

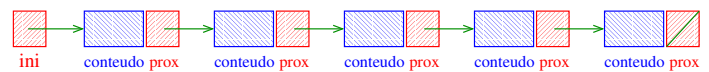
Melhores momentos

Listas encadeadas

AULA 8

Uma **lista encadeada** (= *linked list* = lista ligada) é uma sequência de **células**; cada **célula** contém um **objeto** de algum tipo e o **endereço** da célula seguinte.

Ilustração de uma **lista encadeada** (“sem cabeça”)



Estrutura de uma lista encadeada em C

```
struct celula {
    int conteudo;
    struct celula *prox;
};
typedef struct celula Celula;

Celula *ini;
/* inicialmente a lista esta vazia */
ini = NULL;
```



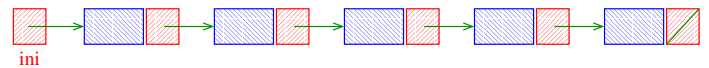
Busca em uma lista encadeada

Esta função **recebe** um inteiro **x** e uma lista **ini**. A função **devolve** o endereço de uma célula que contém **x**. Se tal célula não existe, a função **devolve** NULL.

```
Celula* busca (int x, Celula *ini)
{
    Celula *p;
    p = ini;
    while (p != NULL && p->conteudo != x)
        p = p->prox;
    return p;
}
```

Imprime conteúdo de uma lista

Esta função **imprime** o **conteudo** de cada célula de uma lista encadeada **ini**.



```
void imprima (Celula *ini)
{
    Celula *p;
    for (p=ini; p != NULL; p=p->prox)
        printf("%d\n", p->conteudo);
}
```

AULA 9

Mais listas encadeadas em C

PF 4, S 3.3

<http://www.ime.usp.br/~pf/algorithmos/aulas/lista.html>

Inserção no início de uma lista encadeada

Cria uma célula para guardar um elemento x e insere esta célula no início da lista ini .

```
void insere (int x; Celula *ini)
{
    Celula nova; ERRADO!
    nova.conteudo = x;
    nova.prox = ini;
    ini = &nova;
}
```

A estrutura $nova$ é uma variável “automática”. Ela será devolvida ao sistema no final da execução da função.

Inserção no início de uma lista encadeada

Cria uma célula para guardar um elemento x e insere esta célula no início da lista ini .

```
void insere (int x; Celula *ini)
{
    Celula *nova;
    nova = malloc(sizeof(Celula));
    nova->conteudo = x;
    nova->prox = ini;
    ini = nova; ERRADO!
}
```

Mesmo erro! O parâmetro ini é uma variável local (=automática) que começa inicializada. Ela será devolvida ao sistema...

Inserção no início de uma lista encadeada

Cria uma célula para guardar um elemento x e insere esta célula no início da lista ini .

```
Celula *insere (int x; Celula *ini) {
    Celula *nova;
    nova = malloc(sizeof(Celula));
    nova->conteudo = x;
    nova->prox = ini;
    return nova;
}
```

Certo! Exemplos de chamadas da função

```
ini = insere(11, ini);
ini = insere(valor+3, ini);
```

Inserção no início de uma lista encadeada

Cria uma célula para guardar um elemento x e insere esta célula no início da lista ini .

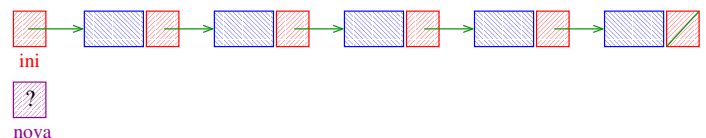
```
void insere (int x; Celula **ini) {
    Celula *nova;
    nova = malloc(sizeof(Celula));
    nova->conteudo = x;
    nova->prox = *ini;
    *ini = nova;
}
```

Certo! Exemplos de chamadas da função

```
insere(11, &ini);
insere(valor+3, &ini);
```

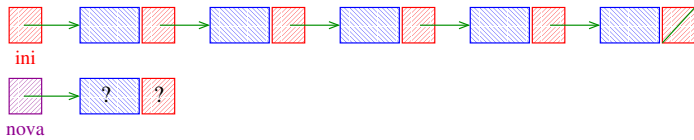
Inserção no início de uma lista encadeada

Cria uma célula para guardar um elemento x e insere esta célula no início da lista ini .



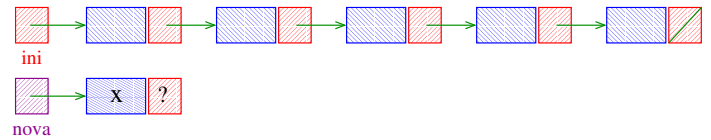
Inserção no início de uma lista encadeada

Cria uma célula para guardar um elemento x e insere esta célula no início da lista ini .



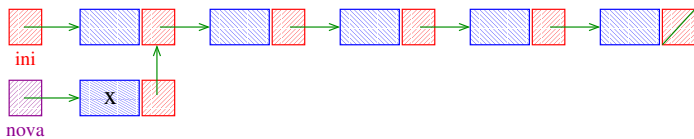
Inserção no início de uma lista encadeada

Cria uma célula para guardar um elemento x e insere esta célula no início da lista ini .



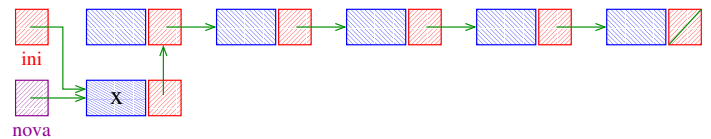
Inserção no início de uma lista encadeada

Cria uma célula para guardar um elemento x e insere esta célula no início da lista ini .



Inserção no início de uma lista encadeada

Cria uma célula para guardar um elemento x e insere esta célula no início da lista ini .



Remoção em uma lista encadeada

Recebe o endereço p de uma célula de uma lista encadeada e remove da lista a célula $p \rightarrow prox$.

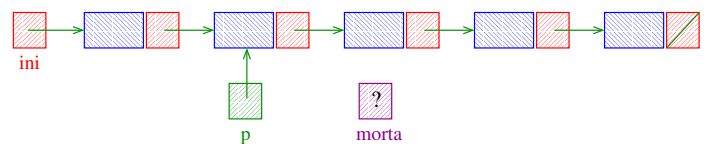
A função supõe que $p \neq \text{NULL}$ e $p \rightarrow prox \neq \text{NULL}$.

```
void remove (Celula *p)
{
    Celula *morta;
    morta = p->prox;
    p->prox = morta->prox;
    free(morta);
}
```

Remoção em uma lista encadeada

Recebe o endereço p de uma célula de uma lista encadeada e remove da lista a célula $p \rightarrow prox$.

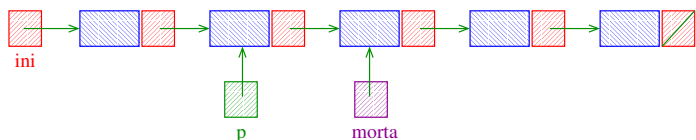
A função supõe que $p \neq \text{NULL}$ e $p \rightarrow prox \neq \text{NULL}$.



Remoção em uma lista encadeada

Recebe o endereço p de uma célula de uma lista encadeada e $remove$ da lista a célula $p \rightarrow prox$.

A função supõe que $p \neq \text{NULL}$ e $p \rightarrow prox \neq \text{NULL}$.

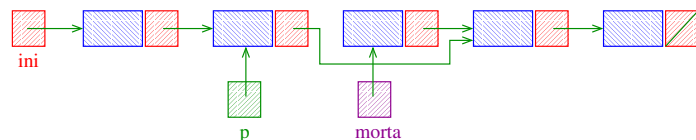


Navigation icons: back, forward, search, etc.

Remoção em uma lista encadeada

Recebe o endereço p de uma célula de uma lista encadeada e $remove$ da lista a célula $p \rightarrow prox$.

A função supõe que $p \neq \text{NULL}$ e $p \rightarrow prox \neq \text{NULL}$.

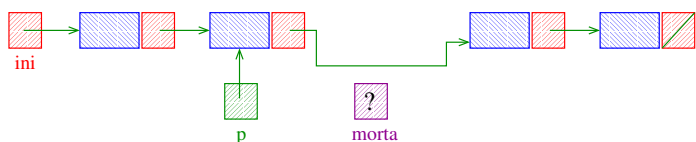


Navigation icons: back, forward, search, etc.

Remoção em uma lista encadeada

Recebe o endereço p de uma célula de uma lista encadeada e $remove$ da lista a célula $p \rightarrow prox$.

A função supõe que $p \neq \text{NULL}$ e $p \rightarrow prox \neq \text{NULL}$.



Navigation icons: back, forward, search, etc.

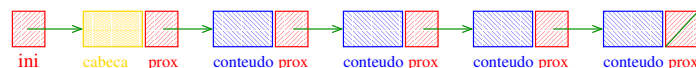
Listas encadeadas com cabeça

O conteúdo da primeira célula é irrelevante: ela serve apenas para marcar o início da lista. A primeira célula é a **cabeça** (= *head cell* = *dummy cell*) da lista.

A primeira célula está sempre no mesmo lugar na memória, mesmo que a lista fique vazia.

Se $ini \rightarrow prox == \text{NULL}$ se e somente se a lista está vazia.

Ilustração de uma lista encadeada "com cabeça"

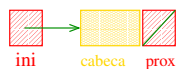


Navigation icons: back, forward, search, etc.

Estrutura de uma lista com cabeça

```
struct celula {
    int conteudo;
    struct celula *prox;
};
typedef struct celula Celula;

Celula *ini, *cabeça;
/* inicialmente a lista esta vazia */
cabeça.prox = NULL;
ini = &cabeça;
```

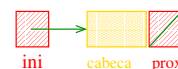


Navigation icons: back, forward, search, etc.

Estrutura de uma lista com cabeça

```
struct celula {
    int conteudo;
    struct celula *prox;
};
typedef struct celula Celula;

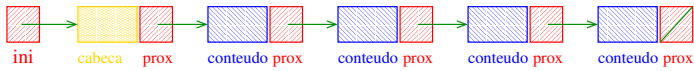
Celula *ini;
/* inicialmente a lista esta vazia */
ini = malloc(sizeof(Celula));
ini->prox = NULL;
```



Navigation icons: back, forward, search, etc.

Imprime conteúdo de uma lista com cabeça

Esta função `imprime` o `conteudo` de cada célula de uma lista encadeada com cabeça `ini`.



```
void imprima (Celula *ini)
{
    Celula *p;
    for (p=ini->prox; p != NULL; p=p->prox)
        printf("%d\n", p->conteudo);
}
```

< > < > < > < > < > < > < >

Busca em uma lista com cabeça

Esta função `recebe` um inteiro `x` e uma lista `ini`. A função `devolve` o endereço de uma célula que contém `x`. Se tal célula não existe, a função `devolve` NULL.

```
Celula* busca (int x, Celula *ini)
{
    Celula *p;
    p = ini->prox;
    while (p != NULL && p->conteudo != x)
        p = p->prox;
    return p;
}
```

< > < > < > < > < > < > < >

Inserção no início de uma lista com cabeça

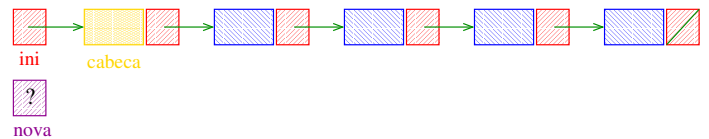
Cria uma célula para guardar um elemento `x` e `insere` esta célula no início da lista com cabeça `ini`.

```
void insere (int x; Celula *ini)
{
    Celula *nova;
    nova = malloc(sizeof(Celula));
    nova->conteudo = x;
    nova->prox = ini->prox;
    ini->prox = nova;
}
```

< > < > < > < > < > < > < >

Inserção no início de uma lista com cabeça

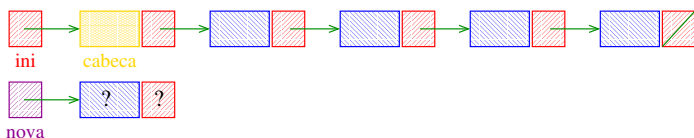
Cria uma célula para guardar um elemento `x` e `insere` esta célula no início da lista com cabeça `ini`.



< > < > < > < > < > < > < >

Inserção no início de uma lista com cabeça

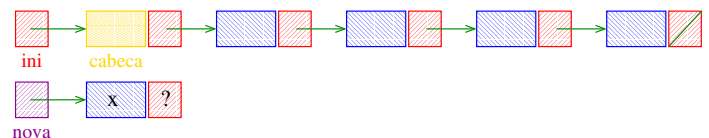
Cria uma célula para guardar um elemento `x` e `insere` esta célula no início da lista com cabeça `ini`.



< > < > < > < > < > < > < >

Inserção no início de uma lista com cabeça

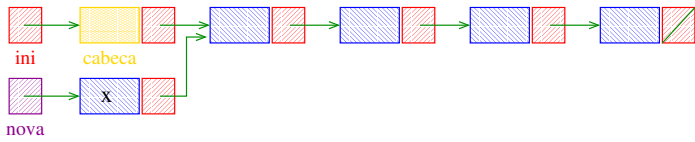
Cria uma célula para guardar um elemento `x` e `insere` esta célula no início da lista com cabeça `ini`.



< > < > < > < > < > < > < >

Inserção no início de uma lista com cabeça

Cria uma célula para guardar um elemento **x** e insere esta célula no início da lista com cabeça **ini**.



Inserção no início de uma lista com cabeça

Cria uma célula para guardar um elemento **x** e insere esta célula no início da lista com cabeça **ini**.

