Algoritmos
e
Estrutura de Dados

Profª. Angela Abreu Rosa de Sá, Drª.

Contato: angelaabreu@gmail.com

Lembrar ...

É um processo INDIVIDUAL!

Depende apenas de você!

IMPORTANTE

- Não se compare com o colega;
- Pergunte QUALQUER dúvida;
- Implemente os exercícios propostos;
- Pratique sempre...inclusive em casa!
- Não desista!

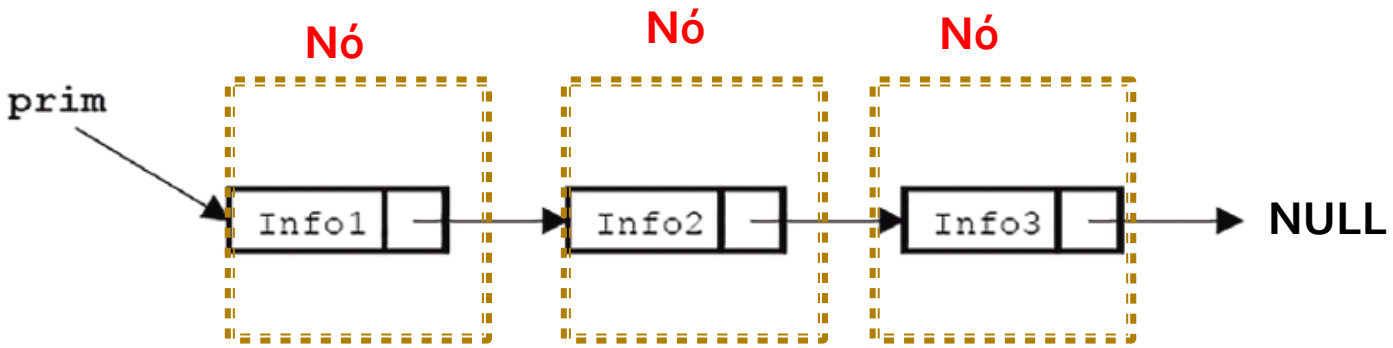
Ementa


1
Listas Ligadas
Definição
Operações
Listas duplamente ligadas
2
Pilhas e filas
Definição
Operações com pilhas
Operações com filas
3
Tabelas de espalhamento
Definição
Operações
Otimização
4
Armazenamento associativo
Definição
Mapas com lista
Mapas com espalhamento

Algoritmos e Estrutura de Dados

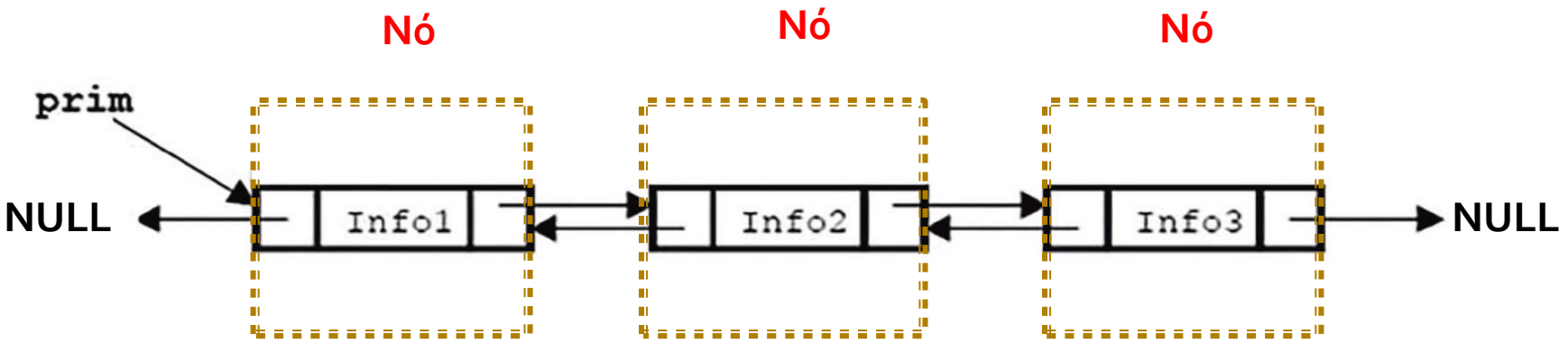
Sumário

Lista Encadeada




```
struct noLista
{
   int informacao;
   struct noLista *proximo;
};
```

Lista Duplamente Encadeada




```
struct noLista
{
   struct noLista *NoAnterior;
   int informacao;
   struct noLista *proximoNo;
};
```

Fila

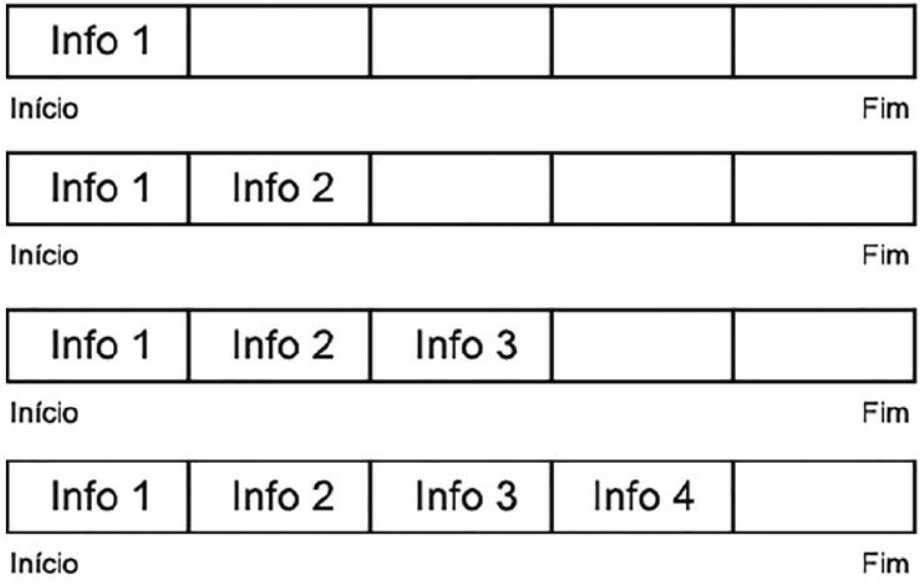
Fila é a representação de um conjunto de elementos no qual podemos *remover esses elementos* por uma extremidade chamada de inicio da fila. Já a outra extremidade, onde são inseridos os elementos, e conhecida como final da fila.

Figura 2.7 | Fila em guichê de aeroporto

Fila

Inserindo elementos na Fila

Figura 2.8 | Entrada de elemento na fila pelo seu final.

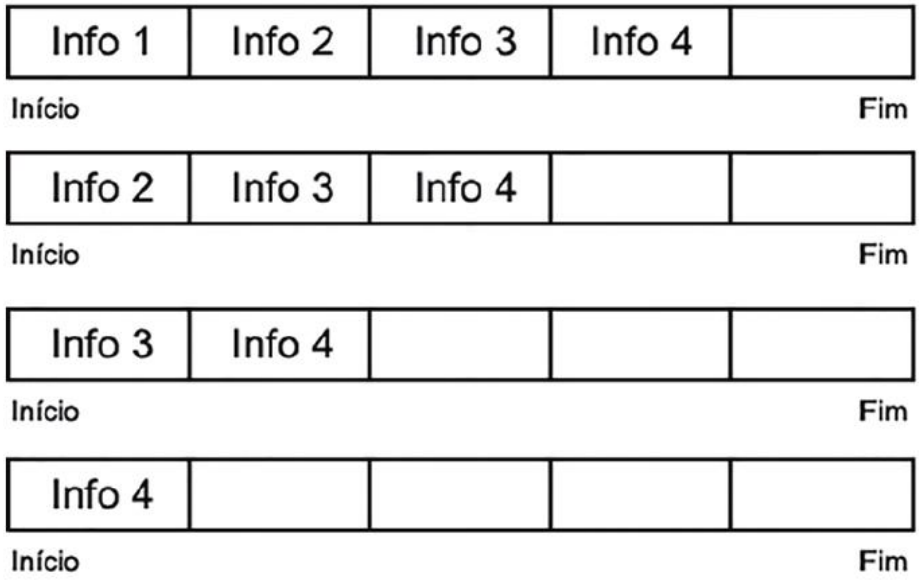


FIFO – ***First In, First Out*** **(o primeiro a entrar, é o primeiro a sair)**

Fila

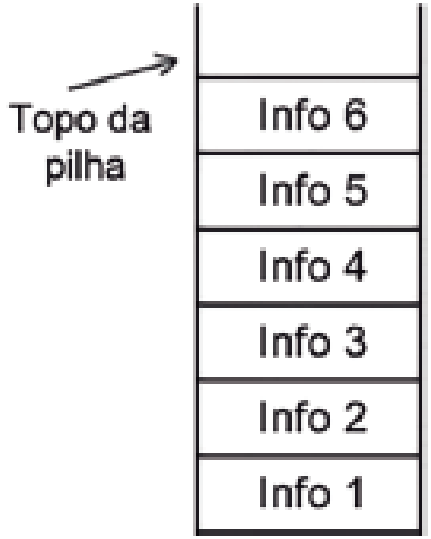
Removendo elementos da Fila

Figura 2.9 | Saída de elemento pelo início da fila



Pilha

Uma pilha tem como definição básica um conjunto de elementos que permite a **inserção e a remoção** de elementos em apenas uma das extremidades da estrutura denominada topo da pilha



Pilha

Os elementos inseridos em uma pilha possuem uma sequencia de Inserção: O **primeiro elemento que entra** na pilha só pode ser **removido por ultimo**, após todos os outros elementos serem removidos.

LIFO – *Last In, First Out* **(o último a entrar, é o primeiro a sair)**
FILO – *First In, Last Out* **(o primeiro a entrar, é o último a sair)**

Pilha

Inserindo elementos na Pilha

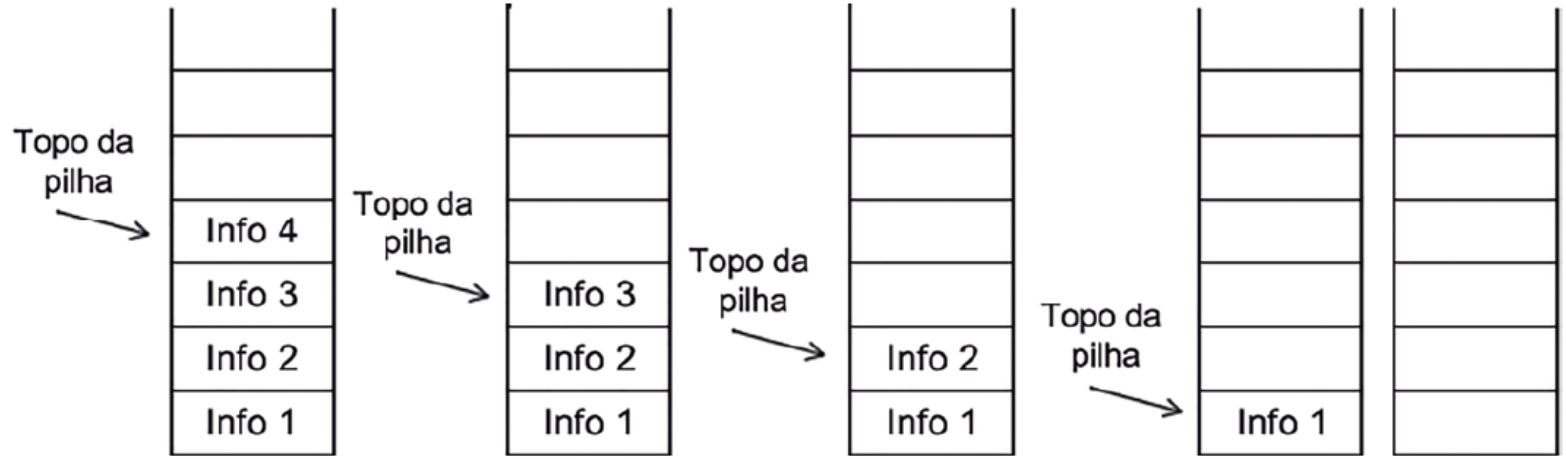
Figura 2.4 | Inserindo elemento na pilha

Fonte: elaborada pelo autor

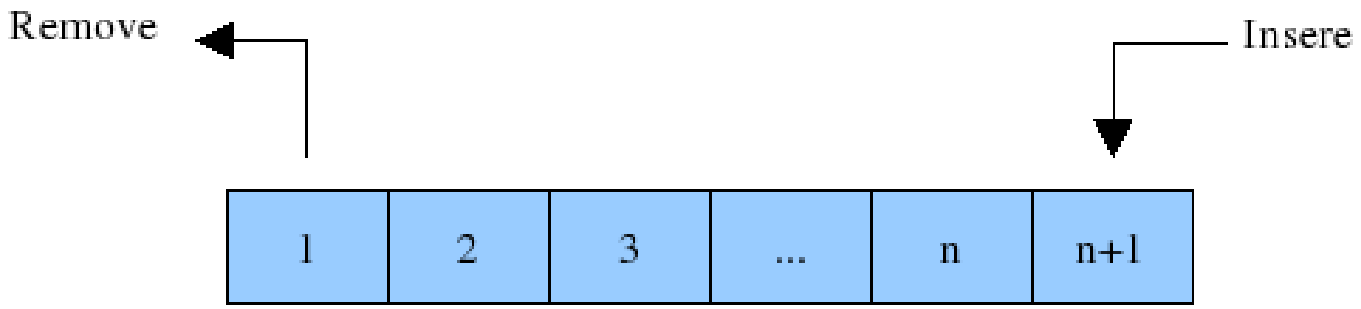
Pilha

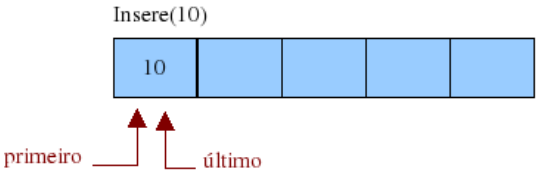
Removendo elementos da Pilha

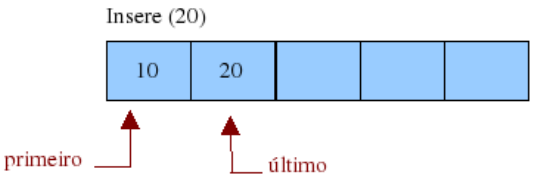
Figura 2.5 | Removendo um elemento da pilha

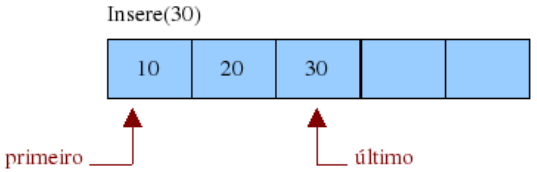


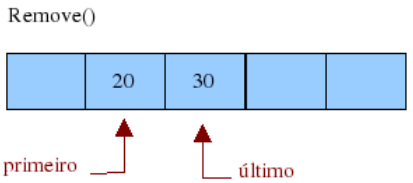
Fila em Vetor Estático

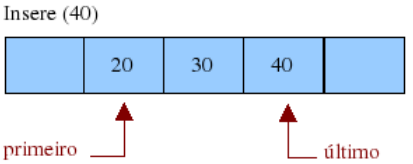


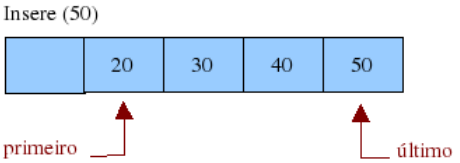




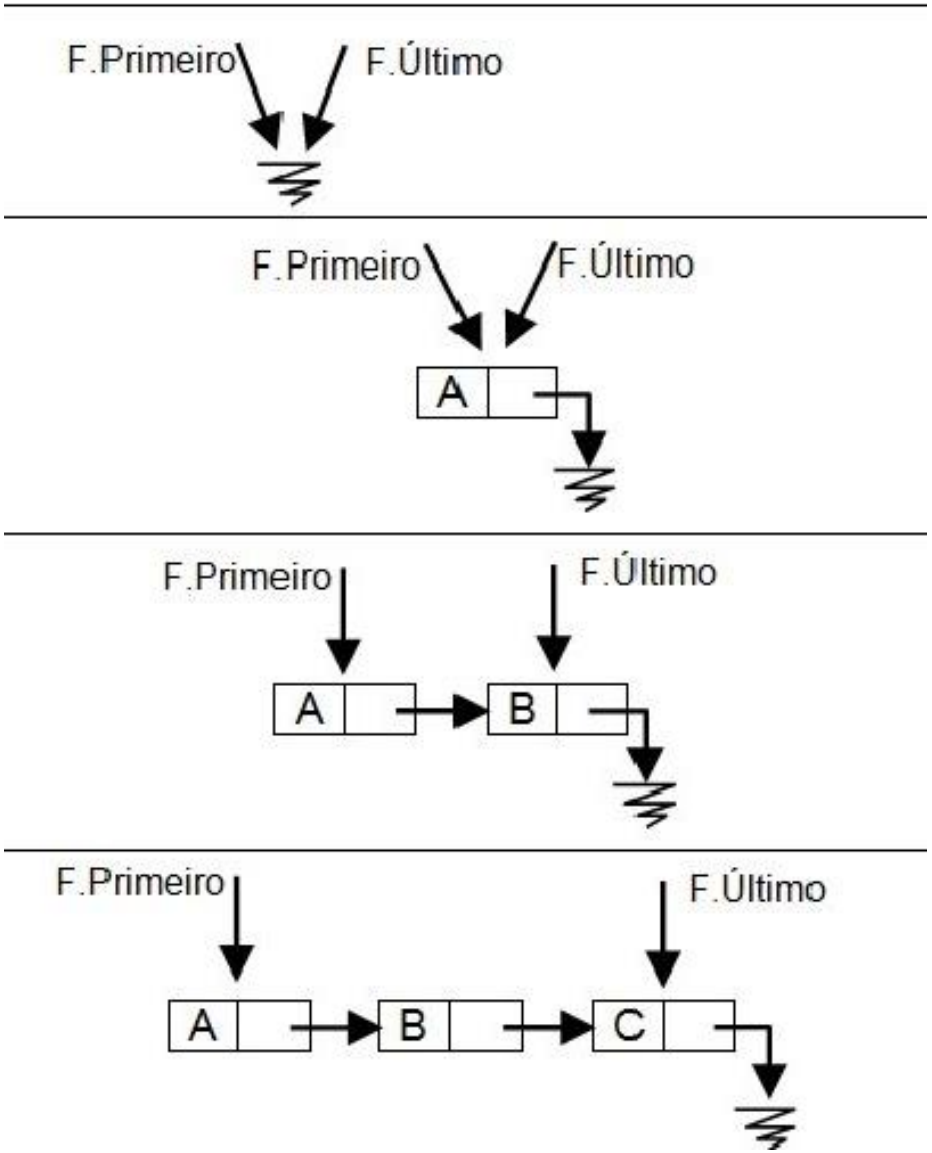








Fila em Lista Encadeada (DINÂMICA)



Operações com FILAS

- **Inserir elementos na Fila** insere sempre no final

- **Remover elemento da Fila** remove sempre do início

- **Saber quantos elementos tem na Fila**

Definição da estrutura da FILA

```c
#include <stdio.h>
#include <stdlib.h>

struct noFila
{
  struct noFila *NoAnterior;
  int informacao;
  struct noFila *proximoNo;
};
```

Inserir na FILA

```c
struct noFila* InserirNaFila(struct noFila* InicioFILA, int NovoNumero)
{
    struct noFila *PercorreFila = InicioFILA; //cópia do início da fila
    struct noFila* novoNo = (struct noFila*)malloc(sizeof(struct noFila)); // alocar memória para o novo nó

    if(PercorreFila != NULL) //se a fila não estiver vazia
    {

        while (PercorreFila->proximoNo != NULL) //percorrer a fila até encontrar o último nó
        {
          PercorreFila = PercorreFila->proximoNo;

        }

        //o que era o último nó, agora será o penúltimo: aponta para o novoNo
        PercorreFila->proximoNo = novoNo; // o que era o último nó (agora será o penúltimo!) aponta para o novo nó
   }

    novoNo->NoAnterior = PercorreFila; //apontar o campo NoAnterior para o nó PercorreFila
    novoNo->informacao = NovoNumero; //inserir a nova informação
    novoNo->proximoNo = NULL; //o novoNo será o último da fila

    if(InicioFILA == NULL)//a fila estava vazia
        return novoNo;
    else return InicioFILA;
}
```

Remover da FILA

```
struct noFila* RemoveDaFila (struct noFila* InicioFILA)
{
    struct noFila *PercorreFila = InicioFILA; //cópia do início da Fila

    if(PercorreFila == NULL)
        return InicioFILA;

        //Remover o primeiro da  Fila
        InicioFILA = PercorreFila->proximoNo; //atualizar o iniício da lista
        InicioFILA->NoAnterior = NULL; //o primiero da fila aponta para null

        free(PercorreFila); //liberar região de memória do nó removido

    return InicioFILA;

}
```

Contar elementos da FILA

```c
int ContarElementosFila(struct noFila  *InicioFILA)
{
    struct noFila *NoAtual = InicioFILA; //copiar o endereço do primeiro nó da fila
    int contador = 0;

    while (NoAtual != NULL) //percorrer a fila até encontrar o último nó = NULL
    {
        contador++; //incrementar o contador de elementos
        NoAtual = NoAtual->proximoNo; //apotar para o próximo nó da fila
    }

    return contador;

}
```

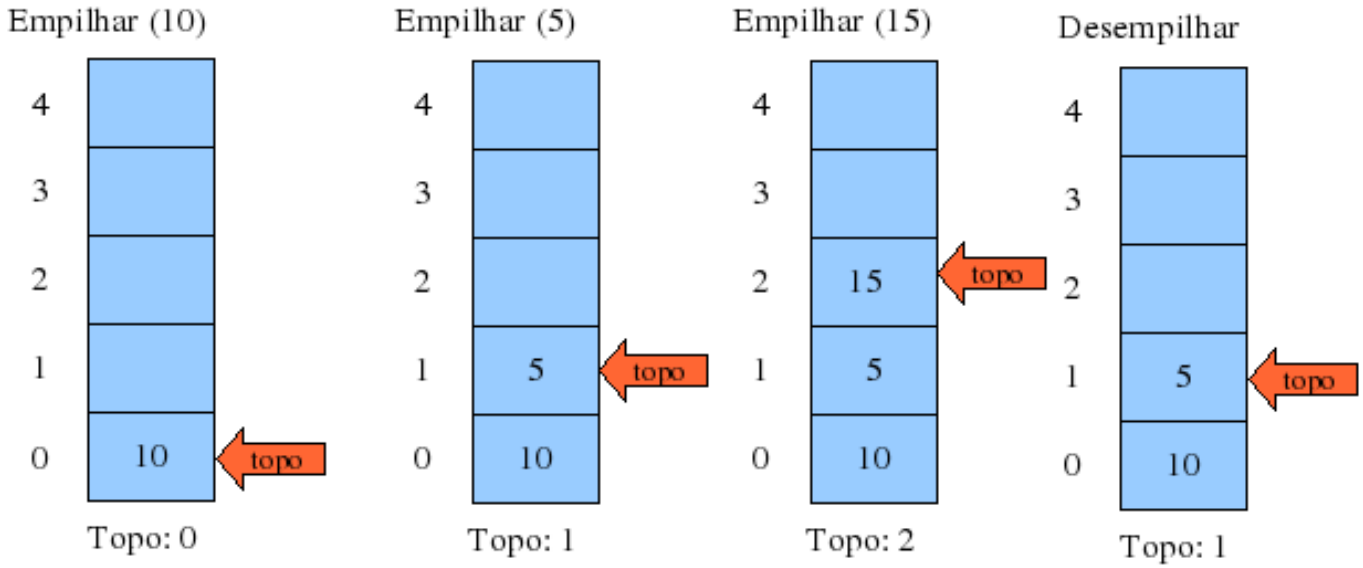
Imprimir FILA

```
void ImprimirFila(struct noFila  *InicioFILA)
{
    struct noFila *NoAtual = InicioFILA; //copiar o endereço do primeiro nó da fila

    while (NoAtual != NULL) //percorrer a fila até encontrar o último nó = NULL
    {
        printf("%d -> ",NoAtual->informacao); //mostrar a informação do nó
        NoAtual = NoAtual->proximoNo; //apotar para o próximo nó da fila
    }

    printf("NULL");

}
```

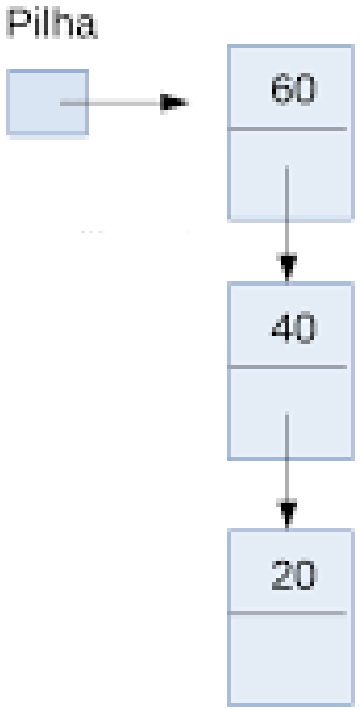
```c
int main()
{
    struct noFila *InicioFILA = NULL;
    printf("*** INSERIR NUMEROS NA FILA **** \n");
    int numero;
    for(int i = 0; i < 5; i++)
    {
        printf("\n -- Digite um numero:");
        scanf("%d", &numero);

        InicioFILA = InserirNaFila (InicioFILA, numero);
    }

    printf("\n\n *** FILA GERADA **** \n");
    //imprimir o conteúdo da fila
   ImprimirFila(InicioFILA);
   int qtde = ContarElementosFila(InicioFILA);
   printf("\n\n -- Quantidade de elementos da Fila: %d",qtde);
  printf("\n\n *** REMOVENDO 2 NUMEROS DA FILA **** \n");
  InicioFILA = RemoveDaFila (InicioFILA);
  InicioFILA = RemoveDaFila (InicioFILA);
  printf("\n\n *** FILA GERADA **** \n");
  //imprimir o conteúdo da fila
   ImprimirFila(InicioFILA);
   qtde = ContarElementosFila(InicioFILA);
   printf("\n\n -- Quantidade de elementos da Fila: %d",qtde);
}
```

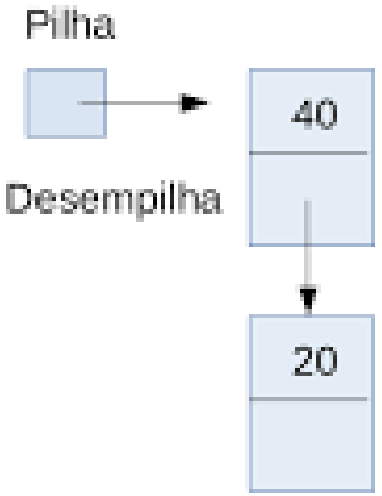
```
*** INSERIR NUMEROS NA FILA ****

 -- Digite um numero:1

 -- Digite um numero:2

 -- Digite um numero:3

 -- Digite um numero:4

 -- Digite um numero:5


 *** FILA GERADA ****
1 -> 2 -> 3 -> 4 -> 5 -> NULL

 -- Quantidade de elementos da Fila: 5

 *** REMOVENDO 2 NUMEROS DA FILA ****


 *** FILA GERADA ****
3 -> 4 -> 5 -> NULL

 -- Quantidade de elementos da Fila: 3
--------------------------------
```

Pilha em Vetor Estático



Pilha em Lista Encadeada (Dinâmica)





Operações com PILHAS

- **EMPILHAR (PUSH)** → **Inserir elementos na Pilha -** insere sempre no início

- **DESEMPILHAR (POP)** → **Remover elemento da Pilha -** remove sempre do início

- **Saber quantos elementos tem na Pilha**

Definição da estrutura da PILHA

```c
#include <stdio.h>
#include <stdlib.h>

struct noPILHA
{
   struct noPILHA *NoAnterior;
   int informacao;
   struct noPILHA *proximoNo;
};
```

EMPILHAR

```c
struct noPILHA* Empilhar(struct noPILHA* InicioPILHA, int NovoNumero)
{

    //alocar memória para o novo Nó da pilha
    struct noPILHA* novoNo = (struct noPILHA*) malloc(sizeof(struct noPILHA));

    //O campo NoAnterior será NULL, pois agora ele será o primeiro da pilha
    novoNo->NoAnterior = NULL;
    //Inserir as informação para o novo nó
    novoNo->informacao = NovoNumero;
    //Apontar o campo "próximo" do novo nó para o local que o InicioDaPilha apontava
    novoNo->proximoNo = InicioPILHA;


    if(InicioPILHA != NULL) // se a pilha não estiver vazia, ligar o nó anterior do início da pilha ao novo nó
       InicioPILHA->NoAnterior = novoNo;


    return novoNo;
}
```

DESEMPILHAR

```
struct noPILHA* Desempilhar (struct noPILHA* topoPILHA)
{
    struct noPILHA* PercorrePilha = topoPILHA; //cópia do início da pilha

    if (PercorrePilha == NULL ) //não tem elemento para ser removido
        return topoPILHA;

    //Remover o primeiro da Fila
    topoPILHA = PercorrePilha->proximoNo; //atualizar o início da pilha
    topoPILHA->NoAnterior = NULL; // o primeiro da pilha  aponta para NULL


    free(PercorrePilha); //liberar região de memória do nó removido

  return topoPILHA;
}
```

Contar elementos da PILHA

```c
int ContarElementosPilha(struct noPILHA  *InicioFILA)
{
    struct noPILHA *NoAtual = InicioFILA; //copiar o endereço do primeiro nó da pilha
    int contador = 0;

    while (NoAtual != NULL) //percorrer a pilha até encontrar o último nó = NULL
    {
        contador++; //incrementar o contador de elementos
        NoAtual = NoAtual->proximoNo; //apotar para o próximo nó da pilha
    }

    return contador;

}
```

Imprimir PILHA

```c
void ImprimirPilha(struct noPILHA  *topoPILHA)
{
    struct noPILHA *NoAtual = topoPILHA; //copiar o endereço do primeiro nó da pilha

    while (NoAtual != NULL) //percorrer a pilha até encontrar o último nó = NULL
    {
        printf("\n %d  ",NoAtual->informacao); //mostrar a informação do nó
        NoAtual = NoAtual->proximoNo; //apotar para o próximo nó da pilha
    }

    printf("\n NULL");

}
```

```c
int main()
{   struct noPILHA *topoPILHA = NULL;
    printf("*** INSERIR NUMEROS NA PILHA **** \n");
    int numero;
    for(int i = 0; i < 5; i++)
    {
        printf("\n -- Digite um numero:");
        scanf("%d", &numero);

        topoPILHA = Empilhar (topoPILHA, numero);
    }

    printf("\n\n *** PILHA GERADA **** \n");
    //imprimir o conteúdo da pilha
   ImprimirPilha(topoPILHA);
   int qtde = ContarElementosPilha(topoPILHA);
   printf("\n\n -- Quantidade de elmentos da PILHA: %d",qtde);
   printf("\n\n *** REMOVENDO 2 NUMEROS DA PILHA **** \n");
   topoPILHA = Desempilhar (topoPILHA);
   topoPILHA = Desempilhar (topoPILHA);
   printf("\n\n *** PILHA GERADA **** \n");
   //imprimir o conteúdo da pilha
   ImprimirPilha(topoPILHA);

   qtde = ContarElementosPilha(topoPILHA);
   printf("\n\n -- Quantidade de elmentos da PILHA: %d",qtde);

}
```

```
*** INSERIR NUMEROS NA PILHA ****

-- Digite um numero:1

-- Digite um numero:2

-- Digite um numero:3

-- Digite um numero:4

-- Digite um numero:5


*** PILHA GERADA ****

5
4
3
2
1
NULL

-- Quantidade de elmentos da PILHA: 5

*** REMOVENDO 2 NUMEROS DA PILHA ****


*** PILHA GERADA ****

3
2
1
NULL

-- Quantidade de elmentos da PILHA: 3
```

Programa completo... FILA

Definição da estrutura da FILA

```c
#include <stdio.h>
#include <stdlib.h>

struct noFila
{
  struct noFila *NoAnterior;
  int informacao;
  struct noFila *proximoNo;
};
```

Inserir na FILA

```
struct noFila* InserirNaFila(struct noFila* InicioFILA, int NovoNumero)
{
    struct noFila *PercorreLista = InicioFILA; //cópia do início da lista
    struct noFila* novoNo = (struct noFila*)malloc(sizeof(struct noFila)); // alocar memória para o novo nó

    if(PercorreLista != NULL) //se a lista não estiver vazia
    {

        while (PercorreLista->proximoNo != NULL) //percorrer a lista até encontrar o último nó
        {
          PercorreLista = PercorreLista->proximoNo;

        }

        //o que era o último nó, agora será o penúltimo: aponta para o novoNo
        PercorreLista->proximoNo = novoNo; // o que era o último nó (agora será o penúltimo!) aponta para o novo nó
   }

    novoNo->NoAnterior = PercorreLista; //apontar o campo NoAnterior para o nó PercorreLista
    novoNo->informacao = NovoNumero; //inserir a nova informação
    novoNo->proximoNo = NULL; //o novoNo será o último da lista

    if(InicioFILA == NULL)//a lista estava vazia
        return novoNo;
    else return InicioFILA;
}
```

Remover da FILA

```
struct noFila* RemoveDaFila (struct noFila* InicioFILA)
{
    struct noFila* PercorreLista = InicioFILA; //cópia do início da lista

    if (PercorreLista == NULL ) //não tem elemento para ser removido
        return InicioFILA;

    //Remover o primeiro da Fila
    InicioFILA = PercorreLista->proximoNo; //atualizar o início da lista
    InicioFILA->NoAnterior = NULL; // o primeiro da lista  aponta para NULL


    free(PercorreLista); //liberar região de memória do nó removido

  return InicioFILA;
}
```

Contar elementos da FILA

```c
int ContarElementosFila(struct noFila  *InicioFILA)
{
    struct noFila *NoAtual = InicioFILA; //copiar o endereço do primeiro nó da lista
    int contador = 0;

    while (NoAtual != NULL) //percorrer a lista até encontrar o último nó = NULL
    {
        contador++; //incrementar o contador de elementos
        NoAtual = NoAtual->proximoNo; //apotar para o próximo nó da lista
    }

    return contador;

}
```

Imprimir FILA

```
void ImprimirFila(struct noFila  *InicioFILA)
{
    struct noFila *NoAtual = InicioFILA; //copiar o endereço do primeiro nó da lista

    while (NoAtual != NULL) //percorrer a lista até encontrar o último nó = NULL
    {
        printf("%d -> ",NoAtual->informacao); //mostrar a informação do nó
        NoAtual = NoAtual->proximoNo; //apotar para o próximo nó da lista
    }

    printf("NULL");

}
```

```c
int main()
{
    struct noFila *InicioFILA = NULL;
    printf("*** INSERIR NUMEROS NA FILA **** \n");
    int numero;
    for(int i = 0; i < 5; i++)
    {
        printf("\n -- Digite um numero:");
        scanf("%d", &numero);

        InicioFILA = InserirNaFila (InicioFILA, numero);
    }

    printf("\n\n *** FILA GERADA **** \n");
    //imprimir o conteúdo da fila
   ImprimirFila(InicioFILA);
   int qtde = ContarElementosFila(InicioFILA);
   printf("\n\n -- Quantidade de elementos da Fila: %d",qtde);
  printf("\n\n *** REMOVENDO 2 NUMEROS DA FILA **** \n");
  InicioFILA = RemoveDaFila (InicioFILA);
  InicioFILA = RemoveDaFila (InicioFILA);
  printf("\n\n *** FILA GERADA **** \n");
  //imprimir o conteúdo da fila
   ImprimirFila(InicioFILA);
   qtde = ContarElementosFila(InicioFILA);
   printf("\n\n -- Quantidade de elementos da Fila: %d",qtde);
}
```

Programa completo... PILHA

```c
#include <stdio.h>
#include <stdlib.h>

struct noPILHA
{
   struct noPILHA *NoAnterior;
   int informacao;
   struct noPILHA *proximoNo;
};
```

```c
struct noPILHA* Empilhar(struct noPILHA* InicioPILHA, int NovoNumero)
{

    //alocar memória para o novo Nó da Lista
    struct noPILHA* novoNo = (struct noPILHA*) malloc(sizeof(struct noPILHA));

    //O campo NoAnterior será NULL, pois agora ele será o primeiro da lista
    novoNo->NoAnterior = NULL;
    //Inserir as informação para o novo nó
    novoNo->informacao = NovoNumero;
    //Apontar o campo "próximo" do novo nó para o local que o InicioDaLista apontava
    novoNo->proximoNo = InicioPILHA;


    if(InicioPILHA != NULL) // se a lista não estiver vazia, ligar o nó anterior do início da lista ao novo nó
       InicioPILHA->NoAnterior = novoNo;


    return novoNo;
}
```

```c
struct noPILHA* Desempilhar (struct noPILHA* topoPILHA)
{
    struct noPILHA* PercorrePilha = topoPILHA; //cópia do início da pilha

    if (PercorrePilha == NULL ) //não tem elemento para ser removido
        return topoPILHA;

    //Remover o primeiro da Fila
    topoPILHA = PercorrePilha->proximoNo; //atualizar o início da pilha
    topoPILHA->NoAnterior = NULL; // o primeiro da pilha  aponta para NULL


    free(PercorrePilha); //liberar região de memória do nó removido

  return topoPILHA;
}
```

```c
int ContarElementosPilha(struct noPILHA  *InicioFILA)
{
    struct noPILHA *NoAtual = InicioFILA; //copiar o endereço do primeiro nó da pilha
    int contador = 0;

    while (NoAtual != NULL) //percorrer a pilha até encontrar o último nó = NULL
    {
        contador++; //incrementar o contador de elementos
        NoAtual = NoAtual->proximoNo; //apotar para o próximo nó da pilha
    }

    return contador;

}
```

```c
void ImprimirPilha(struct noPILHA  *topoPILHA)
{
    struct noPILHA *NoAtual = topoPILHA; //copiar o endereço do primeiro nó da pilha

    while (NoAtual != NULL) //percorrer a pilha até encontrar o último nó = NULL
    {
        printf("\n %d  ",NoAtual->informacao); //mostrar a informação do nó
        NoAtual = NoAtual->proximoNo; //apotar para o próximo nó da pilha
    }

    printf("\n NULL");

}
```

```c
int main()
{   struct noPILHA *topoPILHA = NULL;
    printf("*** INSERIR NUMEROS NA PILHA **** \n");
    int numero;
    for(int i = 0; i < 5; i++)
    {
        printf("\n -- Digite um numero:");
        scanf("%d", &numero);

        topoPILHA = Empilhar (topoPILHA, numero);
    }

    printf("\n\n *** PILHA GERADA **** \n");
    //imprimir o conteúdo da pilha
   ImprimirPilha(topoPILHA);
   int qtde = ContarElementosPilha(topoPILHA);
   printf("\n\n -- Quantidade de elmentos da PILHA: %d",qtde);
   printf("\n\n *** REMOVENDO 2 NUMEROS DA PILHA **** \n");
   topoPILHA = Desempilhar (topoPILHA);
   topoPILHA = Desempilhar (topoPILHA);
   printf("\n\n *** PILHA GERADA **** \n");
   //imprimir o conteúdo da pilha
   ImprimirPilha(topoPILHA);

   qtde = ContarElementosPilha(topoPILHA);
   printf("\n\n -- Quantidade de elmentos da PILHA: %d",qtde);

}
```

Exercícios 😀

FILAS E PILHAS

1- Implemente um programa para inserir 5 números em uma Fila. Imprima a Fila após a inserção de cada elemento.
Obs.: solicite para o usuário digitar um número por vez.

2- Implemente um programa para remover 2 números em uma FILA. Mostre na tela o número que foi removido. E ainda, imprima a Fila antes e após a remoção de cada elemento.

3- Implemente um programa para mostrar a quantidade de elementos em uma FILA.

4- Implemente um programa para inserir 5 números em uma PILHA. Imprima a PILHA após a inserção de cada número. Obs.: solicite para o usuário digitar um número por vez.

5- Implemente um programa para remover 2 números em uma PILHA. Mostre na tela o número que foi removido. E ainda, imprima a Fila antes e após a remoção de cada elemento.

6- Implemente um programa para mostrar a quantidade elementos em uma PILHA.

Muito Obrigada!

Profª. Angela Abreu Rosa de Sá, Drª.

Contato: angelaabreu@gmail.com