1º Lista

Aluno: Angelo Ferro Curso: Ciências da Computação

Professor: Herbert Rocha Disciplina: Linguagens De Programação

1º)

Linguagem de alto nível está muito mais próxima do programador do que do dispositivo, ou seja, é uma linguagem muito mais intuitiva. Já a linguagem de baixo nível é muito mais voltada ao dispositivo tipo o processador. Normalmente envolve números e letras que nada mais são que instruções diretas ao dispositivo.

Um compilador é um programa de sistema que traduz um programa descrito em uma linguagem de alto nível para um programa equivalente em código de máquina para um processador. Para desempenhar essa tarefa, o compilador deve executar dois tipos de atividades. A primeira atividade é a **análise** do código fonte, onde a estrutura e significado do programa de alto nível são reconhecidos. A segunda atividade é a **síntese** do programa equivalente em linguagem simbólica.

2º)

```
#include <stdio.h>

#include <math.h>


int main()

{

   int i,n;

   float soma=1;


   scanf("%d",&n);


   for (i=1; i<=n; i++)

   {

      if (i%2)

         soma=soma*(float)pow(2*i-1,2*i-1)/pow(2*i,2*i);

      else

         soma=soma+(float)pow(2*i-1,2*i-1)/pow(2*i,2*i);
```

```
    }


    printf("%f\n",soma);
    return 0;
}
```

3º)

4º)

5º)

A: As linguagens imperativas são orientadas a ações, onde a computação é vista como uma sequência de instruções que manipulam valores de variáveis. Exemplos: C e Pascal

B: As linguagens funcionais baseiam-se no conceito de função. Pode-se pensar na programação funcional como simplesmente avaliação de expressões. Seu funcionamento: O programador define uma função para resolver um problema e passa essa função para o computador, Uma função pode envolver várias outras funções em sua definição. O Computador funciona então como uma calculadora que avalia expressões escritas pelo programador através de simplificações até chegar a uma forma normal. Exemplos: Haskell e Erlang

C: As linguagens lógicas consistem na definição de relações lógicas que devem ser satisfeitas pela solução procurada. A busca por uma solução acontece automaticamente através de regras de inferência. Exemplos: PROLOG

6º)

<expr> ➔ <expo> * <expr> | <expo>

<expo> ➔ <base> ^ <expo> | <base>

<base> ➔ A | B | C

7º)

Em programa imperativo tal mapeamento é indireto, **via estados** explicitamente modelados por variáveis globais e locais do programa em execução.

No funcional, o modelo computacional é o de uma **função** que, portanto, mapeia entradas em saídas de forma determinística.

8º)

Análise léxica é o processo de analisar a entrada de linhas de caracteres (tal como o código-fonte de um programa de computador) e produzir uma seqüência de símbolos chamado "símbolos léxicos" (lexical tokens), ou somente "símbolos" (tokens), que podem ser manipulados mais facilmente por um parser (leitor de saída). O componente do compilador responsável pela execução desse processo é conhecido como Analisador léxico.

O analisador léxico, ou scanner como também é chamado, faz a varredura do programa fonte caractere por caractere e, traduz em uma seqüência de símbolos léxicos ou tokens. É nessa fase que são reconhecidas as palavras reservadas, constantes, identificadores e outras palavras que pertencem a linguagem de programação. O analisador léxico executas outras tarefas como por exemplo o tratamento de espaços, eliminação de comentários, contagem do número de linhas que o programa possui e etc.

PRIMEIRA 9º)

SEGUNDA 9º)

Chama-se análise semântica a terceira fase da compilação, onde se verificam os erros de semântica no código fonte e se faz a coleta das informações necessárias para a próxima fase da compilação, a geração de código objeto. O objetivo da análise semântica é trabalhar no nível de inter-relacionamento entre partes distintas do programa.

Semântica Operacional descreve o significado de um programa pela execução de suas instruções em uma máquina, seja ela real ou simulada as mudanças no estado da máquina (memória, registradores, etc.) definem o significado da instrução.

Semântica Axiomática é Baseada em lógica (cálculo de predicados) seu Propósito original é a verificação formal de programas. Axiomas ou regras de inferência são definidas para cada tipo de instrução na linguagem (para permitir transformações de expressões para outras expressões) as expressões são chamadas de asserções (predicados).

Semântica Denotacional é Baseado na teoria de funções recursivas tem o método mais abstrato de descrição da semântica foi originalmente desenvolvido Scott e Strachey (1970). A ideia baseia-se no fato de que há maneiras rigorosas de manipular objetos matemáticos, mas não construções de linguagens de programação.

10º)

PROGRAMA A:

X = 122 * Y – 144 {X > 144}

RESPOSTA:

122 * Y – 144 > 144

122 * Y > 144 + 144

Y > 288 / 122

Y > 2,36

PROGRAMA B:

Y = 5 * X – 5

X = Y + 5 {X < 45}

RESPOSTA:

Y + 5 < 45

Y < 40

5 * X – 5 < 40

X < (40 + 5) / 5

X < 9

PROGRAMA C:

```
if (X < 200)
        Y = Y + 2;
else
        Y = Y – 2;
```

{Y > 2}

RESPOSTA:

if:

Y +2 > 2

Y > 2 – 2

Y > 0

Else:

Y – 2 > 2

Y > 2 + 2

Y > 4

11 º)

Linguagem S

<programa> ::= sieg <conteudo_programa> hart

<nome do programa> ::= programa <'a'..'z'|A-Z|0-9|>

<declaração> ::= int|re|pala|a-z|A-Z|0-9|

<comandos> ::= <se>|<escreva>|<leia>|<enquanto>|<para>|/e

<se> ::=se(condição)entao{<resultado>}senao{<resultado_alternativo>}

<escreva> ::= escreva("a-z|A-Z|0-9")!

<leia> ::= leia("a-z|A-Z|0-9")

<enquanto> ::= enquanto(condição)faca{<resultado>}

<para> ::= para(condição)faca{<resultado>}

<quebra_linha> ::= /e