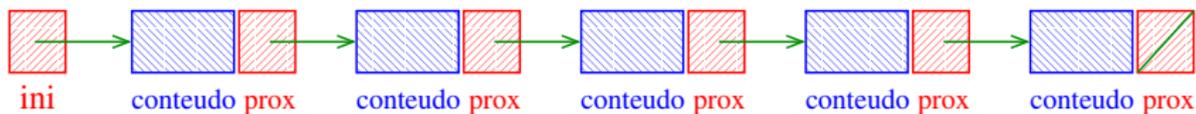


Melhores momentos

AULA 9

Listas

Ilustração de uma lista encadeada



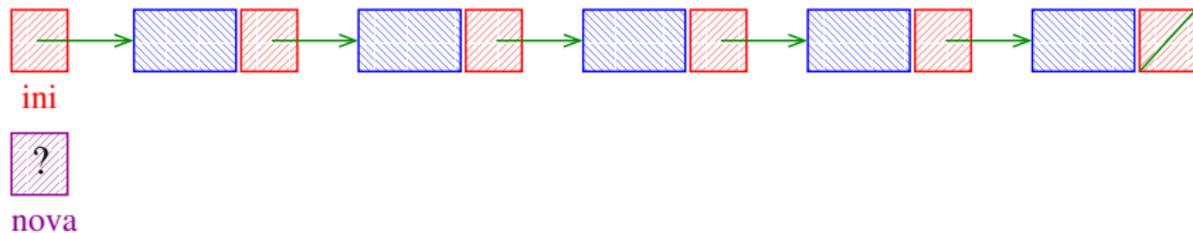
Estrutura de uma lista

```
struct celula {  
    int conteudo;  
    struct celula *prox;  
};  
typedef struct celula Celula;  
  
Celula *ini = NULL; /* inicialize! */  
/* inicialmente a lista esta vazia */
```

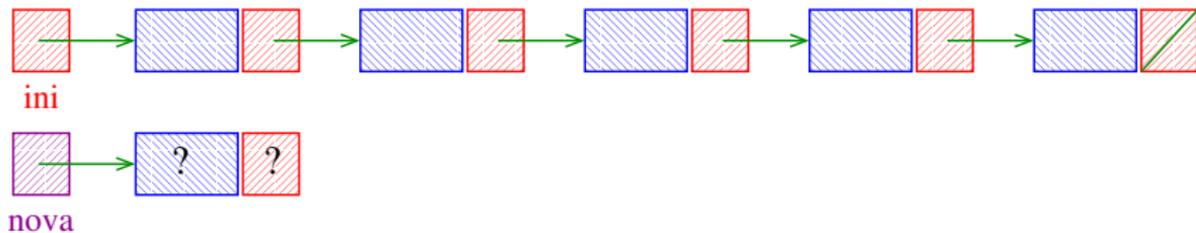


ini

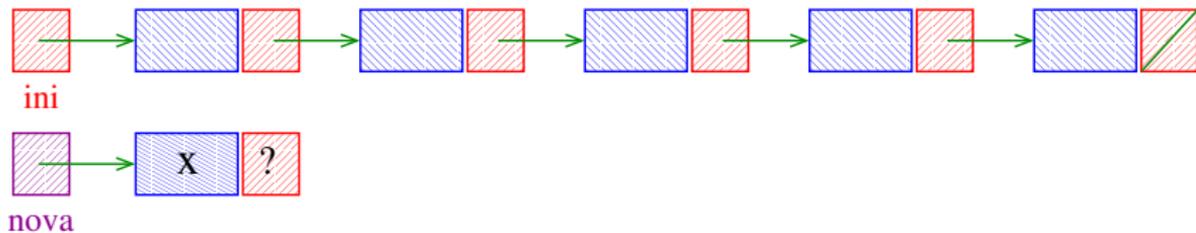
Inserção no início de uma lista



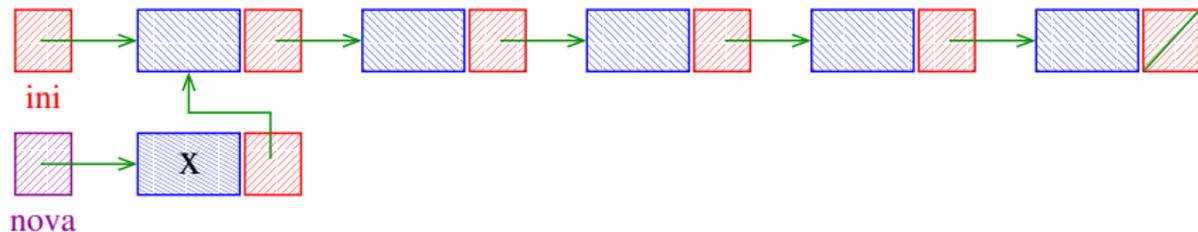
Inserção no início de uma lista



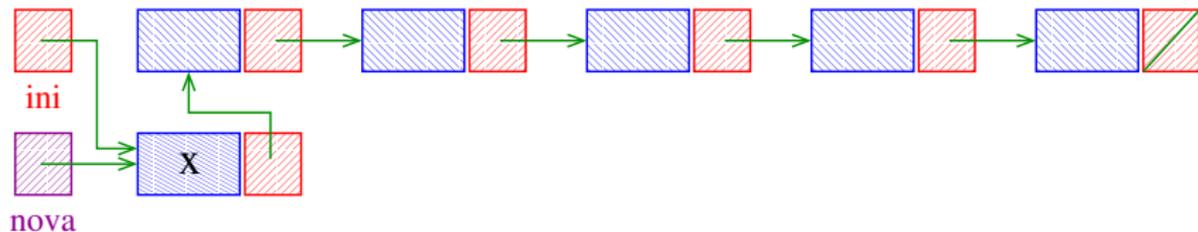
Inserção no início de uma lista



Inserção no início de uma lista



Inserção no início de uma lista



AULA 10

Mais listas encadeadas ainda

PF 4, S 3.3

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

Busca e Remoção em uma lista

Remove, caso exista, a primeira célula da lista **ini** que contém o elemento **x**.

```
Celula *buscaRemove(int x, Celula *ini) {  
    Celula *p, *q;  
    if (ini == NULL) return ini;  
    if (ini->conteudo == x) {  
        q = ini;  
        ini = q->prox;  
        free(q);  
    }  
}
```

Busca e Remoção em uma lista

```
else {  
    p = ini;  
    q = p->prox;  
    while (q!=NULL && q->conteudo!=x){  
        p = q;  
        q = p->prox;  
    }  
    if (q != NULL) {  
        p->prox = q->prox;  
        free(q);  
    }  
}  
return ini;  
}
```

Exemplos de chamadas de buscaRemove

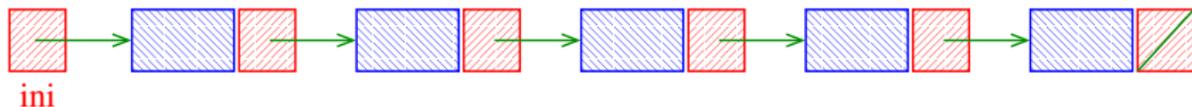
```
Celula *ini, *ini2;  
ini = ini2 = NULL;
```

[...manipulação das listas ...]

```
ini = buscaRemove(22, ini);  
ini2 = buscaRemove(x+1, ini2);  
ini2 = buscaRemove(x+y, ini2);  
ini = buscaRemove(valor, ini);
```

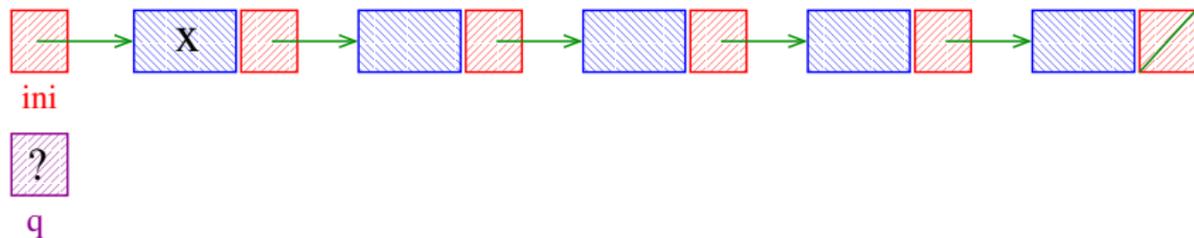
Busca e Remoção em uma lista

Remove, caso exista, a primeira célula da lista **ini** que contém o elemento **x**.



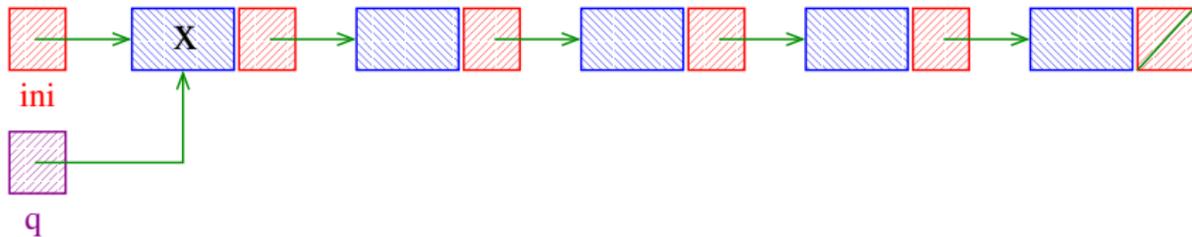
Busca e Remoção em uma lista

Remove, caso exista, a primeira célula da lista **ini** que contém o elemento **x**.



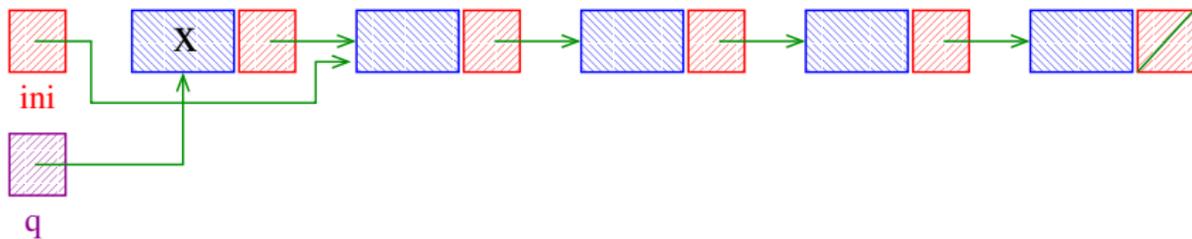
Busca e Remoção em uma lista

Remove, caso exista, a primeira célula da lista **ini** que contém o elemento **x**.



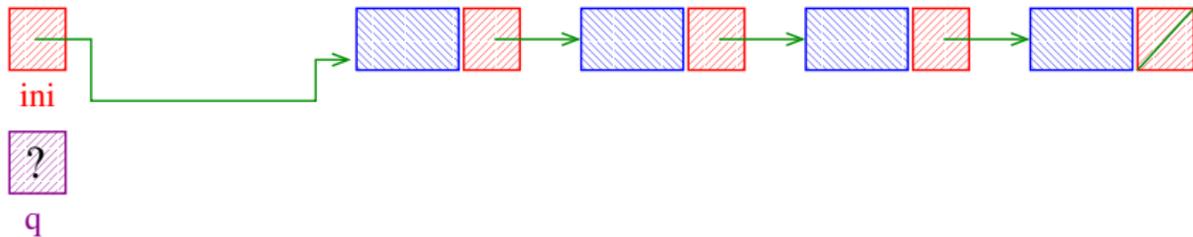
Busca e Remoção em uma lista

Remove, caso exista, a primeira célula da lista **ini** que contém o elemento **x**.



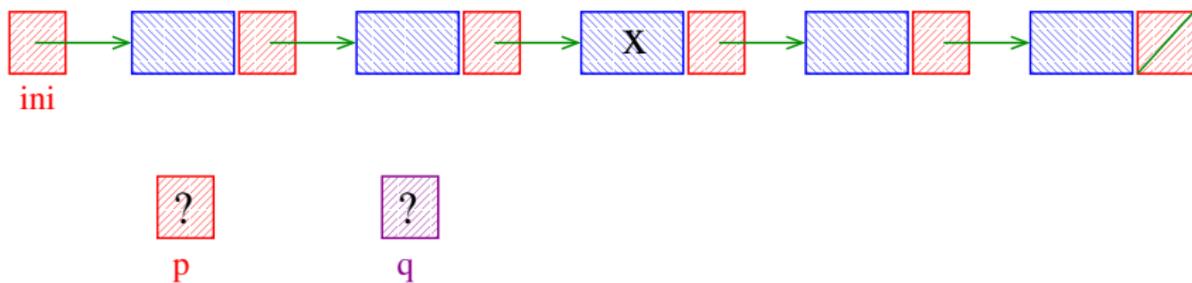
Busca e Remoção em uma lista

Remove, caso exista, a primeira célula da lista **ini** que contém o elemento **x**.



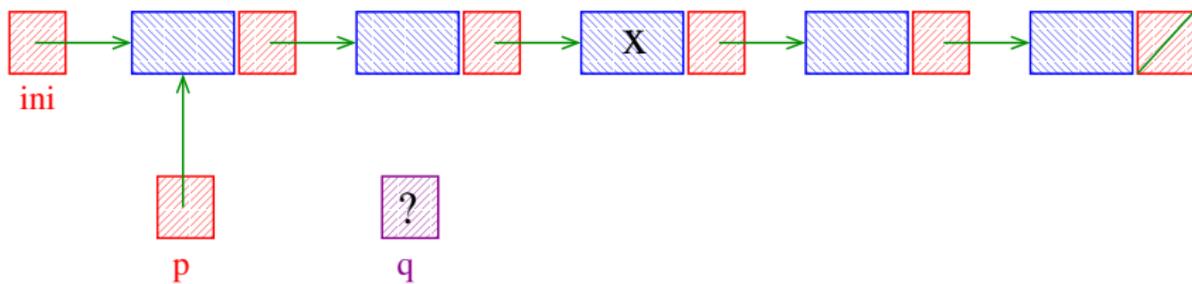
Busca e Remoção em uma lista

Remove, caso exista, a primeira célula da lista **ini** que contém o elemento **x**.



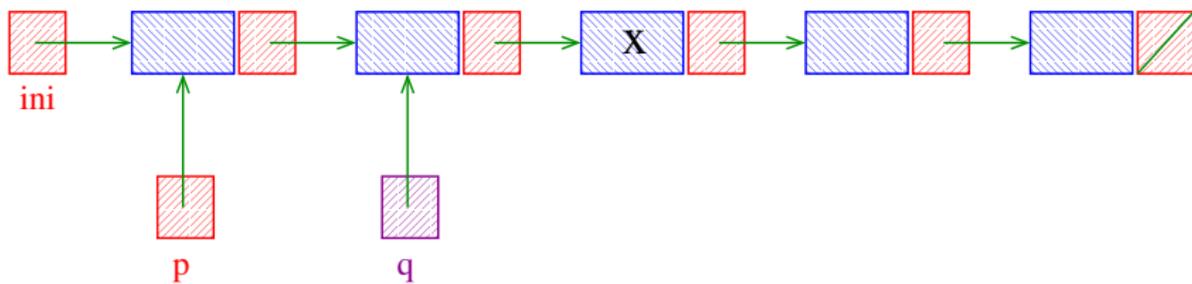
Busca e Remoção em uma lista

Remove, caso exista, a primeira célula da lista **ini** que contém o elemento **x**.



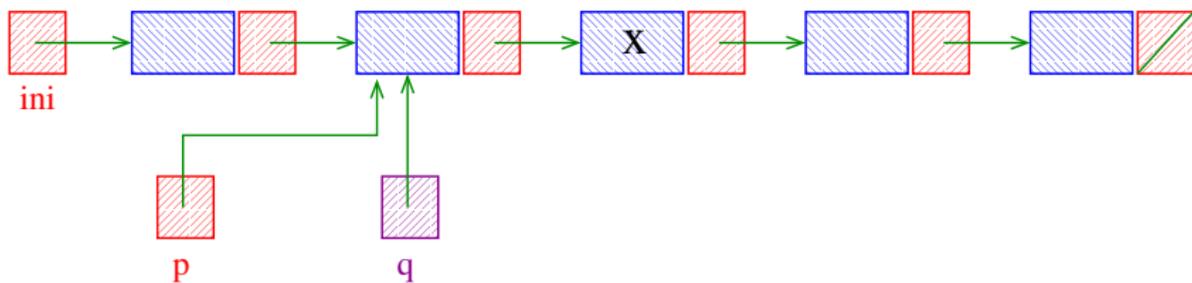
Busca e Remoção em uma lista

Remove, caso exista, a primeira célula da lista **ini** que contém o elemento **x**.



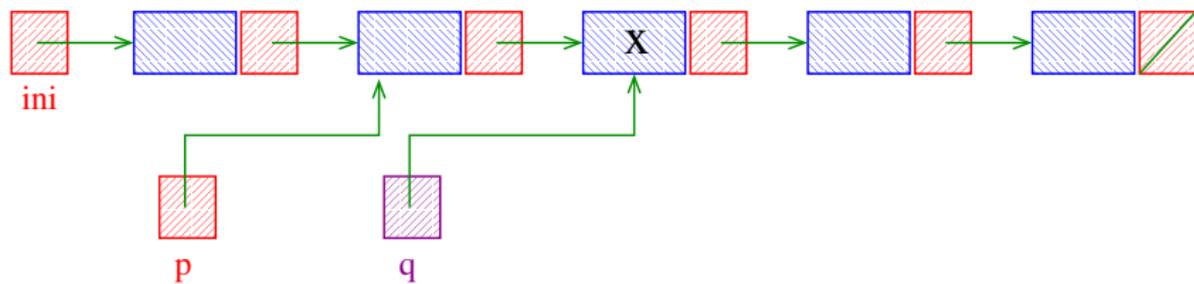
Busca e Remoção em uma lista

Remove, caso exista, a primeira célula da lista **ini** que contém o elemento **x**.



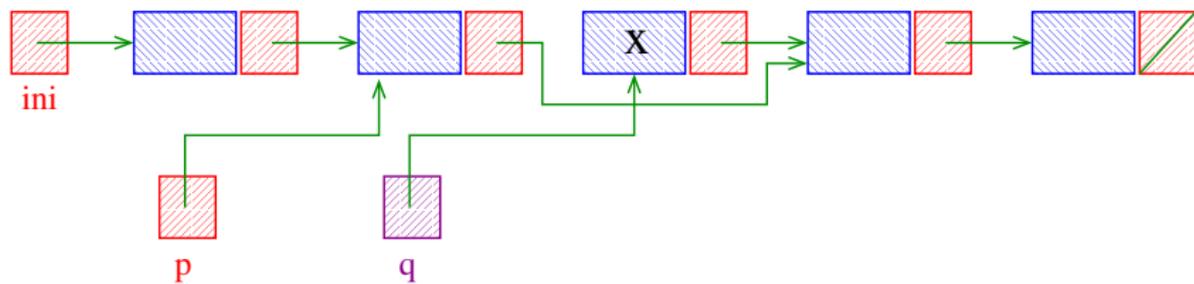
Busca e Remoção em uma lista

Remove, caso exista, a primeira célula da lista **ini** que contém o elemento **x**.



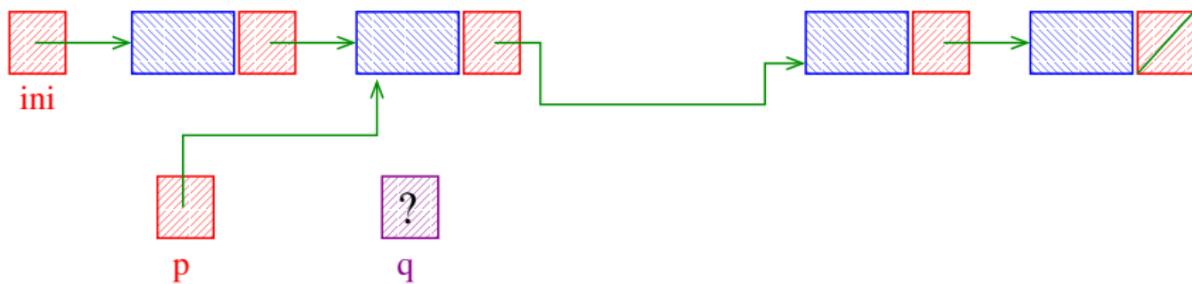
Busca e Remoção em uma lista

Remove, caso exista, a primeira célula da lista **ini** que contém o elemento **x**.



Busca e Remoção em uma lista

Remove, caso exista, a primeira célula da lista **ini** que contém o elemento **x**.



Busca e Remoção em uma lista

Remove, caso exista, a primeira célula da lista **ini** que contém o elemento **x**.

```
void buscaRemove(int x, Celula **ini) {  
    Celula *p, *q;  
    if (*ini == NULL) return;  
    if ((*ini)->conteudo == x) {  
        q = *ini;  
        *ini = q->prox;  
        free(q);  
    }  
}
```

Busca e Remoção em uma lista

```
else {  
    p = *ini;  
    q = p->prox;  
    while (q!=NULL && q->conteudo!=x){  
        p = q;  
        q = p->prox;  
    }  
    if (q != NULL) {  
        p->prox = q->prox;  
        free(q);  
    }  
}  
}
```

Exemplos de chamadas de buscaRemove

```
Celula *ini, *ini2;  
ini = ini2 = NULL;
```

[...manipulação das listas ...]

```
buscaRemove(22, &ini);  
buscaRemove(x+1, &ini2);  
buscaRemove(x+y, &ini2);  
buscaRemove(valor, &ini);
```

Listas com cabeça

PF 4, S 3.3

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

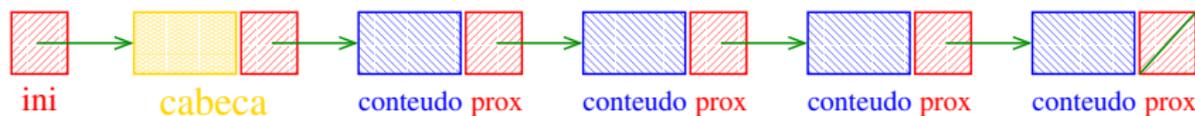
Listas encadeadas com cabeça

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.

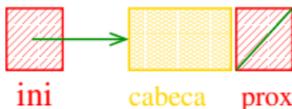
Se **ini** → **prox** == **NULL** se e somente se a **lista está vazia**.

Ilustração de uma **lista encadeada** "com cabeça"



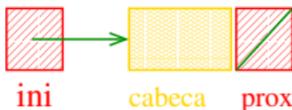
Estrutura de uma lista com cabeça

```
struct celula {  
    int conteudo;  
    struct celula *prox;  
};  
typedef struct celula Celula;  
  
Celula *ini, cabeca;  
/* inicialmente a lista esta vazia */  
cabeca.prox = NULL;  
ini = &cabeca;
```



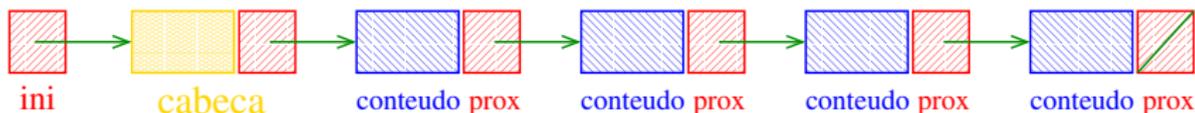
Estrutura de uma lista com cabeça

```
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;
```



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;
}
```

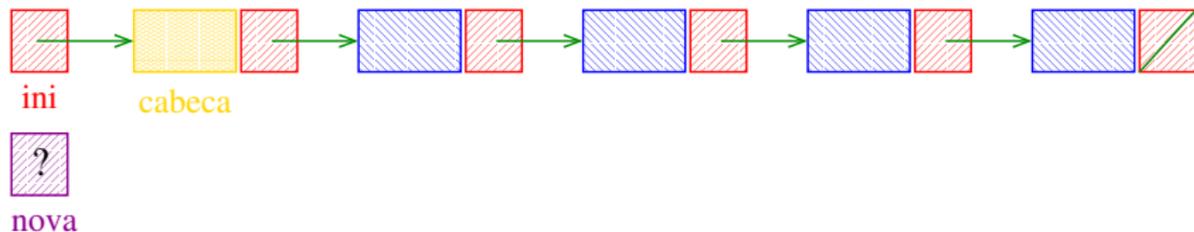
Inserção no início de uma lista com cabeça

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 = mallocSafe(sizeof(Celula));
    nova->conteudo = x;
    nova->prox = ini->prox;
    ini->prox = nova;
}
```

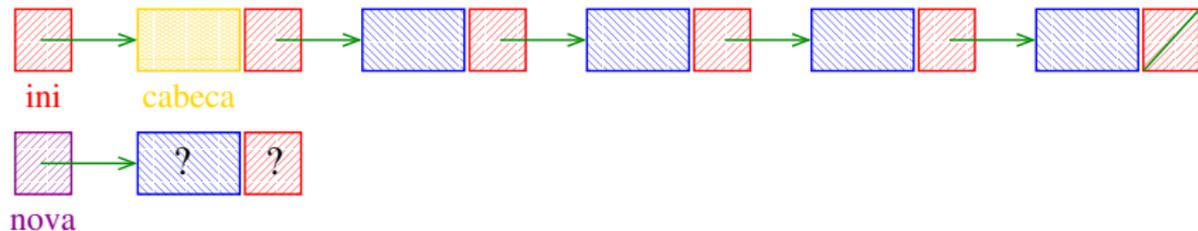
Inserção no início de uma lista com cabeça

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



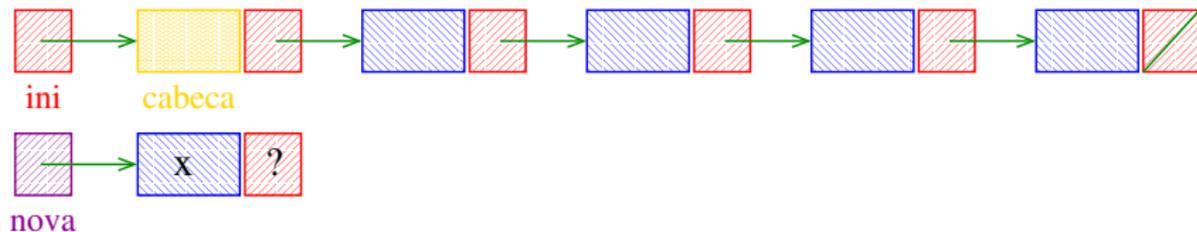
Inserção no início de uma lista com cabeça

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



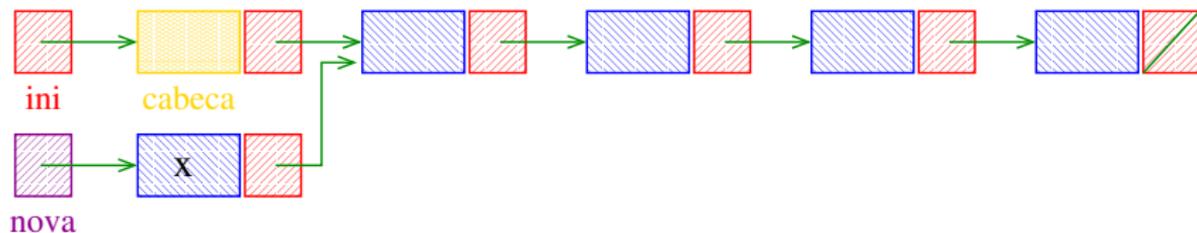
Inserção no início de uma lista com cabeça

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



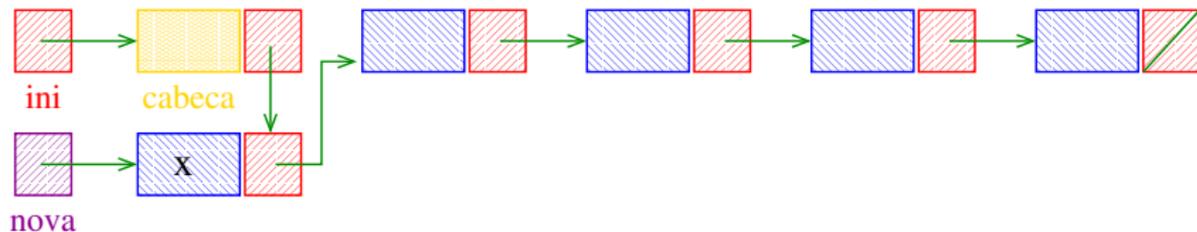
Inserção no início de uma lista com cabeça

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**.



Busca e Remoção em uma lista com cabeça

Remove, caso exista, a primeira célula da lista com cabeça `ini` que contém o elemento `x`.

```
Celula *buscaRemove (int x, Celula *ini){
    Celula *p, *q;
    if (ini == NULL) return ini;
    if (ini->conteudo == x) {
        q = ini;
        ini = q->prox;
        free(q);
    }
}
```

Busca e Remoção em uma lista com cabeça

Remove, caso exista, a primeira célula da lista com cabeça `ini` que contém o elemento `x`.

```
Celula *buscaRemove (int x, Celula *ini){  
    Celula *p, *q;  
    if (ini == NULL) return ini;  
    if (ini->conteudo == x) {  
        q = ini;  
        ini = q->prox;  
        free(q);  
    }
```

Busca e Remoção em uma lista com cabeça

```
else {  
    p = ini;  
    q = p->prox;  
    while (q!=NULL && q->conteudo!=x){  
        p = q;  
        q = p->prox;  
    }  
    if (q != NULL) {  
        p->prox = q->prox;  
        free(q);  
    }  
}  
return ini;  
}
```

Busca e Remoção em uma lista com cabeça

```
else {  
    p = ini;  
    q = p->prox;  
    while (q != NULL && q->conteudo != x) {  
        p = q;  
        q = p->prox;  
    }  
    if (q != NULL) {  
        p->prox = q->prox;  
        free(q);  
    }  
}  
return ini;
```

```
}
```

Busca e Remoção em uma lista com cabeça

```
void buscaRemove (int x, Celula *ini) {  
    Celula *p, *q;  
    p = ini;  
    q = p->prox;  
    while (q!=NULL && q->conteudo!=x){  
        p = q;  
        q = p->prox;  
    }  
    if (q != NULL) {  
        p->prox = q->prox;  
        free(q);  
    }  
}
```

Exemplos de chamadas de buscaRemove

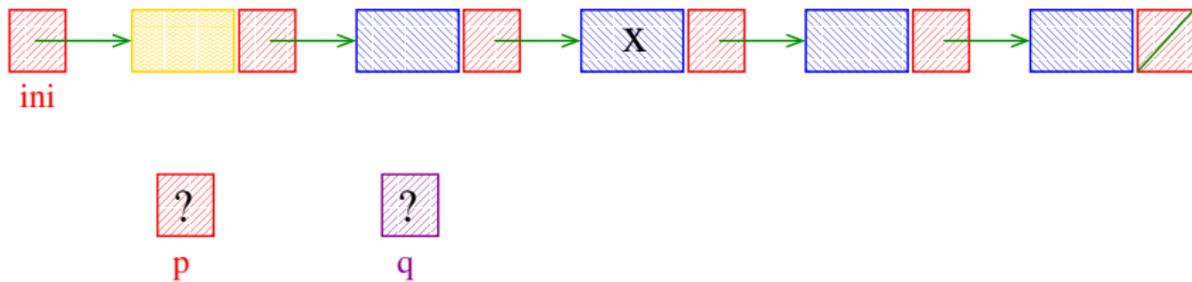
```
Celula *ini, *ini2;  
Celula cabeca;  
ini = &cabeca  
cabeca.prox = NULL;  
ini2 = mallocSafe(sizeof(Celula));  
ini2->prox = NULL;
```

[...manipulação das listas ...]

```
buscaRemove(22,&cabeca);  
buscaRemove(33,ini);  
buscaRemove(x+1,ini2);  
buscaRemove(x+y,ini2);  
buscaRemove(valor,ini);
```

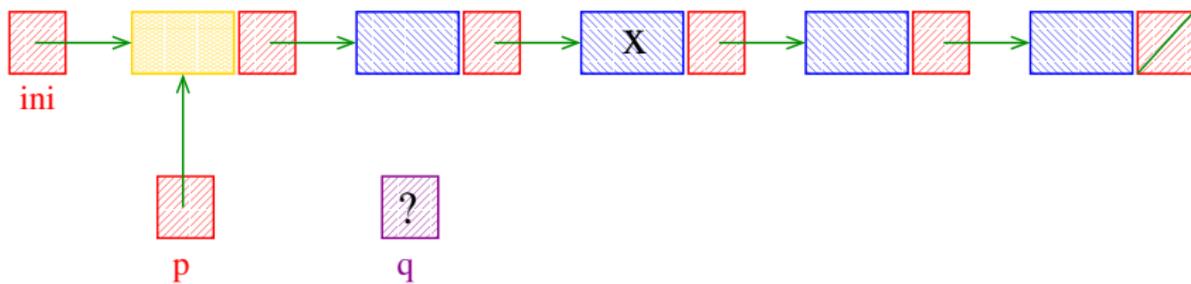
Busca e Remoção em uma lista com cabeça

Remove, caso exista, a primeira célula da lista com cabeça *ini* que contém o elemento *x*.



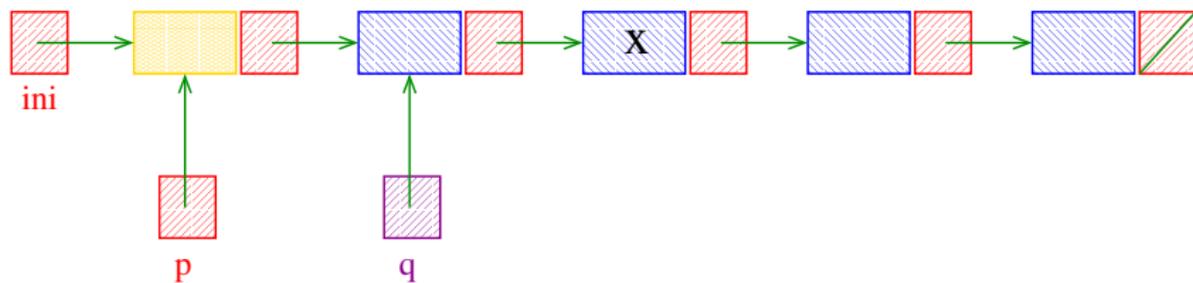
Busca e Remoção em uma lista com cabeça

Remove, caso exista, a primeira célula da lista com cabeça *ini* que contém o elemento *x*.



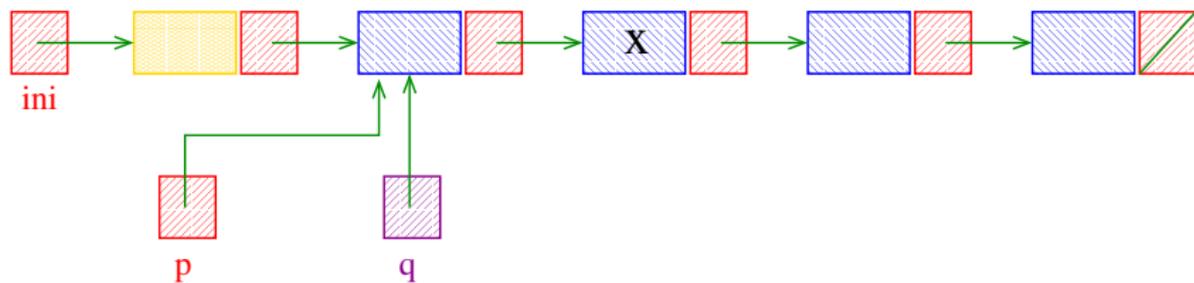
Busca e Remoção em uma lista com cabeça

Remove, caso exista, a primeira célula da lista com cabeça *ini* que contém o elemento *x*.



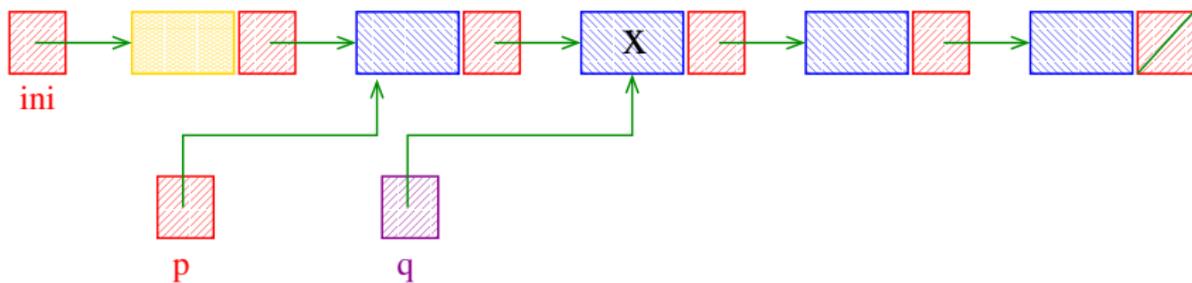
Busca e Remoção em uma lista com cabeça

Remove, caso exista, a primeira célula da lista com cabeça *ini* que contém o elemento *x*.



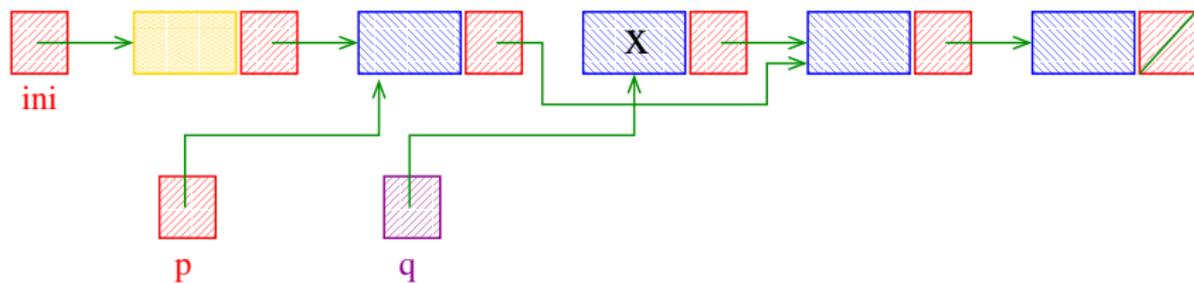
Busca e Remoção em uma lista com cabeça

Remove, caso exista, a primeira célula da lista com cabeça *ini* que contém o elemento *x*.



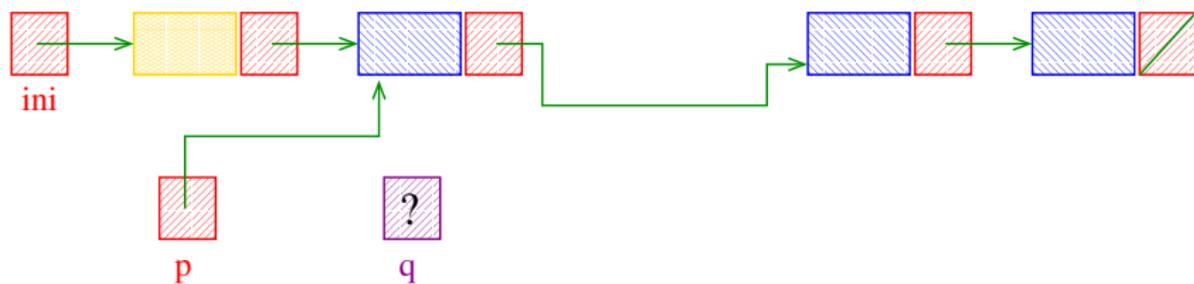
Busca e Remoção em uma lista com cabeça

Remove, caso exista, a primeira célula da lista com cabeça *ini* que contém o elemento *x*.



Busca e Remoção em uma lista com cabeça

Remove, caso exista, a primeira célula da lista com cabeça `ini` que contém o elemento `x`.



Busca e Inserção em uma lista com cabeça

Recebe uma lista com cabeça `ini` e insere uma célula de conteúdo `x` antes da primeira célula de conteúdo `y`. Se nenhuma célula contém `y`, insere a célula com `y` no final da lista.

`Celula *`

```
buscaInsere(int x, int y, Celula *ini) {  
    Celula *p, *q, *nova;  
    nova = mallocSafe(sizeof(Celula));  
    nova->conteudo = x;  
    if (ini == NULL || ini->conteudo == y){  
        nova->prox = ini;  
        ini = nova;  
    }  
}
```

Busca e Inserção em uma lista com cabeça

Recebe uma lista com cabeça `ini` e insere uma célula de conteúdo `x` antes da primeira célula de conteúdo `y`. Se nenhuma célula contém `y`, insere a célula com `y` no final da lista.

`Celula *`

```
buscaInsere(int x, int y, Celula *ini) {  
    Celula *p, *q, *nova;  
    nova = mallocSafe(sizeof(Celula));  
    nova->conteudo = x;  
if (ini == NULL || ini->conteudo == y){  
        nova->prox = ini;  
        ini = nova;  
}
```

Busca e Inserção em uma lista com cabeça

```
else {  
    p = ini;  
    q = p->prox;  
    while (q!=NULL && q->conteudo!=y){  
        p = q;  
        q = p->prox;  
    }  
    p->prox = nova;  
    nova->prox = q;  
}  
return ini;  
}
```

Busca e Inserção em uma lista com cabeça

```
else {  
    p = ini;  
    q = p->prox;  
    while (q!=NULL && q->conteudo!=y) {  
        p = q;  
        q = p->prox;  
    }  
    p->prox = nova;  
    nova->prox = q;  
}  
return ini;  
}
```

Busca e Inserção em uma lista com cabeça

void

```
buscaInsere(int x, int y, Celula *ini){
    Celula *p, *q, *nova;
    nova = mallocSafe(sizeof(Celula));
    nova->conteudo = x;
    p = ini;
    q = p->prox;
    while (q!=NULL && q->conteudo!=y){
        p = q;
        q = p->prox;
    }
    p->prox = nova;
    nova->prox = q;
```

}

Exemplos de chamadas de buscaInsere

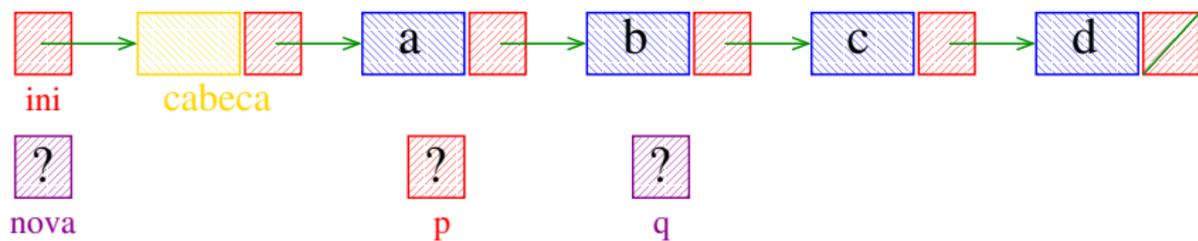
```
Celula *ini, *ini2;  
Celula cabeca;  
ini = &cabeca  
cabeca.prox = NULL;  
ini2 = mallocSafe(sizeof(Celula));  
ini2->prox = NULL;
```

[...manipulação das listas ...]

```
buscaInsere(22,24,&cabeca);  
buscaInsere(33,-10,ini);  
buscaInsere(x+1,y-5,ini2);  
buscaInsere(x+y,ini2);  
buscaInsere(valor,ini);
```

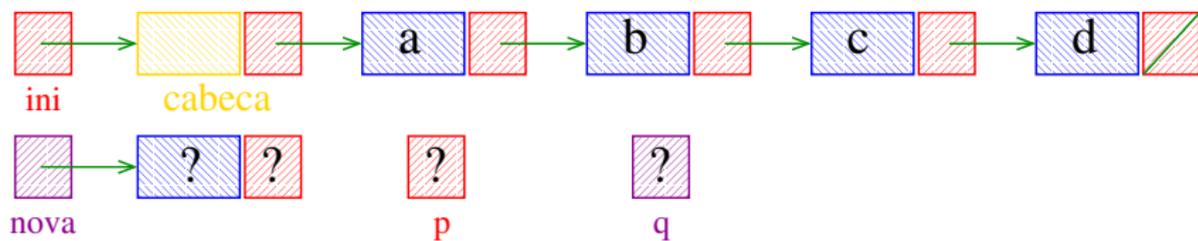
Busca e Inserção em uma lista com cabeça

Recebe uma lista com cabeça *ini* e insere uma célula de conteúdo *x* antes da primeira célula de conteúdo *y*. Se nenhuma célula contém *y*, insere a célula com *y* no final da lista.



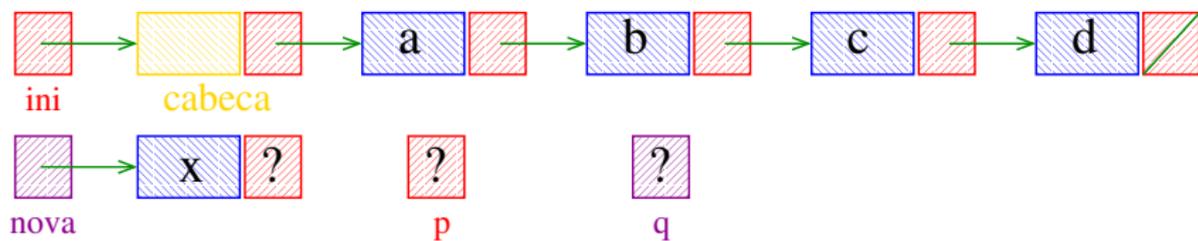
Busca e Inserção em uma lista com cabeça

Recebe uma lista com cabeça *ini* e insere uma célula de conteúdo *x* antes da primeira célula de conteúdo *y*. Se nenhuma célula contém *y*, insere a célula com *y* no final da lista.



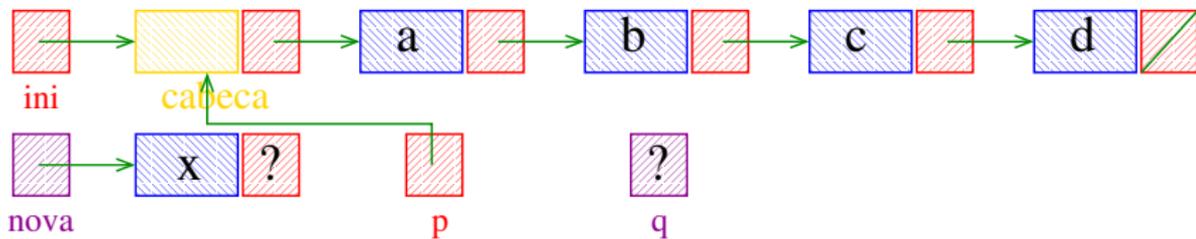
Busca e Inserção em uma lista com cabeça

Recebe uma lista com cabeça *ini* e insere uma célula de conteúdo *x* antes da primeira célula de conteúdo *y*. Se nenhuma célula contém *y*, insere a célula com *y* no final da lista.



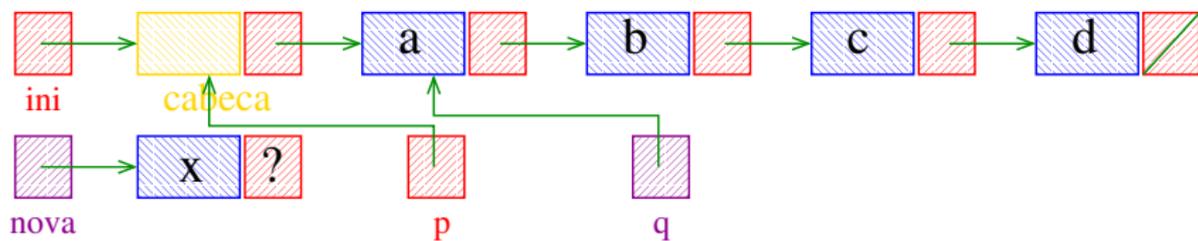
Busca e Inserção em uma lista com cabeça

Recebe uma lista com cabeça *ini* e insere uma célula de conteúdo *x* antes da primeira célula de conteúdo *y*. Se nenhuma célula contém *y*, insere a célula com *y* no final da lista.



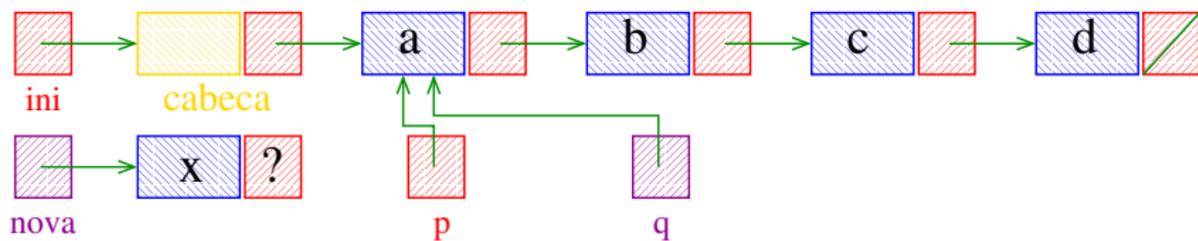
Busca e Inserção em uma lista com cabeça

Recebe uma lista com cabeça *ini* e insere uma célula de conteúdo *x* antes da primeira célula de conteúdo *y*. Se nenhuma célula contém *y*, insere a célula com *y* no final da lista.



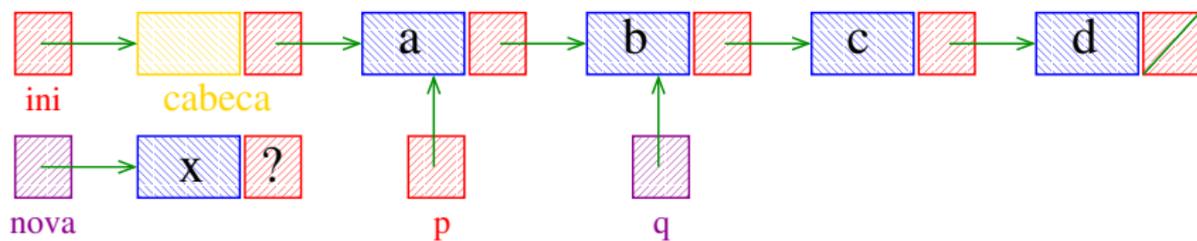
Busca e Inserção em uma lista com cabeça

Recebe uma lista com cabeça *ini* e insere uma célula de conteúdo *x* antes da primeira célula de conteúdo *y*. Se nenhuma célula contém *y*, insere a célula com *y* no final da lista.



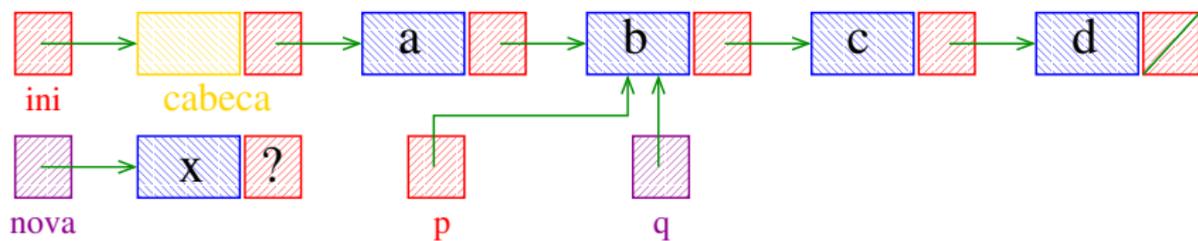
Busca e Inserção em uma lista com cabeça

Recebe uma lista com cabeça *ini* e insere uma célula de conteúdo *x* antes da primeira célula de conteúdo *y*. Se nenhuma célula contém *y*, insere a célula com *y* no final da lista.



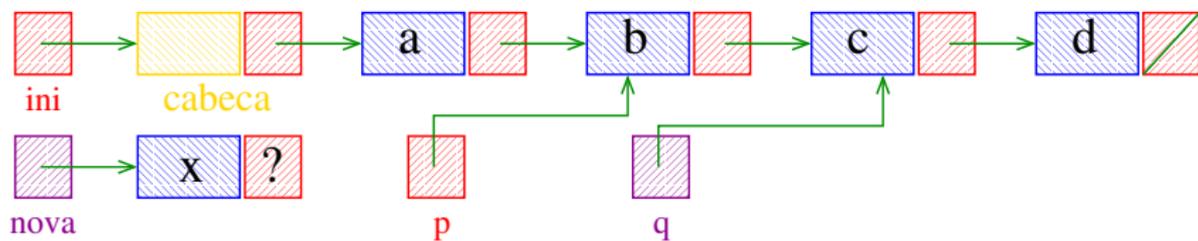
Busca e Inserção em uma lista com cabeça

Recebe uma lista com cabeça *ini* e insere uma célula de conteúdo *x* antes da primeira célula de conteúdo *y*. Se nenhuma célula contém *y*, insere a célula com *y* no final da lista.



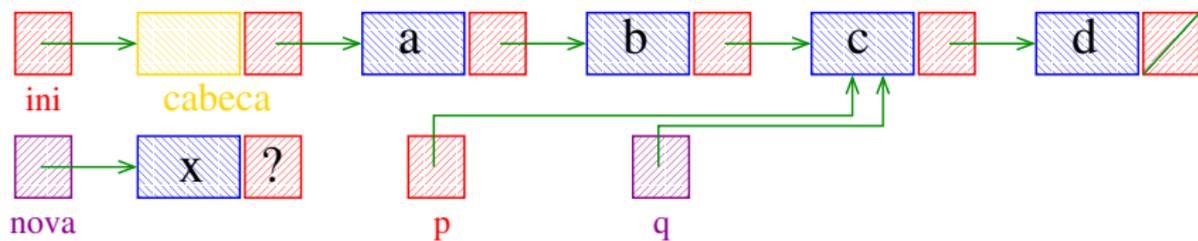
Busca e Inserção em uma lista com cabeça

Recebe uma lista com cabeça *ini* e insere uma célula de conteúdo *x* antes da primeira célula de conteúdo *y*. Se nenhuma célula contém *y*, insere a célula com *y* no final da lista.



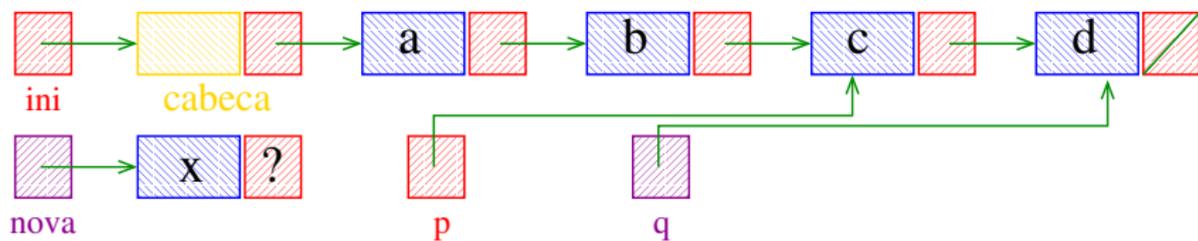
Busca e Inserção em uma lista com cabeça

Recebe uma lista com cabeça *ini* e insere uma célula de conteúdo *x* antes da primeira célula de conteúdo *y*. Se nenhuma célula contém *y*, insere a célula com *y* no final da lista.



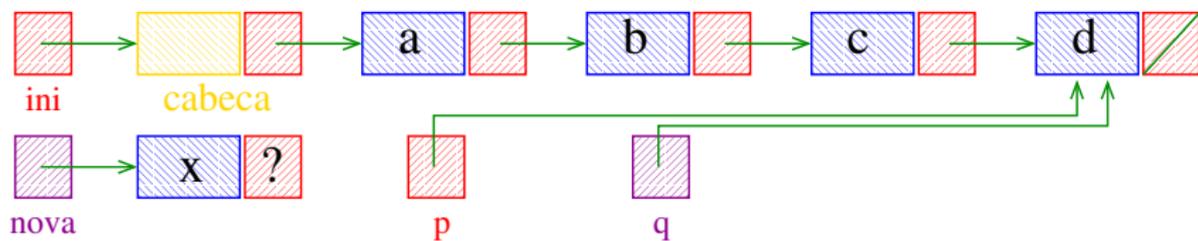
Busca e Inserção em uma lista com cabeça

Recebe uma lista com cabeça *ini* e insere uma célula de conteúdo *x* antes da primeira célula de conteúdo *y*. Se nenhuma célula contém *y*, insere a célula com *y* no final da lista.



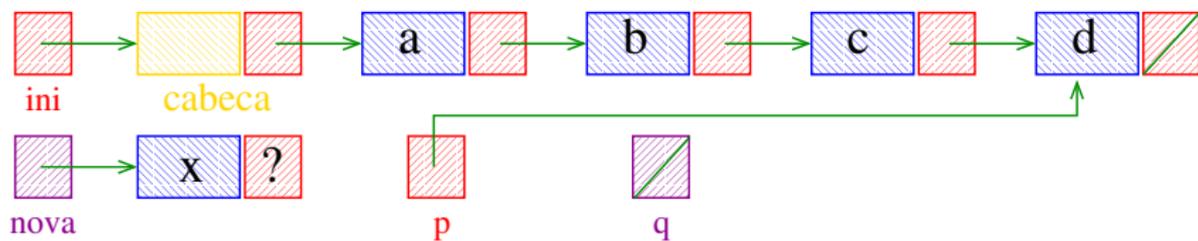
Busca e Inserção em uma lista com cabeça

Recebe uma lista com cabeça *ini* e insere uma célula de conteúdo *x* antes da primeira célula de conteúdo *y*. Se nenhuma célula contém *y*, insere a célula com *y* no final da lista.



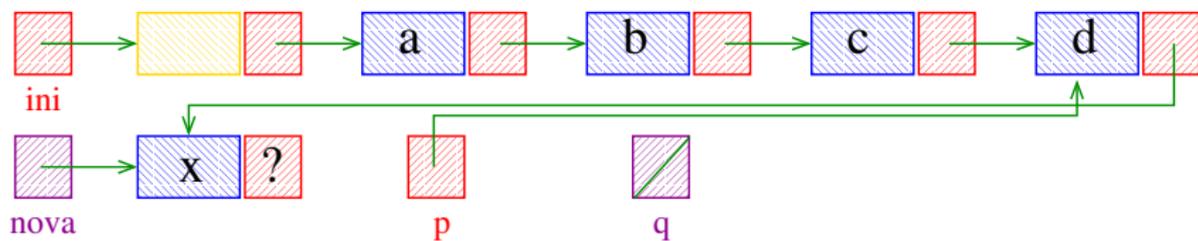
Busca e Inserção em uma lista com cabeça

Recebe uma lista com cabeça *ini* e insere uma célula de conteúdo *x* antes da primeira célula de conteúdo *y*. Se nenhuma célula contém *y*, insere a célula com *y* no final da lista.



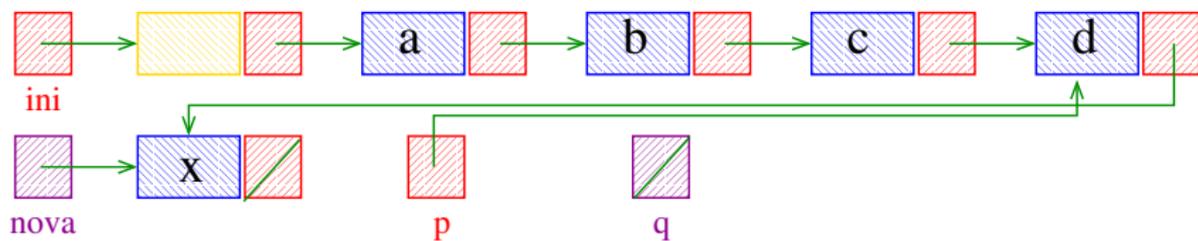
Busca e Inserção em uma lista com cabeça

Recebe uma lista com cabeça *ini* e insere uma célula de conteúdo *x* antes da primeira célula de conteúdo *y*. Se nenhuma célula contém *y*, insere a célula com *y* no final da lista.



Busca e Inserção em uma lista com cabeça

Recebe uma lista com cabeça *ini* e insere uma célula de conteúdo *x* antes da primeira célula de conteúdo *y*. Se nenhuma célula contém *y*, insere a célula com *y* no final da lista.



Busca e Inserção em uma lista com cabeça

Recebe uma lista com cabeça *ini* e insere uma célula de conteúdo *x* antes da primeira célula de conteúdo *y*. Se nenhuma célula contém *y*, insere a célula com *y* no final da lista.

