

Grupo Handbook

Handbook de Questões de TI

comentadas para CONCURSOS

Além do gabarito

Amostra Grátis

Prefácio

Esta é uma amostra grátis da série *Handbook de Questões de TI Comentadas para Concursos – Além do Gabarito*, que apresenta de forma pontual e resumida o formato e o estilo dos comentários produzidos pelo Grupo Handbook de TI.

Este exemplar é composto por questões dos Volumes 1, 2, 3, 4 e 5 da série. Cada volume é desenvolvido em função da banca organizadora e/ou da ênfase na área de TI. Dessa forma, o concurseiro que adquire todos os nossos produtos tem um ótimo material para melhor se preparar para a maioria dos concursos da área de TI. Além disso, sempre que necessário ele também pode se concentrar nos aspectos específicos de cada banca organizadora.

Não perca tempo! Junte-se à comunidade Handbook de TI e bons estudos,

Grupo Handbook de TI

Direitos Autorais

Este material é registrado no Escritório de Direitos Autorais (EDA) da Fundação Biblioteca Nacional. Todos os direitos autorais referentes a esta obra são reservados exclusivamente aos seus autores.

Os autores deste material não proíbem seu compartilhamento entre amigos e colegas próximos de estudo. Contudo, a reprodução, parcial ou integral, e a disseminação deste material de forma indiscriminada através de qualquer meio, inclusive na Internet, extrapolam os limites da colaboração. Essa prática desincentiva o lançamento de novos produtos e enfraquece a comunidade concurseira Handbook de TI.

A série *Handbook de Questões de TI Comentadas para Concursos – Além do Gabarito* é uma produção independente e contamos com você para mantê-la sempre viva.

Grupo Handbook de TI

Canais de Comunicação

A equipe Handbook de TI disponibiliza diversos canais de comunicação para seus clientes.

Loja Handbook de TI

<http://www.handbookdeti.com.br>

Serviço de Atendimento

Comunicação direta com a Equipe Handbook de TI pode ser feita em
<http://www.handbookdeti.com.br/contacts>

Twitter do Handbook de TI

Que acompanhar de perto o trabalho do Grupo Handbook de TI. Cadastre-se no twitter e comece a seguir o grupo Handbook de TI em <http://twitter.com/handbookdeti>

VOLUME 1
ANALISTA DE SUPORTE
BNDES 2008
FUNDAÇÃO CESGRANRIO

1. **Assuntos relacionados:** *Comandos UNIX, Link Simbólico, Hard Link,*

Banca: *CESGRANRIO*

Instituição: *BNDES*

Cargo: *Analista de Suporte*

Ano: *2008*

Questão: *31*

No Linux, que comando é utilizado para criação de links simbólicos?

- (a). `dmesg`
- (b). `rsync`
- (c). `mv -f`
- (d). `ln -s`
- (e). `chmod -l`

Solução:

(A) ERRADA

O comando `dmesg` é um comando do UNIX utilizado para imprimir as mensagens do kernel na saída padrão. Por padrão, as mensagens do kernel são salvas no arquivo `/var/log/dmesg`. O parâmetro mais comum do comando `dmesg` é o `-n`, que serve para controlar o nível de log que será enviado para a saída padrão. Usualmente, o comando `dmesg` é utilizado para diagnosticar problemas durante a etapa de inicialização do sistema.

(B) ERRADA

O `rsync` é um aplicativo UNIX que sincroniza diretórios e arquivos entre dois computadores ou dois pontos distintos em um mesmo computador. O aplicativo trabalha de forma incremental, sincronizando apenas as partes alteradas dos arquivos, poupando a rede e tornando a sincronização mais rápida. O `rsync` também é capaz de preservar links, propriedades e permissões dos arquivos, bem como as datas de criação e modificação.

(C) ERRADA

No UNIX, o comando `mv` é utilizado para renomear um arquivo ou movê-lo de um diretório para outro. Com a opção `-f`, o `mv` irá mover o arquivo sem solicitar a confirmação ao usuário, mesmo que um arquivo de mesmo nome já exista no diretório de destino.

(D) CORRETA

O comando `ln` é utilizado para criar links entre arquivos ou diretórios. Por sua vez, os links são pseudo arquivos que apontam para um arquivo real. No UNIX, existem basicamente dois tipos de links: os hard links e os links simbólicos. Os links simbólicos são criados pela opção `-s` do comando `ln`.

Um hard link é uma cópia de uma entrada do sistema de arquivos. As duas entradas contêm nomes diferentes, mas apontam para o mesmo inode, de modo que o conteúdo e as permissões sejam compartilhados. Embora os hard links não ocupem espaço útil no sistema de arquivos, eles possuem duas limitações básicas. A primeira é que o hard-link e o arquivo precisam estar no mesmo sistema de arquivos, e a segunda é que os hard links não podem

apontar para diretórios.

Os links simbólicos são pequenos arquivos que apontam para outros arquivos, que podem estar localizados em qualquer lugar, inclusive em sistemas de arquivos remotos. Ao contrário dos hard links, os links simbólicos ocupam espaço, embora pequeno, no sistema de arquivos e podem apontar para diretórios. As permissões do arquivo real são herdadas pelos links simbólicos e, caso o arquivo real seja apagado, o link simbólico torna-se um dead link, pelo fato de apontar para um arquivo ou diretório que não mais existe no sistema de arquivos.

(E) ERRADA

O comando `chmod` é utilizado para modificar as permissões de acesso em arquivos ou diretórios no UNIX. Com o `chmod` é possível, por exemplo, definir se um usuário ou um grupo pode ler, alterar ou executar os arquivos. No caso dos diretórios, o privilégio de execução corresponde ao direito de listar seu conteúdo.

2. Assuntos relacionados: *Redes de Computadores, Endereçamento IP, Protocolo ARP,***Banca:** *CESGRANRIO***Instituição:** *BNDES***Cargo:** *Analista de Suporte***Ano:** *2008***Questão:** *32*

Suponha uma rede TCP/IP formada por 3 equipamentos conectados em um mesmo switch:

Estação X, IP 192.168.10.100/24

Estação Y, IP 192.168.10.200/24

Roteador R, IP 192.168.10.1/24

Considerando-se que o default gateway (default route, rota padrão) de cada estação é R, observe as afirmativas abaixo.

- I - Caso X inicie uma conexão TCP destinada a Y, os pedidos de requisição de conexão (SYN) passarão por R.
- II - Todas as mensagens ARP Request enviadas por Y são recebidas por R.
- III - Sem que o endereçamento IP seja alterado, é possível adicionar 253 estações a essa rede.

SOMENTE está(ão) correta(s) a(s) afirmativa(s)

- (a). I
- (b). II
- (c). I e II
- (d). II e III
- (e). I, II e III

Solução:

A afirmativa I é incorreta. Como X e Y pertencem a mesma subrede, as requisições enviadas de X para Y não passarão por R. As requisições partindo de X ou Y só passarão por R caso sejam destinadas a alguma estação localizada em uma subrede diferente de 192.168.10.0/24.

A alternativa II é correta. As mensagens ARP Request (Address Resolution Protocol) tem por objetivo recuperar o endereço MAC de um outro elemento da rede, para o qual é conhecido o endereço IP. Em linhas gerais, quando Y precisa descobrir o endereço MAC de X, o processo é o seguinte:

- Y monta um pacote ARP Request com a pergunta “Quem tem o IP 192.168.10.100?”;
- Y envia o pacote para o endereço de broadcast FF:FF:FF:FF:FF:FF;
- todos os integrantes da subrede recebem o pacote ARP Request;
- ao receber o pacote, X verifica que é capaz de receber a pergunta;
- X monta um pacote ARP Response contendo seu endereço MAC e o envia diretamente a Y;
- Y recebe o ARP Response, e agora está preparado para montar o pacote e endereçá-lo com o MAC de X.

A alternativa III é incorreta. A subrede 192.168.10.0/24 contém 256 endereços. A faixa de endereçamento útil é de 192.168.10.1 até 192.168.10.254, já que os endereços 192.168.10.0 e 192.168.10.255 são os endereços de rede e de broadcast, respectivamente. Ou seja, a subrede em questão pode conter, no máximo, 254 elementos. Como X, Y e R já consumiram 3 desses endereços, podem ser adicionados, no máximo, mais 251 elementos a essa subrede.

VOLUME 2
ANALISTA DE SISTEMAS - ENG. DE
SOFTWARE
PETROBRAS 2008
FUNDAÇÃO CESGRANRIO

3. **Assuntos relacionados:** *Arquitetura de Computadores, Modos de Endereçamento de Memória,***Banca:** CESGRANRIO**Instituição:** Petrobras**Cargo:** Analista de Sistemas - Eng. de Software**Ano:** 2008**Questão:** 21

Um computador tem um registrador R e um conjunto de instruções de um operando, todas com modo de endereçamento indireto. Três destas instruções são especificadas a seguir.

LD: Copia da memória principal para o registrador R.

AC: Adiciona da memória principal ao registrador R.

ST: Move do registrador R para a memória principal.

Considere o programa apresentado abaixo, executado no computador, acessando o bloco de memória principal, cuja situação inicial é mostrada a seguir.

Endereço	Valor Armazenado
00H	01H
01H	02H
02H	03H
03H	04H
04H	05H

LD 01H

AC 02H

ST 03H

AC 00H

ST 01H

LD 03H

ST 00H

Considere que tanto o endereçamento quanto os valores envolvidos nas operações utilizam apenas um byte de memória cada. Após a execução do programa, qual será, em hexadecimal, a soma dos valores armazenados no bloco de memória?

(a). 00H

(b). 04H

(c). 0AH

(d). 10H

(e). 1CH

Solução:

Primeiramente, os conceitos de endereçamento de dados devem estar bem claros. Em uma instrução de programa, há várias maneiras de referenciar um valor, as mais conhecidas são:

- *Imediato*: o valor do operando é especificado diretamente na instrução. Sua principal vantagem é não requerer acesso à memória para obter o operando. A desvantagem é que esse modo impõe uma limitação no tamanho do operando. Suponha que o computador

descrito suporte acesso imediato. A instrução LD 30H faria com que o valor 30H fosse copiado para o registrador R. Entretanto, há ocasiões em que não somente um byte deve ser copiado, por exemplo LD 201040H. Nesse caso, como o valor é armazenado diretamente na instrução, seria necessário aumentar o tamanho da instrução e isso não é possível na maioria das arquiteturas de computador;

- *Direto*: o campo de endereço contém o endereço efetivo do operando na memória. Requer, portanto, apenas um acesso para determinar o valor do operando. Sua limitação é fornecer um espaço de endereçamento limitado. Suponha que o computador descrito suporte endereçamento direto. A instrução LD 01H, faria com que o valor armazenado na posição de memória 01H, ou seja, 02H fosse copiado. Entretanto, se a instrução possuir somente um byte para o endereçamento direto, por exemplo, a quantidade de posições de memória estará limitada em 256 (2^8);
- *Indireto*: o campo de endereço aponta para uma posição de memória que contém o endereço de memória do operando. Sua principal desvantagem é a necessidade de dois acessos à memória. A vantagem em relação ao modo de endereçamento direto é o aumento do espaço de endereçamento, que passa a ser igual 2^n , onde n é o tamanho da palavra na memória. Suponha que o computador tenha somente um byte para endereçar a posição de memória, mas que essa posição de memória corresponda a uma palavra com tamanho de 2 bytes. Um endereçamento na forma indireta, possibilitará o endereçamento de 65536 posições de memória (2^{16}) e não mais 256 como no endereçamento direto. Não é o caso da questão, onde tanto o tamanho permitido para endereçamento na instrução e o tamanho da palavra de memória são iguais a um byte;
- *Registrador*: é semelhante ao modo direto, no entanto, o campo de endereço se refere a um registrador e não a uma posição de memória. Geralmente, esse campo é composto por 3 ou 4 bits, o que permite referenciar de 8 a 16 registradores de propósito geral. Suas vantagens são o tamanho pequeno do campo de endereço e a não necessidade de se acessar à memória. Sua desvantagem é o espaço de endereçamento limitado pelo número de registradores. Por exemplo, poderíamos supor que o computador da questão permitisse endereçamento por registrador e tivesse 16 registradores. Assim, seria possível que um registrador além do R, por exemplo S, pudesse ser endereçado como 05H. Uma instrução da maneira LD 05H copiaria o valor do registrador S para o registrador R;
- *Indireto via Registrador*: semelhante ao modo de endereçamento indireto. O campo de endereço aponta para o registrador que contém a posição de memória do operando. Sua vantagem é a necessidade de um único acesso à memória, um a menos que no modo indireto;
- *Deslocamento*: requer que uma instrução tenha dois campos de endereço, com pelo menos um explícito. O valor de um dos campos é usado diretamente (valor = A). O outro campo é baseado no código da operação, e especifica um registrador cujo conteúdo é adicionado à A, para produzir o endereço efetivo. Os três modos de endereçamento por deslocamento são: *relativo*, *via registrador-base* e *indexado*;
- *Pilha*: a pilha é um bloco reservado de posições de memória. Elementos podem ser colocados e removidos do topo da pilha. O apontador do topo da pilha (*stack-pointer*) é mantido em um registrador. Portanto, de fato, referências a pilha são feitas por endereçamento indireto via registrador.

Já que a questão trata de endereçamento indireto, o valor armazenado no local especificado pelo operando é o endereço de memória do valor que será utilizado na operação. Por exemplo, a instrução LD 01H, carrega, no registrador R, o valor 03H, pois no endereço 01H está

armazenado o endereço 02H, que por sua vez, contém o valor desejado, 03H. Seguindo os passos do programa, teremos:

1. LD 01H, R \leftarrow 03H, R recebe o valor armazenado no endereço 02H;
2. AC 02H, R \leftarrow 03H + 04H \leftarrow 07H, o valor de R é somado ao valor armazenado no endereço 03H;
3. ST 03H, [04H] \leftarrow 07H, a posição de memória 04H recebe o valor do registrador R;
4. AC 00H, R \leftarrow 07H + 02H \leftarrow 09H, o valor de R é somado ao valor da posição 01H;
5. ST 01H, [02H] \leftarrow 09H, a posição de memória 02H recebe o valor de R;
6. LD 03H, R \leftarrow 07H, R recebe o valor armazenado no endereço 04H;
7. ST 00H, [01H] \leftarrow 07H, a posição de memória 01H recebe o valor de R.

Após o término do programa, a situação final do bloco de memória será de acordo com a Tabela 1.

Tabela 1: situação final do bloco de memória.

Endereço	Valor Armazenado
00H	01H
01H	07H
02H	09H
03H	04H
04H	07H

A soma é $01H + 07H + 09H + 04H + 07H = 1CH$, que, em decimal, é 28. Logo, a alternativa correta é a letra (e).

4. **Assuntos relacionados:** *Arquitetura de Computadores, Thread,*

Banca: *CESGRANRIO*

Instituição: *Petrobras*

Cargo: *Analista de Sistemas - Eng. de Software*

Ano: *2008*

Questão: *22*

Alguns sistemas operacionais permitem que seus processos criem múltiplos threads de execução. Em operação normal, o que é previsto que os threads de um mesmo processo do sistema operacional compartilhem?

- (a). Arquivos abertos
- (b). Registradores
- (c). Pilha (stack)
- (d). Variáveis locais de cada thread
- (e). Contador de instrução (program counter)

Solução:

Uma thread é comumente definida como um fluxo único de controle sequencial dentro de um programa. O uso de threads visa reduzir o custo do gerenciamento de processos, que consiste principalmente em:

- criação do processo;
- trocas de contextos entre processos;
- overhead associado a esquemas de proteção de memória;
- comunicação entre processos.

Podemos dizer que as threads pertencentes ao mesmo processo utilizam os recursos alocados no sistema operacional para esse processo, como:

- o espaço de endereçamento na memória;
- os arquivos abertos (handles);
- os objetos de sincronização.

O compartilhamento desses recursos permite que os fluxos de execução (threads) se comuniquem eficientemente. Então, a letra (a) é a opção correta.

Entretanto, threads dentro de um processo (e também entre processos) são escalonadas e executadas independentemente. No momento de sua execução, cada thread recebe alguns recursos próprios, como:

- os registradores;
- a pilha de execução, que lhe dará poder para chamar métodos, passar parâmetros e alocar variáveis locais;
- o contador de instrução (program counter), que é essencial para que o fluxo de execução prossiga.

É importante perceber que para que o acesso a esses recursos exclusivos ocorra, é necessário o chaveamento de contexto entre as threads, ou seja, o estado dos elementos próprios citados deverá ser armazenado e restaurado a cada troca de thread no uso do processador.

Dado o exposto, as alternativas (b), (c), (d) e (e) podem ser eliminadas, pois citam recursos que não são compartilhados entre threads. A Figura 1 exemplifica o compartilhamento das threads dentro de um processo.

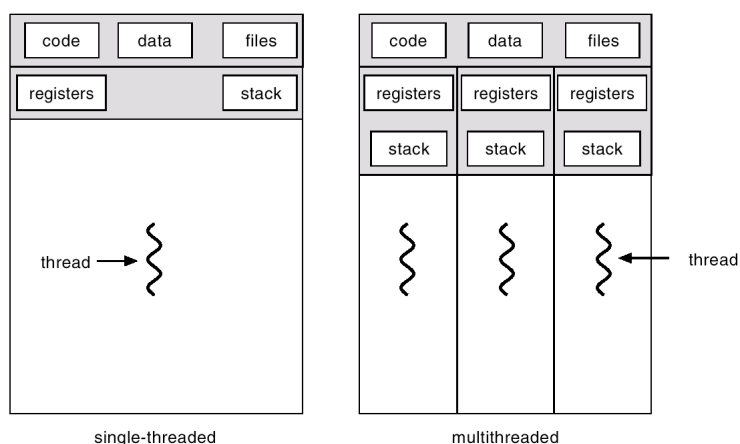


Figura 1: exemplificação de um processo com uma única thread e com múltiplas threads.

Concluindo, as principais vantagens do uso de threads são:

- permite a exploração do paralelismo real oferecido por máquinas multiprocessadas;
- possibilita o aumento do número de atividades executadas por unidade de tempo (throughput);
- permite sobrepor operações de cálculo com operações de I/O e, com isso, reduzir o tempo de resposta;
- o tempo de criação e destruição de threads é inferior ao tempo de criação e destruição de processos, respectivamente;
- o chaveamento de contexto entre threads é mais rápido que o tempo de chaveamento entre processos;
- como threads compartilham o descritor do processo, elas dividem o mesmo espaço de endereçamento, o que permite a comunicação por memória compartilhada sem interação com o núcleo (kernel) do sistema operacional.

VOLUME 3
QUESTÕES DA FCC
PARTE 1
FUNDAÇÃO CARLOS CHAGAS

5. **Assuntos relacionados:** *Sistemas de Enumeração,***Banca:** FCC**Instituição:** MPU**Cargo:** *Analista de Desenvolvimento de Sistemas***Ano:** 2007**Questão:** 31

As representações de números inteiros, positivos e negativos na notação de complemento de dois, bem como os positivos e negativos na notação de excesso, têm os bits de sinal com os respectivos valores:

- (a). 0, 1, 0 e 1
- (b). 1, 0, 0 e 1
- (c). 0, 1, 1 e 0
- (d). 1, 0, 1 e 0
- (e). 0, 0, 1 e 1

Solução:

As notações de complemento de dois e de excesso são as mais conhecidas para a representação de números inteiros. Entretanto, a notação de complemento de dois é mais largamente utilizada na prática.

Na notação de complemento de dois, os valores positivos são formados partindo de uma cadeia de 0s e então contando em binário até que o padrão seja formado por um 0 seguido de 1s, formando os números 0, 1, 2... nessa ordem. Já os números negativos, são formados partindo de uma cadeia de 1s e contando em binário, em ordem decrescente, até que o padrão seja formado de um 1 seguido de 0s, formando os números -1, -2, -3... nessa ordem. Nesta notação, o bit mais à esquerda do padrão indica o sinal do valor representado. Esse bit é mais conhecido como **bit de sinal**. Ou seja, na notação de complemento de dois, o bit de sinal dos números positivos é 0 e dos números negativos é 1.

Na notação de excesso, o valor zero é representado por um 1 seguido de 0s. Os padrões que seguem são representados para representar os números positivos 1, 2, 3...; os que precedem são utilizados para representar os números negativos -1, -2, -3... Note que os números positivos são representados começando com 1 e os números negativos são representados começando com 0.

Na Tabela 2, um quadro comparativo das notações de complemento de dois e de excesso para melhor entendimento de como são representados os números inteiros. Note que a única diferença é o bit de sinal.

Já podemos concluir com o que foi apresentado e de acordo com a Tabela 2 que a resposta a ser marcada é a letra C.

Valor	Complemento de dois	Excesso
7	0111	1111
6	0110	1110
5	0101	1101
4	0100	1100
3	0011	1011
2	0010	1010
1	0001	1001
0	0000	1000
-1	1111	0111
-2	1110	0110
-3	1101	0101
-4	1100	0100
-5	1011	0011
-6	1010	0010
-7	1001	0001
-8	1000	0000

Tabela 2: notações dos números inteiros.

6. Assuntos relacionados: *Sistemas de Enumeração,***Banca:** *FCC***Instituição:** *MPU***Cargo:** *Analista de Desenvolvimento de Sistemas***Ano:** *2007***Questão:** *32*

O resultado da operação lógica “10101011 XOR 11101100” será:

- (a). 10111000
- (b). 01000111
- (c). 10010111
- (d). 11101111
- (e). 10101000

Solução:

A operação XOR é também conhecida como ou-exclusivo ou disjunção-exclusiva. A disjunção exclusiva de um par de proposições p e q , deve significar que p é verdadeiro ou que q é verdadeiro, mas não ambos. A Tabela 3 é a tabela verdade para a operação XOR.

P	Q	P XOR Q
V	V	F
V	F	V
F	V	V
F	F	F

Tabela 3: tabela verdade da operação XOR.

O resultado da operação é feito bit a bit, onde 1 indica verdadeiro e 0 indica falso. Por exemplo, o primeiro bit do primeiro operando (10101011) é 1 e o primeiro bit do segundo operando (11101100) também é 1, logo, pela tabela verdade, concluímos que o primeiro bit do resultado deve ser 0(falso). A operação é feita entre os segundos bits de cada operando e assim por diante. O resultado final será 01000111, tornando a letra B, a alternativa a ser marcada.

VOLUME 4
QUESTÕES FCC
PARTE 2
FUNDAÇÃO CARLOS CHAGAS

7. Assuntos relacionados: *Programação, Algoritmos de Ordenação,***Banca:** *FCC***Instituição:** *TRT 15a Região***Cargo:** *Analista Judiciário - Tecnologia da Informação***Ano:** *2009***Questão:** *21*

São algoritmos de classificação por trocas apenas os métodos

- (a). SelectionSort e InsertionSort.
- (b). MergeSort e BubbleSort.
- (c). QuickSort e SelectionSort.
- (d). BubbleSort e QuickSort.
- (e). InsertionSort e MergeSort.

Solução:

Os algoritmos de ordenação são uns dos principais objetos de estudo na área de computação. Tais algoritmos tem por objetivo colocar os elementos de uma sequência em uma determinada ordem, sendo as mais utilizadas as ordens numéricas e alfabéticas.

Uma das principais razões para se ordenar uma sequência é permitir que os seus elementos sejam acessados de forma mais eficiente. Os métodos de ordenação mais conhecidos e utilizados são os seguintes:

- Ordenação por Troca
 - BubbleSort (Método da Bolha)
 - QuickSort (Método da Troca e Partição)
- Ordenação por Inserção
 - InsertionSort (Método da Inserção Direta)
 - BinaryInsertionSort (Método da Inserção Direta Binária)
- Ordenação por Seleção
 - SelectionSort (Método da Seleção Direta)
 - HeapSort (Método da Seleção em Árvore)
- Outros métodos
 - MergeSort (Método da Intercalação)
 - BucketSort (Método da Distribuição de Chave)
 - CountingSort (Método da Ordenação por Contagem)

Como podemos ver, dentre os métodos apresentados nas alternativas da questão, os únicos baseados na ordenação troca são o BubbleSort e o QuickSort. Portanto, a resposta da questão é a alternativa D. Tais algoritmos são classificados como de troca por que seus processos de ordenação se dão através de trocas entre pares de elementos do vetor. Embora o BubbleSort e o QuickSort se baseiem em trocas, seu funcionamento geral e o seu desempenho diferem substancialmente.

O BubbleSort é, talvez, o método simples de ordenação, que funciona através de sucessivas trocas entre pares de elementos do vetor. O método realiza varreduras no vetor, trocando

pares adjacentes de elementos sempre que o próximo elemento for menor que o anterior. Após uma varredura, o maior elemento está corretamente posicionado no vetor e não precisa mais ser comparado. Dessa forma, após a i -ésima varredura, os i maiores elementos estão ordenados. A complexidade algorítmica de tempo do BubbleSort é $O(n^2)$.

O Quicksort adota uma estratégia conhecida como divisão e conquista. Por isso, usualmente, o Quicksort é implementado utilizando-se o paradigma recursivo. No primeiro passo do Quicksort, um elemento da lista é escolhido como pivô. Em seguida, a lista é reorganizada (por meio de trocas de posição entre os elementos) de modo que todos os elementos anteriores ao pivô sejam menores que ele, e todos os elementos posteriores ao pivô sejam maiores que ele. Ao fim desse processo, que denomina-se particionamento, o pivô estará em sua posição final e haverá duas sublistas não ordenadas.

Em seguida, o algoritmo Quicksort é reaplicado a cada uma das sublistas, sendo a base da recursão as sublistas de tamanho zero ou um que, por definição, estão sempre ordenadas. O processo é finito, pois a cada iteração pelo menos um elemento é posto em sua posição final e não será mais manipulado na iteração seguinte. A complexidade algorítmica de tempo do Quicksort é $O(n \lg n)$.

A seguir, exemplos de implementação do bubble sort e do quicksort usando a linguagem de programação ruby.

```
#BubbleSort em Ruby
def bubblesort(list)
  return list if list.size <= 1 # already sorted
  loop do
    swapped = false
    0.upto(list.size-2) do |i|
      if list[i] > list[i+1]
        list[i], list[i+1] = list[i+1], list[i] # swap values
        swapped = true
      end
    end
    break unless swapped
  end
  return list
end

#QuickSort em Ruby
def quicksort (array)
  return array if array.size <= 1
  pivot = array[0]
  return quicksort (array.select {|y| y < pivot })
    + array.select { |y| y == pivot } +
    quicksort (array.select {|y| y > pivot })
end
```

8. **Assuntos relacionados:** *Programação, Estruturas de Dados, Tabela Hash,***Banca:** FCC**Instituição:** TRT 15a Região**Cargo:** Analista Judiciário - Tecnologia da Informação**Ano:** 2009**Questão:** 22

Uma estrutura de dados especial de armazenamento de informações, cuja ideia central é utilizar uma função que, quando aplicada sobre uma chave de pesquisa, retorna o índice onde a informação deve ser armazenada denomina-se

- (a). vetor de dispersão.
- (b). matriz de dispersão.
- (c). tabela hash.
- (d). árvore binária.
- (e). lista encadeada

Solução:

A resposta da questão é a alternativa C, tabela hash, que também é conhecida como tabela de espalhamento ou de dispersão.

A implementação de uma tabela hash se baseia na escolha de uma função, chamada função hash, que associe uma chave de pesquisa a um índice em uma estrutura de dados. O requisito mais importante de uma função hash é o de distribuir uniformemente as chaves pelos vários índices, de forma eliminar ou minimizar a chance de que mais de uma chave seja mapeada para o mesmo índice, problema conhecido como colisão.

Por mapearem uma chave de pesquisa a um determinado índice de forma direta, as tabelas hash proporcionam um tempo médio de busca constante, ou seja, $O(1)$. Em virtude de seu alto desempenho, as tabelas hash são tipicamente utilizadas na indexação de grandes volumes de informações em bancos de dados e na criação de esquemas associativos de acesso às memória cache. A Figura 2 mostra, de forma básica, como se dá o esquema de mapeamento de chaves em índices usando uma tabela hash.

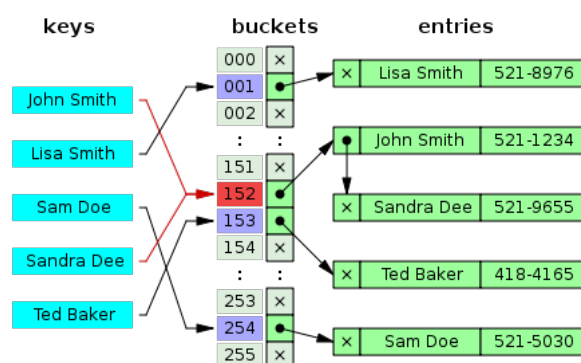


Figura 2: Tabela Hash.

No entanto, a definição de uma boa função hash nem sempre é uma tarefa simples, sendo um trabalho profundamente relacionado à estatística. Portanto, para otimizar a função hash, é necessário conhecer a natureza da chave a ser utilizada e como os valores que a chave pode assumir se distribuem ao longo do domínio. Os métodos de implementação de funções hash mais comuns são o método da **divisão**, o método da **dobra**, e o método da **multiplicação**.

Com relação às limitações das tabelas hash, valem as seguintes observações. As tabelas hash são estruturas de dados que não permite armazenar elementos repetidos, recuperar elementos sequencialmente, nem recuperar antecessores ou sucessores de um determinado elemento já encontrado. Caso essas operações tenham muita importância no sistema, vale a pena considerar a utilização de estruturas de dados como listas encadeadas, árvores, entre outras.

VOLUME 5
DESENVOLVIMENTO DE SOFTWARE
PARTE 1
FUNDAÇÃO CESGRANRIO

9. **Assuntos relacionados:** *Desenvolvimento de Sistemas, Metodologia de Desenvolvimento de Software, Extreme Programming (XP),*

Banca: Cesgranrio

Instituição: BNDES

Cargo: Analista de Sistemas - Desenvolvimento

Ano: 2008

Questão: 31

Que situação favorece a escolha do uso de XP para um projeto de desenvolvimento de software, em oposição à escolha do RUP ou do modelo Cascata?

- (a). Equipe do projeto localizada em diferentes cidades e com poucos recursos de colaboração.
- (b). Equipe do projeto formada por pessoas com alto grau de competitividade.
- (c). Cliente do projeto trabalhando em parceria com a equipe do projeto e sempre disponível para retirar dúvidas.
- (d). Requisitos do software com pequena probabilidade de mudanças.
- (e). Presença de um processo organizacional que exige a elaboração de vários documentos específicos para cada projeto.

Solução:

O Rational Unified Process (RUP) é um framework muito difundido e utilizado que pode ser adaptado a vários tipos de projetos de software. Podem ser derivados do RUP processos para projetos de vários portes, pois este framework define uma grande lista de papéis, artefatos, atividades e fluxos. No entanto, o RUP é tido como muito complexo, e costuma ser visto como um pesado e burocrático, ao contrário das metodologias ágeis.

Em contrapartida, o EXtreme Programming (XP) aparece como uma alternativa mais leve para times de tamanho pequeno e médio porte, que desenvolvem software em um contexto de requisitos vagos e rapidamente modificados. O XP enfatiza a codificação e os testes de códigos, considerando a presença constante dos clientes no desenvolvimento fundamental. Pela característica de simplicidade que esta técnica apresenta, poucos artefatos, papéis e atividades são definidos. Portanto, a resposta da questão é a alternativa C.

Ainda sobre o XP, são muito úteis ainda as seguintes informações. Os quatro valores fundamentais da metodologia XP são a comunicação, simplicidade, feedback e coragem. Para fazer valer tais valores, a metodologia adota como práticas as seguintes:

- **Jogo de Planejamento:** O desenvolvimento é feito em iterações semanais. No início da semana, desenvolvedores e cliente reúnem-se para priorizar as funcionalidades. O escopo é reavaliado semanalmente, e o projeto é regido por um contrato de escopo negociável;
- **Pequenas Versões:** A liberação de pequenas versões funcionais do projeto auxilia muito no processo de aceitação por parte do cliente, que já pode testar uma parte do sistema que está comprando;
- **Metáforas:** A metodologia prega o uso de metáforas, como forma de facilitar a comunicação com o clientes;
- **Time Coeso:** A equipe de desenvolvimento é formada pelo cliente e pela equipe de desenvolvimento;

- **Ritmo Sustentável:** Trabalhar com qualidade, buscando ter ritmo de trabalho saudável, sem horas extras. Horas extras são permitidas quando trouxerem produtividade para a execução do projeto;
- **Reuniões em pé:** Reuniões em pé para não se perder o foco nos assuntos, produzindo reuniões rápidas, apenas abordando tarefas realizadas e tarefas a realizar pela equipe;
- **Posse Coletiva do Código:** O código fonte não tem dono e ninguém precisa solicitar permissão para poder modificar o mesmo. Tal prática tem como objetivo fazer com que a equipe conheça todas as partes do sistema;
- **Programação em Duplas:** Geralmente a dupla é formada por um iniciante na linguagem e outra pessoa funcionando como um instrutor. Como é apenas um computador, o novato é que fica à frente fazendo a codificação, e o instrutor acompanha ajudando a desenvolver suas habilidades. Desta forma o programa sempre é revisto por duas pessoas, evitando e diminuindo assim a possibilidade de erros;
- **Padrões de Codificação:** A equipe de desenvolvimento precisa estabelecer regras para programar e todos devem seguir estas regras. Desta forma, parecerá que todo o código foi editado pela mesma pessoa, mesmo quando a equipe possui uma quantidade maior de programadores;
- **Refatoração:** É um processo que permite a melhoria contínua da programação, que tem como alvos a melhoria da legibilidade do código, maiores modularização e reaproveitamento do código.

10. **Assuntos relacionados:** UML, Diagrama de Classes, Composição Agregada,

Banca: Cesgranrio

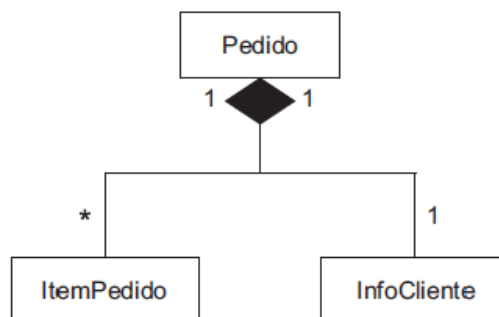
Instituição: BNDES

Cargo: Analista de Sistemas - Desenvolvimento

Ano: 2008

Questão: 35

Considere o relacionamento de “todo-parte” ilustrado no diagrama UML abaixo.



É correto afirmar que

- um objeto da classe InfoCliente pode participar de mais de um relacionamento de composição desempenhando o papel de “parte”.
- um objeto da classe ItemPedido pode participar de mais de um relacionamento de composição desempenhando o papel de “parte”.
- uma instância da classe InfoCliente pode existir antes mesmo que a instância da classe Pedido com que se relacionará tenha sido criada.
- o relacionamento ilustrado acima é ternário.
- a cardinalidade do pedido no relacionamento com ItemPedido igual a 1 não precisaria ser apresentada, uma vez que não poderia assumir outro valor.

Solução:

Lembre-se que diagramas de classe nos permitem identificar tanto o conteúdo de uma classe quanto o relacionamento entre várias classes. Em um diagrama de classe, podemos mostrar as variáveis e métodos membros de uma classe. Podemos também mostrar se uma classe herda de outra, ou se mantém uma referência para outra.

O relacionamento “todo-parte”, também conhecido como composição agregada, ou relacionamento “tem-um” ou “parte-de”, indica que um objeto (o todo) é composto de outros objetos (as partes). Com a composição agregada, o relacionamento entre os objetos é muito mais forte que com a associação, pois nela o todo não pode existir sem suas partes e as partes não podem existir sem o todo. Vários pontos importantes são inerentes a este fato. São estes:

- remoção do todo implica na remoção das partes;
- existe apenas um todo, isto é, as partes não são compartilhadas com outros todos;
- as partes não podem ser acessadas “fora” do todo, ou seja, elas são particulares para o todo;
- uma mensagem destinada a uma parte deve ser enviada para o todo e retransmitida por ele à parte.

Isto significa que a agregação composta deve ser utilizada somente quando um objeto é considerado como uma parte de outro objeto e não apenas uma associação ocasional com existência e visibilidade independentes.

A fim de representar este tipo de relacionamento, utiliza-se uma linha que termina com um símbolo de diamante preenchido, símbolo este colocado contra o todo. Além disso, para evitar qualquer confusão possível, ao todo é atribuída, explicitamente, a multiplicidade de 1 (um), mesmo porque apenas um todo é possível.

A partir do diagrama UML e do que expomos, conseguimos classificar a classe Pedido como sendo o todo e as classes ItemPedido e InfoCliente como as partes do relacionamento. Assim, podemos eliminar as alternativas A e B, pois as partes não podem participar de mais de um relacionamento na composição agregada. Podemos eliminar, também, a alternativa C uma vez que uma parte não pode existir sem o todo. Lembre-se que seria necessário que as 3 (três) entidades estivessem associadas simultaneamente para que tivéssemos um relacionamento ternário, logo, a alternativa D também está errada. Portanto, a alternativa **E** é a correta, pois a multiplicidade 1 (um) atribuída ao todo é utilizada apenas para evitar qualquer confusão possível.

Questao	Resposta
1	D
2	B
3	E
4	A
5	C
6	B
7	D
8	C
9	C
10	E

Índice Remissivo

Algoritmos de Ordenação, 20
Arquitetura de Computadores, 10, 13

Comandos UNIX, 5
Composição Agregada, 27

Desenvolvimento de Sistemas, 25
Diagrama de Classes, 27

Endereçamento IP, 7
Estruturas de Dados, 22
Extreme Programming (XP), 25

Hard Link, 5

Link Simbólico, 5

Metodologia de Desenvolvimento de Software,
25
Modos de Endereçamento de Memória, 10

Programação, 20, 22
Protocolo ARP, 7

Redes de Computadores, 7

Sistemas de Enumeração, 16, 18

Tabela Hash, 22
Thread, 13

UML, 27