UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
INSTITUTO METRÓPOLE DIGITAL
IMD0030 – LINGUAGEM DE PROGRAMAÇÃO I | 2016.2

Projeto de Programação III

Cadastro de Animais de um *Pet Shop* v2.0

1 Objetivo

O objetivo deste trabalho é introduzir melhorias ao programa de cadastro de animais para a loja de animais silvestres *Pet Fera*, implementado no Projeto de Programação II. As melhorias a serem introduzidas dizem respeito à aplicação de conceitos relacionados a STL, bibliotecas e tratamento de exceções. Este documento apresenta as novas melhorias a serem introduzidas na versão 2.0.

2 Melhorias na versão 2.0

Você deverá implementar as seguintes alterações no programa de cadastro de animais para a loja de animais silvestres *Pet Fera*:

1) Substitua a TAD Lista implementada por um *container* do tipo `map` da STL. Realize as correções/adaptações necessárias.

2) A fim de permitir a reutilização de código em projetos futuros, organize o seu modelo de classes em uma biblioteca dinâmica de nome `petfera.so` (Linux) ou `petfera.dll` (Windows). Essa biblioteca deverá ser utilizada na construção dos programas que irão compor o sistema *Pet Fera*.

3) Utilizando a biblioteca criada no item anterior, implemente um programa auxiliar para permitir exportar apenas dados de animais que satisfaçam a um determinado conjunto de critérios.

   ```
   ./exportar -c <classe> -v <veterinario> -t <tratador> <arquivo_saida>
   ```

   Na sintaxe acima, o arquivo executável exportar é invocado passando quatro possíveis argumentos via linha de comando:

   - `-c <classe>`: este argumento opcional indica a classe de animais a serem exportados; por razões de simplicidade, deverá ser indicada apenas uma classe;
   - `-v <veterinario>`: este argumento opcional indica que apenas animais sob a responsabilidade do veterinário informado devem ser exportados; por razões de simplicidade, deverá ser indicado apenas um nome de veterinário;
   - `-t <tratador>`: este argumento opcional indica que apenas animais tratados pelo tratador informado devem ser exportados; por razões de simplicidade, deverá ser indicado apenas um

tratador;

- `<arquivo_saida>`: este argumento obrigatório indica o nome do arquivo de saída, ou seja, do arquivo onde serão guardados os dados a serem exportados; o formato do arquivo de exportação deve seguir o mesmo formato indicado na Tabela 1 do documento que descreve o Projeto de Programação II.

4) Realize o devido tratamento de exceções para as operações de manipulação de arquivos e para a entrada (leitura) de dados por meio da criação das classes de exceção necessárias e lançamento dos respectivos objetos quando for o caso.

5) Uma vez realizadas as tarefas de implementação, você deverá elaborar um relatório simples contendo minimamente (1) uma introdução, a fim de explicar o propósito do relatório e (2) detalhes de implementação, descrevendo como foi feita a sua implementação em termos de arquivos, classes, métodos, etc. e como o programa funciona de maneira geral.

3 Orientações gerais

Você deverá atentar para as seguintes observações gerais no desenvolvimento deste trabalho:

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 e documente-o adequadamente na forma de comentários. Como anteriormente instruído, o código fonte deverá ser anotado para dar suporte à geração automática de documentação utilizando o 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) Para melhor organização do seu código, faça a separação da implementação do programa entre arquivos cabeçalho (`.h`) e corpo (`.cpp`).

4 Autoria e política de colaboração

O trabalho é individual, porém espera-se que os códigos fonte feitos mantidos em um repositório Git local e, posteriormente, unificados em um repositório Git remoto. Cada aluno deverá ter um repositório individual no Gitlab do IMD-UFRN (http://projetos.imd.ufrn.br/), no qual deverão ser disponibilizados todos os arquivos referentes ao programa implementado. O endereço do repositório será informado pelo professor a cada aluno, devendo os mesmos terem feito previamente seu cadastro

no Gitlab.

A atividade de cada aluno deverá ser registrada através de *commits* sobre o repositório Git, que será examinado pelo professor durante a avaliação do trabalho. Além disso, a critério do professor, qualquer aluno poderá ser convocado para uma entrevista cujo objetivo é confirmar a autoria do trabalho desenvolvido. Durante a entrevista, o aluno deverá ser capaz de explicar, com desenvoltura, qualquer parte do trabalho. Portanto, é possível que ocorra, após a entrevista e/ou exame das atividades registradas, redução da nota geral do trabalho.

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

Entrega

Todos os códigos fonte referentes à implementação do trabalho deverão ser disponibilizados sem erros de compilação e devidamente testados e documentados através do repositório Git do aluno no Gitlab do IMD-UFRN (http://projetos.imd.ufrn.br/). Além disso, você deverá submeter, **até as 23h59 do dia 10 de dezembro de 2016**, um único arquivo compactado através da opção Tarefas na Turma Virtual do SIGAA contendo: (1) os mesmos arquivos de código fonte disponíveis no repositório Git; (2) a documentação do projeto na forma de páginas HTML, geradas automaticamente com a ferramenta Doxygen, e; (3) o relatório escrito, preferencialmente em formato PDF. É importante destacar que serão avaliados única e exclusivamente os arquivos submetidos via SIGAA.

5 Avaliação

A avaliação deste trabalho será feita principalmente sobre os seguintes critérios: (1) utilização correta dos conteúdos vistos nas aulas presenciais da disciplina; (2) a corretude da execução do programa implementado, que deve apresentar saída em conformidade com a especificação e as entradas de dados fornecidas; (3) a aplicação correta de boas práticas de programação, incluindo legibilidade, organização e documentação de código fonte, e; (4) qualidade do relatório técnico produzido. O trabalho possuirá **nota máxima de 5,00 (cinco) pontos**, distribuídos de acordo com a seguinte composição:

Item avaliado	**Nota máxima**
Modularização adequada	0,50
Melhorias da versão 2.0	
– Uso do *container* `map` da STL	0,75
– Criação e utilização de biblioteca	0,75
– Programa para exportação de dados	1,00
– Tratamento de exceções	0,50
Uso adequado de ferramentas de compilação (com *Makefile*), depuração e verificação de memória	0,50
Uso correto de controle de versão com Git	0,25
Qualidade do relatório escrito	1,00
Total	**5,00**

Por sua vez, o não cumprimento de algum dos critérios anteriormente especificados poderá resultar nos seguintes decréscimos, calculados sobre a nota total obtida até então:

Falta	**Decréscimo**
Programa apresenta erros de compilação, não executa ou apresenta saída incorreta	−70%
Falta de comentários no código fonte e/ou de documentação gerada com Doxygen	−10%
Implementação na linguagem C ou resultante de mistura entre as linguagens C e C++	−30%
Programa compila com mensagens de aviso (*warnings*)	−50%
Plagiarismo	−100%