Computação eletrônica: Estruturas

Gurvan Huiban
ghuiban@cin.ufpe.br

01 de julho de 2014

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 (`int`);
- Idade (`int`);
- ...

Estrutura `TAluno`

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

Declaração de variável

```
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 ;
};
```

- `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, ...).

Sintaxe: Declaração de variáveis

```
struct Id_Estrutura nome_variavel;
```

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);
- `CEP` (inteiro);

Cuidado!

Definir uma estrutura não implica que uma variável seja definida!

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

Nenhuma variável foi declarada. A estrutura define um tipo de dados, não dados.

Sintaxe: Uso da estrutura

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

```
nome_variavel.Id_Campo
```

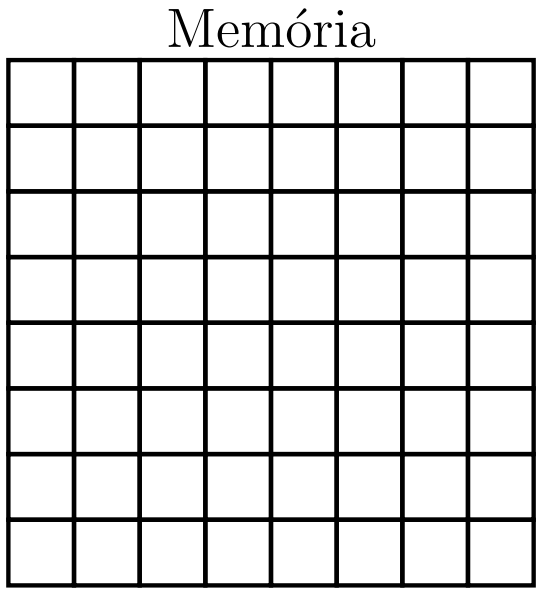
Copia de estrutura

Uma variável de tipo estrutura pode ser copiada integralmente numa outra variável da mesma estrutura:

```
TEndereco e1,e2;
...
e2 = e1;
```

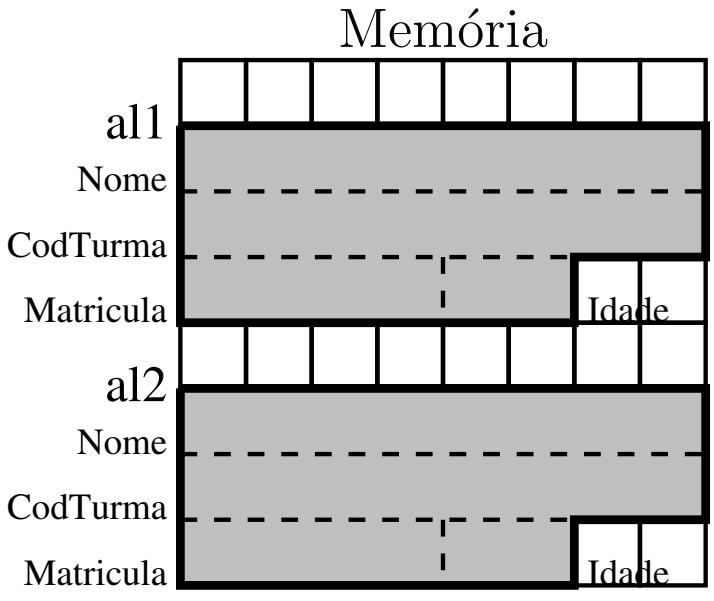
```
struct TAluno
{
  char Nome[250];
  char CodTurma[10];
  int Matricula;
  int Idade;
};

int main(void)
{
  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;
```



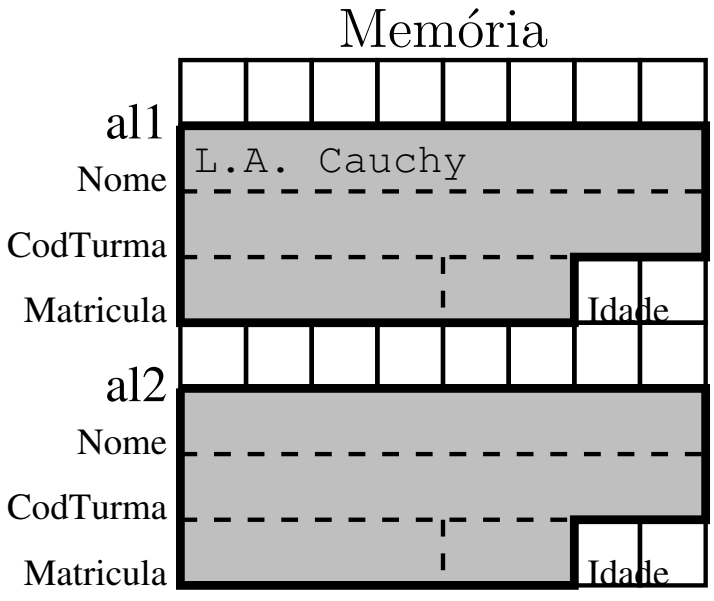
```
struct TAluno
{
  char Nome[250];
  char CodTurma[10];
  int Matricula;
  int Idade;
};

int main(void)
{
  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;
```



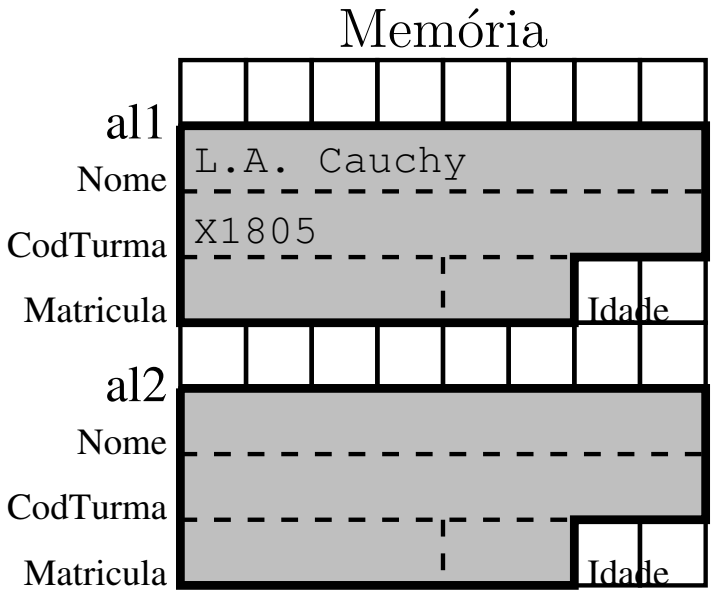
```
struct TAluno
{
  char Nome[250];
  char CodTurma[10];
  int Matricula;
  int Idade;
};

int main(void)
{
  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;
```



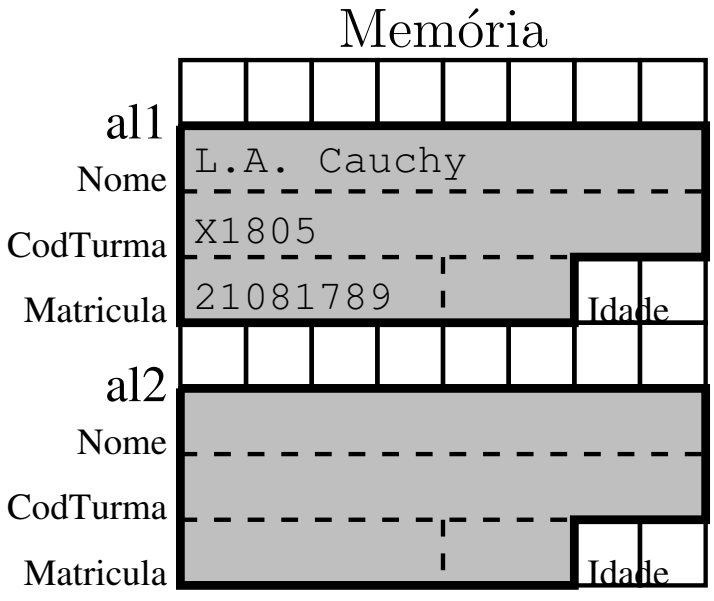
```
struct TAluno
{
  char Nome[250];
  char CodTurma[10];
  int Matricula;
  int Idade;
};

int main(void)
{
  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;
```



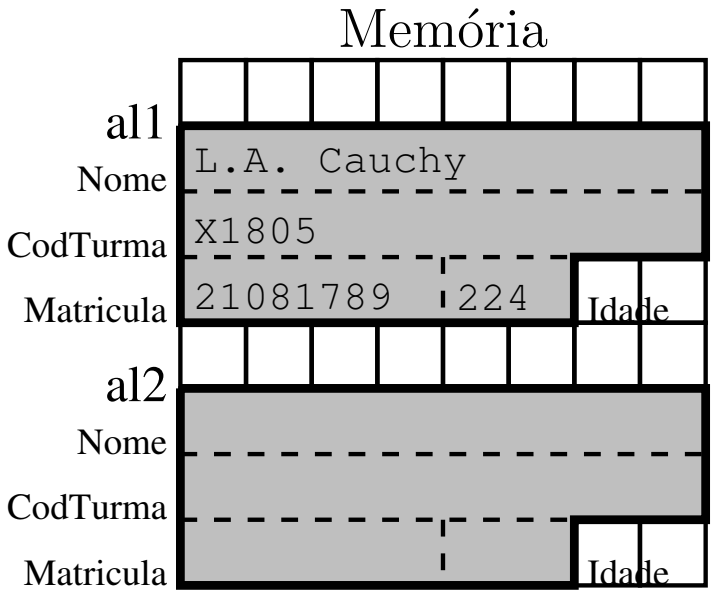
```
struct TAluno
{
  char Nome[250];
  char CodTurma[10];
  int Matricula;
  int Idade;
};

int main(void)
{
  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;
```



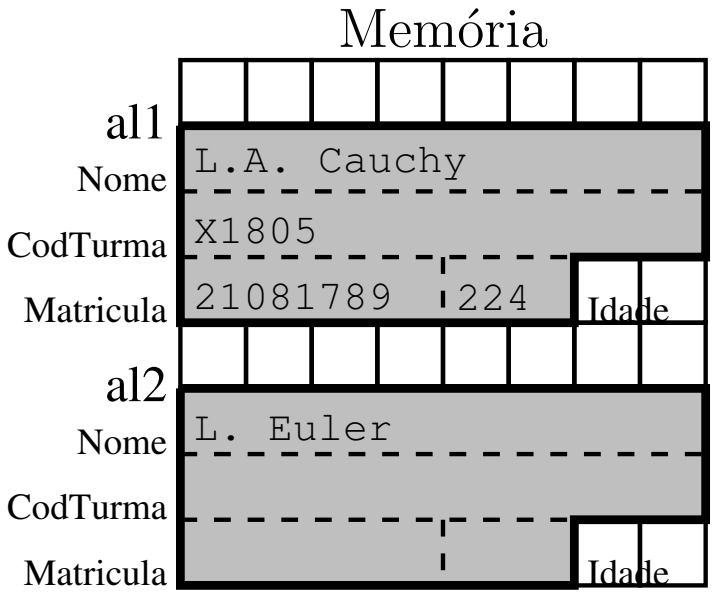
```
struct TAluno
{
  char Nome[250];
  char CodTurma[10];
  int Matricula;
  int Idade;
};

int main(void)
{
  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;
```



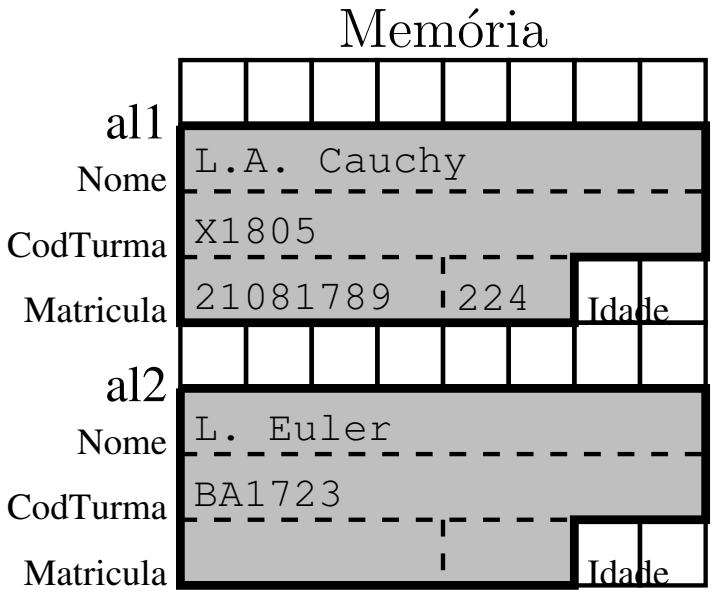
```
struct TAluno
{
  char Nome[250];
  char CodTurma[10];
  int Matricula;
  int Idade;
};

int main(void)
{
  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;
```



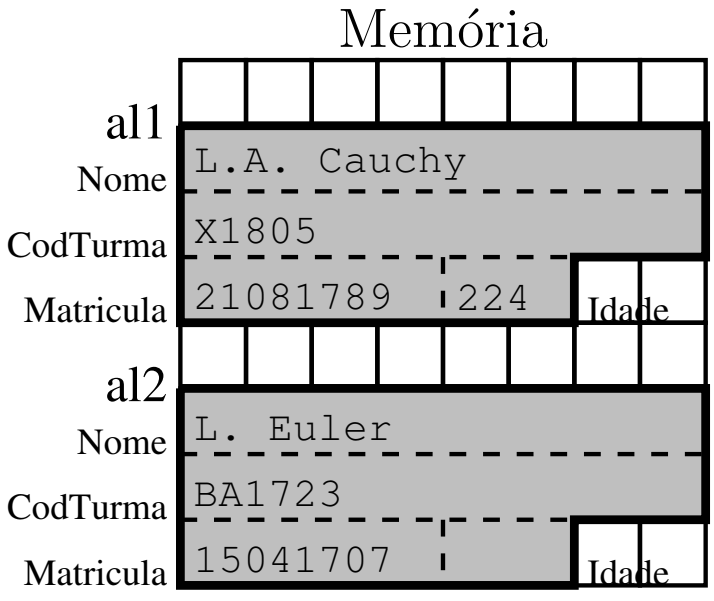
```
struct TAluno
{
  char Nome[250];
  char CodTurma[10];
  int Matricula;
  int Idade;
};

int main(void)
{
  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;
```



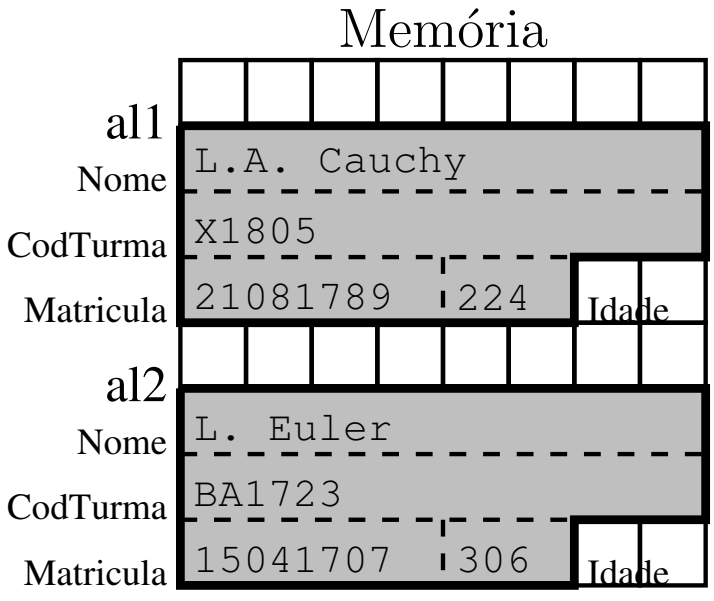
```
struct TAluno
{
  char Nome[250];
  char CodTurma[10];
  int Matricula;
  int Idade;
};

int main(void)
{
  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];
  int Matricula;
  int Idade;
};

int main(void)
{
  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);
- `CEP` (inteiro);

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];
    int matricula;
    int idade;
    struct TEndereco endereco;
};
```

ou

```
struct TAluno
{
    char nome[250];
    char codTurma[10];
    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];
    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;
```

Estrutura pode ser parâmetro de função

```
void imprimirAluno(struct TAluno a)
```

Estrutura pode ser retorno de função

```
struct TAluno preencherAluno(void)
```

typedef

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

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