4a Tarefa - Interpretação

Você já criou três módulos, um para realizar a ANÁLISE LÉXICA, outro para realizar a ANÁLISE SINTÁTICA e outro para realizar a ANÁLISE SEMÂNTICA. O módulo que realiza análise semântica recebe a AST e preenche as estruturas de dados (dicionários) com todas as informações necessárias para proceder a interpretação.

Um possível código para a TRADUÇÃO, que realiza a interpretação, desconsiderando tratamentos de erro e detalhes menos didáticos, é o exemplo abaixo. Considere para efeito de ilustração.

……. arquivo traducao.py …….

```
from lexico.py import Lexico,
from sintatico.py import Sintatico
from semantico.py import Semantico
from interpretador.py import Interpretador

lex = Lexico( 'fonte.bob' )
parser = Sintatico( lex )
ast = parser.analisa()

semantico = Semantico( ast )
semantico.analisa( )     # constrói e alimenta todas as EDs necessárias para a interpretação

tradutor = Interpretador( semantico )
tradutor.interpreta()
```

Parte Zero (sem isso você não fará nada!)

Objetivo: Criar uma ED para gerenciar os ambientes de chamada e de execução.

- Cada ambiente é representado por um dicionário que mantém as informações relevantes sobre os identificadores conhecidos naquele ambiente específico.

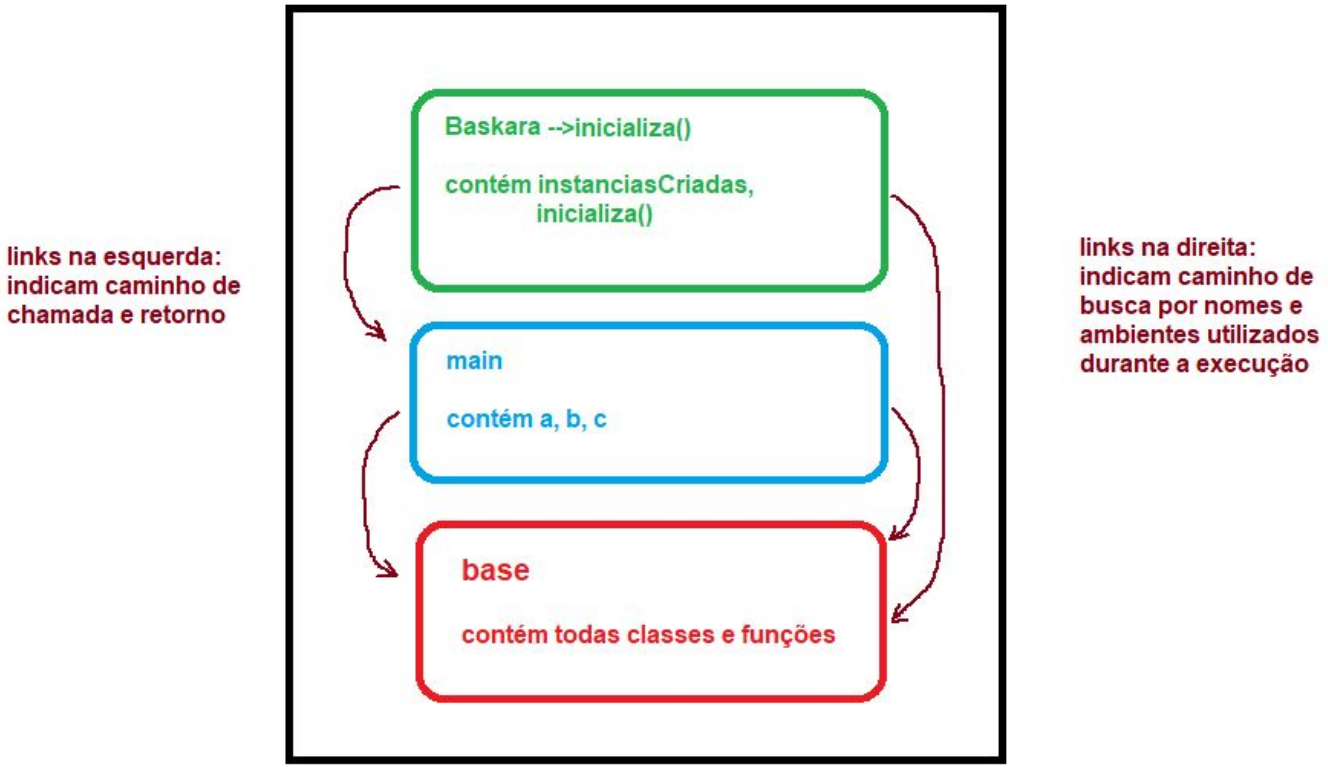
Por exemplo, se a variável X temporária foi declarada na função main, o dicionário que representa o ambiente dessa função deve ter uma entrada ambiente['X'] que retorna todas as amarrações para o nome X. Segue um exemplo concreto para ilustrar:

```
ambiente[ 'x' ] = [ VAR, INT, 34 ]       # variavel simples: tipo e valor
ambiente[ 'vetor' ] = [ ARRAY, 10 ]   # variavel composta: tamanho. Trate um array como uma lista de
                                      # tamanho fixo, podendo armazenar tipos heterogêneos
```

```
ambiente[ ‘print’ ] = [ BUILT_IN ]   # função especial embutida que é reconhecida e tratada pelo
                                     # interpretador (se vire!)
ambiente[ ‘scanf’ ] = [ BUILT_IN ]  # função especial embutida que é reconhecida e tratada pelo
                                     # interpretador (se vire!)
```

- A ED que gerencia os ambientes funciona num esquema de pilha. Cada chamada de função cria um novo ambiente de execução que sobrepõe (esconde) o ambiente do chamador (mesmo mecanismo utilizado para gerenciar dados globais e locais).
- O ambiente da função corrente vai estar no topo da pilha. Quando a execução de uma função finaliza (retorna), o ambiente da função é desempilhado e descartado.

Considere o exemplo do código test_bob.txt postado na 2a tarefa e o esquema da figura apresentada abaixo:



objeto que gerencia os ambientes

O ambiente base contém registro de todas as classes e funções (sem classe) criadas no programa. É por isso que, durante a execução da função main, todos os nomes de classe e função são reconhecidos. Quando o interpretador não encontra o nome no ambiente local, continua a busca nos ambientes linkados. Os links de ambiente de declaração (escopo estático) na direita reconhecem:

- para função livre (sem classe): ambiente local → ambiente base
- para método (função de classe): ambiente local → ambiente de instância → ambiente base
- para método static (função de classe): ambiente local → ambiente de classe → ambiente base
- atenção --- ambientes de instância e de classe consideram a hierarquia de classes

Os links da esquerda marcam os pontos de retorno das chamadas.

Parte1

Objetivo: Preparar os ambientes para iniciar a interpretação e executar a função main()

- Inicie a interpretação, depois do módulo semântico preparar tudo, chamando o método interpreta() que vai realizar os eventos que seguem:
- Prepare o ambiente base.
- Consulte o dicionário de funções, busque a função main()
- Prepare o ambiente da função main()
- Mande executar o corpo da função main() no ambiente preparado

```
tradutor = Interpretador( semantico )
tradutor.interpreta()
```

Parte2

Objetivo: Criar o método executa(ast)

Para interpretar os comandos armazenados na AST, crie um método que recebe como parâmetro a subárvore AST a ser executada e interpreta a mesma.

Parte3

Objetivo: Dar um jeito em tudo e fazer funcionar

Se vire nos 30, não tem jeito de explicar tudo. A maioria das coisas que você precisa é bom senso e criatividade de cientista da computação.

Bom trabalho.