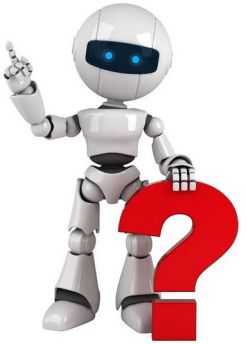
Algoritmos e Estruturas de Dados III

“Listas Encadeadas”

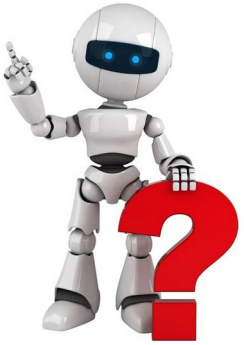
Prof. Dr. Felipe Oliveira

Estruturas de Dados



- Estratégias para organização de dados, permitindo que algoritmos manipulem os dados de maneira mais eficiente.

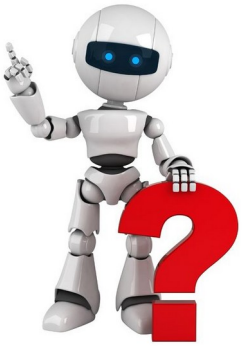
Estruturas de Dados



- Listas encadeadas.

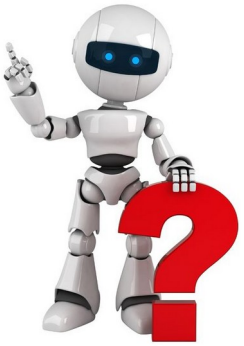
Dinâmicas

Estruturas de Dados

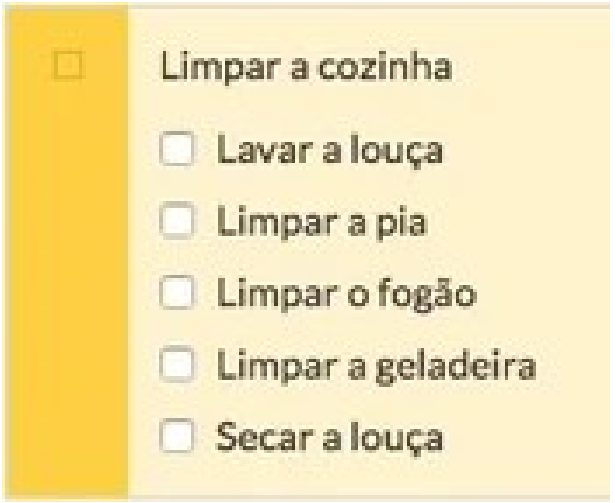


- Lista:
  - Conjunto de elementos, objetos, variáveis, ou qualquer coisa que se possa enumerar e formar um conjunto;

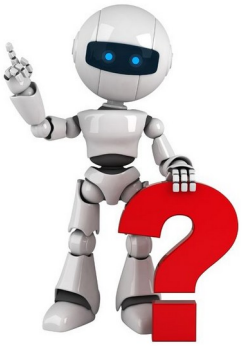
Estruturas de Dados



- Lista:
  - As listas estão presentes em nossa vida.

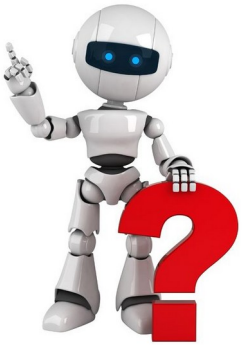


Estruturas de Dados

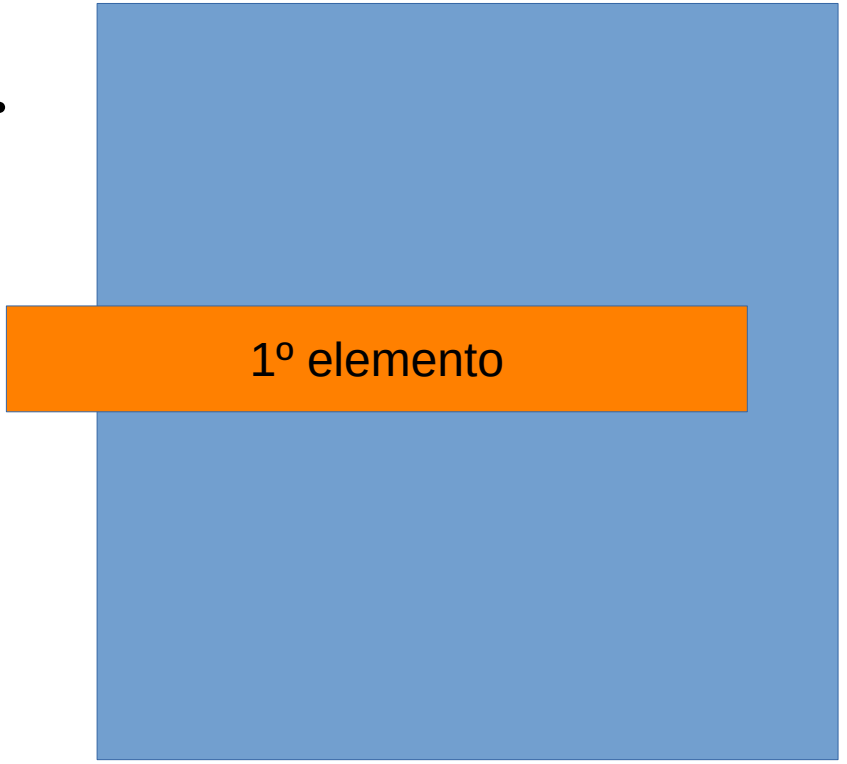


- Comportamento de uma lista:
  - `Lista: vazia.`

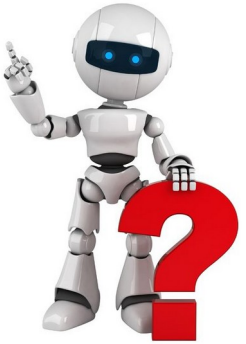
Estruturas de Dados



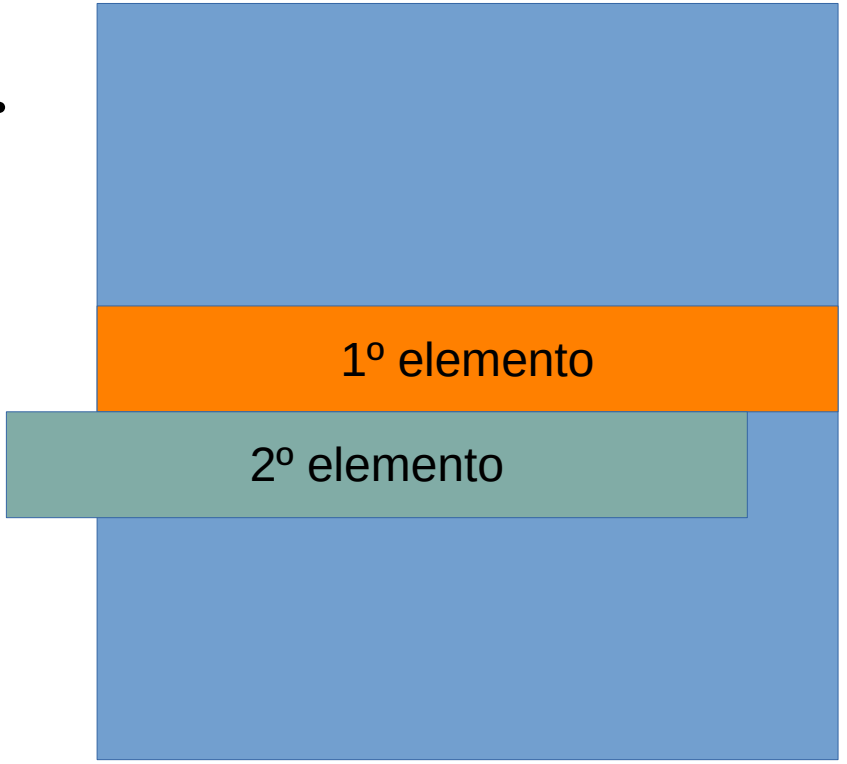
- Comportamento de uma lista:
  - `Lista:`
    - `Inserir 1° elem.`



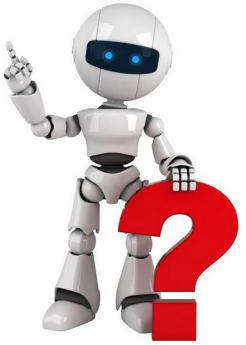
Estruturas de Dados



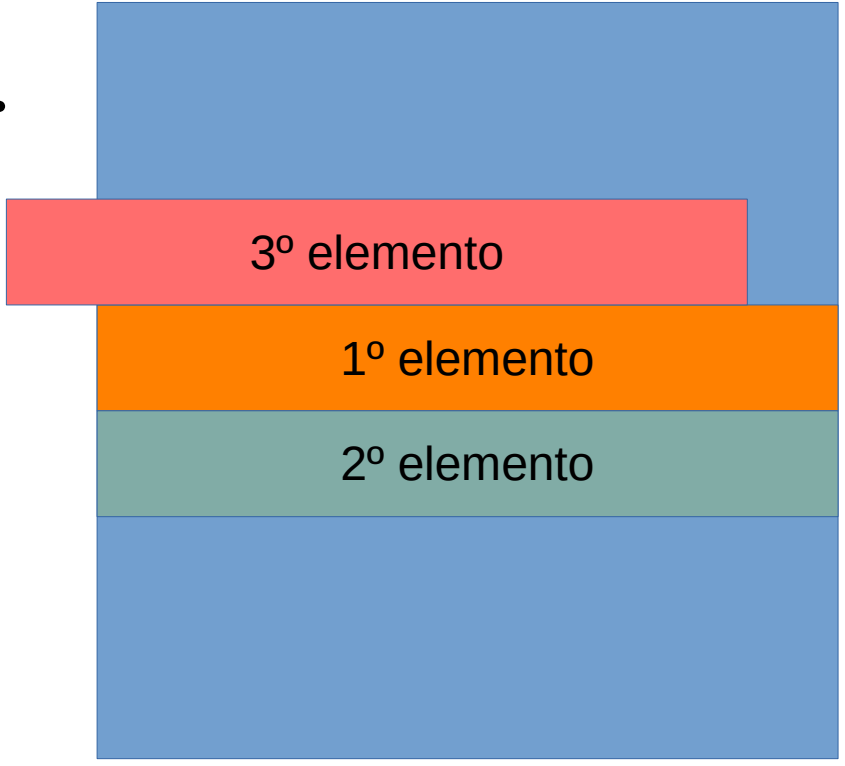
- Comportamento de uma lista:
  - `Lista:`
    - `Inserir 2° elem.`



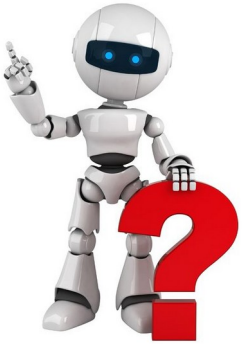
Estruturas de Dados



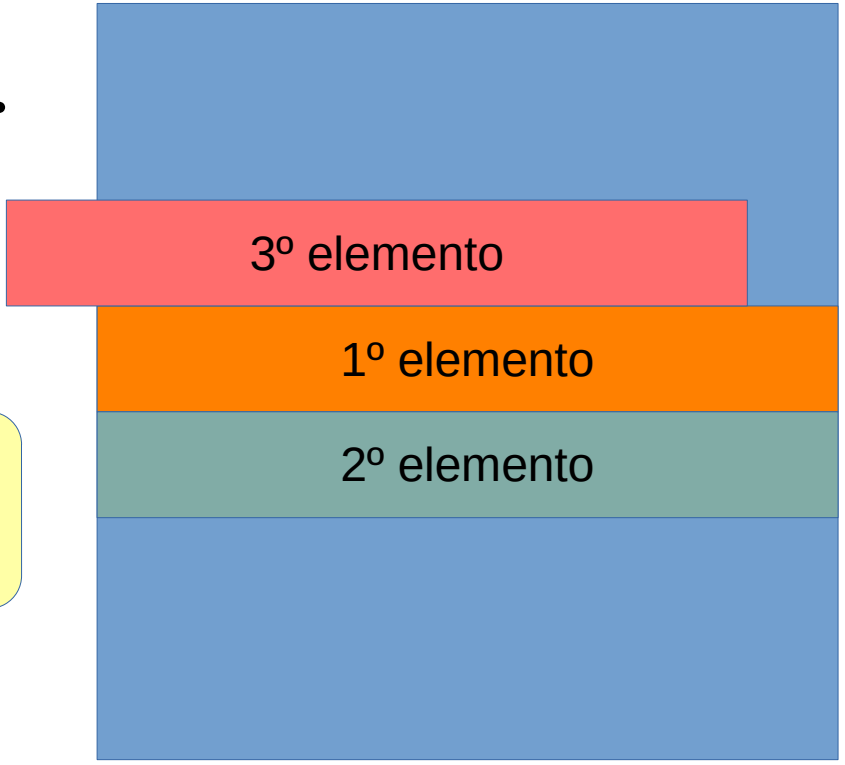
- Comportamento de uma lista:
  - `Lista:`
    - `Inserir 3° elem.`



Estruturas de Dados

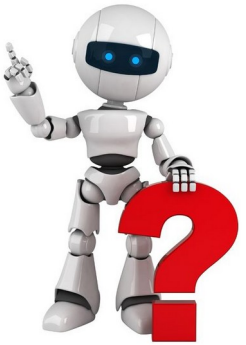


- Comportamento de uma lista:
  - `Lista:`
    - `Inserir 3° elem.`

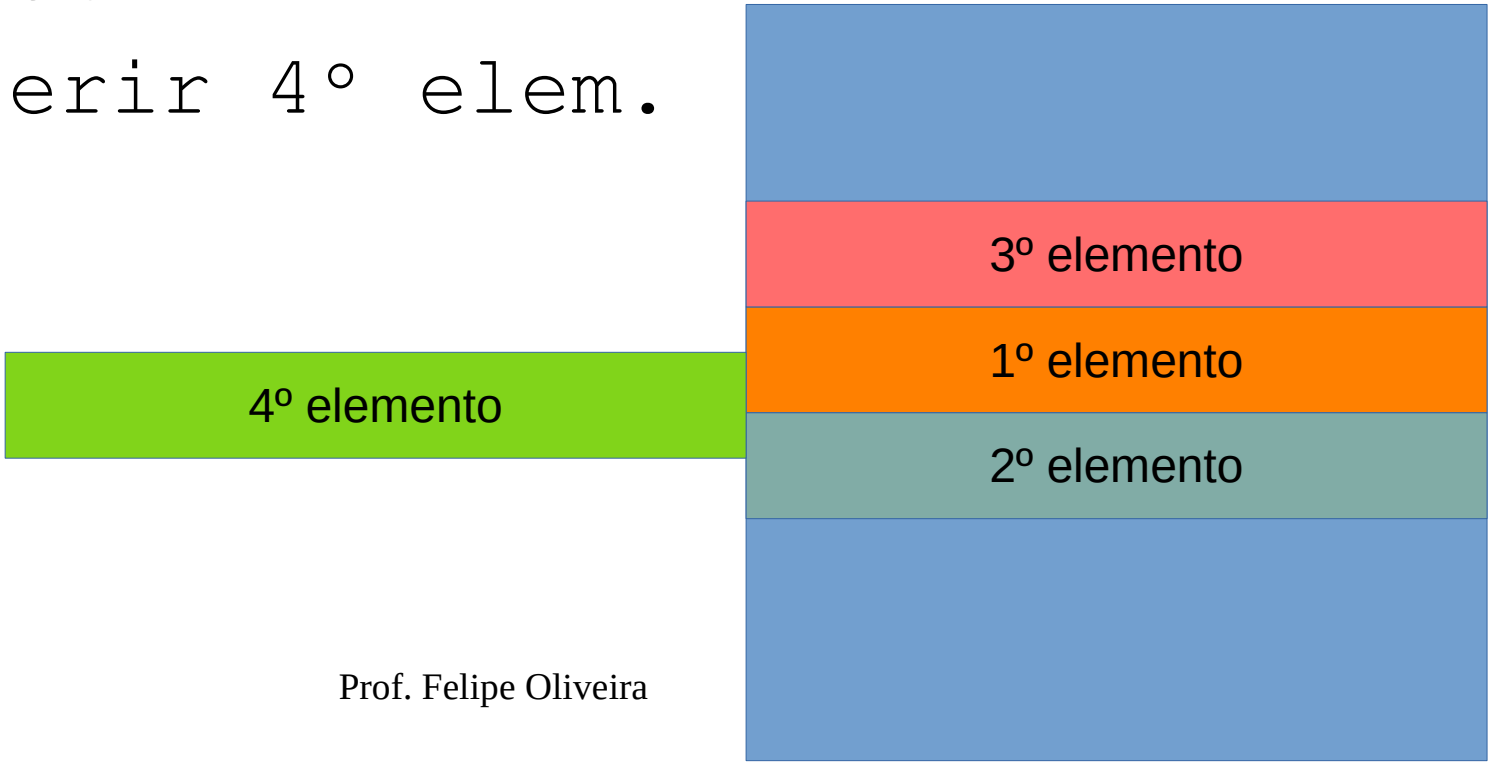


Elementos podem ser inseridos em qualquer posição!!!

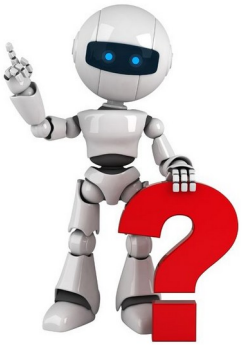
Estruturas de Dados



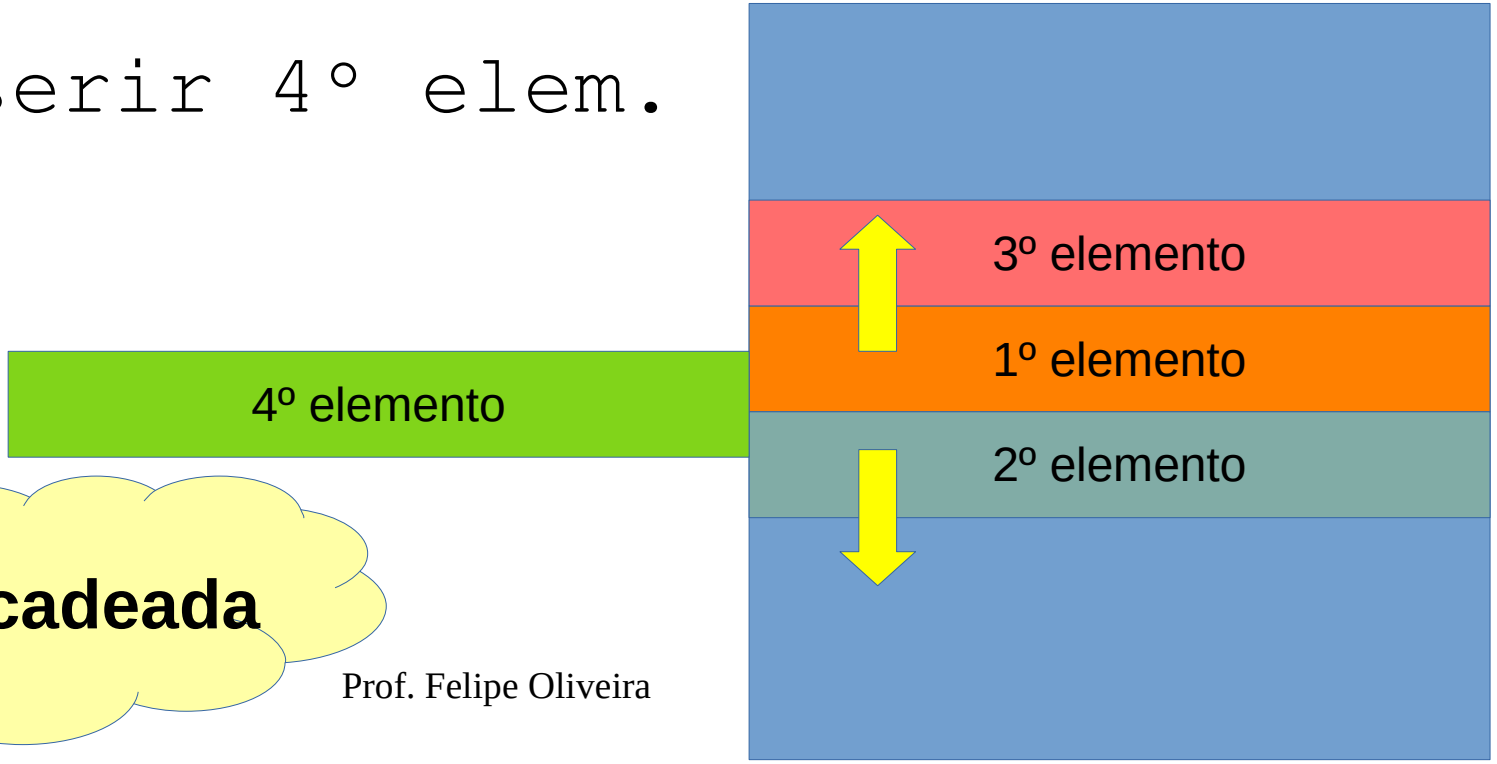
- Comportamento de uma lista:
  - `Lista:`
    - `Inserir 4° elem.`



Estruturas de Dados



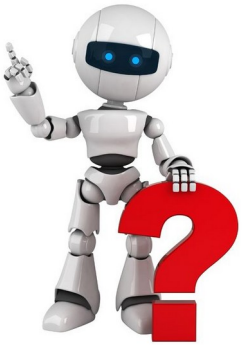
- Comportamento de uma lista:
  - `Lista:`
    - `Inserir 4° elem.`



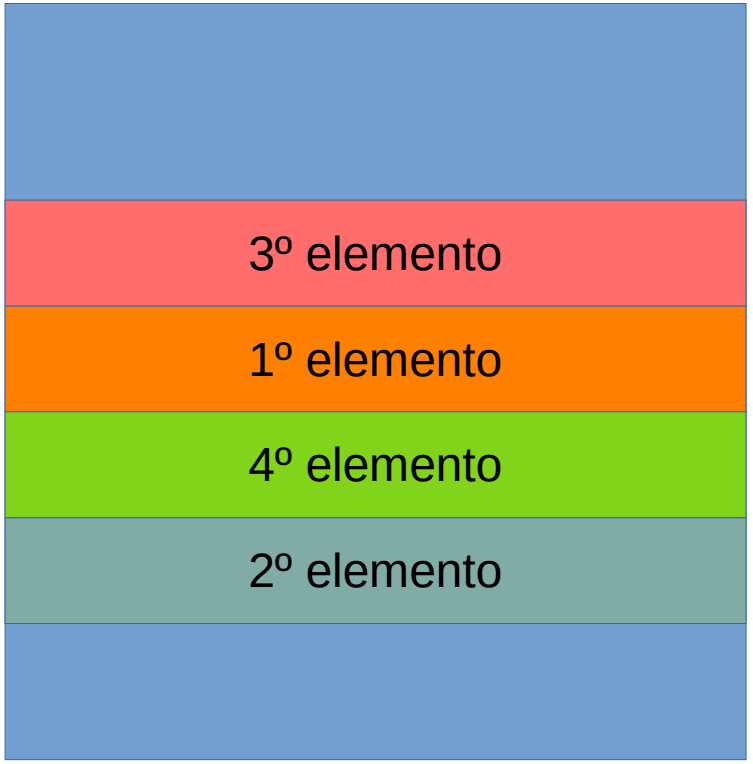
Lista encadeada

Prof. Felipe Oliveira

Estruturas de Dados



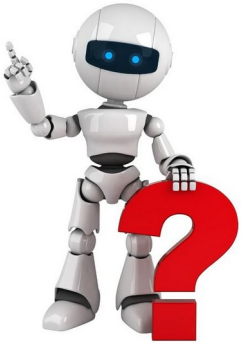
- Comportamento de uma lista:
  - `Lista:`
    - `Inserir 4° elem.`



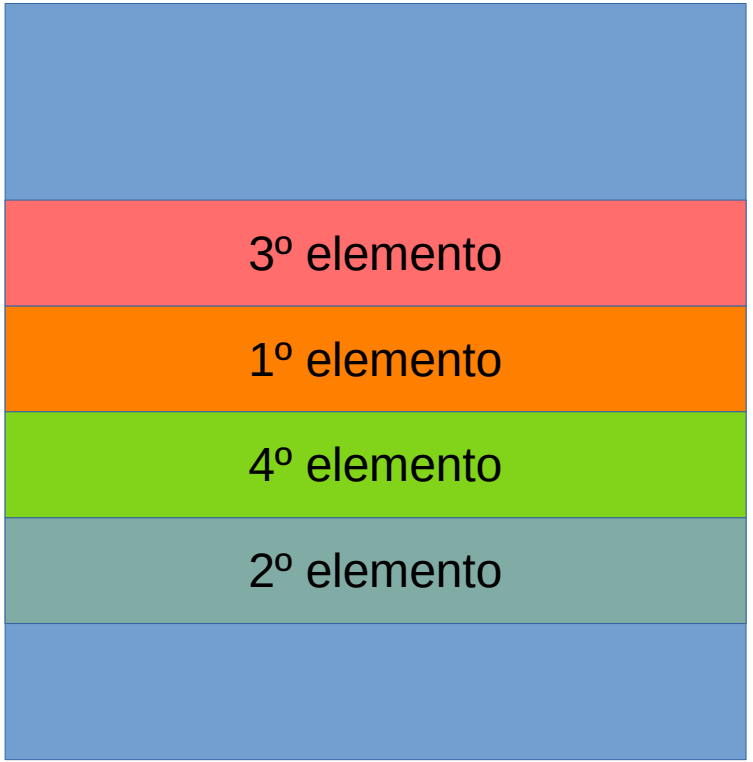
Lista encadeada

Prof. Felipe Oliveira

Estruturas de Dados

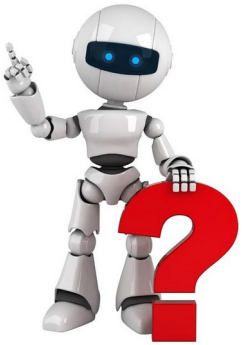


- Comportamento de uma lista:
  - `Lista:`
    - `Remoção.`

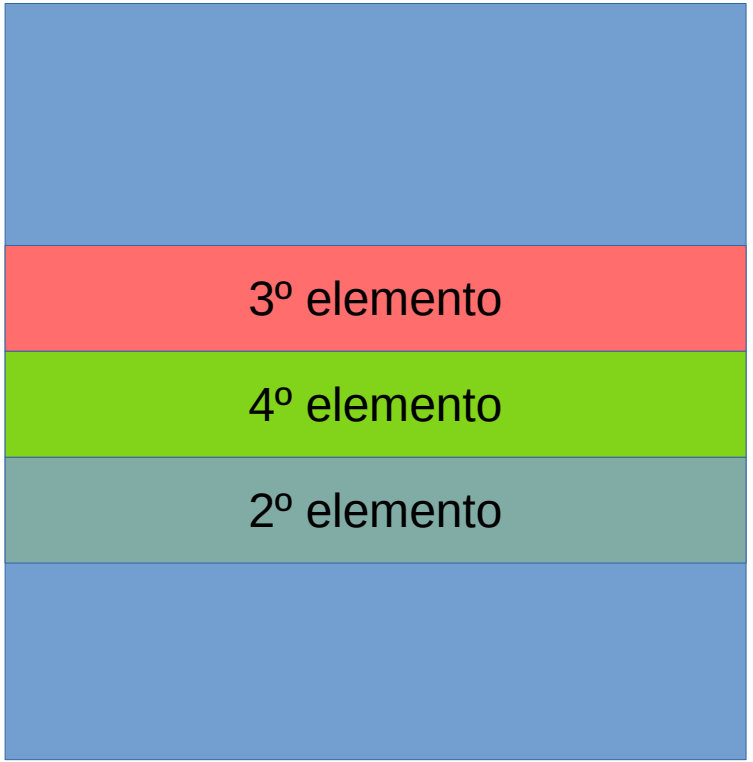


Elementos podem ser removidos de qualquer posição!!!

Estruturas de Dados

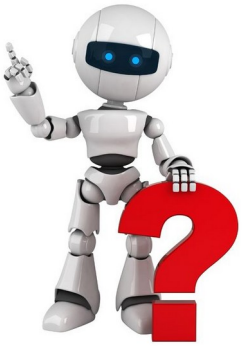


- Comportamento de uma lista:
  - `Lista:`
    - `Remoção.`



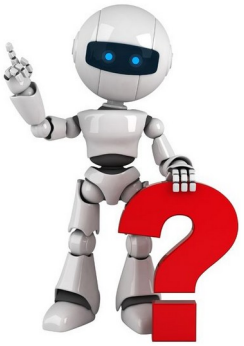
Elementos podem ser removidos de qualquer posição!!!

Estruturas de Dados



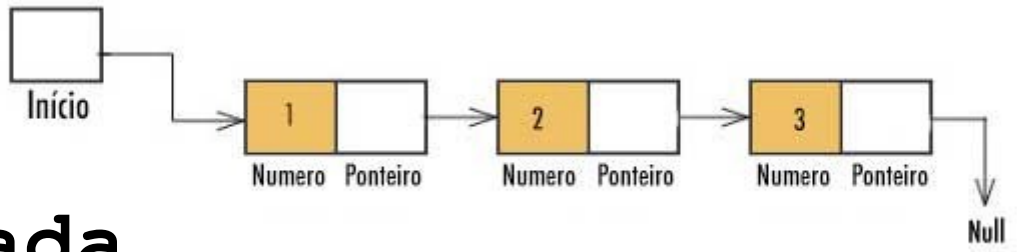
- Implementação de uma lista:
  - Estática;
    - Arrays.
  - **Dinâmica.**
    - **Lista encadeada.**

Estruturas de Dados

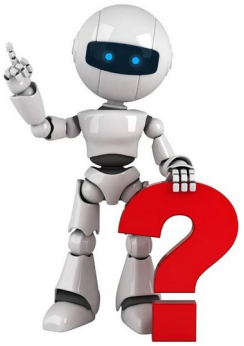


- Implementação de uma lista:
  - Estática;
    - Arrays.
  - **Dinâmica.**
    - **Lista encadeada.**

Cada elemento é armazenado
em uma célula da lista encadeada

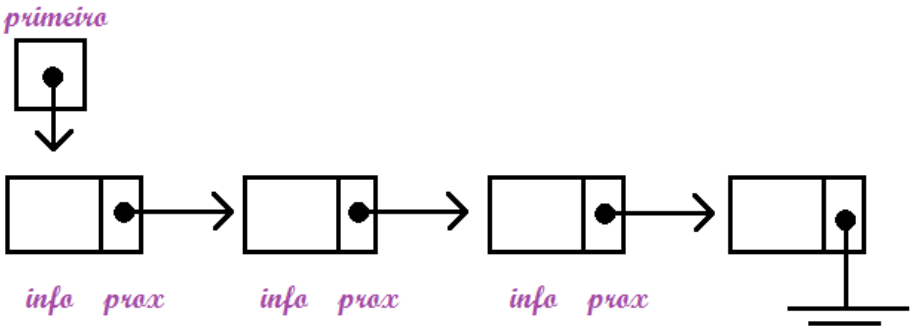


Estruturas de Dados

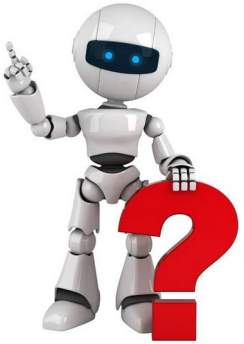


- Implementação de uma lista:
  - Estática;
    - Arrays.
  - **Dinâmica.**
    - **Lista encadeada.**

Cada elemento é armazenado
em uma célula da lista encadeada

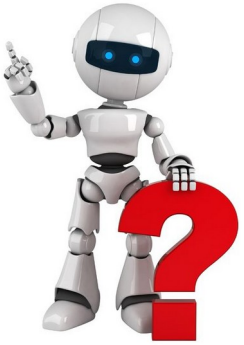


Estruturas de Dados

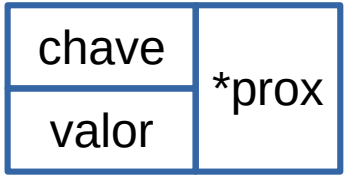


- Funcionamento de uma lista:
  - Criar ponteiro para o início;
  - Inicializar o ponteiro.

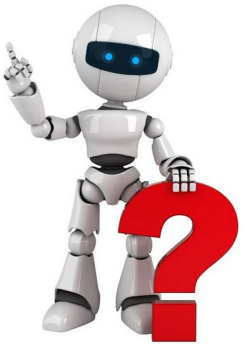
Estruturas de Dados



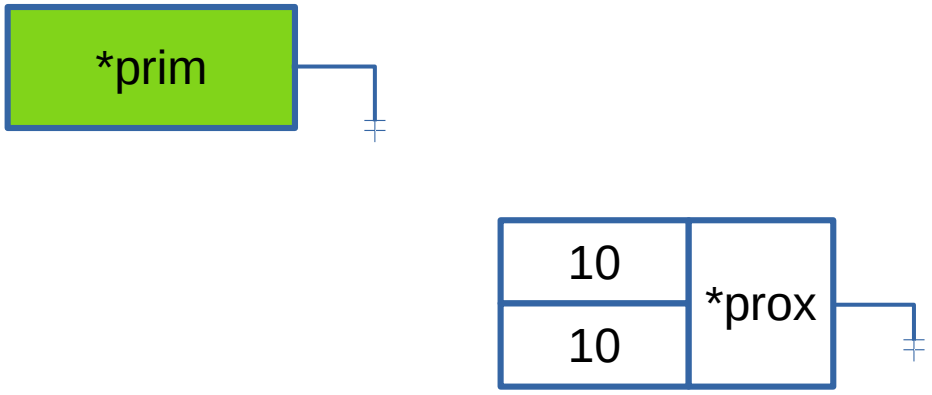
- Funcionamento de uma lista:
  - Criar nó da lista.



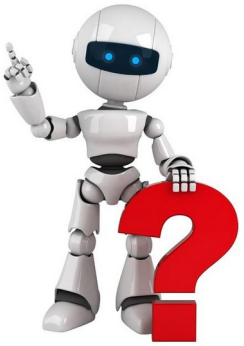
Estruturas de Dados



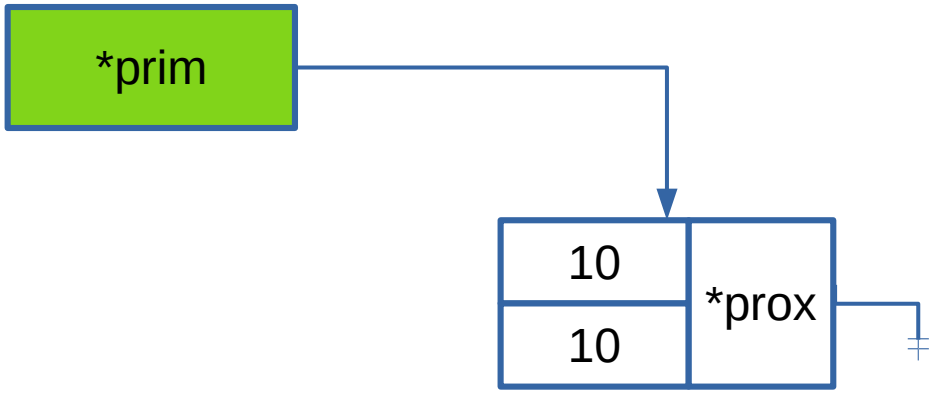
- Funcionamento de uma lista:
  - Preencher nó da lista.



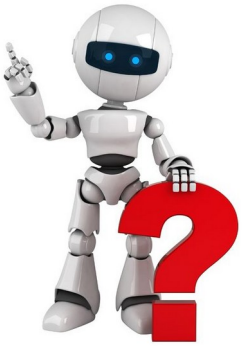
Estruturas de Dados



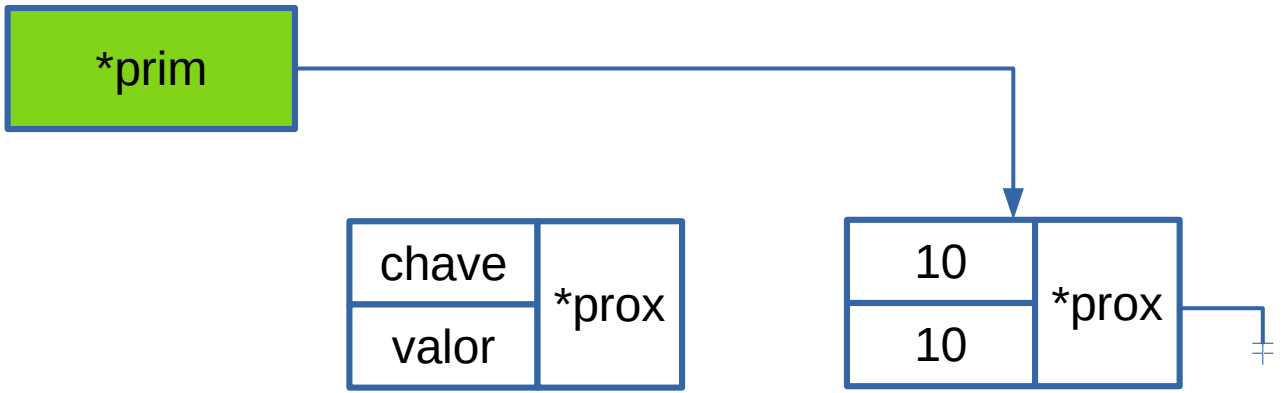
- Funcionamento de uma lista:
  - Conecta à lista.



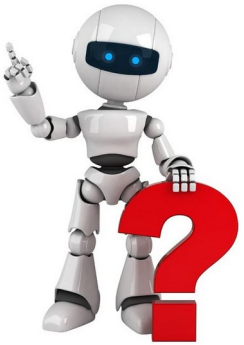
Estruturas de Dados



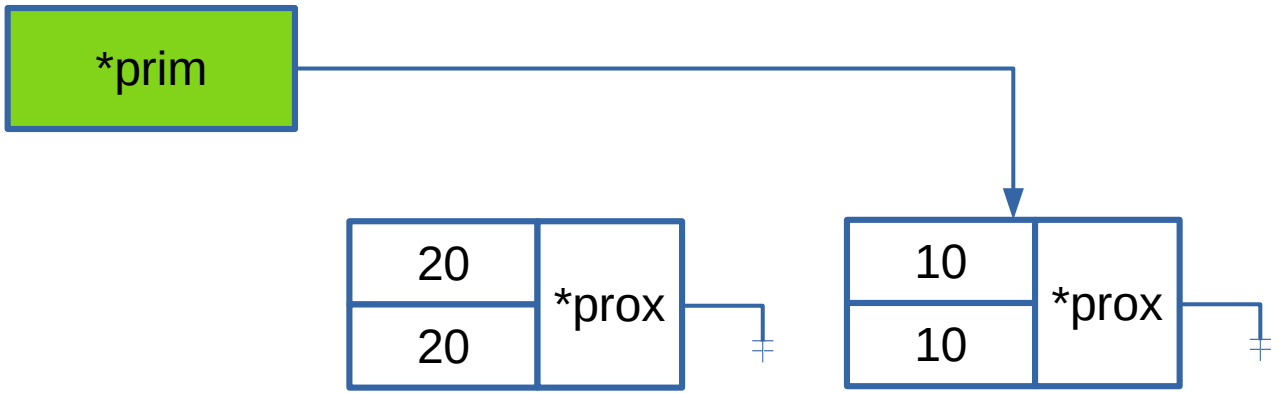
- Funcionamento de uma lista:
  - Criar novo nó da lista.



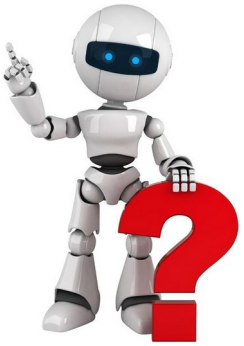
Estruturas de Dados



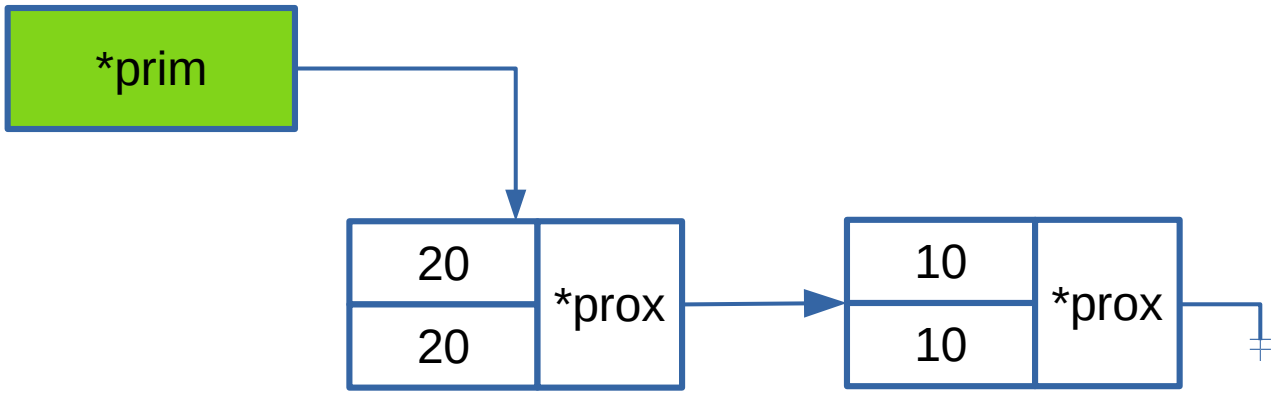
- Funcionamento de uma lista:
  - Preencher novo nó da lista.



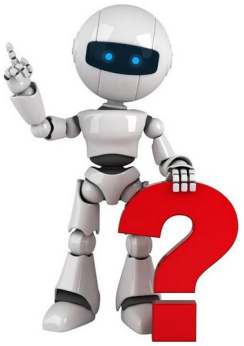
Estruturas de Dados



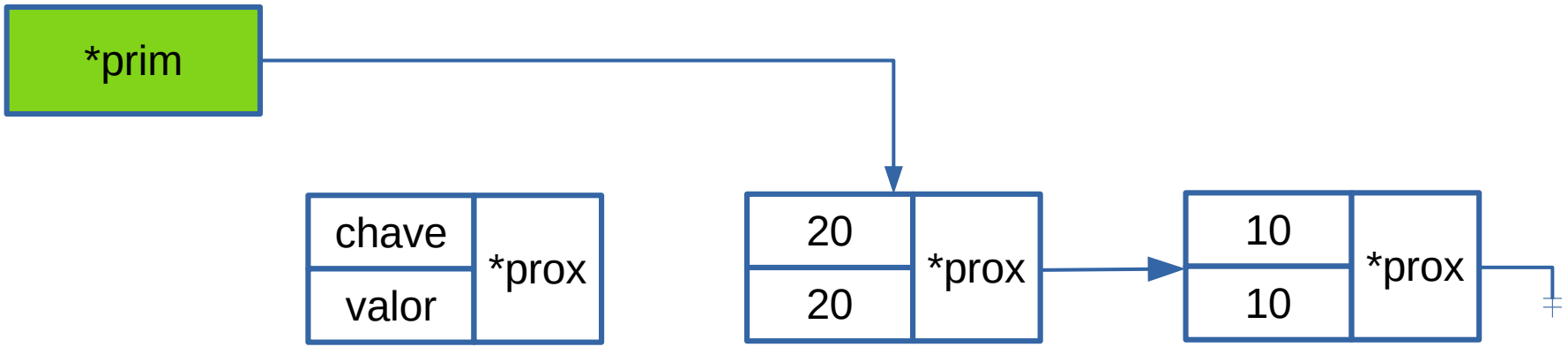
- Funcionamento de uma lista:
  - Conecta à lista.



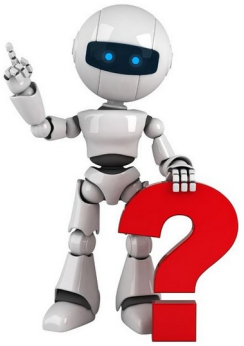
Estruturas de Dados



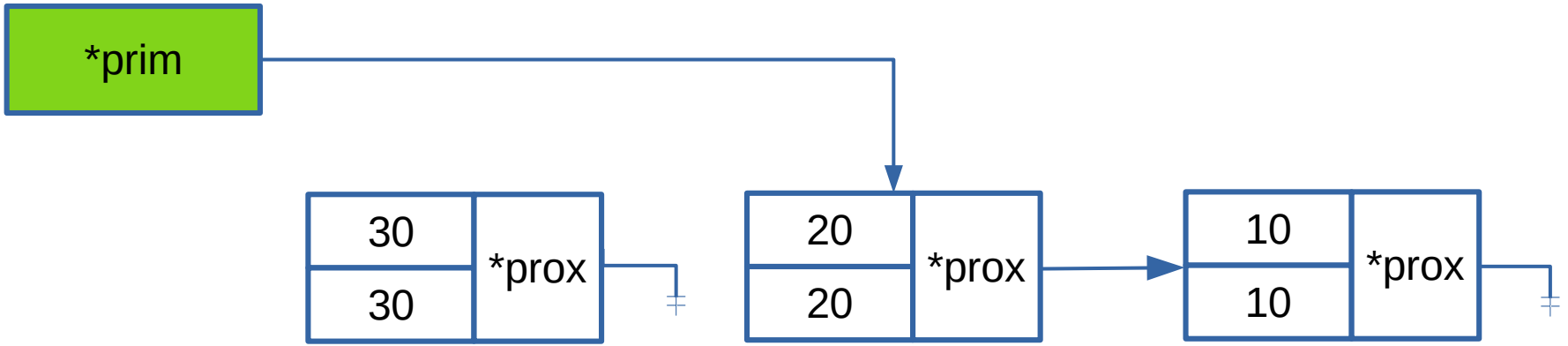
- Funcionamento de uma lista:
  - Criar novo nó da lista.



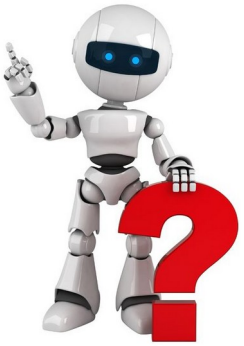
Estruturas de Dados



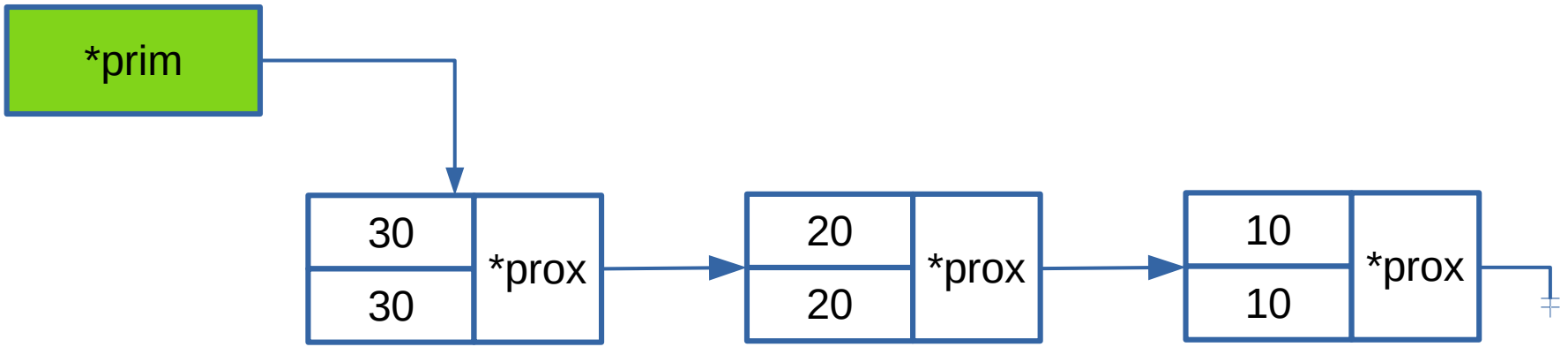
- Funcionamento de uma lista:
  - Preencher novo nó da lista.



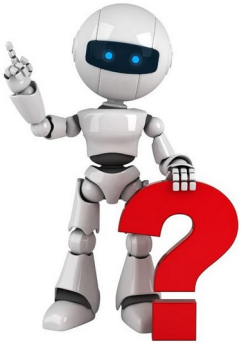
Estruturas de Dados



- Funcionamento de uma lista:
  - Conecta à lista.



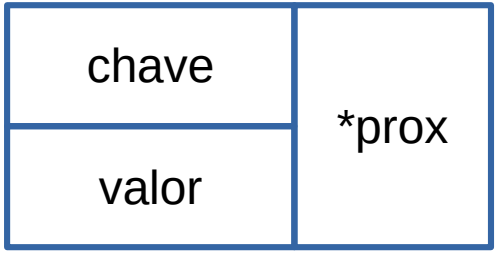
Estruturas de Dados

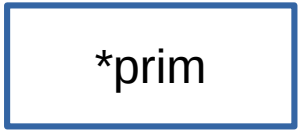


- Implementação de uma lista:
  - Definição das estruturas da lista.

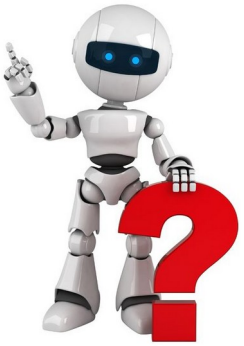
```
struct no_lista
{
        int chave;
        int valor;
        struct no_lista *prox;
};

struct lista
{
        struct no_lista *prim;
};
```





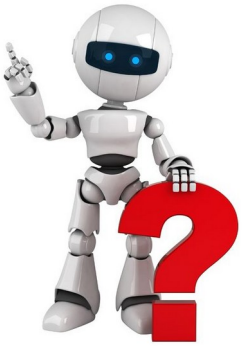
Estruturas de Dados



- Implementação de uma lista:
  - Criar lista.

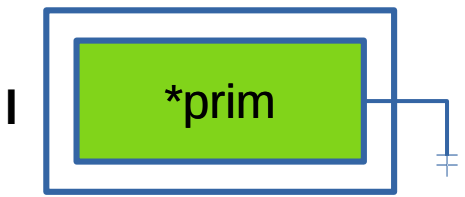
```
int main()
{
        struct lista l;
        inicializa_lista(&l);
}
```

Estruturas de Dados

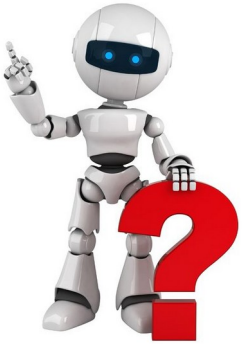


- Implementação de uma lista:
  - Inicializar lista.

```
int main()
{
        struct lista l;
        inicializa_lista(&l);
}
```



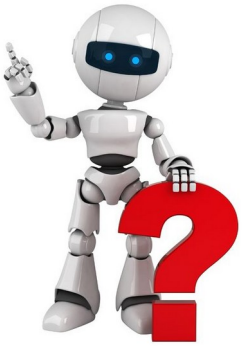
Estruturas de Dados



- Implementação de uma lista:
  - Inicializar lista.

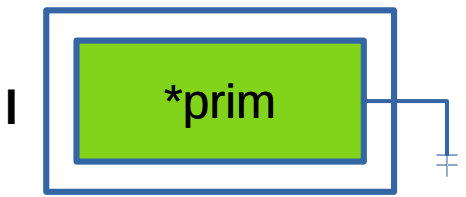
```
void inicializa_lista(struct lista *l)
{
	l->prim=NULL;
}
```

Estruturas de Dados

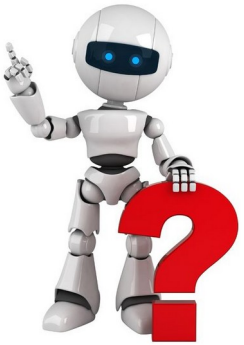


- Implementação de uma lista:
  - Verifica lista vazia.

```
int lista_vazia(struct lista *l)
{
        If( l->prim == NULL )
        {
                return 1;
        }
        else
        {
                return 0;
        }
}
```

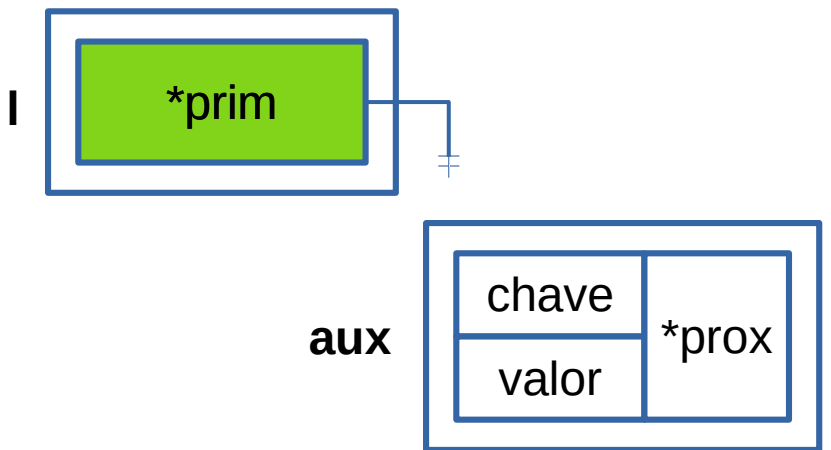


Estruturas de Dados

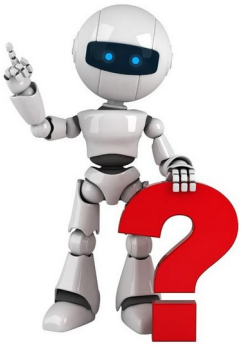


- Implementação de uma lista:
  - Inserção na lista.

```
void inserir_elemento(struct lista *l, int chave, int valor)
{
        struct no_lista *aux;
        aux=(struct no_lista*)malloc(sizeof(struct no_lista));
        aux->chave=chave;
        aux->valor=valor;
        aux->prox=NULL;
        valor = lista_vazia(l);
        if(valor == 1)  {
                l->prim=aux;
        }
        else  {
                aux->prox = l->prim;
                l->prim=aux;
        }
}
```

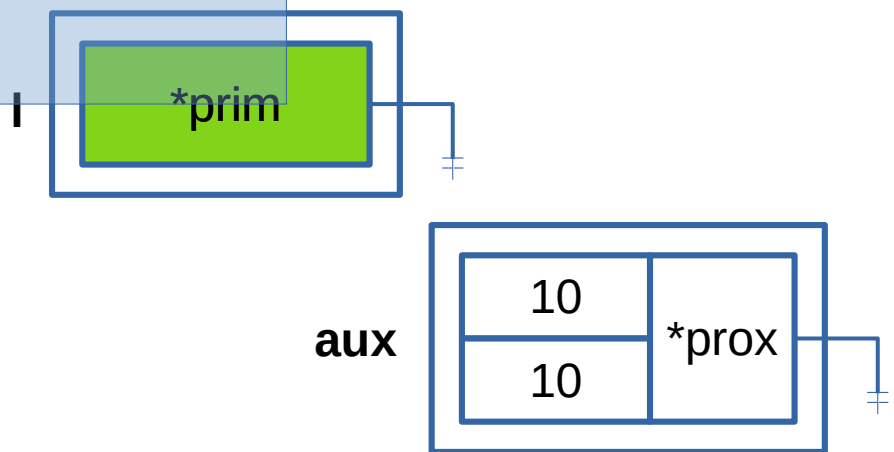


Estruturas de Dados

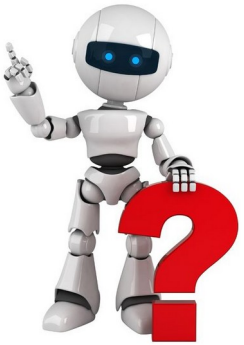


- Implementação de uma lista:
  - Inserção na lista.

```
void inserir_elemento(struct lista *l, int chave, int valor)
{
        struct no_lista *aux;
        aux=(struct no_lista*)malloc(sizeof(struct no_lista));
        aux->chave=chave;
        aux->valor=valor;
        aux->prox=NULL;
        valor = lista_vazia(l);
        if(valor == 1)  {
                l->prim=aux;
        }
        else  {
                aux->prox = l->prim;
                l->prim=aux;
        }
}
```

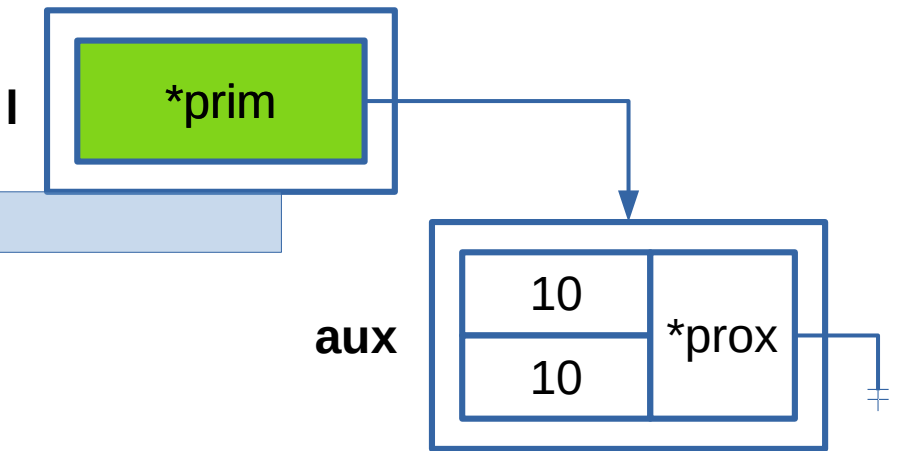


Estruturas de Dados

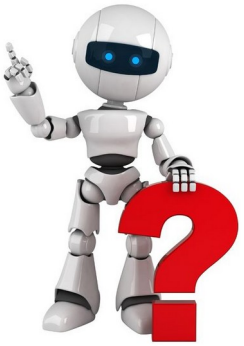


- Implementação de uma lista:
  - Inserção na lista.

```
void inserir_elemento(struct lista *l, int chave, int valor)
{
        struct no_lista *aux;
        aux=(struct no_lista*)malloc(sizeof(struct no_lista));
        aux->chave=chave;
        aux->valor=valor;
        aux->prox=NULL;
        valor = lista_vazia(l);
        if(valor == 1)  {
                l->prim=aux;
        }
        else  {
                aux->prox = l->prim;
                l->prim=aux;
        }
}
```

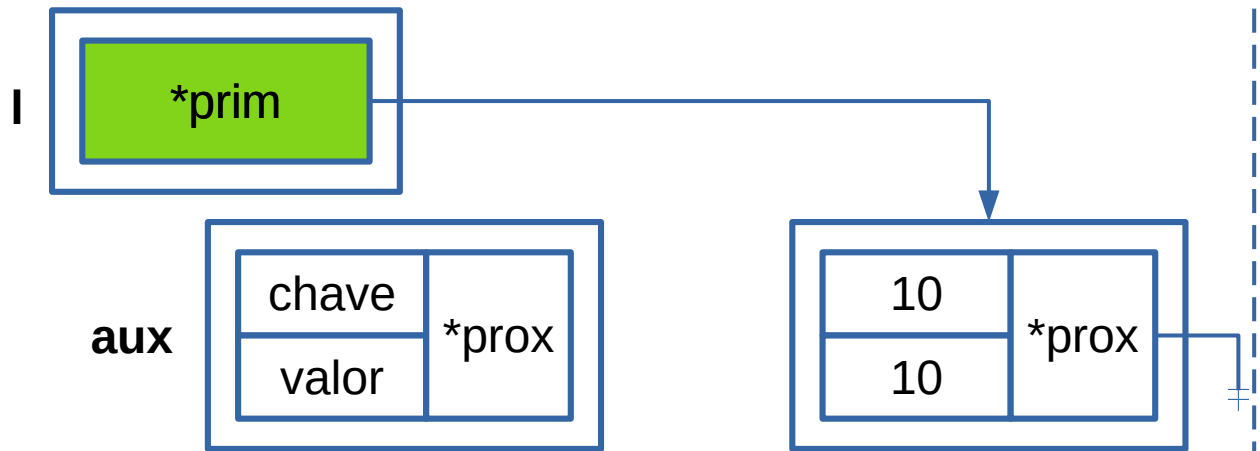


Estruturas de Dados

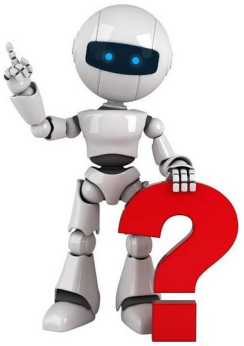


- Implementação de uma lista:
  - Inserção na lista.

```
void inserir_elemento(struct lista *l, int chave, int valor)
{
        struct no_lista *aux;
        aux=(struct no_lista*)malloc(sizeof(struct no_lista));
        aux->chave=chave;
        aux->valor=valor;
        aux->prox=NULL;
        valor = lista_vazia(l);
        if(valor == 1)  {
                l->prim=aux;
        }
        else  {
                aux->prox = l->prim;
                l->prim=aux;
        }
}
```

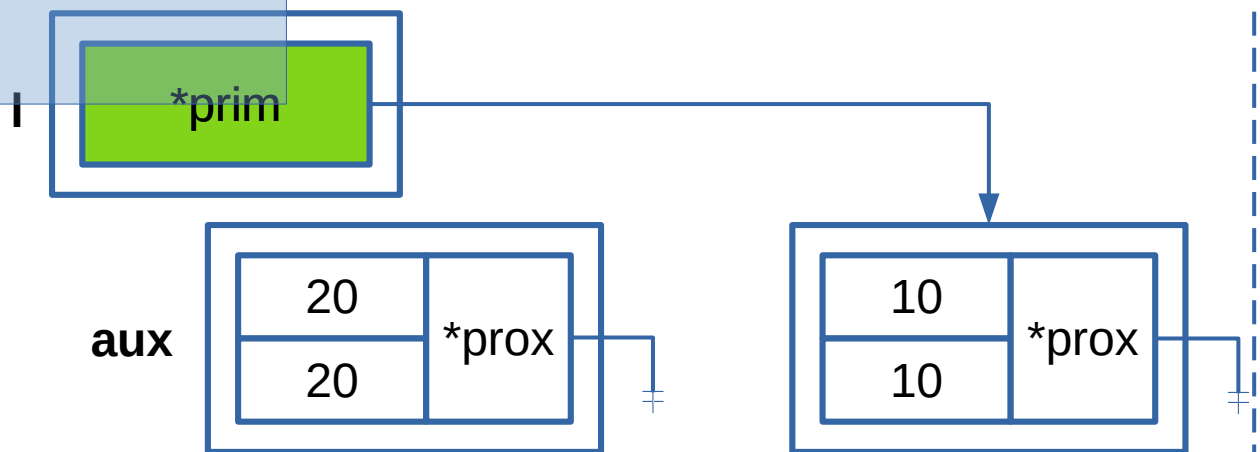


Estruturas de Dados

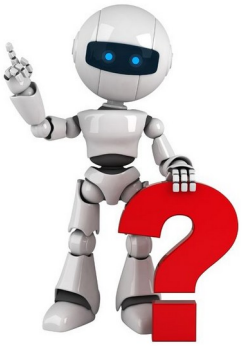


- Implementação de uma lista:
  - Inserção na lista.

```
void inserir_elemento(struct lista *l, int chave, int valor)
{
        struct no_lista *aux;
        aux=(struct no_lista*)malloc(sizeof(struct no_lista));
        aux->chave=chave;
        aux->valor=valor;
        aux->prox=NULL;
        valor = lista_vazia(l);
        if(valor == 1)  {
                l->prim=aux;
        }
        else  {
                aux->prox = l->prim;
                l->prim=aux;
        }
}
```

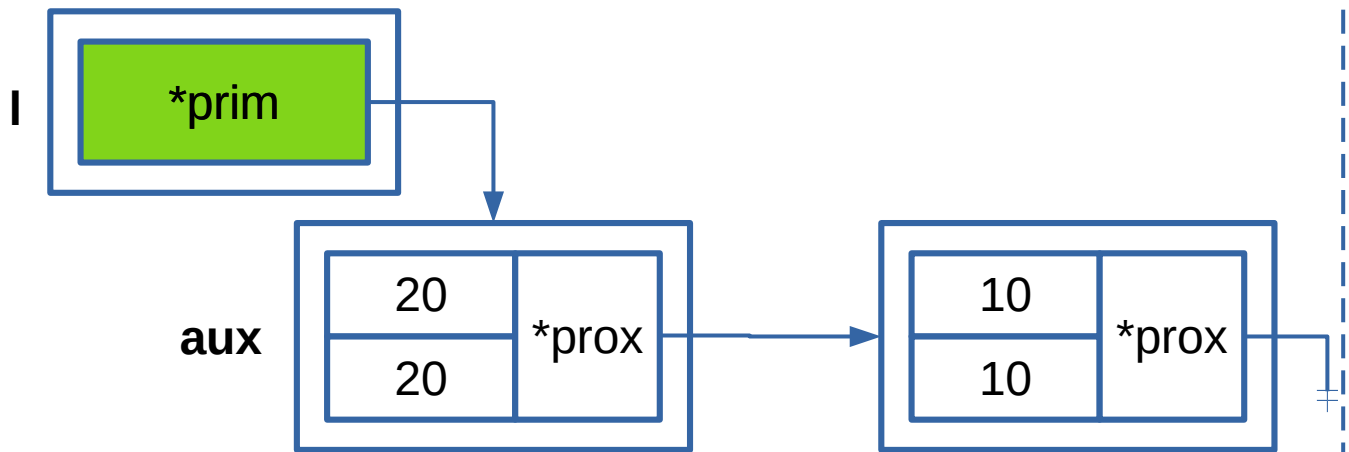


Estruturas de Dados

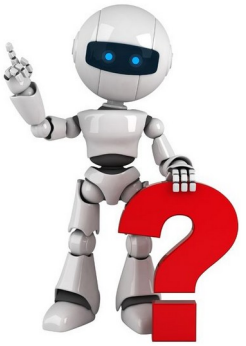


- Implementação de uma lista:
  - Inserção na lista.

```
void inserir_elemento(struct lista *l, int chave, int valor)
{
        struct no_lista *aux;
        aux=(struct no_lista*)malloc(sizeof(struct no_lista));
        aux->chave=chave;
        aux->valor=valor;
        aux->prox=NULL;
        valor = lista_vazia(l);
        if(valor == 1)  {
                l->prim=aux;
        }
        else  {
                aux->prox = l->prim;
                l->prim=aux;
        }
}
```

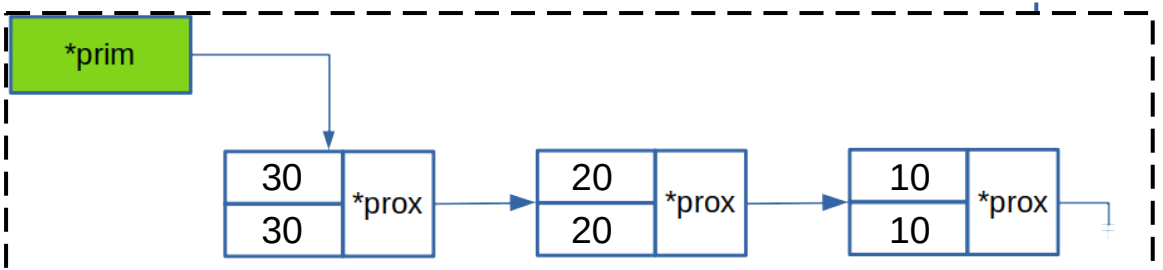


Estruturas de Dados

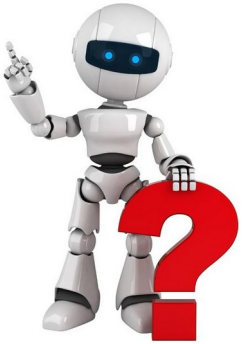


- Implementação de uma lista:
  - Mostra lista.

```
void mostra_lista(struct lista *l)
{
        struct no_lista *aux;
        aux=l->prim;
        while(aux!=NULL)
        {
                printf(" chave %d\t valor: %d\n",aux->chave,aux->valor);
                aux=aux->prox;
        }
}
```

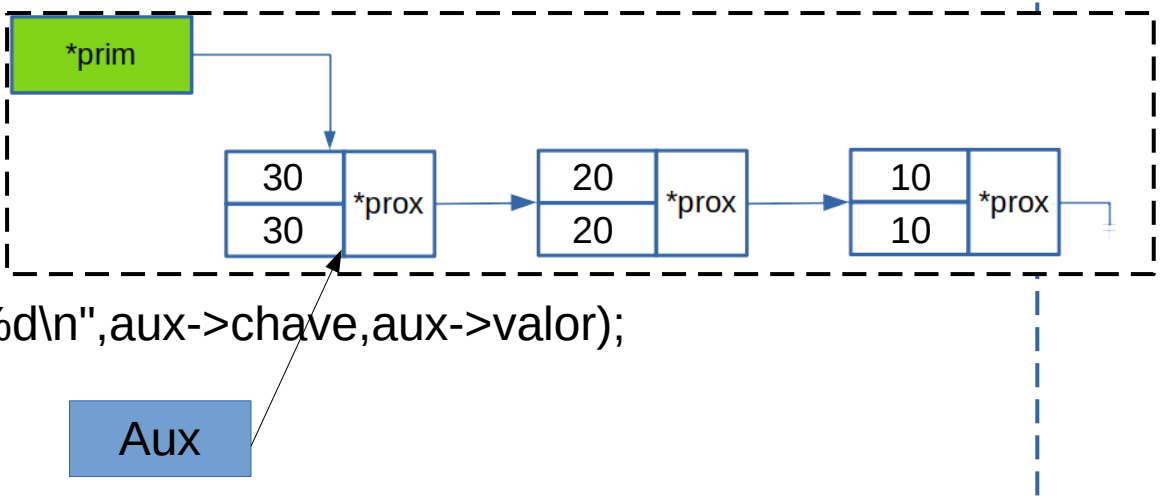


Estruturas de Dados

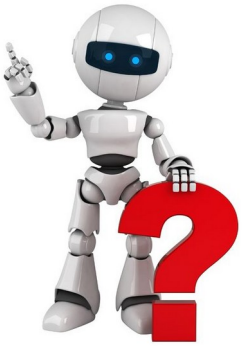


- Implementação de uma lista:
  - Mostra lista.

```
void mostra_lista(struct lista *l)
{
        struct no_lista *aux;
        aux=l->prim;
        while(aux!=NULL)
        {
                printf(" chave %d\t valor: %d\n",aux->chave,aux->valor);
                aux=aux->prox;
        }
}
```

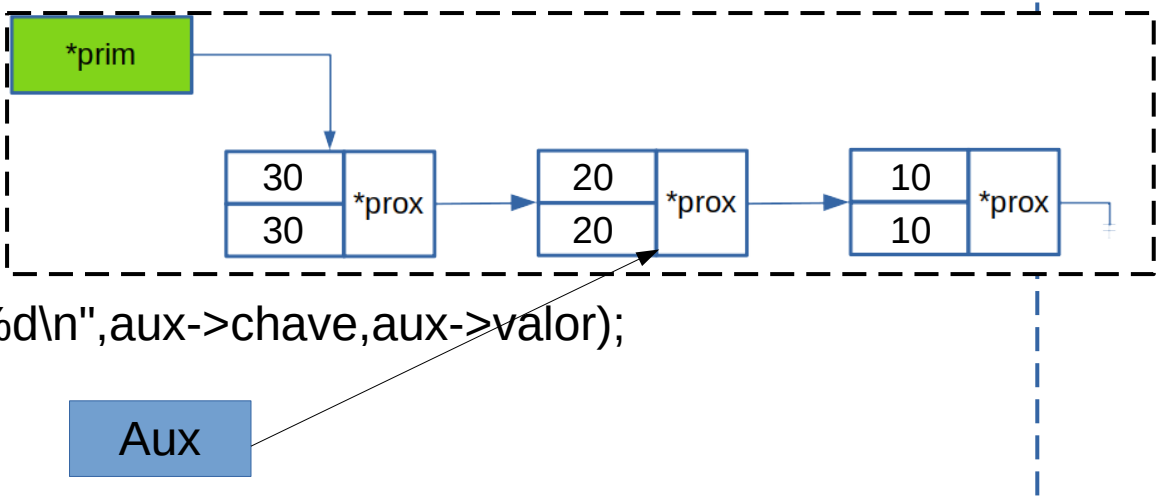


Estruturas de Dados

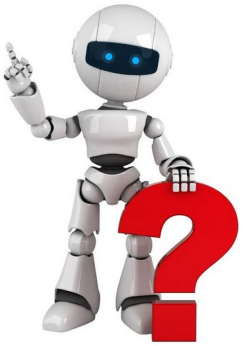


- Implementação de uma lista:
  - Mostra lista.

```
void mostra_lista(struct lista *l)
{
        struct no_lista *aux;
        aux=l->prim;
        while(aux!=NULL)
        {
                printf(" chave %d\t valor: %d\n",aux->chave,aux->valor);
                aux=aux->prox;
        }
}
```

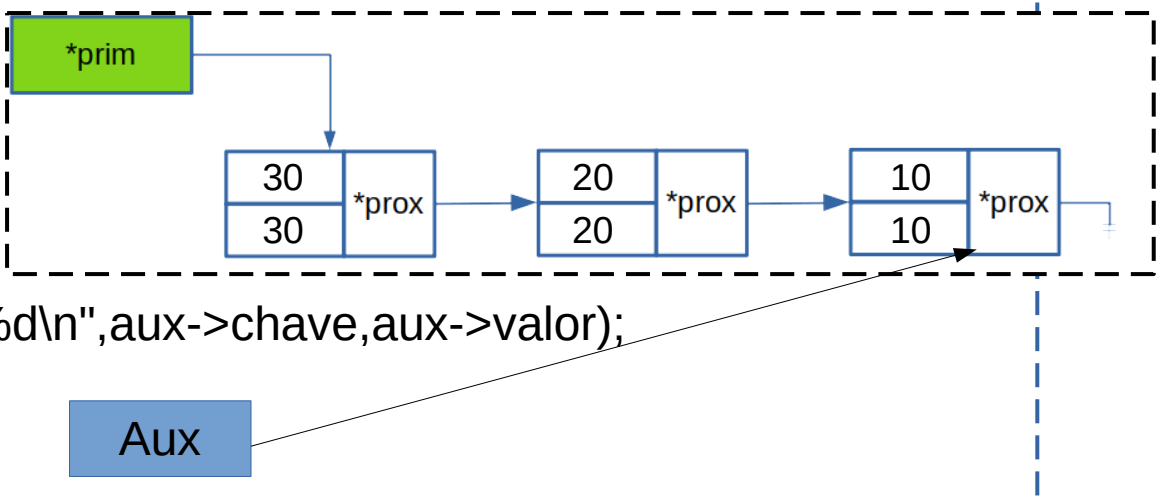


Estruturas de Dados

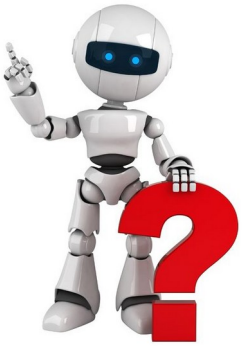


- Implementação de uma lista:
  - Mostra lista.

```
void mostra_lista(struct lista *l)
{
        struct no_lista *aux;
        aux=l->prim;
        while(aux!=NULL)
        {
                printf(" chave %d\t valor: %d\n",aux->chave,aux->valor);
                aux=aux->prox;
        }
}
```

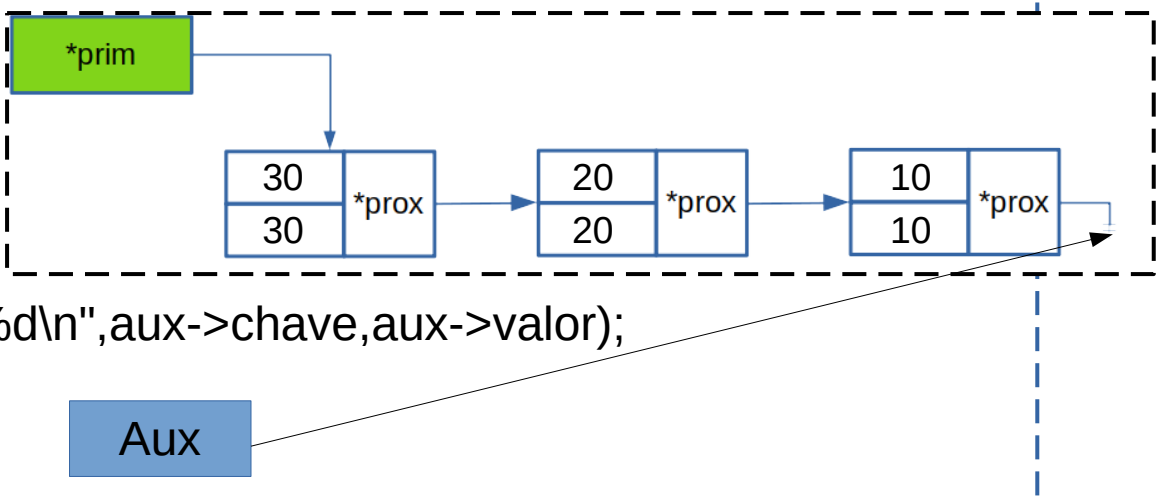


Estruturas de Dados

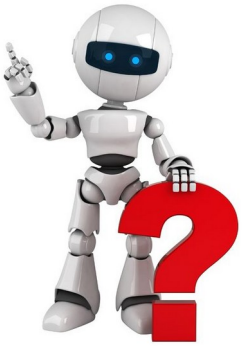


- Implementação de uma lista:
  - Mostra lista.

```
void mostra_lista(struct lista *l)
{
        struct no_lista *aux;
        aux=l->prim;
        while(aux!=NULL)
        {
                printf(" chave %d\t valor: %d\n",aux->chave,aux->valor);
                aux=aux->prox;
        }
}
```

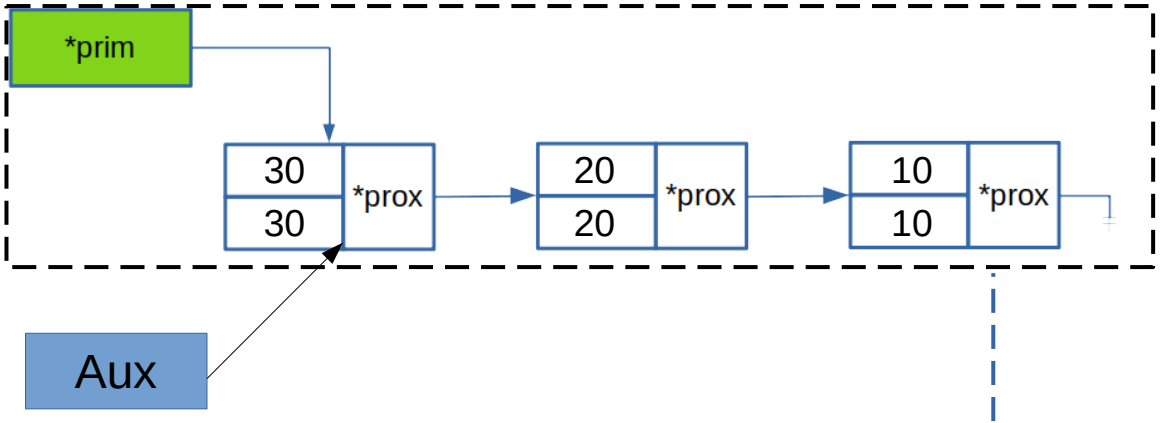


Estruturas de Dados

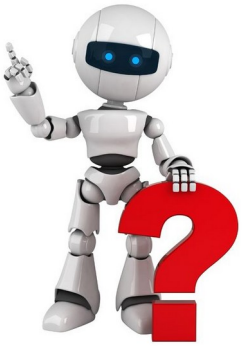


- Implementação de uma lista:
  - Busca na lista.

```
struct no_lista* busca(struct lista *l,int chave)
{
      struct no_lista *aux;
      aux=l->prim;
      while(aux!=NULL)  {
            if(aux->chave==chave)  {
                  return aux;
            }
            aux=aux->prox;
      }
      return NULL;
}
```

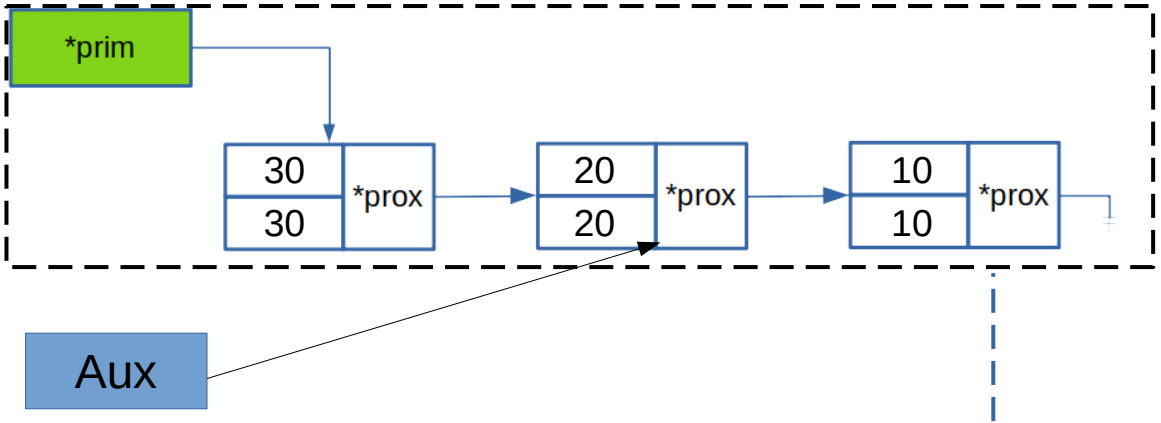


Estruturas de Dados

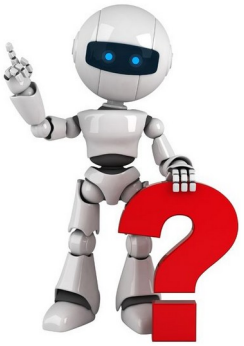


- Implementação de uma lista:
  - Busca na lista.

```
struct no_lista* busca(struct lista *l,int chave)
{
        struct no_lista *aux;
        aux=l->prim;
        while(aux!=NULL)  {
                if(aux→chave==chave)  {
                        return aux;
                }
                aux=aux->prox;
        }
        return NULL;
}
```

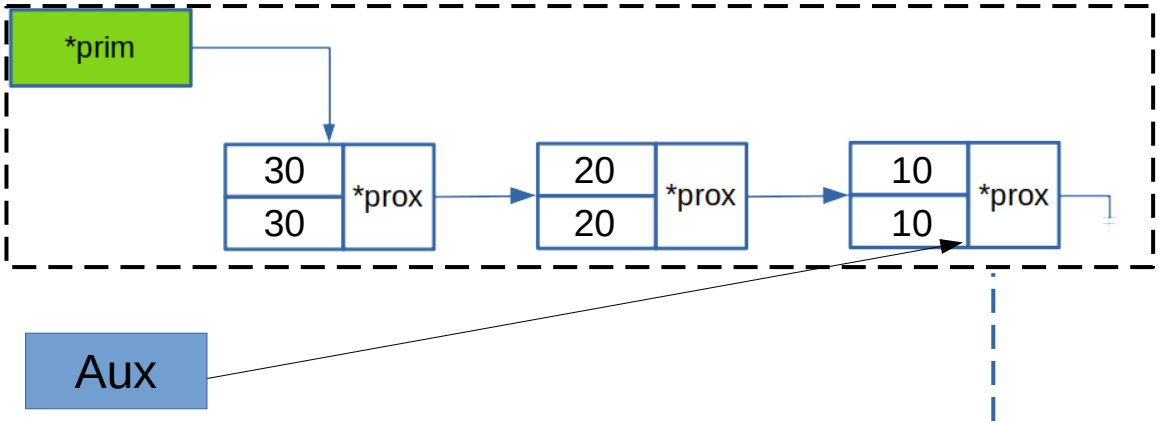


Estruturas de Dados

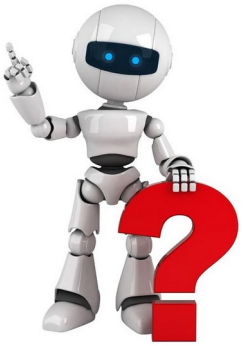


- Implementação de uma lista:
  - Busca na lista.

```
struct no_lista* busca(struct lista *l,int chave)
{
        struct no_lista *aux;
        aux=l->prim;
        while(aux!=NULL)  {
                if(aux→chave==chave)  {
                        return aux;
                }
                aux=aux->prox;
        }
        return NULL;
}
```

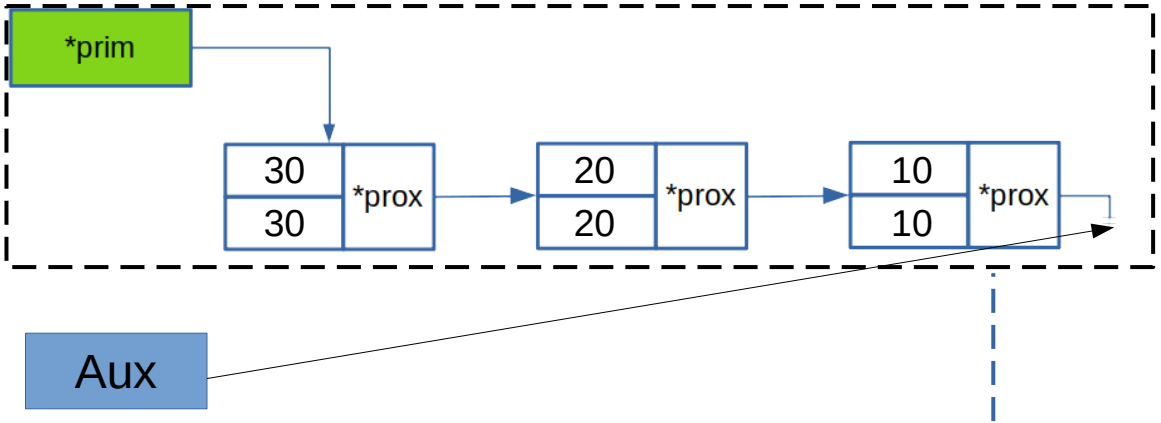


Estruturas de Dados

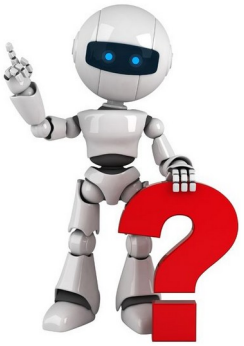


- Implementação de uma lista:
  - Busca na lista.

```
struct no_lista* busca(struct lista *l,int chave)
{
        struct no_lista *aux;
        aux=l->prim;
        while(aux!=NULL)  {
                if(aux→chave==chave)  {
                        return aux;
                }
                aux=aux->prox;
        }
        return NULL;
}
```



Estruturas de Dados



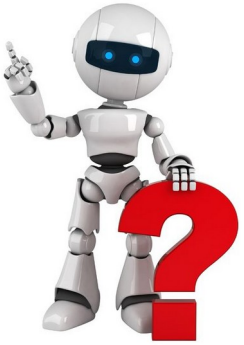
- Implementação de uma lista:
  - Execução da lista.

```
int main()
{
	struct lista l;
	cria_lista(&l);
	inserir_elemento(&l,5,10);
	inserir_elemento(&l,15,20);
	inserir_elemento(&l,25,30);
	inserir_elemento(&l,35,40);
	inserir_elemento(&l,45,50);

	mostra_regioes(&l);
}
```

Dúvidas

Atividade Prática



- Implementação de uma lista encadeada:
  - Inserção na primeira posição;
  - Inserção ordenada;
  - Imprime lista;
  - Busca na lista.

Algoritmos e Estruturas de Dados III

Prof. Dr. Felipe Oliveira
felipeoliveira@ufam.edu.br