LISTA GENÉRICA

LISTA GENÉRICA

- Antes de iniciarmos nossos estudos sobre as listas genéricas, é importante entender o que são e quais as diferenças relacionadas com a lista específica.

LISTA GENÉRICA

- Qualquer TAD específico possui sua aplicação limitada ao domínio do problema computacional ao qual se está resolvendo

- Por exemplo:
  - contexto de diário online de aula, a lista de alunos
  - contexto frota, lista de caminhões

LISTA GENÉRICA

- Não é possível aplicar a implementação de uma lista de frota de veículos a uma lista de alunos, sem que seja necessárias mudanças nas implementações

- Deste modo, a lista genérica é criada com o intuito de ser aplicada a qualquer problema possível

LISTA GENÉRICA

- Quando criarmos uma lista genérica, evitamos o retrabalho, embora tenhamos que utilizar de recursos mais sofisticados de programação

- Uma lista de genérica poderia ser aplicada a:
  - controle de frota
  - controle de alunos
  - controle de funcionários
  - controle de despesas
  - ...

LISTA GENÉRICA

- Nesta apresentação veremos a lista genérica em uma lista estática. É importante destacar que podemos aplicar este conceito sobre uma lista dinâmica.

- Como utilizamos a linguagem C++ na disciplina, para criar uma TAD genérica, utilizaremos o famoso *template*

LISTA GENÉRICA

```
const int MAXIMO = 52;
struct TCartas{
       int naipe;
       int valor;
};

struct TBaralho{
       TCartas cartas[MAXIMO];
       int quantidade;
};
```

específica

```
template <typename TIPO>
struct TElemento{
       TIPO dado;
};

template <typename TIPO, int MAX>
struct TLista{
       TElemento <TIPO>
                     elementos[MAX];
       int quantidade;
};
```

genérica

LISTA GENÉR

Note que na específica há uma aplicação intrínseca a um contexto de jogos de cartas.

Em uma genérica não há qualquer referência a sua aplicação (domínio)

```
const int MAXIMO = 52;
struct TCartas{
       int naipe;
       int valor;
};

struct TBaralho{
       TCartas cartas[MAXIMO];
       int quantidade;
};
```

específica

```
template <typename TIPO>
struct TElemento{
       TIPO dado;
};

template <typename TIPO, int MAX>
struct TLista{
       TElemento <TIPO>
                     elementos[MAX];
       int quantidade;
};
```

genérica

LISTA GENÉRICA

- Uma das maiores confusões dos alunos no contexto de TADs genéricos é a aplicação da técnica em uma linguagem de programação.

- No caso do C++, a confusão ocorre com os ***templates***

LISTA GENÉRICA

```
template <typename TIPO>
struct TElemento{
        TIPO dado;
};

template <typename TIPO, int MA
struct TLista{
        TElemento<TIPO> elementos[MAX];
        int quantidade;
};
```

Neste template, como não sabemos o tipo do elemento a ser aplicado (pode ser aluno, veículo, despesa,...) deixamos a definição do tipo para depois

LISTA GENÉRICA

```
template <typename TIPO>
struct TElemento{
        TIPO dado;
};

template <typename TIPO, int MAX>
struct TLista{
        TElemento<TIPO> elementos
        int quantidade;
};
```

Neste template, como não sabemos o tipo do elemento a ser aplicado (pode ser aluno, veículo, despesa,...) e a capacidade máxima da lista

LISTA GENÉRICA

- Vamos ilustrar isto na função principal (main)

LISTA GENÉRICA

```
#include <iostream>
using namespace std;
#include "lista_generica.h"
#include "poker.h"

int main(){









        return 0;
}
```

Note que quando trabalhamos com uma lista genérica, temos que trabalhar com dois arquivos de bibliotecas: o da lista e o das especificidades do domínio

LISTA GENÉRICA

```
#include <iostream>
using namespace std;
#include "lista_generica.h"
#include "poker.h"

int main(){

        TLista <TCarta, 52> baralho;




        return 0;
}
```

Iremos utilizar o TLista ("lista_generica.h") para criar uma lista de TCartas ("poker.h")

LISTA GENÉRICA

```
#include <iostream>
using namespace std;
#include "lista_generica.h"
#include "poker.h"

int main(){

        TLista <TCarta, 52> baralho;




        return 0;
}
```

Note que, devido ao template, precisamos passar os *parâmetros do template*

Estabelecemos a criação de uma lista de 52 elementos TCarta

LISTA GENÉRICA

```
#include <iostream>
using namespace std;
#include "lista_generica.h"
#include "poker.h"

int main(){

        TLista <TCarta, 52> baralho;




        return 0;
}
```

Todas as operações que são padrões para uma lista (inicializar, remover, inserir, retornar elemento, ...) estão dentro da biblioteca "lista_generica.h"

LISTA GENÉRICA

```
#include <iostream>
using namespace std;
#include "lista_generica.h"
#include "poker.h"

int main(){

        TLista <TCarta, 52> l

        return 0;
}
```

Todas as operações relacionadas ao jogo de poker (embaralhar, distribuir, imprimir baralho), ficam dentro da biblioteca "poker.h"

LISTA GENÉRICA

```
#inc
usin
#inc
#inc

int

        return 0;
}
```

Vamos verificar a biblioteca “lista_generica.h”, para ver como ficam as funções #1

LISTA GENÉRICA

```
#ifndef __LISTA_EST_GENERICA__
#define __LISTA_EST_GENERICA__
template <typename TIPO>
struct TElemento{
        TIPO dado;
};

template <typename TIPO, int MAX>
struct TLista{
        TElemento<TIPO> elementos[MAX];
        int quantidade;
};

template <typename TIPO, int MAX>
void inicializa_lista_est_gen (TLista<TIPO, MAX> &l){
        l.quantidade = 0;
}

#endif
```

Como trabalha com uma estrutura genérica é necessário definir o template para as mesmas

LISTA GENÉRICA

Voltando ao **main**

LISTA GENÉRICA

```
#include <iostream>
using namespace std;
#include "lista_generica.h"
#include "poker.h"

int main(){

        TLista <TCarta, 52> baralho;
        inicializa_lista_est_gen (baralho);




        return 0;
}
```

Mesmo definindo templates para a função, não há necessidade de passarmos os "parâmetros do template", pois o parâmetro "baralho" já carrega as configurações definidas para o template

LISTA GENÉRICA

```
#inc
usin
#inc
#inc

int

        return 0;
}
```

Vamos verificar a biblioteca “lista_generica.h”, para ver como ficam as funções #2

```
#ifndef __LISTA_EST_GENERICA__
#define __LISTA_EST_GENERICA__
//...
template <typename TIPO, int MAX>
bool inserefim_lista_est_gen
                    (TLista<TIPO, MAX> &l,
                     TIPO dado){
      if(l.quantidade >= MAX){
            return false;
      }else{
            TElemento <TIPO> e;
            e.dado = dado;
            l.elementos[l.quantidade] = e;
            l.quantidade++;
            return true;
      }
}
//...
#endif
```

Note que temos um invólucro Telemento, que precisamos criar para acomodar o dado a ser inserido

LISTA GENÉRICA

Voltando ao **main**

LISTA GENÉRICA

```
#include <iostream>
using namespace std;
#include "lista_generica.h"
#include "poker.h"

int main(){

        Tlista <TCarta, 52> baralho;
        inicializa_lista_est_gen (ba      ho);
        TCarta ap;
        ap.naipe = 4;
        ap.valor = 14;
        inserefim_lista_est_gen (baralho, ap);
        return 0;
}
```

deste modo inserirmos um elemento a uma lista genérica

LISTA GENÉRICA

Visualizando benefícios da Lista Genérica

```
#include <iostream>
using namespace std;
#include "lista_generica.h"
#include "poker.h"
#include "carros.h"
#include "produtos.h"

int main(){

        Tlista <TCarta, 52> baralho;
        inicializa_lista_est_gen (baralho);
        TCarta ap;
        //...
        inserefim_lista_est_gen (baralho, ap);

        Tlista <TCarro, 45> carros;
        inicializa_lista_est_gen (carros);
        TCarro c;
        //...
        inserefim_lista_est_gen (carros, c);
        //...
        return 0;
}
```

```
#include <iostream>
using namespace std;
#include "lista_generica.h"
#include "poker.h"
#include "carros.h"
#include "produtos.h"

int main(){

        Tlista <TCarta, 52> baralho;
        inicializa_lista_est_gen (baralho);
        TCarta ap;
        //...
        inserefim_lista_est_gen (baralho, ap);

        Tlista <TCarro, 45> carros;
        inicializa_lista_est_gen (carros);
        TCarro c;
        //...
        inserefim_lista_est_gen (carros, c);
        //...
        return 0;
}
```

Com apenas uma implementação de lista, podemos utilizá-la para carros, poker, produtos, ...