1ª Lista de exercícios de Paradigmas de Programação
Prof. Glauber Cintra – Entrega: 28/jun/2012

Grupo: Cristiane Oliveira, Lucas Diego, Marcelo Silveira e Ulysses Alessandro.

Essa lista deve ser feita por grupos de no **mínimo** 3 e no **máximo** 4 alunos. Cada questão vale 0,7 pontos.

1. **Discorra sobre algumas razões para o estudo de conceitos de linguagens de programação.**

Maior capacidade de utilizar as linguagens de programação.
Maior facilidade de aprender mais linguagens de programação.
Maior facilidade de projetar novas linguagens de programação.
Maior capacidade de escolher a linguagem apropriada para a aplicação a qual se queira desenvolver.
Avanço global da computação.

2. **Dê um exemplo de dois critérios de projeto de linguagens que estão em conflito direto um com o outro.**

A legibilidade, que está relacionada a facilidade de compreensão do código, e a capacidade de escrita, que é a capacidade de expressar computações complexas de forma sucinta, são critérios conflitantes, pois quanto maior a legibilidade, menor a capacidade de escrita e vice-versa.

3. **Explique o que é o *gargalo de Von Neumann*.**

Este é um problema decorrente da arquitetura de computadores proposta por Von Neumann. Advém primordialmente da separação da CPU e da memória e o compartilhamento de um único barramento para comunicação. Como geralmente a velocidade de processamento da CPU é muito superior a velocidade de leitura/escrita da memória, temos que a CPU é forçada a esperar pelos dados na memória principal. Essa espera, perda de eficiência, é chamada de gargalo de Von Neumann.

4. **Diferencie o processo de compilação do processo de interpretação. Explique o que é linkedição.**

A diferença entre os processos consiste na tradução do código fonte. No processo de compilação, ao ser realizado com êxito após suas diversas fases tais como a criação de códigos objeto e a agregação de referências externas ao código, o mesmo gera um arquivo contendo os códigos executáveis em linguagem de máquina, que são carregados para que ocorra a execução da aplicação. No processo de interpretação, não existe arquivo a ser criado em decorrência do processo de tradução. Na interpretação o código é traduzido e executado interativamente instrução a instrução.

A linkedição é um dos processos que ocorre durante a compilação após a criação do código objeto, onde o compilador junta o código de bibliotecas referênciadas (links) com o código objeto, para posteriormente acontecer a geração de código executável.

5. **Qual linguagem usou *ortogonalidade* como principal critério de projeto?**

O ALGOL-68, que inovou para época ao trazer essa característica, pois ela permite a combinação de diversos elementos básicos, dando a origem a elementos mais sofisticados, de forma quase total.

6. **O que é *estado interno* de uma variável?**

O estado interno de uma variável refere-se ao conteúdo da região de memória que esta associada à variável.

7. **Discorra sobre como dar nome às variáveis.**

Os nomes dados as variáveis devem facilitar a compreensão de sua função em um código.

O nome de uma variável é único, na região do código onde essa variável é válida, não sendo assim possível atribuir o mesmo nome a variáveis distintas. O seu nome em geral é composto por caracteres alfanuméricos e possivelmente alguns caracteres especiais tal como o *underline* "_", mas seu primeiro caractere não pode ser um dígito. Nas linguagens chamadas de *case sensitive* ocorre a diferenciação entre letras maiúsculas e minúsculas.

8. **Diferencie *variáveis globais* de *variáveis locais*, *variáveis estáticas* de *variáveis dinâmicas* e *variáveis escalares* de *variáveis compostas*.**

Variáveis globais são variáveis que possuem validade em todo o código fonte do programa, já as locais são variáveis que possuem sua validade restrita a um trecho do código, podendo ser este uma sub-rotina, uma função, bloco de execução, etc.

Variáveis estáticas são variáveis que possuem sua vinculação de memória definida durante a compilação, já as variáveis dinâmicas são variáveis que possuem sua vinculação de memória definida durante a execução da aplicação.

Variáveis escalares são variáveis que armazenam um único valor num dado instante, tais como inteiros e números de ponto flutuante. As variáveis compostas são variáveis que armazenam uma coleção de valores simultaneamente, tais como registros e vetores.

9. **Por que o uso de variáveis globais em funções e procedimentos é desaconselhado? Em que caso tal uso é aceitável?**

O uso de variáveis globais é desaconselhado pois elas dificultam a legibilidade do código, pois seu comportamento passa a ser quase imprevisível, dado que essa variável é válida para qualquer procedimento e função no código, permitindo que os mesmos a modifiquem, tornando difícil seu acompanhamento. O uso de variáveis globais é aconselhável ao se tratar de constante.

Ao definirmos uma constante que temos que utilizar em várias partes do nosso código, facilitando assim tanto a legibilidade quando a programação em si.

10. **No contexto das linguagens de programação, o que é um *alias*? Quais os problemas que podem surgir devido ao uso de *alias*?**

São variáveis que compartilham a mesma região de memória. Seu uso pode dificultar a legibilidade do código, dado que o comportamento de uma variável influi na outra variável, e o vice-versa, dificultando assim o monitoramento do comportamento das variáveis.

11. **O que significa dizer que uma linguagem é *fortemente tipada*? Dê um exemplo de uma linguagem fortemente tipada, de uma linguagem fracamente tipada e de uma linguagem não tipada.**

Linguagens de programação fortemente tipadas são as que exigem a identificação do tipo de cada variável, de modo explicito, quando sua declaração é feita e a mesma permanece com o mesmo tipo durante todo o seu uso.

Linguagem fortemente tipada : Cobol
Linguagem fracamente tipada: C
Linguagem não tipada: Linguagem de montagem (Assembly)

12. **Explique o que estabelece a norma IEEE 754 para a representação de números de precisão simples e dupla em ponto flutuante.**

A norma IEE 754 estabelece a representação dos números de ponto flutuante da seguinte forma:

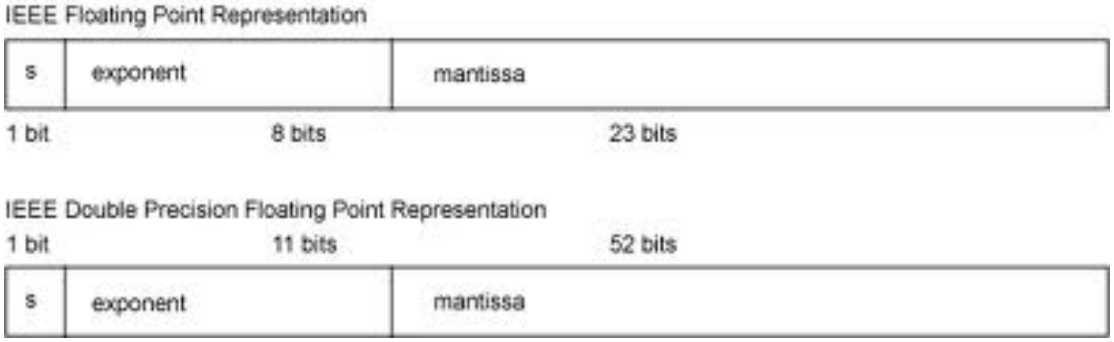
Precisão simples (32 bits):

- 1 bit para o sinal: sendo 0 significando números positivos e 1 negativos.
- 8 bits para o expoente: correspondente a soma de 127 com o expoente de base 2 do número representado.
- 23 bits de mantissa: correspondente a parte fracionária da mantissa do número representado, normalizada entre 1 e 2. Assim sendo sua parte inteira é sempre um bit igual a 1 que não se torna necessário representar.

Precisão Dupla (64 bits):

- 1 bit para o sinal.
- 11 bits para o expoente.
- 53 bits para a mantissa.

A imagem seguinte ilustra essa divisão:



13. **Explique o que são *vetores* e o que são *registros*.**

Vetores são estruturas que armazenam dados de um mesmo tipo. Ele esta dividido em posições, identificadas por índices, que tem o mesmo tamanho em bits. Os vetores ocupam uma região contigua de memória.

Registros são estruturas que armazenam uma coleção de dados de tipos diversos. Ele está dividido em campos, cada um deles identificado por um nome.

14. **Explique o que são vetores estáticos, vetores stack-dinâmicos, vetores heap-dinâmicos de tamanho fixo e vetores heap-dinâmicos de tamanho variável.**

Vetores estáticos: a vinculação de memória e a definição de tamanho ocorrem em tempo de compilação.

Vetores stack-dinâmicos: a vinculação de memória ocorre em tempo de execução e a definição de tamanho ocorre em tempo de compilação.

Vetores heap-dinâmicos de tamanho fixo: a vinculação de memória e a definição de tamanho ocorrem em tempo de execução. Mas seu tamanho se mantém durante a execução do programa uma vez definido.

Vetores heap-dinâmicos de tamanho fixo: a vinculação de memória e a definição de tamanho ocorrem em tempo de execução. Mas seu tamanho pode variar durante a execução do programa.

15. **O que são *ponteiros tipados*?**

Ponteiros tipados são variáveis que guardam um endereço de memória referente ao seu tipo. Ou seja, um ponteiro para um inteiro guarda um endereço de memória referente a um inteiro, não podendo o mesmo guardar um endereço de memória de quaisquer outros tipos de dados.