If688 - Teoria e Implementação de Linguagens de Programação

Alvaro Andrade

Outubro 2018

1 Introdução

A matéria de Teoria e Implementação de Linguagens de Programação ou "Compiladores"como é conhecida por ser equivalente as disciplinas de "if-120 - Compiladores"e "if-473 - Compiladores". Essa cadeira é oferecida no 7º período de ciência da computação pelo professor Leopoldo Motta Teixeira. Seu estudo explora os princípios da logica para computação e desenvolvimento de dados e seu comportamento nos compiladores. O objetivo da cadeira é fornecer os fundamentos para o desenvolvimento da compreensão de teoria de linguagem, análise léxica, representação e geração de códigos e o entendimento da teoria de compiladores.

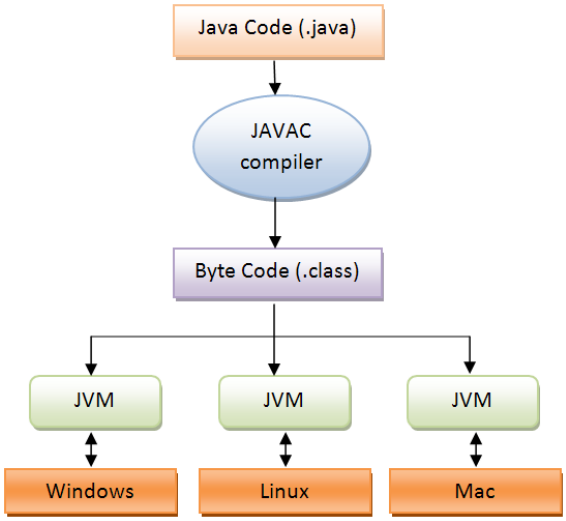

Figura 1: Funcionamento de um compilador

2 Relevância

O estudo de compiladores proporciona um entendimento de como são desenvolvidos as ferramentas de um softwares permitindo que o conhecimento do programador vá além da teoria, compreendendo a fundo o funcionamento de uma das ferramentas fundamentais para o funcionamento de um programa.

2.1 Ponto positivos

- Faz com que o aluno de ciência da computação tenha um entendimento completo sobre o funcionamento de um software.
- Aumenta a compreensão logica do estudante sobre linguagens diversas.

2.2 Pontos negativos

- Alguém que não tenha um bom conhecimento prévio de linguagens de programação tera dificuldades na disciplina.
- A cadeira não introduz a teoria básica de logica computacional.

3 Relação com outas disciplinas

If689 - Informática teórica	Por abordar temas como as primeiras maquinas a funcionar com programação por software, como a Máquina de Turing.
If669 - Introdução a programação	Essa matéria introduz o aluno a logica de programação e mexe com programas que utiliza compiladores para a execução.

Tabela 1: Matérias relacionadas

Referências

[1] Alfred V Aho and Monica S Lam. Ravi sethi, and jeffrey d. ullman. *Compilers, principles, techniques, and tools*, 1985.

[2] Keith Cooper and Linda Torczon. *Engineering a compiler*. Elsevier, 2011.

[3] Dick Grune and Ceriel JH Jacobs. A programmer-friendly ll (1) parser generator. *Software: Practice and Experience*, 18(1):29–38, 1988.

[4] Jens Palsberg. Type-based analysis and applications. In *Proceedings of the 2001 ACM SIGPLAN-SIGSOFT workshop on Program analysis for software tools and engineering*, pages 20–27. ACM, 2001.

[5] Michael Lee Scott. *Programming language pragmatics*. Morgan Kaufmann, 2000.
[2, 1, 4, 5, 3]