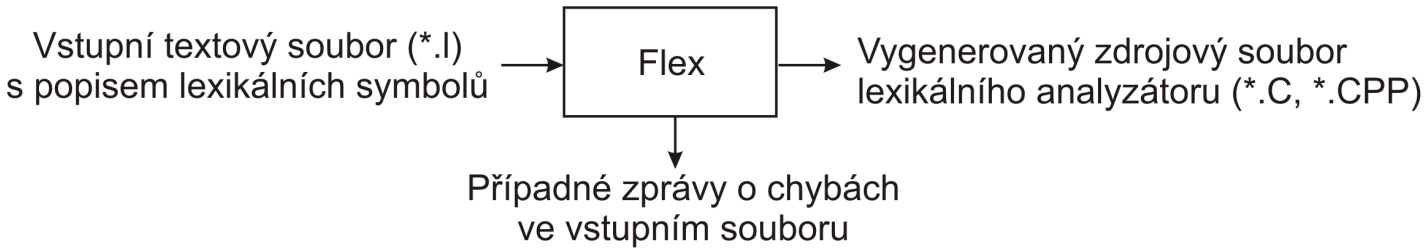
Flex – fast lexical analyzer generator



Formát vstupního souboru:

```
%{
  .. hlavičkové soubory, deklarace a jiné ..
%}
  .. definice ..
%%
  .. pravidla ..
%%
  .. uživatelský kód ..
```

Část, která je mezi symboly `%{` a `%},` Flex nijak nevyužívá a jen ji překopíruje do výstupního souboru. Obsahuje jména hlavičkových souborů a různé deklarace, které jsou zapotřebí pro následný překlad vygenerovaného lexikálního analyzátoru.

V části *definice* jsou definice dílčích regulárních výrazů zapsané jako dvojice:

jméno definice-regulárního-výrazu

Vzory

x	rozpoznání konkrétního znaku x .
.	libovolný znak (byte) vyjma znaku nového řádku.
[xyz]	"třída znaků"; rozpozná některý ze znaků uvedených v závorkách, zde to bude x, y nebo z .
[abj-oZ]	"třída znaků" obsahující interval; zde rozpozná znaky a, b, libovolný znak od j po o nebo znak Z .
[^A-Z]	"negovaná třída znaků", rozpozná libovolný znak, který není ve třídě uveden. Zde rozpozná libovolný znak vyjma velkých písmen.
[^A-Z\n]	libovolný znak vyjma velkých písmena a znaku nového řádku.
r*	žádný nebo libovolný počet výskytů regulárního výrazu *r* .
r+	jeden nebo více výskytů regulárního výrazu *r* .
r?	žádný nebo jeden výskyt regulárního výrazu *r* .
r{2,5}	nejméně dva a nejvíce pět výskytů regulárního výrazu *r* .
r{2,}	dva nebo více výskytů regulárního výrazu *r* .
r{4}	právě čtyři výskyty regulárního výrazu *r* .
{jméno}	použití již definovaného regulárního výrazu s daným *jménem*.

"[xyz]\"foo" Pokud potřebujeme rozpoznat text, který obsahuje znak, jež má specifický význam v zápisu regulárního výrazu, text uzavřeme do uvozovek. Zde rozpozná text `[xyz]"foo` .

x je-li *x* písmeno `a`, `b`, `f`, `n`, `r`, `t` nebo `v`, pak je to dle ANSI-C interpretováno jako `\`*x*. Jinak je to konkrétní znak *x* . Lze použít k zápisu znaků, které jsou zároveň operátory regulárního výrazu, například `\*` .

\0 znak s hodnotou nula (jeho ASCII kód je `0`).

\123 znak s oktalovou hodnotou `123` .

\x2a znak s hexadecimální hodnotou `2a` .

(*r*) závorky pro změnu precedence.

rs regulární výraz *r* následovaný regulárním výrazem *s*, tj. konkatenace *r* a *s* .

r*|*s buďto *r* anebo *s* .

r*/*s *r* jen v případě, je-li za ním *s*, tj. *r* v kontextu *s* . Výstupem této akce je jen text rozpoznaný *r* . Text rozpoznaný *s* je vrácen zpět na vstup.

^*r* *r* jen v případě, kdy je na začátku řádku, tj. na začátku souboru nebo po znaku nového řádku.

***r*$** *r* jen v případě, kdy je na konci řádku, tj. před znakem nového řádku. Ekvivalentní s *r*`/\n` . Některé systémy mohou pro konec řádku používat dvojici znaků `\r\n`, pak by to bylo ekvivalentní s *r*`/\r\n` .

<*s*>*r* *r* jen za počáteční podmínky *s* .

<*s*$_1$,*s*$_2$,*s*$_3$>*r* *r* při některé z počátečních podmínek *s*$_1$, *s*$_2$ nebo *s*$_3$.

<*>*r* *r* při libovolné počáteční podmínce.

<<EOF>> konec souboru.

<*s*$_1$,*s*$_2$><<EOF>> konec souboru při počáteční podmínce *s*$_1$ nebo *s*$_2$.

Příklady.

```
CISLICE  [0-9]
({CISLICE}+"."{CISLICE}*)|("."{CISLICE}+)
({CISLICE}+\.{CISLICE}*)|(\.{CISLICE}+)
ID       [a-zA-Z_][a-zA-Z_0-9]*
PISMENO_ [a-zA-Z_]
ID       {PISMENO_}({PISMENO_}|{CISLICE})*
```

Lze rovněž použít třídy znaků:

```
[:alnum:] [:alpha:] [:blank:]
[:cntrl:] [:digit:] [:graph:]
[:lower:] [:print:] [:punct:]
[:space:] [:upper:] [:xdigit:]
```

Tyto třídy odpovídají znakům rozpoznávanými funkcemi jazyka C: *isalnum(), isalpha()* atd.

Konkatenace má vyšší precedenci než operátor `|` . Tj. výrazy `abc|xy` a `(a(bc))|(xy)` jsou stejné vzory. Operátory násobnosti mají vyšší precedenci než konkatenace (a operátor |). Tj. výraz `xyz*` je stejný vzor jako `xy(z*)` .

Příklad. Následující zápisy jsou ekvivalentní.

```
[[:alnum:]]
[[:alpha:][:digit:]]
[[:alpha:]0-9]
[a-zA-Z0-9]
```

V části *pravidla* jsou pravidla ve tvaru:

```
vzor  akce
```

Vzor musí začínat na novém řádku. *Akce* musí začínat na stejném řádku, na kterém je *vzor*.

V části *uživatelský kód* jsou různé funkce, které využívá (volá) lexikální analyzátor při své činnosti. Pokud je tato část prázdná, oddělovače **%%** mohou být vynechány.

Hodnoty a funkce dostupné pro uživatele

`FILE *yyin` – vstupní textový soubor s popisem lexikálních symbolů.

`char *yytext` – obsahuje text aktuálně rozpoznaného symbolu. Text je ukončen znakem \0.

`int yyleng` – délka textu aktuálně rozpoznaného symbolu.

`char input()` – čte znak ze vstupního souboru, funkce pro jazyk C.

`char yyinput()` – čte znak ze vstupního souboru, funkce pro jazyk C++.

`int yylex()` – funkce vygenerovaného lexikálního analyzátoru.