Compiladors (CL)
GEI (2017-18)

Pràctica 2: Expressions Booleanes i Flux de control

Objectiu

Afegir el reconeixement d'expressions booleanes i del flux de control a la part frontal del compilador iniciat a la Pràctica 1.

Aprenentatges

Utilització conjunta de Flex, Bison i Symtab.

Llenguatge font

Literals i comentaris:

- S'admeten literals enters (e.g. *123*, *-45*), reals (e.g. *3.1416*, *-0.5*), cadenes (e.g. "Hola") i booleans (**true**, **false**).
- Les cadenes estan delimitades per dobles cometes (″), i no poden ocupar més d'una línia.
- Poden haver comentaris com en Java, i.e. comentaris que comencen amb `//` fins a final de línia, i comentaris que es poden estendre vàries línies, iniciats amb `/*` i finalitzats amb `*/`.

Identificadors i expressions:

- Els identificadors i expressions tenen un tipus, que pot ser: enter, real, cadena o booleà.
- Les expressions aritmètiques poden estar formades per literals enters, literals reals, cadenes de caràcters, identificadors (de tipus no booleà), parèntesis, l'operador de concatenació de cadenes (+), operadors aritmètics unaris (+, −) i operadors aritmètics binaris (+, −, *, /, **mod**, **).
- L'ordre de precedència dels operadors aritmètics binaris és: potència (**) més precedència que producte, divisió i mòdul (**mod**), producte igual precedència que divisió i mòdul, aquests major precedència que suma i resta, i suma igual precedència que resta.
- Els operadors aritmètics unaris (manteniment i canvi de signe) tenen igual precedència que la suma i la resta binàries.
- Les expressions booleanes es formen mitjançant els operadors relacionals (>,>=, <, <=, =, <>) aplicats a expressions aritmètiques, i mitjançant els operadors booleans (**not**, **and**, **or**) aplicats a expressions booleanes.
- L'ordre de precedència dels operadors booleans és: **not** major precedència que **and**, i **and** major que **or**.
- Els operadors relacionals (<, <=, >,>=) sobre cadenes generen un error de tipus.
- L'avaluació de les expressions booleanes es pot fer tant en curtcircuit com sense curtcircuit. Per simplicitat, es recomana fer-ho sense curtcircuit en mode

calculadora, i en curtcircuit en mode programa. En curtcircuit, es deixa d'avaluar la resta d'una expressió booleana tan bon punt es coneix el resultat de tota l'expressió. Per exemple, si sabem que en un cert punt del programa `a` és més gran que `b`, aleshores l'expressió booleana `a > b or c < d` segur que és certa independentment del valor de `c < d`, i per tant no cal avaluar `c < d`. Sense curtcircuit s'avaluaria tota l'expressió encara que no fes falta.

Sentències:

- Les sentències poden ser: expressions aritmètiques, expressions booleanes, assignacions, condicionals o iteratives.
- Les expressions aritmètiques, booleanes i assignacions ocupen una línia.
- Les assignacions són del tipus

  id := expressió

  on l'expressió pot ser aritmètica o booleana, i *id* és un identificador.
- En una assignació, el tipus de l'identificador *id* és igual al tipus del resultat de l'expressió corresponent.
- Si s'operen expressions aritmètiques del mateix tipus, el resultat és del mateix tipus.
- Si s'operen aritmèticament enters amb reals, el tipus resultant és real.
- Si es concatena (amb l'operador +) una expressió numèrica amb una cadena, el resultat és una cadena.
- Els identificadors que apareguin en una expressió qualsevol han d'haver estat prèviament inicialitzats, ja que cal conèixer els seus tipus.
- La sintaxi de les sentències condicionals sense alternativa és:

  if (*expressió_booleana*) **then**
  llista_de_sentències
  fi
- La sintaxi de les sentències condicionals amb alternativa és:

  if (*expressió_booleana*) **then**
  llista_de_sentències
  else
  llista_de_sentències
  fi
- La sintaxi de les sentències condicionals amb alternatives múltiples és:

  if (*expressió_booleana*) **then**
  llista_de_sentències
  elsif (*expressió_booleana*) **then**
  llista_de_sentències
  elsif (*expressió_booleana*) **then**
  llista_de_sentències
  ...
  else
  llista_de_sentències
  fi
- La sintaxi de les sentències iteratives indefinides amb condició a l'entrada és:

  while (*expressió_booleana*) **do**
  llista_de_sentències
  done

- La sintaxi de les sentències iteratives indefinides amb condició a la sortidaa és:

  repeat
  llista_de_sentències
  until (*expressió_booleana*)
- La sintaxi de les sentències iteratives definides és:

  for (*id* **in** *rang*) **do**
  llista_de_sentències
  done
- La variable *id* de la iterativa definida ha de ser un identificador nou, no utilitzat anteriorment, i serà de tipus enter.
- Els *rang*s de les iteratives són del tipus:

  expressió_aritmètica_entera..expressió_aritmètica_entera

  Si el valor de l'expressió aritmètica de l'esquerra és més gran que el de la dreta, el rang és buit i no s'executen mai les sentències internes de la iterativa.

Opcions:

- Podeu definir sentències opcionals, amb la sintaxi que vosaltres vulgueu, per incloure al llenguatge noves funcionalitats, com per exemple:
  - Definir i accedir a estructures tipus **array**, **list**, **record**/**struct**, etc.
  - Condicionals tipus **switch**
  - Iteratives tipus **do-while**, loop infinit, instruccions **exit/break** i **continue**
  - Procediments i/o funcions, ja siguin en estructura plana o encaixada, amb pas de paràmetres, etc.

Programa:

- Distingim entre dos modes de funcionament, el mode calculadora (**calc on**) i el mode programa (**calc off**). La sintaxi d'un programa és:

  calc *on_or_off*
  llista_de_sentències

  Per tant, la instrucció **calc** només pot aparèixer al principi del programa.
- En mode calculadora no pot haver-hi sentències iteratives ni condicionals, i l'avaluació d'expressions booleanes és optativa.

Sortides

En mode calculadora:

- Les assignacions mostraran el nom, el seu tipus i el seu valor.
- Les expressions mostraran el seu tipus i el seu valor.

En mode programa:

- Les assignacions i expressions mostraran només el seu tipus, ja que el valor només es pot conèixer en temps d'execució.

Altres sortides:

- Convé anar guardant en un arxiu de log cada producció de la gramàtica reconeguda, per així controlar el progrés i correcció de les anàlisis lèxica i sintàctica. També es pot guardar al mateix arxiu qualsevol altre missatge informatiu.

- Quan hi ha un error al codi font s'ha d'emetre un missatge d'error indicant la posició actual a l'arxiu font, i si l'error és lèxic, sintàctic o semàntic. Mentre més informació es doni dels errors, millor.

Exemples

Entrada en mode calculadora:

```
calc on
// assignacions
x := 2.3
i := 5
z := i * (x + i) - i / 7.2
b := z > 3 or not (i > 0 and i <= 10) or false
s := "Hola"
// expressions
x
i + i * 2
s + ” ” + (s + i) + x
b or not b
```

Entrada en mode programa:

```
calc off

// nested sentences
total := 0.0
i := 1
while (total  < 1000.0) do
   if (i = 1) then
      total := total * 2
   else
      total := total - 1
   fi
   i := -i
done
total

// números de Fibonacci
n := 50
x := 1
x
y := 1
y
for (index in 3..n) do
   fib := x + y
   x := y
   y := fib
   fib
done
```

Lliurament

- Aquesta pràctica és individual.
- El lliurament es farà via Moodle, en les dates indicades al mateix.
- Cal lliurar un arxiu comprimit que contingui:
    - tot el codi font (sense arxius generats en el procés de compilació)
    - exemples i la seva sortida corresponent
    - scripts o Makefile per compilar-ho tot, executar el programa amb els exemples, i netejar-ho tot llevat dels arxius font i els d'exemple.
    - arxiu README (en format txt o pdf) amb instruccions per la compilació i execució, i descripció de tot allò que vulgueu destacar de la vostra pràctica (decisions de disseny, funcionalitat addicional, limitacions, etc.).
- El nom de l'arxiu lliurat ha de contenir el vostre nom i primer cognom.