PROGRAMAÇÃO VISUAL

Prof. Alexandre Krohn

PROJETO DETECÇÃO DE CARGA

Objetivo

- Processar um arquivo com dados de célula de carga

Arquivo

- Arquivo texto com 8 colunas: **data (**formato aaaammdd), **hora** (formato hhMMssmmm), **dado1** (float), **dado2** (float), **dado3** (float), **dado4** (float), **dado5** (float) e **dado6** (float)

CRIAÇÃO DO PROJETO NO VS 2017

Na barra de menus, escolha Arquivo, Novo, Projeto.

- Selecionar Visual C# Windows Forms Application (.NET Framework)
- Atribuir um nome ao projeto: SensorCarga
  - A solução (conjunto de projetos) terá o mesmo nome

Ao criar o projeto, o Visual Studio (VS) gera automaticamente 3 arquivos

- Form1.cs: contém o código que define o comportamento do formulário
- Form1.Designer.cs: contém o código gerado pelos Windows Form Designer que define os componentes visuais do formulário
- Program.cs: código que inicializa o programa e mostra o formulário

CRIAÇÃO DO PROJETO NO VS 2017

```
1  using System;
2  using System.Collections.Generic;
3  using System.ComponentModel;
4  using System.Data;
5  using System.Drawing;
6  using System.Linq;
7  using System.Text;
8  using System.Threading.Tasks;
9  using System.Windows.Forms;
10
11 namespace SensorCarga
12 {
       3 referências
13     public partial class Form1 : Form
14     {
           1 referência
15         public Form1()
16         {
17             InitializeComponent();
18         }
19     }
20 }
```

Define que classes do Framework .NET serão utilizadas no programa

Define o *namespace*: forma de empacotamento de código e conteúdo que torna a sua identificação única

FORMULÁRIO

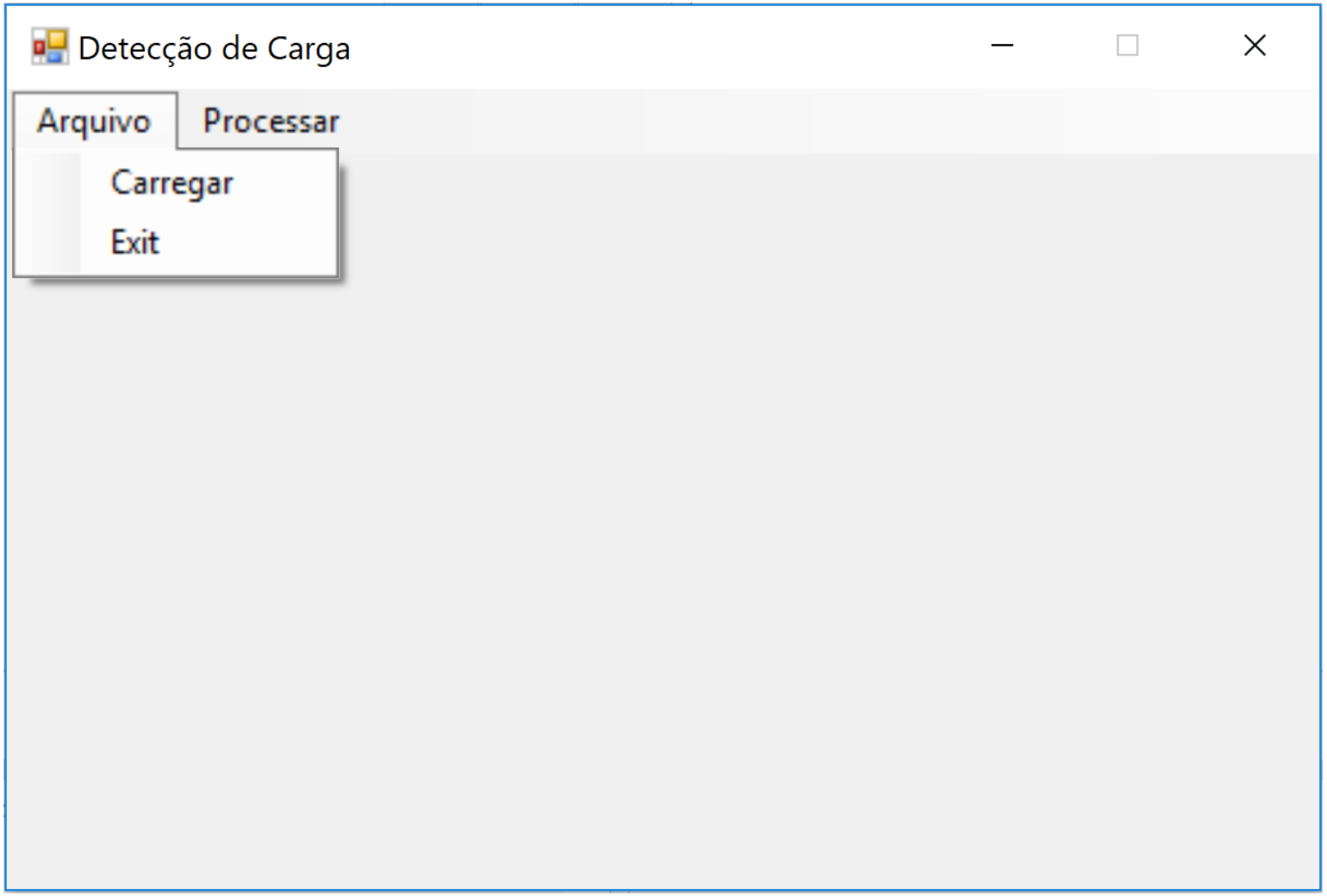
Classe Form fornece o canvas da aplicação onde os controles são colocados (System.Windows.Forms.Form)

1. Renomear o Form1 para FormPrincipal
   - Vai requisitar se deseja renomear as referências: sim

2. Alterar as propriedades do form
   - FormBordStyle: FixedSingle
   - Text: “Detecção de Carga”
   - StartPosition: CenterScreen
   - MaximizeBox: False

Vamos criar um menu!

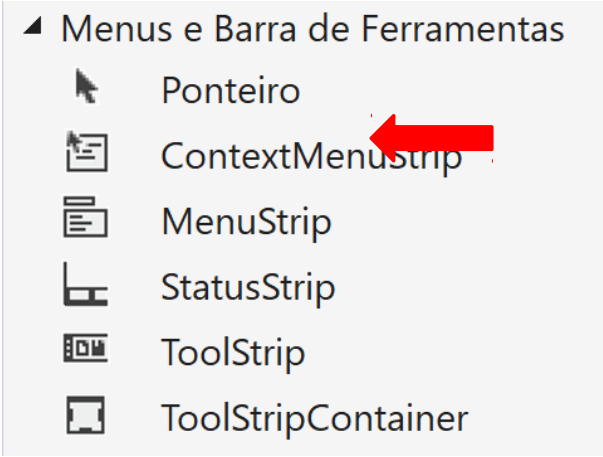
CRIANDO UM MENU PADRÃO



CRIANDO UM MENU PADRÃO

MenuStrip adiciona um controle de menu ao programa

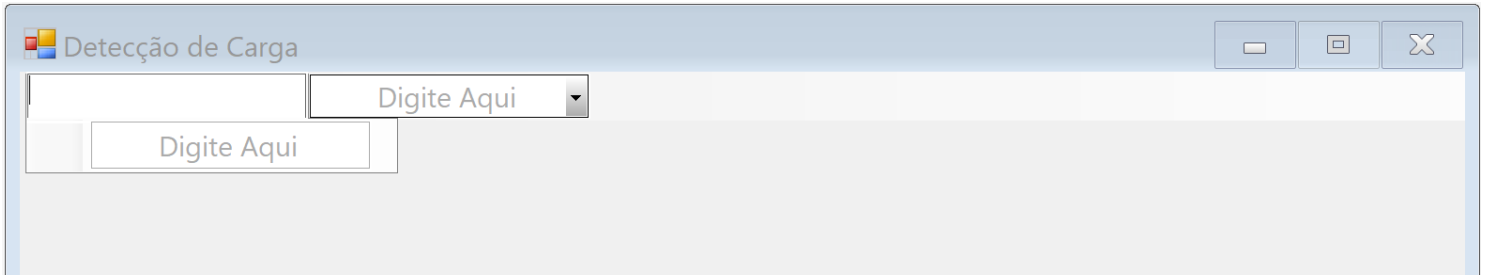
- Clique no texto MenuStrip da Caixa de Ferramentas (lateral esquerda da IDE)



- Arraste (ou dê um duplo-clique) o item até o formulário

CRIANDO UM MENU PADRÃO

Digite os itens do menu (Arquivo, Processar, Carregar, Sair) nas áreas marcadas por **Digite Aqui**



Cada opção selecionável criada no menu é da classe ToolStripMenuItem

- Text: & antes da letra cria um tecla de acesso (ALT + letra)
- Image mostra uma imagem antes da opção do menu
- ShortcutKeys: define um atalho para a opção
- ShortcutKeyDisplayString: mostra o atalho da opção

CRIANDO UM MENU PADRÃO

O identificador do componente/objeto é gerado automaticamente

- O identificador pode ser modificado a fim de representar melhor o objeto
- Encontra-se na propriedade **(Name)**

Ao clicar em um item do menu, um evento é gerado

- Eventos são que nem exceções que são enviadas por um objeto e que podem possuir um código que realize o tratamento

O evento é tratado através de um código definido em um método, a qual deve ser atrelada ao evento Click do item de menu

- Nome do código é tratador/manipulador de eventos – Event Handler

Para adicionar um método para tratar o evento do clique do mouse, deve-se dar um duplo clique na opção do menu

CRIANDO UM MENU PADRÃO

Criar o tratador de clique do mouse na opção de menu Sair: clicar 2 vezes no botão Sair

- VS cria automaticamente o tratador do Evento Click com o nome do controle mais a string do evento (`sairMenuItem_Click()`)

```
private void sairMenuItem_Click(object sender, EventArgs e)
{

}
```

- `sender` é o objeto que disparou o evento (no caso de tratamento múltiplos objetos, o mesmo pode ser utilizado atarvés de um casting)
- e é da classe EventArgs contém informações do evento

Adicionar o método **`Close()`** para fechar o formulário

SELECIONAR O ARQUIVO

Como o objetivo é carregar um arquivo e processá-lo, nós precisamos de uma maneira de poder selecioná-lo de alguma unidade de armazenamento

Para fazer isso, podemos utilizar a classe OpenFileDialog

O componente OpenFileDialog permite que os usuários procurar as pastas do seu computador ou de qualquer computador na rede e selecionar um ou mais arquivos para abrir.

- A caixa de diálogo retorna o caminho e o nome do arquivo que o usuário selecionou na caixa de diálogo.

PROPRIEDADES DO OPENFILEDIALOG

InitialDirectory: o diretório a ser exibido quando a janela de diálogo aparecer pela primeira vez.

```
openFileDialog1.InitialDirectory = @"C:\";
```

Se a propriedade **RestoreDirectory** for definida como True a caixa de diálogo restaura o diretório atual antes de ser fechada

```
openFileDialog1.RestoreDirectory = true;
```

Title: título da janela de diálogo

```
openFileDialog1.Title = “Carregar Arquivo";
```

PROPRIEDADES DO OPENFILEDIALOG

Filter: filtro da janela de diálogo que será usado para filtrar o tipo de arquivos a serem carregados durante a localização.

- Formato: TipoArquivo1[[|TipoArquivo2]...[| TipoArquivoN]]
- TipoArquivo = Rótulo|Extensão1[[; Extensão2]...[; ExtensãoN]]

```
openFileDialog1.Filter = “Arquivos Textos (*.txt)|*.text;*.txt| Todos os Arquivos (*.*)|*.*";
```

PROPRIEDADES DO OPENFILEDIALOG

FilterIndex: índice do filtro atualmente selecionado na caixa de diálogo (default é 1)

```
openFileDialog1.FilterIndex = 2;
```

DefaultExt: extensão padrão do arquivo a ser localizado

```
openFileDialog1.DefaultExt = "txt";
```

CheckFileExists: indica se a janela de diálogo exibirá um aviso se o usuário especificar um nome de arquivo inexistente

```
openFileDialog1.CheckFileExists = true;
```

PROPRIEDADES DO OPENFILEDIALOG

CheckPathExists: indica se a janela de diálogo exibirá um aviso se o usuário especificar um caminho inexistente

```
openFileDialog1.CheckPathExists = true;
```

FileName: nome do arquivo selecionado na janela de diálogo (inclui caminho e extensão)

```
textBox1.Text = openFileDialog1.FileName;
```

SafeFileName retorna o nome e a extensão do arquivo sem o caminho

PROPRIEDADES DO OPENFILEDIALOG

MultiSelect: se esta propriedade for definida como True pode-se selecionar mais de um arquivo e **FileNames (SafeFileNames)** representará todos os arquivos selecionados. Neste caso deve-se usar um laço foreach para exibir os arquivos selecionados

```
openFileDialog1.Multiselect = true;
foreach (String file in
openFileDialog1.FileNames)
{
  StreamReader arq= new StreamReader(file);
}
```

USANDO OPENFILEDIALOG

```
OpenFileDialog selecionaArquivo = new OpenFileDialog();
selecionaArquivo.Title = "Selecione um arquivo de sensor";
selecionaArquivo.InitialDirectory = "C:\\";
selecionaArquivo.Filter = "Células Texto (*.txt)|*.txt|" +
                          "Células Binário (*.bin)|*.bin|" +
                          "Todos os Arquivos|*.txt;*.bin";
selecionaArquivo.RestoreDirectory = false;
if (selecionaArquivo.ShowDialog() == DialogResult.OK)
{
   ...
}
```

EM QUE PONTO ESTAMOS?

1. Criamos o formulário/projeto
2. Adicionamos o menu padrão do projeto (já conseguimos pelo menos fechar a aplicação)
3. Já podemos selecionar um arquivo (ou múltiplos....versão 2.0) para processar

Vamos fazer a leitura do arquivo e obter algumas informações do mesmo (podemos colocar na tela).

Vamos utilizar uma função!

MÉTODOS

Métodos definem o comportamento da classe, e são implementados através de funções

```
tipoVisibilidade tipoDeRetorno nomeDoMetodo
( listaDeParâmetros )
{
   // as instruções do corpo do método ficam aqui
}
```

onde

tipoVisibilidade define a visibilidade do método

tipoDeRetorno define o tipo a ser retornado

nomeDoMetodo define o nome do método a ser invocado

listaDeParâmetros defile a lista de parâmetros

MÉTODOS

```
private float modulo(float real,
          float imaginario)
{
   return (float)Math.Sqrt(real *
         real + imaginario *
         imaginario);
}
```

MÉTODOS - PASSAGEM DE PARÂMETROS

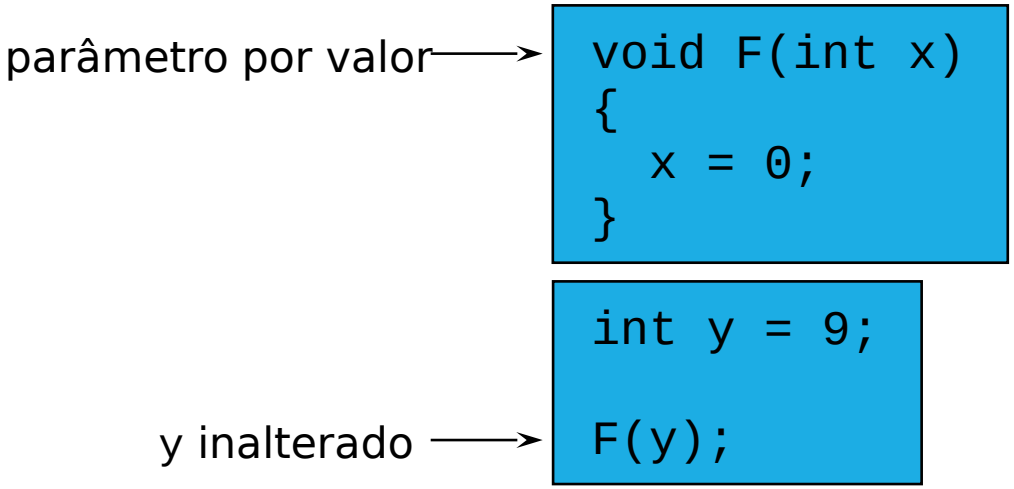
3 Métodos de Passagem de Parâmetros

1. Por valor
2. Por referência
3. Por resultado

MÉTODOS - PASSAGEM DE PARÂMETROS

Por valor

- Valor é copiado para o método
- Qualquer mudança do parâmetro afeta somente localmente



MÉTODOS - PASSAGEM DE PARÂMETROS

Por Referência

- Deve-se utilizar a palavra-chave ref no método e na chamada (obrigatoriamente!)

Parâmetro `ref`, inicialmente valendo 9

```
void G(ref int x)
{
  x++;
}
```

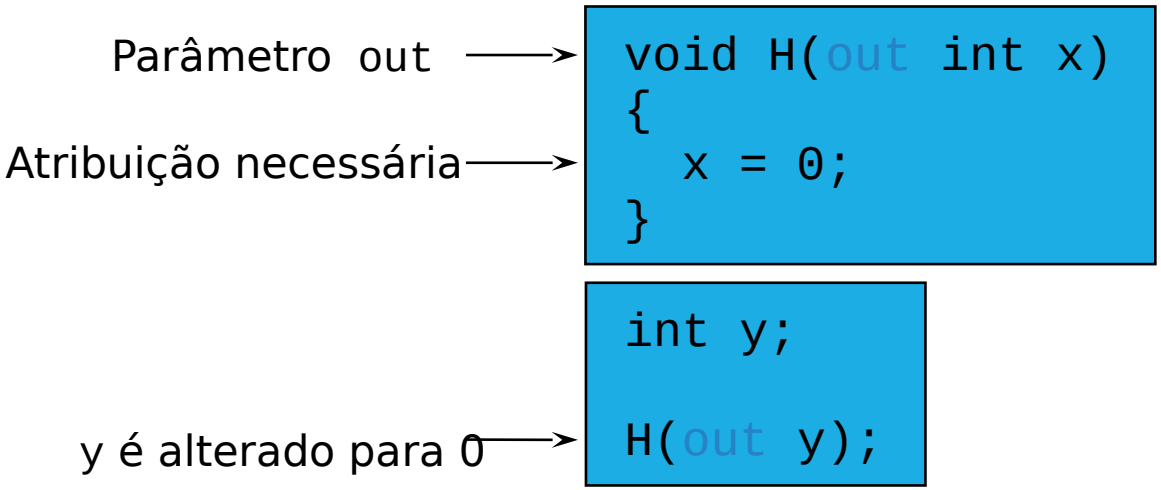
```
int y = 9;

G(ref y);
```

y é alterado para 10

MÉTODOS - PASSAGEM DE PARÂMETROS

Por resultado

- Deve-se utilizar a palavra-chave out no método e na chamada (obrigatoriamente!)



MÉTODOS – DICA!

A mensagem que é mostrada pelo Intelisense pode ser definida após a criação do método.

Basta digitar /// e o VS criará automaticamente código para documentação

```
/// <summary>
/// Função H
/// </summary>
/// <param name="x">parâmetro de teste</param>
void H(out int x)
{
  x = 0;
}
```

VOLTANDO AO NOSSO PROJETO...

Até o momento...

- Criamos um formulário
- Adicionamos um menu padrão com algumas funcionalidades
- Selecionamos um arquivo para processamento
- Realizamos um pré-processamento do arquivo

Agora vamos

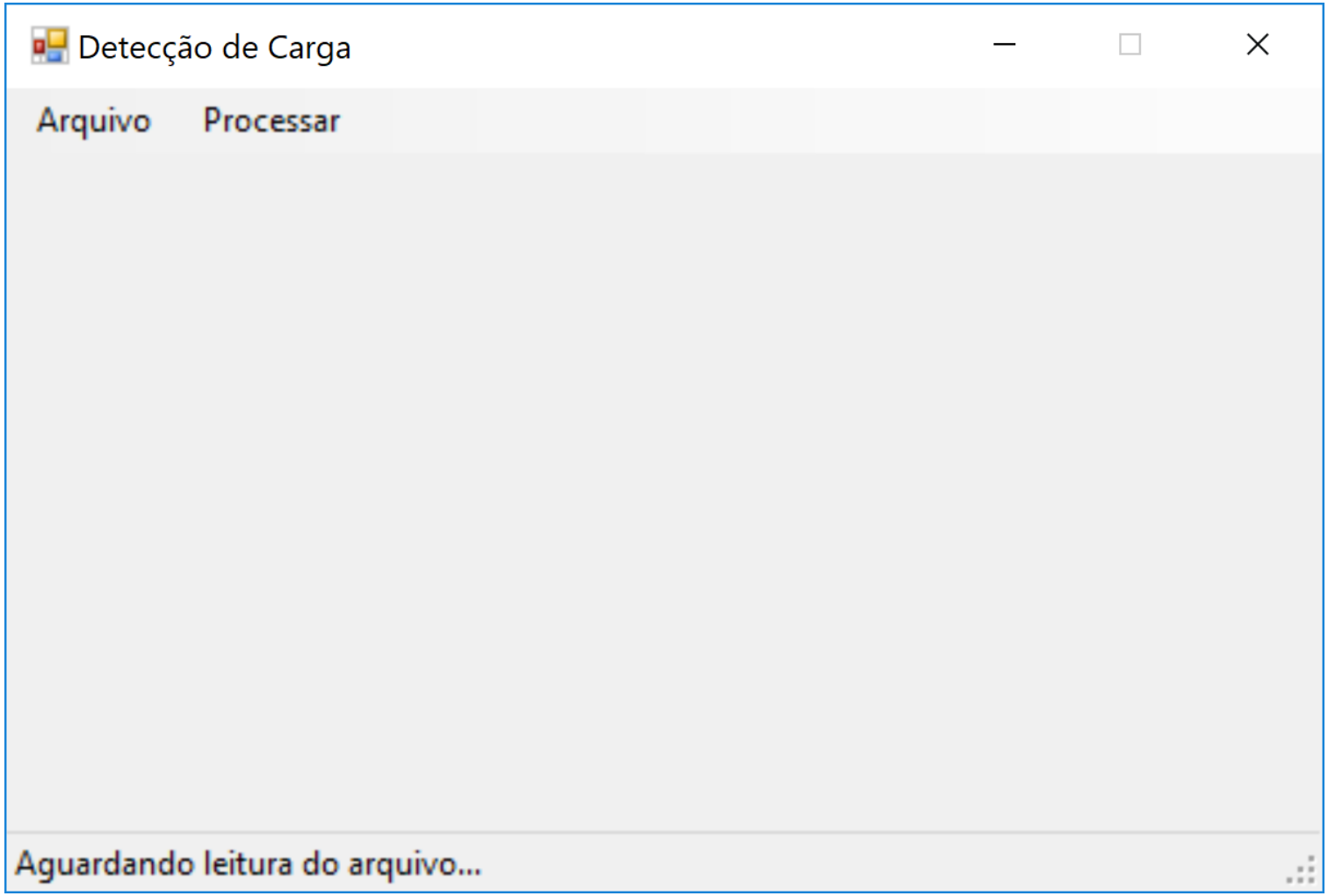
- Criar uma barra de status do programa
- Carregar os dados em memória
- Criar função de processamento

ADICIONANDO UMA BARRA DE STATUS

Uma barra de status para informações gerais do programa através do controle **StatusStrip** (System.Windows.Forms.StatusStrip)

- Adicionar controle **StatusStrip**
- Adicionar **StatusLabel**
  - Clique no StatusStrip e uma lista de controles será apresentada.
  - Selecione **StatusLabel**
  - Renomeie o nome do controle para mensagemStatus
  - Inicialize o texto com a mensagem "Aguardando leitura do arquivo...“ ao iniciar o programa
- Adicionar **ProgressBar**
  - Selecione **ProgressBar**
  - Renomeie o nome do controle para processamentoBar
  - Defina as seguintes propriedades Minimum = 0, Maximum = 100, Visivle = false e Width = 200

ADICIONANDO UMA BARRA DE STATUS



PROCESSAMENTO DO ARQUIVO – CONT.

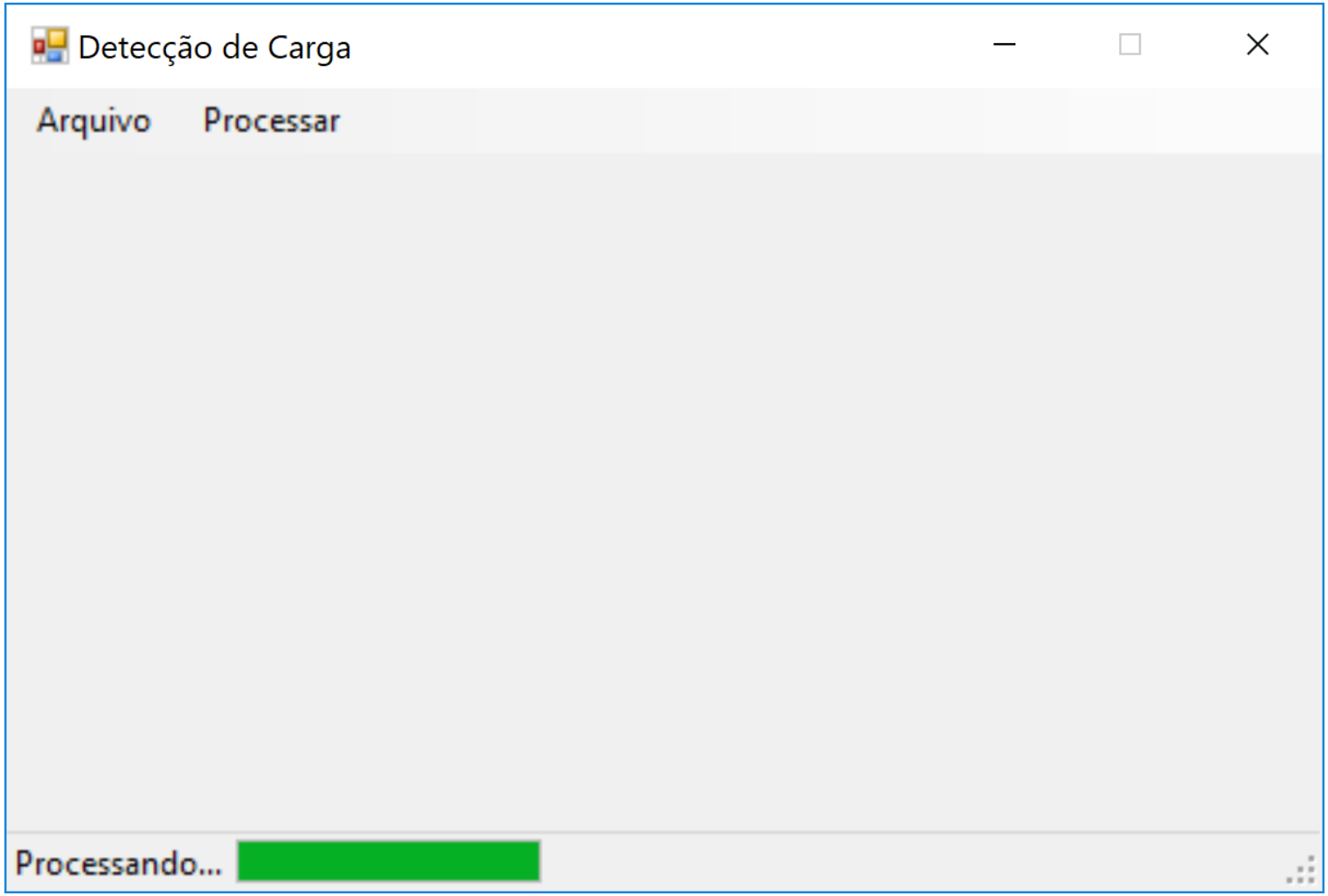
Carregamento do Arquivo

- Após selecionar o arquivo com sucesso, faça um código que carregue o conteúdo do arquivo em duas matrizes (int e float). Uma deve armazenar os tempos e a outra as amostras das células de carga (mais a soma de todas as células por amostra, isto é, uma sétima coluna).
- Antes de começar a leitura, mude a mensagem da barra de status para Processando... e torne a barra de progresso visível. Durante a leitura do arquivo mostre a barra de progresso.

Após a leitura, o programa deverá apresentar uma janela com a mensagem: Dados carregados com sucesso!

- Utilizar MessageBox

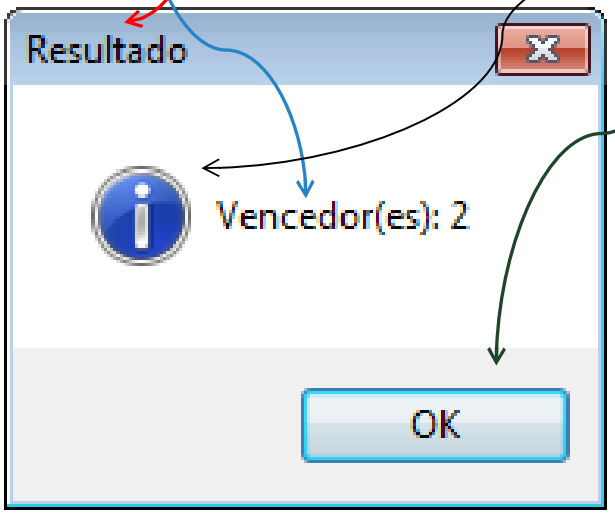
PROCESSAMENTO DO ARQUIVO – CONT.



MESSAGEBOX.SHOW

Mostra uma caixa de mensagem que contem texto, botões, e símbolos que informam e orientam o usuário

```
MessageBox.Show(MensagemString);

MessageBox.Show(MensagemString, TituloString);

MessageBox.Show(MensagemString, TituloString, TipoDeBotão);

MessageBox.Show(MensagemString, TituloString, TipoDeBotão, Ícone);
```



MESSAGEBOX.SHOW

Botões

- MessageBoxButtons.AbortRetryIgnore
- MessageBoxButtons.OK
- MessageBoxButtons.OKCancel
- MessageBoxButtons.RetryCancel
- MessageBoxButtons.YesNo
- MessageBoxButtons.YesNoCancel

O método retorna um valor do tipo **DialogResult**

- DialogResult.Abort
- DialogResult.Cancel
- DialogResult.Ignore
- DialogResult.No
- DialogResult.None
- DialogResult.OK
- DialogResult.Retry
- DialogResult.Yes

Ícones

- MessageBoxIcon.Asterisk
- MessageBoxIcon.Error
- MessageBoxIcon.Exclamation
- MessageBoxIcon.Hand
- MessageBoxIcon.Information
- MessageBoxIcon.None
- MessageBoxIcon.Question
- MessageBoxIcon.Stop
- MessageBoxIcon.Warning

Dica de Produtividade

- Digite mbox e pressione a tecla TAB 2 vezes para que o VS insira o código do método
  - MessageBox.Show("Test");

PROCESSAMENTO DO ARQUIVO – CONT.

Na opção de menu Processar, o programa deve chamar uma função que realize as seguintes etapas:

1. Selecione 2 amostras (somente o valor total). Chamaremos de centros (pode ser um vetor)
2. Utilizando um vetor de acumuladores, acumule os totais nas mesmas posições dos centros ao qual o total é mais próximo (Euclideana). Se o total da amostra 10 é mais próxima do centro 2, some o total da amostra 10 no acumulador 2. A cada soma, incremente um contador para o centro ao qual é mais perto.
3. Após passar por todas as amostras, atualize os novos centros com os acumuladores (normalizados pelos contadores)
4. Vá para o passo 2 (por 20 iterações)

CLASSE RANDOM

Gera números pseudo-aleatórios

Métodos

- Next() retorna um número aleatório não-negativo
- Next(Int32) retorna um número aleatório não negativo [0,maior)
- Next(Int32, Int32) retorna um número aleatório não negativo [menor,maior)
- Sample() retorna um número aleatório [0.0;1.0]