REVISÃO SOBRE ESTRUTURAS DE DADOS LINEARES

GSI512 – Estrutura de Dados II

Prof. Dr. Rafael D. Araújo

rafael.araujo@ufu.br

http://www.facom.ufu.br/~rafaelaraujo

UFU

Objetivo da aula

- Relembrar os conceitos de estruturas de dados não-lineares.

O que são tipos de dados?

- São valores que uma variável pode assumir.
  - *tipo inteiro: pode assumir valores inteiros (positivos e negativos)*
  - *tipo lógico (ou booleano): assume somente dois valores, verdadeiro ou falso*

- Tipos básicos (primitivos):
  - Não-definidos em termos de outros tipos
  - *inteiro, real, lógico e caractere*

Outros tipos de dados

- arranjos (vetores e matrizes)
  - *Estruturas homogêneas*
- estruturas (ou registros)
  - *Estruturas heterogêneas*
- enumeração
- classes
- ponteiros

Vetor é um *array* unidimensional.

Matriz é um *array* bidimensional.

Tipos de dados

- Operações são associadas a um tipo de dados:
  - *Ao executar uma operação, devemos assegurar que seus operandos e seu resultado possuem o tipo de dados correto.*
  - *Por exemplo: ao tipo integer, são associadas as operações adição, subtração, multiplicação e divisão.*

Estruturas de dados

- As estruturas de dados representam maneiras de se **organizar** os dados manipulados por um programa de forma coerente e racional.
  - *Acesso e modificações mais fáceis*
  - *Estruturação conceitual dos dados (*relacionamento lógico entre dados)
- Um programa faz uso de diferentes estruturas de dados, conforme sua necessidade.
  - *Decisão de projeto*

Estruturas de dados

- Lineares:
  - Listas
  - Pilhas
  - Filas
- Não-lineares:
  - Árvores
  - Grafos

Tipo Abstrato de Dados (TAD)

- Um **TAD** pode ser considerado um modelo matemático associado a um conjunto de operações
  - *É uma forma de definir um **novo tipo de dado** de forma abstrata*
  - *Um conjunto de valores e possíveis operações sobre esses valores*
- Separa o conceito (definição do tipo) e a implementação das operações
- A estrutura interna fica "escondida"
  - Caixa-preta

Tipo Abstrato de Dados (TAD)

- Satisfaz as propriedades de
  - *encapsulamento: definição isolada de outras unidades do programa*
  - *invisibilidade e proteção: representação do tipo deve ser acessada somente no ambiente encapsulado*

Tipo Abstrato de Dados (TAD)

- Vantagens
  - **Reutilização**: utilização do mesmo TAD em diversas aplicações diferentes.
  - **Manutenção**: alteração do TAD sem alterar as aplicações que o utilizam
  - **Segurança**: clientes não podem alterar a representação e nem tornar os dados inconsistentes
  - **Correção**: o TAD foi testado e funciona corretamente

Projeto de um TAD

- Escolha de operações adequadas para uma determinada estrutura de dados, visando definir seu comportamento
- A implementação de um TAD escolhe uma estrutura de dados para representá-lo
- Dicas:
  - *definir pequeno número de operações*
  - *conjunto de operações deve ser suficiente para realizar as computações necessárias para quem utiliza o TAD*
  - *cada operação deve ter um propósito bem definido*

TAD em C

- Crie um arquivo .h com as funções

- Crie um arquivo .c com a implementação das funções

TADMatriz.h

```
struct matriz {
   int lin;
   int col;
   int* v;
};
typedef struct matriz Matriz;

/* Função cria matriz: Aloca e retorna uma Matriz
m por n */
Matriz* cria (int m, int n);
```

TADMatriz.c

```
#include <stdlib.h>
#include "TADMatriz.h"

Matriz* cria (int m, int n) {
   ...
}
```

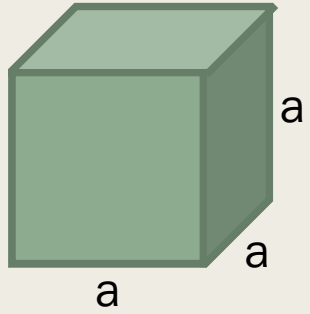
TAD em C

- Inclua o arquivo .h no programa principal (cliente)

MeuProgramaExemplo.c

```
#include <stdio.h>
#include <stdlib.h>
#include "matriz.h"

int main(int argc, char *argv[])
{
   float a,b,c,d;
   Matriz *M;

   // criação de uma matriz
   M = cria(5,5);

   ...

   return 0;
}
```

Exercício 1



- Defina um TAD para representar um cubo. Chame o tipo de dados de Cubo e implemente as seguintes operações:
  1. criar: aloca espaço de memória para um cubo;
  2. liberar: operação que libera a memória alocada por um cubo;
  3. acessar: operação que devolve o comprimento de uma aresta;
  4. alterar_aresta: operação que atribui um novo valor às arestas;
  5. calcular_area_lado: operação que calcula a área de uma das faces do cubo.
  6. calcular_area_total: operação que calcula a área total do cubo.
  7. calcular_volume: operação que calcula o volume do cubo.
- Crie um programa principal (main) e faça um menu para utilizar as operações do no tipo de dados.
- Entregar até às 23h59 do dia 07/03/2021, via Teams.

Listas lineares

- Uma lista linear é um conjunto de n elementos

$$x_1, x_2, \ldots, x_n$$

cuja propriedade estrutural envolve as posições relativas de seus elementos. Supondo $n>0$, temos:

  - x_1 *é o primeiro elemento*
  - *para* $1<k<n$, x_k *é precedido por* x_{k-1} *e seguido por* x_{k+1}
  - x_n *é o último elemento*

Listas lineares

- Todos os elementos são do mesmo tipo de dados.
- Cada elemento é chamado de nó.
- Preserva a relação de ordem entre seus elementos.
- Não necessariamente os elementos estão fisicamente em ordem na memória.
  - ***Lista linear sequencial*** *(ou contígua): quando os elementos estão fisicamente em ordem.*
  - ***Lista linear encadeada****: quando não estão fisicamente em ordem.*

Listas lineares

- Principais operações:
  - *Criação de uma lista*
  - *Inicialização*
  - *Remoção de um elemento da lista (x_k)*
  - *Inserção de um elemento novo antes ou depois de x_k*
  - *Remoção do elemento x_k*
  - *Acesso de um elemento x_k da lista para consulta ou alteração*
  - *Combinação de duas ou mais listas*
  - *Ordenar os elementos da lista*
  - *Copiar a lista para outro espaço*

Exemplos práticos

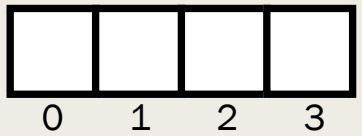
- Consultório médico:
  - *as pessoas na sala de espera estão sentadas em qualquer lugar, porém sabe-se quem é o próximo a ser atendido, e o seguinte, e assim por diante.*
- Lista de aviões que devem decolar:
  - *ordem definida de acordo com a prioridade.*
- Setores de um HD:
  - *lista de setores de disco a serem acessados por um SO.*
- Compras online:
  - *carrinho de compras*
  - *gestão de entregas*

Tipos de alocação

- Alocação estática
  - *Tamanho **pré-definido** durante a **compilação.***
- Alocação dinâmica
  - *Tamanho **dinâmico** definido durante a **execução.***

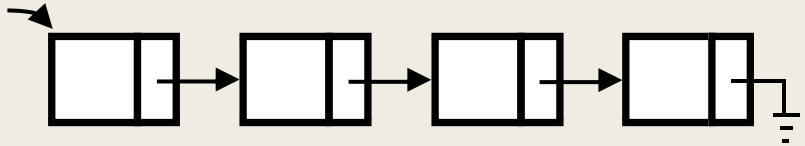
Forma de agrupamento

- Sequencial
  - *Espaço contíguo na memória.*



- Vantagem: fácil acesso a qualquer elemento.
- Desvantagem: inserir ou remover elementos no meio da lista.

- Encadeada
  - *Elementos dispersos na memória.*
  - *Cada elemento armazena sua informação e o endereço da próxima posição*

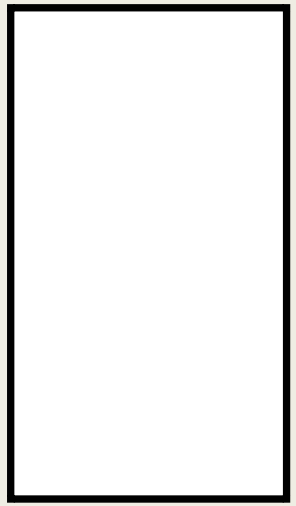


- Vantagem: facilidade de inserção ou remoção de um elemento em qualquer ponto da lista.
- Desvantagem: acesso não indexado aos elementos e gasto adicional de memória.

Políticas de inserção e remoção

- Dependendo da política adotada, a lista recebe um nome diferente
  - *Pilha: LIFO - **L**ast **I**n **F**irst **O**ut*
    - Inserção e remoção somente do topo

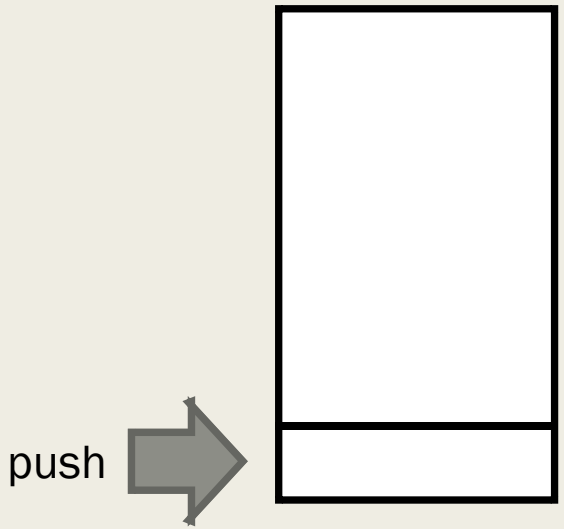
Exemplo: chamada de procedimentos, para armazenar o endereço de retorno (e os parâmetros reais).



Políticas de inserção e remoção

- Dependendo da política adotada, a lista recebe um nome diferente
  - *Pilha: LIFO - **L**ast **I**n **F**irst **O**ut*
    - Inserção e remoção somente do topo

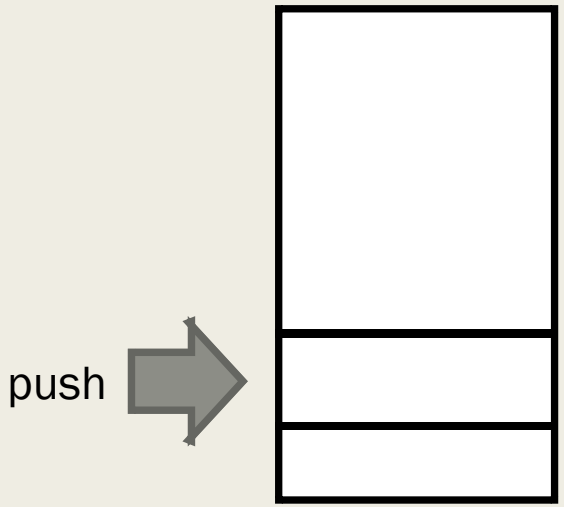
Exemplo: chamada de procedimentos, para armazenar o endereço de retorno (e os parâmetros reais).



Políticas de inserção e remoção

- Dependendo da política adotada, a lista recebe um nome diferente
  - *Pilha: LIFO - **L**ast **I**n **F**irst **O**ut*
    - Inserção e remoção somente do topo

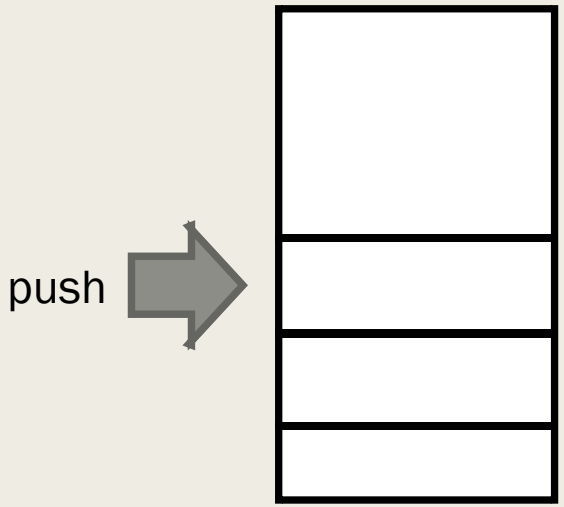
Exemplo: chamada de procedimentos, para armazenar o endereço de retorno (e os parâmetros reais).



Políticas de inserção e remoção

- Dependendo da política adotada, a lista recebe um nome diferente
  - *Pilha: LIFO - **L**ast **I**n **F**irst **O**ut*
    - Inserção e remoção somente do topo

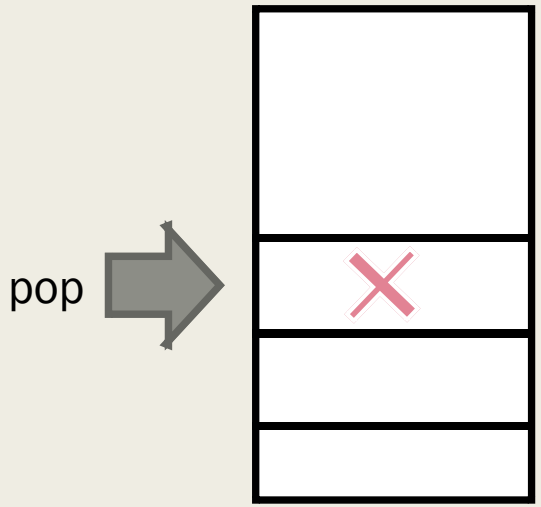
Exemplo: chamada de procedimentos, para armazenar o endereço de retorno (e os parâmetros reais).



Políticas de inserção e remoção

- Dependendo da política adotada, a lista recebe um nome diferente
  - *Pilha: LIFO - **L**ast **I**n **F**irst **O**ut*
    - Inserção e remoção somente do topo

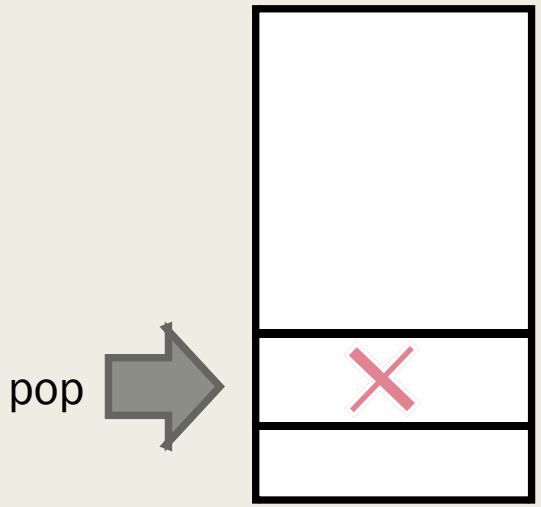
Exemplo: chamada de procedimentos, para armazenar o endereço de retorno (e os parâmetros reais).



Políticas de inserção e remoção

- Dependendo da política adotada, a lista recebe um nome diferente
  - *Pilha: LIFO - **L**ast **I**n **F**irst **O**ut*
    - Inserção e remoção somente do topo

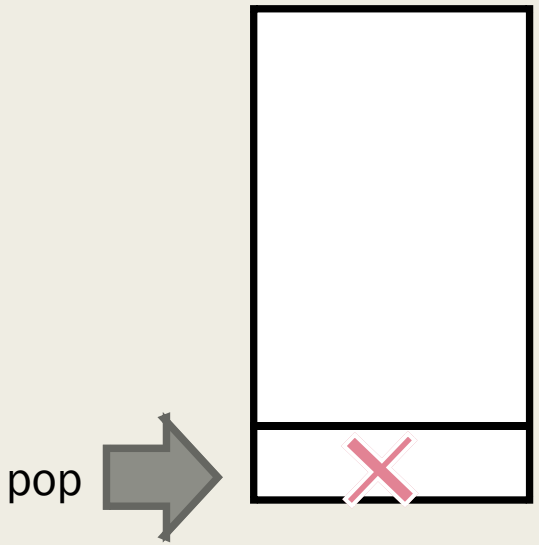
Exemplo: chamada de procedimentos, para armazenar o endereço de retorno (e os parâmetros reais).



Políticas de inserção e remoção

- Dependendo da política adotada, a lista recebe um nome diferente
  - *Pilha: LIFO - **L**ast **I**n **F**irst **O**ut*
    - Inserção e remoção somente do topo

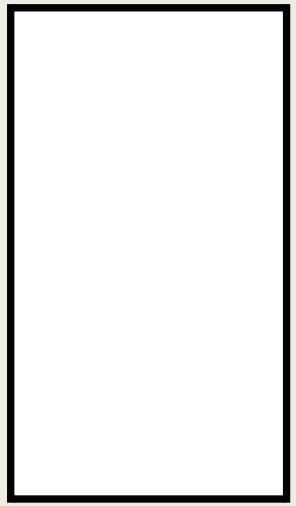
Exemplo: chamada de procedimentos, para armazenar o endereço de retorno (e os parâmetros reais).



Políticas de inserção e remoção

- Dependendo da política adotada, a lista recebe um nome diferente
  - *Pilha: LIFO - **L**ast **I**n **F**irst **O**ut*
    - Inserção e remoção somente do topo

Exemplo: chamada de procedimentos, para armazenar o endereço de retorno (e os parâmetros reais).



Políticas de inserção e remoção

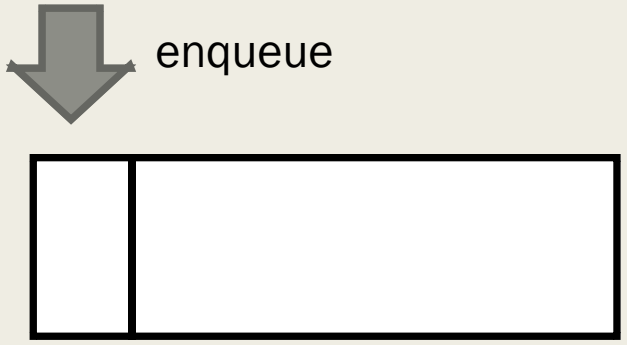
- Dependendo da política adotada, a lista recebe um nome diferente
  - *Fila: FIFO - **F**irst **I**n **F**irst **O**ut*
    - Inserção e remoção em extremidades opostas

Exemplo: os processos prontos para serem executados pelo SO (aguardando apenas a disponibilidade da CPU).

Políticas de inserção e remoção

- Dependendo da política adotada, a lista recebe um nome diferente
  - *Fila: FIFO - **F**irst **I**n **F**irst **O**ut*
    - Inserção e remoção em extremidades opostas

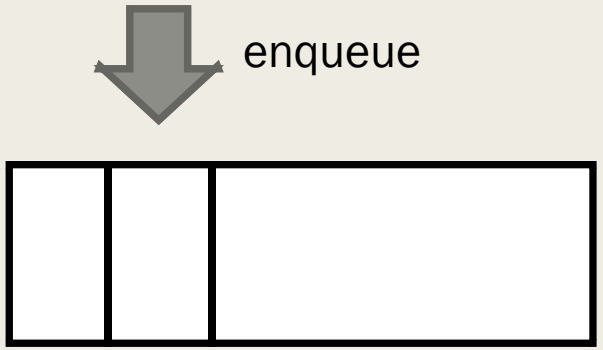
Exemplo: os processos prontos para serem executados pelo SO (aguardando apenas a disponibilidade da CPU).



Políticas de inserção e remoção

- Dependendo da política adotada, a lista recebe um nome diferente
  - *Fila: FIFO - **F**irst **I**n **F**irst **O**ut*
    - Inserção e remoção em extremidades opostas

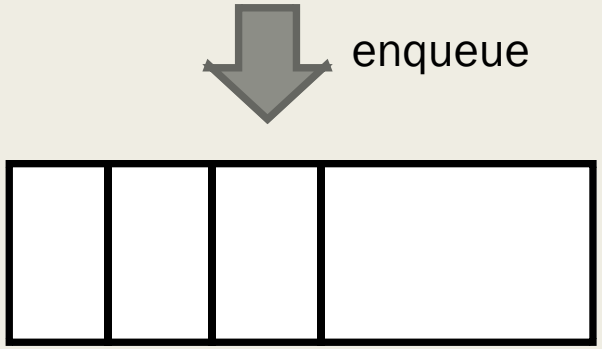
Exemplo: os processos prontos para serem executados pelo SO (aguardando apenas a disponibilidade da CPU).



Políticas de inserção e remoção

- Dependendo da política adotada, a lista recebe um nome diferente
  - *Fila: FIFO - **F**irst **I**n **F**irst **O**ut*
    - Inserção e remoção em extremidades opostas

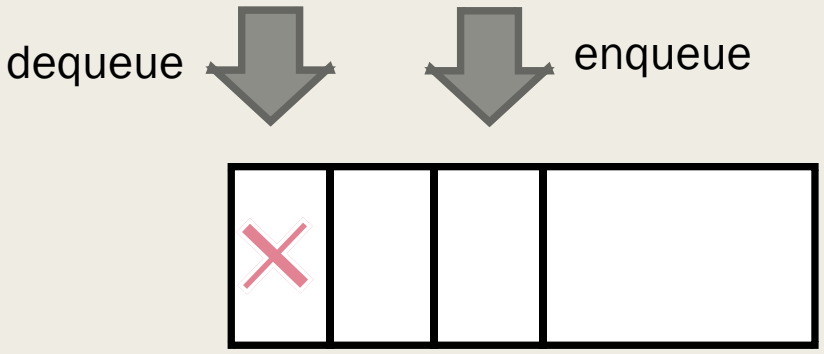
Exemplo: os processos prontos para serem executados pelo SO (aguardando apenas a disponibilidade da CPU).



Políticas de inserção e remoção

- Dependendo da política adotada, a lista recebe um nome diferente
  - *Fila: FIFO - **F**irst **I**n **F**irst **O**ut*
    - Inserção e remoção em extremidades opostas

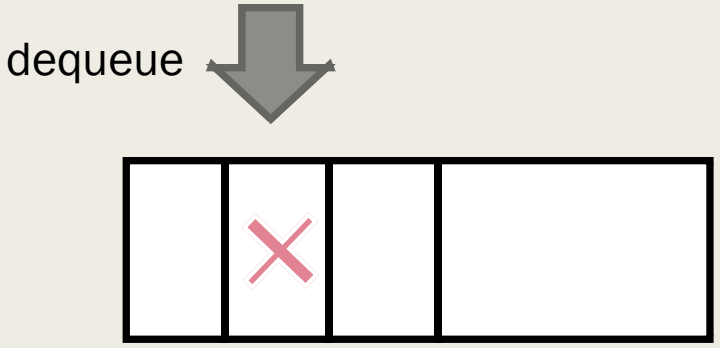
Exemplo: os processos prontos para serem executados pelo SO (aguardando apenas a disponibilidade da CPU).



Políticas de inserção e remoção

- Dependendo da política adotada, a lista recebe um nome diferente
  - *Fila: FIFO - **F**irst **I**n **F**irst **O**ut*
    - Inserção e remoção em extremidades opostas

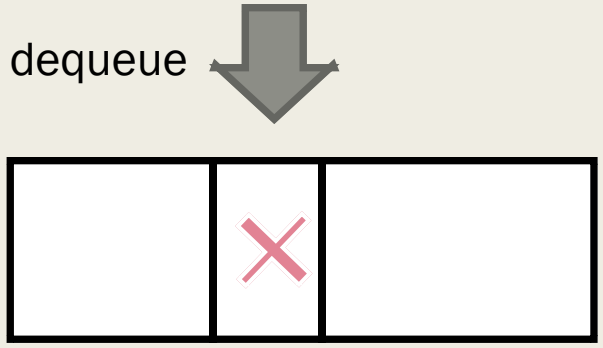
Exemplo: os processos prontos para serem executados pelo SO (aguardando apenas a disponibilidade da CPU).



Políticas de inserção e remoção

- Dependendo da política adotada, a lista recebe um nome diferente
  - *Fila: FIFO - **F**irst **I**n **F**irst **O**ut*
    - Inserção e remoção em extremidades opostas

Exemplo: os processos prontos para serem executados pelo SO (aguardando apenas a disponibilidade da CPU).



Políticas de inserção e remoção

- Dependendo da política adotada, a lista recebe um nome diferente
  - *Fila: FIFO - **F**irst **I**n **F**irst **O**ut*
    - Inserção e remoção em extremidades opostas

Exemplo: os processos prontos para serem executados pelo SO (aguardando apenas a disponibilidade da CPU).

Pilha (*stack*)

- Operações básicas:
  - *push (empilhar): inserir no topo;*
  - *pop (desempilhar): retirar do topo;*
  - *top: observar o topo;*
  - *vazia: verifica se a pilha não contém elementos.*

Fila (*queue*)

- Operações básicas:
  - *enfileirar: inserir no final da fila;*
  - *desenfileirar: retirar do início da fila;*
  - *vazia: verifica se a fila não contém elementos.*

- Implementação especial:
  - *Fila de prioridades*

Exercício 2

- Faça um programa para ler um número inteiro (entre 0 e 1023), converter este número de decimal para binário, usando pilha, e apresentar o resultado da conversão na tela.
- Entregar até às 23h59 do dia 07/03/2021, via Teams.