Laboratorul 6.
Analiză semantică II. Verificarea tipurilor

1 Verificarea tipurilor

În acest laborator, vom extinde funcționalitatea implementată în laboratorul anterior, de rezolvare a simbolurilor, cu posibilitatea luării în calcul a informației de tip, realizării de verificări de tip și afișării unor mesaje de eroare corespunzătoare.

Elementele de noutate sunt următoarele:

- A fost definită **clasa** `TypeSymbol`, ale cărei instanțe sunt simboluri aferente tipurilor.
- Clasa `IdSymbol` conține acum **câmpul** `type`, având tipul `TypeSymbol`.
- Abordarea în două treceri peste AST se menține. În plus, în cea de-a **doua** trecere, în care se rezolvă simbolurile pentru funcții, vom realiza și verificarea de tip. Astfel, metodele vizitatorului aferent celei de-a doua treceri întorc acum `TypeSymbol` în loc de `Void`.

2 Cerințe

1. În prima trecere, în cazul **definițiilor** de variabile globale, funcții și parametri formali, rețineți informația de tip în cadrul simbolul nou creat. Pentru funcții, tipul se referă la cel de retur. Urmăriți comentariile `TODO 1` din clasa `DefinitionPassVisitor`.
2. În a doua trecere, la întâlnirea unei referiri la o **variabilă**, rezolvată deja în prima trecere, precizați tipul acesteia prin valoarea de retur a metodei `visit` corespunzătoare. Similar, pentru **literalii** booleeni, întregi și reali, puteți întoarce direct tipul aferent. Urmăriți comentariile `TODO 2` din clasa `ResolutionPassVisitor`.
3. Pentru expresiile **aritmetice și relaționale**, verificați tipurile operanzilor, și precizați tipul expresiei. Afișați eroare dacă tipurile operanzilor nu corespund (de exemplu `Bool + Int`). Puteți considera fie că tipurile operanzilor trebuie să coincidă, fie că este permisă conversia implicită a unui `Int` la `Float`. Având în vedere uniformitatea verificărilor privitoare la operațiile aritmetice, puteți lua în calcul definirea unei funcții **separate** de verificare, pe care să o apelați în toate metodele `visit` necesare. Urmăriți comentariile `TODO 3` din clasa `ResolutionPassVisitor`.

4. Pentru expresiile `if`, realizați verificarea tip pentru cele trei componente (condiția, ramura `then`, ramura `else`) și întoarceți tipul expresiei. Afișați eroare dacă condiția nu are tipul `Bool`, sau dacă ramurile au tipuri incompatibile. Urmăriți comentariile `TODO 4` din clasa `ResolutionPassVisitor`.

5. Pentru definițiile de variabile cu inițializare, atribuiri, și definițiile de funcții, asigurați-vă de compatibilitatea tipurilor. Pentru **variabile**, tipul expresiei de inițializare sau cu care se realizează o atribuire trebuie să fie compatibil cu tipul variabilei, iar în cazul **funcțiilor**, tipul corpului trebuie să fie compatibil cu tipul de retur declarat. Urmăriți comentariile `TODO 5` din clasa `ResolutionPassVisitor`.

6. Pentru **apelurile** de funcție, verificați dacă numărul parametrilor actuali coincide cu cel al parametrilor formali, și că tipurile sunt compatibile. Urmăriți comentariile `TODO 6` din clasa `ResolutionPassVisitor`.

Atenție! Încercați să nu propagați erorile întâlnite la nivelurile mai joase din AST către nivelurile mai înalte!