

Tema 2

Dezvoltarea unui compilator în ANTLR pentru un mini-limbaj de programare

Enunț

Să se creeze un compilator în ANTLR pentru un mini-limbaj de programare, care să analizeze și să valideze un program scris în mini-limbajul respectiv. Programul trebuie să includă declarații de funcții și variabile, comentarii, structuri decizionale, structuri repetitive și expresii (aritmetice, relaționale, logice etc).

Cerințe

- I. Programul sursă se va citi obligatoriu dintr-un fișier text.
- II. Mini-limbajul va include următoarele categorii de unități lexicale:
 - identificatori: denumiri pentru variabile și funcții;
 - constante numerice și literali (șiruri de caractere cuprinse între ghilimele);
 - cuvinte cheie: int, float, double, string, const, void, if, else, for, while, return;
 - operatori aritmetici: +, -, *, /, %;
 - operatori relaționali: <, >, <=, >=, ==, !=;
 - operatori logici: &&, ||, !;
 - operatori de atribuire: =, +=, -=, *=, /=, %=;
 - operatori de incrementare ++ și decrementare --;
 - delimitatori: paranteze ((,), {, }, virgulă (,), punct și virgulă (;).
- III. Compilatorul va procesa programul sursă pentru a realiza următoarele operațiuni:
 1. Neglijarea spațiilor albe și a comentariilor de tip linie (//) și de tip bloc (/* */).
 2. Obținerea și salvarea într-un fișier text a listei de unități lexicale. Fiecare element al listei este de forma <token, lexemă, index linie>.

3. Colectarea și salvarea în fișiere text distincte a elementelor de sintaxă prezente în program:
- (a) O listă de variabile globale (împreună cu tipul acestora și valorile cu care au fost inițializate la momentul declarării);
 - (b) O listă de funcții. Fiecare element al listei va conține:
 - tipul funcției (iterativă sau recursivă, main sau non-main), tipul returnat, numele și lista parametrilor;
 - lista variabilelor locale (împreună cu tipul acestora și valorile cu care au fost inițializate la momentul declarării);
 - lista structurilor de control (if...else, for, while). Fiecare element al listei este de forma <structură, index linie>.
4. Semnalarea eventualelor erori lexicale, sintactice și semantice, conform detaliilor prezentate pe următoarele pagini ale acestui document. Afișarea mesajelor de eroare se va face obligatoriu într-un fișier text.

Barem

1. Definirea unităților lexicale ale mini-limbajului - 1.5p
2. Neglijarea spațiilor albe și a comentariilor - 0.5p
3. Obținerea listei de unități lexicale - 0.5p
4. Definirea regulilor sintactice ale mini-limbajului - 2.5p
5. Generarea listei de variabile globale - 0.5p
6. Generarea listei de funcții și a detaliilor asociate - 1.5p
7. Semnalarea erorilor - 2p

Oficiu: 1p

Informații privind semnalarea erorilor

- **erori lexicale:** caractere care nu fac parte din vocabularul permis, comentarii și șiruri de caractere care nu au fost închise, șiruri de caractere care se întind pe mai multe rânduri.
- **erori sintactice:** reguli de sintaxă care nu sunt respectate.
- **erori semantice.**

Pentru a asigura corectitudinea semantică a programului sursă, se vor respecta următoarele reguli:

- *unicitatea variabilelor globale:*
Nu este permis ca două variabile globale să aibă același nume.
- *unicitatea funcțiilor:*
Nu este permis să existe două funcții cu același nume și aceeași listă de parametri.
- *validitatea apelurilor unei funcții:*
Nu este permisă apelarea unei funcții care nu a fost definită în program.
- *unicitatea funcției main:*
Programul trebuie să conțină exact o singură funcție **main**, care reprezintă punctul de start al execuției.
- *restricționarea recursivității pentru funcția main:*
Funcția **main** nu poate fi apelată nici de alte funcții, nici de ea însăși.
- *utilizarea variabilelor declarate:*
O variabilă nu poate fi utilizată înainte de a fi declarată.
- *unicitatea variabilelor locale într-o funcție:*
În cadrul unei funcții, nu pot exista două variabile locale cu același nume.
- *evitarea conflictelor între variabilele locale și parametrii unei funcții:*
Numele unei variabile locale nu trebuie să coincidă cu numele unei variabile transmise ca parametru funcției respective.
- *compatibilitatea tipurilor la inițializare:*
Tipul de date al unei variabile trebuie să coincidă cu tipul valorii cu care este inițializată.

Exemple:

`int number = "some text";` – NU este permis.

`double PI = 3.14;` – Este permis.

- *număr corect de argumente într-un apel de funcție:*
Numărul de argumente transmise la apelarea unei funcții trebuie să corespundă numărului de parametri definiți.
- *compatibilitatea tipurilor argumentelor la apelul unei funcții:*
Tipurile argumentelor transmise trebuie să fie compatibile cu tipurile parametrilor funcției.
- *compatibilitatea tipului valorii returnate:*
Tipul valorii întoarse de o funcție trebuie să corespundă tipului declarat al funcției.
- *instrucțiune **return** obligatorie (dacă este cazul):*
Funcțiile care au un tip de retur non-void trebuie să se încheie cu o instrucțiune **return**.
- *limitări asupra variabilelor constante:*
Nu este permisă atribuirea unei noi valori unei variabile declarate ca fiind constantă.