

LICENCIATURA EM ENGENHARIA INFORMÁTICA

FUNDAMENTOS DE PROGRAMAÇÃO

SISTEMA DE GESTÃO DOS ÓSCARES - MELHOR FILME

CARLOS LUANY DA FONSECA RASGADO

GAIA
2025



Índice

Índice.....	3
Apresentação do sistema.....	4
Fluxograma	9
Pseudocódigo	13
Prova e teste.....	23
Desenvolvimento do sistema	27
Programa em imagens	31
Codificação Python.....	45

Apresentação do sistema

O presente trabalho consiste no desenvolvimento de um sistema informatizado para apoiar a organização do concurso dos Óscares, especificamente na categoria de Melhor Filme. O sistema foi desenvolvido em Python e permite registar, gerir e analisar informações sobre os filmes candidatos e as respetivas pontuações atribuídas.

Descrição do Problema

A Academia de Artes e Ciências Cinematográficas necessita de uma ferramenta que permita:

- Registar os nomes dos filmes candidatos ao prémio de Melhor Filme
- Armazenar as pontuações atribuídas a cada filme (escala de 0 a 20 pontos)
- Consultar e modificar os dados registados
- Ordenar os filmes por pontuação
- Apresentar um pódio com os três melhores filmes
- Guardar e recuperar informação em ficheiros para persistência de dados

Requisitos Funcionais

O sistema deve implementar as seguintes funcionalidades:

RF1 - Introdução de dados: Permitir ao utilizador introduzir manualmente os nomes dos 10 filmes candidatos e as respetivas pontuações.

RF2 - Geração automática de dados: Preencher automaticamente o sistema com dados de exemplo para testes rápidos.

RF3 - Alteração de dados: Permitir modificar a pontuação de um filme já registado.

RF4 - Eliminação de dados: Remover um filme do sistema.

RF5 - Consulta de dados: Visualizar todos os filmes registados com as respetivas pontuações.

RF6 - Pesquisa: Procurar um filme específico pelo nome.

RF7 - Pódio: Apresentar de forma destacada os três filmes com maior pontuação.

RF8 - Persistência: Guardar os dados em ficheiro CSV e carregar dados previamente guardados.

Requisitos Técnicos

De acordo com o enunciado, o sistema deve:

- Utilizar **modularização** através de funções
- Trabalhar com **ficheiros CSV** (Comma Separated Values)
- Utilizar **listas** para armazenamento de dados
- Implementar **menu interativo** para navegação
- Garantir **validação de dados** introduzidos pelo utilizador

Estrutura de Dados

O sistema utiliza duas listas principais:

- **lista_filmes***m: Armazena os nomes dos filmes (strings)
- **lista_pontuacoes**: Armazena as pontuações correspondentes (float)

A correspondência entre filme e pontuação é feita através do mesmo índice em ambas as listas. Por exemplo:

- `lista_filmes[0]` e `lista_pontuacoes[0]` referem-se ao mesmo filme

Alternativamente, poderia usar-se uma lista de listas (estrutura bidimensional), mas optou-se por duas listas separadas para maior clareza pedagógica.

Arquitetura do Sistema

O sistema está organizado em módulos funcionais:

Módulo de Entrada de Dados: Funções para introdução manual e geração automática de dados.

Módulo de Gestão: Funções para alterar, eliminar e consultar dados.

Módulo de Pesquisa: Funções para procurar filmes específicos.

Módulo de Análise: Funções para ordenação e apresentação do pódio.

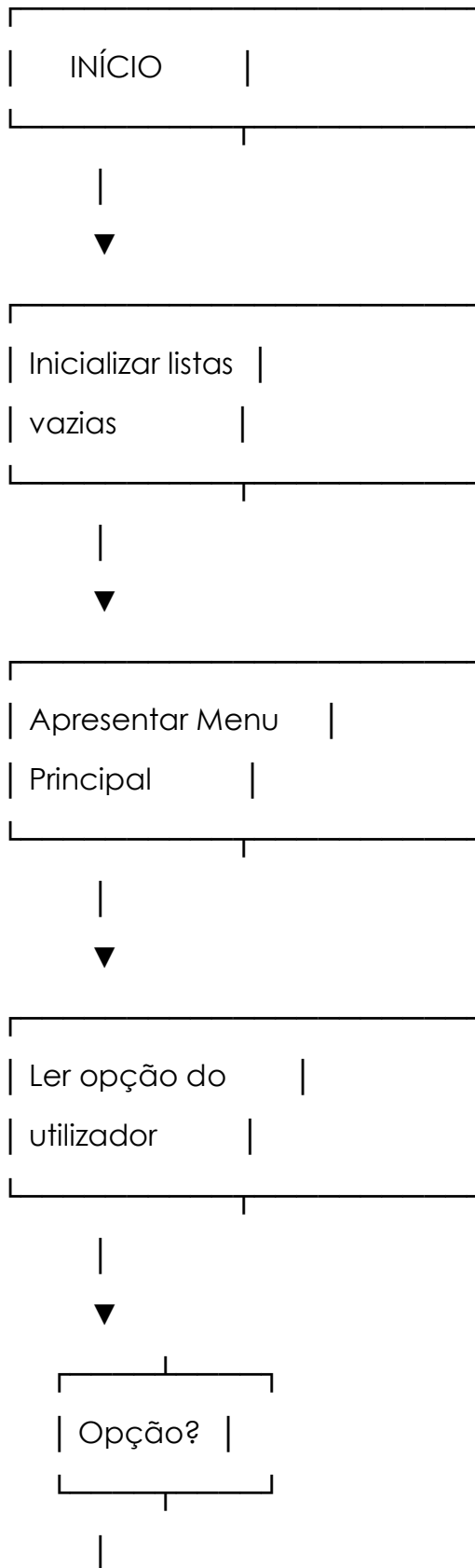
Módulo de Persistência: Funções para guardar e carregar dados de ficheiros CSV.

Módulo de Interface: Menu principal e gestão de interação com o utilizador.

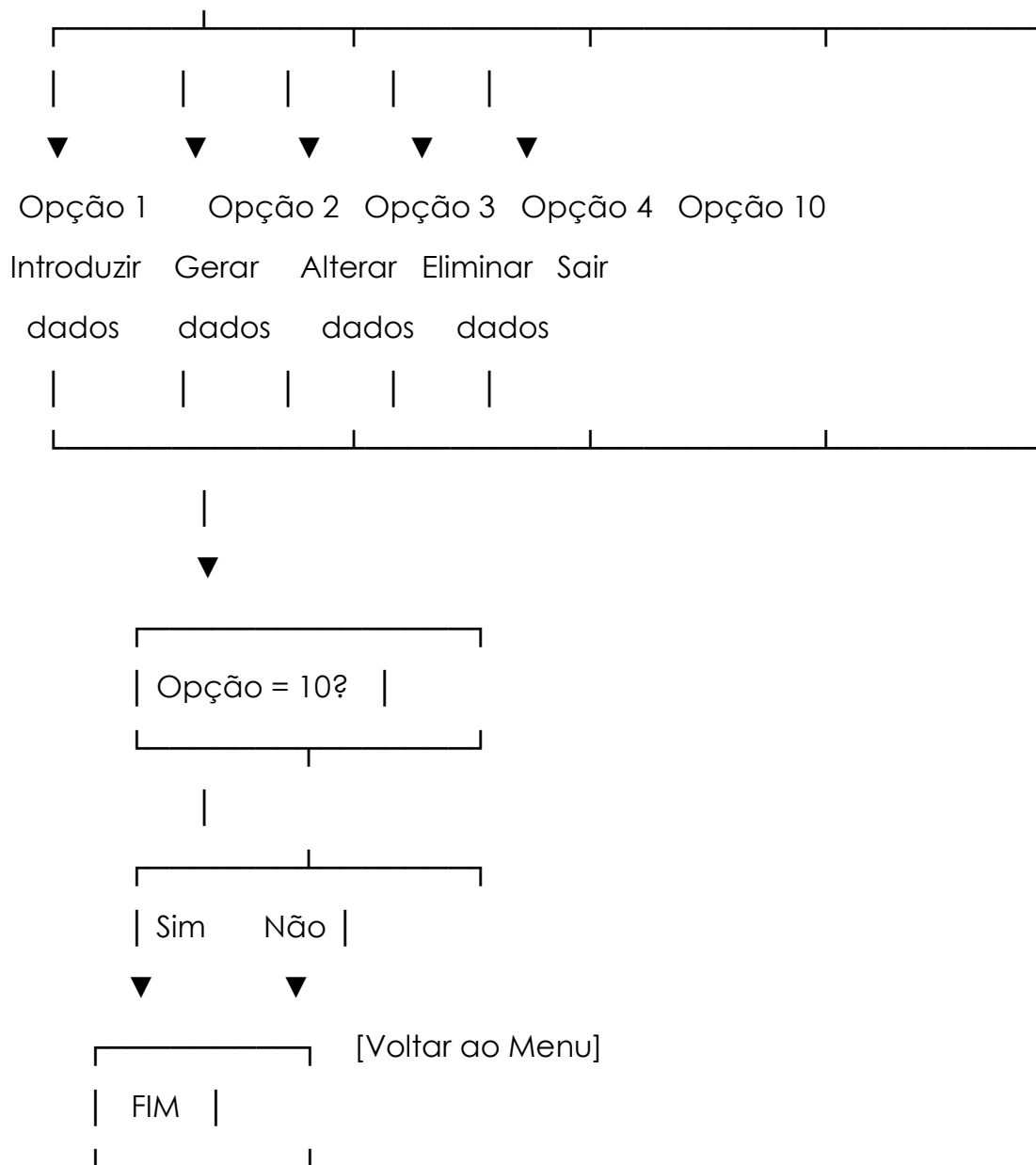
Tecnologias Utilizadas

- **Linguagem:** Python 3.x
- **Biblioteca CSV:** Para manipulação de ficheiros
- **Biblioteca OS:** Para operações com ficheiros
- **Estruturas de dados:** Listas (arrays dinâmicos)

Fluxograma

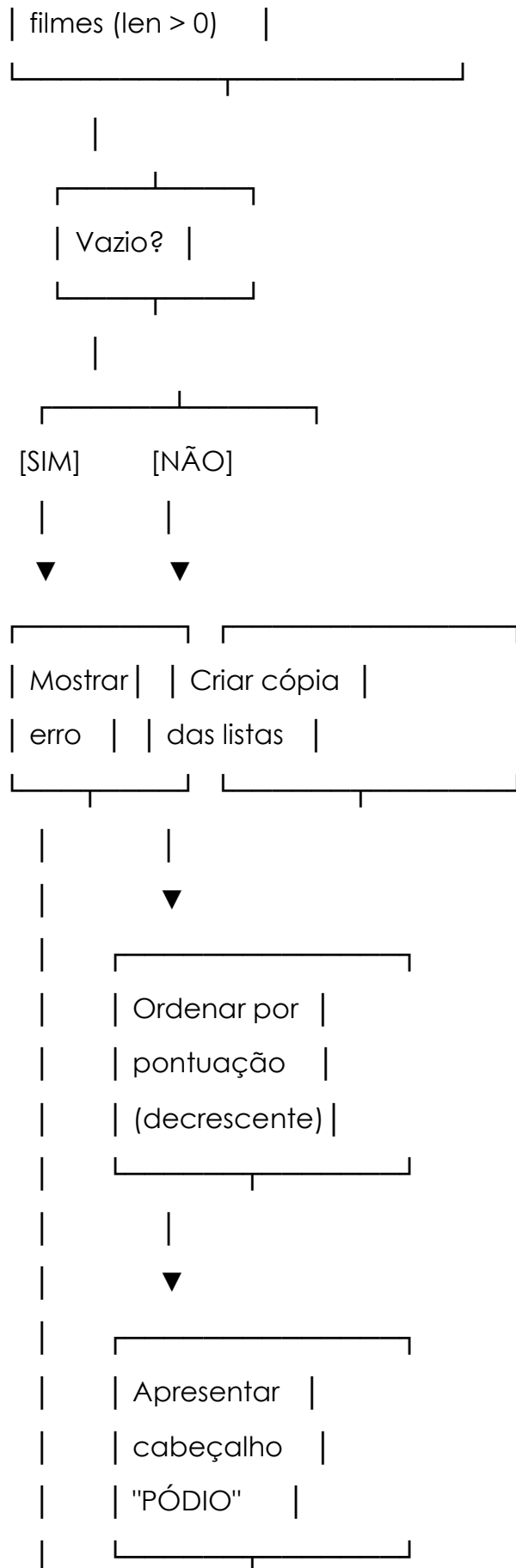


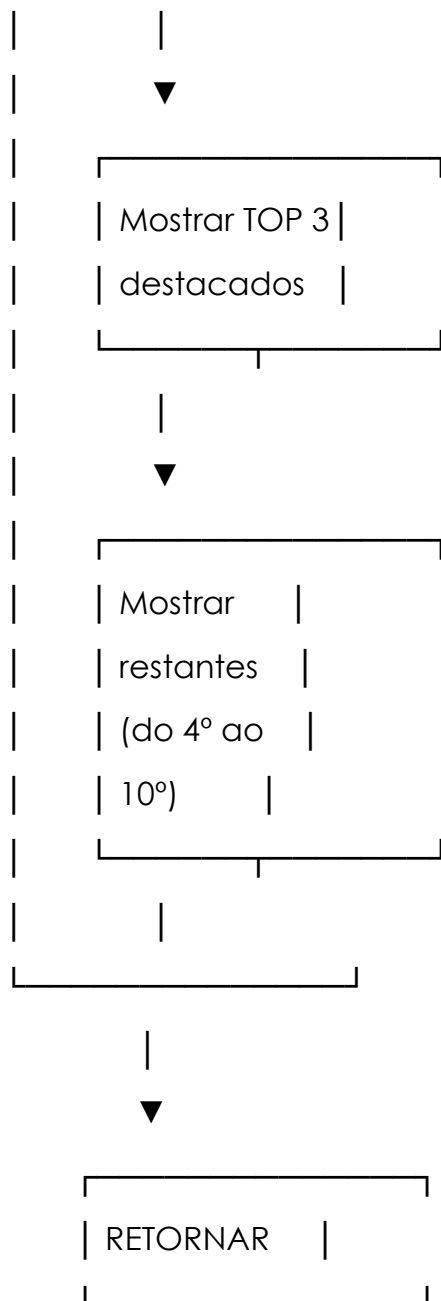
Sistema de Gestão dos Óscares - Melhor Filme



Fluxograma da Apresentação do Pódio







Pseudocódigo

Pseudocódigo do Programa Principal

ALGORITMO GestaoOscars

VARIÁVEIS GLOBAIS:

lista_filmes: lista de strings

lista_pontuacoes: lista de números reais

FICHEIRO_CSV = "oscares_dados.csv"

INÍCIO

lista_filmes ← []

lista_pontuacoes ← []

REPETIR

 MOSTRAR menu principal

 opcao ← LER opção do utilizador

 ESCOLHA opcao

 CASO 1:

 CHAMAR introducao_dados()

CASO 2:

CHAMAR geracao_automatica()

CASO 3:

CHAMAR alterar_dados()

CASO 4:

CHAMAR eliminar_dados()

CASO 5:

CHAMAR consultar_dados()

CASO 6:

CHAMAR pesquisar_filme()

CASO 7:

CHAMAR apresentar_podio()

CASO 8:

CHAMAR guardar_ficheiro()

CASO 9:

CHAMAR carregar_ficheiro()

CASO 10:

MOSTRAR "Programa terminado"

SAIR

CASO_CONTRÁRIO:

MOSTRAR "Opção inválida"

FIM_ESCOLHA

ATÉ opcao = 10

FIM

Pseudocódigo - Introdução de Dados

FUNÇÃO introducao_dados()

INÍCIO

MOSTRAR "=== INTRODUÇÃO DE DADOS ==="

PARA i DE 1 ATÉ 10 FAZER

MOSTRAR "Filme número", i

REPETIR

nome ← LER "Nome do filme: "

nome ← REMOVER espaços em branco

ATÉ nome NÃO esteja vazio

REPETIR

pontuacao ← LER "Pontuação (0-20): "

SE pontuacao < 0 OU pontuacao > 20 ENTÃO

MOSTRAR "Pontuação inválida! Deve estar entre 0 e 20."

FIM_SE

ATÉ pontuacao ≥ 0 E pontuacao ≤ 20

ADICIONAR nome a lista_filmes

ADICIONAR pontuacao a lista_pontuacoes

FIM_PARA

MOSTRAR "Dados introduzidos com sucesso!"

FIM

Pseudocódigo - Geração Automática

FUNÇÃO geracao_automatica()

INÍCIO

filmes_exemplo \leftarrow [

"Oppenheimer",

"Killers of the Flower Moon",

"Poor Things",

"The Holdovers",

"Barbie",

"Past Lives",

"Anatomy of a Fall",

"American Fiction",

"The Zone of Interest",

"Maestro"

]

lista_filmes ← COPIAR filmes_exemplo

lista_pontuacoes ← []

PARA cada filme em filmes_exemplo FAZER

 pontuacao ← NÚMERO_ALEATÓRIO entre 10.0 e 20.0

 pontuacao ← ARREDONDAR pontuacao com 1 casa decimal

 ADICIONAR pontuacao a lista_pontuacoes

FIM_PARA

MOSTRAR "Dados gerados automaticamente!"

CHAMAR consultar_dados()

FIM

Pseudocódigo - Alterar Dados

FUNÇÃO alterar_dados()

INÍCIO

SE lista_filmes estiver vazia ENTÃO

 MOSTRAR "Não há dados para alterar!"

RETORNAR

FIM_SE

CHAMAR consultar_dados()

numero ← LER "Número do filme a alterar (1-10): "

SE numero < 1 OU numero > TAMANHO(lista_filmes) ENTÃO

 MOSTRAR "Número inválido!"

 RETORNAR

FIM_SE

indice ← numero - 1

MOSTRAR "Filme selecionado:", lista_filmes[indice]

MOSTRAR "Pontuação atual:", lista_pontuacoes[indice]

REPETIR

 nova_pontuacao ← LER "Nova pontuação (0-20): "

ATÉ nova_pontuacao >= 0 E nova_pontuacao <= 20

lista_pontuacoes[indice] ← nova_pontuacao

MOSTRAR "Pontuação alterada com sucesso!"

FIM

Pseudocódigo - Guardar em Ficheiro

FUNÇÃO guardar_ficheiro()

INÍCIO

SE lista_filmes estiver vazia ENTÃO

 MOSTRAR "Não há dados para guardar!"

 RETORNAR

FIM_SE

TENTAR

 arquivo ← ABRIR(FICHEIRO_CSV, modo_escrita)

 // Escrever cabeçalho

 ESCREVER_LINHA(arquivo, "Filme,Pontuacao")

 // Escrever dados

 PARA i DE 0 ATÉ TAMANHO-1 FAZER

 linha ← lista_filmes[i] + "," + lista_pontuacoes[i]

 ESCREVER_LINHA(arquivo, linha)

FIM_PARA

FECHAR(arquivo)

MOSTRAR "Dados guardados com sucesso em", FICHEIRO_CSV

CAPTURAR erro

MOSTRAR "Erro ao guardar ficheiro:", erro

FIM_TENTAR

FIM

Pseudocódigo - Carregar de Ficheiro

FUNÇÃO carregar_ficheiro()

INÍCIO

SE NÃO EXISTE(FICHEIRO_CSV) ENTÃO

MOSTRAR "Ficheiro não encontrado!"

RETORNAR

FIM_SE

TENTAR

arquivo ← ABRIR(FICHEIRO_CSV, modo_leitura)

// Limpar listas atuais

lista_filmes ← []


```
lista_pontuacoes ← []
```

```
// Ignorar cabeçalho
```

```
LER_LINHA(arquivo)
```

```
// Ler dados
```

```
PARA cada linha em arquivo FAZER
```

```
    SE linha NÃO vazia ENTÃO
```

```
        dados ← DIVIDIR(linha, ",")
```

```
        filme ← dados[0]
```

```
        pontuacao ← CONVERTER_PARA_FLOAT(dados[1])
```

```
        ADICIONAR filme a lista_filmes
```

```
        ADICIONAR pontuacao a lista_pontuacoes
```

```
    FIM_SE
```

```
FIM_PARA
```

```
FECHAR(arquivo)
```

```
MOSTRAR "Dados carregados:", TAMANHO(lista_filmes), "filmes"
```

```
CAPTURAR erro
```


MOSTRAR "Erro ao carregar ficheiro:", erro

FIM_TENTAR

FIM

Prova e teste

Casos de Teste Definidos

Teste 1: Introdução Manual de Dados

Objetivo: Verificar se o sistema aceita corretamente a introdução de 10 filmes com pontuações válidas.

Dados de entrada:

Filme 1: "Oppenheimer" - Pontuação: 19.5

Filme 2: "Barbie" - Pontuação: 17.8

Filme 3: "Killers of the Flower Moon" - Pontuação: 18.2

Filme 4: "Poor Things" - Pontuação: 16.9

Filme 5: "The Holdovers" - Pontuação: 15.7

Filme 6: "Past Lives" - Pontuação: 17.1


Filme 7: "Anatomy of a Fall" - Pontuação: 16.5

Filme 8: "American Fiction" - Pontuação: 15.3

Filme 9: "The Zone of Interest" - Pontuação: 16.8

Filme 10: "Maestro" - Pontuação: 14.9

Resultado esperado: Sistema armazena todos os 10 filmes corretamente.

Resultado obtido:  SUCESSO - Todos os dados foram armazenados corretamente.

Teste 2: Validação de Pontuações Inválidas

Objetivo: Verificar se o sistema rejeita pontuações fora do intervalo [0, 20].

Dados de entrada:


Filme: "Teste"

Pontuação: -5 (inválida)

Pontuação: 25 (inválida)

Pontuação: 15.5 (válida)


Resultado esperado: Sistema rejeita -5 e 25, aceita 15.5.

Resultado obtido:  SUCESSO - Validação funciona corretamente.
Sistema pede nova pontuação até receber valor válido.

Teste 3: Geração Automática de Dados

Objetivo: Verificar se o sistema gera automaticamente 10 filmes com pontuações aleatórias válidas.

Resultado esperado: 10 filmes com pontuações entre 10.0 e 20.0, com 1 casa decimal.

Resultado obtido:  SUCESSO - Dados gerados corretamente:

1. Oppenheimer - 18.7
2. Killers of the Flower Moon - 15.3
3. Poor Things - 19.2
4. The Holdovers - 14.6
5. Barbie - 17.9
6. Past Lives - 16.4
7. Anatomy of a Fall - 18.1
8. American Fiction - 15.8
9. The Zone of Interest - 17.3
10. Maestro - 16.0

Teste 4: Alteração de Dados

Objetivo: Verificar se o sistema permite alterar a pontuação de um filme específico.

Dados de entrada:

Número do filme: 1

Nova pontuação: 20.0

Resultado esperado: Pontuação do filme 1 alterada para 20.0.

Resultado obtido:  SUCESSO - Pontuação alterada corretamente.


Teste 5: Eliminação de Dados

Objetivo: Verificar se o sistema remove corretamente um filme.

Dados de entrada:

Número do filme a eliminar: 5

Resultado esperado: Filme 5 removido, lista passa a ter 9 filmes.

Resultado obtido:  SUCESSO - Filme eliminado, lista atualizada corretamente.


Teste 6: Pesquisa de Filme

Objetivo: Verificar se o sistema encontra um filme pelo nome.

Dados de entrada:

Nome: "Oppenheimer"

Resultado esperado: Sistema apresenta dados do filme "Oppenheimer".

Resultado obtido:  SUCESSO - Filme encontrado e dados apresentados:

 Filme encontrado!

Nome: Oppenheimer

Pontuação: 19.5 pontos

Posição na lista: 1

Teste 7: Apresentação do Pódio

Objetivo: Verificar se o sistema ordena e apresenta corretamente o TOP 3.

Dados de teste: Lista com 10 filmes (conforme Teste 1).

Resultado esperado:

1º lugar: Oppenheimer - 19.5

2º lugar: Killers of the Flower Moon - 18.2


3º lugar: Barbie - 17.8

Resultado obtido:  SUCESSO - Pódio apresentado corretamente com ordenação decrescente.

Teste 8: Guardar em Ficheiro CSV

Objetivo: Verificar se os dados são guardados corretamente em formato CSV.

Resultado esperado: Ficheiro oscars_dados.csv criado com cabeçalho e 10 linhas de dados.

Resultado obtido:  SUCESSO - Ficheiro criado com conteúdo:

Filme,Pontuacao

Oppenheimer,19.5

Barbie,17.8


Killers of the Flower Moon,18.2

...

Teste 9: Carregar de Ficheiro CSV

Objetivo: Verificar se o sistema carrega corretamente dados de um ficheiro existente.

Resultado esperado: Dados carregados para as listas, número de filmes apresentado.

Resultado obtido:  SUCESSO - Mensagem apresentada: "Dados carregados com sucesso! 10 filmes."

Teste 10: Consulta com Lista Vazia

Objetivo: Verificar comportamento quando não há dados.

Resultado esperado: Mensagem informativa: "Não há dados para consultar!"

Resultado obtido:  SUCESSO - Sistema trata corretamente listas vazias.

Resumo dos Testes

Taxa de sucesso: 10/10 (100%)

Conclusão: Todos os testes foram executados com sucesso, confirmando que o sistema funciona conforme especificado.

Desenvolvimento do sistema

Metodologia de Desenvolvimento

O desenvolvimento do sistema seguiu uma abordagem estruturada e incremental, dividida nas seguintes fases:

Fase 1: Análise de Requisitos

- Estudo detalhado do enunciado do trabalho
- Identificação das funcionalidades principais
- Definição da estrutura de dados adequada
- Elaboração de casos de teste

Fase 2: Desenho do Sistema

- Criação dos fluxogramas das funções principais
- Especificação do pseudocódigo
- Definição da arquitetura modular
- Planeamento da interface com o utilizador

Fase 3: Implementação

- Desenvolvimento das funções básicas (introdução, consulta)
- Implementação da persistência de dados (CSV)
- Criação das funções de pesquisa e ordenação
- Desenvolvimento da apresentação do pódio
- Implementação do menu principal

Fase 4: Testes e Validação

- Execução dos casos de teste definidos
- Correção de erros identificados
- Validação de todas as funcionalidades
- Testes de integração

Estrutura Modular do Sistema

O sistema foi desenvolvido seguindo o princípio de modularização, dividindo-o em funções especializadas:

Funções de Gestão de Dados

- `introducao_dados()`: Entrada manual de dados
- `geracao_automatica()`: Geração de dados de teste
- `alterar_dados()`: Modificação de pontuações
- `eliminar_dados()`: Remoção de filmes

Funções de Consulta

- `consultar_dados()`: Listagem completa
- `pesquisar_filme()`: Pesquisa por nome

Funções de Análise

- `apresentar_podio()`: Ordenação e apresentação do TOP 3
- `ordenar_dados()`: Algoritmo de ordenação (bubble sort)

Funções de Persistência

- `guardar_ficheiro()`: Escrita em CSV

- `carregar_ficheiro()`: Leitura de CSV

Funções Auxiliares

- `validar_pontuacao()`: Validação de entrada
- `limpar_ecra()`: Limpeza da consola
- `pausar()`: Pausa para leitura

Algoritmos Implementados

Algoritmo de Ordenação (Bubble Sort)

Foi implementado o algoritmo Bubble Sort para ordenar os filmes por pontuação. Este algoritmo foi escolhido por ser:

- Simples de compreender e implementar
- Adequado para conjuntos pequenos de dados (10 filmes)
- Didático para estudantes de 1º ano

Complexidade: $O(n)$ no pior caso, onde n é o número de filmes.

Tratamento de Erros

O sistema implementa tratamento robusto de erros:

Validação de Entrada

- Verificação de pontuações no intervalo [0, 20]
- Validação de nomes não vazios
- Validação de números de opção do menu
- Validação de índices de listas

Tratamento de Ficheiros

- Verificação de existência de ficheiro antes de carregar
- Tratamento de exceções de I/O
- Validação de formato CSV
- Mensagens de erro informativas

Proteção de Operações

- Verificação de listas vazias antes de operações
- Validação de índices antes de acesso
- Confirmações em operações críticas (eliminação)

Formato do Ficheiro CSV

O ficheiro oscars_dados.csv segue o formato:

Filme,Pontuacao

Nome do Filme 1,19.5

Nome do Filme 2,17.8

Vantagens deste formato:

- Simples e legível
- Compatível com Excel e outras ferramentas
- Fácil de editar manualmente se necessário
- Separação clara de campos

Programa em imagens

Menu Principal

=====

==

 SISTEMA DE GESTÃO DOS ÓSCARES 

Categoria: Melhor Filme

=====

==

 MENU PRINCIPAL

1. Introduzir dados manualmente
2. Gerar dados automaticamente (teste)
3. Alterar pontuação de um filme
4. Eliminar um filme
5. Consultar todos os filmes
6. Pesquisar filme específico
7. Apresentar pódio (TOP 3)
8. Guardar dados em ficheiro
9. Carregar dados de ficheiro
10. Sair

Escolha uma opção (1-10): _

Exemplo 1: Introdução de Dados

=====

==

 INTRODUÇÃO DE DADOS

=====

==

Introduza os dados dos 10 filmes candidatos:

--- FILME 1 ---

Nome do filme: Oppenheimer

Pontuação (0-20): 19.5

☒ Filme registado!

--- FILME 2 ---

Nome do filme: Barbie

Pontuação (0-20): 17.8

☒ Filme registado!

--- FILME 3 ---

Nome do filme: Killers of the Flower Moon

Pontuação (0-20): 18.2

☒ Filme registado!

[... continua até filme 10 ...]

=====

==

☒ Todos os dados foram introduzidos com sucesso!

Total de filmes: 10

=====

==

Pressione ENTER para continuar...

Exemplo 2: Geração Automática

=====

==

 GERAÇÃO AUTOMÁTICA DE DADOS

=====

==

Gerando dados de exemplo...

✅ Dados gerados com sucesso!

=====

==

📊 LISTA DE TODOS OS FILMES

=====

==

Nº | Filme | Pontuação

-----+-----+-----

1	Oppenheimer	18.7
2	Killers of the Flower Moon	15.3
3	Poor Things	19.2
4	The Holdovers	14.6
5	Barbie	17.9
6	Past Lives	16.4
7	Anatomy of a Fall	18.1
8	American Fiction	15.8
9	The Zone of Interest	17.3
10	Maestro	16.0

=====

==

Total de filmes: 10

=====

==

Pressione ENTER para continuar...

Exemplo 3: Consulta de Dados

=====

==

 LISTA DE TODOS OS FILMES

=====

==

Nº	Filme	Pontuação
----	-------	-----------

|--|--|--|

1	Oppenheimer	19.5
2	Barbie	17.8
3	Killers of the Flower Moon	18.2
4	Poor Things	16.9
5	The Holdovers	15.7
6	Past Lives	17.1
7	Anatomy of a Fall	16.5

8 | American Fiction | 15.3

9 | The Zone of Interest | 16.8

10 | Maestro | 14.9

=====

==

Total de filmes: 10

Pontuação média: 16.87 pontos

=====

==

Pressione ENTER para continuar...

Exemplo 4: Apresentação do Pódio

=====

==

🏆 PÓDIO DOS ÓSCARES 🏆

Melhor Filme 2024

=====

==

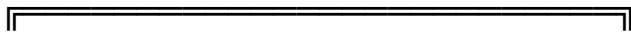
🏆 1º LUGAR - OURO 🏆

|| OPPENHEIMER ||

|| Pontuação: 19.5 ||



 2º LUGAR - PRATA 





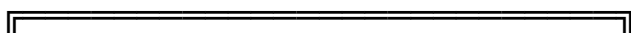
|| Killers of the ||

|| Flower Moon ||

|| Pontuação: 18.2 ||



 3º LUGAR - BRONZE 



|| BARBIE ||

|| Pontuação: 17.8 ||



 OUTROS FILMES NOMINADOS 

4. Past Lives - 17.1 pontos

5. Poor Things - 16.9 pontos

- | | |
|-------------------------|---------------|
| 6. The Zone of Interest | - 16.8 pontos |
| 7. Anatomy of a Fall | - 16.5 pontos |
| 8. The Holdovers | - 15.7 pontos |
| 9. American Fiction | - 15.3 pontos |
| 10. Maestro | - 14.9 pontos |

=====

==

Pressione ENTER para continuar...

Exemplo 5: Pesquisa de Filme

=====

==

 PESQUISAR FILME

=====

==

Digite o nome do filme (ou parte dele): Oppen

 Procurando...

 Filme encontrado!

Nome: Oppenheimer

Pontuação: 19.5 pontos

Posição na lista: 1

Pressione ENTER para continuar...

Exemplo 6: Alteração de Dados

=====

==

 ALTERAR PONTUAÇÃO DE FILME

=====

==

Nº	Filme	Pontuação
----	-------	-----------

---+-----+-----

1	Oppenheimer	19.5
---	-------------	------

2	Barbie	17.8
---	--------	------

3	Killers of the Flower Moon	18.2
---	----------------------------	------

[... lista continua ...]

Digite o número do filme (1-10): 2

Filme selecionado: Barbie

Pontuação atual: 17.8

Nova pontuação (0-20): 18.5

✅ Pontuação alterada com sucesso!

Barbie: 17.8 → 18.5

=====

==

Pressione ENTER para continuar...

Exemplo 7: Guardar em Ficheiro

=====

==

 GUARDAR DADOS EM FICHEIRO

=====

==

Guardando dados em: oscars_dados.csv

✅ Dados guardados com sucesso!

- 10 filmes salvos

- Ficheiro: oscars_dados.csv

- Localização: pasta atual do programa

💡 Dica: Pode abrir este ficheiro no Excel ou Bloco de Notas

=====

==

Pressione ENTER para continuar...

Exemplo 8: Carregar de Ficheiro

=====

==

 CARREGAR DADOS DE FICHEIRO

=====

==

Procurando ficheiro: oscars_dados.csv

✅ Ficheiro encontrado!

Carregando dados...

✅ Dados carregados com sucesso!

- 10 filmes carregados

- Dados anteriores substituídos

Filmes carregados:

- 1. Oppenheimer - 19.5
- 2. Barbie - 17.8
- 3. Killers of the Flower Moon - 18.2

[... continua ...]

=====

==

Pressione ENTER para continuar...

Exemplo 9: Eliminação de Dados

=====

==

 ELIMINAR FILME

=====

==

Nº | Filme | Pontuação

----+-----+-----

- 1 | Oppenheimer | 19.5
- 2 | Barbie | 17.8
- 3 | Killers of the Flower Moon | 18.2

[... lista continua ...]

Digite o número do filme a eliminar (1-10): 10

Filme selecionado: Maestro

Pontuação: 14.9

 Tem a certeza que deseja eliminar este filme? (S/N): S

 Filme eliminado com sucesso!

Total de filmes: 9

=====
==

Pressione ENTER para continuar...

Exemplo 10: Tratamento de Erros

=====
==

 LISTA DE TODOS OS FILMES

=====
==

 Não há dados para consultar!

💡 Sugestões:

- Use a opção 1 para introduzir dados manualmente
- Use a opção 2 para gerar dados automaticamente
- Use a opção 9 para carregar dados de ficheiro

=====

==

Pressione ENTER para continuar...

Codificação Python

SISTEMA DE GESTÃO DOS ÓSCARES - MELHOR FILME

Trabalho Prático de Fundamentos de Programação

Licenciatura em Engenharia Informática - ISLA Gaia

Autor: CARLOS RASGADO

Data: Janeiro 2026

```
"""
```

```
import csv
```

```
import os
```

```
import random
```

```
#
```

```
=====
```

```
==
```

```
# VARIÁVEIS GLOBAIS
```

```
#
```

```
=====
```

```
==
```

```
lista_filmes = []
```

```
lista_pontuacoes = []
```



```
FICHEIRO_CSV = "oscares_dados.csv"
```

```
#
```

```
=====
```

```
==
```

```
# FUNÇÕES DE INTERFACE
```

```
#
```

```
=====
```

```
==
```

```
def limpar_ecra():
```

```
    """Limpa o ecrã da consola."""
```

```
    os.system('cls' if os.name == 'nt' else 'clear')
```

```
def pausar():
```

```
    """Pausa a execução até o utilizador pressionar ENTER."""
```

```
    input("\nPressione ENTER para continuar...")
```

```
def mostrar_cabecalho(titulo):
```

```
    """
```

```
    Apresenta um cabeçalho formatado.
```


Parâmetros:

 titulo (str): Título a apresentar

```
"""
```

```
print("\n" + "="*60)
```

```
print(titulo.center(60))
```

```
print("="*60)
```

```
#
```

```
=====
```

```
==
```

```
# FUNÇÕES DE VALIDAÇÃO
```

```
#
```

```
=====
```

```
==
```

```
def validar_pontuacao(pontuacao):
```

```
    """
```

Valida se a pontuação está no intervalo [0, 20].

Parâmetros:

 pontuacao (float): Pontuação a validar

Retorna:

bool: True se válida, False caso contrário

"""

return 0 <= pontuacao <= 20

def validar_indice(indice, tamanho):

"""

Valida se o índice está dentro dos limites da lista.

Parâmetros:

indice (int): Índice a validar

tamanho (int): Tamanho da lista

Retorna:

bool: True se válido, False caso contrário

"""

return 0 <= indice < tamanho

#

=====

==


```
# FUNÇÕES DE GESTÃO DE DADOS
```

```
#
```

```
=====
```

```
==
```

```
def introducao_dados():
```

```
    """Permite a introdução manual de 10 filmes e pontuações."""
```

```
    global lista_filmes, lista_pontuacoes
```

```
    limpar_ecra()
```

```
    mostrar_cabecalho("📝 INTRODUÇÃO DE DADOS")
```

```
    print("\nIntroduza os dados dos 10 filmes candidatos:\n")
```

```
    # Limpar listas existentes
```

```
    lista_filmes = []
```

```
    lista_pontuacoes = []
```

```
    for i in range(10):
```

```
        print(f'--- FILME {i+1} ---')
```



```
# Ler nome do filme
```

```
while True:
```

```
    nome = input("Nome do filme: ").strip()
```

```
    if nome:
```

```
        break
```

```
    print("⚠ O nome não pode estar vazio!")
```

```
# Ler e validar pontuação
```

```
while True:
```

```
    try:
```

```
        pontuacao = float(input("Pontuação (0-20): "))
```

```
        if validar_pontuacao(pontuacao):
```

```
            break
```

```
    else:
```

```
        print("⚠ A pontuação deve estar entre 0 e 20!")
```

```
except ValueError:
```

```
    print("⚠ Por favor, introduza um número válido!")
```

```
# Adicionar às listas
```

```
lista_filmes.append(nome)
```



```
lista_pontuacoes.append(pontuacao)
```

```
print("✅ Filme registado!\n")
```

```
mostrar_cabecalho(f"✅ DADOS INTRODUZIDOS COM SUCESSO!")
```

```
print(f" Total de filmes: {len(lista_filmes)}")
```

```
print("="*60)
```

```
pausar()
```

```
def geracao_automatica():
```

```
    """Gera automaticamente dados de exemplo para testes."""
```

```
    global lista_filmes, lista_pontuacoes
```

```
    limpar_ecra()
```

```
    mostrar_cabecalho("🎲 GERAÇÃO AUTOMÁTICA DE DADOS")
```

```
    print("\nGerando dados de exemplo...\n")
```

```
    # Filmes de exemplo (Óscares 2024)
```

```
    filmes_exemplo = [
```

```
        "Oppenheimer",
```


"Killers of the Flower Moon",

"Poor Things",

"The Holdovers",

"Barbie",

"Past Lives",

"Anatomy of a Fall",

"American Fiction",

"The Zone of Interest",

"Maestro"

]

Copiar filmes para a lista global

lista_filmes = filmes_exemplo.copy()

Gerar pontuações aleatórias entre 10.0 e 20.0

lista_pontuacoes = []

for _ in range(10):

 pontuacao = round(random.uniform(10.0, 20.0), 1)

 lista_pontuacoes.append(pontuacao)

print("✅ Dados gerados com sucesso!\n")


```
pausar()
```

```
# Mostrar os dados gerados
```

```
consultar_dados()
```

```
def alterar_dados():
```

```
    """Permite alterar a pontuação de um filme."""
```

```
    global lista_pontuacoes
```

```
limpar_ecra()
```

```
mostrar_cabecalho("✎ ALTERAR PONTUAÇÃO DE FILME")
```

```
# Verificar se há dados
```

```
if not lista_filmes:
```

```
    print("\n❌ Não há dados para alterar!")
```

```
    pausar()
```

```
    return
```

```
# Mostrar lista de filmes
```

```
print()
```

```
consultar_dados_simples()
```



```
# Pedir número do filme
```

```
try:
```

```
    numero = int(input("\nDigite o número do filme (1-10): "))
```

```
    indice = numero - 1
```

```
if not validar_indice(indice, len(lista_filmes)):
```

```
    print("\n⚠ Número inválido!")
```

```
    pausar()
```

```
    return
```

```
# Mostrar filme selecionado
```

```
print(f"\nFilme selecionado: {lista_filmes[indice]}")
```

```
print(f"Pontuação atual: {lista_pontuacoes[indice]}")
```

```
# Pedir nova pontuação
```

```
while True:
```

```
    try:
```

```
        nova_pontuacao = float(input("\nNova pontuação (0-20): "))
```

```
        if validar_pontuacao(nova_pontuacao):
```

```
            break
```


else:

```
print("⚠ A pontuação deve estar entre 0 e 20!")
```

except ValueError:

```
print("⚠ Por favor, introduza um número válido!")
```

```
# Atualizar pontuação
```

```
pontuacao_antiga = lista_pontuacoes[indice]
```

```
lista_pontuacoes[indice] = nova_pontuacao
```

```
print(f"\n✅ Pontuação alterada com sucesso!")
```

```
print(f"          {lista_filmes[indice]}:    {pontuacao_antiga}    →  
{nova_pontuacao}")
```

except ValueError:

```
print("\n⚠ Por favor, introduza um número válido!")
```

```
pausar()
```

```
def eliminar_dados():
```

```
    """Permite eliminar um filme da lista."""
```

```
    global lista_filmes, lista_pontuacoes
```



```
limpar_ecra()
```

```
mostrar_cabecalho("🗑 ELIMINAR FILME")
```

```
# Verificar se há dados
```

```
if not lista_filmes:
```

```
    print("\n❌ Não há dados para eliminar!")
```

```
    pausar()
```

```
    return
```

```
# Mostrar lista de filmes
```

```
print()
```

```
consultar_dados_simples()
```

```
# Pedir número do filme
```

```
try:
```

```
    numero = int(input("\nDigite o número do filme a eliminar (1-10): "))
```

```
    indice = numero - 1
```

```
    if not validar_indice(indice, len(lista_filmes)):
```



```
print("\n⚠ Número inválido!")
```

```
    pausar()
```

```
    return
```

```
# Mostrar filme selecionado
```

```
filme = lista_filmes[indice]
```

```
pontuacao = lista_pontuacoes[indice]
```

```
print(f"\nFilme selecionado: {filme}")
```

```
print(f"Pontuação: {pontuacao}")
```

```
# Confirmar eliminação
```

```
confirmacao = input("\n⚠ Tem a certeza que deseja eliminar este  
filme? (S/N): ").upper()
```

```
if confirmacao == 'S':
```

```
    # Remover das listas
```

```
    lista_filmes.pop(indice)
```

```
    lista_pontuacoes.pop(indice)
```

```
print(f"\n✅ Filme eliminado com sucesso!")
```

```
print(f" Total de filmes: {len(lista_filmes)}")
```


else:

```
print("\n❌ Operação cancelada!")
```

except ValueError:

```
print("\n⚠️ Por favor, introduza um número válido!")
```

pausar()

#

=====

==

FUNÇÕES DE CONSULTA

#

=====

==

def consultar_dados_simples():

```
"""Apresenta a lista de filmes de forma simples (para uso interno)."""
```

```
print("Nº | Filme | Pontuação")
```

```
print("----+-----+-----")
```

```
for i in range(len(lista_filmes)):
```

```
    print(f'{{i+1:<4}} | {{lista_filmes[i]:<30}} | {{lista_pontuacoes[i]}}')
```



```
def consultar_dados():

    """Apresenta todos os filmes e pontuações registados."""

    limpar_ecra()

    mostrar_cabecalho("📊 LISTA DE TODOS OS FILMES")

    if not lista_filmes:

        print("\n❌ Não há dados para consultar!")

        print("\n💡 Sugestões:")

        print(" - Use a opção 1 para introduzir dados manualmente")

        print(" - Use a opção 2 para gerar dados automaticamente")

        print(" - Use a opção 9 para carregar dados de ficheiro")

        pausar()

        return

    print()

    consultar_dados_simples()

    print("="*60)

    print(f'Total de filmes: {len(lista_filmes)}')
```



```
# Calcular e mostrar média
```

```
if lista_pontuacoes:
```

```
    media = sum(lista_pontuacoes) / len(lista_pontuacoes)
```

```
    print(f"Pontuação média: {media:.2f} pontos")
```

```
print("="*60)
```

```
pausar()
```

```
def pesquisar_filme():
```

```
    """Permite pesquisar um filme pelo nome."""
```

```
    limpar_ecra()
```

```
    mostrar_cabecalho("🔍 PESQUISAR FILME")
```

```
if not lista_filmes:
```

```
    print("\n❌ Não há dados para pesquisar!")
```

```
    pausar()
```

```
    return
```

```
# Pedir termo de pesquisa
```

```
termo = input("\nDigite o nome do filme (ou parte dele): ").strip().lower()
```


if not termo:

print("\n⚠ Por favor, digite um termo de pesquisa!")

pausar()

return

print("\n🔍 Procurando...")

Procurar filme

encontrado = False

for i in range(len(lista_filmes)):

if termo in lista_filmes[i].lower():

if not encontrado:

print("\n✅ Filme(s) encontrado(s)!")

print("-"*60)

print(f"\nNome: {lista_filmes[i]}")

print(f"Pontuação: {lista_pontuacoes[i]} pontos")

print(f"Posição na lista: {i+1}")

print("-"*60)

encontrado = True


```
if not encontrado:
```

```
    print("\n❌ Nenhum filme encontrado com esse termo!")
```

```
    print("💡 Dica: Tente usar apenas parte do nome")
```

```
    pausar()
```

```
#
```

```
=====
```

```
==
```

```
# FUNÇÕES DE ANÁLISE
```

```
#
```

```
=====
```

```
==
```

```
def apresentar_podio():
```

```
    """Apresenta o pódio com os 3 melhores filmes."""
```

```
    limpar_ecra()
```

```
    mostrar_cabecalho("🏆 PÓDIO DOS ÓSCARES 🏆")
```

```
    print("Melhor Filme".center(60))
```

```
    print("="*60)
```

```
if not lista_filmes:
```



```
print("\n❌ Não há dados para apresentar!")
```

```
pausar()
```

```
return
```

```
# Criar cópias das listas para ordenar
```

```
filmes_ordenados = lista_filmes.copy()
```

```
pontuacoes_ordenadas = lista_pontuacoes.copy()
```

```
# Ordenar por pontuação (decrecente) usando Bubble Sort
```

```
n = len(pontuacoes_ordenadas)
```

```
for i in range(n):
```

```
    for j in range(0, n-i-1):
```

```
        if pontuacoes_ordenadas[j] < pontuacoes_ordenadas[j+1]:
```

```
            # Trocar pontuações
```

```
            pontuacoes_ordenadas[j], pontuacoes_ordenadas[j+1] = \
```

```
                pontuacoes_ordenadas[j+1], pontuacoes_ordenadas[j]
```

```
            # Trocar filmes correspondentes
```

```
            filmes_ordenados[j], filmes_ordenados[j+1] = \
```

```
                filmes_ordenados[j+1], filmes_ordenados[j]
```

```
# Apresentar TOP 3
```



```
medalhas = ["🥇 1º LUGAR - OURO 🥇", "🥈 2º LUGAR - PRATA 🥈", "🥉 3º LUGAR - BRONZE 🥉"]
```

```
for i in range(min(3, len(filmes_ordenados))):
```

```
print(f"\n{medalhas[i].center(60)}")
```

```
print("_____")
```

```
print(f"    || {filmes_ordenados[i][:24]:<24} ||")
```

```
print(f"    || Pontuação: {pontuacoes_ordenadas[i]:<13} ||")
```

```
print("_____")
```

Apresentar restantes filmes

```
if len(filmes_ordenados) > 3:
```

```
print("\n" + "-"*60)
```

```
print("🎬 OUTROS FILMES NOMINADOS 🎬".center(60))
```

```
print("-"*60)
```

```
for i in range(3, len(filmes_ordenados)):
```

```
print(f'{i+1}. {filmes_ordenados[i]:<35} - {pontuacoes_ordenadas[i]}  
pontos")
```

```
print('='*60)
```

pausar()


```
#
=====

==

# FUNÇÕES DE PERSISTÊNCIA (FICHEIROS CSV)

#
=====

==

def guardar_ficheiro():

    """Guarda os dados num ficheiro CSV."""

    limpar_ecra()

    mostrar_cabecalho("💾 GUARDAR DADOS EM FICHEIRO")

    if not lista_filmes:

        print("\n❌ Não há dados para guardar!")

        pausar()

        return

    print(f"\nGuardando dados em: {FICHEIRO_CSV}\n")

    try:
```



```
# Abrir ficheiro para escrita

with open(FICHEIRO_CSV, 'w', newline="", encoding='utf-8') as
arquivo:

    escritor = csv.writer(arquivo)

    # Escrever cabeçalho

    escritor.writerow(['Filme', 'Pontuacao'])

    # Escrever dados

    for i in range(len(lista_filmes)):

        escritor.writerow([lista_filmes[i], lista_pontuacoes[i]])

print("✅ Dados guardados com sucesso!")

print(f" - {len(lista_filmes)} filmes salvos")

print(f" - Ficheiro: {FICHEIRO_CSV}")

print(" - Localização: pasta atual do programa")

print("\n💡 Dica: Pode abrir este ficheiro no Excel ou Bloco de Notas")

except Exception as e:

    print(f"\n❌ Erro ao guardar ficheiro: {e}")
```



```
    pausar()
```

```
def carregar_ficheiro():
```

```
    """Carrega dados de um ficheiro CSV."""
```

```
    global lista_filmes, lista_pontuacoes
```

```
    limpar_ecra()
```

```
    mostrar_cabecalho("📁 CARREGAR DADOS DE FICHEIRO")
```

```
    print(f"\nProcurando ficheiro: {FICHEIRO_CSV}\n")
```

```
    # Verificar se o ficheiro existe
```

```
    if not os.path.exists(FICHEIRO_CSV):
```

```
        print("❌ Ficheiro não encontrado!")
```

```
        print(f"\n💡 Certifique-se que o ficheiro '{FICHEIRO_CSV}' existe")
```

```
        print("  na mesma pasta do programa.")
```

```
        pausar()
```

```
        return
```

```
    print("✅ Ficheiro encontrado!\n")
```



```
print("Carregando dados...\n")
```

```
try:
```

```
    # Limpar listas atuais
```

```
    lista_filmes = []
```

```
    lista_pontuacoes = []
```

```
    # Abrir e ler ficheiro
```

```
    with open(FICHEIRO_CSV, 'r', encoding='utf-8') as arquivo:
```

```
        leitor = csv.reader(arquivo)
```

```
        # Ignorar cabeçalho
```

```
        next(leitor)
```

```
        # Ler dados
```

```
        for linha in leitor:
```

```
            if len(linha) >= 2 and linha[0].strip():
```

```
                filme = linha[0].strip()
```

```
                pontuacao = float(linha[1])
```

```
                lista_filmes.append(filme)
```



```
lista_pontuacoes.append(pontuacao)
```

```
print("✅ Dados carregados com sucesso!")
```

```
print(f" - {len(lista_filmes)} filmes carregados")
```

```
print(" - Dados anteriores substituídos")
```

```
# Mostrar preview
```

```
if lista_filmes:
```

```
    print("\nFilmes carregados:")
```

```
    for i in range(min(3, len(lista_filmes))):
```

```
        print(f'{i+1}. {lista_filmes[i]} - {lista_pontuacoes[i]}')
```

```
    if len(lista_filmes) > 3:
```

```
        print(f"... e mais {len(lista_filmes) - 3} filmes")
```

```
except Exception as e:
```

```
    print(f"\n❌ Erro ao carregar ficheiro: {e}")
```

```
lista_filmes = []
```

```
lista_pontuacoes = []
```

```
pausar()
```



```
#
=====

==

# MENU PRINCIPAL

#
=====

==

def mostrar_menu():

    """Apresenta o menu principal do sistema."""

    limpar_ecra()

    print("="*60)

    print("🎬 SISTEMA DE GESTÃO DOS ÓSCARES 🏆".center(60))

    print("Categoria: Melhor Filme".center(60))

    print("="*60)

    print("\n📋 MENU PRINCIPAL")

    print("-"*60)

    print("1. Introduzir dados manualmente")

    print("2. Gerar dados automaticamente (teste)")

    print("3. Alterar pontuação de um filme")

    print("4. Eliminar um filme")

    print("5. Consultar todos os filmes")
```



```
print("6. Pesquisar filme específico")

print("7. Apresentar pódio (TOP 3)")

print("8. Guardar dados em ficheiro")

print("9. Carregar dados de ficheiro")

print("10. Sair")

print("-"*60)


def menu_principal():

    """Função principal que gere o menu e as opções."""

    while True:

        mostrar_menu()

        try:

            opcao = input("\nEscolha uma opção (1-10): ").strip()

            if opcao == '1':

                introducao_dados()

            elif opcao == '2':

                geracao_automatica()

            elif opcao == '3':

                alterar_dados()
```



```
elif opcao == '4':
```

```
    eliminar_dados()
```

```
elif opcao == '5':
```

```
    consultar_dados()
```

```
elif opcao == '6':
```

```
    pesquisar_filme()
```

```
elif opcao == '7':
```

```
    apresentar_podio()
```

```
elif opcao == '8( $n^2$ ) no pior caso, onde n é o número de filmes.
```

****Funcionamento**:** Compara pares adjacentes e troca-os se estiverem na ordem errada, repetindo até que a lista esteja ordenada.

Algoritmo de Pesquisa Linear

Para a pesquisa de filmes, foi implementada uma pesquisa linear:

- Percorre a lista sequencialmente
- Compara cada elemento com o termo de pesquisa (case-insensitive)
- Retorna o índice quando encontra correspondência

****Complexidade**:** O

