Aula prática 5

Esta aula tem como objetivo introduzir as estruturas lineares; as listas ligadas e filas.

Para cada exercício, consulte a respetiva pasta incluída em **P05.zip**, disponível no Moodle.

Os exercícios 1, 2, e 4 podem fazer no moodle, com o CodeRunner.

1. Crie as funções descritas nas alíneas seguintes e teste a sua implementação, recorrendo ao ficheiro `arquivo.txt`. As funções permitem a manipulação de listas ligadas cujos elementos contêm dados do tipo `string`.
   Nota: para comparar ou utilizar o valor de um iterador, em c++ é necessário fazer cast do mesmo para o tipo de variável que pretende (ex: (int) (*it) ou (string) (*it)).

   a) Crie uma nova lista ligada chamada `lst_jogos`.

   b) Insira os jogos que constam no ficheiro arquivo.txt no fim da lista `lst_jogos`.

   c) Imprima no terminal o número de jogos carregados para a lista:

Resultado (parcial) da execução do programa:
`Foram carregados 43 jogos.`

   d) Imprima no terminal a posição na lista do jogo e o seu nome, seguindo a formatação demonstrada no exemplo abaixo.

Resultado (parcial) da execução do programa:

`Pos 0 -> Grand Theft Auto: Liberty City Stories (PS2)`
`Pos 1 -> FIFA 12 (PS3)`
`...`
`Pos 41 -> Fist of the North Star: Ken's Rage (PS3)`
`Pos 42 -> Shaun White Skateboarding (Wii)`

   e) Ordene alfabeticamente a lista `lst_jogos` e imprima a lista ordenada no terminal.

Resultado (parcial) da execução do programa:

`Lista ordenada:`
`Pos 0 -> Arcade Hits Pack - Gunblade NY + LA Machineguns (Wii)`
`Pos 1 -> Asterix at the Olympic Games (PC)`
`...`
`Pos 41 -> Tomb Raider Legend Platinum (PS2)`
`Pos 42 -> Wacky Races: Crash & Dash Wii (Wii)`

   f) Remova o jogo `Mario Kart (Wii)` da lista e volte a imprimir o seu conteúdo. Sugestão: utilize os métodos find() e erase(). (O find devolve a

posição do elemento ou -1 se não encontrar; o erase devolve um apontador para o elemento seguinte ao apagado).

g) Remova todos os jogos da PS2 da lista e volte a imprimir o seu conteúdo.

Resultado (parcial) da execução do programa:

```
Lista sem jogos da PS2:
Pos 0 -> Arcade Hits Pack - Gunblade NY + LA Machineguns (Wii)
Pos 1 -> Asterix at the Olympic Games (PC)
...
Pos 30 -> Tiger Woods PGA Tour 11 (Wii)
Pos 31 -> Wacky Races: Crash & Dash Wii (Wii)
```

2. Considere a classe `MediaMovel` que consta no ficheiro `MediaMovel.h`. Tal como o nome indica, esta classe destina-se ao cálculo de médias móveis, e apresenta a seguinte formulação:

```
class MediaMovel
{
  public:
    MediaMovel(double valor);
    void update(double valor);
    double getMedia() const;
  private:
    double n;
    list<double> valores;
};
```

a) Implemente a função `update(valor)`, definida no ficheiro `MediaMovel.cpp`, que considera `valor` como o número mais recente a ser considerado para o cálculo da média móvel. Deverá inserir o valor na lista `valores` e atualizar a variável n, que armazena a dimensão da lista.

b) Implemente a função `getMedia()` que calcula e retorna o valor da média móvel dos n números da lista `valores`.

3. Considere a classe RankingNomes que consta no ficheiro RankingNomes.h. Esta classe destina-se à gestão de rankings de pessoas, com um número máximo de posições especificado no atributo max_pos e apresenta a seguinte formulação:

```
class RankingNomes
{
  public:
        RankingNomes(string ficheiro, int num_pos);
        void Insere (string nome, int posicao);
        void Remove (int posicao);
        void Promove(int pos, int numero_pos);
        void Despromove(int pos, int numero_pos);
        void Imprime();
  private:
    int max_pos;
    list<string> nomes;
};
```

a) Implemente o construtor que cria o ranking a partir de um ficheiro de nomes com número de máximo de posições num_pos.

b) Implemente a função da classe void RankingNomes void RankingNomes::Imprime() que imprime o ranking seguindo a formatação demonstrada no exemplo abaixo.

Resultado da execução do construtor e função:

```
-- Teste Construtor --

Ranking de máximo 10 posições:
Pos 1 -> João Silva
Pos 2 -> Maria Santos
Pos 3 -> José Pereira
Pos 4 -> Ana Oliveira
Pos 5 -> Pedro Costa
Pos 6 -> Sofia Ferreira
Pos 7 -> Manuel Rodrigues
Pos 8 -> Inês Almeida
Pos 9 -> Miguel Martins
Pos 10 -> Carolina Soares
```

c) Implemente a função void RankingNomes::Insere(string nome, int pos) que permite inserir um novo nome na posição pos do ranking. Retorna -1 se erro ou 0 se tudo correu bem.

Resultado da execução da função Insere:

```
-- Teste Insere --

Ranking de máximo 10 posições:
Pos 1 -> João Silva
Pos 2 -> Maria Santos
Pos 3 -> José Pereira
Pos 4 -> Ana Oliveira
Pos 5 -> Pedro Costa
Pos 6 -> Sofia Ferreira
Pos 7 -> Manuel Rodrigues
Pos 8 -> Inês Almeida
Pos 9 -> Zeferino Duarte
Pos 10 -> Miguel Martins
 -- Posição inválida --

Ranking de máximo 10 posições:
Pos 1 -> João Silva
Pos 2 -> Maria Santos
Pos 3 -> José Pereira
Pos 4 -> Ana Oliveira
Pos 5 -> Pedro Costa
Pos 6 -> Sofia Ferreira
Pos 7 -> Manuel Rodrigues
Pos 8 -> Inês Almeida
Pos 9 -> Zeferino Duarte
Pos 10 -> Miguel Martins
```

d) Implemente a função `void RankingNomes::Remove(int pos)` que permite remover o nome da posição `pos` do ranking. Retorna -1 se erro ou 0 se tudo correu bem.

```
-- Teste Remove --

 -- Posição inválida --

Ranking de máximo 10 posições:
Pos 1 -> João Silva
Pos 2 -> Maria Santos
Pos 3 -> José Pereira
Pos 4 -> Ana Oliveira
Pos 5 -> Pedro Costa
Pos 6 -> Sofia Ferreira
Pos 7 -> Manuel Rodrigues
Pos 8 -> Inês Almeida
Pos 9 -> Zeferino Duarte
Pos 10 -> Miguel Martins

Ranking de máximo 10 posições:
Pos 1 -> Maria Santos
Pos 2 -> José Pereira
Pos 3 -> Ana Oliveira
Pos 4 -> Pedro Costa
Pos 5 -> Sofia Ferreira
Pos 6 -> Manuel Rodrigues
Pos 7 -> Inês Almeida
Pos 8 -> Zeferino Duarte
Pos 9 -> Miguel Martins
```

e) Implemente a função `int RankingNomes::Promove(int pos, int numero_pos)` que permite promover o nome da posição `pos` do ranking no número de posições designadas por `numero_pos`. Retorna -1 se erro ou 0 se tudo correu bem.

```
-- Teste Promove --

 -- Posição inválida --

Ranking de máximo 10 posições:
Pos 1 -> Maria Santos
Pos 2 -> José Pereira
Pos 3 -> Ana Oliveira
Pos 4 -> Pedro Costa
Pos 5 -> Sofia Ferreira
Pos 6 -> Manuel Rodrigues
Pos 7 -> Inês Almeida
Pos 8 -> Zeferino Duarte
Pos 9 -> Miguel Martins

Ranking de máximo 10 posições:
Pos 1 -> Maria Santos
Pos 2 -> José Pereira
Pos 3 -> Ana Oliveira
Pos 4 -> Pedro Costa
Pos 5 -> Sofia Ferreira
Pos 6 -> Manuel Rodrigues
Pos 7 -> Miguel Martins
Pos 8 -> Inês Almeida
Pos 9 -> Zeferino Duarte

Ranking de máximo 10 posições:
Pos 1 -> Sofia Ferreira
Pos 2 -> Maria Santos
Pos 3 -> José Pereira
Pos 4 -> Ana Oliveira
Pos 5 -> Pedro Costa
Pos 6 -> Manuel Rodrigues
Pos 7 -> Miguel Martins
Pos 8 -> Inês Almeida
Pos 9 -> Zeferino Duarte
```

f) Implemente a função `int RankingNomes::Despromove(int pos, int numero_pos)` que permite despromover o nome da posição `pos` do ranking no número de posições designadas por `numero_pos`. Retorna -1 se erro ou 0 se tudo correu bem.

```
-- Teste Despromove --

 -- Posição inválida --

Ranking de máximo 10 posições:
Pos 1 -> Sofia Ferreira
Pos 2 -> Maria Santos
Pos 3 -> José Pereira
```

```
Pos 4 -> Ana Oliveira
Pos 5 -> Pedro Costa
Pos 6 -> Manuel Rodrigues
Pos 7 -> Miguel Martins
Pos 8 -> Inês Almeida
Pos 9 -> Zeferino Duarte

Ranking de máximo 10 posições:
Pos 1 -> Sofia Ferreira
Pos 2 -> Maria Santos
Pos 3 -> José Pereira
Pos 4 -> Ana Oliveira
Pos 5 -> Pedro Costa
Pos 6 -> Manuel Rodrigues
Pos 7 -> Miguel Martins
Pos 8 -> Inês Almeida
Pos 9 -> Zeferino Duarte

Ranking de máximo 10 posições:
Pos 1 -> Sofia Ferreira
Pos 2 -> Maria Santos
Pos 3 -> José Pereira
Pos 4 -> Ana Oliveira
Pos 5 -> Manuel Rodrigues
Pos 6 -> Miguel Martins
Pos 7 -> Inês Almeida
Pos 8 -> Pedro Costa
Pos 9 -> Zeferino Duarte
```

4.

a) Implemente, no ficheiro **`queue.cpp`**, a função **`posLargestElement`** que devolve a **posição do maior elemento** presente numa fila.

```
int posLargestElement(queue<char> queue_var);
```

A função deve retornar **0** se a fila se encontrar vazia. No caso de elementos iguais, a função retorna a posição do elemento que sairia primeiro da fila.

Nota: A **posição** de um elemento corresponde à sua **ordem**, ou seja, a posição do elemento na cabeça da fila é 1.

O ficheiro **`test.cpp`** testa a implementação da função pedida. Depois de implementada, o programa deverá apresentar:

```
Teste à função <posLargestElement>

Queue: [  ]
Empty queue!

Queue: [   A   B   C   D   E   F  ]
The largest element in the queue is at position 6.

Queue: [   ~   .       +   y   2   f   B   <   &  ]
The largest element in the queue is at position 1.
```

b) Implemente, no ficheiro **`queue.cpp`**, a função **`insertInPosition`** que insere um **elemento** (`elem`) numa determinada **posição** (`pos`) de uma **fila**.

```
int insertInPosition(queue<char>* queue_ptr, char elem,
                     unsigned int pos);
```

`queue_ptr` é um **apontador** para a fila. A função deve retornar **–1** em caso de erro.

Nota: A **posição** de um elemento corresponde à sua **ordem**, ou seja, a posição do elemento na cabeça da fila é 1.

O ficheiro **`test.cpp`** testa a implementação da função pedida. Depois de implementada, o programa deverá apresentar:

```
Testing function <posLargestElement>

Queue: NULL
Insert element X at position 1:
Insert returned an error!

Queue: [   A   B   C   D   E   F  ]
Insert element H at position 8:
Insert returned an error!
Queue: [   A   B   C   D   E   F  ]

Queue: [   A   B   C   D   E   F  ]
Insert element G at position 7:
Insert successful!
Queue: [   A   B   C   D   E   F   G  ]

Queue: [   A   B   C   D   E   F   G  ]
Insert element Y at position 1:
Insert successful!
Queue: [   Y   A   B   C   D   E   F   G  ]

Queue: [   Y   A   B   C   D   E   F   G  ]
Insert element Z at position 4:
Insert successful!
Queue: [   Y   A   B   Z   C   D   E   F   G  ]
```

5. Implemente a função **`reverseQueue`** que inverte os elementos de uma fila, retornando a fila invertida.

```
queue<string> reverseQueue(queue<string> queue_orig);
```

Nota: É sugerido utilizar pilhas na implementação desta função.

Após a implementação da função, o programa deverá apresentar:

```
--- Testing function <reverseQueue> ---

Original queue: [   D8bMX%69   =:w=I'GJ   w=QT>(RT   i^wyht-m   {nKzc0@%   5xSUeH%C
PBEygxs  ]
Reversed queue: [   PBEygxs   5xSUeH%C   {nKzc0@%   i^wyht-m   w=QT>(RT   =:w=I'GJ
D8bMX%69  ]
```