Compiladors (CL)
GEI (2024-25)

Pràctica 1: Calculadora

Objectiu

Fer la part frontal d'un compilador que interpreti programes senzills.

Aprenentatges

Comprendre els conceptes bàsics d'un compilador: taula de símbols, anàlisi lèxica, anàlisi sintàctica, atributs, anàlisi semàntica i comprovació de tipus. Utilització conjunta de Flex, Bison i Symtab.

Llenguatge font

Literals i comentaris:

- S'admeten literals enters (e.g. *123*, *045*), reals (e.g. *3.1416*, *0.5e-04, 1.2345e3*), 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 */. També es permet comentar fins a final de línia amb el caràcter #

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 (+, −, *, /,%, **).
- L'ordre de precedència dels operadors aritmètics binaris és: potència (**) més precedència que producte, divisió i mòdul (**%**), 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**.
- L'avaluació de les expressions booleanes es pot fer tant en curtcircuit com sense curtcircuit. Per simplicitat, es recomana fer-ho sense curtcircuit. 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 `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.

- Les precedències entre operadors (tant aritmètics com booleans) s'han d'implementar a través de la definició de la gramàtica, evitant utilitzar els operadors *%left* i *%right* que proporciona el bison.

Sentències:

- Les sentències poden ser: expressions aritmètiques, expressions booleanes o assignacions.
- 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.

Funcionalitats:

- La calculadora ha de poder suportar funcions trigonomètriques (sin, cos, tan)
- Suporta funció de longitud de cadena LEN i extracció de subcadena SUBSTR(*cadena*; *inici*; *longitud*)
- Suporta l'ús de constants predeterminades PI, E
- Permet representació en mode octal (OCT), binari (BIN), hexagesimal (HEX) o decimal, que és el format per defecte (DEC)

Programa:

- La sintaxi d'un programa és:
  llista_de_sentències

Opcions:

- Podeu afegir opcions diverses a la calculadora: taules, llistes, registres, nous tipus de números (complexes, quaternions, etc.), i qualsevol altre opció que es pugui encabir dins del concepte de calculadora

Sortides

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

- S'ha d'evitar barrejar la sortida de la calculadora (expressions i assignacions) de la sortida de log.
- 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.

Exemple d'arxiu d'entrada

```
// assignacions
x := 1.5
i := 5
z := i * (x + i) - i / 4.0
s := "Hola"
b := z > 3 or not (i > 0 and i <= 10) or false

// expressions bàsiques
x
i
z
s
b

/*
* expressions
*/
i + i * 2
-i + x
s + (s + i) + "--" + x + b
b or not b
```

Lliurament

- Aquesta pràctica és individual.
- El lliurament es farà via Moodle, en les dates indicades.
- 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.