MAC0216 - Técnicas de Programação I

Terceiro Exercício-Programa (EP3)

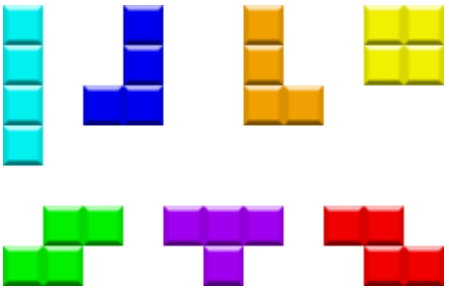
Programação Orientada a Objetos

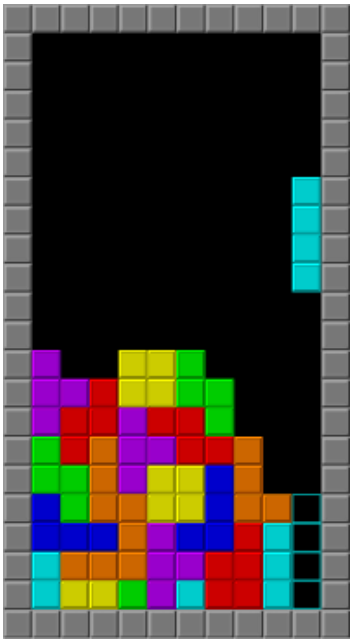
Prazo de entrega: até às 23:59 de 11/12/2024

(Atenção: o exercício pode ser realizado em duplas)

1. Introdução: O Jogo Tetris

O Tetris é um jogo eletrônico clássico de quebra-cabeça, em que o objetivo é organizar peças chamadas **tetriminos** que caem de uma área superior para preencher linhas horizontais na tela. Os **tetriminos** são peças formadas por 4 blocos conectados, com diferentes formas (I, J, L, O, S, T e Z), como mostrado na figura abaixo à esquerda[1]. Cada forma de peça tem uma cor específica no jogo, para facilitar a identificação. A figura abaixo à direita mostra uma imagem de uma tela de Tetris clássico.





Fonte das figuras: Wikipédia (https://pt.wikipedia.org/wiki/Tetris)

A tela do jogo é um retângulo vertical onde as peças caem. Geralmente, ele é composto por uma grade de 10 colunas e 20 linhas. O objetivo do jogo é completar linhas horizontais preenchendo todos os espaços sem deixar lacunas. Quando uma linha é completada, ela é eliminada, e as linhas acima dela se deslocam para baixo.

[1] A Wikipedia contém algumas informações adicionais interessantes sobre os tetriminos: https://en.wikipedia.org/wiki/Tetromino.

Mecânica do Jogo

- **Queda das Peças**: Os tetriminos caem do topo da tela de jogo em direção ao fundo, empilhando-se. Pressionando teclas específicas do teclado, o jogador pode:
    - Girar a peça atual (90° por vez).
    - Mover a peça atual horizontalmente.
    - Acelerar a queda da peça (ou "soltar" a peça diretamente no topo da pilha de peças no fundo da grade da tela).
- **Gerador de Peças**: A cada vez que uma peça é acomodada na pilha, uma nova peça é gerada e inserida no topo da tela. O jogo utiliza um algoritmo para gerar as peças de modo (pseudo)aleatório, garantindo que o jogador receba todas as formas de peças possíveis em um intervalo razoável.
- **Preenchimento de Linhas**: Quando uma ou mais linhas completas são formadas na tela, elas desaparecem e o jogador ganha pontos. A pontuação varia de acordo com o número de linhas eliminadas de uma vez: 1 linha = **Single** (pontuação baixa); 2 linhas = **Double**; 3 linhas = **Triple**; 4 linhas = **Tetris** (pontuação alta).
- **Fim do Jogo**: O jogo termina quando um novo tetrimino não pode ser posicionado porque a pilha de peças alcançou o topo da tela.

Para ver um versão clássica do jogo em funcionamento, acesse https://www.1001jogos.com.br/g/tetris

2. O Exercício-Programa: o Jogo Textris

Neste EP, vocês irão desenvolver em Python uma implementação orientada a objetos do **Textris** – uma versão do clássico jogo Tetris adaptada para o modo texto. A mecânica do jogo será baseada na descrição da Seção 1, mas com a particularidade de que tanto a tela quanto as peças serão representadas no terminal utilizando caracteres.

2.1 Funcionalidades Requeridas

O jogo deve oferecer as seguintes funcionalidades:

1. **Configuração da grade de jogo**:
    - Ao iniciar uma partida, o usuário deve poder definir o tamanho da grade, especificando o número de linhas e colunas.
2. **Criação e movimentação de peças**:
    - Durante a partida, o jogo deve criar, uma por vez, peças de diferentes formatos (sempre compostas por 4 caracteres).
    - Para distinguir os formatos, o jogo usará caracteres variados no lugar de cores.
    - O usuário poderá mover a peça atual para baixo, para a esquerda ou para a direita dentro da grade até que ela seja acomodada na pilha de peças. Assim que isso ocorrer, o jogo deve gerar uma nova peça no topo da tela.
3. **Rotação das peças**:

    - O jogador deve ter a opção de rotacionar a peça atual para a esquerda ou para a direita.
4. **Salvamento e retomada da partida**:
    - O jogador deve poder interromper a partida a qualquer momento e salvar o progresso em um arquivo.
    - O nome do arquivo de salvamento deve seguir o formato: **[nome do jogador] + [data e hora da gravação]**.
    - O jogador poderá carregar uma partida salva posteriormente, retomando-a exatamente de onde parou.
5. **Pontuação e rankings**:
    - Durante a partida, o jogo deve exibir continuamente a pontuação atual.
    - O jogador deve poder visualizar as melhores pontuações já alcançadas no jogo em execução, acompanhadas dos nomes dos respectivos jogadores.

Observação Importante

Diferentemente do Tetris tradicional, no Textris do EP3, as peças criadas no topo da tela **não cairão automaticamente** em direção ao fundo. Elas só se movimentarão quando o jogador pressionar as teclas apropriadas (como as setas para baixo, esquerda e direita).

Essa decisão foi tomada para simplificar o desenvolvimento do EP, uma vez que implementar a queda automática, simultânea à interação do jogador, exigiria o uso de técnicas que não são abordadas em MAC0216.

2.2 Classes

Seu programa precisará ter pelo menos as classes abaixo, que devem se integrar:

- Jogo
- Partida
- Peça

Há outras classes que podem fazer sentido a depender da implementação feita:

- Tela
- Jogador
- Ranking
- Subclasses de Peça

2.3 Documentação

Os comentarios nos códigos-fonte explicando as classes e o programa principal precisarão ser suficientes para que o programa **doxygen** possa gerar uma documentação do software em arquivos html e demais formatos necessários para que a documentação possa ser lida em um navegador web. Certifique-se que a documentação seja gerada corretamente seguindo as recomendações do doxygen para codigos em Python: https://www.doxygen.nl/manual/docblocks.html#pythonblocks .

2.4 Automação da Geração da Documentação e da Execução dos Testes

A geração da documentação e a execução dos testes devem ser automatizados por meio do programa **make**. O **Makefile** incluído na entrega precisa conter pelo menos os seguintes alvos:

- all: que vai gerar a documentação e rodar os testes
- doc: que vai apenas gerar a documentação
- tests: que vai apenas rodar os testes
- clean: que vai limpar todos os arquivos intermediários gerados pelos outros alvos acima

2.5 Testes

Para garantir que todos os métodos das classes estão corretamente implementados, você deve escrever testes unitários usando pytest.

2.6 Repositório de Código

Você deverá usar o git e o github para o controle de versões do código e também para trabalhar de forma colaborativa com a sua dupla.

Atenção: crie o repositório do EP3 no github como um repositório privado, de modo que apenas as pessoas autorizadas por você possam ter acesso ao código. Deixar o repositório público pode levar a casos de plágio.

Na entrega do EP3, um dos membros da duplas deverá enviar no e-Disciplinas tanto um zip dos arquivos da implementação quanto o link do repositório. A dupla também deve conceder acesso ao repositório no github para a professora e os monitores (logins: kellyrb, liviaruegger e matheusilver).

O arquivo README.md do repositório da sua implementação do EP3 deve possuir no mínimo as seguintes seções:

- AUTOR(ES): com nome, NUSP e endereço de e-mail
- DESCRIÇÃO: explicando o que o programa faz
- COMO EXECUTAR: explicando como o programa deve ser executado, os eventuais argumentos de linha de comando do programa (o que é o argumento, como ele precisa estar formatado, etc.). Fale também sobre como a documentação é gerada.
- TESTES: fornecendo informações sobre os testes embutidos na entrega feita.
- DEPENDÊNCIAS: descreverndo o que é necessério para rodar o programa. Informações como a versão do interpretador Python e o SO onde você executou e sabe que ele funciona são importantes de serem colocadas aqui.

É importante destacar que os commits feitos no repositório serão analisados na correção. Para o caso do desenvolvimento do EP em dupla, é esperado que os dois integrantes da dupla tenham contribuições significativas no código no repositório.

3 Exemplos de Execução

No EP3, vocês estão livres para definir a forma de apresentação do jogo em modo texto e também como se dará as interações com o usuário do jogo. Por exemplo, vocês podem escolher as teclas que quiserem para controlar a movimentação das peças no jogo.

Na sequência, são mostrados alguns exemplos de possíveis interações entre um jogador e o Textris, apenas para fornecer algumas ideias a partir das quais vocês possam trabalhar. O que está destacado em amarelo é digitado pelo usuário.

1) Início do jogo:

```
$ python ep3.py
*** Jogo Textris - um tetris em modo texto ***
Opções do jogo:
- <i> para iniciar uma nova partida
- <c> para carregar uma partida gravada e continuá-la
- <p> para ver as 10 melhores pontuações
- <s> para sair do jogo
Digite a opção desejada: i
Digite o nome do jogador: Kelly
Digite o número de linhas da tela do jogo: 8
Digite o número de colunas da tela do jogo: 12
```

2) Início da partida (tela do jogo impressa em um terminal “limpo”). A peça inserida pelo jogo no topo da tela (escolhida de forma aleatória) é a de forma J :

```
 ____________
|      *     |
|      *     |
|     **     |
|            |
|            |
|            |
|            |
|            |
|____________|
Pontuação:  0
Teclas do jogo:
← move esquerda | → move direita | ↓ move baixo
<Page Down> rotaciona esquerda | <Page Up> rotaciona direita
<s> sai da partida, <g> grava e sai da partida
```

3) Tela do jogo após jogar pressionar algumas teclas para movimentações de peças:

```
 ____________
|            |
|            |
|            |
|            |
|   %%       |
|  %%  #     |
|*  ###$$$$  |
|***$$$$     |
|____________|
```

```
Pontuação: 0
Teclas do jogo:
← move esquerda | → move direita | ↓ move baixo
<Page Down> rotaciona esquerda | <Page Up> rotaciona direita
<s> sai da partida, <g> grava e sai da partida
```

Observe a posição da peça S na tela acima. Quando essa peça for movida para baixo, completará uma linha preechida, o que resulta em pontuação (como mostrado na tela abaixo). Note que, na tela abaixo, a linha preenchida não aparece mais (porque ela foi removida assim que preenchida).

4) Tela do jogo após completar o preenchimento da penúltima linha da grade:

```
 __________
|     $$$$ |
|          |
|          |
|          |
|          |
|          |
|  %% #    |
|***$$$$   |
__________
Pontuação: 1
Teclas do jogo:
← move esquerda | → move direita | ↓ move baixo
<Page Down> rotaciona esquerda | <Page Up> rotaciona direita
<s> sai da partida, <g> grava e sai da partida
```

5) Tela do jogo após a pilha de peças chegar no topo da tela (ou seja, fim de partida). Observe que o jogo volta ao menu inicial depois de encerrada uma partida:

```
 __________
|     $$$$ |
|      *   |
|   *  *   |
|   * **   |
| @**%%%   |
|@@ %%%%%  |
|@+++%%%% *|
|&&++++%% *|
__________
Fim da partida!
Pontuação final: 3


*** Jogo Textris - um tetris em modo texto ***
Opções do jogo:
- <i> para iniciar uma nova partida
- <c> para carregar uma partida gravada e continuá-la
- <p> para ver as 10 melhores pontuações
- <s> para sair do jogo
Digite a opção desejada:
```

4 Dicas para a Implementação

- Exiba a tela do jogo sempre no canto superior esquerdo da janela do terminal. Para fazer isso, basta executar um comando para limpar o terminal antes de fazer a exibição da tela. Dessa forma, para atualizar a visualização da tela do jogo a cada vez que o jogador pressionar uma tecla de movimentação de peça, basta limpar o terminal e re-exibir a tela do jogo com peça já em sua nova posição.

  Em Python, dá para limpar o terminal realizando uma chamada ao sistema operacional por meio do módulo **os**, como mostrado abaixo:.

  ```
  import os
  os.system('cls||clear')
  ```

- Para fazer a captura das teclas pressionadas pelo jogador durante uma partida do jogo, você deve usar a função **readkey()** do módulo **readchar**. A função readkey() devolve o valor da tecla assim que ela é pressionada. Diferentemente da função input(), ela não espera o usuário digitar <ENTER> para devolver o valor. É por essa razão que ela é mais apropriada que a input() para uso no jogo. Veja um exemplo de uso abaixo:

  ```
  from readchar import readkey, key

  print("Pressione teclas!")
  while True:
    tecla = readkey()
    if "a" < tecla < "z" or "A" < tecla < "Z":
        print("Pressionou letra:", tecla)
    elif tecla == key.DOWN:
        print("Pressionou seta para baixo")
    elif tecla == key.ENTER:
        print("Pressionou ENTER")
        print("Tchau!")
        break
  ```

  Além de key.DOWN e key.ENTER, há outras teclas especiais que readkey() captura. Outros exemplos de teclas que podem ser úteis para o jogo são: `key.LEFT`, `key.RIGHT`,`key.PAGE_DOWN`,`key.PAGE_UP`.

  O módulo readchar não é um módulo padrão de Python, ele precisa ser instalado antes de ser usado:

  ```
  pip install readchar
  ```

- Para gravar um objeto em um arquivo (ou ler de um arquivo um objeto gravado anteriormente), vocês podem usar o módulo **pickle**[2]. Segundo sua documentação, o módulo pickle implementa protocolos binários para serializar e desserializar uma estrutura de objeto Python. “Pickling” é o processo pelo qual uma hierarquia de objetos Python é convertida em um fluxo de bytes, e “unpickling” é a operação inversa, em que um fluxo de bytes (de um arquivo binário ou objeto byte ou similar) é convertido de volta em uma hierarquia de objetos. O código abaixo exemplifica o uso do pickle:

```
import pickle

# Definindo a classe Pessoa
class Pessoa:
    def __init__(self, nome, email):
        self.nome = nome
        self.email = email

    def __repr__(self):
        return f"Pessoa(nome={self.nome}, email={self.email})"

# Criando uma instância da classe
pessoa1 = Pessoa("João", "joao@example.com")

# Salvando o objeto em um arquivo
with open("pessoa.pkl", "wb") as arquivo:
    pickle.dump(pessoa1, arquivo)      # salva pessoa1 em arquivo

# Carregando o objeto do arquivo
with open("pessoa.pkl", "rb") as arquivo:
    pessoa_carregada = pickle.load(arquivo)

print("Objeto carregado:")
print(pessoa_carregada)
```

5 Critérios da Correção do EP

A nota do EP3 será composta da seguinte forma:

- 30% Funcionalidades implementadas corretamente conforme o enunciado.
- 10% Qualidade da interface com o usuário. Usabilidade do jogo.
- 30% Qualidade do projeto orientado a objetos; classes com responsabilidade única.
- 10% Qualidade dos nomes das variáveis, atributos, métodos, classes. Os nomes revelam a intenção?
- 10% Testes automatizados – quantidade e qualidade dos testes, o quanto eles garantem que o sistema está funcionando corretamente.
- 10% Qualidade da documentação e explicação de como instalar e executar o programa em README.md.

Pontos extras poderão ser atribuídos a funcionalidades adicionais implementadas no jogo.

[2] https://docs.python.org/pt-br/3/library/pickle.html