Data: 12/12/2011
Prof. Dr. Jean M. Simão - DAELN/UTFPR Ctba-Centro.

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ - Campus de Curitiba (Brasil) – **Dep. Acadêmico de Eletrônica** (DAELN). Disciplina: **Fundamentos de Programação 2 - IF62C**/Turma: **S11/S12**. **Prof**: Jean M. **Simão**. Curso: **Eng. Eletrônica - Prova sobre linguagem C++ / Orientação a Objetos – Exame de Recuperação.**

Nome do Aluno: _____ **Turma**: ____
Horário de Começo: _____ Horário de Fim: _____

Leia toda a prova, pois os enunciados estão completados uns nos outros.
Utilize os bons princípios de projeto e programação orientada a objetos.

(Questão - 1) Em um programa C++ (para *console*), crie uma classe abstrata chamada de *Partícula* com uma única função virtual pura. Entretanto, esta classe pode ter outras funções que não sejam virtuais puras. Ainda, esta classe terá um atributo protegido inteiro chamado *número* e outro atributo protegido real chamado *peso.*

(Questão - 2) Crie uma classe *Próton* (com atributo booleano *núcleo*), uma classe *Nêutron* (com atributo ponteiro de *Próton*) e uma classe *Elétron* (com um atributo inteiro *nível*), todas derivadas de *Partícula* e cada qual com uma função-membro chamada "*void informe_natureza()*". No caso de *Próton* tal função-membro informará (em tela) "*Próton - positivo"* precedido de seu *número* e *peso*, no caso de *Nêutron* informará "*Nêutron - neutro"* precedido de seu *número* e *peso* e, por fim, no caso de *Elétron* informará "*Elétron - negativo"* precedido de seu *número* e *peso.*

(Questão - 3) Crie uma classe *Lista_Particulas* para ponteiros de objetos de *Partícula*, baseando-se na classe *Vector* da *STL*, que na prática servirá para ponteiros de objetos de subclasses de *Partícula* uma vez que esta classe é abstrata. Mais precisamente, a classe *Lista_Particulas* deverá agregar fortemente um objeto da classe *Vector* parametrizada com (o tipo) ponteiro de *Partícula*. Ainda, a classe *Lista_Particulas* deverá ter um método para incluir, bem como outro para listar. Por fim, a listagem apresentará a 'natureza' de cada objeto cujo apontamento esteja registrado na lista, fazendo uso de *iterator*. Outrossim, esta classe deverá ser a única classe de lista desta prova, visando assim o uso pleno de polimorfismo.

(Questão - 4) Crie uma classe *Átomo*, derivada da classe *Partícula*, com uma função-membro chamada "*void informe_natureza()*" que informará (em tela) "*Átomo"*. Cada objeto de *Átomo* poderá armazenar em si ponteiros de objetos *Elétron* por meio de uma função *void incluir_Elétron (Elétron* pe)*. Similarmente, cada objeto de *Átomo* poderá armazenar em si ponteiros de objetos *Próton* por meio de uma função *void incluir_Próton (Próton * pp)*. Entretanto, a cada apontamento de objeto *Próton* incluído, será instanciado dinamicamente um objeto de *Nêutron*, o qual será relacionado ao objeto *Próton* em questão. Ainda, cada objeto *Átomo* armazenará os apontamentos de objetos *Elétron*, *Próton* e *Nêutron* como apontamentos de objetos *Partícula* por meio de uma lista para tal. Por fim, a classe *Átomo* deve permitir atualizar seu 'peso' cada vez que uma 'partícula' lhe for 'adicionada' justamente adicionando o peso desta partícula.

(Questão - 5) Crie uma classe *Molécula* onde serão criados dinamicamente objetos de *Átomo*, bem como objetos de *Próton* e *Elétron*. Certamente, os objetos de *Próton* e de *Elétron* serão associados a objetos de *Átomo* neste âmbito. Outrossim, a classe *Molécula* terá uma lista que conterá os apontamentos destes objetos de *Átomo* criados, bem como dos objetos de *Próton* e de *Elétron* criados. Ainda, a classe *Molécula* terá um método *void executar()* que permitirá listar a natureza dos objetos considerados na citada lista. Por fim, a classe *Molécula* deve permitir a cada objeto *Molécula* decorrente calcular seu 'peso' baseado (apenas) nos objetos de *Átomo* que lhe foram 'agregados'.

A interpretação faz parte do conteúdo da prova! Inclua comentários (se for o caso) para deixar explicitas suas decisões.