

# **Técnicas de Programação**

**Prof. Aurivam**

## Sumario

Aula 1-Conceito C Sharp. ....	3
Conceito de Projeto, bibliotecas, namespace. ....	5
Aula 2-Variável e Constante e Declaração de variáveis. ....	10
Aula 3- Tipos de dados. ....	11
Aula 4- Entrada de dados ....	12
Aula 5- Saída de dados. ....	15
Aula6- Operadores Aritméticos.....	17
Aula7- Estrutura de controle condicional if/else.....	19
Aula 8-Operadores relacionais. ....	22
Aula 9-Depuração e Debug.....	25
Aula 10- Versionamento de software.....	29
Aula 11 - GitHub ....	31
Aula 12-Estrutura de Repetição- FOR.....	36
Aula 13- Estrutura de controle condicional switch “ case ”. ....	38
Aula 14 – Classe ....	41
Aula 15- Métodos.....	44
Aula 16- Lista ....	47
Aula 17- Botões e propriedades do Windows forms.....	49
Aula 18 – Desenvolvendo no Windows Forms – calculo ....	50
Aula-19 Desenvolvimento no Windows forms consumo de carro.....	51
Aula-20 Desenvolvimento no Windows forms listbox. ....	52

## Aula 1-Conceito C Sharp.

**Linguagem de programação-** Linguagem escrita e formal que especifica um conjunto de instruções e regras usadas para gerar programas (software).

Um software pode ser desenvolvido para rodar em um computador, dispositivo móvel ou em qualquer equipamento que permita sua execução..

**Programar**, na prática, é escrever um texto que será transformado em um software.

**"C Sharp"**, é uma linguagem de programação, da Plataforma .NET, derivada de C/C++, simples, moderna e orientada à objetos. Possui o poder do C/C++ aliado com a alta produtividade do Visual Basic.

É distribuído juntamente com Microsoft Visual Studio.NET, e tem acesso a toda a plataforma do Next Generation Windows Services (NGWS), que incluem uma poderosa biblioteca de classes e um mecanismo de execução comum.

O visual Studio usa plataformas de desenvolvimento de software como por exemplo, o Windows forms, o Windows Presentation foundation (WPF) e Microsot Silverlight.

**O visual Studio suporta linguagens com Visual Basic, Visual C#, Visual C++, Visual F# e Javascript.**

Essas linguagens aproveitam as funcionalidades da plataforma .NET, que permite o acesso a tecnologia-chave que simplificam o desenvolvimento de aplicações desktop, web.

### Vantagens do C#

Orientada a Objetos.

Todos os métodos e atributos devem ser declarados dentro de classes.

Atributos e métodos estáticos de classes públicas podem servir como substitutos para variáveis e métodos globais.

Entre os diversos recursos do ambiente **C# Visual Studio** podemos destacar:

**O Editor de Código (Code Editor)**, usado para manipular o código fonte;

**O Compilador C# (C# Compiler)**, utilizado para converter o código fonte em um programa executável;

**O Depurador do Visual Studio (Visual Studio Debugger)**, usado para testar seus programas;

**A Caixa de Ferramentas (Toolbox)** e o **Editor de Formulários (Windows Forms Designer)**, para a rápida criação de interfaces com o usuário usando o mouse;

**O Explorador de Soluções (Solution Explorer)**, útil para o gerenciamento de arquivos de projeto e configurações;

**O Editor de Projetos (Project Designer)**, usado para configurar o compilador, caminhos de instalação e demais recursos;

**O Visualizador de Classes (Class View)**, usado para navegar através das classes definidas no seu código fonte;

**A Janela de Propriedades (Properties Window)**, utilizada para definir as propriedades e eventos nos controles da sua interface com o usuário;

### Código fonte.

Esse texto deve ser escrito em uma **linguagem de programação** e é chamado de **código**, mas não é um código lido apenas por uma máquina, é um código que pode ser lido por um ser humano.

Conceito de Projeto, bibliotecas, namespace.

The image shows a screenshot of a C# program in Visual Studio with several annotations explaining key concepts:

```
1 using System;
2
3 namespace curso {
4     class Program {
5
6         static void Main(string[] args) {
7             Console.WriteLine("Olá mundo!");
8             Console.ReadLine();
9         }
10    }
11 }
12
13
14
```

**Importação de bibliotecas** (pointing to line 1): `using System;`

**namespace: é uma forma de organizar as classes do seu projeto em módulos. Cada arquivo pertencerá a um namespace que você declarar** (pointing to line 3): `namespace curso {`

**Declaração da classe. Todo código em C# fica dentro de uma classe. A classe define dados e operações. No caso de um programa mínimo como este, temos apenas uma classe (chamada "Program") e uma única operação: o método "Main". Este é um assunto de Programação Orientada a Objetos.** (pointing to line 4): `class Program {`

**static é um prefixo que indica que a operação (no caso o "Main") pode ser executada independentemente de se instanciar um objeto desta classe. Este é um assunto de Programação Orientada a Objetos.** (pointing to line 6): `static`

**O prefixo void indica que a operação (no caso o "Main") não retorna nenhum valor, ou seja, apenas executa uma ação e termina.** (pointing to line 6): `void`

**Isto indica que a operação "Main" pode receber parâmetros para sua execução. Alguns programas modo console usam isso.** (pointing to line 6): `Main(string[] args)`

**Using** serve para carregar as bibliotecas do sistema, ou seja, muitas funções e comandos já estão prontas para serem utilizadas, portanto temos que carregá-las no projeto para que possamos usufruir.

**Biblioteca** em linguagem de programação **são** rotinas/ funcionalidades padronizada da **linguagem** de programação **C#** que contém operações comuns como tratamento de entrada/saída e cadeia de caracteres.

**Namespace** é o nome dado ao ambiente de desenvolvimento do projeto. Em um projeto podemos criar vários arquivos (classes) e com isso para cada arquivo ou classe criado, utilizaremos o mesmo *namespace*, facilitando a organização do projeto.

**Class** diretiva utilizada em linguagem orientada a objetos para identificar o tipo de objeto a ser modelado e o arquivo deverá ter sempre o mesmo nome da classe.

**Static void main (string[] args)** é o cabeçalho do método principal do projeto, ou seja, no momento da execução do projeto, será executada a sequência lógica de comandos que estiverem escritas a partir do início do desenvolvimento deste método. O nome deste método é **main**.

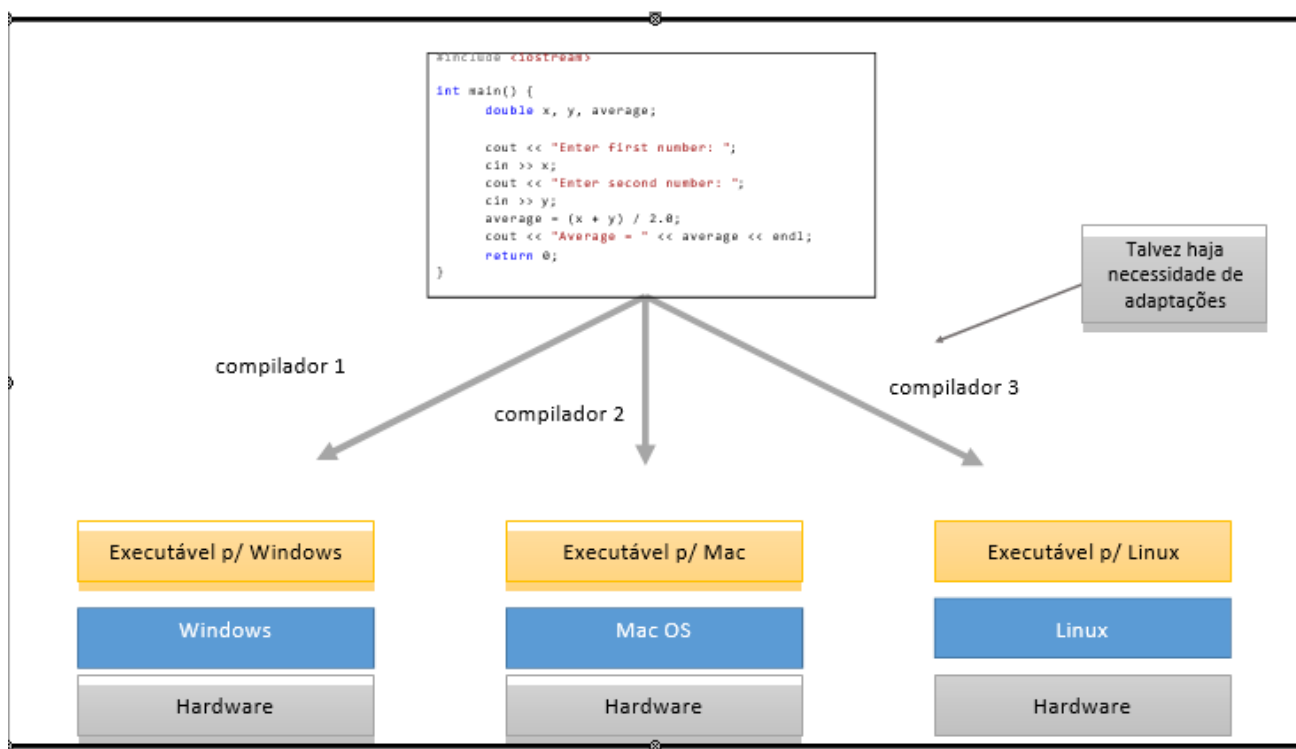
As chaves “{}” servem para delimitar um bloco de instruções, o início e término de métodos, classes, *namespaces* ou comandos.

Indentação do código. ctrl + k+ D

**Quebra de linha das chaves.** Por convenção o c#, você ver as chaves abertas abaixo da estrutura do código, alguns programadores usam o abre chaves na frente.

## Compilação

Compilação é o ato / processo de traduzir um código fonte feito em uma linguagem de programação para uma linguagem do computador, para que suas instruções sejam executadas pelo processador, ou seja, cria o executável de um programa do código fonte.

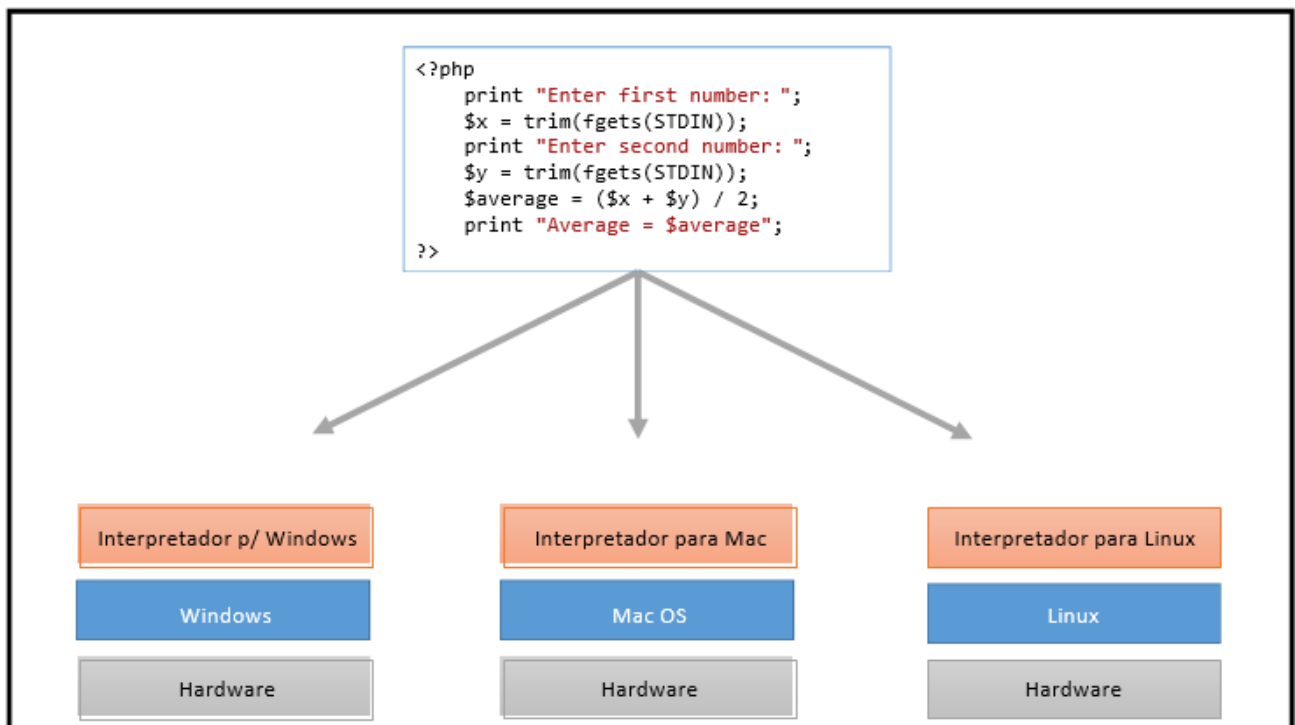


**Código fonte** não é compreendido pelo computador e sim pelo programador. Compilador específico para o sistema operacional transforma o código fonte em executável que será executado pelo computador.

### Erros em tempo de Compilação –

Caso a sintaxe (regra) de alguma instrução for escrita errada, na própria linha de comando irá aparecer um sublinhado em vermelho, alertando que algo não está escrito corretamente, aí basta analisar

## Interpretador



Um programa inscrito em linguagem interpretada como PHP ele não vai ser compilado para gera um executável.

Para cada plataforma diferente, eu vou ter um interpretador, que é um software que é capaz de ler o código fonte , converter em executável e ai sim vai será executado.

### Vantagens

Manutenção, escrevo o código uma vez só e executa em sistemas diferente.

### Desvantagens

Esse processo é muito mais lento do que uma execução de um programa compilado.



Sintaxe do Programa do c# ( sharp) modo Console.

`Console.WriteLine(" Digite os dados do produto: ");`Quebra linha e Escreva.

`Produto = Console.ReadLine();`Lê a linha e guarda na variável.

`Console.WriteLine();`é responsável por exibir uma mensagem da tela e logo em seguida pular uma linha.

`Console.Write(" O nome do produto é : " + nome);`Concatenação

`Console.ReadLine();`Programa aguardar usuário apertar algum tecla

`Console.ReadKey();`é responsável por pausar a execução do programa.

`Console.ReadLine();`é reconhecida como uma string.

Portanto, para que a informação seja armazenada em uma variável de outro tipo (diferente de *string*), temos que aplicar a conversão de tipo adequada a cada necessidade.

Exemplo de conversão:

```
int N = int.Parse(Console.ReadLine());
```

Conversão de string para boolean :

```
bool R = bool.Parse(Console.ReadLine());
```

Conversão de string para char

```
char L = char.Parse(Console.ReadLine());
```

**Endentação-** Importante endentação seu código, o que estiver dentro da estrutura, afasta pra direita para facilitar a organização do código.

CTRL + K + D.

## Aula 2-Variável e Constante e Declaração de variáveis.

Para armazenar informações em um programa, é preciso reservar espaço na memória.

E para isso temos que declarar variáveis, para que estas possam receber informações e assim pode ser utilizada durante a execução do programa.

Como o próprio nome já explica, uma variável pode ter seu conteúdo alterado durante a execução de um programa.

**Variável:** são nomes dados aos espaços reservados na memória para que sejam armazenados valores, que posteriormente possam ser utilizados durante a execução o programa. No caso as variáveis *a* e *b* possuem valores do tipo inteiro.

**Constantes :** São nomes dados a espaços reservados na memoria onde os valores não poderão ter seu conteúdo alterado durante a execução de um programa, ou seja, diferente das variáveis, as constantes manterão o mesmo valor do início ao fim da execução.

Uma constante é uma variável,que armazenará um valor ÚNICO, um valor que NÃO mudará com o tempo de execução do programa.

**Concatenação:** O símbolo de adição “+” é usado para concatenar, ou seja, montar uma mensagem para que seja exibida uma mensagem organizada na tela. Podemos definir como “juntar” texto com variáveis para que seja exibida a mensagem na tela da forma desejada.

**Operação:** Neste caso está realmente sendo realizada uma operação de adição (soma) de duas variáveis

## Aula 3- Tipos de dados.

**Byte** 0 até 255 Não aceita numero negativo.

**Sbyte** 0 até 127

**Short** -32,768 ..32767

**Int** 2147483647 números inteiros com “4 bytes.”

**Long** -9223,372,036,854 com “8 bytes”

**Float** . flutuante, acessa numeros quebrados e com “4 bytes.”

**Double** -precisão, numeros quebrados e com “8 bytes.”

**Char** é usado para conter um único caractere unicode

**Bool** boleando- verdadeiro ou falso. Quando declaramos o bool sem valor ( true e false) o sistema atribui false automaticamente.

### Funções para Valor Minimo ou Maximo.

#### Exemplo

Int N1= int.MaxValue

Int N2=int.MinValue

Sbyte N3= Sbyte.MaxValue

Sybyte N4= Sbyte.MinValue

### Restrições da linguagem - Nome de variáveis.

Não pode começar com dígitos: use letra;

Não pode ter espaço em branco;

Não usar acentos ou til

#### Errado

Int 5minutos;

Int salário;

Int salario do funcionário;

#### Certo

Int 5minutos;

Int Salario;

Int salarioDofuncionario;

## Aula 4- Entrada de dados

Agora será apresentada uma maneira de fazer com que o usuário digite a informação e esta fique armazenada em uma variável.

Para isso utilizamos o **comando- console.ReadLine();**

Toda a informação capturada pelo Console.ReadLine() é reconhecida como uma string. Portanto para que a informação seja armazenada em uma variável de outro tipo diferente de (string), temos que aplicar a conversão do tipo adequada a cada necessidade.

**Exemplo(console.ReadLine());**      **Comando de entrada de dados**

int n= int.Parse (console.ReadLine());

double n= double. Parse (console.ReadLine ());

**Conversão de string para double:**

v= double. Parse (console.ReadLine ());

**Conversão de string para boolean:**

R= bool.Parse (console.ReadLine());

**Conversão de string para char:**

L= char. Parse (console.ReadLine());

**Conversão de string para int:**

N = int.Parse (console.ReadLine());

**To.string=** limitador de casas decimais.

ToString("f2").

O teclado em português, o separador de casa decimais é a vírgula e não o ponto. Para resolver isso adiciona a biblioteca Using System.Globalization para inverter para vírgula.

**Culture.info.Invariantculture-** conversor de vírgula para ponto no C#.

using System.Globalization

**Exemplo**

➤ Console.WriteLine(Saldo.ToString("f3", CultureInfo.InvariantCulture));

**Declaração de variáveis.**

```
int n1 = int.Parse(Console.ReadLine());  
char ch = char.Parse(Console.ReadLine());  
double n2 = double.Parse(Console.ReadLine());
```

```
Console.WriteLine(n1);
```

```
Console.WriteLine(ch);
```

```
Console.WriteLine(n2);
```

```
Console.ReadLine();
```

**1-**Desenvolva aplicação via console no C# que leia o nome, peso, e data de nascimento. Apresente as informações no final da aplicação.

**2-**Desenvolva Uma aplicação para executar a seguinte interação com o usuário:

le os valores destacados em vermelho e depois exibir os dados na tela.

Entre com o nome completo: **Alex Green**

Qual é o seu peso? **73**

Qual é a data de Nascimento? **23/12/1978**

Apresente na tela as informações:

**Meu nome é Alex Green, tenho 73 kilos e nasci em 23/12/1978**

## Exercícios

1-Desenvolva código fonte via console C# chamada **SomaValores** que leia dois números inteiros (via console) e exiba o resultado da soma dos valores informados.

2-Desenvolva código fonte via console C# chamada **Operacoes** que leia dois números reais (via console) e exiba o resultado da adição, subtração, multiplicação e divisão dos valores informados.

3-Desenvolva código fonte via console C# chamada **CotacaoDolarParaReal** que:

O usuário deve informar a cotação do dólar no dia, depois o usuário informa a quantidade de dólar .O programa deve converter para reais o valor e exibir o valor convertido em real.

4- Desenvolva código fonte via console C# chamada **Consumocarro** que leia a quantidade de quilômetros percorridos e a quantidade de combustível colocada no carro. Exibe o consumo do veículo por km rodado.

5- Desenvolva código fonte via Console C# chamada **Compra**. O usuário deve informar nome do produto, valor e quantidade. Exibe no final o nome do produto comprado e o total da compra.

## Aula 5- Saída de dados.

Saída é quando o programa informa dados para o usuário. No caso aqui, imprimir informação na tela.

**Console. Write**> escreve sem quebra de linha

**Console. Writeline**> coloca quebra de linha.

Exemplo double x = 10.35784

**Console.Writeline (x. toString("F2"),cultureInfo.InvariantCulture));**

Importar a biblioteca `system globalization;`

Concatenar vários elementos +

Exemplo

**Console.WriteLine**( " o valor do troco é " + x " reais");

### Exercício Saída de dados.

Em um novo programa, inicie as seguintes variáveis:

```
string produto1= "computador";  
string produto2= " mesa de escritório";
```

```
byte idade= 30;  
int código= 52;  
char gênero= 'm';
```

```
double preço1= 2100.00;  
double preco2= 650,50  
double medida= 53.234567
```

Em Seguida, usando os valores das variáveis, produza a seguinte saída:

```
Produtos;  
computador, cujo preço é 2100,00.  
Mesa de escritório, cujo preço é R$ 650,00.  
Registro: 30 anos de idade, código 52 e gênero: M
```

## Exercícios

1. Desenvolva código fonte via console C# chamada **CalculaMediaAluno** que leia o nome de um aluno e as notas das três provas que ele obteve no semestre. No final informar o nome do aluno e a sua média (aritmética).
2. Desenvolva código fonte via console C# chamada **sucessorantecessor** que leia um valor para a variável . Exibir o antecessor e sucessor dessa variável N.
3. Desenvolva código fonte via console C# chamada **CalculaComissao** que realize o cálculo de comissão dos vendedores de uma loja, levando-se em consideração que a comissão do vendedor será de 5% do total de vendas de acordo com os seguintes dados:
  - a. Nome do vendedor.
  - b. Preço unitário da peça.
  - c. Quantidade vendida.
  - d. Exiba as seguintes informações nome do vendedor,, valor unitário, quantidade vendida e comissão do vendedor.
4. A Loja Tabajara está vendendo seus produtos em 5 (cinco) prestações sem juros. Construa uma Aplicação console C# chamada **Prestacoes** que receba o valor da compra e exiba o valor das prestações.
5. Desenvolva código fonte via console C# chamada **PrecoCusto** que receba o preço de custo de um produto e mostre o valor de venda. Sabe-se que o preço de custo receberá um acréscimo de acordo com um percentual informado pelo usuário.
6. Desenvolva código fonte via console chamada **ParOuImpar** que receba um número inteiro e informar se este número é par ou ímpar.



## Aula 6- Operadores Aritméticos.

**Operador de Atribuição** -O operador de atribuição em Codigos é representado = indicando que algum valor será atribuído em alguma variável. **Por exemplo:** dobro = n1\*2;

### Operadores de atribuição

Operador	Exemplo	Significado
=	<b>a = 10;</b>	a <b>RECEBE</b> 10
<b>+=</b>	<b>a += 2;</b>	a <b>RECEBE</b> a + 2;
<b>-=</b>	<b>a -= 2;</b>	a <b>RECEBE</b> a - 2;
<b>*=</b>	<b>a *= 2;</b>	a <b>RECEBE</b> a * 2;
<b>/=</b>	<b>a /= 2;</b>	a <b>RECEBE</b> a / 2;
<b>%=</b>	<b>a %= 3;</b>	a <b>RECEBE</b> a % 3;

**Operador aritmético**- Os operadores aritméticos são aqueles que nos permitem realizar operações matemáticas, com dados numéricos. Além da adição, subtração, multiplicação e divisão, podemos utilizar também alguns outros operadores.

+ (Adição)

- (Subtração)

\* (Multiplicação)

/ (Divisão)

% (Resto)

> (Maior)

< (Menor)

<= (Menor igual)

>= (Maior igual)

!= (Diferente)

== (Igualdade)

**Por exemplo:** soma= n1+n2; ou sub = n1-n2;

**Procedência de operadores-** Operações dentro de parênteses são sempre executadas primeiro, como nas operações matemáticas. Em C#, operadores multiplicativos (\*,/,%) tem precedências sobre os aditivos(+, -).

Vamos dar a ordem de procedência da seguinte expressão:

Exemplo.

$$3+4 *2 = 11$$

$$(3+4) *2= 14$$

Para efetuar a soma primeiro, podemos utilizar os parênteses:

Exemplo.

$$3+4 *2 = 11$$

$$(3+4) *2= 14$$

Perceba que a ordem de procedência altera o resultado, por isso devemos ter atenção com a procedência.

## Aula7- Estrutura de controle condicional if/else.

Nesta estrutura entra uma palavra mágica chamada "se", que dependendo do resultado da operação que for realizada, executa algumas instruções, "senão" executa outras instruções. Em outras palavras, permite a escolha de um grupo de instruções para serem executadas de acordo com aceitação ou não de certas condições.

Composta:

### Simples

```
if ( condição){  
  Comando 1  
  Comando 2  
}
```

### Composta:

```
}
```

```
If (condição){  
  Comando 1  
  Comando 2  
}
```

```
Else {  
  Comando 1  
  Comando 2  
}
```

### Encadeamentos

```
If (condição){  
  Comando 1  
  Comando 2  
}
```

```
Else if{  
  Comando 3  
  Comando 4
```

```
Else if{  
  Comando 5  
  Comando 6
```

```
Else {  
Comando 7  
Comando 9
```

### Estrutura condicional

```
static void Main(string[] args)  
{  
    string nome;  
    int idade;  
  
    Console.WriteLine(" Boa noite ");  
  
    Console.WriteLine(" Qual é o seu nome? ");  
    nome = Console.ReadLine();  
  
    Console.WriteLine(" Qual é a sua idade?");  
    Idade = int.Parse(Console.ReadLine());  
  
    if( idade > 50)  
    {  
        Console.WriteLine(" Você já estar ficando velho ");  
    }  
  
    else  
    {  
        Console.WriteLine(" Você ainda é novinho belo ");  
    }  
  
    Console.WriteLine(" Você se chama " + nome + " e vc tem " + idade + " anos ");  
    Console.ReadLine();
```

### Exercicios

1-Desenvolva uma Aplicação console C# chamada **SomaValores** que leia dois numeros e exiba o maior valor.

2-Desenvolva uma Aplicação console C# chamada **Operacoes** que leia dois números reais (via console) e exiba o resultado da adição, subtração, multiplicação e divisão dos valores informados.

3-Desenvolva uma Aplicação console C# chamada **CotacaoDolarParaReal** que:

O usuário deve informar a cotação do dólar no dia, depois o usuário informa a quantidade de dólar  
O programa deve converter para reais o valor e exibir o valor convertido em real.

4- Desenvolva aplicação console C# chamada **Consumocarro** que leia a quantidade de quilômetros percorridos e a quantidade de combustível colocada no carro. Se consumo maior que 8, Exibe mensagem “carro econômico” senão “carro gastão”.

5 Desenvolva um programa em C# chamada -rodizio que leia o final da placa do veículo e exiba que dia o veículo não pode circular. Usando estrutura de if/else para que o programa execute conforme o modelo.

1 e 2= Segunda-feira

3 e 4= terça-feira

5 e 6= quarta-feira

7 e 8= quinta-feira

9 e 0= sexta-feira

6-Desenvolva uma Aplicação console C# chamada Diferenca que leia dois números e apresente a diferença do maior para o menor.

7- Desenvolva uma Aplicação console C# chamada MaiorMenor que leia dois números e exiba mensagem informando o valor do maior número e o valor do menor número. Se os dois números forem iguais, exibir mensagem “valores informados são iguais”.

8-Desenvolva uma Aplicação console C# chamada Intervalo que leia um número inteiro. Verifique por meio de condição se o valor fornecido está na faixa entre 0 (zero) e 9 (nove). Caso o valor fornecido esteja dentro da faixa, apresentar a mensagem “valor válido”. Caso contrário, apresentar a mensagem “valor inválido”.

9-Desenvolva uma Aplicação console C# chamada ParOuImpar para ler um número inteiro e informar se este número é par ou ímpar.

10- Fazer um programa para ler um número inteiro, e depois dizer se este número é negativo ou não.

## Aula 8-Operadores relacionais.

**Operador lógico-** Os operadores lógicos servem para combinar resultados de expressões, retornando se o resultado final é verdadeiro ou falso.

==	igual
&&	E
	ou

**&&(E/AND)** o resultado de uma expressão usando o operador lógico AND é verdadeiro somente se todas as condições forem verdadeiras.

```
int a = 10;
```

```
Console.WriteLine( a<11 & a>21);  
False
```

```
int a = 10;
```

```
Console.WriteLine(a <= 10 & a >= 10);
```

**||** o resultado de uma expressão usando operador lógico OR é falsa, somente se todas as condições forem falsas.

```
int a = 10;
```

```
Console.WriteLine(a < 9 || a >= 12);
```

1--Desenvolva uma Aplicação console C# chamada ServicoMilitar que leia os dados de **"4" pessoas** (nome, idade e altura). Se altura maior 1,80 e idade maior que 18, informar está apta ou senão para cumprir o serviço militar obrigatório. **(FOR)**

2-Desenvolva uma Aplicação console C# chamada AumentoSalarial para ler um salário e atualizá-lo de acordo com a tabela abaixo.

FAIXA SALARIAL	AUMENTO
Até 600,00	30%
601,00 a 1.100,00	25%
1101,00 a 2.400,00	20%
2401,00 a 3.550,00	15%
Acima de 3.550,00	15%

3-Um determinado clube de futebol pretende classificar seus atletas em categorias e para isto ele contratou um programador para criar uma Aplicação console C# chamada ClassificacaoAtleta que executasse esta tarefa. Para isso o clube criou uma tabela que continha a faixa etária do atleta e sua categoria. A tabela está demonstrada abaixo:

IDADE	CATEGORIA
De 05 a 10	INFANTIL
De 11 a 15	JUVENIL
De 16 a 20	JUNIOR
De 21 a 25	PROFISSIONAL

Construa uma Aplicação console C# chamada clubeFutebol que solicite o nome e a idade de um atleta e imprima a sua categoria.

4-Elabore um código para testar se uma senha digitada é igual a "MinhaSenha". Se a senha estiver correta escreva "Acesso permitido", do contrário emita a mensagem "Você não tem acesso ao sistema".

5-Desenvolva um código que receba o dia da semana (número) e informe o dia da semana (literal).

DIA	DESCRIÇÃO
1-	Domingo
2	Segunda-Feira
3	Terça-Feira
4	Quarta-Feira
5	Quinta-feira
6	Sexta-Feira
7	Sábado

**Operadores de atribuição.**

são formas usadas para otimizar uma linha de instrução, que usa o valor da própria variável (antes o operador) para realizar o cálculo e atribuir nela mesma o novo resultado.

**+=** atribuição aditiva

**-=** atribuição subtrativa

**\*=** atribuição multiplicativa

**/=** atribuição divisão

**%=** atribuição resto da divisão.

Exemplo:

```
int a = 10;
```

```
Console.WriteLine(a);
```

```
a += 2; ou a++;
```

```
Console.WriteLine(a);
```

```
a *= 3;
```

```
Console.WriteLine(a);
```

```
a -= 4; ou a--;
```

```
Console.WriteLine(a);
```

1-Desenvolva uma aplicação em C# chamada operador que leia um número e exiba seu sucessor e antecessor.

2-Desenvolva uma aplicação via console em C# chamada intervalo, que o usuário Informe um número no intervalo de 20 à 30. Se o número digitado estiver no intervalo, exiba "o número está no intervalo" senão "número inválido".

3-Desenvolva uma aplicação em C# chamada idadeusuario, que leia nome do usuário, idade e peso. Se peso for menor que 70 e idade menor que 20, aplicação deve exibir "Parabéns você ainda tá novinho", do contrário "Você está velho eh".

4-Desenvolva uma aplicação em C# chamada compracerta, que leia o nome do produto, o valor do produto e a quantidade. Aplicação deve efetuar valorTotal da compra e parcelar em 3 vezes. Se o valor da compra for maior que R\$ 300,00, aplicação deve exibir "Bom cliente", do contrário "Cliente gasta pouco". Aplicação deve calcular o ICMS de 12% do valor total da compra e exibe em mensagem "valor de ICMS" e o parcelamento se houver.



## Aula 9-Depuração e Debug

Durante o processo de codificação de seu programa erros podem acontecer, o processo de “depuração e debug” são úteis para resolver estes problemas.

**Depuração e Debug-** Se trata de corrigir erros durante o processo de codificação e execução de seu programa.

Durante a codificação vários tipos de erros podem ocorrer como:

- Erros de sintaxe.
- Erros de declaração.
- Erros de caractere inválido.
- Erros de conversão de tipo.
- Erros de espera de expressão.
- Outros...

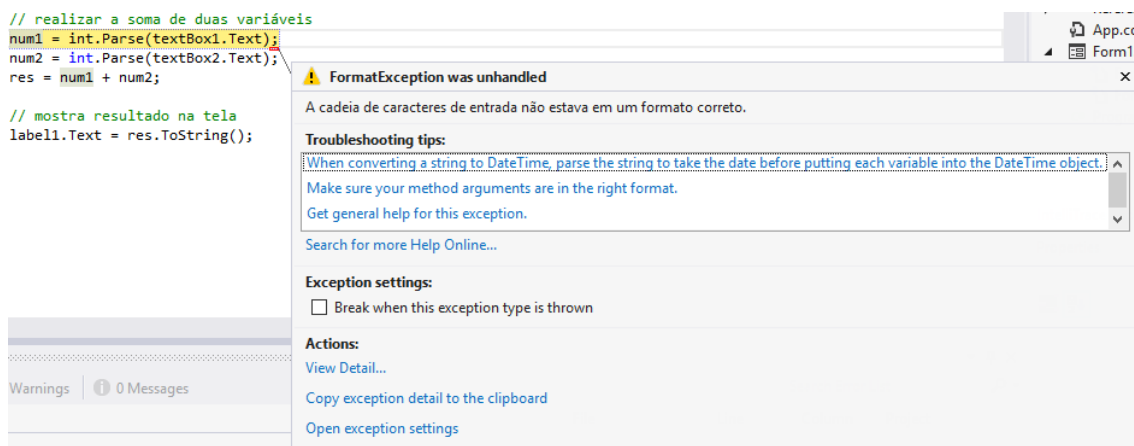
Exemplo:

**Erro de Declaração** – Quando digitamos erroneamente um comando, exemplo:

```
// declaração de váriaveis|  
integer num;
```

O correto seria “int” para declarar um tipo primitivo inteiro, neste caso se o mouse for colocado sobre a linha ondulada em vermelha a mensagem “The type or name space name “integer” couldnotbefound...” será apresentada.

**Erros em tempo de execução “RuntimeError”** – Ocorrem durante o processo de execução da aplicação, como por exemplo ao receber o valor indevido para uma variável, exemplo:



Neste caso tratamentos de erro como o uso de “try-catch”, conversões de tipo ou até mesmo observar se está sendo feita a atribuição corretamente podem ser necessários.

Se o mouse for colocado por cima da linha tracejada em vermelho podemos observar a mensagem de erro “Cannot implicitly convert type ‘string’ to ‘int’”.

Existem também os chamados erros humanos “**Human Errors**”, que são causados por erros de lógica, que acabam por nos dar resultados diferentes do esperado ao concluir a ação do programa, como por exemplo, ao solicitar a soma de dois números e o resultado obtido é uma subtração. Este tipo de erro é um dos mais difíceis de resolver e para isso o Visual Studio nos proporciona a ferramenta de debug em tempo de execução.

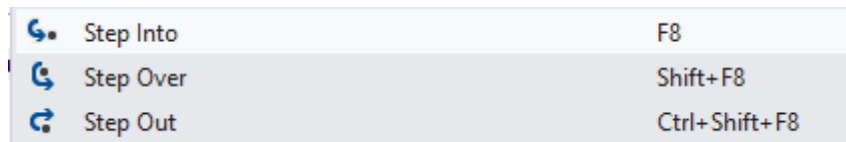
O tratamento deste tipo de debug pode ser feito analisando todo o código a ser executado, como fazendo com que o programa seja interrompido em pontos estratégicos para análise “Breakpoints”.

### Percorrendo todo o código

- Verifique se o Visual Studio está no modo “Debug”.



Para dar início ao Debug podemos ir na aba de menu “Debug” e selecionar “StepInto” ou clicar nos ícones correspondentes.



Step over- debug de uma tela só.

Step Into- debug de varias telas.

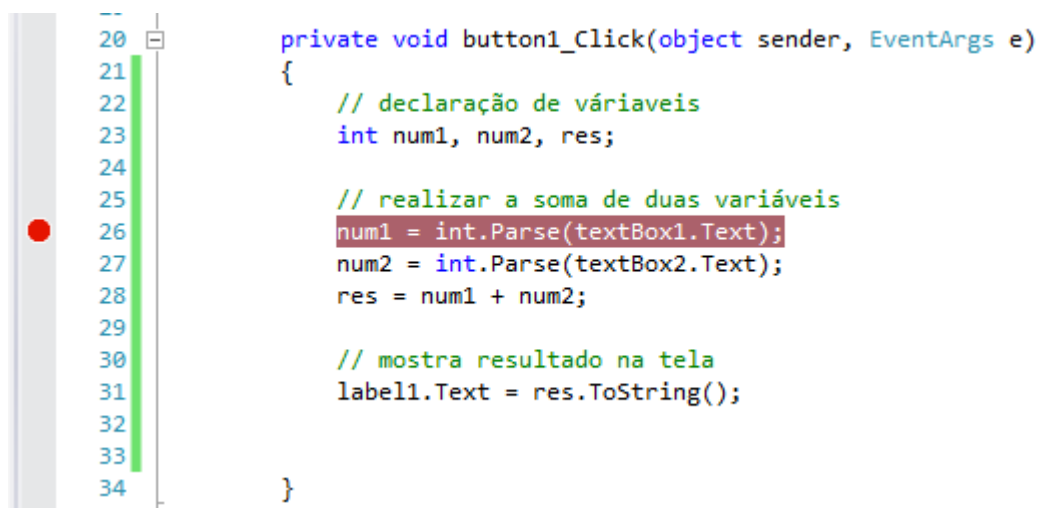
Para avaliarmos o processo de execução do código utilizamos “Step Over” para avançar ou “Step Out” para retroceder.

Ao colocarmos o mouse por cima do objeto a ser analisado teremos seu resultado apresentado, exemplo:

```
// realizar a soma de duas variáveis  
num1 = int.Parse(textBox1.Text);  
num1 12 (textBox2.Text);  
res = num1 + num2;
```

## “Breakpoints”

- Adicione “breakpoints” utilizando o menu “Debug” e selecionando “Toggle Breakpoint” ou utilize a tecla de atalho “F9”, um “breakpoint” também pode ser adicionado clicando com o mouse na lateral da tela de codificação.

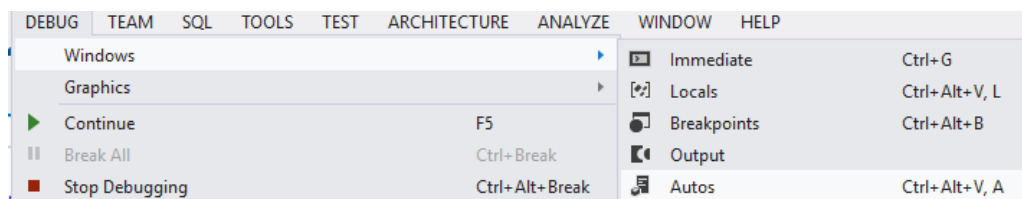


Para continuar o debug basta utilizar “Step Over” e “Step Out”.

Podemos analisar as variáveis e objetos dentro de uma janela ao invés de ficarmos colocando o mouse por cima de cada objeto do código.

Para que isso seja possível selecione na aba de menu “Debug” a opção “Windows - Autos”.

Obs: O modo Debug deve estar ativo (F8 ou F11).



O modo Debug deve estar ativo –

F9 Cria um ponto de breakpoints

F5 Inicia debug do código até o breakpoint marcado.

F11 step over acompanha execução linha por linha do código depois do breakpoint marcado.).

## Aula 10- Versionamento de software

Cada vez que um software é construído e publicado (build and deploy) ele recebe uma nova versão de controle.

O versionamento de software é um processo de controle de versões estabelecido por meio de numerações diferentes.

Isso permite que os programadores saibam quando e quais alterações foram realizadas, acompanhando as mudanças aplicadas no software.

Além disso, permite que os usuários finais identifiquem as novidades e reconheçam as versões mais atualizadas.

### Porquê usar?

Algumas razões levam à necessidade de criação de uma nova versão de um software. As principais são:

- correções de bugs;
- inclusão ou extensão de funcionalidades;
- mudança de arquitetura ou re-fatoração de código;
- correções pequenas e ajustes estéticos.

Um esquema de versionamento de software bem montado deve usar identificadores baseados em sequências, em que cada versão é fornecida com um ou mais números e/ou letras em ordem crescente.

Os primeiros números representam as mudanças com o maior nível de significância.

Depois disso, eles vão demonstrando pequenas atualizações. Veja o exemplo abaixo:

- v1.2 para v1.3: pode indicar uma alteração importante na estrutura do software, como a inclusão e/ou exclusão de ferramentas;
- v1.01 para v1.02: pode representar uma pequena correção de bugs (falhas operacionais).

### Envio e resgate de versões

A principal função do *sistema de controle de versão (versionamento)* é armazenar todo o histórico de desenvolvimento do documento, desde o primeiro envio até sua última versão.

Isso permite que seja possível resgatar uma determinada versão de qualquer data mais antiga, evitando desperdício de tempo no desenvolvimento para desfazer alterações quando se toma algum rumo equivocado.

Cada "envio" é na maioria dos sistemas chamado de "**commit**" (às vezes "*submit*"), ou seja, efetivar as alterações no (ou "submeter" ao) repositório.

Cada envio produz uma nova versão no repositório e é armazenado como "uma fotografia" do momento.



## Aula 11 - GitHub

Git é um sistema de controle de versão de arquivos. Através deles podemos desenvolver projetos na qual diversas pessoas podem contribuir simultaneamente no mesmo, editando e criando novos arquivos e permitindo que os mesmos possam existir sem o risco de suas alterações serem sobrescritas.

O [Github](#) é um serviço web que oferece diversas funcionalidades extras aplicadas ao git. Resumindo, você poderá usar gratuitamente o github para hospedar seus projetos pessoais.

Além disso, quase todos os projetos/frameworks/bibliotecas sobre desenvolvimento open source estão no github, e você pode acompanhá-los através de novas versões, contribuir informando bugs ou até mesmo enviando código e correções. Se você é desenvolvedor e ainda não tem github, você está atrasado e essa é a hora de correr atrás do prejuízo.

**GitHub** é um serviço de armazenamento remoto de repositório git.

interface com usuario via web.

Padrão da industria para armazenamento de projetos de código aberto.

È maior hospedeiro de codigo fonte do mundo.

Planos pagos para repositorio privados.

### **Configurando o Git.**

Existem 2 pequenos passos para configurar o seu GIT para ter um acesso mais simplificado ao github. Aqui estaremos estabelecendo que, sempre que necessitar, você irá fornecer o seu login e senha ao GitHub.

Então, com o seu terminal git aberto, vamos digitar:

```
$ git config --global user.name "YOUR NAME"
```

```
$ git config --global user.email "YOUR EMAIL ADDRESS"
```

```
Barth@DESKTOP-20LVI6S MINGW64 ~  
$ git config --global user.name "wagnerbarth"  
  
Barth@DESKTOP-20LVI6S MINGW64 ~  
$ git config --global user.email "wagner@barth.com.br"
```

Exemplo:

Para verificar se tudo deu certo use o comando **"less .gitconfig"**

Obs: Você deve estar no diretório **"home"** representado por um **"~"** (til), caso não esteja no diretório home execute o comando **"cd ~"**

### Clonando um projeto para o seu computador.

Este processo traz para a sua máquina o projeto que está disponível no repositório, assim podemos nanutilizá-lo.

Para realizar o processo de clonagem utilizamos o comando:

```
git clone https://github.com/<username>/site.git
```

```
Barth@DESKTOP-20LVI6S MINGW64 ~  
$ git clone https://github.com/wagnerbarth/aulaTP/  
Cloning into 'aulaTP'...  
warning: You appear to have cloned an empty repository.
```

Obs: Como não havia nenhum arquivo em nosso projeto a mensagem de **"Warning"** foi apresentada.

### Subindo arquivos / diretórios para o repositório

Copie um arquivo ou um diretório contendo seus arquivos e, no diretório do seu projeto, digite o comando:



git status

```
Barth@DESKTOP-20LVI6S MINGW64 ~
$ cd aulaTP/

Barth@DESKTOP-20LVI6S MINGW64 ~/aulaTP (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        decisao-e-loop/

nothing added to commit but untracked files present (use "git add" to track)
```

Como podemos observar foi percebido que o seu diretório local possui uma pasta ou arquivo que não está no repositório, **“Untracked files”**.

Para subirmos os arquivos usamos o comando:

git add<files>

```
Barth@DESKTOP-20LVI6S MINGW64 ~/aulaTP (master)
$ git add decisao-e-loop/

Barth@DESKTOP-20LVI6S MINGW64 ~/aulaTP (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   decisao-e-loop/decisao-e-loop.sln
        new file:   decisao-e-loop/decisao-e-loop.v11.suo
        new file:   decisao-e-loop/decisao-e-loop/App.config
        new file:   decisao-e-loop/decisao-e-loop/Program.cs
        new file:   decisao-e-loop/decisao-e-loop/Properties/AssemblyInfo.cs
        new file:   decisao-e-loop/decisao-e-loop/bin/Debug/decisao-e-loop.exe
        new file:   decisao-e-loop/decisao-e-loop/bin/Debug/decisao-e-loop.exe.config
        new file:   decisao-e-loop/decisao-e-loop/bin/Debug/decisao-e-loop.pdb
        new file:   decisao-e-loop/decisao-e-loop/bin/Debug/decisao-e-loop.vshost.exe
        new file:   decisao-e-loop/decisao-e-loop/bin/Debug/decisao-e-loop.vshost.exe.config
        new file:   decisao-e-loop/decisao-e-loop/bin/Debug/decisao-e-loop.vshost.exe.manifest
        new file:   decisao-e-loop/decisao-e-loop/decisao-e-loop.csproj
        new file:   decisao-e-loop/decisao-e-loop/decisao-e-loop.csproj.user
        new file:   decisao-e-loop/decisao-e-loop/Debug/DesignTimeResolveAssemblyReferencesInput.cache
        new file:   decisao-e-loop/decisao-e-loop/obj/Debug/TemporaryGeneratedFile_036C0B5B-1481-4323-8D20-8F5ADCB23D92.cs
        new file:   decisao-e-loop/decisao-e-loop/obj/Debug/TemporaryGeneratedFile_5937a670-0e60-4077-877b-f7221da3dda1.cs
        new file:   decisao-e-loop/decisao-e-loop/obj/Debug/TemporaryGeneratedFile_E7A71F73-0F8D-4B9B-B56E-8E70B10BC5D3.cs
        new file:   decisao-e-loop/decisao-e-loop/obj/Debug/decisao-e-loop.csproj.FileListAbsolute.txt
        new file:   decisao-e-loop/decisao-e-loop/obj/Debug/decisao-e-loop.exe
        new file:   decisao-e-loop/decisao-e-loop/obj/Debug/decisao-e-loop.pdb
```

Podemos observar que novos arquivos foram adicionados ao repositório, porém, eles ainda não estão por definitivo, este processo apenas adicionou os arquivos a um “index” do Git.

## Commit (adicionar ao repositório)

Ao aplicarmos um **“commit”** os arquivos serão adicionados ao repositório local e ao Git.

Devemos sempre informar o que está sendo feito neste **“commit”** pois estas informações serão úteis para levantarmos os dados necessários sobre as versões. Para realizar um **“commit”** utilize o comando:

**gitcommit -m “Comentários...”**

```
Barth@DESKTOP-20LVI6S MINGW64 ~/aulaTP (master)
$ git commit -m "Início do projeto -decisão-e-loop- primeira versão 1.0"
[master (root-commit) 66dfb48] Início do projeto -decisão-e-loop- primeira versão 1.0
20 files changed, 351 insertions(+)
create mode 100644 decisao-e-loop/decisao-e-loop.sln
create mode 100644 decisao-e-loop/decisao-e-loop.v11.suo
create mode 100644 decisao-e-loop/decisao-e-loop/App.config
create mode 100644 decisao-e-loop/decisao-e-loop/Program.cs
create mode 100644 decisao-e-loop/decisao-e-loop/Properties/AssemblyInfo.cs
create mode 100644 decisao-e-loop/decisao-e-loop/bin/Debug/decisao-e-loop.exe
create mode 100644 decisao-e-loop/decisao-e-loop/bin/Debug/decisao-e-loop.exe.config
create mode 100644 decisao-e-loop/decisao-e-loop/bin/Debug/decisao-e-loop.pdb
create mode 100644 decisao-e-loop/decisao-e-loop/bin/Debug/decisao-e-loop.vshost.exe
create mode 100644 decisao-e-loop/decisao-e-loop/bin/Debug/decisao-e-loop.vshost.exe.config
create mode 100644 decisao-e-loop/decisao-e-loop/bin/Debug/decisao-e-loop.vshost.exe.manifest
create mode 100644 decisao-e-loop/decisao-e-loop/decisao-e-loop.csproj
create mode 100644 decisao-e-loop/decisao-e-loop/decisao-e-loop.csproj.user
create mode 100644 decisao-e-loop/decisao-e-loop/obj/Debug/DesignTimeResolveAssemblyReferencesInput.cache
create mode 100644 decisao-e-loop/decisao-e-loop/obj/Debug/TemporaryGeneratedFile_036C0B5B-1481-4323-8D20-8F5ADCB23D92.cs
create mode 100644 decisao-e-loop/decisao-e-loop/obj/Debug/TemporaryGeneratedFile_5937a670-0e60-4077-877b-f7221da3dda1.cs
create mode 100644 decisao-e-loop/decisao-e-loop/obj/Debug/TemporaryGeneratedFile_E7A71F73-0F8D-4B9B-B56E-8E70B10BC5D3.cs
create mode 100644 decisao-e-loop/decisao-e-loop/obj/Debug/decisao-e-loop.csproj.FileListAbsolute.txt
create mode 100644 decisao-e-loop/decisao-e-loop/obj/Debug/decisao-e-loop.exe
create mode 100644 decisao-e-loop/decisao-e-loop/obj/Debug/decisao-e-loop.pdb
```

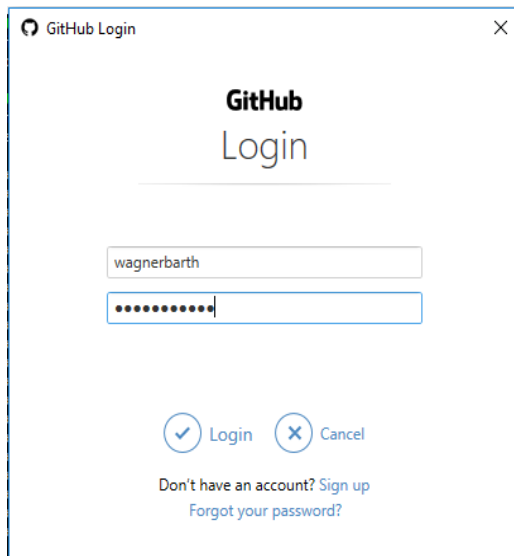
## Publicando o projeto no GitHub

Para publicarmos o projeto n GitHub e deixarmos disponível para que todos os usuários utilizem o projeto devemos utilizar o comando:

**gitpush**

```
Barth@DESKTOP-20LVI6S MINGW64 ~/aulaTP (master)
$ git push
```

Será solicitado o usuário e senha para que o processo possa ser realizado.



Após o processo de “**push**” serão apresentadas as informações de carregamento (publicação) no GitHub.

```
Barth@DESKTOP-20LVI6S MINGW64 ~/aulaTP (master)
$ git push
Enumerating objects: 23, done.
Counting objects: 100% (23/23), done.
Delta compression using up to 4 threads
Compressing objects: 100% (18/18), done.
Writing objects: 100% (23/23), 26.49 KiB | 3.78 MiB/s, done.
Total 23 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), done.
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:   https://github.com/wagnerbarth/aulaTP/pull/new/master
remote:
To https://github.com/wagnerbarth/aulaTP/
 * [new branch]      master -> master
```

## Aula 12-Estrutura de Repetição- FOR

Estrutura de repetição usado para repetir o Código determinadas vezes.

O início, o for executa a primeira vez. Se condição for verdade ele faz o incremento, se falso e pula fora.

Observação: usa-se quando se você conhece antecipadamente a quantidade de repetições.

**For ( int i=1; i<=4, i ++){For( **início**; **condição**; **incremento**)**

Comandos 1

Comandos 2

}

```
for (int i = 1; i < 0; i++) {
```

```
Console.WriteLine(" Boa noite Jovem ");
```

```
Console.WriteLine();
```

```
Console.WriteLine(" Gloria a Deus ");
```

```
}
```

```
Console.ReadLine();
```

```
int n = 0;
```

```
for (inti = 1; i<= 4; i++)    {
```

```
Console.WriteLine(" Digite o valor de n ");
```

```
n = int.Parse(Console.ReadLine());
```

```
}
```

```
Console.ReadLine();
```

1. Desenvolva uma Aplicação console C# chamada Intervalo100 que exiba os números de 1 a 100 na tela.
2. Desenvolva uma Aplicação console C# chamada números que exiba os números de 1 a 50 na tela em ordem decrescente.
3. uma Aplicação console C# chamada NumerosImpares que exiba os números ímpares entre 100 e 200.
4. Desenvolva uma Aplicação console C# chamada ServicoMilitar que leia os dados de "4" pessoas (nome, sexo, idade e altura). Se altura maior 1,80 e idade maior que 18, informar está apta ou não para cumprir o serviço militar obrigatório.
5. Desenvolva uma aplicação em C# que leia a idade e exiba quantidade pessoas possuem mais de 18 anos e a quantidade de pessoas menores de 18 anos. O código deverá ler a idade de 6 pessoas
6. Desenvolva uma Aplicação console C# chamada ControleDeEstoque que receba o preço de custo e o preço de venda de 8 produtos. Mostre como resultado se houve lucro, prejuízo ou empate para cada produto. Informe média de preço de custo e do preço de venda.
7. Desenvolva uma Aplicação console C# chamada CalculaComissaoVendedor que realize o cálculo de comissão dos vendedores de uma loja, levando-se em consideração que a comissão do vendedor será de 5% do total de vendas de acordo com os seguintes dados:
  - a. Nome do vendedor.
  - c. Código da peça.
  - d. Descrição da peça.
  - e. Preço unitário da peça.
  - f. Quantidade vendida.
  - g. Exiba as seguintes informações:  
nome do vendedor, código e descrição da peça, valor unitário, quantidade vendida e comissão do vendedor.
2. Utilizando estrutura de repetição for, Desenvolva uma aplicação console C# chamada testePositivo, que leia um número inteiro, e depois dizer se este número é negativo ou positivo. Repetição de 4 vezes  
Exemplos: Entrada: -10, 8 0;  
Saída: negativo, não negativo, não negativo.

## Aula 13- Estrutura de controle condicional switch “ case ”.

Switch/case é uma estrutura de condição que define o código a ser executado com base em uma comparação de valores. Para que isso fique mais claro, vejamos a sintaxe do `switch/case`:

```
switch (N)
{
case valor1:
    // comando1
    // comando 2
break;
case valor2:
    // comando 1
    // comando 2
break;

default:
    //comando;
break:
}
```

Na linha 1, em `switch (variável ou valor)`, definimos a variável ou valor que desejamos comparar. Na linha 3, informamos que se o valor declarado neste **case** for igual ao contido no **switch**, `código 1` será executado. O mesmo comportamento se aplica ao segundo `case`. Ademais, caso o valor contido no `switch` não seja atendido em uma das condições, nenhum bloco será executado.

E o comando `break`? O comando **break** é utilizado para especificar a última linha de código a ser executada dentro da condição. Se não declarado, os códigos implementados dentro dos **cases** subsequentes serão executados.

**Exemplo. Switch case**

```
int n1 = 1;

Console.WriteLine(" Escreva um numero :");
n1 = int.Parse(Console.ReadLine());

switch (n1)
{

case 2:
Console.WriteLine(" o dia da semana é segunda-feira");
break;

case 3:
Console.WriteLine(" o dia da semana é terça-feira ");
Break;

case 4:
Console.WriteLine(" o dia da semana é Quarta-feira");
break;

case 5:
Console.WriteLine(" o dia da semana é Quinta-feira");
break;

case 6:
Console.WriteLine(" o dia da semana é sexta-feira");
break;

case 7:
Console.WriteLine(" o dia da semana é sábado");
break;

default:
Console.WriteLine(" Numero invalido");
break;
}

Console.ReadLine();
```

**Exercicios switch case.**

1-Desenvolva um programa em C# chamado switch-desconto que receba a quantidade do produto, o valor do produto. O usuário deve escolher na tela a opção de desconto ( exibido na tela as opções) usando estrutura de switch case para que o programa execute conforme o modelo que deva ser impresso na tela:

Opção 1 -Desconto de 10%.

Opção 2- Sem desconto.

Opção 3- Desconto 15%.

O programa deverá calcular o desconto conforme escolha.

2-Desenvolva um programa em C# chamado switch-calculadora que mostre as seguintes opções ( adição, subtração, multiplicação e divisão) e realize os que se pede em cada uma delas. O usuário deverá informar dois números de entrada e escolher a opção desejada. Use estrutura switch case.

1 – Adição;

2 – Subtração;

3 – Multiplicação;

4 – Divisão;

3-Desenvolva um programa em C# chamada switch-aumento para ler um salário e utilizando estrutura switch case para Calcular o aumento de salario, o desconto de INSS. O usuário deve escolher a opção desejada ( exibir as opções). Apresentar o Novo salário atualizado e o desconto de INSS sobre o novo salário.

FAIXA SALARIAL	AUMENTO
<b>Opção 1</b>	9%
<b>Opção 2</b>	15%
<b>Opção 3</b>	20%

4-Desenvolva um programa em C# chamada swith-salario para leia a quantidade de aulas dadas pelo professor e realize o cálculo conforme o nível do professor.

Para cada nível haverá um acréscimo de 30% de descanso remunerado no total final

Utilizando estrutura switch case para atualizar de acordo com a tabela abaixo :

<b>Professor Nível 1</b>	<b>R\$12,00</b>
<b>Professor Nível 2</b>	<b>R\$17,00</b>
<b>Professor Nível 3</b>	<b>R\$25,00</b>



## Aula 14 – Classe

Classe é uma estrutura de dados que combina ações (métodos , função ou operação) e atributos (dados, campos) em uma única unidade.

Uma classe fornece uma definição para *instâncias* da classe criadas dinamicamente, também conhecidas como *objetos*.

As classes dão suporte à **herança e polimorfismo**, mecanismos nos quais *classes derivadas* podem estender e especializar *classes base*.

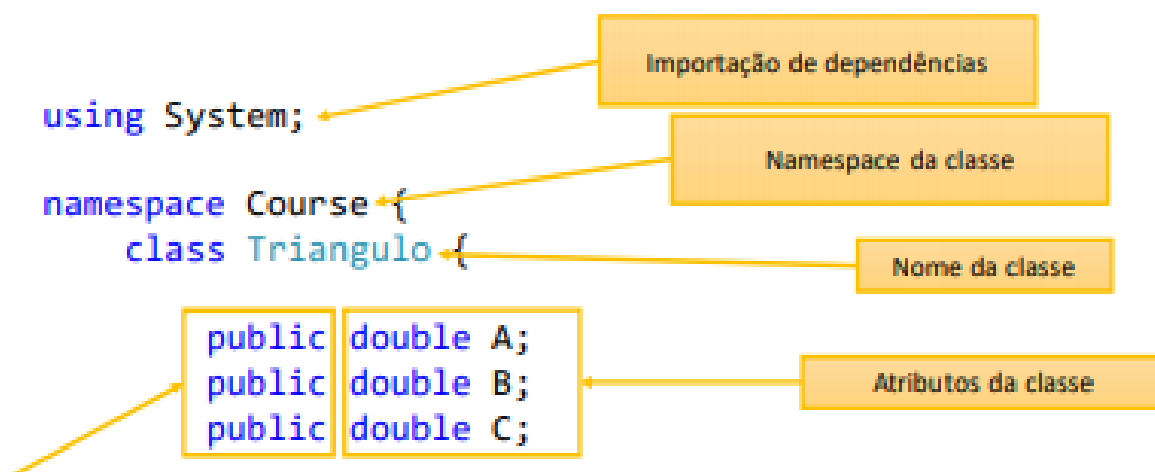
Classe É um tipo estruturado que pode conter (membros):

Atributos (dados / campos)

Métodos (funções / operações)

**A classe também pode prover muitos outros recursos, tais como:**

- Construtores
- Sobrecarga
- Encapsulamento
- Herança
- Polimorfismo



Class **public**: permite acesso a qualquer código externo a **classe**.

Class **private**: proíbe qualquer acesso externo à própria **classe**, inclusive das **classes** filhas.

Class **protected**: permite acesso às **classes** filhas, mas proíbe a qualquer outro acesso externo.

**Instanciação** é um processo por meio do qual se realiza a cópia de um objeto (classe) existente.

Uma classe, a qual tem a função de determinar um tipo de dado, deve ser instanciada para que possamos utilizá-la.

**Exemplo:**

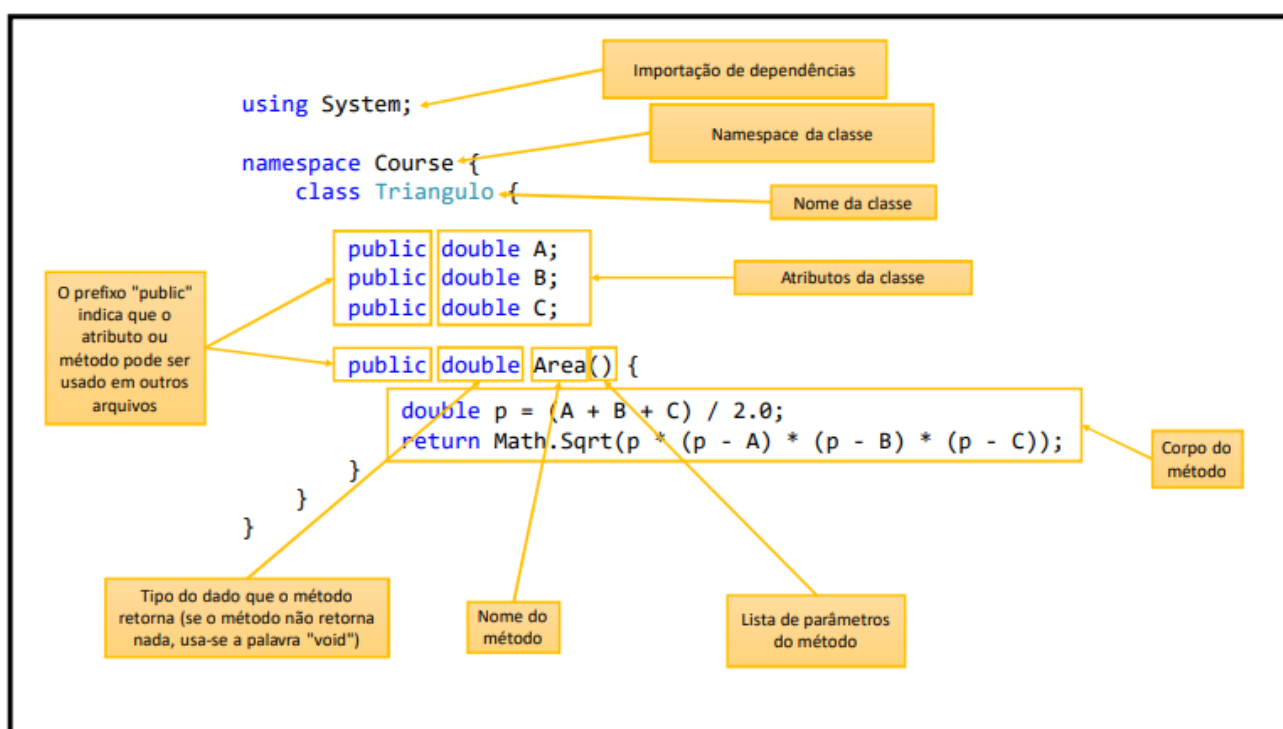
```
Pessoa p1 = new Pessoa();
Pessoa p2 = new Pessoa();
```

```
//Faço a declaração da minha classe e a instancio, tudo na mesma linha.
```

```
//Defino o valor dos campos criados na classe Pessoa.
```

```
Console.WriteLine(" Digite o nome da primeira pessoa");
p1.nome = Console.ReadLine();
Console.WriteLine(" Digite o nome da segunda pessoa");
p2.nome = Console.ReadLine();
```

**Método** é uma função dentro da classe. O método também é um membro da classe, por isso deve ficar dentro da classe.



1-Fazer um programa em C# para ler nome e salário de dois funcionários. Depois, mostrar o funcionário com maior salário. Para isso, crie uma classe chamada Funcionário e faça instanciação com a classe principal.

Dados do primeiro funcionário:

Nome: **Carlos Silva**

Salário: **6300.00**

Dados do segundo funcionário:

Nome: **Ana Marques**

Salário: **6700.00**

<b>Funcionario</b>
-Nome String
-Salario double

2-Fazer um programa em C# para ler os dados de duas pessoas, (nome, idade ) depois mostrar o nome da pessoa mais velha. Para isso, crie uma classe chamada Pessoa e faça instanciação com a classe principal .

Dados da primeira pessoa:

Nome: **Maria**

Idade: **17**

Dados da segunda pessoa:

Nome: **Joao**

Idade: **16**

<b>Pessoa</b>
-Nome String
-Idade int

3-Faça um programa que leia cotação do dólar, a quantidade dólar. Faça uma classe chamada Conversor que possua um método chamado ValorTotal para descobrir o total em reais. Também deve ser criado um método chamado IOF para descobrir o valor de 6% de IOF sobre o total.

<b>Cotação</b>
- Qte int
- Cotação double

4-Fazer um programa para ler o nome de um aluno e as três notas que ele obteve nos três trimestres do ano . Ao final, mostrar a média das notas e nome do aluno.

<b>Aluno</b>
-Nome String
-Nota double

## Aula 15- Método.

Um método é um bloco de código que contém uma série de instruções. Um programa faz com que as instruções sejam executadas chamando o método e especificando os argumentos de método necessários.

Toda Classe em C# é um subclasse da classe Object

Object Possui os seguinte métodos:

Get Type- retorna o tipo de objeto;

Equals- compara se o objeto é igual a outro;

GetHashCode- retorna um código hash do objeto;

Tostring- converte o objeto para string;

### Exemplo

```
{  
} class Produto  
{  
    public string Nome;  
    public double Preco;  
    public int Quantidade;  
  
    public double ValorTotalEstoque()  
    {  
        return Preco * Quantidade;  
    }  
}
```

## Exercicio

1-Faça um código fonte chamado codeDolar que leia cotação do dólar, a quantidade dólar. Faça uma classe chamada Conversor que possua um método chamado ValorTotal para descobrir o total em reais. Também deve ser criado um método chamado IOF para descobrir o valor de 6% de IOF sobre o total.

<b>Conversor</b>
- Qte int
- Cotação double
+ Valortotal():double
+ IOF() : double

2-Fazer um código fonte chamado codeEscola que leia o nome de um aluno e as três notas que ele obteve nos três trimestres do ano . Ao final, mostrar a média das notas e nome do aluno.

<b>Aluno</b>
-Nome String
-Nota 1 double
-Nota 2 double
+ Notafinal () double
+ Media (): double

3-Fazer um código fonte chamado Codefuncionario para ler os dados de um funcionário (nome, salário bruto e imposto). Criar um método chamado SalarioLiquido que apresente o salario liquido no final.Em seguida, mostrar os dados do funcionário (nome e salário líquido).

<b>Funcionario</b>
-Nome String
-SalarioBruto double
-Imposto double
+ SalarioLiquido():double

4-Fazer um código fonte chamado code2funcionario, que leia os dados de um funcionário (nome, salário bruto e imposto). Crie uma classe chamada Funcionario. Em seguida, mostrar os dados do funcionário (nome e salário líquido). Em seguida, crie método aumentar o salário do funcionário com base em uma porcentagem dada (somente o salário bruto é afetado pela porcentagem) e mostrar novamente os dados do funcionário. Use a classe projetada abaixo.

<b>Funcionario</b>
-Nome String
-SalarioBruto double
-Imposto double
+ SalarioLiquido():double
+Aumentarsalario() : double

5-Fazer um código fonte chamado cad2 que leia os dados de um produto em estoque ( nome, preço e quantidade em estoque)

Mostrar os dados do produto( nome, preço e quantidade em estoque;  
Realiza um entrada no estoque e mostrar novamente os dados do produto;  
Realizar um saída no estoque e mostra novamente os dados do produto.

<b>Produto</b>
-Nome String
-Preço double
-Quantidade int
+ Valortotal (): double
+ AdicionarEstoque():double
- RemoverEstoque() double

## Aula 16- Lista

lista é uma estrutura de dados, homogênea ( dados do mesmo tipo) e ordenada ( elementos acessados por meio de posições) . Inicia vazia e seus elementos são alocados sob demanda;

### Classe: list

Namespace: system.collections.Generic.

### Vantagens

tamanho variável

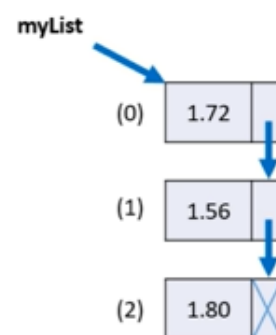
facilidade para realizar inserções e deleções;

### Desvantagens

acesso sequencial aos elementos;

## Listas

- Lista é uma estrutura de dados:
  - Homogênea (dados do mesmo tipo)
  - Ordenada (elementos acessados por meio de posições)
  - Inicia vazia, e seus elementos são alocados sob demanda
  - Cada elemento ocupa um "nó" (ou nodo) da lista
- Classe: List
- Namespace: System.Collections.Generic
- Vantagens:
  - Tamanho variável
  - Facilidade para se realizar inserções e deleções
- Desvantagens:
  - Acesso sequencial aos elementos \*



ETEC

```

List<string> Nome = new List<string>();
// crio lista de nomes tipo string;

Nome.Add(" Ana");
Nome.Add(" kIKO");
Nome.Add(" MARCIA ");
Nome.Insert(2, " Sofia ");

foreach (string obj in Nome)
{

    Console.WriteLine(obj);
}
Console.WriteLine( Nome.Count);

Console.ReadLine();

```

**Add-** adicionar na lista.

**Insert-** inserir nome na posição indicada.

**Nome.count=** informa tamanho da lista.

**Remove-** remove da lista.

```

// cria lista tipo string ,nome automovel

List <string> Automovel = new List<string> ();
Automovel.Add(" GOL");
Automovel.Add(" UNO ");
Automovel.Add(" FUSCA");
Automovel.Insert(1, " Ferrari ");

foreach ( string obj in Automovel){
    Console.WriteLine(obj);
}
// count, exibe tamanho de lista.
Console.WriteLine(" Tamanho da lista é : "+ Automovel.Count + " automoveis ");
Console.ReadLine();

Automovel.Remove(" FUSCA");
// REMOVE DA LISTA

Console.WriteLine();

foreach (string obj in Automovel)
{
    Console.WriteLine(obj);
}

Console.ReadLine();

```



## Aula 17- Botões e propriedades do Windows forms.

**label**= É um rotulo para seu aplicativo.

**button**= Representa um controle de botão do Windows.

**textBox**= São caixas de texto que armazenaram dados.

**ComboBox**= caixa onde seleciona o texto .

**Listbox**= caixa onde adicionados ou removidos listas.

**pictureBox**= Selecionar um imagem para o aplicativo.

### Exemplo de comando no Windows Forms

```
// declaração de variáveis  
  
int n1;  
  
n1 = Convert.ToInt16(textBox1.Text);  
// convert- converte o texto recebido em int, tipo da variavel n1.
```

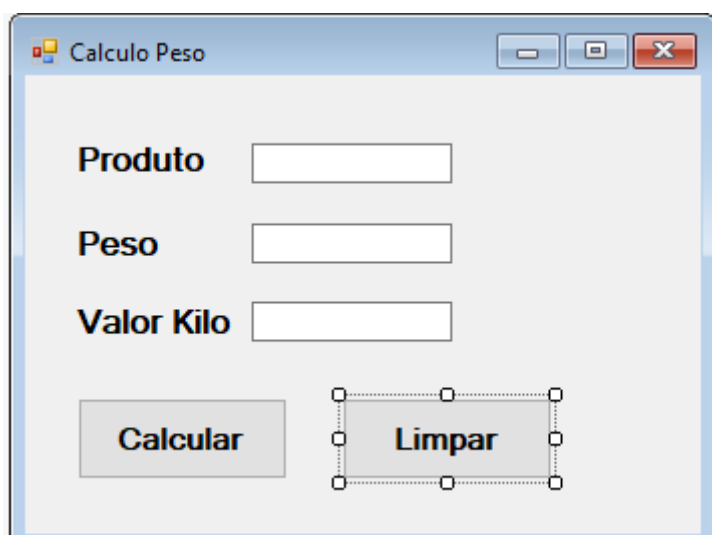
### Botão Limpar

```
textBox1nome.Clear();  
textBox2peso.Clear();  
textBox3altura.Clear();
```

## Aula 18 – Desenvolvendo no Windows Forms – calculo .

Desenvolva código fonte no Windows Forms que realize calculo do peso do produto X valor do produto. O código fonte deve ler nome do produto, peso do produto e o valor por kilo e exibir ao final.

Conforme modelo abaixo.



```
private void btn_calcular_Click(object sender, EventArgs e)
{
    string Nome;
    double Peso, Valor, Calculo;

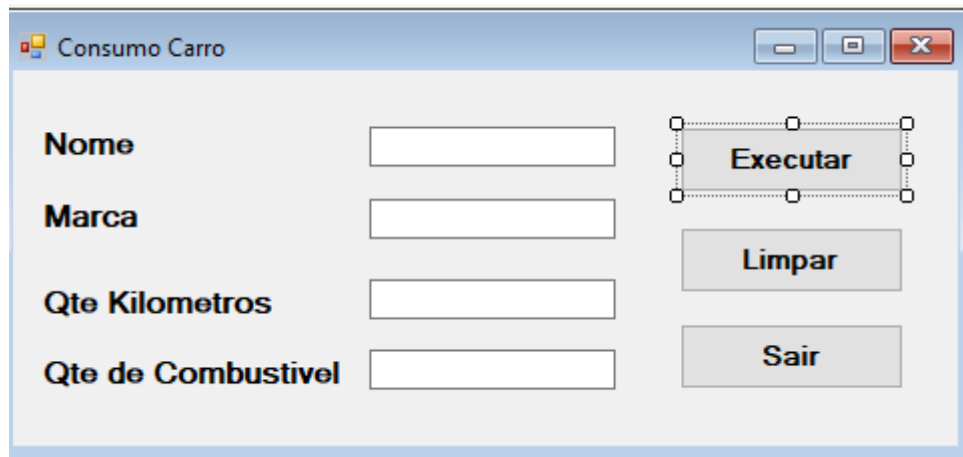
    Nome = textBox1.Text;
    Peso = Convert.ToDouble(textBox2.Text);
    Valor = Convert.ToDouble(textBox3.Text);
    Calculo = Valor * Peso;
    MessageBox.Show("O valor total do Produto " + Nome + " é R$ " + Calculo);
}

private void btn_limpar_Click(object sender, EventArgs e)
{
    textBox1.Clear();
    textBox2.Clear();
    textBox3.Clear();
}
```

## Aula-19 Desenvolvimento no Windows forms consumo de carro.

Desenvolva código fonte no Windows que realize cálculo do consumo de combustível de um determinado veículo. O código fonte deve ler o nome do veículo, marca, quantidade de quilômetros rodados, quantidade de combustível utilizado. Ao final, exibir se o veículo é econômico ou gasto utilizando estrutura condicional if.

Conforme modelo abaixo.



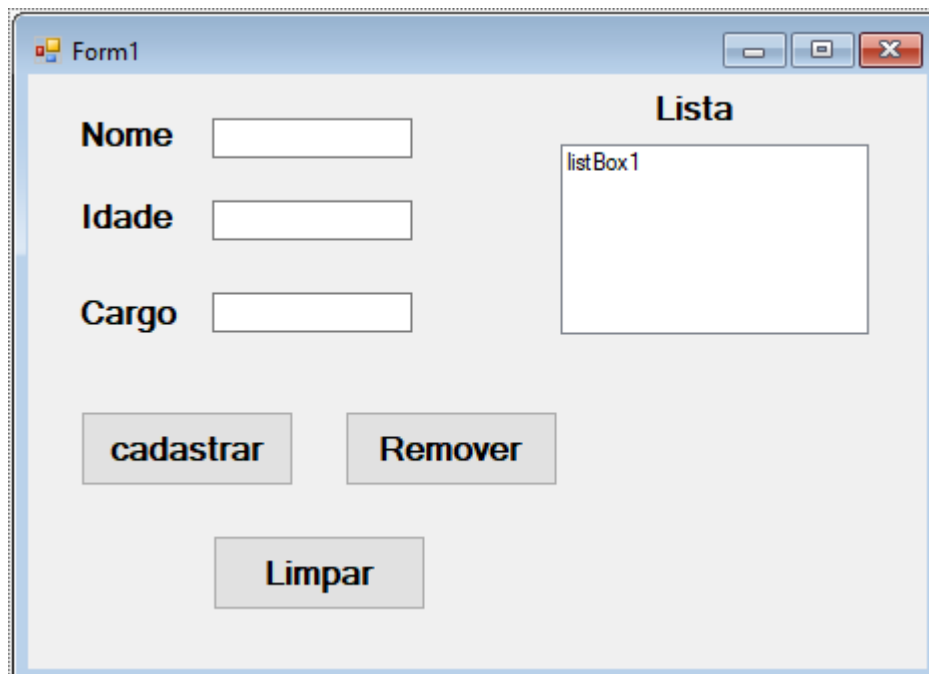
```
string Nome, Marca;
int Qtecombustivel;
int Qtekilometro, consumo;

Nome = textBox1.Text;
Marca = textBox2.Text;
Qtecombustivel = Convert.ToInt16(textBox3.Text);
Qtekilometro = Convert.ToInt16(textBox4.Text);
consumo = Qtecombustivel / Qtekilometro;

if( consumo<8){
    MessageBox.Show(Nome + " Veiculo gastão e consumo de " + consumo + " km por
litro");
}
else{
    MessageBox.Show( Nome + " Veiculo economico e consumo de " + consumo + " km
por litro");
}
```

## Aula-20 Desenvolvimento no Windows forms listbox.

Desenvolva código fonte no Windows que leia nome, idade e cargo e armazene numa Listbox. Adicionar botão para adicionar e remover da Listbox conforme modelo abaixo.



```
private void btn_adicionar_Click(object sender, EventArgs e)
{
    string nome, cargo;
    int idade;

    nome = textBox1.Text;
    idade = Convert.ToInt16(textBox2.Text);
    cargo = textBox3.Text;

    listBox1.Items.Add(string.Format("{0} {1} {2} ", nome, idade, cargo));
    messagebox.show(" Item adicionado");
}

private void btn_remover_Click(object sender, EventArgs e)
{
    listBox1.Items.Remove(listBox1.SelectedItem);
}

private void btn_limpar_Click(object sender, EventArgs e)
{
    textBox1.Clear();
    textBox2.Clear();
    textBox3.Clear();
}
```

**Combobox**= selecionar, caixa de seleção de dados;

**Quando usar:** EX. formulario de cadastro;

**Combobox**- tambem é uma classe, possui (metodos)

**Métodos-**

}

### Exercicio Classe Correção

1-Fazer um programa para ler os dados de um funcionário (nome, salário bruto e imposto). Em seguida, mostrar os dados do funcionário (nome e salário líquido).

Em seguida, aumentar o salário do funcionário com base em uma porcentagem dada (somente o salário bruto é afetado pela porcentagem) e mostrar novamente os dados do funcionário. Use a classe projetada abaixo.

```
Funcionario f = new Funcionario();

Console.WriteLine(" Digite o nome do funcionario:");
f.Nome = Console.ReadLine();
Console.WriteLine(" Digite o Salario Bruto :");
f.SalarioBruto = double.Parse(Console.ReadLine());
Console.WriteLine(" Digite o imposto do salario: ");
f.Imposto = double.Parse(Console.ReadLine());
Console.WriteLine(" Informações do funcionario: "+ f);
Console.WriteLine(" Digite a porcentagem de aumento:");
double percent= double.Parse(Console.ReadLine());
f.AumentaSalario(percent);
Console.WriteLine(" Informações do funcionario: "+ f);

Console.ReadLine();

public string Nome;
public double SalarioBruto;
public double Imposto;

public double SalarioLiquido()
{
    return SalarioBruto - Imposto;
}

public void AumentaSalario(double porcentagem)
{
    SalarioBruto = SalarioBruto + (SalarioBruto * porcentagem/100.0);
}

public override string ToString()
{
    return Nome
        + " R$, "
        + SalarioLiquido().ToString("f2", CultureInfo.InvariantCulture);
}
```

}