Compiladors (CL)
GEI (2025-26)

Pràctica 3: Creació d'un compilador complet

Objectiu

Fer les parts frontal i dorsal d'un compilador per un llenguatge, ampliació del de la pràctica 2.

Aprenentatges

Generació de codi intermedi mitjançant back-patching.

Llenguatge font

Literals i comentaris:

- Els mateixos que a la pràctica 2, afegint els literals booleans **true** i **false**.

Identificadors i expressions:

- Les expressions i operadors aritmètics funcionen igual que a la pràctica 2.
- 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 s'ha de fer en 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 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.
- 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, assignacions, condicionals o iteratives.
- Les expressions aritmètiques i assignacions funcionen igual que a la pràctica 2. (taules unidimensionals i canvis de tipus explícits inclosos)
- Hi ha dos tipus de sentències condicionals: sense i amb alternativa.
- 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

- **S'ha d'implementar les sentències condicionals amb switch**

```
switch ( expressió )
case valor_constant:
          llista_sentencies
 break;
 default : llista_sentencies
 break;

 fswitch
```

- S'ha d'implementar opcions de control de flux de control iteratives o condicionals: sortida incondicional break;

- Hi ha quatre tipus de sentències iteratives: incondicionals, amb condició a l'inici, amb condició al final, i indexades.
- La sintaxi de les sentències iteratives incondicionals és:

  repeat *expressió_aritmètica_entera* **do**
  llista_de_sentències
  done

  El seu significat és executar la llista de sentències tantes vegades com indiqui el valor de l'expressió aritmètica entera. Podeu decidir què fer si aquest valor és negatiu o zero.
- La sintaxi de les sentències iteratives amb condició a l'inici és:

  while *expressió_booleana* **do**
  llista_de_sentències
  done
- La sintaxi de les sentències iteratives amb condició al final és:

  do
  llista_de_sentències
  until *expressió_booleana*
- La sintaxi de les sentències iteratives indexades és:

  for *id* **in** *rang* **do**
  llista_de_sentències
  done

  La variable *id* queda declarada com de tipus enter.
  En la generació del 3AC implementeu l'optimització de codi basat en unrolling de bucles, quan la iteració és inferior a un límit (exemple 5)

- 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.

Programa:

- La sintaxi d'un programa és:

  llista_de_sentències

Opcions:

- Opcions per a expressions aritmètiques i assignacions: tipus i literals caràcter, tipus i literals cadena, altres tipus numèrics (long, double, enters modulars), taules multidimensionals, llistes, registres, apuntadors.
- Opcions per a expressions booleanes, assignacions i flux de control: variables booleanes.
- Opcions per a instruccions de flux de control condicionals: condicionals amb múltiples alternatives (**elsif**)
- Avaluació d'expressions que només involucren literals, sense generar C3A.
- Altres opcions: procediments i/o funcions, àmbits encaixats, etc.
- Altres optimitzacions de codi que vulguis implementar: reutilització de temporals, constant folding, eliminació de codi mort ...

Codi de tres adreces

En el document *C3A.pdf* es descriu el llenguatge intermedi de codi de tres adreces de forma completament general. Per fer aquesta pràctica no es necessita tot, sinó només un subconjunt d'aquest llenguatge.

Sortides

Sortida principal:

- La sortida ha de ser un programa en codi de tres adreces (C3A), respectant la descripció del C3A que es troba en el document *C3A.pdf*.
- Generació del codi de tres adreces, utilitzant back-patching per les expressions booleanes i pel flux de control.
- Les sentències que són només expressions (i.e. sense assignació) indiquen que es vol imprimir el seu valor per pantalla (i.e. utilitzant els procediments C3A predefinits `PUT`).
- Les expressions aritmètiques que només involucren literals es poden avaluar en memòria en temps de compilació, sense generar C3A pel seu càlcul.

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.
- S'ha d'evitar barrejar la sortida C3A 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

Entrada:

```
// nested sentences
i := 1
total := (1.0 + 0 * 1) - i
while total < 1000.0 do
   if (i = 1) then
      total := total * 2
   else
      total := total - 1
   fi
   i := -i
   if total > 666.6 or total = 500.0 then
      i := 111
   fi
done
total
```

Sortida (podria ser lleugerament diferent):

```
 1: i := 1
 2: $t01 := I2F i
 3: $t02 := 1.0 SUBF $t01
 4: total := $t02

 5: IF total LTF 1000.0 GOTO 7
 6: GOTO 22

 7: IF i EQ 1 GOTO 9
 8: GOTO 12
 9: $t03 := total MULF 2.0
10: total := $t03
11: GOTO 14
12: $t04 := total SUBF 1.0
13: total := $t04

14: $t05 := CHSI i
15: i := $t05

16: IF total GTF 666.6 GOTO 20
17: GOTO 18
18: IF total EQ 500.0 GOTO 20
19: GOTO 5
20: i := 111
21: GOTO 5

22: PARAM total
23: CALL PUTF,1
24: HALT
```

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.