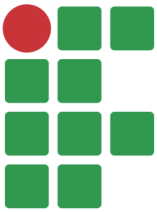

INSTITUTO FEDERAL DA PARAÍBA
CAMPUS CAMPINA GRANDE
BACHARELADO EM ENGENHARIA DA COMPUTAÇÃO
DISCIPLINA DE POO e LAB. POO
PROF. VICTOR ANDRÉ PINHO DE OLIVEIRA

Atividade Unidade II - 2 - Fundamentos C++

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?

   Já que esse método não haverá modificação de atributos, então é interessante utilizar const para o design da classe.

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

   Vai dar erro, em objetos const só é possível utilizar métodos const.

3. O que é composição?

   Quando uma classe possui objetos de outra classe como atributos.

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

   Elas são definidas fora do escopo da classe, porém podem acessar membros privados e públicos de seus objetos.

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

   O ponteiro this possui o endereço do objeto. Ele aponta para o endereço de memória do objeto. Pode ser usado para explicitamente acessar os membros do objeto ou para permitir chamadas de métodos em cascata.

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

new = Aloca memória para um objeto.

new[] = Aloca memória para um array.

delete = Desaloca memória para um objeto.

Delete[] = Desaloca memória para um array.

Pode ser usado para alocar ponteiros para funções.

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

São membros de classe, logo são compartilhados por todos os objetos. Todos os objetos instanciados terão o mesmo atributo ou métodos static, diferentemente de não static, onde cada objeto terá o seu atributo ou método.

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.

9. (Classe Agenda) Crie uma classe **Agenda** que armazena 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];
};
```

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;
}
```

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;
};
```

11. (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