



DEI
DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
TÉCNICO LISBOA

Estruturas auto-referenciadas & Listas

K&R: Capítulo 6

IAED, 2013/2014



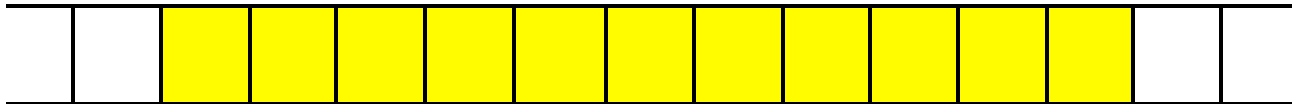
Sinopse

- **Estruturas Auto-Referenciadas: das tabelas às listas**
 - Exemplo de aplicação:
 - Listas de inteiros
 - Pilhas de inteiros
 - Listas de strings
-
- Regras de Scope (“âmbito” de variáveis em C)
 - Abstract Data Types (ADTs)

Tabelas/Vectores

- Colecção de items
 - Inteiros, reais, caracteres
 - Estruturas
 - Tabelas, Ponteiros
- Guardados em posições consecutivas de memória

`int tab[N];`



- Programador é responsável por respeitar limites

Tabelas/Vetores

- Em C tabelas podem ser
 - De dimensão fixa
 - Alocadas dinamicamente

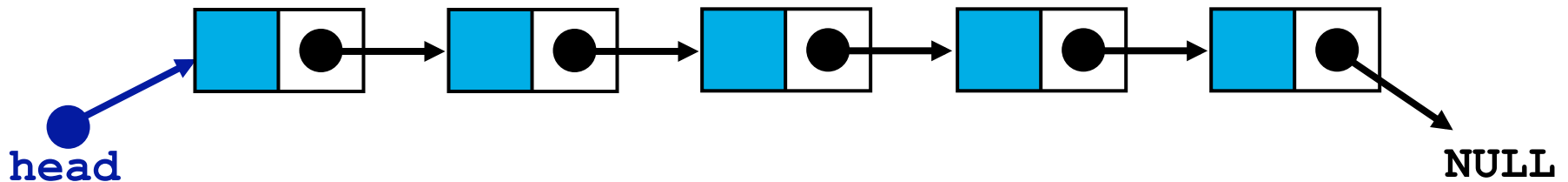
```
#define N 100
int tab1[N];
int *tab2 = (int *) malloc(N*sizeof(int));
```

- Acesso alternativo a tabelas

```
x = tab2[i];
y = *(tab2+i);
```

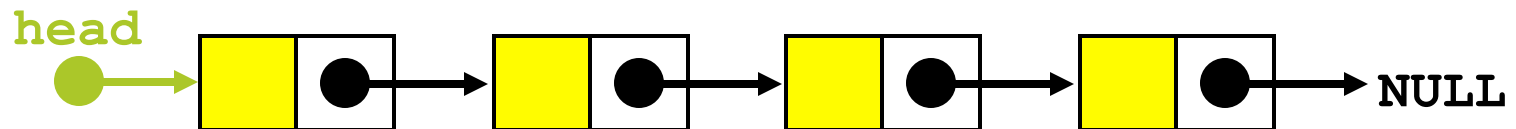
Tabelas/Vectores

- Vantagens
 - Manipulação simples
 - Facilidade de acesso ao n-ésimo elemento
- Desvantagens
 - Tamanho limitado
 - Necessidade de realocar e copiar todos os elementos se desejar aumentar dimensão da tabela
 - Desperdício de memória
- Alternativa: *Listas*



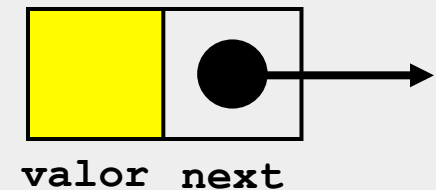
Lista Simplesmente Ligada

- Conjunto de nós



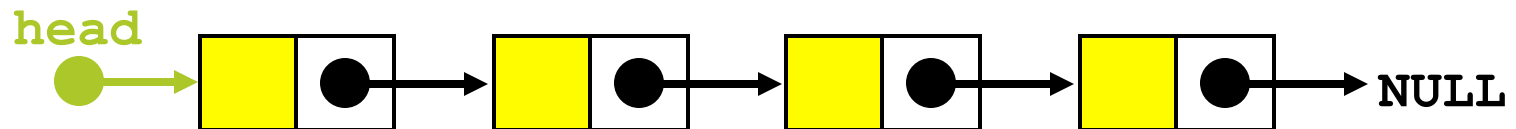
- Cada nó contém
 - Informação útil
 - Ponteiro para o próximo nó

```
struct node {  
    int valor;  
    struct node *next;  
};
```



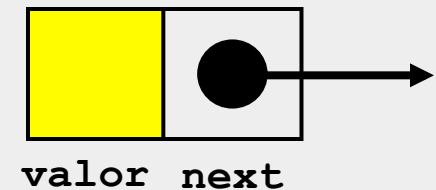
Lista Simplesmente Ligada

- Conjunto de nós



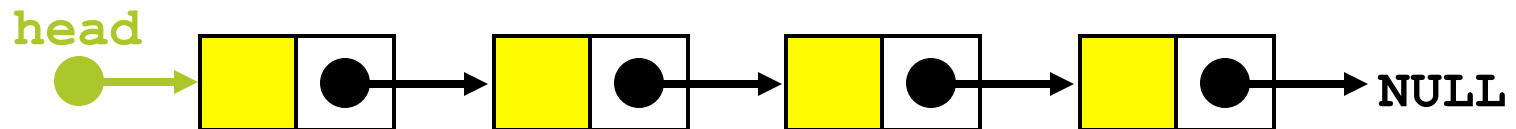
- Cada nó contém
 - Informação útil
 - Ponteiro para o próximo nó

```
struct node {  
    int valor;  
    struct node *next;  
};
```



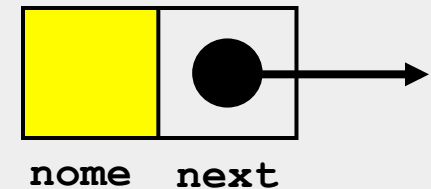
Lista Simplesmente Ligada

- Conjunto de nós



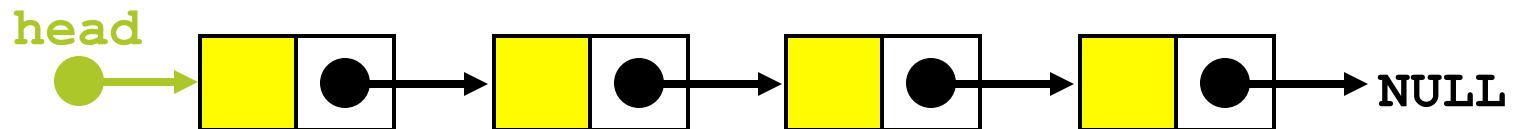
- Cada nó contém
 - Informação útil
 - Ponteiro para o próximo nó

```
struct node {  
    char nome[256];  
    struct node *next;  
};
```



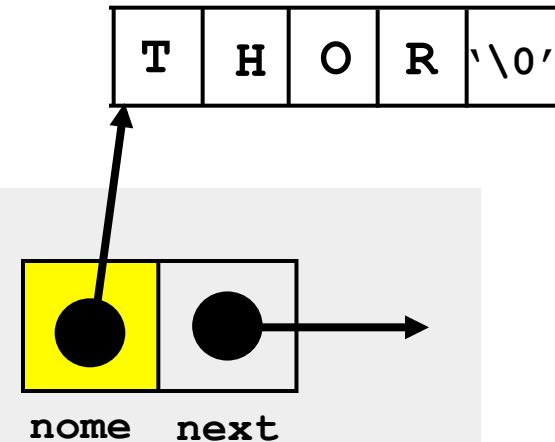
Lista Simplesmente Ligada

- Conjunto de nós



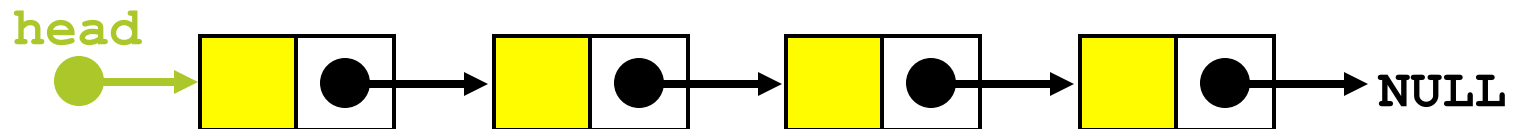
- Cada nó contém
 - Informação útil
 - Ponteiro para o próximo nó

```
struct node {  
    char *nome;  
    struct node *next;  
};
```



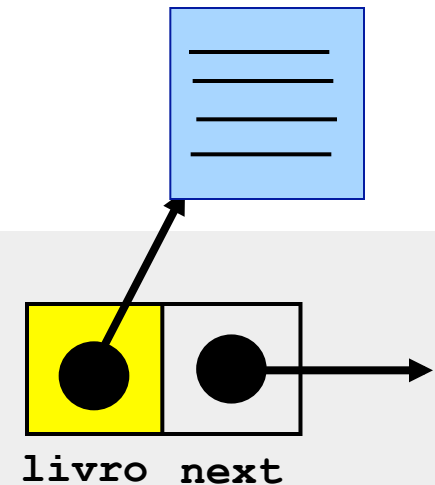
Lista Simplesmente Ligada

- Conjunto de nós



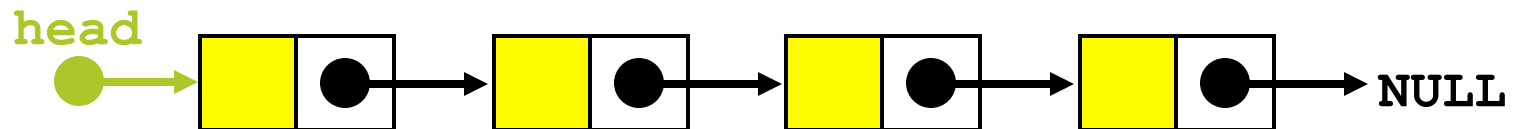
- Cada nó contém
 - Informação útil
 - Ponteiro para o próximo nó

```
struct node {  
    Livro *livro;  
    struct node *next;  
};
```



Lista Simplesmente Ligada

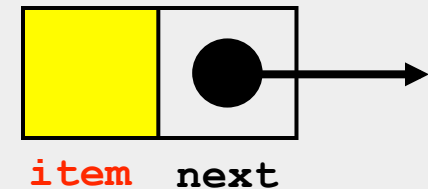
- Conjunto de nós



- Cada nó contém
 - Informação útil
 - Ponteiro para o próximo nó

*A nossa
abstração
habitual*

```
struct node {  
    Item item;  
    struct node *next;  
};
```



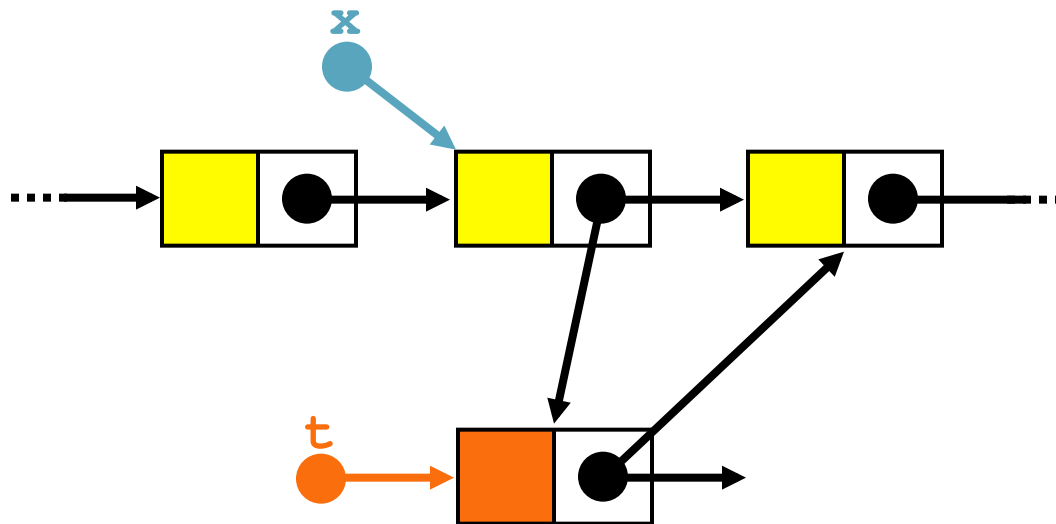


Listas

- Vantagens
 - Tamanho ilimitado (limite na memória disponível na máquina)
 - Alocação de memória apenas para os elementos que queremos representar
 - Inserção e remoção simples
- Desvantagens
 - Mais difícil o acesso ao n -ésimo elemento

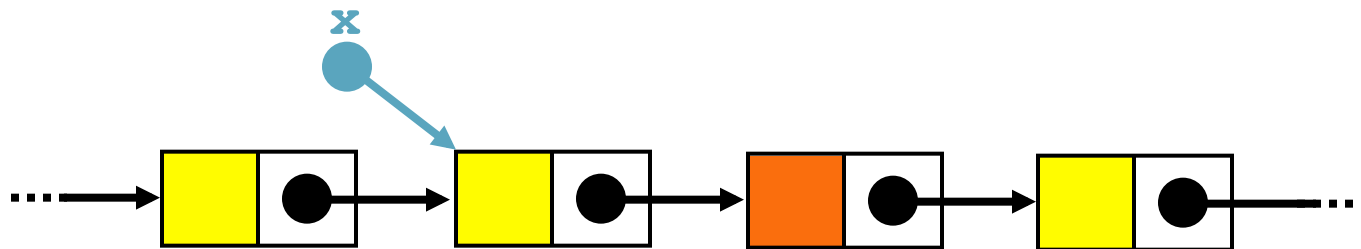
Inserção na Lista de um elemento *t depois de *x

```
t->next = x->next;  
x->next = t;
```



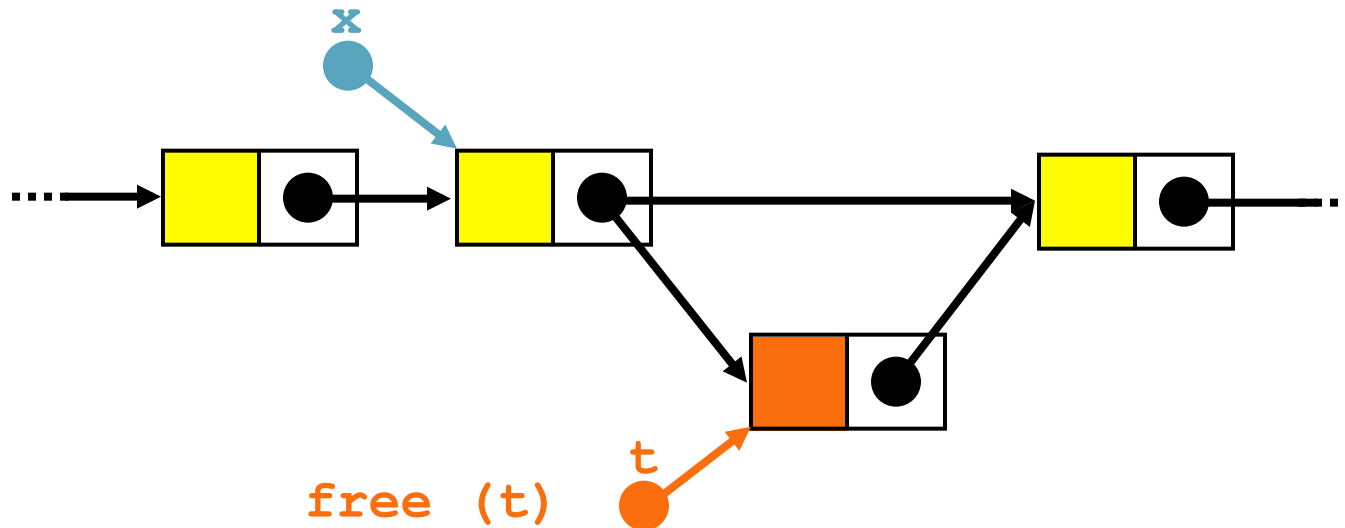
Inserção na Lista

```
t->next = x->next;  
x->next = t;
```



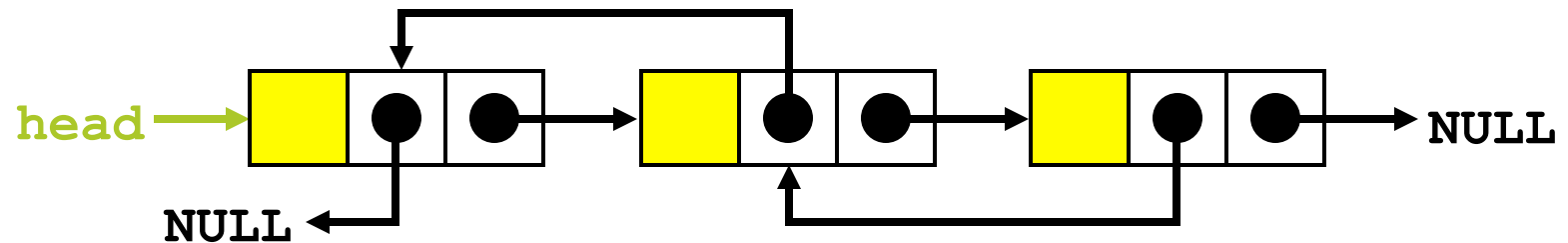
Remoção do elemento depois de x da Lista

```
t=x->next  
x->next = t->next;  
free(t);
```



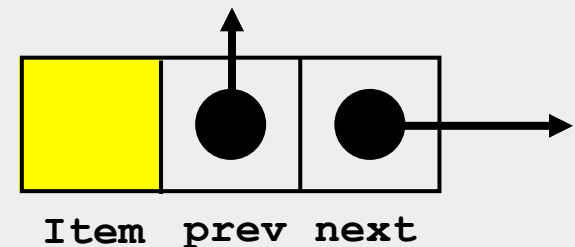
Lista Duplamente Ligada

- Conjunto de nós



- Cada nó contém
 - Informação útil
 - Ponteiro para o próximo nó e para o nó anterior

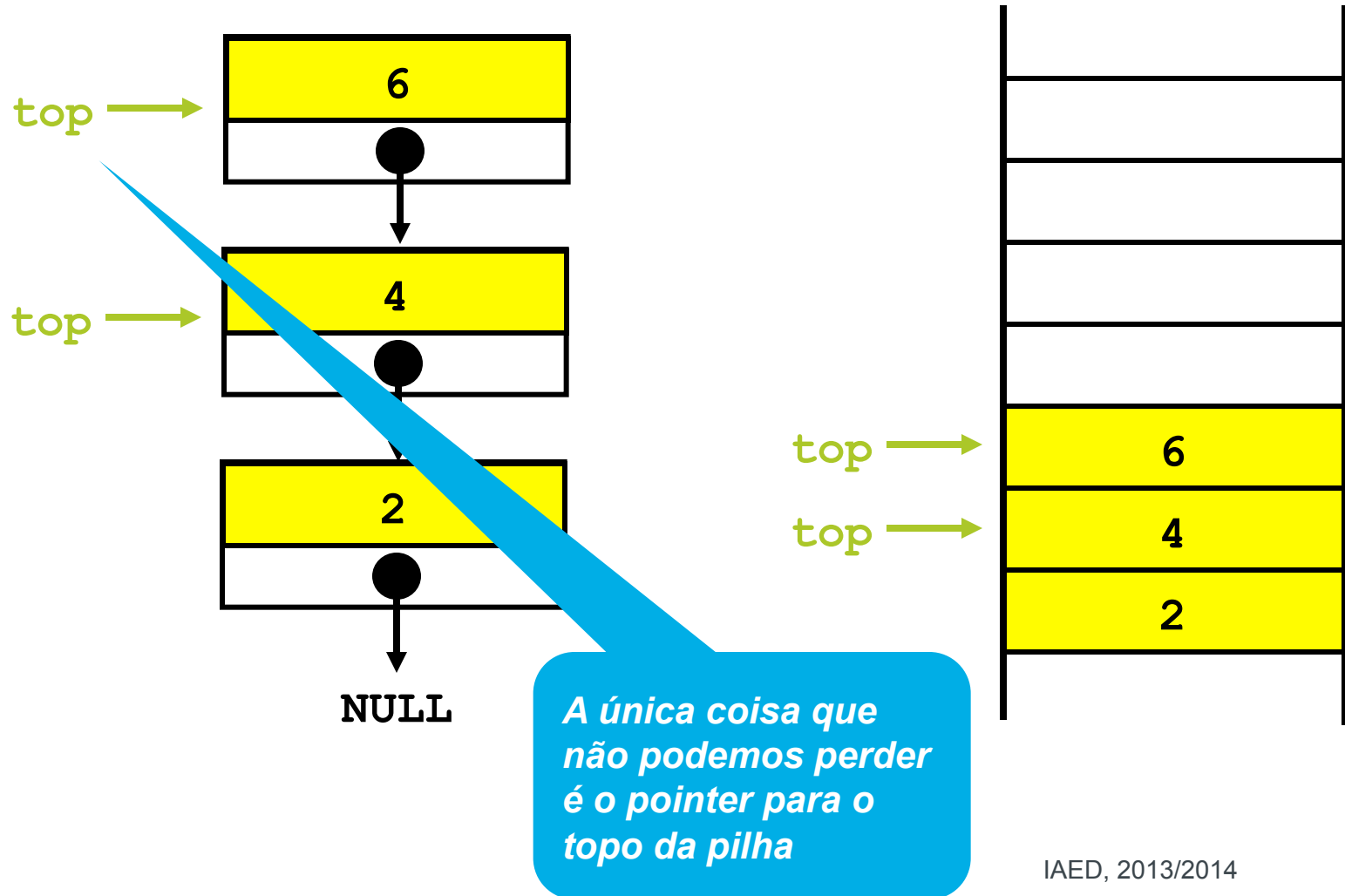
```
struct node {  
    Item item;  
    struct node *prev, *next;  
};
```



Sinopse

- Estruturas Auto-Referenciadas: das tabelas às listas
 - **Exemplo de aplicação:**
 - **Listas de inteiros**
 - Pilhas de inteiros
 - Listas de strings
-
- Regras de Scope (“âmbito” de variáveis em C)
 - Abstract Data Types (ADTs)

Exemplo: Pilha Dinâmica de Inteiros



Exemplo: Pilha Dinâmica de Inteiros

```
#include <stdio.h>
#include <stdlib.h>
```

```
struct node{
    int value;
    struct node*next;
};
```

Variável global

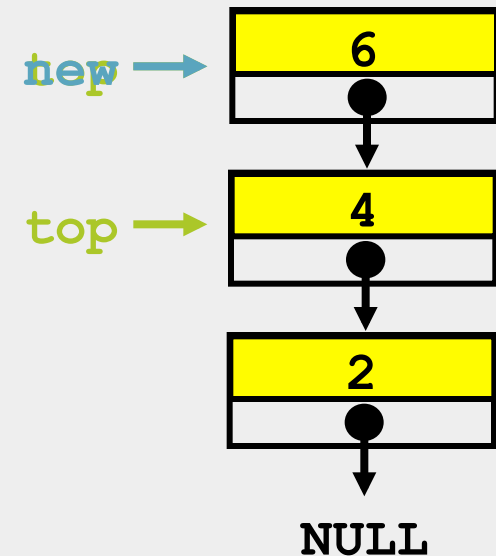
```
static struct node *top;
```

```
void init() /* inicializa a pilha */
{
    top = NULL;
}
```

Exemplo: Pilha Dinâmica de Inteiros

```
void push(int value) /* introduz novo elemento no topo */
{
    struct node *new;

    new = (struct node *) malloc(sizeof(struct node));
    new->value = value;
    new->next = top;
    top = new;
}
```



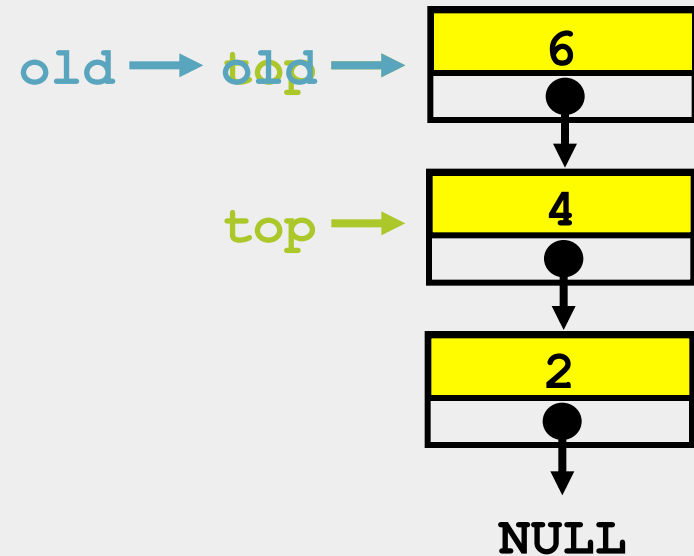
Exemplo: Pilha Dinâmica de Inteiros

```
int is_empty() /* pergunta se está vazia */  
{  
    return top == NULL;  
}
```

Exemplo: Pilha Dinâmica de Inteiros

```
int pop() /* apaga o topo e retorna o valor apagado */
{
    int value;
    struct node *old;

    if (!is_empty()) {
        value = top->value;
        old = top;
        top = top->next;
        free(old);
        return value;
    }
    else
        return -1;
}
```





Exemplo: Pilha Dinâmica de Inteiros

Problema:

- Como obter o número de elementos da Pilha?

Solução:

- Necessário percorrer toda a Pilha!!

Utilização de typedef

- É usual utilizar `typedef` na manipulação de estruturas auto-referenciadas

```
struct node{  
    int value;  
    struct node *next;  
};  
  
typedef struct node  Node;  
typedef struct node* Node_ptr;
```


Utilização de typedef

- É usual utilizar `typedef` na manipulação de estruturas auto-referenciadas

```
struct node{  
    int value;  
    struct node *next;  
};  
  
typedef struct node  Node;  
typedef struct node* link;
```

Sinopse

- Estruturas Auto-Referenciadas: das tabelas às listas
 - Exemplo de aplicação:
 - Listas de inteiros
 - Pilhas de inteiros
 - **Listas de strings**
-
- Regras de Scope (“âmbito” de variáveis em C)
 - Abstract Data Types (ADTs)

Lista de Strings

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

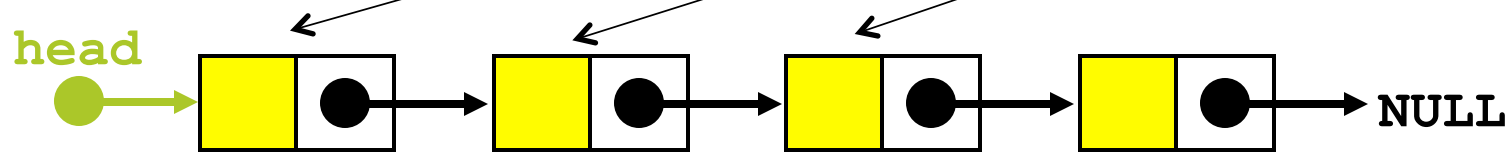
typedef struct node {
    char *text;
    struct node *next;
} *link;
```

Criar lista com Argumentos da linha de comandos

```
$ ./myprogram bolo1 bolo2 bolo3 bolo4 bolo5
```

```
int main(int argc, char* argv[]){  
    ...  
}
```

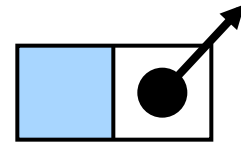
argc = 6
argv[0] = "myprogram"
argv[1] = "bolo1"
argv[2] = "bolo2"
argv[3] = "bolo3"
...



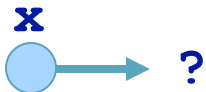
Criar lista com Argumentos da linha de comandos

```
int main(int argc, char* argv[])
{
    int i;
    link head = NULL;
    /* inserir todos os elementos na lista*/
    for(i = 1; i < argc; i++)
        head = insertEnd(head, argv[i]);
    print(head); /* imprime toda a lista*/
    /* remover o i-gesimo elemento (lido do stdin) */
    scanf("%d", &i);
    head = delete(head, argv[i]);
    print(head); /* voltamos a imprimir toda a lista */
    return 0;
}
```

Novo Elemento

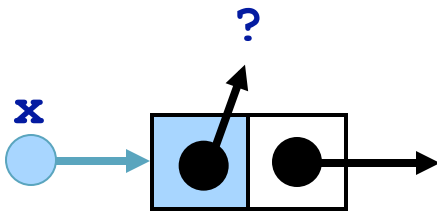


```
link NEW(char* text)
{
    link x = [redacted ?];
    x->text =
        [redacted ?];
    strcpy(x->text, text);
    x->next = NULL;
    return x;
}
```



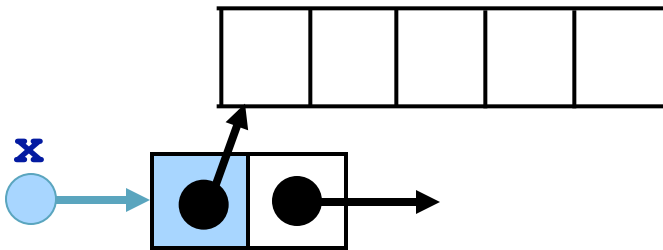
Novo Elemento

```
link NEW(char* text)
{
    link x = (link) malloc(sizeof(struct node));
    x->text =
        ?;
    strcpy(x->text, text);
    x->next = NULL;
    return x;
}
```



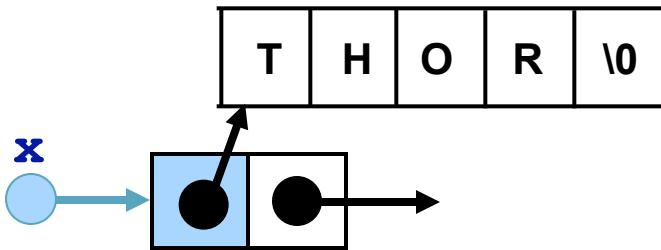
Novo Elemento

```
link NEW(char* text)
{
    link x = (link) malloc(sizeof(struct node));
    x->text =
        (char*) malloc(sizeof(char) * (strlen(text)+1));
    strcpy(x->text, text);
    x->next = NULL;
    return x;
}
```



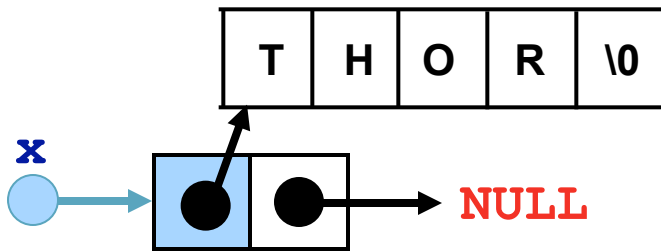
Novo Elemento

```
link NEW(char* text)
{
    link x = (link) malloc(sizeof(struct node));
    x->text =
        (char*) malloc(sizeof(char) * (strlen(text)+1));
    strcpy(x->text, text);
    x->next = NULL;
    return x;
}
```



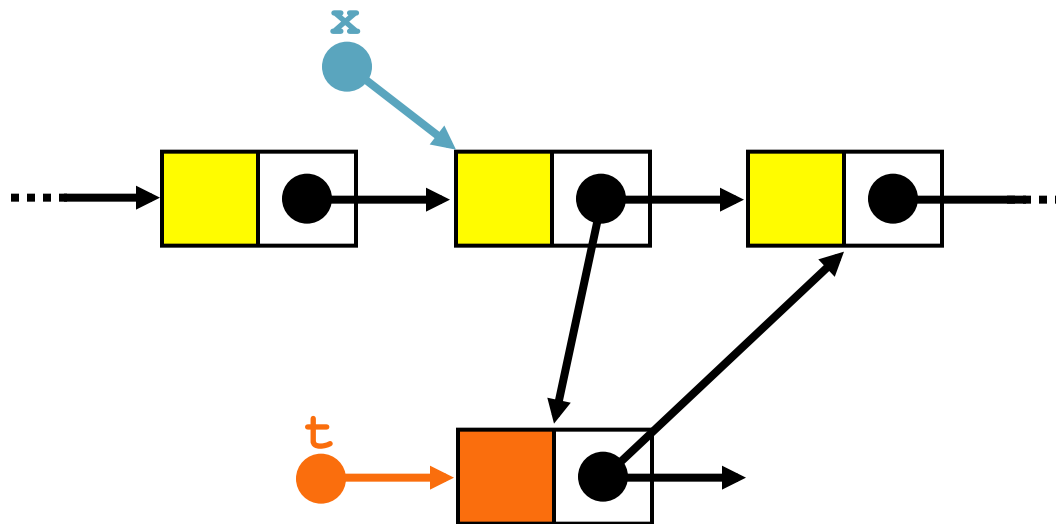
Novo Elemento

```
link NEW(char* text)
{
    link x = (link) malloc(sizeof(struct node));
    x->text =
        (char*) malloc(sizeof(char) * (strlen(text)+1));
    strcpy(x->text, text);
    x->next = NULL;
    return x;
}
```



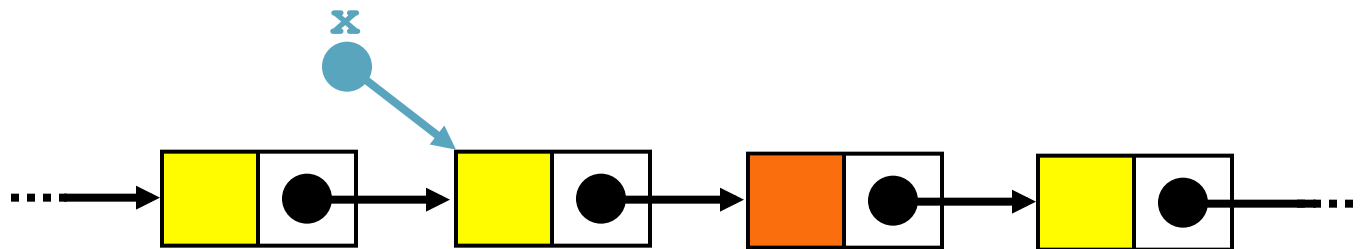
Inserção na Lista

```
t->next = x->next;  
x->next = t;
```



Inserção na Lista

```
t->next = x->next;  
x->next = t;
```



Inserção no Início (mais fácil!)

```
link insertBegin(link head, char* text)
{
    link x = NEW(text);
    x->next = ?;
    return x;
}
```

Inserção no Início

```
link insertBegin(link head, char* text)
{
    link x = NEW(text);
    x->next = head;
    return x;
}
```

*Tenho sempre de
retornar a head!*


```
int main() {
    (...)
    head = insertBegin(head, "Bolo");
    (...)
}
```

Inserção no Fim

```
link insertEnd(link head, char* text)
{
    link x;
    if(head == NULL)
        return NEW(text);
    for( [redacted] )
        ;
    x->next = NEW(text);
    return head;
}
```

Se a lista estiver vazia, o novo elemento passa a ser a head

Inserção no Fim

```
link insertEnd(link head, char* text)
{
    link x;
    if(head == NULL)
        return NEW(text);
    for(  )
        ;
    x->next = NEW(text);
    return head;
}
```

*Caso contrário
tenho de
encontrar o fim da
lista. Como?*

Inserção no Fim

```
link insertEnd(link head, char* text)
{
    link x;
    if(head == NULL)
        return NEW(text);
    for(x = head; x->next != NULL; x = x->next)
        ;
    x->next = NEW(text);
    return head;
}
```

Imprimir a Lista

```
void print(link head)
{
    link t;
    for(t = head; t != NULL; t = t->next)
        printf("%s\n", t->text);
}
```

Procura na Lista

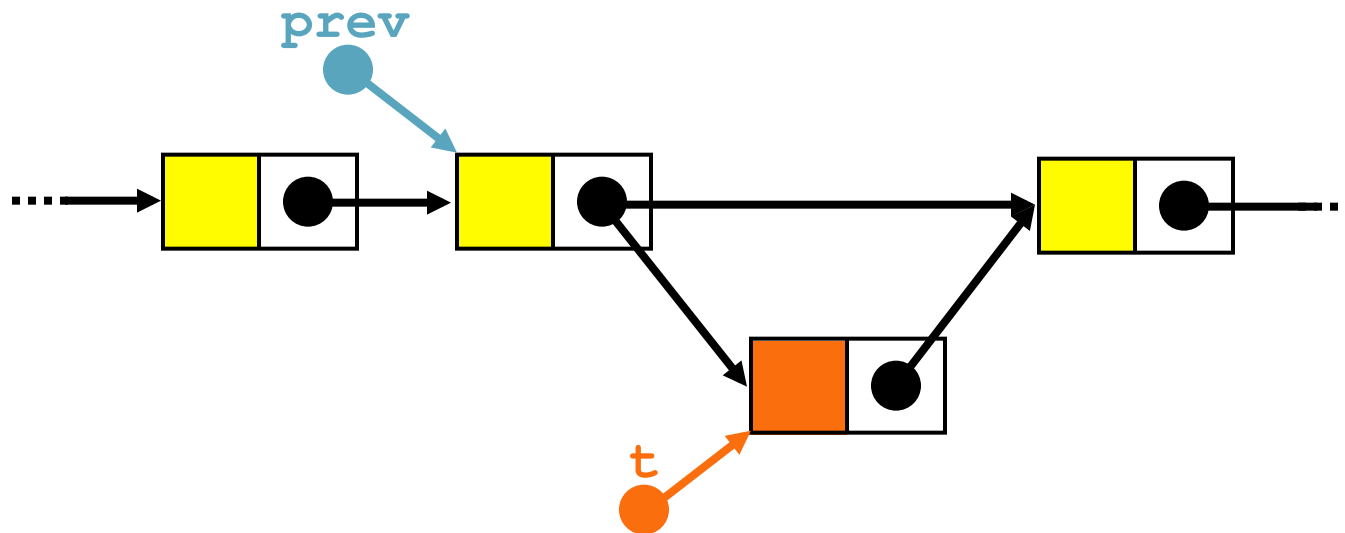
```
link lookup(link head, char* text)
{
    link t;
    for(t = head; t != NULL; t = t->next)
        if( ? )
            return t;
    return NULL;
}
```

Procura na Lista

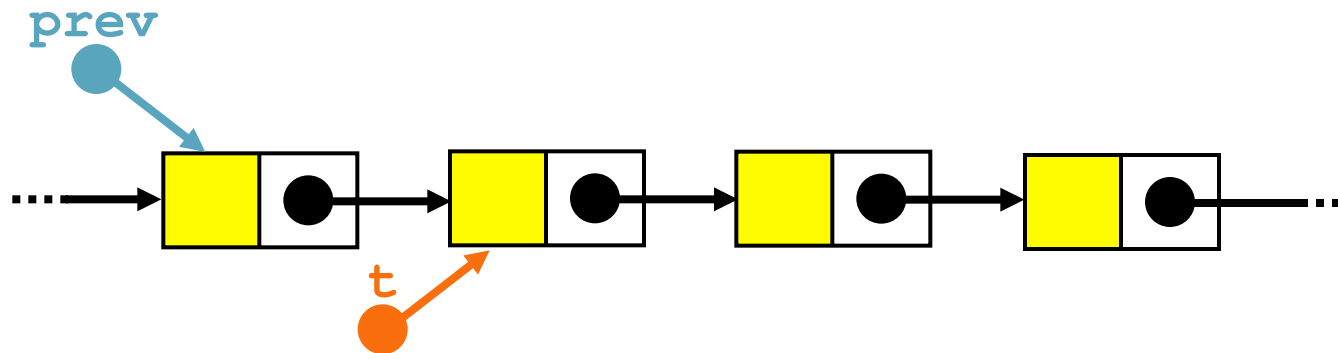
```
link lookup(link head, char* text)
{
    link t;
    for(t = head; t != NULL; t = t->next)
        if(strcmp(t->text, text) == 0)
            return t;
    return NULL;
}
```

Remoção da Lista


```
t = prev->next;  
prev->next = t->next;
```



Remoção da Lista com Procura



Remoção da Lista com Procura

```
link delete(link head, char* text)
{
    link t, prev;
    for(t = head, prev = NULL; t != NULL;
        prev = t, t = t->next) {
        if(strcmp(t->text, text) == 0) {
            if(t == head)
                ;
            else
                prev->next = t->next;
            free(t);
        }
    }
    return head;
}
```

Remoção da Lista com Procura

```
link delete(link head, char* text)
{
    link t, prev;
    for(t = head, prev = NULL; t != NULL;
        prev = t, t = t->next) {
        if(strcmp(t->text, text) == 0) {
            if(t == head)
                head = t->next;
            else
                prev->next = t->next;
            free(t);
        }
    }
    return head;
}
```

*Será que nos
falta alguma
coisa?!*

Remoção da Lista com Procura

```
link delete(link head, char* text)
{
    link t, prev;
    for(t = head, prev = NULL; t != NULL;
        prev = t, t = t->next) {
        if(strcmp(t->text, text) == 0) {
            if(t == head)
                head = t->next;
            else
                prev->next = t->next;
            free(t->text);
            free(t);
        }
    }
    return head;
}
```

Inserção no Início – Sem Valor de Retorno

```
link insertBegin(link head, char* text)
{
    link x = NEW(text);
    x->next = head;
    return x;
}
```

```
void insertBegin(link *headptr, char *text) /*alternativa*/
{
    link x = NEW(text);
    x->next = *headptr;
    *headptr = x;
}
```