ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO
Departamento de Engenharia de Computação e Sistemas Digitais

PCS 3111 - Laboratório de Programação Orientada a Objetos para Engenharia Elétrica

Aula 10 – Persistência de Objetos

Atenção

1. Código inicial a ser usado na resolução dos exercícios encontra-se **disponível no e-Disciplinas**.
2. Apenas declarações das classes são fornecidas como código inicial, cabendo ao aluno assegurar o carregamento único dos cabeçalhos (*headers*), definir as classes, e incluir bibliotecas necessárias.
3. Os nomes, os atributos, os métodos, e as respectivas assinaturas das classes dadas **devem seguir o especificado** em cada exercício para fins de correção automática.
4. A função `main` não deve ser submetida. Caso contrário, a correção automática retornará um *Compilation Error.*

Exercício 1
Implemente uma nova classe, **PersistenciaPessoa**, que será responsável pela persistência dos objetos da classe **Pessoa** entregue (e que não deve ser alterada). Essa classe possui os seguintes métodos:

```
class PersistenciaPessoa {
public:
    PersistenciaPessoa(string arquivo);
    virtual ~PersistenciaPessoa();

    void inserir(Pessoa *p);
};
```

- O construtor **PersistenciaPessoa** recebe o endereço do arquivo em que os dados serão salvos.
- O método **inserir** deve inserir os dados da pessoa passada como argumento, adicionando-os ao final do arquivo.

O arquivo a ser usado por `PersistenciaPessoa` deve possuir o seguinte formato:

```
nomeDaPessoa1
valorPorHora1
horasDiarias1
...
nomeDaPessoan
valorPorHoran
horasDiariasn
```

Ou seja, em um arquivo serão armazenados os dados (nome, horas diárias e valor por hora) de n **Pessoas**. A primeira linha do arquivo deverá conter o nome da primeira pessoa. A segunda

linha deve conter o valor por hora da primeira **Pessoa**, isto é, quanto que ela ganha por hora trabalhada. A terceira linha deve conter as horas diárias da primeira **Pessoa**, isto é, quantas horas por dia ela trabalha. Na quarta linha deve ser armazenado o nome da segunda Pessoa, e assim por diante. **Note que o arquivo termina com um "\n"**!

Exemplo: Sejam duas **Pessoas**, "Foo" e "Bar", cujos valoresPorHora sejam, respectivamente, 12 e 13, e cujas horasDiarias sejam, respectivamente, 8 e 6. Ao salvar "Foo" e depois "Bar", teremos um arquivo texto com a seguinte apresentação (em um editor de texto):

```
Foo
12
8
Bar
13
6

```

ATENÇÃO: Admita que o nome da **Pessoa não** contém espaços!

Exercício 2
Adicione à classe **PersistenciaPessoa** o método

```
Pessoa** obter(int& quantidade);
```

que retorna um vetor (alocado dinamicamente) com as **Pessoas** armazenadas no arquivo e a quantidade de pessoas desse vetor, por referência. Considere que o arquivo contém no máximo 10 **Pessoas**. Desse modo, deve-se reconstruir todas as **Pessoas** armazenadas no arquivo, com seus respectivos nomes e atributos persistidos.

Assuma que quem chamou o método será o responsável por destruir o objeto.

Esse método deve ser implementado de modo a lidar com possíveis problemas de leitura e formatação do arquivo:

- Caso o arquivo não seja encontrado, a função deve jogar a exceção **invalid_argument** da biblioteca padrão.
- Caso o arquivo esteja vazio, a função deve retornar **NULL**.
- Caso o arquivo não siga a formatação esperada, deve-se jogar a exceção **logic_error** da biblioteca padrão.
    - **DICA**: repare que a formatação correta espera que a primeira linha contenha uma string, a segunda um double e a terceira um int, sucessivamente, para n **Pessoas** salvas.

Testes do Judge

Exercício 1

- inserir Pessoa em arquivo vazio
- inserir Pessoa em arquivo inexistente
- inserir Pessoa em arquivo não vazio

Exercício 2

- obter para arquivo inexistente
- obter para arquivo vazio
- obter para arquivo com tipos de dados incorretos
- obter para 1 Pessoa
- obter para várias Pessoas