Ministerul Educaţiei şi Cercetării al Republicii Moldova

Universitatea Tehnică a Moldovei

Facultatea Calculatoare, Informatică şi Microelectronică

LABORATORY WORK NO.6

Topic: Parser

Elaborated:

st. gr.

FAF-223 Ciornii Alexandr

Verified:

asist. univ.

Dumitru Cretu

Chişinău - 2024

Objectives:

1. Get familiar with parsing, what it is and how it can be programmed [1].
2. Get familiar with the concept of AST [2].
3. In addition to what has been done in the 3rd lab work do the following:
    i. In case you didn't have a type that denotes the possible types of tokens you need to:
        a. Have a type *TokenType* (like an enum) that can be used in the lexical analysis to categorize the tokens.
        b. Please use regular expressions to identify the type of the token.
    ii. Implement the necessary data structures for an AST that could be used for the text you have processed in the 3rd lab work.
    iii. Implement a simple parser program that could extract the syntactic information from the input text.

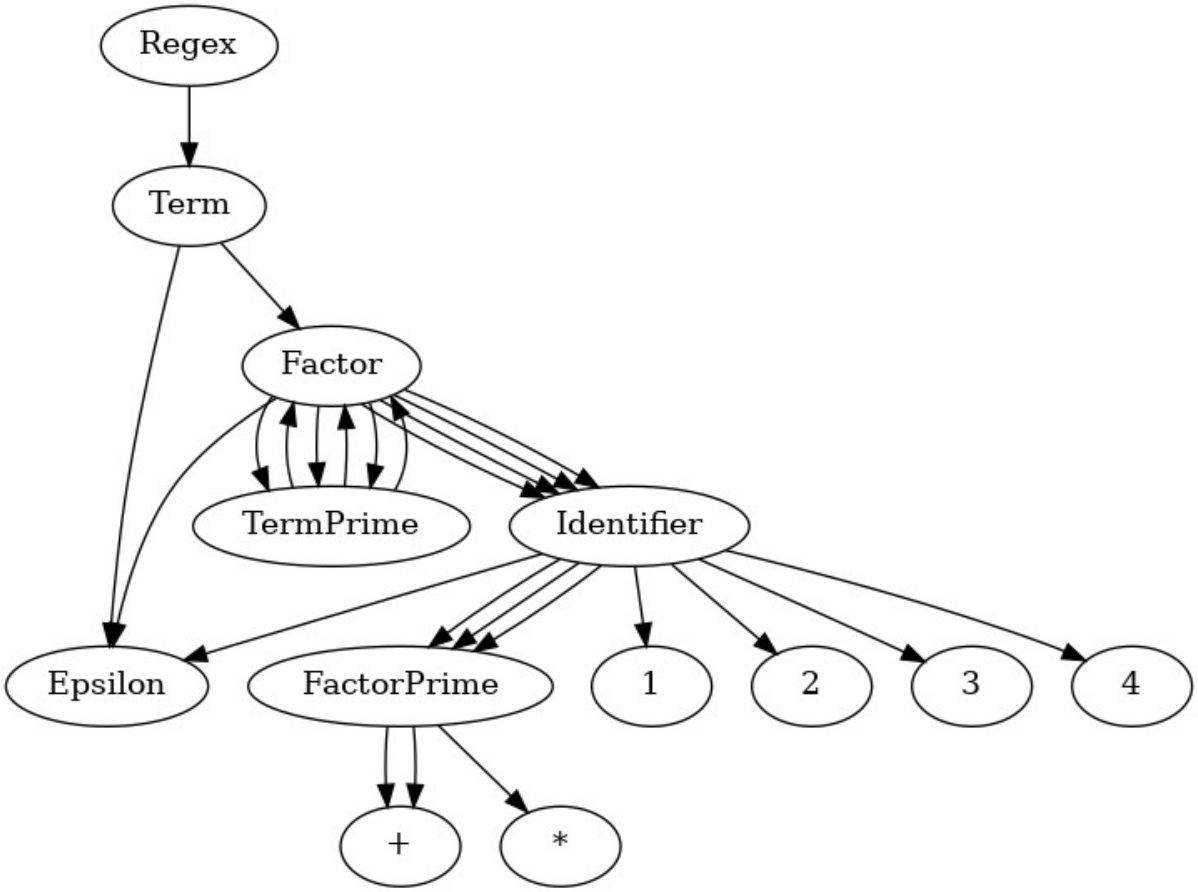
Implementation:

It’s literally code from 3rd lab (lexer) with addition of

```
public class ASTNode
{
    public string NodeType { get; set; }
    public List<ASTNode> Children { get; set; }

    public ASTNode(string nodeType)
    {
        NodeType = nodeType;
        Children = new List<ASTNode>();
    }

    public void Print(int indent = 0)
    {
        Console.WriteLine(new string(' ', indent) + NodeType);
        foreach (var child in Children)
        {
            child.Print(indent + 2);
        }
    }

    public string ToDotString()
    {
        string dot = $"\"{NodeType}\"";
        foreach (var child in Children)
        {
            dot += $" -> \"{child.NodeType}\"";
        }
        dot += ";";
        foreach (var child in Children)
        {
            dot += "\n" + child.ToDotString();
```

```
        }
        return dot;
    }
}
```

Also here’s results for 1+2+3*4 input



Enter a regular expression:
1+2+3*4
Tokens:
Type: Identifier, Value: 1
Type: Plus, Value: +
Type: Identifier, Value: 2
Type: Plus, Value: +
Type: Identifier, Value: 3
Type: Asterisk, Value: *
Type: Identifier, Value: 4

AST Visualization:
Regex
  Term
    Factor
      Identifier
        1

```
    FactorPrime
      +
  TermPrime
    Factor
      Identifier
        2
      FactorPrime
        +
    TermPrime
      Factor
        Identifier
          3
        FactorPrime
          *
      TermPrime
        Factor
          Identifier
            4
          Epsilon
        Epsilon
Epsilon

AST visualization saved as /home/urmjsty/software_eng/RiderProjects/LFALab/AST/ast.png
```