



PUC Minas

Pontifícia Universidade Católica de Minas Gerais
Engenharia de Software
Algoritmos e Estruturas de Dados 1
Prova 3 - 35 pontos

34

ALUNO: Luca Ferrari Azolim

Questão 1 - 3 pontos

Suponha termos uma função chamada troca(&X, &Y) cuja finalidade seja trocar o conteúdo de X com o de Y, ou seja quando essa função executa o valor de X volta em Y e vice-versa. Acompanhe a execução do trecho de código abaixo e informe os valores que as variáveis A, B, C e D irão assumir ao atingir o final do código.

```
int A=30, B=20, C=10, D=40;
for (i = 1; i <= 5; i++) {
    if (i % 2 == 0)
        troca (&A, &B);
    else if (i % 3 == 0)
        troca (&B, &C);
    else if (i % 4 == 0)
        troca (&A, &C);
    else if (i % 5 == 0)
        troca (&C, &D);
    else troca (&D, &A);
}
```

O resultado correto será:

- a) A = 20, B = 20, C = 30, D = 10
- b) A = 40, B = 30, C = 20, D = 10
- c) A = 10, B = 20, C = 30, D = 40
- d) A = 40, B = 10, C = 30, D = 20

A= 10; B= 20; C= 30; D= 40;

	1	2	3	4	5
A	30	40	20	20	10
B	20	20	40	10	20
C	10	10	10	40	40
D	40	30	30	30	40

Questão 2 - 3 pontos

```
struct DADO {
    int apontador [10];
    struct DADO *vetor;
} registro[20];
```

A estrutura declarada acima pode ser corretamente descrita como:

- a) Um vetor de registros que contém um vetor de apontadores da estrutura DADO
- b) Um registro contendo um vetor de inteiros e um apontador para a estrutura DADO
- c) Um vetor de registros contendo um vetor de inteiros e um apontador para a estrutura DADO
- d) Um registro contendo um vetor de inteiros e um vetor de apontadores para a estrutura DADO

Questão 3 - 3 pontos

Relacione os conceitos existentes na primeira coluna com as características ou definições colocadas na segunda coluna.

- | | |
|-----------------|---|
| I. Registro: | (IV) Tipo Abstrato de Dado |
| II. Arquivo: | (I) Coleção de Dados Heterogêneos |
| III. Apontador: | (V) Cadeia de Caracteres |
| IV. Classe: | (II) Armazenamento de dados permanente |
| V. string: | (III) Contém endereço de outra variável |

Assinale a opção que representa a sequência correta a ser colocada nos parêntesis

- a) IV, I, V, II, III
- b) II, IV, I, V, III
- c) IV, V, III, II, I
- d) II, IV, V, III, I

Questão 4 - 3 pontos

O que faz o seguinte procedimento?

```
void XYZ( arquivo *vetor, int n)
{
    int i=0;
    FILE *arq;
    arq = fopen("cotacao_dolar.dat", "wb");

    for(i=0; i<n; i++) {
        fwrite( &vetor[i], sizeof(arquivo), 1, arq);
    }
    fclose(arq);
}
```

- a) Recebe um ponteiro para um vetor do tipo arquivo, abre o arquivo "cotacao_dolar.dat" como leitura e carrega os dados no vetor passado como parâmetro.
- b) Recebe um vetor do tipo inteiro, abre o arquivo "cotacao_dolar.dat" como leitura e carrega os campos no vetor.
- c) Recebe um ponteiro para um vetor do tipo arquivo, abre o arquivo "cotacao_dolar.dat" como escrita e carrega os dados no vetor passado como parâmetro
- d) Recebe um ponteiro para um vetor do tipo arquivo, cria o arquivo "cotacao_dolar.dat" como escrita e grava os dados do vetor passado como parâmetro.

Questão 5 - 3 pontos

Com relação à Programação Orientada a Objetos, analise as afirmações a seguir.

- I. Uma classe derivada (ou subclasse) herda atributos e métodos definidos em sua classe-pai (superclasse), podendo sobrepor métodos da classe-pai e/ou criar novos atributos e métodos conforme suas necessidades específicas.
- II. Um método sobrecarregado é aquele que existe em uma classe derivada e tem o mesmo nome e parâmetros ao método original implementado na superclasse. X
- III. Em C++ métodos podem ter somente três possíveis formas de visibilidade: a pública permite que métodos sejam invocados apenas por métodos externos à sua hierarquia de classe; a privada permite que sejam invocados apenas por métodos da própria classe; a protegida permite que sejam invocados apenas por métodos das suas subclasses.

Assinale:

- a) se somente a afirmativa I estiver correta.
- b) se somente a afirmativa II estiver correta.
- c) se somente as afirmativas I e III estiverem corretas.
- d) se somente as afirmativas II e III estiverem corretas.

Questão 6 - 3 pontos

Um ponteiro é muito útil quando se trata de manipular elementos existentes em memória. Sobre o uso de ponteiros pode-se dizer que:

- a) A utilização de ponteiro como parâmetro em uma função se dá através da passagem por valor. X
- b) O ponteiro é um tipo abstrato de dados, definido com o comando struct em C, que contém informações sobre agregados. O que permite o uso destes agregados em funções. X
- c) O ponteiro é um tipo que contém o endereço de memória de uma certa variável, situação que favorece a implementação de passagem por referência em funções ✓
- d) A única vantagem de utilidade ponteiro é o fato de permitir manipular tipos definidos com o uso do comando struct. X

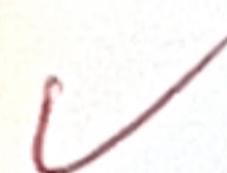
Questão 7 - 3 pontos

Sistemas lineares de equações são uma forma bastante usada nas diversas engenharias para modelar problemas práticos tais como: cálculo de correntes de circuitos com diversas malhas, ou a distribuição de carga em um sistema de vigas. Em um computador representamos os coeficientes destas equações como uma matriz A , cujas linhas representam uma equação e as colunas os coeficientes de uma mesma variável em cada equação (podendo ser zero ou não); um vetor Y que corresponde ao valor escalar normalmente presente nas equações e um vetor b que representa as variáveis a serem calculadas e deverá conter a resposta ao final. Assim a formula deste sistema se resume em: $A.b = Y$. Abaixo apresentamos um algoritmo utilizando o método de eliminação de Gauss para resolver um sistema linear qualquer.

```

    inicio // Eliminação de Gauss
        Inteiro: I, J, K, N;
        para K de 0 até N-1 faça
            para J de K+1 até N-1 faça
                A[K,J] ← A[K, J] / A[K, K];
            fimpara;
            Y[K] ← B[K] / A[K,K];
            A[K,K] ← 1;
            para I de K+1 até N-1 faça
                para J de K+1 até N-1 faça
                    A[I, J] ← A[I, J] - A[I, K] * A[K, J];
                fimpara;
                B[I] ← B[I] - A[I, K] * Y[K];
                A[I,K] ← 0;
            fimpara;
        fimpara;
    fim.

```



Qual a codificação em C que implementa corretamente o algoritmo de Eliminação de Gauss apresentado?

a)

```

int i, j, k, n;
for (k = 0; k <= n-1; k++) {
    for (j = k+1; j <= n-1; j++) {
        A[k][j] = A[k][j] / A[k][k];
        Y[k] = B[k] / A[k][k];
        A[k][k] = 1;
        for (i = k+1; i <= n-1; i++)
            for (j = k+1; j <= n-1; j++)
                A[i][j] = A[i][j] - A[i][k] * A[k][j];
        B[i] = B[i] - A[i][k] * Y[k];
        A[i][k] = 0;
    }
}

```

b)

```

int i, j, k, n;
for (k = 0; k <= n-1; k++) {
    for (j = k+1; j <= n-1; j++)
        A[k][j] = A[k][j] / A[k][k];
    Y[k] = B[k] / A[k][k];
    A[k][k] = 1;
    for (i = k+1; i <= n-1; i++) {
        for (j = k+1; j <= n-1; j++)
            A[i][j] = A[i][j] - A[i][k] * A[k][j];
        B[i] = B[i] - A[i][k] * Y[k];
        A[i][k] = 0;
    }
}

```

c)

```

int i, j, k, n;
for (k = 0; k <= n-1; k++)
    for (j = k+1; j <= n-1; j++)
        A[k][j] = A[k][j] / A[k][k];
    Y[k] = B[k] / A[k][k];
    A[k][k] = 1;
    for (i = k+1; i <= n-1; i++)
        for (j = k+1; j <= n-1; j)
            A[i][j] = A[i][j] - A[i][k] * A[k][j];
    B[i] = B[i] - A[i][k] * Y[k];
    A[i][k] = 0;

```



d)

```

int i, j, k, n;
for (k == 0; k < n; k++) {
    for (j == k; j < n; j++)
        A[k][j] == A[k][j] / A[k][k];
    Y[k] == B[k] / A[k][k];
    A[k][k] == 1;
    for (i == k; i < n; i++) {
        for (j == k; i < n; i++)
            A[i][j] == A[i][j] - A[i][k] * A[k][j];
        B[i] == B[i] - A[i][k] * Y[k];
        A[i][k] == 0;
    }
}

```



Questão 8 - 3 pontos

Foi feita uma pesquisa com um número indeterminado de assinantes de TV a Cabo sendo perguntado: quantos eram os canais mais assistidos, quantos filmes por mês cada usuário assistia e a idade dos entrevistados. Calcular a quantidade média de canais assistidos, filmes por mês por usuário e a idade média dos usuários. Os dados terminam quando for digitada uma quantidade negativa de canais de TV

a)

```
int main()
{
    int usuarios=0, canais, idade, filmes;
    float sCanais=0, sIdade=0, sFilmes=0;

    do {
        cout << "Quantidade canais assistidos (-1 termina): ";
        cin >> canais;
        cout << "Entre com a idade do usuario: ";
        cin >> idade;
        cout << "Quantidade de filmes assistidos: ";
        cin >> filmes;
        sIdade += idade;
        sCanais += canais;
        sFilmes += filmes;
        usuarios++;
    } while (canais >= 0);
    if (usuarios > 0) {
        cout << "\nMedia de canais: " << sCanais / usuarios;
        cout << "\nMedia de idades: " << sIdade / usuarios;
        cout << "\nMedia de filmes : " << sFilmes / usuarios;
    }
}
```

b)

```
int main()
{
    int usuarios=0, canais, idade, filmes;
    float sCanais=0, sIdade=0, sFilmes=0;

    while (canais >= 0) {
        cout << "Quantidade canais assistidos: ";
        cin >> canais;
        cout << "Entre com a idade do usuario: ";
        cin >> idade;
        cout << "Quantidade de filmes assistidos: ";
        cin >> filmes;
        sIdade += idade;
        sCanais += canais;
        sFilmes += filmes;
        usuarios++;
    }
    if (usuarios > 0) {
        cout << "\nMedia de canais: " << sCanais / usuarios;
        cout << "\nMedia de idades: " << sIdade / usuarios;
        cout << "\nMedia de filmes : " << sFilmes / usuarios;
    }
}
```

c)

X

```
int main()
{
    int usuarios=0, canais, idade, filmes;
    float sCanais=0, sIdade=0, sFilmes=0;

    cout << "Quantidade canais assistidos (-1 termina): ";
    cin >> canais;
    while (canais >= 0) {
        cout << "Entre com a idade do usuario: ";
        cin >> idade;
        cout << "Quantidade de filmes assistidos: ";
        cin >> filmes;
        sIdade += idade;
        sCanais += canais;
        sFilmes += filmes;
        usuarios++;
        cout << "Quantidade canais assistidos (-1 termina): ";
        cin >> canais;
    }
    if (usuarios > 0) {
        cout << "\nMedia de canais: " << sCanais / usuarios;
        cout << "\nMedia de idades: " << sIdade / usuarios;
        cout << "\nMedia de filmes : " << sFilmes / usuarios;
    }
}
```

d)

```
int main()
{
    int usuarios=0, canais, idade, filmes;
    float sCanais=0, sIdade=0, sFilmes=0;

    for (canais = 0; canais >= 0; canais++, usuarios++) {
        cout << "Quantidade canais assistidos: ";
        cin >> canais;
        cout << "Entre com a idade do usuario: ";
        cin >> idade;
        cout << "Quantidade de filmes assistidos: ";
        cin >> filmes;
        sIdade += idade;
        sCanais += canais;
        sFilmes += filmes;
    }
    if (usuarios > 0) {
        cout << "\nMedia de canais: " << sCanais / usuarios;
        cout << "\nMedia de idades: " << sIdade / usuarios;
        cout << "\nMedia de filmes : " << sFilmes / usuarios;
    }
}
```

Questão 9 - 3 pontos

A herança é uma das características mais poderosas e importantes da orientação a objetos, pois permite o reaproveitamento de atributos e métodos. Em aplicações que utilizam herança,

- a) não é possível sobrescrever em uma subclasse, um método de sua superclasse.
- b) cada superclasse pode ter apenas uma subclasse.
- c) é possível a um objeto de uma subclasse tanto usar um método da superclasse quanto usar uma versão redefinida deste método pela subclasse.
- d) somente as superclasses poderão ter métodos ou construtores sobrecarregados.

Questão 10 - 3 pontos

Na orientação a objetos, o conceito de encapsulamento corresponde à propriedade de

- a) Receber, por uma classe, uma mensagem sem parâmetros.
- b) Esconder ou ocultar detalhes da implementação de uma dada classe de outras classes.
- c) Utilizar estruturas de matrizes quadradas nos programas desenvolvidos.
- d) Usar variáveis e constantes do tipo inteiro nos métodos das classes implementadas.

Questão 11 - 5 pontos

Informe os termos relacionados aos conceitos de Orientação a Objetos que completam as frases, escolhendo dentre as palavras: atributos, classe, construtor, encapsulamento, herança, mensagem, métodos, objeto, privado, públicos.

- a) Em uma classe apenas os seus membros públicos são conhecidos externamente.
- b) Um membro privado é aquele que não pode ser acessado externamente. A essa capacidade de isolar os membros internos de uma classe ocultando a informação de como uma classe foi implementada chamamos de encapsulamento.
- c) Herança é uma forma de criarmos uma nova classe aproveitando o que já temos pronto em outra. Essa construção se faz de forma hierárquica e pode ter diversos níveis
- d) O construtor pode ser entendido como um método especial, cuja finalidade é ser executado quando um novo objeto de uma classe é criado.

Questão 12 - 5 pontos

Associe a o exemplo de código da 1ª coluna com o conceito que corresponde melhor ao da 2ª: (5 pontos)

- | | |
|--|--|
| (b) class Coordenador : Public Professor { | a. Definição de uma classe |
| (d) Pessoa *fulano = new Pessoa(); | b. Definindo uma herança |
| (a) class Professor { | c. Definição de um construtor |
| (c) public: Hora(int h, int m, int s); | d. Criação de uma nova instância de objeto |
| (e) public: int calculaIdade(Data d1); | e. Definição de um método |