Paradigme de programare
Tema 3 – Prolog

Termen de predare: **25.05.2017** (soft + hard) - **11:55 AM**

Ultima actualizare: 22.05.2017 (v.3)
v.1 : versiunea initiala (enunt + schelet de cod + teste publice)
v2: eliminare fraza "errors.txt"
v.3: prelungire deadline 12 ore pana pe 25.05.2017, 11:55 AM

1. Introducere

Se da limbajul IMP++ definit de urmatoarea gramatica:

<symbol> :: [a-zA-Z]+
*<value> :: 0 | [1-9][0-9]**
*<op> :: + | - | * | == | <*
<expr> :: <expr> <op> <expr> | <symbol> | <value>
<assign> :: <symbol> = <expr>
<prog> :: <assign>;
| <prog> <prog>
| if (<expr>) then {<prog>} else {<prog>}
| for (<assign>; <expr>; <assign>) {<prog>}
| assert <expr>;
| return <expr>;
| ;

Nu este necesar ca ultima instructiune din codul sursa sa fie de tip return, atat timp cat ultima instructiune executata este de acest tip. De exemplu, urmatorul program este valid si are ca rezultat valoarea 5.

a = 10;
if (a < 15) then { return 5;}
else {;}
a = 12;

Programul este invalid doar atunci cand se ajunge la sfarsitul programului fara sa se fi executat vreo instructiune de tip ***return***.
Consideram toate variabilele ca fiind implicit globale, iar orice modificare adusa unei variabile in orice bloc al programului este vizibila oriunde in program si nu se limiteaza la regiunea respectiva(de exemplu in interiorul unui bloc ***then***). Prin urmare, programul urmator intoarce rezultatul ***4***.

if (1) then { x = 5;}
else {;}
y = x – 1;
return y;

Daca instructiunea ***if*** de mai sus primeste o conditie ce ***nu*** este adevarata, programul devine ***invalid*** deoarece variabile ***x*** este folosita fara a fi initializata anterior.

De asemenea, pentru ca rezultatul unui program sa fie valid, trebuie ca intregul program sa fie corect din punct de vedere ***sintactic***. De exemplu, programul urmator va produce o eroare, chiar daca instructiunea ***return*** apare inainte de linia care genereaza eroarea (*vezi* ***gramatica***).

return 0;
a = c ** 2;

Erorile semantice nu afecteaza validitatea programului pana la evaluarea efectiva a instructiunii in care apar. Singura eroare semantica posibila este folosirea unei variabile neinitializate. Programul urmator este valid si are ca rezultat valoarea ***0***;

return 0;
a = c + 2;

Se considera ca rezultatul celor doua operatii logice (***'<'*** si ***'=='***) este numeric si poate avea doua valori posibile ***0***(*fals*) si ***1***(*adevarat*).

IMPORTANT! In ordine descrescatoare a prioritatii operatiilor avem ****, + si -, <, ==.***
In aceste conditii, rezultatul programului urmator este ***0***:

return 1 + 2 * 3 < 5 == 1;

Conditia asociata instructiunilor ***if, for si assert*** este considerata ***falsa*** atunci cand expresia asociata se evalueaza la ***0***, si ***adevarata*** cand [expresia] se evalueaza la ***1***. Cuvintele cheie (***if, then, else, for, assert si return***) nu trebuie sa apara pe post de ***<symbol>***. In caz contrar, trebuie semnalata o eroare.

2. Schelet de cod

In cadrul fisierului ***"tema3.pl"*** veti gasi un parser ajutator si definitia predicatului ***parseInput***, care este folosit pentru a evalua rezultatul prelucrarii unui fisier sursa. Parserul ajutator ***lparse*** primeste un string pe care il transforma intr-o lista de ***tokens***. Acestea pot fi de forma ***[a-zA-Z0-9]+, ';', '<', '+', '-', '*', '(', ')', '{', '}', '='*** sau ***'=='.*** Orice altceva(inclusiv spatii libere) este ignorat.

Practic, acest parser transforma string-ul respectiv intr-o lista de elemente constituente ale limbajului *IMP++*, cu *mentiunea importanta* ca ***nu*** disting intre valori si simboluri; ambele pot fi cuprinse de un *token* de tip ***[a-zA-Z0-9]+***. Ramane in sarcina voastra sa va asigurati ca simbolurile si valorile sunt conforme cu definitia limbajului.

Pentru a vedea mai bine cum arata rezultatul parserului ajutator, puteti evalua expresia:

read_file_to_string*('C:\\tema3pp\\tests\\public\\t001.in', S, []), lparse(S,L), write(L), fail.*

Calea de mai sus depinde, bineinteles, de locul unde se afla directorul ce contine testele publice (sau orice alte fisiere pe care doriti sa le parsati), si numele fisierului. Dupa cum se poate observa din scheletul de cod, predicatul ***parseInput(F,R)*** incepe prin a citi intr-un string continutul fisierului ***F***, pe care il transforma intr-o lista de tokens cu ajutorul lui ***lparse***, si in cele din urma face apel la ***parseInputAux*** pentru a obtine rezultatul.

3. Cerinte

Se cere implementarea predicatului ***parseInputAux***, astfel incat ***parseInput*** sa obtine valoarea corecta asociata evaluarii fisierului sursa primit ca parametru.

Un program *IMP++* poate duce la urmatoarele valori:

- '***a***', daca cel putin o instructiune *assert* esueaza;
- '***e***', daca s-a intalnit o *eroare sintactica* sau *semantica*; nu trebuie aduse precizari suplimentare cu privire la natura erorii;
- '***x***', daca programul a fost executat fara erori, insa am ajuns la sfarsitul lui fara a intalni vreo instructiune *return*.
- O valoare numerica, obtinuta in urma executiei unei instructiuni *return*.

Pentru cazul in care programul nu respecta mai mult de o singura regula, ordinea de tratare a erorilor va fi: ***'e' > 'a' > 'x'***. Acesta inseamna ca daca intr-un program esueaza cel putin o instructiune de *assert* si in acelasi timp programul are si o *eroare semantica/sintactica*, atunci in urma rularii programului, se va afisa ***'e',*** intrucat eroarea semantica/sintactica are o prioritate de tratare mai mare.

IMPORTANT! Nu modificati definitia lui ***parseInput***.

4. Testare

Puteti gasi testele publice in directorul ***"tests\public\"***. Fisierele ".out" contin rezultatul asteptat de la fiecare test. Testarea se va face automat. Pe langa testele incluse in arhiva va exista si un set de teste private. Fiecare set de teste valoreaza ***50%*** din punctajul temei.

5. Trimitere

Tema trebuie trimisa intr-o arhiza ".zip" care sa contina doar doua fisiera (fara vreun director intermediar): scheletul de cod completat(fisierul ***"tema3.pl"***) si un fisier ***README*** in care sa explicati cum ati rezolvat tema. Numele arhivei trebuie sa fie de forma ***"grupa_nume_prenume.zip".***

6. Punctaj

Tema valoreaza ***1.5 puncte*** din nota finala. Punctele sunt impartite astfel:

- ***0.75p*** pentru testele publice
- ***0.75p*** pentru testele private