**Integrantes:** Gabriel Dias Barros, Matheus Santos Silva Pereira

* Considerando a lista da Seção Aspectos Teóricos, como poderiam ser os ponteiros externos para as pilhas e filas utilizando dois tipos tStack e tLine, de forma a facilitar a manipulação dessas estruturas de dados?

Utilizando ponteiros para a pilha onde a informação que interessa é apenas o topo e na Fila outro ponteiro que apenas interessa o primeiro elemento.

* Como poderia-se implementar uma fila de prioridade?

Pode ser criada uma fila de prioridade ordenada na inserção de elementos na lista, que facilitaria na remoção dos mesmos, ou fazer uma inserção desordenada, mas sempre que for fazer uma remoção é necessária uma função que faça a ordenação sempre que for precisar retirar um elemento da lista.

* Uma função de ordenação seria útil?

Sim, facilitaria na inserção dos elementos da lista, mas com um custo computacional maior para remoção dos elementos.

* Uma função de inserção de um elemento ordenado seria útil?

Sim, facilitaria na remoção dos elementos da lista, mas com um custo computacional maior na inserção de elementos na lista, devido a ter uma ordenação para cada elemento inserido.

* Para remover o menor elemento de uma fila ascendente, qual operação seria necessária? *remove\_front(l)* ou *remove\_rear(l)*?

Para remover o menor elemento de uma fila ascendente, onde os elementos estão organizados em ordem crescente, a operação necessária seria remove\_front(l).

* O que seria necessário adicionar aos nós da lista apresentada na seção Aspectos Teóricos, de forma que ela seja duplamente encadeada?

Para tornar a lista encadeada apresentada na seção Aspectos Teóricos uma lista duplamente encadeada, precisamos adicionar um ponteiro adicional para o nó anterior em cada nó da lista.

* O que seria necessário adicionar aos nós da lista apresentada na seção Aspectos Teóricos, de forma que ela seja heterogênea?

Para tornar a lista apresentada na seção Aspectos Teóricos heterogênea, podemos adicionar um ponteiro para os dados dentro da estrutura do nó. Isso permitirá que cada nó da lista contenha dados de diferentes tipos, tornando a lista heterogênea.

* Como poderia ser criado um nó de cabeçalho para a lista apresentada na seção Aspectos Teóricos, de forma que esse nó de cabeçalho possuísse a informação da quantidade de nós existentes na lista?

Para criar um nó de cabeçalho precisamos modificar a estrutura para incluir um campo adicional para representar o nó de cabeçalho e também uma variável para armazenar o tamanho da lista. Um exemplo seria:

typedef struct {

tNode\* header; // Nó de cabeçalho

int size; // Tamanho da lista

} tList;

### **C++**

* Listas (linked lists):
  + A linguagem C++ oferece suporte para listas encadeadas (linked lists) através da implementação manual de nós e ponteiros, assim como em C. No entanto, bibliotecas como a STL (Standard Template Library) fornecem implementações de listas duplamente encadeadas (std::list) e listas circulares (std::forward\_list) que podem ser facilmente utilizadas.
* Pilhas (stacks):
  + A STL fornece a classe std::stack, que é uma adaptação da estrutura de dados pilha e fornece funcionalidades para inserção (push) e remoção (pop) de elementos no topo da pilha.
* Filas (queues):
  + Assim como as pilhas, a STL oferece a classe std::queue, que é uma implementação da estrutura de dados fila. Ela fornece operações para inserir elementos no final da fila (push) e remover elementos do início da fila (pop).

### **Python**

* Listas (lists):
  + A linguagem Python possui um tipo de dados embutido chamado list, que é uma sequência ordenada de itens. As listas em Python podem ser facilmente manipuladas usando métodos embutidos, como append() para adicionar elementos, pop() para remover elementos, e muitos outros métodos úteis.
* Pilhas (stacks):
  + As pilhas em Python podem ser implementadas facilmente usando a estrutura de lista embutida. Os métodos append() e pop() podem ser utilizados para adicionar e remover elementos no topo da pilha, respectivamente. Além disso, o módulo collections fornece a classe deque, que pode ser usada para implementar uma pilha eficiente.
* Filas (queues):
  + Para implementar uma fila em Python, podemos usar a estrutura de lista embutida. No entanto, para uma fila mais eficiente, o módulo queue oferece a classe Queue, que fornece operações seguras para enfileirar e desenfileirar elementos.

### **MATLAB**

* Listas (lists):
  + Em MATLAB, as listas podem ser implementadas usando vetores (arrays). Os arrays em MATLAB são coleções ordenadas de elementos que podem ser facilmente manipuladas usando funções e operadores embutidos.
* Pilhas (stacks):
  + As pilhas podem ser implementadas em MATLAB usando arrays. Para adicionar elementos no topo da pilha, podemos usar a função push ou o operador de indexação ([]), e para remover elementos do topo da pilha, podemos usar a função pop.
* Filas (queues):
  + Da mesma forma que as pilhas, as filas podem ser implementadas em MATLAB usando arrays. Para enfileirar elementos no final da fila, podemos usar a função enqueue ou o operador de concatenação ([]), e para desenfileirar elementos do início da fila, podemos usar a função dequeue