Compilers 2021/2022

Projekt 2: kalkulator

Piotr Plebański
121309

15 września 2022

1 Opis

Projekt implementuje kalkulator wyrażeń arytmetycznych. Zastosowano następującą gramatykę bezkontekstową: Symbol E oznacza docelowe wyrażenie arytmetyczne, a symbol num - liczbę naturalną (ciąg cyfr).

$$E \rightarrow TK|-FR$$

$$K \rightarrow +TK|-TK|\varepsilon$$

$$T \rightarrow GL$$

$$L \rightarrow *GL|/GL|\varepsilon$$

$$F \rightarrow G|-F$$

$$R \rightarrow \$|\varepsilon$$

$$G \rightarrow num|(E)$$

Oznaczenia symboli terminalnych:

- *num* – liczba naturalna,
- *+,-,*,/* – operatory arytmetyczne,
- *$* – koniec tekstu,
- *()* – nawiasy grupujące,
- ε – puste słowo.

Gramatyka ta uwzględnia:

- wyłącznie liczby całkowite (również jako wynik dzielenia),
- operatory arytmetyczne: +,-,*,/ z uwzględnieniem priorytetów operatorów,
- nieprzerwany ciąg negacji "-. . . G".

Gramatyka nie uwzględnia białych znaków.

2 Parser

Zastosowano parser *top-down* typu LL(1). Zbiory FIRST, FOLLOW oraz tabela parsingu przedstawione są w tab . 1.

	FIRST	FOLLOW	+	-	*	/	(	)	$	num
E	num,(,-	),$		[-, F, R]			[T, K]			[T, K]
K	+,-,ε	),$	[+, T, K]	[-, T, K]				ε	ε	
T	num, (	+,-,),$					[G, L]			[G, L]
L	*,/,ε	+,-,),$	ε	ε	[*, G, L]	[/, G, L]		ε	ε	
F	num,(,-	$, ε		[-, F]			[G]			[G]
R	$, ε	),$						ε	[$]	
G	num, (	+,-,*,/,),$					[(, E,)]			num

Tabela 1: Tablica parsingu.

3 Algorytm

1. Analiza leksykalna. Rezultatem jest lista tokenów.
2. Analiza składniowa. Parser sprawdza zgoność listy tokenów z gramatyką poprzez budowanie drzewa parsowania. Do znajdowania oczekiwanych tokenów używana jest tablica parsingu (tab. 1) oraz pomocniczy stos. Rezultatem jest drzewo, jeśli wyrażenie jest poprawne.
3. Ewaluacja drzewa. Węzły drzewa (E, T, F, G i n) posiadają reguły pozwalające na ich ewaluację. Obliczanie wartości wyrażenia dla poddrzew odbywa się rekurencyjnie. Ewaluacja kończy się powodzeniem, jeśli wyrażenie nie zawiera nieobsługiwanych operacji, np. dzielenia przez 0.

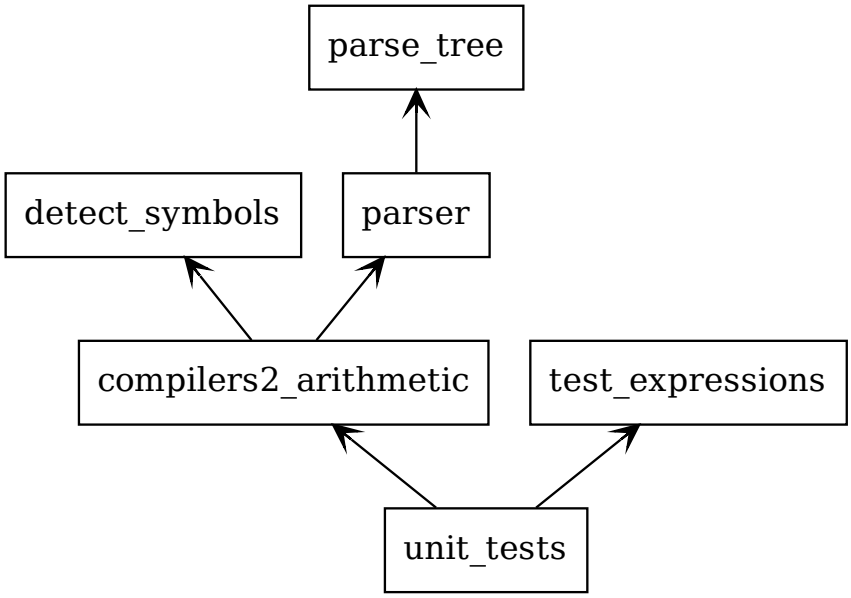
4 Struktura projektu

Projekt został zrealizowany w języku Python 3.8 używając jedynie pakietów z biblioteki standardowej.

4.1 Moduły w projekcie

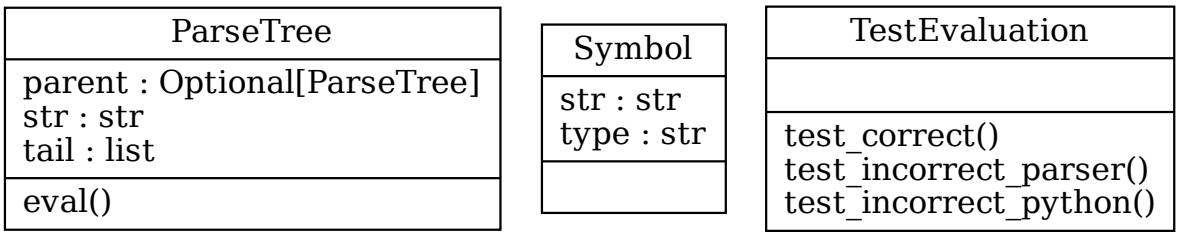
- compilers2_arithmetic.py – główny plik z funkcją walidującą wyrażenie podane przez użytkownika,
- detect_symbols.py – lexer, zawiera również klasę *Symbol* opisującą tokeny,
- parser.py – parser,
- parse_tree.py – klasa *ParseTree*, której instancje są węzłami drzewa parsowania, posiada również metodę do rekurencyjnej ewaluacji węzła,
- test_expressions.py – zbiór wyrażeń do testów,
- unit_tests.py – testy jednostkowe.

4.2 Diagram zależności między modułami



Rysunek 1: Diagram zależności między modułami.

4.3 Diagram klas



Rysunek 2: Diagram klas użytych w projekcie.