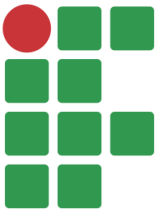

INSTITUTO FEDERAL DA PARAÍBA
CAMPUS CAMPINA GRANDE
BACHARELADO EM ENGENHARIA DA COMPUTAÇÃO
DISCIPLINA DE POO e LAB. POO
PROF. VICTOR ANDRÉ PINHO DE OLIVEIRA

Atividade Unidade II - 2 - Classes: um exame mais profundo 2

Lista Única

Instruções

Responda às questões abaixo. Pode usar este próprio documento. Questões práticas devem ser anexadas separadamente.

As primeiras 7 questões valem 1. As questões 8, 9 e 10 valem 3. A questão 11 vale 4.

Questões

1. Por que devemos qualificar um método como const?

   Resposta: Para especificar que o determinado método não irá alterar o estado da classe.

2. O que acontece se um objeto const tentar invocar um método não-const? Explique.

   Resposta: O compilador não irá permitir, pois um objeto const só pode utilizar de métodos const, para que não ocorra alteração do estado do objeto.

3. O que é e qual a diferença entre associação, composição e agregação?

   Composição é quando um objeto obrigatoriamente é formado pela a junção com outro objeto.

   Agregação é quando um objeto pode ter outro associado a ele, porém sua existência não depende da presença desse outro objeto.

   Associação é quando um objeto interage com outro objeto, mas não necessariamente por composição ou agregação

4. Explique a noção de amizade (**friend**) em C++. Fale sobre prós e contras.

   Resposta: A função friend é quando o programador declara que a função possui uma “amizade” com a classe, fazendo com que a função possua acesso aos atributos da classe. Os prós seria a praticidade do processo e otimização do código,

o contra seria que, caso seja implementado de uma forma errada, fere o princípio de encapsulamento.

5. O que é o ponteiro **this**? Para quem ele aponta? De que maneiras ele pode ser usado?

   Resposta: O ponteiro this é como se fosse um atributo criado implicitamente dentro de uma classe que irá apontar para a própria classe, ele pode ser utilizado para se referir a um atributo da própria classe dentro de um método quando o parâmetro tiver o mesmo nome da classe, ou quando se deseja retornar a própria classe nos métodos sets, para facilitar a leitura e compreensão no código main.

6. Para que servem os operadores new, new [], delete e delete []? Quando usar **new []** em detrimento de **new**?

   Resposta: O new é utilizado para alocar dinamicamente a quantidade necessária de um espaço de memória com o tamanho do tipo passado, o new[] segue a mesma lógica, entretanto, ele é usado para alocar um array daquele tipo, ou seja, o ponteiro irá apontar para um array de determinado tamanho daquele tipo.

   O delete tem a responsabilidade de apagar o que o ponteiro está alocando, ou seja, se o ponteiro aponta para uma região de memória de um int, o delete terá a responsabilidade apagar essa região de memória alocada.

7. O que são membros static? Quais as implicações de uma Classe ter membros static?

   Resposta: Métodos static são métodos comuns a todas as classes que só trabalham com atributos static, a implicação seria a existência de um atributo static, que seria comum a ambas as classes, sendo necessário (em tese) um método static.

8. (Classe *Pessoa*) Crie uma classe para representar uma pessoa, com os atributos privados de nome, idade e altura. Crie os métodos públicos necessários para *sets* e *gets* e também um métodos para imprimir os dados de uma pessoa. Não deixe de qualificar os devidos métodos const.

   Resposta: main.cpp - Questão 8 - Replit

9. (Classe *Agenda*) Crie uma classe **Agenda** que armazena até 10 pessoas e seja capaz de operações como:

```
class Agenda{
```

```
public:
   void armazenaPessoa(string nome, int idade, float altura);
   void armazenaPessoa(const Pessoa &p);

   void removePessoa(string nome);

   int buscaPessoa(string nome) const; // informa em que posição da
agenda está a pessoa
   void imprimePovo() const; // imprime todos os dados de todas as
pessoas da agenda
   void imprimePessoa(int i) const; // imprime os dados da pessoa que
está na posição 'i' da agenda
private:
   Pessoa pessoas[10];
   int numPessoas;
};
```

Teste sua agenda com o programa apresentado abaixo.

```
#include <iostream>
using std::cout, std::endl;

#include "Agenda.h"


int main () {
      char linha[] = "-----------------------------------------\n";
      Agenda A;

      A.armazenaPessoa("Abel", 22, 1.78);
      A.armazenaPessoa(Pessoa("Tiago", 20, 1.80));
      A.imprimePovo();
      cout << linha;

      int posicao = A.buscaPessoa("Tiago");
      if (posicao > 0)
      A.imprimePessoa(posicao);
      cout << linha;

      A.removePessoa("Tiago");
      A.imprimePovo();
      cout << linha;
```

```
      return 0;
}
```

Resposta: https://replit.com/@LiedsonAugusto/Questao-9-1#main.cpp

10. (Classe *Agenda* Melhorada) Modifique a Classe Agenda anterior de modo que a quantidade de Pessoas que a Agenda pode armazenar seja definida no construtor. Certifique-se de realizar a alocação e a desalocação de memória de forma adequada. Adeque a implementação para suportar essa nova possibilidade.

```
class Agenda{
public:
   Agenda(int tPessoas = 1);
   ~Agenda();
   void armazenaPessoa(string nome, int idade, float altura);
   void armazenaPessoa(const Pessoa &p);

   void removePessoa(string nome);

   int buscaPessoa(string nome) const; // informa em que posição da
agenda está a pessoa
   void imprimePovo() const; // imprime todos os dados de todas as
pessoas da agenda
   void imprimePessoa(int i) const; // imprime os dados da pessoa que
está na posição 'i' da agenda
private:
   Pessoa *pessoas

;
   int qtdePessoas;
};
```

Resposta: https://replit.com/@LiedsonAugusto/Questao-10-1#main.cpp

11. (Classe *ArrayList*) Aproveite a Classe ArrayList criada em sala e adicione os seguintes métodos:

    a. busca : que recebe um índice e, caso exista elemento naquele índice, retorne o respectivo elemento
    b. remove : que recebe um índice e, caso exista elemento naquele índice, remova o respectivo elemento
    c. removeEl : que recebe um elemento e, caso ele exista, remova todas as ocorrências deste

    Resposta: main.cpp - Questão 11 - Replit

12. (Classe *ArrayList*) Ainda sobre a Classe ArrayList, escreva as seguintes funções friend:
    a. somaArray: que recebe um objeto ArrayList por referência e um valor inteiro a ser somado a todos os elementos
    b. subArray: que recebe um objeto ArrayList por referência e um valor inteiro para subtrair todos os elementos
    c. mulArray: que recebe um objeto ArrayList por referência e um valor inteiro para multiplicar todos os elementos
    d. divArray: que recebe um objeto ArrayList por referência e um valor inteiro para dividir todos os elementos

    Resposta: main.cpp - Questão 12 - Replit

13. (Classe *IntegerSet*) Crie uma Classe IntegerSet (Conjunto de Inteiros) pela qual cada objeto pode "armazenar um conjunto de inteiros" no intervalo de 0 a 99. Internamente, a Classe deve ter um array de 100 posições, onde cada posição representa o respectivo inteiro. Assim, se o elemento 0 do array estiver setado como 1, significa que o inteiro 0 está presente no conjunto. Se o elemento estiver setado como 0, significa que o inteiro não está no conjunto.

    Forneça 2 construtores. Um construtor-padrão, que não recebe argumentos. Esse deve criar um IntegerSet (conjunto) vazio. E um construtor que recebe um array de inteiros e o tamanho, para inicializar o array interno.

    Forneça os seguintes métodos:

    a. **insertElement**: insere um novo inteiro k no conjunto (seta a posição k do array para 1)

    b. **deleteElement**: deleta um inteiro k do conjunto (seta a posição k do array para 0)

    c. **print**: imprime apenas os inteiros presentes no conjunto, separados por espaço

    Forneça as seguintes funções friend:

    d. **unionOfSets**: recebe dois IntegerSet (conjuntos) e cria e retorna um terceiro conjunto que representa a união dos conjuntos

    e. **intersectionOfSets**: recebe dois IntegerSet (conjuntos) e cria e retorna um terceiro conjunto que representa a interseção dos conjuntos

    Resposta: https://replit.com/@LiedsonAugusto/Questao-13#main.cpp