Stack – Pilha
LIFO "Last In, First Out" (último a entrar, primeiro a sair).

Aplicações:

Navegador
Expressões aritméticas
Compiladores
Chamada de subrotinas (pilha de execução)

Estrutura:

Funções:
empty() – Retorna um valor lógico indicando se a pilha está cheia.
size() – Retorna ao tamanho da pilha.
top() ou peek() – Retorna uma referência ao elemento mais ao topo da pilha.
push(elemento) – Insere o elemento "elemento" no topo da pilha.
pop() – Remove o elemento do topo da pilha.

A complexidade de todas as operações é O(1).

Implementações comuns:
Dinâmica, usando uma lista encadeada onde cada elemento aponta para o próximo.
Estática, usando um array que cresce conforme necessário.

Exemplo-1-pilha.py
Exemplo-1-pilha-completo.py
Exemplo-1-pilha-OO.py
Exemplo-1-pilha-struct.c

Queue – Fila
FIFO "First In, First Out" (primeiro a entrar, primeiro a sair)

Aplicações:
Impressoras
Escalonamento de processos
Pedidos

Funções:
empty() – Retorna um valor lógico indicando se a fila está vazia.
size() – Retorna o tamanho da fila.
enqueue(elemento) – Adiciona um elemento ao final da fila.
dequeue() – Remove o elemento do início da fila.

front() – Retorna o elemento no início da fila sem removê-lo.

A complexidade de todas as operações é O(1).

Implementações comuns:
Dinâmica, usando uma lista encadeada com dois ponteiros (frente e fim).
Estática, usando um array circular para gerenciar o início e o final da fila.

Exemplo-2-fila.py
Exemplo-2-fila-OO.py

Deque (Doubly Ended Queue) – Fila de duas pontas
Double-ended Queue: permite inserções e remoções de elementos tanto no início quanto no fim.

Aplicações:
Gerenciamento de buffer: Suporte a operações de fila e pilha.
Jogo e simulações: Manipulação eficiente de dados com acesso em ambas as extremidades.

Funções:
empty() – Retorna um valor lógico indicando se o deque está vazio.
size() – Retorna o tamanho do deque.
push_front(elemento) – Adiciona um elemento no início do deque.
push_back(elemento) – Adiciona um elemento no fim do deque.
pop_front() – Remove o elemento do início do deque.
pop_back() – Remove o elemento do fim do deque.
front() – Retorna o elemento do início do deque.
back() – Retorna o elemento do fim do deque.

A complexidade de todas as operações é O(1).

Implementações comuns:
Dinâmica, usando uma lista duplamente encadeada.
Estática, usando um array circular para suportar operações nas duas extremidades.

Exemplo-3-deque-OO.py

Lista – Lista Simples
Estrutura de dados linear que pode armazenar elementos de forma contígua ou não contígua, permitindo acesso direto e sequencial aos elementos.

Aplicações:
Armazenamento de dados: Listagem de elementos em memória.

Implementação de outras estruturas: base para pilhas e filas.

Funções:
empty() – Verifica se a lista está vazia.
size() – Retorna o tamanho da lista.
get(indice) – Retorna o elemento no índice especificado.
insert(indice, elemento) – Insere um elemento no índice especificado.
remove(indice) – Remove o elemento no índice especificado.

Complexidade:
Dinâmica: Inserção e remoção podem ser O(1) se a posição for conhecida.
Estática: Acesso e alteração de elementos são O(1). Inserção e remoção podem ser O(n).

Implementações comuns:
Dinâmica, usando uma lista encadeada.
Estática, usando uma lista (array) com tamanho fixo ou dinâmico.

Variações:
Lista Circular: variação da lista encadeada, onde o último nó aponta para o primeiro nó, formando um ciclo.
Lista Simplesmente Encadeada: estrutura de dados onde cada elemento (nó) possui um valor e um ponteiro para o próximo nó.
Lista Duplamente Encadeada: estrutura de dados onde cada nó possui um valor, um ponteiro para o próximo nó e um ponteiro para o nó anterior.

Exemplo-4-lista-OO.py
Exemplo-5-lista-circular-OO.py

Obs.: é comum também criar a função init() para iniciar a estrutura.

Implementar as estruturas apresentadas em C/C++.