**Algoritmo, Automação e Programa de Computador**

**Algoritmo:** Sequência finita de instruções para se resolver um problema. Aplica-se a diversas áreas de conhecimento.

**Automação:** Consiste em utilizar máquina(s) para executar o procedimento desejado de forma automática ou semiautomática.

**Programa ~ Algoritmo:** Programas de computador **são algoritmos** executados pelo computador (em linhas gerais).

Conclusão**:** O computador é uma máquina que **automatiza** a execução de **algoritmos**.

Obs: Apenas algoritmos computacionais: Processamento de dados e Cálculos.

**O que é preciso para fazer um programa de computador?**

* Uma **linguagem de programação:** regras **léxicas** e **sintáticas** para se escrever o programa.
* Uma **IDE**: Software para editar e testar o programa.
* Um **compilador**: Software para transformar o **código fonte** em **código objeto**.
* Um **gerador de código** ou **máquina virtual**: software que permite que o programa seja executado.

**Linguagem de programação**

É um conjunto de regras **léxicas** (ortografia) e **sintáticas** (gramática) para se escrever programas.

**IDE – Ambiente Integrado de Desenvolvimento**

**IDE:** É um conjunto de softwares utilizado para construção de programas.

Exemplos:

C/C++: Code Blocks

Java: Eclipse, NetBeans

C#: Microsoft Visual Studio

**Funcionalidades de uma IDE**

* Edição de código fonte (endentação, autocompletar, destaque de palavras, etc.)
* Depuração e testes
* Construção do produto final (build)
* Sugestão de modelos (templates)
* Auxiliar em várias tarefas do seu projeto
* Etc.

**Código Fonte e Objeto, Compilação, Interpretação e Máquina Virtual**

**Código fonte:** É aquele escrito pelo programador em linguagem de programação.

**Compilação:** É um processo que transforma o código fonte no que chamamos de **Código Objeto**. Esse processo é feito por um software chamado **compilador**. Durante o processo de compilação, o compilador faz a análise léxica e a análise sintática. Se houver erros na parte léxica ou sintática o programador é obrigado a resolver.

**Código Objeto:** Após o código fonte ser compilado e transformado em código objeto, este código objeto passa por um gerador de código para que seja feito um processo chamado **construção** (build).

**Interpretação:** O interpretador lê o código fonte e realiza a análise léxica, análise sintática e geração de código sob demanda.

**Abordagem híbrida:** O código fonte passa por uma precompilação que faz a análise léxica e sintática, gerando um código objeto que recebe o nome de **Bytecode**. O bytecode é interpretado por uma **máquina virtual** que faz a geração de código sob demanda.

**Entrada de Dados em C#**

O comando **Console.WriteLine()** é utilizado para exibir dados na tela.

O comando **Console.ReadLine()** é utilizado para receber dados digitados. Todos os dados digitados são recebidos em string.

Para converter string para outros formatos é preciso usar o Parse.

Ex: double x = double.Parse(Console.ReadLine());

**Split**

O split é usado para separar dados de um vetor, por exemplo.

Ex: string[] vetor = Console.ReadLine().Split(' ');

**Operações Matemáticas**

O comando **Math.Sqrt(x)** é utilizado para calcular a raiz quadrada de um número.

O comando **Math.pow(x)** é utilizado para calcular a potenciação de um número.

O comando **Math.abs(x)** é utilizado para mostrar o valor absoluto de um número.

**Vetores**

Um vetor corresponde a uma coleção de dados de tamanho fixo, indexada, unidimensional e homogênea.

**Indexada:** os elementos são acessados por meio de índices.

**Unidimensional:** uma dimensão.

**Homogênea:** todos dados são do mesmo tipo.

Vetores são também chamados de **arranjos** unidimensionais. Um arranjo deve ser alocado previamente, antes de ser utilizado. Uma vez alocado, sua quantidade de elementos é fixa.

**Declaração de Vetor**

(Int, Double ou String)[ ] NomeDoVetor;

**Instanciação**

NomeDoVetor = new (Int, Double, String)[ 2 ];

**Acessar os elementos de um vetor** **com For**

For ( int i = 0; i < 5; i++) {

NomeDoVetor[ i ] = i + 10;

}