

Princípios de Desenvolvimento de Algoritmos **MAC122**

Prof. Dr. Paulo Miranda
IME-USP

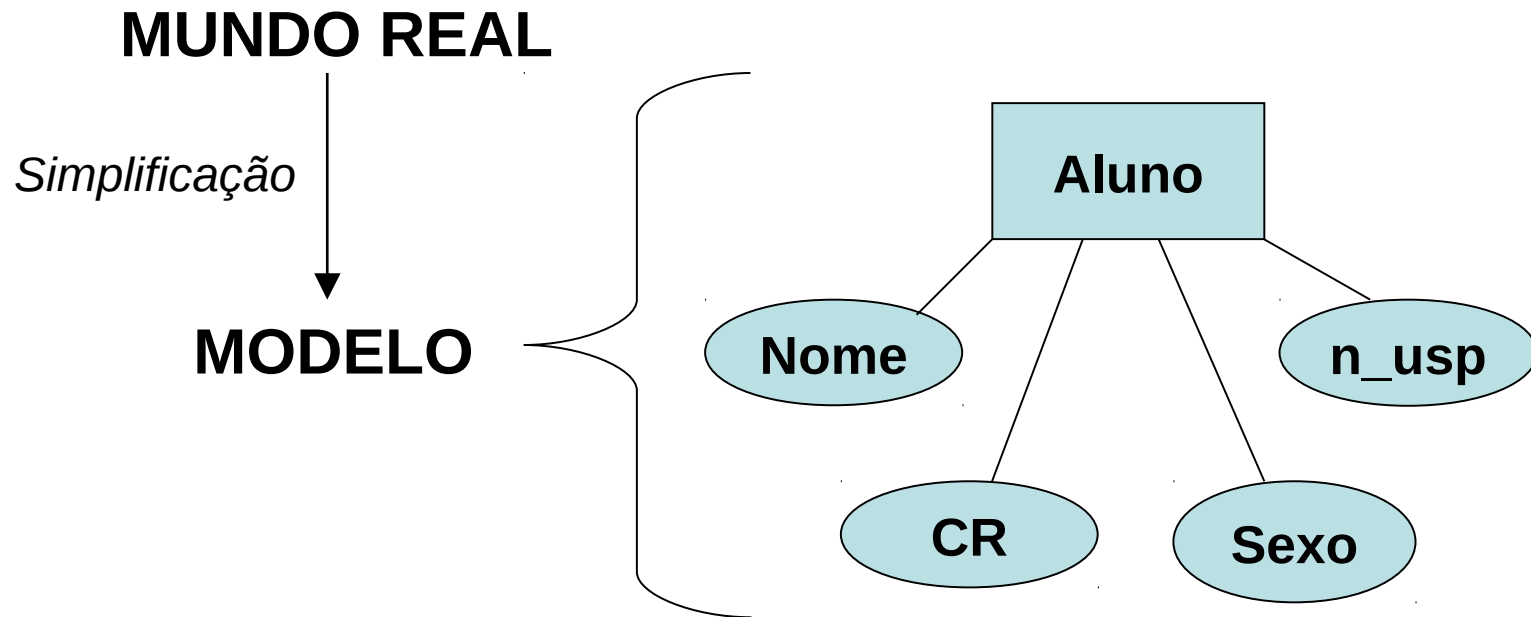
Registros

Registros/Estruturas

- **Motivação:**
 - **MODELAGEM DE DADOS:**
 - MUNDO REAL:
 - Objetos/conceitos complexos.
 - MODELO:
 - Representações que simplificam a realidade.
 - Tornam possível o estudo e a manipulação.
 - Úteis para um determinado propósito.

Registros/Estruturas

- **Motivação:**
 - **MODELAGEM DE DADOS:**



Registros/Estruturas

- **Motivação:**

- **MODELAGEM D**

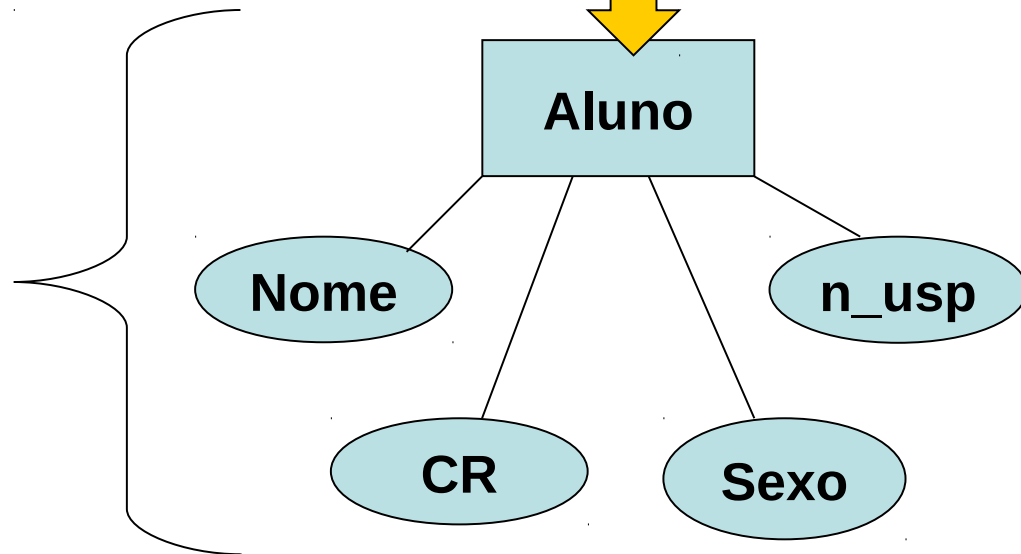
Entidade:

Algo que é relevante para o sistema.
Pode estar associada a um elemento concreto,
ou a um conceito “abstrato”.

MUNDO REAL

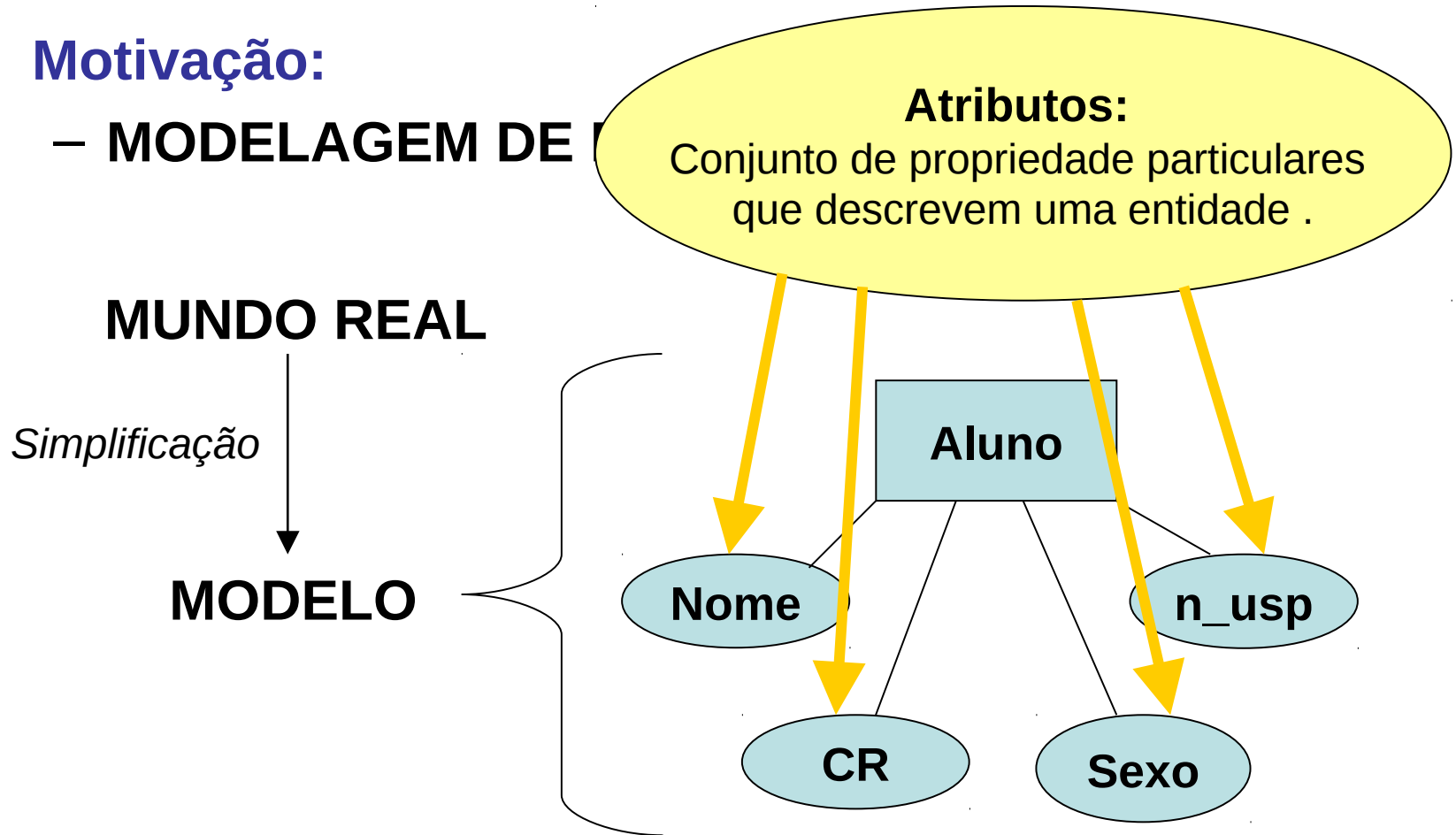
Simplificação

MODELO



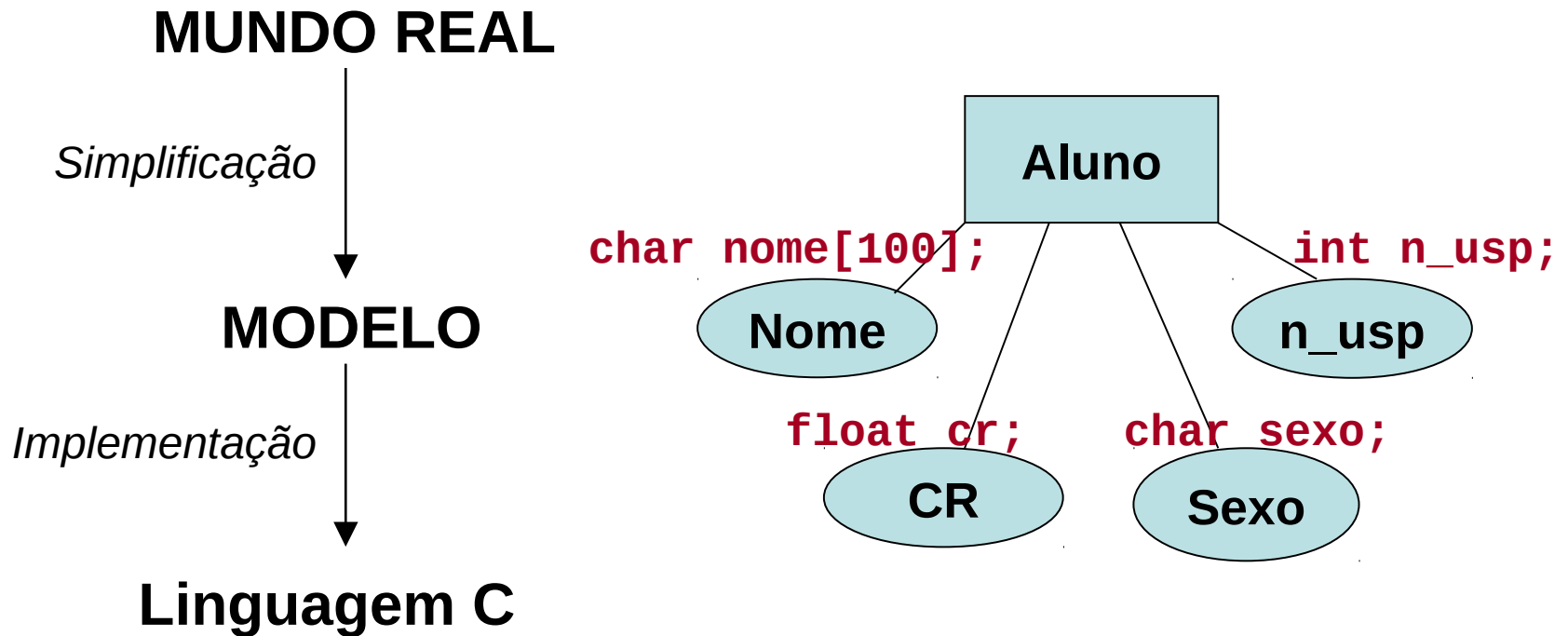
Registros/Estruturas

- **Motivação:**
 - **MODELAGEM DE**



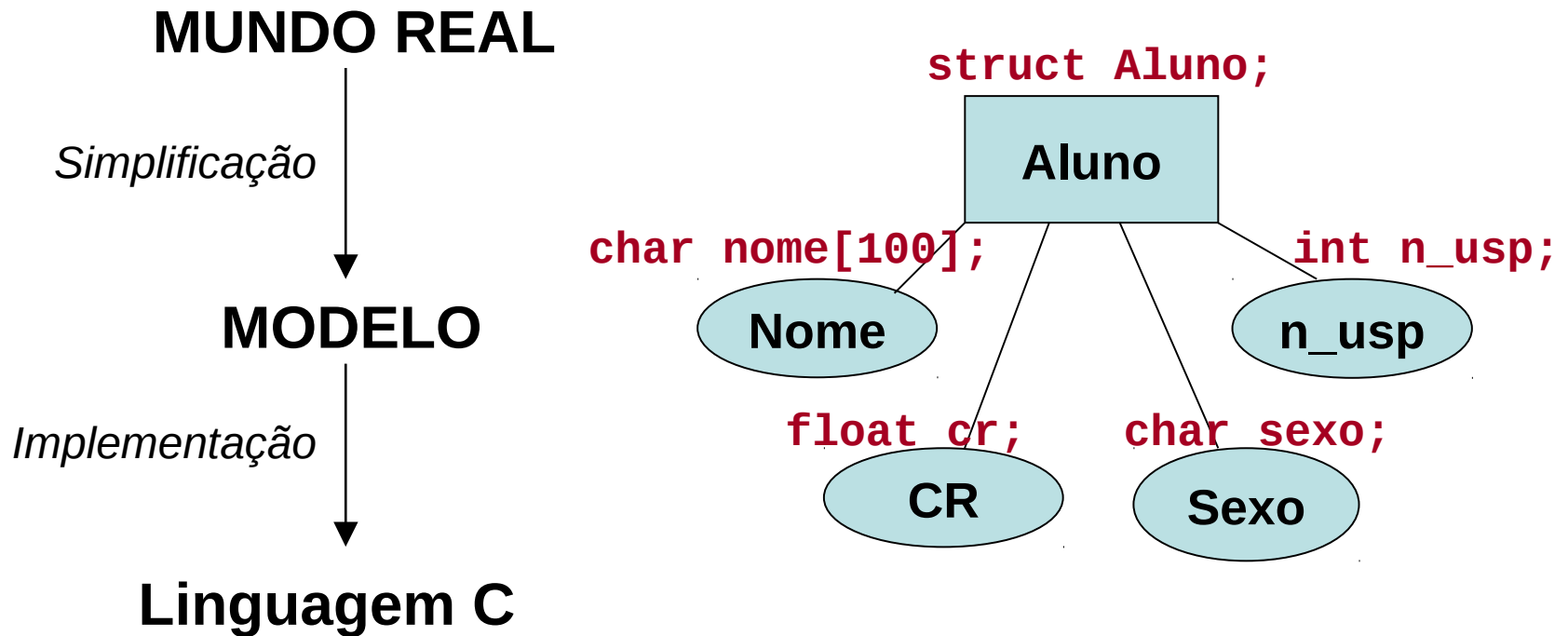
Registros/Estruturas

- **Problema:**
 - Como representar um modelo em linguagem C?
 - Conjunto de variáveis.



Registros/Estruturas

- **Problema:**
 - Como representar um modelo em linguagem C?
 - Conjunto de variáveis.
 - Podem ser agrupadas em estruturas.



Registros/Estruturas

- **Definição:**

- Uma estrutura é um tipo de dado cujo formato é definido pelo programador.
- Ela agrupa elementos, chamados membros ou campos da estrutura, que não necessitam ser do mesmo tipo.

Exemplo:

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

struct Aluno{
    char   nome[100];
    int    n_usp;
    float  cr;
    char   sexo;
    int    curso;
};

int main(){
    struct Aluno aluno;

    strcpy(aluno.nome, "Fulano da Silva");
    aluno.n_usp = 9233481;
    aluno.cr = 1.0;
    aluno.sexo = 'M';
    aluno.curso = 45070;
    return 0;
}
```

Exemplo:

Definição do formato da estrutura.

Tipo **struct Aluno** é definido.

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

```
struct Aluno{
    char   nome[100];
    int    n_esp;
    float  cr;
    char   sexo;
    int    curso;
};
```

```
int main(){
    struct Aluno aluno;

    strcpy(aluno.nome, "Fulano da Silva");
    aluno.n_esp = 9233481;
    aluno.cr = 1.0;
    aluno.sexo = 'M';
    aluno.curso = 45070;
    return 0;
}
```

Exemplo:

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

```
struct Aluno{
    char   nome[100];
    int    n_esp;
    float  cr;
    char   sexo;
    int    curso;
};
```

“Etiqueta” da estrutura

Membros da estrutura

```
int main(){
    struct Aluno aluno;

    strcpy(aluno.nome, "Fulano da Silva");
    aluno.n_esp = 9233481;
    aluno.cr = 1.0;
    aluno.sexo = 'M';
    aluno.curso = 45070;
    return 0;
}
```

Exemplo:

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

```
struct Aluno{
    char   nome[100];
    int    n_usp;
    float  cr;
    char   sexo;
    int    curso;
};
```

Declaração de uma
variável do tipo
definido.

```
int main(){
    struct Aluno aluno;
```

```
    strcpy(aluno.nome, "Fulano da Silva");
    aluno.n_usp = 9233481;
    aluno.cr = 1.0;
    aluno.sexo = 'M';
    aluno.curso = 45070;
    return 0;
}
```

Exemplo:

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

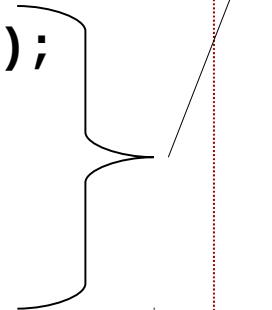
```
struct Aluno{
    char   nome[100];
    int    n_usp;
    float  cr;
    char   sexo;
    int    curso;
};
```

```
int main(){
    struct Aluno aluno;
```

```
    strcpy(aluno.nome, "Fulano da Silva");
    aluno.n_usp = 9233481;
    aluno.cr = 1.0;
    aluno.sexo = 'M';
    aluno.curso = 45070;
    return 0;
```

```
}
```

Acessando membros
da estrutura.
nome_registro.campo



Registros/Estruturas

- Operações de entrada e saída (leitura e escrita):
 - A **leitura/escrita** de estruturas pela **entrada/saída** padrão, deve ser feita **campo a campo**, como se os membros fossem variáveis independentes.

```
/* fgets(aluno.nome, 99, stdin); */  
scanf(" %[^\\n]", aluno.nome);  
scanf("%d", &aluno.n_usp);  
scanf("%f", &aluno.cr);  
scanf(" %c", &aluno.sexo);  
scanf("%d", &aluno.curso);
```

```
printf("Nome: %s\\n", aluno.nome);  
printf("NUSP: %d\\n", aluno.n_usp);  
printf("CR: %.2f\\n", aluno.cr);  
printf("Sexo: %c\\n", aluno.sexo);  
printf("Curso: %d\\n", aluno.curso);
```

Registros/Estruturas

- **Atribuição entre estruturas:**
 - O conteúdo de uma variável estrutura pode ser atribuído a outra variável estrutura do mesmo tipo.
 - Todos campos da estrutura de origem são atribuídos aos membros correspondentes da estrutura de destino.

```
int main(){
    struct Aluno aluno1;
    struct Aluno aluno2;
    ...

    aluno1 = aluno2;
    ...
    return 0;
}
```

Estruturas Aninhadas

- Estruturas com campos que são estruturas.
- Recurso para a criação de tipos de dados complexos.

```
struct Data{
    int dia;
    int mes;
    int ano;
};

struct Endereco{
    char rua[100];
    char bairro[100];
    char cidade[100];
    int cep;
};
```

```
struct Aluno{
    char nome[100];
    int n_usp;
    float cr;
    char sexo;
    int curso;
    struct Data nascimento;
    struct Endereco endereco;
};
```


Estruturas Aninhadas

- Estruturas com campos que são estruturas.
- Recurso para a criação de tipos de dados complexos.

```
int main(){
    struct Aluno aluno;

    strcpy(aluno.nome, "Fulano da Silva");
    aluno.n_usp = 9233481;
    aluno.cr = 1.0;
    aluno.sexo = 'M';
    aluno.curso = 45070;
    aluno.nascimento.dia = 13;
    aluno.nascimento.mes = 9;
    aluno.nascimento.ano = 1980;
    strcpy(aluno.endereco.rua, "Av. Dr. Alberto Sarmiento");
    strcpy(aluno.endereco.bairro, "Castelo");
    strcpy(aluno.endereco.cidade, "Campinas");
    aluno.endereco.cep = 1306275;
    ...
}
```