UNIVERSIDADE FEDERAL DE MINAS GERAIS

PABLO CORREA COSTA
SARAH ALINE CAETANO CRUZ
YANNE ASSIS ALVES

JOGO DE CARTAS: PACIÊNCIA

BELO HORIZONTE
2022

RESUMO

Com o objetivo de aplicar os conceitos de abstração, encapsulamento, composição, herança e polimorfismo em Boas Práticas com base na programação orientada a objetos, foi desenvolvido o jogo Paciência a partir da linguagem C++ por alunos da disciplina Programação e Desenvolvimento de Software II da Universidade Federal de Minas Gerais (UFMG), fazendo com que tais conceitos fossem aprendidos de forma mais significativa em prática.

Palavras-chave: Paciência, Jogo de Cartas, Programação Orientada a Objetos, Programação em C++, Boas Práticas.

ABSTRACT

With the objective of applying the concepts of abstraction, encapsulation, inheritance and polymorphism in Good Practices with object-oriented programming, a game of Solitaire was developed using the C++ language, by students of the class Programming and Software Development II, at the Federal University of Minas Gerais (UFMG), making such concepts more approachable in practice.

Keywords: Solitaire, Card Game, Object Oriented Programming, C++ Programming, Good Practices.

1. Introdução

Para auxiliar no aprendizado de boas práticas na programação orientada a objetos, os alunos da disciplina Programação e Desenvolvimento de Sistemas II foram orientados a desenvolver um programa que satisfizesse o que foi ensinado em sala de aula. Foi-se, então, escolhido o jogo Paciência, seguindo o tema “jogos de cartas”, para tal finalidade.

Paciência, também conhecido como Solitaire ou Klondike, é um jogo de cartas muito famoso pela sua peculiaridade de possibilitar ao usuário jogar sem a necessidade de uma outra pessoa. O jogo requer um baralho de 52 cartas e seu objetivo principal é organizar, a partir de um deck embaralhado, 4 pilhas finais (cada uma de um naipe diferente) em ordem ascendente - de Ás a Rei.

Para auxiliar o jogador, é criado um tableau que consiste em 7 pilhas de cartas distribuídas de formas que a primeira tenha uma carta, a segunda duas cartas, a terceira três e assim por diante até que existam sete pilhas. Apenas a carta do topo da pilha estará virada para o jogador.

No tableau, as cartas podem ser movimentadas de uma pilha para outra, desde que a carta que será movida seja da cor contrária e de um número inferior àquela do topo da pilha de destino. Por exemplo, caso eu tenha uma pilha cuja carta no topo seja um 10 de copas(vermelho) e possua uma pilha com um 9 de espadas(preto), eu posso mover o 9 para a pilha do 10, desbloqueando uma nova carta na antiga pilha do 9 e transformando o 9 no topo da pilha do 10.

O jogo conta com duas modalidades sendo, vira uma carta e vira três cartas. O primeiro tipo apenas uma carta é virada por vez no estoque, essa modalidade é indicada para quem não é muito experiente no jogo e quer aprender as regras. A segunda modalidade três cartas são viradas por vez no estoque, porém apenas a carta de cima do descarte pode ser utilizada, essa modalidade é mais difícil e necessita de mais estratégias para vencer, pois é muito comum ficar sem jogadas (GENIOL, 2022), sendo esta a modalidade escolhida neste projeto.

2. Referência Bibliográfica

O jogo paciência já foi desenvolvido antes por Lucy Mari Tabuti, Sandra Puga e Luiz Antonio Moura Neto na linguagem Java, no ano de 2014. O projeto foi desenvolvido com o objetivo de ilustrar alguns dos assuntos abordados na disciplina de Estrutura de Dados possibilitando que o aprendizado torne-se significativo ao estudante. Para isso, foi utilizado o estudo de caso denominado “Projeto-Paciência”, o qual foi ilustrado por meio de um Objeto de Aprendizagem (OA) que, de forma lúdica, utiliza as estruturas de recursividade, pilha, fila e lista ligada.

A metodologia de construção do objeto de aprendizagem será apresentada na forma como os professores abordaram cada tópico e que será ilustrado por meio de linguagem algorítmica e de imagens. No início do desenvolvimento do Projeto Paciência, os alunos identificam como é a manipulação de dados que são representados pelas cartas do baralho como uma variável composta de número e naipe. A partir da especificação da variável Carta, os professores sugerem que os alunos utilizem um vetor (array), uma estrutura que permite o armazenamento de n elementos do mesmo tipo de dado, para armazenar as 52 cartas do baralho.

A cada nova partida do jogo, o baralho necessita ser embaralhado, isto é, as cartas no vetor precisam ser misturadas aleatoriamente. O conceito de recursividade é de difícil assimilação por parte dos alunos e este desenvolvimento os auxilia na compreensão do processo.O próximo passo é discutir como acontece a manipulação dos dados. A primeira estrutura apresentada é a Pilha. Na estrutura de Pilha, as informações são armazenadas na ordem inversa à sua inserção, ou seja, o primeiro elemento a ser inserido é o último a sair. Após entender o conceito, os alunos discutem o algoritmo para o seu desenvolvimento em termos computacionais. A próxima estrutura a ser estudada é a Fila, onde o primeiro elemento que entra é o primeiro a sair.

Enfim, o conceito de lista ligada é apresentado. Diferente do vetor, as informações em uma lista ligada estão organizadas de maneira esparsa na memória do computador, por este motivo, as informações precisam estar ligadas umas às outras. Para isso, cada elemento deve saber quem são os seus antecessores e quem são os seus sucessores. Ao abordar este tópico o professor explica o conceito de “nó”. O “nó” é um registro, adicionado dos campos para armazenamento da referência ao antecessor e ao sucessor. Uma lista ligada aceita a manipulação aleatória dos dados, permitindo que inserções e remoções sejam realizadas em qualquer posição.

A partir do momento em que as estruturas de dados foram estudadas e desenvolvidas, inicia-se o processo de implementação das estruturas, bem como, das regras lógicas do jogo. Para o início do jogo, as cartas do baralho devem ser distribuídas em sete listas ligadas. As demais cartas que não foram distribuídas nas listas anteriormente mencionadas devem ser agrupadas para que possam ser exibidas e utilizadas de forma que a primeira carta inserida seja a primeira a ser retirada.

Além disso, pela necessidade do jogo, ao inserir uma nova carta nesse agrupamento, a mesma deve ser agrupada depois de todas as outras, evidenciando a estrutura FILA (FIFO – First in/First out, primeiro a entrar, último a sair). Durante o jogo, conforme as cartas são manipuladas, elas são organizadas por naipe em quatro blocos que são criados através das estruturas de dados PILHAS. Pilhas são estruturas de dados que seguem o conceito de LIFO (Last in/First out, tradução: último que entra, primeiro que sai). Durante o desenvolvimento do jogo, as estruturas de filas, pilhas e listas ligadas trocam informações entre si, assim, a lógica de como as informações são enviadas de uma para outra estrutura segue as regras do jogo, sendo este o motivador para o aprendizado das estruturas de dados e o que facilita a sua implementação.

Ao decorrer do desenvolvimento do objeto de aprendizagem Projeto Paciência, pelos alunos, observa-se que o aprendizado de um conteúdo abstrato pôde ser realmente entendido de forma lúdica. A construção do aprendizado, neste caso, se deu pela construção do jogo de baralho usando uma linguagem de programação e, principalmente, pelo entendimento das regras do jogo utilizando um baralho físico com as suas 52 cartas e de sua manipulação para entender as regras do jogo e aplicá-la aos conceitos de recursão, pilha, fila e lista ligada.

3. Desenvolvimento

Para o desenvolvimento deste trabalho, na linguagem C++, foram usadas quatro classes, sendo elas: deck, carta, pilha e paciência.

A classe Deck é responsável por embaralhar as cartas para que o jogo comece. E nesta classe temos as funções:

- Popula: popula o deck com as 52 combinações de cartas existentes entre 13 números e 4 naipes.
- Embaralhar: que gera um número aleatório até 52 (número de cartas no deck) e faz a troca de lugar da carta no lugar i com a carta no lugar sorteado, no fim o baralho estará misturado.
- PrintaDeck: Passa pelas 52 cartas e printa.
- PopulaVetor: A função de popular com cartas do deck com uma pilha e também é usado para popular o tableau.

A classe Carta é responsável por pegar o naipe, a cor das cartas e também para virar carta.Todos esses atributos são passados para a classe paciência, pois lá isso será realmente usado. E nesta classe temos as funções:

- ViraCarta: muda o lado da carta.
- GetValorPaciência: Ela define inteiros para as cartas A,T,J,Q,K.
- GetValor: recebe o valor da carta.
- GetNaipe: recebe o naipe da carta.
- GetIsVirada:

A classe pilha é um TAD Pilha responsável por construir as 7 pilhas do tableau, as 4 pilhas de naipes final e a pilha do deck (mão do jogador). E nesta classe temos as funções, que serão compostas na classe Paciência posteriormente.:

- Push: Empilha uma carta.
- Push_Copia_Do_Valor: Função específica para popular um vetor de pilhas, utiliza apenas os valores da carta.
- Clear: Zera uma pilha.

- Vazia: Verifica se a pilha está vazia
- Topo: Retorna a carta do topo da pilha.
- Remove: Remove a carta com o índice requerido.
- Tamanho: Retorna o número de cartas da pilha.
- Pop: Desempilha uma carta.
- Move: Adiciona uma carta no topo de uma pilha destino e retira uma carta do topo de uma pilha de origem.

A classe Paciência é responsável por fazer todo o jogo funcionar, lá é onde funciona toda lógica do código.

Lá usamos a classe Deck para criar um deck de paciência. Com as funções de popular o deck e de embaralhar para criá-lo aleatoriamente. E depois populamos o tableau com as pilhas de cartas previamente populadas. Ela também é responsável por mostrar as cartas descartadas da rodada, começar e terminar um movimento igualmente, garantir que o movimento seja invalido a partir de uma pilha vazia, permite o movimento do rei para casa vazia.

Nesta classe temos as funções:

- CartasMao: leva 3 cartas para mão;
- PrintDetalheNaipe: printa a tela do jogo e atualiza a cada movimento;
- IsMovimentoNaipeValido: Determina os movimentos válidos e também que o naipe só pode ser movido para uma casa do naipe oposto e também define se a carta pode ser movida para a pilha final.
- MoveNaipe: O movimento só pode ser feito se for com naipe válido;
- MoveEntrePilha: Verifica o movimento entre pilhas e vira cartas caso ocorra;
- MoveDeckParaPilha: Realiza um movimento do deck para uma pilha do tableau;
- MovePilhaParaPilha: Realiza um movimento entre as pilhas do tableau;
- IsMovimentoPilhaValido: Um valor deve ser estritamente 1 abaixo da carta de destino;
- JogoCompleto: Finaliza o jogo caso todas as casas da pilha final estejam completas.

4. Experimentos Computacionais

Para executar o programa, primeiramente deve-se baixar os arquivos do link do github (link), que também foi disponibilizado no email da professora. Com isso, basta ir até a pasta onde os arquivos foram baixados e extraí-los.

No prompt de comandos (ou em qualquer editor de código de preferência) basta ir até o local dos arquivos e digitar o comando *Make*. Os arquivos *.o* serão gerados e para iniciar o jogo basta digitar *./main*.

Ao testar o código, percebe-se que ele funciona de forma esperada na visão do usuário (Figura 1), sendo possível jogar normalmente o jogo de Paciência, seguindo suas regras.

Figura 1- Visão do Usuário

Fonte: Acervo pessoal.

Para jogar, é preciso seguir as legendas que aparecem para o usuário, começando com a escolha do tipo da próxima jogada entre as opções:

1. Mover a carta entre pilhas
2. Mover a carta para pilha final
3. Comprar
0. Sair

Após escolher uma opção, o programa responde de forma esperada, seja permitindo que o jogador compre três cartas (opção 3); mova uma carta das pilhas enumeradas de 6 à 0 ou de sua mão enumerada 7 para alguma outra pilha onde o movimento seja permitido (opção 1); guarde uma carta na pilha final, desde que junto com seu tipo e numeração ordenada (opção 2); ou parando o jogo (opção 0).

O jogo mostrou-se satisfatório em questão de desempenho, funcionando como esperado, seja durante a partida ou após uma vitória ou derrota.

5. Conclusão

Diante do apresentado no decorrer deste trabalho, a implementação do jogo de cartas Paciência, utilizando a linguagem C++, mostrou-se muito satisfatória, possibilitando um código em que pudesse rodar o jogo, juntamente aos conceitos de Abstração, Encapsulamento, Herança e Polimorfismo contribuindo com o aprimoramento e repertório de conhecimento dos que participaram no projeto. O objetivo foi alcançado, possibilitando que o jogo seja executado sem nenhum erro aparente. Ademais, como propostas para trabalhos futuros, propõe-se a implementação de uma interface gráfica almejando um serviço mais profissional e que possa ser disponibilizado para diversos usuários.

6. Bibliografia

Tabuti, Lucy Mari, Sandra Puga, e Marcos Moreira Brito. "Construção do objeto de aprendizagem Paciência como recurso didático para o ensino da disciplina Estruturas de Dados." *Nuevas Ideas en Informática Educativa Memorias del IXX Congresso Internacional. TISE*. 2014. Disponível em: http://www.tise.cl/volumen10/TISE2014/tise2014_submission_29.pdf. Acesso em: 20/11/2022.

Gonçalves, Ricardo Emanuel Ferreira. "Jogo digital para o ensino dos fundamentos da programação." (2011). Disponível em: https://repositorio-aberto.up.pt/bitstream/10216/62088/1/000149236.pdf. Acesso em: 20/11/2022.

GENIOL. “Paciência.” Disponível em: https://www.geniol.com.br/paciencias/paciencia/. Acesso em: 28/11/2022.

Juraj's Blog. “Implementing Solitaire in C”. Disponível em: https://jborza.com/games/2020/07/12/solitaire-cli.html. Acesso em: 28/11/2022.

Professor Elétrico. “Compilando Código C/C++ com makefile”. Disponível em https://professoreletrico.com/posts/programacao/compilando-codigo-c-cpp-com-makefile/. Acesso em: 01/12/2022