Метод анализа

Алгоритм рекурсивного спуска для анализа логических выражений в грамматике языка Pascal G[‹Expr›]:

1. Начинаем с символа ‹Expr›.

2. Метод для символа ‹Expr› вызывает метод для символа ‹OrExpr›.

3. Метод для символа ‹OrExpr› вызывает метод для символа ‹AndExpr›.

4. Метод для символа ‹AndExpr вызывает метод для символа ‹NotExpr›.

5. Метод для символа ‹NotExpr› проверяет наличие оператора отрицания '.NOT.', если он есть, продолжает непосредственно с метода для символа ‹RelExpr›.

6. Метод для символа ‹RelExpr› проверяет тип текущего токена. Если это идентификатор, логический литерал или число, то переходим к следующему токену.

7. Метод для символа ‹RelExpr› проверяет, является ли текущий токен оператором сравнения. Если это так, вызываем метод для символа ‹AddExpr›.

8. Метод для символа ‹AddExpr› вызывает метод для символа ‹MulExpr›, затем проверяет наличие оператора арифметической операции и вызывает метод для символа ‹MulExpr› еще раз.

9. Метод для символа ‹MulExpr› вызывает метод для символа ‹UnaryExpr› .

10. Метод для символа ‹UnaryExpr› вызывает метод для символа ‹Factor› и проверяет наличие операторов умножения, деления или остатка от деления, вызывая метод для символа ‹Factor› при их наличии.

11. Метод для символа ‹Factor› проверяет тип текущего токена. Если это идентификатор, целочисленное или вещественное число, то переходим к следующему токену. Если это символ '(', вызываем метод для символа ‹UnaryExpr›, затем проверяем наличие символа ')', иначе возвращаем ошибку.

12. Если текущий токен не является допустимым для данного выражения, возвращаем ошибку.

13. Если достигнут конец выражения и ошибок не возникло, возвращаем успешный результат