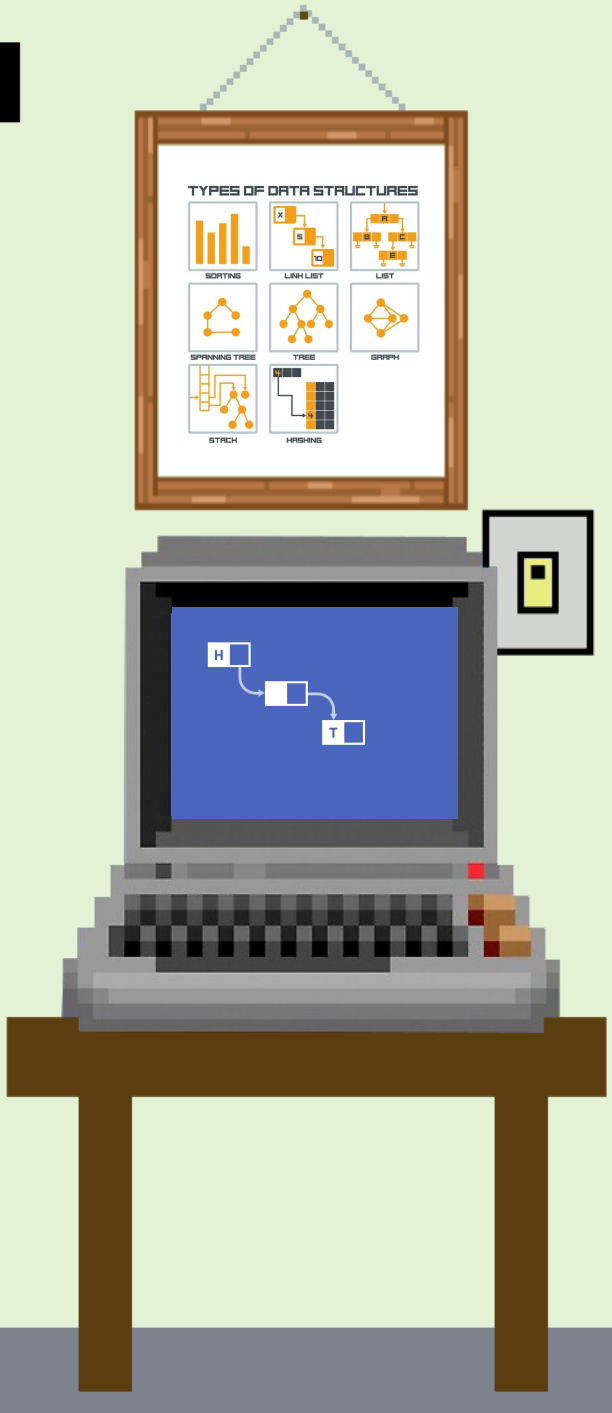
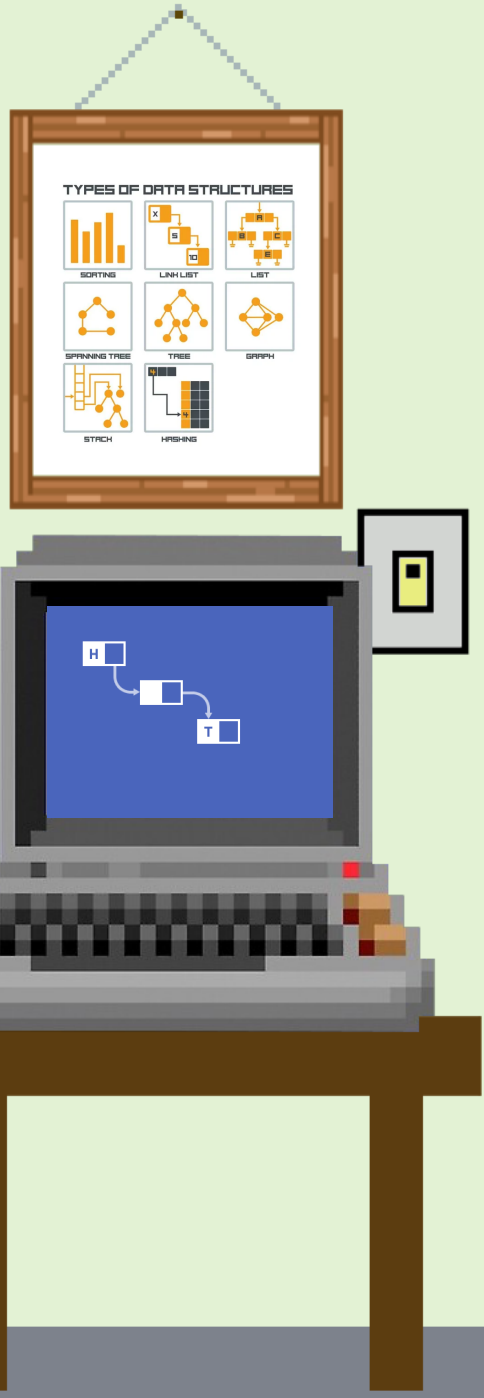
Listas Encadeadas Duplas

Estruturas de Dados I



Listas Encadeadas Duplas

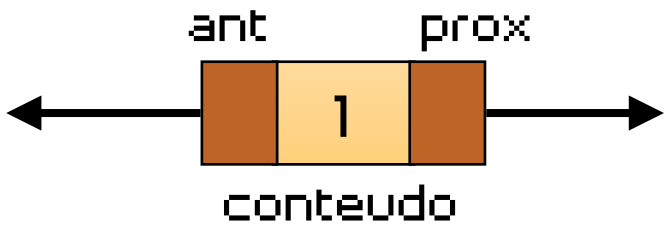
- Cada elemento tem um ponteiro para o próximo elemento e um ponteiro para o elemento anterior
- Dado um ponteiro para o último elemento da lista, é possível percorrer a lista em ordem inversa
- Vantagens
  - Acesso aos elementos anteriores
- Desvantagens
  - Atualizar os ponteiros é mais trabalhoso

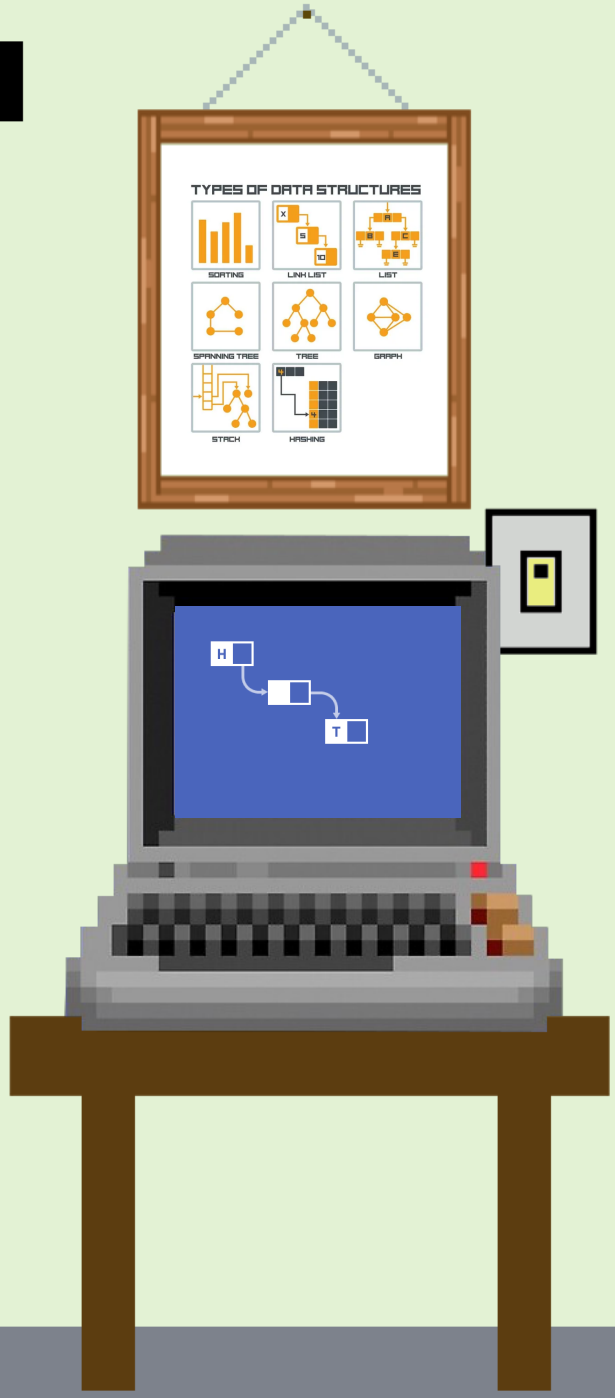


Listas Encadeadas Duplas

- Células armazenam o seu conteúdo, o endereço da célula seguinte e o endereço da célula anterior
- A última célula da lista deve apontar para NULL
- O anterior da cabeça deve apontar para NULL

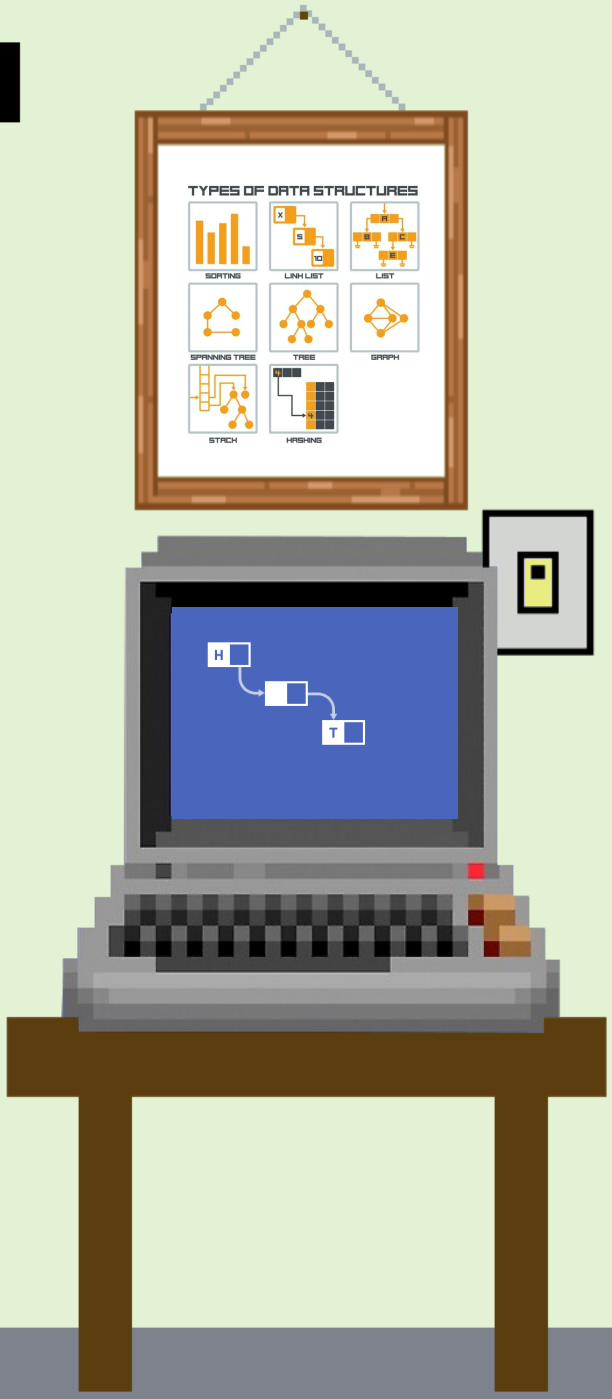
```
typedef struct cel {
        int conteudo;
        struct cel *prox;
        struct cel *ant;
} celula;
```





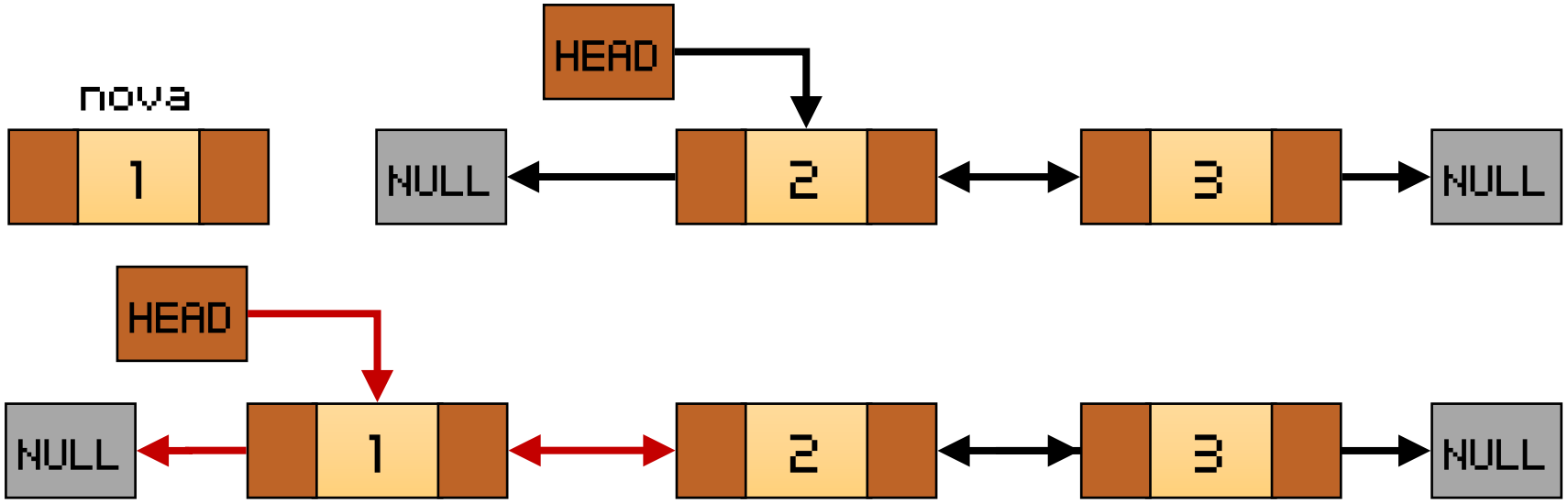
Principais Operações

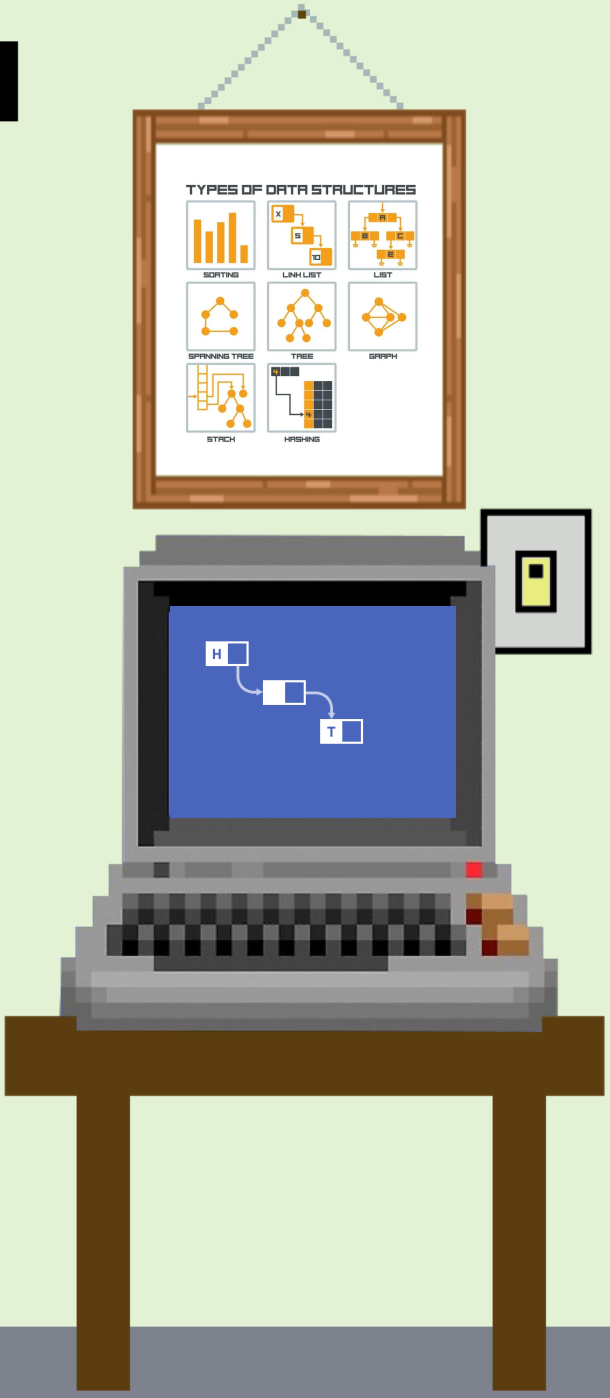
- Inserir elemento
  - Início
  - Meio
  - Fim
- Imprimir lista
- Buscar elemento
- Editar elemento
- Remover elemento



Inserir

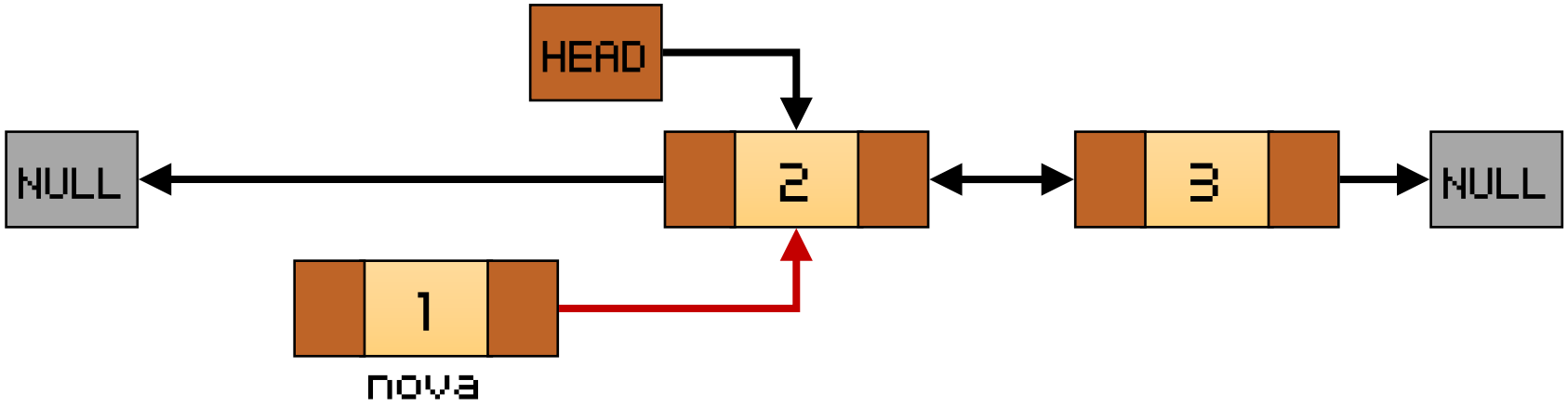
- Inicio
  - nova->prox = *ptr_cabeca;
  - nova->ant = NULL;
  - *ptr_cabeca->ant = nova;
  - *ptr_cabeca = nova;

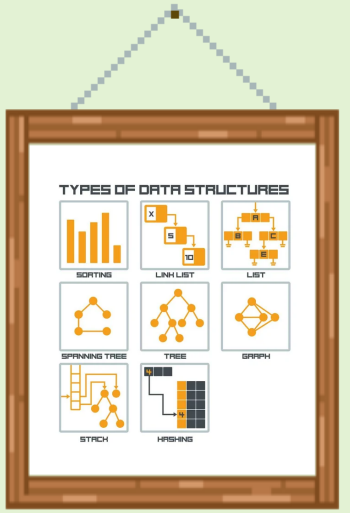


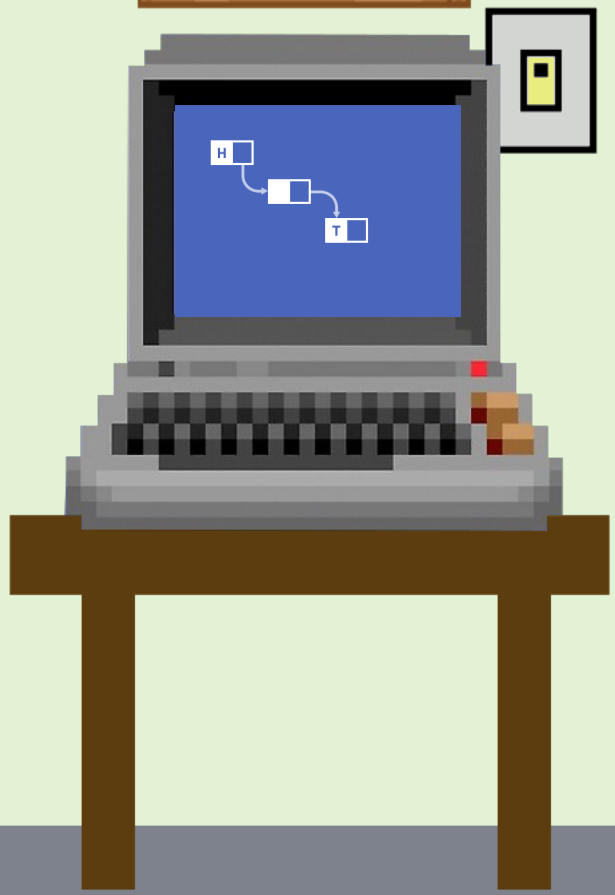


Inserir

- Inicio
  - nova->prox = *ptr_cabeca;
  - nova->ant = NULL;
  - *ptr_cabeca->ant = nova;
  - *ptr_cabeca = nova;

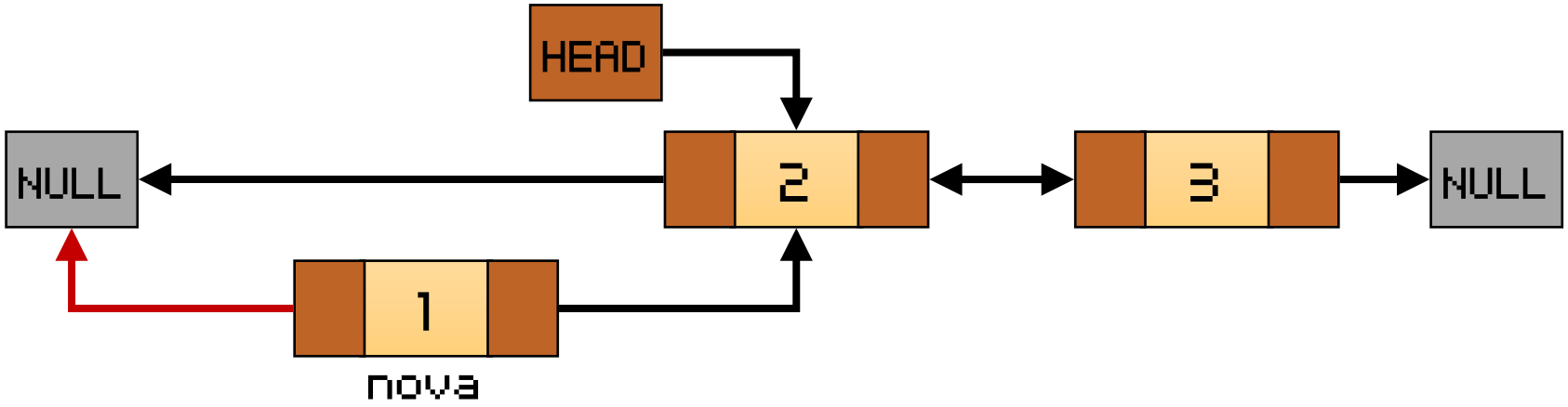


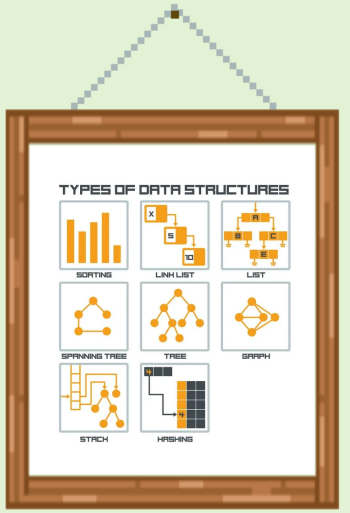


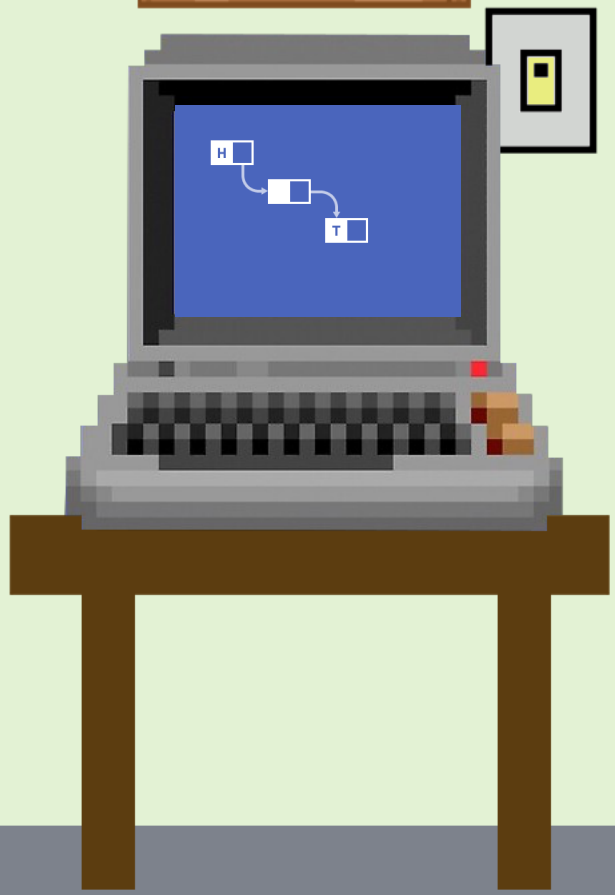


Inserir

- Inicio
  - nova->prox = *ptr_cabeca;
  - nova->ant = NULL;
  - *ptr_cabeca->ant = nova;
  - *ptr_cabeca = nova;

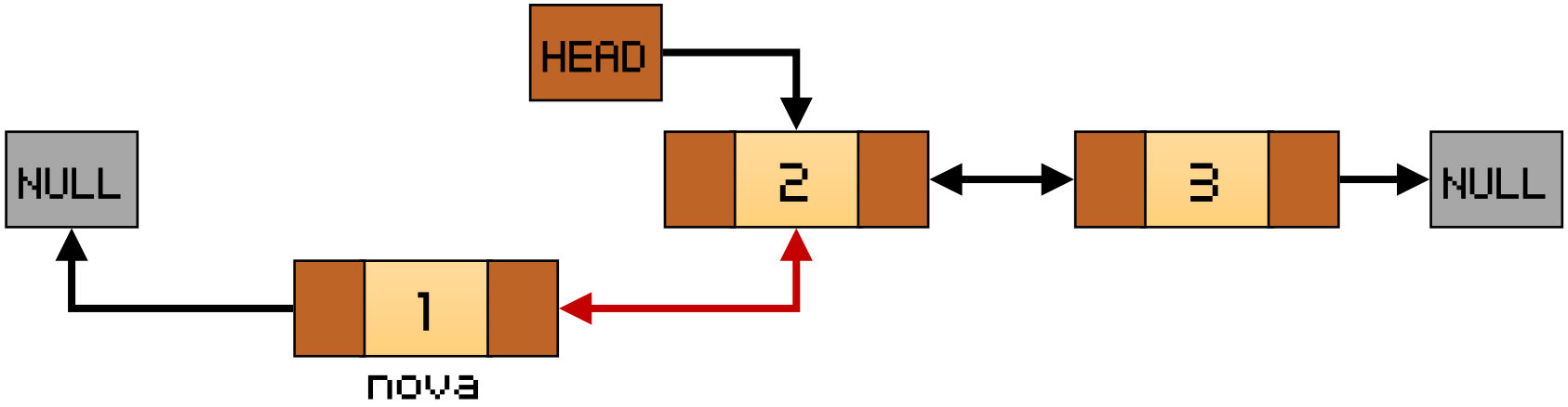


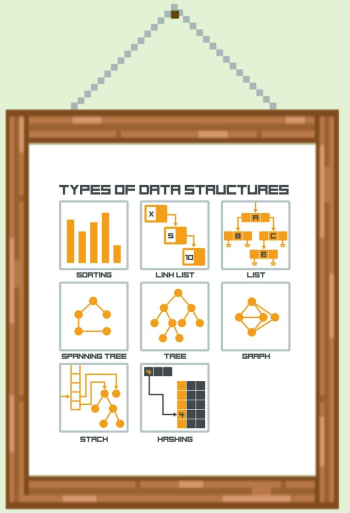


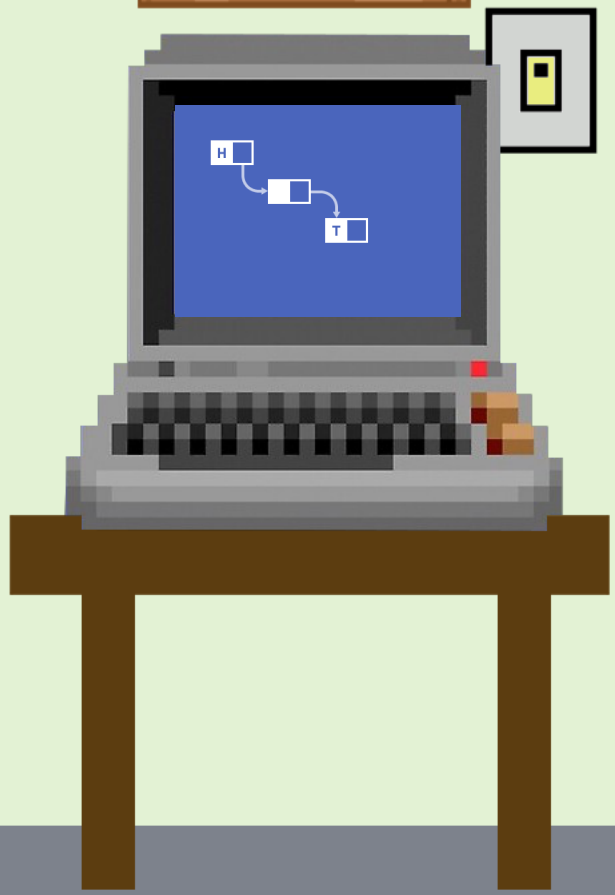


Inserir

- Inicio
  - nova->prox = *ptr_cabeca;
  - nova->ant = NULL;
  - *ptr_cabeca->ant = nova;
  - *ptr_cabeca = nova;

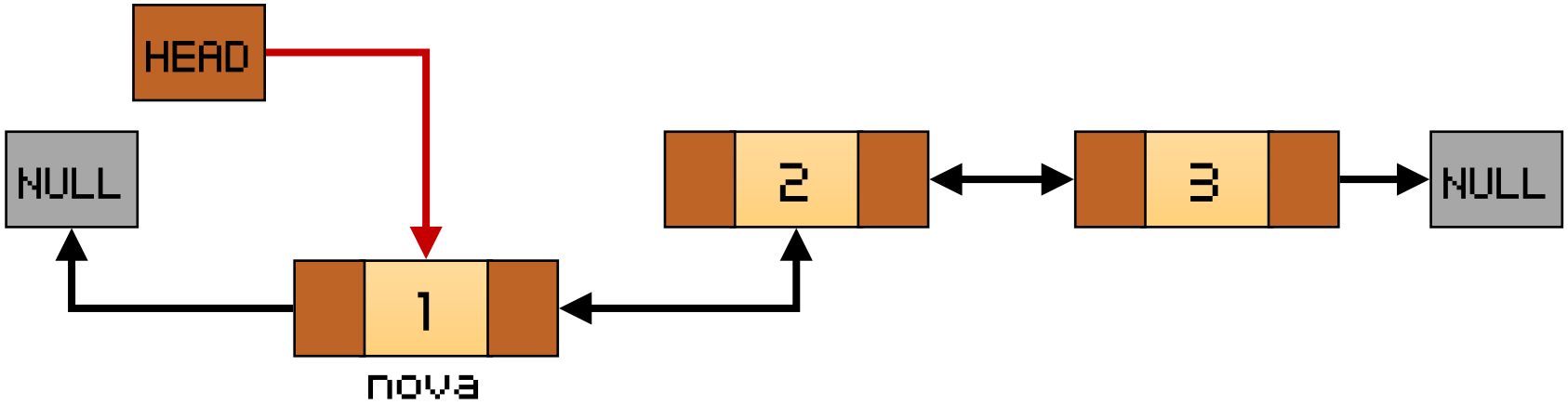


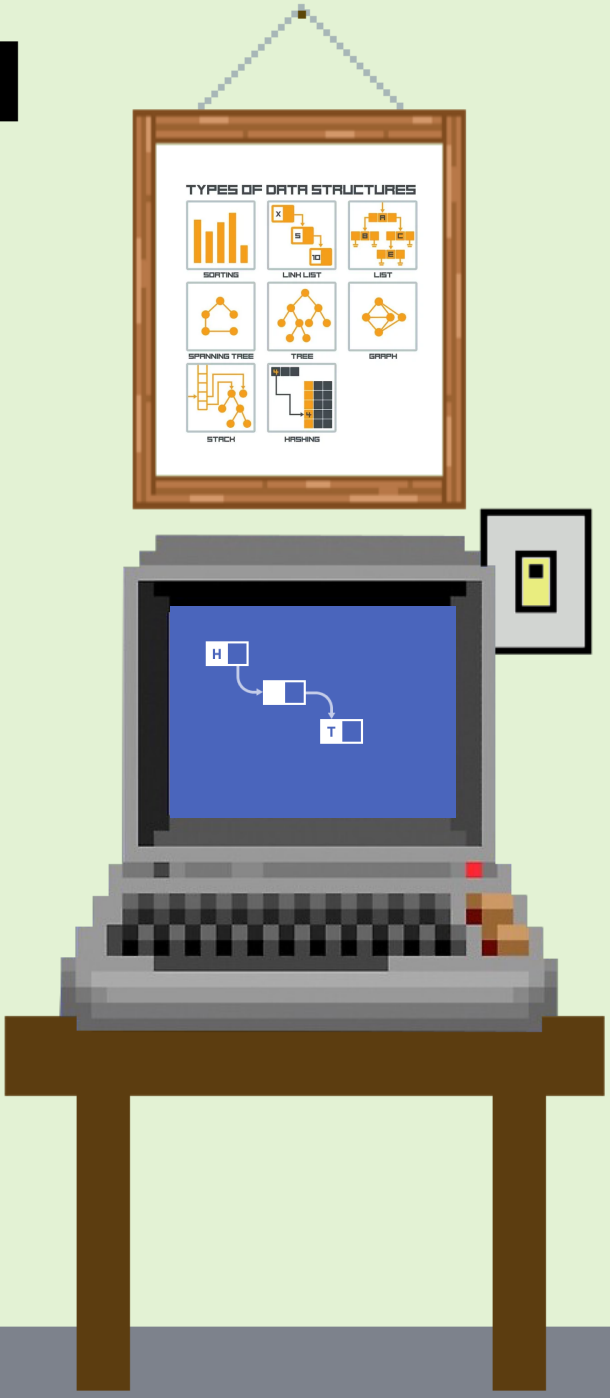




Inserir

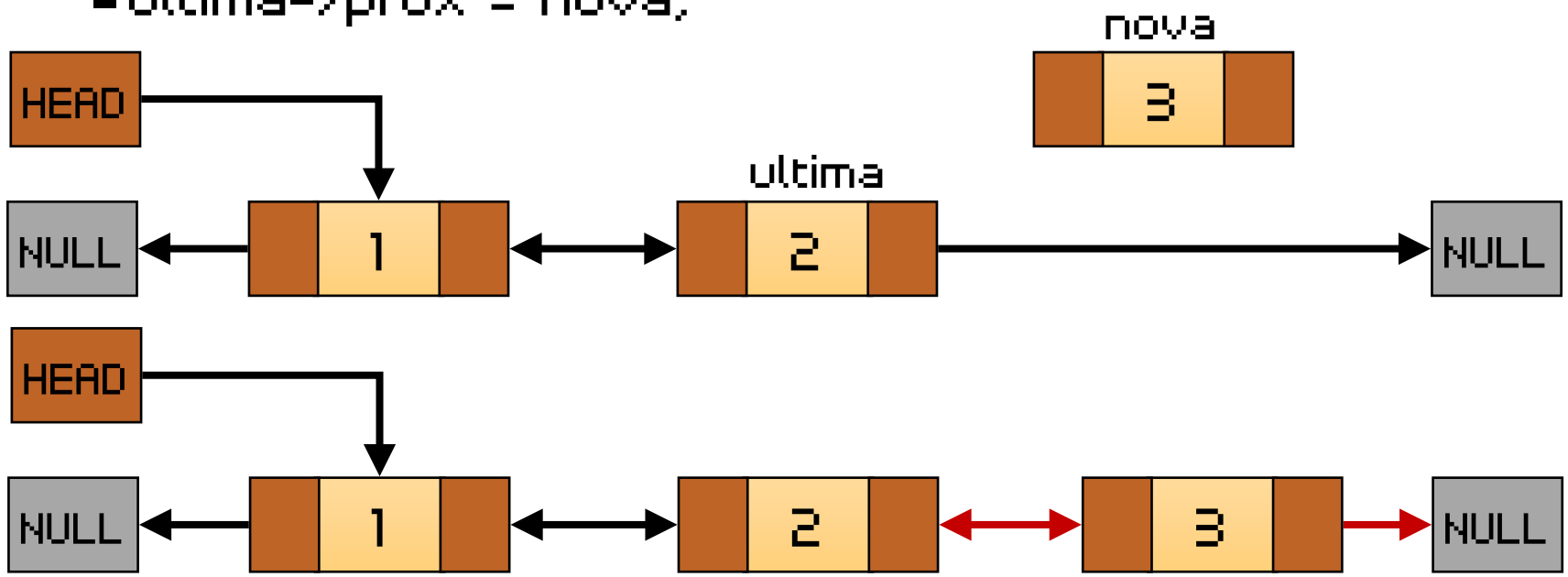
- Inicio
  - nova->prox = *ptr_cabeca;
  - nova->ant = NULL;
  - *ptr_cabeca->ant = nova;
  - *ptr_cabeca = nova;

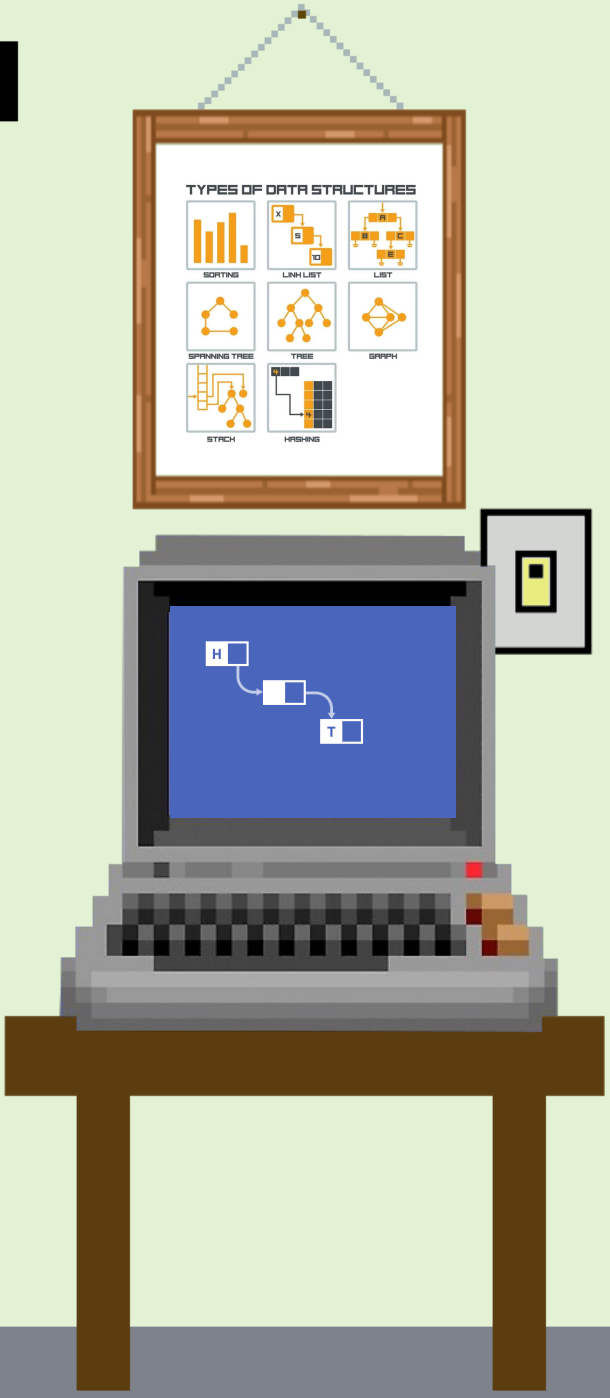




Inserir

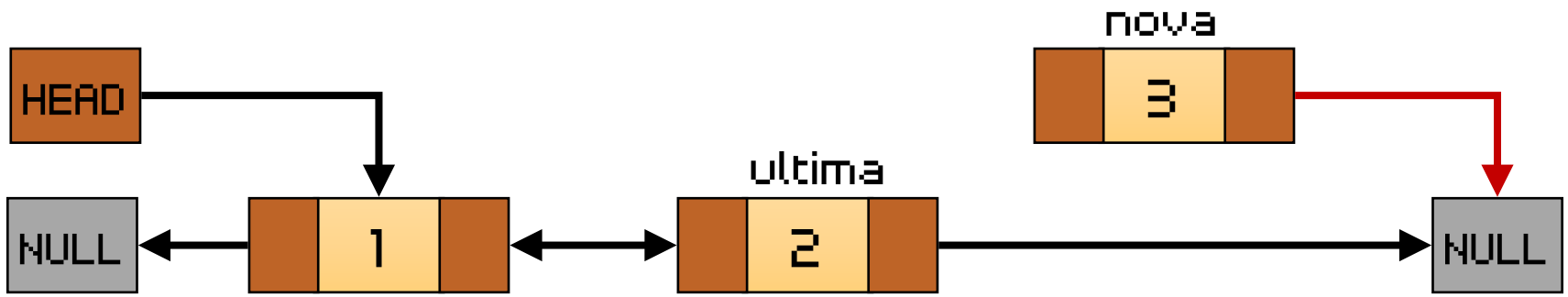
- Fim
  - nova->prox = NULL;
  - nova->ant = ultima;
  - ultima->prox = nova;

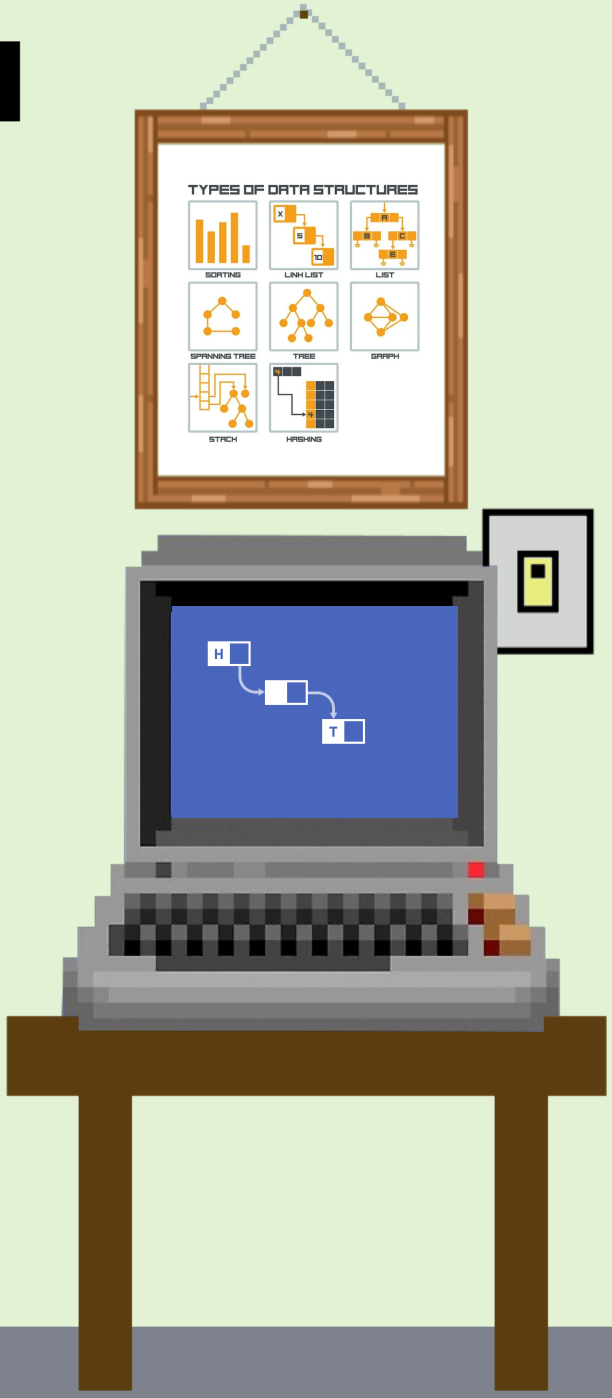




Inserir

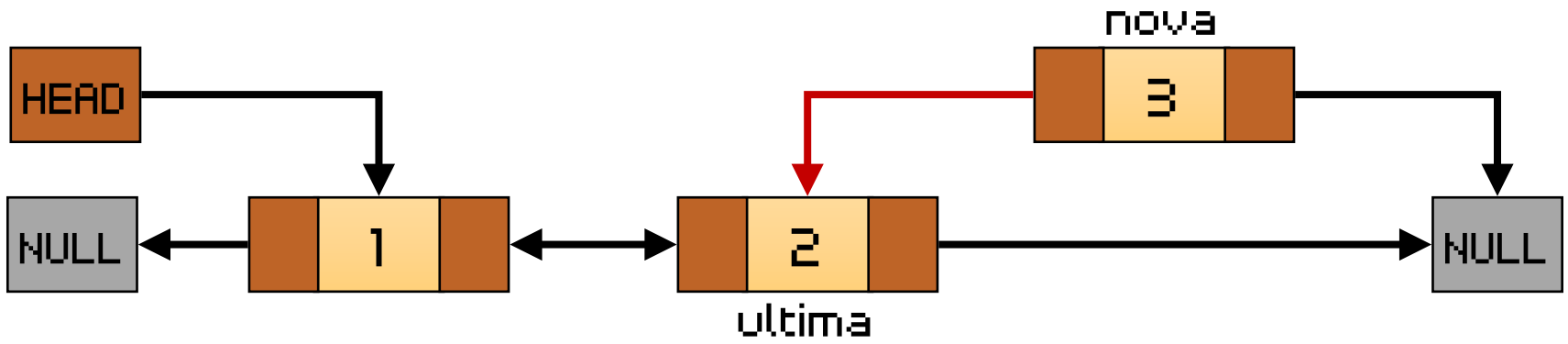
- Fim
  - nova->prox = NULL;
  - nova->ant = ultima;
  - ultima->prox = nova;

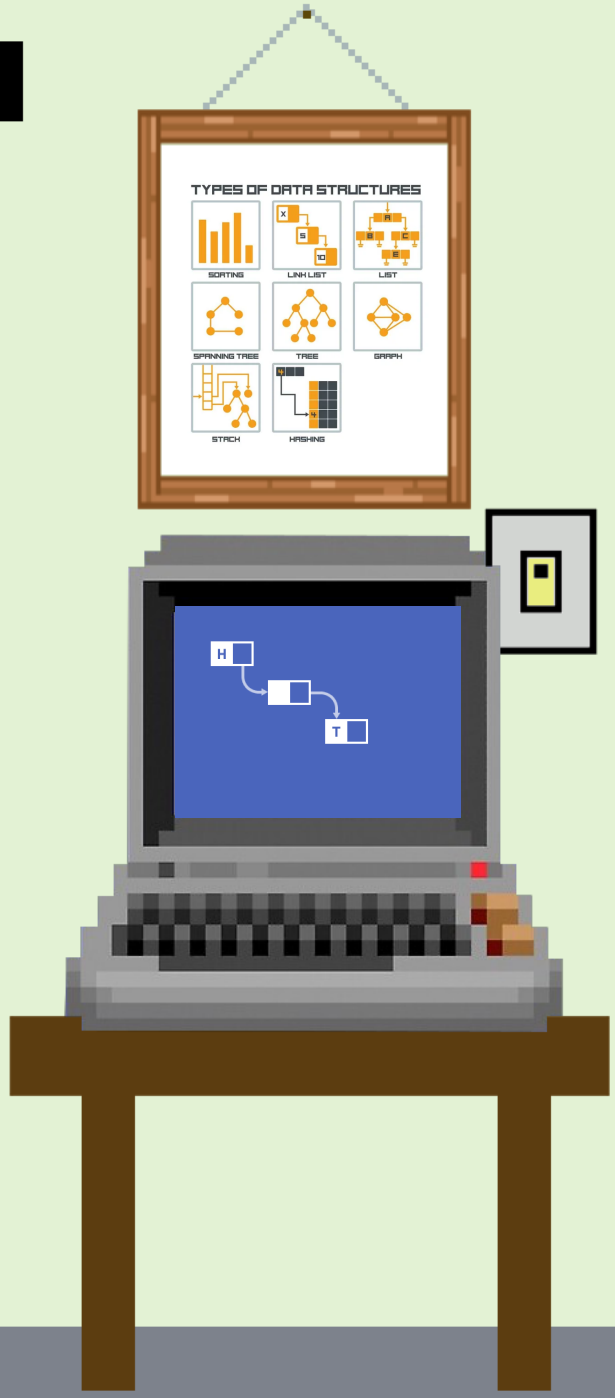




Inserir

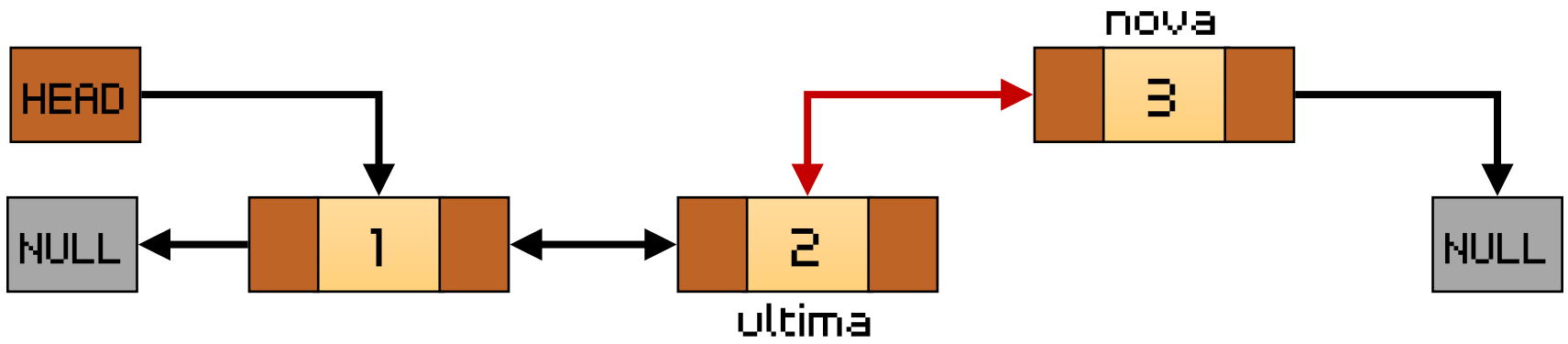
- Fim
  - nova->prox = NULL;
  - nova->ant = ultima;
  - ultima->prox = nova;

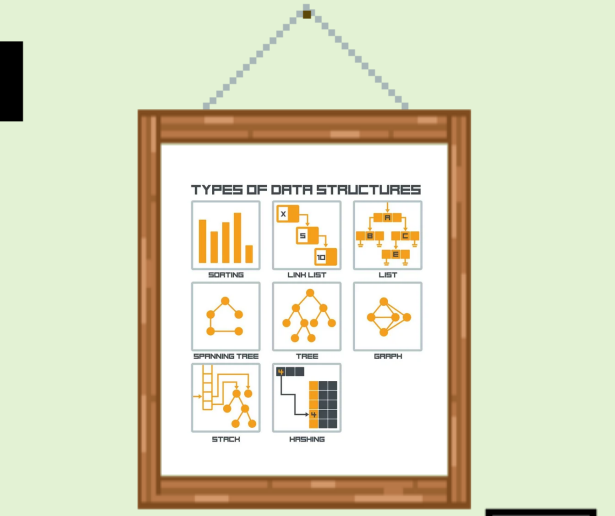


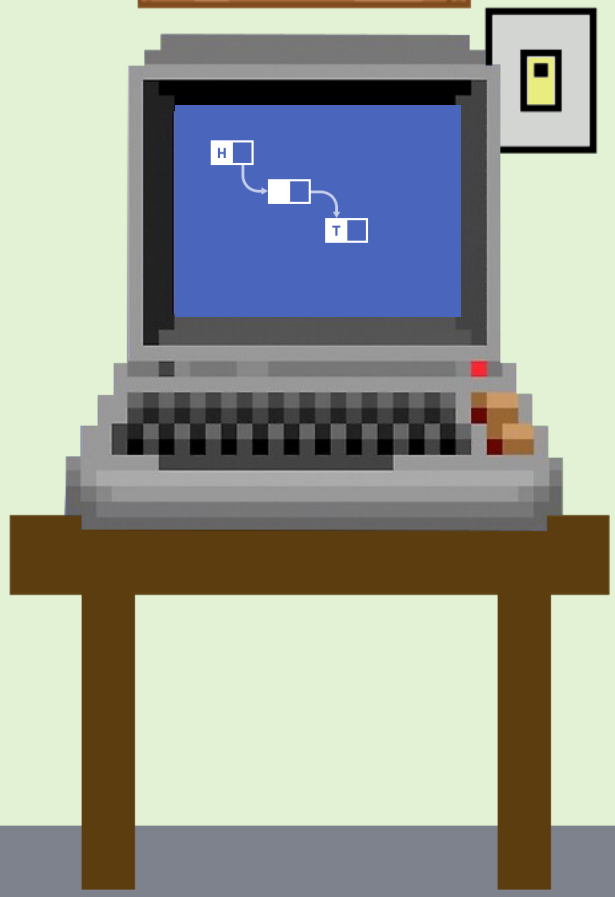


Inserir

- Fim
  - nova->prox = NULL;
  - nova->ant = ultima;
  - ultima->prox = nova;

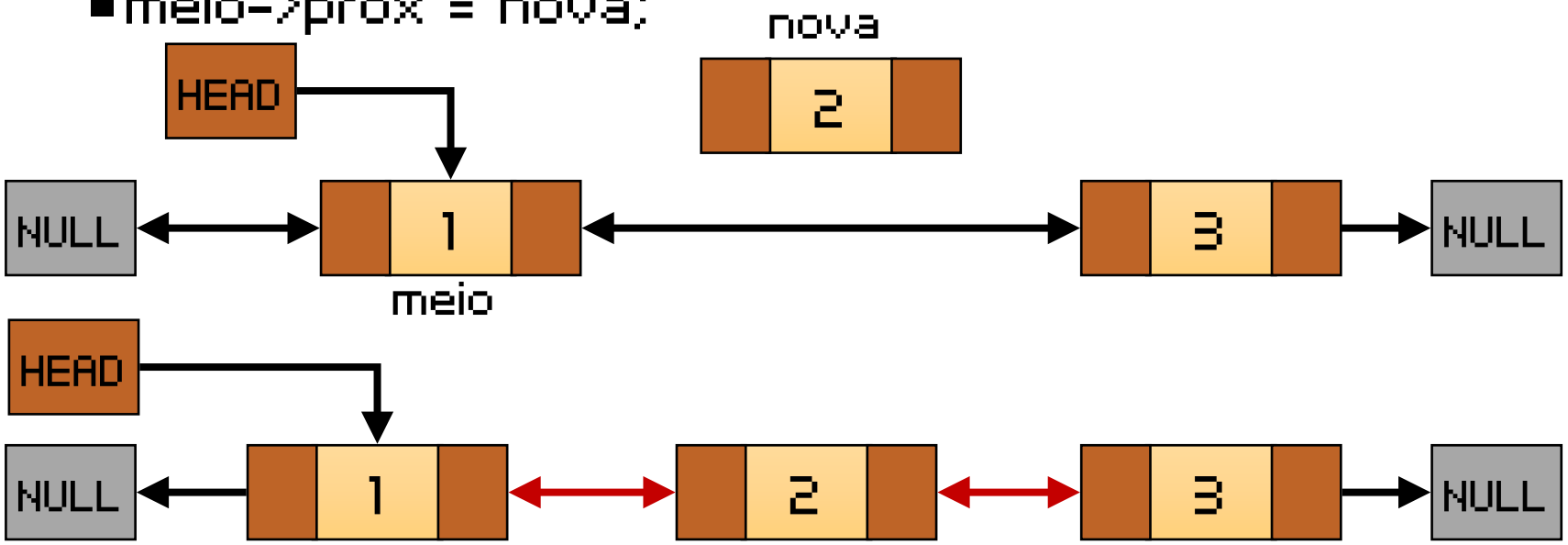


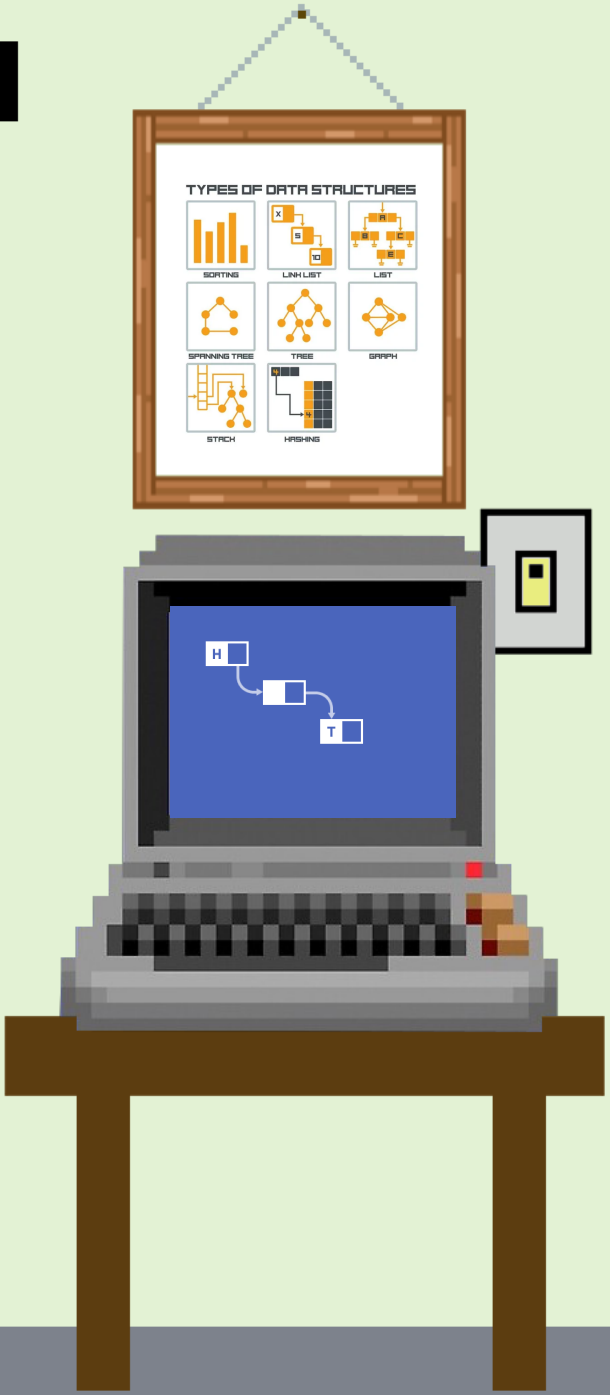




Inserir

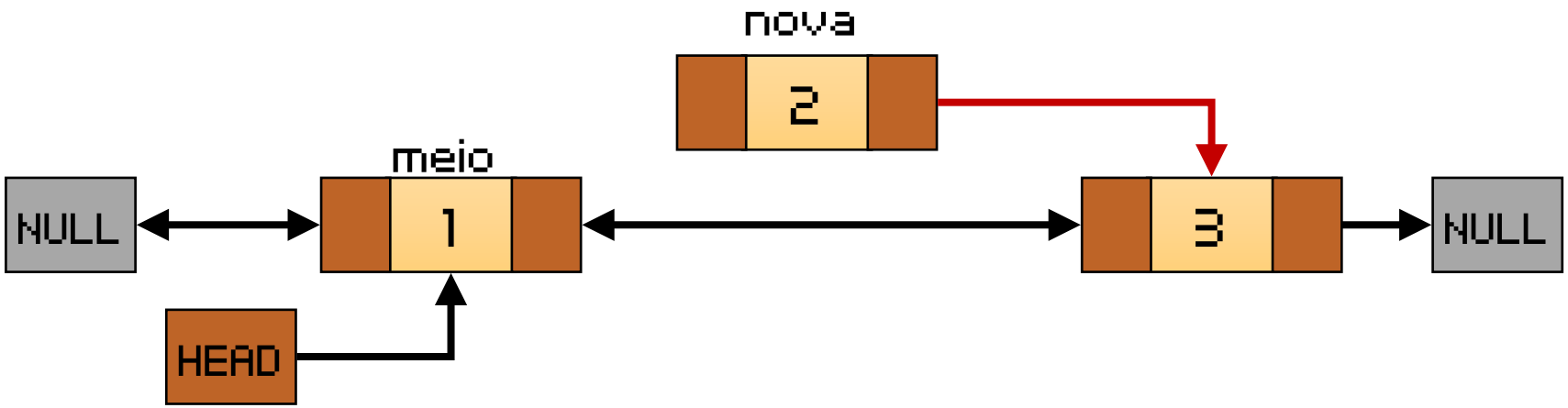
- Meio
  - nova->prox = meio->prox;
  - nova->ant = meio;
  - meio->prox->ant = nova;
  - meio->prox = nova;

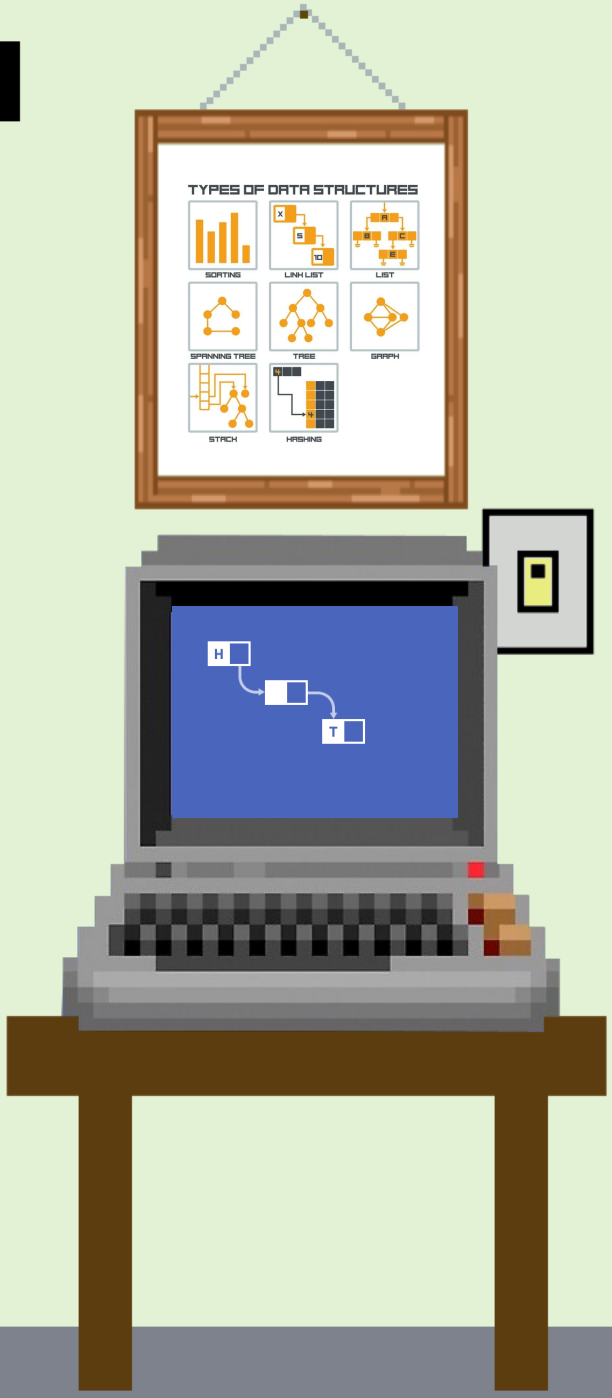




Inserir

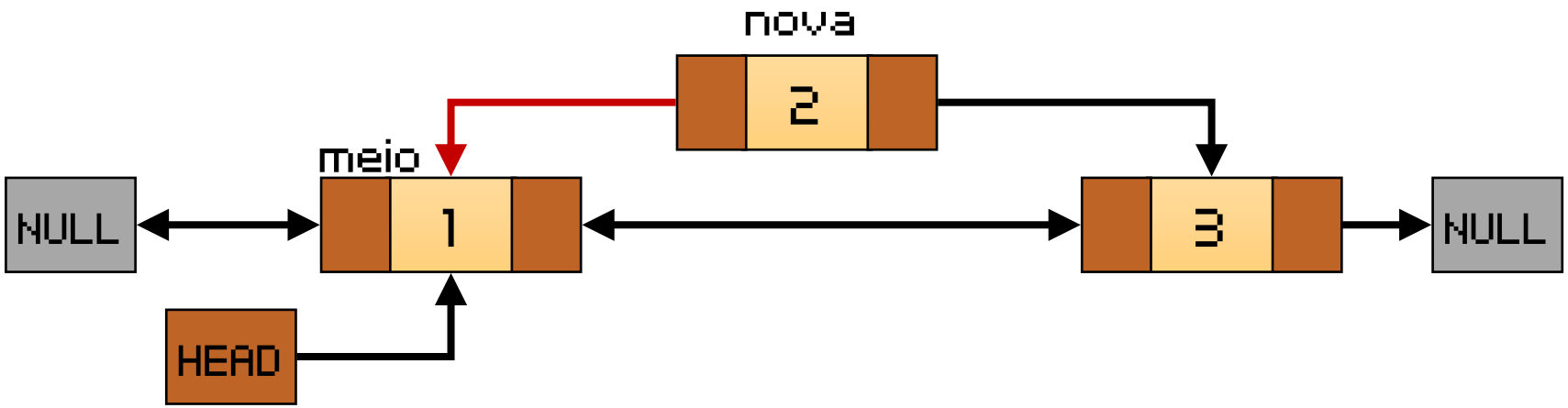
- Meio
  - nova->prox = meio->prox;
  - nova->ant = meio;
  - meio->prox->ant = nova;
  - meio->prox = nova;

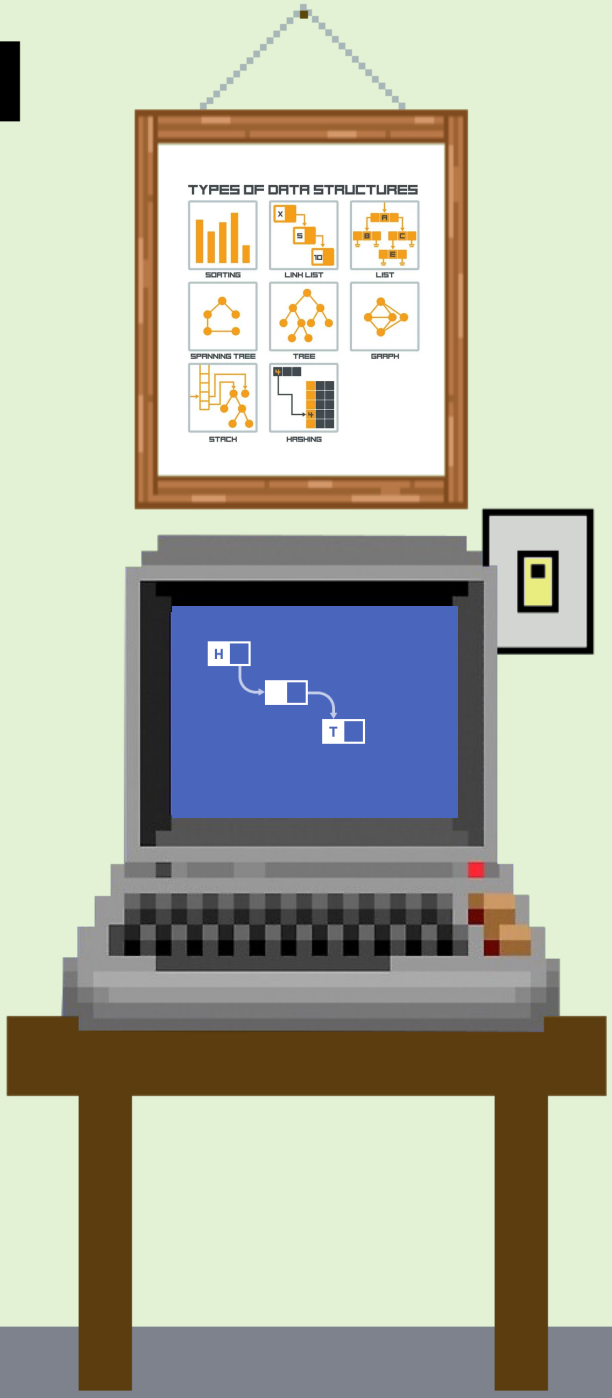




Inserir

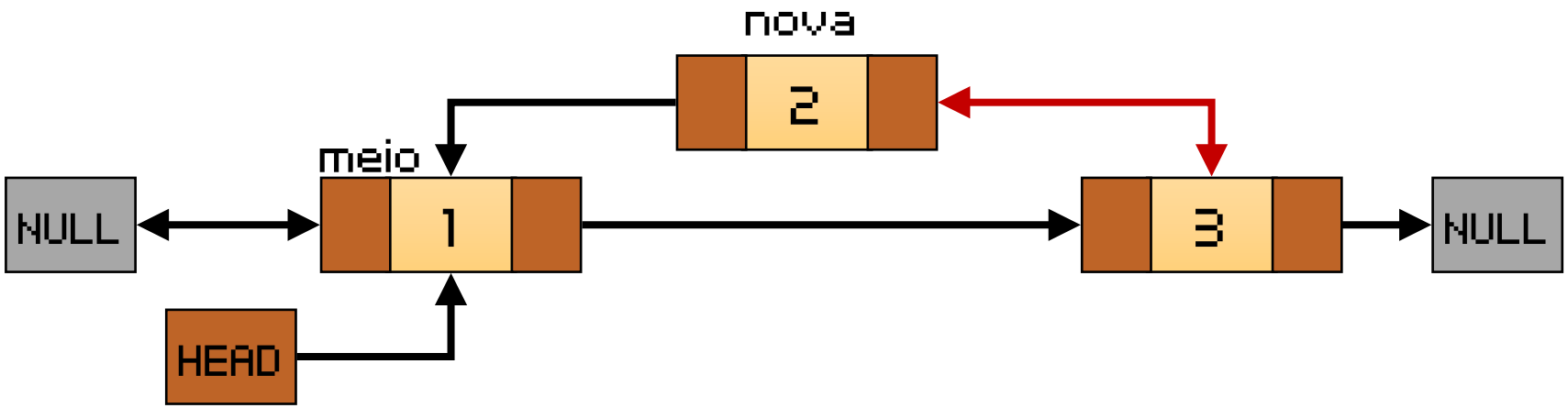
- Meio
  - nova->prox = meio->prox;
  - nova->ant = meio;
  - meio->prox->ant = nova;
  - meio->prox = nova;

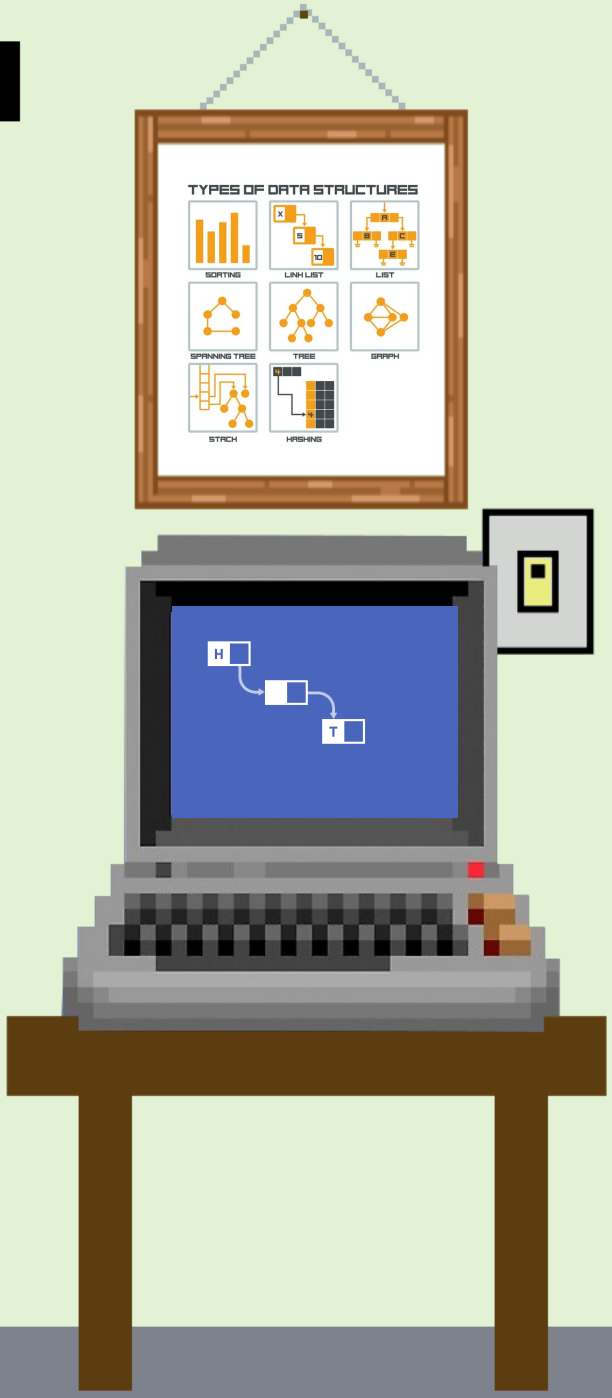




Inserir

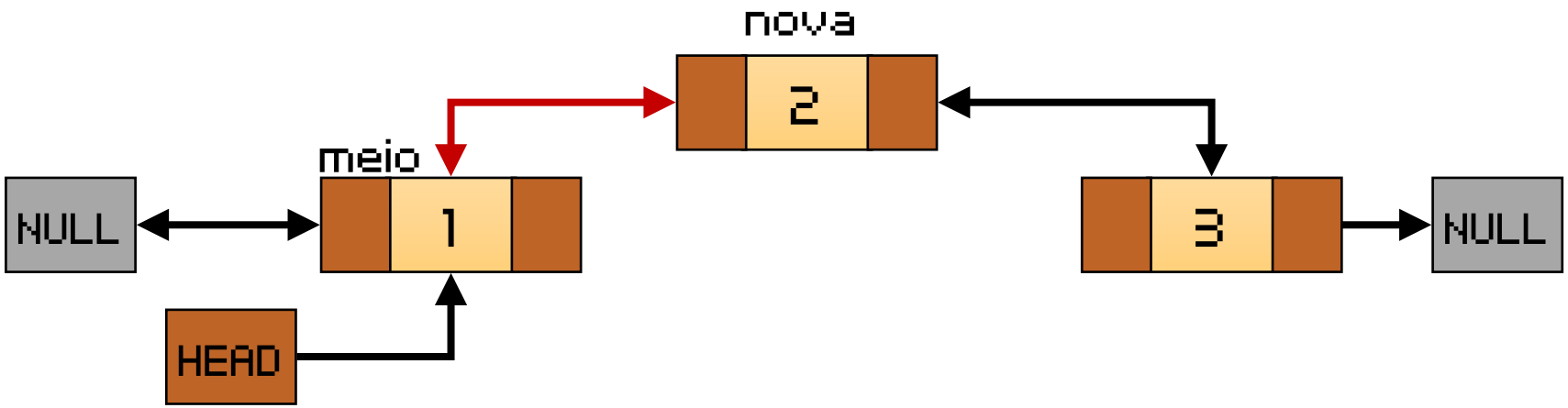
- Meio
  - nova->prox = meio->prox;
  - nova->ant = meio;
  - meio->prox->ant = nova;
  - meio->prox = nova;

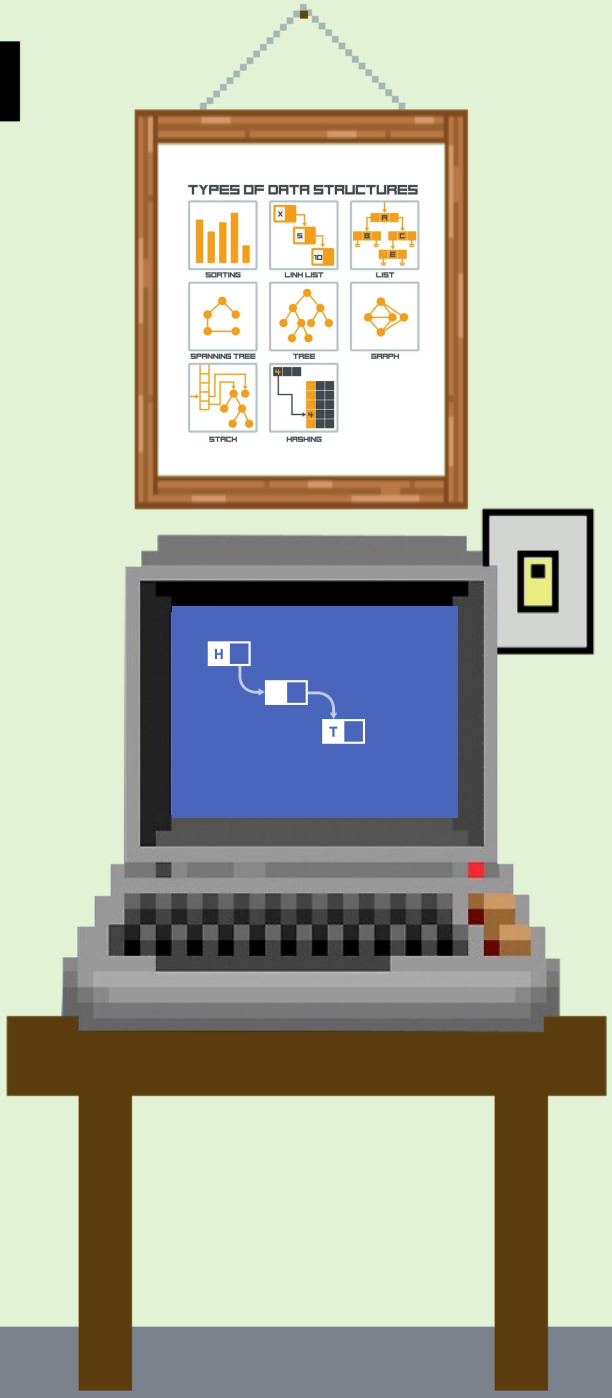




Inserir

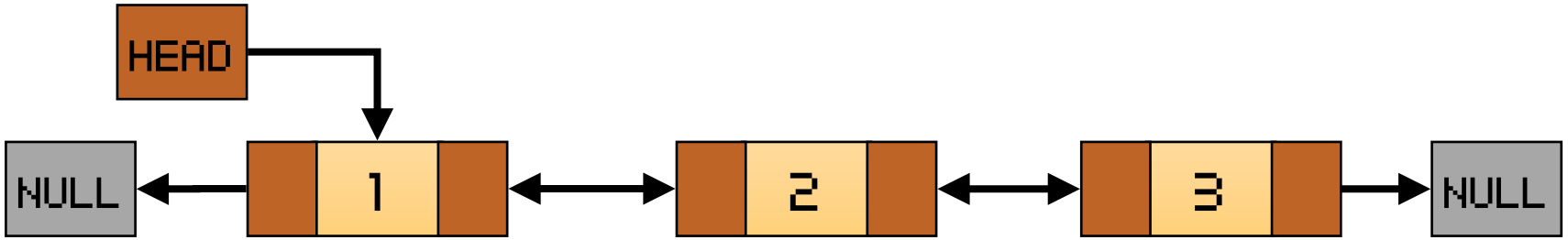
- Meio
  - nova->prox = meio->prox;
  - nova->ant = meio;
  - meio->prox->ant = nova;
  - meio->prox = nova;

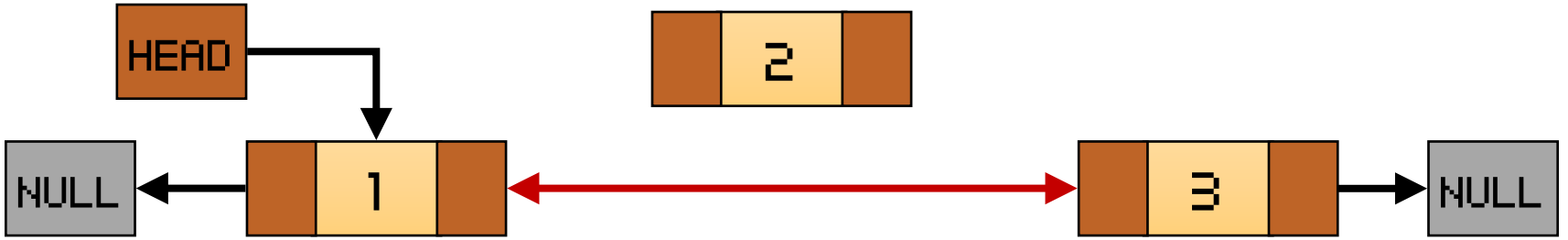


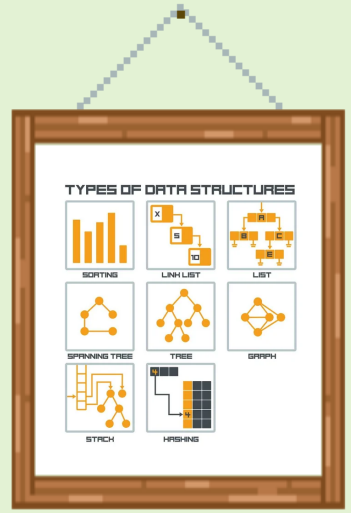


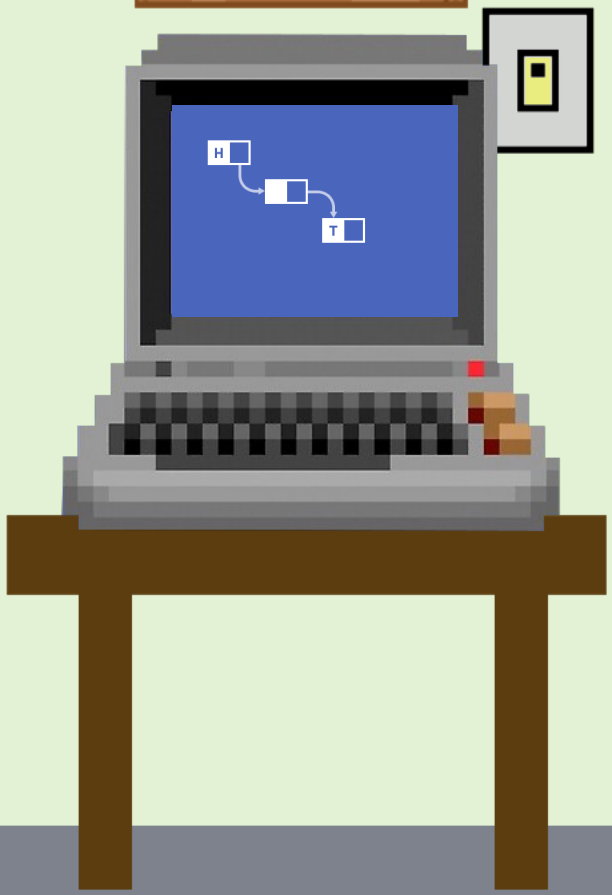
Remover

- rem->prox->ant = rem->ant;
- rem->ant->prox = rem->prox;
- rem->prox=NULL;
- rem->ant=NULL;



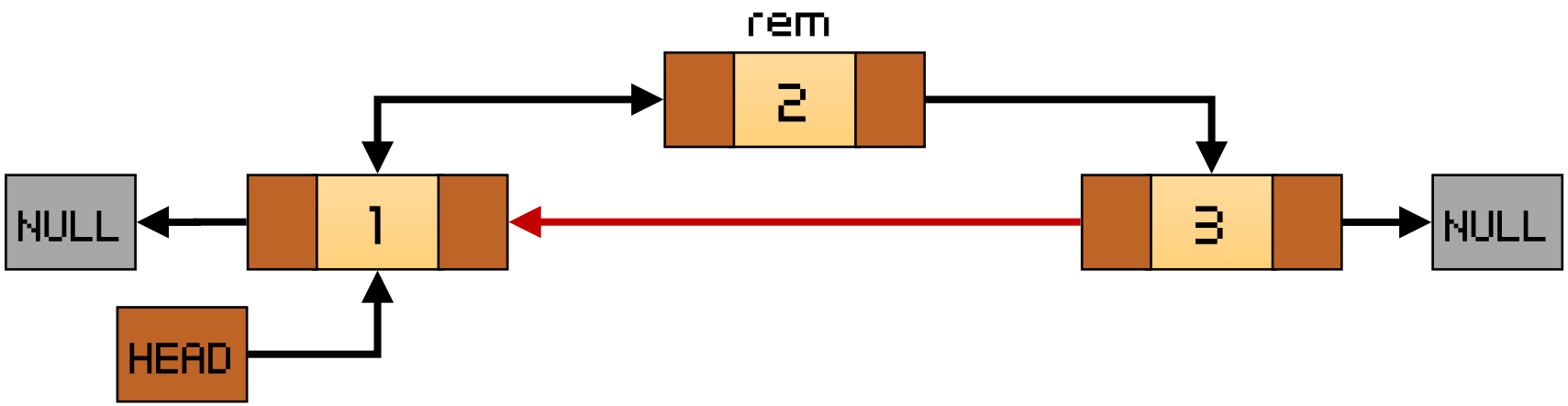


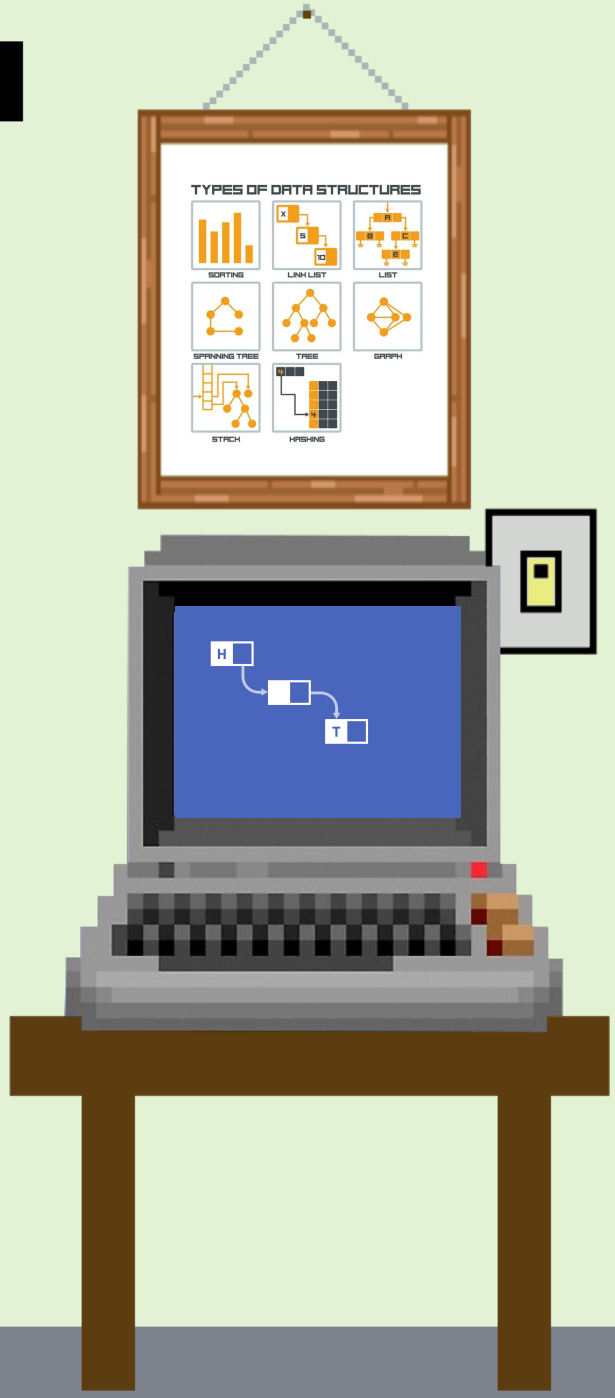




Remover

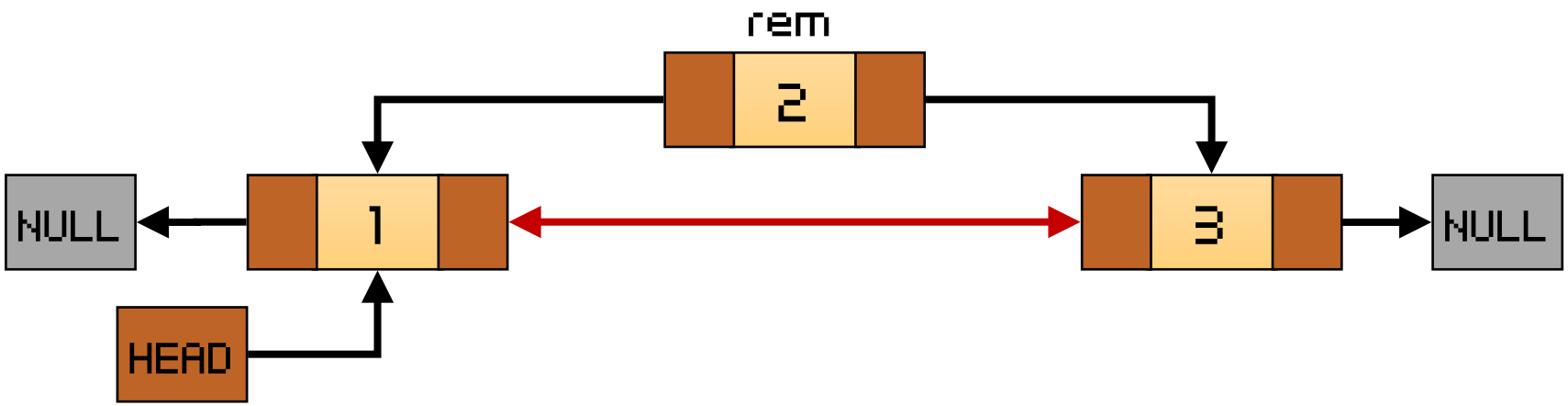
- rem->prox->ant = rem->ant;
- rem->ant->prox = rem->prox;
- rem->prox=NULL;
- rem->ant=NULL;

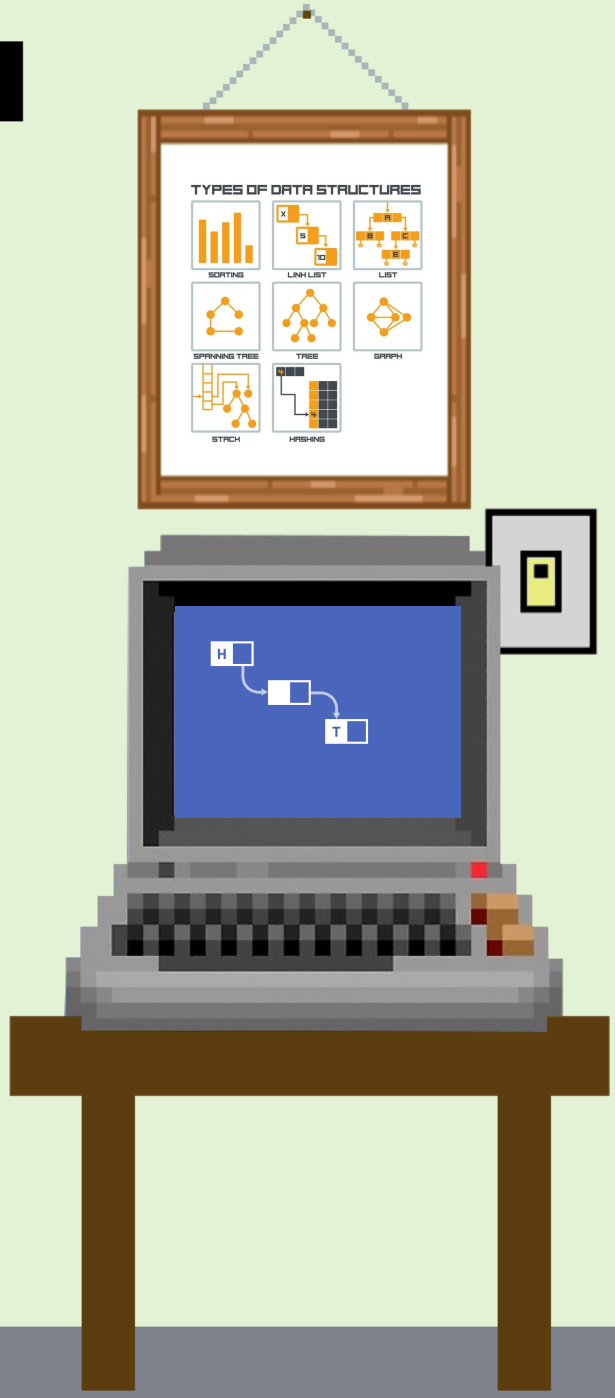




Remover

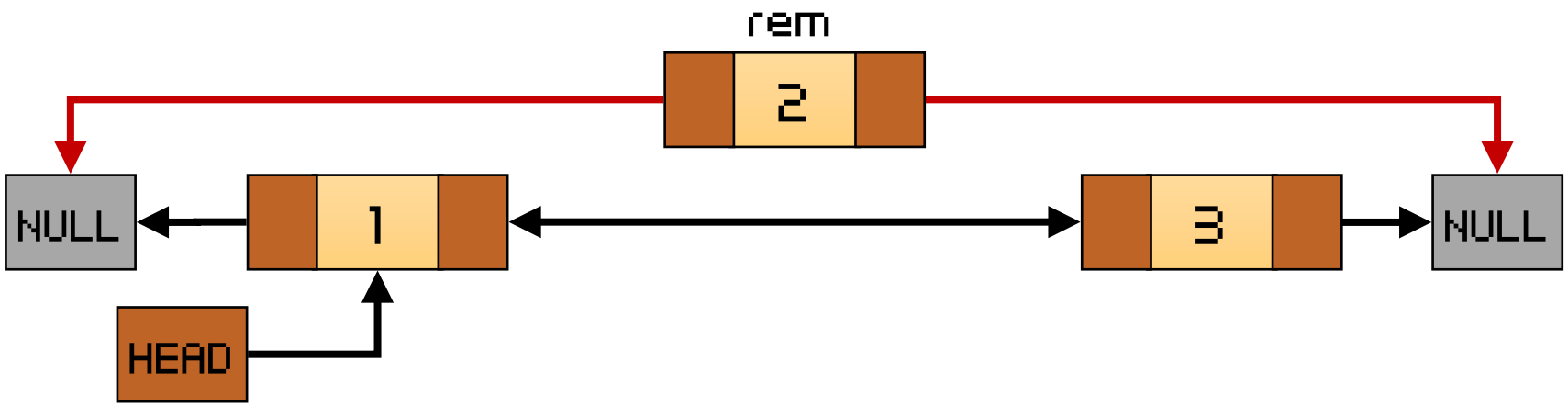
- rem->prox->ant = rem->ant;
- rem->ant->prox = rem->prox;
- rem->prox=NULL;
- rem->ant=NULL;

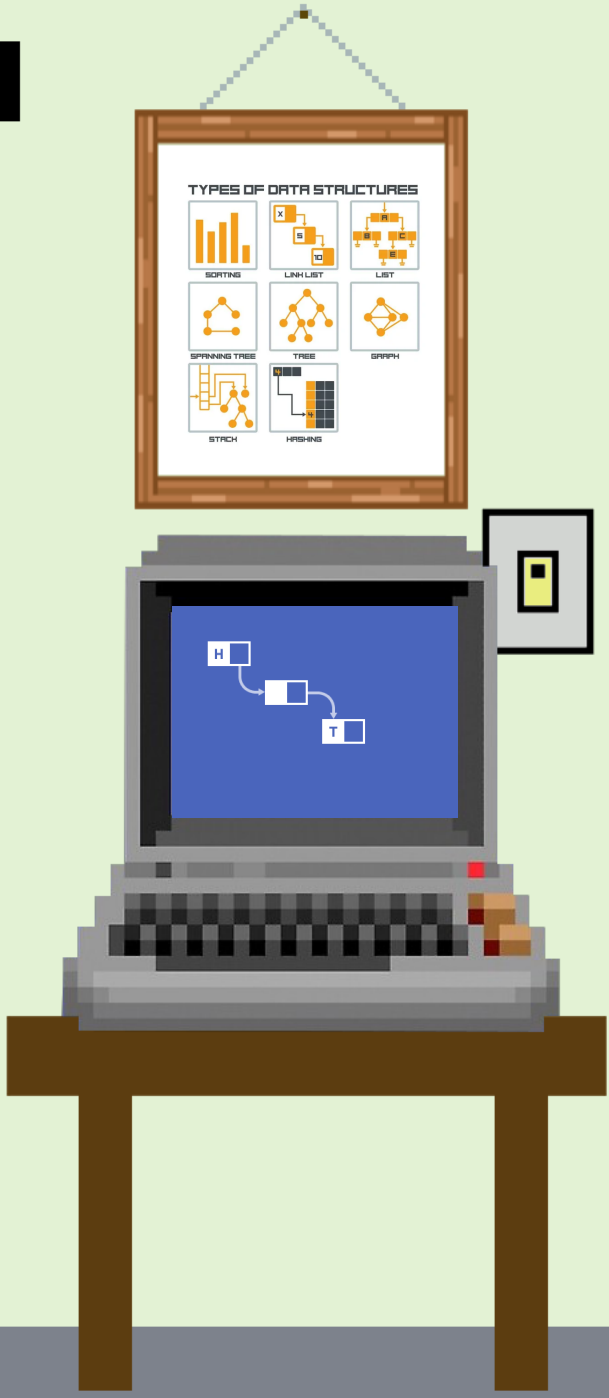




Remover

- rem->prox->ant = rem->ant;
- rem->ant->prox = rem->prox;
- rem->prox=NULL;
- rem->ant=NULL;





Atividade

- Implemente as funções descritas abaixo sobre listas encadeadas duplas
  - Localizar um elemento da lista e trocar de posição com o próximo
  - Receber duas listas e concatenar em uma só
  - Receber uma lista e dividir em duas na metade