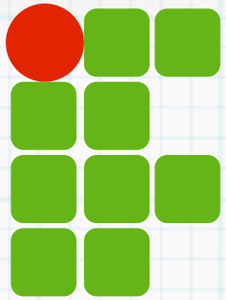
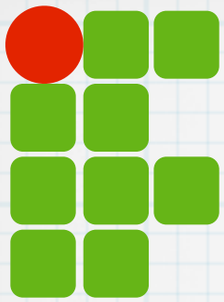


INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
RIO GRANDE DO NORTE

Algoritmos

Lista ligada

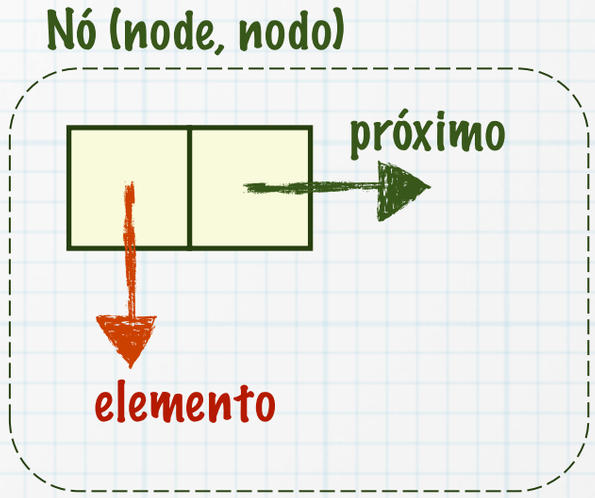


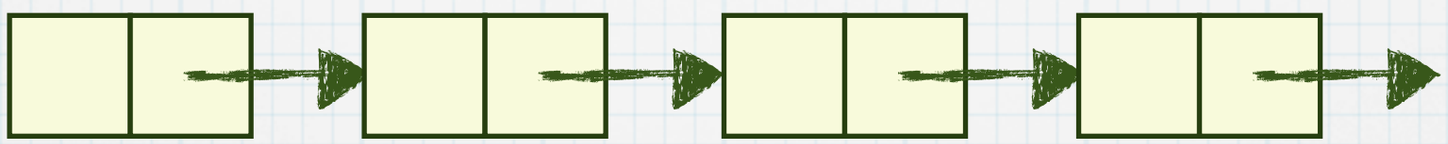
Agenda

* Introdução
* Delimitação
* Principais operações
* Exercícios

Lista ligada

* Lista de nós
* Cada nó contém
  * Um ou mais elementos
  * Um ponteiro para o próximo nó



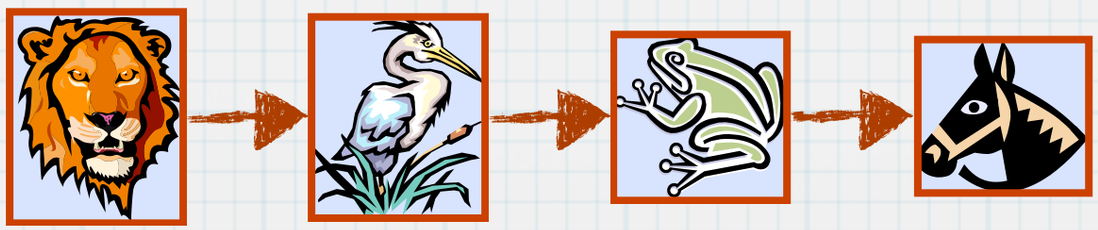


Lista ligada

* Um membro de uma struct é um ponteiro para a struct

```
struct ListaLigada {
    /* Dados do nó */
    struct ListaLigada *proximo;
};
```

  * Necessário alocar cada elemento dinamicamente
* Conjunto de nós
  * Cada nó tem um ponteiro para o próximo nó
* Necessário conhecer o primeiro nó



Lista ligada

* Lista de números inteiros
  * A struct contém um campo do tipo int e um ponteiro para o próximo

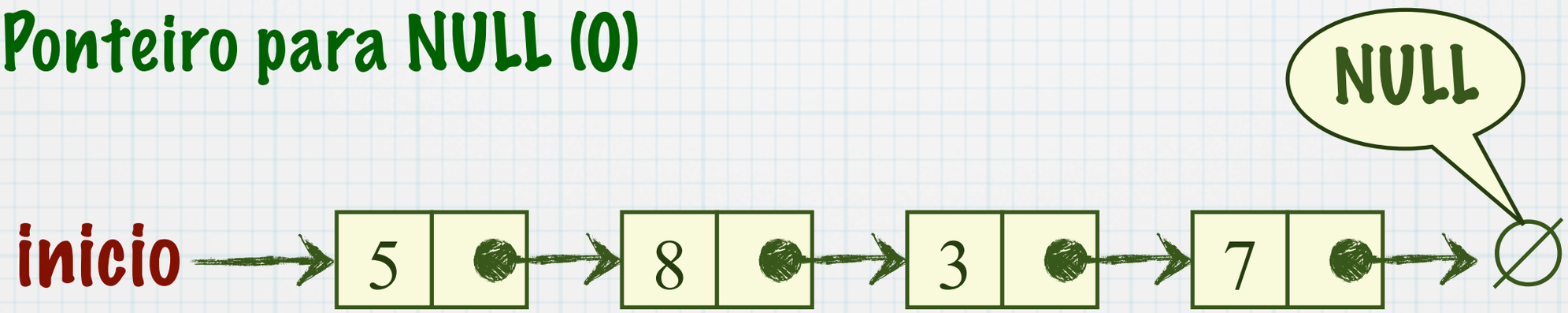
```c
struct ListaNumero {
  int numero;
  struct ListaNumero *proximo;
};
```

Delimitação da lista

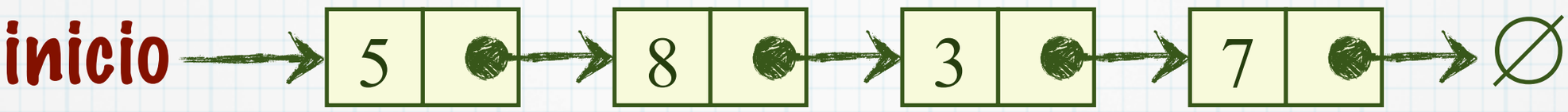
* Qual o início?
  * Ponteiro para o primeiro elemento da lista

```
struct ListaNumero *inicio;
```

* Qual o fim?
  * Ponteiro para NULL (0)

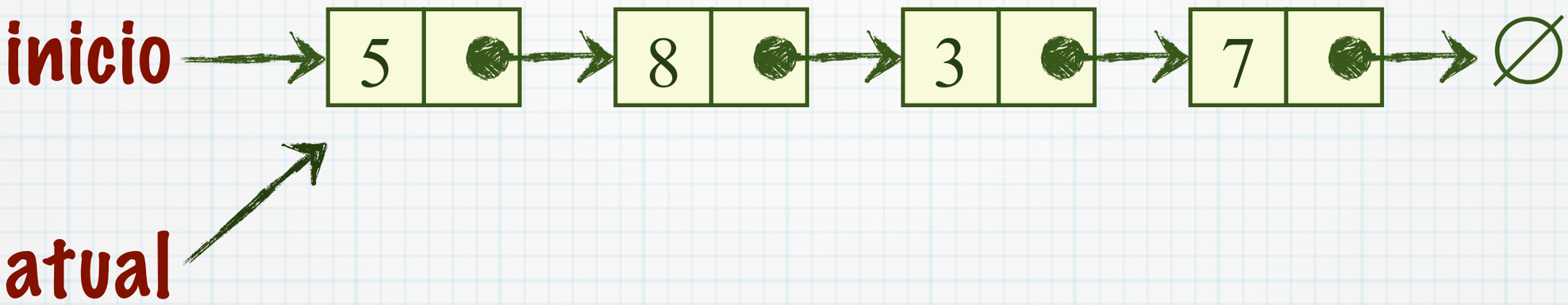


Percorrer lista



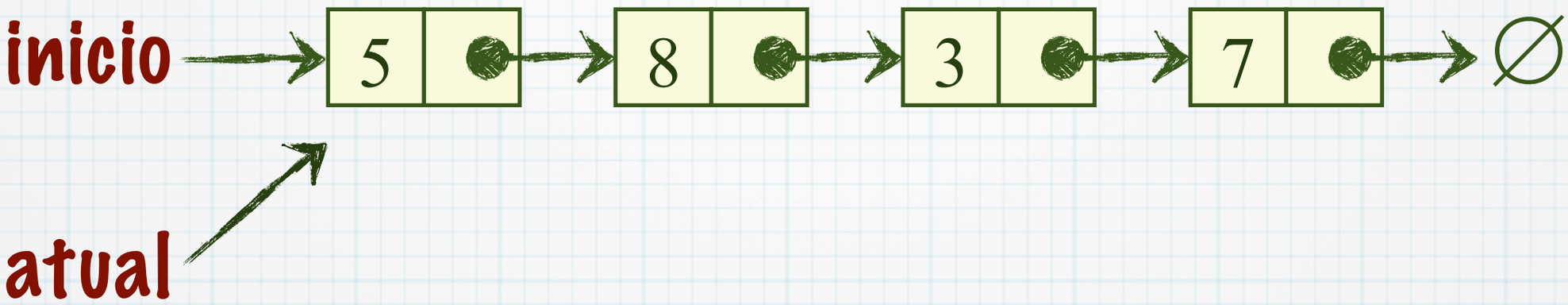

```
...
struct ListaNumero * atual;
atual = inicio;
printf("Lista de todos os números cadastrados\n\n");
while (atual != NULL) {
  printf("   %d\n", atual->numero);
  atual = atual->proximo;
}
...
```

Percorrer lista



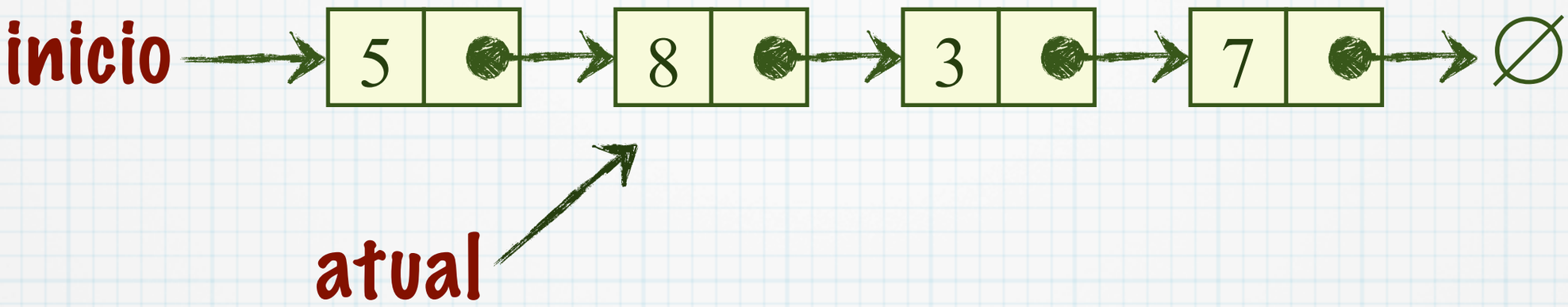

```
...
struct ListaNumero * atual;
atual = inicio;
printf("Lista de todos os números cadastrados\n\n");
while (atual != NULL) {
  printf("   %d\n", atual->numero);
  atual = atual->proximo;
}
...
```

Percorrer lista



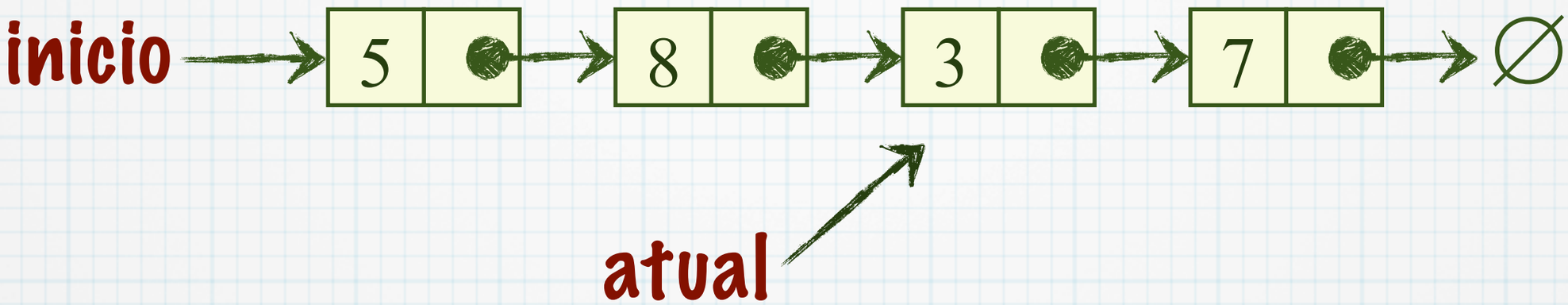

```
...
struct ListaNumero * atual;
atual = inicio;
printf("Lista de todos os números cadastrados\n\n");
while (atual != NULL) {
   printf("   %d\n", atual->numero);
   atual = atual->proximo;
}
...
```

Percorrer lista



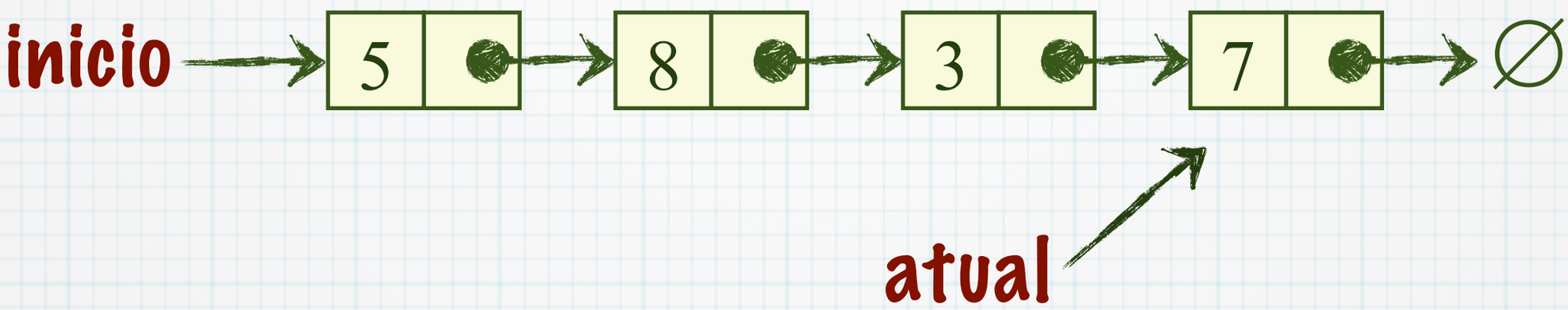

```
...
struct ListaNumero * atual;
atual = inicio;
printf("Lista de todos os números cadastrados\n\n");
while (atual != NULL) {
   printf("   %d\n", atual->numero);
   atual = atual->proximo;
}
...
```

Percorrer lista



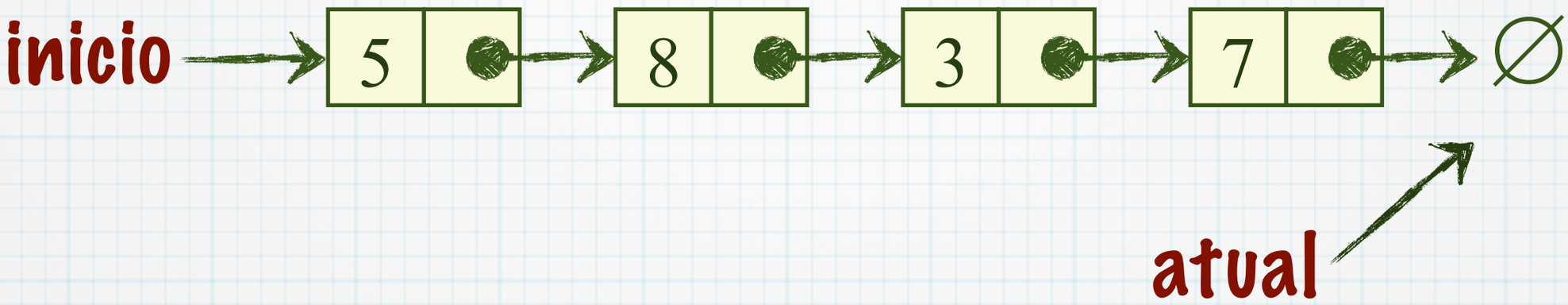

```
...
struct ListaNumero * atual;
atual = inicio;
printf("Lista de todos os números cadastrados\n\n");
while (atual != NULL) {
   printf("   %d\n", atual->numero);
   atual = atual->proximo;
}
...
```

Percorrer lista



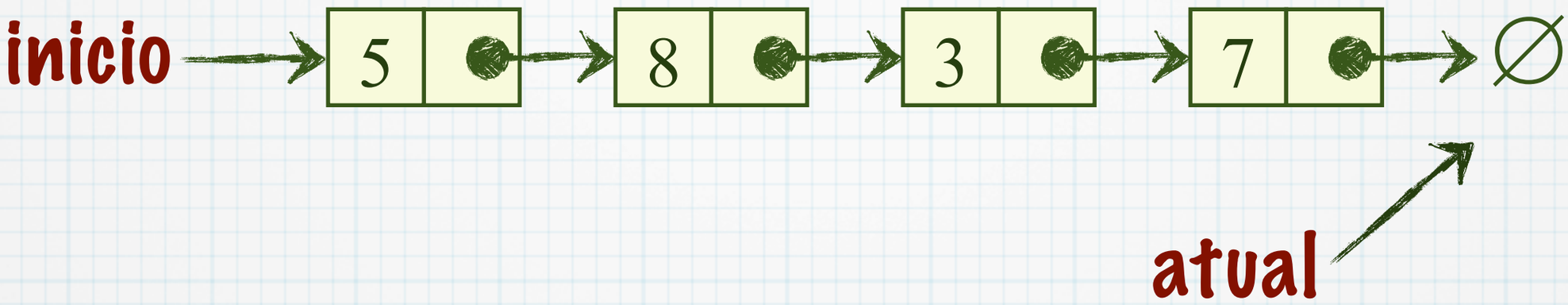

```
...
struct ListaNumero * atual;
atual = inicio;
printf("Lista de todos os números cadastrados\n\n");
while (atual != NULL) {
   printf("   %d\n", atual->numero);
   atual = atual->proximo;
}
...
```

Percorrer lista

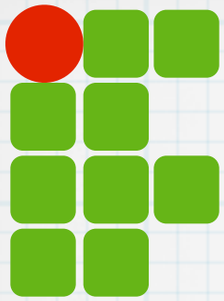



```
...
struct ListaNumero * atual;
atual = inicio;
printf("Lista de todos os números cadastrados\n\n");
while (atual != NULL) {
   printf("   %d\n", atual->numero);
   atual = atual->proximo;
}
...
```

Percorrer lista




```
...
struct ListaNumero * atual;
atual = inicio;
printf("Lista de todos os números cadastrados\n\n");
while (atual != NULL) {
  printf("   %d\n", atual->numero);
  atual = atual->proximo;
}
...
```



Operações em uma lista

* Inserir nó
  * No início, no fim, no meio
* Buscar nó
* Remover nó
  * Do início, do fim, do meio

Inserção no início

1. Aloca novo nó

```
novo = (struct ListaNumero*) malloc
(sizeof (struct ListaNumero));
```

2. Inserir novo elemento

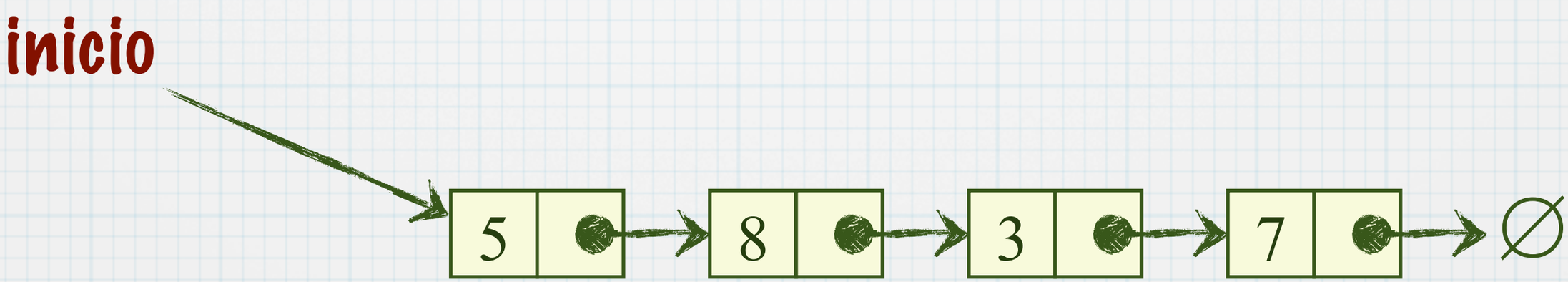
```
novo->numero = numero;
```

3. Novo nó aponta para antigo início

```
novo->proximo = inicio;
```

4. Início aponta para novo nó

```
inicio = novo;
```



Inserção no início

1. Aloca novo nó

```
novo = (struct ListaNumero*) malloc
(sizeof (struct ListaNumero));
```

2. Inserir novo elemento

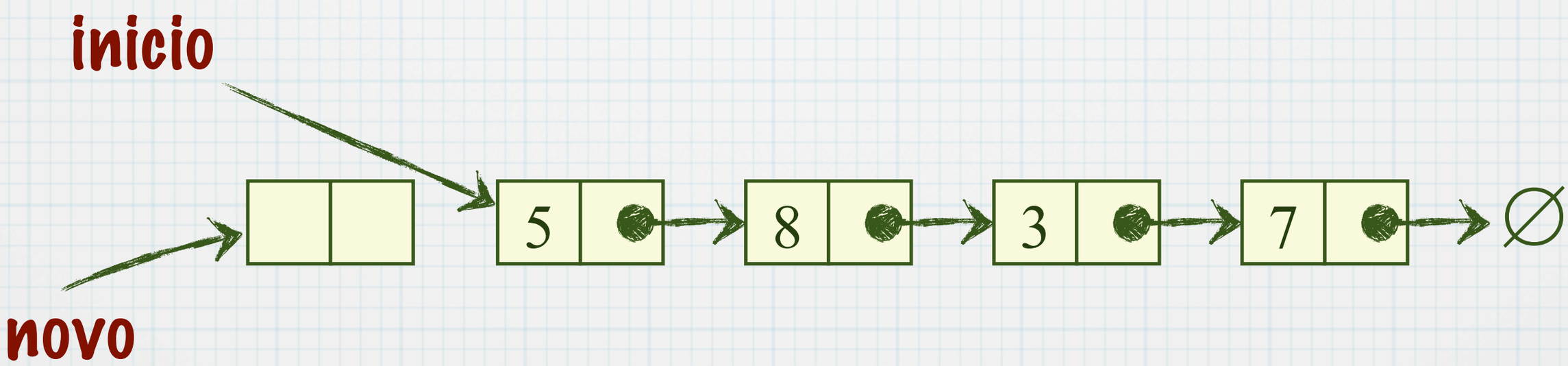
```
novo->numero = numero;
```

3. Novo nó aponta para antigo início

```
novo->proximo = inicio;
```

4. Início aponta para novo nó

```
inicio = novo;
```



Inserção no início

1. Aloca novo nó

```
novo = (struct ListaNumero*) malloc
(sizeof (struct ListaNumero));
```

2. Inserir novo elemento

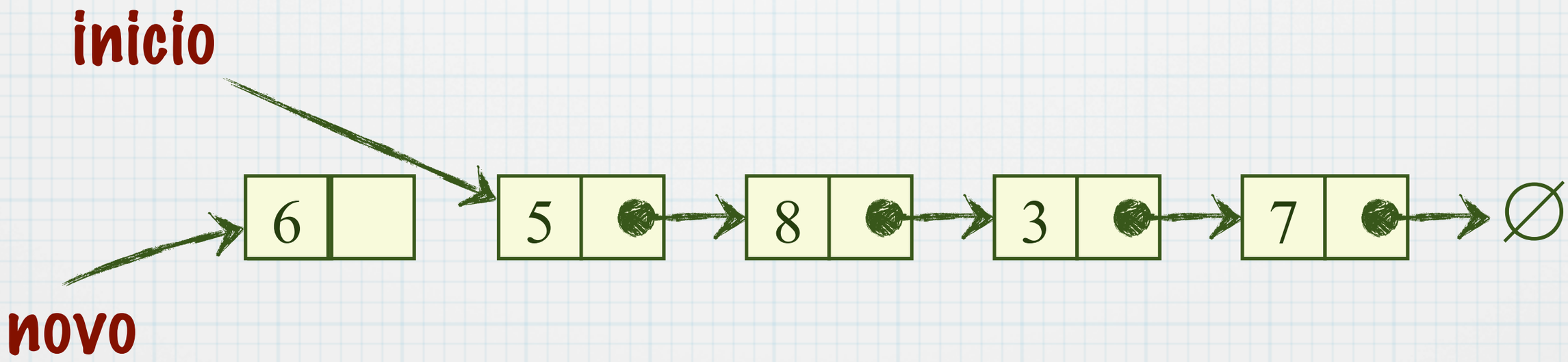
```
novo->numero = numero;
```

3. Novo nó aponta para antigo início

```
novo->proximo = inicio;
```

4. Início aponta para novo nó

```
*        inicio = novo;
```



Inserção no início

1. Aloca novo nó

```
novo = (struct ListaNumero*) malloc
(sizeof (struct ListaNumero));
```

2. Inserir novo elemento

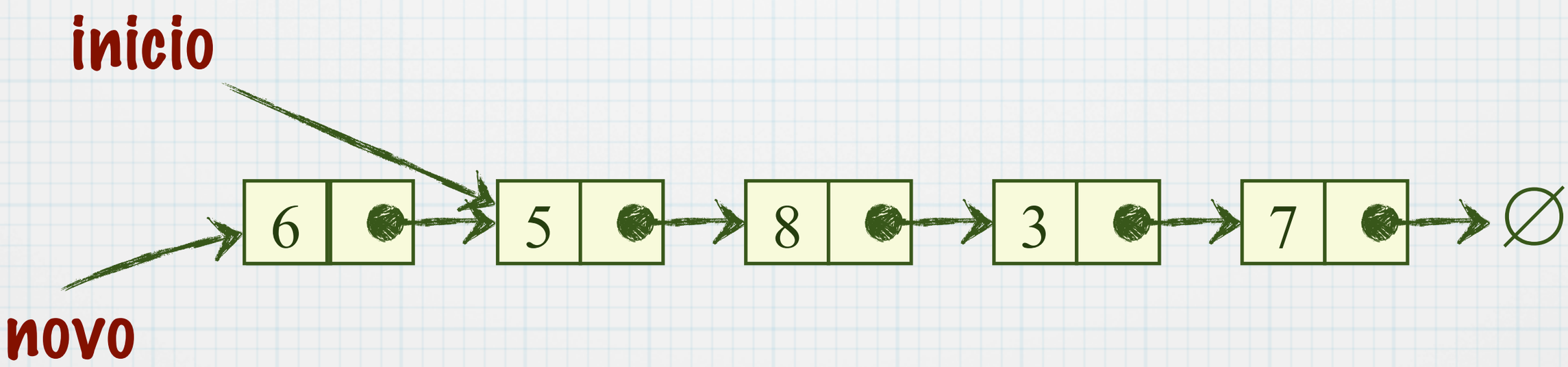
```
novo->numero = numero;
```

3. Novo nó aponta para antigo início

```
novo->proximo = inicio;
```

4. Início aponta para novo nó

```
inicio = novo;
```



Inserção no início

1. Aloca novo nó

```
novo = (struct ListaNumero*) malloc
(sizeof (struct ListaNumero));
```

2. Inserir novo elemento

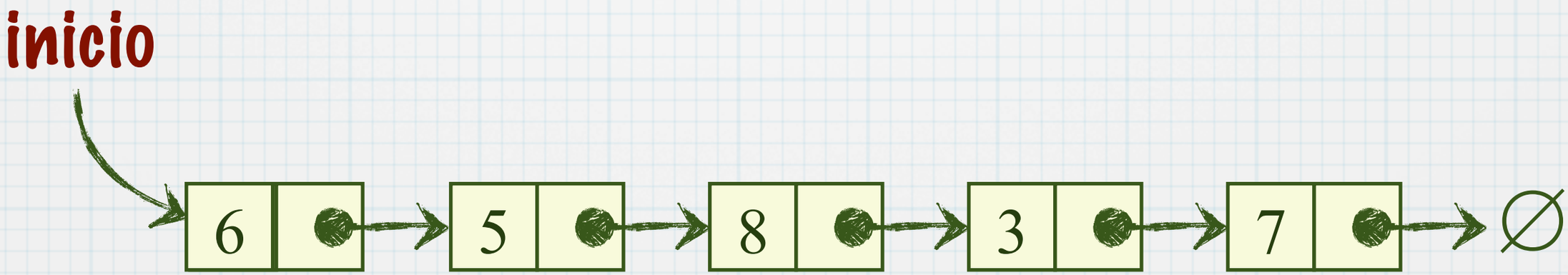
```
novo->numero = numero;
```

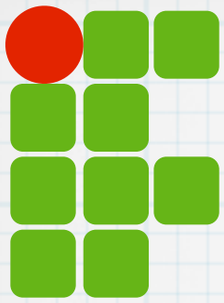
3. Novo nó aponta para antigo início

```
novo->proximo = inicio;
```

4. Início aponta para novo nó

```
inicio = novo;
```





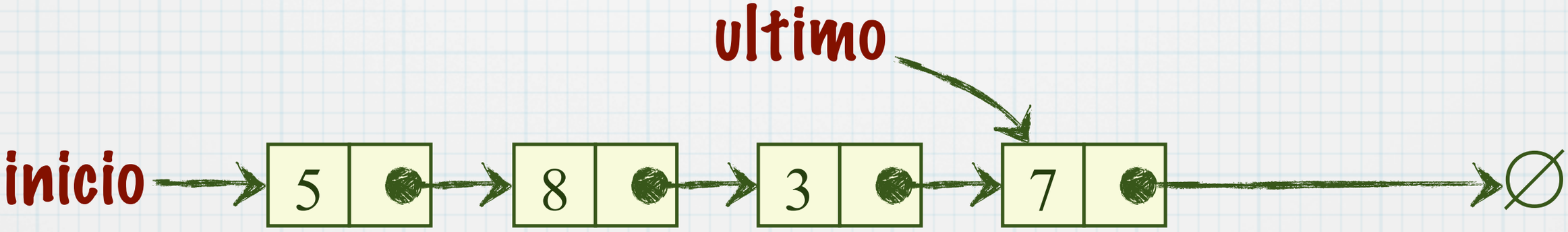
Inserção no início

```
int inserirInicio(int numero) {
   struct ListaNumero *novo;
   int retorno = 1;
   novo = (struct ListaNumero*)
           malloc(sizeof(struct ListaNumero));
   novo->numero = numero;
   novo->proximo = inicio;
   inicio = novo;
   return retorno;
}
```

Inserção no fim

* Necessário ter ponteiro para o último nó

```
ultimo = inicio;
while (ultimo->proximo != NULL)
  ultimo = ultimo->proximo;
```



Inserção no fim

1. Aloca novo nó

```
novo = (struct ListaNumero*) malloc
(sizeof (struct ListaNumero));
```

2. Inserir novo elemento

```
novo->numero = numero;
```

3. Novo nó aponta para nulo

```
novo->proximo = NULL;
```

4. Último aponta para novo

```
* ultimo->proximo =
novo;
```

Inserção no fim

1. Aloca novo nó

```
novo = (struct ListaNumero*) malloc
(sizeof (struct ListaNumero));
```

2. Inserir novo elemento

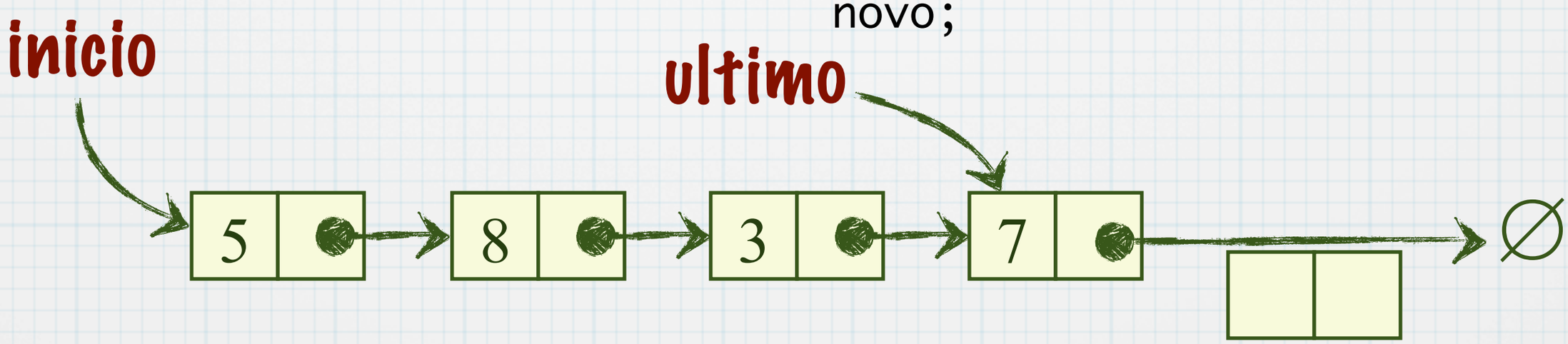
```
novo->numero = numero;
```

3. Novo nó aponta para nulo

```
novo->proximo = NULL;
```

4. Último aponta para novo

```
*      ultimo->proximo =
    novo;
```



Inserção no fim

1. Aloca novo nó

```
novo = (struct ListaNumero*) malloc
(sizeof (struct ListaNumero));
```

2. Inserir novo elemento

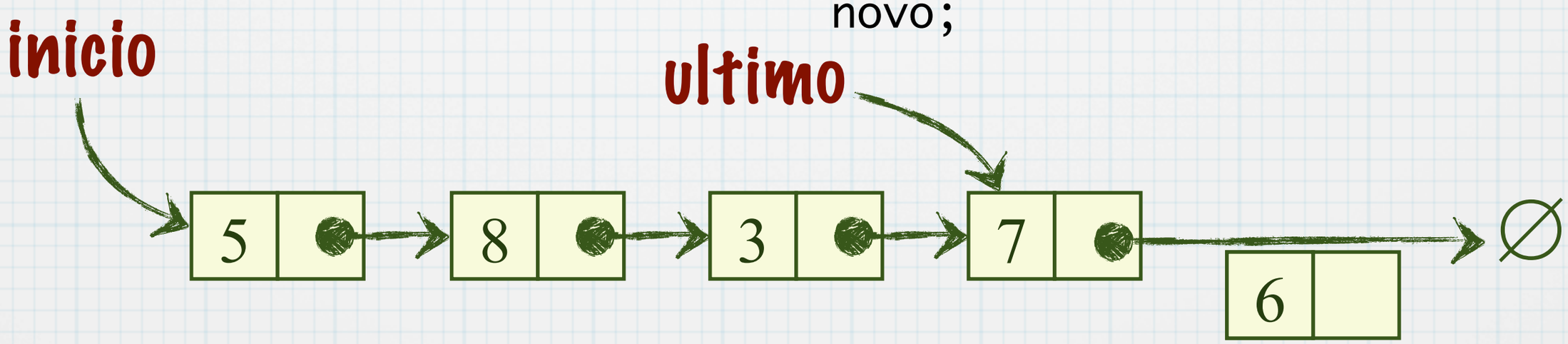
```
novo->numero = numero;
```

3. Novo nó aponta para nulo

```
novo->proximo = NULL;
```

4. Último aponta para novo

```
*      ultimo->proximo =
novo;
```



Inserção no fim

1. Aloca novo nó

```
novo = (struct ListaNumero*) malloc
(sizeof (struct ListaNumero));
```

2. Inserir novo elemento

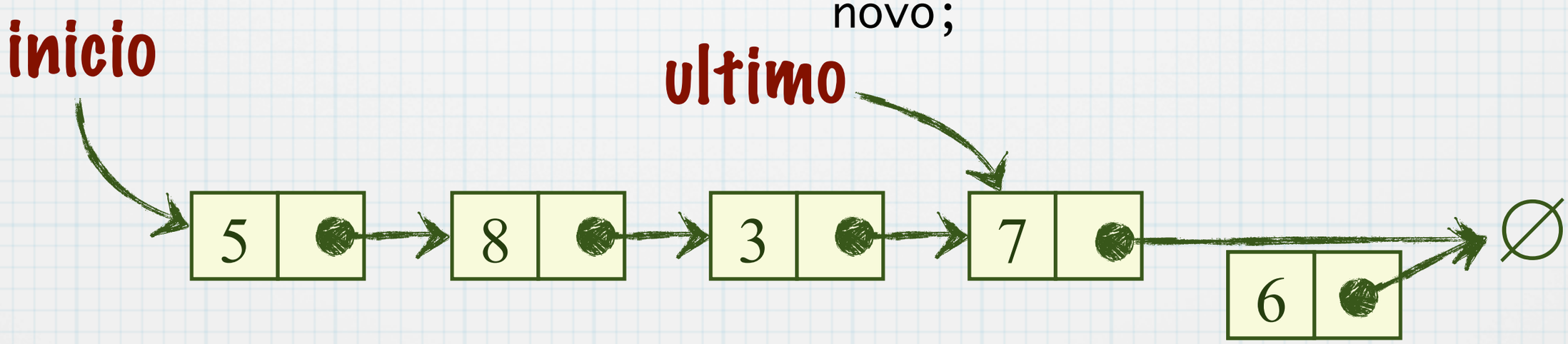
```
novo->numero = numero;
```

3. Novo nó aponta para nulo

```
novo->proximo = NULL;
```

4. Último aponta para novo

```
*      ultimo->proximo =
    novo;
```



Inserção no fim

1. Aloca novo nó

```
novo = (struct ListaNumero*) malloc
(sizeof (struct ListaNumero));
```

2. Inserir novo elemento

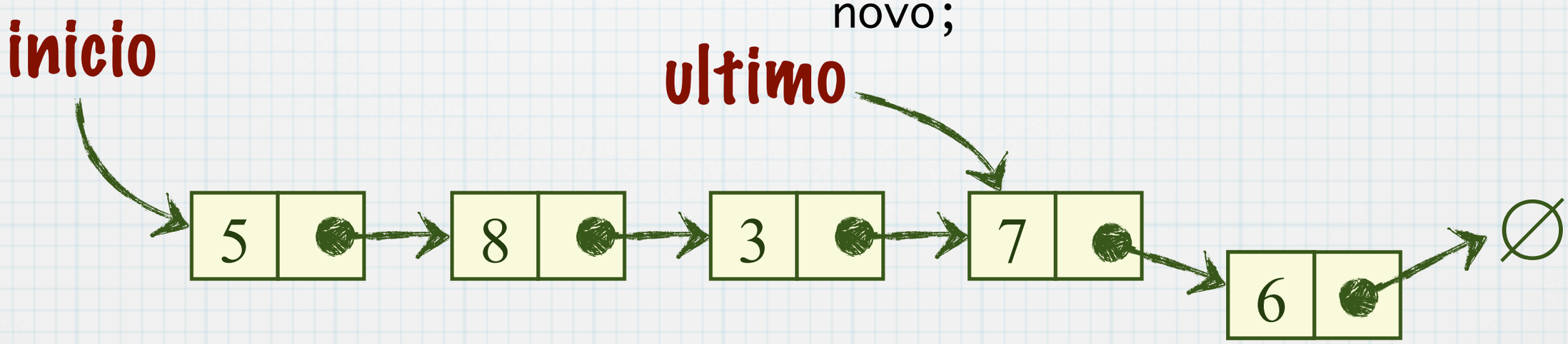
```
novo->numero = numero;
```

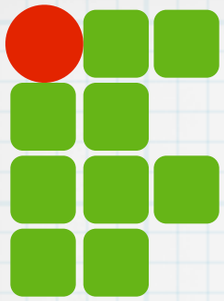
3. Novo nó aponta para nulo

```
novo->proximo = NULL;
```

4. Último aponta para novo

```
*      ultimo->proximo =
    novo;
```





Inserção no fim

```c
int inserirFim(int numero) {
    struct ListaNumero *novo, *ultimo;
    int retorno = 1;
    novo = (struct ListaNumero*) malloc(sizeof(struct ListaNumero));
    if (novo == NULL) { /* Verifica se a memória foi alocada */
        retorno = -1;
    } else {
        novo->numero = numero;
        novo->proximo = NULL; /* O novo é o fim da lista */
        if (inicio == NULL) { /* Se a lista estiver vazia */
            inicio = novo; /* o novo será o único elemento da lista */
        } else { /* Se ja houver elementos na lista */
            /* Percorre a lista para encontrar o último elemento */
            ultimo = inicio;
            while (ultimo->proximo != NULL)
                ultimo = ultimo->proximo;
            ultimo->proximo = novo; /* Atualiza referência do último */
        }
    }
    return retorno;
}
```

Remoção do início

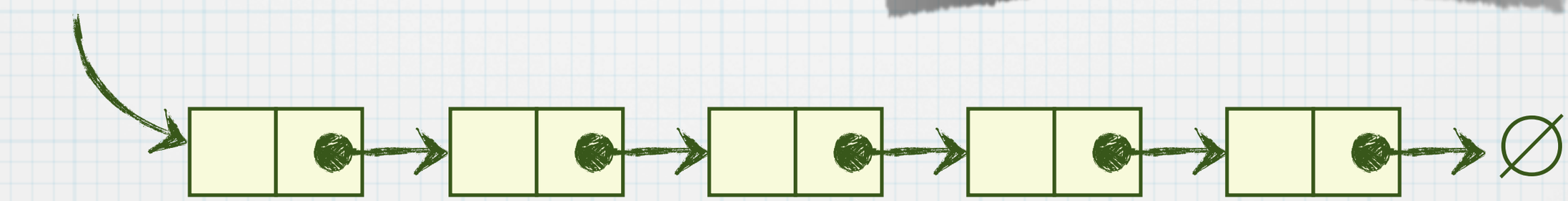
1. Atualiza início

```
    velho = inicio;
*       inicio = inicio ->proximo;
```

2. Libera espaço usado

```
    free(velho);
```

```
int removeInicio() {
    struct ListaNumero *velho;
    int retorno = 1;
    velho = inicio;
    if (velho == NULL) {
        retorno = -1;
    } else {
        inicio = inicio->proximo;
        free(velho);
    }
    return retorno;
}
```



Remoção do início

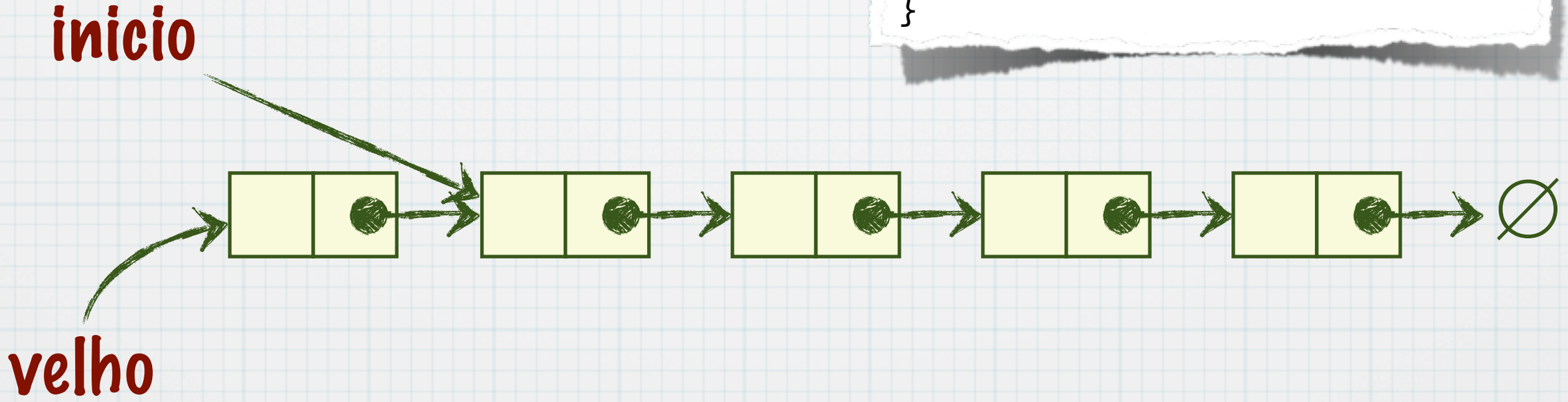
1. Atualiza início

```
    velho = inicio;
*       inicio = inicio ->proximo;
```

2. Libera espaço usado

```
    free(velho);
```

```
int removeInicio() {
    struct ListaNumero *velho;
    int retorno = 1;
    velho = inicio;
    if (velho == NULL) {
        retorno = -1;
    } else {
        inicio = inicio->proximo;
        free(velho);
    }
    return retorno;
}
```



Remoção do início

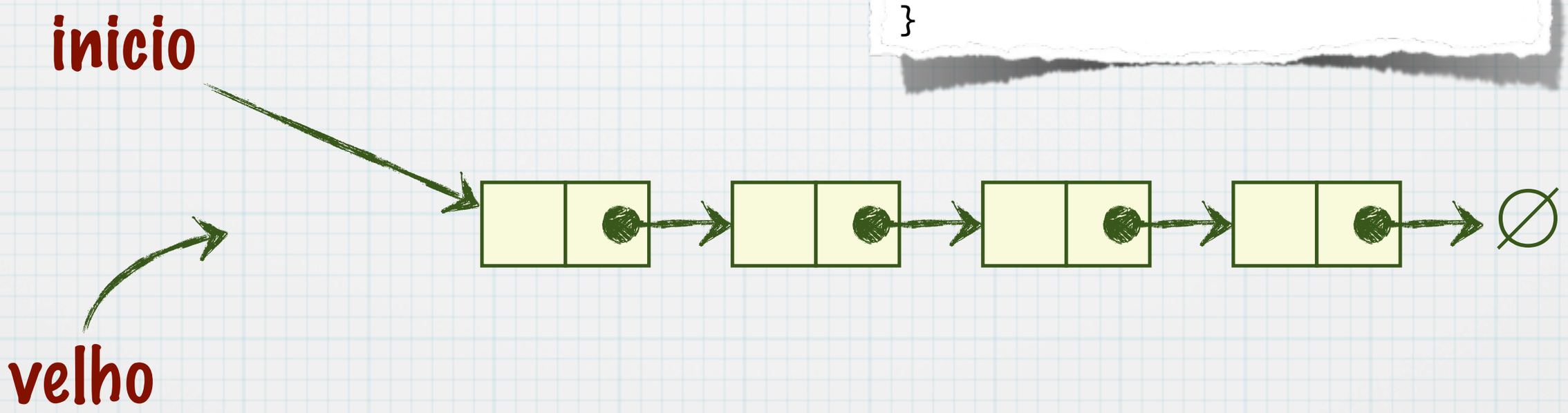
1. Atualiza início

```
velho = inicio;
inicio = inicio ->proximo;
```

2. Libera espaço usado

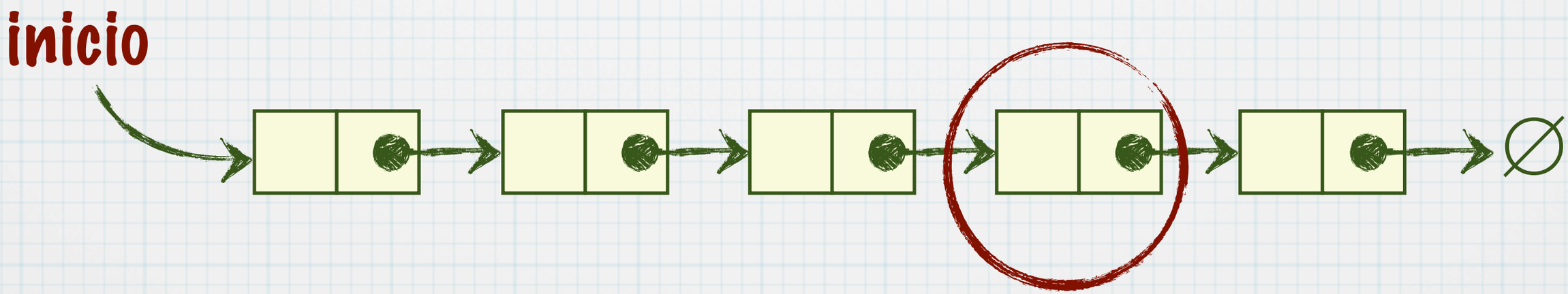
```
free(velho);
```

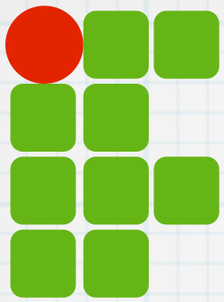
```
int removeInicio() {
    struct ListaNumero *velho;
    int retorno = 1;
    velho = inicio;
    if (velho == NULL) {
        retorno = -1;
    } else {
        inicio = inicio->proximo;
        free(velho);
    }
    return retorno;
}
```



Remoção do fim

* Precisamos encontrar o elemento anterior ao último
* Esse elemento apontará para nulo
* Exercício

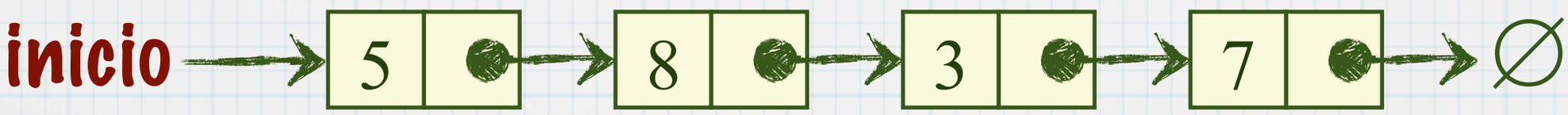




Buscar um número

* Devemos percorrer toda a lista e comparar cada elemento com o parâmetro passado

```
int buscaNumero(int numero) {
   struct ListaNumero * atual;
   int retorno = -1;
   atual = inicio;
   while (atual != NULL) {
      if (atual->numero == numero) {
         retorn = 1;
         break;
      }
      atual = atual->proximo;
   }
   return retorno;
}
```



Dúvidas?