Introducao a Programacao: Estruturas

Prof. Pericles Miranda

Slides fornecidos pelo prof. Gurvan Huiban

Tipo estruturado homogêneo

- Tipo de variável que contém elementos de um mesmo tipo;
- Tipicamente: vetores unidimensionais, vetores multidimensionais.

Tipo estruturado heterogêneo

- Tipo de variável que contém elementos que podem ser de tipos diferentes;
- Chamado Estrutura;
- Chamamos Campos os elementos de uma estrutura;
- Em C, definido com a palavra chave `struct`.

Exemplo: Estrutura

Queremos definir um tipo de variável para armazenar *alunos* com as informações:

- Nome (cadeia de caracteres);
- Código da turma (cadeia de caracteres);
- Matricula (`long int`);
- Idade (`int`);
- ...

Estrutura TAluno

```
struct TAluno
{
   char nome[250];
   char codTurma[10];
   long int matricula;
   int idade;
}

struct TAluno aluno;
```

Sintaxe: Declaração do tipo

```
struct Id_Estrutura {
  Tipo_dado_1 Id_Campo_1 ;
  Tipo_dado_2 Id_Campo_2 ;
  ...
  Tipo_dado_n Id_Campo_n ;
};
struct Id_Estrutura nome_variavel;
```

- `Id_Estrutura`: Identificador da estrutura;
- `Id_Campo`: Identificadores do campo;
- `Tipo_dado`: Tipo de dado do campo. Pode ser qualquer (`char`, `float`, vetor, outro estrutura, ...).

Exercício: Estrutura de endereço

Definir uma estrutura `TEndereco` para armazenar endereços, com os campos seguintes:

- `Rua` (255 caracteres);
- `Número` (inteiro);
- `Complemento` (64 caracteres);
- `Bairro` (32 caracteres);
- `Cidade`(32 caracteres);
- `CEP1` (`long int` contendo os 5 primeiros dígitos);
- `CEP2` (inteiro contendo os 3 últimos dígitos);
- `Estado` (2 caracteres).

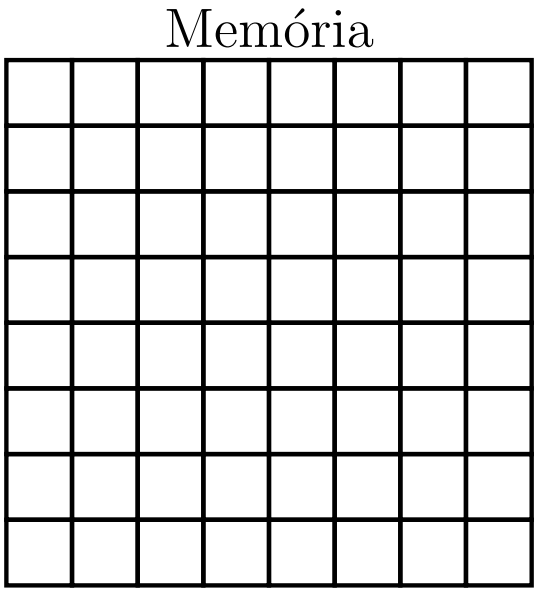
Sintaxe: Uso da estrutura

Acesso ao campo `Identificador_do_campo` de uma variável `nome_variavel` de tipo estrutura:

```
nome_variavel.Id_Campo
```

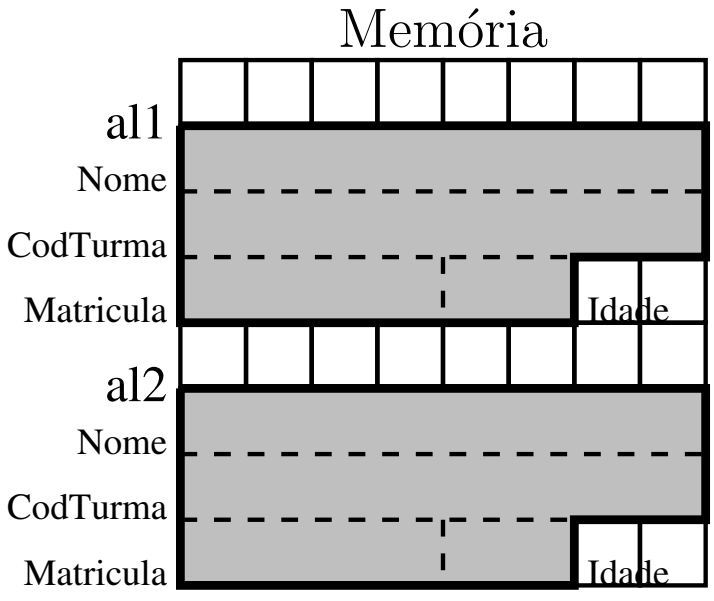
```
struct TAluno
{
  char Nome[250];
  char CodTurma[10];
  long int Matricula;
  int Idade;
};

main()
{
  struct TAluno al1, al2;
  strcpy(al1.Nome, "L.A. Cauchy");
  strcpy(al1.CodTurma, "X1805");
  al1.Matricula = 21081789;
  al1.Idade = 224;
  strcpy(al2.Nome, "L. Euler");
  strcpy(al2.CodTurma, "Ba1723");
  al2.Matricula = 15041707;
  al2.Idade = 306;
```



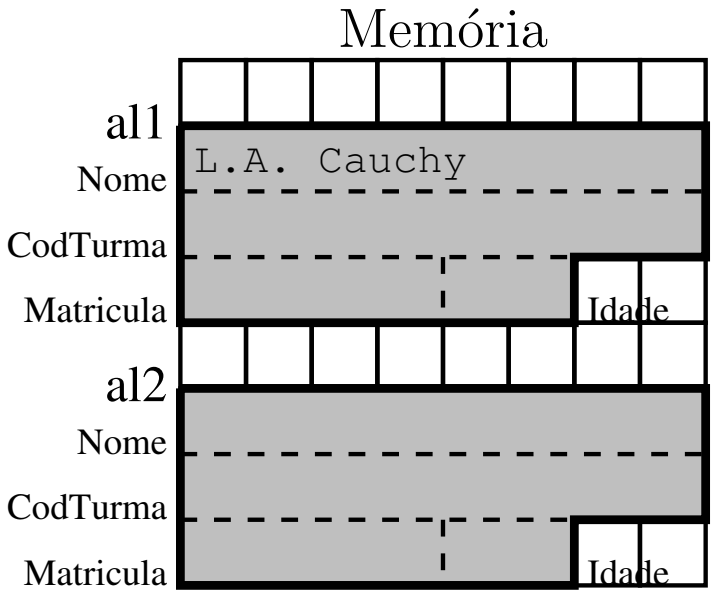
```
struct TAluno
{
  char Nome[250];
  char CodTurma[10];
  long int Matricula;
  int Idade;
};

main()
{
  struct TAluno al1, al2;
  strcpy(al1.Nome, "L.A. Cauchy");
  strcpy(al1.CodTurma, "X1805");
  al1.Matricula = 21081789;
  al1.Idade = 224;
  strcpy(al2.Nome, "L. Euler");
  strcpy(al2.CodTurma, "Ba1723");
  al2.Matricula = 15041707;
  al2.Idade = 306;
```



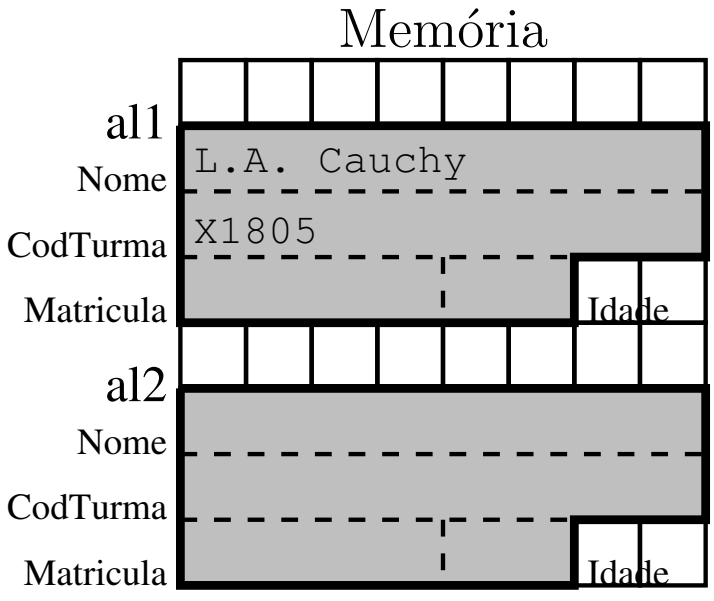
```
struct TAluno
{
  char Nome[250];
  char CodTurma[10];
  long int Matricula;
  int Idade;
};

main()
{
  struct TAluno al1, al2;
  strcpy(al1.Nome, "L.A. Cauchy");
  strcpy(al1.CodTurma, "X1805");
  al1.Matricula = 21081789;
  al1.Idade = 224;
  strcpy(al2.Nome, "L. Euler");
  strcpy(al2.CodTurma, "Ba1723");
  al2.Matricula = 15041707;
  al2.Idade = 306;
```



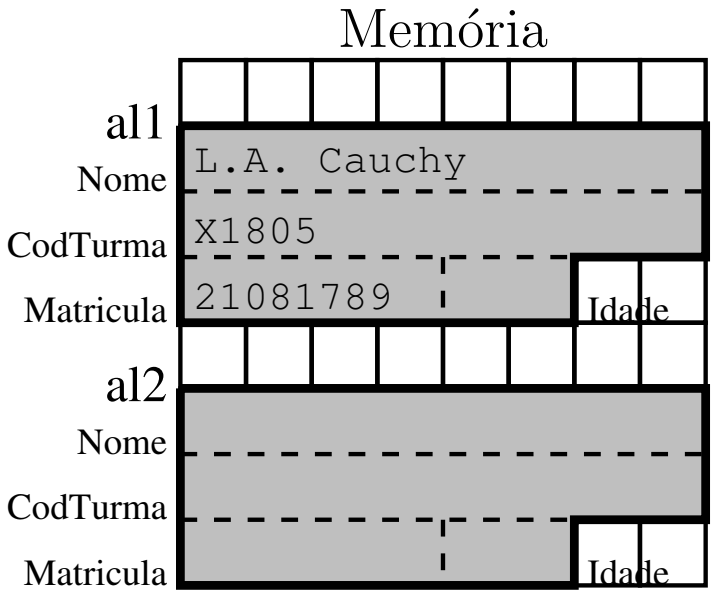
```
struct TAluno
{
  char Nome[250];
  char CodTurma[10];
  long int Matricula;
  int Idade;
};

main()
{
  struct TAluno al1, al2;
  strcpy(al1.Nome, "L.A. Cauchy");
  strcpy(al1.CodTurma, "X1805");
  al1.Matricula = 21081789;
  al1.Idade = 224;
  strcpy(al2.Nome, "L. Euler");
  strcpy(al2.CodTurma, "Ba1723");
  al2.Matricula = 15041707;
  al2.Idade = 306;
```



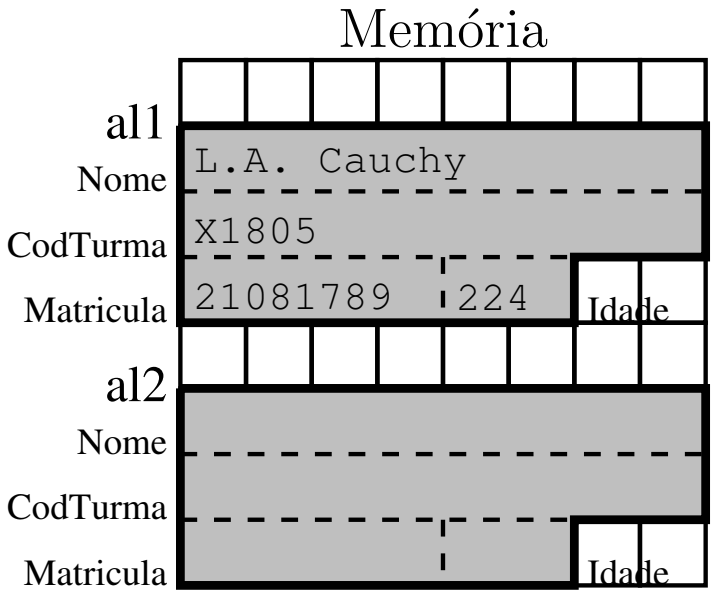
```
struct TAluno
{
  char Nome[250];
  char CodTurma[10];
  long int Matricula;
  int Idade;
};

main()
{
  struct TAluno al1, al2;
  strcpy(al1.Nome, "L.A. Cauchy");
  strcpy(al1.CodTurma, "X1805");
  al1.Matricula = 21081789;
  al1.Idade = 224;
  strcpy(al2.Nome, "L. Euler");
  strcpy(al2.CodTurma, "Ba1723");
  al2.Matricula = 15041707;
  al2.Idade = 306;
```



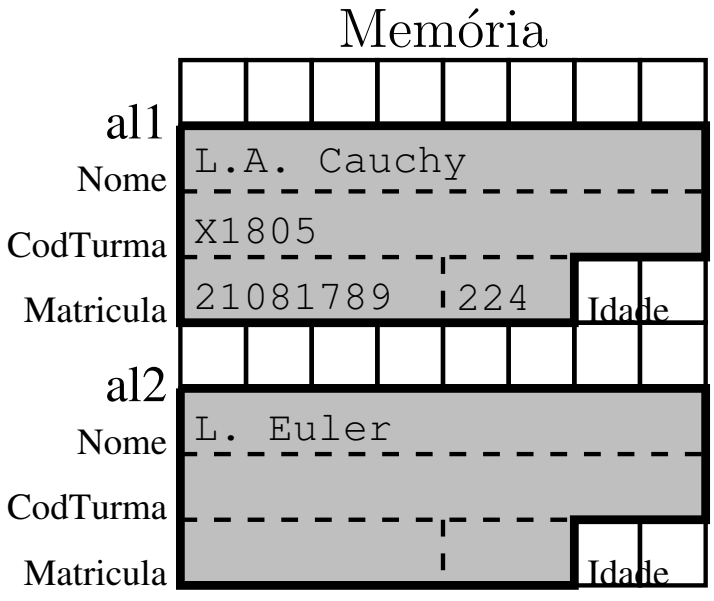
```
struct TAluno
{
  char Nome[250];
  char CodTurma[10];
  long int Matricula;
  int Idade;
};

main()
{
  struct TAluno al1, al2;
  strcpy(al1.Nome, "L.A. Cauchy");
  strcpy(al1.CodTurma, "X1805");
  al1.Matricula = 21081789;
  al1.Idade = 224;
  strcpy(al2.Nome, "L. Euler");
  strcpy(al2.CodTurma, "Ba1723");
  al2.Matricula = 15041707;
  al2.Idade = 306;
```



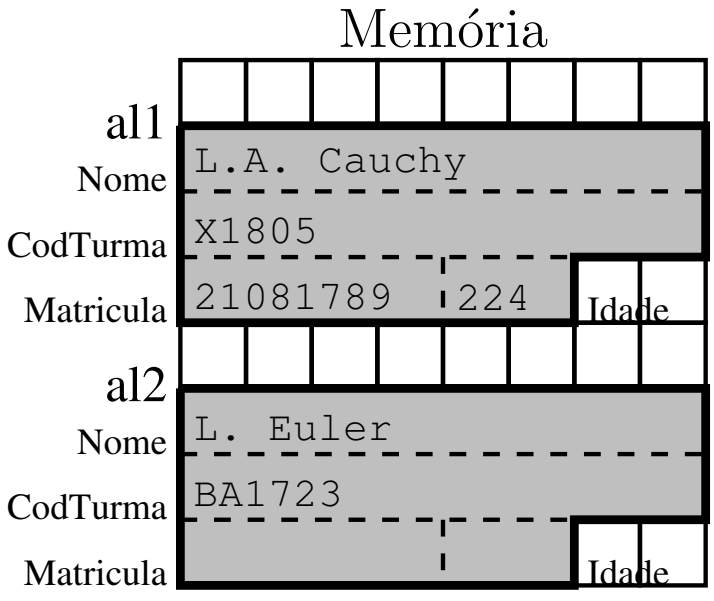
```
struct TAluno
{
  char Nome[250];
  char CodTurma[10];
  long int Matricula;
  int Idade;
};

main()
{
  struct TAluno al1, al2;
  strcpy(al1.Nome, "L.A. Cauchy");
  strcpy(al1.CodTurma, "X1805");
  al1.Matricula = 21081789;
  al1.Idade = 224;
  strcpy(al2.Nome, "L. Euler");
  strcpy(al2.CodTurma, "Ba1723");
  al2.Matricula = 15041707;
  al2.Idade = 306;
```



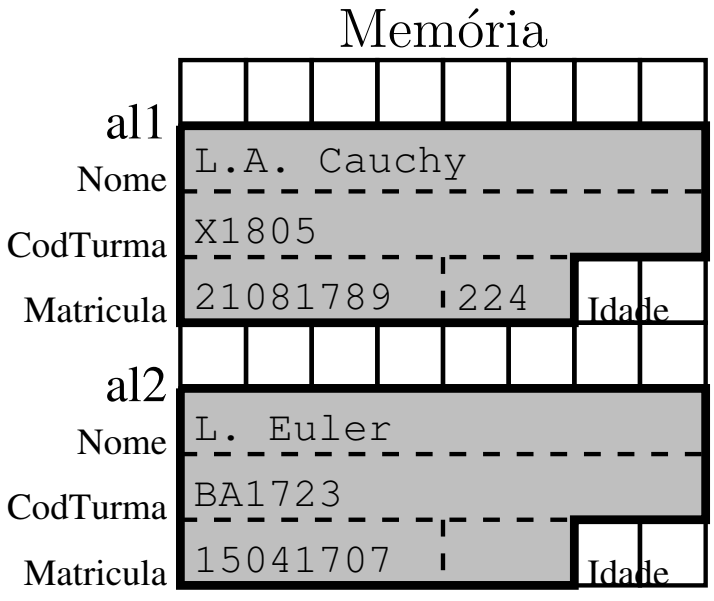
```
struct TAluno
{
  char Nome[250];
  char CodTurma[10];
  long int Matricula;
  int Idade;
};

main()
{
  struct TAluno al1, al2;
  strcpy(al1.Nome, "L.A. Cauchy");
  strcpy(al1.CodTurma, "X1805");
  al1.Matricula = 21081789;
  al1.Idade = 224;
  strcpy(al2.Nome, "L. Euler");
  strcpy(al2.CodTurma, "Ba1723");
  al2.Matricula = 15041707;
  al2.Idade = 306;
```



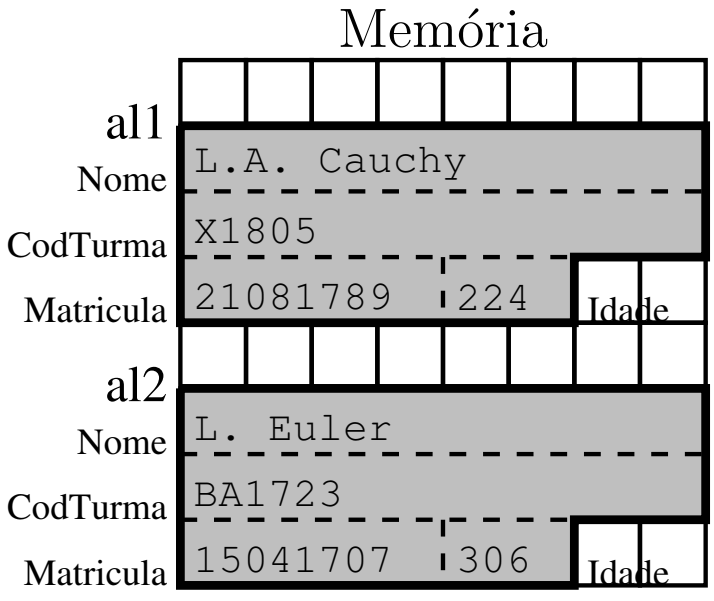
```
struct TAluno
{
  char Nome[250];
  char CodTurma[10];
  long int Matricula;
  int Idade;
};

main()
{
  struct TAluno al1, al2;
  strcpy(al1.Nome, "L.A. Cauchy");
  strcpy(al1.CodTurma, "X1805");
  al1.Matricula = 21081789;
  al1.Idade = 224;
  strcpy(al2.Nome, "L. Euler");
  strcpy(al2.CodTurma, "Ba1723");
  al2.Matricula = 15041707;
  al2.Idade = 306;
```



```
struct TAluno
{
  char Nome[250];
  char CodTurma[10];
  long int Matricula;
  int Idade;
};

main()
{
  struct TAluno al1, al2;
  strcpy(al1.Nome, "L.A. Cauchy");
  strcpy(al1.CodTurma, "X1805");
  al1.Matricula = 21081789;
  al1.Idade = 224;
  strcpy(al2.Nome, "L. Euler");
  strcpy(al2.CodTurma, "Ba1723");
  al2.Matricula = 15041707;
  al2.Idade = 306;
```



Exercício: Preencher um endereço

Escrever um programa em C que peça ao usuário os dados para preencher uma variável do tipo `TEndereco` como definido anteriormente:

- `Rua` (255 caracteres);
- `Número` (inteiro);
- `Complemento` (64 caracteres);
- `Bairro` (32 caracteres);
- `Cidade`(32 caracteres);
- `CEP1` (`long int` contendo os 5 primeiros dígitos);
- `CEP2` (inteiro contendo os 3 últimos dígitos);
- `Estado` (2 caracteres).

Pedir por último o número e o CEP.

Uma estrutura pode conter outras estruturas

```
struct TEndereco
{
    char rua[255];
    int numero;
    char complemento[64];
    ...
    char estado[2];
};
struct TAluno
{
    char nome[250];
    char codTurma[10];
    long int matricula;
    int idade;
    struct TEndereco endereco;
};
```

ou

```
struct TAluno
{
    char nome[250];
    char codTurma[10];
    long int matricula;
    int idade;
    struct TEndereco
    {
        char rua[255];
        int numero;
        char complemento[64];
        ...
        char estado[2];
    } endereco;
};
```

Acesso

Acesso ao campo `complemento` de um campo `endereco` de tipo `struct TEndereco` de uma variável `aluno` de tipo `struct TAluno`:

```
aluno.endereco.complemento;
```

Uma estrutura pode conter vetores

```
struct TAluno
{
    char nome[250];
    char codTurma[10];
    long int matricula;
    int idade;
    struct TEndereco endereco;
    int notas[4];
};
```

Acesso

```
aluno.notas[2];
```

Podemos definir vetores de estruturas

```
struct TAluno turma[20];
```

Acesso

```
turma[1].nome;
```

typedef

- Permite nomear o tipo definido por uma estrutura;
- Simplifica o uso do tipo

```
typedef struct
{
    char nome[250];
    char codTurma[10];
    long int matricula;
    int idade;
    struct TEndereco endereco;
    int notas[4];
} TAluno;
main()
{
  TAluno aluno;
```

Exercício: Turma

- Escreva um programa que recebe as matrículas, os nomes e as notas das avaliações bimestrais de uma relação de alunos.

  O programa deve emitir um relatório incluindo as matrículas, os nomes e as médias anuais, calculadas como médias aritméticas das avaliações bimestrais.

- Usar a estrutura seguinte:
  - `matricula` : 9 caracteres;
  - `nome` : 30 caracteres;
  - `Notas` : vetor de 3 reais;
  - `Media` : real.
- O cadastro dos alunos termina quando a matrícula 0 for entrada;
- Tem no máximo 60 alunos.