Pràctica de Compilador 2018-2019
Compilador de LANS

Breu Informe de la Pràctica

Marc Cané Salamià
Enric Rodríguez Galán

Professor: Jordi Coll Caballero

Feina feta

Pel que respecte a la feina realitzada durant el transcurs de la pràctica, s'ha intentat fer la part obligatoria i petits fragments de la part opcional. De la part opcional, s'ha implementat parcialment tot el que fa referència als àlies i poc més. Degut a que sabiem que el temps jugava en la nostre contra, vam decidir prioritzar la elaboració de la major part possible de la feina obligatoria.

Modificacions entre entregues

El canvi més notable que hem fet des de la primera entrega és que ara el lexer (regles lèxiques) i el parser (regles sintàctiques) estan a dins del mateix fitxer .g4 per estalviar-nos problemes.

De la part lèxica, hem mogut el token *TK_BOOLEAN* per problemes de prioritats (detectava els valors booleans com a identificadors) i hem actualitzat els seus posibles valors a 'cert' i 'fals', ja que anteriorment eren 'true' i 'false' i el llenguatge no els defineix així.

De la part sintàctica, vam arreglar la gramàtica per a que no tingues associativitat definida (teniem aquesta opció comentada i vam redefinir una gramàtica amb associativitat dreta errònia).

No vam modificar cap més aspecte de la primera entrega.

Estructura de la taula de símbols

La taula de símbols està formada per objectes de tipus Registre (no hem fet cap jerarquia de classes amb herència ni res per l'estil). Aquest objecte conté els camps següents:

- `String text` : Camp que emmagatzema el contingut del token que representa aquest símbol. Per exemple, una variable tindria escrit el seu identificador aquí, com les constants, les funcions, ...
- `String supertype` : Tipus general del registre. Ens diu si es una variable, una constant, una funció, un alias...
- `String type` : Tipus més especific del registre. Ens diu si es de tipus enter, real, boolea… Més envall es parla amb més detall sobre aquest camp, ja que té certs matisos que cal saber.
- `int line` : Línia en el codi font on s'ha generat aquest registre.
- `int pos` : Columna en el codi font on s'ha generat aquest registre.
- `int intval` : Valor enter si en té. A data de la redacció d'aquest document aquest camp no s'ha usat per res. El vam afegir el primer dia per si ens era necessari, però al final no l'hem utilitzat.
- `Long dir` : Direcció del element al Bytecode. Si es una variable, serà el seu número de variable. Si es una constant, serà el seu id. Etc.

Per entendre millor quin *type* i *supertype* té cada tipus de registre que creem, hem elaborat la següent taula explicativa on s'expressen els possibles valors que aquests camps poden agafar:

Element	Supertipus	Tipus
Constant	`CONSTANT_SUPERTYPE`	Tipus bàsics
Variable	`VARIABLE_SUPERTYPE`	Tipus bàsics, o identificador del tipus no bàsic
Funció	`FUNCTION_SUPERTYPE`	Tipus bàsics
Acció	`ACTION_SUPERTYPE`	
Tupla	`TUPLE_SUPERTYPE`	Identificador de la tupla (nom)
Àlies	`ALIAS_SUPERTYPE`	Tipus bàsic al que fa àlies.
Vector	`VECTOR_SUPERTYPE`	Tipus bàsic dels elements del vector

Com es pot veure al codi, la part opcional no està implementada, i aquest disseny pot no ésser el més adequat.

Aspectes rellevants de la implementació

Un aspecte que cal mencionar ja que sinó a l'hora de corregir la pràctica pot portar molts mals de cap és el fet de que no propaguem l'error de tipus a les expressions. És a dir, si tenim una expressió en la que es produeix un error de tipus, aquest no es propaga cap a la resta de l'expressió.

Per exemple, en la expressió 4.5\2+3, a l'hora de mirar el dos tipus dels operands de la suma aquesta no serà conscient de que l'operació de divisió entera ha donat un error de tipus, i suposarà que aquesta ha anat bé. Això ho implementem propagant el tipus del resultat de l'operació al operador esquerre de la següent, tant entre nivells diferents de la jerarquia com en el mateix nivell, i només validem les comprovacions necessàries entre els dos operadors quan és estricament necessari amb aquest tipus propagat. Altrament, es comprovaria només la compatibilitat entre l'operand dret i la operació, ja que l'operand esquerre ja hauria estat degudament validat en el "pas" anterior.

D'aquesta manera el que aconseguim és mostrar un sol missatge d'error per cada punt d'origen d'errors, i aquest error no fa que la resta de comprovacions de tipus de les expressions donguin error de tipus.

Un altre aspecte a tenir en compte és el fet de les variables no inicialitzades. És responsabilitat del programador inicialitzar-les variables abans de usar-les.

També cal dir que és possible que hi hagi antics fragments de codi comentats amb caràcter acadèmic (per veure que hem fet malament i com no tornar-ho a fer).

Apart d'això, res més a dir.