Mapeamentos
(IED-001)

Prof. Dr. Silvio do Lago Pereira

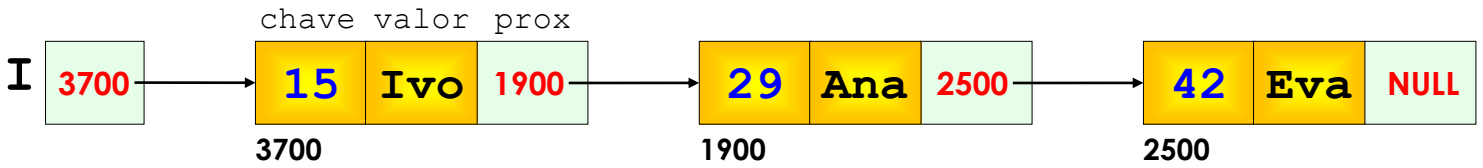
Departamento de Tecnologia da Informação

Faculdade de Tecnologia de São Paulo

Mapeamentos

Mapeamento

é uma lista encadeada ordenada cujos nós associam **valores** a **chaves**.



Observações:

- As chaves devem ser **únicas** (não pode haver chaves repetidas).
- Os valores são **ordenados** em função das chaves.

Mapeamento é útil para guardar coleção dinâmicas, em que os dados são acessados por chave!

Mapeamentos

Exemplo 1. O tipo `Map`

```
typedef int Chave;
typedef char Valor[22];
typedef struct map {
    Chave chave;
    Valor valor;
    struct map *prox;
} *Map;
```

Map ≡ struct map *

Observações:

- O tipo **`Chave`** indica o tipo das chaves armazenadas no mapeamento.
- O tipo **`Valor`** indica o tipo dos valores associados às chaves no mapeamento.
- O tipo **`Map`** é usado para declarar um ponteiro de mapeamento (que aponta o primeiro nó).

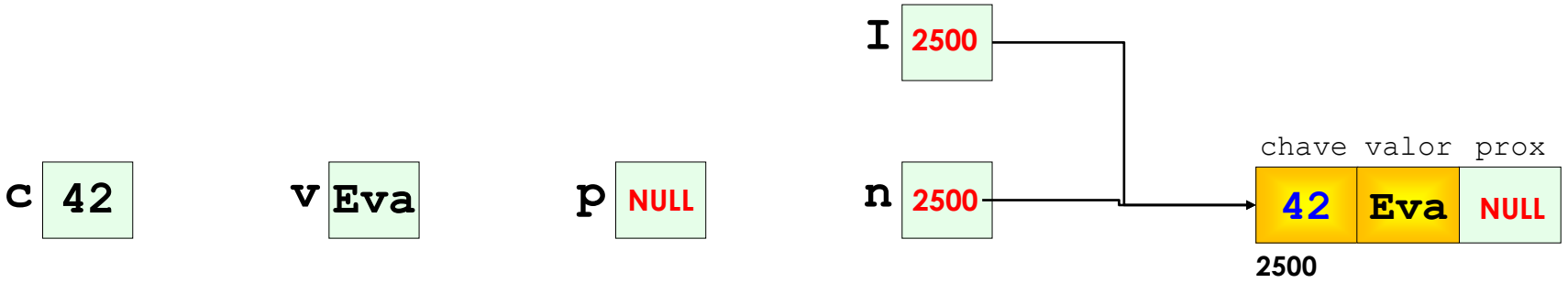
Note que os tipos **`Chave`** e **`Valor`** podem ser redefinidos, em função da aplicação!

Mapeamentos

Exemplo 2. Criação de nó de mapeamento

```
Map no_map(Chave c, Valor v, Map p) {
    Map n = malloc(sizeof(struct map));
    n->chave = c;
    strcpy(n->valor,v);
    n->prox = p;
    return n;
}
```

```
Map I = no_map(42,"Eva",NULL);
```



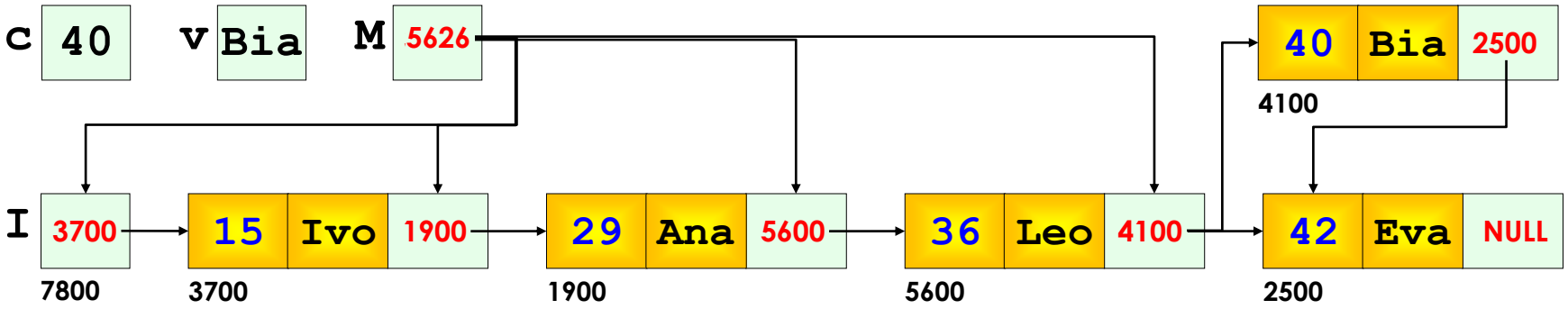
Usando a função **`no_map()`** é possível criar qualquer mapeamento desejado!

Mapeamentos

Exemplo 3. Inserção em mapeamento

```
void insm(Chave c, Valor v, Map *M) {
    while( *M && c>(*M)->chave )
        M = &(*M)->prox;
    if( *M && c==(*M)->chave )
        strcpy((*M)->valor,v);
    else *M = no_map(c,v,*M);
}
```

```
insm(40,"Bia",&I)
```



Note que essa operação substitui o valor associado à chave, se ela já existir no mapeamento!

Mapeamentos

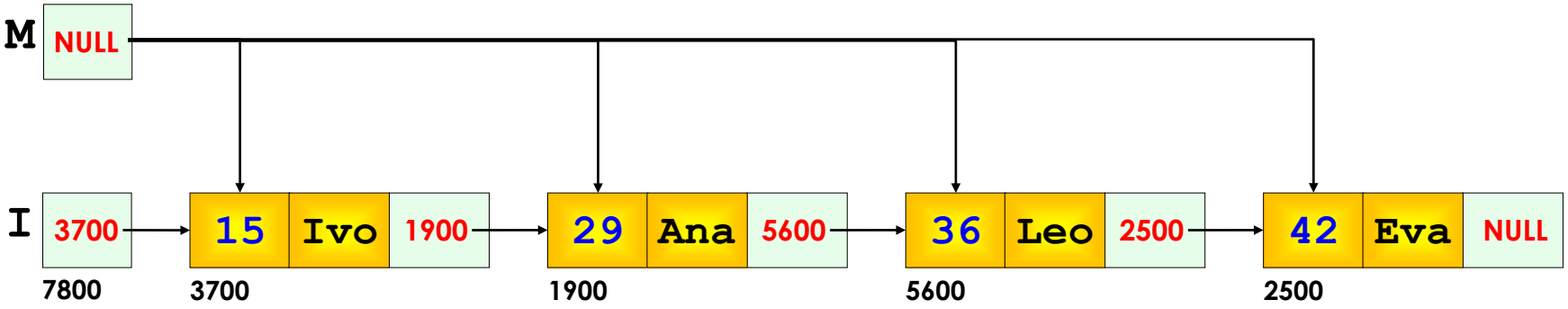
Exemplo 4. Exibição de mapeamento

```
void exibem(Map M) {
   printf("[");
   while( M ) {
      printf("(%d,%s)",M->chave,M->valor);
      if( M->prox ) printf(",");
      M = M->prox;
   }
   printf("]\n");
}
```

```
exibem(I)
```

vídeo

```
[(15,Ivo) ,(29,Ana) ,(36,Leo) ,(42,Eva) ]
```



Mapeamentos

Exercício 1. Inserção e exibição

Complete e execute o programa a seguir.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
...
int main(void) {
    Map I = NULL;
    insm(36,"Leo",&I);
    insm(15,"Ivo",&I);
    insm(42,"Eva",&I);
    insm(29,"Ana",&I);
    exibem(I);
    insm(29,"Bia",&I);
    exibem(I);
    return 0;
}
```

Exercício 2. Inserção recursiva

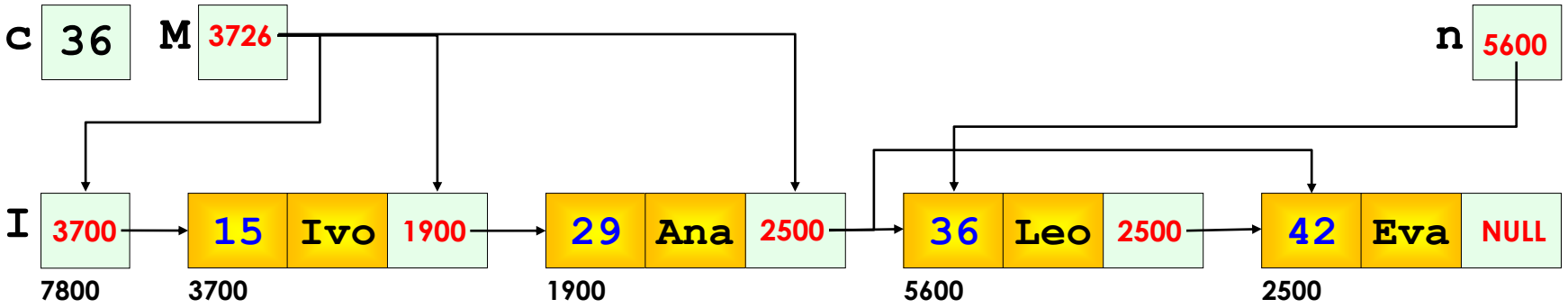
Crie a função recursiva **`insmr(c,v,&I)`**, que insere o par **`(c,v)`** no mapeamento **`I`**.

Mapeamentos

Exemplo 5. Remoção em mapeamento

```
void remm(Chave c, Map *M) {
    while( *M && c>(*M)->chave )
        M = &(*M)->prox;
    if( *M==NULL || c!=(*M)->chave ) return;
    Map n = *M;
    *M = n->prox;
    free(n);
}
```

```
remm(36,&I)
```



Note que, se a chave não existir no mapeamento, a função não causa nenhuma alteração nele!

Mapeamentos

Exercício 3. Remoção

Complete e execute o programa a seguir.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
...
int main(void) {
    Map I = NULL;
    insm(36,"Leo",&I); insm(15,"Ivo",&I);
    insm(42,"Eva",&I); insm(29,"Ana",&I);
    exibem(I);
    remm(29,&I);
    exibem(I);
    remm(42,&I);
    exibem(I);
    return 0;
}
```

Exercício 4. Remoção recursiva

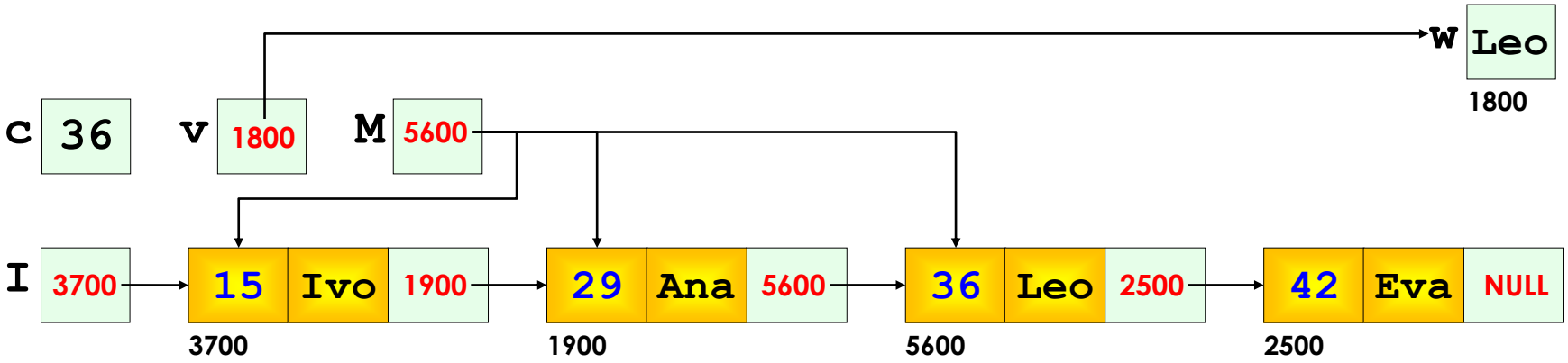
Crie a função recursiva **`remmr(c,&I)`**, que remove a chave **`c`** do mapeamento **`I`**.

Mapeamentos

Exemplo 6. Pertinência em mapeamento

```
int pertm(Chave c, Valor v, Map M) {
    while( M && c>M->chave )
        M = M->prox;
    if( M && c==M->chave )
        strcpy(v,M->valor);
    return (M && c==M->chave);
}
```

```
pertm(36,w,I)
```



Note que, se a chave não existir no mapeamento, a variável apontada por **v** não é alterada!

Mapeamentos

Exercício 5. Pertinência

Complete e execute o programa a seguir.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
...
int main(void) {
   Valor w;
   Map I = NULL;
   insm(36,"Leo",&I); insm(15,"Ivo",&I); insm(42,"Eva",&I);
   exibem(I);
   if( pertm(42,w,I) ) printf("Valor da chave 42: %s\n",w);
   else puts("Chave 42 inexistente!");
   if( pertm(10,w,I) ) printf("Valor da chave 10: %s\n",w);
   else puts("Chave 10 inexistente!");
   return 0;
}
```

Exercício 6. Pertinência recursiva

Crie a função recursiva **`pertmr(c,v,&I)`**, que verifica se a chave **`c`** está no mapeamento **`I`**.

Mapeamentos

Exercício 7. Destruição iterativa

Crie a função iterativa **`destroim(&I)`**, que destrói o mapeamento **`I`**.

Exercício 8. Destruição recursiva

Crie a função recursiva **`destroimr(&I)`**, que destrói o mapeamento **`I`**.

Exercício 9. Chaves e valores inteiros

Altere a implementação de mapeamentos para criar mapeamentos em que tanto as chaves quanto seus valores associados são números inteiros.

```
typedef int Chave;
typedef int Valor;
```

Exercício 10. Chaves e valores cadeia

Altere a implementação de mapeamentos para criar mapeamentos em que tanto as chaves quanto seus valores associados são cadeias de caracteres.

```
typedef char Chave[22];
typedef char Valor[22];
```

Fim