MC921 - Construção de Compiladores

Plano de Desenvolvimento da Disciplina

Professor: Sandro Rigo
Carga horaria: 90H
Creditos: 6
Horário:

- Aulas - 3a 08-10H, 8a 08-10H
- Laboratorios - 3a 10-12h, 5a 10-12h

Objetivo

O estudante deve desenvolver um compilador. Serão apresentados conceitos e técnicas da teoria de compiladores e projetos práticos serão desenvolvidos de forma associada.

Pré-requisitos

Ao início da disciplina, o estudante deve ser capaz de:

- Escrever algoritmos iterativos e recursivos para resolver problemas simples.
- Desenvolver e testar programas utilizando estruturas de dados (variável, tupla, lista, dicionário) e objetos (classe) na linguagem de programação Python.
- Escrever programa simples em linguagem de montagem e descrever as operações resultantes da execução deste programa na arquitetura do computador.

Programa da Disciplina

1. Visão geral da compilação
2. Analise léxica
3. Analise sintática
4. Analise semântica
5. Geração de código
6. Análise de fluxo de dados
7. Otimização de código
8. Compiladores no mundo real

Atividades

Aulas

As aulas teóricas serão presenciais e não serão gravadas. O docente disponibilizará os slides usados nas aulas.

Avaliações teóricas

Atividades teóricas serão aplicados através do Google Classroom. Essas atividades serão individuais e avaliadas.

Seminários

Assuntos de interesse da turma e tecnologias recentes em compiladores serão apresentados durante o semestre pelos estudantes. Essa atividade tem como o objetivo promover discussões, incentivar o pensamento crítico e exercitar a apresentação de conteúdos técnicos. As apresentações serão realizadas em grupos. A presença na apresentação de outros grupos contará para sua nota de seminários.

Projetos práticos

Ao longo do semestre, os estudantes deverão desenvolver um compilador. O desenvolvimento será dividido em 5 projetos práticos. Os projetos são trabalhos em grupo (máximo de 2 alunos por grupo).

Para cada projeto serão fornecidos um notebook e um repositório template contendo uma descrição detalhada do projeto, diretrizes de programação, trechos de código, etc. Os projetos usarão o ambiente GitHub Classroom. Os estudantes precisam clonar os repositórios localmente para realizar o trabalho e enviá-los para teste antes do prazo final da avaliação. O sistema de *Action* do GitHub executará os testes. Todas as entradas de teste para os projetos são abertas e não há testes fechados. A saída correta para cada teste é aberta, e sua avaliação levará em consideração não apenas a correção da execução, mas também o desempenho de alguns projetos. O GitHub fechará automaticamente o sistema de envio após o prazo final de cada projeto e não haverá extensões. Portanto, recomendamos fortemente que o aluno envie seu trabalho mesmo que o teste esteja incompleto.

O atendimento dos projetos será realizado presencialmente durante os horários de laboratório (4h por semana) e virtualmente no mural do Google Classroom.

Avaliação

Serão propostos alguns trabalhos teóricos e seminário (T) e 5 laboratórios práticos (P). A média (M) será calculada da seguinte forma:

MP = 0.1 P1 + 0.2 P2 + 0.2 P3 + 0.2 P4 + 0.3 P5
MT = Média aritmética dos trabalhos teóricos e seminário

M = 0.7 MP + 0.3 MT, se MP e MT acima de 5, caso contrário, M = min(MP, MT)

Exame

- Caso no final da disciplina o estudante tenha média $2.5 \leq M < 5.0$, poderá fazer o exame (E).
- Neste caso, a média final será calculada como MF = (M+E)/2, com nota máxima igual a 5.0.
- O exame será composto de uma prova teórica (ET) e um projeto prático (EP). A nota do exame será calculada como E = 0.7 EP + 0.3 ET, se EP e ET acima de 5, caso contrário, M = min(EP, ET)
- Data de realização: 11/7/24 as 10h

Informações Importantes

- As datas referentes às entregas, tanto dos projetos quanto dos exercícios, estarão disponíveis na Google classroom do curso.
- A presença é obrigatória em todas as aulas. Frequência inferior a 75% causa reprovação.
- Casos de plágio entre os trabalhos ou de conteúdos externos serão tratados com rigor. Qualquer tentativa de fraude nos trabalhos ou projetos resultará em nota final de M = 0 (zero) para todos os envolvidos e o caso será encaminhado à coordenação do curso.

Cronograma

Os projetos terão um prazo para ser implementado variando entre 1 e 4 semanas em função da complexidade. Cada trabalho teórico terá pelo menos uma semana de prazo de antecedência, Seguem as datas previstas:

- Entrega do projeto 1 - Lexer: 12/3
- Entrega do projeto 2 - Parser: 2/4
- Entrega do trabalho 1 - Front-end: 18/4
- Entrega do projeto 3 - Semantic: 30/4
- Entrega do projeto 4 - Codegen: 11/6
- Entrega do trabalho 2 - Back-end: 13/6
- Entrega do projeto 5 - Dataflow: 20/6/
- Exame: 11/07

Os seminários serão organizados em junho. As datas serão combinadas previamente com os estudantes.

Bibliografia

O curso é baseado nos seguintes livros, ou edições mais novas dos mesmos.

- Andrew Appel. Modern Compiler Implementation in Java.
- Aho, Sethi and Ullman. Compilers: Principles, techniques, and tools.
- Keith Cooper and Linda Torczon. Engineering a Compiler.