Laboratorul 4.
Analiză Sintactică II.
Construcția AST.

Abstract Syntax Tree (AST)

AST-ul este o structură de date construită în memorie, de către compilator, ca o rafinare a arborelui de derivare, produs direct de analizorul sintactic.

Această structură nu are o formă standardizată (depinde de implementarea și cerințele fiecărui compilator și limbaj de programare), dar în general are următoarele caracteristici:

- Toate tipurile de nod, care corespund unor tipuri diferite de instrucțiuni, sunt reprezentate prin clase ce extind aceeași clasă de bază.
- Nu conține noduri „intermediare", fiind independent de gramatica particulară utilizată (ex: pentru a recunoaște o constantă numerică, inițial se generează în arborele de derivare o cale de forma `expr - term - integer`). Astfel, etapele ulterioare, precum analiza semantică, nu suferă modificări datorate eventualelor actualizări ale gramaticii.
- Dacă unele structuri de sintaxă există doar ca indicații pentru analiză (ex: parantezele din expresiile matematice, pentru stabilirea ordinii corecte a operațiilor), nodurile lor pot fi eliminate din AST, atâta timp cât ordinea corectă a operațiilor apare corect în arbore.
- Dacă unele structuri de sintaxă sunt echivalente (ex: reprezentări diferite ale constantelor numerice, `if...then...endif` vs. `if...then...else NOP endif`), ele pot fi reprezentate în AST sub o formă unificată.

Exerciții:

Ne propunem ca în laboratorul de astăzi să construim și să afișăm AST-ul unui program CPLang. **Salvați fișierele laboratorului anterior sau copiațiile într-un proiect nou pentru a nu pierde rezolvarea!** Copiați fișierele din arhivă în proiectul laboratorului anterior și urmăriți `TODO`-urile din fișierele `ASTNode.java`, `ASTVisitor.java` și `Test.java`.

1. Definirea ierarhiei de clase pentru AST

Pornind de la clasa-părinte (`ASTNode`), extindeți cu clase specializate pentru fiecare tip de nod ce poate apărea în AST (expresii `if`, operații aritmetice etc.).

- Asigurați-vă că salvați corespunzător copiii unui nod, luând în considerare numărul acestora
- Pentru definiții și apeluri de funcții puteți avea câmpuri de forma `LinkedList<TipNod>` în care să salvați argumentele apărute între parantezele rotunde.
- Utilizați manualul CPLang pentru inventarierea tuturor construcțiilor sintactice ale limbajului.

2. Construcția AST-ului și afișarea sa

Completați în fișierul `Test.java` cei doi visitori:

- `astConstructionVisitor`, instanță de `CPLangParserBaseVisitor`, va parcurge arborele de derivare produs de analiza sintactică (vezi laboratorul anterior) în vederea construirii AST-ului
- `printVisitor`, instanță de `ASTVisitor`, va parcurge AST-ul pentru a afișa conținutul fiecărui nod, utilizând indentare cu tab-uri în funcție de nivelul nodului în arbore.

Afișând nodul părinte al AST-ului, recursiv se va declanșa afișarea întregului arbore.