Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação

POO - Trabalho Prático

Prof. Maurício Acconcia Dias

Leonardo Bellini dos Reis nº 10284802
Marcelo Kiochi Hatanaka nº 10295645

São Carlos, 26 de Junho de 2018

Sumário

1 - Introdução

Este trabalho consiste no desenvolvimento de um jogo utilizando uma linguagem orientada a objetos, visando aplicar nele todos os conceitos vistos em aula. O jogo desenvolvido foi um “Super Trunfo” adaptado, e foi implementado na linguagem C++, no sistema operacional Linux.

Este relatório contém uma descrição detalhada do jogo: o que é, como jogar, regras, modos de jogo, e outras possibilidades. Além disso, contém o diagrama de classes a nível de implementação do código e uma lista das aplicações do conteúdo visto em aula implementados no jogo.

2 - O jogo

O jogo desenvolvido foi um “Super Trunfo” com adaptações.
O “Super Trunfo” , pela definição da Wikipedia, é:

“Super Trunfo é um jogo de cartas colecionáveis distribuído no Brasil pela Grow, que consiste em tomar todas as cartas em jogo dos outros participantes por meio de escolhas de características de cada carta (ex velocidade, altura, longevidade). O jogo comporta de dois a oito participantes e tem classificação livre, podendo ser disputado por qualquer pessoa alfabetizada.”

As adaptações feitas ao nosso jogo serão explicadas no decorrer do relatório.

2.1 - Como jogar

O jogo comporta apenas dois jogadores. As cartas do baralho são divididas igualmente entre os montes dos participantes, e o turno de quem começa é decidido aleatoriamente. Duas cartas (uma de cada jogador) e seus atributos aparecerão na tela (essas são as primeiras cartas de cada monte). A carta do jogador da vez será revelada, enquanto a do outro terá seu nome e valores dos atributos escondidos. O jogador da vez deve escolher um atributo que julga ter um valor superior ao mesmo atributo da carta do oponente. Após a escolha do atributo, a outra carta é revelada e os valores dos atributos são comparados. Aquele cuja carta possui o atributo de maior valor vence a rodada e ganha a carta do oponente, adicionando as duas no final do seu monte, e ganhando o direito de escolher o próximo atributo. Novas cartas aparecem na tela, e outra rodada começa. Ganha o jogador que ficar com todas as cartas do baralho.

2.1.1 - O Super Trunfo

Entre as cartas do baralho, uma delas é marcada como o “Super Trunfo”. O Super Trunfo é uma carta que não depende do valor de nenhum atributo para vencer a rodada. O jogador que tiver essa carta automaticamente ganha a rodada, a não ser que a carta do oponente seja do tipo A. Cada baralho possui: 1 Super Trunfo, ¼ das cartas de tipo A, e o restante das cartas de tipo B.

2.1.2 - Critério de desempate

No jogo original, caso ocorra um empate, as cartas permanecem na mesa e a próxima rodada decide quem as ganhará. No entanto, no nosso jogo, as cartas nunca empatam.

Se os atributos forem iguais, a carta “mais fraca” vence. Para decidir qual carta é a “mais fraca”, é feito um cálculo com as duas, comparando seus atributos com os maiores atributos presentes no baralho. Aquela que possuir a menor porcentagem em relação a esses valores é considerada a mais fraca e ganha a rodada. Esse sistema de desempate foi feito para bonificar o jogador que, com uma carta “pior” em mãos, faz uma boa escolha no jogo, ao mesmo tempo que pune aquele que, com uma carta “melhor”, faz uma má escolha.

Se as porcentagens das duas cartas forem iguais, o jogador que possuía a vez de jogar perde. Da mesma forma, esse critério foi feito para punir uma má escolha.

2.2 - Modos de jogo

O jogo possui mais de um modo de jogo. Pode-se escolher jogar sozinho, contra uma inteligência artificial; com outra pessoa; ou apenas assistir duas inteligências artificiais jogando (modo criado para testes).

2.2.1 - *Singleplayer* (*Player* x *Bot*)

Ao escolher esse modo, o jogador pode optar por 5 diferentes níveis de dificuldade:

Fácil: o jogador jogará contra o “Bot Iniciante”. Esse bot simula um jogador que não sabe jogar direito, não conhece as cartas, e não escolhe os atributos direito.

Médio: representado pelo “Bot Amador”. Esse bot simula um jogador que já conhece o jogo e o baralho. Ele sabe quais valores dos atributos são altos e os escolhe.

Difícil: representado pelo “Bot Experiente”. Ele simula um jogador experiente, que decorou todas as cartas do baralho. Ele sabe qual atributo da sua carta tem a maior chance de ganhar e o escolhe.

Muito Difícil: jogado contra o “Bot Profissional”. Esse bot joga como um profissional. Sempre que você ganha uma rodada, ele decora suas cartas ganhas e a posição delas no seu monte. Depois de rodar seu monte inteiro, ele sabe quais

são todas as suas cartas, e pode escolher um atributo com a certeza de que vai ganhar.

Impossível: jogado contra o “Bot Trapaceiro”. É um bot que trapaceia no jogo.
Ele olha sua carta antes de escolher, troca a própria carta para vencer a rodada, e aumenta seu ganho de pontos. Apesar do nome, essa dificuldade não é impossível de se vencer (Bot Profissional ganha, às vezes). A única forma de vencê-lo é possuindo o Super Trunfo.

2.2.2 - Multiplayer (Player 1 x Player 2)

Esse modo de jogo é jogado por duas pessoas, uma contra a outra. Como a única carta mostrada antes da escolha do atributo é a carta de quem escolhe, não há problema se o outro jogador vê-la, visto que não poderá fazer nada com essa informação.

2.2.3 - Demonstração (Bot x Bot)

Esse modo de jogo foi criado para testes dos bots, para verificar suas escolhas e compará-los em confronto. Nesse caso, ao escolher esse modo, você escolhe qual serão os níveis de dificuldade de cada bot.

2.3 - Pontuação

A fim de diminuir o tempo total do jogo, decidimos implementar um sistema de pontos e pontuação limite para o fim do jogo. Antes de começar um jogo, você escolhe a pontuação a ser atingida (ou escolhe não terminar o jogo pela pontuação). Assim, o jogo possui duas formas de acabar: ou ganha-se todas as cartas do baralho, ou atinge-se a pontuação escolhida.

Os pontos acumulados por cada jogador são ganhos em cada rodada que ganham. Os pontos são calculados com base na diferença percentual de cada carta (o mesmo cálculo para desempate). Assim, ganhar de uma carta “mais forte” lhe dará mais pontos, e ganhar de uma carta “mais fraca”, menos pontos.

2.4 - Baralhos

Criamos vários baralhos para o jogo. Dessa forma, antes de começar a jogar, você escolhe com qual baralho deseja jogar. Além disso, criamos a possibilidade de criar e personalizar novos baralhos (com novos temas) para jogar. Essa personalização envolve: criar baralhos com diferentes números de cartas, criar cartas e escolher seus nomes e atributos, alterar nome de um baralho criado, alterar

nome de um atributo, alterar nome de uma carta, alterar valor de um atributo de uma carta e deletar um baralho.

3 - Diagrama de classes

As classes criadas para o jogo foram: Atributo, Carta, Baralho, Jogador, Pessoa, BotFacil, BotMedio, BotDificil, BotProf, BotImpossivel.

Um Atributo possui um nome e um valor.

Uma Carta possui um conjunto de Atributos.

Um Baralho possui um conjunto de Cartas.

Um Jogador possui um Baralho e um método virtual puro para escolha de atributos.

Uma Pessoa é uma classe derivada de Jogador, e possui um método próprio de escolha de atributos.

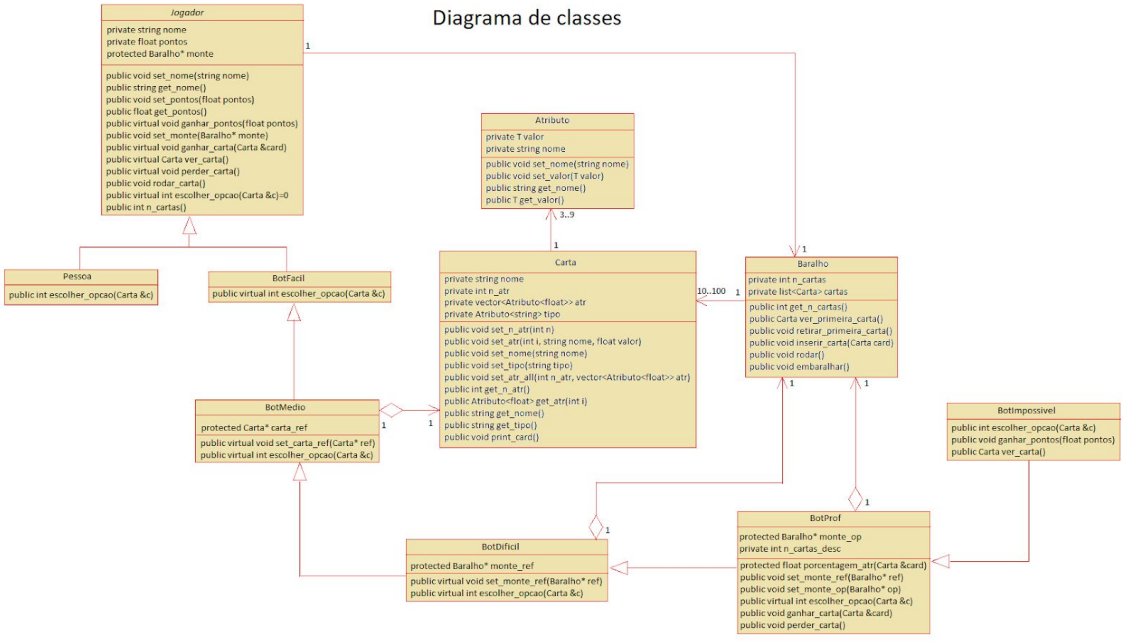
Um BotFacil é uma classe derivada de Jogador, e possui um método próprio de escolha de atributos.

Um BotMedio é uma classe derivada de BotFacil, e possui um método próprio para escolha de atributo.

Um BotDificil é uma classe derivada de BotMedio, e possui um método próprio para escolha de atributo.

Um BotProf é uma classe derivada de BotDificil, e possui um método próprio para escolha de atributo.

Um BotImpossivel é uma classe derivada de BotProf, e possui um método próprio para escolha de atributo.



4 - Aplicações do conteúdo visto

Neste tópico, citaremos alguns exemplos de aplicações do conteúdo visto que usamos na implementação do nosso jogo.

Classes

Criamos 10 classes para o desenvolvimento do jogo: Atributo, Carta, Baralho, Jogador, Pessoa, BotFacil, BotMedio, BotDificil, BotProf e BotImpossivel.

Herança

Utilizamos herança na criação das classes dos bots e da classe Pessoa. Todas essas classes herdam da classe Jogador.

Polimorfismo

Utilizamos polimorfismo para transformar objetos da classe Pessoa ou da classe de algum bot na classe Jogador. Esse polimorfismo foi utilizado na função que inicia o jogo, assim, o método chamado para escolha dos atributos era feito pela classe Jogador, mas o método executado era das classes derivadas, cada uma com seu próprio método.

Try-Catch

Utilizamos o tratamento de erros do try-catch em diversas funções, principalmente na abertura de arquivos. No entanto, decidimos não utilizá-lo em algumas situações, pois o retorno da função já era suficiente e polui menos o código.

Streams

Utilizamos arquivos para salvar todos os baralhos do jogo. Manipulamos arquivos diversas vezes, visto que implementamos diversas funções para alterar esses baralhos.

Sobrecarga de operadores

Utilizamos sobrecarga de operadores na função de imprimir os atributos de uma carta (<<), na função de inserir nova carta em um baralho (+=), e na função de retirar uma carta de um baralho (--).

Template

Utilizamos templates na criação da classe Atributo. O valor do atributo é uma variável template.

Manipulação de strings

Manipulamos strings para formatar impressão dos atributos das cartas.

Pré-processador

Utilizamos diversos “defines” para facilitar legibilidade e modularização do código.

STL

Utilizamos containers da classe <list> para agrupar as Cartas na classe Baralho. Além disso, utilizamos a classe <vector> diversas vezes para agrupar strings na manipulação dos baralhos.

Alocação dinâmica de Memória

A maioria dos objetos instanciados no arquivo principal foram dinamicamente alocados.

Threads

Utilizamos threads para criação de nomes aleatórios (na escolha de nomes das cartas, na criação de baralhos). Uma thread criava uma consoante aleatória, enquanto a outra criava uma vogal aleatória.

5 - Observações

5.1 - Interface gráfica

O jogo não possui interface gráfica, no entanto, formatamos a saída do terminal para que a visualização e a dinâmica do jogo ficassem bem organizadas. Além disso, tratamos todos os erros da entrada pelo terminal, ou seja, se o usuário tentar inserir algo inválido, o jogo continuará executando e apenas um aviso aparecerá na tela.

5.2 - Compilação e execução

Para compilar, utilize o Makefile: make
Para executar, também : make run

5.3 - Baralhos bloqueados

Alguns baralhos estão bloqueados para modificações. São os primeiros baralhos, criados junto com o jogo. Os baralhos modificáveis são aqueles criados pela ferramenta de criação de baralhos do jogo. Tomamos a decisão de bloqueá-los

para que o usuário nunca perca as informações originais. Se for preciso, pode-se criar cópias desses baralhos com as modificações que desejar.

6 - Conclusão e considerações finais

O projeto foi finalizado dentro do prazo, e segue a especificação fornecida. Para compensar a falta de uma interface gráfica, tentamos organizar a impressão das informações ao máximo, além de adicionar vários elementos para aumentar o número de aplicações do conteúdo visto. No geral, consideramos o resultado satisfatório, destacando a implementação da manipulação de novos baralhos e dos bots, que ficaram bem interessantes.

Com a realização desse trabalho, foi possível aumentar nosso conhecimento em relação à programação orientada a objetos, assim como aprimorar nossa lógica e nossas técnicas de programação.