Trabalho de Compiladores I

Lucas Antônio Nogueira Silva - 0049974

Introdução

Neste trabalho 2 foi realizado a continuação do trabalho 1 de compiladores, que foi realizado a criação do semântico e da tabela de símbolos.

Tabela de símbolos

```
class Tabela:
    def __init__(self):
        self.tabela = {}

    def existeIdent(self, nome):
        if nome in self.tabela:
            return True
        else:
            return False

    def declaraIdent(self, nome, valor):
        if not self.existeIdent(nome):
            self.tabela[nome] = valor
            return True
        else:
            return False

    def pegaValor(self, nome):
        return self.tabela[nome]

    def atribuiValor(self, nome, valor):
        self.tabela[nome] = valor
```

Em resumo, a classe Tabela fornece funcionalidades para verificar a existência de um identificador na tabela, declarar novos identificadores, obter e atribuir valores aos identificadores existentes. Ela mantém internamente um dicionário chamado tabela para armazenar os identificadores e seus valores.

Atualização do código

Algumas atualizações foram necessárias no código do arquivo sintatico.py para a execução do trabalho 2. Logo abaixo explicarei as atualizações necessárias.

```
def testaVarNaoDeclarada(self, var, linha):
    if self.deuErro:
        return
    if not self.tabSimb.existeIdent(var):
        self.deuErro = True
        msg = "Variavel " + var + " nao declarada."
        self.semantico.erroSemantico(msg, linha)
        quit()
```

Em resumo, esse trecho de código verifica se uma variável não declarada é usada. Se a variável não existe na tabela de símbolos, ocorre um erro semântico, e a execução do programa é interrompida.

```
if type(ident) == list:
    for i in ident:
        if i in self.tabSimb.tabela:
            print('Essa variavel ja foi declarada.')
            quit()

        self.tabSimb.declaraIdent(i, tipo)

else:
    if ident in self.tabSimb.tabela:
        print('Essa variavel ja foi declarada.')
        quit()

    self.tabSimb.declaraIdent(ident, tipo)
```

Em resumo, esse trecho de código lida com a declaração de identificadores em um programa. Se o identificador ou a lista de identificadores já estiverem presentes na tabela de símbolos, uma mensagem de erro é impressa e a execução é encerrada. Caso contrário, os identificadores são declarados na tabela de símbolos com seus respectivos tipos.

```
if aux != None:
    listAux = []
    listAux.append(ident)
    listAux.extend(aux)

    return listAux

return ident
```

Em resumo, esse trecho de código lida com a concatenação de uma lista auxiliar aux com um valor identificador ident. Se a variável aux não for None, uma nova lista é criada contendo o valor da variável ident seguido pelos elementos da lista aux. Caso contrário, se a variável aux for None, o valor da variável ident é retornado diretamente.

```
aux = self.LIST_ID()
if (not self.tabSimb.existeIdent(aux)):
    self.semantico.erroSemantico(f'Variável {aux} não declarada', self.tokenAtual.linha)
    quit()
```

Em resumo, esse trecho de código verifica se o identificador ou identificadores contidos em **aux** não estão declarados na tabela de símbolos. Se algum dos identificadores não estiver declarado, ocorre um erro semântico e a execução do programa é interrompida.