Universidade Federal de Ouro Preto – UFOP
Instituto de Ciências Exatas e Biológicas – ICEB
Departamento de Computação – DECOM
Disciplina: Programação Orientada a Objetos
Professor: Marco Antonio M. Carvalho

Lista de Exercícios 06 – Genéricos

Instruções

- *Todos os exercícios que envolvem programas devem ser resolvidos por programas em linguagem **C**++;*
- *Na solução dos exercícios, devem ser utilizados os conceitos listados no cabeçalho desta lista;*
- *Eventuais dúvidas podem ser sanadas com o professor.*

1. Crie uma **função genérica** *quickSort*(). Crie também um *driver* que lê, ordena e imprime vetores dos tipos *int*, *float*, *double* e *char* e também para objetos de uma classe criada por você, com pelo menos 3 atributos.

2. **Sobrecarregue** a função genérica *printArray*() vista na aula para receber mais dois parâmetros: o índice inicial e o índice final da impressão. Esta função também deve ser genérica. Crie um *driver*.

3. Crie uma **função** *saldo()* **genérica** que recebe um ponteiro para um objeto da classe Conta (exercício 3 da lista anterior) e imprima seu saldo. Lembre-se que a função deve ser genérica, e portanto, seus parâmetros não devem ter tipo definido. Crie um *driver* com um vetor de ponteiros para *Conta*, cada ponteiro deve apontar para um dos objetos de cada classe concreta da hierarquia ser passado como parâmetro para a função genérica.

4. Crie uma **função genérica** *localizar*(), que recebe como parâmetros um vetor, um elemento do mesmo tipo que os elementos do vetor e um inteiro informando o tamanho do vetor. A função deve tentar localizar o elemento representado pelo segundo parâmetro no vetor. Caso o elemento seja encontrado, a função deve retornar o índice do elemento, caso contrário, deve retornar -1.

5. Crie uma **classe genérica** *Vetor*, que recebe como parâmetros o número e o tipo de elementos do vetor, ambos padronizados com *int* e 10. Nesta classe devem ser criados um construtor e destrutor para alocar o vetor, um método para retornar a quantidade de elementos no vetor (chamado *getSize*()) e outro para adicionar um elemento (chamado *add*()). Sobrecarregue o operador **[]**. **Não utilize a classe *vector***.

6. Crie uma **função global** *friendVetorInt*(), amiga apenas da especialização da classe *Vetor< int >*. Esta função deve inicializar todas as posições do vetor com o valor -1. Crie um driver que testa a função com uma especialização *Vetor< int >* e uma especialização *Vetor< float >*. **Este exercício deve ser feito em um arquivo diferente do exercício anterior.**

7. Crie uma classe genérica *MembroStatic* que recebe como parâmetro o tipo dos seus objetos. A classe deve possuir um membro *static* inteiro, que é incrementado em seu construtor e decrementado em seu destrutor. Crie um *driver* que gera 3 especializações desta classe e mostre que realmente cada especialização da classe genérica possui uma cópia do membro *static*.