Listas ordenadas
(IED-001)

Prof. Dr. Silvio do Lago Pereira

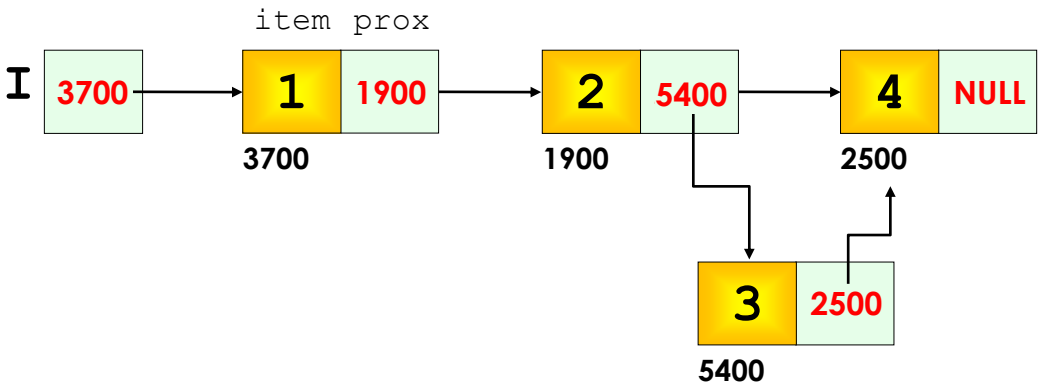
Departamento de Tecnologia da Informação

Faculdade de Tecnologia de São Paulo

Listas ordenadas

Lista ordenada

é uma lista encadeada cujos itens são sempre inseridos em ordem crescente.



Observações:

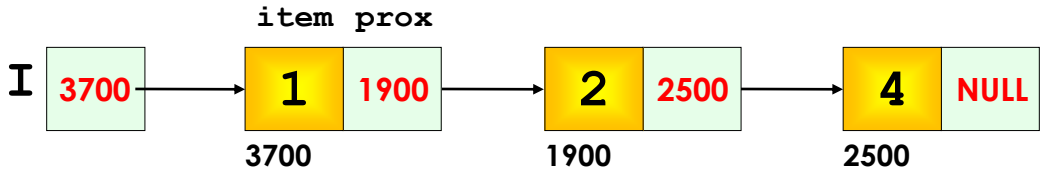
- Cada novo item inserido na lista entra na posição correta para manter a ordem.
- Nenhum item precisa ser movido na memória para que a ordem seja mantida.
- Todas as funções para listas podem ser usadas com listas ordenadas.

Lista ordenada é útil para guardar uma coleção dinâmicas em que a ordem é importante!

Listas ordenadas

Exemplo 1. O tipo `Lista` (recapitulação)

```
typedef int Item;
typedef struct no {
    Item item;
    struct no *prox;
} *Lista;
```



Observações:

- O tipo **`Item`** indica o tipo dos itens armazenados na lista.
- O tipo **`Lista`** é usado para declarar um ponteiro de lista (que aponta o primeiro nó).
- Se **`I`** é um ponteiro de lista, então **`I->item`** é o primeiro item da lista.
- Se **`I`** é um ponteiro de lista, então **`I->prox`** é o endereço do segundo nó da lista (resto).

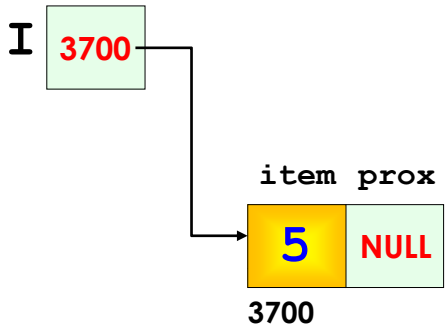
Note que o tipo **`Item`** pode ser redefinido, em função da aplicação que usa o tipo **`Lista`**!

Listas ordenadas

Exemplo 2. Criação de nó (recapitulação)

```
Lista no(Item x, Lista p) {
    Lista n = malloc(sizeof(struct no));
    n->item = x;
    n->prox = p;
    return n;
}
```

```
Lista I = no(5,NULL);
```

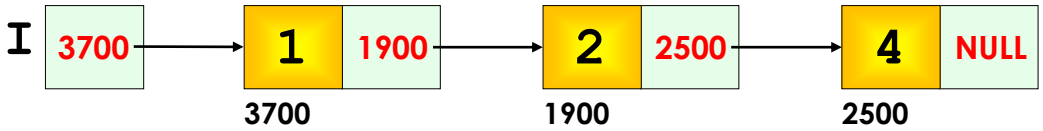


Usando a função `no()` é possível criar qualquer lista desejada!

Listas ordenadas

Exemplo 3. Criação de lista ordenada com a função `no()`

```
Lista I =                3700                ;
```



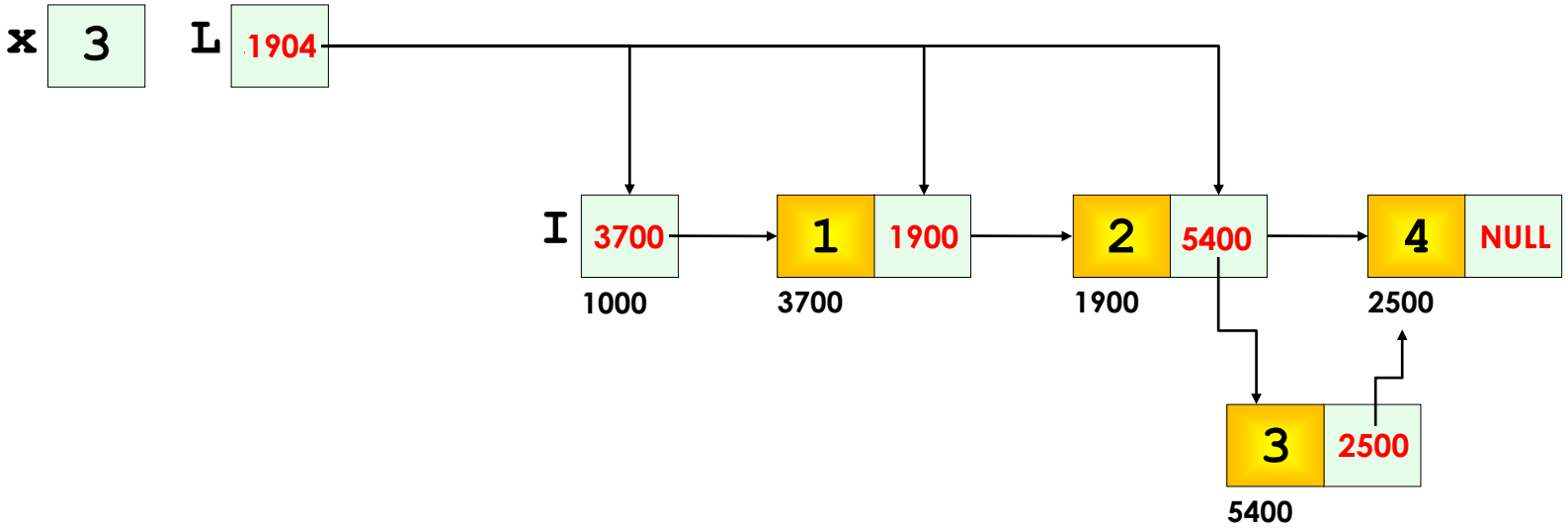
Por que usar a função **`no()`** não é a forma mais adequada de se criar uma lista ordenada?

Listas ordenadas

Exemplo 4. Inserção em lista ordenada

```
void ins(Item x, Lista *L) {
    while( *L != NULL && (*L)->item < x )
        L = &(*L)->prox;
    *L = no(x,*L);
}
```

`ins(3,&I)`



Note que essa operação permite a inserção de itens repetidos!

Listas ordenadas

Exercício 1. Programa para inserção em lista ordenada

Complete e execute o programa a seguir.

```
#include <stdio.h>
#include <stdlib.h>

...
int main(void) {
    Lista I = NULL;
    ins(4,&I);
    ins(1,&I);
    ins(3,&I);
    ins(5,&I);
    ins(2,&I);
    exibe(I);
    return 0;
}
```

Listas ordenadas

Exercício 2. Inserção em lista ordenada sem repetição

Crie a função iterativa **`ins_sr(x,&L)`**, que insere o item **`x`** na lista ordenada **`L`** somente se **`x`** não estiver em **`L`**. Em seguida, faça um programa para testar o funcionamento da função.

Exercício 3. Inserção recursiva em lista ordenada

Crie a função recursiva **`ins_rec(x,&L)`**, que insere um item **`x`** numa lista ordenada **`L`**. Em seguida, faça um programa para testar o funcionamento da função.

Exercício 4. Exibição recursiva crescente de lista ordenada

Crie a função recursiva **`exibe_crescente(L)`**, que exibe em ordem crescente todos os itens da lista ordenada **`L`**. Depois, faça um programa para testar o funcionamento da função.

Exercício 5. Exibição recursiva decrescente de lista ordenada

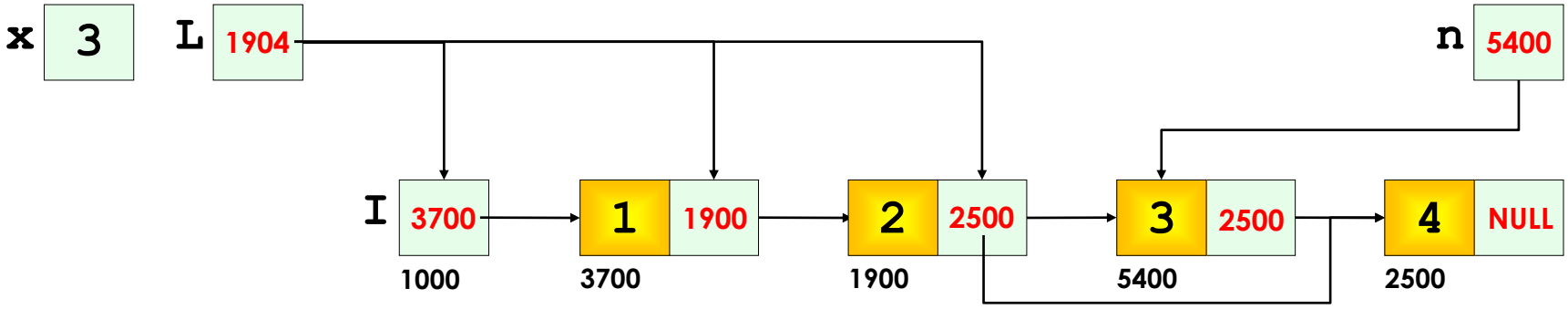
Crie a função recursiva **`exibe_decrescente(L)`**, que exibe em ordem decrescente todos os itens da lista ordenada **`L`**. Depois, faça um programa para testar o funcionamento da função.

Listas ordenadas

Exemplo 5. Remoção em lista ordenada

```
void rem(Item x, Lista *L) {
    while( *L != NULL && (*L)->item < x )
        L = &(*L)->prox;
    if( *L == NULL || (*L)->item > x ) return;
    Lista n = *L;
    *L = n->prox;
    free(n);
}
```

```
rem(3,&I)
```



Note que, caso o item não seja encontrado, a lista não é alterada!

Listas ordenadas

Exercício 6. Programa para remoção em lista ordenada

Complete e execute o programa a seguir.

```
#include <stdio.h>
#include <stdlib.h>

...
int main(void) {
    Lista I = NULL;
    ins(4,&I); ins(1,&I); ins(3,&I); ins(5,&I); ins(2,&I);
    rem(3,&I);
    exibe(I);
    return 0;
}
```

Exercício 7. Remoção de todas as ocorrências em lista ordenada

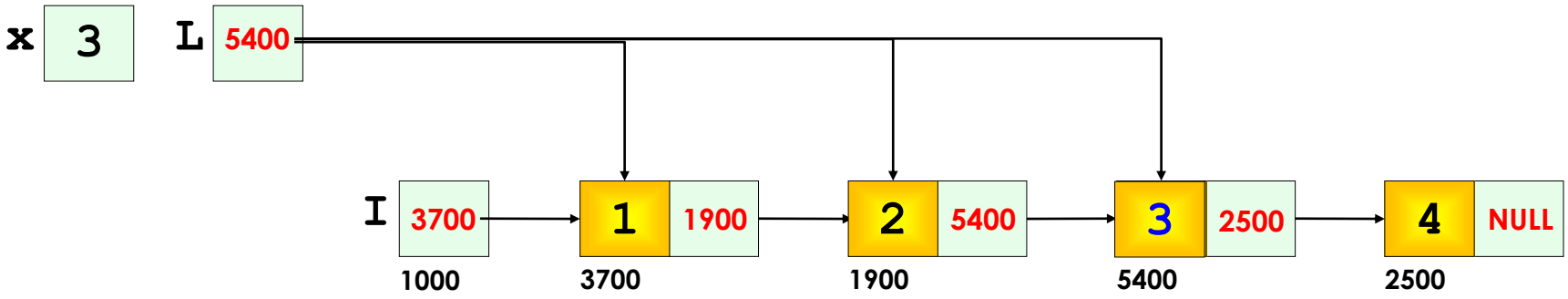
Crie a função iterativa **`rem_todo(x,&L)`**, que remove toda ocorrência do item **`x`** na lista ordenada **`L`**. Em seguida, faça um programa para testar o funcionamento da função.

Listas ordenadas

Exemplo 6. Pertinência em lista ordenada

```
int pert(Item x, Lista L) {
    while( L != NULL && L->item < x )
        L = L->prox;
    return (L != NULL && L->item == x);
}
```

`pert(3,I)`



Note que a função devolve 0 (falso) quando o item não é encontrado na lista!

Listas ordenadas

Exercício 8. Programa para verificação de pertinência em lista ordenada

Complete e execute o programa a seguir.

```
#include <stdio.h>
#include <stdlib.h>

...

int main(void) {
    Lista I = NULL;
    ins(4,&I); ins(1,&I); ins(3,&I); ins(2,&I);
    printf("%d\n",pert(5,I));
    printf("%d\n",pert(3,I));
    return 0;
}
```

Exercício 9. Verificação de pertinência recursiva

Crie a função recursiva **`pert_rec(x,L)`**, que verifica se o item **`x`** está na lista ordenada **`L`**. Em seguida, faça um programa para testar o funcionamento da função.

Fim