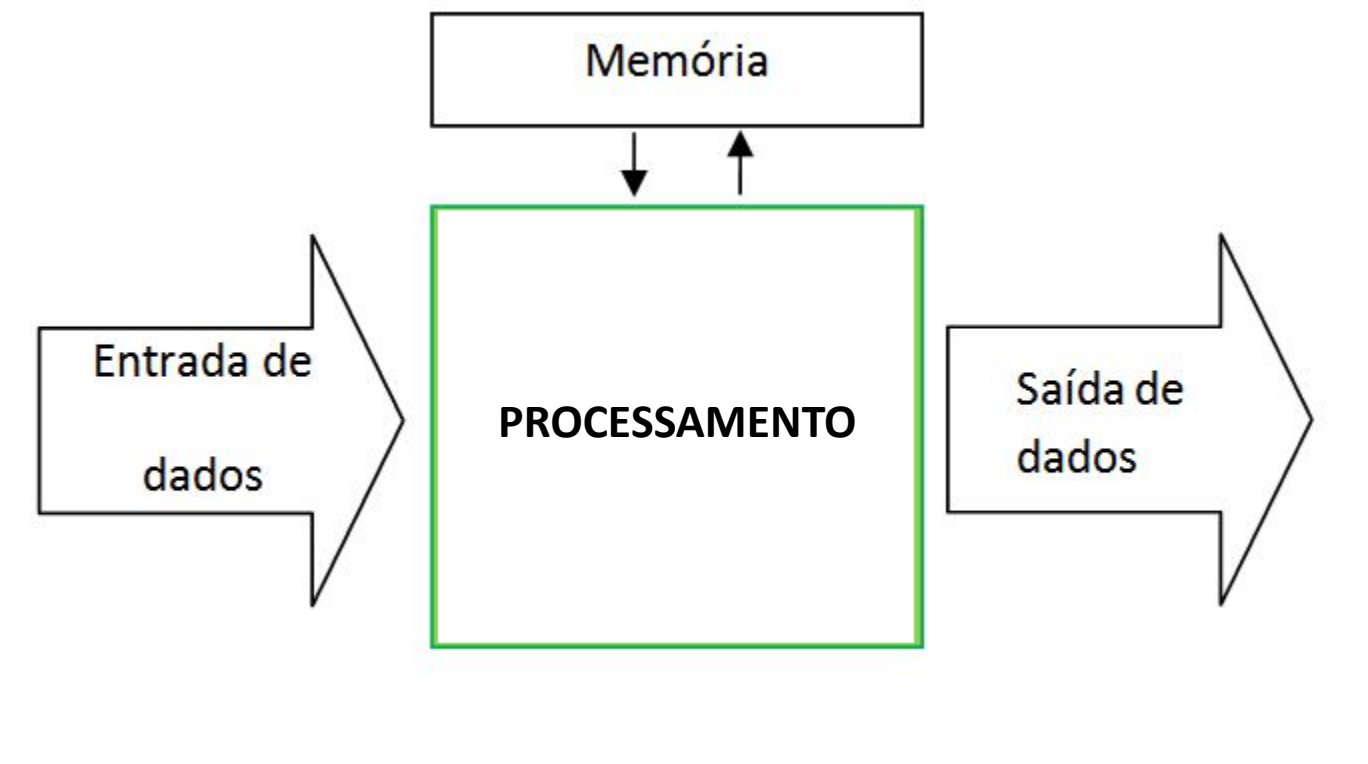
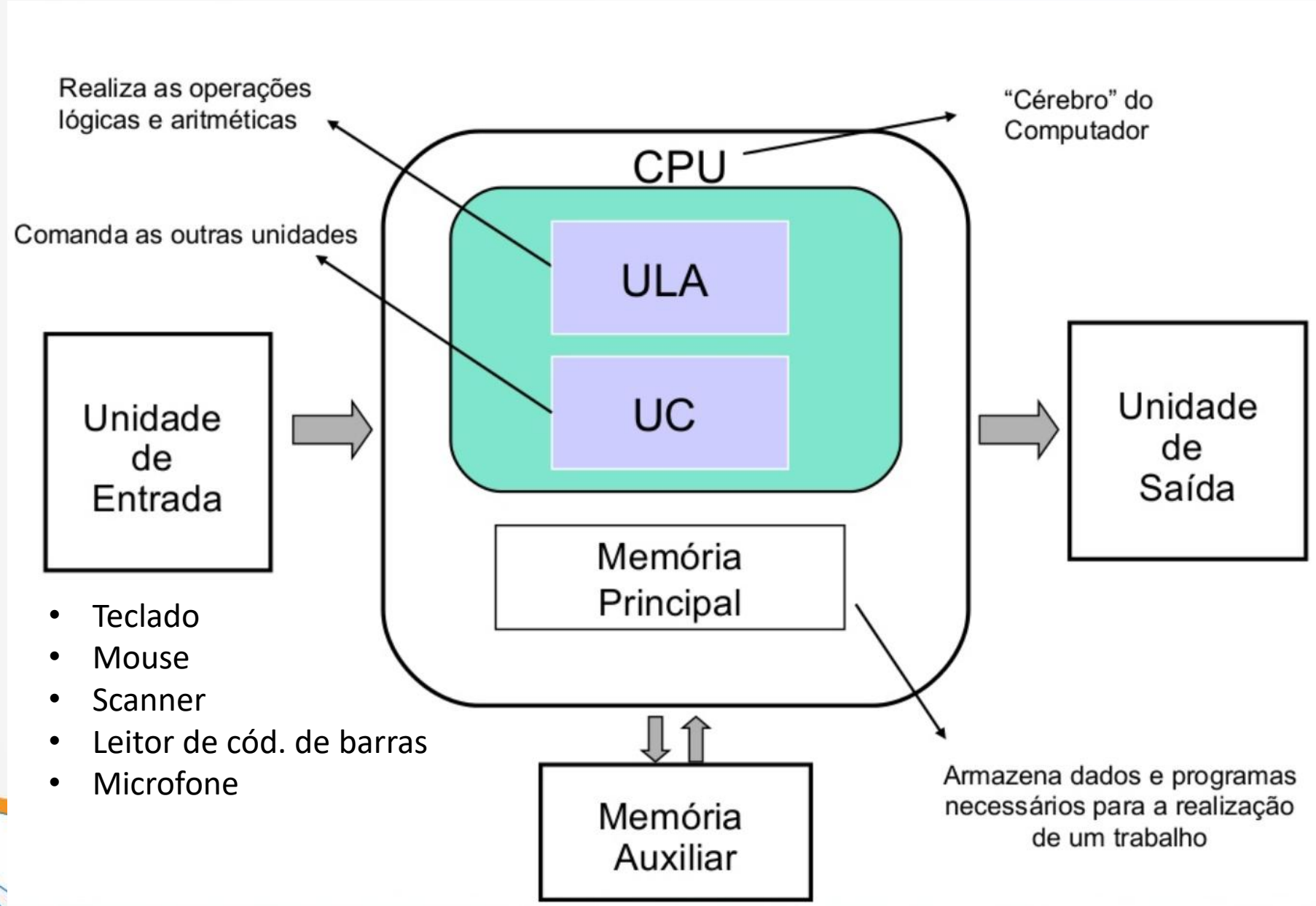
Programação em C#

Funcionamento e componentes do computador



Funcionamento e componentes do computador



- Teclado
- Mouse
- Scanner
- Leitor de cód. de barras
- Microfone

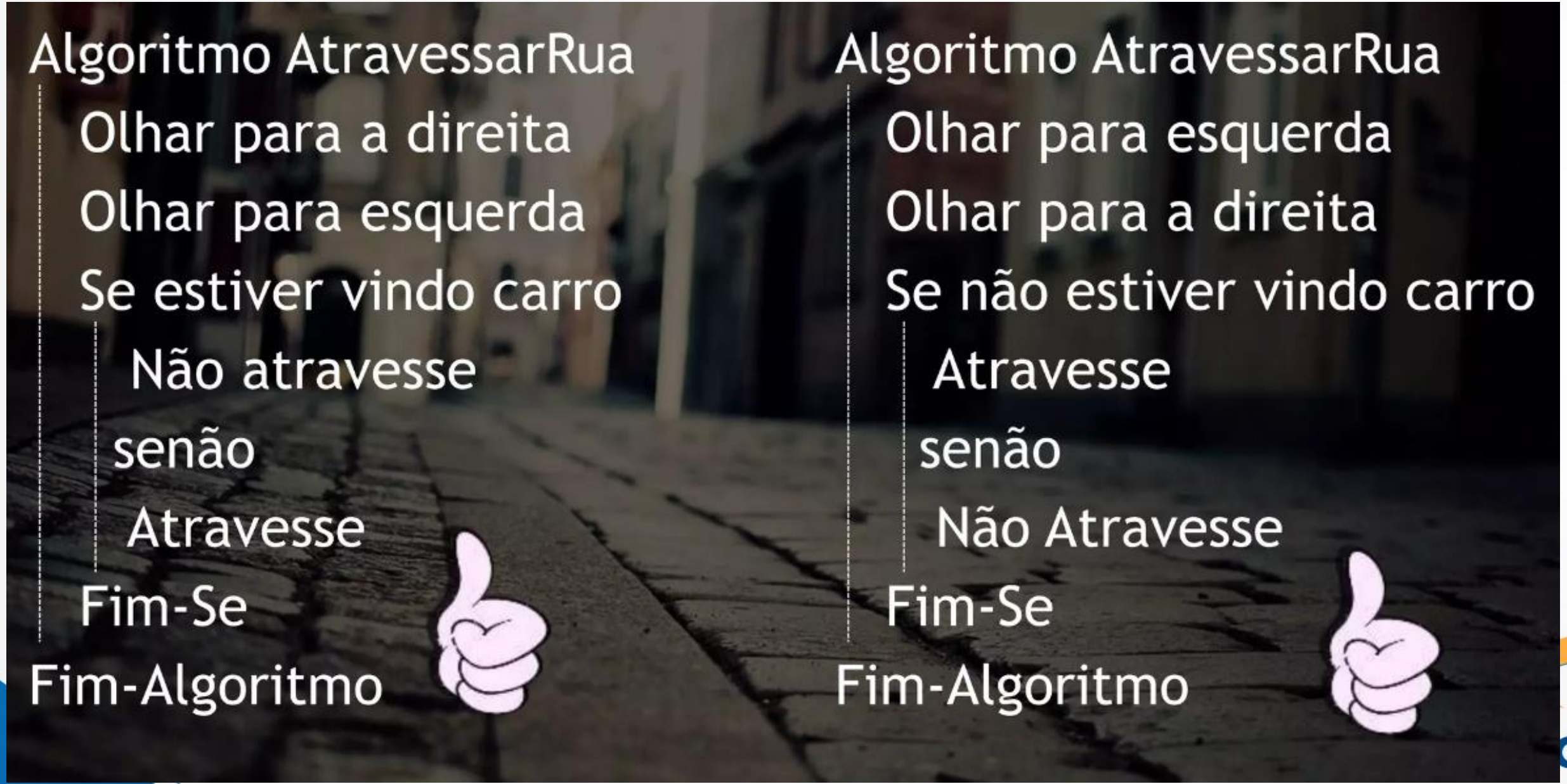
- Monitor
- Impressora
- Auto falante
- Drives

Algoritmos

- São conjuntos de passos finitos e organizados que, quando executados, resolvem um problema.
- É genérico e pode ser implementada em qualquer linguagem.
- Pode ou não usar a estrutura de uma linguagem de programação.
- Algoritmo é o caminho para a solução de um problema, e em geral, os caminhos que levam a uma solução são muitos.

Não existe apenas um único algoritmo para cada problema.

Algoritmos


Algoritmo AtravessarRua
Olhar para a direita
Olhar para esquerda
Se estiver vindo carro
Não atravesse
senão
Atravesse
Fim-Se
Fim-Algoritmo
Algoritmo AtravessarRua
Olhar para esquerda
Olhar para a direita
Se não estiver vindo carro
Atravesse
senão
Não Atravesse
Fim-Se
Fim-Algoritmo

Componentes básicos de um algoritmo

Um algoritmo é composto por:

- Variáveis
- Comandos

- Variáveis são informações lidas ou geradas pelo programa

- Comandos básicos:
  - Atribuição
  - Comandos de Entrada
  - Comandos de Saída
  - Condicionais
  - Laços

Lembre-se:

Algoritmos não se aprendem:

- Copiando algoritmos
- Estudando algoritmos

Algoritmos só se aprendem:

- Construindo algoritmos
- Testando algoritmos


C#
Microsoft
.NET

Declaração de Variáveis

C# é uma linguagem de programação **fortemente Tipada**, ou seja, toda variável ao ser criada deve ter um tipo de dados bem definido.

A declaração de variáveis no C# começa pelo tipo de dados seguindo pelo nome de identificação da variável. Ex:

```
string nome;
decimal nota1, nota2;
```

Os principais tipos de dados são:

- string (caracter/literal)
- int (inteiro)
- Double/decimal (real/decimal)
- Bool (booleano/lógico)

Convenções de Nomenclatura em Programação

- **camelCase** ex: nomeDoProduto
- **PascalCase** ex: NomeDoProduto
- **snake_case** ex: nome_do_produto
- **kebab-case** ex: nome-do-produto

Em C# costuma-se utilizar:

- PascalCase para Classes, Métodos e Propriedades
- camelCase para variáveis locais.

Convenções de Nomenclatura em Programação

Algumas dicas:

- Clareza e Significado
- Evite Abreviações
- Evite Nomes Genéricos
- Não use espaços
- Não use caracteres especiais

OBS: C# é uma linguagem Case Sensitive, ou seja, faz diferenciação de letras maiúsculas e minúsculas.

Atribuição de Dados

No C# o comando de atribuição de dados é o “ = ”

```
string nome;
decimal nota1, nota2;

nome = “Fulano”;
nota1 = 10;
nota2 = 8;
```

Comando de Entrada

Em uma aplicação do tipo Console, podemos utilizar o comando Console.ReadLine(). Comando semelhante ao “Leia” do Português Estruturado

Ex:

```
string nome;
nome = Console.ReadLine()
```

Comando de Saída

Em uma aplicação do tipo Console podemos utilizar o Console.WriteLine
Comando semelhante ao “Escreva” do Português Estruturado
Console.WriteLine(“Escreve na página”)

Ex:

```
string nome;
nome = “Roni”;
Console.WriteLine(“O nome é: ” + nome);
```

OBS: Utilize o “+” para concatenar textos e variáveis

Operadores

Aritméticos:

+	Soma
-	Subtração
*****	Multiplicação
/	Divisão
%	Resto divisão

Relacionais:

==	igual
!=	diferente
>	maior
<	menor
>=	maior ou igual
<=	menor ou igual

Lógicos:

!	Não
&&	E
||	OU

Operadores

Tabela Verdade dos operadores lógicos:
e (&&), ou (||) e não (!).

Tabelas da verdade

Operação não

A	não A
F	= V
V	= F

Operação e

A	B	A e B
F	F	= F
F	V	= F
V	F	= F
V	V	= V

Operação ou

A	B	A ou B
F	F	= F
F	V	= V
V	F	= V
V	V	= V

Expressões Aritméticas

Prioridades: Na resolução das expressões aritméticas, as operações possuem uma hierarquia.

Prioridade	Operadores
1ª	parênteses mais internos
2ª	/ * %
3ª	+ -

Nota: no caso de operadores com igual prioridade, a expressão deve ser resolvida de esquerda para direita.

Expressões Lógicas

Prioridades

Entre operandos lógicos:

Prioridade	Operadores
1ª	! (não)
2ª	&& (e)
3ª	|| (ou)

Expressões – Precedência Geral

Prioridades

Entre todos os operadores:

Prioridade	Operadores
1ª	parênteses mais internos
2ª	operadores aritméticos
3ª	operadores relacionais
4ª	operadores lógicos

Bloco de Código (instruções)

- Um bloco de código é um conjunto de uma ou mais linhas de código definidas por um símbolo de chave de abertura e fechamento { }. Ele representa uma unidade de código completa com uma única finalidade em seu sistema de software. EX:

```
public class Xyz
{
    Instrução 1;
    Instrução 2;
    Instrução 3;
}
```

Estrutura de Decisão Simples (if)

```
if (condicao)
{
        “Ações Aqui Se Condição Verdadeiro”
}
```

Estrutura de Decisão Composta (if... else)

```
if (condicao)
{
        “Ações Aqui Se Condição Verdadeiro”
}
else
{
        “Ações Aqui Se Condição Falsa”
}
```

Estrutura de Decisão Encadeada (if... elseif)

```
if (condicao1)
{
        “Ações Aqui Se Condição1 Verdadeiro”
}
else if (condicao2)
{
        “Ações Aqui Se Condição2 Verdadeiro”
}
else
{
        “Ações Aqui Se Condição Falsa”
}
```

Estrutura de Múltipla Escolha (Select Case)

```
switch (variavel)
{
        case x:
                Ações de x;
                break;
        case y:
                 Ações de y;
                break;
        case z:
                 Ações de z;
                break;
        default:
                 caso contrario;
                break;
}
```

Estrutura de Repetição

Com teste no Início (while):

```
while (condicao1)
{
        "Ações enquanto a condição verdadeira"
}
```

Obs: Use:
continue Para pular as ações e ir para o próximo loop
break Sai do While imediatamente

Estrutura de Repetição

Com teste no Fim (do... while):

```
do
{
        “Ações enquanto a condição não é satisfeita”
}
while (condicao1)
```

Estrutura de Repetição

Com variável de controle (for):

```
for (int i = 0; i < 10; i++)
{
        “Ações a ser repetidas”
}
```

Obs: Nesse caso irá repetir 10 x