

**Cetec**  
Capacitações

**CENTRO PAULA SOUZA**

GOVERNO DO ESTADO  
**SÃO PAULO**

GOVERNO FEDERAL  
**BRASIL**  
PAÍS RICO É PAÍS SEM POBREZA



# **Atualização Técnica e Pedagógica de Professores no componente de Lógica de Programação com C# (console)**

**Semana 5**

**Structs/Registros**

**Prof. Tiago Jesus de Souza**

## Introdução

Quando utilizamos vetor e/ou matrizes notamos que é possível armazenar vários dados, porém de apenas um determinado tipo para cada variável.

Com estruturas (structs) podemos armazenar vários dados de tipos diferentes de forma estruturada, ou seja, podemos criar uma estrutura com vários campos, sendo que cada campo poderá ter seu próprio tipo.

Fazendo uma analogia, em banco de dados podemos criar uma tabela com vários campos. Cada campo ter um tipo específico (string, real, inteiro,...). As informações que são armazenadas em cada linha da tabela são chamadas de registros. Para isso são usados sistemas gerenciadores de banco de dados.

Em Linguagem C#, é possível formar estes registros usando structs, onde iremos definir quais são os campos com seus respectivos tipos e depois podemos criar variáveis, vetores e matriz para armazenar informações que serão o registros.

## Sintaxe de uma Struct

Sintaxe para a criação de uma struct:

```
public struct <nome da struct>{  
    public <tipo> <variável 1>;  
    public <tipo> <variável 2>;  
    public <tipo> <variável 3>;  
    .  
    .  
    public <tipo> <variável N>;  
}
```

Onde o nome da struct deve seguir as mesmas regras para definição de variáveis (sem acentuação, sem espaços, iniciar com números e não conter caracteres especiais)

O tipo pode ser int, double, string, boolean entre outros ou até mesmo uma struct.

## Criando uma Struct

*Nome da struct*

```
public struct Pessoa{  
    public string nome;  
    public int idade;  
    public double altura;  
}
```

*Variáveis(campos)*

*Tipos*

Pessoa		
<i>nome</i>	<i>idade</i>	<i>altura</i>

Observando a estrutura criada, podemos verificar que quando declaramos uma variável que seja do tipo desta struct, esta variável poderá armazenar o nome, idade e altura de uma mesma pessoa.



## Declarando variável do tipo Struct

Para declararmos uma variável do tipo struct, seguimos as mesmas regras para definição de uma variável qualquer. A única diferença é que agora iremos declarar uma variável que é do tipo de uma struct foi criada pelo programador.

A declaração de uma variável é a seguinte:

***Pessoa p1;***

Onde p1 é a variável declarada que é do tipo Pessoa, que por ser uma struct, esta variável pode armazenar o nome, idade e altura de uma pessoa.

Podemos também declarar várias variáveis, por exemplo se quisermos armazenar os dados de 5 pessoas, podemos usar:

***Pessoa p1, p2, p3, p4, p5;***

## Atribuindo valores na variável do tipo Struct

A atribuição de valores para os campos de uma struct é realizada da seguinte forma:

```
p1.nome = "Camilo";  
p1.idade = 34;  
p1.altura = 1,72;
```

p1		
nome	idade	altura
Camilo	34	1,72

Notem que para acessar um campo da variável **p1**, devemos usar o **.** (ponto) e na sequencia o nome da variável (campo).

Caso o usuário digite a informação, utilizamos os mesmos comandos destinados a entrada de de dados, como por exemplo:

```
p2.nome = Console.ReadLine();  
p2.idade = int.Parse(Console.ReadLine());  
p2.altura = double.Parse(Console.ReadLine());
```

## Exemplo 1 – Uso de struct

Neste exemplo apenas irá armazenar os dados de duas pessoas e exibir as informações na tela.

```
public struct Pessoa
{
    public string nome;
    public int idade;
    public double altura;
}

static void Main(string[] args)
{
    Pessoa pessoa1, pessoa2;

    pessoa1.nome = "Fulana da Silva";
    pessoa2.nome = "Cicrano de Tal";

    pessoa1.idade = 30;
    pessoa2.idade = 65;

    pessoa1.altura = 1.51;
    pessoa2.altura = 1.84;

    Console.WriteLine("\n\n{0} com {1} anos de idade e mede: {2:N2}m de altura \n\n", pessoa1.nome, pessoa1.idade, pessoa1.altura);
    Console.WriteLine("\n\n{0} com {1} anos de idade e mede: {2:N2}m de altura \n\n", pessoa2.nome, pessoa2.idade, pessoa2.altura);

    Console.ReadKey();
}
```



## Exemplo 1 – Uso de struct - Resultado

Neste exemplo, apenas foram declaradas duas variáveis, onde foram atribuídos valores para seus respectivos campos e na sequência foram exibidos os dados.

```
file:///C:/Aulas C#/Aula5_exemplo1/Aula5_exemplo1/bin/Debug/Aula5_exemplo1.EXE  
  
Fulana da Silva com 30 anos de idade e mede: 1,51m de altura  
  
Cicrano de Tal com 65 anos de idade e mede: 1,84m de altura
```

## Exemplo 2 – Uso de struct com entrada de dados

Neste exemplo o usuário deverá digitar a data de nascimento (dia, mês ano) e depois exibir na tela.

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

static void Main(string[] args)
{
    Data dataNasc;

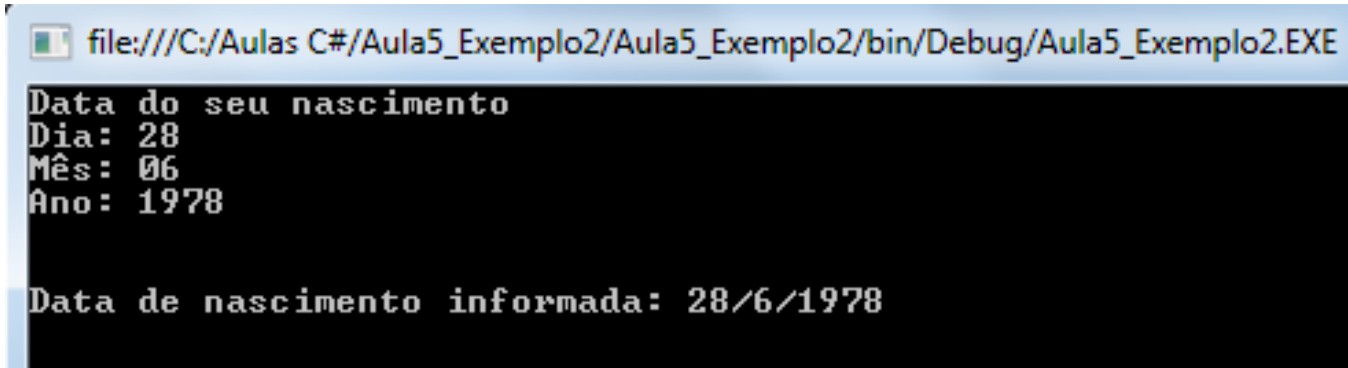
    Console.WriteLine("Data do seu nascimento");
    Console.Write("Dia: ");
    dataNasc.dia = int.Parse(Console.ReadLine());
    Console.Write("Mês: ");
    dataNasc.mes = int.Parse(Console.ReadLine());
    Console.Write("Ano: ");
    dataNasc.ano = int.Parse(Console.ReadLine());

    Console.WriteLine("\n\nData de nascimento informada: ");
    Console.WriteLine("{0}/{1}/{2} \n\n", dataNasc.dia, dataNasc.mes, dataNasc.ano);

    Console.ReadKey();
}
```

## Exemplo 2 – Uso de struct com entrada de dados - Resultado

Neste exemplo foi criada uma struct chamada Data, uma variável para armazenar a data de nascimento, onde o usuário teve que digitar o dia, mês e ano de nascimento e na sequencia foram exibidos os dados cadastrados.



```
file:///C:/Aulas C#/Aula5_Exemplo2/Aula5_Exemplo2/bin/Debug/Aula5_Exemplo2.EXE
Data do seu nascimento
Dia: 28
Mês: 06
Ano: 1978

Data de nascimento informada: 28/6/1978
```

## Vetor de registros

Nos exemplos anteriores, foram criadas variáveis para cada registro. No entanto, dependendo do problema a ser resolvido, onde exija um maior número de registros, devemos usar vetor.

Por exemplo, suponhamos que queremos armazenar os dados de 35 alunos de uma turma, o que pensamos inicialmente seria declarar variáveis aluno1, aluno2, aluno3, aluno4, aluno5, aluno6, ..., aluno34 e aluno35. Acho que já conseguem imaginar o trabalho que vai dar e quantidade de linhas de código apenas para a entrada dos dados desses alunos, correto? Imaginem então, se quisermos exibir somente os dados dos alunos com nota inferior a 7,0? Dessa forma, muitos programadores não teriam continuado com esta profissão.

Para solucionar este problema, utilizaremos um vetor de struct, onde apenas com uma variável, podemos armazenar vários registros.

## Criando uma struct Aluno

Criaremos uma struct chamada *Aluno* com os campos *nome*, *matricula* e *nota*.

```
public struct Aluno{  
    public string nome;  
    public int matricula;  
    public double nota;  
}
```

Aluno		
<i>nome</i>	<i>matricula</i>	<i>nota</i>

## Declarando vetor de registros do tipo Aluno

Para declararmos um vetor do tipo da struct Aluno, devemos usar a seguinte instrução:

```
Aluno[] alunos = new Aluno[35];
```

Desta forma está sendo criado um vetor Chamado **alunos** do tipo **Aluno** capaz de armazenar no máximo 35 registros de alunos.

Lembrando que o índice (posição) de um elemento no vetor inicia em 0 (zero).

alunos			
	<i>nome</i>	<i>matricula</i>	<i>nota</i>
0			
1			
2			
3			
4			
5			
...			
34			



## Atribuindo valores no vetor

Para atribuição de valores no vetor, seguimos as mesmas definições citadas anteriormente, porém agora devemos indicar em qual posição do vetor iremos armazenar os dados e em qual campo.

```
alunos[0].nome = "Camila"
```

```
alunos[0].matricula = 143;
```

```
alunos[0].nota = 7.5;
```

```
alunos[1].nome = "Adriano"
```

```
alunos[1].matricula = 231;
```

```
alunos[1].nota = 4.5;
```

E assim por diante...

alunos			
	<i>nome</i>	<i>matricula</i>	<i>nota</i>
0	Camila	143	7,5
1	Adriano	231	4,5
2			
3			
4			
5			
...			
34			

## Atribuindo valores no vetor usando estrutura de repetição

Usando ainda como base o vetor de alunos, imaginem que um usuário deverá cadastrar os dados dos 35 alunos. Se fossemos fazer um `Console.WriteLine` e `Console.ReadLine` para cada informação de cada aluno, teríamos muito trabalho, correto?

Portanto, neste caso, utilizaremos uma estrutura de repetição, uma vez que já é conhecido o número de registros que devemos cadastrar.

```
for (int x=0; x < 35; x++){  
    Console.WriteLine("\nNome do {0}º aluno: ", x+1);    alunos[x].nome = Console.ReadLine();  
    Console.WriteLine("Matrícula: ");    alunos[x].matricula = int.Parse(Console.ReadLine());  
    Console.WriteLine("Nota: ");    alunos[x].nota = double.Parse(Console.ReadLine());  
}
```

## Exibindo valores do vetor usando estrutura de repetição

Para automatizar o processo de exibição dos dados, também utilizaremos uma estrutura de repetição.

```
for (int x=0; x < 35; x++){  
    Console.WriteLine("\nNome do {0}º aluno: {1}", x+1, alunos[x].nome);  
    Console.WriteLine("Matrícula: {0}", alunos[x].matricula);  
    Console.WriteLine("Nota: {0:N1}", alunos[x].nota);  
}
```

*Obs.: Vale lembrar que neste caso sabemos a quantidade de elementos válidos no vetor.*

## Exemplo 3 – Vetor de struct

```
public struct Aluno
{
    public string nome;
    public int matricula;
    public double nota;
}

static void Main(string[] args)
{
    const int TAM = 5;
    Aluno[] alunos = new Aluno[TAM];

    Console.WriteLine("**** CADASTRANDO OS DADOS ****\n");
    for (int x=0; x < TAM; x++){
        Console.Write("\nNome do {0}º aluno: ", x+1);    alunos[x].nome = Console.ReadLine();
        Console.Write("Matrícula: ");                  alunos[x].matricula = int.Parse(Console.ReadLine());
        Console.Write("Nota: ");                        alunos[x].nota = double.Parse(Console.ReadLine());
    }

    Console.Clear();
    Console.WriteLine("**** EXIBINDO OS DADOS ****\n");
    for (int x=0; x < TAM; x++){
        Console.WriteLine("\nNome do {0}º aluno: {1}", x+1, alunos[x].nome);
        Console.WriteLine("Matrícula: {0}", alunos[x].matricula);
        Console.WriteLine("Nota: {0:N1}", alunos[x].nota);
    }
    Console.ReadKey();
}
```

## Exemplo 3 – Vetor de struct - Resultado

```
file:///C:/Aulas C#/Aula5_Exemplo3/Aula5_Exemplo3/bin/
**** CADASTRANDO OS DADOS ****

Nome do 1º aluno: Jun
Matrícula: 823
Nota: 8,2

Nome do 2º aluno: Eduardo
Matrícula: 746
Nota: 9,0

Nome do 3º aluno: Frank
Matrícula: 943
Nota: 9,5

Nome do 4º aluno: Jitsu
Matrícula: 245
Nota: 6,2

Nome do 5º aluno: Luis Fernando
Matrícula: 527
Nota: 5,5
```

```
file:///C:/Aulas C#/Aula5_Exemplo3/Aula5_Exemplo3/bin/D
**** EXIBINDO OS DADOS ****

Nome do 1º aluno: Jun
Matrícula: 823
Nota: 8,2

Nome do 2º aluno: Eduardo
Matrícula: 746
Nota: 9,0

Nome do 3º aluno: Frank
Matrícula: 943
Nota: 9,5

Nome do 4º aluno: Jitsu
Matrícula: 245
Nota: 6,2

Nome do 5º aluno: Luis Fernando
Matrícula: 527
Nota: 5,5
```

## Exemplo 4 – Passando registro por parâmetro

Neste exemplo será enviado um registro por parâmetro para o método *exibirDados*.

Este método irá receber o registro e irá exibir os dados que estão armazenados neste registro.

```
public struct Livro {
    public string titulo;
    public int ano;
    public float preco;
}

static void Main(string[] args) {
    Livro livro;

    Console.WriteLine("**** CADASTRANDO UM LIVRO ****\n");
    Console.Write("\nTítulo do livro: "); livro.titulo = Console.ReadLine();
    Console.Write("\nAno de Lançamento: "); livro.ano = int.Parse(Console.ReadLine());
    Console.Write("\nPreço: "); livro.preco = float.Parse(Console.ReadLine());

    Console.WriteLine("\n\n**** EXIBINDO OS DADOS DO LIVRO ****\n");
    exibirDados(livro);

    Console.ReadKey();
}

static void exibirDados(Livro l) {
    Console.WriteLine("\nTítulo do livro: {0}", l.titulo);
    Console.WriteLine("\nAno de Lançamento: {0}", l.ano);
    Console.WriteLine("\nPreço: R${0:N2}", l.preco);
}
```



## Exemplo 5 – Passando registro por parâmetro - Resultado

Com o resultado apresentado, nota-se que a sequência que foram digitadas as informações de um livro e na sequência foram exibidos todos os dados desse livro.

```
file:///C:/Aulas C#/Aula5_Exemplo4/Aula5_Exemplo4/bin/Debug/Aula5_Exemplo4.EXE
**** CADASTRANDO UM LIVRO ****

Título do livro: Lógica de Programação
Ano de Lançamento: 2013
Preço: 54,99

**** EXIBINDO OS DADOS DO LIVRO ****

Título do livro: Lógica de Programação
Ano de Lançamento: 2013
Preço: R$54,99
```

## Exemplo 5 – Método que retorna um registro

Neste exemplo foi criado um Método chamado *cadastrarLivro*, que será responsável por realizar a entrada de dados e retornar todos os dados cadastrados para o método principal.

Na sequência os dados serão exibidos.

```
public struct Livro
{
    public string titulo;
    public int ano;
    public float preco;
}

static void Main(string[] args)
{
    Livro livro;

    livro = cadastrarLivro();
    exibirDados(livro);

    Console.ReadKey();
}

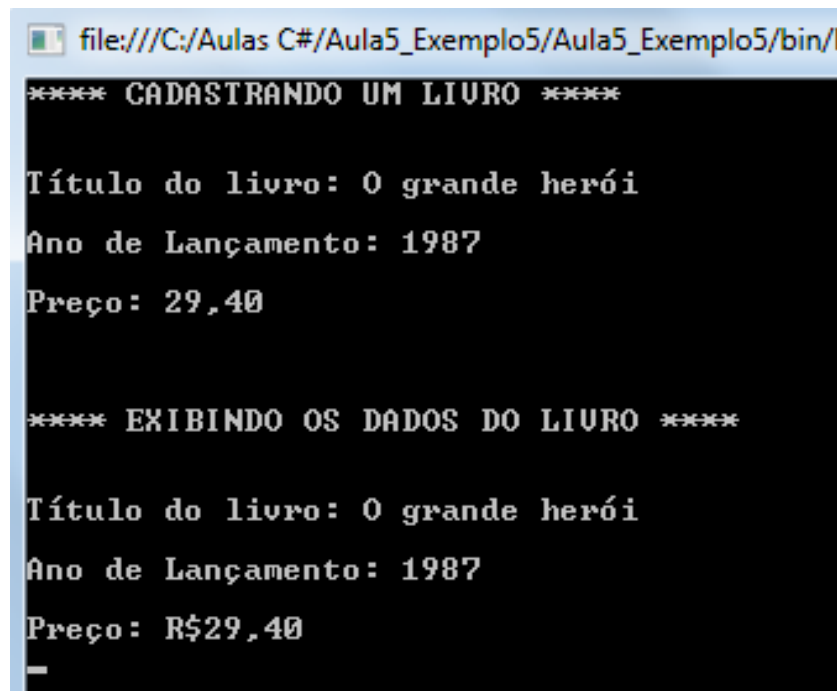
static Livro cadastrarLivro()
{
    Livro l;
    Console.WriteLine("**** CADASTRANDO UM LIVRO ****\n");
    Console.Write("\nTítulo do livro: "); l.titulo = Console.ReadLine();
    Console.Write("\nAno de Lançamento: "); l.ano = int.Parse(Console.ReadLine());
    Console.Write("\nPreço: "); l.preco = float.Parse(Console.ReadLine());
    return l;
}

static void exibirDados(Livro l)
{
    Console.WriteLine("\n\n**** EXIBINDO OS DADOS DO LIVRO ****\n");
    Console.WriteLine("\nTítulo do livro: {0}", l.titulo);
    Console.WriteLine("\nAno de Lançamento: {0}", l.ano);
    Console.WriteLine("\nPreço: R${0:N2}", l.preco);
}
```

## Exemplo 5 – Método que retorna um registro - Resultado

O método que realiza o cadastro de livro, possui uma variável auxiliar do tipo Livro para armazenar os dados digitados pelo usuário.

Logo após todas as entradas serem realizadas pelo usuário, este método *cadaststrarLivro* retornará para o método principal um registro com todas as informações de um livro, cujos valores serão atribuídos em outra variável (livro).



```
file:///C:/Aulas C#/Aula5_Exemplo5/Aula5_Exemplo5/bin/1
***** CADASTRANDO UM LIVRO *****
Título do livro: O grande herói
Ano de Lançamento: 1987
Preço: 29,40
***** EXIBINDO OS DADOS DO LIVRO *****
Título do livro: O grande herói
Ano de Lançamento: 1987
Preço: R$29,40
_
```

## Vetor de registros

Como na maioria dos casos temos que armazenar um determinado número de registros, ou seja, por exemplo armazenar várias informações de várias pessoas, podemos unir as duas técnicas já apresentadas, surgindo assim um vetor de registros.

Como exemplo, usaremos como base a struct do exemplo 5 (Livro).

```
Livro[ ] livros = new Livro[5];
```

Índices/Posições

livros			
	<i>titulo</i>	<i>ano</i>	<i>preco</i>
0	Lógica em C#	2013	89,90
1	Flash CS4	2012	55,90
2	Guia CSS	2000	22,90

Conteúdo do registro da linha 2

## Exemplo 6 – Cadastro e exibição de dados usando vetor de registros

Neste exemplo, o objetivo é cadastrar 3 registros de livros e exibir todos os dados dos livros cadastrados

Observem que foi necessária a declaração de apenas 1 variável para armazenar os 6 livros, sendo que cada livro possui 3 informações.

```
public struct Livro
{
    public string titulo;
    public int ano;
    public float preco;
}

static void Main(string[] args)
{
    const int TAM = 3;
    Livro[] livros = new Livro[TAM];

    Console.WriteLine("\n**** CADASTRANDO LIVROS ****\n");
    for (int x = 0; x < TAM; x++)
        livros[x] = cadastrarLivro();

    Console.WriteLine("\n\n**** EXIBINDO OS DADOS DOS LIVROS ****\n");
    exibirDados(livros);
    Console.ReadKey();
}

static Livro cadastrarLivro()
{
    Livro l;
    Console.Write("\ntítulo do livro: "); l.titulo = Console.ReadLine();
    Console.Write("Ano de Lançamento: "); l.ano = int.Parse(Console.ReadLine());
    Console.Write("Preço: "); l.preco = float.Parse(Console.ReadLine());
    return l;
}

static void exibirDados(Livro[] l)
{
    for (int x = 0; x < l.Length; x++)
    {
        Console.WriteLine("\ntítulo do livro: {0}", l[x].titulo);
        Console.WriteLine("\tano de Lançamento: {0}", l[x].ano);
        Console.WriteLine("\tPreço: R${0:N2}", l[x].preco);
    }
}
```

## Exemplo 6 – Cadastro e exibição de dados usando vetor de registros

Foram digitados os dados de todos os livros na sequência todas as informações de todos os livros foram exibidas na tela

Podemos notar que o método cadastrarLivro retornará um registro que será armazenado na posição  $x$  do vetor (no método principal-*main*).

E para exibir os dados, foi passado o vetor de livros por parâmetro para o método exibirDados exibir as informações de todos os registros

```
file:///C:/Aulas C#/Aula5_Exemplo6/Aula5_Exemplo6/bin,
**** CADASTRANDO LIVROS ****
Título do livro: O Segredo
Ano de Lançamento: 2005
Preço: 39,00
Título do livro: Cebolinha
Ano de Lançamento: 1980
Preço: 25,90
Título do livro: Doki
Ano de Lançamento: 2010
Preço: 87,89
**** EXIBINDO OS DADOS DOS LIVROS ****
Título do livro: O Segredo
Ano de Lançamento: 2005
Preço: R$39,00
Título do livro: Cebolinha
Ano de Lançamento: 1980
Preço: R$25,90
Título do livro: Doki
Ano de Lançamento: 2010
Preço: R$87,89
```



## Matriz de registros

Também é possível criar uma matriz de registros. Sendo assim, em cada posição da matriz é possível armazenar vários dados de tipos diferentes.

Livro[ , ] livros = new Livro[5,4];

Nesta matriz existe 20 (5x4) estando em posições para armazenar em cada posição um livro, com todas suas respectivas informações.

Imaginem uma estante com essas dimensões/repartições e um livro estando colocado em cada posição desta prateleira, é dessa forma que podemos visualizar.

## Exemplo 7 – Cadastrando e Exibindo matriz de registros

Neste exemplo, no método Main, que está realizando chamadas aos métodos cadastrarLivro e exibirDados.

Logo após o usuário digitar os dados dos 6 livros, serão exibidas todas as informações destes livros no formato visual de uma matriz.

```
public struct Livro
{
    public string titulo;
    public int ano;
    public float preco;
}

static void Main(string[] args)
{
    const int NUM_L = 3, NUM_C = 2;
    Livro[,] livros = new Livro[NUM_L, NUM_C];

    Console.WriteLine("\n**** CADASTRANDO LIVROS ****\n");
    for (int l = 0; l < NUM_L; l++)
    {
        for (int c = 0; c < NUM_C; c++)
        {
            livros[l,c] = cadastrarLivro();
        }
    }
    Console.Clear();
    Console.WriteLine("\n**** EXIBINDO OS DADOS DOS LIVROS ****\n");
    exibirDados(livros);
    Console.ReadKey();
}
```

## Exemplo 7 – Cadastrando e Exibindo matriz de registros - *Continuação*

```
static Livro cadastrarLivro()
{
    Livro l;
    Console.WriteLine("\nTítulo do livro: "); l.titulo = Console.ReadLine();
    Console.WriteLine("Ano de Lançamento: "); l.ano = int.Parse(Console.ReadLine());
    Console.WriteLine("Preço: "); l.preco = float.Parse(Console.ReadLine());
    return l;
}

static void exibirDados(Livro[,] lv)
{
    int px=3, py=3; //variáveis para controlar a posição do cursor na tela
    for (int l = 0; l < lv.GetLength(0); l++)
    {
        for (int c = 0; c < lv.GetLength(1); c++)
        {
            Console.SetCursorPosition(px, py+1);
            Console.WriteLine("\tTítulo: {0}", lv[l,c].titulo);
            Console.SetCursorPosition(px, py+2);
            Console.WriteLine("\tAno: {0}", lv[l,c].ano);
            Console.SetCursorPosition(px, py+3);
            Console.WriteLine("\tPreço: R${0:N2}", lv[l,c].preco);
            px += 25; //para deixar um espaço a direita p/ proximo registro
        }
        px = 0; //quando pular de linha retonar na posição inicial
        py += 5; //quando pular de linha, pular 5 linhas a partir da posição atual
    }
}
```

## Exemplo 7 – Cadastrando e Exibindo matriz de registros-Resultado

A lado esquerdo temos o resultado apresentando as informações digitadas pelo usuário.

Abaixo temos o resultado da apresentação dos dados.

```
file:///C:/Aulas C#/Aula5_Exemplo7/Aula5_E
**** CADASTRANDO LIVROS ****

Título do livro: Livro 1
Ano de Lançamento: 2001
Preço: 20,00

Título do livro: Livro 2
Ano de Lançamento: 2002
Preço: 30,00

Título do livro: Livro 3
Ano de Lançamento: 2003
Preço: 40,00

Título do livro: Livro 4
Ano de Lançamento: 2004
Preço: 50,00

Título do livro: Livro 5
Ano de Lançamento: 2005
Preço: 60,00

Título do livro: Livro 6
Ano de Lançamento: 2006
Preço: 70,00
```

```
file:///C:/Aulas C#/Aula5_Exemplo7/Aula5_Exemplo7/bin/Debug/Aula5_Exemplo7.EXE
**** EXIBINDO OS DADOS DOS LIVROS ****

Título: Livro 1      Título: Livro 2
Ano: 2001           Ano: 2002
Preço: R$20,00     Preço: R$30,00

Título: Livro 3      Título: Livro 4
Ano: 2003           Ano: 2004
Preço: R$40,00     Preço: R$50,00

Título: Livro 5      Título: Livro 6
Ano: 2005           Ano: 2006
Preço: R$60,00     Preço: R$70,00
```

## Bibliografia

- MSDN, Microsoft. **Guia de Programação C#**. Disponível:  
< <http://msdn.microsoft.com/pt-br/library/vstudio/ah19swz4.aspx> >. Acesso em 23 abr 2013  
  
< [http://msdn.microsoft.com/en-us/library/aa288471\(v=vs.71\).aspx](http://msdn.microsoft.com/en-us/library/aa288471(v=vs.71).aspx) >. Acesso em 25 abr 2013
- <http://www.dotnetperls.com/struct>