Informatyka, 3 rok
Wydział EAIiIB
Teoria Kompilacji i Kompilatory

Projekt zaliczeniowy

KALKULATOR

Krzysztof Misiak
Filip Pasternak

1 Opis działania kalkulatora

Celem projektu jest przedstawienie działania prostego kalkulatora.

Funkcjonalności kalkulatora :

- przetwarzanie podstawowych działań algebraicznych (dodawanie, odejmowanie , mnożenie, dzielenie, potęgowanie, negacja)
- wczytywanie i wypisywanie działań w konsoli
- wczytywanie i zapisywanie działań do pliku
- wyświetlanie błędu w przypadku nieprawidłowego działania wejściowego
- operacje na liczbach rzeczywistych
- obsługa funkcji trygonometrycznych
- rozpoznawanie notacji wykładniczej

Projekt realizowany jest w języku Java. Docelowo zostanie zaimplementowany graficzny interfejs użytkownika.

2 Opis gramatyki

Produkcje używanej gramatyki według notacji Backusa-Naura

<wejscie> ::= ε
|<wejscie> <linia>

<linia> ::= ‘\n’
| <wyrazenie> ‘\n’

<trygonometria> ::= sin(<wyrazenie>)
| cos<wyrazenie>
| tg(<wyrazenie>)
| ctg(<wyrazenie)

<wyrazenie> ::= <liczba>
| <trygonometria>
| <wyrazenie> + <wyrazenie>
| <wyrazenie> – <wyrazenie>
| <wyrazenie> * <wyrazenie>
| <wyrazenie> / <wyrazenie>
| – <wyrazenie>
| <wyrazenie> ^ <wyrazenie>
| (<wyrazenie>)

<zero> ::= 0

<cyfra niezerowa> ::= 1
| 2
| 3
| 4
| 5
| 6
| 7
| 8
| 9

<cyfra> ::= <zero>
| <cyfra niezerowa>

<ciag cyfr> ::= <cyfra>
| <cyfra><ciag cyfr>

<calkowita> ::= <cyfra>
| <cyfra niezerowa><ciag cyfr>

<liczba> ::= <calkowita>
| <calkowita> . <ciag cyfr>

3 Skaner

3.1 Opis skanera

Skaner został skonstruowany tak, aby na podstawie opisanej w poprzednim rozdziale gramatyki wyodrębniać z otrzymanego na wskazanym wejściu łańcucha znaków tokenów charakterystycznych dla kalkulatora. Tokeny reprezentowane są przez instancje klasy Yytoken, która to posiada pola ‘type’ i ‘attribute’ oznaczające odpowiednio rodzaj tokena i opcjonalny atrybut (przykładowo typ: Liczba, którego atrybutem jest wartość tej liczby).

W późniejszych krokach tworzenia kompilatora zostaną stworzone dodatkowe klasy dziedziczące po Yytoken, charakteryzujące odmienne ich rodzaje i implementujące odpowiednie właściwości i zachowania.

Skaner ten umożliwia wprowadzanie na wejściu komentarzy w notacji języka Java (komentarz liniowy i blokowy) oraz JavaDoc. Napotykając takowe ignoruje je, dzięki czemu nie mają wpływu na kod wynikowy.

3.2 Sposób tworzenia

Do stworzenia skanera wykorzystane zostało narzędzie JFlex, które pozwala na generowanie analizatorów leksykalnych w języku Java w oparciu o plik z rozszerzeniem ‘.flex’. Plik ten dzieli się na trzy części (oddzielone znakiem %%).

Pierwsza zawiera kod w języku Java, który zostanie wstawiony bez zmian do plików wynikowych (.java). Umożliwia stworzenie dodatkowych elementów potrzebnych w programie (np. bufory, klasy) czy też dołączenie dodatkowych plików, bibliotek. Została użyta do stworzenia głównej klasy programu i implementacji funkcji main, w której ciele wywoływana jest metoda yylex() klasy Yylex, będącej właściwym skanerem. Zawiera także definicję klasy Yytoken reprezentującej żeton.

Druga sekcja to miejsce na definiowanie makr dla skanera. Polega to na podaniu nazwy makra i odpowiadającego mu wyrażenia regularnego. Makra te można wykorzystać w kolejnej części pliku flex opisanej niżej. W drugiej części można wykorzystać także dodatkowe opcje dla skanera lub poźniejszego parsera, określać stany w jakich może znajdować się skaner (na bieżącym etapie tworzenia nie były potrzebne żadne dodatkowe opcje, gdyż zależą one w dużej mierze od właściwej formy parsera).

Ostatnia sekcja pliku flex jest kluczowa z punktu widzenia skanera. Określa się tutaj jego konkretne zachowanie w zależności od dopasowanego wzorca. Dla poszczególnych wzorców podane są instrukcje w języku Java, które podejmowane są w momencie wykrycia.

4 Parser

4.1 Opis Parsera

Parser został utworzony z wykorzystaniem narzędzia CUP. Na podstawie pliku parser.cup generowana jest klasa java, w której zawarte są wygenerowane metody oraz metody odziedziczone po interfejsie java_cup.runtime.Parser. Parser został tak utworzony, by korzystać z wcześniej stworzonej klasy Scanner. W konstruktorze ustawiany jest obiekt konkretnego skanera, który generuje symbole(tokeny).

W tej klasie jest też definiowana gramatyka, symbole terminalne oraz nieterminalne, a także priorytety poszczególnych działań, w tym nawiasowanie.

4.2 Sposób tworzenia

Plik parsera składa się z kilku części:

- parser code – część kodu dołączona do wynikowej klasy
- init with – częśc kodu odpowiadająca za inicjalizację
- scan with – część kodu odpowiadająca za wczytywanie kolejnych tokenów
- gramatyka – definicja symboli terminalnych, nieterminalnych, priorytetów operatorów, reguł gramatyki