Lex/Flex wiadomości podstawowe;:

Przyponijmy, że analizą leksykalną nazywamy podział danych wejściowych składających się z ciągu znaków (tekstu, zawartości pliku, itp..) na ciąg okreslonych symboli leksykalnych nazywanych tokenami. Tokeny opisaywane są zazywaczaj za pomocą wyrażeń regularnych. Każde wyrażenie regularne opisuje pewien język – zbiór ciągów znaków dających się do niego dopasować.

Programy wykonujące analizę leksyklaną nazywamy lekserami. Program taki może zostać zaimplementowany bezposrednio w dowolnym języku programowania. Można również wykorzystać w tym celu narzędzia umożliwiające generowanie kodu źródłowego analizatora leksykalnego na podstawie przygotowanej specyfikacji działania. Przykłądem tego typu narzędzia jest program lex lub odpowiadający mu w systemie GNU program flex.

Zadanie wstępne: Przypomnij informacje dotyczące analizy leksykalnej oraz programu lex, które poznałeś w trakcie wykładu.

## Przygotowanie specyfikacji analizatora leksykalnengo

Programm lex służy do tworzenia analizatorów leksykalnych na podstawie przygotowanej specyfikacji. Powinna ona określać strukturę poszukiwanych tokenów oraz reguły zachowania w przypadku pozytywnego dopasowania. Specyfikaja dla programu lex ma postać pliku tekstowego o następującym formacie.

*%(   
 deklaracje użytkownika  
 %)  
 definicje wyrażeń regularnych  
 %%  
 reguły dopasowania  
 %%  
 definicje funkcji i procedur*

Część pierwsza zawierająca deklaracje użytkownika jest kopiowana do wynikowego źródła leksera bez zmian. W tym miejscu powinny być umieszczone deklaracje dołączenia specyficznych plików nagłówkowych, deklaracje zmiennych globalnych, funkcji i procedur użytkownika, itp.

W części drugiej umieszczane są definicje wyrażeń regularnych wykorzystywanych do rozpoznawania symboli leksykalnych. Do konstrukcji wyrażeń możemy używać następujących operacji:

* | - suma dwóch wyrażeń regularnych
* [a-m] – operator zakresu
* () – grupowanie
* \*- domknięcie Kleene’go (zero lub więcej wystąpień)
* +- co najmniej jedno wystąpienie
* ?- co najwyżej jedno wystąpienie
* . – dowolny znak(poza znakiem nowego wiersza)
* ^ - lewy kontekst – pasuje do do początku wiersza
* $- prawy kontekst – pasuje do końca wiersza
* „…” – dopasowanie dosłowne.

Zdefiniowane proste wyrażenia mogą służyć do budowania bardziej skomplikowanych wyrażeń. Do wcześniej zdefiniowanego wyrażenia odwołujemy się umieszczając je w nawiasach { oraz }.  
 {wyrazenie}

Część trzecia zawiera reguły określające zachowanie programu po pozytywnym dopasowaniu okreslonych wyrażeń regularnych. Reguły te są postaci:  
 wzorzec { kod języka C }  
Zachowanie leksera po dopasowaniu wzorca określanie jest za pomocą fragmentów kodu w języku C. Możemy więc korzystać ze zmiennych i funkcji zdefiniowanych części deklaracji. Ponadto możemy używać między innymi poniższych zmiennych specjalnych:

* yytext – ostatnio dopisany wzorzec (char\*)
* yyleng – długość ostatnio dopasowanego wzorca (int)
* yyin – wejście programu (FILE \*)
* yyout – wyjście programu (FILE \*)

W częście czwartej umieszczane są definicje funkcji i procedur. W sczególności możemy umieścić w tym miejscu definicję funkcji main. Analiza leksykalna jest wykonywana przez funkcję yylex() wygenerowaną przez program lex. W przypadku nie dostarczenia implementacji funkcji main, lex wygeneruje ją z domyślną zawartością składającą się z pojedynczego wywołania funkcji yylex().

**Kompilacja i uruchomienie**

Kompilacja plików źródłowych opisujących działanie analizatora leksykalnego przebiega dwuetapowo. W pierwszym etapie na podstawie przygotowanej specyfikacji program lex generuje kod źródłowy analizatora leksykalnego.

lex specyfikacja.l

Wygenerowany przez powyższe polecenie kod źródłowy umieszczany jest w pliku o nazwie lex.yy.c. W drugim etapie jest on kompilowany za pomocą kompilator języka C. Aby kompilacja zakończyła się sukcesem wymagane jest dołączenie biblioteki fl.