1º LISTA DE EXERCÍCIOS DE LINGUAGENS DE PROGRAMAÇÃO

Victor Barbosa Rocha

1) conceitue e descreva as diferenças entre (i) linguagem de alto-nível e (ii) linguagem de baixo-nível. Descreva o papel e a importância do compilador no processo de criação de programas de computador.

A linguagem de baixo nível é aquela mais próxima da linguagem da máquina, tipicamente elas requerem um maior tempo de desenvolvimento dos seus códigos, mas o arquivo executável é rápido, exemplo Assembly e Cobol. Por outro lado, temos as linguagens de alto nível, que são mais próximas da linguagem humana, o que facilita a leitura, aprendizado e tempo de desenvolvimento do código, contudo o arquivo executável é mais lento, por exemplo Python e Perl.

O compilador é um programa que serve para traduzir um código fonte de uma linguagem de alto nível para uma linguagem de baixo nível que seja entendida pelo computador. O compilador é importante, entre outras coisas, para facilitar e tornar mais acessível o desenvolvimento de programas, pois junto com as linguagens de alto nível, abriu margem para o crescimento do número de programadores, já que simplifica e escreve para a arquitetura utilizada.

5) descreva as seguintes categorias de linguagens de programação e apresente o nome de duas linguagens de programação com seus respectivos exemplos.

IMPERATIVAS: as linguagens imperativas são aquelas que literalmente mandam o computador realizar tais ações e em determinada ordem, assim mudando o estado das variáveis. São exemplos, Assembly e C.

FUNCIONAIS: as funcionais claramente trabalham com funções matemáticas, evitando assim dados mutáveis. Não são tanto usadas na indústria, mas bastante na parte acadêmica. Exemplo, Haskell e Ocaml.

LÓGICAS: essas linguagens se baseiam na lógica matemática formal. Os programas desse tipo de linguagens consistem em frases que expressão fatos e regras sobre algum problema. Dentre os representantes temos: Prolog e a Planner (que foi a primeira).

MARCAÇÃO/HIBRIDA: são conjuntos de códigos aplicados a um texto ou a dados, com o fim de adicionar informação particulares sobre esse texto ou dado. Tipo JSTL e XSLT.

6) escreva uma gramática no formato BNF e não ambígua para expressões envolvendo as variáveis A, B, C e os operadores \* (multiplicação) e ^ (exponenciação). A precedência deve seguir as regras usuais da matemática.

<program> ::= **begin** <stmt\_list> **end**

<stmt\_list> ::= <stmt> | <stmt> <stmt\_list>

<stmt> ::= <var> = <digit>

<stmt> ::= <var> = <expression>

<var> ::= ‘A’|’B’|’C’

<digit> ::= ‘0’ .. ‘9’

<operador> ::= ‘\*’| ’^’

<expression> ::= <var><operador><expression>| <var> <operador><factor> | <factor>| <var>

<factor> ::= <var> ^ <var>

7) descreva o que é um paradigma de programação.

Paradigma de programação é uma forma de classificar linguagens de programação baseadas em suas características, por exemplo, o modelo de execução e até organização do código. Os paradigmas mais comuns são Imperativos, Funcionais, Declarativos, Orientados a objetos, procedural, lógico ou simbólico.

9) defina análise semântica e descreva: Semântica Operacional; Semântica Axiomática; e Semântica Denotacional.

Análise semântica é a descrição do significado das expressões, das instruções e das unidades de programa. Simplesmente, essa análise é importante para que os programadores saibam o que determinados comandos fazem.

A semântica Operacional é mais para saber sobre como o programa é executado e quais as operações são realizadas, do que somente os resultados dessa computação.

A semântica Axiomática é para saber quais as proposições lógicas ou asserções são válidas para o programa.

A semântica Denotacional é para saber o que o programa significa e que objetos matemáticos ele denota.

10) utilizando a semântica axiomática apresente a pré-condição para os seguintes programas.

PROGRAMA A: {P} = {y > 2,3}

x = 122 \* y - 144

{x > 144}

122 \* y – 144 > 144

122\*y > 288

Y > 288/122

Y > 2,3

PROGRAMA B: {P} = {x < 9}

**y = 5 \* x - 5**

**x = y + 5**

X = 5x – 5 + 5

X = 5x

**{x < 45}**

**5x < 45**

**X<9**

PROGRAMA C: {P} = {y > 4}

if(x < 200) y = y + 2;

else y = y – 2;

{y > 2}

Y + 2> 2

Y > 0

Y – 2 > 2

**Y > 4**

PROGRAMA D: {P} = {i <= N}

No caso da condição não atender, a única pré-condição válida é que i seja igual a N. Se a condição atender uma vez, i tem que ser igual a N-1. Para a condição ser atendida mais de uma vez, i tem que ser menor que N-1. Se i tem que ser < ou = a N, a pré-condição tem que ser {i<=N}.

while i < N do

i = i + 1

end

{i == N}

PROGRAMA E: {P} = Sorry...