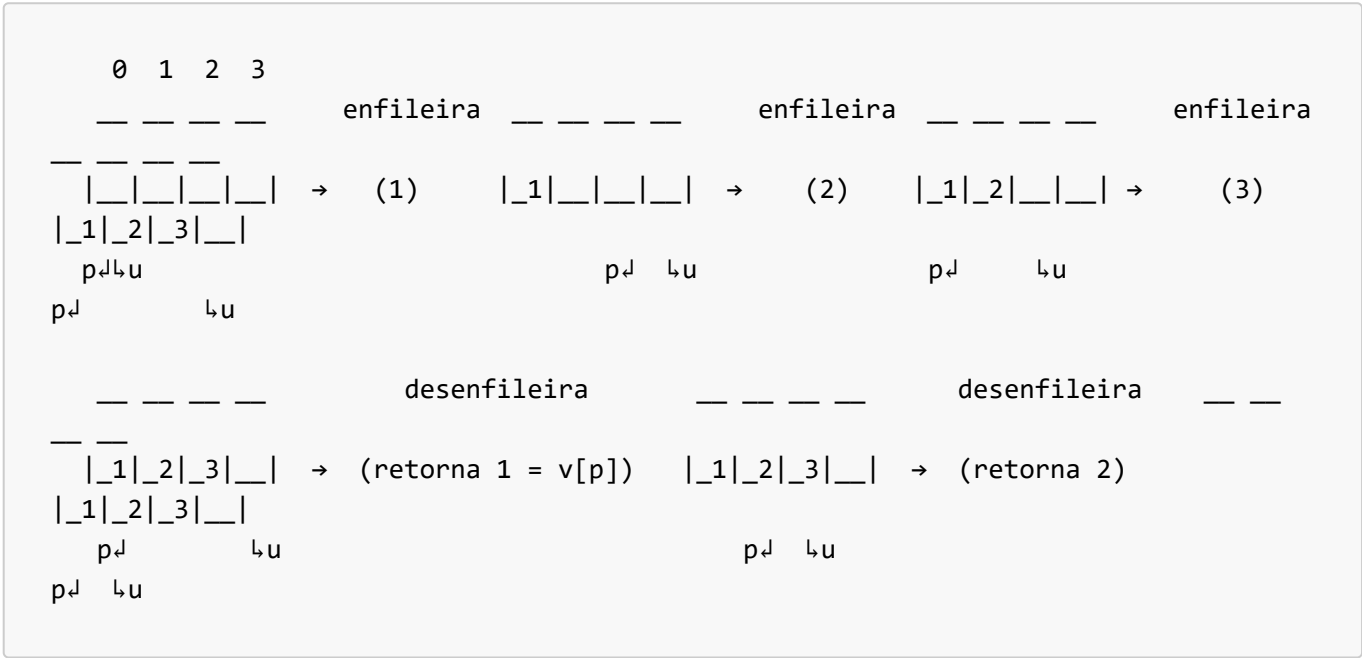
Filas

- É uma estrutura em que os elementos são inseridos e removidos
- **Regra**: Primeiro que entra é o primeiro que sai (FIFO: First-in, first-out
- **Aplicação:** Casos em que desejamos construir uma memória de dados e recuperá-los na mesma ordem que foram inseridos

VETOR

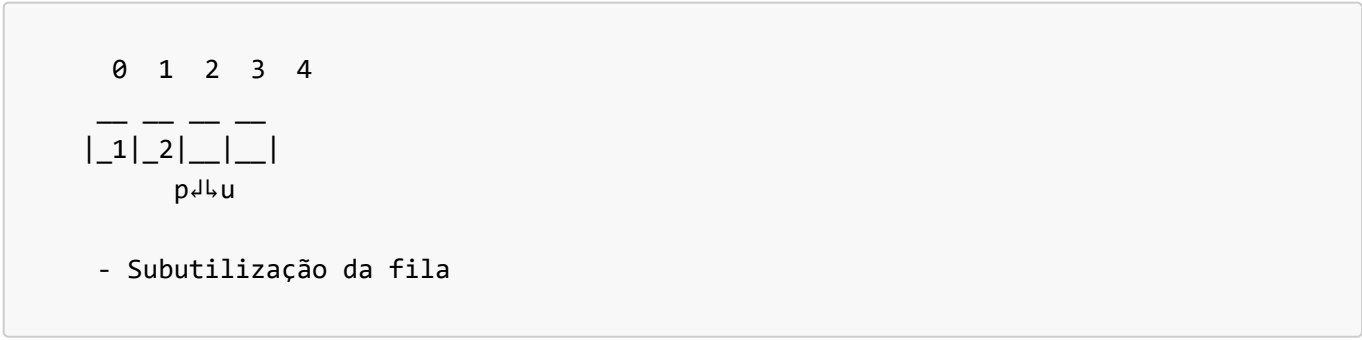


▹ Desenfileira não remove necessariamente o elemento - Muda a posição do **p** e do **u**

PROBLEMAS

- Desperdício de memória

Fila vazia → p==u



Fila cheia → u==N (onde N é o tamanho do vetor)

```
  - 2x enfileira
  - 1x desenfileira
  - 1x enfileira

    0  1  2  3
   __ __ __ __     enfileira __ __ __ __    desenfileira  __ __ __ __   enfileira
__ __ __ __
  |_1|_2|_3|_4|  →    2x    |_1|_2|_3|_3| →     1x       |_1|_2|_3|_4| →   2x
|_1|_2|_3|_4|
  p↲↳u                      p↲     ↳u                      p↲  ↳u
p↲       ↳u
```

Fila circular - Solução da fila cheia

- Quando u chegar ao final, voltarmos ao começo do vetor caso tenha espaço

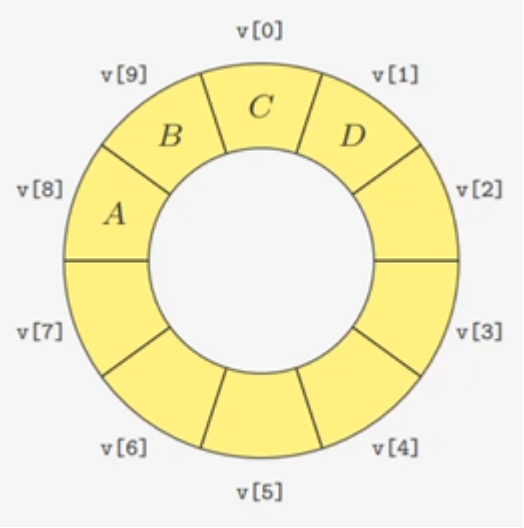
```
    0  1  2  3  4  5  6  7  8  9               0  1  2  3  4  5  6  7  8
9
   __ __ __ __ __ __ __ __ __ __    Enfileira   __ __ __ __ __ __ __ __ __
__   Desenfileira
  |_M|_N|_O|_P|_Q|_R|_S|_T|__|__|   →  8x    →
|_M|_N|_O|_P|_Q|_R|_S|_T|__|__|   →  8x
  p↲↳u                                    p↲                      ↳u

      0  1  2  3  4  5  6  7  8  9               0  1  2  3  4  5  6  7
8  9
     __ __ __ __ __ __ __ __ __ __     Enfileira  __ __ __ __ __ __ __ __
__ __     Enfileira
   → |_M|_N|_O|_P|_Q|_R|_S|_T|__|__|  →  (A)   →
|_M|_N|_O|_P|_Q|_R|_S|_T|_A|__|  →   (B)
                          p↲↳u                                    p↲
↳u


      0  1  2  3  4  5  6  7  8  9               0  1  2  3  4  5  6  7
8  9
     __ __ __ __ __ __ __ __ __ __     Enfileira  __ __ __ __ __ __ __ __
__ __     Enfileira
  →  |_M|_N|_O|_P|_Q|_R|_S|_T|_A|_B|  →  (C)   →
|_C|_N|_O|_P|_Q|_R|_S|_T|_A|_B|  →   (D)
     ↳u                   p↲                      ↳u                 p↲

      0  1  2  3  4  5  6  7  8  9               0  1  2  3  4  5  6  7
8  9

     __ __ __ __ __ __ __ __ __ __
  →  |_C|_D|_O|_P|_Q|_R|_S|_T|_A|_B|
           ↳u              p↲
```



Como representar?

OBS: A última posição da fila não pode ser utilizada

- Fila vazia → **p==u**
- Fila cheia
  - **p==0** e ~~**u==N**~~ **u==N-1**

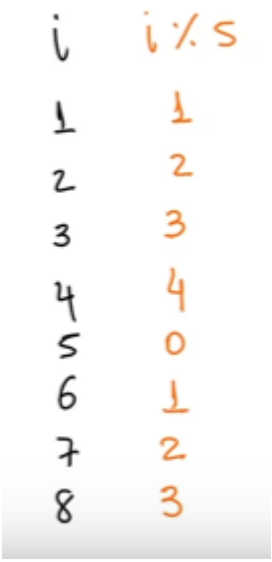
```
                        0  1  2  3                  0  1  2  3
                       __ __ __ __                __ __ __ __
~~4x enfileira:~~     |_1|_2|_3|_4|     ☒   →   |_1|_2|_3|▨|
                      p↲           ↳u            p↲        ↳u
```

  - u == p-1

```
  0  1  2  3                              0  1  2  3
 __ __ __ __                            __ __ __ __
|_2|_3|__|_1|   → Enfilera (4) ☒   →   |_2|_3|▨|_1|
   ↳u    p↲                                 u↲  ↳p
```

Para evitar erros:

1. Fazer 2 if's
2. Usar as condições:
   - P==0 e u==N-1 **ou** u==p-1
   - p=0 e u+1=N **ou** u+1=p
3. União das duas condições: (u+1)%N==p ---> % = resto da divisão Comportamento circular -> USO DO MÓDULO %



Implementação VETOR

Não é adequado, da muito trabalho

```
typedef struct {
	int *dado;
	int N; // tamanho
	int p; // inicio
	int u; // ultimo
} fila;
```

1. Criação

```
fila *cria_fila(){
	fila f=malloc(sizeof(fila));
	f->N=10;
	f-> dado = malloc(f->N*sizeof(int);
	f->p = f->u=0; //fila vazia
	return f;
}
```

2. Inserção - Pode falhar se a fila estive cheia ---> CUSTA O(n) ou O(n)/2

```
int enfileira(fila *f, int x){
	if((f->u+1)%f->N==f->p){ // (u+1)%N==p --> se a filha esta cheia...
		if (!redimensiona(f)) return 0;
	f->dado[f->u]=x;

	// f->u++; //u incrementado
	// if(f->u==N) f->u=0;

	f->u=(f->u+1)%f->N; // substitui as duas linhas comentadas
```

```
        return 1;         // Quando f1->u+1 der igual a N (último esta na última
posição e tem que ficar vazio) vai dar resto 0 e vai significar que o vetor esta
cheio, então f->u=0.
        }
}
```

3. Remoção

```
int desenfileira(fila *f, int *y){
        if(f->p==f->u) return 0; //se a fila é vazia return 0, pq da errado
        *y=f->dado[f->p]; // y é o 1º da fila
        f->p=(f->p+1)%f->N; //p+1 pq quando chefa ao final quer q volter ao incio

}
```

4. Utilização: (main)

```
fila *f = cria_fila();
enfileira(f,1);
enfileira(f,2);
enfileira(f,3);
int y;
desenfileira(f,&y); //y=1
...
free(f->dado);
free(f);
```