Descrição do Trabalho Prático 04 de Compiladores

Eduardo Miranda

Abril 2024

1 Introdução

O trabalho prático (TP) consiste na implementação de uma **Máquina Virtual (Interpretador)** para uma linguagem intermediária que será definida a seguir. Tal interpretador deve ser criado na linguagem Python assim como as etapas anteriores.

2 Proposta de TP

Conforme mencionado, o TP proposto consiste em implementar um interpretador para uma linguagem intermediária.

2.1 Sobre o Interpretador

O programa resultante deve possibilitar que o usuário informe um arquivo escrito na linguagem Python que retorne uma lista de tuplas. Estas tuplas definirão como o programa deve ser executado no Interpretador.

2.1.1 Possíveis tipos de Tokens

O interpretador deve ser capaz de processar as seguintes tuplas:
Operadores aritméticos:

- Adição: ("+","guardar","operando1","operando2");
- Subtração: ("-","guardar","operando1","operando2");
- Multiplicação: ("*","guardar","operando1","operando2");
- Divisão Real: ("/","guardar","operando1","operando2");
- Módulo: ("%","guardar","operando1","operando2");
- Divisão Inteira: ("//","guardar","operando1","operando2");;

- Uno adição: ("+","guardar","operando1",None);
- Uno subtração: ("-","guardar","operando1",None);

As tuplas de operações aritméticas comuns devem processar como esperado, realizando a soma, subtração, multiplicação e afins dos operandos 1 e 2 e salvando o resultado na variavel "guardar".

Os operadores unários de adição e subtração devem estar antes do operando1 para "negativação" do operando caso seja a operação de uno subtração.

Ex: guardar = -5; //Operação com somente um operando

Operadores lógicos, relacionais e atribuição:

- OU lógico: ("||","guardar","operando1","operando2");
- E lógico: ("&&","guardar","operando1","operando2");
- NÃO lógico: ("!","guardar","operando1",None);
- Igualdade: ("==","guardar","operando1","operando2");
- Diferença: ("<>","guardar","operando1","operando2");
- Maior que: (">","guardar","operando1","operando2");>;
- Maior igual que: ("≥","guardar","operando1","operando2");
- Menor que: ("<","guardar","operando1","operando2");
- Menor igual que: ("≤","guardar","operando1","operando2");
- Atribuição: ("=","guardar","operando1",None);

Nos casos de operadores logicos e relacionais a lógica permanece a mesma para os operadores aritméticos.

Exceto para o operador unário NÃO LÓGICO que a lógica é a mesma para os operadores unários de adição e subtração.

No operador de atribuição deve ser guardado o valor do "operando1" na "variável" guardar.

Funções de Alteração de Fluxo:

- IF: ("IF","CONDICAO","LABEL1","LABEL2");
- JUMP: ("JUMP","LABEL",None,None);

Para a função de alteração de fluxo **IF**, deve-se mudar o ponteiro de leitura da posição atual para a posição em que o LABEL1 está caso a variável "CONDICAO" seja verdadeira, caso contrário deve-se mudar o ponteiro de leitura para a posição em que o "LABEL2" está.

Já na função de alteração de fluxo **JUMP** o ponteiro de leitura deve-se mudar para a posição em que o label está indepentemente de quaisquer condições.

Chamadas de Sistema:

- PRINT: ("CALL","PRINT","VALOR",None);
- SCAN: ("CALL","SCAN","VARIAVEL",None);

Para a chamada de sistema print o conteúdo de "VALOR" deve ser printado na tela.

Já para a chamada scan deve ser feita a leitura do teclado e o conteúdo desta leitura deve ser salvo na variável "VALOR".

Labels:

- Label: ("LABEL","NOMELABEL",None,None);

Os labels indicam pontos em que o ponteiro de execução do programa pode ser alterado. Os mesmos devem ser salvos em uma lista antes de começar a interpretação do código efetivamente.

3 Algumas Dicas

Durante o desenvolvimento, é importante não se perder nos detalhes. Portanto, é recomendado que a implementação inicie somente pelas funcionalidades básicas. Só depois de garantir que as funcionalidades básicas estão funcionando conforme planejado, deve-se considerar a implementação de melhoramentos e funcionalidades adicionais. Também recomenda-se que trechos mais complicados do código sejam acompanhados de comentários que esclareçam o seu funcionamento/objetivo/parâmetros de entrada e resultados.

O TP também será avaliado em termos de:

- Modularização;
- Legibilidade (nomes de variáveis significativos, código bem formatado, uso de comentários);
- Consistência (formatação uniforme);
- Otimização do código;

4 Funcionalidades

As seguintes funcionalidades devem estar funcionando corretamente ao entregar o trabalho prático:

4.1 Funcionalidade 1

A partir de uma lista de tuplas ao executar o interpretador para algum arquivo escrito na linguagem python arbitrário a execução deste código intermediário seja efetuada com sucesso.

Execução esta que se inicia na primeira tupla da lista e segue para a tupla subsequente até chegarmos na última tupla e a programação deve ser encerrada, exceto em casos de Funções de Alteração de Fluxo nas quais o ponteiro de execução do programa deve 'saltar' para a posição indicada pelo label ao invés de somente fazer o simples incremento.

4.2 Funcionalidade 2

Casos como os de acesso a valores de labels e operandos que não existem ainda devem ser tratados como erros.

Todavia o acesso de variáveis que ainda não tenham sido instanciadas ao se 'salvar' uma primeira vez seja em qualquer operação ou atribuição esta variável deve ser criada.

5 Submissão

Além de serem submetidos via link no Portal Didático da disciplina, o TP também deve ser enviado para o endereço eletrônico `eduardomiranda@cefetmg.br` , tendo como assunto TP_Compiladores_2024_1. No corpo do email deve aparecer o nome completo de cada integrante do grupo e um arquivo .zip com o programa e o manual de utilização.