UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

IMPLEMENTAÇÃO DE COMPILADOR

Marcos Silva Laydner
Nathan Sargon Werlich
Higor Nocetti
Sadi Júnior Domingos Jacinto

Professor orientador: Rafael de Santiago

Florianópolis
2020

Marcos Silva Laydner
Nathan Sargon Werlich
Higor Nocetti
Sadi Júnior Domingos Jacinto

IMPLEMENTAÇÃO DE COMPILADOR

Trabalho prático da disciplina INE5622 – Introdução a Compiladores, consistindo na implementação de um compilador (analisador léxico e sintático), com o uso da ferramenta ANTLR4, necessário para obtenção de nota.

Professor orientador: Rafael de Santiago

Florianópolis
2020

Sumário

1 CONTRIBUIÇÃO DOS MEMBROS

Participante	**Contribuição**
Marcos Silva Laydner	Tipos obrigatórios Operações obrigatórias
Nathan Sargon Werlich	Laços *for* e *while* Estruturas de controle *if-then-else* e *switch-case*
Higor Nocetti	Características adicionais da linguagem Códigos de teste
Sadi Júnior Domingos Jacinto	Elaboração do Relatório Definição de funções

2 DESCRIÇÃO DA LINGUAGEM

A linguagem criada chama-se **sadbeep**. A mesma consiste em uma linguagem que não se utiliza de tipagem, além de apresentar a possibilidade de definições de variáveis, e um código inteiro propriamente dito, fora de funções. Além disso, a criação de variáveis não é obrigatória, visto que um código como o do exemplo abaixo é válido:

```
1 1 + 1;
2 "Teste de string não atribuída a nenhuma variável.";
3 true;
```

2.1 REQUISITOS OBRIGATÓRIOS DA LINGUAGEM

2.1.1 TIPOS

- **INT:**
  Para o tipo inteiro, foi considerado qualquer quantidade, maior ou igual à 1, de dígitos de 0 à 9, inicialmente sem limite de quantidade de dígitos.

- **FLOAT:**
  Para o tipo de ponto flutuante, foram considerados dois grupos de dígitos, de qualquer quantidade, maior ou igual à 1 estando no intervalo de 0 à 9, concatenados por um ponto. Assim como o tipo inteiro, a princípio, não foi definido nenhuma limitação da quantidade máxima de dígitos.

  Visando facilitar a leitura da gramática, esses dois tipos (INT e FLOAT) foram representados por um mesmo *token*, chamado **NUMBER**.

```
1 NUMBER: INT | FLOAT;
2 INT: [0-9]+;
3 FLOAT: [0-9]+.[0-9]+;
```

- **BOOL:**
  Um dos tipos adicionais da linguagem, definido devido a utilidade do tipo *booleano* em operações lógicas e condicionais utilizadas por laços como *for* e *while*, e por estruturas condicionais como o *if-then-else*.

```
1 TRUE: 'true';
2 FALSE: 'false';
3 BOOL: TRUE | FALSE;
```

- **STRING:**
  O segundo tipo adicional da linguagem, a definição de *string* usada nesse trabalho comporta tanto caractéres isolados quanto textos longos. Além disso, existem dois caractéres usados para se delimitar e definir uma *string*, a saber: ' e ".

```
1 STRING: '\''~[\r\n']* '\'' | '"'~[\r\n']*'"';
```

2.1.2 OPERADORES

Os operadores implementados foram baseados no código de exemplo disponibilizado no *moodle*, sendo divididos em três categorias:

1. **exp:** Operações lógicas.

```
1 exp: left=summ (op=('>' | '<' | '>=' | '<=' | '==' | '!=')
     right=exp)*;
```

2. **mult:** Operações de multiplicação, divisão e módulo.

```
1 mult: left=atom (op=('*' | '/' | '\%') right=mult)*;
```

3. **summ:** Operações de adição e subtração.

```
1 summ: left=mult (op=('+' | '-') right=summ)*;
```

Importante observar que a ordem de precedência dos operadores é definida pelo atributo "*left*", indicando quais operadores tem precedência sobre quais outros. Assim, tem-se *mult* com a maior precedência, seguida de *summ*[1], e tendo *exp* como o de menor precedência[2].

2.1.3 FUNÇÕES

A definição de uma função se dá com o prefixo *func* seguido do nome da função, uma lista de argumentos dentro de dois parênteses, sendo possível tal lista estar vazia, e um conjunto de colchetes, dentro dos quais se encontra, obrigatóriamente, o corpo da função. A sintaxe simplicada é:

func nome(args) { body; }

E sua gramática é:

```
1 function_def : 'func' name=ID '(' args? ')' block;
2 args : ID (',' ID)*;
3 block: '{' expr* '}';
```

Sendo que a chamada de um função pode ocorrer dentro ou fora de uma função, além de permitir o uso de recursão. Para chamar uma função, basta invocar o nome da mesma, passando para ela uma lista de parâmetros, que podem ser variáveis, valores ou outras funções, A sintaxe é:

call_func(args);

E a gramática:

```
1 call : name=ID '(' exprs? ')' ';';
```

[1] graças ao uso de *left=mult*

[2] *left=summ*

2.1.4 *FOR* e *WHILE*

2.1.5 *IF-THEN-ELSE* e *SWITCH CASE*

2.2 CARACTERÍSTICAS ADICIONAIS DA LINGUAGEM

Fora os tipos adicionais, comentados na seção de tipos, a presente linguagem apresenta, como característica adicional, o suporte à comentários, sejam eles de linha única ou de múltiplas linhas, seguindo a sintaxe de comentários da linguagem *Java*.

```
1 COMMENT: '/*' .*? '*/' -> skip;
2 LINE_COMMENT: '//' ~[\r\n]* -> skip;]
```

3 EXEMPLOS DE CÓDIGOS