Implementação de Lista Dinâmica Encadeada, Pilha e Fila

Mariana Bastos dos Santos

Junho 2019

1 Introdução

Na computação, existem diversos mecanismos de organização de dados para atender aos diferentes requisitos de processamento. Esses mecanismos são as estruturas de dados, as quais definem como distribuir e relacionar os dados disponíveis, tornando mais eficientes os algoritmos que manipulam esses dados. Sendo assim, este trabalho tem como objetivo implementar três diferentes estruturas de dados: lista encadead, pilha e fila. Utilizando liguagem c++ para o desenvolvimento, são criadas as classes LDE, pilha e fila, sendo que a primeira será herdada pelas demais, possuindo os métodos de inserção no fim e no início da lista e remoção no fim e no início da lista. A classe pilha usará os métodos de inserção e remoção no início da lista, para empilhar e desempilhar, respectivamente, os elementos. A classe fila usará o métodos de inserção e remoção no fim da lista, para enfileirar e desenfileirar, respectivamente, os elementos.

2 Trabalhos Relacionados

[1] propõe um algoritmo de escalonamento de tarefas baseado em algoritmos de clustering fuzzy. Para isso, foi combinado o algoritmo de clusterização fuzzy c-means baseado em kernel e o algoritmo FIFO (First-In-First-Out) aprimorado para projetar um novo algoritmo de planejamento. No algortimo de clusterização fuzzy c-means baseado em kernel, usamos função de

base radial (RBF) como função kernel. Os resultados do experimento mostram que o algoritmo proposto encurta o tempo de execução das tarefas e aumenta a utilização dos recursos.

3 Conceitos Fundamentais

3.1 Lista dinâmica encadeada

Listas dinâmicas encadeadas (LDE) são estruturas de dados que utilizam alocação dinâmica de memória para armazenar elementos, ou seja, podem crescer e diminuir dinamicamente. Cada elemento da lista é chamado de nó e possui um ponteiro para o próximo da lista (encadeamento). A lista é representada por um ponteiro para o primeiro elemento, e desse pode-se alcançar o segundo, seguindo o encadeamento e assim sucessivamente. Para cada novo elemento inserido na estrutura, um espaço na memória é alocado dinamicamente, mas a alocação do espaço não é contígua. Cada elemento da lista deve guardar a informação e o endereço do próximo nó, e também deve haver um ponteiro especial para o primeiro elemento da lista e para o último, sendo que o ponteiro para o último elemento deve especificar algum tipo de final, como por exemplo nulo.

3.2 Fila

Também conhecida como FIFO (Fisrt-In-First-Out), é uma estrutura de dados onde o primeiro elemento é processado primeiro e o mais novo elemento é processado depois. Por exemplo, numa bilheteria as pessoas chegam, pegam os ingressos e vão embora. As pessoas entram numa fila para serem atendidas de maneira organizada. A pessoa que entrar na fila primeiro receberá o ingresso primeiro e sairá da fila. A pessoa que entrar na fila depois, receberá o ingresso depois que a pessoa na frente dele. Sendo assim, a pessoa que entrar na fila por último será o último a receber os ingressos.

3.3 Pilha

Também conhecida como LIFO (Last-In-Last-Out), é uma estrutura de dados onde o primeiro elemento é processado por último e o último elemento é processado primeiro. Por exemplo, em um tubo onde se armazenam bolas,

diferentes tipos de bolas são colocadas. A bola que entrar no tubo por último, será tirada primeiro. A bola que estiver entre a primeira colocada e a última colocada, só será retirada quando a última for retirada. Sendo assim a bola que entrar primeiro no tubo será a última a sair.

4 Implementação

Neste trabalho foram implementadas as estruturas de dados, LDE, pilha e fila. Para a implementação de ambas foi utilizada a linguagem C++.

Na Figura 1 é possível observar o diagrama da implementação das classes *no*, *LDE*, *Pilha* e *Fila*. Na classe no, observa-se os atributos *numero*, o qual se refere ao número a ser armazenado no nó, e o ponteiro do próximo nó (*no *proximo*).

A classe *no* é herdada pela classe *LDE*, pois essa utiliza a classe *no* em sua estrutura. Os atributos da *LDE* são o nó que indica o início da lista (*no *inicio*) e o nó que indica o fim da lista (*no *fim*) e cada um desses nós possuem os atributos *numero* e ponteiro do próximo nó, herdados da classe *no*.

Na classe *LDE* o método *criarNo* cria o primeiro nó da lista, funcionando como um inicializador, o qual evita a verificação de primeiro elemento nos métodos de inserção. O método *insereInicio* é responsável por inserir os elementos, apenas no início da lista, deslocando os demais elementos já existentes. O método *insereFim* é responsável por inserir os elementos, apenas no fim da lista, atualizando sempre o ponteiro do último elemento. O método *removeInicio* remove os elementos do inicio da lista, tendo que atualizar o ponteiro de início e garantir que não perca a referência. O método *removeFim* remove os elementos que estão no final da lista e assim como o método *removeInicio*, deve garantir que não perca a referência, mas do último ponteiro. O método *buscar* procura, a partir de um dado número, o elemento ao qual esse número pertence e então retorna o número, do contrário sinaliza que não foi encontrado. O método *imprimir* percorre toda a lista e imprime cada um dos números armazenados nos nós.

A classe *Pilha* herda a classe *LDE*. A classe *Pilha* possui o atributo *topo* que corresponde ao elemento armazenado no topo da pilha, sendo representado pelo nó de inicio da classe *LDE*. O método *getTopo* atribui o número do primeiro elemento da lista, ao topo da pilha. O método *inicializa* utiliza o método *criarNo* da classe *LDE* e insere o primeiro elemento na pilha. O

método *empilha* utiliza o método da classe *LDE insereInicio*. O método *desempilha* utiliza o método da classe *LDE removeInicio*. E por fim, o método *imprimir* utiliza o método *imprimir* da classe *LDE*.

A classe *fila* também herda a classe *LDE*. A classe *Fila*, possui os atributos *primeiro* e *ultimo* que correspondem ao primeiro e último elemento, respectivamente da fila, sendo esses representados pelos nós de inicio e fim da classe *LDE*. Os métodos *getPrimeiro* e *getUltimo* atribuem os números do primeiro e último elemento da lista, ao primeiro e último elemento da fila. O método *inicializa* utliza o método *criarNo* para inserir o primeiro elemento da fila. O método *enfileirar* utiliza o método *insereFim* da classe *LDE* e o método *desempilha* utiliza o método *removeFim* da classe *LDE*. E por fim é utilizado o método *imprimir* da classe *LDE* no método *imprimir*.

5 Experimentos e Resultados

Os experimentos realizado nesse trabalho para a classe *LDE* foi aplicado nas classes *Pilha* e *Fila*, por serem classes que herdam a primeira.

No primeiro experimento, observado na Figura 2, foram testados os métodos, *inicializa*, *empilha*, *desempilha* e *imprimir* da classe *Pilha* e herdam, respectivamente, os métodos *criarNo*, *insereInicio*, *removeInicio* e *imprimir* da classe *LDE*. No experimento foi impresso cada um dos elementos, seguindo a ordem de alocação na lista, ou seja, foram inseridos os números, na sequência: 10, 2, 3 e 4. É possível observar, ainda na Figura 2, que foi empilhada de forma correta, pois a ordem na impressão da pilha segue com o número 10 na última posição e o número 4 no topo da pilha. Em seguida foram removidos os números e o resultado apresentado é o esperado, o primeiro elemento retirado é o topo, representado inicialmente pelo número 4, seguido por 3, 2 e por fim 10.

No segundo experimento, observado na Figura 3, foram testados os métodos, *inicializa*, *enfileira*, *desenfileira* e *imprimir* da classe *Fila* e herdam, respectivamente, os métodos *criarNo*, *insereFim*, *removeFim* e *imprimir* da classe *LDE*. No experimento foi impresso cada um dos elementos, seguindo a ordem de alocação na lista, ou seja, foram inseridos os números, na sequência: 10, 2, 3 e 4. É possível observar, ainda na Figura 3, que foi enfileirada de forma correta, pois a ordem na impressão da fila segue com o número 10 na primeira posição e o número 4 na última posição da fila. Em seguida foram removidos os números e o resultado apresentado é o esperado, o primeiro elemento a

sair da fila é o primeiro da fila, representado inicialmente por 10, seguido por 2, 3 e pelo último elemento 4.

6 Conclusão

Esse trabalho propôs a implementação das estruturas de dados lista dinamica encadeada, fila e pilha. Nos experimentos foram alcançados resultados esperados para o objetivo do trabalho, no qual cada um dos algoritmos apresentaram comportamento esperado.

Referências

[1] Jian Li, Tinghuai Ma, Meili Tang, Wenhai Shen, and Yuanfeng Jin. Improved fifo scheduling algorithm based on fuzzy clustering in cloud computing. *Information*, 8:25, 2017.

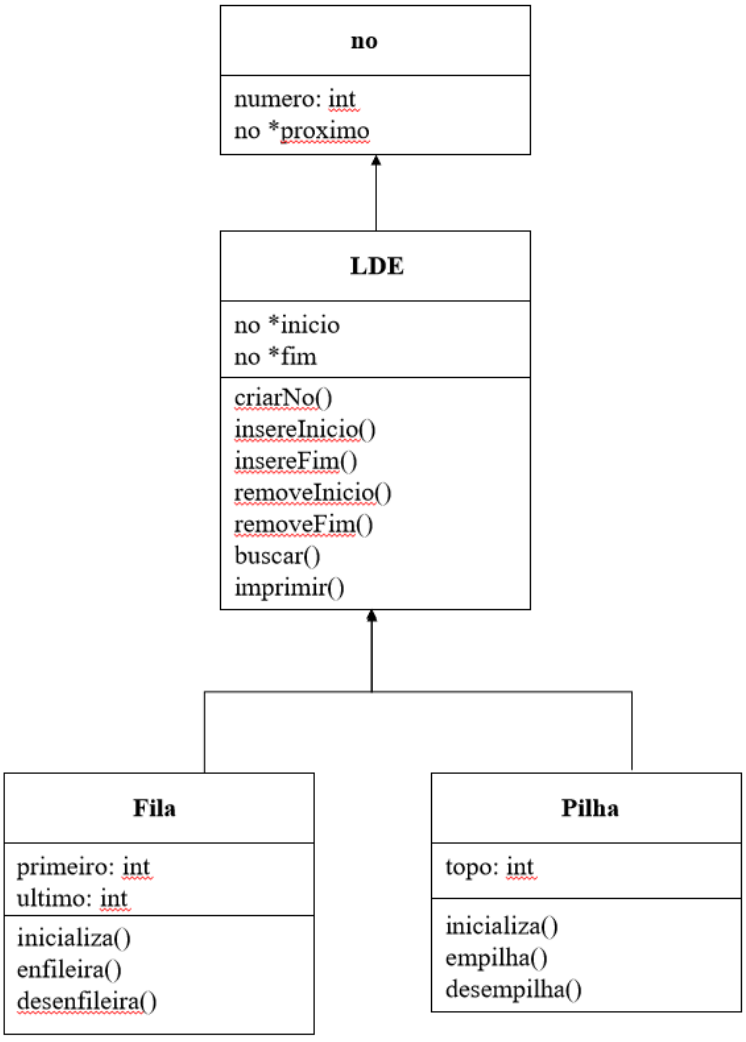

Figura 1: Diagrama das classes: no, LDE, Pilha e Fila

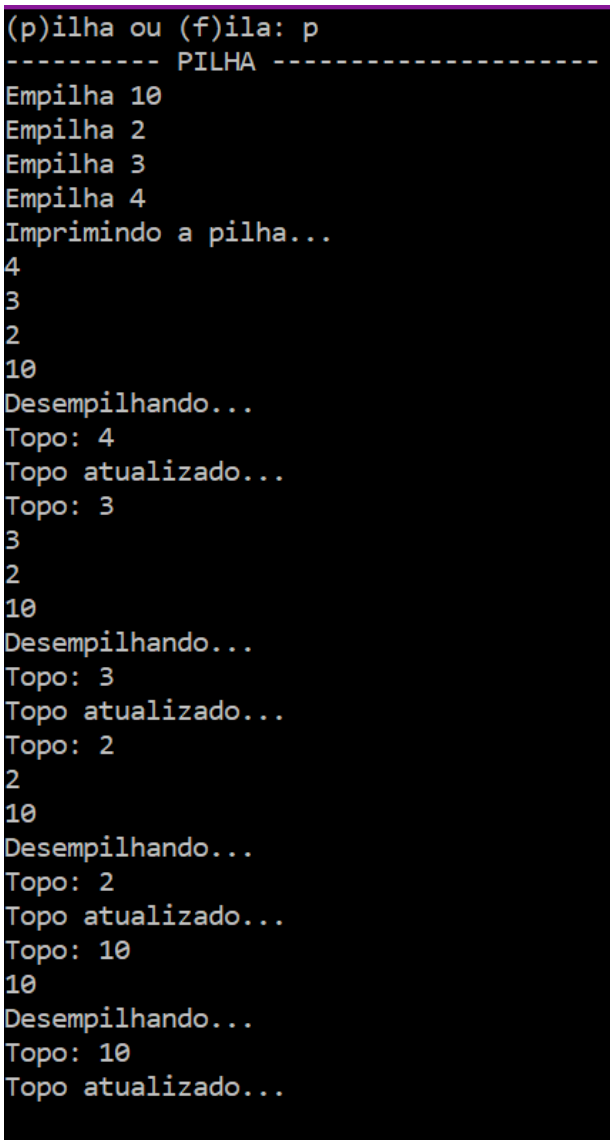

Figura 2: Resultado da classe pilha

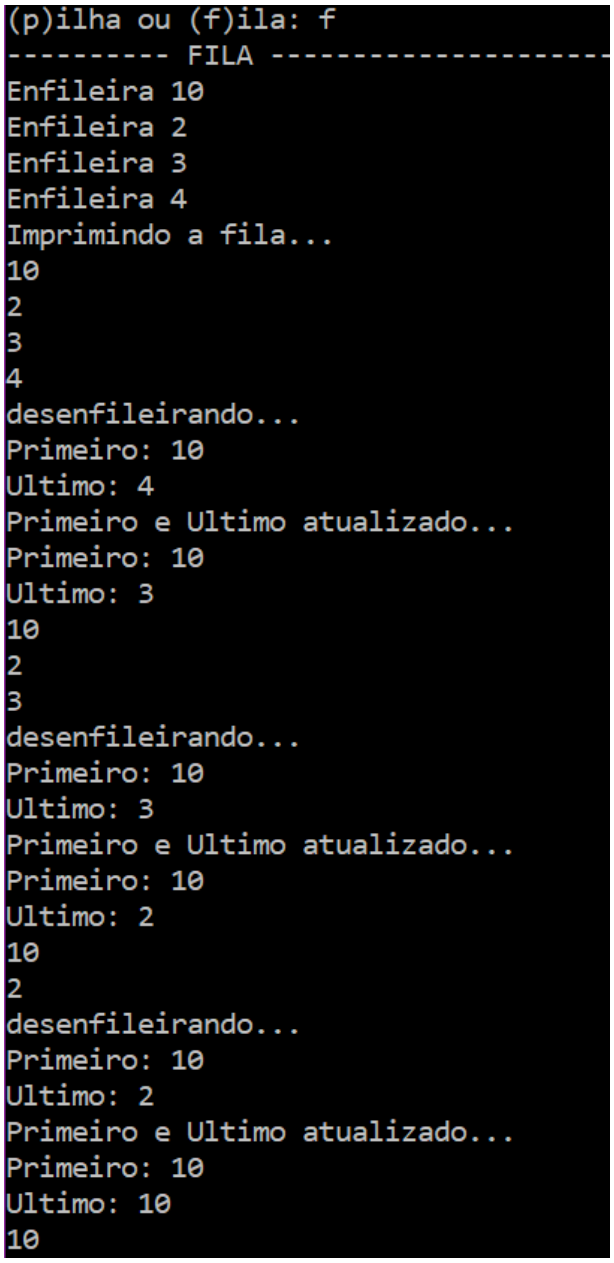

Figura 3: Resultado da classe fila