Utilizando a ferramenta ANTLR para desenvolvimento de um Interpretador e Tradutor

Bruno Fantineli

Murillo Henryque

Rafael Ribeiro

Universidade Estadual do Norte do Paraná (UENP)

Bandeirantes – PR – Brasil

[2]Centro de Ciências Tecnológicas

Universidade Estadual do Norte do Paraná – Bandeirantes, PR – Brasil

Resumo. Desenvolvimento de um interpretador para a linguagem Mini Pascal em java e a tradução de seu código fonte para C++ utilizando a ferramenta ANTLR.

1. Introdução

O interpretador de Mini Pascal foi desenvolvido como trabalho final para a matéria de compiladores no curso de ciência da computação, juntamente com esse interpretador foi necessário a criação de um tradutor, que ao receber o código em Mini Pascal, tinha como saída o mesmo código em C++. Foi utilizado o ANTLR que é usado como um plugin para o NetBeans.

2. Desenvolvimento

Com o uso do ANTLR é possível desenvolver compiladores e interpretadores de uma maneira fácil, com a declaração de todos os Tokens e a gramática da linguagem, podendo modificar qualquer parte dela.

```
ID          : [a-zA-Z][a-zA-Z0-9]*;
EQ          : '==';
STR         : ["].*?["];
NUM         : [-]?[0-9]+(.[0-9]+)?;
PLUS        : '+';
MINUS       : '-';
TIMES       : '*';
DIV         : '/';
DF          : '!=';
LT          : '<';
GT          : '>';
LTE         : '<=';
GTE         : '>=';
WS          : [\n\r \t]+ -> skip;
```

Figura 1. Declaração de Tokens

A imagem acima demonstra como funciona a declaração dos Tokens, primeiro é necessário definir a gramática do Token e em seguida entre aspas simples, existe a definição dos caracteres que representam esse token.

A utilização dos tokens é feita na gramática do interpretador. Na gramática é definido as funções que a linguagem deve ter. Como por exemplo as funções while, for e as decisões com if, if else e else. Defini também como funciona a declaração das variáveis e as atribuições para elas. Como vai funcionar os comandos de escrita a os comandos para impressão na tela. Existiu um pouco de dificuldade na implementação dos arrays, mas foi possível a utilização deles, e não foi possível a implementação das funções e procedures.

```
progr : PROGRAM ID EOL          {traducao+=      "#include<iostream>\n";
                                traducao+="using namespace std;\n";
                                traducao+="int main()"+open+"\n";

                                } block    #progrRule;

block : varBlock? statementBlock    #blockRule;

varBlock : VAR var+ #varBlockRule;

var: ID (','ID)* VARDEC simpletype EOL    {traducao+=$ID.text + ";" + "\n";} #varIDDec

    | ID OPCOL {aux+=$ID.text;} expr CLCOL                                  #indexedVar
    | ID VARDEC {aux+=$ID.text;}  arraytype                  #varArrayType
    | ID                            {aux+=$ID.text;}              #varID;
```

Figura 1. Parte da gramática de MiniPascal

A gramática do MiniPascal é um pouco diferente de C++, a atribuição de valores para uma variável é necessário a utilização de dois pontos antes do igual, e também a utilização de begin e end depois de cada if ou else.

3. Utilização do Interpretador

Para a utilização do interpretador é necessário um arquivo com a extensão .basic que inclue o código em Mini Pascal, esse arquivo é utilizado como entrada para o programa que passa pela analise sintática e léxica, caso seja aceito ele mostra o resultado d0 programa e a saída imprimi esse resultado, juntamente com um arquivo de extensão .cpp, já traduzido para C++.

```
program teste;

var
    x : integer;
    y : integer;

begin
    x := 15;
    y := 15;
    if x == 15 ther
    begin
    writeln(y)
    end
end
```

Figura 2. Input.basic

Após utilizar esse código como exemplo, a saída é o resultado desse programa e a tradução em C++.

```
#include<iostream>
using namespace std;
int main(){
int x;
int y;
x = 15;
y = 15;
if(x==15){
cout << y << endl;

}
}
```

Figura 3. Tradução para C++