Aula 28

MC 102 - Algoritmos e Programação de Computadores

Listas Ligadas I.

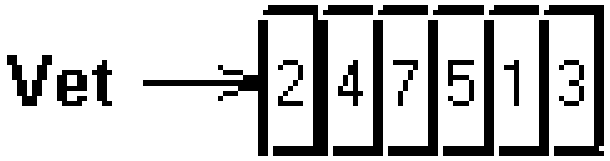
Listas Ligadas

A utilização de listas ligadas resolve vários problemas típicos que, se usando vetores, seria muito trabalhoso.

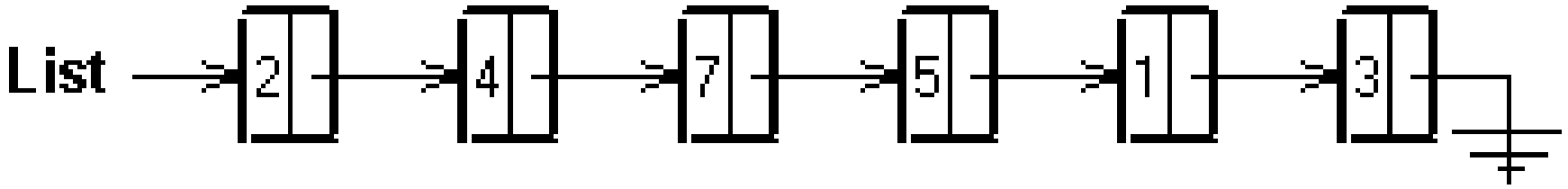
Algumas problemas exigem que certas prioridades dos elementos dos vetores sejam garantidas como, por exemplo, a ordem em que os elementos são inseridos define a ordem que serão removidos (como uma fila qualquer).

Listas Ligadas x Vetores

Vetor:



Lista Ligada:



Listas Ligadas x Vetores

Vantagens:

- Não há a necessidade de redimensionar pois considera cada elemento individualmente
  - Num vetor, o redimensionamento é obrigatório.
- Não é preciso deslocar valores para garantir alguma ordem dos elementos
  - Manter o vetor ordenado
  - Garantir a ordem de chegada / saída

Desvantagens:

- Não é possível fazer acesso imediato (usar indexador)
  - Será necessário percorrer alguns elementos

Lista Ligada

Uma Lista Ligada representa um conjunto de elementos denominados **nós** onde as informações são armazenadas.

Cada nó deve obrigatoriamente possuir um apontador para o próximo elemento da lista, caracterizando assim uma ligação entre os nós.

Portando, um nó é um registro que contém a **informação** a ser armazenada e o **apontador** para o próximo nó.

Lista Ligada - Implementação

```
/* Define o tipo de dado que será utilizado na lista */
typedef unsigned int TipoDado;

typedef struct no {
  TipoDado info;  /* Campo para a informação */
  struct no *prox; /* Campo p/ o apontador do próximo nó */
} no;

typedef no * pno; /* Tipo apontador para nó    */
typedef no * lista; /* Tipo Lista Ligada Simples */
```

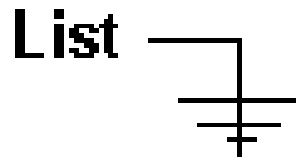
Lista Ligada - Propriedades

Propriedades de uma Lista Ligada Simples

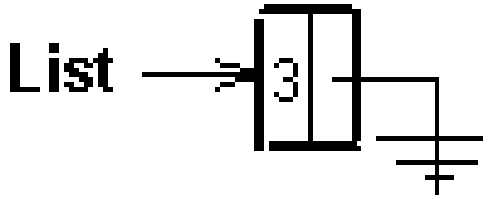
- Uma lista vazia aponta sempre para NULL
- O último nó da lista sempre aponta para NULL
- Qualquer nó da lista sempre aponta para o nó seguinte, permitindo um “passeio” do primeiro ao último
- Todo nó criado deve sempre apontar para NULL, até que sua posição na lista seja definida

Lista Ligada - Propriedades

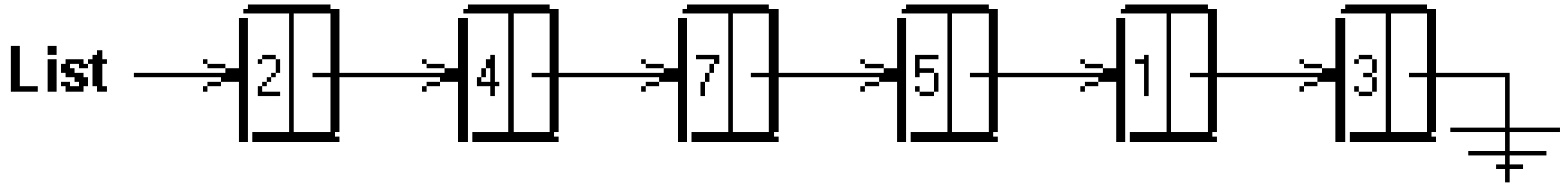
Lista Vazia:



Lista com um único nó:



Lista com vários nós:



Lista Ligada - Operações

- Inicializa uma lista vazia
- Criar um novo nó
- Inserir um nó na lista
- Procurar um nó na lista
- Imprimir a lista
- Remover um nó da lista
- Liberar a lista
  - Remover todos os elementos

Lista Ligada - InicializaLista

- Inicializa uma lista vazia

  A lista deve sempre ser inicializada como vazia

```
void InicializaLista(lista *L) {
   *L = NULL;
}
```

Lista Ligada - CriaNo

- Criar um novo nó

  Um novo nó é alocado na memória dinâmica

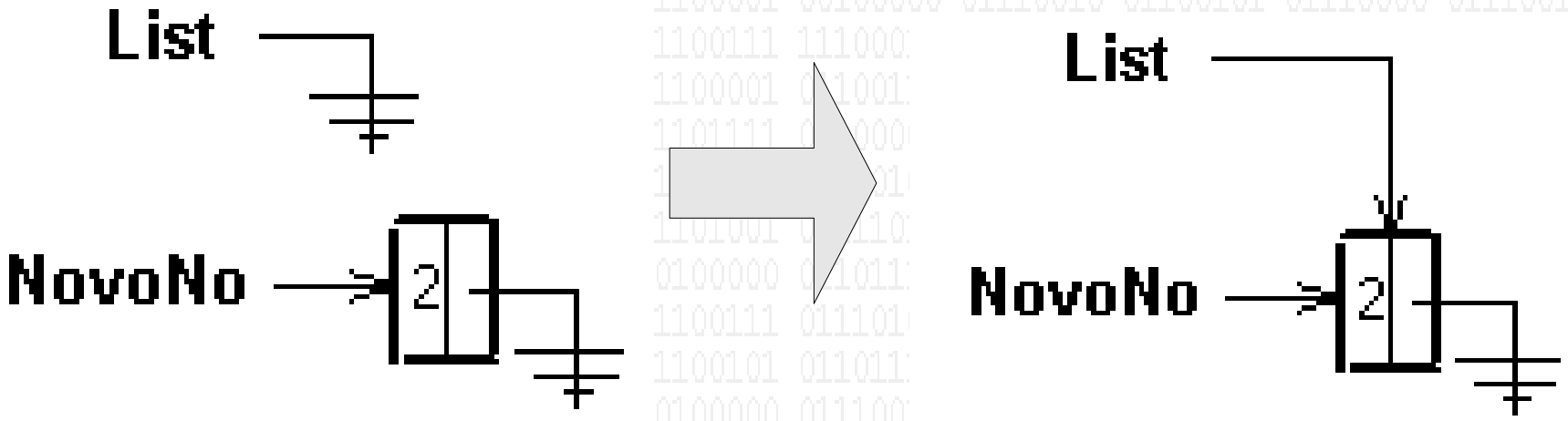
```
pno CriaNo(TipoDado Info) {
  pno No;

  No = (pno) malloc(sizeof(no));
  No->info = Info;
  No->prox = NULL;
  return No;
}
```

Lista Ligada - Inserção

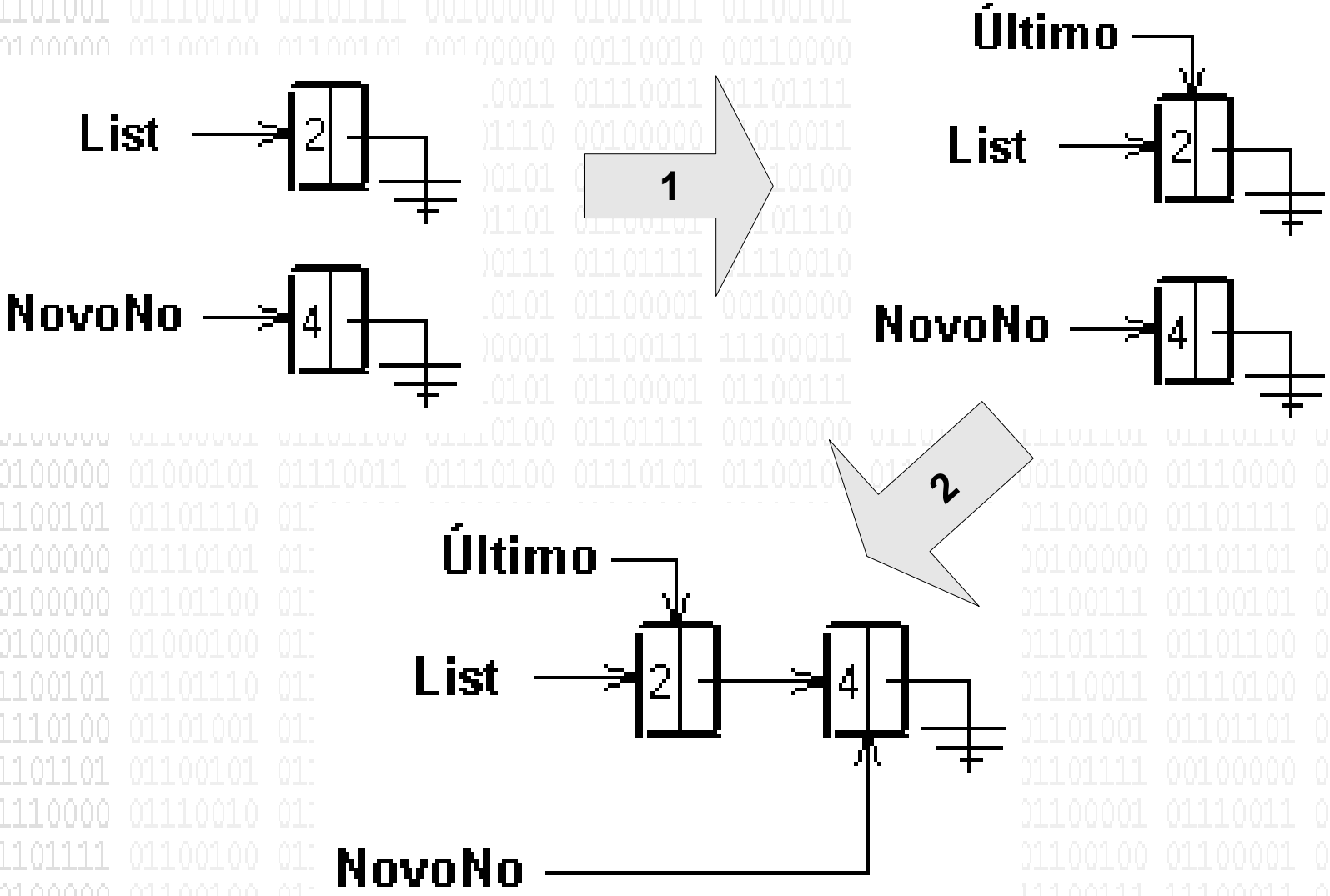
Supondo que novos elementos são inseridos no final da lista, o procedimento de inserção atuará da seguinte maneira:

- Em uma lista vazia, insere o elemento e este passa a ser o primeiro elemento da lista
- Em uma lista com pelo menos um elementos, procura o último elemento e insere o novo elemento após o último. O último agora passa a ser o novo elemento inserido na lista.

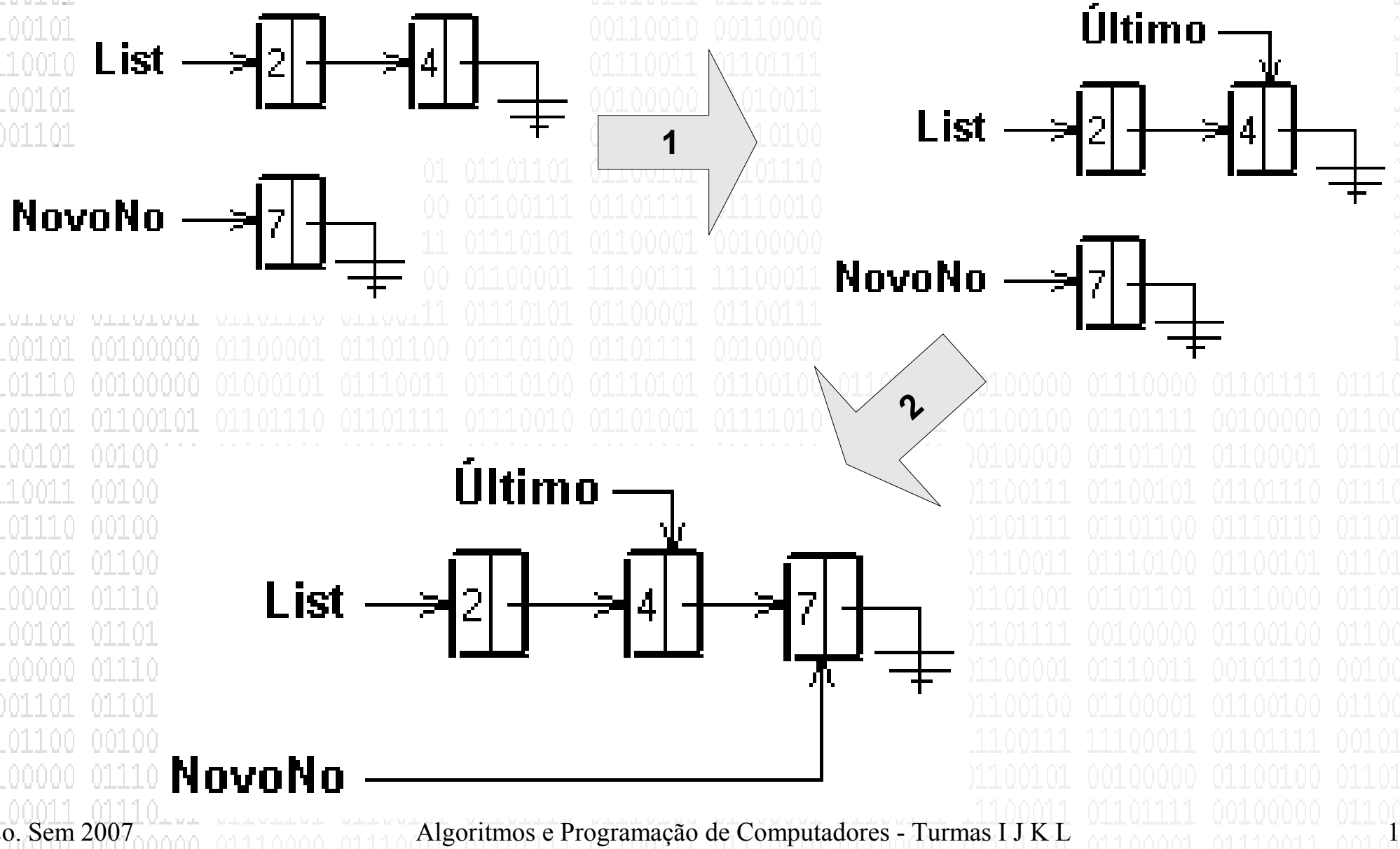
Inserção do elemento 2



Inserção do elemento 4



Inserção do elemento 7

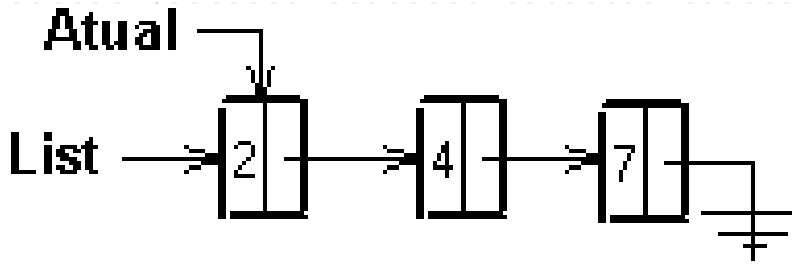


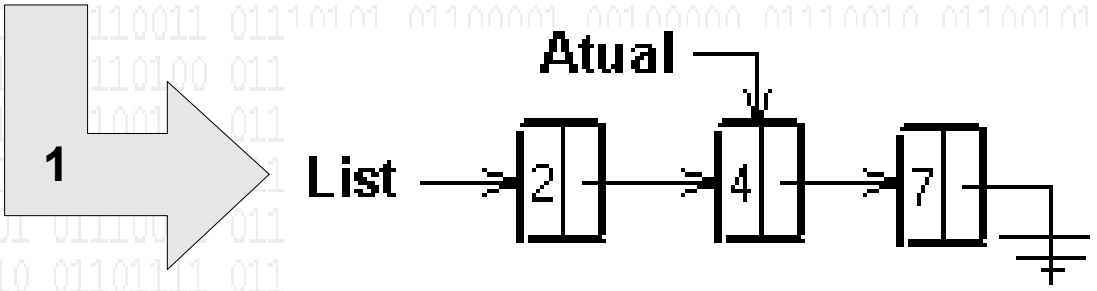
Lista Ligada - Busca

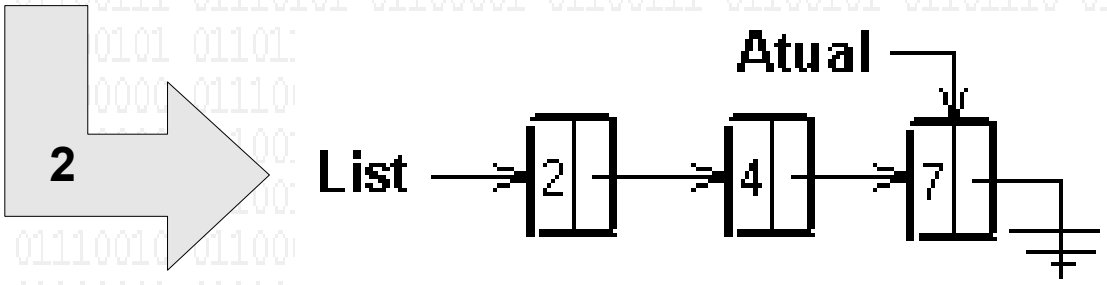
Para procurar uma informação, o resultado da busca deverá ser um apontador para o nó cuja informação foi encontrada, ou o apontador NULL caso contrário. Portanto, o algoritmo será:

- Se a lista é vazia, retorna NULL
- Senão, verifica se o nó atual é possui a informação desejada.
  - Se a informação foi encontrada, retorna o endereço do nó atual.
  - Senão, se for o último elemento, retorna NULL. Caso contrário, passa para nó seguinte e continua procurando.

Busca pelo elemento 7





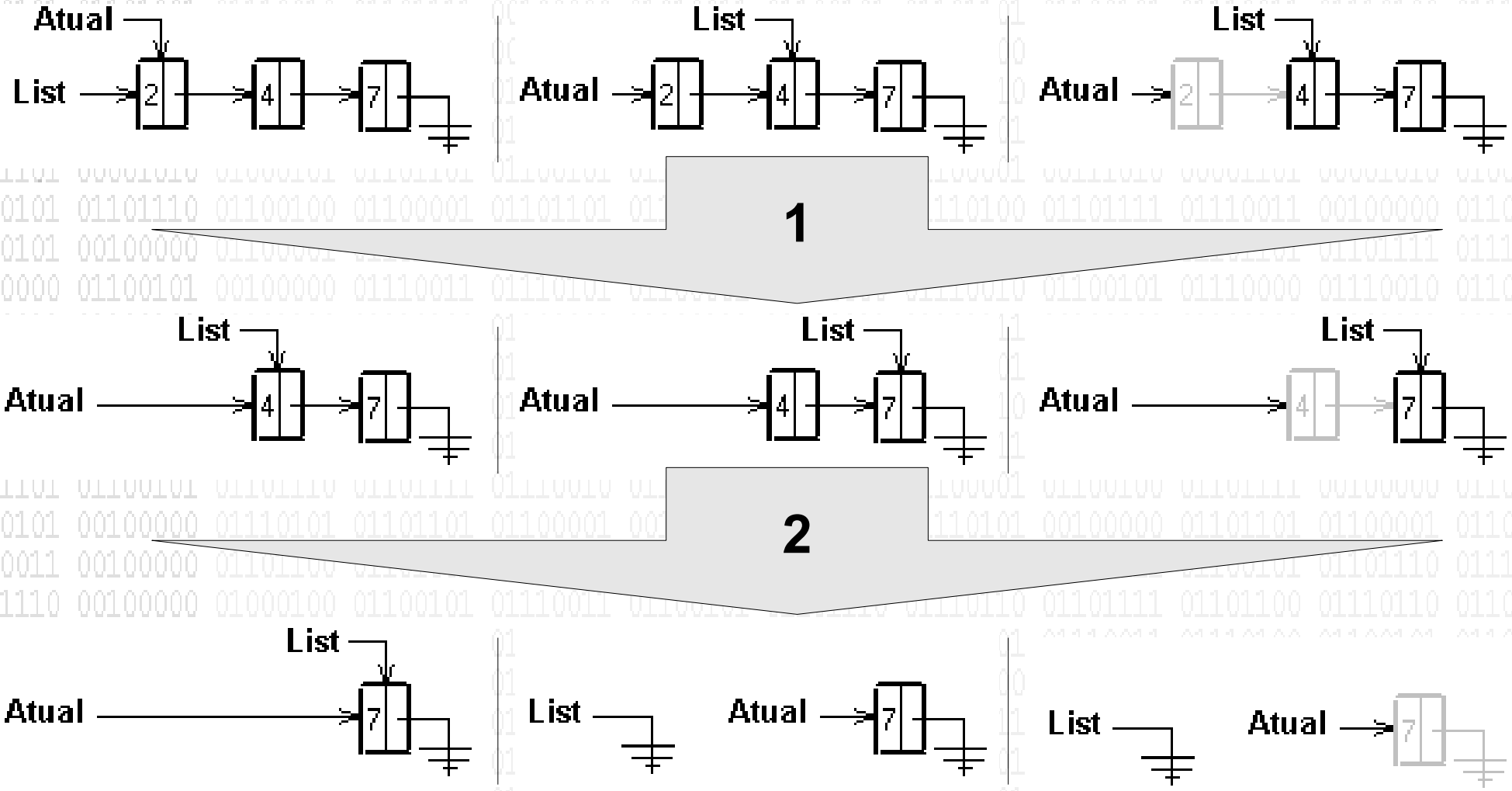


Lista Ligada - LiberaLista

A memória alocada para cada nó de uma lista deve ser liberada após a utlização da lista. O algorimto para liberar todos os nós de uma lista é semelhante ao de busca, da forma:

- Enquanto a lista não estiver vazia
  - Guarda a posição do nó atual
  - Atualiza o primeiro elemento da lista como sendo o nó seguinte ao atual
  - Remove o nó atual

LiberaLista



LiberaLista - Iterativo

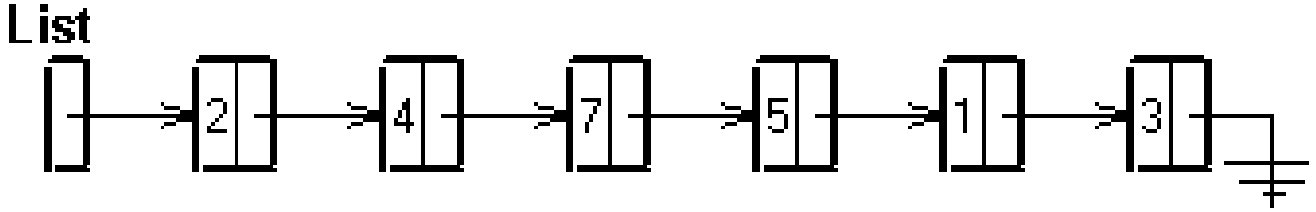
```
void LiberaLista(lista *L) {
  pno p = *L;

  while (p != NULL) {
    *L = p->prox;
    free(p);
    p = *L;
  }
}
```

LiberaLista - Recursivo

```
void LiberaLista(lista *L) {
  pno p = *L;
  if (p != NULL) {
   LiberaLista(&(p->prox));
    free(p);
    *L = NULL;
  }
}
```

Lista Ligada Simples com Nó Cabeça

Um nó cabeça é definido como um nó que contém apenas um apontador para o primeiro nó da lista. A informação do nó cabeça é irrelevante. Neste caso, a lista deixa de ser um apontador e passa a ser um nó (o próprio nó cabeça)



Lista Ligada Simples com Nó Cabeça

- O nó cabeça está sempre na mesma posição de memória, mesmo que a lista esteja vazia.
- Uma lista vazia é quando o campo prox do nó cabeça é igual a NULL
- A informação contida no nó cabeça é irrelevante

Observação!

- Listas Simples com ou sem nó cabeça se diferenciam muito pouco, representando mais uma opção do programador