Diego Miranda - 2210996 Felipe Cancella - 2210487

O que foi implementado:

Toda a gramática pedida, com as seguintes alterações, para facilitar a implementação.

- IF-THEN-ELSE/IF-THEN com FIM para indicar término
- varlist "normal" e varlist_m personalizada para as variáveis do monitor
- EVAL com implementação da segunda opção (EVAL cmds VEZES ID FIM)

Gramática final:

```
-> programa$
            -> INICIO varlist MONITOR varlist m EXECUTE cmds TERMINO
programa
varlist
            -> id, varlist | id
varlist_m
            -> id, varlist_m | id
            -> cmd cmds | cmd
cmds
            -> id = expr | ZERO (id)
cmd
cmd
            -> IF id THEN cmds FIM | IF id THEN cmds ELSE cmds FIM
cmd
            -> EVAL cmds VEZES id FIM
            -> ENQUANTO id FACA cmds FIM
cmd
            -> expr + expr | expr * expr | id | NUMBER
expr
obs.:
"+" = PLUS, "*" = TIMES, "," = COMMA, "(" = LPAREN, ")" = RPAREN, "=" = EQUALS
```

Como foi implementado:

Estruturas iniciais usadas para auxiliar na operação do analisador.

```
Lista de tokens: tokens = ('ID', 'NUMBER', 'EQUALS', ...)

Expressões regulares para os tokens: t_EQUALS = r'=', t_PLUS = r'\+', ...

Palavras-chave: keywords = {'INICIO': 'INICIO', 'MONITOR': 'MONITOR', ...}

Tabela de símbolos: symbol_table = {}

Lista de variáveis monitoradas: monitorados = []

Lista com o código C gerado: c_code = []
```

Depois de criar essas estruturas auxiliares, criamos as regras para os identificadores.

```
Regras: t_ID, t_NUMBER, t_newline, t_error
```

E assim montamos o lexer.

Já para o parser, montamos cada regra (sintática) da gramática.

O passo mais importante nas regras, além de identificar cada gramática, é o armazenamento do equivalente ao processo interpretado na lista c_code, que guarda as linhas de código do programa em linguagem C.

Abaixo temos um exemplo da regra para o item expr da gramática.

Exemplo:

Nesse caso, o parser ao identificar a expressão "monta" ela e armazena em p[0]. Posteriormente, essa e outras operações serão armazenadas no vetor de código c pelo programa, na regra p_programa.

O que funciona/não funciona:

Tudo:)

Como executar:

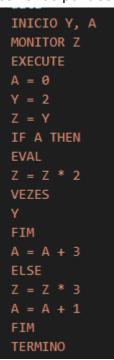
Para executar, basta substituir o texto da variável data na main por uma das opções do arquivo testes_provolone.py. Depois da execução, um arquivo "arquivo_de_saida.c" será gerado, com o código equivalente na linguagem C.

Testes utilizados:

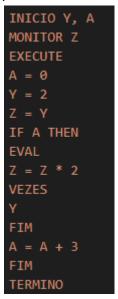
Teste 1: Implementação de monitoramento de variável, EVAL, tendo mais de um comando dentro de um ENQUANTO, usando também operação de soma:

```
INICIO Y, A
MONITOR Z
EXECUTE
A = 1
Y = 2
Z = Y
ENQUANTO A FACA
EVAL
Z = Z + 2
VEZES
Y
FIM
A = 0
FIM
TERMINO
```

Teste 2: Implementação de monitoramento de variável, IF-THEN-ELSE, com mais de um comando por trecho, além de operações de soma e multiplicação:



Teste 3: Implementação de monitoramento de variável, IF-THEN, com mais de um comando por trecho, além de operações de soma e multiplicação:



Teste 4: Implementação de monitoramento de mais de uma variável, IF-THEN-ELSE, com mais de um comando por trecho, além de operações de soma, multiplicação e ZERO:

