Ciência da Computação Integrada

Aula 4 – Formação Específica, parte II.

Prof. Msc. Álvaro A. Colombero Prado

Exercício 1 - Circuitos digitais combinatórios e multiplexadores

- Um prédio de 4 andares, sendo o primeiro andar térreo, é servido por 2 elevadores. Por motivo de economia de energia, o elevador 2 só é acionado se for solicitado em mais de 2 andares. Considere um circuito proposto para habilitar o acionamento do elevador 2 conforme é mostrado a seguir.
- Ele utiliza um multiplexador 4x1, cuja saída é selecionada através da composição dos sinais A e B, que indicam se os andares 1 e 2 solicitaram o serviço do elevador.
- Assim, o valor $AB=10_{(2)}$ indica que o primeiro andar solicitou o elevador, mas não o segundo. Os sinais C e D indicam se os andares 3 e 4 solicitarem o serviço, respectivamente.

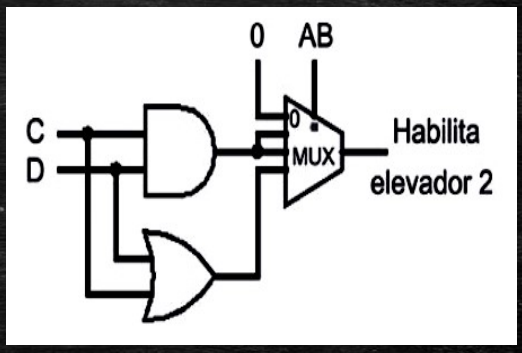
Exercício 1 - Circuitos digitais combinatórios e multiplexadores

- Com base na análise do circuito proposto para o problema, avalie as seguintes asserções e a relação proposta entre elas.

I. O circuito não atende às especificações do projeto.

PORQUE

II. A entrada superior do multiplexador com valor constante 0 indica que a saída será 0, independentemente dos valores dos sinais A, B, C e D.



Exercício 1 - Circuitos digitais combinatórios e multiplexadores

- A respeito dessas asserções, assinale a opção correta.

A. As asserções I e II são proposições verdadeiras, e a II é uma justificativa da I.

B. As asserções I e II são proposições verdadeiras, e a II não é uma justificativa da I.

C. A asserção I é uma proposição verdadeira, e a II é uma proposição falsa.

D. A asserção I é uma proposição falsa, e a II é uma proposição verdadeira.

E. As asserções I e II são proposições falsas.

Exercício 1 - Circuitos digitais combinatórios e multiplexadores - Teoria

- Circuitos digitais combinatórios são aqueles cuja saída depende unicamente da(s) sua(s) entrada(s) em dado instante e que não têm nenhuma memória de sua situação anterior. Circuitos lógicos digitais sequenciais dependem não apenas de sua entrada em um dado instante, mas também de seu histórico de ativação.
- Um dos dispositivos utilizados em projetos de circuitos digitais é o multiplexador,também chamado de mux. Ele funciona como uma espécie de "chave seletora": recebe várias entradas e seleciona apenas uma, a que será copiada na sua saída. A decisão sobre qual das entradas vai ser selecionada depende de um conjunto de entradas de seleção e de variáveis binárias que especificam quais dos sinais de entrada serão copiados para a saída.

Exercício 1 - Circuitos digitais combinatórios e multiplexadores - Teoria

- Como exemplo, suponha ummultiplexador que apresente apenas duas entradas e uma saída, às vezes chamado de multiplexador 2-para-1, mostrado na figura 1.

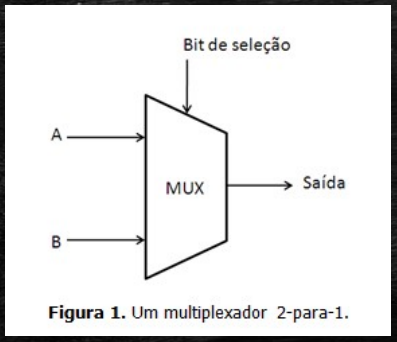


Figura 1. Um multiplexador 2-para-1.

Tabela 1. Tabela verdade do multiplexador da figura 1.

A	B	Bit de seleção(S)	Saída(O)
0	0	0	0
0	1	0	0
1	0	0	1
1	1	0	1
0	0	1	0
0	1	1	1
1	0	1	0
1	1	1	1

Exercício 1 - Circuitos digitais combinatórios e multiplexadores - Teoria

- Observe que, quando o bit de seleção é igual a 0, a saída do multiplexador é igual ao valor da entrada A. Quando o bit de seleção é igual a 1, a saída do multiplexador é igual ao valor da entrada em B. Um circuito como esse corresponde à equação que segue:

$$O = (A \cdot \overline{S}) + (B \cdot S)$$

- Multiplexadores podem ter diversos tamanhos. Na figura 2, vemos um exemplo de um multiplexador 4-para-1. Como temos o dobro de entradas (quatro), precisamos de dois bits para selecionar uma das quatro portas.

Exercício 1 - Circuitos digitais combinatórios e multiplexadores - Teoria

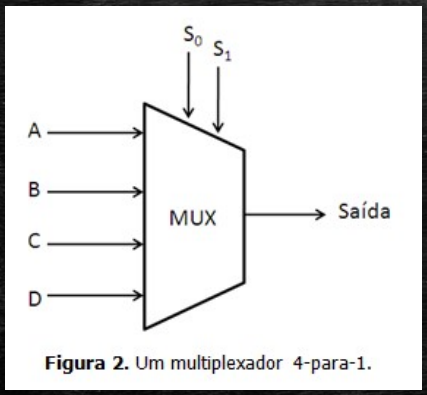


Figura 2. Um multiplexador 4-para-1.

Tabela 2. Tabela verdade do multiplexador da figura 2.

S_0	S_1	Saída (O)
0	0	A
0	1	B
1	0	C
1	1	D

Observe que, dependendo da combinação dos bits S_0 e S_1, temos, na saída, os valores de A, B, C ou D.

Exercício 1 - Resolução da questão e análise das asserções

- Para resolver a questão, devemos compreender o circuito da figura do enunciado. Sabemos que as entradas de seleção A e B geram um total de 4 combinações possíveis: $00_{(2)}$,$10_{(2)}$,$01_{(2)}$ e $11_{(2)}$.
- Como o elevador só deve ser acionado caso seja chamado em mais de dois andares (ou seja, em três ou quatro andares) e caso A e B sejam iguais a zero, o elevador não deve ser habilitado, independentemente dos valores de C e D. Assim, se $AB=00_{(2)}$ devemos ter como saída do multiplexador zero sempre. Isso corresponde ao zero ligado à primeira porta de entrada do multiplexador. Se tanto A quanto B estiverem acionados (ou seja, se $AB=11_{(2)}$), basta apenas um dos andares C ou D ser ativado (ou ambos) para que o elevador seja acionado.

Exercício 1 - Resolução da questão e análise das asserções

- É por isso que a última porta de entrada do multiplexador (a porta selecionada caso $AB=11_{(2)}$) deve ser resultado de um OU entre as entradas C e D. Finalmente, se apenas uma das entradas de controle (A ou B) estiver com nível um e a outra estiver com o nível zero (não importa qual delas), é necessário que ambas as entradas C e D estejam selecionadas para que o elevador seja habilitado, já que apenas assim teremos três andares chamando o elevador.
- Dessa forma, podemos afirmar que o circuito atende às especificações do projeto, o que torna a asserção I falsa. Além disso, a entrada superior do multiplexador igual a zero afeta a saída somente se $AB=00_{(2)}$, e não sempre, como diz a asserção II, que, dessa forma, também é falsa.
- Alternativa correta: E.

Exercício 2 – Linguagens de programação - PROLOG

- Diferentes implementações da linguagem de programação PROLOG permitem predicados com parâmetros, aceitam as operações de conjunção e disjunção lógica, utilizando os símbolos vírgula (conjunção) e ponto e vírgula (disjunção), e a negação lógica com opredicado not.
- Considere que um programador propôs as cláusulas mostradas a seguir, definidas em uma linguagem de programação como PROLOG, como parte da verificação de critérios para seleção de candidatos a uma chapa de presidente e vice-presidente de uma empresa. Estas cláusulas apresentam as premissas para chegar às conclusões "selecionados", "desconsiderados" e "descartado", a partir da possibilidade da existência de fatos ou regras com o identificador superior.

Exercício 2 – Linguagens de programação - PROLOG

```
superior(jorge).
superior(ana).
selecionados(P,Q) :- superior(P), superior(Q).
desconsiderados(P,Q) :- not(superior(P)); not(superior(Q)).
descartado(P) :- not(superior(P)).
```

Exercício 2 – Linguagens de programação - PROLOG

- Considerando apenas as colocações e cláusulas acima e a hipótese de mundo fechado (closed world assumption), avalie as afirmações a seguir.

I. Para todos os valores dos parâmetros P e Q, o predicado "selecionados" retornará o valor lógico falso.
II. Para todos os valores de P e Q, os predicados "selecionados" e "desconsiderados" retornarão valores lógicos diferentes.
III. A conjunção dos predicados "selecionados" e "desconsiderados", para quaisquer valores de P e Q, retornará um valor lógico verdadeiro.
IV. Para qualquer valor do parâmetro P, o predicado "descartado" retornará um valor verdadeiro.
V. A disjunção dos predicados "selecionados" e "desconsiderados", para quaisquer valores de P e Q, retornará um valor lógico verdadeiro.

É correto apenas o que se afirma em
A. I e II. B. I e III. C. II e V. D. III e IV. E. IV e V.

```
superior(jorge).
superior(ana).
selecionados(P,Q) :- superior(P), superior(Q).
desconsiderados(P,Q) :- not(superior(P)); not(superior(Q)).
descartado(P) :- not(superior(P)).
```

Questão 2 – Introdução Teórica

- **1.1. Paradigmas de linguagens de programação**
- Muitas das linguagens apresentam características semelhantes e podem ser agrupadas em paradigmas. Segundo Tucker e Noonan (2008), "um paradigma de programação é um padrão de resolução de problemas que se relaciona a um determinado gênero de programas e linguagens".
- Podemos identificar quatro paradigmas na atualidade (TUCKER e NOONAN, 2008), conforme segue.

Questão 2 – Introdução Teórica

- **Programação imperativa**: nesse paradigma, o programador lista uma série de comandos de modo explícito, essencialmente dizendo ao computador como resolver um problema. É o modelo mais comum e mais antigo. Temos, como exemplos, as linguagens C, Fortran e Perl (entre outras).
- **Programação orientada a objetos**: nesse paradigma, vários objetos comunicam-se trocando mensagens. Linguagens como Java, C# e C++ suportam esse tipo de paradigma.

Questão 2 – Introdução Teórica

- **Programação Funcional**: esse paradigma foi inspirado na ideia matemática de função. Nesse tipo de linguagem, a computação é feita pela avaliação das funções que compõem o programa, de forma similar à matemática. Alguns exemplos de linguagens funcionais são as linguagens Lisp, Haskell e Scheme.
- **Programação Lógica**: nessa linguagem, inspirada pela matemática e pela lógica formal, um programa contém um conjunto de declarações (regras) e a computação é uma consequência lógica obtida dessas afirmações. A linguagem PROLOG é um exemplo desse tipo de paradigma.

Questão 2 – Linguagem PROLOG

- Todas as linguagens de programação requerem "lógica" para que um programa seja escrito e executado, e o computador em si é uma máquina lógico-digital. Contudo, aqui o termo "lógica" refere-se ao fato de que um programa é uma coleção de declarações sobre determinado domínio de problema, e a computação é a conclusão lógica derivada dessas declarações.

- Nas linguagens lógicas, o programador está mais preocupado com o detalhamento do que o programa deve fazer do que "como" o programa deve fazer. O programador uma série de cláusulas, essencialmente afirmações sobre determinado problema. Depois, uma pergunta deve ser feita para ser respondida com base nas cláusulas do programa. O computador vai tentar deduzir a resposta com base nas cláusulas apresentadas.

Questão 2 – Linguagem PROLOG

- Temos duas abordagens: "mundo fechado" e "mundo aberto". Utilizamos a hipótese "mundo fechado" (closed world assumption), na qual apenas dizemos que algo é verdadeiro, se o pudermos provar por meio de uma cláusula ou de uma dedução. A outra possibilidade, chamada de "mundo aberto", admite que algo possa ser verdadeiro, mesmo que isso não seja passível de ser provado apenas com a informação disponível.

Questão 2 – Análise das afirmativas

- Para resolvermos esse problema, criamos uma tabela (tabela 1), contendo as cláusulas descritas no enunciado :

Tabela 1. Tabela verdade para a situação do enunciado.

P	Q	P e Q (selecionados)	P ou Q	Não P	Não Q	(Não P) ou (Não Q) (desconsiderados)
F	F	F	F	V	V	V
F	V	F	V	V	F	V
V	F	F	V	F	V	V
V	V	V	V	F	F	F

Questão 2 – Análise das afirmativas

- I – Afirmativa incorreta.
  JUSTIFICATIVA. Observe que tanto Jorge quanto Ana têm formação superior, o que significa que o predicado "selecionados" retorna verdadeiro. Dessa forma, a afirmativa está incorreta.
- II – Afirmativa correta.
  JUSTIFICATIVA. Observe que as colunas 3 e 7 sempre apresentam valores opostos: quando o valor de uma coluna é Verdadeiro (V), o valor da outra coluna é Falso (F). Assim, verificamos que os valores retornados pelos predicados serão sempre opostos.
- III – Afirmativa incorreta.
  JUSTIFICATIVA. Uma conjunção equivale ao operador lógico "E". Como os predicados "selecionados" e "desconsiderados" sempre retornam valores opostos, uma conjunção entre esses predicados equivale a fazer um "E" lógico entre uma variável e sua negação, por exemplo, "X E NÃO(X)", que sempre vai retornar o valor falso (F), e não verdadeiro (V), como é dito na afirmativa III.

```
superior(jorge).
superior(ana).
selecionados(P,Q) :- superior(P), superior(Q).
desconsiderados(P,Q) :- not(superior(P)); not(superior(Q)).
descartado(P) :- not(superior(P)).
```

Questão 2 – Análise das afirmativas

- IV – Afirmativa incorreta.
  JUSTIFICATIVA. Basta observarmos que se P= "jorge", descartado = não(V) = Falso. Logo, a afirmativa IV está incorreta.
- V – Afirmativa correta.
  JUSTIFICATIVA. Uma disjunção equivale ao operador lógico "OU". Como os predicados "selecionados" e "desconsiderados" sempre retornam valores opostos, uma disjunção entre esses predicados equivale a fazer um "OU" lógico entre uma variável e sua negação, por exemplo, "X OU NÃO(X)", que sempre vai retornar o valor lógico Verdadeiro (V).
- Alternativa correta: C.

```
superior(jorge).
superior(ana).
selecionados(P,Q) :- superior(P), superior(Q).
desconsiderados(P,Q) :- not(superior(P)); not(superior(Q)).
descartado(P) :- not(superior(P)).
```

Questão 3 – Funções - Recursão

- Qual o valor de retorno da função a seguir, caso n = 27?

```
int recursao (int n) {
        if (n <= 10) {
                return n * 2;
        }
        else {
                return recursao(recursao(n/3));
        }
}
```

A. 8.
B. 9.
C. 12.
D. 16.
E. 18.

Questão 3 – Introdução Teórica

- **Recursão**
  Quando, em um programa, temos uma função (ou procedimento) chamando a si mesma, isso é denominado mecanismo de recursão. À primeira vista, o fato de uma função chamar a si mesma pode parecer estranho, e talvez até mesmo errado: se uma função chama a si mesma de forma contínua, quando esse processo termina?
- Ao criar uma função recursiva, o programador deve ter o cuidado de evitar situações nas quais o programa nunca termine, com uma função chamando a si mesma sem nunca ocorrer um critério de parada. Por exemplo, na função do enunciado, quando n é menor do que 10 ou igual a 10, a função retorna o dobro do valor de n.
- Para casos maiores, a função chama a si mesma com o valor de n/3. Dessa forma, há uma condição na qual ocorre recursão e outra condição na qual a função retorna algum valor.
  Em um programa bem comportado, sempre deve haver interrupção do processo de recursão. A função retorna a um valor que vai ser utilizado em cada chamada anterior da função. Programas bem comportados devem levar um tempo finito para a sua execução.

Questão 3 – Introdução Teórica

- Um cuidado que se deve ter ao utilizar recursão, mesmo quando não ocorre uma
  situação com infinitas chamadas, é evitar que o número de chamadas seja muito grande, para que não ocorra estouro de pilha: cada chamada a uma função implica na criação de um item a mais em uma região da memória chamada pilha de execução; se o número de elementos na pilha de execução crescer demasiadamente, a pilha pode estourar, levando ao fim indesejado da execução do programa.
- Existem algumas técnicas de otimização que podem evitar situações de estouro de pilha. Uma das mais utilizadas é a chamada de recursão final própria (ou, no inglês, tail recursion). Nesse caso, uma chamada pode ser feita sem a necessidade de se adicionar um quadro na pilha de chamada, evitando o seu crescimento desenfreado.

Questão 3 – Resolução

- Para a resolução da função dada na questão, podemos fazer um teste de mesa. Vamos enumerar as linhas do fragmento de programa:

```
1) int recursao (int n) {
2)       if (n <= 10) {
3)               return n * 2;
4)       }
5)       else {
6)               return recursao(recursao(n/3));
7)       }
8) }
```

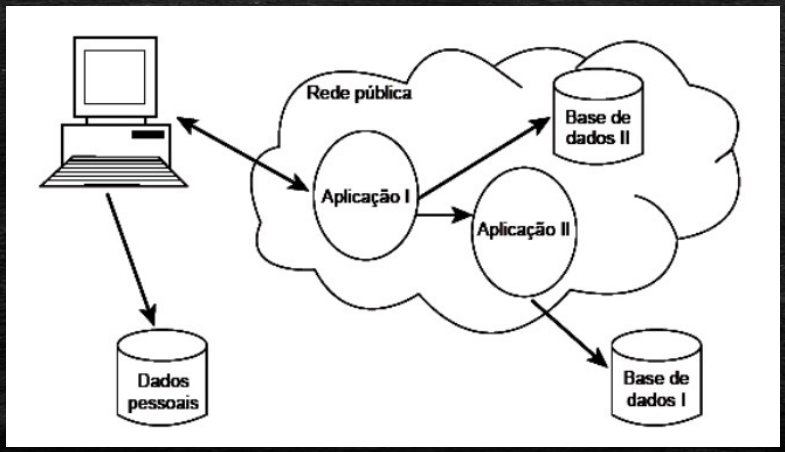
Quadro 1. Teste de mesa do problema do enunciado.

LINHA	n	Retorno
6	27	recursao(recursao(9))
3	9	18
6	27	recursao(18)
6	18	recursao(recursao(6))
3	6	12
6	18	recursao(12)
6	12	recursao(recursao(4))
3	4	8
6	12	recursao(8)
3	8	16

Dessa forma, o valor retornado para recursão (27) é 16.
Alternativa correta: D.

Questão 4 – Sistemas Distribuídos

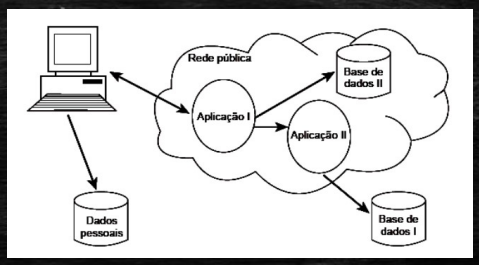
- Considere o arranjo computacional apresentado a seguir:



Questão 4 – Sistemas Distribuídos

- A característica fundamental esperada para tais sistemas de modo a ter o menor impacto sobre a experiência do usuário final é:

A. a transparência entre as entidades do sistema.
B. a linguagem de programação orientada a eventos.
C. o hardware com elevada taxa de processamento de dados.
D. a base de dados deve estar localizada no mesmo espaço fixo.
E. a independência quanto à disponibilidade de conexão à rede de comunicação de dados.



Questão 4 – Introdução Teórica

- **Sistemas distribuídos**
  Até a década de 1970, computadores eram recursos caros, sofisticados e centralizados. O custo de produção de uma única máquina era tão elevado que, em geral, apenas governos ou grandes corporações tinham condições de adquirir esses equipamentos.
- O uso dos computadores era feito por meio de lotes (batches), submetidos de uma só vez ao computador, usualmente na forma de um baralho de cartões perfurados.
- Ao final da execução, o usuário retirava os resultados em algum escaninho, ou seja, nenhum usuário acessava o computador diretamente: todos os programas eram executados e controlados pelo "operador do sistema", que submetia os lotes ao computador central, controlava a sua execução, trazia os resultados impressos e colocava-os no escaninho público.

Questão 4 – Introdução Teórica

- Com a evolução do uso dos sistemas em lotes, surgiram os sistemas interativos. Neles, o usuário acessa o computador e controla diretamente a execução do programa e de todos os periféricos ligados à máquina. Nas primeiras versões, os sistemas interativos eram monousuários, ou seja, apenas um único usuário podia utilizar o computador de cada vez.
- Posteriormente, devido ao elevado custo dessas máquinas e à crescente demanda por serviços computacionais, surgiu o "compartilhamento temporal", ou, em inglês, o timesharing. Nesse caso, uma única máquina era diretamente acessada por meio de um conjunto de terminais e os recursos eram distribuídos pelos diversos usuários, cada qual em seu terminal. Dessa forma, para cada usuário, parecia que havia um computador "só para ele". No entanto, na realidade, havia apenas um único computador rodando subdivisões independentes para cada usuário. Assim, uma máquina era capaz de atender a mais de um usuário ao mesmo tempo. Esses sistemas são chamados de sistemas multiusuários.

Questão 4 – Introdução Teórica

- Com o aumento do número de máquinas e de usuários, houve a necessidade de interligar essas "ilhas de processamento" e surgiram as primeiras redes de computadores. Com o desenvolvimento das redes de computadores e dos sistemas operacionais capazes de executar mais de um processo simultaneamente (sistemas operacionais multitarefa) e, também, com a capacidade de esses sistemas darem suporte a mais de um usuário simultaneamente (sistemas multiusuários), surgiu a possibilidade do compartilhamento de recursos tanto local quanto remotamente.
- A comunicação entre computadores remotos possibilitou que sistemas fossem desenvolvidos de forma distribuída, tirando-se proveito não apenas dos recursos locais de uma única máquina, mas de várias máquinas remotas, postas em contato por meio da rede. O resultado desse procedimento chama-se sistema distribuído, o qual é composto por recursos que executam, simultaneamente, em várias máquinas diferentes e se comunicam por meio da troca de mensagens, que são transportadas pela rede.

Questão 4 – Análise das alternativas

A – Alternativa correta.
JUSTIFICATIVA. Em um sistema distribuído, no qual diferentes aplicações e bases de dados comunicam-se de forma complexa, é importante que existam interfaces bem definidas entre as diversas entidades do sistema. Além disso, essas interfaces precisam ser transparentes, ou seja, devem permitir que as entidades se comuniquem sem expor a implementação interna. Dessa forma, a realização de alterações internas nessas entidades, como a correção de defeitos ou a execução de melhorias, não afetam a interface de acesso e fazem com que as aplicações que dependam dessas entidades continuem funcionando normalmente.

B – Alternativa incorreta.
JUSTIFICATIVA. Ainda que o uso de uma linguagem orientada a eventos seja útil na construção desse tipo de sistema, sua simples utilização não garante que o usuário final tenha a experiência adequada.

Questão 4 – Análise das alternativas

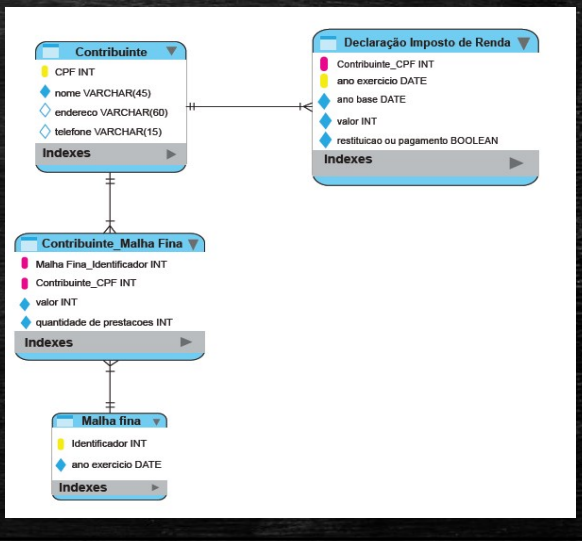
C – Alternativa incorreta.
JUSTIFICATIVA. Uma vez que as aplicações rodem em um sistema distribuído e se comuniquem por meio de uma rede pública, provavelmente há um ambiente bastante heterogêneo com relação ao hardware utilizado. Ainda há a influência da velocidade da rede pública, que está fora do controle da aplicação.

D – Alternativa incorreta.
JUSTIFICATIVA. A base de dados não precisa estar fisicamente próxima do local em que as aplicações são executadas, uma vez que elas podem ser acessadas remotamente pela rede.

E – Alternativa incorreta.
JUSTIFICATIVA. Um sistema distribuído, como o apresentado na figura do enunciado, certamente depende do acesso à rede para o seu funcionamento, uma vez que as aplicações rodam em diferentes máquinas e em diferentes locais (mas acessíveis via rede pública).

Questão 5 – Banco de Dados

- O modelo lógico de dados fornece uma visão da maneira como os dados serão armazenados. A figura a seguir representa o modelo lógico de um ambiente observado em um escritório contábil:



Questão 5 – Banco de Dados

Em relação ao modelo, avalie as afirmativas a seguir.

I. A entidade Declaração Imposto de Renda é uma entidade fraca.

II. O relacionamento entre Contribuinte e Malha Fina é do tipo N:M (muitos para muitos).

III. O atributo CPF da entidade Contribuinte tem a função de chave estrangeira na entidade Declaração Imposto de Renda e no relacionamento Contribuinte_MalhaFina.

IV. A entidade Malha Fina não possui chave primária somente chave estrangeira.

V. O relacionamento Contribuinte_MalhaFina é um relacionamento ternário.

É correto apenas o que se afirma em:

a) I, II e III.

b) I, II e IV.

c) I, IV e V.

d) II, III e V.

e) III, IV e V.

Questão 5 – Análise das Alternativas

- I – Afirmativa correta.

  JUSTIFICATIVA. A entidade "Declaração de Imposto de Renda" depende da existência da entidade "Contribuinte" e tem como discriminador o atributo "ano exercício".

- II – Afirmativa correta.

  JUSTIFICATIVA. A tabela "Contribuinte_Malha Fina" faz o papel de uma tabela de ligação entre as entidades "Contribuinte" e "Malha Fina", justamente porque o relacionamento é do tipo N:M (muitos para muitos).

- III – Afirmativa correta.

  JUSTIFICATIVA. Tanto na entidade "Contribuinte_Malha Fina" quanto na entidade "Declaração Imposto de Renda", é necessário o atributo CPF, que aponta para a entidade "Contribuinte". Logo, esse atributo é uma chave estrangeira.

Questão 5 – Análise das Alternativas

- IV – Afirmativa incorreta.

  JUSTIFICATIVA. O atributo "Identificador" é a chave primária da entidade "Malha Fina".

- V – Afirmativa incorreta.

  JUSTIFICATIVA. O relacionamento "Contribuinte_Malha Fina" é um relacionamento entre duas entidades, as entidades "Contribuinte" e "Malha Fina" e, portanto, é um relacionamento binário.

- Alternativa correta: A.

Questão 6 - Recursividade

- "Uma função é denominada recursiva quando ela é chamada novamente dentro de seu corpo. Implementações recursivas tendem a ser menos eficientes, porém facilitam a codificação e seu entendimento". CELES, W.; CERQUEIRA, R.; RANGEL, J. L. Introdução a estrutura de dados. Rio de Janeiro, 2004 (com adaptações).
- Considere a função recursiva f(), a qual foi escrita em linguagem C:

```
1 int f( int v[], int n){
2     if(n == 0)
3        return 0;
4     else{
5        int s;
6        s = f(v, n-1);
7        if( v[n-1] > 0 ) s = s + v[n-1];
8            return s;
9     }
10 }
```

Questão 6 - Recursividade

- Suponha que a função f() é acionada com os seguintes parâmetros de entrada: f({2,-4,7,0,-1,4},6);
- Nesse caso, o valor de retorno da função f() será:
  A. 8.
  B. 10.
  C. 13.
  D. 15.
  E. 18.

```
1 int f( int v[], int n){
2     if(n == 0)
3        return 0;
4     else{
5        int s;
6        s = f(v, n-1);
7        if( v[n-1] > 0 ) s = s + v[n-1];
8            return s;
9     }
10 }
```

Questão 6 – Introdução Teórica:

- **Recursividade**
  Em um programa computacional, quando temos uma função (ou procedimento) chamando a si mesma, chamamos esse mecanismo de recursividade. À primeira vista, o fato de uma função chamar a si mesma pode parecer estranho e talvez até mesmo errado: se uma função chama a si mesma de forma contínua, quando o processo irá parar?
- Ao criar uma função recursiva, o programador deve evitar situações em que o programa nunca termine, com uma função chamando a si mesma sem nunca estabelecer um critério de parada. Dessa forma, deve existir uma condição na qual ocorra recursividade e outra condição na qual a função retorne algum valor.

Questão 6 – Introdução Teórica:

- Em um programa bem comportado, sempre deve haver um momento em que o processo de recursividade é interrompido. A função retorna a um valor que vai ser utilizado em cada chamada anterior da função. Normalmente, queremos trabalhar com programas que devem levar um tempo finito para essa execução e, preferencialmente, o menor tempo possível.
- Outro cuidado que devemos tomar ao utilizarmos recursividade, mesmo quando não temos uma situação com infinitas chamadas, é que, se o número de chamadas for muito grande, pode ocorrer uma situação chamada de estouro de pilha. Cada chamada para uma função implica a criação de um item a mais em uma região da memória chamada, a pilha de execução.

Questão 6 – Introdução Teórica:

- Se o número de elementos na pilha de execução crescer demasiadamente, a pilha pode estourar, levando ao fim da execução do programa. Esse problema não é exclusivo de programas que utilizam recursividade, mas é mais comum nesses casos, pois um número excessivo de chamadas a uma função pode levar ao estouro da pilha de execução.
- Existem algumas técnicas de otimização que podem evitar situações de estouro de pilha. Uma das mais utilizadas é a chamada de recursividade final própria, ou, em inglês, tail recursion. Nesse caso, uma chamada pode ser feita sem a necessidade de se adicionar um quadro na pilha de chamadas, o que evita seu crescimento desenfreado.

Questão 6 – Análise da Questão:

- Para a resolução do problema, podemos construir uma tabela na qual simulamos a execução do programa. Observe que essa tabela será construída na ordem em que a função f(v,n) retorna aos valores, e não na ordem em que ela é chamada. Isso acontece porque essa é uma função recursiva e o primeiro valor a ser retornado é 0, na linha 3, quando n é igual a 0. A função f(v[0], 0) retorna a s=0 na linha 6. Como v[0]=2>0, sabemos que s=s+2=0+2=2. Sendo assim, f(v,1)=2. Esse valor é novamente retornado à linha 6. Com f(v[1],2), mas v[1]=-4<0, portanto, f(v,2)=2.
- Esse processo é repetido até que se chegue ao valor f(v,6)=13, conforme tabela 1.

Questão 6 – Análise da Questão:

Tabela 1. Resultado de f(v,n) para a execução do programa.

N	f(v,n)
0	0
1	2
2	2
3	9
4	9
5	9
6	13

Alternativa correta: C.

Questão 7 – Listas Ligadas

- Para fins estatísticos, uma empresa precisa armazenar os trajetos que seus representantes comerciais percorrem entre pontos de venda. É importante que para cada local visitado sejam armazenadas, além da informação do próprio local, o local de origem do representante (ponto de venda anterior), o local de destino (ponto de venda posterior), as distâncias percorridas e os tempos de viagem. Esse procedimento permite que estes trajetos possam ser analisados, de forma rápida, do local de origem ao local de destino, bem como no sentido inverso, do local de destino (final do trajeto) ao local de origem (início do trajeto).

Questão 7 – Listas Ligadas

- O analista responsável pelo sistema, que utilizará os dados armazenados e produzirá os relatórios estatísticos, projetou o seguinte esboço de uma classe que representa um ponto de venda:

```
public class Local {
     private String nome_estabelecimento;
     private String endereco;
     private Local origem;
     private Local destino;
     private float distancia_origem;
     private float tempo_origem;
}
```

Questão 7 – Listas Ligadas

- A respeito do esboço da classe, avalie as afirmativas a seguir.

1. O esboço acima representa uma lista duplamente encadeada.
2. A utilização de um nó de uma estrutura de dados do tipo árvore de busca multivias de grau três seria a solução ideal para o problema porque providenciaria a economia de recursos de memória e de disco.
3. A utilização de uma árvore de pesquisa binária para a solução do problema é normal, desde que o atributo de ordenação da árvore seja distancia_origem.

- É correto o que se afirma em

a) I, apenas.
b) II, apenas.
c) I e III, apenas.
d) II e III, apenas.
e) I, II e III.


```
public class Local {
     private String nome_estabelecimento;
     private String endereco;
     private Local origem;
     private Local destino;
     private float distancia_origem;
     private float tempo_origem;
}
```

Questão 7 - Introdução Teórica – Listas Ligadas

- A construção de um programa é profundamente influenciada pela escolha das estruturas de dados que dão suporte ao seu funcionamento. O livro intitulado *Algoritmos + Estruturas de Dados = Programas*(WIRTH, 1976) mostra como esses dois conceitos (algoritmos e estruturas de dados) estão relacionados e a importância deles para o trabalho do programador.
- Com relação às estruturas de dados, podemos observar as listas (estruturas de dados lineares) e as árvores (estruturas de dados não lineares).
- No seu formato mais simples, uma lista ligada pode ser vista como um conjunto de registros que apresentam ligações entre si, pelo menos em um sentido (também chamada de lista encadeada simples), como mostrado na figura 1.

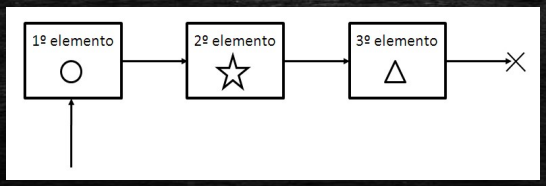


Figura 1. Representação gráfica de uma lista ligada simples com três elementos.

Questão 7 - Introdução Teórica – Listas Ligadas

- Cada membro da lista deve ter uma referência (ou ponteiro) para, pelo menos, outro elemento. Dessa forma, os elementos da lista estão "ligados" por referências. Isso permite que esses elementos sejam acessados de acordo com uma interface bem definida e de acordo com as referências.
- O programador deve estar atento à forma como os elementos devem ser inseridos na lista: no início, no fim ou em algum ponto intermediário. Em alguns casos, pode ser necessário impor um critério de inserção, a fim de garantir que a lista apresente uma ordem particular.
- Um exemplo bastante utilizado de lista é o chamado "lista duplamente ligada" (ou "lista duplamente encadeada"). Nesse caso, cada elemento da lista tem uma referência tanto para o elemento seguinte quanto para o elemento anterior, permitindo que o programa percorra a lista em dois sentidos.

Questão 7 - Introdução Teórica – Listas Ligadas

- Caso ocorra a inserção de um elemento novo na lista, o programador deve atualizar ambas as referências (exceto no caso de inserção nas extremidades).
- Além das estruturas de dados lineares, como as listas ligadas, existem também estruturas de dados não lineares, como as árvores. Árvores são utilizadas para representar dados de natureza hierárquica (CELES, CERQUEIRA e RANGEL, 2004).
- Tipicamente, em uma estrutura de dados linear, cada elemento apresenta uma ligação para, no máximo, outros dois elementos (o elemento posterior e o elemento anterior). Dessa forma, ao percorrermos esse tipo de estrutura, obtemos um caminho tipicamente linear. Em estruturas de dados nãolineares, podem existir várias opções diferentes de percursos. Uma das estruturas de dados não lineares mais importantes corresponde às árvores.

Questão 7 - Introdução Teórica – Listas Ligadas

- Existem diversos tipos de árvores, construídos com diferentes finalidades, como as árvores binárias e as árvores B. Em uma árvore binária, cada elemento tem, além de uma referência para o nó pai, até duas referências para os nós filhos.
- Formalmente, uma árvore pode ser definida de forma recursiva como um conjunto finito *T* de um ou mais nós com as características a seguir (KNUTH, 1997).
  - Existe um nó especial chamado de raiz da árvore.
  - Os nós restantes (excluindo o nó raiz) são particionados em $m \geq 0$ conjuntos $T_1, T_2, \ldots, T_m$ e cada um deles também é uma árvore. As árvores $T_1, T_2, \ldots, T_m$ são chamadas de subárvores da raiz.

Questão 7 – Análise das afirmativas

- I - Afirmativa correta.

  JUSTIFICATIVA. A estrutura apresentada tem duas referências para objetos da mesma classe (a classe Local). Logo, trata-se de uma lista duplamente encadeada.

- II - Afirmativa incorreta.

  JUSTIFICATIVA. Uma árvore de busca multivias (também chamada de árvore B) é uma forma de organização de árvore especialmente projetada para grandes volumes de dados, nos casos em que é impossível armazenar todos esses dados na memória principal do computador. Nesses casos, o computador tem de utilizar alguma forma de armazenamento secundário, que, normalmente, apresenta tempos de acesso muito mais elevados do que os da memória principal.

- III - Afirmativa incorreta.

  JUSTIFICATIVA. Não é necessária a utilização de uma árvore para a resolução desse problema, uma vez que uma lista duplamente encadeada resolve o problema de forma eficiente e correta.

- Alternativa correta: A.

Questão 8 – Árvores Binárias

- Existem várias maneiras de se percorrer uma árvore binária. A função a seguir, escrita em pseudocódigo, percorre uma árvore na ordem esquerda-raiz-direita, conhecida por varredura e-r-d recursiva.
- A função erd() recebe por parâmetro a raiz r de uma árvore e faz uso de seus elementos esq, dir e cont, que representam, respectivamente, ponteiros para uma subárvore à esquerda de r, uma subárvore à direita de r e o conteúdo de r.

```
função erd(árvore r)
{
     se(r != NULO)
     {
          erd(r->esq);
          escreva(r->conteúdo);
          erd(r->dir);
     }
}
```

Questão 8 – Árvores Binárias

- Considere a Árvore Binária a seguir:
- A sequência correta de exibição do conteúdo da árvore utilizando a função erd() é:

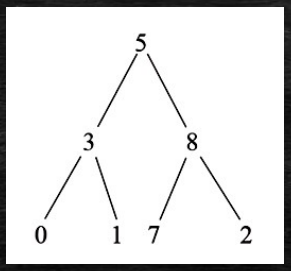
a) 5,3,8,0,1,7,2.

b) 0,1,7,2,3,8,5.

c) 0,3,5,1,7,8,2.

d) 0,3,1,5,7,8,2.

e) 2,7,8,5,0,3,1.



Questão 8 – Árvores Binárias - Teoria

- Listas ligadas, pilhas e vetores são estruturas de dados muito úteis para representação de vários tipos de informações em programas computacionais. Contudo, nem sempre conseguimos representar informações utilizando esses tipos de estruturas. Isso ocorre à medida que o relacionamento entre os nós começa a se tornar mais complexo, com mais possibilidades de interligação.
- Uma das formas de estrutura de dados mais comuns para a representação de informações hierárquicas é a árvore (CELES, CERQUEIRA e RANGEL, 2004). Na computação, uma árvore corresponde a uma estrutura que contém um nó raiz (desenhado no topo) e uma série de nós filhos (que correspondem aos ramos).
- Existem vários tipos de árvores, cada uma com uma finalidade específica. Um dos tipos mais comuns e úteis é a chamada árvore binária. Nesse caso, cada nó pode ter zero, um ou dois nós filhos. Os últimos elementos da árvore, normalmente desenhados na sua porção inferior, são chamados de nós folhas.

Questão 8 – Árvores Binárias - Teoria

- Uma das operações importantes que sempre devemos ser capazes de executar em uma estrutura de dados é chamada de percurso. Por exemplo, frequentemente queremos percorrer uma lista, visitando cada um dos seus elementos. Ou então percorrer um vetor, lendo cada um dos seus elementos ou fazendo outra operação com esses elementos. Também podemos percorrer uma árvore binária visitando cada um dos seus elementos.
- É conveniente utilizar funções recursivas para situações em que queremos percorrer estruturas de dados como árvores binárias e listas ligadas. Devido ao fato de essas estruturas terem referências (ou ponteiros) para outros elementos da mesma classe (ou tipo), a utilização de funções que seguem essas referências é bastante conveniente.

Questão 8 – Análise da questão

- Para resolvermos a questão, enumeramos três pontos importantes da função "erd", mostrados na figura 1. Devemos observar que um número só é impresso quando a função "escreva" é chamada. Quando r for NULO, a função "erd" retorna sem efetuar nenhuma ação. Caso contrário, a função "erd" é chamada de forma recursiva nos pontos 1 e 3.
- Para que a função "escreva" seja chamada, a função "erd" chamada no ponto 1 deve retornar, o que ocorre apenas quando r for igual a NULO. Logo, a primeira impressão deve ser a de um dos nós folhas da árvore. Nós folhas são os últimos nós de uma árvore computacional. Observe que, em computação, a árvore é desenhada "de cabeça para baixo", com a raiz no topo do desenho e as folhas na parte de baixo.

Questão 8 – Análise da questão

```
função erd( árvore r)
{
      se( r != NULO )
      {
         (1) erd( r->esq);
         (2) escreva(r->conteúdo);
         (3) erd(r->dir);
      }
}
```

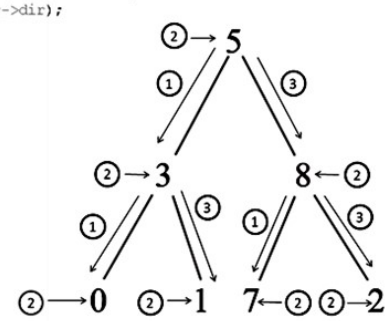


Figura 1. Representação gráfica de uma árvore binária ao lado da função "erd" utilizada para percorrer os elementos de uma árvore.

Questão 8 – Análise da questão

- Na figura 2, temos o desenho da mesma árvore do enunciado, com a ordem de impressão dos elementos dentro de quadrados ao lado dos nós da árvore. Compare essa figura e a figura 1 e observe os sentidos das setas. O número dentro das circunferências na figura 1 mostra o ponto no código em que o programa toma determinada decisão. Observe que:
  - no ponto 1, sempre percorremos o ramo esquerdo da árvore;
  - no ponto 2, sempre imprimimos um elemento;
  - no ponto 3, sempre percorremos o ramo direito.
  - **Alternativa correta: D.**

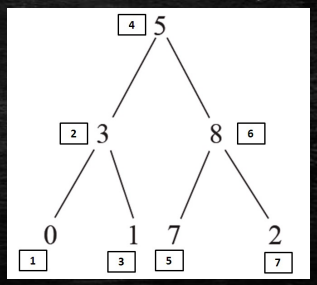
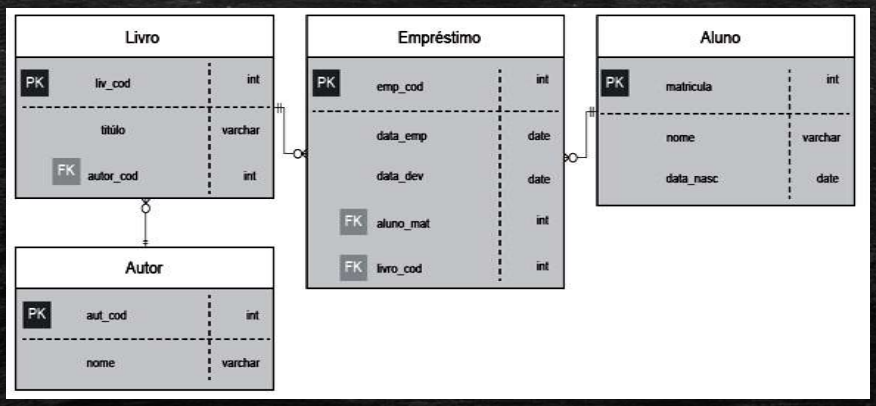


Figura 2. Ordem de impressão dos elementos da árvore binária.

Questão 9 – Bancos de Dados

- Observe o modelo de Entidade – Relacionamento ora apresentado:



Questão 9 – Bancos de Dados

- O modelo de entidade relacionamento apresentado, representa, de forma sucinta, uma solução para persistência de dados de uma biblioteca. Considerando que um livro está emprestado quando possuir um registro vinculado a ele na tabela "Empréstimo", e essa tupla não possuir valor na coluna "data_dev", o comando SQL que deve ser utilizado para listar os títulos dos livros disponíveis para empréstimo é:

A.
```
select titulo from livro
except
select l.titulo from emprestimo e inner join livro l
on e.livro_cod = l.liv_cod where e.data_dev is null
```

B.
```
select titulo from livro
union
select l.titulo from emprestimo e inner join livro l
on e.livro_cod = l.liv_cod where e.data_dev is null
```

C.
```
select titulo from livro
except
select l.titulo from emprestimo e inner join livro l
on e.livro_cod = l.liv_cod where e.data_dev is not null
```

D.
```
select titulo from livro
union select l.titulo from emprestimo e left join livro l
on e.livro_cod = l.liv_cod where e.data_dev is null
```

E.
```
select titulo from livro
except
select l.titulo from emprestimo e right join livro l
on e.livro_cod = l.liv_cod where e.data_dev is not null
```

Questão 9 – Análise das Alternativas

- O resultado desejado corresponde a "encontrar todos os livros, exceto aqueles que estão emprestados". Um livro está emprestado se está presente na tabela "Empréstimo" e não ter valor na coluna "data_dev" (ou seja, essa coluna deve conter o valor NULL). Dessa forma, para selecionarmos todos os livros que estão emprestados, fazemos o que segue:

```
select l.titulo from emprestimo e inner join livro l
on e.livro_cod = l.liv_cod where e.data_dev is null
```

Questão 9 – Análise das Alternativas

- Lembrando que o comando "inner join" retorna apenas os registros que apresentam o mesmo valor na coluna "livro_cod" da tabela "empréstimo" e da coluna "liv_cod" na tabela "livro". Além disso, observe a condição "e.data_dev is null", ou seja, apenas os registros que não apresentarem valores em "data_dev".
- Se quiséssemos selecionar todos os livros, independentemente de estarem emprestados ou não, deveríamos fazer:

```
select titulo from livro
```

Questão 9 – Análise das Alternativas

- Essa consulta vai retornar todos os títulos presentes na tabela livro. Excluiremos os registros que estão emprestados, retornados na consulta anterior. Isso pode ser feito utilizando-se o comando except, que elimina, da primeira consulta, os registros obtidos pela segunda consulta, conforme segue.

```
select titulo from livro
except
select l.titulo from emprestimo e inner join livro l
on e.livro_cod = l.liv_cod where e.data_dev is null
```

- Alternativa correta: A

Questão 10 – Bancos de Dados

- *O modelo relacional representa o banco de dados como uma coleção de relações (tabelas). Na terminologia formal do modelo relacional, uma linha é chamada de "tupla", o título da coluna é denominado "atributo" e a tabela é chamada de "relação". O tipo de dado que descreve os tipos de valores que podem aparecer em cada coluna é denominado "domínio". Um banco de dados relacional pode impor vários tipos de restrições nos dados armazenados.*

  ELMASRI, R. NAVATHE, S.B. Sistema de Banco de Dados: Fundamentos e Aplicações. Rio de Janeiro: LTC, 2002.

Questão 10 – Bancos de Dados

- Restrições que permitem controlar situações como, por exemplo, "o salário de um empregado não deve exceder o salário do supervisor do empregado" e utilizam mecanismos chamados *triggers* (gatilhos) na sua implementação, são do tipo:

a) restrições de domínio.

b) restrições de unicidade.

c) restrições de integridade referencial.

d) restrições de integridade da entidade.

e) restrições de integridade semântica.

Questão 10 – Análise das alternativas

- A – Alternativa incorreta.

  JUSTIFICATIVA. A restrição descrita no enunciado não limita o tipo de dados de um atributo e, portanto, não é uma restrição de domínio.

- B – Alternativa incorreta.

  JUSTIFICATIVA. A restrição descrita no enunciado não garante unicidade.

- C – Alternativa incorreta.

  JUSTIFICATIVA. A restrição descrita no enunciado não menciona tipo de relacionamento entre entidades e, portanto, não pode ser vista como uma restrição de integridade referencial.

Questão 10 – Análise das alternativas

- D – Alternativaincorreta.

  JUSTIFICATIVA. A restrição descrita no enunciado não garante que mais de um empregado tenha os valores iguais em uma tabela, apenas garante que o empregado ganhe menos do que o seu supervisor. Por exemplo, dois funcionários poderiam ter o mesmo salário, desde que esse salário fosse menor do que o salário do supervisor.

- E – Alternativa correta.

  JUSTIFICATIVA. A restrição descrita no enunciado é uma regra de negócio. Logo, é uma restrição de integridade semântica.

Até a Próxima!!!