INE5622 - Introdução a Compiladores

Implementação de Compilador 2 (IC2) - Análise semântica e execução de código

Equipe

Vinicius Pizetta de Souza	16207731
Jackson Dener Wrublak	11201028
José Felipe da Silva	17202643

Papel no Desenvolvimento

Ocorreram 5 encontros com todos os membros participando do desenvolvimento de IC2.

Implementação do compilador

Para o desenvolvimento do projeto, foi utilizada a gramática ANTLR4.
Utilizamos a gramática entregue no IC1 e a complementamos com itens faltantes.
Foi implementado, testado semanticamente os tipos de dados, instâncias, definições e chamadas de funções, estruturas de controle, laços de repetição, operações e suas precedências.

Gramática vjj.g4

```
grammar vjj;

start: func* EOF;

func: DEF name = ID '(' args? ')' statms;
```

```
args: ID (',' ID)*;

statms: '{' statm* '}' | statm;

statm:

   ID '=' expr ';'                                          # assign

   | PRINT expr ';'                                         # print

   | IF cond = expr then = statms (ELSE otherwise = statms)?   # if

   | WHILE cond = expr statms                               # while

   | FOR '(' cond = expr ';' cond = expr ';' expr ')' statms   # for

   | SWITCH '(' atom ')' switchStatms+                      # switch

   | CLASS classdef                                         # class

   | RETURN expr ';'                                        # return;


whileStatement: '{' statm* '}';

switchStatms: '{' CASE atom ':' switchStatms* '}';

types: TYPE_INT | TYPE_FLOAT | TYPE_BOOLEAN | TYPE_NULL;

classdef: CLASS atom ('(' (args)? ')')? ':' statms;

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

exprs: expr (',' expr)*;

expr:

   left = summ (

       op = ('>' | '<' | '<=' | '>=' | '==') right = expr

   )*;

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

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

```
atom:
   '(' expr ')'
   | INT
   | FLOAT
   | NULL
   | BOOLEAN
   | ID
   | INPUT
   | call;
TYPE_INT: 'int';
TYPE_FLOAT: 'float';
TYPE_BOOLEAN: 'bool';
TYPE_NULL: 'null';
DEF: 'def';
RETURN: 'return';
PRINT: 'print';
CASE: 'case';
IF: 'if';
ELSE: 'else';
WHILE: 'while';
FOR: 'for';
SWITCH: 'switch';
OR: 'or';
AND: 'and';
```

```
NOT: 'not';

CLASS: 'class';

INPUT: 'input';

ID: [a-zA-Z]+ [0-9a-zA-Z]*;

INT: [0-9]+;

FLOAT: [0-9]+ '.' [0-9]+;

NULL: 'None';

BOOLEAN: 'True' | 'False';

WS: [ \r\n\t]+ -> skip;
```

Comandos Utilizados

Criar ambiente virtual

virtualenv venv

Ativar ambiente virtual

source venv/bin/activate

Instalar requisitos do ambiente virtual

pip install -r requirements.txt

Compilar em python

python3 -m vjj -h

Testar arquivo .vjj FUNCIONANDO

python3 -m vjj input.vjj

Testar arquivo .vjj COM ERRO

python3 -m vjj zetta.vjj

Melhorias na gramática

Implementamos na gramática .g4, todos os pontos que foram requisitados no IC1:

- Tipagem (INT, FLOAT, BOOLEAN E STRING)
- Uso de classes
- Chamadas de funções python(def)
- Estruturas if-else e switch
- Laços for e while
- Operações matemáticas
- Precedência de operações usando parênteses
- Característica adicional do grupo é o uso do Tipo Null = None

Norma

Seguimos a norma e ele foi escrito em python 3 e roda em sistemas Linux.

Testes

Para testes deixamos 2 arquivos na pasta raiz, o arquivo input.vjj possui o algoritmo fibonacci fornecido, e o arquivo erro.vjj possui um código com erro sintático.