

# **COMPUTAÇÃO**

## **QUESTÃO DISCURSIVA 1**

### **Padrão de resposta**

O estudante deve ser capaz de apontar algumas vantagens dentre as seguintes, quanto à modalidade EaD:

- (i) flexibilidade de horário e de local, pois o aluno estabelece o seu ritmo de estudo;
- (ii) valor do curso, em geral, é mais baixo que do ensino presencial;
- (iii) capilaridade ou possibilidade de acesso em locais não atendidos pelo ensino presencial;
- (iv) democratização de acesso à educação, pois atende a um público maior e mais variado que os cursos presenciais; além de contribuir para o desenvolvimento local e regional;
- (v) troca de experiência e conhecimento entre os participantes, sobretudo quando dificilmente de forma presencial isso seria possível (exemplo, de pontos geográficos longínquos);
- (vi) incentivo à educação permanente em virtude da significativa diversidade de cursos e de níveis de ensino;
- (vii) inclusão digital, permitindo a familiarização com as mais diversas tecnologias;
- (viii) aperfeiçoamento/formação pessoal e profissional de pessoas que, por distintos motivos, não poderiam frequentar as escolas regulares;
- (ix) formação/qualificação/habilitação de professores, suprimindo demandas em vastas áreas do país;
- (x) inclusão de pessoas com comprometimento motor reduzindo os deslocamentos diários.

## **QUESTÃO DISCURSIVA 2**

### **Padrão de resposta**

O estudante deve abordar em seu texto:

- identificação e análise das desigualdades sociais acentuadas pelo analfabetismo, demonstrando capacidade de examinar e interpretar criticamente o quadro atual da educação com ênfase no analfabetismo;
- abordagem do analfabetismo numa perspectiva crítica, participativa, apontando agentes sociais e alternativas que viabilizem a realização de esforços para sua superação, estabelecendo relação entre o analfabetismo e a dificuldade para a obtenção de emprego;
- indicação de avanços e deficiências de políticas e de programas de erradicação do analfabetismo, assinalando iniciativas realizadas ao longo do período tratado e seus resultados, expressando que estas ações, embora importantes para a eliminação do analfabetismo, ainda se mostram insuficientes.

### QUESTÃO DISCURSIVA 3

#### Padrão de resposta

Algoritmo iterativo

```
int fibonacci(n) {
    prevFib ← 0,
    currFib ← 1
    if n == 1
        return 0
    if n == 2
        return 1
    for i ← 1 to n - 2 /* repetir n-2 vezes */ {
        temp ← prevFib + currFib
        prevFib ← currFib
        currFib ← temp
    }

    return currFib
}
```

Algoritmo recursivo

```
int fibonacci(n) {

    if n == 1
        return 0
    if n == 2
        return 1

    else

        return fibonacci(n-1) + fibonacci(n-2)

}
```

Discussão:

A solução recursiva clássica possui a vantagem de ser implementada diretamente a partir da definição do problema, mas tem a grande desvantagem de possuir uma ordem de complexidade exponencial. A versão iterativa tem complexidade linear o que a torna mais vantajosa em termos de eficiência, mas exige mais atenção na implementação.

#### QUESTÃO DISCURSIVA 4

##### Padrão de resposta

a) registro nodo com campos: chave do tipo inteiro; esq e dir do tipo apontadores para registro nodo.

Qualquer notação em português estruturado, de forma imperativa ou orientada a objetos deve ser considerada, assim como em uma linguagem de alto nível como o Pascal, C e Java. O importante é a presença dos campos sublinhados e do uso de apontadores ou autoreferências.

b) função CriaABP(v: vetor de inteiros; i, j: inteiros) retorna apontador para registro nodo // i, j são os índices inicial e final do vetor

```
| Cria novo nodo apontado por p, o qual deve ser uma variável local  
  
| pos = (i + j) / 2 // determina a posição central do vetor  
  
| p->chave = v[pos] // guarda o elemento v[pos] no novo nodo criado  
  
| se i < j entao // ainda não se está no nível das folhas  
  
| | p->esq = CriaABP(v, i, pos-1) // chama recursivamente para a sub-árvore da  
| esquerda  
  
| | p->dir = CriaABP(v, pos+1, j) // chama recursivamente para a sub-árvore da  
| direita  
  
| senão p->esq = p->dir = NULL // nível das folhas  
  
| retorna p
```

Chamada principal: raiz = CriaABP(v, 1, n) onde raiz aponta para o nodo raiz da árvore.

Qualquer notação em português estruturado, de forma imperativa ou orientada a objetos deve ser considerada, assim como em uma linguagem de alto nível como o Pascal, C e Java. A função deve ser recursiva e não pode usar comparações para encontrar o elemento a ser inserido, nem utilizar operações de inserção que façam comparações implicitamente. A condição de parada da recursão (nível das folhas) deve estar clara e os parâmetros para chamada recursiva devem estar corretos. Os apontadores dos nodos-folhas devem ser aterrados.

## **QUESTÃO DISCURSIVA 5**

### **Padrão de resposta**

Em cada um dos mapeamentos o participante do exame deve indicar claramente como calcular o endereço de um determinado bloco da memória principal na memória cache. Isso pode ser feito pela divisão do endereço de 32 bits em campos (Palavra, Linha, Rótulo e Conjunto) ou por uma breve descrição de como cada campo é usado, incluindo seu tamanho. A seguir são apresentadas descrições detalhadas de cada esquema, visando facilitar a correção do item. A resposta, portanto, não precisa incluir todas as informações de cada esquema, mas deve diferenciá-los claramente.

#### Mapeamento direto:

No mapeamento direto cada bloco da memória principal é mapeado em uma única posição da cache e seu endereço deve ser dividido da seguinte forma:

Rótulo Linha Palavra

13    17    2

Dois bits são usados para identificar a palavra (byte) dentro do bloco (ou linha). São necessários 17 bits para determinar em qual das 128K linhas da cache o bloco será mapeado. Os 13 bits mais significativos do endereço devem ser comparados com o rótulo da cache naquela linha para saber se aquele é o bloco atualmente mapeado.

#### Mapeamento totalmente associativo:

No mapeamento totalmente associativo cada bloco da memória principal pode ser mapeado em qualquer posição da cache e seu endereço deve ser dividido da seguinte forma:

Rótulo Palavra

30    2

Dois bits são usados para identificar a palavra dentro do bloco. Todos os demais bits (30) são usados como rótulo para identificar o bloco na memória cache.

#### Mapeamento associativo por conjunto:

No mapeamento associativo por conjunto (4 vias) cada bloco da memória principal é mapeado em um conjunto com 4 linhas e seu endereço deve ser dividido da seguinte forma:

Rótulo Conjunto Palavra

15    15    2

Dois bits são usados para identificar a palavra dentro do bloco. São necessários 15 bits para determinar em qual dos 32K conjuntos o bloco será mapeado. Os 15 bits mais significativos do endereço devem ser comparados com os rótulos da cache naquele conjunto para saber se o bloco está atualmente mapeado.

Vantagens e desvantagens:

O mapeamento direto é o mais simples de ser implementado e o circuito resultante é mais rápido e não requer algoritmo de substituição. Entretanto, em geral, as taxas de acertos (cache hit) são menores.

O mapeamento totalmente associativo é o que tem as maiores taxas de acerto. Entretanto, é o mais complexo dos três. Os circuitos resultantes são maiores, mais caros e mais lentos. Além disso, requer um algoritmo de substituição. Normalmente esse mapeamento é utilizado em memórias cache de pequena capacidade.

O mapeamento associativo por conjunto é uma solução de compromisso (trade-off) entre as duas opções anteriores. Tem como vantagens ser mais simples que o totalmente associativo e, em geral, mais eficiente, em termos de taxas de acerto, do que o mapeamento direto.