Modificació de la Pràctica de CL 1 de juny de 2007

Tot seguit us expliquem la modificació de la pràctica que haureu de realitzar. Us donem també una petita guia d'ajuda que no heu de seguir forçosament. Aquesta guia conté informació de cóm s'avaluarà la modificació que us demanem, basant-se en uns jocs de proves que podeu trobar al racó en `examens.fib.upc.edu` accedint a l'entrega corresponent. Aquests jocs de proves estan inclosos al fitxer `infoexamen.tar`, el contingut del qual es pot extreure fent `tar xf infoexamen.tar`, per poder executar després `tar xf jp.tar`. El fitxer `infoexamen.tar` conté també d'altres fitxers que us permetran realitzar una autoavaluació. Al final d'aquest document us expliquem cóm fer aquesta autoavaluació i també l'entrega via el racó. És *molt recomanable* crear un directori nou `/tmp/cl`, copiar allà la vostra pràctica, descomprimir allà també `infoexamen.tar` i `jp.tar`, realitzar una entrega desde el principi, i anar fent entregues a mesura que la modificació de la vostra pràctica vagi superant més jocs de proves.

Sisplau, feu l'autoavaluació i l'entrega preliminar al principi, i també un cop hagueu acabat la pimera modificació. Assegureu-vos, sobretot, de fer l'última entrega 15 minuts abans de l'hora límit per a completar l'examen, moment en el que el racó es tancarà.

Volem afegir al llenguatge CL la possibilitat de fer servir expressions amb l'operador d'exponenciació, `e1^e2`, on `e1` i `e2` són dues expressions aritmètiques. El resultat d'avaluar `e1^e2` és el resultat de l'operació potència on la base és el resultat d'avaluar `e1` i l'exponent és el resultat d'avaluar `e2`.

Aquí teniu un exemple:

```
Program
   Vars
     a int
     b int
     y real
     z real
   Endvars

   a:=5
   b:=2
   y:= b^a
   z:= y^(a div b)
   writeln(z)
EndProgram
```

L'execució del programa anterior escriuria per pantalla:

```
1024.000000
```

Guia i puntuació:

- **3 punts** de la nota s'obtenen en passar el joc de proves genèric que es troba al fitxer `jpbasic1`, juntament amb la sortida esperada. La resta ja depèn estrictament de les modificaciones que s'especifiquen a continuació.
- Primer cal retocar l'anàlisi sintàctica. Necessitem un nou token `POTENCIA`, i introduir a les expressions bàsiques una nova regla `exppot : expsimple POTENCIA expsimple` expressada de la manera adequada per tal que no es produeixin ambiguitats, i es permeti associativitat esquerra de l'operació. També cal tenir en compte que l'operador exponenciació ^ té més prioritat que l'operador producte $*$, i que caldrà crear un node a l'arbre sintàctic d'un nou tipus: `Nexppot` o "expresió potència".

- A l'anàlisi semàntica d'una expressió `e1^e2` haurem de fer una comprovació dels tipus dels operands i assignar un tipus a l'expressió resultant. La base `e1` pot ser de tipus enter o de tipus real i l'exponent `e2` ha de ser de tipus enter. Donat que l'exponent podria ser un enter negatiu i no ho podem saber en temps de compilació, considerarem que el tipus resultant de `e1^e2` és sempre real.

  Amb totes aquestes consideracions, en fer el `TypeCheck` d'un node `Nexppot`, haurem de considerar dos tipus addicionals d'errors:

  ```
  Potencia con base no numerica.
  Potencia con exponente no entero.
  ```

  (1 punt) Això ens permetria passar un joc de proves com el que segueix:

```
 1: PROGRAM
 2:   VARS
 3:     x int
 4:     a int
 5:     y real
 6:     z real
 7:     b bool
 8:   ENDVARS
 9:   IF NOT x=2 THEN
10:     WHILE NOT 3>4 DO
11:       IF (---4^(y+3)>0 = b) THEN
12:         x:= 4^(x*4)
13:       ENDIF
14:     ENDWHILE
15:     IF NOT -3=-3 OR (b OR b AND b) THEN
16:       z:= (b^a*2)^x
17:     ENDIF
18:   ELSE
19:     x:=6/3
20:   ENDIF
21: ENDPROGRAM
```

  Que hauria de donar els següents errors:

```
L. 9: Operador Not con tipos incompatibles.
L. 9: Operador = con tipos incompatibles.
L. 10: Operador Not con tipos incompatibles.
L. 10: Operador > con tipos incompatibles.
L. 11: Potencia con exponente no entero.
L. 12: Asignacion con tipos incompatibles.
L. 15: Operador Not con tipos incompatibles.
L. 15: Operador = con tipos incompatibles.
L. 16: Potencia con base no numerica.
L. 19: Asignacion con tipos incompatibles.
```

  (1 punt) També hem de passar el següent joc de proves:

```
 1: PROGRAM
 2:   VARS
 3:     x int
 4:     a int
 5:     y real
 6:     z real
 7:     b bool
 8:   ENDVARS
 9:   IF x^(y*2)-z^3 > 0 THEN
10:     x := a^x + 3 div 5
```

```
11:   ELSE
12:     z := x^3^a-(z-2)^(y/2)
13:   ENDIF
14: ENDPROGRAM
```

Amb el resultat:

```
L. 9: Potencia con exponente no entero.
L. 10: Asignacion con tipos incompatibles.
L. 12: Potencia con exponente no entero.
```

- Considerem ara la generació de codi i tractem successivament tres casos, de menor a major dificultat:

Cas 1: Base real, exponent positiu.
La seqüència d'instruccions QCode que cal generar podria seguir aquest esquema:

```
// inicialitza acumulador a 1
pcon 1
itor
// calcula base
CodeGenerateRightExpr(a->h1)
// calcula exponent
CodeGenerateRightExpr(a->h2)
// etiqueta inici bucle
label1:
// avalua si l'exponent ha arribat a zero
pusp
indi
// si es zero, sortir del bucle (label2)
plab label2
fjmp
// decrementa exponent
pcon 1
subi
// obte adreca de l'acumulador per fer un store al final
pusp
pcon -8
addi
// crea copia del valor de l'acumulador
pusp
pcon -12
addi
indi
// obte valor de la base
pusp
pcon -12
addi
indi
// multiplica copia de l'acumulador per la base i guarda resultat a l'acumulador
mulr
stor
// torna al principi del bucle
plab label1
ujmp
// etiqueta de sortida del bucle
label2:
// allibera espai de la base i exponent, deixa acumulador a la pila
pcon -8
pspa
```

(2 punts) Això ens permetria passar un joc de proves com el que segueix: .

```
 1: PROGRAM
 2:   VARS
 3:     x real
 4:     a int
 5:     y int
 6:     z real
 7:     b bool
 8:   ENDVARS
 9:   x:=3
10:   y:=2
11:   a:=4
12:   z:= x^(y*2)-x^3
13:   IF x^(y*2)-x^3 > 0 THEN
14:     z := x^a + 3 div 5
15:   ELSE
16:     z := x^3^a-(z-2)^(a div 2)
17:   ENDIF
18:
19:   writeln(z)
20: ENDPROGRAM
```

Que quan executem sobre la Q-machine el codi generat al compilar-lo, dóna com a resultat:

```
81.000000
```

Cas 2: Base entera, exponent positiu.
Per cobrir aquest cas, cal ampliar la seqüència generada en l'apartat anterior per tal que si la base és entera, es converteixi a real el seu valor. Simplement cal afegir, si s'escau, la instrucció `itor` darrera de la crida a `CodeGenerateRightExpr` corresponent a la base.

(1 punts) Amb això podrem passar un joc de proves on l'única diferència amb l'anterior és que la variable x ara és de tipus `enter`. El resultat produit també és `81.000000`.

```
 1: PROGRAM
 2:   VARS
 3:     x int
 4:     a int
 5:     y int
 6:     z real
 7:     b bool
 8:   ENDVARS
 9:   x:=3
10:   y:=2
11:   a:=4
12:   z:= x^(y*2)-x^3
13:   IF x^(y*2)-x^3 > 0 THEN
14:     z := x^a + 3 div 5
15:   ELSE
16:     z := x^3^a-(z-2)^(a div 2)
17:   ENDIF
18:
19:   writeln(z)
20: ENDPROGRAM
```

Cas 3: Base real o entera, exponent negatiu. Si l'exponent és negatiu (cosa que es coneix en temps d'execució), es pot aplicar que: $x^{-a} = \frac{1}{x^a} = (\frac{1}{x})^a$. Per tant, cal generar codi que comprovi si l'exponent és negatiu, i si ho és, (a) canviï el signe de l'exponent, (b) inverteixi el valor de la base, i (c) segueixi normalment amb el bucle de càlcul de la potència. Si l'exponent és positiu, cal anar directament al pas (c).

(2 punts) Això ens permetrà passar el següent joc de proves:

```
 1: PROGRAM
 2:   VARS
 3:     x int
 4:     a int
 5:     y int
 6:     z real
 7:     b bool
 8:   ENDVARS
 9:
10:   x:=3
11:   y:=2
12:   a:= -4
13:   z:= x^a
14:
15:   writeln(z)
16:
17:   IF x^(y*2)-x^3 < 0 THEN
18:     z := x^a + 3 div 5
19:   ELSE
20:     z := x^a^3-(z-2)^(a div 2)
21:   ENDIF
22:
23:   writeln(z)
24: ENDPROGRAM
```

El qual, a l'executar-se, dóna com a resultat:

```
0.012346
-0.253113
```

Autoevaluació:

S'ha d'executar `./fesentrega.sh` per a crear automàticament un fitxer anomenat `entrega.tar`. Si aquesta comanda emet algún missatge aviseu inmediatament a algun dels professors de l'assignatura, inclús si després tot sembla funcionar, doncs el més probable és que no realitzeu la entrega correctament i us quedeu sense avaluació.

Si aleshores executeu `./checker.sh`, al cap d'una estona us indicarà quins jocs de proves heu superat juntament amb la nota assignada fins al moment. El `checker` és una ajuda bastant fidedigna per a comprovar que les modificacions funcionen, però és responsabilitat vostra comprovar que es generen els errors esperats, ni més ni menys.

Entrega:

Connecteu-vos a pràctiques via web a `examens.fib.upc.edu`. Heu d'entregar el fitxer `entrega.tar` que heu creat tal com s'indica a l'autoevaluació.