Compilador de LANS

30 de setembre de 2024

1. Instruccions i criteris d'avaluació

Les pràctiques representen un 40% de la nota total de l'assignatura, i la nota mínima per aprovar l'assignatura és 4.5/10. El 100% de la nota de pràctiques de l'assignatura correspondrà al resultat d'aquesta pràctica. La pràctica s'ha de desenvolupar en parelles. Hi haurà dues dates d'entrega a través de Moodle:

- 24/11/2024: Entrega de la part lèxica i sintàctica. Si no es fa aquesta entrega o no es compleixen raonablement les especificacions, es descomptarà fins a 1 punt de la nota final de pràctiques. Després de la correcció de l'entrega, s'han de corregir els possibles errors que s'hagin detectat.
- 02/02/2025: Entrega final. Després de l'entrega via Moodle, s'haurà de defensar la pràctica presencialment, dia i hora a determinar per cada grup.

La nota màxima és 10, i es repartirà de la següent manera:

- Anàlisi lèxic: fins a 1 punt. S'ha de fer tot per aprovar.
- Anàlisi sintàctic: fins a 2 punts. S'ha de fer tot per aprovar.
- Anàlisi semàntic: fins a 2 punts part obligatòria, fins a 1.5 punts part opcional.
- Generació de codi: fins a 2 punts part obligatòria, fins a 1.5 punts part opcional.

És obligatori per aprovar completar l'anàlisi semàntic i la generació de codi de:

- Expressions (tots els operadors) i assignació.
- Operacions de lectura i escriptura.
- Condicional.
- Bucles.

Per optar a la màxima puntuació:

- Vectors.
- Tuples.
- Accions i funcions.

Compilador LANS El Llenguatge d'Alt Nivell Senzill (LANS) és un llenguatge de programació inspirat en el pseudocodi imperatiu que s'ha utilitzat durant molts anys en aquesta universitat. L'objectiu d'aquesta pràctica és el desenvolupament d'un compilador que rebrà com a entrada un codi escrit en Llenguatge d'Alt Nivell Senzill (LANS) i ens generarà un fitxer Bytecode (.class), que podrà ser executat amb la Java Virtual Machine (JVM). El propi compilador també es desenvoluparà en Java. La pràctica es desenvoluparà utilitzant ANTLR4 (ANother Tool for Language Recognition), que ens sistematitzarà la construcció dels analitzadors lèxics i sintàctics.

2. Tipus de dades En LANS tenim tipus bàsics de dades i tipus definits.

2.1 Tipus bàsics

Els tipus bàsics de dades que tenim en LANS són els següents:

- enter: Representa un nombre enter.
- real: Representa un nombre real. El separador de decimal és el punt, i podem utilitzar notació científica. Exemples vàlids de nombres reals: 0.1, 2.10, 3.1416, 6.023E23, 12.0E-5.
- booleà: Representa un Booleà. Pot prendre com a valor cert o fals.

Les variables de tipus bàsics no se declararan. Agafant el tipus de la primera assignació. Un cop tenen el tipus sempre mantindran el mateix tipus.

2.2 Tipus definits

Amb LANS podem definir nous tipus de dades, que poden ser tuples o vectors per a tipus bàsics. Els tipus definits es poden utilitzar per definir variables del programa principal així com de qualsevol acció o funció. Els paràmetres de les accions i funcions també poden ser de tipus definits, però el valor de retorn de les funcions no.

3. Estructura general d'un programa LANS

Un programa LANS s'escriu en un sol fitxer, i sempre especifica un nom de programa. El fitxer que generarà el compilador rebrà el mateix nom que el programa compilat. Per exemple, el programa HelloWorld es compilarà per defecte en un fitxer HelloWorld.class.

L'estructura general d'un fitxer LANS conté els següents elements (l'ordre en què apareixen ha de ser aquest):

{Bloc de declaració de tipus} ?

{Bloc d'accions i funcions} ?

programa nom

{Bloc de declaració de variables de tipus definits}?

{sentencia}+

fprograma

3.1 Bloc de declaració de tipus

El bloc de declaració de tipus té la forma següent:

tipus

declaració_de_tipus_nou+

ftipus

Un tipus nou pot ser un vector de tipus bàsic:

nom_tipus_nou : vector tipus_bàsic mida enter{,enter}*;

El valor obligatori mida és un nombre enter que determina la mida del vector. Els vectors comencen per l'índex 0.

Finalment, un nou tipus també pot ser una tupla de camps de tipus bàsic:

mom_tipus_nou : tupla {id : tipus bàsic }+ ftupla;

Fixeu-vos que les tuples han de tenir un camp com a mínim.

3.2 Bloc d'accions i funcions

En el bloc de d'accions i funcions, podem declarar tantes accions i/o funcions com vulguem (o cap). Les declaracions d'una acció i una funció tenen la forma següent:

```
accio nom_accio({parametres formals}?)
        {Bloc de declaració de variables de tipus definits}?
        sentència*
facció
funcio nom_funcio({parametres formals}?)  retorna tipus_bàsic
        {Bloc de declaració de variables de tipus definits}?
        sentència* retorna expr
        tipus bàsic;
ffuncio
```

Els paràmetres formals tenen la forma següent:

{ent|entsor}? id : tipus_bàsic{,{ent|entsor}? id:tipus_bàsic}*

El modificador de paràmetre ens indica si la variable és d'entrada (ent) o si és d'entrada sortida (entsor). Si no s'especifica el modificador, el paràmetre és d'entrada. Les funcions només poden rebre paràmetres d'entrada.

3.3 Bloc de declaració de variables de tipus definit

El bloc de declaració de variables de tipus definit té la forma següent:

```
variables
        {id: tipus;}*
fvariables
```

Totes les variables (normals o definides) només són accessibles des del programa principal o acció o funció on s'hagin declarat.

4. Identificadors

Els identificadors (noms de variables, tipus, accions, funcions i camps de tupla) han de ser una paraula sense espais que només pot contenir caràcters de la a a la z, ja sigui majúscules o minúscules o dígits 0-9 . No poden començar amb un dígit.

5. Comentaris

Els comentaris en LANS són com els de C. Tenim comentaris d'una sola línia que comencen amb //, i s'acaben quan trobem un salt de línia. També podem fer comentaris multi-línia, que comencen amb /* i acaben quan trobem la primera ocurrència de */.

6. Expressions

Expressions Una expressió pot ser:

Un valor constant de tipus bàsic

Una variable

Un accés a tupla

Un accés a vector

Una crida a una funció

Una operació sobre una o vàries expressions

Els operadors que acceptem són:

+ , - : suma, resta. Definits sobre enters i reals.

*, / : multiplicació, divisió. Definits sobre enters i reals.

\, %: divisió entera, resta de divisió entera (mòdul). Definits entre enters, el resultat és un enter.

~: menys unari. Definit sobre enters i reals. Aquest operador canvia de signe l'expressió de la seva dreta.

== , !=: igualtat, desigualtat. Definit sobre tots els tipus bàsics. El resultat és Booleà.

<, <=, >, >=: més petit, més petit igual, més gran, més gran igual. Definits sobre tots els tipus bàsics. El resultat és Booleà.

no , & , |: negació lògica, and lògica, or lògica. Definits sobre Booleà i el resultat és Booleà.

Les promocions d'enter a real són admeses i automàtiques quan sigui convenient. Per exemple, són expressions correctes "3.5 + 1", "2==4.7", "a ? 3 : 7.8", i en aquest exemples promocionem 1, 2 i 3 d'enter a real. No són admeses altres conversions entre tipus bàsics.

Els vectors s'indexen amb []: nom_variable[expr entera] [expr_entera]

Els camps de les tuples s'accedeixen amb un punt: id variable.nom camp Les funcions s'accedeixen passant els paràmetres reals (expressions) entre parèntesis i separats per coma:

nom_funcio({expr,{expr}*}?)

La prioritat dels operadors és, de més a menys prioritari:

no ~

* / \ %

+ -

!= < <= > >= ==

& | no

En cas d'operadors d'igual prioritat, tindrem associativitat per l'esquerra, exemple: 4 − a + b − 3 = ((4 − a) + b) − 3. Podem modificar prioritats i associativitat posant expressions entre parèntesis.

7. Sentències

Una sentència pot ser una assignació, un condicional, un bucle de tipus per, un bucle de tipus mentre, una crida a una acció, o una instrucció de lectura/escriptura.

7.1 Assignació

Una assignació té la forma següent:

nom_variable := expr;

Podem assignar posicions d'un vector i camps d'una tupla, però no assignarem tot un vector ni tota una tupla. Podem assignar un enter a un real, però no al revés.

7.2 Condicional

Un condicional té la forma següent:

si expr_booleana llavors sentencia*

{altrasi expr_booleana llavors sentencia*}*

{altrament sentencia*}?

fsi

7.3 Per

El bucle per itera una variable entera en un rang inclusiu pels dos extrems:

```
per id en rang(num{,num}?) fer
        sentencia*
fper
```

El rang va de 0 fins a num-1 si nomes tenim un numero o de num1 fins num2-2 en el cas de dos numeros

7.4 Mentre

El bucle mentre repeteix les instruccions mentre la condició sigui certa:

```
mentre expressio_booleana fer
sentenciai+
fmentre
```

7.4 Crida a acció

Una acció es crida de la mateixa manera que una funció:

```
nom_accio({expri{,expr}*}?)
```

7.4 Lectura/escriptura

Tenim la instrucció de lectura llegir i les instruccions d'escriptura escriure i escriureln com a accions integrades al llenguatge. La instrucció llegir rep un sol paràmetre de sortida que és una variable de tipus bàsic on guardarem el valor llegit per teclat:

```
llegir(id{:tipus_bàsic}?);
```

Si la variable ja té tipus no cal donar el tipus bàsic de la lectura

La instrucció escriure rep un nombre variable de paràmetres (almenys n'ha de rebre un), i escriu per pantalla la concatenació dels diferents valors rebuts:

```
escriure(expr,{,expr}*);
```

La instrucció escriureln és semblant a la instrucció escriure, però al final de tot escriu un salt de línia. A més, la instrucció escriureln pot no rebre cap paràmetre.

Tan escriure com escriureln poden rebre expressions de tipus string. Un string serà un text que va entre cometes dobles.