

Compiladors: Examen final de laboratori.

9 de juny de 2026

ATENCIÓ: Al Racó trobareu els jocs de proves i codi necessari per a fer l'examen.
ABANS DE COMENÇAR A FER RES, llegiu les instruccions del final de l'enunciat per veure com descarregar-lo i instal·lar-lo.

ATENCIÓ: Cal entregar l'examen en un fitxer `.tgz` pujat al Racó. *Llegiu les instruccions del final de l'enunciat per veure com generar-lo.*

PUNTUACIÓ: Els tres primers punts de la nota de laboratori s'obtenen amb els jocs de proves de la pràctica base. La resta s'obtenen superant els jocs de proves específics de l'examen. La correcció és **automàtica**, a través dels jocs de proves d'aquest enunciat, més un conjunt addicional de jocs de proves privats.

IMPORTANT: L'examen consta de dos exercicis independents. Podeu fer-los en qualsevol ordre. Es recomana fer cada exercici incrementalment, resolent cada joc de proves abans de passar al següent.

1 Instruccions `break` i `continue` (3 punts)

Volem afegir a l'ASL les instruccions `break` i `continue` que permeten sortir d'un bucle, o saltar-ne a la següent iteració, respectivament.

Per exemple, el següent programa suma els elements en posicions parells d'un array:

```
1 func main()
2     var i: int
3     var s: float
4     var a: array[10] of float
5
6     i = 0;
7     s = 0;
8     while true do
9         if i == 10 then
10             break; // si s'ha acabat l'array, parem
11         endif
12         if i%2==1 then
13             continue; // si la posicio es senar, saltar-la
14         endif
15         s = s + a[i];
16         i = i + 1;
17     endwhile
18     write s;
19 endfunc
```

Donat que pot haver-hi diferents nivells de bucles aniuats, la instrucció `break` té un argument opcional consistent en una *constant entera* (amb valor 1 per defecte) que indica el número de nivells dels quals cal sortir. Així doncs, `break` i `break 1` són equivalents, i surten del `while` actual, mentre que `break 2` sortiria del `while` actual i de l'immediatament superior.

Joc de proves 1 (0.5 punts). Començarem modificant la gramàtica per afegir les dues noves instruccions. La gramàtica forçarà l'argument opcional del **break** a ser **INTVAL**, pel que no cal comprovar el tipus al TypeCheck.

El primer joc de proves comprova que la gramàtica accepta les noves instruccions.

Un cop fets els canvis, el primer joc de proves:

```
1 func f(i : bool, b:array[10] of int) : int
2   var j,a : int
3   write (a + b);
4   a = a > 1;
5   if b[i]>0 then
6     return a;
7   endif
8 endfunc
9
10 func main()
11   var i, j: int
12   var c1, c2: char
13   var x, y: float
14   var A: array[10] of int
15   var B: array[10] of float
16   var C: array[10] of char
17
18   while i<10 do
19     if A[i]%2 == 0 then
20       continue;
21     else
22       i = i + 1;
23       x = c1 * B[j];
24     endif
25
26     while x != y*C[j/i+1] - A[c1] do
27       break;
28     endwhile
29   endwhile
30
31   write y+x;
32 endfunc
```

hauria de produir els següents errors semàntics:

Line 3:11 error: Operator '+' with incompatible types.
Line 4:4 error: Assignment with incompatible types.
Line 5:7 error: Array access with non integer index.
Line 23:16 error: Operator '*' with incompatible types.
Line 26:17 error: Operator '*' with incompatible types.
Line 26:31 error: Array access with non integer index.

Joc de proves 2 (1 punt). A continuació comprovarem que l'ús de **break** i **continue** sigui correcte. Cal comprovar que tant el **break** com el **continue** estan dins d'un bucle, i que el nivell d'aniuament del **break** sigui igual o superior al valor del seu argument INTVAL.

Per fer-ho, cal controlar en quin nivell d'aniuament es troba cada instrucció. Declareu una variable global **int whileDepth=0** al **TypeCheckVisitor**. Al començament de cada visita a una instrucció **while**, cal incrementar aquesta variable, i cal decrementar-la abans d'acabar la visita. Així, les instruccions **break** i **continue** podran comprovar en quina profunditat d'aniuament es troben, i donar un error si és necessari.

Al mòdul auxiliar *SemErrors* proporcionat amb l'exàmen trobareu els errors específics d'aquestes noves instruccions.

Amb això, el segon joc de proves:

```

1 func f(i : bool,
2       b:array[10] of int) : int
3   var j,a : int
4   write (a + b);
5   a = a > 1;
6   if b[i]>0 then
7     break;
8   else
9     break 2;
10  endif
11 endfunc
12
13 func main()
14   var i, j: int
15   var c1, c2: char
16   var x, y: float
17   var A: array[10] of int
18   var B: array[10] of float
19   var C: array[10] of char
20
21   while i<10 do
22     if A[i]%2 == 0 then
23       continue;
24     else
25       i = i + 1;
26       x = c1 * B[j];
27       if not i then
28         break 2;
29       endif
30     endif
31
32     while x != y/C[j*i]-A[c1] do
33       if c1 != c2 then
34         break 3;
35       else
36         break 2;
37       endif
38     endwhile
39   endwhile
40
41   continue;
42   write y+x;
43
44 endfunc

```

ha de produir els errors:

```

Line 4:11 error: Operator '+' with incompatible types.
Line 5:4 error: Assignment with incompatible types.
Line 6:7 error: Array access with non integer index.
Line 7:5 error: Break/continue out of a while.
Line 9:5 error: Break/continue out of a while.
Line 26:16 error: Operator '*' with incompatible types.
Line 27:5 error: Operator 'not' with incompatible types.
Line 28:5 error: Wrong number of levels in break/continue.
Line 32:17 error: Operator '/' with incompatible types.
Line 32:27 error: Array access with non integer index.
Line 34:11 error: Wrong number of levels in break/continue.
Line 41:2 error: Break/continue out of a while.

```

Joc de proves 3 (1 punt). El següent pas es generar el codi per a les noves instruccions. Per superar aquest joc de proves, només cal tractar el **continue** i el **break** 1.

El codi generat és simplement una instrucció de salt incondicional. El **continue** salta al principi del bucle, i el **break** salta al final del bucle.

Per saber a quina etiqueta cal saltar, és necessari recordar quina etiqueta s'ha assignat al bucle en curs: Afegiu al CodeGenVisitor una nova variable global `std::vector<std::string> labelStack` per apilar les etiquetes dels bucles aniuats.

Cada cop que es visita una instrucció **while** i es genera una nova etiqueta, cal apilar-la. A l'acabar la visita, es desapila. Així, el cim de la pila conté sempre l'etiqueta del bucle en curs i podem saber on cal saltar per fer el **break** o el **continue**.

Recordeu que els vectors en C++, tenen els mètodes `push_back(x)`, `pop_back()` i `back()` per afegir, treure, o consultar el darrer element, respectivament.

<pre> 1 func f(M:array[10] of int) : int 2 var i,k,n,t : int 3 read t; 4 n=0; i=0; 5 while i<t+1 do 6 i = i + 1; 7 n = n + M[i+1]; 8 if i==4 then 9 continue; 10 endif 11 i = i+1; 12 if i>7 then 13 break; 14 endif 15 endwhile 16 return n; 17 endfunc </pre>	Un cop fets els canvis, el tercer joc de proves: al llegir l'entrada següent: <pre>14</pre> ha de produir la sortida: <pre> x=6.4 y=1 x=12.8 x=19.2 x=25.6 x=32 y=33 y=39.4 y=45.8 y=52.2 79.2 </pre>
<pre> 18 19 func main() 20 var i, j: int 21 var x, y: float 22 var A: array[10] of int 23 var B: array[10] of float 24 25 i=0; 26 while i<20 do 27 A[i] = i*3; B[i] = i*2.4; 28 i = i+1; 29 if i==10 then break; endif 30 endwhile 31 32 j = 1; 33 while j<10 do 34 if j<6 then 35 x = j + A[j] + B[j]; 36 write "x="; write x; write '\n'; 37 if x>12 then 38 j = j+1; continue; 39 endif 40 endif 41 y = j + A[j-1] + B[j-1]; 42 write "y="; write y; write '\n'; 43 j = j+1; 44 endwhile 45 46 write B[i-2] + f(A); write '\n'; 47 endfunc </pre>	

Joc de proves 4 (0.5 punt). Finalment gestionarem els breaks de més d'un nivell. Simplement, per un break n cal usar la posicio n-èssima a partir del top de la pila (és a dir labelStack.size()-n) per obtenir l'etiqueta on saltar.

Amb això, el següent joc de proves: <pre> 1 func f(M:array[10] of int) : int 2 var i,k,n,t : int 3 read t; read k; 4 n=0; i=1; 5 while i<k+1 do 6 n = n + M[i+1]; 7 while i<4 do 8 break 1; 9 endwhile 10 i = i+1; 11 while i>7 do 12 n = n + t; 13 i = i + 1; 14 if i>10 then 15 break 2; 16 endif 17 endwhile 18 endwhile 19 return n; 20 endfunc 21 22 func main() 23 var i, j: int 24 var x, y: float 25 var A: array[10] of int 26 var B: array[10] of float 27 28 i=0; 29 while i<10 do 30 A[i] = i*3; B[i] = i*2.4; 31 i = i+1; 32 endwhile 33 34 i=0; j=0; 35 while i<20 do 36 i = i+1; 37 while j<20 do 38 B[j] = B[j] + A[i]; 39 if B[j] > 10 then 40 break 2; 41 endif 42 j = j+1; 43 endwhile 44 endwhile 45 46 j = 1; 47 while j<15 do 48 j = j+1; 49 if j<10 then 50 x = j + A[j] + B[j]; 51 write "x="; write x; write '\n'; 52 if x>15 then continue; endif 53 endif 54 y = j + A[j-10] + B[i]; 55 write "y="; write y; write '\n'; 56 endwhile 57 write B[i] + f(A); write '\n'; 58 endfunc </pre>	al llegir l'entrada següent: <pre> 1234 8 </pre>
	ha d'escriure: <pre> x=15.8 x=22.2 x=25.6 x=32 x=38.4 x=44.8 x=51.2 x=57.6 y=15.4 y=19.4 y=23.4 y=27.4 y=31.4 y=35.4 3812.4 </pre>

2 Comparació lexicogràfica d'arrays (4 punts)

El segon exercici consisteix en dotar el llenguatge ASL d'una operació de comparació d'arrays que permet comparar dos arrays i veure si el primer es lexicogràficament menor que el segon. La sintaxi d'aquesta expressió és: $A \ll B$ on A i B són arrays.

Recordeu que l'ordre lèxicogràfic consisteix en comparar els elements en la mateixa posició de A i B. El primer element que sigui diferent determina si A es menor o major que B. Si tots els elements són iguals, la seqüència més curta és la menor.

Per exemple:

- ['a','b','c','d'] és menor que ['a','b','d']
- ['a','b','c'] és menor que ['a','b','c','d']
- [1,2,3,4] és menor que [1,2,4]
- [1,2,3] és menor que [1,2,3,4]

Per tant, podem escriure codi de l'estil

```
1 func main()
2   var i,j: int
3   var b: bool
4   var A,B: array [10] of char
5   var X: array [15] of int
6   var Y: array [10] of float
7
8   i = 0;
9   j = 3;
10  if A << B then
11    Y[i] = Y[i] + X[j];
12
13    b = X << Y and not (B << A);
14    if b then
15      b = not (X << X);
16      A[i] = 'w';
17      B[j] = A[i+1];
18    endif
19  endif
20
21 endfunc
```

Joc de proves 5 (1 punt). El primer pas és afegir l'operació << a la gramàtica. Doneu-li prioritat immediatament inferior a la dels altres operadors relacionals, i tracteu-la amb una regla separada de forma que tingui el seu propi visitor, per no afectar al codi existent. També caldrà fer les comprovacions semàntiques oportunes: A i B han de ser arrays, i els seus elements de tipus comparables. El resultat de l'expressió serà un booleà.

Així, el primer joc de proves:

```

1 func ff(a: array[10] of int, c:array[20] of int) : bool
2     return a<<c or not a[2] << b[1];
3 endfunc
4
5 func main()
6     var i,j: int
7     var b: bool
8     var A,B: array [10] of char
9     var X: array [15] of int
10    var Y: array [10] of float
11
12    i = 0;
13    j = B[j+1];
14    if A << B then
15        X[i] = X[i] + Y[j];
16
17        b = X << Y and not (X << A);
18        if b then
19            b = ff(A,B) and ff(X,Y);
20            b = not B << Y;
21            A[i] = 'w';
22            X[j] = A[i+1];
23        endif
24    endif
25 endfunc

```

genera els errors:

```

Line 2:19 error: Operator 'not' with incompatible types.
Line 2:28 error: Operator '<<' with incompatible types.
Line 2:31 error: Identifier 'b' is undeclared.
Line 13:5 error: Assignment with incompatible types.
Line 15:12 error: Assignment with incompatible types.
Line 17:29 error: Operator '<<' with incompatible types.
Line 19:18 error: Parameter #1 with incompatible types in call to 'ff'.
Line 19:20 error: Parameter #2 with incompatible types in call to 'ff'.
Line 19:30 error: Parameter #1 with incompatible types in call to 'ff'.
Line 19:32 error: Parameter #2 with incompatible types in call to 'ff'.
Line 20:15 error: Operator 'not' with incompatible types.
Line 20:21 error: Operator '<<' with incompatible types.
Line 22:9 error: Assignment with incompatible types.

```

Joc de proves 6 (1 punt). A continuació començarem amb la generació de codi. En aquest joc de proves, els arrays són de la mateixa mida, i els elements del mateix tipus.

El codi d'aquest joc de proves:

```

1 func equal(A: array[10] of float,
2           B: array[10] of float) : bool
3     return not (A<<B or B<<A);
4 endfunc
5
6 func main()
7     var i,j,n: int
8     var b: bool
9     var A,B: array [10] of int
10    var C,D: array [10] of char
11    var X,Y: array [10] of float
12
13    read n;
14    while n!=999 do
15        if n==888 then
16            // 888 means arrays of int
17            i = 0;
18            while i<10 do
19                read A[i]; X[9-i] = A[i]; i = i+1;
20            endwhile
21            i = 0;
22            while i<10 do
23                read B[i]; Y[9-i] = B[i]; i = i+1;
24            endwhile
25            if A << B then
26                write "A smaller than B\n";
27            else if not (B<<A) then
28                write "A is equal than B\n";
29            endif
30            endif
31
32            if equal(X,Y) then write "X == Y\n";
33            else if X<<Y then write "X < Y\n";
34            else write "X > Y\n";
35            endif
36            endif
37        else
38            // not 888 means arrays of char
39            i=0;
40            while i<10 do
41                read C[i]; i=i+1;
42            endwhile
43            i=0;
44            while i<10 do
45                read D[i]; i=i+1;
46            endwhile
47
48            if C << D then
49                write "C smaller than D\n";
50            else if not (D << C) then
51                write "C is equal than D\n";
52            else
53                write C[1]; write D[2]; write "\n";
54            endif
55            endif
56        endif
57
58        read n;
59    endwhile
60 endfunc

```

al processar l'entrada:

```

666
a b c d e f g h i j
a b c d e f g h i k

888
0 1 2 3 4 5 6 7 8 9
0 1 2 3 5 5 7 7 9 9

666
a b c d e f g h i j
a b c d e f g h i j

888
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1

888
1 1 1 1 1 1 1 1 1 1
2 2 2 2 2 2 2 2 2 2

888
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 4 3 2 1

666
a b c d e f g h i j
a b c d e g g g g g

999

```

ha de generar la sortida:

```

C smaller than D
A smaller than B
X < Y
C is equal than D
A is equal than B
X == Y
A smaller than B
X < Y
X > Y
C smaller than D

```

Joc de proves 7 (1 punt). El següent pas consistirà en generar codi per al cas en que un dels arrays es de reals i l'altre d'enters. Cal que el codi generat faci les coercions necessaries abans de fer les comparacions de cada element.

En aquest joc de proves, els arrays encara tenen la mateixa mida.

El codi d'aquest joc de proves: <pre> 1 func mewIF(A: array[10] of int, 2 B: array[10] of float) : float 3 var i: int 4 i = 0; 5 while i<10 and A[i]<B[i] do 6 write A[i]; 7 write " "; 8 i = i+1; 9 endwhile 10 write " --\n"; 11 if i==10 then i=0; endif 12 return A[i] + B[i]; 13 endfunc 14 15 func mewFI(A: array[10] of float, 16 B: array[10] of int) : float 17 var i: int 18 i = 0; 19 while i<10 and A[i]<B[i] do 20 write A[i]; 21 write " "; 22 i = i+1; 23 endwhile 24 write " --\n"; 25 if i==10 then i=0; endif 26 return A[i] + B[i]; 27 endfunc 28 29 func main() 30 var i,j,n: int 31 var b: bool 32 var A: array [10] of int 33 var X: array [10] of float 34 35 read n; 36 while n!=999 do 37 i = 0; 38 while i<10 do read A[i]; i=i+1; 39 endwhile 40 i = 0; 41 while i<10 do read X[i]; i=i+1; 42 endwhile 43 44 if A << X then 45 write "A smaller than X\n"; 46 else if X << A then 47 write "A is bigger than X\n"; 48 else 49 write "A and X are equal\n"; 50 endif 51 endwhile 52 53 write mewIF(A,X); write "\n"; 54 write mewFI(X,A); write "\n"; 55 56 read n; 57 endwhile 58 endfunc </pre>	al llegir l'entrada: <pre> 333 0 1 2 3 4 5 6 7 8 9 0.0 1.0 2.0 3.0 4.0 5.0 7.0 8.0 9.0 9.0 222 1 1 1 1 1 1 1 1 1 1 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 111 1 1 1 1 1 1 1 1 1 1 2.0 2.0 2.0 2.0 2.0 2.0 2.0 2.0 2.0 2.0 555 0 1 2 3 4 5 6 7 8 9 0.0 1.0 2.0 3.0 4.0 5.0 4.0 3.0 2.0 1.0 999 </pre> ha de produir la sortida: <pre> A smaller than X -- 0 -- 0 A and X are equal -- 2 -- 2 A smaller than X 1 1 1 1 1 1 1 1 1 1 -- 3 -- 3 A is bigger than X -- 0 -- 0 </pre>
--	---

Joc de proves 8 (1 punt). Finalment, tractarem el cas en que els arrays són de mida diferent. Si s'arriba al final de l'array més curt (perque tots els elements són iguals) l'array més curt és el menor dels dos.

El codi d'aquest joc de proves: <pre> 1 func mew (A: array[10] of float, 2 B: array[15] of int) : float 3 var i: int 4 i = 0; 5 while i<10 and A[i]<B[i] do 6 write A[i]; 7 write " "; 8 i = i+1; 9 endwhile 10 write " --\n"; 11 if i==10 then i=0; endif 12 return A[i] + B[i]; 13 endfunc 14 15 func main() 16 var i,j,n: int 17 var b: bool 18 var A,B: array [15] of int 19 var X: array [10] of float 20 21 read n; 22 while n!=999 do 23 i = 0; 24 while i<15 do read A[i]; i=i+1; 25 endwhile 26 i = 0; 27 while i<10 do read X[i]; i=i+1; 28 endwhile 29 30 if A << X then 31 write "A smaller than B\n"; 32 else if X << A then 33 write "A is bigger than B\n"; 34 else 35 write "A and B are equal\n"; 36 endif 37 endwhile 38 39 B = A; 40 write A << B; write "\n"; 41 write B << X; write "\n"; 42 write not (B << X or X << B); 43 write "\n"; 44 write mew(X,B); write "\n"; 45 46 read n; 47 endwhile 48 endfunc </pre>	al llegir l'entrada següent: <pre> 333 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 0.0 1.0 2.0 3.0 5.0 5.0 6.0 7.0 4.0 3.0 222 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 111 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 0.0 1.0 2.0 3.0 5.0 5.0 6.0 7.0 8.0 10.0 555 0 1 2 3 4 5 6 7 8 9 8 7 6 5 4 0.0 1.0 2.0 3.0 4.0 5.0 4.0 3.0 2.0 1.0 999 </pre> ha de generar la sortida: <pre> A smaller than B 0 1 0 -- 0 A is bigger than B 0 0 0 -- 2 A smaller than B 0 1 0 -- 0 A is bigger than B 0 0 0 -- 0 </pre>
--	---

Informació important

FITXERS PER A L'EXAMEN: Al Racó (`examens.fib.upc.edu`) trobareu un fitxer `examen.tgz` amb el següent contingut:

- `final-lab-CL-2026.pdf`: Aquest document, amb l'enunciat i les instruccions.
- `jps`: Subdirectori amb jocs de proves (`jp_chkt_XX.asl`) i `jp_genc_YY.asl`), i la seva corresponent sortida esperada (`jp_chkt_XX.err`) per als jocs de proves de validació semàntica, `jp_genc_YY.in/.out` per als jocs de proves de generació de codi). En els JPs de generació, no es compara el codi generat, sinó la sortida que produeix la tVM en executar-lo.
- `common`: Subdirectori amb el mòdul auxiliar `SemErrors` ampliat amb el codi necessari per a l'examen.
- `evalua.sh`: Script que executa tots els jocs de proves i diu si se superen o no.
- `empaqueta.sh`: Script que crea un fitxer `examen-nom.cognom.tgz` amb la vostra solució. Aquest és el fitxer que cal pujar al Racó.

PASSOS A SEGUIR:

- Feu una còpia de les carpetes `asl`, `common` i `tvm` de la vostra pràctica a un directori `examen`.

```
mkdir examen
cp -r practica/asl practica/common practica/tvm examen/
```

- Canvieu al nou directori `examen`, i descomprimiu-hi el fitxer `examen.tgz` del Racó:

```
cd examen
tar -xzf examen.tgz
```

Això extreurà el contingut del paquet, **afegint** al vostre directori `examen` els fitxers llistats anteriorment.

IMPORTANT: Feu-ho en l'ordre especificat (primer una còpia de la vostra pràctica i després descomprimir el `.tgz`). Fer-ho en l'ordre invers causarà que us falti codi necessari a `common` i que els JPs no siguin els adequats.

- Trebal·leu normalment a la carpeta `examen/asl`.

```
cd asl
make pristine
make antlr
make -j 10
...
```

(Si la compilació és lenta per sobrecàrrega del servidor, podeu executar l'script `fast-make.sh`)

- Per executar tots els jocs de proves i veure si els passeu, executeu `../evalua.sh`.
- Per veure les diferències entre la sortida del vostre `asl` i la sortida esperada en un joc de proves concret de type check, podeu fer:

```
./asl ../jps/jp_chkt_XX.asl | diff -y - ../jps/jp_chkt_XX.err
```

(Podeu ignorar la línia "There are semantic errors: no code generated")
- Per veure les diferències entre la sortida del vostre `asl` i la sortida esperada en un joc de proves concret de generació de codi, podeu fer:

```
./asl ../jps/jp_genc_XX.asl > jp_XX.t
../tvm/tvm jp_XX.t < ../jps/jp_genc_XX.in | diff -y - ../jps/jp_genc_XX.out
```
- Executeu `../empaqueta.sh` per crear el fitxer d'entrega `../examen-USERNAME.tgz` que cal pujar al Racó. Els paquets creats sense usar aquest script seran qualificats com **NO PRESENTAT**.