

**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 7**

**Gravação e Leitura de Dados em Arquivo Texto**

**Prof. Tiago Jesus de Souza**

## Introdução

Dados/Informações são muito importantes para todos nós, mas fundamental para as empresas.

Tratamos todos os exercícios até o momento, solicitando para que o usuário digitasse as informações e quando finalizava o programa, tudo era perdido.

Mas agora, no material desta semana, será mostrado como armazenar informações em arquivo e depois recuperá-las.

O C# utiliza o *Stream* para ler e escrever em arquivos. Sempre que uma aplicação lê ou escreve em arquivos ou liga a outro computador numa rede, envia e recebe bytes.

Assim, sempre que se pretender ler ou escrever dados num arquivo, deve ser utilizado ou criado um objeto *Stream*.

## Streams

Um *stream* é uma representação abstrata de um dispositivo serial.

Um dispositivo como uma impressora, um local da memória ou qualquer objeto que suporte a leitura e a escrita nele de modo linear.

## Classe Stream - Escrita

A classe Stream está disponível na biblioteca IO, portanto é necessário incluir no arquivo da classe que será utilizada: **Using System.IO;**

Para criar um objeto do tipo *writer* (escrita) devemos fazer:

```
StreamWriter wr = new StreamWriter(@"c:\pasta\arquivo.txt", true);
```

Colocar o **@** antes do path do arquivo faz com que o compilador não interprete a barra “\” como sendo uma mudança de linha ou tabulação (\n ou \t), e sim como uma string.

A seguir podemos escrever no arquivo a função de escrita:

```
wr.WriteLine("esta frase será gravada dentro do arquivo");
```

Não se esquecer de fechar o arquivo no final da escrita para que não fique aberto a outros programas:

```
wr.Close();
```



## Classe Stream - Leitura

Para a leitura de arquivos utiliza-se o *StreamReader*:

```
StreamReader rd = new StreamReader(@"c:\pasta\arquivo.txt");
```

Para melhor exemplificar, vamos criar um arquivo onde se escreverá duas linhas e, depois de escrito, vai ler essas duas linhas.

No ciclo while, irá ler todas as linhas enquanto estas tiverem algo escrito (*EndOfStream*).

Tal como na escrita, também na leitura devemos fechar o arquivo com *Close()*.

```
StreamWriter wr = new  
StreamWriter(@"c:\pasta\doc.txt", true);  
wr.WriteLine("Primeira linha");  
wr.WriteLine("Segunda linha");  
wr.Close();  
StreamReader rd = new  
StreamReader(@"c:\pasta\doc.txt");  
while(!rd.EndOfStream){  
    string linha = rd.ReadLine();  
    Console.WriteLine(linha);  
}  
rd.Close();
```

## Classes File e Directory

### Métodos úteis da classe "File":

Método	Descrição
Create ( )	Criar um arquivo em um determinado caminho
Open ( )	FileStrem no caminho especificado
Copy( )	Copiar arquivo em um local específico

### Métodos úteis da classe "Directory"

Método	Descrição
Delete ( )	Exclui o diretório e todos os arquivos nele contido
Move ( )	Mover um diretório para um novo local
GetFiles( )	Retorna uma array de objetos File

## Classes File e FileInfo

O objeto FileInfo é como se fosse um arquivo em um disco.

Para escrevermos ou lermos um arquivo, é necessário criarmos um objeto Stream.

Vamos criar um objeto FileInfo, indicando o nome do arquivo e seu diretório.

```
FileInfo aFile = new FileInfo ("c:/arquivo.txt");
```

A classe *File* requer um parâmetro de string especificando a localização do arquivo para cada chamada de método.

As duas chamadas efetuam o mesmo resultado.

```
FileInfo aFile = new FileInfo("teste.txt");  
  
If (aFile.Exists)  
    Console.WriteLine("Esse arquivo existe");
```

```
If (File.Exists("teste.txt"))  
    Console.WriteLine("Esse arquivo existe");
```



## Classes File e FileInfo

Se você estiver realizando uma única chamada de método do objeto, é recomendado que utilize a classe estática *File*.

Caso o seu aplicativo estiver realizando várias operações em um arquivo, o ideal seria instanciar o objeto *FileInfo* e usar os seus métodos, economizando mais tempo porque o objeto já estará referenciando o arquivo correto no sistema de arquivos, enquanto uma classe estática terá de encontrá-lo a cada vez.

## Arquivo Texto

A criação de um arquivo texto em um projeto é interessante quando não existe uma importância com a estrutura dos dados a serem armazenados. Mas se por exemplo, tivéssemos que armazenar dados de um banco de dados que possui registros, com seus campos de tipos pré-definidos, não deveríamos criar um arquivo neste formato.

Este tipo de arquivo é muito usado para gravação de logs de sistemas, por exemplo, gravar o nome e horário em que um usuário realizou seu login no sistema, ou gravar em um arquivo texto algumas informações que gerem um texto informativo. Sem a preocupação de lidar com a estrutura.

## Exemplo 1 – Trabalhando com arquivo texto

Neste exemplo, foi criado um diretório para armazenar o Arquivo.

Depois foram inseridas frases neste arquivo, e na sequência foram Exibidas as informações contidas no arquivo .txt

```
static void Main(string[] args)
{
    //Instrução para a criação do diretório Teste
    Directory.CreateDirectory(@"c:\Teste\");

    //Cria um objeto para manipular a escrita no arquivo
    StreamWriter wr = new StreamWriter(@"c:\Teste\Exemplo1.txt", true);
    wr.WriteLine("Primeira linha");
    wr.WriteLine("Segunda linha");
    wr.Close();

    //Cria um objeto para manipular a leitura do arquivo
    StreamReader rd = new StreamReader(@"c:\Teste\Exemplo1.txt");
    while (!rd.EndOfStream)
    {
        string linha = rd.ReadLine();
        Console.WriteLine(linha);
    }
    rd.Close();

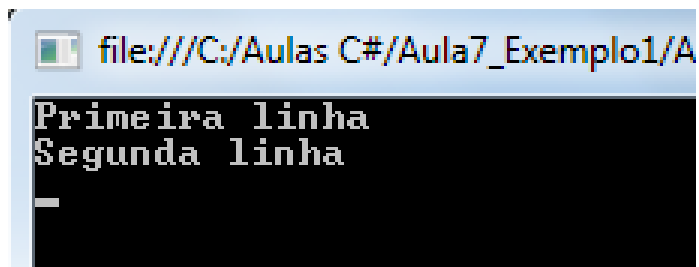
    Console.ReadKey();
}
```

## Exemplo 1 – Trabalhando com arquivo texto - Resultado

Neste caso, já foi predefinido na linha de código os textos para que fossem inseridos no arquivo teste.txt.

Mas poderia ser qualquer informação digitada pelo usuário, que seria armazenada em uma variável e depois envia-la para o arquivo.

A forma como usar, vai depender da proposta do exercício.



```
file:///C:/Aulas C#/Aula7_Exemplo1/A
Primeira linha
Segunda linha
-
```

## Exemplo 2 – Trabalhando com arquivo texto – outra forma

Neste exemplo, foi utilizado outro método para que seja realizado a gravação dos dados no arquivo. O resultado é a criação do arquivo Exemplo2.txt. Verifique na diretório.

```
static void Main(string[] args)
{
    //Primeiro devemos escolher o nome do arquivo e onde iremos cria-lo
    String nome_arquivo = "c:\\\\Teste\\Exemplo2.txt";

    //Depois devemos verificar se o arquivo existe, se não existir então criamos o arquivo
    if (!System.IO.File.Exists(nome_arquivo))
        System.IO.File.Create(nome_arquivo).Close();

    //Agora usamos a classe TextWriter para escrever no arquivo
    System.IO.TextWriter arquivo = System.IO.File.AppendText(nome_arquivo);
    arquivo.WriteLine("Escrevendo no arquivo");

    //Fechamos o arquivo
    arquivo.Close();
}
```

## Exemplo 3 – Leitura de arquivo texto

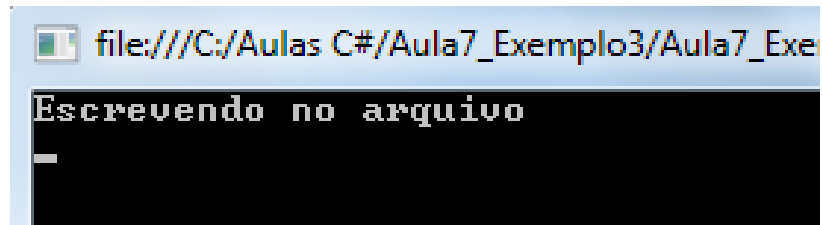
Neste exemplo, está sendo realizada a leitura dos dados armazenados no arquivo txt.

```
static void Main(string[] args) {  
  
    string caminho = "C:\\\\Teste\\\\Exemplo2.txt";  
  
    // Cria Objetos  
    System.IO.StreamReader sr;  
    string linhaAtual;  
  
    // Verifica se o Arquivo não Existe  
    if (!System.IO.File.Exists(caminho)) {  
        throw (new System.IO.FileNotFoundException("Não foi Possível Localizar o Arquivo Especificado"));  
    }  
  
    // Inicializa o StreamReader  
    using (sr = new System.IO.StreamReader(caminho)) {  
        while (!sr.EndOfStream)  
        {  
            // Recupera a Linha  
            linhaAtual = sr.ReadLine();  
            Console.WriteLine(linhaAtual);  
        }  
    }  
    Console.ReadKey();  
}
```



## Exemplo 3 – Leitura de arquivo texto - Resultado

O resultado apresentado de refere ao conteúdo do arquivo Exemplo2.txt, criado anteriormente.



```
file:///C:/Aulas C#/Aula7_Exemplo3/Aula7_Exe
Escrevendo no arquivo
-
```

## Arquivo Binário

- Um arquivo binário tem uma importante característica de não guardar somente valores do tipo texto, mas de outros tipos.
- Basta definir qual vai ser a sua estrutura de armazenamento a qualquer hora e momento.
- No primeiro momento esses arquivos são rápidos, pois, dependendo de seu tipo armazenado dentro não necessita conversão, como acontece muito com valores textos.
- Ideal para trabalhar com structs e objetos.

## Exemplo 4 –Adicionando dados em um Arquivo Binário

Neste exemplo, apenas foram atribuídos valores nas variáveis e na sequencia gravados em arquivo binário.

```
static void Main(string[] args)
{
    // Instanciando as variáveis
    int idade = 25;
    string nome = "Sérgio";
    bool temFilhos = false;
    double altura = 1.79;

    // criando o arquivo
    FileStream stream = new FileStream(@"C:\\Teste\\Exemplo4.bin", FileMode.Create);

    // instanciando a variável do tipo BinaryWriter
    BinaryWriter writer = new BinaryWriter(stream);

    // usando o método Write para escrever no arquivo.
    writer.Write(idade);
    writer.Write(nome);
    writer.Write(temFilhos);
    writer.Write(altura);

    writer.Flush();
    writer.Close();
}
```

## Exemplo 5 – Recuperando dados de um Arquivo Binário

Neste exemplo, os valores que foram atribuídos no arquivo do exemplo anterior, foram trazidos para este exemplo, armazenando os valores organizadamente nas variáveis e na sequência sendo exibidos.

```
static void Main(string[] args)
{
    // instanciando a variavel stream
    FileStream stream =
        new FileStream(@"C:\\Teste\\Exemplo4.bin", FileMode.Open);

    // instanciando a variável do tipo BinaryReader
    BinaryReader reader = new BinaryReader(stream);

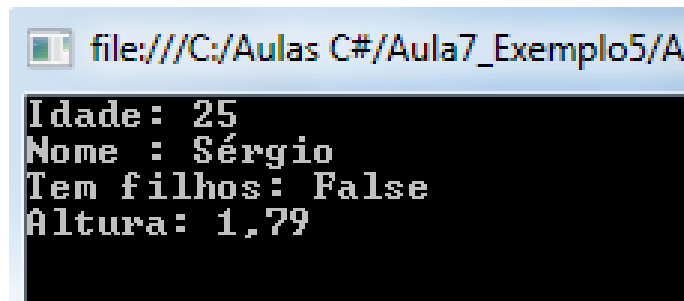
    // variáveis recebendo os valores lidos pelo arquivo binário
    int idade = reader.ReadInt32();
    string nome = reader.ReadString();
    bool temFilhos = reader.ReadBoolean();
    double altura = reader.ReadDouble();

    // fecha o acesso com o arquivo
    reader.Close();

    // exibe o resultado
    Console.WriteLine("Idade: {0}", Convert.ToString(idade));
    Console.WriteLine("Nome : {0}", nome);
    Console.WriteLine("Tem filhos: {0}", Convert.ToString(temFilhos));
    Console.WriteLine("Altura: {0}", Convert.ToString(altura));
    Console.ReadKey();
}
```

## Exemplo 5 – Recuperando dados de um Arquivo Binário - Resultado

Analisando o resultado, notem que as informações foram recuperadas adequadamente de acordo com o seu tipo, ou seja, foram gravadas no arquivo de forma que conseguissem ser recuperadas quando solicitadas.



```
file:///C:/Aulas C#/Aula7_Exemplo5/A
Idade: 25
Nome : Sérgio
Tem filhos: False
Altura: 1,79
```

## Exemplo 6 – Menu Básico com Arquivo Binário

Neste exemplo, estaremos usando uma struct como base de registro para ser armazenado no arquivo, laço de repetição para sair do programa somente quando usuário digitar a opção 0 (zero).

No menu existem as opções para Cadastrar Pessoas e outra para Listar Pessoas.

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

static void Main(string[] args)
{
    Pessoa p;
    int op = 0;

    do{
        Console.Clear();
        Console.WriteLine("1 - Cadastrar dados");
        Console.WriteLine("2 - Listar dados");
        Console.WriteLine("0 - Sair");
        Console.Write("Digite a opção desejada: ");
        op = int.Parse(Console.ReadLine());

        switch (op) {
```



## Exemplo 6 – Menu Básico com Arquivo Binário - *Continuação*

Opção responsável por realizar o cadastro e gravar o registro em arquivo.

```
case 1:
    // instanciando a variavel stream
    FileStream streamCadastrar = new FileStream(@"C:\\Teste\\Exemplo6.bin", FileMode.Append);

    // instanciando a variável do tipo BinaryWriter
    BinaryWriter writer = new BinaryWriter(streamCadastrar);
    Console.WriteLine("\nDigite o nome: ");
    p.nome = Console.ReadLine();
    Console.WriteLine("Digite a idade: ");
    p.idade = int.Parse(Console.ReadLine());
    Console.WriteLine("Digite a altura: ");
    p.altura = double.Parse(Console.ReadLine());
    writer.Write(p.nome);
    writer.Write(p.idade);
    writer.Write(p.altura);
    writer.Flush();
    writer.Close();

break;
```

## Exemplo 6 – Menu Básico com Arquivo Binário - *Continuação*

Opção responsável por abrir o arquivo e exibir os dados dos registros do arquivo.

```
case 2:
    FileStream streamListar = new FileStream(@"C:\\Teste\\Exemplo6.bin", FileMode.Open);

    // instanciando a variável do tipo BinaryReader
    BinaryReader reader = new BinaryReader(streamListar);

    // o peekChar verifica o próximo byte do arquivo sem avançar o cursor,
    // se não houver mais bytes para serem lidos, ele devolve valor negativo
    while(reader.PeekChar() > -1)
    {
        // variáveis recebendo os valores lidos pelo arquivo binário
        p.nome = reader.ReadString();
        p.idade = reader.ReadInt32();
        p.altura = reader.ReadDouble();
        Console.WriteLine("\n{0} - {1} - {2}", p.nome, p.idade, p.altura);
    }

    // fecha o acesso com o arquivo
    reader.Close();
    break;
```

## Exemplo 6 – Menu Básico com Arquivo Binário - *Continuação*

Opção responsável por sair do programa e caso o usuário digitar qualquer opção diferente das indicadas, é informado ao usuário.

E somente finalizará o programa, quando o usuário digitar 0 (zero).

```
        case 0:
            break;

        default: Console.WriteLine("Opção inválida");
            break;
    }
    Console.WriteLine("\n\nPressione alguma tecla para continuar!!");
    Console.ReadKey();
}while(op!=0);
}
```

## Exemplo 6 – Menu Básico com Arquivo Binário - Resultado

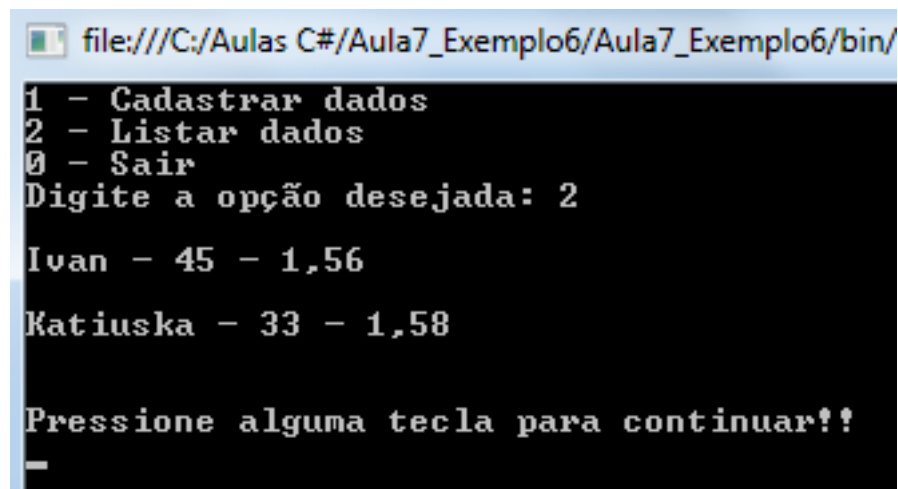
Telas representando a realização de dois cadastros de pessoas.

```
file:///C:/Aulas C#/Aula7_Exemplo6/Aula7_Exemplo6/bin/  
1 - Cadastrar dados  
2 - Listar dados  
0 - Sair  
Digite a opção desejada: 1  
  
Digite o nome: Ivan  
Digite a idade: 45  
Digite a altura: 1,56  
  
Pressione alguma tecla para continuar!!  
_
```

```
file:///C:/Aulas C#/Aula7_Exemplo6/Aula7_Exemplo6/bin,  
1 - Cadastrar dados  
2 - Listar dados  
0 - Sair  
Digite a opção desejada: 1  
  
Digite o nome: Katiuska  
Digite a idade: 33  
Digite a altura: 1,58  
  
Pressione alguma tecla para continuar!!  
_
```

## Exemplo 6 – Menu Básico com Arquivo Binário - Resultado

Telas apresentando a listagem de todos os cadastros que estão no arquivo.



```
file:///C:/Aulas C#/Aula7_Exemplo6/Aula7_Exemplo6/bin/  
1 - Cadastrar dados  
2 - Listar dados  
0 - Sair  
Digite a opção desejada: 2  
  
Ivan - 45 - 1,56  
  
Katuska - 33 - 1,58  
  
Pressione alguma tecla para continuar!!  
-
```

Se na sequencia for realizado o cadastro de mais pessoas, as informações serão adicionadas no arquivo e uma nova listagem atualizada será apresentada, quando for selecionada a opção 2.

## Exemplo 7 – Trabalhando com métodos, vetor e arquivo binário

Este exemplo demonstra a possibilidade de trabalhar com vetor para carregar todos os registros que estão no arquivo, evitando assim ficar abrindo e fechando o arquivo a cada operação que é realizada.

E também facilita na manipulação das informações.

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

//método principal
static void Main(string[] args)
{
    Pessoa[] p = new Pessoa[500];
    int op = 0;
    //x é o contador oficial da quantidade de registros cadastrados
    int x = carregarDados(p);

    do
    {
        op = menu();
        switch (op)
        {
            case 1:
                p[x] = entradaDados(); //armazenando um registro no vetor
                cadastrar(p[x]); //chamando o método para gravar registro no arquivo
                x++;
                break;

            case 2:
                Console.WriteLine("\n **** Listagem **** ");
                for (int i = 0; i < x; i++)
                    exibirDados(p[i]);
                break;

            case 0:
                break;

            default: Console.WriteLine("Opção inválida");
                break;
        }
        pausar();
    } while (op != 0);
}
```



## Exemplo 7 – Trabalhando com métodos, vetor e arquivo binário - *Continuação*

Fazendo a criação de métodos, o código do método principal fica mais limpo (claro) e a estruturação do código fica mais organizada.

```
//método responsável por exibir o menu, ler e retornar a opção escolhida
static int menu()
{
    Console.Clear();
    Console.WriteLine("1 - Cadastrar dados");
    Console.WriteLine("2 - Listar dados");
    Console.WriteLine("0 - Sair");
    Console.Write("\nDigite a opção desejada: ");
    return int.Parse(Console.ReadLine());
}

//método responsável por exibir os dados de um registro
static void exibirDados(Pessoa aux)
{
    Console.WriteLine("\n" + aux.nome + "\t");
    Console.WriteLine(aux.idade + "\t");
    Console.WriteLine(aux.altura);
}

//método responsável por ler um registro e retorna-lo para o main
static Pessoa entradaDados()
{
    Pessoa aux;

    Console.WriteLine("\nDigite o nome: ");
    aux.nome = Console.ReadLine();
    Console.WriteLine("Digite a idade: ");
    aux.idade = int.Parse(Console.ReadLine());
    Console.WriteLine("Digite a altura: ");
    aux.altura = double.Parse(Console.ReadLine());

    return aux;
}
```

## Exemplo 7 – Trabalhando com métodos, vetor e arquivo binário

– *Continuação*

```
static void pausar()
{
    Console.WriteLine("\n\nPressione alguma tecla para continuar!!");
    Console.ReadKey();
}

//método responsável por cadastrar o registro no arquivo
static void cadastrar(Pessoa aux)
{
    // instanciando a variavel stream
    FileStream streamCadastrar = new FileStream(@"C:\\Teste\\Exemplo7.bin", FileMode.Append);

    // instanciando a variável do tipo BinaryWriter
    BinaryWriter writer = new BinaryWriter(streamCadastrar);

    writer.Write(aux.nome);
    writer.Write(aux.idade);
    writer.Write(aux.altura);
    writer.Flush();
    writer.Close();
}
```

## Exemplo 7 – Trabalhando com métodos, vetor e arquivo binário

– *Continuação*

```
//método responsável por carregar os registros que estão no arquivo, em um vetor
static int carregarDados(Pessoa[] p)
{
    FileStream streamListar;

    //Se o arquivo não existir, cria o arquivo, senão, apenas abre o arquivo
    if (!File.Exists(@"C:\\Teste\\Exemplo7.bin"))
        streamListar = new FileStream(@"C:\\Teste\\Exemplo7.bin", FileMode.Create);
    else
        streamListar = new FileStream(@"C:\\Teste\\Exemplo7.bin", FileMode.Open);

    //serve de contador de registros
    int cont = 0;

    // instanciando a variável do tipo BinaryReader
    BinaryReader reader = new BinaryReader(streamListar);

    while (reader.PeekChar() > -1)
    {
        // variáveis recebendo os valores lidos pelo arquivo binário
        p[cont].nome = reader.ReadString();
        p[cont].idade = reader.ReadInt32();
        p[cont].altura = reader.ReadDouble();
        cont++;
    }
    // fecha o acesso com o arquivo
    reader.Close();
    return cont;
}
```

## Bibliografia

- MSDN, Microsoft. **Guia de Programação C#**. Disponível:  
< <http://msdn.microsoft.com/pt-br/library/vstudio/6ka1wd3w.aspx> >. *Acesso em 26 abr 2013*
- < <http://msdn.microsoft.com/pt-br/library/system.io.file.aspx> >. *Acesso em 28 abr 2013*
- < <http://msdn.microsoft.com/en-us/library/system.io.binaryreader.peekchar.aspx> >. *Acesso em 28 abr 2013*
- < <http://www.tiagolemos.com.br/2009/01/22/c-para-iniciantes-criar-e-escrever-em-um-arquivo-txt/> >. *Acesso em 01 mai 2013*
- < <http://imasters.com.br/artigo/12197/dotnet/leitura-e-escrita-em-arquivos-com-c/> >. *Acesso em 01 mai 2013*
- < <http://www.linhadecodigo.com.br/artigo/347/crie-um-arquivo-txt-e-acrescente-valores-usando-csharp.aspx> >. *Acesso em 01 mai 2013*