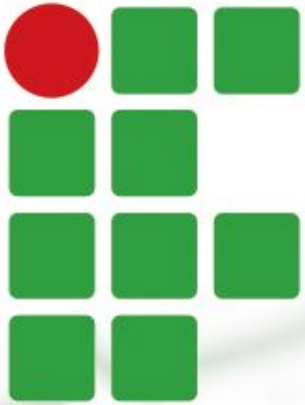


INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Estruturas de Dados 2

- Generic Lists -

Prof. MSc. Adriano Antunes Prates

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).

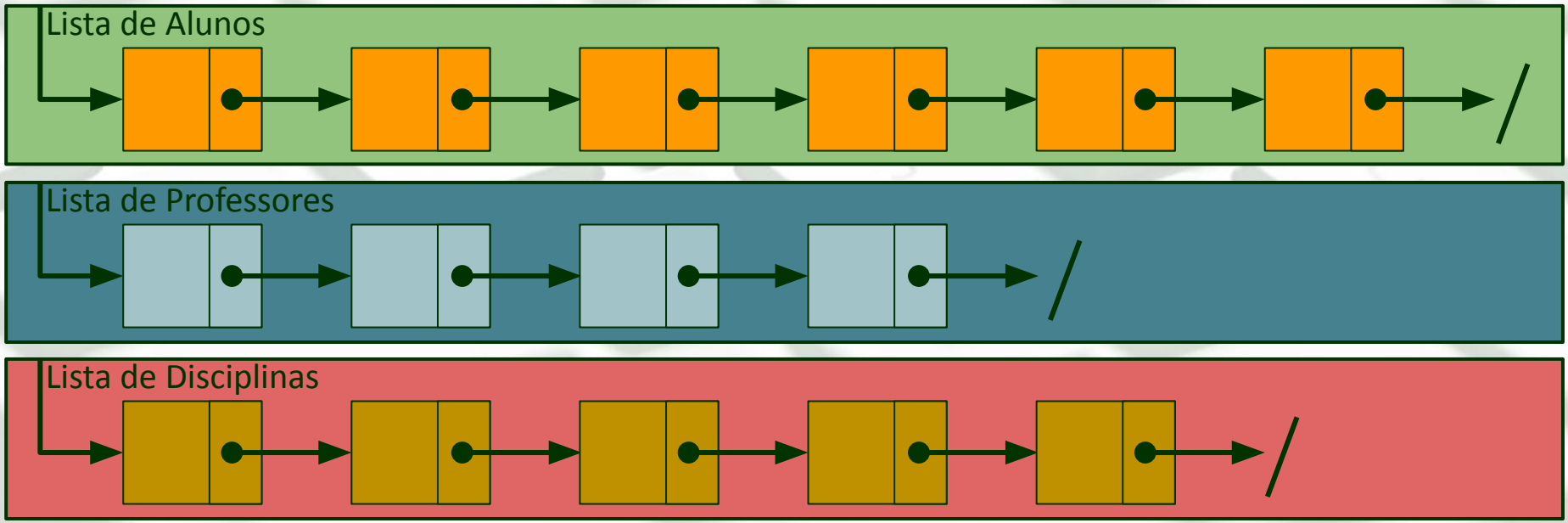
- Para tornar mais claro, imagine a seguinte solução…

  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…

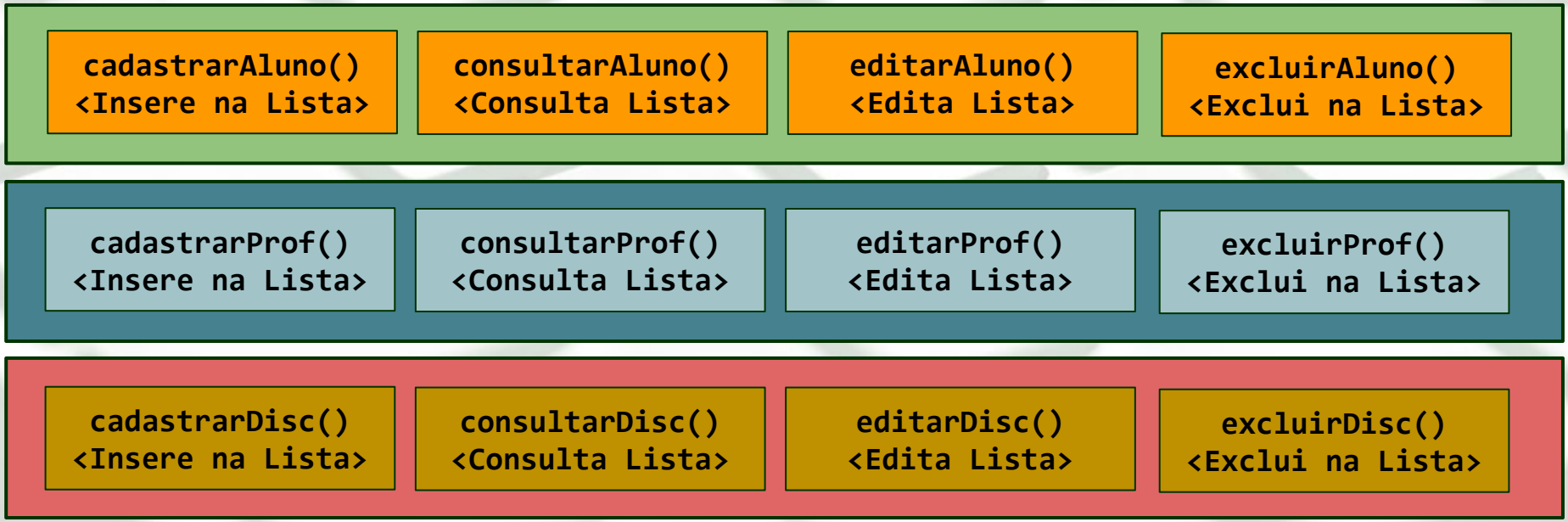
Problema Atual

- 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

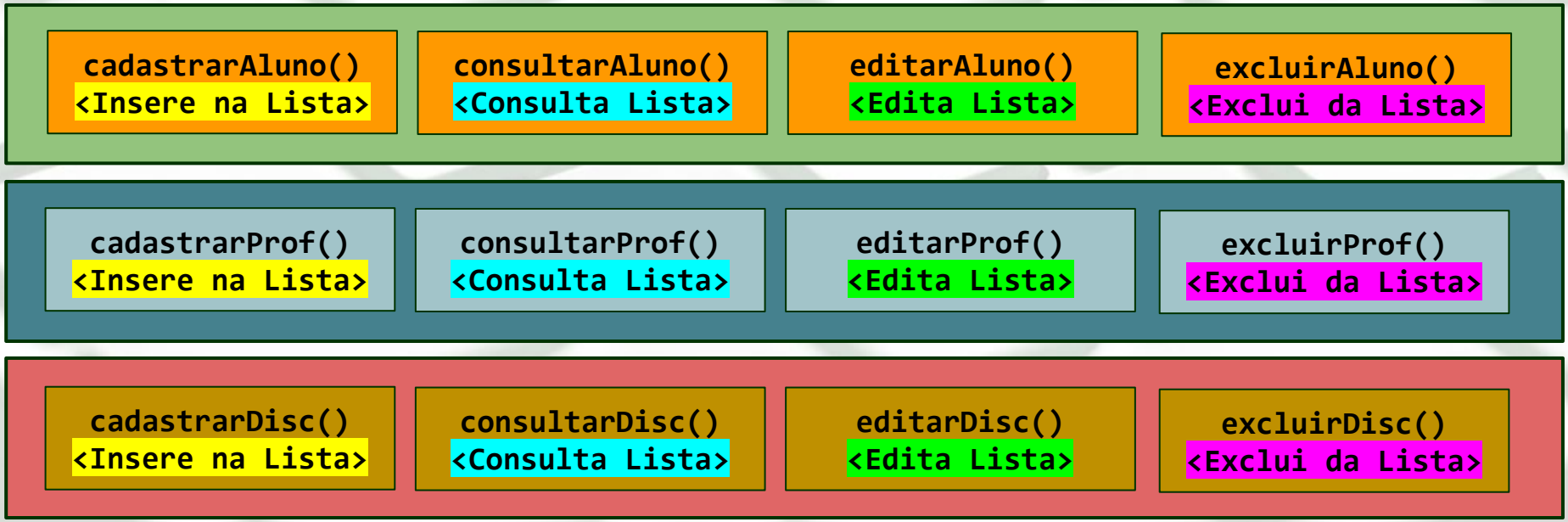
- **Perceba que as operações (algoritmos) para manutenção das listas são as mesmas, independentemente da entidade**, mas implementamos cada operação em cada entidade separadamente…



INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Problema Atual

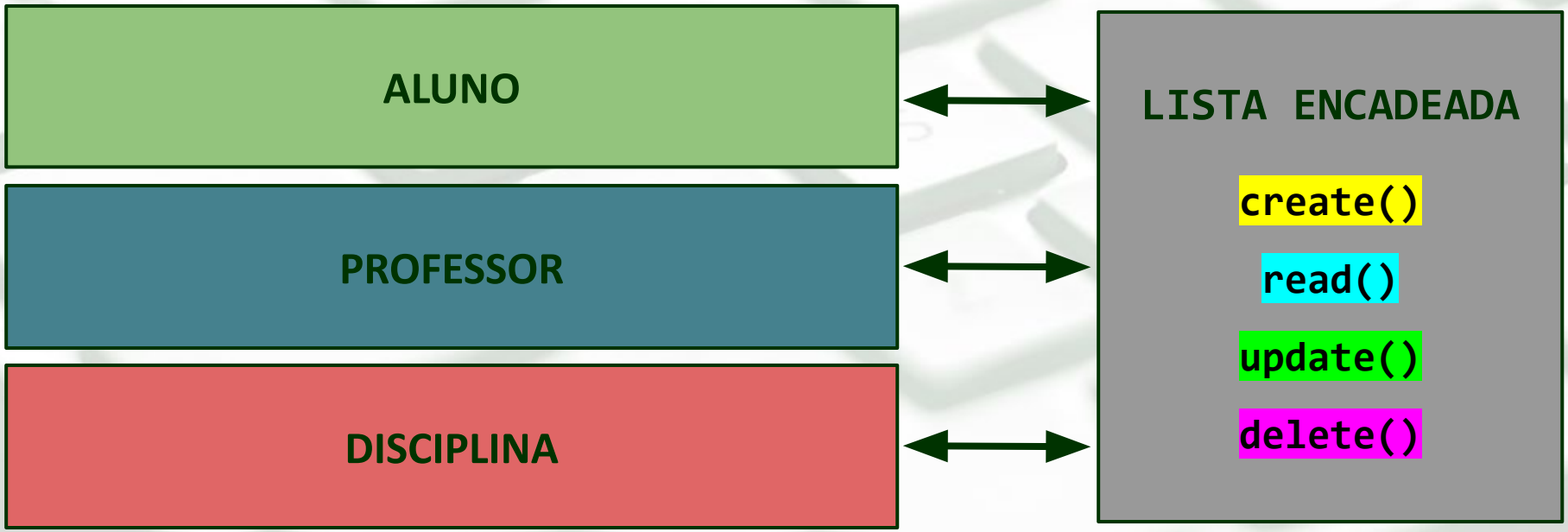
- **Perceba que as operações (algoritmos) para manutenção das listas são as mesmas, independentemente da entidade**, mas implementamos cada operação em cada entidade separadamente… Isso é **ALTO ACOPLAMENTO.**



INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Problema Atual

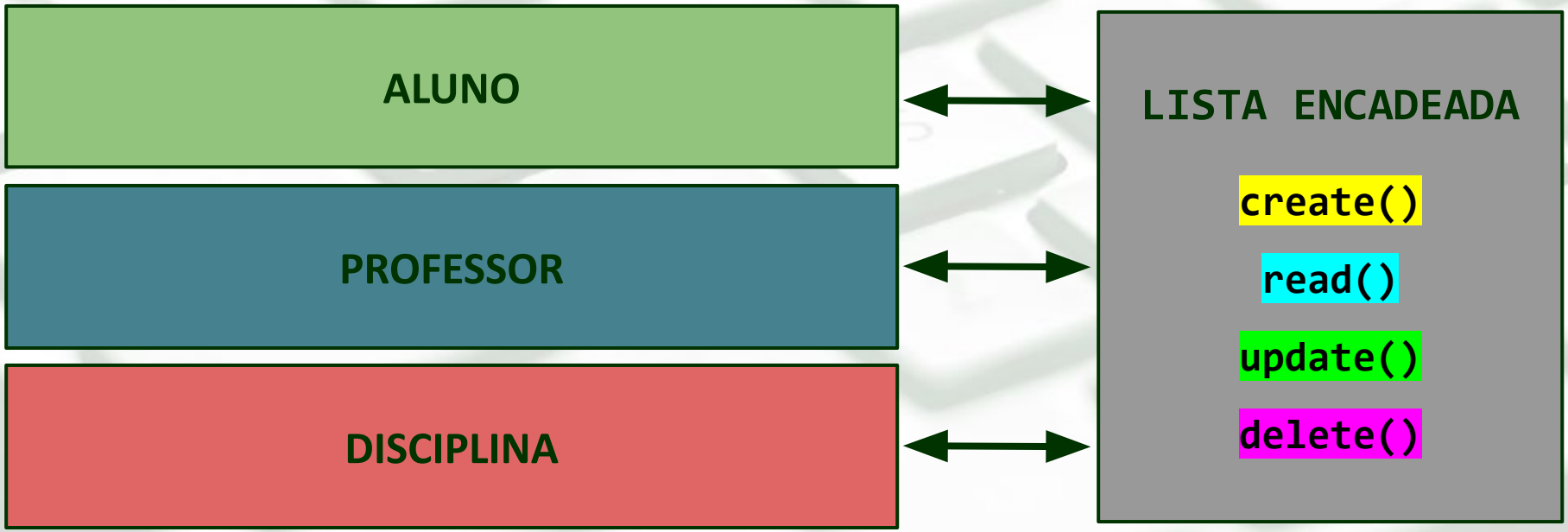
- "No mundo ideal" o mais interessante é que as operações sobre a Estrutura de Dados **Lista Encadeada** (controle) sejam **INDEPENDENTES** às entidades que a utiliza (modelo).



INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

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;
```

INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

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;
```

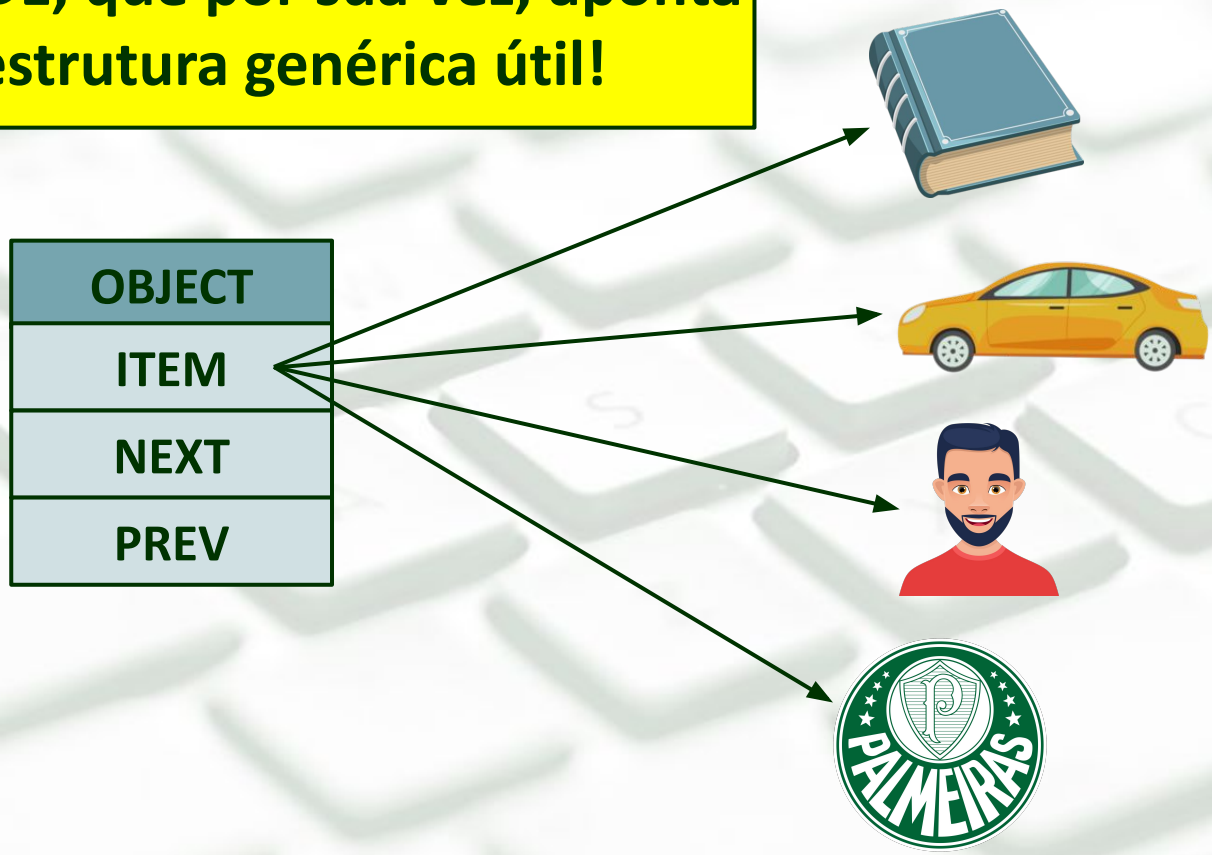
```
typedef Node* Object;
```

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

INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Objetos Genéricos

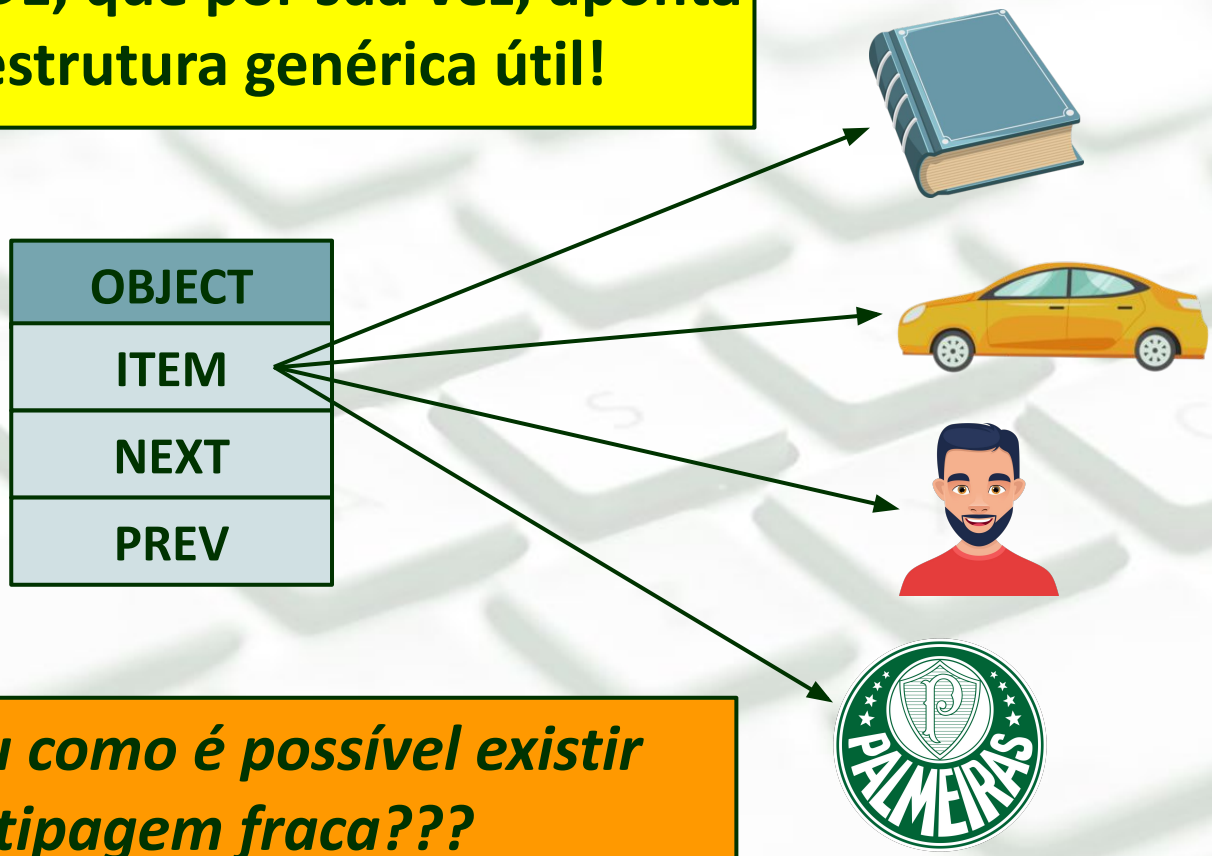
OBJECT é portanto, um PONTEIRO para uma struct NODE, que por sua vez, aponta para uma estrutura genérica útil!



INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

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???

INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

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...

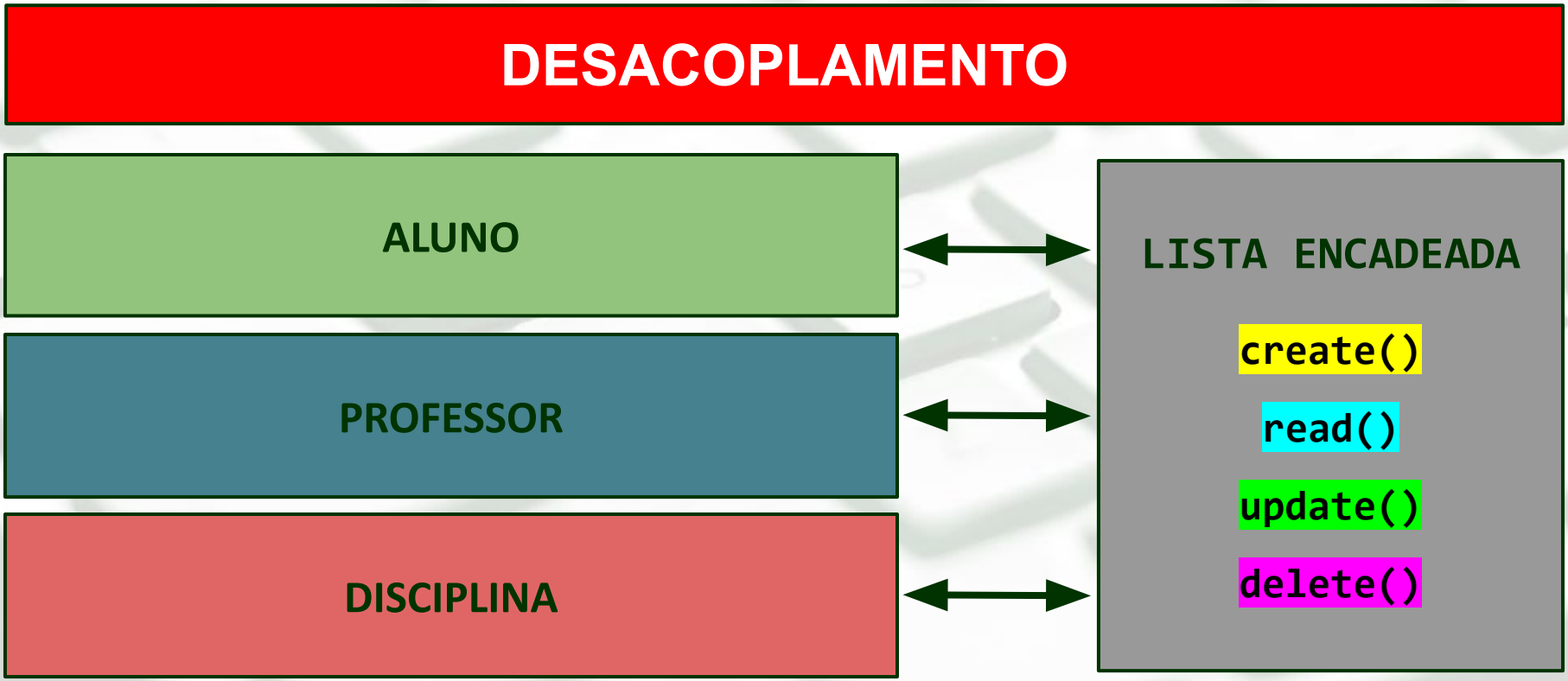
```
#define new(TYPE) new_##TYPE()

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);
}
```

INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

2º Desafio

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



INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

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
    Object end; //Final da lista
    int size;   //tamanho da lista
       //outros dados da lista que desejar
       //armazenar para acesso direto.
}List;
```

INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Listas Genéricas

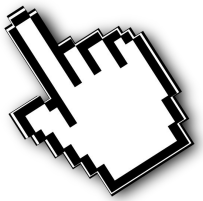
- **Repositório oficial do Python (List Object)**

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

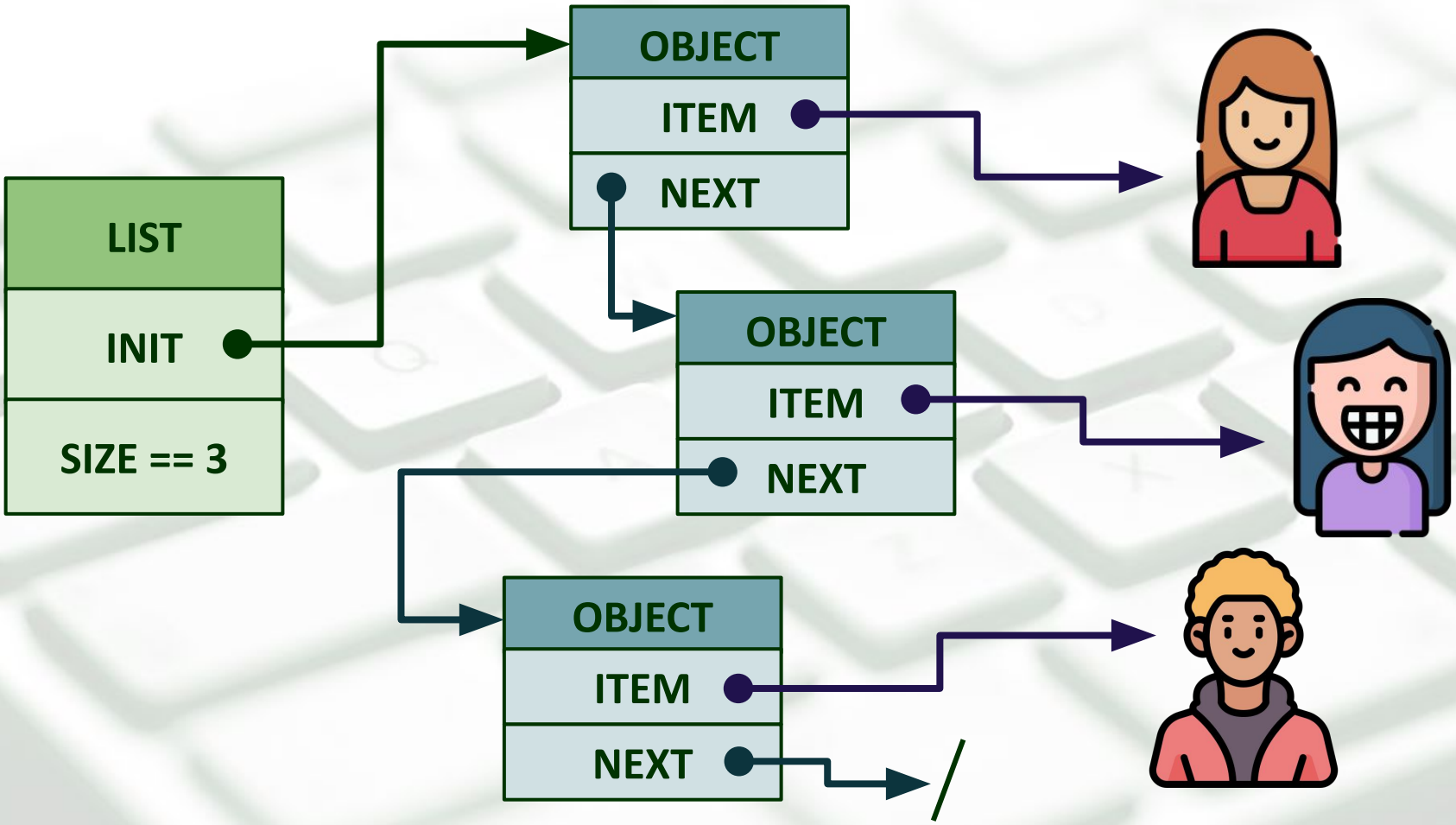
```
1   /* List object interface
2
3       Another generally useful object type is a list of object pointers.
4       This is a mutable type: the list items can be changed, and items can be
5       added or removed. Out-of-range indices or non-list objects are ignored.
6
7       WARNING: PyList_SetItem 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  */
```



INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Listas Genéricas v1.0

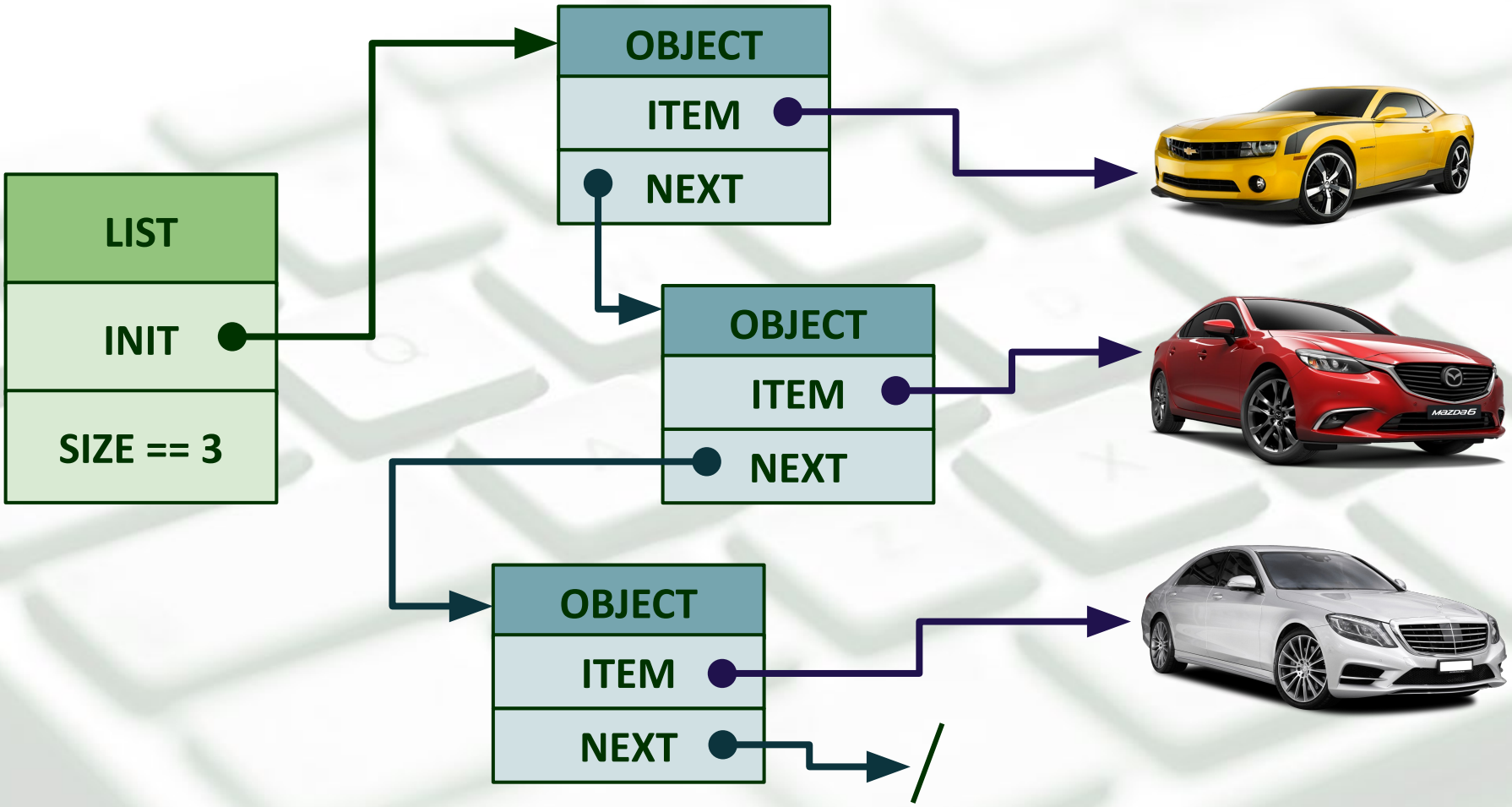
- **Projeto de Implementação de Lista Genérica…**



INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Listas Genéricas v1.0

- **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

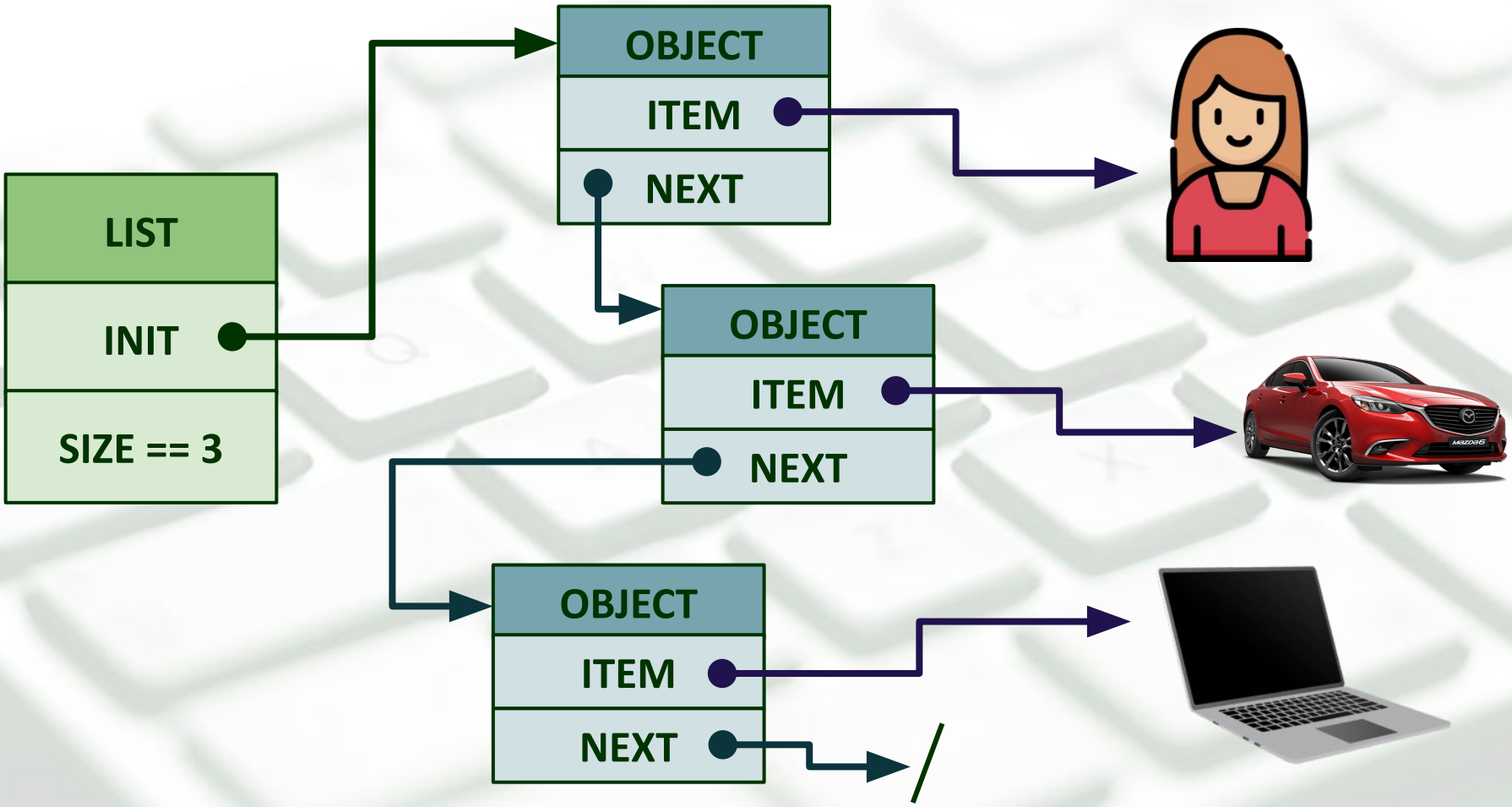
- **Exemplo de implementação em Linguagens de alto nível**

```
from collections import deque

def main():
    lst = deque()   #DEQUE => Double-Ended Queue
    lst.append(Carro("Hilux", "Toyota"))
    lst.append(Pessoa("João", 30))
    lst.append(Carro("T-Cross", "VW"))
    lst.append(Pessoa("Maria", "25"))

    print(list(lst))
```

INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Listas Genéricas v2.0

- **Projeto de Implementação de Lista Genérica…**



Conceito de Abstração

- **Abstração** é o processo de **ocultar detalhes complexos** de um sistema, **expondo apenas as funcionalidades essenciais**.
- **Abstração** permite gerenciar a complexidade, criando sistemas mais modulares, com melhor produtividade e mais fáceis de compreender.

Para dirigir um carro, você não precisa saber exatamente como o motor funciona internamente.

O volante, pedais e câmbio são abstrações que permitem a interação e uso do sistema (carro).

Conceito de Abstração

- Entretanto, camadas de abstração sempre introduzem:
  - **Maior *Overhead*:** e consumo de recursos, devido a necessidade de tradução/conversão das chamadas (de algo simples => algo complexo).
  - **Perda do "controle fino" das operações:** Ao usar abstrações, o desenvolvedor perde o controle fino sobre as otimizações que poderiam ser feitas em baixo nível. Quanto maior o nível de abstração (alto nível), mais complexo/impraticável é descer ao baixo nível.

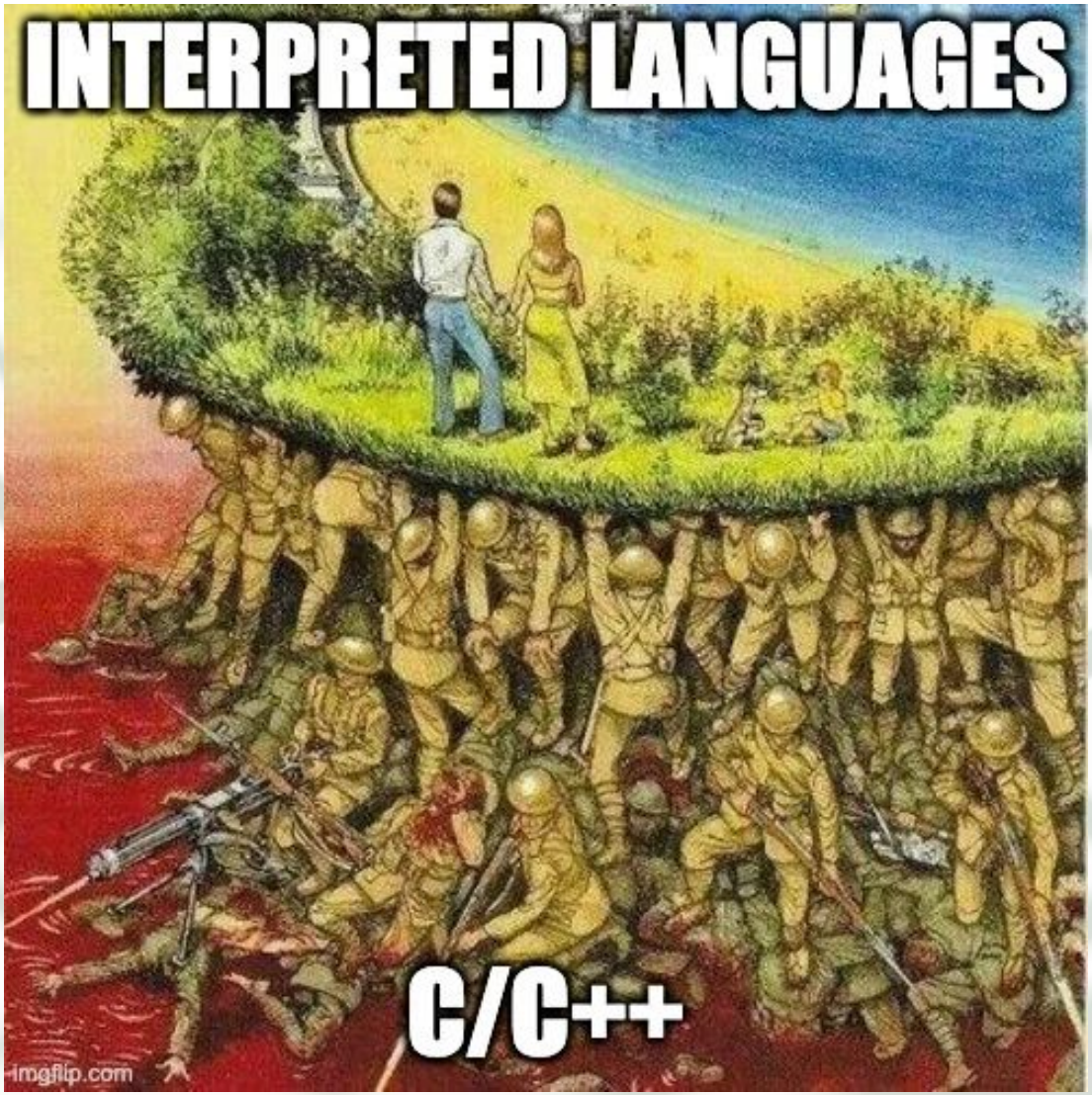
INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Conceito de Abstração

Produtividade

vs.

Overhead



INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

Bora CODAR!!!

- **Lista de Objetos Genéricos v1.0**
  - Lista duplamente encadeada, podendo ser tratada como Pilha ou Fila.

object.h	list.h
Definição da Struct **Node**	Definição do Tipo **List**
Definição do Tipo **Object**	**void list_push()**
Object new_Object()	**void list_enqueue()**
Object destroy()	**Object list_pop()**
entidade.h	**Object list_dequeue()**
Definição de uma Entidade e suas funções para validação da lista.	**void list_print()**

Bora CODAR!!!

- **Requisitos Técnicos...**

- Uso de ponteiros deverá ser transparente (SEM necessidade de *)
- Interface padronizada para construção de objetos dinâmicos.

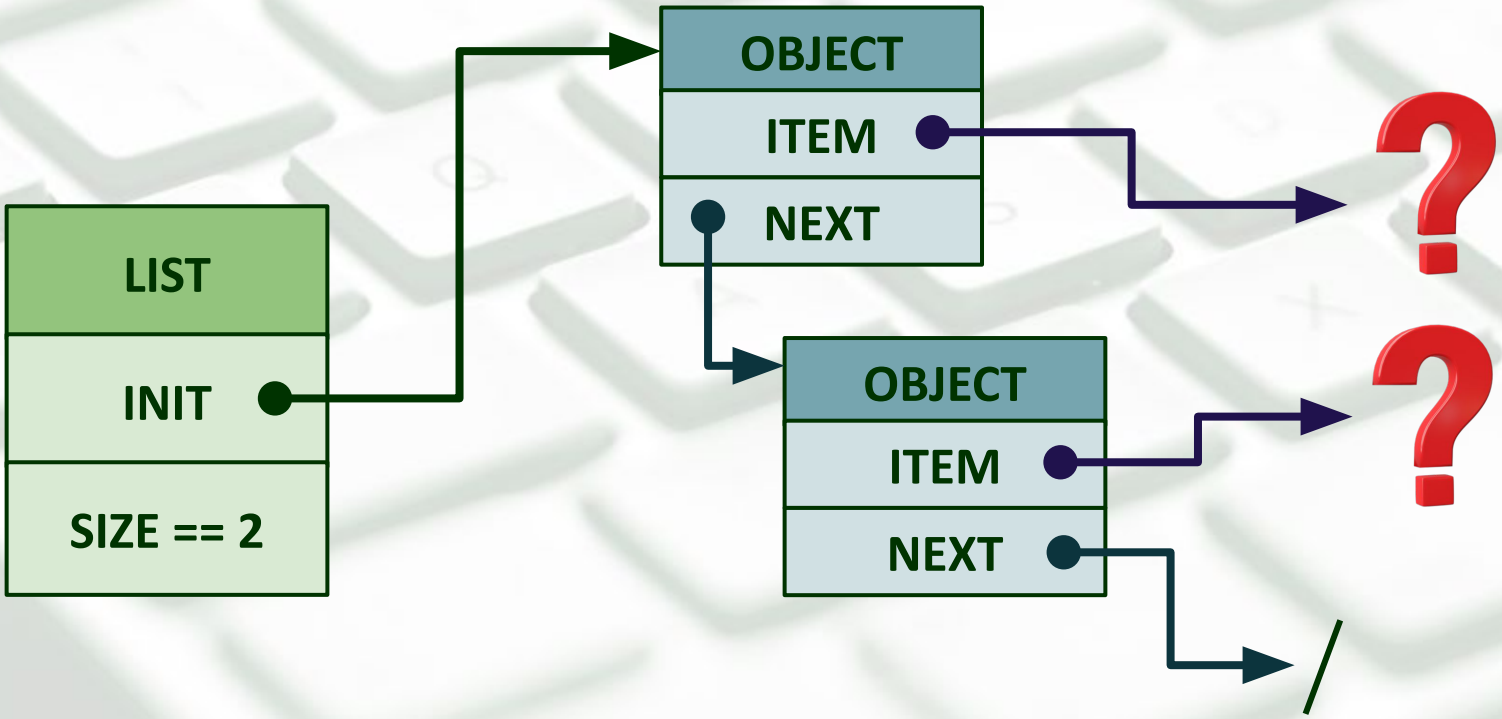
```
int main(){
     List* lst = new_List();
     Object* d1 = new_Disciplina();
     setDisciplina(d1,"ED2",3,80);
     list_push(lst,d1);
     d1 = new_Disciplina();
     setDisciplina(d1,"R2",5,80);
     list_push(lst,d1);
     Object* ob = list_pop(lst);
     printDisciplina(ob);
     destroy(ob);
     list_print(lst);
}
```

```
int main(){
     List lst = new(List);
     Object d1 = new(Disciplina);
     setDisciplina(d1,"ED2",3,80);
     list_push(lst,d1);
     d1 = new(Disciplina);
     setDisciplina(d1,"R2",5,80);
     list_push(lst,d1);
     Object ob = list_pop(lst);
     printDisciplina(ob);
     destroy(ob);
     list_print(lst);
}
```

INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

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**?



INSTITUTO FEDERAL
Norte de Minas Gerais
Campus Januária

3º Desafio

- Como
  "*indefi* íficas
  (*p.ex. i* tens?

Pesquise qual é a melhor forma de resolver esse problema.