

Compiladors: Examen final de laboratori.

8 de juny de 2023

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 Instrucció foreach (3 punts)

Volem afegir a l'ASL l'estructura repetitiva **foreach**, que permet iterar sobre els elements d'un array.

El format de la nova instrucció és: **foreach x in A do ... endfor**, on **x** és una variable i **A** és un array. La variable **x** ha d'estar declarada i ser del mateix tipus que els elements de **A**. Per exemple:

```
1 func main()
2   var a : array[5] of float
3   var b : array[10] of int
4   var z,p : float
5   var i : int
6
7   read p;
8   foreach z in a do
9     if z>p then
10      i = 0;
11      while i<10 do
12        b[i] = 0;
13        i = i + 1;
14      endwhile
15      p = z*2;
16    endif
17  endfor
18
19  write p;
20 endfunc
```

Joc de proves 1 (0.5 punts). Començarem modificant només la gramàtica per afegir els tokens necessaris i la instrucció **foreach**

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

```
1 func main()
2   var a : array[5] of float
3   var b : array[10] of int
4   var z,p : float
5   var i : int
6
7   read p;
8   foreach z in a do
9     if z>p and not b then
10      i = 0;
11      while i<10 do
12        b[i] = z;
13        i = i + 1;
14      endwhile
15      p = z*2;
16    endif
17  endfor
18
19  write p;
20  write q;
21 endfunc
```

hauria de produir la sortida:

Line 9:16 error: Operator 'not' with incompatible types.
Line 12:16 error: Assignment with incompatible types.
Line 20:8 error: Identifier 'q' is undeclared.

Joc de proves 2 (0.5 punts). Ara farem el Typecheck de la nova instrucció. Caldrà comprovar que la variable de control és compatible amb el tipus dels elements de l'array, i que la segona variable és realment un array. Observeu que un array d'enters es pot iterar amb una variable real, pero no al revés.

<p>Un cop fets els canvis, el segon joc de proves:</p> <pre> 1 2 func wasa(M : array[5] 3 of float): bool 4 var elem : int 5 var elem2 : float 6 var b : bool 7 var p,q,s : int 8 9 foreach elem in M do 10 write elem; 11 foreach x in M do 12 s = s + x; 13 endfor 14 endfor 15 16 foreach M in M do 17 elem2 = 1/M[0]; 18 foreach elem in x do 19 q = q * -elem; 20 endfor 21 endfor 22 23 foreach elem in b do 24 foreach x in y do 25 b = true; 26 endfor 27 foreach b in M do 28 a = M; 29 endfor 30 endfor 31 32 foreach elem2 in M do 33 write elem2 - 1/elem2; 34 endfor 35 endfunc 36 37 func main() 38 var a : array[5] of float 39 var b : array[10] of int 40 var z,p : float 41 var i,j : int 42 43 read p; 44 foreach z in b do 45 if z>p and not b then 46 a[i+1] = wasa(b); 47 p = z*2; 48 endif 49 endfor 50 51 write p; 52 write q; 53 endfunc </pre>	<p>hauria de produir la sortida:</p> <pre> Line 9:3 error: Foreach with incompatible variable and array elements. Line 11:14 error: Identifier 'x' is undeclared. Line 12:16 error: Identifier 'x' is undeclared. Line 16:3 error: Foreach with incompatible variable and array elements. Line 18:22 error: Identifier 'x' is undeclared. Line 23:3 error: 'foreach' requires an array parameter. Line 24:14 error: Identifier 'x' is undeclared. Line 24:19 error: Identifier 'y' is undeclared. Line 27:6 error: Foreach with incompatible variable and array elements. Line 28:9 error: Identifier 'a' is undeclared. Line 45:16 error: Operator 'not' with incompatible types. Line 46:15 error: Assignment with incompatible types. Line 46:22 error: Parameter #1 with incompatible types in call to 'wasas'. Line 52:8 error: Identifier 'q' is undeclared. </pre>
---	---

Joc de proves 3 (1 punt). A continuació generarem el codi corresponent al `foreach`. El codi generat ha de recórrer tots els elements de l'array, assignant a la variable de control el corresponent en cada iteració. Per ara, no cal que us preocupeu de les coercions enter-real, ja que no n'hi ha en aquest joc de proves.

<pre> 1 func main() 2 var i: int 3 var a : array[10] of int 4 var b : array[5] of float 5 var elemt : int 6 var elemf : float 7 8 i = 0; 9 while i < 10 do 10 read elemt; 11 a[i] = elemt*2; 12 i = i+1; 13 endwhile 14 15 foreach elemt in a do 16 i = 0; 17 while i < 5 do 18 b[i] = 1.0/(2*elemt+1); 19 i = i+1; 20 endwhile 21 22 write elemt; 23 write "\n"; 24 foreach elemf in b do 25 write " "; 26 write elemf*3.5; 27 endfor 28 write "\n"; 29 endfor 30 endfunc </pre>	<p>Amb això, el següent joc de proves:</p> <p>al llegir l'entrada següent:</p> <pre> 31 1 4 5 9 8 11 7 12 22 </pre> <hr/> <p>ha d'escriure:</p> <pre> 62 0.028 0.028 0.028 0.028 0.028 2 0.7 0.7 0.7 0.7 0.7 8 0.205882 0.205882 0.205882 0.205882 0.205882 10 0.166667 0.166667 0.166667 0.166667 0.166667 18 0.0945946 0.0945946 0.0945946 0.0945946 0.0945946 16 0.106061 0.106061 0.106061 0.106061 0.106061 22 0.0777778 0.0777778 0.0777778 0.0777778 0.0777778 14 0.12069 0.12069 0.12069 0.12069 0.12069 24 0.0714286 0.0714286 0.0714286 0.0714286 0.0714286 44 0.0393258 0.0393258 0.0393258 0.0393258 0.0393258 </pre>
---	---

Joc de proves 4 (1 punt). Finalment, permetrem coercions enter-real entre els elements de l'array i la variable de control.

Així passarem el darrer joc de proves:	que amb les dades d'entrada: 4 77 1 2 9 20 21 66 18 1
<pre> 1 func main() 2 var i: int 3 var x : float 4 var a : array[10] of int 5 var b : array[5] of float 6 var elemi : int 7 var elemf : float 8 9 i = 0; 10 while i < 10 do 11 read x; 12 b[i/2] = x*2; 13 a[i] = i; 14 i = i+1; 15 endwhile 16 17 foreach elemf in a do 18 write elemf; 19 write "\n"; 20 foreach x in b do 21 write " "; 22 write elemf - x/3.5; 23 endfor 24 write "\n"; 25 endfor 26 endfunc </pre>	produeix la sortida: 0 -44 -1.14286 -11.4286 -37.7143 -0.571429 1 -43 -0.142857 -10.4286 -36.7143 0.428571 2 -42 0.857143 -9.42857 -35.7143 1.42857 3 -41 1.85714 -8.42857 -34.7143 2.42857 4 -40 2.85714 -7.42857 -33.7143 3.42857 5 -39 3.85714 -6.42857 -32.7143 4.42857 6 -38 4.85714 -5.42857 -31.7143 5.42857 7 -37 5.85714 -4.42857 -30.7143 6.42857 8 -36 6.85714 -3.42857 -29.7143 7.42857 9 -35 7.85714 -2.42857 -28.7143 8.42857

2 Funcio *built-in* reduce (4 punts)

El segon exercici consisteix en afegir al llenguatge ASL amb una funció predefinida **reduce**, que redueixi un array a un element aplicant repetidament una funcio donada.

La sintaxi de la funció és: **reduce(a,f)**, on:

- **a** és un array i **f** és una funció.
- La funció té exactament dos paràmetres, del mateix tipus que els elements de l'array.
- El resultat de la funció és del mateix tipus que els elements de l'array.

El resultat que calcula l'operació **reduce(a,f)** es defineix com:

- Si l'array té un sol element, el resultat de **reduce** és $a[0]$.
- Si l'array té més d'un element, el resultat de **reduce** és:

$$reduce(a, f) = f(f(\dots f(f(f(a[0], a[1]), a[2]), a[3])\dots), a[n])$$

Per exemple, el programa següent calcula la suma i el màxim dels elements d'un vector:

```
1 func suma(a: float, b: float) : float
2   return a+b;
3 endfunc
4
5 func max(a: float, b: float) : float
6   if a>b then return a;
7   else return b;
8   endif
9 endfunc
10
11 func main()
12   var a : array[10] of float
13   var s : float
14
15   s = reduce(a,suma);
16   write s;
17   s = reduce(a,max);
18   write s;
19 endfunc
```

Joc de proves 5 (0.5 punts). El primer pas és afegir el token **reduce** a la gramàtica, i afegir les crides al *built-in* dins de les expressions.

El primer joc de proves:	genera els errors:
<pre> 1 func suma(a: float, b: float) : float 2 return a+b; 3 endfunc 4 5 func last(a: char, b: char) : char 6 if a>=b then return a; 7 else return b; 8 endif 9 endfunc 10 11 func main() 12 var a : array[10] of float 13 var s : float 14 var b : array[20] of char 15 var m : char 16 17 s = s + p; 18 s = reduce(a,suma); 19 write s; 20 c = reduce(b,last); 21 write c; 22 23 m = s*s - 1; 24 a = m[i] + b[0]; 25 endfunc </pre>	<pre> Line 17:10 error: Identifier 'p' is undeclared. Line 20:2 error: Identifier 'c' is undeclared. Line 21:8 error: Identifier 'c' is undeclared. Line 23:4 error: Assignment with incompatible types. Line 24:4 error: Assignment with incompatible types. Line 24:6 error: Array access to a non array operand. Line 24:8 error: Identifier 'i' is undeclared. Line 24:11 error: Operator '+' with incompatible types. </pre>

Joc de proves 6 (1 punt). A continuació farem la comprovació de tipus. Cal comprovar:

- Que els primer argument de **reduce** és un array. Trobareu a la versió ampliada de *SemErrors* l'error **arrayIsRequired**.
- Que el segon argument de **reduce** és una funció, amb exactament dos paràmetres del mateix tipus, i amb el resultat del mateix tipus que els paràmetres. Trobareu a la versió modificada de *TypesMgr* la funcio **reduceValidFunction** per fer la comprovació, i a *SemErrors*, el nou error **reduceInvalidFunction**.
- Que els elements de l'array són del mateix tipus que el resultat de la funció. Trobareu a *SemErrors*, el nou error **reduceIncompatibleArrayAndFunction**.

Si es troba cap dels errors anteriors en els arguments de **reduce**, el resultat del **reduce** es decorarà amb tipus **error**. Si tot es correcte, el tipus del resultat del **reduce** serà el tipus de retorn de la funció passada com a segon argument.

El segon joc de proves:	genera els errors:
<pre> 1 func suma(a: float, 2 b: float) : float 3 return a+b; 4 endfunc 5 6 func last(a: char, 7 b: char) : char 8 if a>=b then return a; 9 else return b; 10 endif 11 endfunc 12 13 func pff(a: int, 14 b: char) : float 15 return a+b; 16 endfunc 17 18 func void(x:bool) 19 write x; 20 endfunc 21 22 func main() 23 var a : array[10] of float 24 var s : float 25 var b : array[20] of char 26 var m : char 27 28 s = s + p; 29 s = reduce(a,suma); 30 c = reduce(b,last); 31 32 s = reduce(b,pff); 33 y = reduce(a,void); 34 m = reduce(s,suma); 35 36 m = s*s - 1; 37 a = m[i] + b[0]; 38 endfunc </pre>	<pre> Line 15:10 error: Operator '+' with incompatible types. Line 28:10 error: Identifier 'p' is undeclared. Line 30:2 error: Identifier 'c' is undeclared. Line 32:6 error: Reduce requires a proper function (T x T -> T). Line 33:2 error: Identifier 'y' is undeclared. Line 33:6 error: Reduce requires a proper function (T x T -> T). Line 34:6 error: 'reduce' requires an array parameter. Line 36:4 error: Assignment with incompatible types. Line 37:4 error: Assignment with incompatible types. Line 37:6 error: Array access to a non array operand. Line 37:8 error: Identifier 'i' is undeclared. Line 37:11 error: Operator '+' with incompatible types. </pre>

Joc de proves 7 (0.5 punts). El següent pas consistirà en generar codi per a l'operació de reducció. Començarem suposant que l'array només té un element, i per tant, el resultat del reduce és aquest element, independentment de quina sigui la funció.

El codi d'aquest joc de proves:	ha de generar la sortida:
<pre> 1 func suma(n: int, m: int): int 2 return n+m; 3 endfunc 4 5 func minim(n: int, m: int): int 6 if n < m then return n; endif 7 return m; 8 endfunc 9 10 func maxim(n: int, m: int): int 11 if n > m then return n; endif 12 return m; 13 endfunc 14 15 func maximC(c1: char, c2: char): char 16 if c1 > c2 then return c1; endif 17 return c2; 18 endfunc 19 20 func main() 21 var A: array[1] of int 22 var i, k: int 23 var r: float 24 var c: char 25 var AC: array[1] of char 26 27 A[0] = 231; 28 write "A[0]="; write A[0]; write "\n"; 29 30 r = reduce(A,suma)/10.0; write r; write "\n"; 31 k = reduce(A,minim); write k; write "\n"; 32 k = reduce(A,maxim); write k; write "\n"; 33 34 AC[0] = 'h'; 35 c = reduce(AC, maximC); 36 write c; write "\n"; 37 endfunc </pre>	<pre> A[0]=231 23.1 231 231 h </pre>

Joc de proves 8 (1 punt). Seguidament, usarem una funció de resultat constant, i per tant el resultat del reduce en un array de més d'un element, serà aquesta constant. Si l'array té un sol element, el resultat seguirà sent el primer element de l'array.

Observeu que per passar aquest joc de proves, és suficient amb cridar la funció un cop (amb el primer i segon elements de l'array), i no és necessari fer encara el bucle, ja que totes les crides successives donarien el mateix resultat.

<p>El codi d'aquest joc de proves:</p> <pre> 1 func catorze(n: int, m: int): int 2 return 14; 3 endfunc 4 5 func lletraF(c1: char, c2: char): char 6 return 'F'; 7 endfunc 8 9 func main() 10 var A: array[10] of int 11 var B: array[1] of int 12 var i, k: int 13 var r: float 14 var c: char 15 var AC: array[12] of char 16 var BC: array[1] of char 17 18 i = 9; 19 while i >= 0 do 20 read A[i]; 21 write "A["; write i; write "]="; 22 write A[i]; write "\n"; 23 i = i - 1; 24 endwhile 25 26 r = reduce(A,catorze)/10.0; 27 write r; write "\n"; 28 k = reduce(A,catorze); 29 write k; write "\n"; 30 31 B[0] = A[3]; 32 write reduce(B,catorze); write "\n"; 33 B[0] = A[6]; 34 write reduce(B,catorze); write "\n"; 35 36 i = 0; 37 while i < 12 do 38 // read chars from input 39 // (skips whitespaces) 40 read AC[i]; 41 i = i + 1; 42 endwhile 43 c = reduce(AC, lletraF); 44 write c; write "\n"; 45 46 BC[0] = AC[7]; 47 c = reduce(BC, lletraF); 48 write c; write "\n"; 49 endfunc </pre>	<p>al llegir l'entrada següent:</p> <pre> 6 3 22 99 56 12 7 9 66 4 hola que tal </pre> <hr/> <p>ha de generar la sortida:</p> <pre> A[9]=6 A[8]=3 A[7]=22 A[6]=99 A[5]=56 A[4]=12 A[3]=7 A[2]=9 A[1]=66 A[0]=4 1.4 14 7 99 F t </pre>
--	---

Joc de proves 9 (1 punt). Finalment, generarem el codi per a aplicar la funció repetidament a cada element de l'array. Per fer-ho, el codi generat haurà de:

- Crear un temporal per acumular el resultat i inicialitzar-lo amb el valor del primer element de l'array.
- Cridar la funció amb el valor de l'acumulador com a primer argument, i el del següent element de l'array com a segon argument, i guardar el resultat com a nou valor de l'acumulador. Repetir per a cada element de l'array.

<p>El codi d'aquest joc de proves:</p> <pre> 1 func suma(n: int, m: int): int 2 return n+m; 3 endfunc 4 5 func minim(n: int, m: int): int 6 if n < m then return n; endif 7 return m; 8 endfunc 9 10 func resta(n: int, m: int): int 11 return n-m; 12 endfunc 13 14 func AND(a: bool, b: bool) : bool 15 return a and b; 16 endfunc 17 18 func maximC(c1: char, c2: char): char 19 if c1 > c2 then return c1; endif 20 return c2; 21 endfunc 22 23 func main() 24 var A: array[10] of int 25 var B: array[10] of bool 26 var i, k: int 27 var r: float 28 var c: char 29 var AC: array[13] of char 30 31 i = 9; 32 while i >= 0 do 33 read A[i]; 34 B[i] = (A[i]%2 == 0); 35 i = i - 1; 36 endwhile 37 38 r = reduce(A,suma)/10.0; 39 write r; write '\n'; 40 k = reduce(A,minim); 41 write k; write '\n'; 42 k = reduce(A,resta); 43 write k; write '\n'; 44 45 i = 0; 46 while i < 13 do 47 // read chars from input 48 // (skips whitespaces) 49 read AC[i]; 50 i = i + 1; 51 endwhile 52 c = reduce(AC, maximC); 53 write c; write "\n"; 54 55 if reduce(B,AND) then 56 write "all even\n"; 57 else 58 write "some odd\n"; 59 endif 60 endfunc </pre>	<p>al llegir l'entrada següent:</p> <pre> 22 5 71 9 8 17 15 1 43 16 molt i tu que </pre> <hr/> <p>ha de generar la sortida:</p> <pre> 20.7 1 -175 u some odd </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-2023.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 els mòduls auxiliars `SemErrors` i `TypesMgr` ampliat amb el codi necessari per a l'examen.
- `tvm`: Subdirectori amb la màquina virtual.
- `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` i `common` de la vostra pràctica a un directori `examen`.

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

- Treballeu normalment a la carpeta `examen/asl`.

```
cd asl  
make antlr  
make  
...
```

(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**.