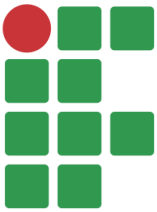

INSTITUTO FEDERAL DA PARAÍBA
CAMPUS CAMPINA GRANDE
BACHARELADO EM ENGENHARIA DA COMPUTAÇÃO
DISCIPLINA DE POO e LAB. POO
PROF. VICTOR ANDRÉ PINHO DE OLIVEIRA

Atividade Unidade II - 1 - Fundamentos C++

Lista Única

# Instruções

Responda às questões abaixo. Pode usar este próprio documento.

As primeiras 5 questões valem 1. As questões 6 a 8 demais valem 2. A questão 9 vale 4.

# Questões

1. Qual a utilidade dos métodos inline?

   Resposta: Os métodos inline tem como principal utilidade otimizar a classe, em contrapartida, irá consumir mais memória.

2. O que acontece quando atribuímos um objeto a outro (mesmo tipo)?

   Resposta: O objeto que está atribuindo o outro para si, recebe para seus atributos os mesmos atributos do outro objeto.

3. O que acontece quando definimos métodos private? Qual a utilidade?

   Resposta: O método quando é colocado na estrutura private faz com que o usuário não tenha acesso aquele método, essa prática é útil quando se quer repartir algum método public em menores, onde não faz sentido o usuário ter acesso a aquela parte do método.

4. O que é, para que serve e quando é invocado o destrutor de uma Classe?

   Resposta: O destrutor ganha um sentido melhor quando possui algum atributo que é um ponteiro

5. Quais as implicações de um método get retornar uma referência não-const para um atributo private?

   Resposta: Caso o método get retorne uma referência, ele estará infligindo o conceito de encapsulamento, pois ele estará retornando uma referência para algum atributo private, tornando possível a alteração do mesmo.

6. (Aprimorando a Classe Time) Modifique a Classe Time da última versão em “sala” (Aula7Ex3) para incluir um método tick que incrementa 1 segundo à hora. O objeto Time sempre deve permanecer em um estado consistente. Escreva um programa que testa o seu método tick. Certifique-se de testar os seguintes casos:

    a. Incrementar para o próximo minuto. Ex.: 11h50m59 + 1s -> 11h51m00

    b. Incrementar para a próxima hora. Ex.: 11h59m59 + 1s -> 12h00m00

    c. Incrementar para o próximo dia. Ex.: 23h59m59 + 1s -> 0h00m00

    Resposta: main.cpp - Questão 6 - Replit

7. (Aprimorando a Classe Time) Forneça um construtor que seja capaz de utilizar a hora atual da função time() - disponível através da biblioteca <ctime> - para inicializar um objeto da Classe Time. Para tanto:

    a. Use a última versão de “sala” da Classe Time (Aula7Ex3)

    b. Remova o argumento default mais a esquerda (da hora) do construtor

    c. Adicione um novo construtor sem parâmetros (sobrecarregando o construtor) que inicializa o objeto com a hora atual

    d. Pesquise como pegar a hora atual do sistema a partir da função time() da biblioteca <ctime>.

    e.

    Resposta: main.cpp - Questão 7 - Replit

8. (Classe Retangulo) Crie uma Classe Retangulo com atributos altura e largura, cada um dos quais assume o padrão 1.0 (no construtor). Forneça métodos get e set para esses atributos. Os métodos set devem verificar se o valor passado é maior que 0.0 e menor que 20.0. Caso não seja, deve ser atribuído o valor 1 ao atributo. Além disso, forneça métodos que:

    a. Calcula e retorna o perímetro do retângulo

    b. Calcula e retorna a área do retângulo

    Resposta: main.cpp - Questão8 - Replit

9. (Classe Pessoa) Crie uma Classe Pessoa com atributos nome, CPF e CPFvalido. Você deverá definir apenas um construtor e este deverá receber o nome e o CPF. Forneça métodos get e set para os dois primeiros atributos. A Classe deverá validar o CPF (acesse este link para conhecer o algoritmo de validação) e em função disso setar o valor para o atributo CPFvalido (true ou false). Considere que o CPF seja passado como uma sequência de 11 dígitos, sem pontos e sem traços. Escreva também um método chamado apresentar que exibe na tela o nome e o CPF. Ao exibir o CPF, deverá constar entre parênteses a situação do CPF (“Válido” ou “Inválido”).

Resposta: main.cpp - Questão 9 - Replit

10. (Classe HugeInteger) Crie uma Classe HugeInteger que utiliza um array de 40 elementos (char) para armazenar inteiros de até 40 dígitos. Forneça os métodos:

    a. input: recebe uma string contendo o inteiro. O método deve verificar se realmente está recebendo um número. Não precisa considerar sinal (+ ou -), pois os números quando corretos sempre serão positivos.

    b. output: imprime o número na saída padrão

    c. add: soma objeto com outro HugeInteger

    d. isEqualTo, isNotEqualTo, isGreaterThan, isLessThan, isGreaterThanOrIqual e isLessThanOrIqual (==,!=,>,<,>=,<=): compara o objeto com outro objeto HugeInteger passado como argumento. O retorno deve ser do tipo bool, isto é, true ou false.

    Dicas:

    - Você pode armazenar o número de forma invertida, corrigindo na hora de exibir. Apenas uma opção, você pode encontrar um jeito melhor de fazer.
    - Você pode definir um atributo para armazenar o tamanho em dígitos do número. Isso vai te ajudar na hora de comparar os objetos.
    - Não precisa se preocupar com situações excepcionais. Concentre-se na solução em si.