Lista de Exercícios 3: Resolução

Guilherme de Abreu (nUSP: 12543033)
Hélio Cardoso (nUSP: 10310227)
Laura Camargos (nUSP: 13692334)
Sandy Dutra (nUSP: 12544570)
Theo dos Santos (nUSP: 10691331)

28 de março de 2025

Exercício 1:

Considere o seguinte programa escrito em LALG:

```
program p1;
var x: integer;
begin
      read(x);
      x := x * 2;
      write(x);
end.
```

Em um processo de compilação, qual seria a saída da

a. análise léxica?

b. análise sintática?

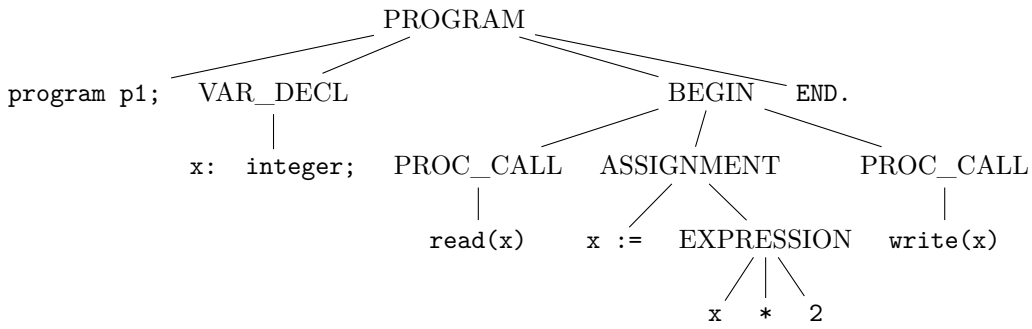
c. análise semântica?

Resolução:

a. uma sequência de palavras (tokens) associadas a lexemas reconhecidos pela linguagem. Ex.:

```
<program, keyword> <p1, id>
<var, keyword> <x, id2> <:, symbol>
<integer, keyword><;, symbol>
<begin, keyword>
<read, keyword> <(, symbol> <x, id2> <), symbol>
<;, symbol>
<x, id2> <:=, operator> <x, id2> <*, operator> <2, number>
<;, symbol>
<write, keyword> <(, symbol> <x, id2> <), symbol>
<;, symbol>
<end., keyword> <., symbol>
```

b. uma árvore sintática abstrata (AST), uma estrutura de dados que organiza os tokens hierarquicamente, segundo as normas gramaticais da linguagem. Uma possível representação em gráfica simplificada desta seria:



c. finalmente, a análise semântica é o estágio que verifica se o código-fonte é significativo para além da sua sintaxe. Isto é, para além de estar bem formatado ele se traduz em instruções válidas. Isto inclui:

- Verificação de tipos
- Verificação de escopo
- Validação de chamadas de função/procedimento
- A construção de uma tabela de símbolos que sumariza as etapas anteriores

Um exemplo de tabela de símbolos para presente código seria:

Identifier	**Type**
x	integer

Não apresentando falhas nestas sucessivas etapas, o código-fonte é habilitado a ser compilado

Exercício 2:

Defina os seguintes correlatos ao processo de compilação:

a. Interpretador

b. Processador de macro

c. Editor de ligação

d. Montador (Assembler)

e. Pré-processador

f. Editor/IDE

g. Depurador

Resolução:

a. **Interpretador:** programa o qual executa um dado código-fonte à partir de sua leitura, linha a linha, durante o tempo de execução.

b. **Processador de macro:** ferramenta que processa substituições de texto em um dado código-fonte antes de sua compilação interpretação. Isto é, este reconhece dadas invocações de macro (como o `define` na linguagem C) e a substituem por definições a estas atribuídas.

c. **Editor de ligação (Linker):** programa o qual combina uma série de arquivos (denominados "objetos") gerados pelo compilador em um único arquivo executável, ao resolver as referências entre estes.

d. **Montador (Assembler):** programa o qual provê tradução entre uma linguagem de programação de baixo nível denominada "linguagem de montagem"e correspondentes instruções de máquina. Como a relação entre instruções de montagem e instruções de máquina se aproximam de uma correspondência de 1 para 1, tal linguagem permite ao programador ganhos em controle e desempenho em troca de perdas em termos de menor portabilidade e maior complexidade.

e. **Pré-processador:** programa o qual opera transformações no código-fonte antes deste ser processado pelo compilador. Este abarca a inclusão de arquivos, a ativação do processador de macros, a compilação condicional, dentre outras várias funcionalidades.

f. **Editor/IDE:** programa o qual provê um conjunto de ferramentas em uma interface unificada para permitir ao usuário um ambiente de desenvolvimento eficiente para dentre outras funcionalidades escrever, testar e depurar outros programas à partir do código-fonte destes.

g. **Depurador:** programa o qual permite o teste, análise e depuração de um dado programa alvo por meio do controle de execução, inspeção de variáveis e rastreio de erros no código-fonte deste. Ou seja, depuradores são ferramentas que auxiliam programadores a encontrar e reparar falhas em dados programas.

Exercício 3:

Quais as principais similaridades e diferenças entre interpretadores e compiladores? Qual método é mais vantajoso?

Resolução:

São similaridades entre compiladores e interpretadores:

- **Propósito:** tanto compiladores quanto interpretadores são ferramentas pelas quais um dado código-fonte pode ser traduzido a uma forma que pode ser executada por um computador.
- **Detecção de erro:** Ambos realizam tarefas de análise léxica, sintática e semântica para assegurar que o código-fonte é válido. E podem orientar desenvolvedores a identificar e corrigir falhas antes da execução deste.

Enquanto as principais divergências entre interpretadores e compiladores encontram-se descritas na tabela 1.

Em suma, as características de linguagens interpretadas as tornam propícias de serem utilizadas em ambientes de desenvolvimento dinâmicos como, por exemplo, na prototipagem, na análise de dados, ou na escrita de scripts. Ou seja, situações as quais o custo adicional de sucessivos processos de compilação não se justifica. Enquanto, por outro lado linguagens compiladas são preferíveis quando é crítico o melhor desempenho e eficiência dos programas gerados.

Tabela 1: Principais divergências entre interpretadores e compiladores

Aspecto	**Interpretador**	**Compilador**
Execução	Executa o código-fonte traduzido em linguagem de máquina, linha a linha, em tempo de execução	Executa uma tradução do código-fonte feita previamente, em sua inteiridade.
Saída	Não gera arquivo executável, o código é executado diretamente	Produz arquivo executável.
Tempo de execução	Maior, pois o código é traduzido e executado simultaneamente.	Menor, a execução é feita diretamente.
Consumo de memória	Maior, pois o interpretador precisa ser carregado a memória durante execução.	Menor, o código é executado de forma independente.
Tratamento de erros	Erros são todos detectados e relatados durante execução	Alguns erros são detectados e relatados durante compilação
Portabilidade	Maior, dado código-fonte pode ser executado em qualquer plataforma alvo que possua um interpretador.	Menor, o código-fonte necessita ser compilado para gerar um arquivo executável cada plataforma alvo.
Depuração	Mais ágil pois se tem uma resposta imediata a mudanças feitas no código.	Mais lenta pois cada modificação exige recompilação.

Exercício 4:

Quais as características de uma linguagem que determinam que ela deve ser compilada ou interpretada? Esta questão refere-se à linguagem em si, independentemente do uso que é feito dela.

Resolução:

Não há elementos na linguagem que conclusivamente apontem se esta a de ser interpretada ou compilada, tido que estas são características da implementação da linguagem Não obstante, há características que fortemente sugerem a linguagem ser utilizada para um fim e não outro. Por exemplo, linguagens dinamicamente tipadas tendem a ser interpretadas, pois esta é uma facilidade que pode ser resolvida em tempo de execução.