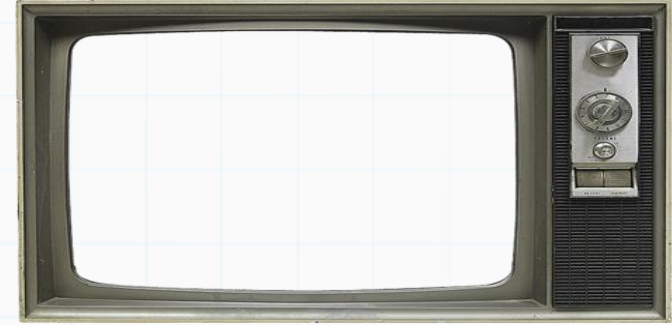


Programação Estruturada

Professor : Yuri Frota

yuri@ic.uff.br

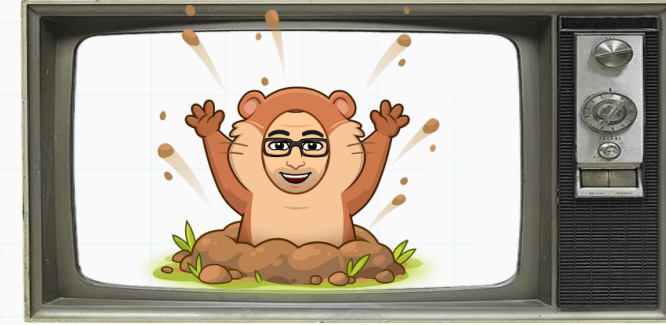


- Utilize os arquivos fornecidos main.c e tad.c com o TAD básico de listas encadeadas para fazer as questões a seguir



Listas encadeadas- LAB

1) Dado duas listas, escreva uma função para concatenar as duas listas em uma nova lista, apenas fazendo redirecionamento de ponteiros (**sem alocar nenhum nó novo**), e **sem utilizar/alocar estruturas auxiliares (vetores ou listas)**.



Código da main.c

```
int main()
{
    int k;

    lista *L1 = NULL;
    L1 = insere_lista(L1, 8);
    L1 = insere_lista(L1, 6);
    L1 = insere_lista(L1, 9);
    L1 = insere_lista(L1, 2);
    imprime_lista(L1);

    lista *L2 = NULL;
    L2 = insere_lista(L2, 7);
    L2 = insere_lista(L2, 4);
    L2 = insere_lista(L2, 10);
    imprime_lista(L2);

    lista *L3 = concatena_listas(L1,L2);
    imprime_lista(L3);

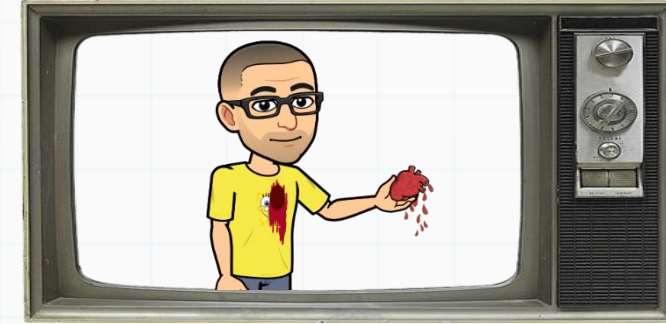
    L3 = exclui_lista (L3);
    return 0;
}
```

Exemplo de execução:

```
L = 2, 9, 6, 8,
L = 10, 4, 7,
L = 2, 9, 6, 8, 10, 4, 7,
```

Use só o que
aprendemos
até hoje

Listas encadeadas- LAB



2) Escreva uma função que dada uma lista L e um número inteiro $k > 0$, imprima os últimos k elementos da lista. Não utilizar/alocar estruturas auxiliares (vetores ou listas).

Código da main.c

```
int main()
{
    int k;
    lista *L = NULL;

    L = insere_lista(L, 34);
    L = insere_lista(L, 3);
    L = insere_lista(L, 15);
    L = insere_lista(L, 25);
    L = insere_lista(L, 65);
    L = insere_lista(L, 7);
    L = insere_lista(L, 98);
    imprime_lista(L);

    imprime_k_lista(L, 3);
    imprime_k_lista(L, 1);
    imprime_k_lista(L, 7);
    imprime_k_lista(L, 9);

    L = exclui_lista (L);
    return 0;
}
```

Exemplo de execução:

```
L = 98, 7, 65, 25, 15, 3, 34,
3 ultimos elementos = 15, 3, 34,
1 ultimos elementos = 34,
7 ultimos elementos = 98, 7, 65, 25, 15, 3, 34,
k invalido
```

Use só o que
aprendemos
até hoje

Listas encadeadas- LAB

3) Escreva uma função que remova duplicações em uma lista ordenada sem utilizar/alocar estruturas auxiliares (vetores ou listas).

Código da main.c

```
int main()
{
    lista *L = NULL;

    L = insere_lista(L, 33);
    L = insere_lista(L, 33);
    L = insere_lista(L, 33);
    L = insere_lista(L, 33);
    L = insere_lista(L, 28);
    L = insere_lista(L, 28);
    L = insere_lista(L, 18);
    L = insere_lista(L, 2);
    L = insere_lista(L, 2);
    L = insere_lista(L, 2);
    imprime_lista(L);

    L = remove_duplicados (L);
    imprime_lista(L);

    L = exclui_lista (L);
    return 0;
}
```

Exemplo de execução:

```
L = 2, 2, 2, 18, 28, 28, 33, 33, 33, 33,
sem duplicacao
L = 2, 18, 28, 33,
```



Use só o que
aprendemos
até hoje

Listas encadeadas- LAB

4) Escreva uma função que remova da lista todas as ocorrências de um valor “el”. A função deve devolver o novo início da lista. sem utilizar/alocar estruturas auxiliares (vetores ou listas).

Código da main.c

```
#include <stdio.h>
#include <stdlib.h>
#include "tad.c"

int main()
{
    int k;

    lista *L = NULL;
    L = insere_lista(L, 4);
    L = insere_lista(L, 4);
    L = insere_lista(L, 3);
    L = insere_lista(L, 4);
    L = insere_lista(L, 4);
    L = insere_lista(L, 4);
    L = insere_lista(L, 5);
    L = insere_lista(L, 7);
    L = insere_lista(L, 4);
    imprime_lista(L);

    L = remove_elemento(L, 4);
    imprime_lista(L);

    L = exclui_lista(L);

    return 0;
}
```

Exemplo de execução:

```
L = 4, 7, 5, 4, 4, 3, 4, 4,
L = 7, 5, 3,
```



Use só o que aprendemos até hoje



Listas encadeadas- LAB

copia e cola



```
int main()
{
    lista *L1 = NULL;
    lista *L2 = NULL;

    L1 = insere_lista(L1, 12); L1 = insere_lista(L1, 10);
    L1 = insere_lista(L1, 8);  L1 = insere_lista(L1, 8);
    L1 = insere_lista(L1, 6);  L1 = insere_lista(L1, 4);
    L1 = insere_lista(L1, 3);  L1 = insere_lista(L1, 3);
    L1 = insere_lista(L1, 1);
    imprime_lista(L1);

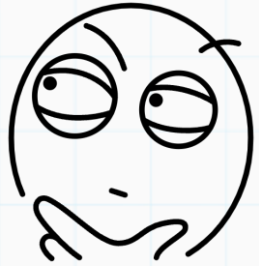
    L2 = insere_lista(L2, 21); L2 = insere_lista(L2, 18);
    L2 = insere_lista(L2, 15); L2 = insere_lista(L2, 15);
    L2 = insere_lista(L2, 13); L2 = insere_lista(L2, 12);
    L2 = insere_lista(L2, 12);
    imprime_lista(L2);

    L1 = concatena_listas(L1, L2); L1 = remove_duplicados(L1);
    L1 = remove_elemento(L1, 10);  L1 = insere_lista(L1, 14);
    L1 = remove_elemento(L1, 3);   L1 = insere_lista(L1, 9);
    L1 = remove_elemento(L1, 18);  L1 = insere_lista(L1, 4);
    L1 = remove_elemento(L1, 12);
    imprime_lista(L1);

    int valor_final = 1*(L1->info) + 2*(L1->prox->info) + 3*(L1->prox->prox->info);
    valor_final      = valor_final+ 4*(L1->prox->prox->prox->info) + 5*(L1->prox->prox->prox->prox->info);
    printf("valor final = %d\n",valor_final);

    L1 = exclui_lista (L1);
    return 0;
}
```

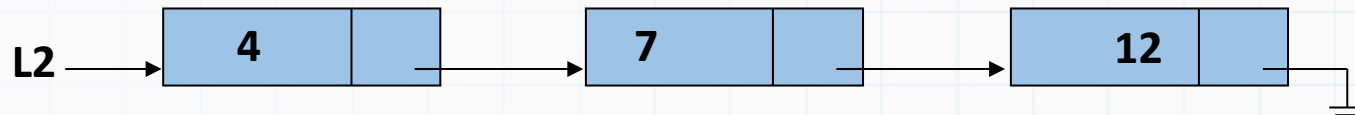
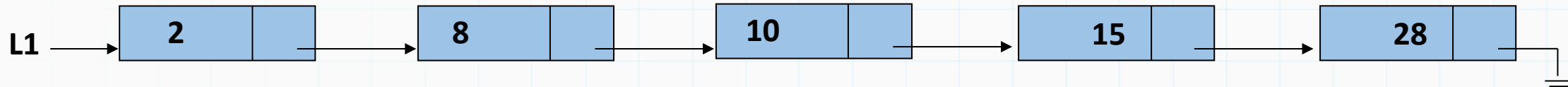
Dado que as funções anteriores foram implementadas, se rodarmos o código ao lado, qual seria o valor impresso final ?



Listas encadeadas- LAB



5) Dado duas listas ordenadas, escreva uma função que mescle (“merge”) as duas listas em uma nova lista ordenada, apenas fazendo redirecionamento de ponteiros (sem alocar novos nos), e sem utilizar/alocar estruturas auxiliares (vetores ou listas). Veja exemplo:



Em BDs massivos, as informações armazenadas nos nós são da ordem de Terabytes, tornando impraticável copiar informações entre os nós.

Então qualquer alteração feita na estrutura da lista deve ser feita apenas redirecionando ponteiros

Vamos utilizar 2 ponteiros auxiliares para percorrer as listas: inicio e fim.

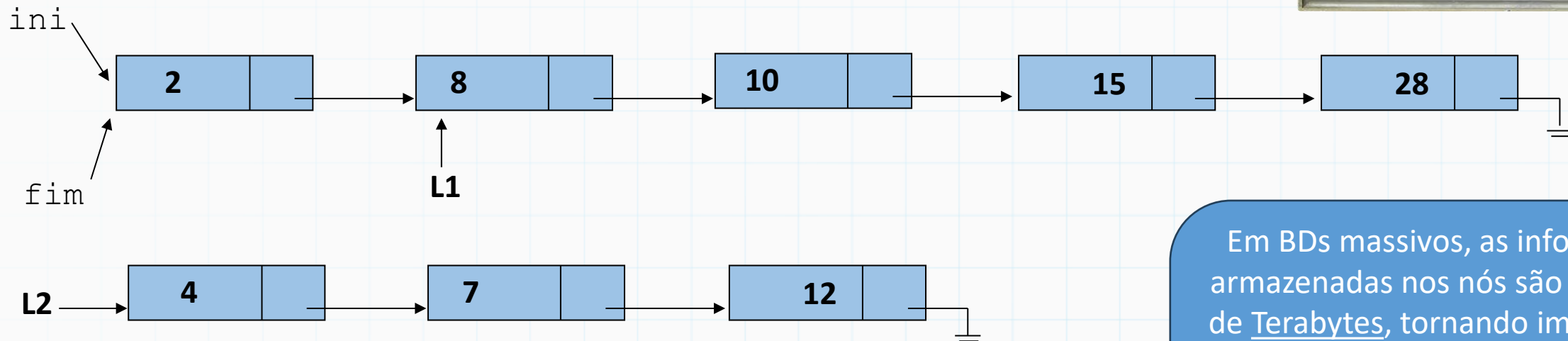
ini fim
↓ ↓

Use só o que aprendemos até hoje

Listas encadeadas- LAB



5) Dado duas listas ordenadas, escreva uma função que mescle (“merge”) as duas listas em uma nova lista ordenada, apenas fazendo redirecionamento de ponteiros (sem alocar novos nos), e sem utilizar/alocar estruturas auxiliares (vetores ou listas). Veja exemplo:



Em BDs massivos, as informações armazenadas nos nós são da ordem de Terabytes, tornando impraticável copiar informações entre os nós.

Então qualquer alteração feita na estrutura da lista deve ser feita apenas redirecionando ponteiros

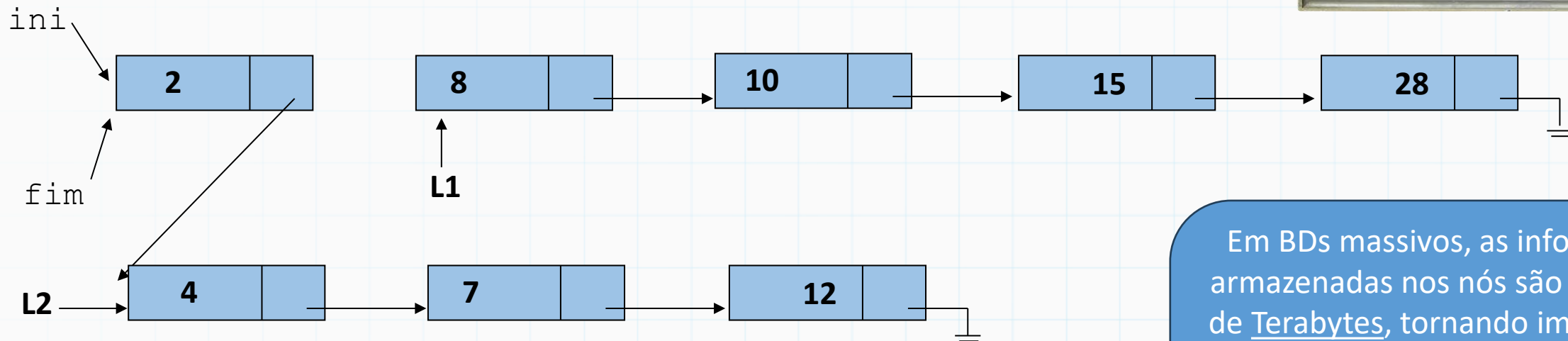
- Para o primeiro elemento vemos quem tem o menor valor (entre o primeiro elemento de L1 e o primeiro de L2), para atribuirmos inicialmente os ponteiros ini e fim para esse menor elemento.
- Depois avançarmos o ponteiro da lista (neste caso L1)

Use só o que aprendemos até hoje

Listas encadeadas- LAB



5) Dado duas listas ordenadas, escreva uma função que mescle (“merge”) as duas listas em uma nova lista ordenada, apenas fazendo redirecionamento de ponteiros (sem alocar novos nos), e sem utilizar/alocar estruturas auxiliares (vetores ou listas). Veja exemplo:



Em BDs massivos, as informações armazenadas nos nós são da ordem de Terabytes, tornando impraticável copiar informações entre os nós.

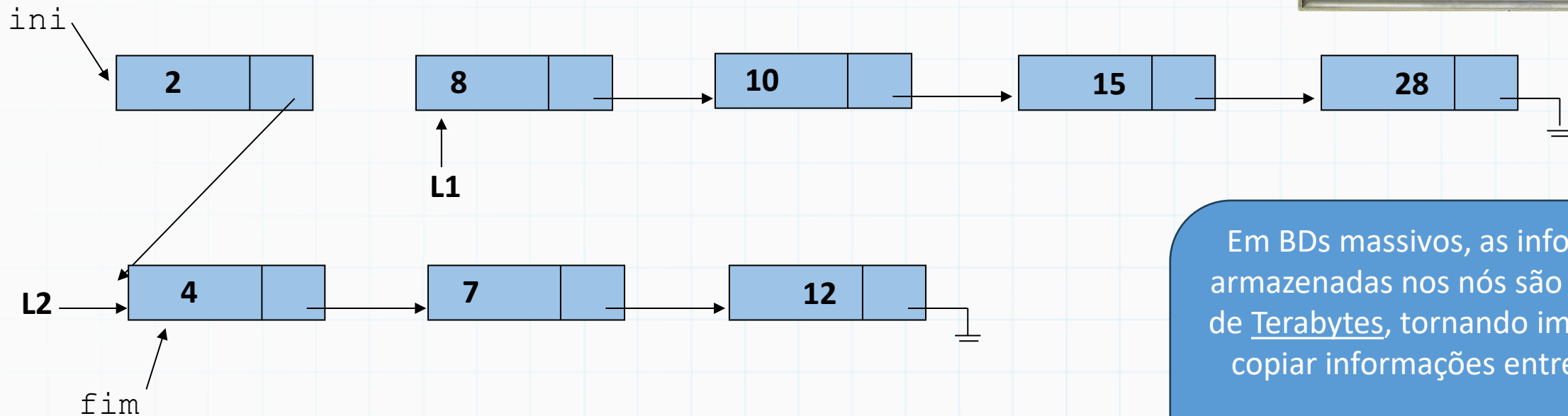
Então qualquer alteração feita na estrutura da lista deve ser feita apenas redirecionando ponteiros

- iterativamente, vamos comparar os valores que L1 aponta (que é 8) e que L2 aponta (que é 4) para saber quem é o menor
 - 4 é menor que 8, o próximo elemento será de L2
- vamos atualizar o próximo elemento da nova lista (o próximo depois do fim), logo, próximo do fim aponta para L2
 - que vai ter que ser o 4

Listas encadeadas- LAB



5) Dado duas listas ordenadas, escreva uma função que mescle (“merge”) as duas listas em uma nova lista ordenada, apenas fazendo redirecionamento de ponteiros (sem alocar novos nós), e sem utilizar/alocar estruturas auxiliares (vetores ou listas). Veja exemplo:



Em BDs massivos, as informações armazenadas nos nós são da ordem de Terabytes, tornando impraticável copiar informações entre os nós.

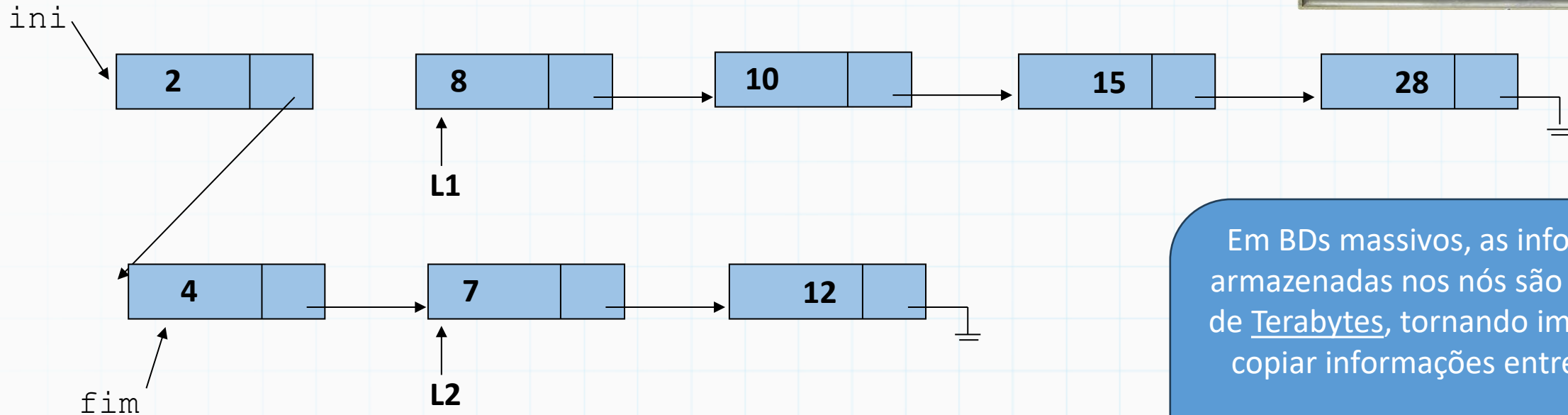
Então qualquer alteração feita na estrutura da lista deve ser feita apenas redirecionando ponteiros

- atualizamos o fim agora para apontar para o novo fim da nova lista, logo, fim vai apontar para L2

Listas encadeadas- LAB



5) Dado duas listas ordenadas, escreva uma função que mescle (“merge”) as duas listas em uma nova lista ordenada, apenas fazendo redirecionamento de ponteiros (sem alocar novos nós), e sem utilizar/alocar estruturas auxiliares (vetores ou listas). Veja exemplo:



Em BDs massivos, as informações armazenadas nos nós são da ordem de Terabytes, tornando impraticável copiar informações entre os nós.

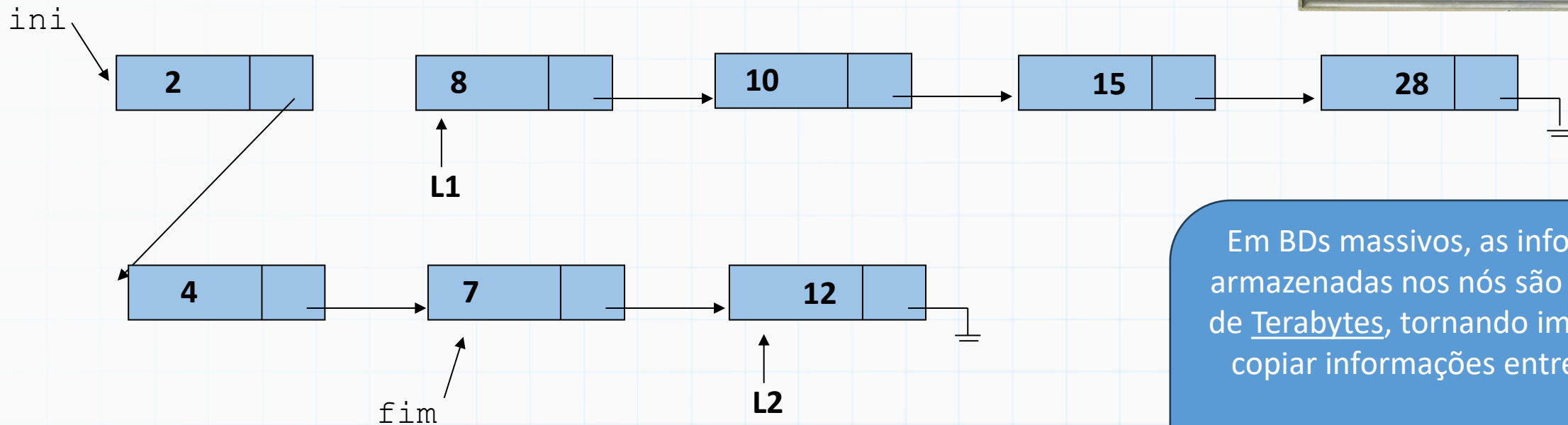
Então qualquer alteração feita na estrutura da lista deve ser feita apenas redirecionando ponteiros

- O ponteiro da lista escolhida (L2) agora passa para o próximo elemento
 - **L2 passa para o próximo**
- O processo agora se repete, veja que novamente o valor apontado por L2 é menor que o apontado por L1

Listas encadeadas- LAB



5) Dado duas listas ordenadas, escreva uma função que mescle (“merge”) as duas listas em uma nova lista ordenada, apenas fazendo redirecionamento de ponteiros (sem alocar novos nos), e sem utilizar/alocar estruturas auxiliares (vetores ou listas). Veja exemplo:



Em BDs massivos, as informações armazenadas nos nós são da ordem de Terabytes, tornando impraticável copiar informações entre os nós.

Então qualquer alteração feita na estrutura da lista deve ser feita apenas redirecionando ponteiros

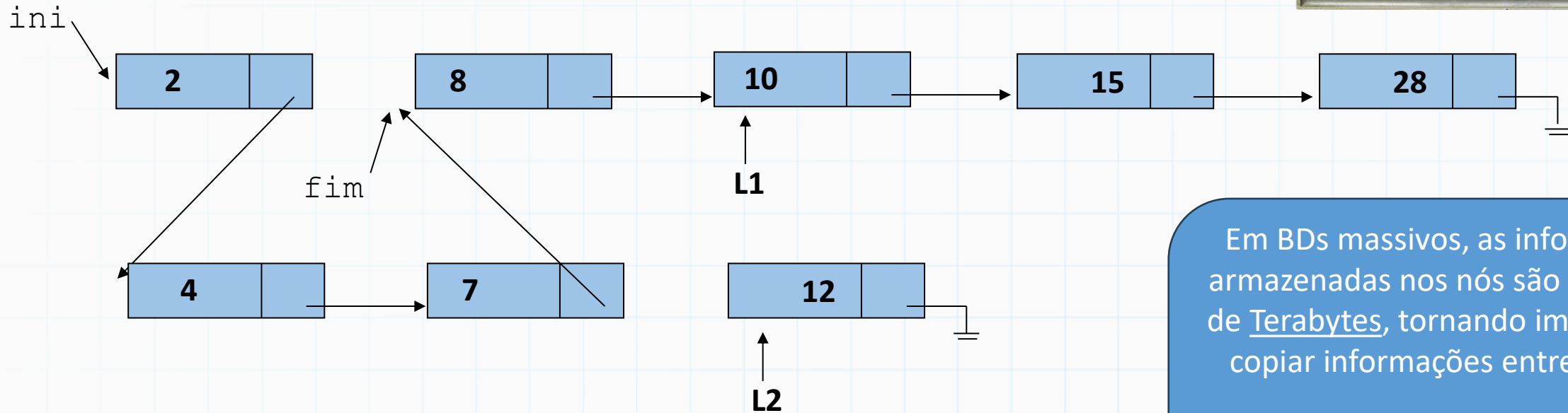
- Dado mais uma iteração teremos esse cenário.

- Veja que agora, o valor apontado por L1 é menor, logo teremos:

Listas encadeadas- LAB



5) Dado duas listas ordenadas, escreva uma função que mescle (“merge”) as duas listas em uma nova lista ordenada, apenas fazendo redirecionamento de ponteiros (sem alocar novos nos), e sem utilizar/alocar estruturas auxiliares (vetores ou listas). Veja exemplo:



Em BDs massivos, as informações armazenadas nos nós são da ordem de Terabytes, tornando impraticável copiar informações entre os nós.

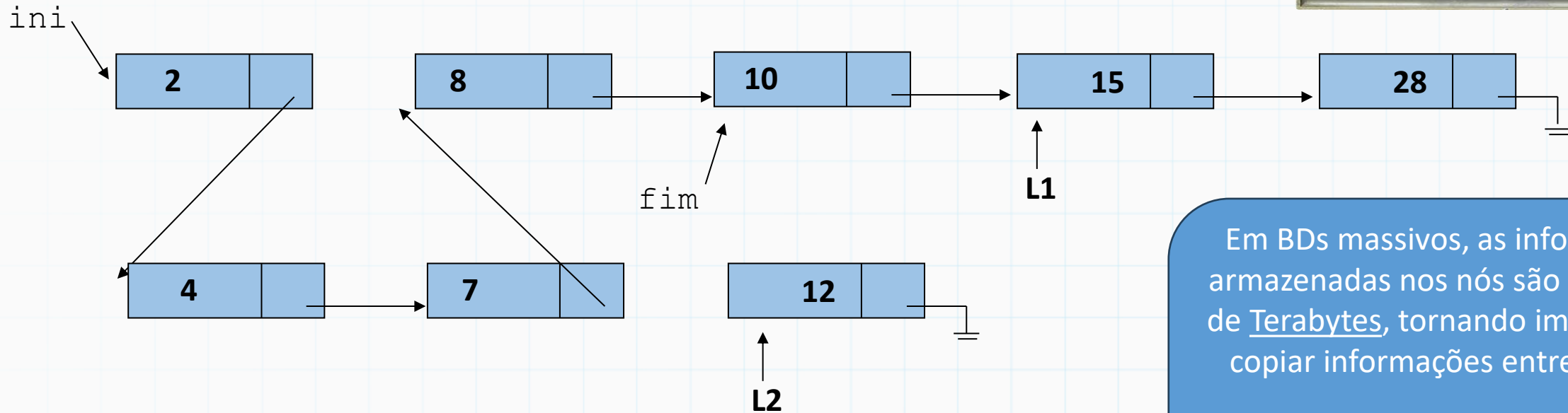
Então qualquer alteração feita na estrutura da lista deve ser feita apenas redirecionando ponteiros

- O processo continua enquanto tiver elementos nas duas listas

Listas encadeadas- LAB



5) Dado duas listas ordenadas, escreva uma função que mescle (“merge”) as duas listas em uma nova lista ordenada, apenas fazendo redirecionamento de ponteiros (sem alocar novos nos), e sem utilizar/alocar estruturas auxiliares (vetores ou listas). Veja exemplo:



Em BDs massivos, as informações armazenadas nos nós são da ordem de Terabytes, tornando impraticável copiar informações entre os nós.

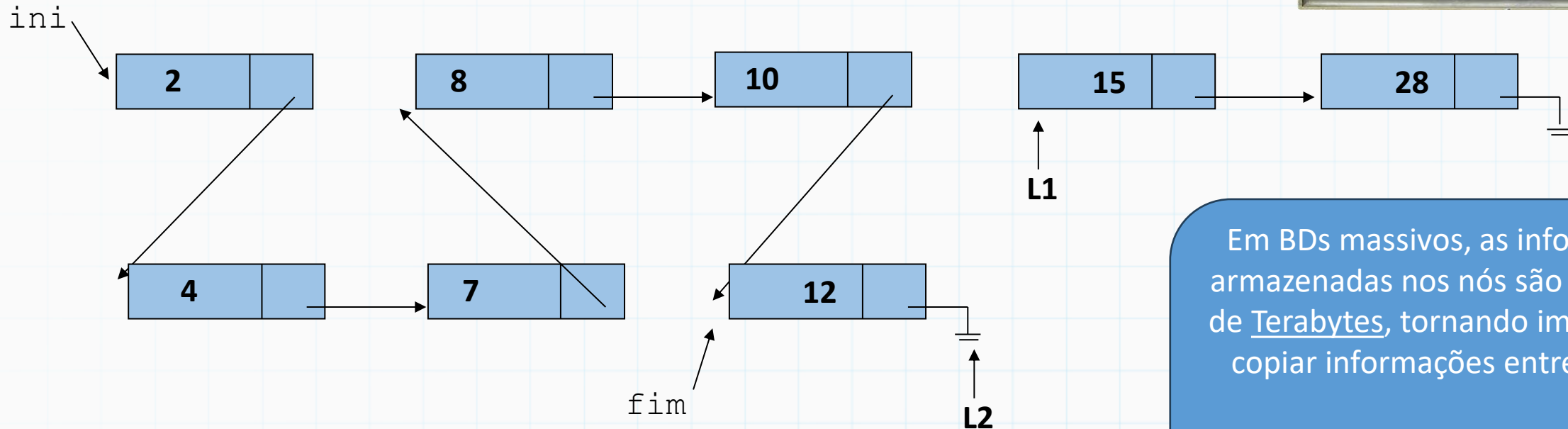
Então qualquer alteração feita na estrutura da lista deve ser feita apenas redirecionando ponteiros

- O processo continua enquanto tiver elementos nas duas listas

Listas encadeadas- LAB



5) Dado duas listas ordenadas, escreva uma função que mescle (“merge”) as duas listas em uma nova lista ordenada, apenas fazendo redirecionamento de ponteiros (sem alocar novos nos), e sem utilizar/alocar estruturas auxiliares (vetores ou listas). Veja exemplo:



Em BDs massivos, as informações armazenadas nos nós são da ordem de Terabytes, tornando impraticável copiar informações entre os nós.

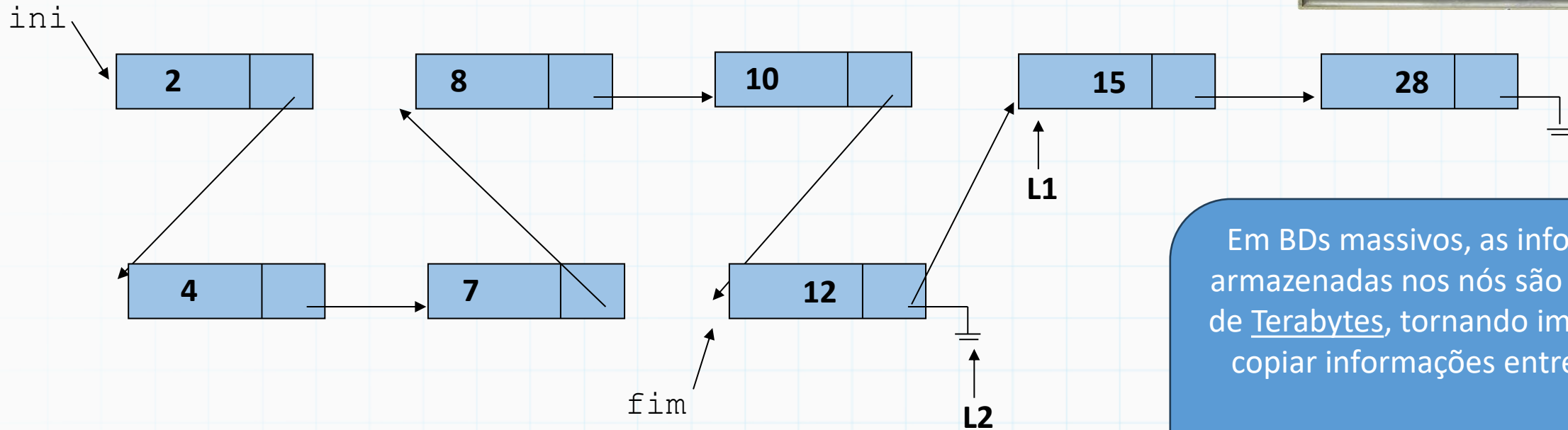
Então qualquer alteração feita na estrutura da lista deve ser feita apenas redirecionando ponteiros

- Opa, uma das listas chegou no fim, vamos parar.
- **Porem ainda falta fazer algo !**

Listas encadeadas- LAB



5) Dado duas listas ordenadas, escreva uma função que mescle (“merge”) as duas listas em uma nova lista ordenada, apenas fazendo redirecionamento de ponteiros (sem alocar novos nos), e sem utilizar/alocar estruturas auxiliares (vetores ou listas). Veja exemplo:



Em BDs massivos, as informações armazenadas nos nós são da ordem de Terabytes, tornando impraticável copiar informações entre os nós.

Então qualquer alteração feita na estrutura da lista deve ser feita apenas redirecionando ponteiros

- o próximo do fim tem que apontar para o início do “restinho” da lista que não acabou (nesse caso, o próximo do fim aponta para L1)

- Pronto, podemos agora retornar o ponteiro ini como a nova lista

Veja o exemplo de execução a seguir:



Listas encadeadas- LAB



5) Dado duas listas ordenadas, escreva uma função que mescle (“merge”) as duas listas em uma nova lista ordenada, apenas fazendo redirecionamento de ponteiros (sem alocar novos nos), e sem utilizar/alocar estruturas auxiliares (vetores ou listas). Veja exemplo:

Código da main.c

```
int main()
{
    int k;

    lista *L1 = NULL;
    L1 = insere_lista(L1, 28);
    L1 = insere_lista(L1, 15);
    L1 = insere_lista(L1, 10);
    L1 = insere_lista(L1, 8);
    L1 = insere_lista(L1, 2);
    implime_lista(L1);

    lista *L2 = NULL;
    L2 = insere_lista(L2, 14);
    L2 = insere_lista(L2, 12);
    L2 = insere_lista(L2, 7);
    implime_lista(L2);

    lista *L3 = mescla_listas(L1,L2);
    implime_lista(L3);

    L3 = exclui_lista (L3);
    return 0;
}
```

Exemplo de execução:

```
L = 2, 8, 10, 15, 28
L = 7, 12, 14,
L = 2, 7, 8, 10, 12, 14, 15, 28
```

Em BDs massivos, as informações armazenadas nos nós são da ordem de Terabytes, tornando impraticável copiar informações entre os nós.

Então qualquer alteração feita na estrutura da lista deve ser feita apenas redirecionando ponteiros

Use só o que aprendemos até hoje

Listas encadeadas- LAB

copia e cola



```
int main()
{
    lista *L1 = NULL;
    lista *L2 = NULL;
    lista *L3 = NULL;

    L1 = insere_lista(L1, 19); L1 = insere_lista(L1, 16);
    L1 = insere_lista(L1, 14); L1 = insere_lista(L1, 14);
    L1 = insere_lista(L1, 11); L1 = insere_lista(L1, 9);
    L1 = insere_lista(L1, 7); L1 = insere_lista(L1, 7);
    L1 = insere_lista(L1, 5); L1 = insere_lista(L1, 2);

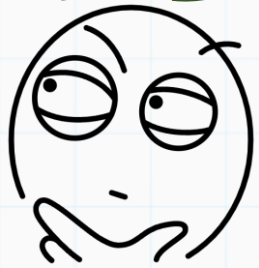
    L2 = insere_lista(L2, 20); L2 = insere_lista(L2, 18);
    L2 = insere_lista(L2, 17); L2 = insere_lista(L2, 15);
    L2 = insere_lista(L2, 13); L2 = insere_lista(L2, 13);
    L2 = insere_lista(L2, 10); L2 = insere_lista(L2, 6);
    L2 = insere_lista(L2, 4); L2 = insere_lista(L2, 3);

    L3 = mescla_listas(L1, L2); L3 = remove_duplicados(L3);
    L3 = remove_elemento(L3, 7); L3 = insere_lista(L3, 12);
    L3 = remove_elemento(L3, 14); L3 = insere_lista(L3, 8);
    L3 = remove_elemento(L3, 3); L3 = insere_lista(L3, 5);
    imprime_lista(L3);

    int valor_final = 1*(L3->info) + 2*(L3->prox->info) + 3*(L3->prox->prox->info);
    valor_final = valor_final + 4*(L3->prox->prox->prox->info) + 5*(L3->prox->prox->prox->prox->info);
    printf("valor final = %d\n", valor_final);

    L3 = exclui_lista (L3);
    return 0;
}
```

Agora que temos o mesclar, dado que as funções anteriores foram implementadas, se rodarmos o código ao lado, qual seria o valor impresso final ?



Listas encadeadas- LAB



6) Dado uma lista e um inteiro $k > 0$, escreva uma função que rotacione a lista a direita k vezes, apenas fazendo redirecionamento de ponteiros, sem utilizar/alocar estruturas auxiliares (vetores ou listas).

Código da main.c

```
int main()
{
    int k;

    lista *L = NULL;
    L = insere_lista(L, 5);
    L = insere_lista(L, 4);
    L = insere_lista(L, 3);
    L = insere_lista(L, 2);
    L = insere_lista(L, 1);
    imprime_lista(L);

    L = rotaciona_lista(L, 1);
    imprime_lista(L);
    L = rotaciona_lista(L, 3);
    imprime_lista(L);
    L = rotaciona_lista(L, 5);
    imprime_lista(L);
    L = rotaciona_lista(L, 6);
    imprime_lista(L);

    L = exclui_lista(L);
    return 0;
}
```

Exemplo de execução:

```
L = 1, 2, 3, 4, 5,
L = 5, 1, 2, 3, 4,
L = 2, 3, 4, 5, 1,
L = 2, 3, 4, 5, 1,
k invalido
L = 2, 3, 4, 5, 1,
```

Em BDs massivos, as informações armazenadas nos nós são da ordem de Terabytes, tornando impraticável copiar informações entre os nós.

Então qualquer alteração feita na estrutura da lista deve ser feita apenas redirecionando ponteiros

Use só o que aprendemos até hoje

Listas encadeadas- LAB

copia e cola



```
int main()
{
    lista *L1 = NULL;
    lista *L2 = NULL;
    lista *L3 = NULL;

    L1 = insere_lista(L1, 20); L1 = insere_lista(L1, 18);
    L1 = insere_lista(L1, 16); L1 = insere_lista(L1, 14);
    L1 = insere_lista(L1, 14); L1 = insere_lista(L1, 12);
    L1 = insere_lista(L1, 10); L1 = insere_lista(L1, 7);
    L1 = insere_lista(L1, 7); L1 = insere_lista(L1, 4);

    L2 = insere_lista(L2, 21); L2 = insere_lista(L2, 19);
    L2 = insere_lista(L2, 17); L2 = insere_lista(L2, 15);
    L2 = insere_lista(L2, 13); L2 = insere_lista(L2, 11);
    L2 = insere_lista(L2, 9); L2 = insere_lista(L2, 9);
    L2 = insere_lista(L2, 6); L2 = insere_lista(L2, 3);

    L3 = mescla_listas(L1, L2); L3 = remove_duplicados(L3);
    L3 = remove_elemento(L3, 14); L3 = insere_lista(L3, 8);
    L3 = remove_elemento(L3, 3); L3 = insere_lista(L3, 5);
    L3 = rotaciona_lista(L3, 4); L3 = remove_elemento(L3, 18);
    L3 = insere_lista(L3, 12); imprime_lista(L3);

    int valor_final = 1*(L3->info) + 2*(L3->prox->info) + 3*(L3->prox->prox->info);
    valor_final = valor_final+ 4*(L3->prox->prox->prox->info) + 5*(L3->prox->prox->prox->prox->info);
    printf("valor final = %d\n",valor_final);

    L3 = exclui_lista (L3);
    return 0;
}
```

Agora que temos o rotacionar, dado que as funções anteriores foram implementadas, se rodarmos o código ao lado, qual seria o valor impresso final ?



Listas encadeadas- LAB



7) Vamos refazer o exercício 3. Escreva uma função que remova duplicações em uma lista ordenada sem utilizar/alocar estruturas auxiliares (vetores ou listas) **E USANDO APENAS UM LAÇO !!!**

Código da main.c

```
int main()
{
    lista *L = NULL;

    L = insere_lista(L, 33);
    L = insere_lista(L, 33);
    L = insere_lista(L, 33);
    L = insere_lista(L, 33);
    L = insere_lista(L, 28);
    L = insere_lista(L, 28);
    L = insere_lista(L, 18);
    L = insere_lista(L, 2);
    L = insere_lista(L, 2);
    L = insere_lista(L, 2);
    imprime_lista(L);

    L = remove_duplicados (L);
    imprime_lista(L);

    L = exclui_lista (L);
    return 0;
}
```

Exemplo de execução:

```
L = 2, 2, 2, 18, 28, 28, 33, 33, 33, 33,
sem duplicacao
L = 2, 18, 28, 33,
```

Use só o que
aprendemos
até hoje



Listas encadeadas- LAB

DESAFIO

8) Escreva uma função que receba uma lista onde seus elementos são dígitos, e diga se a lista é um palíndromo, mas você não pode usar vetores !!!

Use só o que aprendemos até hoje

```
int main()
{
    lista *L = NULL;
    L = insere_lista(L, 5);
    L = insere_lista(L, 4);
    L = insere_lista(L, 3);
    L = insere_lista(L, 4);
    L = insere_lista(L, 5);
    imprime_lista(L);

    if (palindromo_lista(L) == 1)
        printf("e' palindromo\n");
    else
        printf("nao e' palindromo\n");

    L = exclui_lista (L);

    return 0;
}
```

L = 5, 4, 3, 4, 5,
e' palindromo

```
int main()
{
    lista *L = NULL;
    L = insere_lista(L, 5);
    L = insere_lista(L, 1);
    L = insere_lista(L, 2);
    L = insere_lista(L, 2);
    L = insere_lista(L, 1);
    imprime_lista(L);

    if (palindromo_lista(L) == 1)
        printf("e' palindromo\n");
    else
        printf("nao e' palindromo\n");

    L = exclui_lista (L);

    return 0;
}
```

L = 1, 2, 2, 1,
e' palindromo

DESAFIO

```
int main()
{
    lista *L = NULL;
    L = insere_lista(L, 9);
    L = insere_lista(L, 3);
    L = insere_lista(L, 5);
    L = insere_lista(L, 1);
    L = insere_lista(L, 9);
    imprime_lista(L);

    if (palindromo_lista(L) == 1)
        printf("e' palindromo\n");
    else
        printf("nao e' palindromo\n");
    L = exclui_lista (L);

    L = insere_lista(L, 9);
    L = insere_lista(L, 3);
    L = insere_lista(L, 3);
    L = insere_lista(L, 8);
    imprime_lista(L);

    if (palindromo_lista(L) == 1)
        printf("e' palindromo\n");
    else
        printf("nao e' palindromo\n");

    L = exclui_lista (L);
    return 0;
}
```

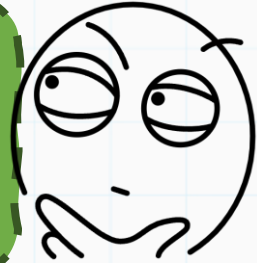
L = 9, 1, 5, 3, 9,
nao e' palindromo
L = 8, 3, 3, 9,
nao e' palindromo



Listas encadeadas- LAB



Dado que as funções anteriores foram implementadas, se rodarmos o código ao lado, qual seria o valor impresso final ?



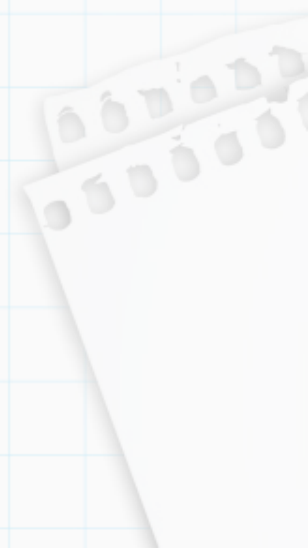
Agora que temos o mesclar, dado que as funções anteriores foram implementadas, se rodarmos o código ao lado, qual seria o valor impresso final ?



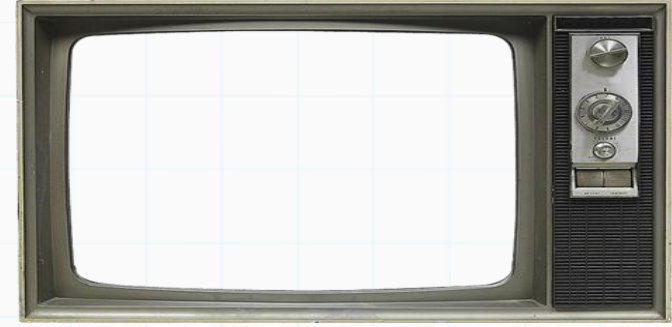
Agora que temos o rotacionar, dado que as funções anteriores foram implementadas, se rodarmos o código ao lado, qual seria o valor impresso final ?



Todos ganham



Até a próxima



Slides baseados no curso de Aline Nascimento