UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
INSTITUTO METRÓPOLE DIGITAL
IMD0030 – LINGUAGEM DE PROGRAMAÇÃO I

Laboratório 5

Classes, Objetos, Métodos Construtores, Método Destrutor e Sobrecarga de Operadores

Objetivo

O objetivo deste exercício é colocar em prática conceitos do paradigma de Programação Orientada a Objetos (POO) na linguagem de programação C++, em particular pela implementação de classes, objetos, métodos construtores, método destrutor e sobrecarga de operadores.

Orientações gerais

Você deverá observar as seguintes observações gerais na implementação deste exercício:

1) Apesar da completa compatibilidade entre as linguagens de programação C e C++, seu código fonte **não** deverá conter recursos da linguagem C nem ser resultante de mescla entre as duas linguagens, o que é uma má prática de programação. Dessa forma, deverão ser utilizados **estritamente** recursos da linguagem C++.

2) Durante a compilação do seu código fonte, você deverá habilitar a exibição de mensagens de aviso (*warnings*), pois elas podem dar indícios de que o programa potencialmente possui problemas em sua implementação que podem se manifestar durante a sua execução.

3) Aplique boas práticas de programação. Codifique o programa de maneira legível (com indentação de código fonte, nomes consistentes, etc.) e documente-o adequadamente na forma de comentários. Anote ainda o código fonte para dar suporte à geração automática de documentação utilizando a ferramenta Doxygen (http://www.doxygen.org/). Consulte o documento extra disponibilizado na Turma Virtual do SIGAA com algumas instruções acerca do padrão de documentação e uso do Doxygen.

4) Busque desenvolver o seu programa com qualidade, garantindo que ele funcione de forma correta e eficiente. Pense também nas possíveis entradas que poderão ser utilizadas para testar apropriadamente o seu programa e trate adequadamente possíveis entradas consideradas inválidas.

5) Lembre-se de aplicar boas práticas de modularização, em termos da implementação de diferentes funções e separação entre arquivos cabeçalho (`.h`) e corpo (`.cpp`).

6) A fim de auxiliar a compilação do seu projeto, construa **obrigatoriamente** um `Makefile` que faça uso da estrutura de diretórios apresentada anteriormente em aula.

7) Garanta o uso consistente de alocação dinâmica de memória. Para auxilia-lo nesta tarefa, você pode utilizar o Valgrind (http://valgrind.org/) para verificar se existem problemas de gerenciamento de memória.

Autoria e política de colaboração

Este trabalho deverá ser realizado **individualmente**. O trabalho em cooperação entre estudantes da turma é estimulado, sendo admissível a discussão de ideias e estratégias. Contudo, tal interação não deve ser entendida como permissão para utilização de (parte de) código fonte de colegas, o que pode caracterizar situação de plágio. Trabalhos copiados em todo ou em parte de outros colegas ou da Internet serão sumariamente rejeitados e receberão nota zero.

Apesar de o trabalho ser feito individualmente, você deverá utilizar o sistema de controle de versões Git no desenvolvimento. Ao final, todos os arquivos de código fonte do repositório Git local deverão estar unificados em um repositório remoto Git hospedado em algum serviço da Internet, a exemplo do GitHub, Bitbucket, Gitlab ou outro de sua preferência. A fim de garantir a boa manutenção de seu repositório, configure corretamente o arquivo `.gitignore` em seu repositório Git.

Entrega

Você deverá submeter um único arquivo compactado no formato `.zip` contendo todos os códigos fonte resultantes da implementação deste exercício, sem erros de compilação e devidamente testados e documentados, **até as 23h59 do dia 4 de maio de 2017** através da opção *Tarefas* na Turma Virtual do SIGAA. Você deverá ainda informar, no campo *Comentários* do formulário de submissão da tarefa, o endereço do repositório Git utilizado.

Avaliação

O trabalho será avaliado sob os seguintes critérios: (i) utilização correta dos conteúdos vistos anteriormente e nas aulas presenciais da disciplina; (ii) a corretude da execução dos programas implementados, que devem apresentar saída em conformidade com a especificação e as entradas de dados fornecidas; (iii) a aplicação correta de boas práticas de programação, incluindo legibilidade, organização e documentação de código fonte, e; (iv) a utilização correta do repositório Git, no qual deverá estar registrado todo o histórico da implementação por meio de *commits*. A presença de mensagens de aviso (*warnings*) ou de erros de compilação e/ou de execução, a modularização inapropriada e a ausência de documentação são faltas que serão penalizadas. Este trabalho contabilizará nota de até 1,0 ponto na 2ª Unidade da disciplina.

Programa 1

Implemente um programa em C++ que atenda aos seguintes critérios:

a) Contenha uma classe que representa um funcionário, registrando seu nome, salário e data de admissão;
b) Contenha também uma classe que representa uma empresa, registrando seu nome, CNPJ e lista de funcionários;
c) O programa deverá permitir criar uma empresa;
d) O programa deverá permitir que se adicione a uma empresa alguns funcionários (a ser criado e armazenado usando alocação dinâmica) e não deve permitir adicionar um funcionário que já tenha sido anteriormente adicionado, sendo neste caso exibida uma mensagem de erro. Os dados referentes aos funcionários a serem adicionados deverão ser lidos a partir de um arquivo no formato CSV (*Comma-Separated Values*). **Dica:** utilize a sobrecarga dos operadores de igualdade (==) e de extração de *stream* (>>).
e) O programa deverá permitir listar os dados de todos os funcionários de uma empresa, sobrecarregando-se o operador de inserção em *stream* (<<);
f) O programa deverá permitir que seja dado um aumento de *X*% a todos os funcionários de uma determinada empresa;
g) O programa deverá permitir listar os dados de todos os funcionários de uma empresa em período de experiência, ou seja, contratados há menos de 90 dias considerando a data corrente.

Dica: Observe que as funcionalidades exigidas nesta questão podem (e devem) ser resolvidas com a alteração das classes (por exemplo, criando novos métodos) e não manipulando de forma mirabolante os objetos no programa principal. Não complique o que é simples!

Programa 2

Utilizando classes, implemente em C++ um jogo de dados hipotético, no qual cada jogador deve lançar dois dados em sua vez. A soma dos valores dos dados deve ser acumulada para cada jogador. O objetivo é ficar o mais próximo e abaixo do valor *N* a ser estabelecido como configuração geral do jogo e visível a todos os jogadores. Ao obter um valor agregado superior a *N*, o jogador é considerado excluído da rodada. A cada vez de jogar, o jogador pode optar por jogar os dados ou parar (e não jogar mais na rodada). Uma rodada é finalizada quando: (1) resta apenas um jogador, uma vez que os outros foram excluídos; (2) quando não há mais jogadores a jogar, ou seja, todos os jogadores "ativos" decidiram parar, ou; (3) quando um dos jogadores atinge exatamente o valor **N**. Vence a rodada o jogador que permanecer na rodada (ou seja, não for excluído) e obtiver o número de pontos agregados mais próximo de *N*. Ao final de cada rodada, o programa deverá listar os pontos agregados obtidos por cada jogar e declarar o campeão da rodada.