

# Estrutura de Dados

Prof. Rafael Nunes

# Listas

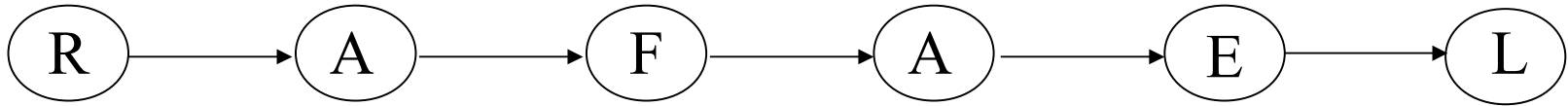
*Aula 05 - Parte 1*

Vamos Relembrar

O que é uma Lista?

# Lista

***Estruturas*** de armazenamento  
***de dados*** linear



Já sabemos que podemos ter  
*mais de um tipo* de lista?

Quais???

# Tipos de Listas

- Listas *lineares*
  - estáticas
  - dinâmicas
- Como manipulá-las:
  - *filas*
  - *pilhas*

# Filas x Pilhas



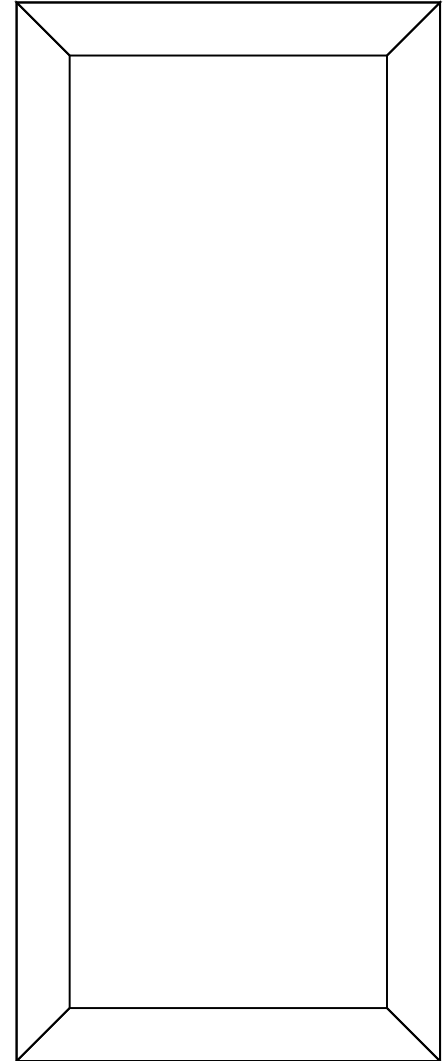
# Lista Estática

- Estrutura da Lista
- Aloca (armazena) um tamanho fixo na memória para armazenar todo o tamanho da lista.
- Os nós (ou células) contêm os dados de um tipo determinado tipo (int, char, ou um tipo definido pelo usuário: “struct”)
- Vejamos um exemplo...

# Lista Estática

- Armazena um tamanho fixo na memória para todo o tamanho da Lista.

**Memória RAM**  
(Computador)

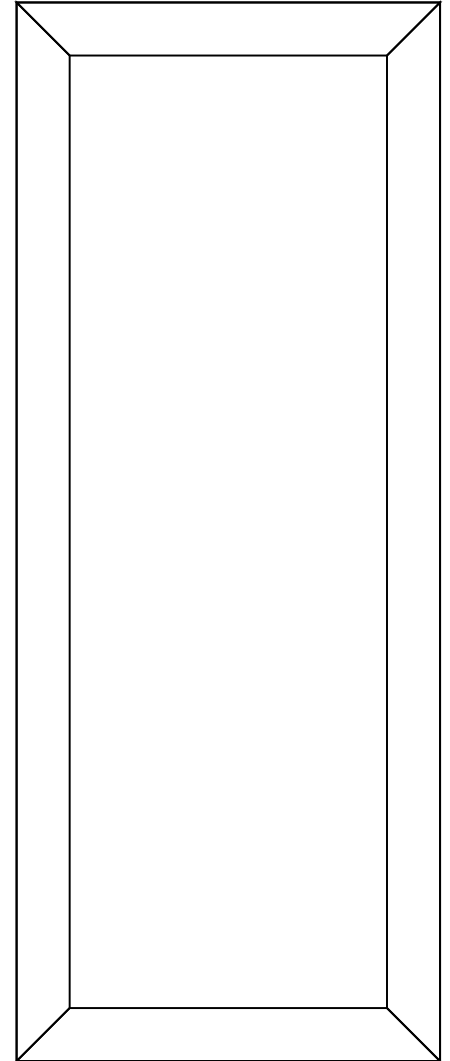


# Lista Estática

**Memória RAM**  
(Computador)

- Armazena um tamanho fixo na memória para todo o tamanho da Lista.

```
1. int main()  
2. {  
3.     char Listax[6];  
4.     ...  
5. }
```

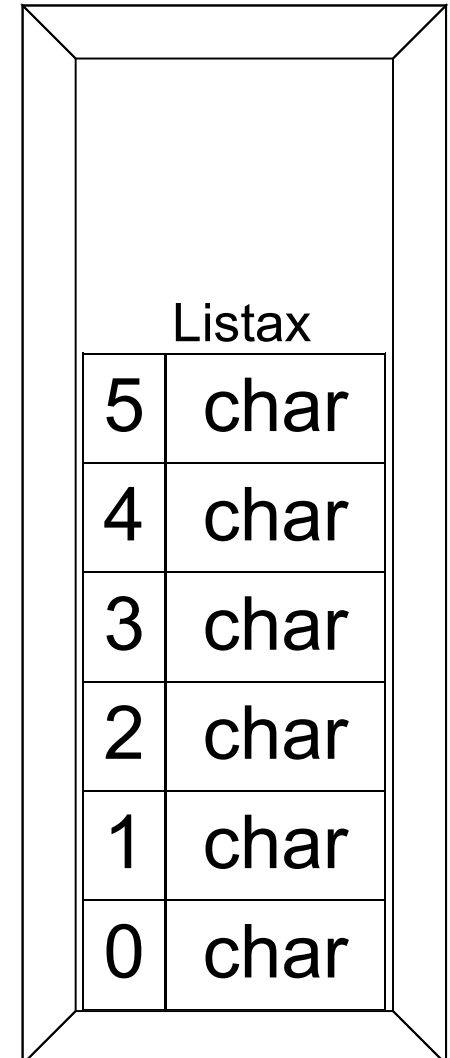


# Lista Estática

**Memória RAM**  
(Computador)

- Armazena um tamanho fixo na memória para todo o tamanho da Lista.

```
1. int main()  
2. {  
3.     char Listax [6];  
4.     ...  
5. }
```



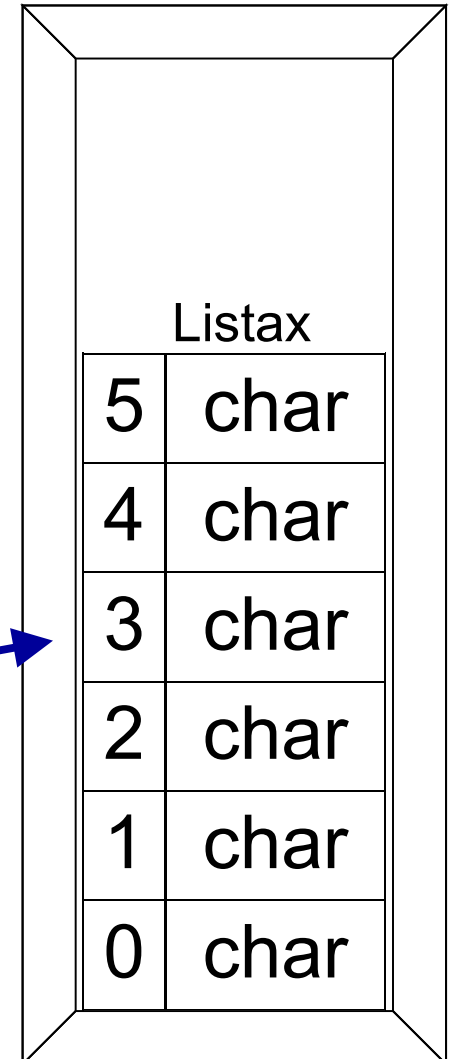
# Lista Estática

**Memória RAM**  
(Computador)

- Armazena um tamanho fixo na memória para todo o tamanho da Lista.

```
1. int main()  
2. {  
3.     char Listax [6];  
4.     ...  
5. }
```

**Espaço  
Armazenado!**



# Lista Estática

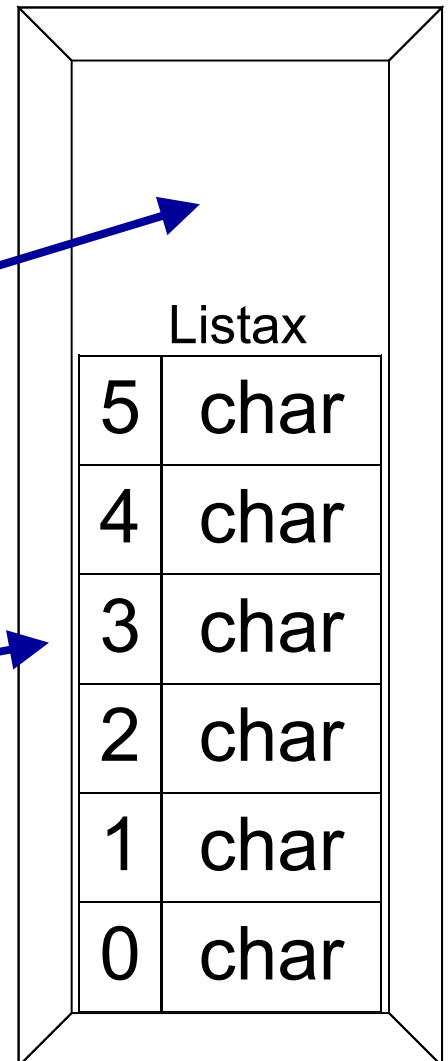
**Memória RAM**  
(Computador)

- Armazena um tamanho fixo na memória para todo o tamanho da Lista.

```
1. int main()  
2. {  
3.     char Listax [6];  
4.     ...  
5. }
```

**Espaço Livre**

**Espaço Armazenado!**



# Lista Estática

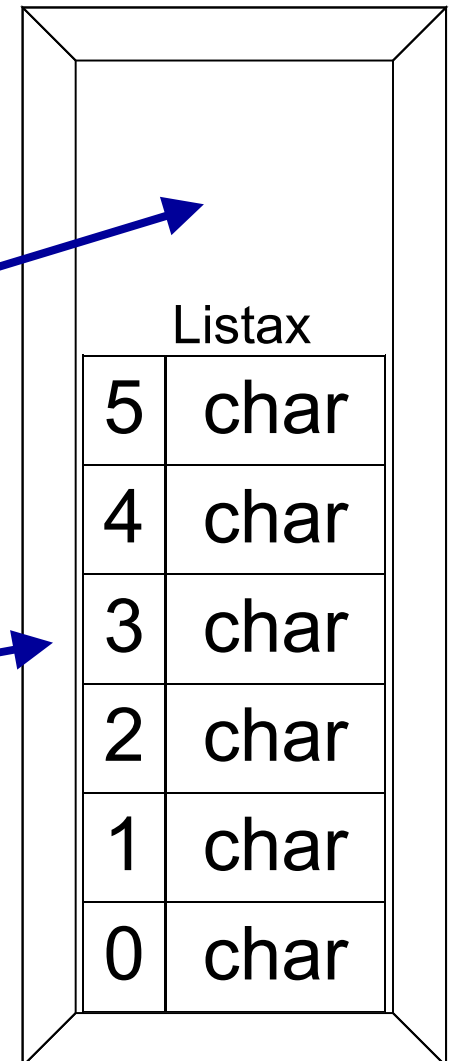
**Memória RAM**  
(Computador)

- Armazena um tamanho fixo na memória para todo o tamanho da Lista.

```
1. int main()  
2. {  
3.     char Listax [6];  
4.     ...  
5. }
```

**Espaço Livre**  
**A lista não  
pode ser  
aumentada**

**Espaço  
Armazenado!**



# Lista Estática

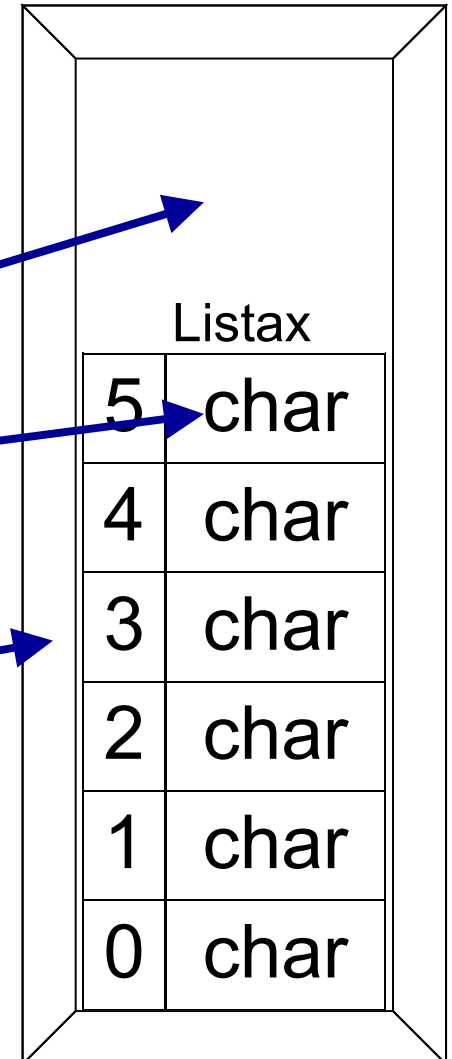
**Memória RAM**  
(Computador)

- Armazena um tamanho fixo na memória para todo o tamanho da Lista.

```
1. int main()  
2. {  
3.     char Listax [6];  
4.     ...  
5. }
```

**Espaço Livre**  
**Todos os nós tem o mesmo tipo**

**Espaço Armazenado!**





# Lista Estática

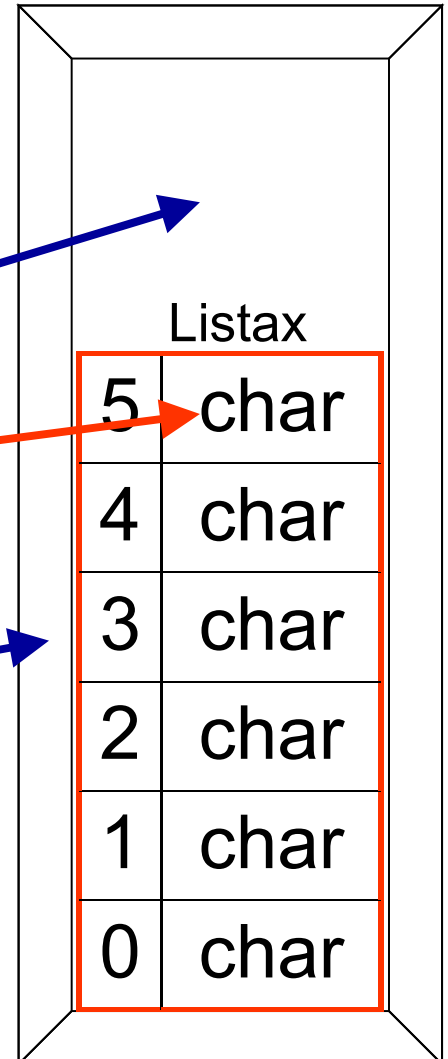
**Memória RAM**  
(Computador)

- Armazena um tamanho fixo na memória para todo o tamanho da Lista.

```
1. int main()  
2. {  
3.     char Listax [6];  
4.     ...  
5. }
```

**Espaço Livre**  
**Tamanho armazenado é de 6 x char**

**Espaço Armazenado!**



# Exemplo de Funcionamento

*Manipulação através de Fila*

# Manipulação da Lista - Fila

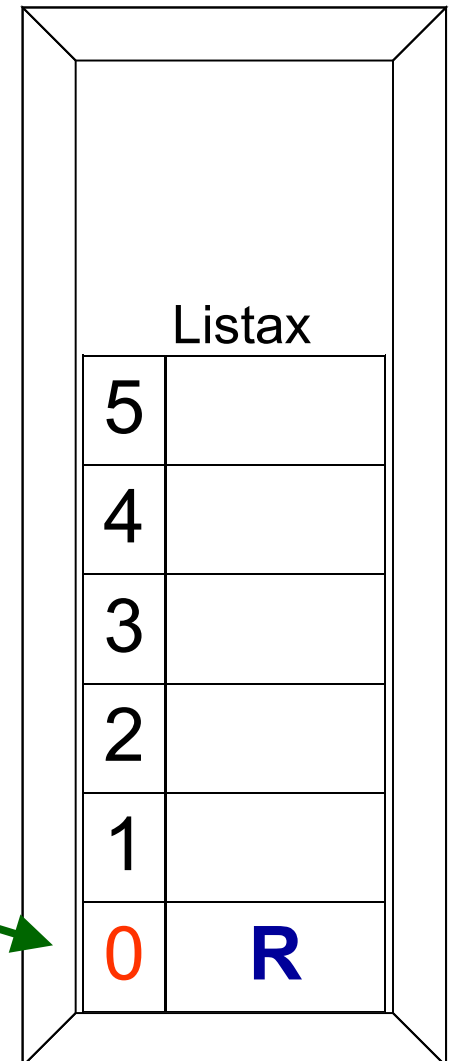
- FIFO: Primeiro a entrar, primeiro a sair
- *Inserir no Final*
- Remove do Início
- Reestrutura a Lista

# Lista Estática (Fila)

**Memória RAM**  
(Computador)

```
void Insere(char x, char *ListaAux)
{
    ListaAux [pos++] = x;
    ListaAux [++pos] = '\0';
}
```

```
1.  int main()
2.  {
3.      char Listax [6];
4.      Insere('R', Listax);
5.  }
```

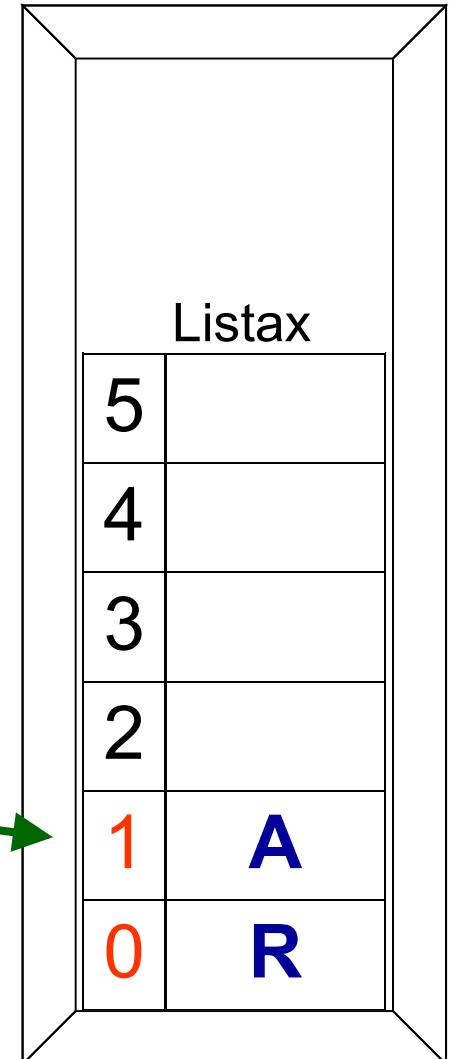


# Lista Estática (Fila)

**Memória RAM**  
(Computador)

```
void Insere(char x, char *ListaAux)
{
    ListaAux [pos++] = x;
    ListaAux [++pos] = '\0';
}
```

```
1.  int main()
2.  {
3.      char Listax [6];
4.      Insere('R', Listax);
5.      Insere('A', Listax);
6.      ...
7.  }
```



# Lista Estática (Fila)

**Memória RAM**  
(Computador)

```
void Insere(char x, char *ListaAux)
{
    ListaAux [pos++] = x;
    ListaAux [++pos] = '\0';
}
```

```
1.  int main()
2.  {
3.      char Listax [6];
4.      Insere('R', Listax);
5.      Insere('A', Listax);
6.      Insere('F', Listax);
7.  }
```

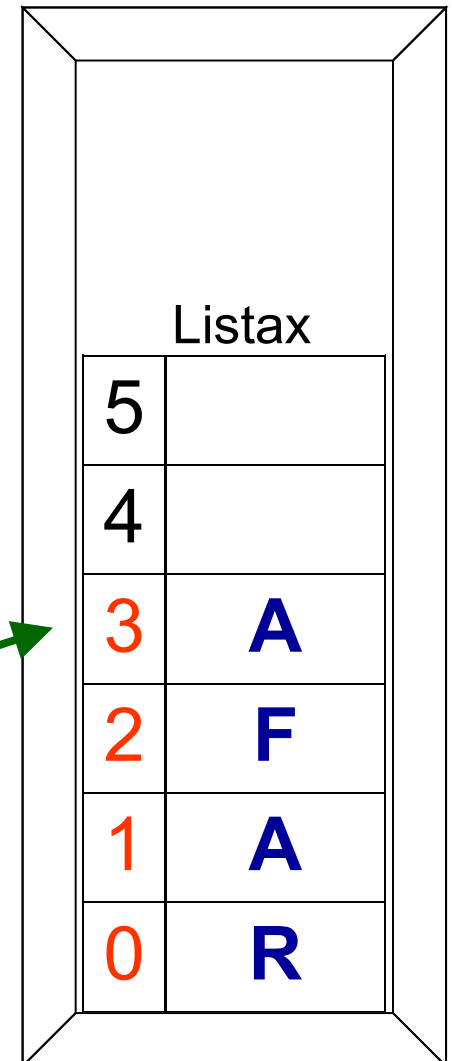
Listax	
5	
4	
3	
2	F
1	A
0	R

# Lista Estática (Fila)

**Memória RAM**  
(Computador)

```
void Insere(char x, char *ListaAux)
{
    ListaAux [pos++] = x;
    ListaAux [++pos] = '\0';
}
```

```
1.  int main()
2.  {
3.      char Listax [6];
4.      Insere('R', Listax);
5.      Insere('A', Listax);
6.      Insere('F', Listax);
7.      Insere('A', Listax);
8.  }
```



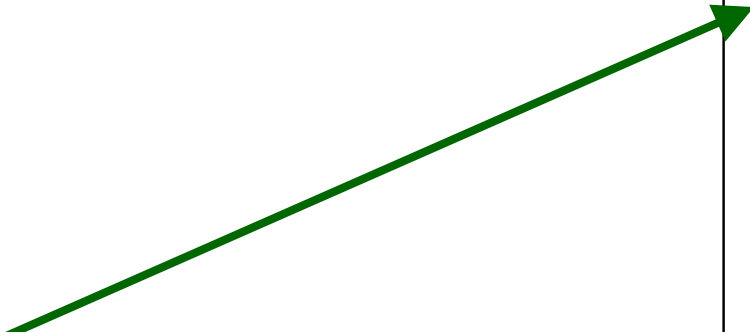
Listax	
5	
4	
3	A
2	F
1	A
0	R

# Lista Estática (Fila)

**Memória RAM**  
(Computador)

```
void Insere(char x, char *ListaAux)
{
    ListaAux [pos++] = x;
    ListaAux [++pos] = '\0';
}
```

```
1. int main()
2. {
3.     char Listax [6];
4.     Insere('R', Listax);
5.     Insere('A', Listax);
6.     Insere('F', Listax);
7.     Insere('A', Listax);
8.     Insere('E', Listax);
9. }
```



5	
4	E
3	A
2	F
1	A
0	R



# Lista Estática (Fila)

**Memória RAM**  
(Computador)

```
void Insere(char x, char *ListaAux)
{
    ListaAux [pos++] = x;
    ListaAux [++pos] = '\0';
}
```

```
1. int main()
2. {
3.     char Listax [6];
4.     Insere('R', Listax);
5.     Insere('A', Listax);
6.     Insere('F', Listax);
7.     Insere('A', Listax);
8.     Insere('E', Listax);
9.     Insere('L', Listax);
10. }
```

*Podemos caminhar  
sequencialmente  
através das posições  
da memória (pos++)*

Listax	
5	L
4	E
3	A
2	F
1	A
0	R

# Manipulação da Lista - Fila

- FIFO: Primeiro a entrar, primeiro a sair
- Insere no Final
- ***Remove do Inicio***
- Reestrutura a Lista

# Lista Estática (Fila)

**Memória RAM**  
(Computador)

```
void Remove(char *vet) {  
    int i, j=1, tam;  
    tam = strlen(vet);  
    for (i=0; i< tam; i++) {  
        vet [i] = vet [j++];  
    }  
}  
  
1.  int main()  
2.  {  
3.      char Listax [6];  
4.      Remove(Listax);  
  
5.  }
```

Listax	
5	L
4	E
3	A
2	F
1	A
0	R

# Lista Estática (Fila)

**Memória RAM**  
(Computador)

```
void Remove(char *vet) {  
    int i, j=1, tam;  
    tam = strlen(vet);  
    for (i=0; i< tam; i++) {  
        vet [i] = vet [j++];  
    }  
}
```

```
1. int main()  
2. {  
3.     char Listax [6];  
4.     Remove(Listax);  
5. }
```

**Remove do  
Início!**

**O primeiro que  
entrou na Listax**

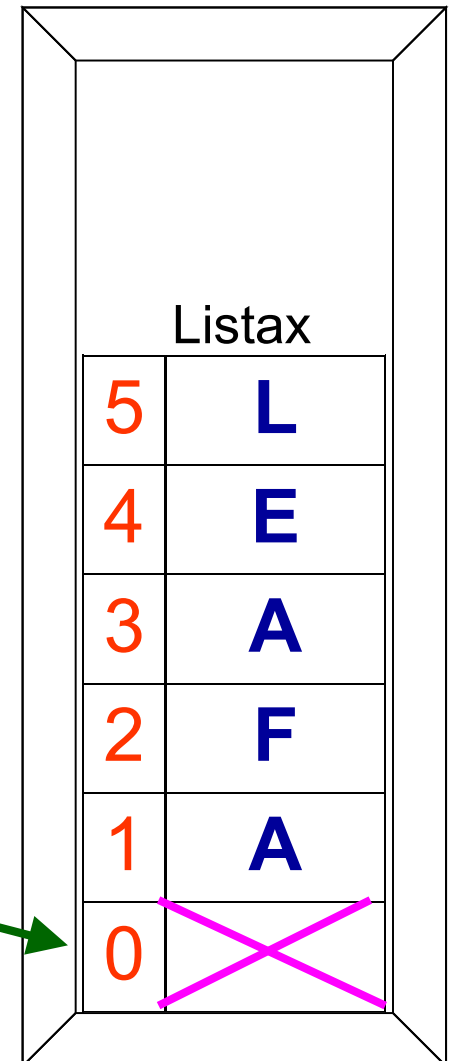
Listax	
5	L
4	E
3	A
2	F
1	A
0	R

# Lista Estática (Fila)

**Memória RAM**  
(Computador)

```
void Remove(char *vet) {  
    int i, j=1, tam;  
    tam = strlen(vet);  
    for (i=0; i< tam; i++) {  
        vet [i] = vet [j++];  
    }  
}
```

```
1. int main()  
2. {  
3.     char Listax [6];  
4.     Remove(Listax);  
5. }
```



# Lista Estática (Fila)

**Memória RAM**  
(Computador)

```
void Remove(char *vet) {  
    int i, j=1, tam;  
    tam = strlen(vet);  
    for (i=0; i< tam; i++) {  
        vet [i] = vet [j++];  
    }  
}
```

```
1. int main()  
2. {  
3.     char Listax [6];  
4.     Remove(Listax);  
5. }
```

*É necessário  
reestruturar a  
Listax*

Listax	
5	L
4	E
3	A
2	F
1	A
0	

# Lista Estática (Fila)

**Memória RAM**  
(Computador)

```
void Remove(char *vet) {  
    int i, j=1, tam;  
    tam = strlen(vet);  
    for (i=0; i< tam; i++) {  
        vet [i] = vet [j++];  
    }  
}  
  
1.  int main()  
2.  {  
3.      char Listax [6];  
4.      Remove(Listax);  
  
5.  }
```

Listax	
5	L
4	E
3	A
2	F
1	
0	A

# Lista Estática (Fila)

**Memória RAM**  
(Computador)

```
void Remove(char *vet) {  
    int i, j=1, tam;  
    tam = strlen(vet);  
    for (i=0; i< tam; i++) {  
        vet [i] = vet [j++];  
    }  
}  
  
1.  int main()  
2.  {  
3.      char Listax [6];  
4.      Remove(Listax);  
  
5.  }
```

Listax	
5	L
4	E
3	A
2	
1	F
0	A



# Lista Estática (Fila)

**Memória RAM**  
(Computador)

```
void Remove(char *vet) {  
    int i, j=1, tam;  
    tam = strlen(vet);  
    for (i=0; i< tam; i++) {  
        vet [i] = vet [j++];  
    }  
}  
  
1.  int main()  
2.  {  
3.      char Listax [6];  
4.      Remove(Listax);  
  
5.  }
```

Listax	
5	L
4	E
3	
2	A
1	F
0	A

# Lista Estática (Fila)

**Memória RAM**  
(Computador)

```
void Remove(char *vet) {  
    int i, j=1, tam;  
    tam = strlen(vet);  
    for (i=0; i< tam; i++) {  
        vet [i] = vet [j++];  
    }  
}  
  
1.  int main()  
2.  {  
3.      char Listax [6];  
4.      Remove(Listax);  
  
5.  }
```

Listax	
5	L
4	
3	E
2	A
1	F
0	A

# Lista Estática (Fila)

**Memória RAM**  
(Computador)

```
void Remove(char *vet) {  
    int i, j=1, tam;  
    tam = strlen(vet);  
    for (i=0; i< tam; i++) {  
        vet [i] = vet [j++];  
    }  
}  
  
1.  int main()  
2.  {  
3.      char Listax [6];  
4.      Remove(Listax);  
  
5.  }
```

Listax	
5	
4	L
3	E
2	A
1	F
0	A

Até a próxima...