi dizkiogu: **newfor ascending** denean, lehenbiziko adierazpenaren balioa bigarren adierazpenaren balioa baino txikiagoa edo berdina izan behar duela, eta aldiz **newfor descending** denean, lehenbiziko adierazpenaren balioa bigarren adierazpenaren balioa baino handiago edo berdina izan behar duela. Aginduaren egiaztapen semantiko **estatikoa** edo **dinamikoa** egin beharko genuke? Arrazonatu zure erantzuna eta gehitu ieazakiozu SZIEari egiaztapen hori aurrera eramateko ekintzak.