



INSTITUTO FEDERAL

Norte de Minas Gerais

Campus Januária

Estruturas de Dados 2

- Generic Lists -



INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Situação Problema





INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Situação Problema

Imagine o desenvolvimento de uma engine para jogo MMO-RPG.





INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Situação Problema

Imagine o desenvolvimento de uma engine para jogo MMO-RPG.

Teremos que pensar em estruturas de dados para representar: Jogadores Online, Inimigos, Itens coletados, etc...





INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Situação Problema

Imagine o desenvolvimento de uma engine para jogo MMO-RPG.

Teremos que pensar em estruturas de dados para representar: Jogadores Online, Inimigos, Itens coletados, etc...

Estruturas Dinâmicas seriam essenciais para melhor desempenho.





Problema Atual

- **Problema Atual**: Existe um **forte acoplamento** entre a ***Entidade*** (objetos que queremos representar) e a ***Estrutura de Dados*** adotada para armazená-la (*p.ex.* Lista Encadeada).



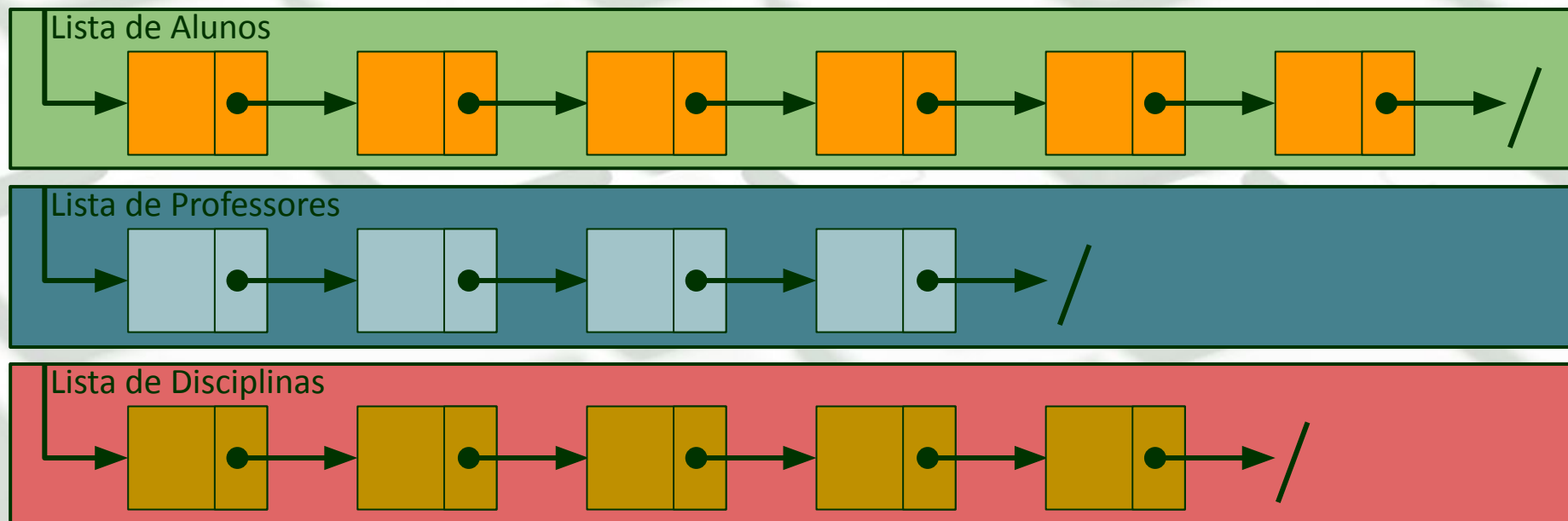
Problema Atual

- **Problema Atual**: Existe um **forte acoplamento** entre a **Entidade** (objetos que queremos representar) e a **Estrutura de Dados** adotada para armazená-la (*p.ex.* Lista Encadeada).
- Para tornar mais claro, imagine a seguinte solução...
Faça um programa que armazene, em listas dinâmicas, as seguintes estruturas: Aluno, Professor, Disciplina.
 - **Aluno** => Núm. Matrícula, Nome, Curso, etc...
 - **Professor** => CPF, Nome, Titulação, etc...
 - **Disciplina** => Código, Nome, Curso, Período, etc...



Problema Atual

- Perceba na solução que, **ao invés de implementar uma única Estrutura de Dados do tipo Lista Encadeada**, implementamos 03 tipos distintos de Lista Encadeada...





Problema Atual

- Perceba que as operações (algoritmos) para manutenção das listas são as mesmas, independentemente da **entidade**, mas implementamos cada operação em cada entidade separadamente...

cadastrarAluno()
<Insere na Lista>

consultarAluno()
<Consulta Lista>

editarAluno()
<Edita Lista>

excluirAluno()
<Exclui na Lista>

cadastrarProf()
<Insere na Lista>

consultarProf()
<Consulta Lista>

editarProf()
<Edita Lista>

excluirProf()
<Exclui na Lista>

cadastrarDisc()
<Insere na Lista>

consultarDisc()
<Consulta Lista>

editarDisc()
<Edita Lista>

excluirDisc()
<Exclui na Lista>



Problema Atual

- Perceba que as operações (algoritmos) para manutenção das listas são as mesmas, independentemente da **entidade**, mas implementamos cada operação em cada entidade separadamente... Isso é **ALTO ACOPLAMENTO**.

cadastrarAluno()
<Insere na Lista>

consultarAluno()
<Consulta Lista>

editarAluno()
<Edita Lista>

excluirAluno()
<Exclui da Lista>

cadastrarProf()
<Insere na Lista>

consultarProf()
<Consulta Lista>

editarProf()
<Edita Lista>

excluirProf()
<Exclui da Lista>

cadastrarDisc()
<Insere na Lista>

consultarDisc()
<Consulta Lista>

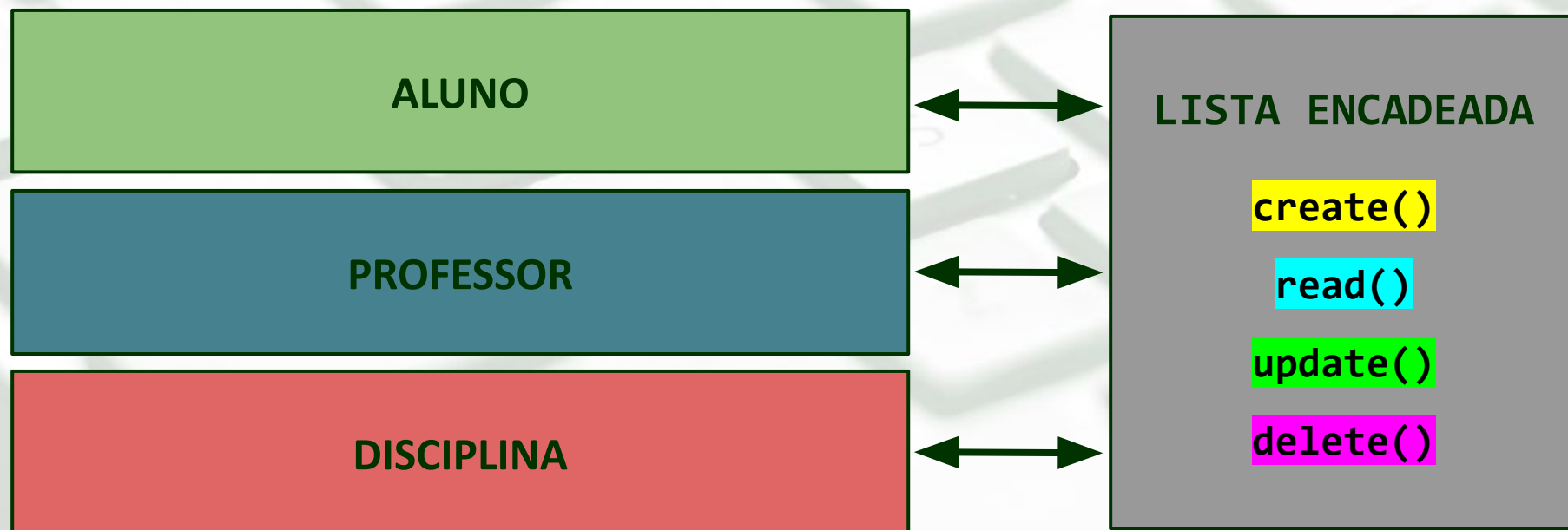
editarDisc()
<Edita Lista>

excluirDisc()
<Exclui da Lista>



Problema Atual

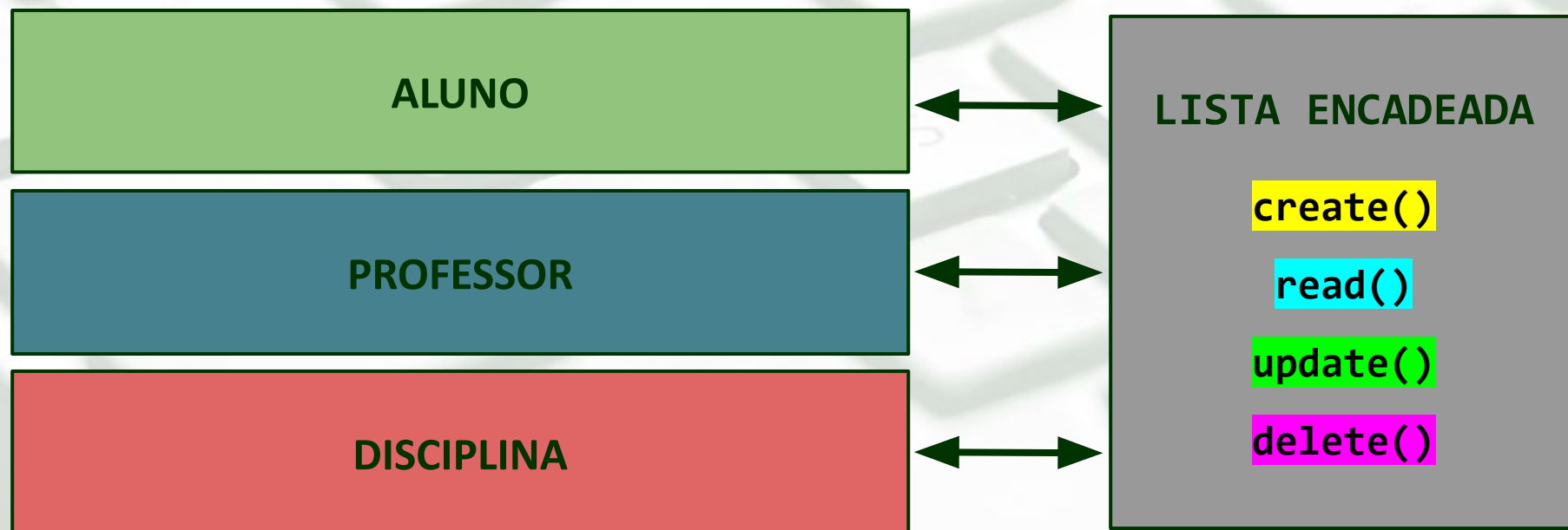
- “No mundo ideal” o mais interessante é que as operações sobre a Estrutura de Dados **Lista Encadeada** (controle) sejam **INDEPENDENTES** às entidades que a utiliza (modelo).





Listas Genéricas

- Nesta proposta, temos que implementar um tipo de **Lista Genérica**, onde todas as **operações podem ser compartilhadas por qualquer entidade**.





1º Desafio

- Como evitar o alto acoplamento sendo que, na própria entidade definimos o tipo de lista a ser utilizada...

```
typedef struct Aluno{  
    int numMatricula;  
    char nome[100];  
    char curso[100];  
    struct Aluno *prox;  
}Aluno;
```



1º Desafio

- Como evitar o alto acoplamento sendo que, na própria entidade definimos o tipo de lista a ser utilizada...

```
typedef struct Aluno{  
    int numMatricula;  
    char nome[100];  
    char curso[100];  
    struct Aluno *prox;  
}Aluno;
```

JÁ COMEÇOU ERRADO!

Se a ideia é separar a entidade (Aluno) da estrutura de dados (Lista Encadeada) não podemos deixar esses dois conceitos tão acoplados!



Objetos Genéricos

■ Como definir então um objeto genérico???

```
typedef struct Node{  
    void* item;  
    struct Node* next;  
    struct Node* prev;  
}Node;
```



Objetos Genéricos

■ Como definir então um objeto genérico???

```
typedef struct Node{  
    void* item;  
    struct Node* next;  
    struct Node* prev;  
}Node;
```





Objetos Genéricos

■ Como definir então um objeto genérico???

```
typedef struct Node{  
    void* item;  
    struct Node* next;  
    struct Node* prev;  
}Node;
```

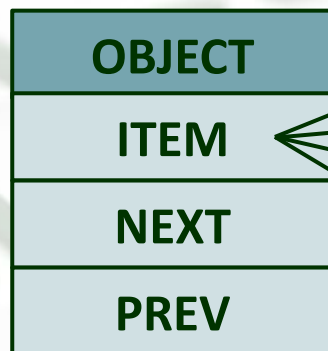
```
typedef Node* Object;
```

*Como criaremos **NODE's** de forma **dinâmica**, chamaremos esse tipo de **Object***



Objetos Genéricos

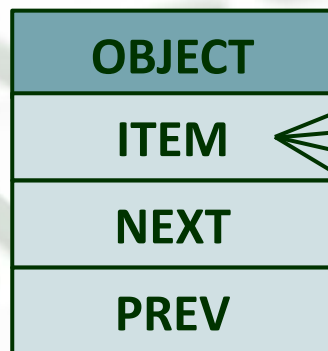
OBJECT é portanto, um PONTEIRO para uma struct NODE, que por sua vez, aponta para uma estrutura genérica útil!





Objetos Genéricos

OBJECT é portanto, um PONTEIRO para uma struct NODE, que por sua vez, aponta para uma estrutura genérica útil!



PS... Já pensou como é possível existir linguagens de tipagem fraca???



Objetos Genéricos

OBJECT é portanto, um PONTEIRO para uma struct NODE, que por sua vez, aponta

Clique e leia depois...

How Python objects are implemented in C?



Xinzhe Li, PhD in Language Intelligence · Follow
2 min read · Jul 10, 2022

PREV

PS... Já pensou como é possível existir linguagens de tipagem fraca???





Objetos Genéricos

■ Construindo **Objetos Genéricos** através de uma Interface Única...

```
#define new(TYPE) new_##TYPE()
```

```
Object new_Node(){  
    //...  
}
```

```
Object new_Professor(){  
    Object obj = new(Node);  
    //...  
}
```

```
Object new_Disciplina(){  
    Object obj = new(Node);  
    //...  
}
```

```
int main(){  
    Object prof = new(Professor);  
    Object disc = new(Disciplina);  
}
```



2º Desafio

- Mas um único objeto por si só não é capaz de definir toda uma lista...

DESACOPLAMENTO

ALUNO

PROFESSOR

DISCIPLINA

LISTA ENCADEADA

`create()`

`read()`

`update()`

`delete()`



Listas Genéricas

■ Como definir então uma...

LISTA de OBJETOS genéricos

```
typedef struct Node{  
    void* item;  
    struct Node* next;  
    struct Node* prev;  
}Node;
```

```
typedef Node* Object;
```

Protótipo Inicial de uma Lista Genérica

```
typedef struct{  
    Object init; //início da lista  
    int size; //tamanho da lista  
    //outros dados da lista que desejar  
    //armazenar para acesso direto.  
}List;
```




Listas Genéricas

■ Repositório oficial do Python (List Object)

cpython / Include / listobject.h

 vstinner gh-114329: Fix PyList_GetItemRef() limited C API definition (#117520)

Code

Blame

55 lines (44 loc) · 1.87 KB



Raw



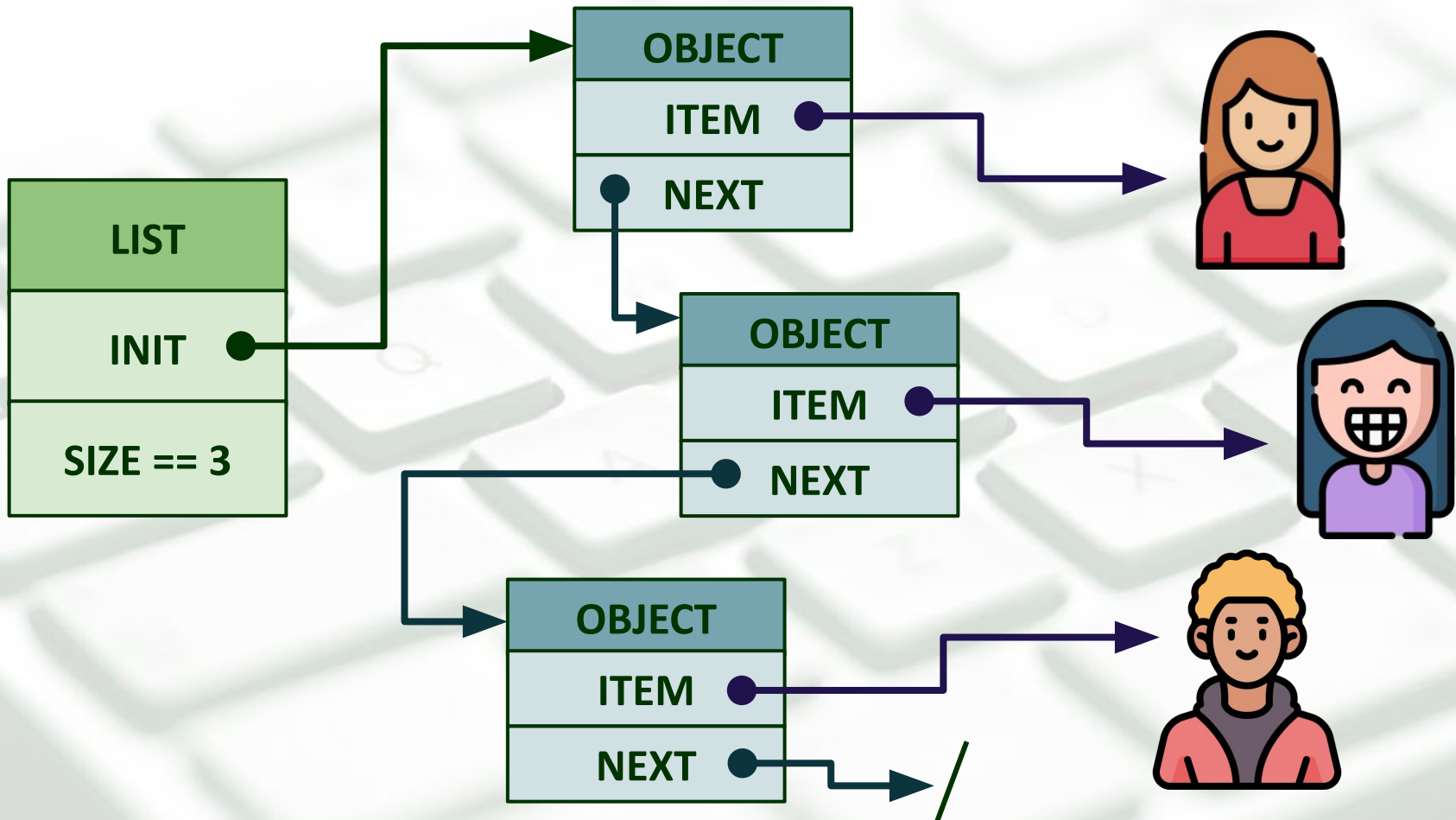
```
1  /* List object interface
2
3  Another generally useful object type is a list of object pointers.
4  This is a mutable type: the list items can be changed, and items can be
5  added or removed. Out-of-range indices or non-list objects are ignored.
6
7  WARNING: PyList_SetItem does not increment the new item's reference count,
8  but does decrement the reference count of the item it replaces, if not nil.
9  It does *decrement* the reference count if it is *not* inserted in the list.
10 Similarly, PyList_GetItem does not increment the returned item's reference
11 count.
12 */
```





Listas Genéricas

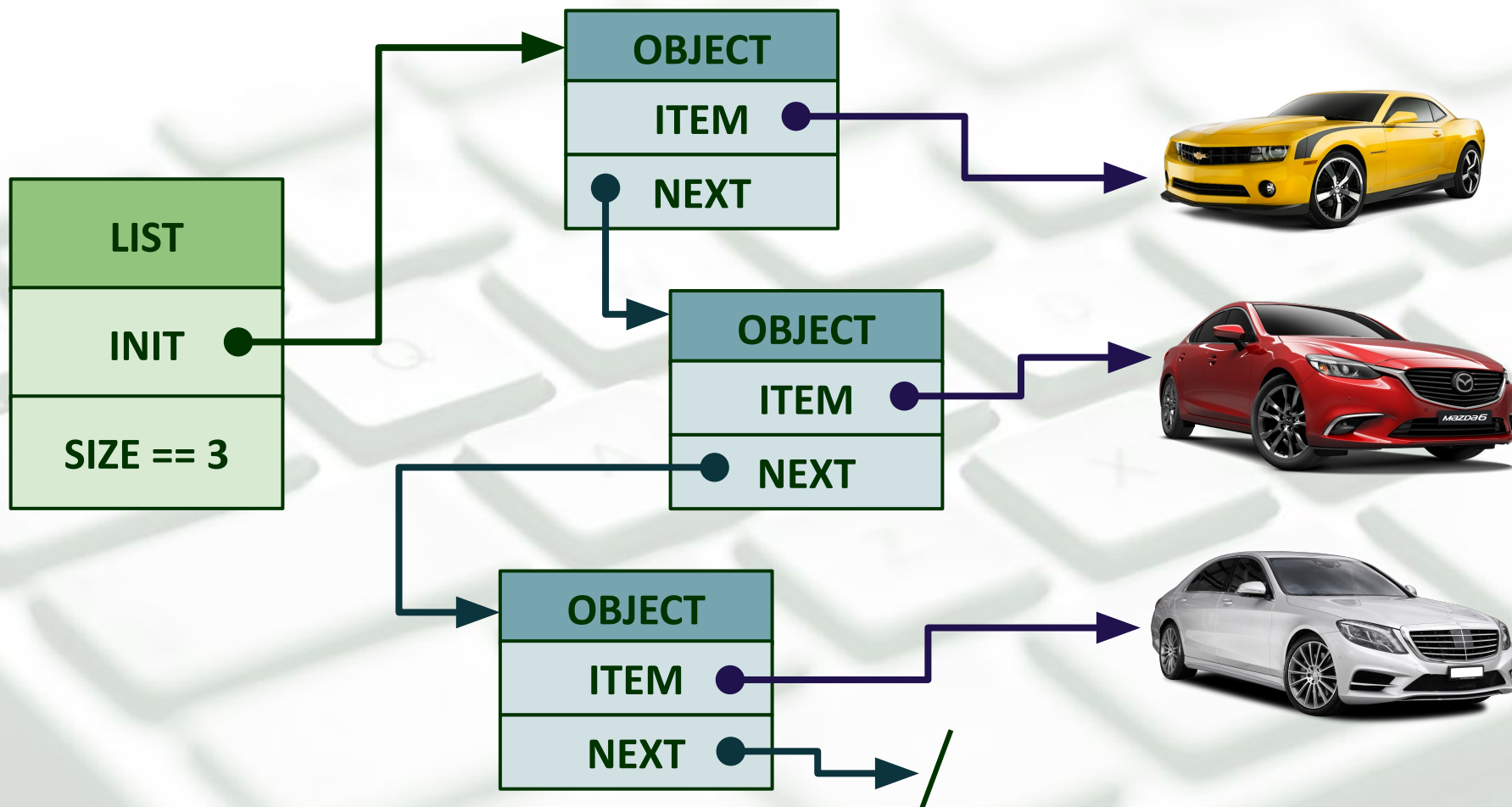
■ Projeto de Implementação de Lista Genérica...





Listas Genéricas

■ Projeto de Implementação de Lista Genérica...





Implementação Real

■ Exemplo de implementação em Linguagens de alto nível

```
using namespace std;

int main() {
    std::list<Carro> cars;

    cars.push_front(Carro("Territory", "Ford"));
    cars.push_front(Carro("T-Cross", "VW"));
    cars.push_front(Carro("Toro", "Fiat"));
    cars.push_front(Carro("Hilux", "Toyota"));

    for (const auto & c : cars)
        cout << c << endl;

    return 0;
}
```



Implementação Real

■ Exemplo de implementação em Linguagens de alto nível

```
import java.util.LinkedList;  
  
public class Main{  
    public static void main(String[] args) {  
        LinkedList<Carro> cars = new LinkedList<Carro>();  
        cars.add(new Carro("Hilux", "Toyota"));  
        cars.add(new Carro("Toro", "Fiat"));  
        cars.add(new Carro("T-Cross", "VW"));  
        cars.add(new Carro("Territory", "Ford"));  
        System.out.println(cars);  
    }  
}
```



Implementação Real

■ Exemplo de implementação em Linguagens de alto nível

```
from collections import deque

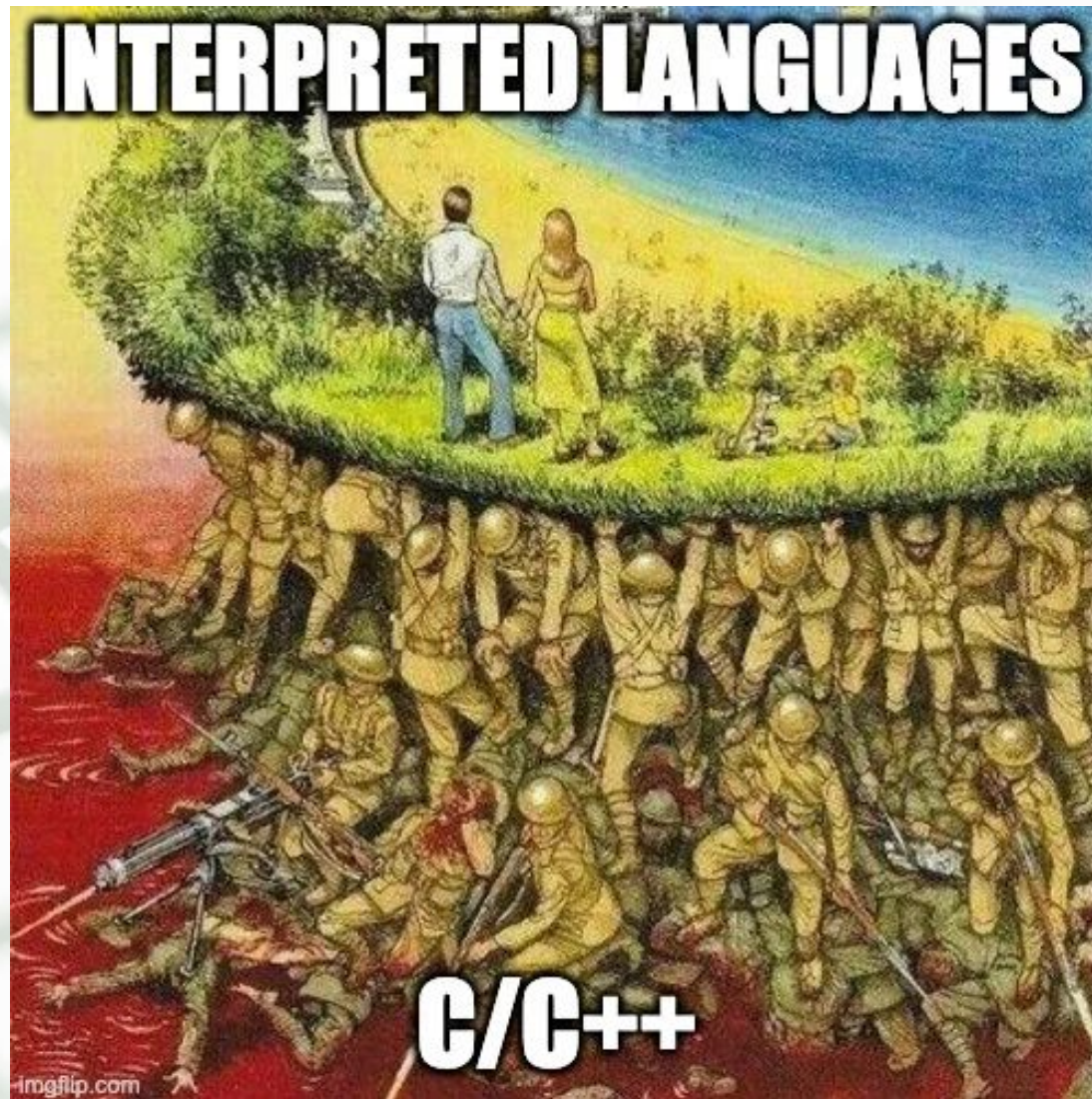
def main():
    cars = deque()    #DEQUE => Double-Ended Queue
    cars.append(Carro("Hilux", "Toyota"))
    cars.append(Carro("Toro", "Fiat"))
    cars.append(Carro("T-Cross", "VW"))
    cars.append(Carro("Territory", "Ford"))

    print(list(cars))
```




INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Implementação Real





Bora CODAR!!!



- Crie duas LIBs, contendo as seguintes funções...

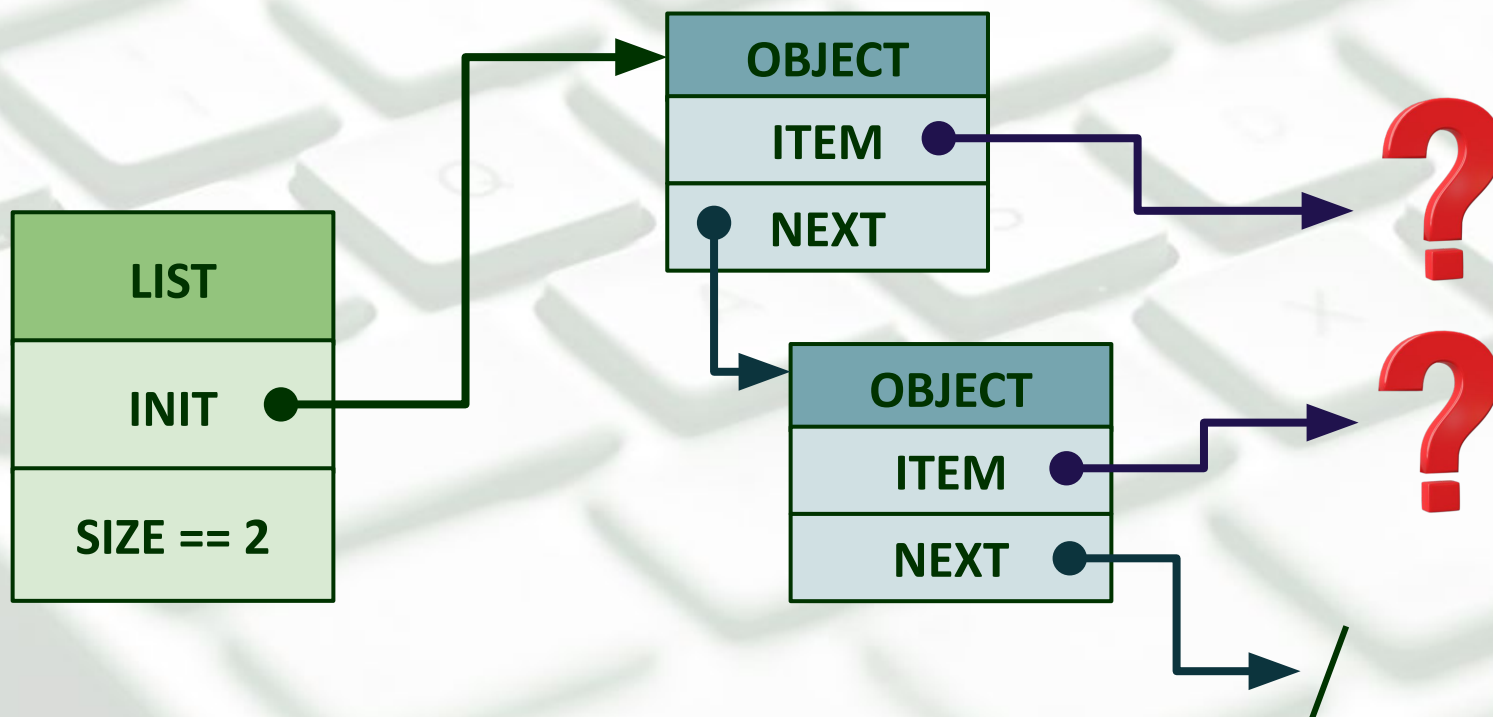
object.h	list.h
Definição da Struct Node	Definição do Tipo List
Definição do Tipo Object	void list_push()
Object new_Object()	void list_pop()
Object destroy()	void list_print()

Para testar as LIBs, defina uma **struct Carro** (Modelo, Marca, Ano) e use a **Lista Genérica como Estrutura de Dados** para armazenar uma quantidade indefinida de Carros.



3º Desafio

- Como os itens de uma Lista Genérica são “**indefinidos**”, como executar funções específicas (*p.ex. imprimir os dados do Nó*) para estes itens?





3º Desafio

- Como o problema é "indefinido" (p.ex. i

Pesquise qual é a melhor forma de resolver esse problema.

íficas
tens?

