Laboratório III
Herança

1. Objetivo

Trabalho individual com 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 herança.

2. 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) Você deverá utilizar apenas um editor de texto simples (tais como o Gedit ou o Sublime) e o compilador em linha de comando, por meio do terminal do sistema operacional Linux.

3) 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.

4) 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. A título de sugestão, anote o seu 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.

5) 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.

6) Para melhor organização do seu código, implemente diferentes funções e faça a separação da implementação do programa entre arquivos cabeçalho (.h) e corpo (.cpp).

3. Tarefas

Implemente em C++ um jogo de batalha de conjuradores de monstros, onde o jogador irá duelar contra o computador e aquele que perder a batalha, perde o monstro conjurado.

Para isso, vocês devem seguir o seguinte fluxo de trabalho:

1. Ler Grimório do player.
2. Ler Grimório do computador.
3. Iniciar duelo.
    1. Duelo é realizado por turno.
    2. Cada jogador escolhe entre um monstro para o duelo
    3. Pode-se escolher em cada turno uma ação passiva e ativa do monstro.
    4. A cada ação, o monstro deve "falar" essa ação.
4. Jogador vencedor adiciona no seu grimório o monstro do perdedor.
5. Monstro do jogador perdedor é removido do grimório do jogador perdedor.

3.1. Especificação do Projeto

O código desenvolvido deve seguir as especificações abaixo:

1. Os Grimórios são arquivos no computador;
2. As operações de adição e remoção do monstro deve ser refletida em seus respectivos arquivos.
3. Ao ler os grimórios, os monstros devem ser colocados em uma lista.
4. O monstro do perdedor deve ser removido da lista e do arquivo e adicionado na lista e no arquivo do vencedor.
5. Caso um dos jogadores fique sem monstro, não é possível que o duelo ocorra.

3.2. Classificação dos monstros:

Os monstros deverão ter os tipos:

- Besta, seus ataques são com base em sua força física é sua fúria, tendo vantagens em cima de inimigos mágicos e desvantagens em inimigos alados.
- Mágicos, seus ataques são com base em seu poder mágico, tendo vantagens em cima de inimigos alados e desvantagens em inimigos bestas.
- Alados, seus ataques são com base em sua agilidade, tendo vantagens contra inimigos bestas e desvantagens contra inimigos mágicos.

As classes deverão seguir o esquema de hierarquia da Figura 1.

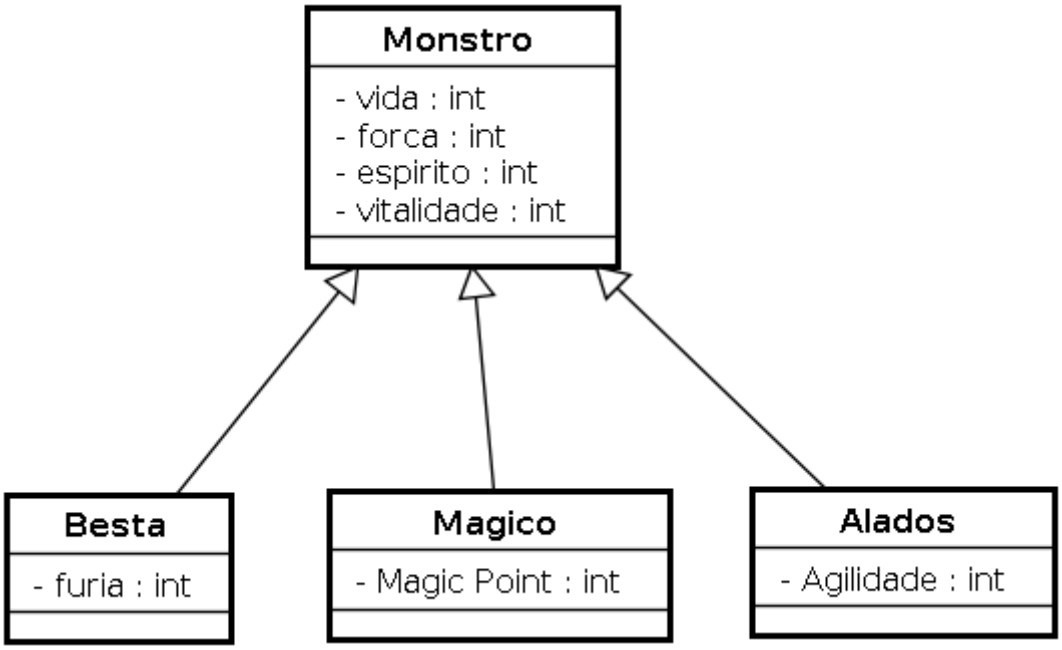

Figura 1: estrutura hierárquicas das classes.

3.3. Organização do Projeto

O código deverá ser devidamente comentado e anotado para dar suporte à geração automática de documentação no formato de páginas Web (HTML) utilizando a ferramenta Doxygen. Para maiores informações, você poderá acessar a página do Doxygen na Internet (http://www.doxygen.org/).

O código do projeto deve seguir a configuração de pastas e arquivos:

1. /bin – código executável
2. /src – código fonte
3. /docs – documentação
4. makefile
5. `README` – arquivo contendo informações sobre: configuração, compilação e execução. Também deve conter uma sessão com as informações sobre quais arquivos e as linhas que contêm: Uso de alocação dinâmica de memória. Sobrecarga de funções; Pilha e fila implementada para o livro e os usuários; Sobrecarga de Operadores.

4. 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. Além disso, você deverá submeter, até 23:59 do dia 09 de junho de 2017, 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;

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 para a segunda unidade, distribuídos de acordo com a seguinte composição:

Itens Avaliados	Nota máxima
Implementação das classes dos monstros com hierarquia.	0,50
Leitura e escrita de arquivos.	0,50
Implementação das vantagens e desvantagens dos monstros.	0,50
Implementação Duelo.	1,40
Verificação da existência de monstros dos arquivos.	0,20
Uso de construtores	0,10
Uso consistente de alocação dinâmica de memória e uso do This.	0,20
Aplicação adequada de sobrecarga de funções.	0,25
Documentação do código.	0,50
Uso de listas.	0,25
Sobrecarga de operadores	0,20
Criação do Makefile.	0,15
Criação do Readme.	0,25
Total	5

Por sua vez, o não cumprimento de algum dos critérios abaixo 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 do README e/ou Makefile	-0,5
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%