Pilhas e Filas Encadeadas

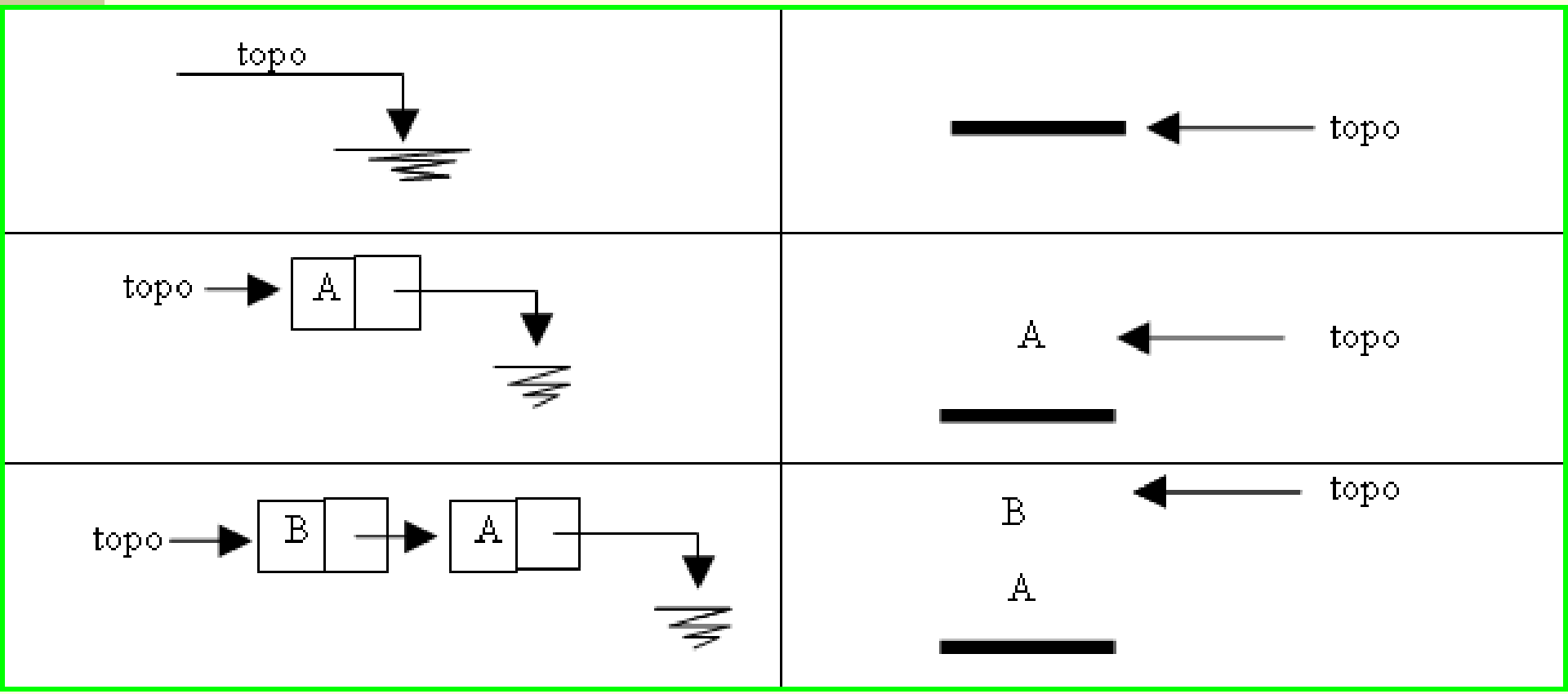
Algoritmos e Estruturas de Dados I

Pilha

- Lista linear: pilha

- Represente graficamente o funcionamento da pilha, representando a pilha vazia, a entrada e a saída de elementos
  - Quais e quantos ponteiros são necessários?

Pilha



Exercício

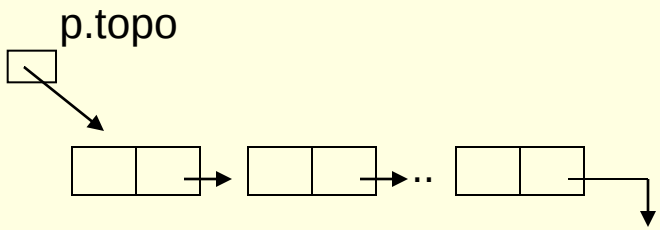
- Implementar as rotinas da *pilha* utilizando a lista encadeada e dinâmica
  - Create, Push, Pop, IsEmpty

Operações – alocação encadeada dinâmica

```
typedef struct elem{
    tipo_info info;
    struct elem *lig;
}tipo_elem;

typedef struct{
    tipo_elem *topo;
}pilha;

pilha p;
```



Obs: p.topo dá o endereço do elemento no topo

1. create (P) - cria uma pilha P vazia

```
void create (pilha *p){
    p->topo = NULL;
}
```

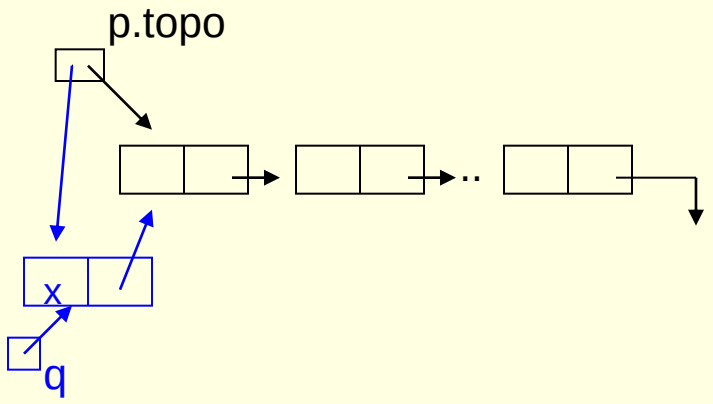
2. insere x no topo de P (empilha): push (x, P)

```
boolean push (pilha *p, tipo_info x){

   tipo_elem *q = malloc(sizeof(tipo_elem));

   if (*q == NULL)
     /*não possui memória disponível*/
      return FALSE;

   q->info = x;
   q->lig = p->topo
   p->topo = q;
   return TRUE;
}
```



3. testa se P está vazia

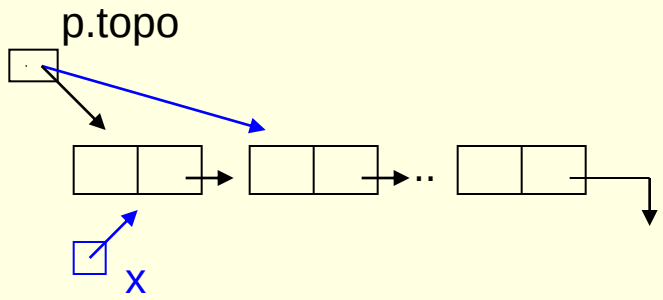
```
boolean isEmpty (pilha *p){
     return (p->topo == NULL);
}
```

4. acessa o elemento do topo da pilha (sem remover) - testar antes da chamada se a pilha não está vazia!!!

```
tipo_elem *topo (pilha *p){
     return p->topo;
}
```

5. Remove e retorna o elemento (todo o registro) eliminado

```
boolean pop(pilha *p, tipo_elem *x){
    if (!IsEmpty(p)){
    *x = p->topo;
    p->topo = p->topo->lig;
    return TRUE;}
    else return FALSE;
}
```



Exercícios

- Implementar as operações dos TADs:
  - pilha estática
  - pilha dinâmica

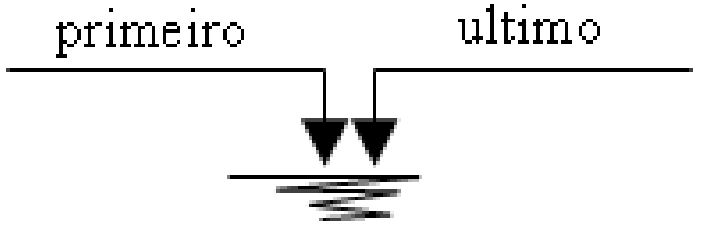
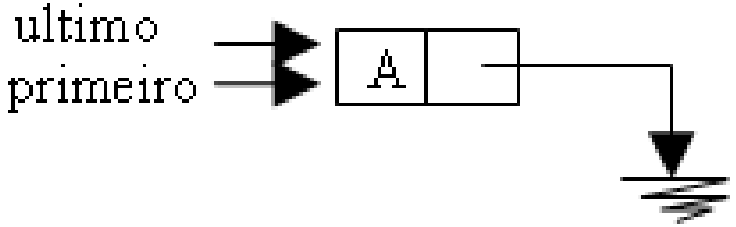
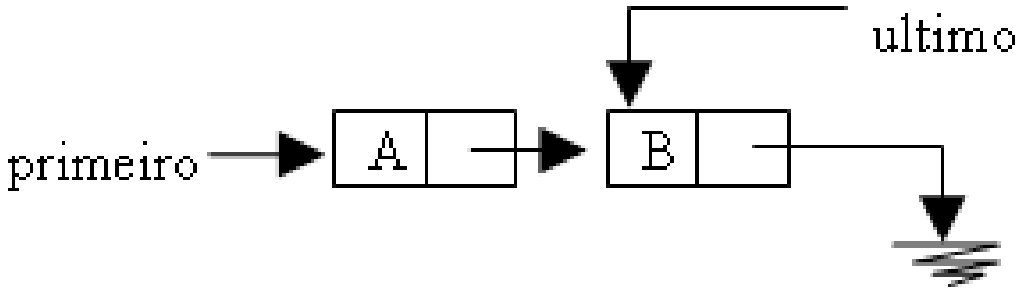
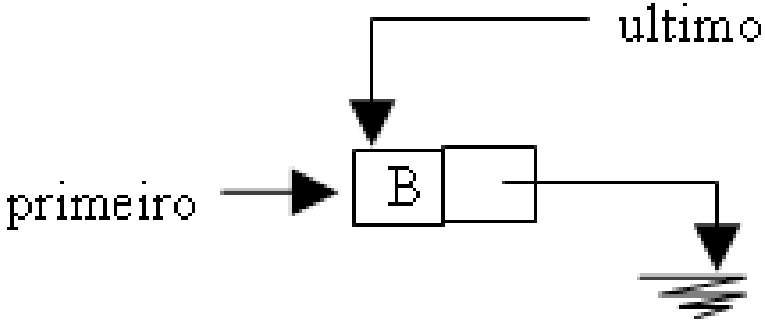
Reflita:

- Há vantagens de se implementar as Pilhas de forma dinâmica? Quais?
- Há desvantagens? Quais?
- Há vantagens em implementá-las estaticamente no array?
- Há desvantagens? Quais?

Fila

- Lista linear: fila

- Represente graficamente o funcionamento da fila, representando a fila vazia, a entrada e a saída de elementos
  - Quais e quantos ponteiros são necessários?

fila vazia	
entra A	
entra B	
sai o primeiro	

Exercício

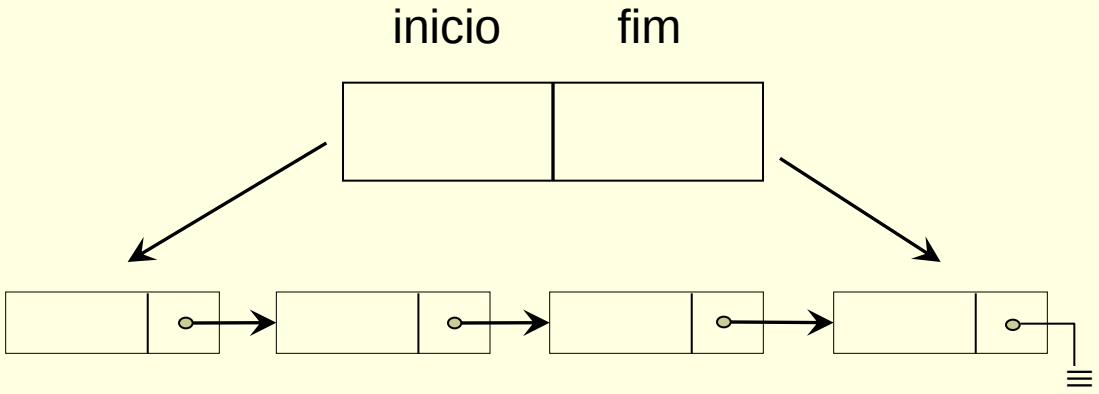
- Implementar as rotinas da *fila* utilizando a lista encadeada e dinâmica
  - Cria, Entra, Sai, IsEmpty, IsFull

Implementações de Filas: Dinâmica

```
#define tipo_info int

typedef struct elem{
    tipo_info info;
    struct elem *lig;
}tipo_elem;

typedef struct{
    tipo_elem *inicio;
    tipo_elem *fim;
}fila;
fila q;
```



```
void create(fila *q){
    /*Cria uma fila vazia. Deve ser usado antes de qualquer
   outra operação*/
    q->inicio = NULL;
    q->fim = NULL;
}
```

```
boolean IsEmpty (fila *q){
    /*Retorna true se fila não contém elementos, false caso
   contrário*/
    return (q->inicio == NULL);
}

void Empty (fila *q){
    /*Reinicializa uma fila existente q como uma fila vazia
   removendo todos os seus elementos.*/
    tipo_elem *ndel, *nextno;

    if(!IsEmpty(q)){
        nextno = q->inicio;
        while (nextno != NULL){
            ndel = nextno;
            nextno = nextno->lig;
            free(ndel);
        }
    }
    create(q);
}
```

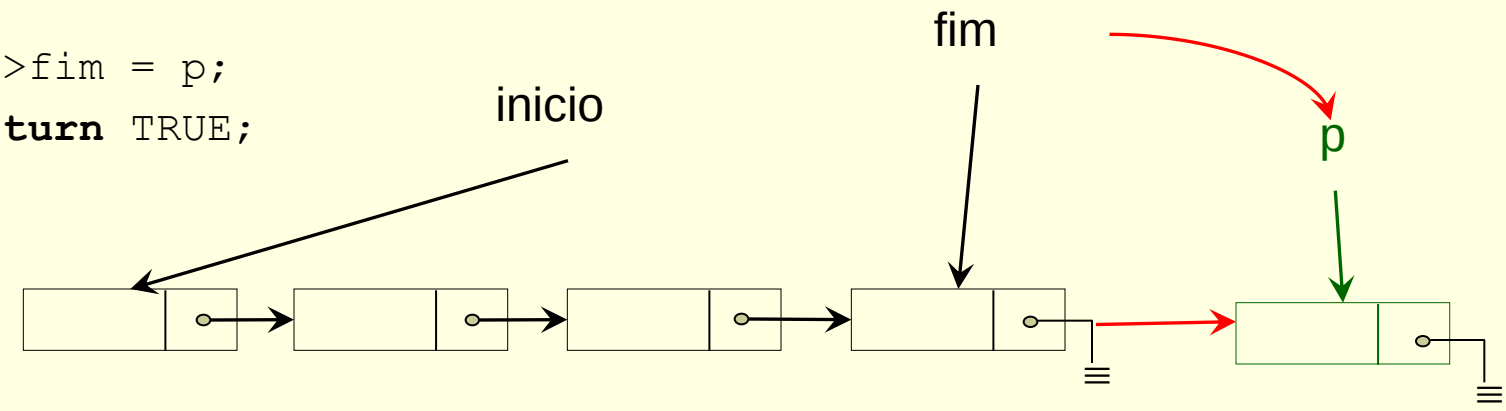
```
boolean Entra(fila *q, tipo_info info){
    /*Adiciona um item no fim da fila q. Retorna true se
   operação realizada com sucesso, false caso contrário*/
    tipo_elem *p;

    p = malloc(sizeof(tipo_elem));
    if (p == NULL)
        return FALSE;

    p->info = info;
    p->lig = NULL;

    if (IsEmpty(q))
        q->inicio = p;
    else
        q->fim->lig = p;

    q->fim = p;
    return TRUE;
}
```



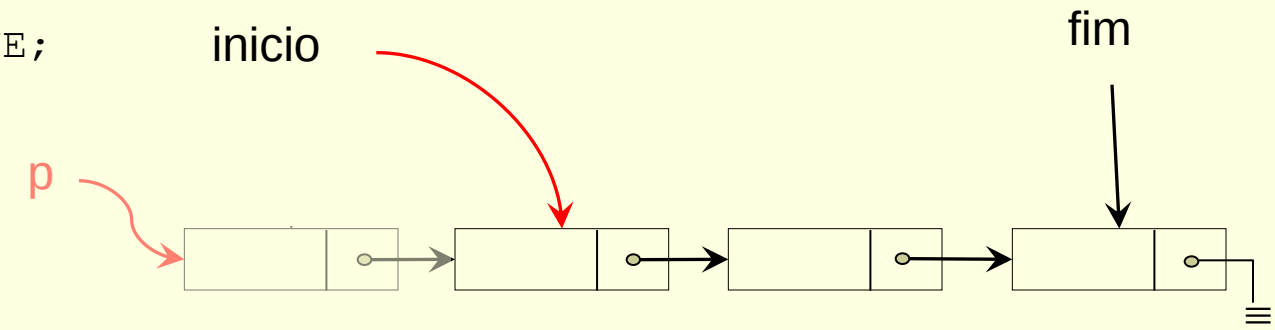
```
boolean Sai(fila *q, tipo_info *info){
    /*Remove um item do início da fila q. Retorna true se
   operação realizada com sucesso, false caso contrário*/
    tipo_elem *p;

    if (IsEmpty(q))
        return FALSE;

    p = q->inicio;
    *info = p->info;
    q->inicio = p->lig;

    if (q->inicio == NULL)
        q->fim = NULL;

    free(p);
    return TRUE;
}
```



```
int tamanho (fila *q){
    /*Retorna o tamanho da fila*/
    tipo_elem *p;
    int cont = 0;

    p = q->inicio;
    while (p != NULL){
        cont ++;
        p = p->lig;
    }

    return cont;
}


boolean começo_fila (fila *q, tipo_info *item){
    /*Mostra o começo da fila sem remover o item. Retorna true
   se operação realizada com sucesso, false caso contrário*/
    if (IsEmpty(q))
        return FALSE;

    *item = q->inicio->info;
    return TRUE;
}
```

Análise dos 2 tipos de Representação

- Vantagens da Fila Estática (Anel):
  - não envolve custos da alocação dinâmica
- Desvantagens da Fila Estática:
  - previsão de tamanho máximo
- Vantagens da Fila Dinâmica:
  - ocupa espaço estritamente necessário
- Desvantagens da Fila Dinâmica:
  - custos usuais da alocação dinâmica (tempo de alocação, campos de ligação)

Quando usar

- Representação Estática (Anel):
  - quando fila tiver tamanho pequeno ou seu comportamento for previsível
- Representação Dinâmica:
  - nos demais casos

Exercícios

1. Implemente um procedimento reverso que reposiciona os elementos na fila de forma que o início se torne fim e vice-versa. Use uma pilha.
   - I F → I F
   - 1 → 2 → 3 3 → 2 → 1
2. Obtenha uma representação mapeando uma pilha P e uma fila F em um único array V[n]. Escreva algoritmos para inserir e eliminar elementos destes 2 objetos de dados. O que você pode dizer sobre a conveniência de sua representação?