

Laboratório de Programação II

Pilha e Fila

Universidade Federal de Juiz de Fora
Departamento de Ciência da Computação

Aula de Hoje

- ▶ TAD Pilha
 - ▶ Implementação com lista simplesmente encadeada
- ▶ TAD Fila
 - ▶ Implementação com lista simplesmente encadeada

TAD Nó

- ▶ Vamos trabalhar com representações de Pilha e Fila com encadeamento dos nós. O TAD Nó já foi apresentado para listas encadeadas, mas vamos revisá-lo.

```
class No
{
    public:
        No ()                {  };
        ~No ()               {  };
        int getInfo ()        { return info; };
        No* getProx ()        { return prox; };
        void setInfo(int val) { info = val; };
        void setProx(No *p)   { prox = p; };

    private:
        int info;             // informacao do No
        No* prox;             // ponteiro para o proximo No
};
```

Pilha Encadeada

```
class PilhaEncad
{
    public:
        PilhaEncad();
        ~PilhaEncad();
        int getTopo();
        void empilha(int val); // insere No no topo
        int desempilha();      // elimina No do topo
        bool vazia();

    private:
        No* topo; // ponteiro para o No do topo
};
```

Fila Encadeada

```
class FilaEncad
{
    public:
        FilaEncad();
        ~FilaEncad();
        int getInicio();
        void enqueue(int val); // insere No no fim
        int dequeue();        // elimina No do inicio
        bool vazia();

    private:
        No *inicio; // ponteiro para No do inicio
        No *final;  // ponteiro para No do fim
};
```

Exercícios

1. Desenvolver a função `imprime()`, tanto para o TAD Pilha quanto para o TAD Fila, a qual deve mostrar os elementos da pilha do topo para o final e da fila do início para o fim.

```
void PilhaEncad::imprime();  
void FilaEncad::imprime();
```

2. Modifique o TAD Pilha Encadeada de forma que este guarde o **número de itens que estão na Pilha**. Implemente uma operação `tamanho()` que retorna quantos itens estão armazenados na pilha.

```
int PilhaEncad::tamanho();
```

Exercícios

3. Implementar uma função `inverte()` que, dados um vetor `vet` com n números inteiros, cria e retorna um novo vetor com os elementos de `vet` na **ordem inversa**. O novo vetor deve ser alocado de forma dinâmica. A função `inverte()` deve ser implementada e testada no programa principal (`main.cpp`).

Protótipo:

```
int* inverte(int *vet, int n);
```

Exercícios

4. Considere uma fila F não vazia. Utilizando as operações de fila (vazia, enfileira e desenfileira), uma pilha de forma auxiliar com suas operações (vazia, empilha e desempilha) e uma variável auxiliar do tipo `int`, escreva uma função que **inverte a ordem** dos elementos da fila.

```
void inverteFila(FilaEncad *f);
```

5. Desenvolver uma função para, dadas duas filas F1 e F2, criar e retornar uma nova fila que representa a **concatenação destas duas filas** (sem repetição de valores). Considere que as filas F1 e F2 não possuem valores repetidos e tanto F1 quanto F2 tornam-se vazias após a concatenação.

```
FilaEncad* concatena(FilaEncad *f1,  
                    FilaEncad *f2);
```


Exercícios

6. Utilizando apenas as operações de manipulação de pilhas (vazia, empilha, desempilha, getTopo), uma pilha auxiliar pAux e uma variável do tipo `int`, escrever uma função para **remover um item com valor x** de uma posição qualquer da pilha.

Exemplo: Pilha (9, 2, 5, 6, 1) Topo = 9

Remove(5)

Saída: Pilha (9, 2, 6, 1) Topo = 9

```
void removeDaPilha(PilhaEncad *p, int x);
```