Limbaje formale și tehnici de compilare

analiza de domeniu

Analiza de domeniu analizează declarațiile de simboluri (variabile, funcții, argumente, tipuri de date etc.) și introduce simbolurile declarate într-o structură de date specifică, denumită **Tabela de simboluri** (**TS**).
TS este implementată ca o stivă de **domenii**, un domeniu fiind o secțiune de program care grupează simboluri cu aceeași vizibilitate. De exemplu, în limbajul Quick avem **domeniul global**, care grupează variabilele globale și funcțiile și **domeniile locale**, specifice fiecărei funcții, care grupează argumentele și variabilele locale ale acelei funcții.
La începutul programului, în TS se va crea domeniul local, iar apoi, când se parsează o funcție, se va crea domeniul acelei funcții. Când se termină de parsat funcția, se șterge domeniul ei din TS, împreună cu toate simbolurile din el și se revine în domeniul global.

Implementarea analizei de domeniu pentru limbajul Quick

În limbajul Quick analiza de domeniu are loc simultan cu analiza sintactică. Pentru aceasta, în unele reguli sintactice se va insera cod care implementează acțiunile necesare analizei de domeniu în acel punct al analizorului sintactic. Toate aceste fragmente de cod sunt date în fișierul „**Quick - analiza de domeniu.pdf**”, între acolade și folosind culoarea albastru.
Aceste fragmente de cod se vor implementa în analizorul sintactic propriu, iar dacă implementarea este corectă, la execuția compilatorului, folosind ca intrare exemplul din laborator, mesajele care vor apărea pe ecran vor fi de forma celor listate în fișierul „**afișare_ad.txt**”.
Pentru ca unele secvențe de cod să funcționeze, ele au nevoie să știe ce atom a fost consumat. Pentru aceasta se va folosi o variabilă globală „**Atom *consumed;**”, iar în funcția **consume**, dacă se consumă atomul curent, se va adăuga o linie de forma „**consumed=&atomi[idxCrtAtom];**”, înainte de a se trece la următorul atom.

Observație: funcția **tkerr** folosită în secvențele de cod este ușor diferită față de cea prezentată în laboratorul de la analizorul lexical, în sensul că nu mai are nevoie să i se dea ca argument atomul curent de la analiza sintactică, ci ea folosește automat acest atom. Tot restul funcționalității este identic.
Se poate folosi și varianta de **tkerr** de la laboratorul cu analizorul lexical, dar atunci trebuie modificate corespunzător apelurile ei din secvențele de cod.

Secvențele de cod pentru analiza de domeniu folosesc anumite funcții, tipuri de date și variabile globale, care sunt declarate și implementate în fișierele „**ad.h**” și „**ad.c**”. Aceste fișiere vor trebui incluse în compilatorul propriu.
Fișierul „**ad.h**” se poate include ca orice fișier antet local, folosind **#include "ad.h"**. Fișierul „**ad.c**” se poate adăuga proiectului sau copia conținutul acestuia în fișierele proprii.
De exemplu, dacă se folosește **gcc** în linia de comandă, compilarea unui proiect format din mai multe fișiere se face folosind o comandă de forma:

```
gcc -Wall -o compilator main.c alex.c asin.c ad.c
```

Dacă se folosesc medii de dezvoltare gen **Visual Studio** sau **Code::Blocks**, se poate adăuga fișierul „**ad.c**” în lista de fișiere sursă a proiectului.

Evaluarea activității

Evaluarea acestei activități se va face conform următoarei proceduri:

1. Se va folosi ca intrare în compilator fișierul exemplu din laborator, iar mesajele afișate vor trebui să fie de forma celor listate în fișierul „**afișare_ad.txt**”

2. Se vor introduce în exemplu unele erori, de exemplu două variabile globale cu același nume sau două argumente cu același nume, iar compilatorul va trebui să genereze mesajele de eroare specifice
3. Studentul va fi necesar să știe să explice cum funcționează codul din fișierul „**ad.c**” (de exemplu cum s-a implementat adăugarea la o listă simplu înlănțuită)