Universidade Federal da Paraíba	Linguagem de Programação I
Centro de Informática	Semestre: 2019.2
Departamento de Informática	Professor: Tiago Maritan

1ª Lista de Exercícios

Entrega: 16/12/2019

1) Sobre o Paradigma de Programação Orientada a objetos, responda as seguintes questões:
(a) O que são classes e objetos? Explique a diferença entre os dois.
(b) Para que serve um construtor e quando e como ele pode ser utilizado?
(c) Quais são os benefícios do encapsulamento? Exemplifique.

2) Implemente uma classe `Pessoa` em C++ com os atributos `nome` (do tipo `string`), `idade` (do tipo `int`) e `telefone` (do tipo `string`). Considere também que ela possui dois construtores (um que só recebe o nome e outro que recebe nome, idade e telefone) e métodos para obter e alterar esses atributos (métodos `get()` e `set()`). Em seguida, crie um arquivo main.cpp, contendo o método `main` que cria dois objetos do tipo Pessoa e testa os seus métodos.

3) Implemente uma classe `Invoice` em C++ que possa ser utilizado por uma loja de suprimentos de Informática para representar uma fatura de um item vendido na loja. Uma fatura deve incluir as seguintes informações como atributos:

- Número do item faturado;
- Descrição do item;
- Quantidade comprada do item;
- Preço unitário do item.

Sua classe deve ter um construtor que inicialize os quatro atributos. Se a quantidade não for positiva, ela deve ser configurada como 0. Se o preço por item não for positivo ele deve ser configurado como 0.0. Crie métodos `get()` e `set()` para acessar e configurar cada um dos seus atributos. Além disso, forneça um método chamado `getInvoiceAmount()` que calcula o valor da fatura (isso é, multiplica a quantidade pelo preço por item) e depois retorna o valor como um `double`. Em seguida, crie um arquivo main.cpp, com o método `main()`, que cria alguns objetos da classe `Invoice` e testa os seus métodos.

4) Implemente uma classe `Data` em C++ cuja instância (objeto) represente uma data. Esta classe deverá dispor dos seguintes métodos:

- Dois construtores: um construtor sem parâmetro e um construtor com três parâmetros que verifica se a data é válida e inicializa os seus atributos. Caso a data não esteja correta, ele deve configurá-la como caso 01/01/0001;
- Um método `compara()` que recebe um outro objeto da Classe Data como parâmetro, compara-o com a data corrente e retorna:
    - 0 se as datas forem iguais;
    - 1 se a data corrente for maior que a do parâmetro;

    - -1 se a data do parâmetro for maior que a corrente.
- Um método chamado `getMesExtenso()` que retorna o mês da data corrente por extenso;
- Um método `isBissexto()` que retorna verdadeiro se o ano da data corrente for bissexto e falso caso contrário;
- Métodos `get()` e `set()` para acessar e configurar cada um dos seus atributos;

Em seguida, crie um arquivo main.cpp, com o método `main()`, que cria alguns objetos da classe `Data` e testa seus métodos.

5) Implemente uma classe em C++ chamada `Horario` cuja instância represente um horário composto pelos seguintes atributos: `hora`, `minuto` e `segundo`. A classe deve representar esses componentes de horário e deve apresentar os métodos descritos a seguir:

- Dois construtores: um construtor sem parâmetro e um construtor com três parâmetros que verifica se o horário é válido e inicializa os seus atributos. Caso o horário não esteja correta, ele deve configurá-lo como caso 00:00:00;
- Um método chamado `setHorario()`, que deve receber os valores de hora, minuto e segundo como parâmetro;
- Métodos `get()` e `set()` para acessar e configurar cada um dos seus atributos;
- Um método para `avancarHorario()` o horário para o próximo segundo (lembre-se de atualizar o minuto e a hora, quando for o caso).

Em seguida, crie um arquivo main.cpp, com o método `main`, que cria alguns objetos da classe `Relogio` e testa os seus métodos.

6) Implemente uma classe em C++ chamada `Voo` cuja instância (objeto) represente um voo que acontece em determinada `Data` e em determinado `Horário`. Cada voo possui no máximo 100 lugares, e a classe deve controlar a ocupação das vagas. A classe deve conter os seguintes métodos:

- Um construtor que configura os dados do voo, como por exemplo, o número do voo, a sua data e horário;
- Um método `proximoLivre()` que retorna o número da próxima cadeira livre;
- Um método `verifica()` que verifica se o número da cadeira recebido como parâmetro está ocupada;
- Um método `ocupa()` que ocupa uma determinada cadeira do voo, cujo número é recebido como parâmetro, e retorna `true` se a cadeira ainda não estiver ocupada (operação foi bem sucedida), e `false`, caso contrário;
- Um método `vagas()` que retorna o número de cadeiras vagas disponíveis (não ocupadas) no voo;
- Um método `getNumVoo()` que retorna o número do voo;
- Um método `getData()` que retorna a data do voo (na forma de objeto);
- Um método `getHorario()` que retorna o horário do voo;

Crie também um arquivo main.cpp, com o método `main`, que cria alguns objetos da classe `Voo` e testa seus métodos.

7) Considere um Sistema de Controle de Pagamentos de Funcionários de uma empresa. Esse sistema apresenta uma classe chamada `Pagamento`, com os atributos `valorPagamento`, do tipo `double` e `nomeDoFuncionario`, do tipo `String` e métodos para obter e alterar esses atributos (métodos `get` e `set`).

Esse sistema apresenta também uma classe `ControleDePagamentos`, que apresenta um atributo chamado `pagamentos`, que é um array de objetos do tipo `Pagamento` e o método `setPagamentos`. Essa classe apresenta também um método chamado `calculaTotalDePagamentos`, que não tem parâmetros e deve retornar o valor total dos pagamentos do sistema. Nessa classe há ainda o método `existePagamentoParaFuncionario (string nomeFuncionario)` que verifica se dentre os pagamentos guardados no `ControleDePagamentos` há algum deles que se refere ao funcionário passado como parâmetro, retornando `true` neste caso e `false`, caso contrario.

a) Implemente a classe `Pagamento`.

b) Implemente a classe `ControleDePagamentos`.

c) Escreva um arquivo main.cpp, com o método main, que crie um objeto do tipo `ControleDePagamentos` e que configure ao menos dois pagamentos. Faça com que o programa imprima o total dos pagamentos utilizando o método correspondente da classe `ControleDePagamentos`.