Proiect

Limbaje formale si compilatoare

Crearea unui compilator folosind FLEX si BISON

Brânzea Ana-Maria,

Grupa 40317A

Cuprins

1. Programe folosite
2. Funcționalitate și mod de utilizare
3. Fișierul .l
4. Fișierul .y
5. Bibliografie
6. Programe folosite

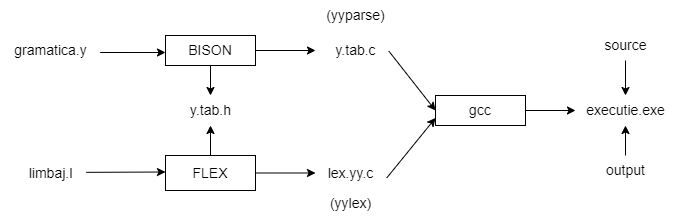
Flex și Bison sunt două instrumente de analiză lexicală și sintactică utilizate împreună pentru a genera analizatori sintactici.

**Flex** este un generator de analizatori lexicali. El funcționează prin definirea unui set de reguli pentru identificarea și analizarea componentelor lexicale ale unui limbaj, cum ar fi cuvinte-cheie, operatori și literali. Acestea sunt definite într-un fișier de intrare numit "fișier flex" și sunt utilizate pentru a genera codul C sau C++ care poate fi utilizat pentru a analiza textul în limbajul specificat.

**Bison** este un generator de analizatori sintactici. El funcționează prin definirea unui set de reguli gramaticale pentru o limbă specificată, care sunt definite într-un fișier de intrare numit "fișier bison". Acestea sunt utilizate pentru a genera codul C sau C++ care poate fi utilizat pentru a analiza gramatica limbajului specificat.

Arborele sintactic va fi traversat depth first, în acest timp generându-se codul. Unele compilatoare pot produce direct cod executabil, în vreme ce altele au drept rezultat programul translatat în limbaj de asamblare.

**Utilizarea Flex și Bison**

Fig1: succesiunea de operații implicate in realizarea unui compilator folosind FLEX si BISON ****

BISON citește descrierea gramaticii din fișierul gramatica.y si generează parserul (si anume funcția yyparse()) in fișierul y.tab.c si definițiile tokenilor in y.tab.h.

FLEX citește descrierile pattern-urilor din limbaj.l si creează pe baza regulilor descrise scanner-ul, descris de funcția yylex, conținută in fișierul lex.yy.c.

In final, parser-ul si scanner-ul sunt compilate împreună pentru a forma executabilul executie.exe. Din funcția main se cheamă yyparse pentru a rula compilatorul. Acesta cheamă yylex pentru a obține următorul token din fișierul de intrare.

Translatorul se creează prin următoarele comenzi folosind PowerShell window:

> bison -y -d gramatica.y // creează y.tab.h si y.tab.c

> flex limbaj.l // creează lex.yy.c

> gcc -o executie.exe y.tab.c lex.yy.c // compilare; se creează executie.exe

1. Funcționalitate și mod de utilizare

Executie.exe va rula in terminal. Acesta permite rezolvarea unor calcule matematice folosind cuvinte cheie.

**Operații permise:**

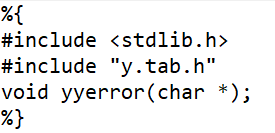
**Operații aritmetice de baza:** adunare, scădere, înmulțire, împărțire

**Modulo:** a%b (restul împărțirii lui a la b, rezultatele vor fi valide doar pentru intregi, altfel se va afișa 0) **Modul:**  |-a|= a sau |a|=a

Instrucțiunile trebuie separate de ’;’.

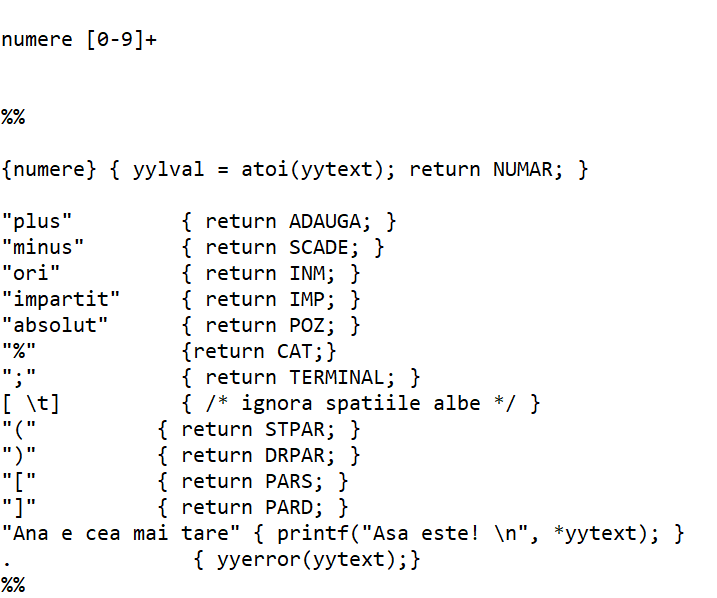
1. Fișierul .l

Fișierul limbaj.l conține trei secțiuni delimitate prin %%: definiții, reguli si subrutine.

**Secțiunea de definiții:**

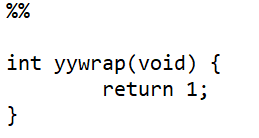
Se include fișierul y.tab.h si biblioteca C <stdlib.h> (pentru a putea folosi functiile atoi().

**Secțiunea de reguli:**

****

Orice numar va fi recunoscut ca token-ul NUMAR, valoarea numerica fiind însuși numărul.(funcția atoi() va converti string-ul de input într-un int)

**Secțiunea de subrutine:**

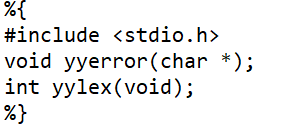


Funcția yywrap() verifica daca fișierul de input a ajuns la final.

1. Fisierul .y

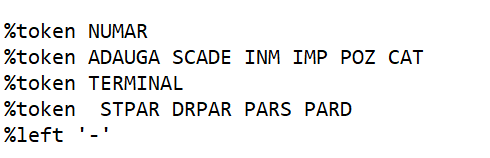
Fișierul gramatica.y va conține secțiunile definiții, reguli si subrutine.

**Secțiunea definiții:**

****

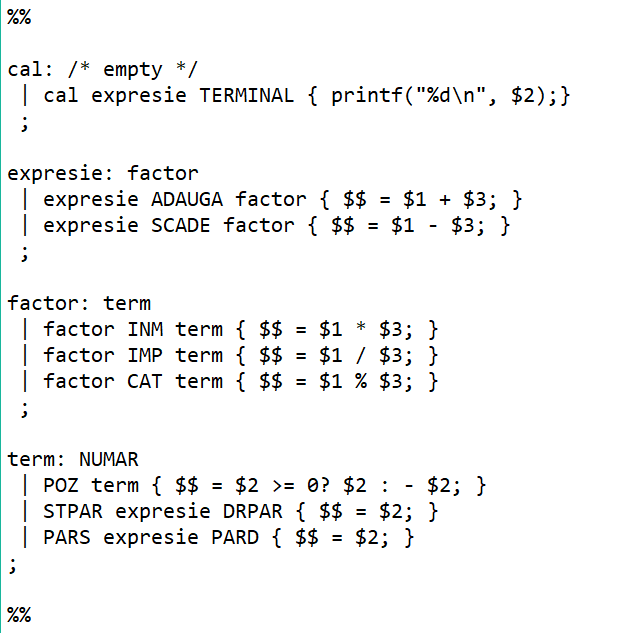
- Se include biblioteca necesara

- Se defineste functia yylex

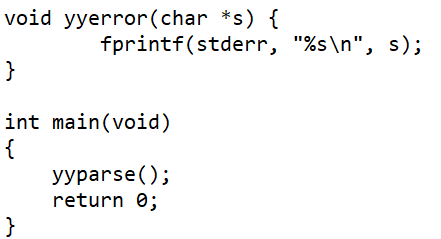


Se specifica *tokenii* si neterminali.

**Secțiunea de reguli:**

****

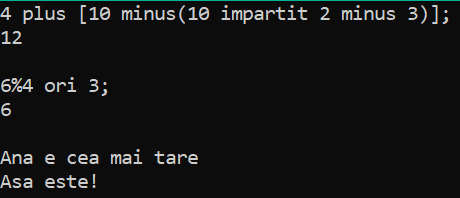
**Secțiunea de subrutine:**

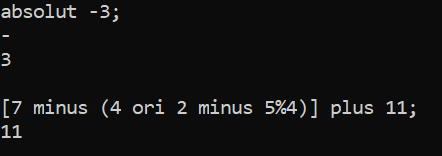
****

- Se defineste functia yyerror()

- In main se va apela yyparse();

Execuție





1. Bibliografie

* Manual\_Bison\_2021.pdf
* 3\_Generatorul de analizoare lexicale BISON.pdf
* 2\_Generatorul de analizoare lexicale FLEX.pdf
* 1\_Introducere in FLEX si BISON.pdf
* <https://www.oreilly.com/library/view/flex-bison/9780596805418/ch01.html>
* https://begriffs.com/posts/2021-11-28-practical-parsing.html