Danny Machado - 822141543
Julia Caroline de Paiva Silva - 822150064
Thais Aires Paiva - 822147596
João Victor Barreto - 822138704
Luís Gabriel da Silva - 822159775

Compilador

Projeto A3

São Paulo
2023

SUMÁRIO

1. INTRODUÇÃO.....3
2. GRAMÁTICA.....4
3. ANTLR4..... 6

Link Vídeo do projeto: Compilador JokerScript

1. INTRODUÇÃO

Desenvolver um compilador é uma tarefa desafiadora. Um compilador é um tipo de software que traduz código-fonte escrito em uma linguagem de programação de alto nível para um código equivalente em linguagem de máquina ou outra forma executável.

A linguagem de programação é fundamental para a computação, isto porque desempenha um papel estabelecer comunicação entre o computador e humano, o que a torna fundamental. Não obstante o desenvolvimento e implementação de linguagens de programação é uma tarefa árdua com fortes bases na engenharia de software e teoria da computação, o que torna a sua criação um dos desafios mais empolgantes e enriquecedores que um programa enfrenta.

O presente projeto tem como objetivo desenvolver um compilador utilizando a linguagem Java como base. O compilador a ser desenvolvido terá a capacidade de traduzir programas escritos em JokerScript para código de máquina, possibilitando sua execução em diferentes plataformas computacionais. Este trabalho visa explorar os conceitos teóricos de compiladores, bem como a aplicação prática desses conhecimentos na criação de um software funcional e educativo.

JokerScript vai proporcionar uma maior simplicidade na idealização, lógica e compreensão de códigos tanto por parte dos desenvolvedores, quanto dos visualizadores. Também é importante destacar que a atribuição de palavras chaves (tokens), poderá ser vista e atribuída de uma forma mais clara do que em outras linguagens de programação, devido a utilização da língua nativa brasileira.

2. GRAMÁTICA

A gramática JokerScript define as regras para iniciar o compilador e gerar a árvore abstrata. O processo começa com o comando `**inicioprograma**` e o arquivo de texto a ser compilado.

Existem dois tipos iniciais de declaração que o compilador reconhece: declaração de variáveis (`**Stat**`) ou declaração de funções. Uma declaração de variáveis pode começar com um identificador seguido por '=' e uma expressão, seguido por ';', ou pode ser apenas uma expressão.

As funções seguem regras específicas, obrigatoriamente precisam ter a seguinte estrutura:

```
FUNCAO ID '(' ID (',' ID)* ')' '{' stat* '}' ;
```

As expressões podem ser representadas de várias maneiras, incluindo identificadores, dígitos, negação de uma expressão (`'**nao**' **expr**`), combinação de duas expressões com 'e' (`'**expr** '**e**' **expr**'`) ou 'ou' (`'**expr** '**ou**' **expr**'`).

Os tokens, representando palavras reservadas e caracteres especiais, têm expressões regulares associadas a eles:

- `E: 'e'`
- `OU: 'ou'`
- `NAO: 'nao'`
- `IGUAL: '='`
- `PONTO: ','`
- `PONTOEVIRGULA: ';'`
- `PARENESQ: '('`
- `PARENDIR: ')'`
- `CHAVESQ: '{'`
- `CHAVDIR: '}'`
- `FUNCAO: 'funcao'`
- `DIGITO: [0-9]+`
- `ID: [a-zA-Z][a-zA-Z0-9]*`
- `WS: [ \t\n\r\f]+ -> skip` (para ignorar espaços em branco)

Expr.g4

```
Click here to ask Blackbox to help you code faster |
grammar Expr;

options { tokenVocab=ExprLexer; }


inicioprograma
    : stat* EOF
    | function* EOF
    ;

stat: ID '=' expr ';'
    | expr
    ;

function : FUNCAO ID '(' ID (',' ID)* ')' '{' stat* '}' ;


Code Suggestions

expr: ID
    | DIGITO
    | 'nao' expr
    | expr 'e' expr
    | expr 'ou' expr
    ;

E : 'e' ;
OU : 'ou' ;
NAO : 'nao' ;
IGUAL : '=' ;
PONTO : ',' ;
PONTOEVIRGULA : ';' ;
PARENESQ : '(' ;
PARENDIR : ')' ;
CHAVESQ : '{' ;
CHAVDIR : '}' ;
FUNCAO: 'funcao' ;
DIGITO : [0-9]+ ;
ID: [a-zA-Z][a-zA-Z0-9]* ;
WS: [ \t\n\r\f]+ -> skip ;

```

3. FERRAMENTA ANTLR4

O ANTLR4 (ANother Tool for Language Recognition, versão 4) é uma poderosa ferramenta para a geração de analisadores léxicos, sintáticos e semânticos, amplamente utilizada em aplicações de processamento de linguagens. Aqui estão algumas de suas principais características:

Análise Léxica e Sintática: ANTLR4 é capaz de processar a estrutura lexical e sintática de uma linguagem, separando e interpretando tokens e regras gramaticais.

Gramáticas: Utiliza gramáticas definidas em um formato específico, geralmente em arquivos .g4. Essas gramáticas definem as regras para identificação de tokens (léxicos) e estruturas (sintáticos).

Geração de Código: A partir de uma gramática, ANTLR4 gera código-fonte em várias linguagens de programação, como Java, C#, Python, JavaScript, entre outras, para realizar a análise.

Análise de Dependências: Além de simplesmente analisar a entrada, ANTLR4 pode construir árvores de análise (parse trees) e gráficos de dependência para representar a estrutura da linguagem analisada.

Extensível e Personalizável: Permite a personalização através de visitantes ou ouvintes que são acionados durante a análise. Isso permite a manipulação de estruturas analisadas de forma mais granular.

Ampla Aplicação: É usado em diversos domínios, como interpretação de linguagens de programação, processamento de linguagem natural, geração de compiladores e intérpretes, entre outros.

Comunidade e Recursos: Possui uma comunidade ativa e uma variedade de recursos de aprendizado, incluindo documentação, tutoriais e exemplos de código.

ANTLR4 é notável por sua eficiência, flexibilidade e a capacidade de lidar com linguagens complexas, tornando-o uma escolha popular para muitos desenvolvedores e pesquisadores na área de processamento de linguagens.

O ANTLR4 foi usado no JokerScript para gerar classes em Java definindo TOKENS, Interpretadores, os analisadores léxicos, sintáticos e a árvore de análise (Parse Tree) usando interface GUI