

Universidade do Vale do Paraíba
Colégio Técnico Antônio Teixeira Fernandes
Disciplina POO - 2017

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#

Exemplo:

```
<?php
$vet01 = array();
$vet01[] = "Sistemas operacionais";
$vet01[] = "Compiladores";
$vet01[] = "Bancos de dados";

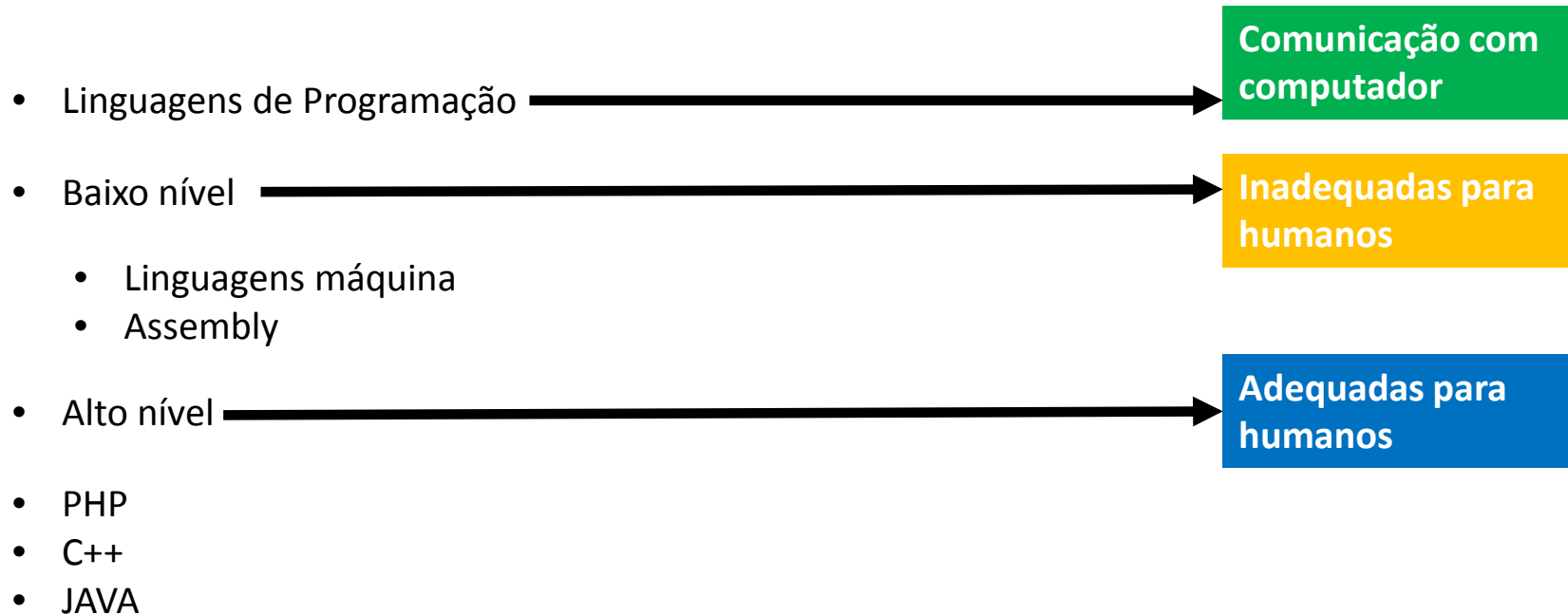
$vet02 = array(1, 2, 3, 4, 5);

$vet03 = array(0 => 0, 2 => 3, 10 => "sem 10");

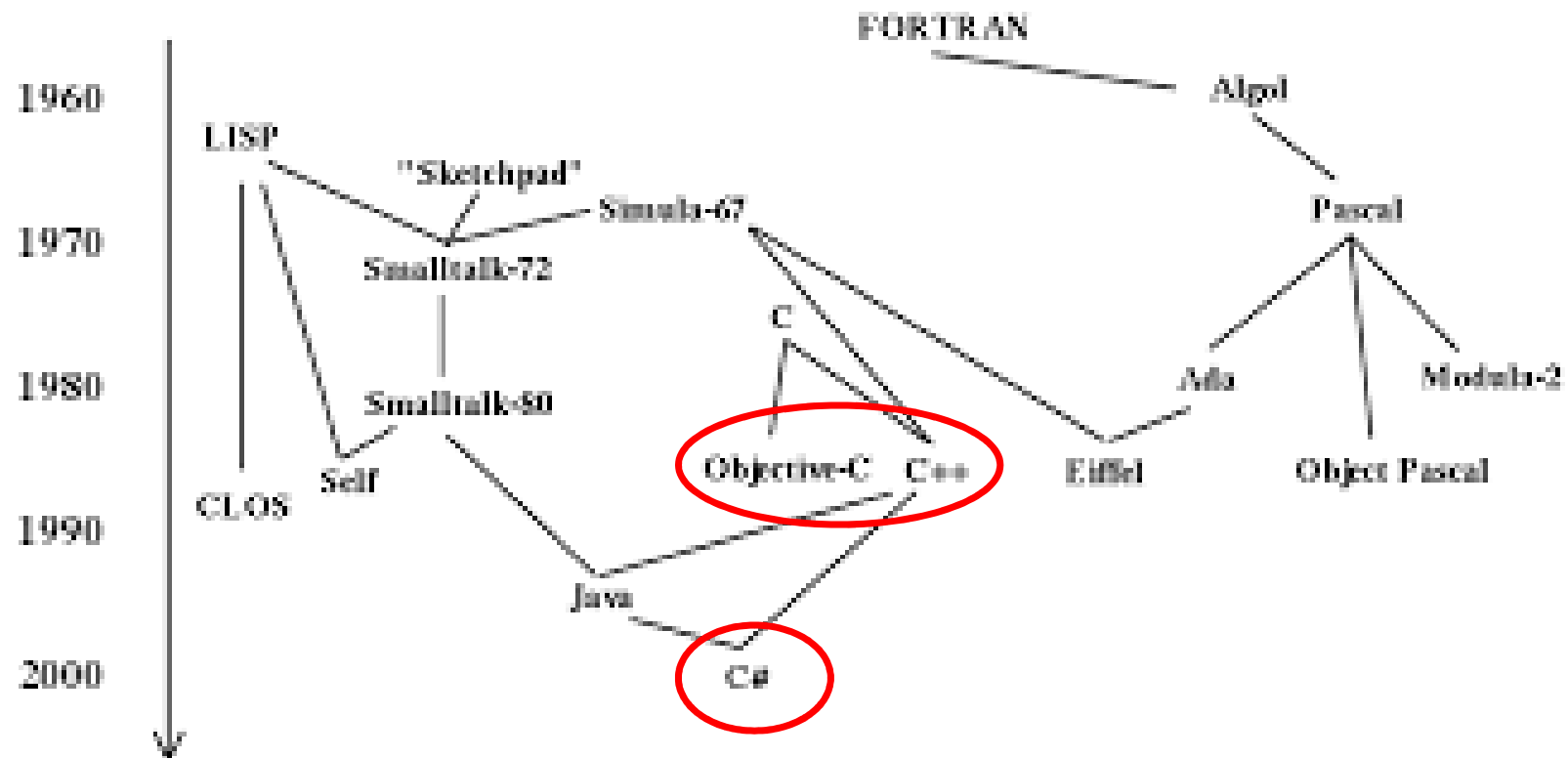
for ($i = 0; $i < count($vet01); $i++) {
    echo $vet01[$i] . "<br />";
}

?>
```

Linguagens de Programação



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#.



Histórico C#

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

```
        'resource_id' => $role_details['id'],
        'resource_id' => $resource_details['id'],
    );
    if ( $this->rule_exists( $resource_details['id'], $role_details['id'] ) ) {
        if ( $access == false ) {
            // Remove the rule as there is currently no need for it
            $details['access'] = ! $access;
            $this->_sql->delete( 'acl_rules', $details );
        } else {
            // Update the rule with the new access value
            $this->_sql->update( 'acl_rules', array( 'access' => $access ) );
        }
    }
    foreach( $this->rules as $key=>$rule ) {
        if ( $details['role_id'] == $rule['role_id'] && $details['resource_id'] == $rule['resource_id'] ) {
            if ( $access == false ) {
                unset( $this->rules[ $key ] );
            } else {
                $this->rules[ $key ]['access'] = $access;
            }
        }
    }
}
```

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

```
55 function array_from_string(a) {  
56     a = a.replace(/(\r\n|\n\r)/g, " ");  
57     a = replaceAll(" ", " ", a);  
58     a = a.replace(/ +(?= )/g, "");  
59     return a.split(" ");  
60 }  
61 $("#unique").click(function() {  
62     var a = array_from_string($("#efis").val()), b = $("#elast,w").val();  
63     if (c < 2 * b - 1) {  
64         return function("check" + c), this.trigger("click");  
65     }  
66     for (b = 0; b < a.length; b++) {  
67         if (a[b] && a[b] != a[b]) a.splice(b, 1);  
68     }  
69 }
```

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).

Estrutura de um programa C# Aplicação (Console)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

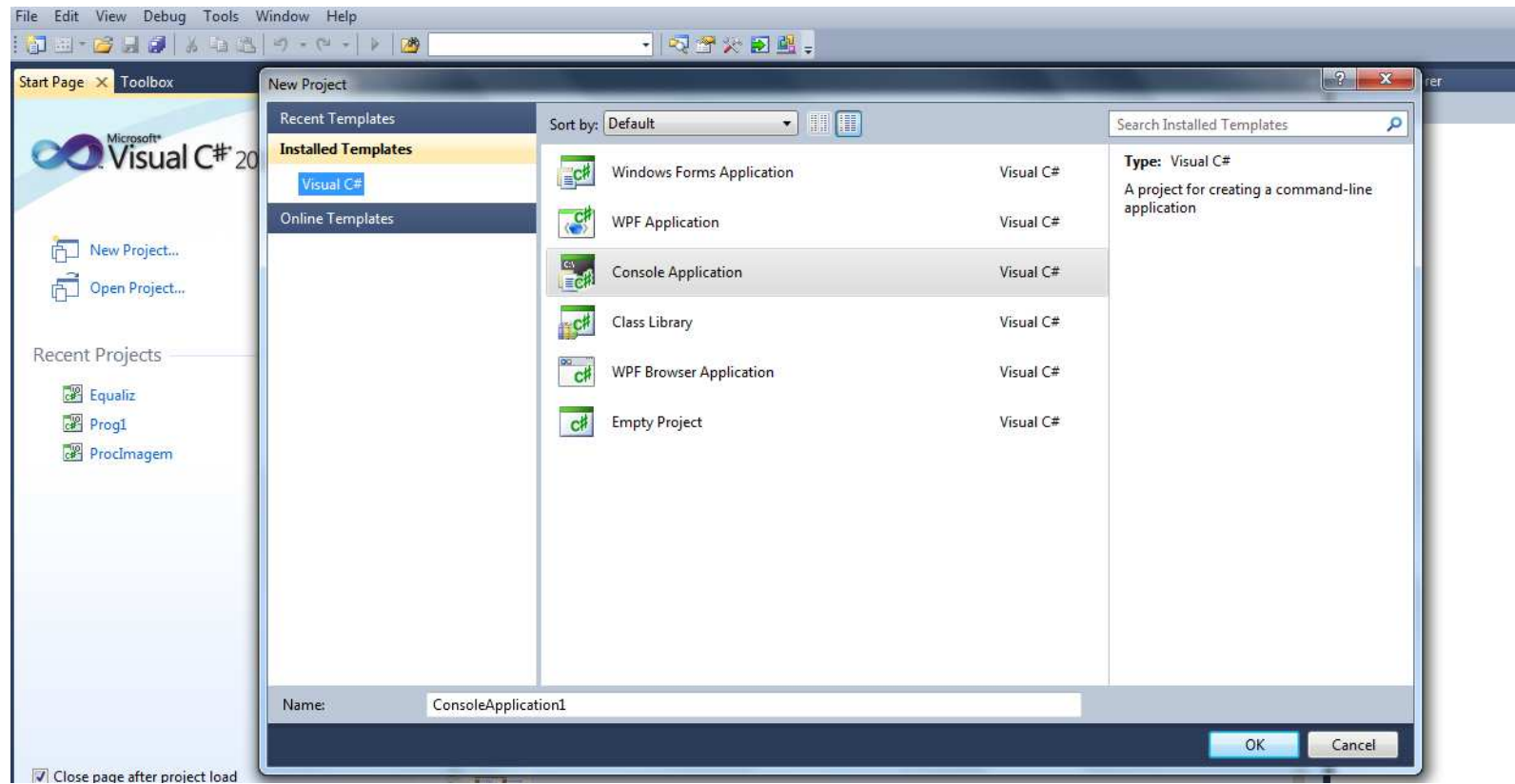
namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            .....
        }
    }
}
```

→ 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 Application (Aplicativo Windows Forms) (*)	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

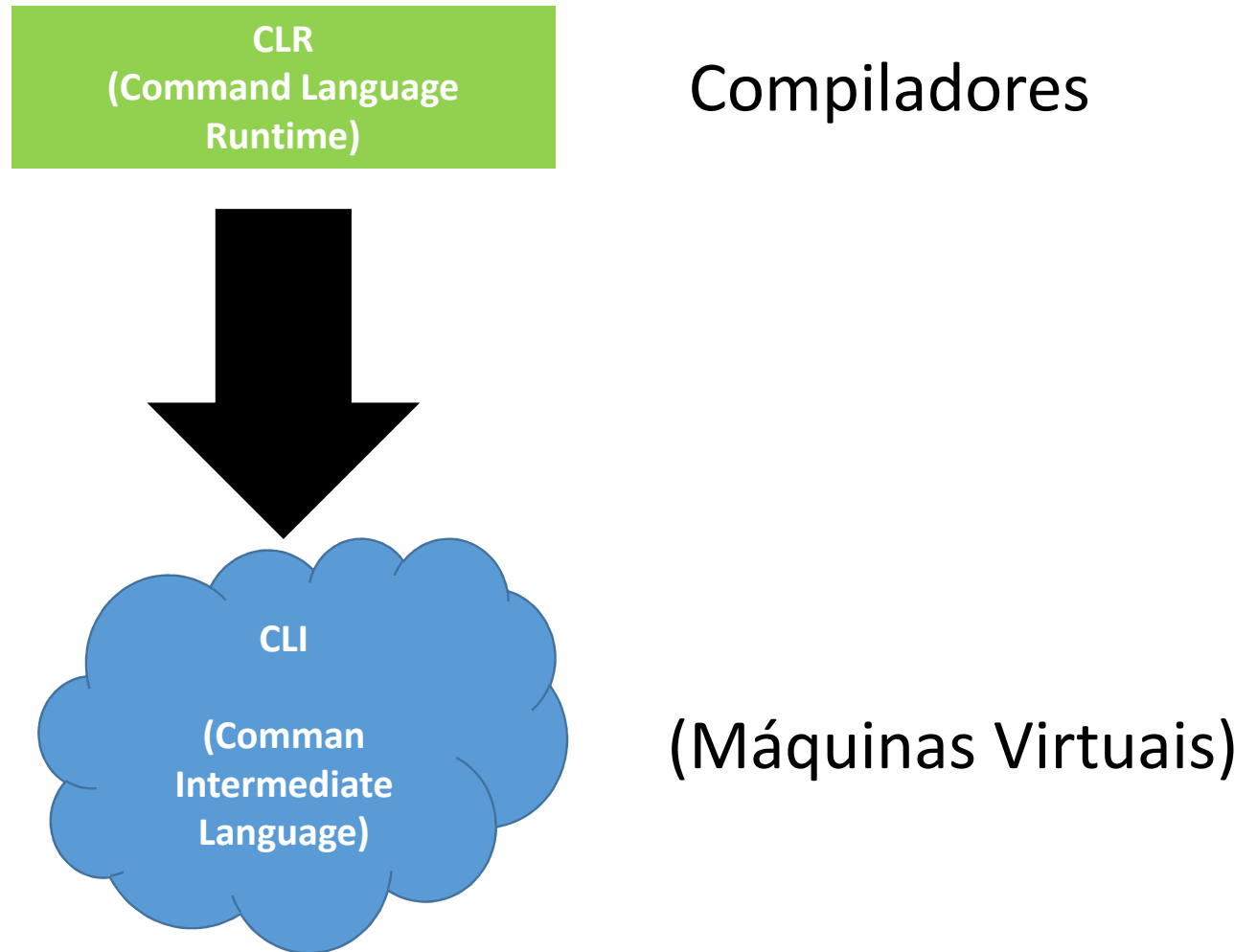
Característica principais de uma Linguagem de Programação (C#)

- 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).

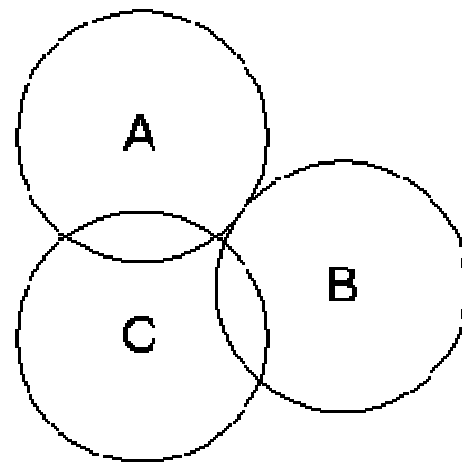
Compiladores e VM (Máquinas Virtuais)



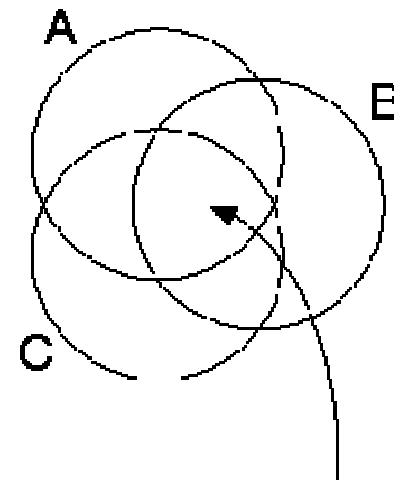
Conceito de Framework

Conceito de Framework.

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



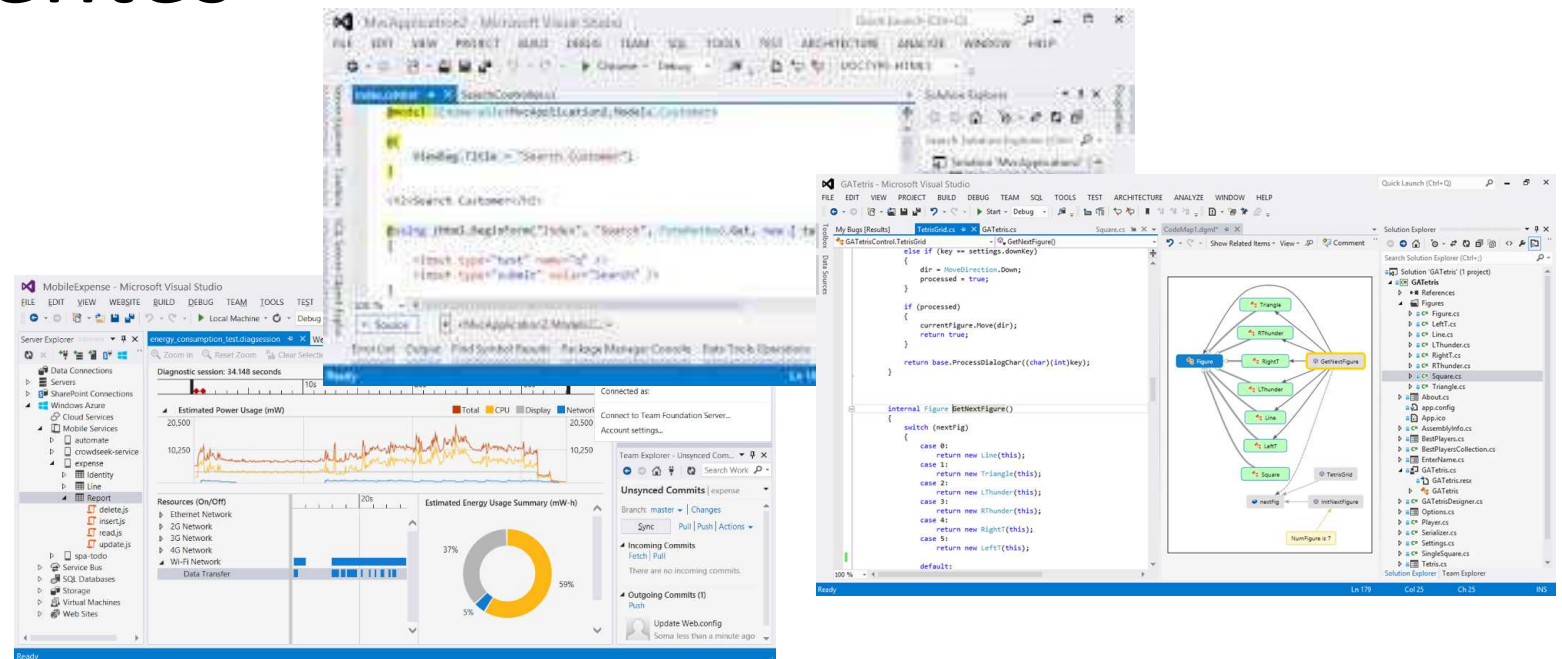
Impossível criar
Framework



Interseção grande
Possível criar Framework

- Aplicações Orientadas a Objetos;
- Componentes

- ## •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.

Memória RAM

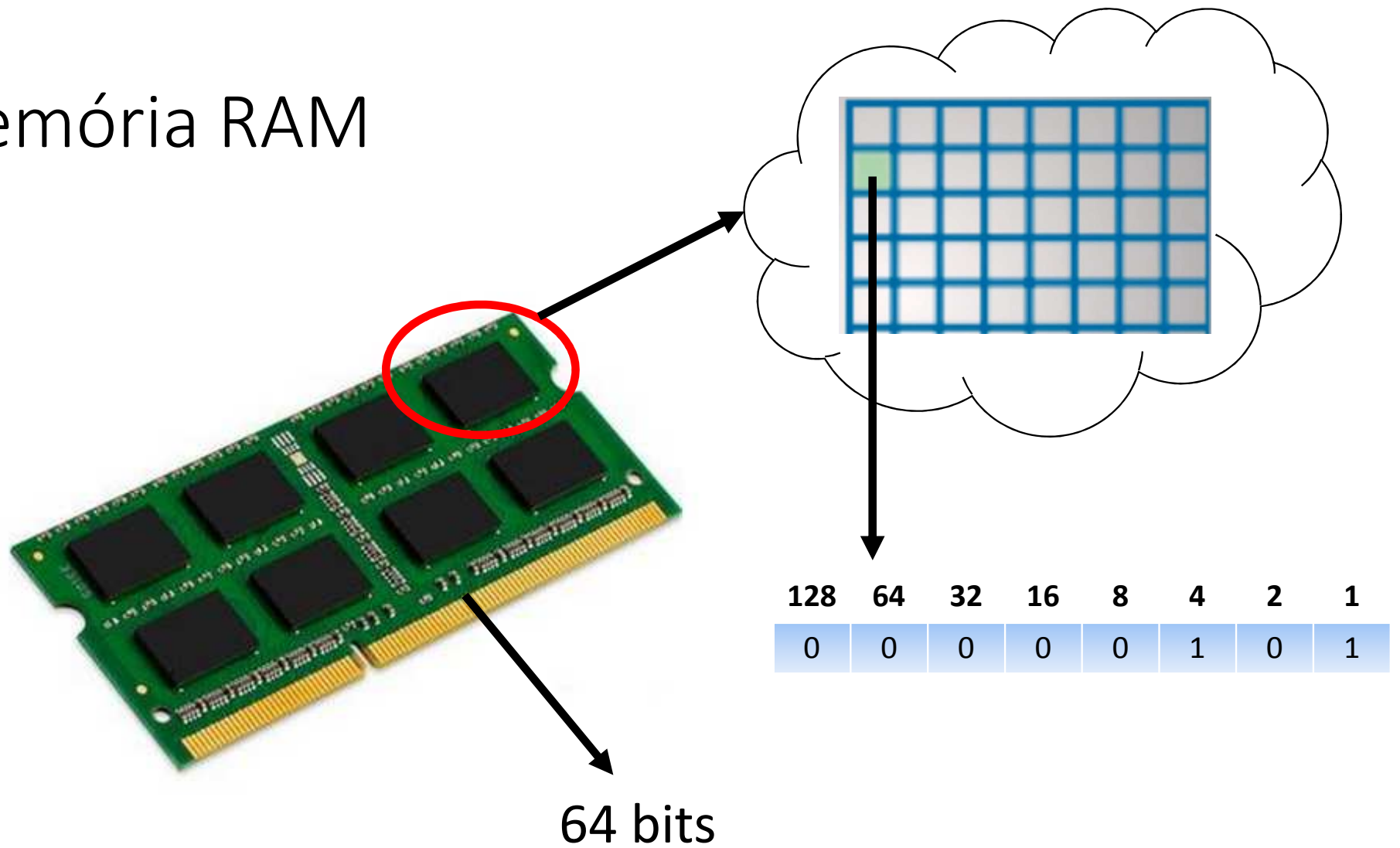
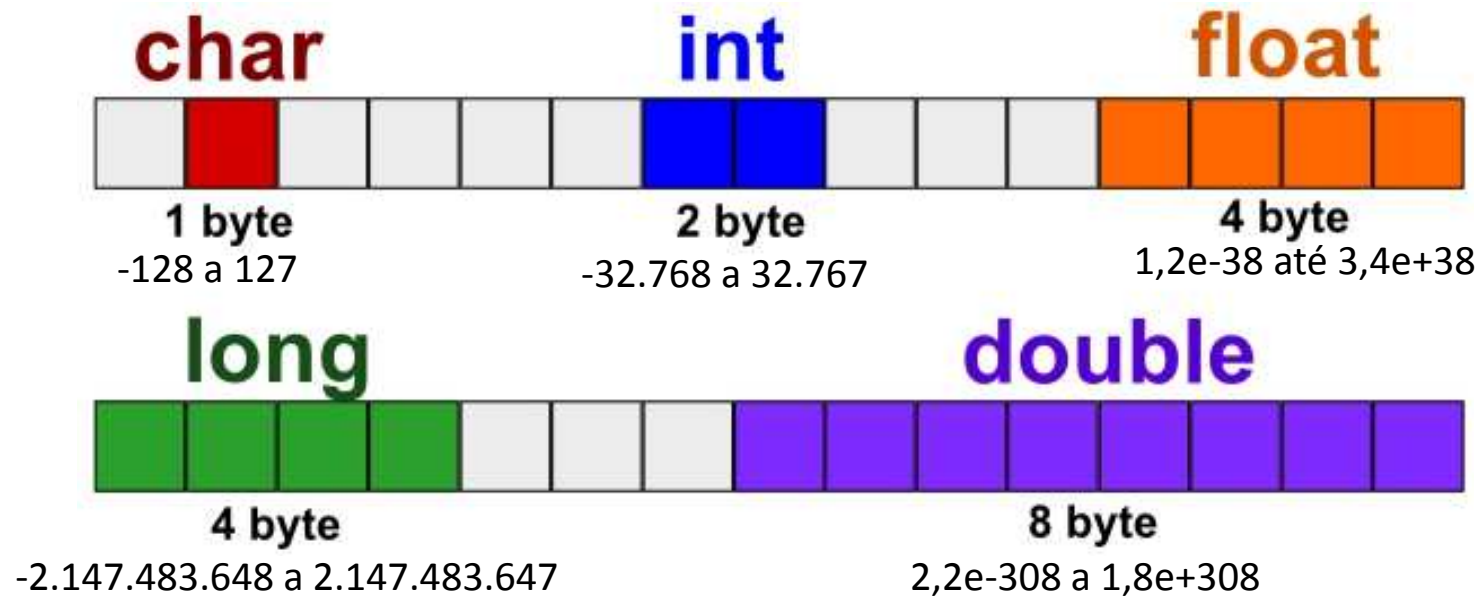


Tabela de tipos de dados



char

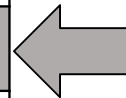


Vetor de caracteres.

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
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	M	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

|| - ou 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;
```

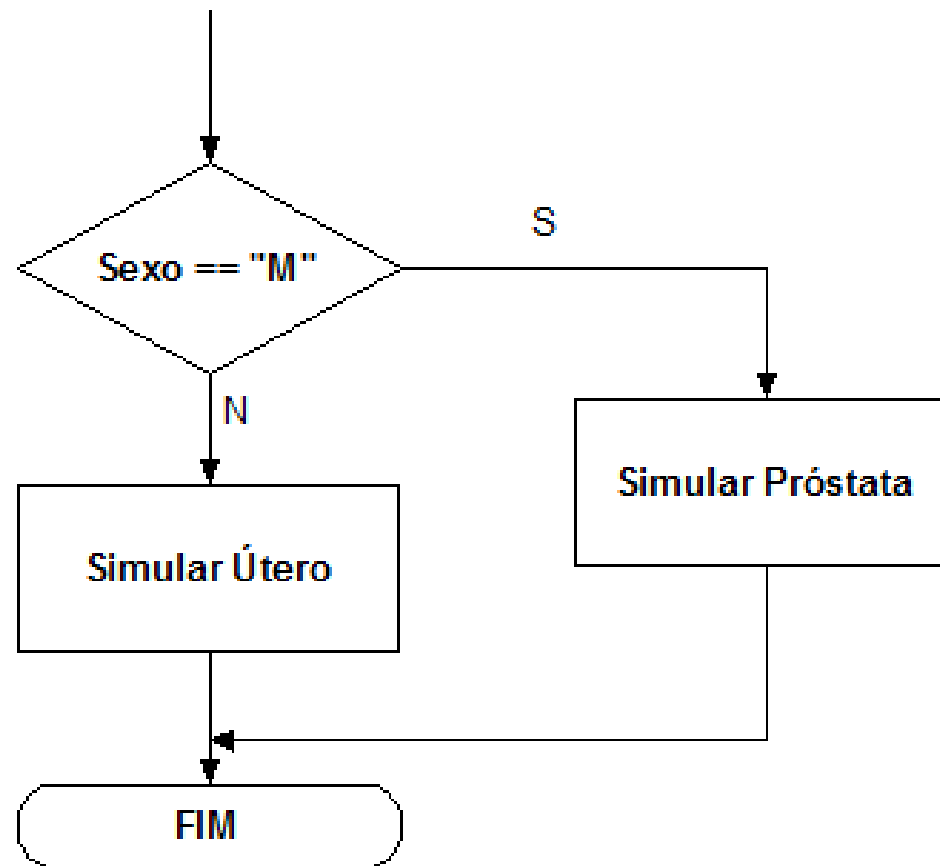
```
int k = y++; // Resultado y = 5 e k = 4
```

Linguagem de Programação C#

Estrutura Básica

- Atribuição
- Condição
- Repetição

Decisão Composta



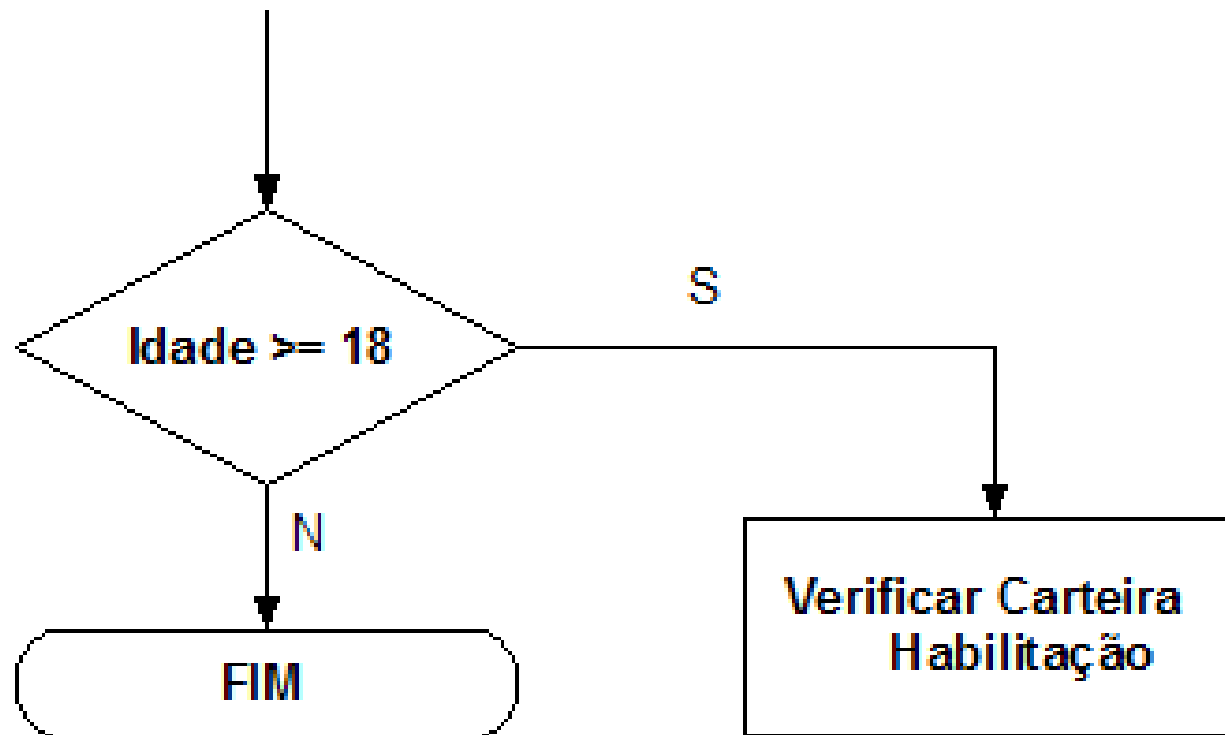
Condicional if()

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

Condiciona! if()

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

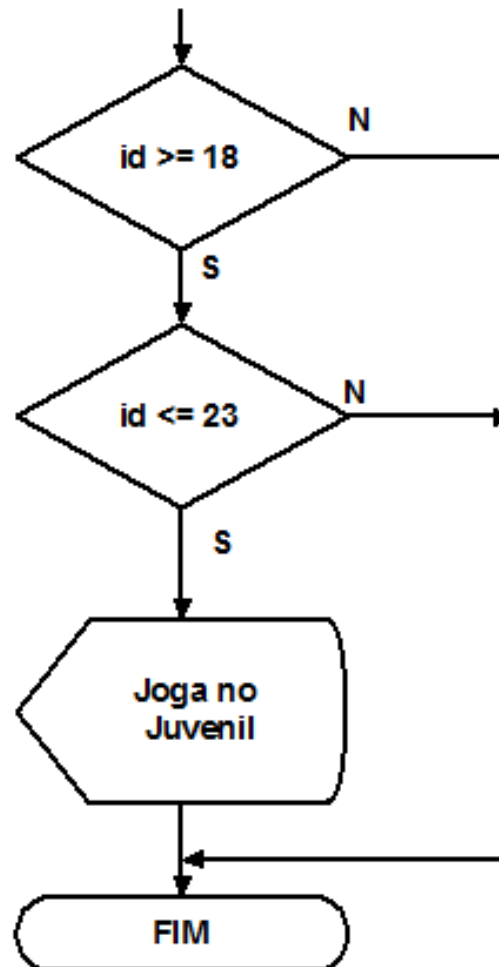
Decisão Simples



Exemplo de if()

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

Decisão Dupla duas condições



Séries de Instruções Condicionais

switch(): cadeia de condição

```
switch(<expressão>) {  
    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()

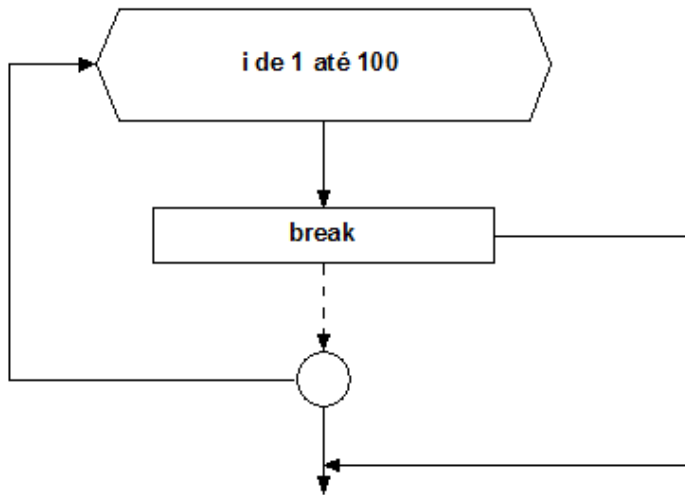
```
for(<início>;<condição>;<atualização>)  
{  
    [bloco de instruções]  
}
```

Exemplo do for()

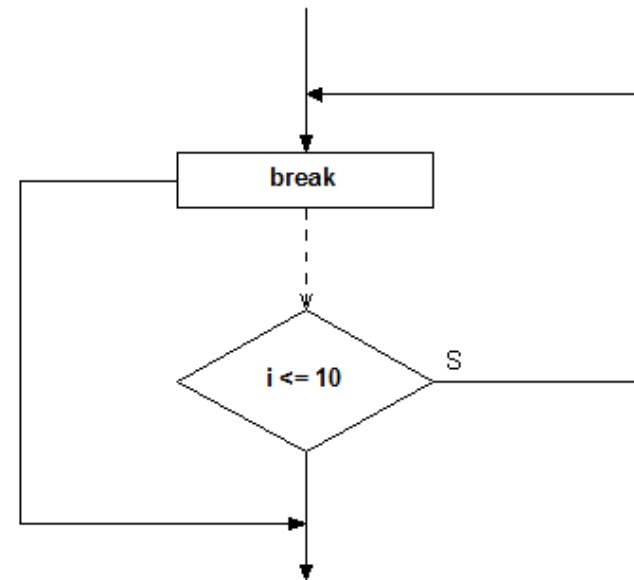
```
int num = 8;  
int contador = 0;  
for (int i = 0; i <= 10; i++)  
{  
    num = num - 1;  
    contador = contador + 1;  
}
```

Quebra e permanência de ciclo em estruturas de Controle.

Quebra de ciclo (break)



Estrutura for (para)

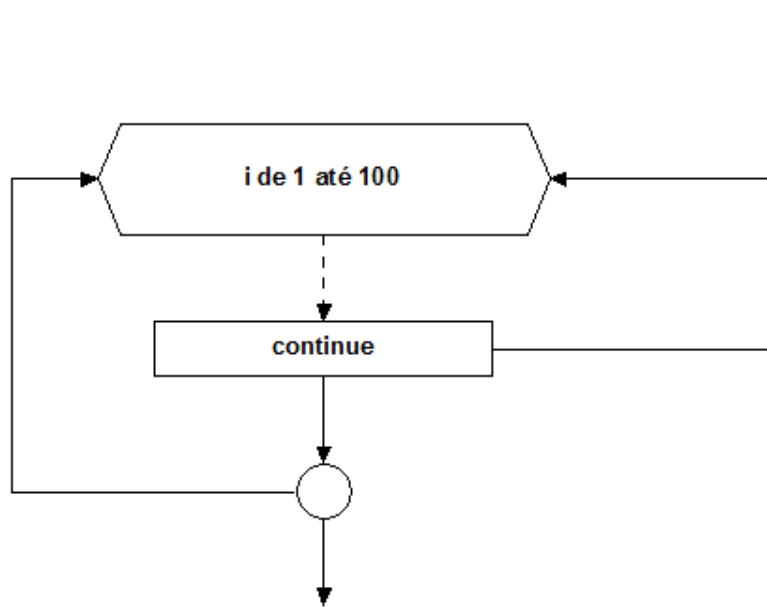


Estrutura while (enquanto)

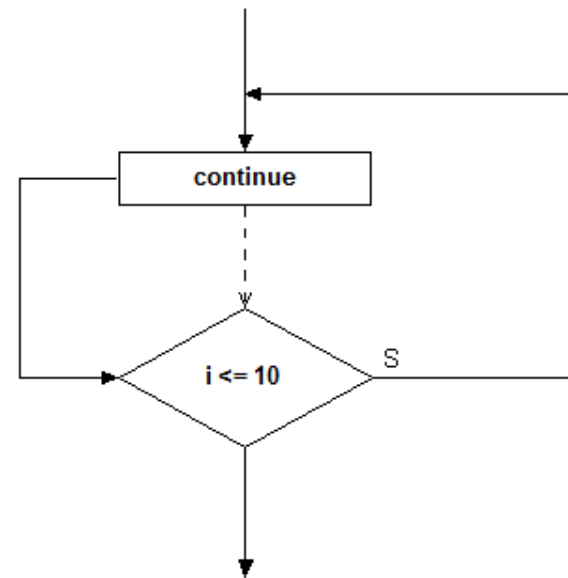
Exemplo Prático (break)

```
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)



Estrutura while (enquanto)

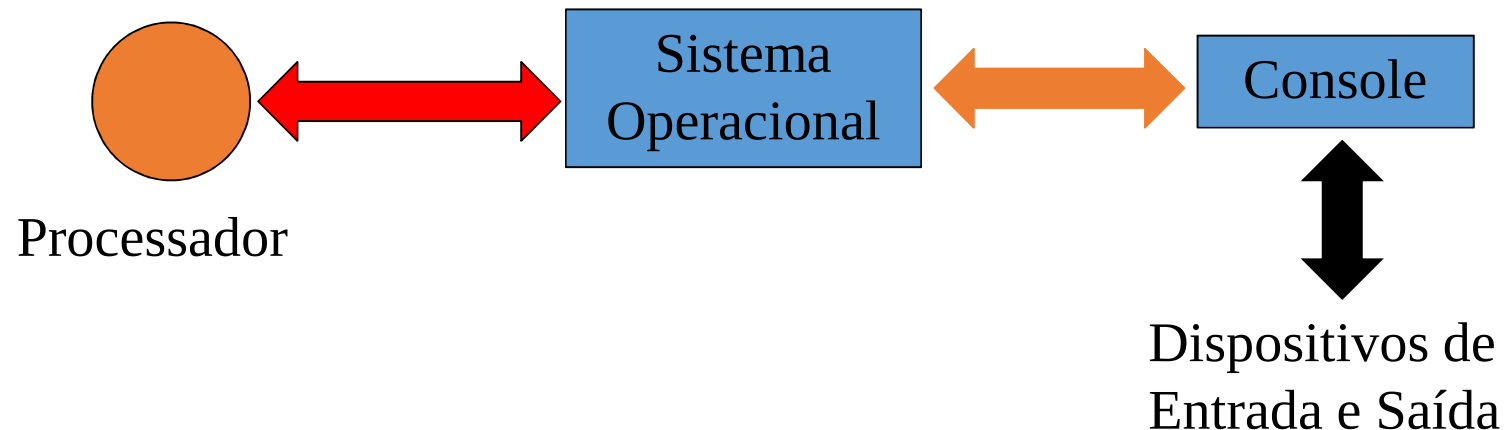
Exemplo Prático (continue)

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

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);
```