# RESUMO

* 1. **– CRITÉRIOS DE AVALIAÇÃO DE LINGUAGENS**

Ainda na pauta sobre o avanço das tecnologias, precisamos entender que há muitas pessoas, em muitos lugares, desenvolvendo coisas novas todos os dias. Então cabe a nós, escolhermos o que melhor nos supre em questão de linguagens de programação. Os critérios de avaliação são úteis quando queremos comparar linguagens, afim de obter o que há de melhor quando precisamos desenvolver um projeto profissionalmente. Critérios como legibilidade, facilidade de escrita e confiabilidade são ditos como os mais importantes pelo simples fato de estarem mais presentes no plano de negócios, fatores que mais pesam na hora de escolher uma linguagem.

**Legibilidade**: Quando desenvolvemos um software, precisamos estar atentos ao fato de que haverá futuras manutenções e atualizações, portanto devemos prepara – lo desde o início do desenvolvimento para que haja facilidade em fazê – lo, e para que outras pessoas também entenda o código sem muitas complicações. A documentação também auxilia nesse processo.

**Facilidade de escrita:** Por mais complexa e cheia de recursos que a linguagem seja, uma boa facilidade de escrita é essencial para torna – lá uma linguagem aceitável. As tecnologias avançam depressa, não há tempo para aprender todas as “artimanhas” de uma determinada linguagem. Isso também reflete na legibilidade.

**Confiabilidade:** A network de computadores cresceu exponencialmente de uns anos pra cá, isso requereu que as linguagens acompanhassem esse crescimento e demandassem de mais tecnologias para proteger o dados que passam pelos seus softwares, assim tornando – os mais “parrudos”, à prova de falhas e interferências externas, como os hackers.

**Simplicidade e ortogonalidade:** Se uma linguagem tem um grande número de construções, alguns programadores não estarão familiarizados com todas. Essa situação pode levar ao uso incorreto de alguns recursos e a uma utilização escassa de outros que podem ser mais elegantes ou mais eficientes (ou ambos) do que os usados. Pode até mesmo ser possível, conforme destacado por Hoare (1973), usar recursos desconhecidos acidentalmente, com resultados inesperados. Logo, um número menor de construções primitivas e um conjunto de regras consistente para combiná-las (isso é, ortogonalidade) é muito melhor do que diversas construções primitivas. Um programador pode projetar uma solução para um problema complexo após aprender apenas um conjunto simples de construções primitivas. Por outro lado, muita ortogonalidade pode prejudicar a facilidade de escrita. Erros em programas podem passar despercebidos quando praticamente quaisquer combinações de primitivas são legais. Isso pode levar a certos absurdos no código que não podem ser descobertos pelo compilador