Revivendo a Infância: Um Jogo Desenvolvido Em Python

Everton Bruno Silva dos Santos[1]

[1]Universidade Estadual de Feira de Santana (UEFS)
Caixa Postal 44036-900 – Feira de Santana – BA – Brazil
Av. Transnordestina, s/n, Novo Horizonte

evertonbrunosds@gmail.com

Resumo. *O software descrito ao longo desse relatório foi implementado por meio de algoritmos, visando modernizar um jogo de cartas criado por um grupo de amigos. Para isso, o produto segue todas a regras do jogo original informadas por seus idealizadores, de modo que venha a ser uma solução que resolve o problema proposto.*

1. Introdução

Um grupo de amigos quando jovens inventaram um jogo, porém mesmo com o passar dos anos eles sempre se recordam do que fizeram na época de suas infâncias, então visando reviver essa nostalgia nos dias atuais, resolveram moderniza-lo. Assim, como propósito de tornar esse jogo de cartas mais condizente com as tecnologias de hoje, foi necessário refaze-lo eletronicamente.

Este relatório tem por objetivo descrever o que foi necessário para fazer com que esse projeto pudesse funcionar nos computadores utilizando-se dos recursos contidos na linguagem de programação Python. Ao longo da sessão de desenvolvimento procura-se abordar os questionamentos que possibilitaram a criação desse software, bem como abordar o modo como a solução está organizada.

2. Desenvolvimento

A solução é organizada da seguinte forma:

1. As bibliotecas *os* e *random* são importadas, para só então serem criados os procedimentos e funções;

2. Após o procedimento principal ter entrado em execução, o processo de *loguin* e cadastro é feito a medida em que os usuários inserem seus *niknames* no programa;

3. Ao longo de toda a execução do programa variáveis locais, matrizes, listas, tuplas, dicionários e objetos são criados.

4. Os arquivos situados dentro da pasta “GameData” como “cartas.txt” e “PlayerData.txt” são inseridos na memória *RAM* do computador para logo se tornam listas e posteriormente matrizes que são incridas atributos de um objeto;

5. O laço principal é iniciado dando a opção do primeiro jogador escolher o modo de jogo, sendo manual ou automático. Por fim, as cartas são exibidas em quanto o mesmo se revesa na escolha do tipo de disputa com o segundo jogador.

Para que escrever e ler arquivos? O processo de escrita de arquivos é necessário a este projeto uma vez que usuários devem ser cadastrados caso ainda não tenham seus

nicknames disponíveis no arquivo “PlayerData.txt”. Enquanto que, o processo de leitura corresponde a possibilidade de acessar os dados das cartas usadas no jogo, bem como acessar os dados dos usuário que já foram cadastrados, possibilitando a exibição de suas respectivas pontuações.

Desse modo, na linguagem de programação *Python*, a leitura e escrita de arquivos fica a cargo da função *open* [python.org 2016]. Além disso, ela possibilita a inserção de argumentos, esses que fizeram com que o jogo fosse capaz de ler e escrever arquivos sempre que necessário.

Como as classes foram usadas? Buscando usar uma estrutura capaz de armazenar dados de forma agrupada sem haver a necessidade de saber onde estão localizados seus índices para buscar determinada informação, uma classe foi definida e por meio desta um objeto foi criado contendo métodos capazes de retornar os dados dos jogadores da partida, bem como atributos que compreendem as cartas das mãos de ambos os jogadores, todas as cartas da pilha e além disso o objeto conta também com a possibilidade de atualizar os percentuais de vitória dos jogadores envolvidos na partida. No projeto, o objeto criado recebe em seus atributos as matrizes oriundas dos arquivos necessários para o jogo, sendo esses o arquivo de cartas e o que contém os dados dos jogadores.

Em que uma função ou procedimento pode ser útil? Na programação é possível efetuar o reaproveitamento de linhas de comando através de procedimentos e funções, mas no caso específico das funções espera-se o retorno de algum dado como resultado e no caso dos procedimentos espera-se apenas que algum comando seja efetuado, o que caracteriza o procedimento como uma função infrutífera [Forbellone 2000].

Visando modularizar o projeto projeto, funções e procedimentos foram criadas, por exemplo para trabalhar com a leitura e processamento de dados. Evitados a repetição exaustiva de determinados trechos do código, o que o tornaria mais extenso e difícil de compreender.

Como os dados importados são trabalhados na memória RAM? Há princípio, ao observar como se organizada a estrutura das informações contidas no arquivo “cartas.txt” notou-se que seria possível transforma-lo em uma matriz, o que veio a ser útil ao longo do projeto.

Após a importação, um laço percorre cada linha do arquivo e o comando *split(';')* faz com que cada linha se torne uma lista na memória, separando seus dados após cada sinal de **ponto e vírgula** e os inserindo em uma outra lista, criando uma matriz. Esse processo tem a finalidade de acessar cada dado separadamente por meio de índices. Para que os principais dados do jogo pudessem ser manipulados em outras funções de maneira mais compreensível, todas as matrizes criadas foram armazenadas em um objeto.

Como as cartas que correspondem as linhas de uma matriz foram embaralhadas? Levando em conta que as cartas deve ser embaralhadas antes de cada jogo, por meio da importação da biblioteca *random*, se fez possível a utilização da função *randint* que é capaz de sortear números em um intervalo de outros dois valores numéricos.

Assim o processo de embaralhar as cartas do jogo ocorre por meio da remoção de elementos da matriz de cartas de forma aleatória, sendo que a cada vez que um elemento é removido da matriz, o mesmo é inserido em uma outra matriz de forma desordenada.

Ao fim, uma nova matriz contendo as cartas desordenadas é dada como retorno da função de desordenamento.

Para isso, foi criada uma função contendo em si um laço que permanece ativo em quanto a matriz possuir elementos. Logo, o comprimento (quantidade de elementos) da matriz é usado para ser o valor máximo a ser sorteado em uma faixa de valores que inicia no primeiro índice da matriz.

Como as cartas que correspondem as linhas de uma matriz foram ordenadas? Quando o usuário decide o modo de disputa as cartas devem ser ordenadas por valor, energia e força. Além disso, ao ser selecionado o modo de jogo manual as cartas devem sempre estar em ordem alfabética, a partir desses fatos surgiu a necessidade de se utilizar um método de ordenação.

Ainda que o *BubleSort* não seja o método de ordenação mais eficiente, ao avaliar que o arquivo que contém as cartas não é muito extenso e que esse algoritmo é de simples compreensão, o mesmo foi adotado para efetuar quatro possíveis formas de ordenação, sendo: alfabética, valor, força e energia.

Possuindo dois laços, esse algoritmo foi implementado no jogo de modo que, verificasse se determinado elemento da coluna de uma matriz era menor que um outro contido na linha seguinte, afim de trocar as posições dessas linhas caso a condição fosse verdadeira. Fazendo com que pouco pouco os dados fossem ordenados de forma crescente.

Como a disputa de jokempô foi desenvolvida? Levando em conta que o modo de disputa jokempô se iguala ao jogo **"pedra, papel e tesoura"**, nota-se que as possibilidades de combinação desses três itens podem resultar em nove possibilidades. E ao considerar que em cada partida poderão haver apenas dois jogadores, a solução para esse modo de disputa foi desenvolvida num esquema que pode ser observado na (tabela 1).

Tabela 1. Descrição do processo.

POSSIBILIDADES	RESULTADOS	QUEM VENCE
PedraPedra	True,True	EMPATE
PedraPapel	False,True	JOGADOR 2
PedraTesoura	True,False	JOGADOR 1
PapelPedra	True,False	JOGADOR 1
PapelPapel	True,True	EMPATE
PapelTesoura	False,True	JOGADOR 2
TesouraPedra	False,True	JOGADOR 2
TesouraPapel	True,False	JOGADOR 1
TesouraTesoura	True,True	EMPATE

O processo ocorre da seguinte maneira:

1. Os valores de jokempô de ambos os jogadores da disputa são concatenados.

2. O resultado dessa concatenação é dada como chave de um dicionário que conta com todas as possibilidades de jokempô.

3. O dicionário retorna uma *tupla* cujos valores são as combinações de *true* e ou

false em diferentes ordens.

4. Essa *tupla* obtida por meio da chave é comparada em condicionais que verificam se a disputa deu empate, bem como se o jogador um ou dois venceu a disputa de acordo a ordem e valores contidos na *tupla*.

3. Conclusões

O relatório apresentou a elaboração da solução em software para o problema proposto, esse que se referente a criação de um jogo eletrônico baseado em um pré-existente inventado por um grupo de amigos durante suas infâncias. A solução foi construída na linguagem de programação python constituindo uma interface gráfica de linha de comando, sendo as informações inseridas na memória do computador não só por meio de arquivos, mas também pela entrada de dados pelo teclado do usuário. A solução resolve o problema proposto fazendo o uso de recursos presentes na linguagem python como variáveis, funções, procedimentos, classes, objetos, listas, condicionais, estruturas de repetição matrizes e dicionários.

No decorrer do desenvolvimento do produto, problemas tiveram que ser sanados, processo descrito ao longo do desenvolvimento desse relatório. Algumas soluções necessitaram de um certo nível de compreensão textual e domínio de programação.

Os pontos positivos desse software correspondem ao modo como os dados são importados, processados, armazenados e exibidos, considerando em cada ação as regras do jogo original implantadas em forma de algoritmo. Todavia, os pontos de melhoria desse software giram em torno do fato de que o arquivo de cartas não pode conter nenhuma linha que não siga o padrão típico das cartas usadas no jogo, do contrário o mesmo pode apresentar falhas.

Referências

Forbellone, A. V. L. (2000). *Lógica de Programação: A Construção de Algoritmos e Estrutura de Dados*. Makron Books.

python.org (2016). Input and output - reading and writing files. Disponível em: <https://docs.python.org/3/tutorial/inputoutput.html>. Acesso em: 14 Ago 2019.