

# **AULA 03**

# **ESTRUTURA DE DADOS**

---

**Lista linear sequencial**

**Norton T. Roman & Luciano A. Digiampietri**

# Lista linear

Estrutura de dados na qual **cada elemento é precedido por um elemento e sucedido por outro** (exceto o primeiro que não tem predecessor e o último que não tem sucessor).

Os elementos estão em uma dada **ordem** (por exemplo, a ordem de inclusão ou ordenados por uma chave).

# Lista linear sequencial

É uma **lista linear** na qual a **ordem lógica** dos elementos (a ordem “vista” pelo usuário) **é a mesma ordem física** (em memória principal) dos elementos. Isto é, elementos vizinhos na lista estarão em posições vizinhas de memória.

# Lista linear sequencial

## Modelagem:

Modelaremos usando um **arranjo** de registros;  
**Registros** conterão as informações de interesse do usuário;  
Nosso arranjo terá um **tamanho fixo** e controlaremos o número de elementos com uma **variável adicional**.

# Modelagem

```
#define MAX 50
```

```
typedef int TIPOCHAVE;
```

```
typedef struct{  
    TIPOCHAVE chave;  
    // outros campos...  
} REGISTRO;
```

```
typedef struct {  
    REGISTRO A[MAX];  
    int nroElem;  
} LISTA;
```

# Funções de gerenciamento

Implementaremos funções para:

**Inicializar a estrutura**

**Retornar a quantidade de elementos válidos**

**Exibir os elementos da estrutura**

**Buscar por um elemento na estrutura**

**Inserir elementos na estrutura**

**Excluir elementos da estrutura**

**Reinicializar a estrutura**

# Inicialização

**Para inicializar uma estrutura qualquer, precisamos pensar nos valores adequados para cada um dos campos de nossa estrutura**

# Inicialização

**Para inicializar uma estrutura qualquer, precisamos pensar nos valores adequados para cada um dos campos de nossa estrutura**

**Para inicializar uma lista sequencial já criada pelo usuário, só precisamos colocar o valor 0 (zero) no número de elementos válidos**



# Inicialização

```
void inicializarLista(LISTA l) {  
    l.nroElem = 0;  
}
```

# Inicialização

```
void inicializarLista(LISTA l) {  
    l.nroElem = 0;  
}
```

**Há algum problema com este código?**

# Inicialização

```
void inicializarLista(LISTA l) {  
    l.nroElem = 0;  
}
```

**Há algum problema com este código?  
Qual a diferença entre os códigos?**

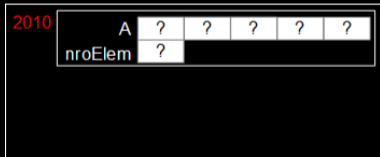
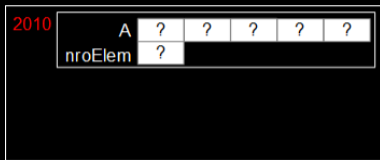
```
void inicializarLista(LISTA* l) {  
    l->nroElem = 0;  
}
```

# Inicialização

```
void inicializarLista(LISTA l) {  
    l.nroElem = 0;  
}
```

**Há algum problema com este código?  
Qual a diferença entre os códigos?**

```
void inicializarLista(LISTA* l) {  
    l->nroElem = 0;  
}
```

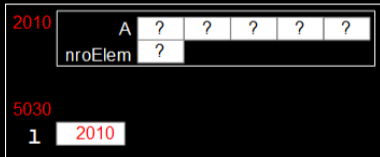
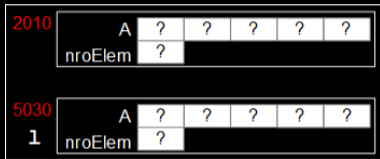


# Inicialização

```
void inicializarLista(LISTA l) {  
    l.nroElem = 0;  
}
```

**Há algum problema com este código?  
Qual a diferença entre os códigos?**

```
void inicializarLista(LISTA* l) {  
    l->nroElem = 0;  
}
```

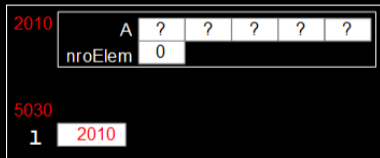
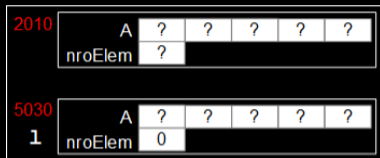


# Inicialização

```
void inicializarLista(LISTA l) {  
    l.nroElem = 0;  
}
```

Há algum problema com este código?  
Qual a diferença entre os códigos?

```
void inicializarLista(LISTA* l) {  
    l->nroElem = 0;  
}
```

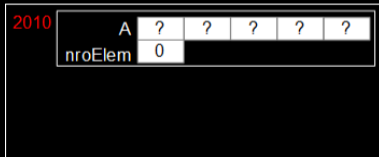
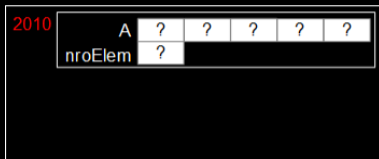


# Inicialização

```
void inicializarLista(LISTA l) {  
    l.nroElem = 0;  
}
```

**Há algum problema com este código?  
Qual a diferença entre os códigos?**

```
void inicializarLista(LISTA* l) {  
    l->nroElem = 0;  
}
```



# Retornar número de elementos

Para esta estrutura basta retornar o valor do campo  
*nroElem*



# Retornar número de elementos

```
int tamanho(LISTA* l) {  
    return l->nroElem;  
}
```

# Exibição/Impressão

Para exibir os elementos da estrutura precisaremos iterar pelos **elementos** válidos e, por exemplo, **imprimir suas chaves**.

# Exibição/Impressão

```
void exibirLista(LISTA* l){
    int i;
    printf("Lista: \n ");

    printf("\n\n");
}
```

# Exibição/Impressão

```
void exibirLista(LISTA* l){
    int i;
    printf("Lista: \n ");
    for (i=0; i < l->nroElem; i++)
        printf("%i ", l->A[i].chave);
    printf("\n\n");
}
```

# Exibição/Impressão

```
void exibirLista(LISTA* l){
    int i;
    printf("Lista: \n ");
    for (i=0; i < l->nroElem; i++)
        printf("%i ", l->A[i].chave);
    printf("\n\n");
}
```

2010	A	21	9	55	?	?
	nroElem	3				
5030	1	2010				

# Exibição/Impressão

```
void exibirLista(LISTA* l){
    int i;
    printf("Lista: \" ");
    for (i=0; i < l->nroElem; i++)
        printf("%i ", l->A[i].chave);
    printf("\n\n");
}
```

2010	A	21	9	55	?	?
	nroElem	3				
5030	1	2010				

Saída:

\$ Lista: " 21 9 55 "

# Buscar por elemento

A função de busca deverá:

**Receber uma chave** do usuário

**Retornar a posição** em que este elemento se encontra na lista (caso seja encontrado)

**Retornar -1** caso não haja um registro com essa chave na lista

# Busca sequencial

```
int buscaSequencial(LISTA* l, TIPOCHAVE ch) {
```



# Busca sequencial

```
int buscaSequencial(LISTA* l, TIPOCHAVE ch) {  
    int i = 0;  
    while (i < l->nroElem){  
        if(ch == l->A[i].chave) return i;  
        else i++;  
    }  
    return -1;  
}
```

# **Inserção de um elemento**

**O usuário passa como parâmetro um registro a ser inserido na lista**

**Há diferentes possibilidades de inserção:**

**No início**

**No fim**

**Ordenada pela chave**

**Numa posição indicada pelo usuário**

# Inserção de um elemento

O usuário passa como parâmetro um registro a ser inserido na lista

Há diferentes possibilidades de inserção:

No início

No fim

Ordenada pela chave

**Numa posição indicada pelo usuário**

# Inserção de um elemento

Como inserir?

Se a lista **não estiver cheia** e o **índice** passado pelo usuário for **válido**: **desloca** todos os elementos posteriores uma posição para a direita; **insere** o elemento na posição desejada, **soma um** no campo *nroElem* e **retorna *true***

Caso contrário retorna *false*



# Inserção em posição específica

```
bool inserirElemLista(LISTA* l, REGISTRO reg, int i){
    int j;
    if ((l->nroElem == MAX) || (i < 0) || (i > l->nroElem))
        return false;

}
```

# Inserção em posição específica

```
bool inserirElemLista(LISTA* l, REGISTRO reg, int i){
    int j;
    if ((l->nroElem == MAX) || (i < 0) || (i > l->nroElem))
        return false;
    for (j = l->nroElem; j > i; j--) l->A[j] = l->A[j-1];
    l->A[i] = reg;
    l->nroElem++;
    return true;
}
```

# Inserção em posição específica

```
bool inserirElemLista(LISTA* l, REGISTRO reg, int i){
    int j;
    if ((l->nroElem == MAX) || (i < 0) || (i > l->nroElem))
        return false;
    for (j = l->nroElem; j > i; j--) l->A[j] = l->A[j-1];
    l->A[i] = reg;
    l->nroElem++;
    return true;
}
```

2010

A	21	9	55	?	?
nroElem	3				



# Inserção em posição específica

```
bool inserirElemLista(LISTA* l, REGISTRO reg, int i){
    int j;
    if ((l->nroElem == MAX) || (i < 0) || (i > l->nroElem))
        return false;
    for (j = l->nroElem; j > i; j--) l->A[j] = l->A[j-1];
    l->A[i] = reg;
    l->nroElem++;
    return true;
}
```

2010	A	21	9	55	?	?
	nroElem	3				
l	2010	i	2			
reg	33	j	?			

# Inserção em posição específica

```
bool inserirElemLista(LISTA* l, REGISTRO reg, int i){
    int j;
    if ((l->nroElem == MAX) || (i < 0) || (i > l->nroElem))
        return false;
    for (j = l->nroElem; j > i; j--) l->A[j] = l->A[j-1];
    l->A[i] = reg;
    l->nroElem++;
    return true;
}
```

2010	A	21	9	55	?	?
	nroElem	3				
l	2010	i	2			
reg	33	j	3			

# Inserção em posição específica

```
bool inserirElemLista(LISTA* l, REGISTRO reg, int i){
    int j;
    if ((l->nroElem == MAX) || (i < 0) || (i > l->nroElem))
        return false;
    for (j = l->nroElem; j > i; j--) l->A[j] = l->A[j-1];
    l->A[i] = reg;
    l->nroElem++;
    return true;
}
```

2010	A	21	9	55	55	?
	nroElem	3				
l	2010	i	2			
reg	33	j	3			

# Inserção em posição específica

```
bool inserirElemLista(LISTA* l, REGISTRO reg, int i){
    int j;
    if ((l->nroElem == MAX) || (i < 0) || (i > l->nroElem))
        return false;
    for (j = l->nroElem; j > i; j--) l->A[j] = l->A[j-1];
    l->A[i] = reg;
    l->nroElem++;
    return true;
}
```

2010	A	21	9	55	55	?
	nroElem	3				
l	2010	i	2			
reg	33	j	2			

# Inserção em posição específica

```
bool inserirElemLista(LISTA* l, REGISTRO reg, int i){
    int j;
    if ((l->nroElem == MAX) || (i < 0) || (i > l->nroElem))
        return false;
    for (j = l->nroElem; j > i; j--) l->A[j] = l->A[j-1];
    l->A[i] = reg;
    l->nroElem++;
    return true;
}
```

2010	A	21	9	33	55	?
	nroElem	3				
l	2010	i	2			
reg	33	j	2			

# Inserção em posição específica

```
bool inserirElemLista(LISTA* l, REGISTRO reg, int i){
    int j;
    if ((l->nroElem == MAX) || (i < 0) || (i > l->nroElem))
        return false;
    for (j = l->nroElem; j > i; j--) l->A[j] = l->A[j-1];
    l->A[i] = reg;
    l->nroElem++;
    return true;
}
```

2010	A	21	9	33	55	?
	nroElem	4				
l	2010	i	2			
reg	33	j	2			

# Inserção em posição específica

```
bool inserirElemLista(LISTA* l, REGISTRO reg, int i){
    int j;
    if ((l->nroElem == MAX) || (i < 0) || (i > l->nroElem))
        return false;
    for (j = l->nroElem; j > i; j--) l->A[j] = l->A[j-1];
    l->A[i] = reg;
    l->nroElem++;
    return true;
}
```

2010

A	21	9	33	55	?
nroElem	4				

# Exclusão de um elemento

O usuário passa a chave do elemento que ele quer excluir

Se houver um elemento com esta chave na lista, “**exclui este elemento**”, **desloca** todos os elementos posteriores uma posição para a esquerda, **diminui** em um o campo *nroElem* e **retorna *true***

Caso contrário, retorna *false*



# Exclusão de um elemento

```
bool excluirElemLista(TIPOCHAVE ch, LISTA* l) {
```

# Exclusão de um elemento

```
bool excluirElemLista(TIPOCHAVE ch, LISTA* l) {  
    int pos, j;  
    pos = buscaSequencial(l,ch);
```

# Exclusão de um elemento

```
bool excluirElemLista(TIPOCHAVE ch, LISTA* l) {  
    int pos, j;  
    pos = buscaSequencial(l,ch);  
    if(pos == -1) return false;
```

# Exclusão de um elemento

```
bool excluirElemLista(TIPOCHAVE ch, LISTA* l) {  
    int pos, j;  
    pos = buscaSequencial(l,ch);  
    if(pos == -1) return false;  
    for(j = pos; j < l->nroElem-1; j++)  
        l->A[j] = l->A[j+1];  
}
```

# Exclusão de um elemento

```
bool excluirElemLista(TIPOCHAVE ch, LISTA* l) {
    int pos, j;
    pos = buscaSequencial(l,ch);
    if(pos == -1) return false;
    for(j = pos; j < l->nroElem-1; j++)
        l->A[j] = l->A[j+1];
    l->nroElem--;
    return true;
}
```

# Reinicialização da lista

Para esta estrutura, para reinicializar a lista basta colocar 0 (zero) no campo *nroElem*

# Reinicialização da lista

```
void reinicializarLista(LISTA* l) {  
    l->nroElem = 0;  
}
```

# **AULA 03**

# **ESTRUTURA DE DADOS**

---

**Lista linear sequencial**

**Norton T. Roman & Luciano A. Digiampietri**