# Listas em C++

EDUARDO HABIB BECHELANE MAIA

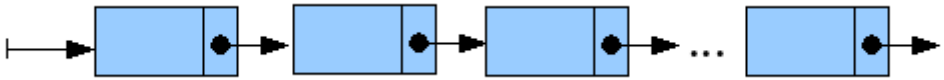
HABIB@CEFETMG.BR

# Listas lineares

1. É uma das formas mais simples de interligar elementos
2. Estrutura em que as operações inserir, retirar e localizar são definidas.
3. Podem crescer ou diminuir de tamanho durante a execução de um programa, de acordo com a demanda.
4. Itens podem ser acessados, inseridos ou retirados de uma lista.
5. Duas listas podem ser concatenadas para formar uma lista única, ou uma pode ser partida em duas ou mais listas.
6. Adequadas quando não é possível prever a demanda por memória, permitindo a manipulação de quantidades imprevisíveis de dados, de formato também imprevisível.

# Listas Lineares

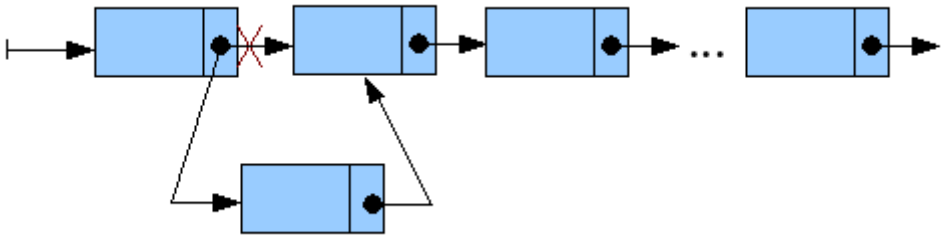
1. Diferente do que ocorre em vetores, nas listas lineares os elementos não estão necessariamente armazenados sequencialmente na memória



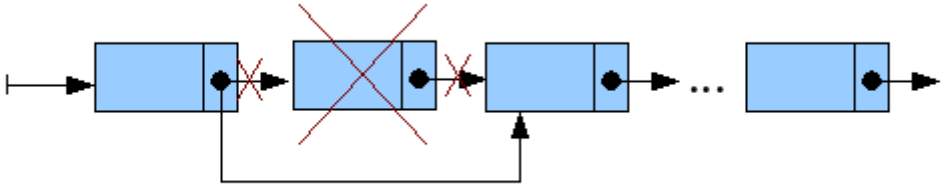
2. Exemplo de lista duplamente encadeada

# Inserção e remoção

1. Inserção em lista:



2. Remoção



# Exemplo No.hpp

```
#ifndef NO_HPP
#define NO_HPP
#include <iostream>
#include "No.hpp"

using namespace std;

class No
{
private:
        int v;
        No* prox;

public:
        No(int v);
        int obterValor();

        No* obterProx();

        void setProx(No* p);
};

#endif
```

# Exemplo No.cpp

```
#include <iostream>
#include "No.hpp"

No::No(int v) // construtor
{
        this->v = v;
        this->prox = NULL;

}

int No::obterValor() // obtém o valor
{
        return this->v;
}

No* No::obterProx() // obtém o próximo No
{
        return prox;
}

void No::setProx(No* p) // seta o próximo No
{
        prox = p;
}
```

# Lista.hpp

```
#ifndef LISTA_HPP
#define LISTA_HPP
using namespace std;
class Lista
{
private:
        No* cabeca; // primeiro
elemento
        No* cauda; // último elemento
public:
        Lista();
        Lista(int v);
        virtual ~Lista();
        void mostrarTodos();
        bool vazia();
        void inserir_inicio(int v);
        void inserir_final(int v);
        int size();
        bool existe(int v);
        void remover();
};
#endif
```

# Exemplo Lista.cpp

```
#include <iostream>
#include "No.hpp"
#include "Lista.hpp"

Lista::Lista()
{
        // se não passar elemento, então cabeca e cauda são NULL
        cabeca = NULL;
        cauda = NULL;
}

Lista::Lista(int v)
{
        // se passar elemento, então cria novo No
        cabeca = new No(v);
        cauda = cabeca;
}

Lista::~Lista() // destrutor
{
   No* atual = cabeca;
   while (atual != NULL)
   {
      No* temp = atual;
      atual = atual->obterProx();
      delete temp;
   }
}
```

# Exemplo Lista.cpp

```
// mostra todos os elementos da lista
void Lista::mostrarTodos()
{
        cout << "\nImprimindo todos os elementos...\n";
        No* c = cabeca;

        if(vazia())
                cout << "A lista NÃO possui
elementos!!\n";
        else
        {
                while(c) // percorre a lista
                {
                        cout << c->obterValor() << endl;
                        c = c->obterProx();
                }
                cout << endl;
        }
}
```

# Exemplo Lista.cpp

```
bool Lista::vazia() // verifica se a lista está vazia
{
	return (cabeca == NULL);
}

// insere no início (semelhante a push_front da list)
void Lista::inserir_inicio(int v)
{
	No* novo_no = new No(v);

	if(vazia())
	{
		cabeca = novo_no;
		cauda = novo_no;
	}
	else
	{
		novo_no->setProx(cabeca);
		cabeca = novo_no;
	}
}
```

# Exemplo Lista.cpp

```
// insere no final (semelhante a push_back)
void Lista::inserir_final(int v)
{
        No* novo_no = new No(v);

        if(vazia())
        {
                cabeca = novo_no;
                cauda = novo_no;
        }
        else
        {
                cauda->setProx(novo_no);
                cauda = novo_no;
        }
}
```

# Exemplo Lista.cpp

```
// retorna o tamanho da lista
int Lista::size()
{
	if(vazia()) // se for vazia, então retorna 0
		return 0;

	No* c = cabeca;
	int tam = 0;

	// percorre a lista
	do
	{
		c = c->obterProx();
		tam++;
	}
	while(c);

	return tam;
}
```

# Exemplo Lista.cpp

```
// verifica se um elemento existe na lista
bool Lista::existe(int v)
{
	No* c = cabeca;

	while(c)
	{
		if(c->obterValor() == v)
			return true;
		c = c->obterProx();
	}
	return false;
}
```

# Exemplo Lista.cpp

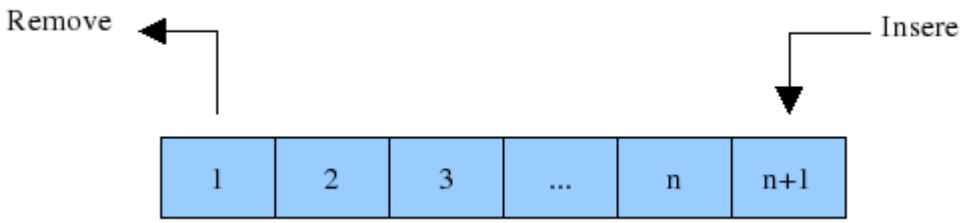
```
// remove da lista, remoção do final
void Lista::remover()
{
	if(!vazia())
	{
		// se houver só 1 elemento
		if(cabeca->obterProx() == NULL)
		{
			delete cabeca;  // Libera memória
			cabeca = NULL;
			cauda = NULL;  // A cauda também deve ser atualizada
		}
		else if(cabeca->obterProx()->obterProx() == NULL) // 2 elementos
		{
			delete cabeca->obterProx();  // Libera memória
			cabeca->setProx(NULL);
			cauda = cabeca;  // Atualiza a cauda
		}
		else // > 2 elementos
		{
			No* ant_ant = cabeca;
			No* ant = cabeca->obterProx();
			No* corrente = cabeca->obterProx()->obterProx();

			while(corrente)
			{
				No* aux = ant;
				ant = corrente;
				ant_ant = aux;
				corrente = corrente->obterProx();
			}
			delete ant_ant->obterProx();  // Libera memória
			ant_ant->setProx(NULL);  // Seta o prox como NULL
			cauda = ant_ant;  // Atualiza a cauda
		}
	}
}
```

# Exemplo main.cpp

```
#include <iostream>
#include <locale>
#include "No.hpp"
#include "Lista.hpp"

int main(int argc, char *argv[])
{
   Lista l;
   setlocale(LC_ALL,"Portuguese");
   if(l.vazia())
      cout << "A lista está vazia!!\n";
   else
      cout << "NÃO está vazia!!\n";

   l.mostrarTodos();
   if(l.existe(10))
      cout << "\n existe na lista!!\n";
   else
      cout << "\nNÃO existe!!\n";

   l.inserir_final(10);
   l.inserir_final(20);
   l.inserir_final(30);
   l.inserir_final(40);
   l.inserir_inicio(50);

   l.mostrarTodos();

   if(l.vazia())
      cout << "A lista está vazia!!\n";
   else
      cout << "NÃO está vazia!!\n";

   if(l.existe(10))
      cout << "\nExiste na lista!!\n";
   else
      cout << "\nNÃO existe!!\n";

   l.remover();
   l.mostrarTodos();
   cout<<"Tamanho da lista: "
<<l.size();
   return 0;
  }
```

# Filas

1. São estruturas de dados do tipo FIFO (first-in first-out)
   a. Primeiro a entrar, primeiro a sair



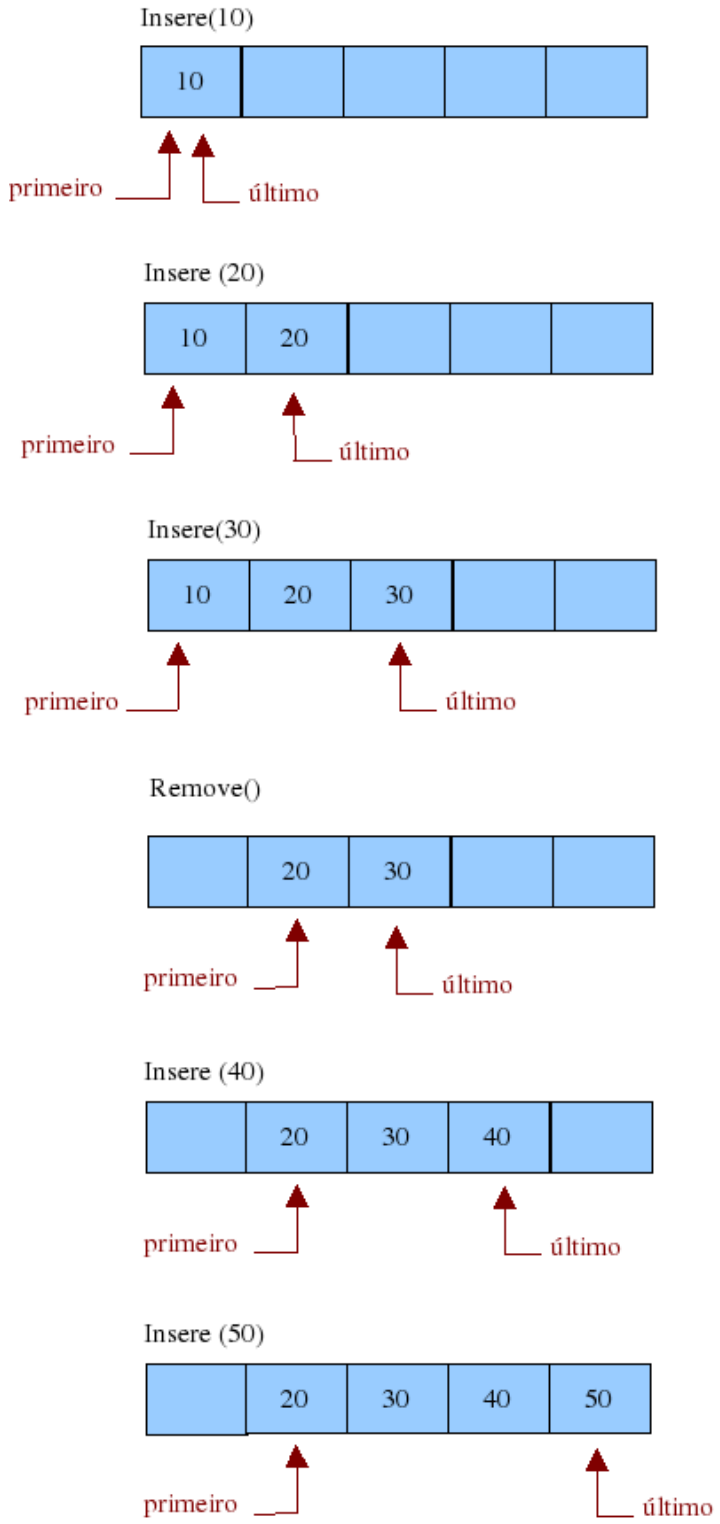

2. Exemplos:
   a. Controle de impressão
   b. Fila de pessoas
   c. etc

# Operações

1. criação da fila;
2. enfileirar;
3. desenfileirar;
4. mostrar a fila;
5. verificar se a fila está vazia;

# Exemplo

**Imagens retiradas de https://www.cos.ufrj.br/~rfarias/cos121/filas.html**



# Referências

https://www.cos.ufrj.br/~rfarias/cos121/aula_11.html

https://www.cos.ufrj.br/~rfarias/cos121/filas.html