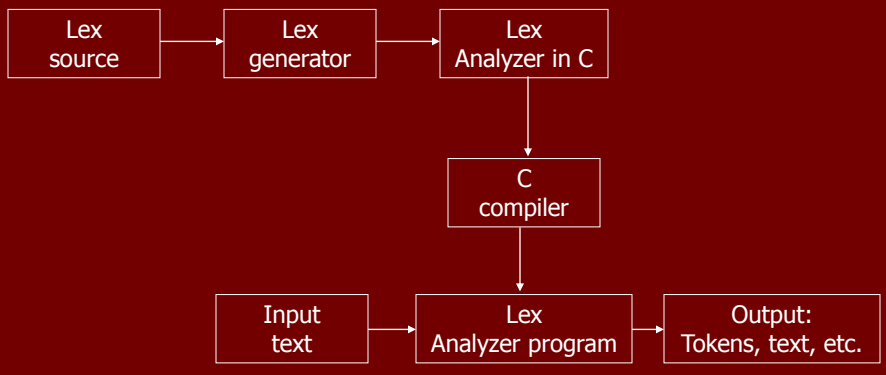
FLEX
(Fast LEXical analyzer)

FLEX

- FLEX je softverski alat za generisanje skenera (leksičkog analizatora): programa koji prepoznaje leksičke šablone u tekstu.
- FLEX se koristi za rešavanje široke klase problema:
  - obrada teksta,
  - pisanje kompajlera,
  - translacija jednog niza karaktera u drugi.

Kreiranje i korišćenje leksičkog analizatora



Kreiranje i korišćenje leksičkog analizatora

- lex source (lex specification)
  - Specifikacija skenera zapisuje se u ulaznoj datoteci. Specifikacija se zadaje u obliku parova regularnih izraza i akcija zapisanih u C-kodu; ovi parovi nazivaju se leksička pravila.
- lex program generator
  - LEX generator čita specifikaciju iz ulazne datoteke.
- lex Analyzer
  - Izlaz iz LEX generatora je leksički analizator u C kodu
- lex Analyzer Program
  - skener u C-u se kompajlira pomoću C kompajlera i generiše se izvršni program - lexical analyzer.

Pisanje flex specifikacije

- FLEX specifikacija se sastoji iz najviše tri sekcije:
  - sekcija definicija
  - sekcija pravila
  - sekcija korisničkog koda
- Sekcija *pravila* je obavezna. Ostale dve sekcije su opcione, ali ako se koriste, moraju se navesti u zadatom redosledu.

Sekcija pravila

- Sekcija pravila otvara se sa graničnikom %%. Ako se koristi sekcija korisničkog koda, navodi se drugi graničnik %% za kraj sekcije pravila.
- Svako leksičko pravilo sadrži:
  - regularni izraz i
  - akcije koje se preduzimaju ako je prepoznat regularni izraz
- Kad god ulaz sadrži string koji lex nije prepoznao, (jer za takav string nije zadato leksičko pravilo), lex na izlazu prikazuje ulazni string.

Regularni izrazi

- Regularni izrazi su stringovi sa ili bez operatora.
- Najprostiji regularni izrazi su stringovi bez operatora, na primer:
  SEDAM;
  JABUKA;
  PLAVO;
- Navedenim regularnim izrazima odgovara svaki ulazni string koji sadrži navadene karaktere.

Regularni izrazi

- Neka su R i S neki regularni izrazi. Operatori koji se koriste u regularnim izrazima su:
  - . regularnom izrazu odgovara bilo koji znak izuzev newline
  - R+ regularnom izrazu odgovara bar jedan R
  - R* regularnom izrazu odgovara nula ili više R-ova
  - R? regularnom izrazu odgovara opciono pojavljivanje R,
  - R|S regularnom izrazu ogovara ili R ili S
  - RS iza regularnog izraza R sledi regularni izraz S
  - [] oznaka za klasu znakova. Postoje tri vrste klase znakova:
    - ✓ prosta klasa znakova,
      [ABC] primer proste klase znakova; regularnom izrazu odgovara ili A ili B ili C
    - ✓ klasa znakova za utvrđenim granicama i
      [a-z] primer klase znakova sa utvrđenim granicama; regularnom izrazu odgovara bilo koje malo slovo engleske azbuke.
    - ✓ negacija klase znakova.
      [^A-Za-z] primer negacije klase znakova; regularnom izrazu odgovara bilo koji znak koji nije slovo

Primer

- Primer za identifikaciju promenljive u mnogim programskim jezicima

  [a-zA-z][0-9a-zA-Z]*

Korektene promenljive:
- a
- R2
- SUMA1
- Ba

Loše promenljive:
- A_B
- 5B
- $C

Lex operatori

- Lex operatori se uključuju u specifikaciju kao znaci koji se pretražuju
  - Navođenjem znaka \ ispred operatora ili
  - Stavljanjem operatora između znakova “

Prepoznavanje pozitivnog celog broja se može zadati sa

\+?[0-9]+ ili "+"?[0-9]+

Čime će biti korektno prepoznati brojevi poput:

+95

75

+ 1

Lex pravila

- Kada skener prepozna string koji odgovara regularnom izrazu na početku pravila, onda obraća pažnju na desni deo pravila koji može da sadrži jednu ili više akcija. Akcije se zapisuju notacijom programskog jezika C.

```
-[0-9]     printf("%s\n", yytext);
```

- Kada lex prepozna u ulaznom tekstu negativan broj, takav broj se memoriše u globalni znakovni niz **yytext**. Dužina stringa memoriše se u globalnu promenljivu **yyleng**.
- Ako se akcija realizuje sa dve ili više C naredbi, onda se one, kao i u C-u, stavljaju u vitičaste zagrade.
- Primer u kome treba prikazati treći znak u prvom prepoznatom celom broju:

```
[0-9]+  {   if (yyleng > 2)
                printf ("%c", yytext[2]);
            }
```

I/O funkcije

- FLEX podržava i tri I/O funkcije:
  - input () - za čitanje znaka
  - unput (c) - za vraćanje poslednjeg učitanog znaka
  - output (c) - za prikazivanje znaka
- Jedan način za ignorisanje svih znakova izmedju dva dupla znaka navoda je korišćenje input ():

```
\"      while ( input () != '"' );
```

Sekcija definicija

- Početak sekcije definicija označava se sa %{, a kraj sa %}.
- Sekcija definicija obično sadrži:
  - deklaracije lokalnih i globalnih promenljivih,
  - "include" direktive i definicije makroa.
- Deklaracije promenljivih imaju isti oblik i funkciju kao i u programskom jeziku C, a to isto važi i za include direktive.

Sekcija definicija

- Sekcija definicija moze da sadrži i deklaraciju regularnih izraza. Opšti oblik je:

  <ime> <regularni izraz>

  <ime> obavezno počinje slovom engleske azbuke iza koga sledi jedno ili više slova, cifara, donja crta '_ ' ili gornja crta '-'.
- Primer deklaracija:

  DIGIT [0-9]+
- U sekciji pravila referencira se na deklarisano ime tako što se ime stavlja u vitičaste zagrade {DIGIT}

Sekcija korisničkog koda

- Potprogrami se koriste u lex-u iz istih razloga zbog kojih se koriste i u drugim programskim jezicima.
- Kod akcija koje se koriste za više pravila može se napisati samo jednom i pozivati kada je potrebna.
- Drugi razlog za korišćenje rutina je uprošćavanje sekcije pravila, bez obzira da li se potprogram koristi u jednom ili u više pravila.

Sekcija korisničkog koda

- Potprogram koji ignoriše komentar u programskom jeziku C gde se komentar zapisuje:
  /* komentar */;
- Napomenimo da komentar ne može biti ugnježden.

```
 "/*"     preskoci ();
%%
preskoci ()  {
   for ( ; ; )  {
        while (input() != '*');
         if (input() != '/')
              unput(yytext[yyleng-1]);
          else return;
   }
}
```

Translacija i izvršavanje Lex-a

- Za dobijanje leksičkog analizatora u C-u:

  lex <ime-prog-lex>.l

  gde je <ime-prog-lex>.l datoteka koja sadrži lex specifikaciju, a FLEX generiše leksički analizator u C-u, koji se nalazi u datoteci yy.lex.c
- Kompajliranje i linkovanje se vrši kao i za svaki drugi C program:

  gcc –o <ime-prog> lex.yy.c -lfl

Primer

- Napisati lex specifikaciju za prebrojavanje broja znakova i novih redova u nekom ulaznom tekstu koji se nalazi u datoteci *tekst.dat*.

Primer

```
%{
#include <stdio.h>
int  yyred = 0;
int  yyznak = 0;
%}
%%
\n      yyred++;
.       yyznak++ ;

%%
main()
 {   extern FILE*yyin;
     yyin = fopen("tekst.dat","r");
     yylex();
     printf(" broj znakova = %d \n", yyznak);
     printf(" broj novih redova = %d \n", yyred);
     fclose(yyin);
}
```