Compiladores 2024.1 - Trabalho 2
Entrega: 05/06
Avaliação: 4 pontos
Gerenciamento de escopo e verificação de tipos por meio de tabela de símbolos

Abaixo segue a descrição de uma linguagem fictícia simplificada com único propósito de permitir atribuições de valores e criação de escopo. Portanto, suas operações são limitadas, não servindo como linguagem de propósito geral.

A linguagem tem os seguintes comandos:

- **BLOCO**

```
BLOCO tk_id_bloco
(...)
FIM tk_id_bloco
```

- **DECLARAÇÕES E ATRIBUIÇÕES,** seguem padrão descrito pela gramática:

```
DEC -> TIPO LIST AT
LIST -> AT ,
LIST ->
AT -> ID
AT -> ID = CONST
AT -> ID = ID
ID -> tk_identificador
CONST -> tk_numero
CONST -> tk_cadeia
TIPO -> NUMERO
TIPO -> CADEIA
```

- **PRINT**

```
PRINT tk_idenficiador
```

- **Tokens:**
  - **Identificadores (tk_identificador) –** nomes para atributos são descritos iniciados por uma letra, e pode conter outras letras, dígitos ou _.
  - **Identificadores de bloco (tk_id_bloco) –** nomes para blocos são iniciados e terminados por _, possui 1 ou mais letras e dígitos.
  - **Tipos de dados:**
    - **numero (tk_numero) –** números inteiros ou reais (ex: 10, 10.0, +10, -1.345). Valor padrão é 0, se não atribuído nenhum.
    - **cadeia (tk_cadeia) –** sequência de caracteres entre aspas duplas (ex: "", "cadeia", " * - * "). valor padrão é cadeia vazia "", se não atribuído nenhum

- **Outros:**
  - **Indentação não é obrigatória,**
  - **A linguagem é fortemente tipada, NÃO permite o uso de variáveis sem declaração prévia ou faz conversão de tipos**.

Exemplo de um programa:

```
BLOCO _principal_
        NUMERO a = 10, b = 20
        CADEIA x

        PRINT b
        PRINT a
        x= “Ola mundo”
        x=a
        PRINT x
        BLOCO _n1_
                CADEIA a = “Compiladores”
                NUMERO c
                c=-0.45
                PRINT b
                PRINT c
        FIM _n1_

        BLOCO _n2_
                CADEIA b = “Compiladores”
                PRINT a
                PRINT b
                a=11
                CADEIA a= “Bloco2”
                PRINT a
                PRINT c
                BLOCO _n3_
                        NUMERO a=-0.28, c
                        PRINT a
                        PRINT b
                        PRINT c
```

```
                    c=a
                    PRINT c
                    a=40
                    PRINT a
                    print c
                FIM _n3_
        FIM _n2_
        PRINT c
        PRINT a
FIM _principal_
```

Saída esperada:

```
20
10
Erro linha 8, tipos não compatíveis
“Ola mundo”
20
-0.45
10
“Compiladores
“Bloco2”
Erro linha  25 - Variável não declarada
-0.28
“Compiladores”
0
-0.28
40
-0.28
Erro linha  38 - Variável não declarada
11
```

Sua tarefa:

Você deverá implementar um programa, em qualquer linguagem, para gerenciar o escopo dessa linguagem, acusar erro quando existir e mostrar os valores de cada PRINT.

Em um fluxo completo (front-end) de um compilador, os passos seriam os seguintes:

1. Análise léxica – Reconhecer tokens da linguagem
2. Analisar sintático – Construção de tabela de derivação a partir da gramática e tokens retornados pelo léxico
3. Análise semântica – Verificação de escopo

Para simplificação do trabalho, **não precisaremos executar a análise léxica e sintática**. Vamos considerar que todo programa é sempre **bem formatado lexicalmente e sintaticamente**.

No entanto, você deverá implementar a **análise semântica para gerenciamento de escopo e verificação de tipos.**

Você deverá:

- implementar uma tabela de símbolos e atualizá-la a cada abertura e fechamento de escopo, bem como, a cada declaração e atribuição de valores.

  A tabela de símbolos deve ter no mínimo os seguintes atributos: token (categoria), lexema, tipo de dado, valor.

- mostrar o valor de todos os prints

- Informar quando houver algum erro semântico em relação ao escopo ou verificação de tipos. Possíveis erros:
  - Tipos não compatíveis – ex, uma atribuição apenas pode ser realizada com elementos do mesmo tipo
  - Variável não declarada – ex, um print de variável que não existe naquele escopo.
  - **Ao identificar erros semânticos, informar e prosseguir o processamento. Não deve parar o programa.**

- Você deverá escolher um dentre os seguintes métodos para gestão de escopo: op1, única tabela de símbolos, onde cada lexema tem sua pilha de escopos; op2: pilha de escopos, onde cada posição da pilha guarda uma tabela de símbolos
- Entrega: relatório com código comentado (ou link para código); Entradas e saídas para os exemplos fornecidos; Explicação de qual método utilizou para gerenciar escopo; Um pseudo código (algoritmo) em alto nível de como seu programa funciona e gerencia o escopo e verificação de tipos.