Nome: Arthur Henrique Sousa Cruz **Matrícula:** 2020160256

Pré-requisitos:

Para a execução do código C++ 14. Os arquivos de cada diretório são compilados utilizando o comando “make” dentro do respectivo diretório.

Estruturas:

As estruturas estão divididas em pastas. Cada pasta contém seu próprio Makefile e seu próprio arquivo para testes.

Lista Sem Cauda:

Para a implementação de uma Lista sem cauda foram criados dois arquivos: ListaSemCauda.hpp e ListaSemCauda.cpp. Foi utilizado o conceito de classes para a implementação das TAD’s (ressaltando que a Orientação a Objetos não foi o foco). A classe Elemento representa um elemento da lista e contém os seguintes atributos e métodos:

- **int dado:** Atributo que armazena o dado no elemento.
- **Elemento* próximo:** Ponteiro para o próximo elemento da Lista.
- **Elemento(int dado):** Construtor da classe.
- **set_proximo(Elemento* proximo):** O argumento do método se torna o elemento da sequência do elemento atual.
- **get_dado():** Retorna o dado armazenado.
- **get_proximo():** Retorna o elemento seguinte ao elemento atual.

Por sua vez a lista é composta pelos seguintes atributos e métodos:

- **Elemento* inicio:** Atributo que indica o primeiro Elemento da Lista.
- **ListaSemCauda():** Construtor da classe.
- **~ListaSemCauda():** Destrutor. Desaloca os elementos que estão presentes na Lista.
- **insere_inicio(int dado):** Insere um elemento no início da lista. Se o ponteiro de início da lista é nulo, ele passa a apontar para o novo elemento e a função finaliza. Caso contrário, o início passa a ser o próximo do novo elemento e o novo elemento passa a ser o início.
- **insere_fim(int dado):** Insere um elemento no final da lista. e o ponteiro de início da lista é nulo, ele passa a apontar para o novo elemento e a função finaliza. Caso contrário a lista é percorrida até o último elemento e o novo elemento é inserido ao fim do último.
- **get_lista_string():** Método criado para teste. Retorna uma string com todos os elementos da lista.

O arquivo main.cpp foi criado para testar as estruturas.

Lista Com Cauda:

Para a implementação de uma Lista sem cauda foram criados dois arquivos: ListaComCauda.hpp e ListaComCauda.cpp. Foi utilizado o conceito de classes para a implementação das TAD’s (ressaltando que a Orientação a Objetos não foi o foco). A classe Elemento se mantém igual à da lista sem cauda.

A lista é composta pelos seguintes atributos e métodos:

- **Elemento* inicio:** Atributo que indica o primeiro Elemento da Lista.
- **Elemento*** **fim:** Atributo que indica o último Elemento da Lista.

- **ListaComCauda():** Construtor da classe.
- **~ListaComCauda():** Destrutor. Desaloca os elementos que estão presentes na Lista.
- **insere_inicio(int dado):** Insere um elemento no início da lista. Se o ponteiro de início da lista é nulo, ele e o ponteiro para o final da lista passam a apontar para o novo elemento e a função finaliza. Caso contrário, o início passa a ser o próximo do novo elemento e o novo elemento passa a ser o início.
- **insere_fim(int dado):** Insere um elemento no final da lista. Se o ponteiro de início da lista é nulo, ele e o ponteiro para o final da lista passam a apontar para o novo elemento e a função finaliza. Caso contrário o novo elemento passa a ser o próximo do último e o ponteiro para o último elemento passa a apontar para o novo elemento.
- **get_lista_string()**: Método criado para teste. Retorna uma string com todos os elementos da lista.

O arquivo main.cpp foi criado para testar as estruturas.

Pilha:

A implementação da Pilha foi baseada na Lista sem cauda. Essa Lista é mais adequada para a implementação da pilha, pois a inserção e remoção ocorrem em somente uma das pontas da estrutura. Foram criados dois arquivos para a pilha: Pilha.hpp e Pilha.cpp. Foi utilizado o conceito de classes para a implementação das TAD's (ressaltando que a Orientação a Objetos não foi o foco). A classe Elemento se mantém igual à da lista sem cauda.

A pilha é composta pelos seguintes atributos e métodos:

- **Elemento* inicio:** Atributo que indica o primeiro Elemento da Pilha.
- **Pilha():** Construtor da classe.
- **~Pilha():** Destrutor. Desaloca os elementos que estão presentes na Pilha.
- **insere_na_pilha(int dado):** Insere um elemento no início da lista. Se o ponteiro de início da lista é nulo, ele passa a apontar para o novo elemento e a função finaliza. Caso contrário, o início passa a ser o próximo do novo elemento e o novo elemento passa a ser o início.
- **remove_da_pilha(int dado):** Remove o último elemento inserido na Pilha. Se a pilha está vazia, o programa finaliza com erro. Caso contrário o início passa a apontar para o segundo elemento da lista e o primeiro é deletado da memória. O dado armazenado é salvo antes da deleção e retornado após ela.

O arquivo main.cpp foi criado para testar as estruturas.

Fila:

Foram criadas duas filas, uma baseada na lista sem cauda (fila sem cauda) e outra baseada na lista com cauda (fila com cauda). Sua única variação foi a inserção no final da lista. Os arquivos que as representam são, respectivamente, FilaSemCauda.hpp, FilaSemCauda.cpp e FilaComCauda.hpp, FilaComCauda.cpp. Foi utilizado o conceito de classes para a implementação das TAD's (ressaltando que a Orientação a Objetos não foi o foco). A classe Elemento se mantém igual à da lista sem cauda, porém ela foi implementada separadamente nos arquivos Elemento.hpp e Elemento.cpp.

Por sua vez a lista é composta pelos seguintes atributos e métodos:

- **Elemento* inicio:** Atributo que indica o primeiro Elemento da Lista.

- **Elemento* fim:** Atributo que indica o último Elemento da Lista (Somente na Fila com cauda).
- **FilaSemCauda e FilaComCauda():** Construtores de suas respectivas classes.
- **~ListaSemCauda e ~ListaComCauda():** Destrutores. Desalocam os elementos que estão presentes nas respectivas Filas.
- **remove_inicio(int dado):** Remove o primeiro elemento da Fila. Se a fila está vazia o programa finaliza com erro. Caso contrário o início se torna o segundo elemento da lista e o primeiro é deletado. O dado armazenado é salvo antes da deleção e retornado após ela.
- **insere_fim(int dado):** Insere um elemento no final da lista.
  - Para a Fila com Cauda: Se o ponteiro de início da lista é nulo, ele e o ponteiro para o final da lista passam a apontar para o novo elemento e a função finaliza. Caso contrário o novo elemento passa a ser o próximo do último e o ponteiro para o último elemento passa a apontar para o novo elemento.
  - Para a Fila sem Cauda: Se o ponteiro de início da lista é nulo, ele passa a apontar para o novo elemento e a função finaliza. Caso contrário a lista é percorrida até o último elemento e o novo elemento é inserido ao fim do último.

  O arquivo main.cpp foi criado para testar as estruturas.

- **Comparação de Complexidade**
  - **Sem Cauda**:
    - Inserção: Como se tem um ponteiro apenas para o início da fila é necessário percorrer todos os elementos da lista para chegar ao final. Assim sendo, a inserção no pior caso ocorre em $O(N)$, sendo N o número de elementos da lista.
    - Remoção: A remoção é feita no início da lista e por isso ocorre em $\Theta(1)$.
  - **Complexidade Com Cauda:**
    - Inserção: Como se tem um ponteiro para o final da fila é possível inserir na fila após um acesso. Dessa forma a complexidade da inserção no pior caso cai para $O(1)$.
    - Remoção: A remoção é feita no início da lista e por isso ocorre em $\Theta(1)$.