Escola Politécnica de Pernambuco

Disciplina: Compiladores
Semestre 2026.2
Prof. Luis Carlos (lcsm@poli.br)
Classroom: xwkyqdfy

Website:
http://disciplinas.prof-luis.cloud-ip.cc/

Agenda

- Desenvolvimento de Linguagens
  - Model Driven Development
  - Domain Specific Languages
- Motivação
- Problema de Compiladores
- Conteudo da Disciplina
-

Motivação I

- Implementar o tipo pessoa com nome e endereço.

```
1
2  class Pessoa {
3      String nome;
4      String endereço;
5  }
```

- Eh só isso???
  - NAO!!!!!!

Motivação I

- Implementar o tipo pessoa com nome e endereço.

```
1
2  class Pessoa {
3     private String nome;
4     String nome() { return this.nome; }
5     void nome(String nome) { this.nome = nome; }
6     String endereço;
7     String endereco() { return this.endereco; }
8     void endereco(String end) { this.endereco = end; }
9
10    Pessoa(String n, String e) {
11         nome = n;
12         endereco = e;
13    }
14    public Strint toString() {
15        return "Pessoa("+nome+","+endereco+")";
16
17    }
18    ....
19 }
```

- Muita Repetição
- Como tornar o desenvolvimento mais simples...

Motivação II

- Escrever um programa que desenha uma linha na tela usando Java/Swing:

```
import javax.swing.*;
import java.awt.*;

public class DesenhoLinha extends JPanel {

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);

        // Desenha uma linha de (50, 50) até (250, 150)
        g.drawLine(50, 50, 250, 150);
    }

    public static void main(String[] args) {
        JFrame janela = new JFrame("Desenho de uma linha");

        janela.add(new DesenhoLinha());
        janela.setSize(320, 240);
        janela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        janela.setLocationRelativeTo(null);
        janela.setVisible(true);
    }
}
```

Motivação III:

- O que faz este código???

```
1  class Q {
2      Vector<Estudante> estudantes = new Vector<Estudante>()
3      Vector<String> consulta() {
4          Vector<String> result = new Vector<String>()
5          for (int c = 0; c<estudantes.size(); c++) {
6              Estudante corr = estudantes.get(c);
7              if (corr.idade > 20) result.add(corr.nome);
8          }
9          return result;
10     }
11     
12 }
```

Motivação III:

- O que faz este código???

```
1 SELECT nome FROM Estudantes WHERE idade > 20
```

Motivação IV

- Programa para “desenhar” uma função do 2º grau:
  -

main.c

```c
#include <stdio.h>

int main()
{
    float a,b,c;
    printf("Digite os coeficientes: ");
    scanf("%f %f %f", &a,&b, &c);
    for (float x=0; x<100;x=x+.01)
        printf("f(%f) = %f\n", x, a*x*x+b*x+c);

    return 0;
}
```

Motivação IV

- Programa para “desenhar” uma função do 2º grau.
- O programa só funciona para este tipo de função.
- E se o usuário precisar de outro tipo?
  - Funções trigonométricas
  - Exponenciais / logaritmicas
  - Mistas...
- O código precisaria ser refeito e compilado
- Eu gostaria de um programa mais “poderoso”:
  - Tratasse diferentes funções ser precisar ser alterado.

main.c

```c
#include <stdio.h>

int main()
{
    float a,b,c;
    printf("Digite os coeficientes: ");
    scanf("%f %f %f", &a,&b, &c);
    for (float x=0; x<100;x=x+.01)
        printf("f(%f) = %f\n", x, a*x*x+b*x+c);

    return 0;
}
```

Resumo

- Programar usando apenas “linguagens de programação gerais”:
  - Precisa de muita habilidade de programação
  - Dependente de tecnologia
  - Pouco flexível
  - Dificil de Entender
  - Muitos Elementos Repetição.
- Utilizar uma linguagem adequada facilitaria o desenvolvimento de aplicações.

Domain Specific Languages

- Linguagens de Propósito Gerais
- VS
-
  - –

Desenvolvimento Orientado a Modelos (MDD)

- O que é?
  - O Desenvolvimento Orientado a Modelos (MDD - Model-Driven Development) é uma abordagem que utiliza modelos (linguagens) como os principais artefatos no processo de desenvolvimento de software. Esses modelos são usados para especificar, desenhar, analisar e até mesmo gerar código automaticamente.

Desenvolvimento Orientado a Modelos (MDD)

- Vantagens do MDD:
  - Abstração Elevada:
  - Produtividade Aumentada:
  - Facilidade de Comunicação:
  - Reuso de Modelos:
  - Manutenção Simplificada:

PROBLEMA

- A maioria das linguagens de programação possuem suporte a programas que lidam com dados simples:

```
int main() {
    int idade;
    printf(“Qual a sua idade?”);
    scanf(“%d”, &idade);
    int nasc = 2026 - idade;
    printf(“você deve ter nascido por volta de %d”,nasc);
}
```

  - Dados mais complexos devem ser programados diretamente
    - Processadores de Linguagens

Processador de Linguagem

- Programa que recebe dados de entrada
  - Escritos usando uma linguagem complexa
- Armazena essa entrada na memória e realiza computações
- Produzem saida
  - Saida também complexas.

Tipos de Processadores

- Interpretadores
  - Lêem programas e imprimem o resultado da excecução destes
- Analisadores:
  - Lêem programas e calculam propriedades do programa.
- Embelezadores / Otimizadoes / Obfuscadores:
  - Lêem programas e produzem uma versão “melhorada” (ou piorada) dos mesmos
- Compiladores / tradutores:
  - Lêem programas em uma linguagem de origem e produzem programas equivalentes em uma linguagem de destino.
- Outros tipos.

Disciplina de Compiladores

- Estudo de técnicas para programa processadores de linguagens:
  - Reconhecimento
  - Modelagem e Representação na memória
  - Análise e transformação
  - Geração de Código.

Conhecimento Básico da Disciplina

- Teoria da Computação
  - Lingagens e Automatos
- Estruturas de Dados:
  - Listas, Árvores, Pilhas e Filas.
- Linguagens de Programação
  - Uma linguagem OO
  - Ferramentas de Desenvolvimento.

Revisem estes conteúdos...

Disciplina de Compiladores

- Estudo de técnicas para programa processadores de linguagens:
  - Reconhecimento
  - Modelagem e Representação na memória
  - Análise e transformação
  - Geração de Código.

Conhecimento Básico da Disciplina

- Teoria da Computação
  - Lingagens e Automatos
- Estruturas de Dados:
  - Listas, Árvores, Pilhas e Filas.
- Linguagens de Programação
  - Uma linguagem OO
  - Ferramentas de Desenvolvimento.

Revisem estes conteúdos...

Estrutura da Disciplina

- Formalismos e Ferramentas:
  - Expressões Regulares e Gramáticas
  - Fases de um Compilador Tradicional:
    - Front-End
    - Back-end
  - Ferramenta de Geração de Compiladores
  - Estudo de Caso (Expressões e Linguagem Imperativa)

Avaliação

- 1ª Avaliação
- 2ª Avaliação + Trabalho.
  - Projetar uma linguagem de domínio específico.

Bibliografia

- Bibliografia Básica:
  - Compiladores, Principios, técnicas e Ferramentas
    Aho, Sethi, Ullman
  - James Appel, Modern Compiler Construction in Java
  - Páginas do Wikipedia
- Ferramentas:
  - Ajudam a construção de processadores de linguagens.
  - ANTLR, LARK, Xtext, PEG.js, ...
-

Que ferramentas vamos utilizar???

- Ferramenta “simples” baseada em programação:
  - ANTLR
  - LARK
- Ferramenta baseada em frameworks complexos:
  - Xtext (eclipse / Java)
  - TextX (python)
- Ferramentas Baseadas em Formalismos de “Alto-nível”
  - Spoofax / Stratego.

O que é uma linguagem?

- Conceitos Básicos:
  - Alfabeto (Σ): Conjunto de Simbolos
  - Palavra: Sequencia de caracteres pertencentes a um alfabeto
  - Conjunto das Palavras (Σ*): Todas as sequencias de possivêis dos caracteres de um alfabeto.
  - Linguagem (L in Σ*): Subconjunto do conjunto de palavras que determina as palavras válidas da linguagem.
- Questões:
  - Como **determinar** o conjunto L (conjunto infinito)?
  - Como construir uma máquina que **decida** se uma palavra pertence a L

Teoria da Computação

- Determinar uma linguagem:
  - Expressões Regulares
  - Gramáticas Livres de Contexto
- Reconhecedor para Decidir uma Linguagem:
  - Autômatos Finitos
  - Outras Máquinas de Estados.
- Assuntos da Próxima aula...
  - Façam revisão de Teoria da Computação.