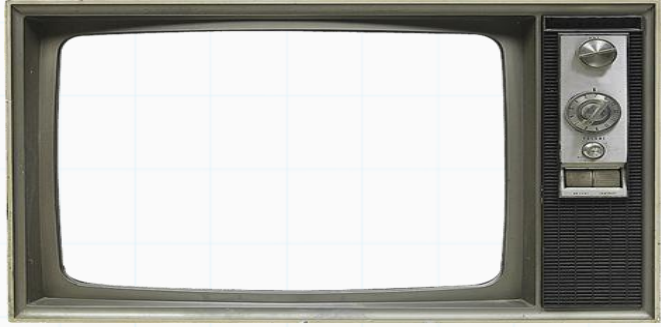
Programação Estruturada



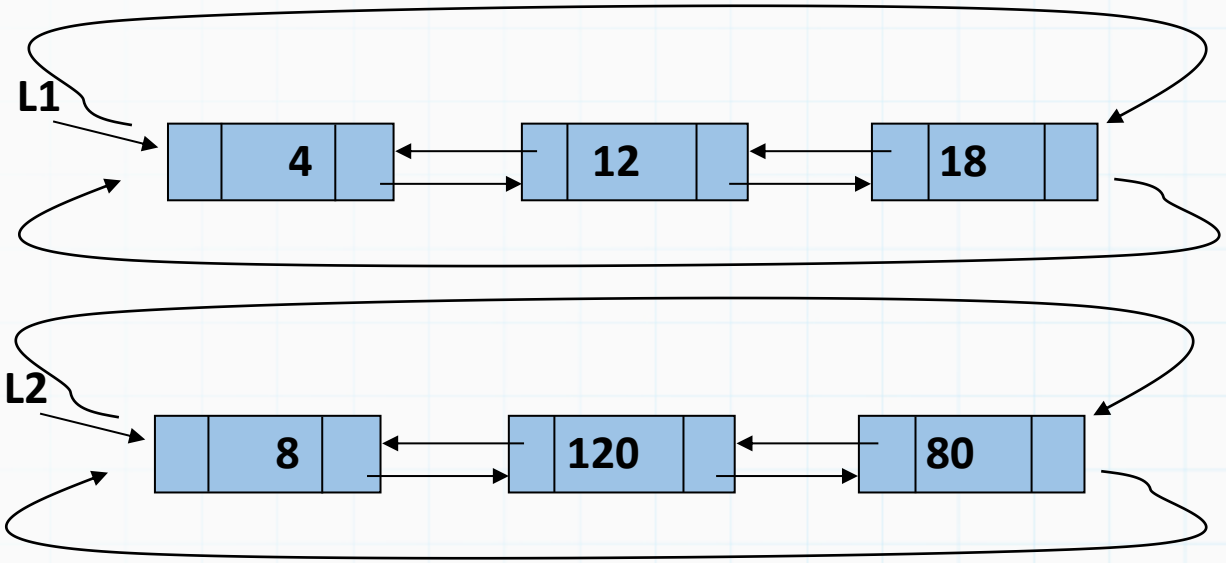
Professor : Yuri Frota

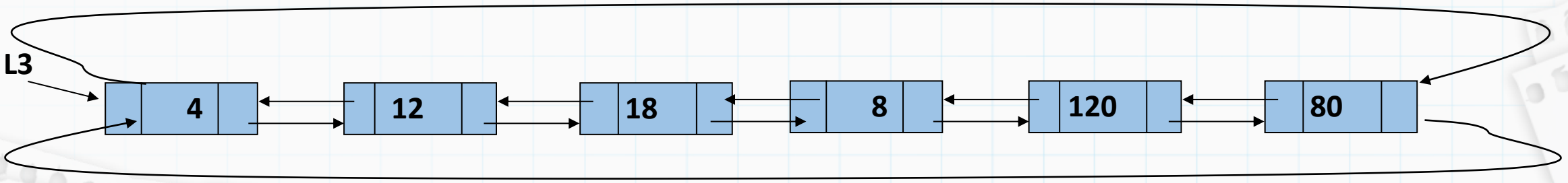
yuri@ic.uff.br

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

Listas circulares duplamente encadeadas - LAB

1) Dado duas listas circulares e duplamente encadeadas, escreva uma função para concatenar as duas listas em uma nova lista, apenas fazendo redirecionamento de ponteiros, sem alocar novos nos, ou utilizar/alocar estruturas auxiliares (vetores ou listas).





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

Veja exemplo da execução abaixo:

Use só o que aprendemos até hoje

Listas circulares duplamente encadeadas - LAB

1) Dado duas listas circulares e duplamente encadeadas, escreva uma função para concatenar as duas listas em uma nova lista, apenas fazendo redirecionamento de ponteiros, sem alocar novos nos, ou 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);
    implime_lista(L1);

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

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

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

Exemplo de execução:

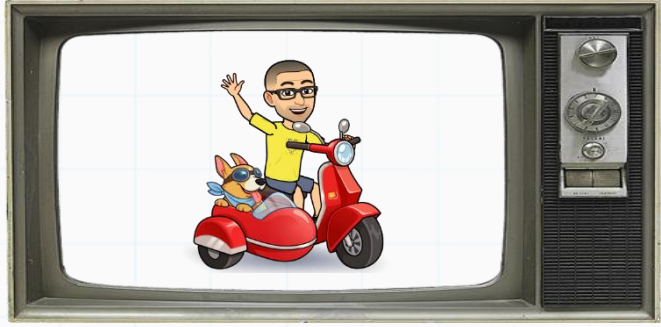
```
L = 2, 9, 6, 8,
L = 10, 4, 7,
L = 2, 9, 6, 8, 10, 4, 7,
```

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 circulares duplamente encadeadas - LAB



2) Dado uma lista circular e duplamente encadeada com n elementos, escreva uma função para remover um elemento na posição n-1>k>=0

main.c

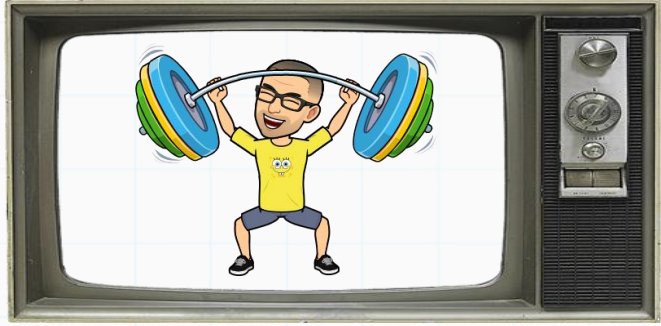
```
int main()
{
    lista *L1 = NULL;
    L1 = insere_lista_pos_DC(L1, 2, 0);
    L1 = insere_lista_pos_DC(L1, 6, 1);
    L1 = insere_lista_pos_DC(L1, 7, 2);
    L1 = insere_lista_pos_DC(L1, 23, 3);
    L1 = insere_lista_pos_DC(L1, 13, 4);
    imprime_lista_DC(L1);

    L1 = remove_lista_pos_DC(L1, 2);
    imprime_lista_DC(L1);
    L1 = remove_lista_pos_DC(L1, 4);
    imprime_lista_DC(L1);
    L1 = remove_lista_pos_DC(L1, 3);
    imprime_lista_DC(L1);
    L1 = remove_lista_pos_DC(L1, 0);
    imprime_lista_DC(L1);
    L1 = remove_lista_pos_DC(L1, 0);
    imprime_lista_DC(L1);
    L1 = remove_lista_pos_DC(L1, 0);
    imprime_lista_DC(L1);

    exclui_lista_DC(L1);
    return 0;
}
```

Execução

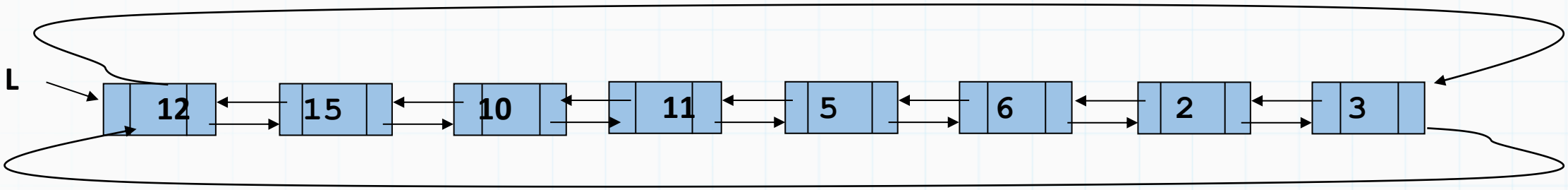
```
L = 2, 6, 7, 23, 13,
L = 2, 6, 23, 13,
Posicao invalida
L = 2, 6, 23, 13,
L = 2, 6, 23,
L = 6, 23,
L = 23,
L = vazio
```

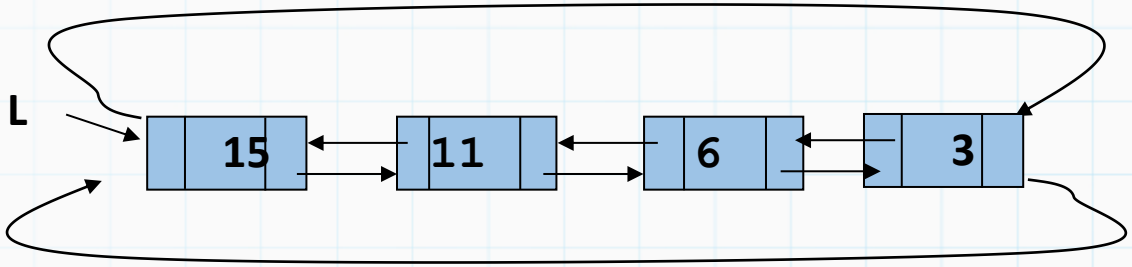
Use só o que aprendemos até hoje

Listas circulares duplamente encadeadas - LAB



3) Dado uma lista circular e duplamente encadeada com n elementos, escreva uma função para remover os elementos da lista em que o próximo elemento a ele é maior que ele, sem utilizar/alocar estruturas auxiliares (vetores ou listas).

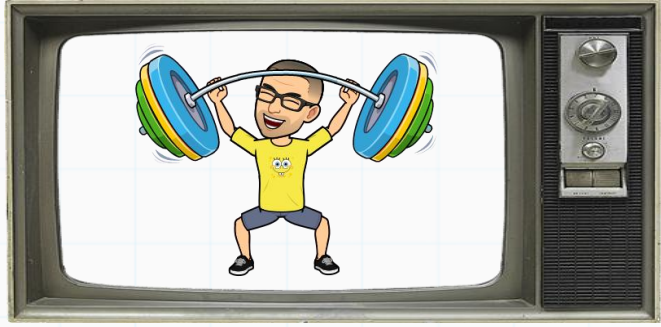




Veja exemplo de execução a seguir

Use só o que aprendemos até hoje

Listas circulares duplamente encadeadas - LAB



3) Dado uma lista circular e duplamente encadeada com n elementos, escreva uma função para remover os elementos da lista em que o próximo elemento a ele é maior que ele, sem utilizar/alocar estruturas auxiliares (vetores ou listas).

main.c

```
int main()
{
    lista *L1 = NULL;
    L1 = insere_lista_pos_DC(L1, 12, 0);
    L1 = insere_lista_pos_DC(L1, 15, 1);
    L1 = insere_lista_pos_DC(L1, 10, 2);
    L1 = insere_lista_pos_DC(L1, 11, 3);
    L1 = insere_lista_pos_DC(L1, 5, 4);
    L1 = insere_lista_pos_DC(L1, 6, 5);
    L1 = insere_lista_pos_DC(L1, 2, 6);
    L1 = insere_lista_pos_DC(L1, 3, 7);
    imprime_lista_DC(L1);

    L1 = remove_proximos_maiores(L1);
    imprime_lista_DC(L1);

    exclui_lista_DC(L1);
    return 0;
}
```

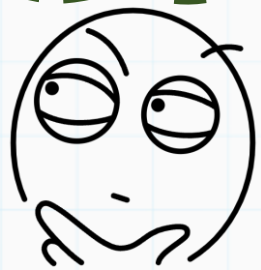
Execução

```
L = 12, 15, 10, 11, 5, 6, 2, 3,
remove onde proximos maiores
   12 menor que 15, remove
   10 menor que 11, remove
   5 menor que 6, remove
   2 menor que 3, remove
L = 15, 11, 6, 3,
```

Listas circulares duplamente encadeadas - LAB

copia e cola

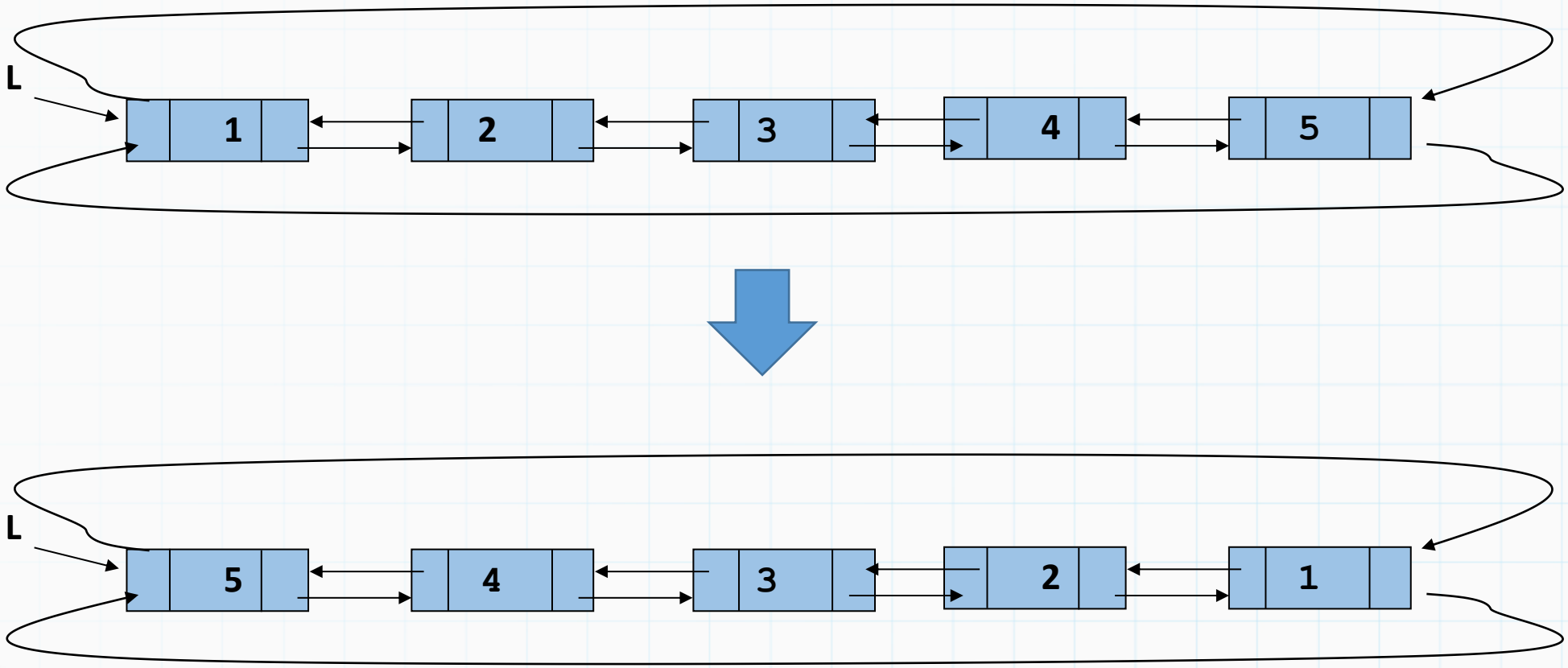
Dado que as funções anteriores foram implementadas, se rodarmos o código ao lado, qual seria o valor impresso final ?



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

    /* Geração de L1 */
    L1 = insere_lista_pos_DC(L1, 8, 0);  L1 = insere_lista_pos_DC(L1, 3, 1);
    L1 = insere_lista_pos_DC(L1, 11, 2); L1 = insere_lista_pos_DC(L1, 6, 3);
    L1 = insere_lista_pos_DC(L1, 14, 4); L1 = insere_lista_pos_DC(L1, 5, 5);

    /* Geração de L2 */
    L2 = insere_lista_pos_DC(L2, 9, 0);  L2 = insere_lista_pos_DC(L2, 2, 1);
    L2 = insere_lista_pos_DC(L2, 13, 2); L2 = insere_lista_pos_DC(L2, 7, 3);
    L2 = insere_lista_pos_DC(L2, 10, 4);

     /* Sequência de operações */
    L1 = insere_lista_pos_DC(L1, 12, 2); L2 = remove_lista_pos_DC(L2, 1);
    L1 = concatena_DC(L1, L2);           L1 = remove_proximos_maiores(L1);
    L1 = insere_lista_pos_DC(L1, 4, 3);  L1 = remove_lista_pos_DC(L1, 5);
    L1 = insere_lista_pos_DC(L1, 15, 0); L1 = remove_proximos_maiores(L1);
    imprime_lista_DC(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_DC(L1);
    return 0;
}
```

Listas circulares duplamente encadeadas - LAB

4) Dado uma lista circular e duplamente encadeada escreva uma função para inverter os elementos da lista, apenas fazendo reatribuição de ponteiros.



Exemplo de execução a seguir:

Use só o que aprendemos até hoje

Listas circulares duplamente encadeadas - LAB

4) Dado uma lista circular e duplamente encadeada escreva uma função para inverter os elementos da lista, apenas fazendo reatribuição de ponteiros.

main.c

```
int main()
{
    lista *L1 = NULL;
    L1 = insere_lista_pos_DC(L1, 1, 0);
    L1 = insere_lista_pos_DC(L1, 2, 1);
    L1 = insere_lista_pos_DC(L1, 3, 2);
    L1 = insere_lista_pos_DC(L1, 4, 3);
    L1 = insere_lista_pos_DC(L1, 5, 4);
    imprime_lista_DC(L1);

    printf("inverte\n");
    L1 = inverte_lista_DC(L1);
    imprime_lista_DC(L1);

    exclui_lista_DC(L1);
    return 0;
}
```

Execução

```
L = 1, 2, 3, 4, 5,
inverte
L = 5, 4, 3, 2, 1,
```

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

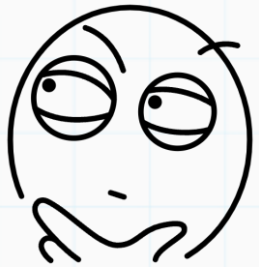
Então qualquer alteração feita na estrutura da lista deve ser feita apenas redirecionando ponteiros

Use só o que aprendemos até hoje

Listas circulares duplamente encadeadas - LAB

copia e cola

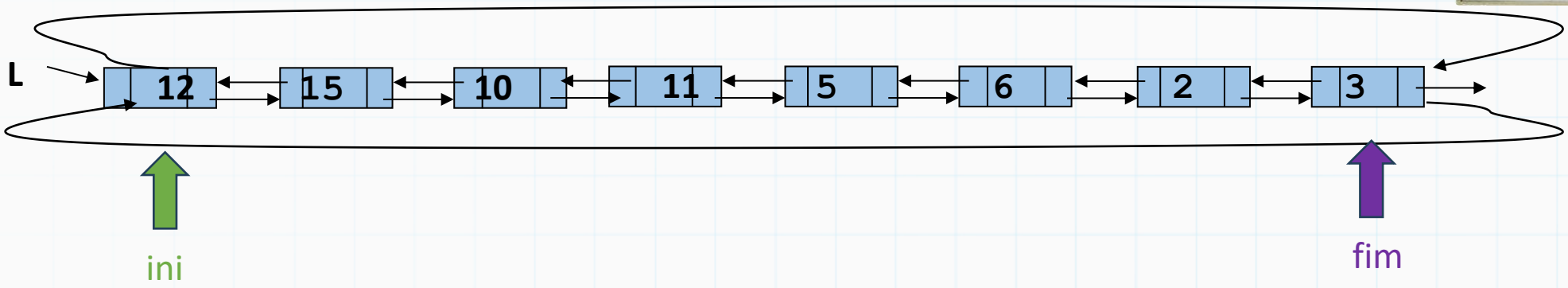
Agora com a inversão, se rodarmos o código ao lado, qual seria o valor impresso final ?



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

    /* Geração de L1 */
    L1 = insere_lista_pos_DC(L1, 8, 0);  L1 = insere_lista_pos_DC(L1, 3, 1);
    L1 = insere_lista_pos_DC(L1, 11, 2); L1 = insere_lista_pos_DC(L1, 6, 3);
    L1 = insere_lista_pos_DC(L1, 14, 4); L1 = insere_lista_pos_DC(L1, 5, 5);

    /* Geração de L2 */
    L2 = insere_lista_pos_DC(L2, 9, 0);  L2 = insere_lista_pos_DC(L2, 2, 1);
    L2 = insere_lista_pos_DC(L2, 13, 2); L2 = insere_lista_pos_DC(L2, 7, 3);
    L2 = insere_lista_pos_DC(L2, 10, 4);

     /* Sequência de operações */
    L1 = insere_lista_pos_DC(L1, 12, 2); L2 = remove_lista_pos_DC(L2, 1);
    L1 = concatena_DC(L1, L2);           L1 = remove_proximos_maiores(L1);
    L1 = insere_lista_pos_DC(L1, 4, 3);  L1 = remove_lista_pos_DC(L1, 5);
    L1 = insere_lista_pos_DC(L1, 15, 0); L1 = remove_proximos_maiores(L1);
    imprime_lista_DC(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_DC(L1);
    return 0;
}
```

Listas circulares duplamente encadeadas - LAB

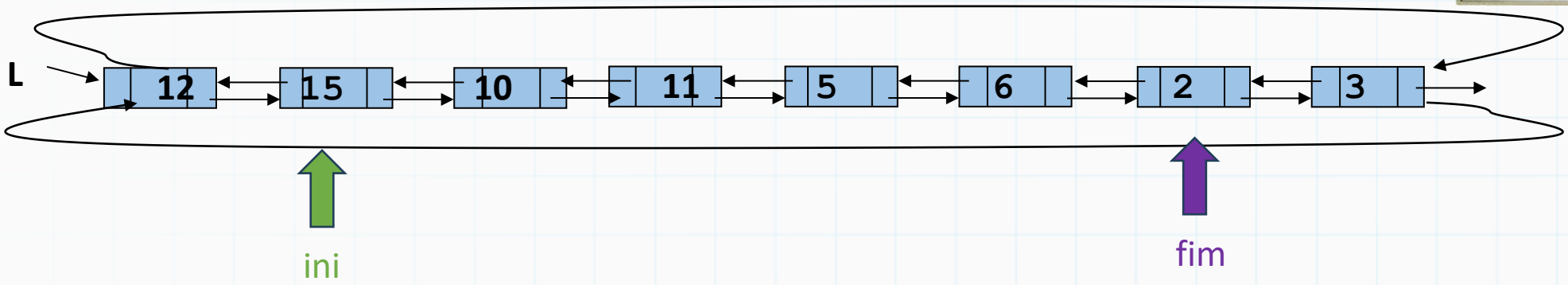
5) Dado uma lista circular e duplamente encadeada escreva uma função para colocar todos os elementos pares antes dos impares, sem alocar nos e sem utilizar/alocar estruturas auxiliares (vetores ou listas), da seguinte forma:



- Faça isso usando 2 ponteiros (ini e fim)

Listas circulares duplamente encadeadas - LAB

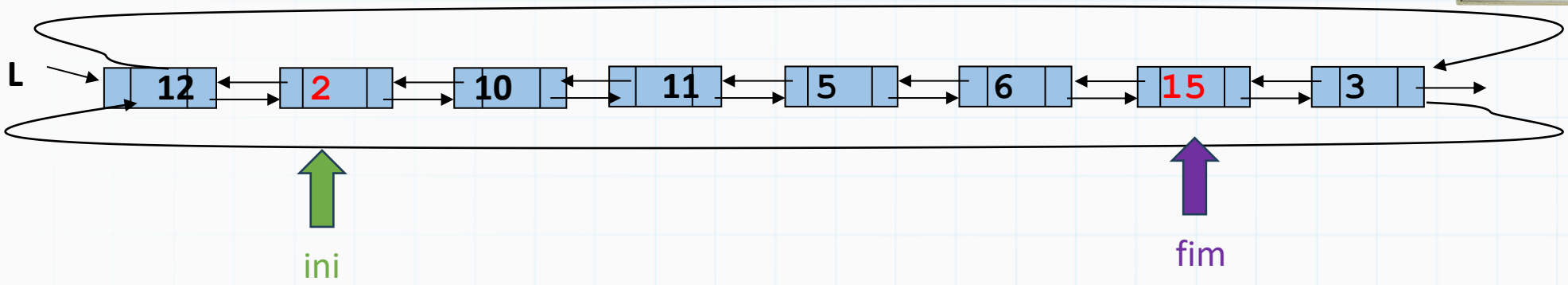
5)) Dado uma lista circular e duplamente encadeada escreva uma função para colocar todos os elementos pares antes dos impares, sem alocar nos e sem utilizar/alocar estruturas auxiliares (vetores ou listas), da seguinte forma:



- Faça isso usando 2 ponteiros (ini e fim)
- ini anda para frente ate encontrar um impar
- fim anda para tras ate encontrar um par

Listas circulares duplamente encadeadas - LAB

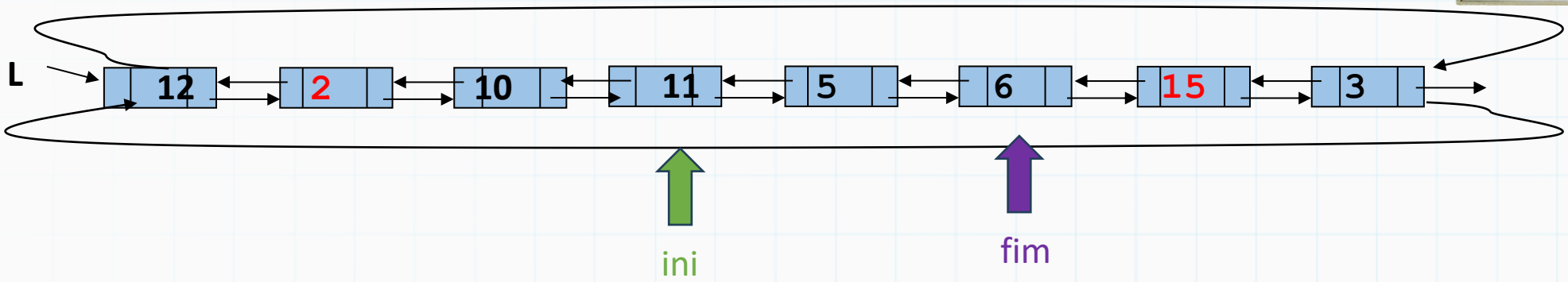
5)) Dado uma lista circular e duplamente encadeada escreva uma função para colocar todos os elementos pares antes dos impares, sem alocar nos e sem utilizar/alocar estruturas auxiliares (vetores ou listas), da seguinte forma:



- Faça isso usando 2 ponteiros (ini e fim)
- ini anda para frente ate encontrar um impar
- fim anda para tras ate encontrar um par
- Ai troca as infos, e volta a andar

Listas circulares duplamente encadeadas - LAB

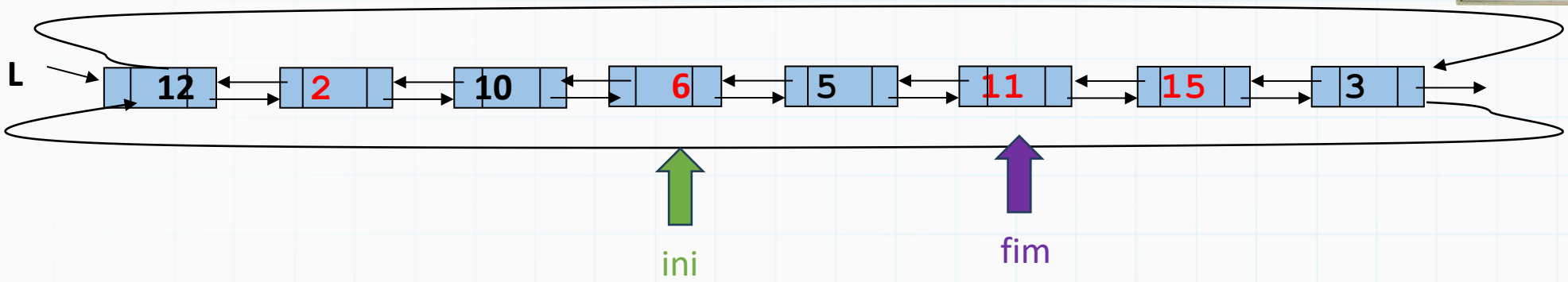
5)) Dado uma lista circular e duplamente encadeada escreva uma função para colocar todos os elementos pares antes dos impares, sem alocar nos e sem utilizar/alocar estruturas auxiliares (vetores ou listas), da seguinte forma:



- Faça isso usando 2 ponteiros (ini e fim)
- ini anda para frente ate encontrar um impar
- fim anda para tras ate encontrar um par
- Ai troca as infos, e volta a andar

Listas circulares duplamente encadeadas - LAB

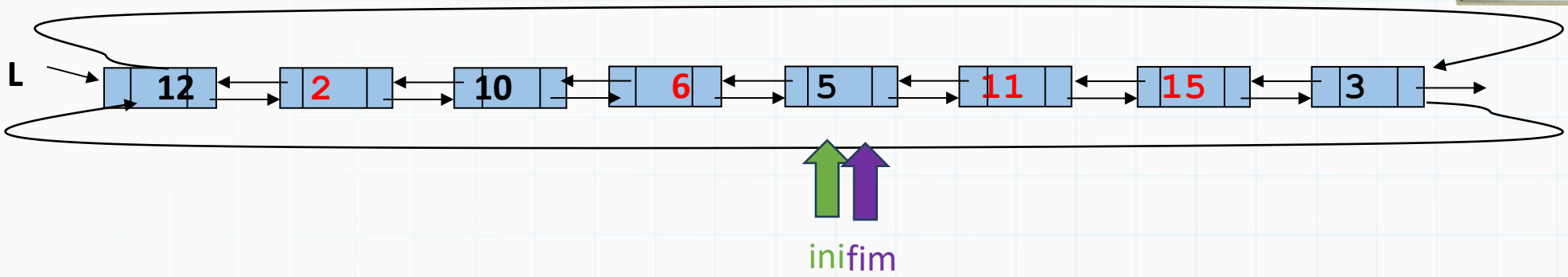
5) Dado uma lista circular e duplamente encadeada escreva uma função para colocar todos os elementos pares antes dos impares, sem utilizar/alocar estruturas auxiliares (vetores ou listas), da seguinte forma:



- Faça isso usando 2 ponteiros (ini e fim)
- ini anda para frente ate encontrar um impar
- fim anda para tras ate encontrar um par
- Ai troca as infos, e volta a andar

Listas circulares duplamente encadeadas - LAB

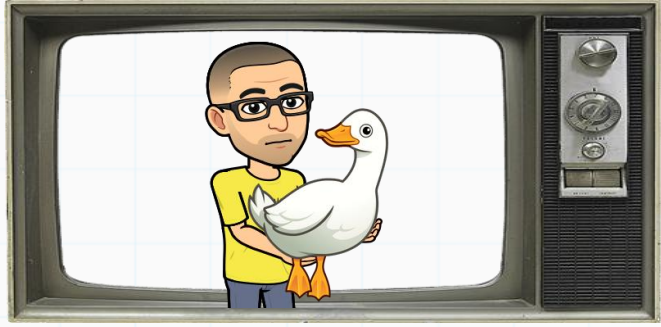
5)) Dado uma lista circular e duplamente encadeada escreva uma função para colocar todos os elementos pares antes dos impares, sem alocar nos e sem utilizar/alocar estruturas auxiliares (vetores ou listas), da seguinte forma:



- Faça isso usando 2 ponteiros (ini e fim)
- ini anda para frente ate encontrar um impar
- fim anda para tras ate encontrar um par
- Ai troca as infos, e volta a andar
- O método para quando os ponteiros se encontram (ou passam um pelo outro)

... Veja exemplo da execução :

Listas circulares duplamente encadeadas - LAB



5)) Dado uma lista circular e duplamente encadeada escreva uma função para colocar todos os elementos pares antes dos impares, sem alocar nos e sem utilizar/alocar estruturas auxiliares (vetores ou listas), da seguinte forma:

main.c

```
int main()
{
    lista *L1 = NULL;
    L1 = insere_lista_pos_DC(L1, 17, 0);
    L1 = insere_lista_pos_DC(L1, 15, 1);
    L1 = insere_lista_pos_DC(L1, 8, 2);
    L1 = insere_lista_pos_DC(L1, 12, 3);
    L1 = insere_lista_pos_DC(L1, 10, 4);
    L1 = insere_lista_pos_DC(L1, 5, 5);
    L1 = insere_lista_pos_DC(L1, 4, 6);
    L1 = insere_lista_pos_DC(L1, 1, 7);
    L1 = insere_lista_pos_DC(L1, 7, 8);
    L1 = insere_lista_pos_DC(L1, 6, 9);
    imprime_lista_DC(L1);

    printf("par impar\n");
    L1 = par_impar_DC(L1);
    imprime_lista_DC(L1);

    exclui_lista_DC(L1);
    return 0;
}
```

Execução

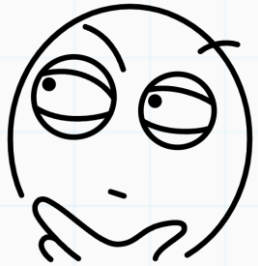
```
L = 17, 15, 8, 12, 10, 5, 4, 1, 7, 6,
par impar
L = 6, 4, 8, 12, 10, 5, 15, 1, 7, 17,
```

Use só o que aprendemos até hoje

Listas circulares duplamente encadeadas - LAB

copia e cola

Agora com a separação pares/impares , se rodarmos o código ao lado, qual seria o valor impresso final ?



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

    L1 = insere_lista_pos_DC(L1, 7, 0);  L1 = insere_lista_pos_DC(L1, 4, 1);
    L1 = insere_lista_pos_DC(L1, 12, 2); L1 = insere_lista_pos_DC(L1, 5, 3);
    L1 = insere_lista_pos_DC(L1, 10, 4); L2 = insere_lista_pos_DC(L2, 9, 0);
    L2 = insere_lista_pos_DC(L2, 14, 1); L2 = insere_lista_pos_DC(L2, 3, 2);
    L2 = insere_lista_pos_DC(L2, 8, 3);  L2 = insere_lista_pos_DC(L2, 11, 4);

    L1 = insere_lista_pos_DC(L1, 6, 2);  L2 = remove_lista_pos_DC(L2, 2);
    L1 = concatena_DC(L1, L2);           L1 = par_impar_DC(L1);
    L1 = remove_proximos_maiores(L1);    L1 = insere_lista_pos_DC(L1, 13, 1);
    L1 = remove_lista_pos_DC(L1, 4);     L1 = inverte_lista_DC(L1);
    imprime_lista_DC(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_DC(L1);
    return 0;
}
```

Listas circulares duplamente encadeadas - LAB

6) Implemente agora uma função que dado uma lista circular duplamente encadeada, ordene a lista pelo método da bolha apenas fazendo reatribuição de ponteiros (isto é, não pode mexer nas infos dos nos).


```
for (i=0; i<n; i++)
    for (j=0; j<n-1; j++)
        if (v[j] > v[j+1])
        {
            temp        = v[j];
            v[j]        = v[j+1];
            v[j+1]      = temp;
        }
```

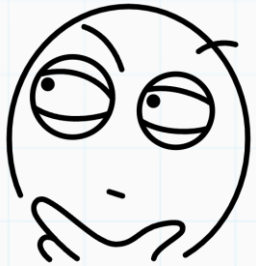
Em BDs massivos, as informações armazenadas nos nós são da ordem de Terabytes, tornando impraticável copiar informações entre os nós.

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

Use só o que aprendemos até hoje

Listas circulares duplamente encadeadas - LAB

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

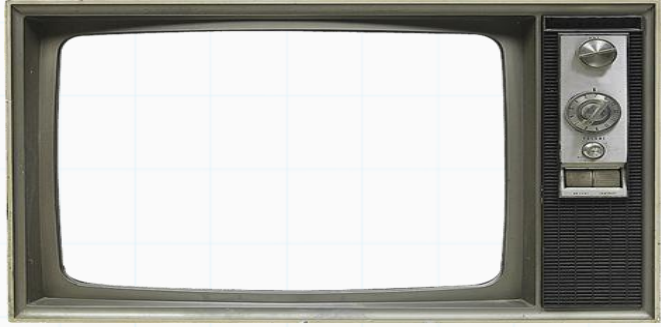


Agora com a inversão, se rodarmos o código ao lado, qual seria o valor impresso final ?

Agora com a separação pares/impares , se rodarmos o código ao lado, qual seria o valor impresso final ?

Todos ganham

Até a próxima



Slides baseados no curso de Aline Nascimento