Interpretor

Tema 2

responsabil: Caramizaru Horea Alexandru

1 Introducere

Tema va avea ca obiectiv construirea unui interpretor pentru un limbaj simplu.

Exemplu de cod :

```
var = 10
nr = 5
sum = var - nr * 4
max = (sum>var)?nr:var
fin = 2 * ( sum - max ) - nr
```

Tipurile de date existente vor fi numere intregi cu semn.

Operatorii pentru tipul int:

+ (adunare - operator unar sau binar)

- (scadere - operator unar sau binar)

* (inmultire - operator binar)

? (operator ternar) (a>b)?a:b

Precedenta operatiilor este:

(1) *

(2) + -

(3) ?

(4) =

Pentru acelas nivel al precedentei nu conteaza ordinea operatiilor.

Operanzi:

variabile : "var" , "n"

constante : 22 , -24

Pentru constante care nu contin semn se considera automat ca sunt pozitive.

Precedenta operatiilor poate fi schimbata cu ajutorul parantezelor rotunde.

Pentru realizarea acestui interpretor va trebui sa construiti :

-arbore de parsare

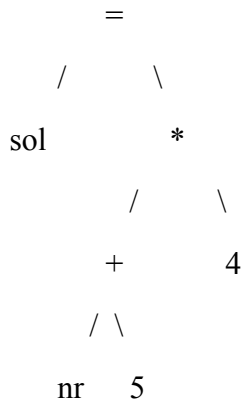
-analiza semantica

-evaluare expresie

Arbore de parsare

Un exemplu de arbore de parsare pentru expresia:

sol = (nr + 5) * 4



Pentru a construi arborele de parsare va trebui definita o ierarhie de clase. Clasa de baza se va numi Nod iar din acesta se vor extinde clase pentru Expresii, Termeni, Factori, Operatori. Termenii pot fi constante sau variabile.

Pentru: sol = (nr + 5) * 4

= + * sunt operatori

nr + 4 este expresie

(..) si 4 sunt factori

Pentru realizarea structurii arborescente se vor utilza colectii corespunzatoare din Java. Pentru fiecare arbore construit se va afisa incepand de la nivelul 0 al arborelui in felul urmator.

Pentru exemplul de mai sus:

E

T = E

T = F * T

T = (T + T) * F

Unde:

"E" reprezinta expresie.

"T" reprezinta termen (constanta sau variabila)

"F" reprezinta factor (constanta sau variabila)

Mai multe detalii puteti sa vedeti aici:

http://www.csse.monash.edu.au/~lloyd/tildeProgLang/Grammar/Arith-Exp/

http://www.infoarena.ro/problema/evaluare

Pentru situatiile in care pot exista mai multi arbori posibili orice solutie este admisa.

Exemplu:

a = 3

b = a + 3 + a

T = T + E sau T = E + T sunt considerate corecte in procesul de construire al arborelui pentru linia 2.

Analiza semantica

Analiza semantica are ca scop determinarea corectitudinii intregii expresii.

Erorile care pot aparea:

var = 9

1 = var (membrul stang nu este o variabila la linia 2 coloana 1)

total = var + b (b nedeclarata la linia 3 coloana 9)

Pentru fiecare linie se va afisa in fisierul de iesire eroarea corespunzatoare sau "Ok!"

Exemplu:

b nedeclarata la linia 2 coloana 1

Ok!

membrul stang nu este o variabila la linia 3 coloana 9

In cazul in care intr-o expresie apar mai multe erori se va afisa decat prima de la stanga spre dreapta.

Valorile continute de variabile pot fi suprascrise:

a = 4

b = 9

a = a + b - 2

Evaluarea expresiei:

Rezultatul evaluarii expresiei va fi afisat pentru fiecare linie.

In cazul in care o linie de instructiuni contine erori se va afisa "error".

Exemplu:

input:

a = 4

b = 9

d = val + a

a = a + b - 2

rezultat:

a = 4

b = 9

error

a = 11

Precizari si clarificari:

1. numele fisierului de intrare se va da ca argument in linia de comanda (primul parametru)
2. input-ul nu va contine erori diferite de cele de la nivelul de analiza semantica
3. fiecare instructiune ocupa exact o linie
4. Pentru realizarea temei trebuie folosita o versiune de Java nu mai recenta de Java 7, aceasta fiind versiunea care ruleaza pe VMChecker.

5. output-ul va consta in 3 fisiere:
   - <fisier_intrare>_pt (arbore parsare)
   - <fisier_intrare>_sa (analiza semantica)
   - <fisier_intrare>_ee (evaluare expresie)

RESPECTAREA REGULILOR DE IMPLEMENTARE ESTE OBLIGATORIE.

Pe langa implementarea efectiva, va trebui sa generati si un javadoc pentru clasele create, folosind comentarii în cod si generarea automata oferita de Netbeans / Eclipse, precum și un readme în care să explicați deciziile luate în implementarea temei și problemele întampinate.

Testarea va fi automata, astfel incat va trebui să aveti si un makefile cu o regula run care sa ruleze clasa si care contine metoda ”main”.

Punctajul pe tema va consta din:

- 70% teste
- 10% coding style
- 20% readme + JavaDoc

Pentru intrebari, clarificari, etc. voi fi prezent la curs pe 25 noiembrie. (dupa prima saptamana)