





Universidade do Vale do Paraíba Colégio Técnico Antônio Teixeira Fernandes Disciplina POO - 2019

Material III-Bimestre

Introdução e conceitos fundamentais da Linguagem C#

http://www1.univap.br/wagner

Prof. Responsável

Wagner Santos C. de Jesus











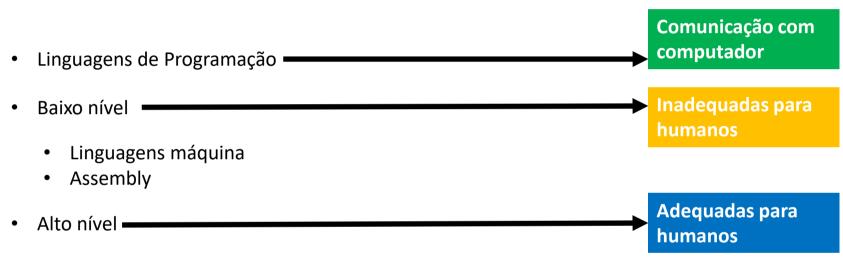
Introdução a Linguagem C#



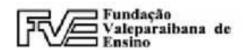




Linguagens de Programação



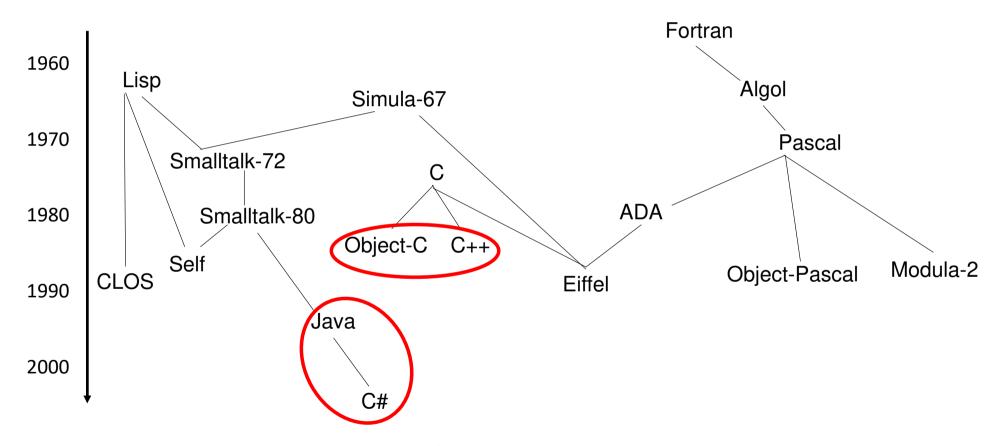
- PHP
- C++
- JAVA







Estudo sobre Linguagens de programação



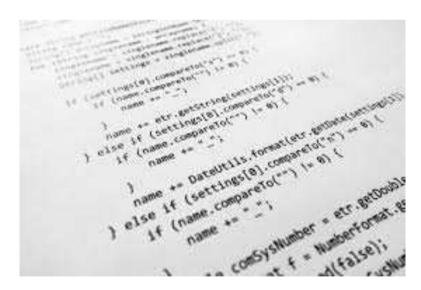






Arquitetura (.NET)

A linguagem C# foi criada junto com a arquitetura .NET. Embora existam várias outras linguagens que suportam essa tecnologia (como VB.NET, C++, J#), C# é considerada a linguagem símbolo do .NET, devido as funcionalidade da plataforma .NET terem sido desenvolvidas em C#.



Prof. Wagner Santos C. de Jesus wagner@univap.br







Histórico C#

A linguagem C# (Sharp) foi influencia da linguagens C++ e Java que ganhou popularidade na comunidade de informática quando a Microsoft decidiu incluí-la no pacote Visual Studio.

```
'resource_id' presource_details['id'],

if ( $this->rule_exists{ $resource_details['id'], $rule_details['id'],

if ( $access == false ) (

// Remove the rule as there is currently as most for it
$details['access'] = !$access;

$this-> sql->delete( 'acl_rules', $details);

} else {

// Update the rule with the new access unlow
$this-> sql->update( 'acl_rules', array( 'access' == page)

}

foreach( $this->rules as $keyestrule ) {

if ( $details['role_id'] == $rule['rele_id'] &= $details['role_id'] &= $rule['role_id'] &= $rul
```

Prof. Wagner Santos C. de Jesus wagner@univap.br







C# (CSharp) é uma linguagem de <u>programação orientada a objetos</u> criada pela Microsoft, faz parte da sua plataforma .Net. A companhia baseou C# na linguagem C++ e Java.







Linguagem de Programação Características Operacionais







Características

- O Java utiliza comentários Javadoc e o C# utiliza comentários baseados em XML;
- O Java utiliza a JVM, C# e .Net Framework.
- Compiladores de linha de comando.
- Compiladores para Windows, Windows Mobile, Linux, Mac OS X e Solaris.
- Orientação à Objetos.
- Manipulação de Banco de Dados.







Característica

- C# é mais seguro com os tipos que C++.
- A sintaxe para a declaração de vetores é diferente ("int[] a = new int[5]" ao invés de "int a[5]").
- Membros de enumeração são colocados em seu próprio espaço de nomes (namespace).
- C++ não possui modelos (templates), mas C# 2.0 possui genéricos (generics).
- Propriedades estão disponíveis, as quais permitem que métodos sejam chamados com a mesma sintaxe de acesso a membros de dados.
- Recursos de reflexão completos estão disponíveis
- 2003 tornou-se um padrão ISO (ISO/IEC 23270)- ECMA (**European Computer Manufacturers Association**) ECMA 334 (Padrões).







```
using System;
using System.Collections.Generic;
using System.Ling;
using System.Text;
namespace ConsoleApplication1
  class Program
    static void Main(string[] args)
```

Estrutura de um programa C# Aplicação (Console)

Determina pacotes que deverão ser usados pelo programa.

Declara escopo que contém um conjunto de classes relacionadas.

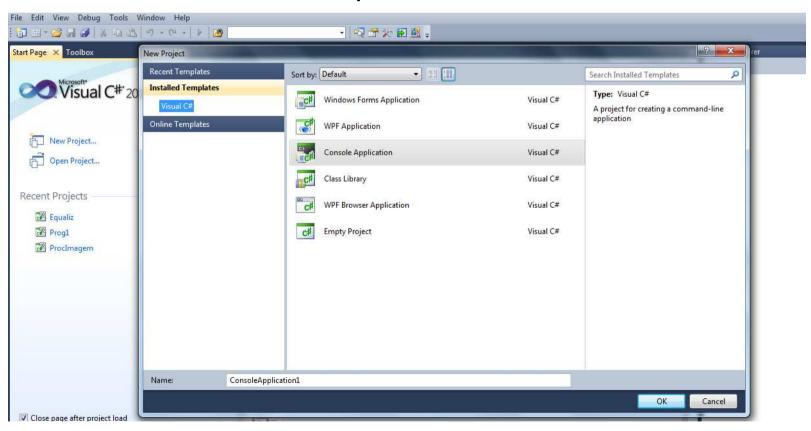
→ Área de Template







Janela Principal do Ambiente









Principais modelos para criação de aplicações.

Modelo	Tipos de Projetos Criados
Windows Forms Aplication (Aplicativo Windows Foms) (*)	Aplicação desktop para o sistema operacional Windows
WPF Application (Aplicativo WPF)	Aplicação Desktop para criação de templates (Código Reutilizáveis)
Console Application (Aplicativo de console) (*)	Aplicação que ocorre apartir de linha de comandos do Windows.







Linguagem de Programação Características Operacionais







Característica principais de uma Linguagem de Programação (C#)

- Sintaxe (Escrita)
- Semântica (Significado)
- Pragmática (Lógica)







Conceito

- Interpretadores Captura o código de fonte na linguagem correspondente e converte-o em linguagem de Máquina (atividade realizada por linhas de código).
- Compilador Captura o código de fonte na linguagem correspondente e converte-o em linguagem de Máquina (atividade realizada em uma única vez).



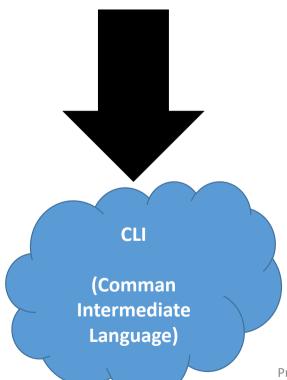




Compiladores e VM (Máquinas Virtuais)

CLR (Command Language Runtime)

Compiladores



(Máquinas Virtuais)

Prof. Wagner Santos C. de Jesus wagner@univap.br







Conceito de Framework

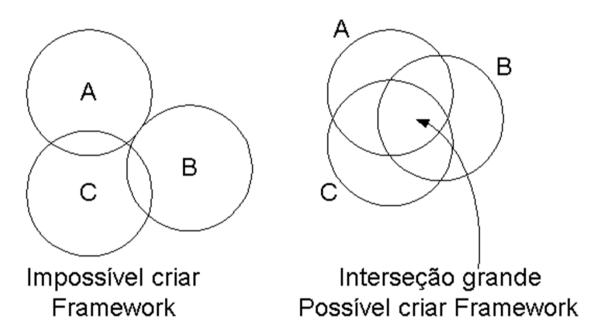






Conceito de Framework.

Um framework vem a ser uma ferramenta que possui como características captura a funcionalidade comum a várias aplicações.





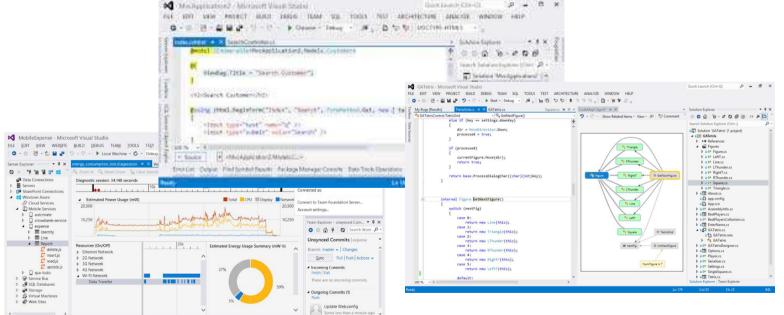




Características de FrameWork:

Aplicações Orientadas a Objetos;

Componentes









Tipos de Dados

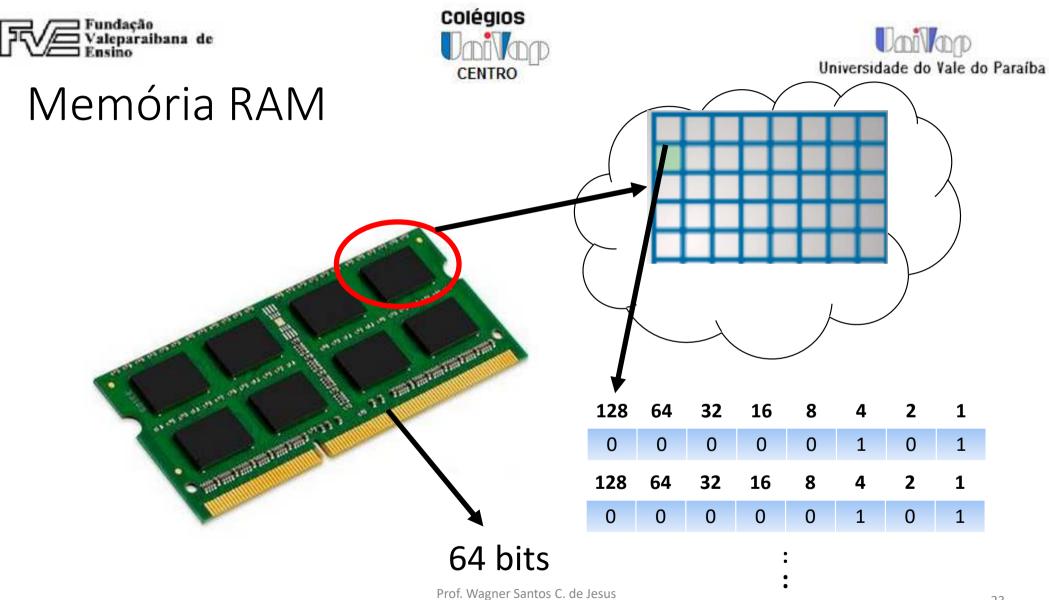






Conceito de Tipos de Dados

O tipo de dados representa o tipo de informação que uma variável ou uma constante pode armazenar estabelecendo seu limite mínimo e máximo de valores suportados.



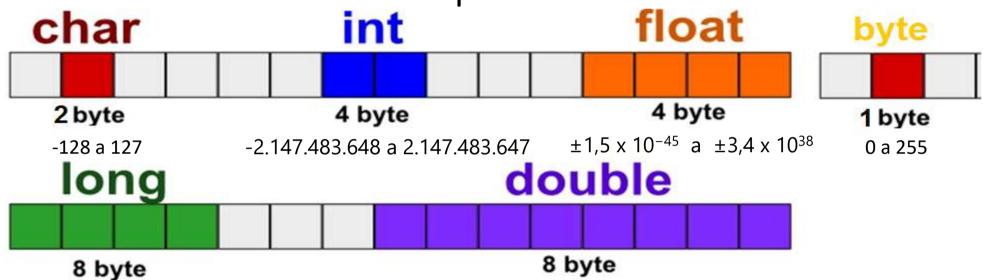
wagner@univap.br







Tabela de tipos de dados



-9.223.372.036.854.775.808 a 9.223.372.036.854.775.807

 $\pm 5.0 \times 10^{-324}$ a $\pm 1.7 \times 10^{308}$

char



Vetor de caracteres.







Exemplo

Inteiro de 3 bits (int3)

	4	2	1	
0	0	0	0	
1	0	0	1	
2	0	1	0	
3	0	1	1	
4	1	0	0	
5	1	0	1	
6	1	1	0	1
7	1	1	1	

$$2^{n}$$

$$n = 3$$
byte a = 5;







Tipos em C# (u - unsigned)

Tipo de Dados	Armazenamento
Byte, Ushort, Uint e Ulong	Números inteiros positivos
Sbyte, Short, Int e Long	Números inteiros positivos e negativos
Float, Double e Decimal	Números inteiros e reais (Positivos e negativos)
Char e String	Caracteres e cadeias de caracteres.
Bool	Valores lógicos







Literais (Valores pré-definidos)

Tipo de Dados	Sufixo	Exemplo
uint	U	uint valor = 35U;
long	L	long valor = 35L;
ulong	UL	ulong valor = 10UL;
float	F	Float valor = 20.3F;
double	D	double valor = 20.3D;
decimal	М	decimal valor = 20.3M;







Tipos Principais

Tipo de Dados	Tamanho	Exemplo
Int	4 bytes	int valor = 35;
long	8 bytes	long valor = 35;
byte	1 byte	byte valor = 255;
float	4 bytes	float valor = 20.3;
double	8 bytes	double valor = 20.3;
decimal	16 bytes	decimal valor = 20.3;
char	2 bytes	Apenas um caractere
string	-	Conjunto de caracteres
bool	1 byte	true/false (Booleano)







Operações Matemáticas de Lógicos







Operadores Aritméticos

- + Adição
- Subtração
- * Multiplicação
- / Divisão
- % Resto da Divisão (Módulo)







Operadores de atribuição

- = Atribuição simples
- += Atribuição com adição
- -= Atribuição com subtração
- *= Atribuição com multiplicação
- /= Atribuição com divisão
- %= Atribuição com módulo







Operadores Lógicos

! - Não inverso

&& - e lógico







Operadores Relacionais

- = = Igual
- != Diferente de
- < Menor que
- > Maior que
- >= Maior ou Igual
- <= Menor ou Igual







Operadores de Incremento e decremento

- ++ Incremento
- -- Decremento







Exemplo atribuição

```
int a = 5;
a++;  // Resultado de a = 6.
int y = 4;
int k = y++; // Resultado y = 5 e k = 4
```







Linguagem de Programação C# Estrutura Básica

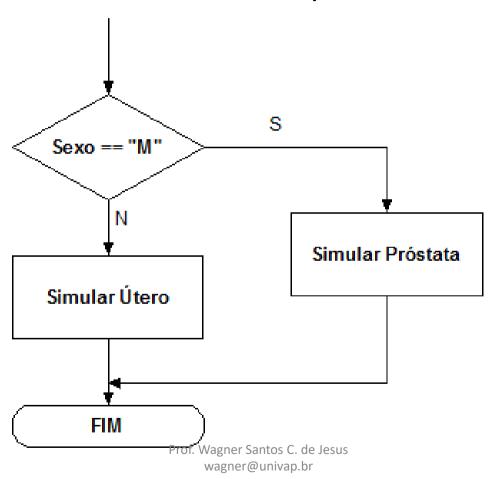
- Atribuição
 - Condição
- Repetição







Decisão Composta









Condicional if()

```
if(<condição>) {
     [Bloco-Instruções-1]
}
else {
     [Bloco-Instruções-2]
}
```







Condicional if()

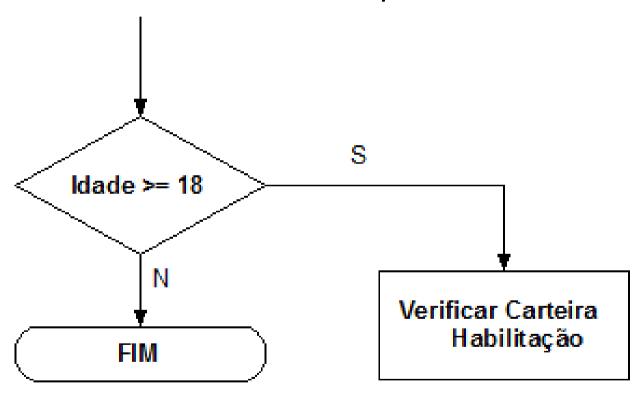
```
if(<condição>)
    [Instrução-1]
else
    [Instrução-2]
```







Decisão Simples









Exemplo de if()

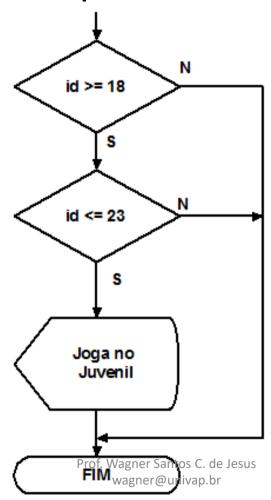
```
char sexo = 'M';
if(sexo == 'F') {
  idade = 45;
  cintura = 20;
  nasc = "Brasileira";
}
```







Decisão Dupla duas condições









Séries de Instruções Condicionais







```
switch(<expressão>) {
switch(): cadeia de condição
                                          case valor1:
                                            [bloco de instruções]
                                            break;
                                          case valor2:
                                            [bloco de instruções]
                                            break;
                                          default:
                                             [bloco de instruções]
                                             break;
```







Exemplo switch()

```
switch(categoria) {
  case 1:
     bonus = salario * 0.1;
     break;
  case 6:
     bonus = salario * 0.07;
     break;
  case 7:
      bonus = 100;
      break;
  default:
      bonus = 0;
         break;
```

A instrução break deve ser usada para que haja abandono da estrutura.







Estrutura de repetição







while():Enquanto a condição verdadeira executa.

```
while (<condição>) {
    [bloco de instruções]
}
    do{
     [bloco de instruções]
    } while (<condição>);
```







Exemplo: while()

```
int num = 20;
int contador = 0;
while (num > 10)
{
    num = num - 1;
    contador = contador + 1;
}
```







Exemplo: do while()

```
int num = 8;
int contador = 0;
do {
   num = num - 1;
   contador = contador + 1;
} while (num != 0);
```







Instrução for()

```
for(<inicio>;<condição>;<atualização>)
{
    [bloco de instruções]
}
```







Exemplo do for()

```
int num = 8;
int contador = 0;
for (int i = 0; i <= 10; i++)
{
    num = num - 1;
    contador = contador + 1;
}</pre>
```







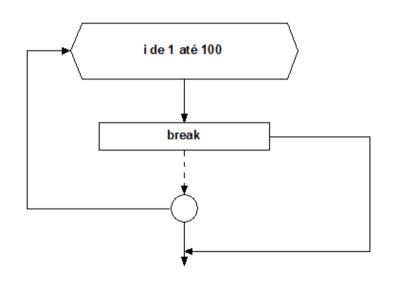
Quebra e permanência de ciclo em estruturas de Controle.



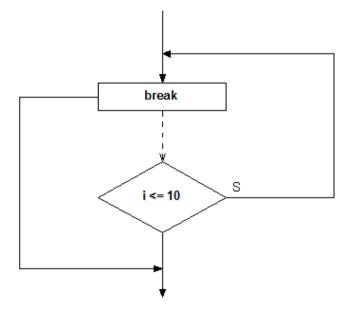




Quebra de ciclo (break)



Estrutura for (para)



Estrutura while (enquanto)







Exemplo Prático (break)

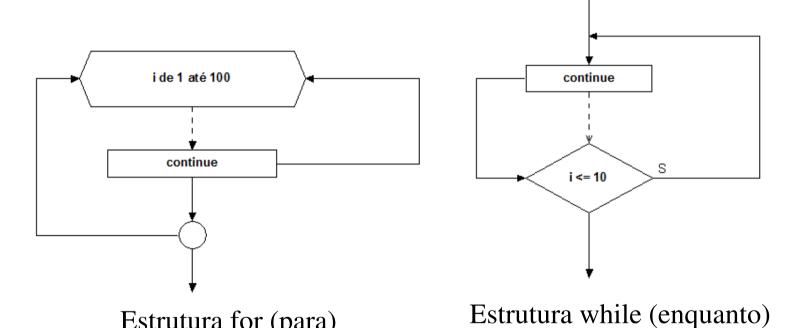
```
int num = 8;
int contador = 0;
while (num != 0) {
  if (contador == 2)
    break;
  num = num - 1;
  contador = contador + 1;
}
```







Permanência de ciclo (continue)



Estrutura for (para)

Prof. Wagner Santos C. de Jesus wagner@univap.br







Exemplo Prático (continue)

```
int num = 8;
 int contador = 0;
while (num != 0) {
   if (contador == 2)
      num = 0;
      continue;
    num = num - 1;
    contador = contador + 1;
    Prof. Wagner Santos C. de Jesus
       wagner@univap.br
```







Entrada e Saída de Dados

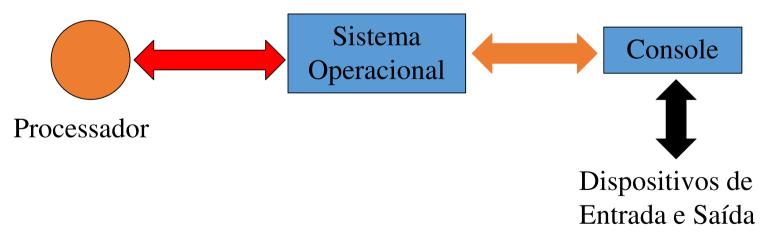






Classe: Console

Representa a entrada, as saídas, e os fluxos de erro padrão para aplicativos do console. Esta classe não pode ser herdada.









Entrada de dados Métodos read e readLine()

Método	Descrição
read()	Realiza a leitura de um caractere pelo teclado.
readLine()	Realiza a leitura de uma String pelo teclado.







Saída de dados Métodos write e writeLine()

Método	Descrição
write()	Realiza a escrita de uma string sem salto de linhas.
writeLine()	Realiza a escrita de uma String com salto de linhas.







Exemplo de Entrada e Saída

String str = Console.ReadLine(); Console.WriteLine(str);







Conversão de Tipos de Dados







Método: Parse()

Converte uma String em um número de acordo com seu tipo de dado especificado.







Exemplo de conversão de tipos

```
double num1 = double.Parse(Console.ReadLine());
double num2 = double.Parse(Console.ReadLine());
double soma = num1 / num2;
Console.WriteLine(soma);
```