Rozdział 3

Analiza leksykalna

W tym rozdziale pokażemy, jak skonstruować analizator leksykalny (lekser). W samodzielnej implementacji leksera pomocne będzie rozpoczęcie od diagramu lub innego opisu leksemów dla każdego tokenu. Następnie będziemy mogli napisać kod rozpoznający każde wystąpienie każdego leksemu w danych wejściowych i zwracający informacje o zidentyfikowanym tokenie.

Możemy również utworzyć lekser automatycznie, specyfikując wzorce leksemów w *generatorze analizatorów leksykalnych* i kompilując te wzorce w kod, który będzie funkcjonował jako lekser. Podejście to ułatwia modyfikowanie utworzonego analizatora leksykalnego, gdyż musimy napisać od nowa jedynie odpowiednie wzorce, a nie cały program. Przyśpiesza to również proces implementacji leksera, gdyż programista specyfikuje oprogramowanie na bardzo wysokim poziomie wzorców i polega na generatorze pod względem tworzenia szczegółowego kodu. W podrozdziale 3.5 przedstawimy generator analizatorów leksykalnych o nazwie *Lex* (lub *Flex* w jego nowszym wcieleniu).

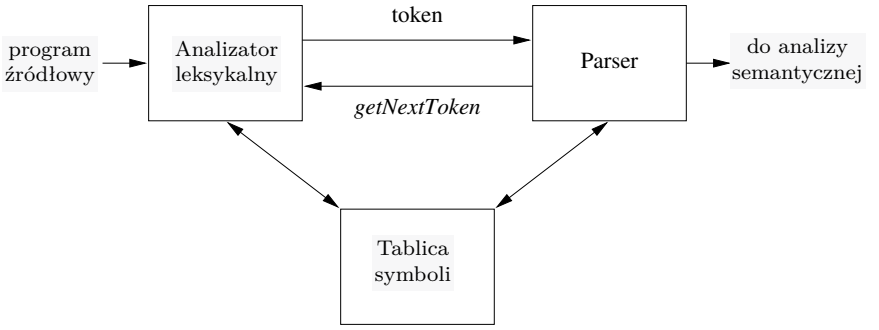
Badanie generatorów analizatorów leksykalnych zaczniemy od wprowadzenia wyrażeń regularnych, poręcznej notacji dla specyfikowania wzorców leksemów. Pokażemy, jak można przekształcać tę notację, najpierw w niedeterministyczne automaty, a później w automaty deterministyczne. Te dwie notacje mogą zostać użyte jako wejście do „sterownika", czyli kodu, który symuluje te automaty i używa ich do ustalania kolejnego tokenu. Ten sterownik i specyfikacje automatów stanowią jądro analizatora leksykalnego.

3.1. Rola analizatora leksykalnego

Będąc pierwszą fazą działania kompilatora, głównym zadaniem leksera jest odczytywanie z wejścia znaków programu źródłowego, grupowanie ich w leksemy i utworzenie tokenu dla każdego leksemu ze źródłowego programu i potraktowanie takiej sekwencji tokenów jako wyjścia. Strumień tokenów jest następnie

przekazywany do parsera w celu analizy składniowej. Analizator leksykalny typowo odwołuje się również do tablicy symboli. Gdy lekser odkryje leksem tworzący identyfikator, musi wprowadzić ten leksem do tablicy symboli. W niektórych przypadkach informacje dotyczące rodzaju identyfikatora mogą być czytane z tablicy symboli przez lekser, aby ułatwić ustalenie właściwego tokenu, który musi zostać przekazany do parsera.

Te interakcje zostały przedstawione na rysunku 3.1. Zazwyczaj interakcja jest implementowana przez wywołanie analizatora leksykalnego przez parser. To wywołanie, przedstawione tutaj przez polecenie *getNextToken*, powoduje, że analizator leksykalny odczytuje kolejne znaki z wejścia, aż zidentyfikuje następny leksem i utworzy dla niego odpowiedni token, który zwróci do parsera.



RYSUNEK 3.1: Interakcje między analizatorem leksykalnym a parserem

Ponieważ lekser jest tą częścią kompilatora, która czyta tekst źródłowy, może wykonywać pewne inne zadania poza identyfikowaniem leksemów. Jednym z nich jest usuwanie komentarzy i „białych znaków" (spacji, znaków nowego wiersza, tabulatorów, a być może również innych znaków służących do oddzielania tokenów w ciągu wejściowym). Inne zadanie to korelowanie komunikatów błędów generowanych przez kompilator z programem źródłowym. Na przykład lekser może rejestrować liczbę kolejnych napotkanych znaków nowego wiersza, dzięki czemu może przypisać numer wiersza do każdego komunikatu błędu. W niektórych kompilatorach analizator leksykalny wykonuje kopię programu źródłowego z komunikatami błędów wstawionymi w odpowiednich pozycjach. Jeśli program źródłowy używa preprocesora makr, rozwijanie makr również może być wykonywane przez analizator leksykalny.

Niekiedy leksery są podzielone na dwa procesy wykonywane kaskadowo:

(a) *Skaner* realizuje proste operacje, które nie wymagają tokenizacji danych wejściowych, takie jak usuwanie komentarzy i kompresja następujących po sobie białych znaków w jeden.

(b) Właściwy *analizator leksykalny* jest bardziej złożoną częścią, która tworzy tokeny z wyjścia skanera.