Compiladors

Pràctica 3: Creació d'un compilador complet

NEUS OLLER MATAS

Enginyeria Informàtica

Universitat Rovira i Virgili

1 Introducció

Aquesta pràctica implementa un compilador complet per a un llenguatge ampliat basat en la pràctica 2. S'ha afegit suport per a expressions booleanes, sentències condicionals i iteratives amb diverses opcions de control de flux. El compilador genera codi intermedi en format de Tres Adreces (C3A), seguint l'especificació proporcionada.

2 Instruccions de Compilació

```
bison -d parser.y

flex scanner.l

make

./calculator prova/prova_nom.txt

make clean    # netejar
```

3 Decisions de disseny

3.1 Gestió de la taula de símbols

Es manté una taula de símbols personalitzada per a variables definides per l'usuari. Les variables temporals generades no s'inclouen a la taula per simplificar-ne la visualització. Cada entrada inclou informació sobre el tipus (INTEGER, FLOAT, BOOLEAN, ARRAY, etc.) i el valor corresponent.

3.2 Generació de codi de tres adreces

S'utilitza la funció addQuad() per generar quads. Les expressions booleanes es gestionen amb back-patching, permetent un control precís del flux. Les sentències condicionals i iteratives es tradueixen a salts condicionals amb etiquetes generades dinàmicament.

3.3 Expressions literals

Les expressions que només inclouen literals s'avaluen en temps de compilació, evitant la generació innecessària de quads.

3.4 Estructura de la gràmatica

La gramàtica està dissenyada per complir amb les especificacions del llenguatge, evitant l'ús de %left i %right per definir precedències. S'han creat regles jerarquitzades per establir l'ordre d'avaluació d'operadors. Reutilitzant el codi de la primera pràctica.

3.5 Iteracions i bucle

3.5.1 REPEAT-DO-DONE

Aquest tipus d'iteració s'executa almenys una vegada i repeteix el bloc d'instruccions fins que es compleix una condició específica. Es crea un comptador d'iteracions que s'inicialitza al principi del bucle i que s'incrementa automàticament a cada iteració. Quan la condició final es compleix, el bucle es deté. El codi intermedi generat utilitza quàdruples per gestionar el comptador i els salts condicionals, assegurant-se que es compleix el flux correcte.

3.5.2 WHILE-DO

El bucle while-do avalua la condició booleana abans de cada iteració. Si la condició és certa (TRUE), s'executa el bloc d'instruccions i es torna a avaluar la condició. Si la condició és falsa (FALSE), el bucle finalitza. Aquesta estructura es tradueix en codi intermedi amb salts condicionals que gestionen la repetició del bloc segons l'avaluació de la condició.

3.5.3 DO-UNTIL

A diferència del while-do, el bucle do-until assegura que el bloc d'instruccions s'executi almenys una vegada abans d'avaluar la condició. La condició es verifica al final de cada iteració, i el bucle es repeteix fins que la condició es compleix. En el codi intermedi, això es reflecteix amb un salt condicional al final del bloc, que controla si es torna a executar el bucle o si finalitza.

3.5.4 FOR

El bucle for gestiona automàticament una variable d'índex dins d'un interval especificat. A cada iteració, la variable d'índex s'incrementa o decreix segons l'interval definit, i el bucle finalitza quan es compleix la condició de límit. El codi intermedi generat inclou quàdruples per inicialitzar la variable, comparar-la amb el límit i actualitzar-ne el valor a cada iteració, seguint estrictament la semàntica del bucle.

3.6 Condicionals

3.6.1 IF-THEN-ELSE

El condicional $\boldsymbol{if - then - else}$ introdueix un bloc alternatiu d'instruccions que s'executa si la condició és falsa. En aquest cas, es generen dos salts condicionals: un per accedir al bloc THEN si la condició és certa, i un altre per accedir al bloc ELSE si la condició és falsa.

3.6.2 IF-THEN

El condicional if-then avalua una condició booleana i, si aquesta és certa (TRUE), executa el bloc d'instruccions associat al THEN. Si la condició és falsa (FALSE), no es fa res i el programa continua.

Pel que fa a la implementació tècnica, el compilador utilitza backpatching per gestionar les llistes de salts pendents, com les llistes de trues (truelist) i les llistes de falses (falselist). Això permet connectar les instruccions generades parcialment amb les etiquetes finals un cop completat el codi intermedi. Tot i que els condicionals funcionen correctament, el suport per expressions booleanes amb curtcircuit no està implementat, fet que provoca que totes les condicions es verifiquin completament, independentment del resultat parcial.

4 Funcions pròpies

- $addQuad()$: per afegir una nova quàdruple a la llista $quad\_list$. Rep un nombre variable d'arguments i els emmagatzema en una estructura de tipus quàdruple (quad). La quàdruple es guarda a la llista, i l'índex de quàdruples actual (currQuad) s'incrementa. Finalment les afegeix al codi intermedi a la llista de quads (quads_list)

- $newTemp()$: Genera noms únics per a variables temporals durant la generació de codi intermedi. Si es proporciona un prefix, aquest s'afegeix al nom del temporal, cosa que resulta útil per identificar temporals associats a Arrays o altres estructures.

- $arithmeticCalc()$: Calcula expressions aritmètiques entre dues variables, gestionant tipus i generant instruccions adequades en codi de tres adreces.

- Tractament de la potència; $powFunction()$, $potenciaRecursiva()$:
  - $powFunction()$: calcula la potència d'un nombre elevant a un exponent.
  - $potenciaRecursiva()$: es cridada per la funció $powFunction()$ i calcula la potència de manera recursiva, ja que en cas d'haver estat més d'una vegada elevat, doncs que sigui capaç de calcular-ho.

- $printQuads()$: Mostra el codi intermedi en format C3A, respectant les especificacions del document C3A.pdf.

- $mostrarCustomSymtab()$: Mostra el contingut de la taula de símbols personalitzada. Exclou temporals de la taula de símbols per millorar la llegibilitat. Inclou tipus d'informació com INTEGER, FLOAT, i ARRAY.

- $customSymEnter()$: Afegeix una entrada a la taula de símbols personalitzada. Actualitza automàticament l'entrada si ja existeix el nom. Gestiona correctament el tipus de dades (p. ex., ARRAY si el nom té el prefix d'una taula).

- $customSymLookup()$: Busca una entrada a la taula de símbols personalitzada pel seu nom.

- $generateBooleanOperation()$: Genera el codi intermedi per a expressions booleanes.

- $Backpatch()$: Assigna etiquetes a salts pendents.

5 Estat del projecte

Tot i que la majoria de funcionalitats s'han implementat correctament, les següents funcionalitats estan pendents de desenvolupament per manca de temps:

1. Expressions booleanes amb curtcircuit

Les expressions booleanes es poden generar utilitzant operadors relacionals (<, <=, >, >=, =, <>) i lògics (not, and, or).

2. Sentències condicionals amb switch

Les sentències switch haurien d'haver generat salts i etiquetes dinàmiques segons el valor de l'expressió, però aquesta funcionalitat no s'ha inclòs.