Iată o sinteză clară și concisă cu lucrurile importante de reținut din cele trei documente despre **LEX** și analiza lexicală:

**🔍 Scopul Analizei Lexicale și a LEX-ului**

* Analiza lexicală transformă codul sursă (un șir de caractere) într-o secvență de **simboluri/tokeni** semnificativi.
* LEX este un **generator de analizatoare lexicale** care creează funcția yylex() ce citește fluxul de intrare și îl împarte conform **expresiilor regulate** specificate.
* Este folosit singur sau împreună cu YACC pentru analiză sintactică (yyparse()).

**📄 Structura fișierului LEX**

Format general:

{definiții}

%%

{reguli}

%%

{subrutine utilizator}

* **Definiții**: macrodefiniri, includeri, condiții de start.
* **Reguli**: expresii regulate (stânga) + acțiuni C (dreapta).
* **Subrutine utilizator**: funcții suplimentare, de ex. yywrap().

**✒️ Expresii Regulate în LEX**

* . – orice caracter (cu excepția newline)
* [abc] – clase de caractere
* [^abc] – complementul unei clase
* \*, +, ? – repetiții: zero+, unu+, opțional
* {m,n} – între m și n repetiții
* x|y – alternativă (x sau y)
* "text" – potrivire exactă
* x/y – **lookahead** (x urmat de y)
* ^ – început de linie; $ – sfârșit de linie
* <condiție>expr – expresie activă doar în starea de start condiție

**⚙️ Funcții și variabile utile**

* yytext – textul potrivit cu expresia
* yyleng – lungimea textului
* input(), output(c), unput(c) – manipulare caracter cu fluxul
* yywrap() – apelată la EOF; returnează 1 pentru terminare, 0 pentru continuare

**🧠 Controlul ambiguității și lookahead**

* **Reguli ambigue**: LEX alege:
  1. cea mai lungă potrivire
  2. regula care apare prima dacă mai multe au aceeași lungime
* Operatorul / și funcțiile yyless(n), yymore() permit lookahead/lookbehind și ajustări în potriviri.

**🧩 Condiții de start și contexte**

* Permite schimbarea regulilor active în funcție de contextul liniei.
* Se folosesc cu:
  + %start <nume>
  + BEGIN <nume> pentru activare
  + BEGIN 0 pentru revenire la starea implicită

**🛠️ Exemple practice (din laboratoare)**

1. **Eliminare spații/tab-uri**:
2. [ \t]+$ ;
3. [ \t]+ printf(" ");
4. **Identificare numere întregi/float (stil FORTRAN)**
5. **Histogramă după lungimea cuvintelor** – contorizează lungimea cuvintelor cu lung[yyleng]++
6. **Recunoaștere stringuri cu ghilimele și escape**:
   * Se folosesc yymore() și verificarea ultimului caracter cu yytext[yyleng-1]
7. **Contorizare he și she simultan** folosind REJECT

**🔁 Pași pentru utilizarea LEX**

1. Scrii fișierul .l
2. Compilezi cu lex file.l → generează lex.yy.c
3. Compilezi cu gcc lex.yy.c -ll → obții executabil

**⭐ Alte lucruri de reținut**

* Nu toate expresiile sunt sigure. De ex., ".\*" poate duce la potriviri greșite — preferabil "[^\"]\*" pentru stringuri.
* Comentarii C sunt permise, dar doar în anumite zone (ex. linii indentate sau între %{ ... %}).
* REJECT forțează Lex să încerce o altă regulă chiar dacă una a fost deja potrivită.

Vrei să-ți ofer și un rezumat tip fișă PDF cu aceste informații, ușor de imprimat sau de învățat?