INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
CEARÁ
CAMPUS ARACATI

Bacharelado em Ciência da Computação

Disciplina: Compiladores
Aula 01 - Introdução ao processo de compilação

Professor: Diego Rocha Lima

Introdução

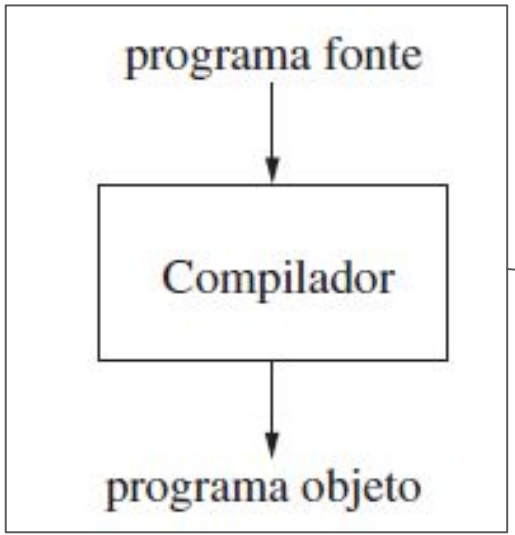
Linguagem de programação:

- Uma notação para descrever operações a serem executadas por uma máquina;

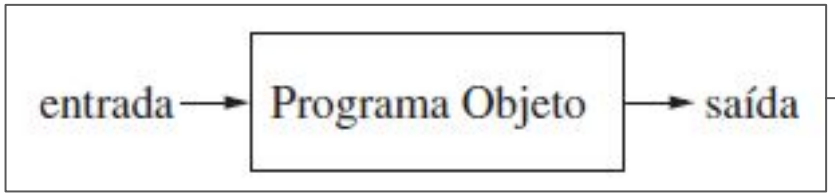
Compilador:

- Software que faz a tradução desses comandos para um formato que possa ser executado por um computador.

Etapas de processamento

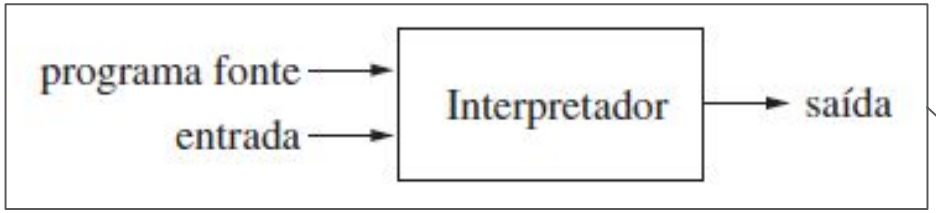


O compilador recebe um programa fonte (código fonte) como entrada e o transforma em um arquivo executável pelo sistema (programa objeto)

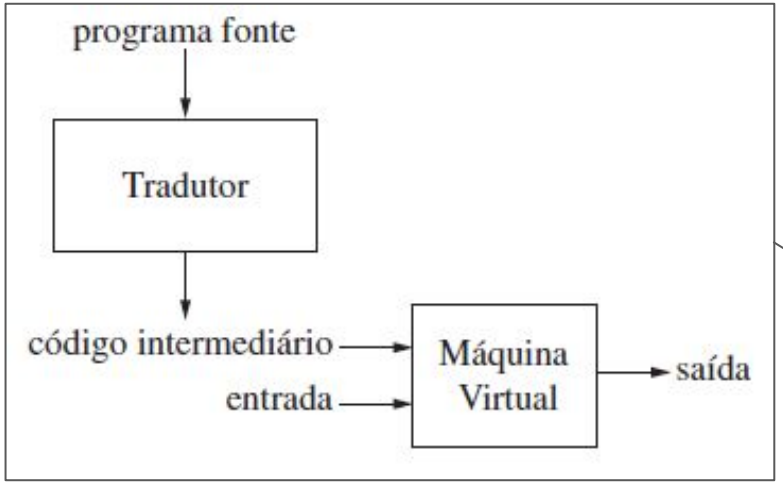


O programa objeto recebe a entrada de dados e executa, fornecendo a saída do programa

Etapas de processamento

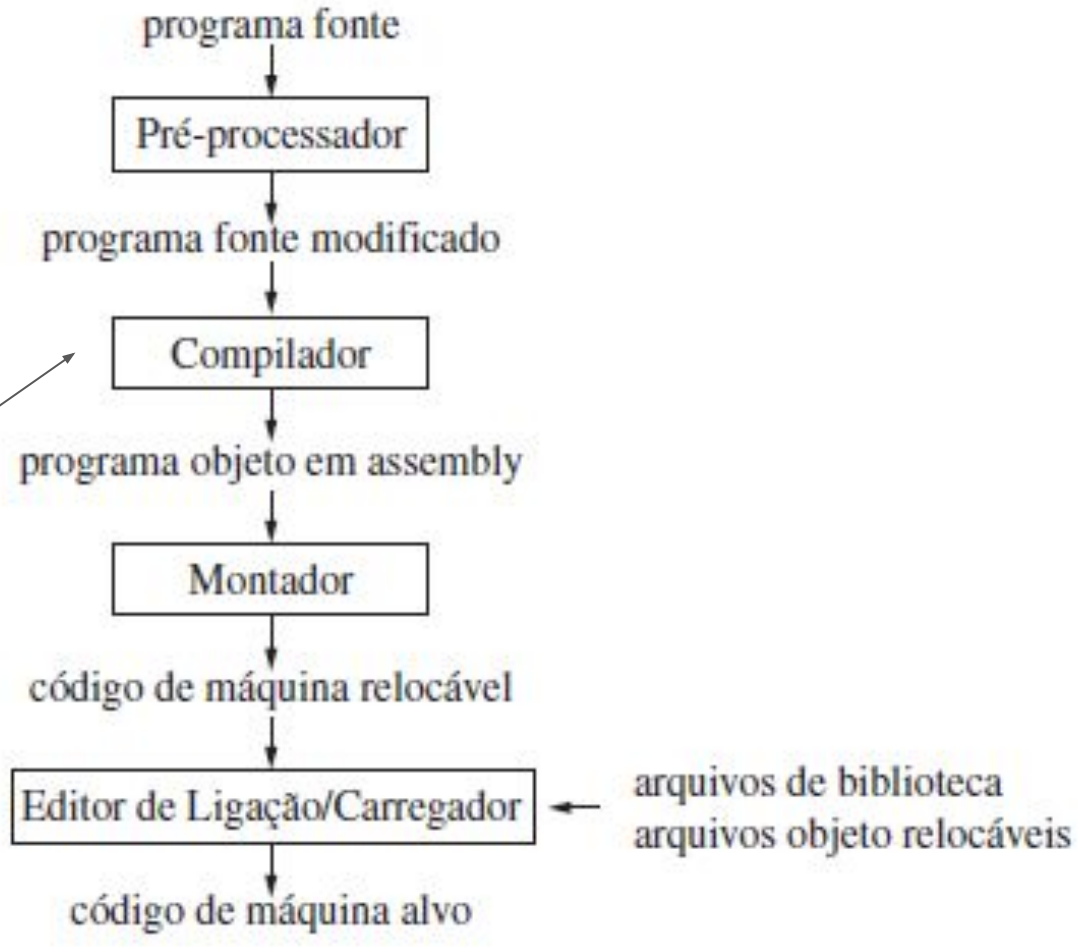


Um interpretador executa diretamente as operações especificadas no programa fonte sobre as entradas fornecidas pelo usuário.
Freqüentemente oferece um melhor diagnóstico
de erro do que um compilador, pois executa o programa fonte instrução por instrução.



Combinam compilação e interpretação

Um sistema de processamento de linguagem completo

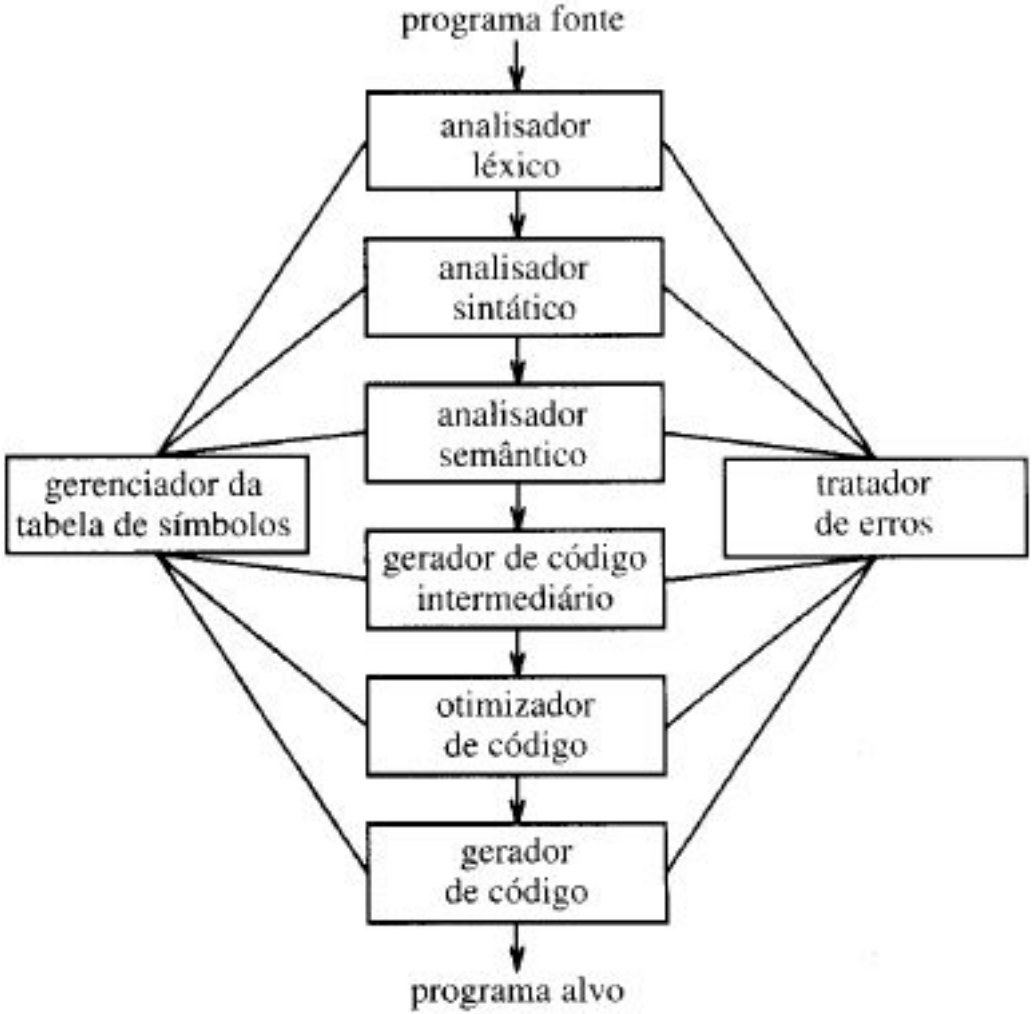


Estudaremos com detalhes esta etapa

Estrutura de um compilador

Existem duas partes nesse mapeamento:

- análise: léxica, sintática e semântica, onde são verificados erros no código fonte.
- síntese: etapas de criação do programa objeto



Etapas de Análise: Léxica

- responsável por ler todas as entradas, caractere a caractere, identificar os **lexemas,** transformá-los em **tokens** e guardar o seu valor específico.

Ex.:

```
montante = valor_inicial + tx_juros * 60
```

- **montante** é o token **<id,1>**, onde id será seu identificador e **1** é sua posição na tabela de símbolos;
- O símbolo **=** será p token **< = >**, sem valor associado, da mesma forma os tokens **< + >** e **< * >**
- **valor_inicial** será **<id,2>** e **tx_juros** será **<id,3>**
- **60** será o token **<60>**

Etapas de Análise: Léxica

Assim:

```
montante = valor_inicial + tx_juros * 60
```

Ficará da seguinte forma na tabela de símbolos:

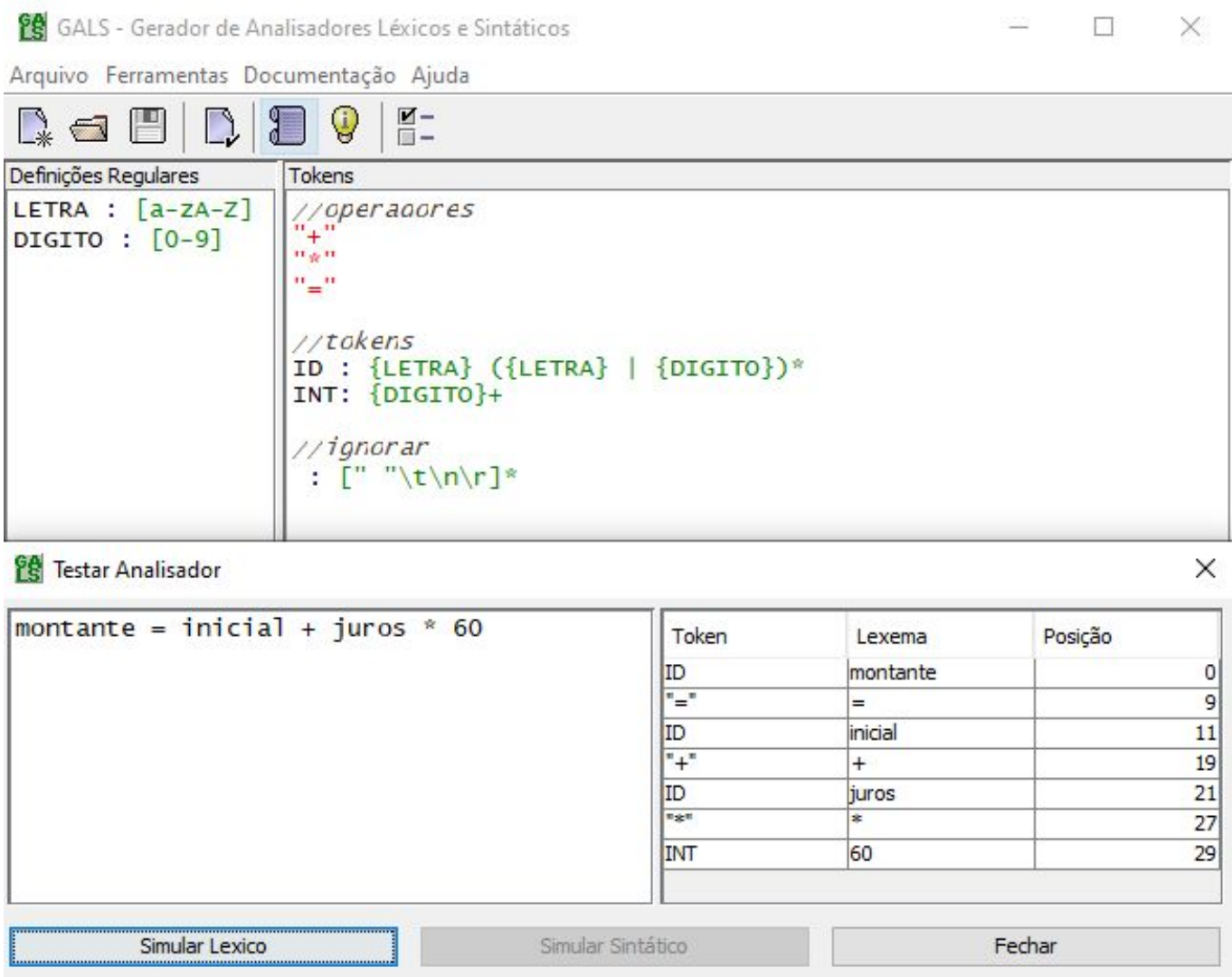
<id,1> <=> <id,2> <+> <id,3> <*> <60>

Obs.:

- A análise léxica também é responsável por descartar caracteres que são irrelevantes para a formação dos tokens, como espaços em branco, tabs e quebras de linha;
- Apenas os tokens são passador para a próxima etapa de compilação;

Exemplificando com o GALS

- Os lexemas são identificados através das **expressões gramaticais**;
- Cada lexema identificado deve estar associado a um token, ou o token será o próprio lexema



Etapas de Análise: Sintática

- Esta etapa utiliza **gramáticas** para encontrar representações para os tokens

Código fonte: **`montante = valor_inicial + tx_juros * 60`**

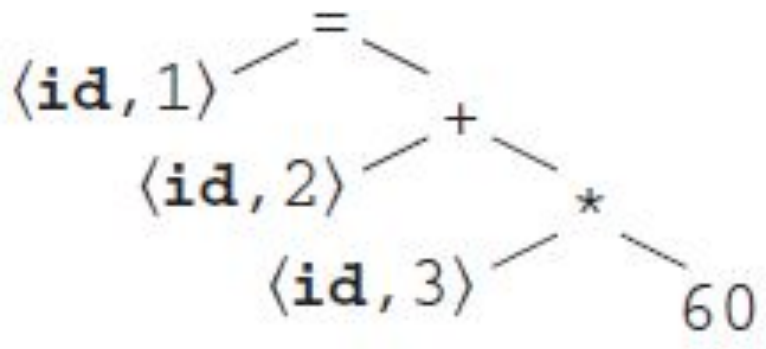
Análise léxica: **`<id,1> <=> <id,2> <+> <id,3> <*> <60>`**

Análise sintática: Faz a derivação com base em uma regra gramatical.

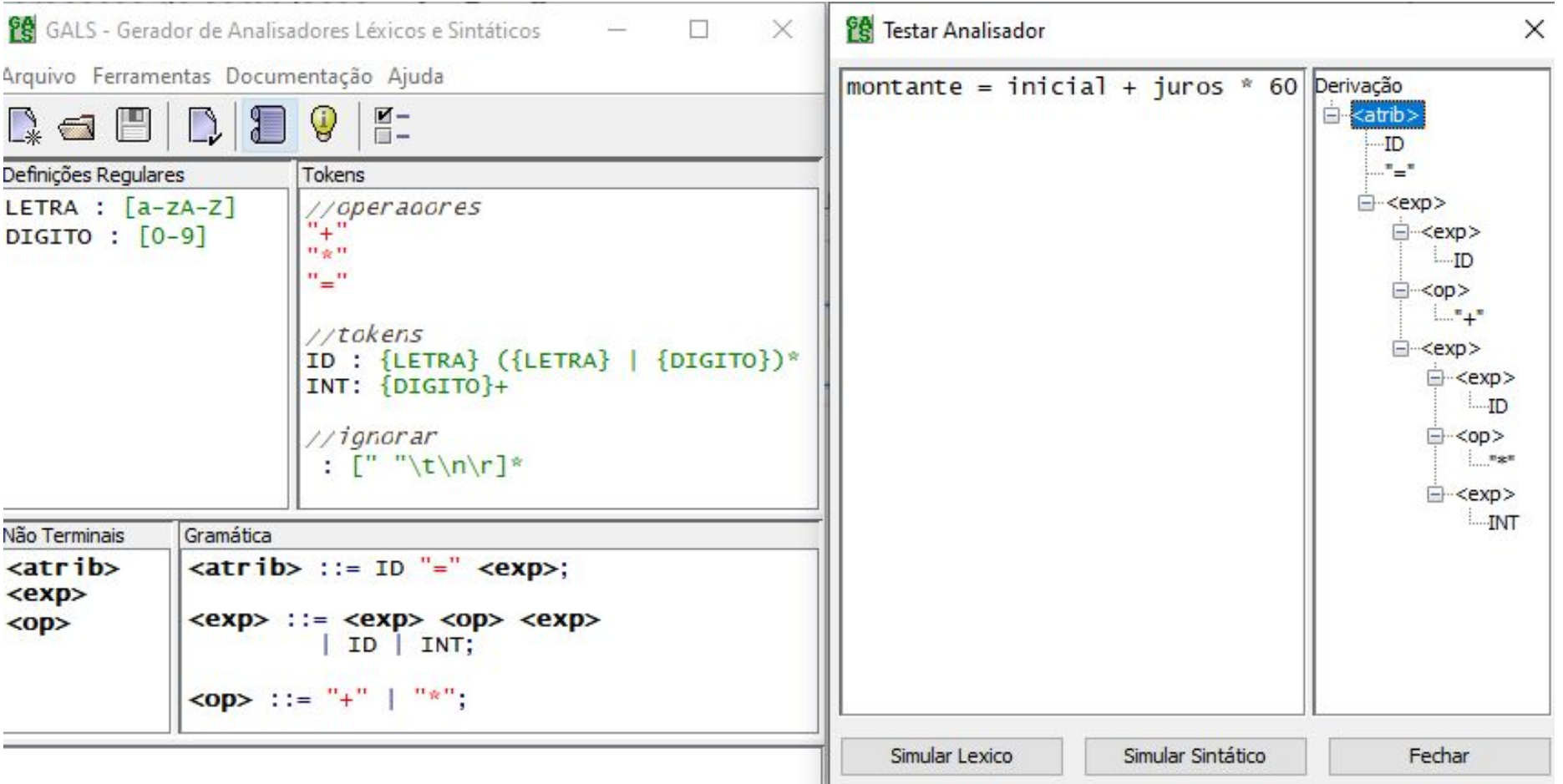
<atrib> → ID “=” <exp>

<exp> → <exp> <op> <exp>
| ID | INT

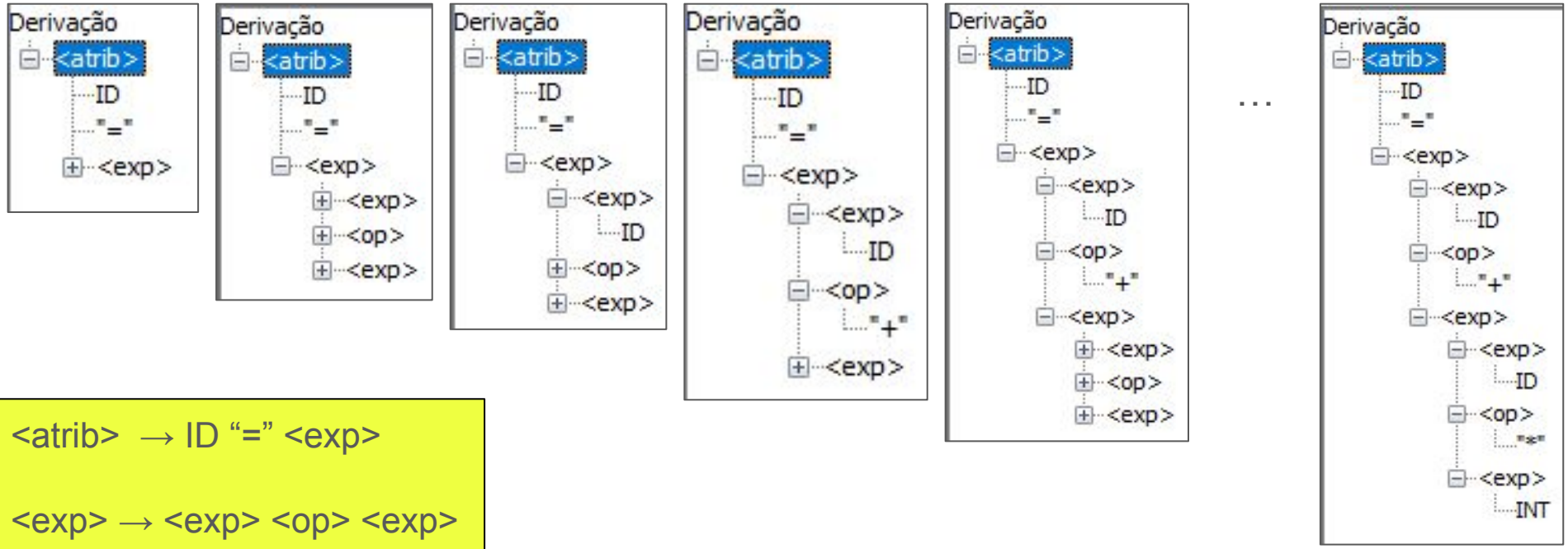
<op> → “+” | “*”



Exemplificando com o GALS



Passo a passo da derivação



<atrib> → ID “=” <exp>

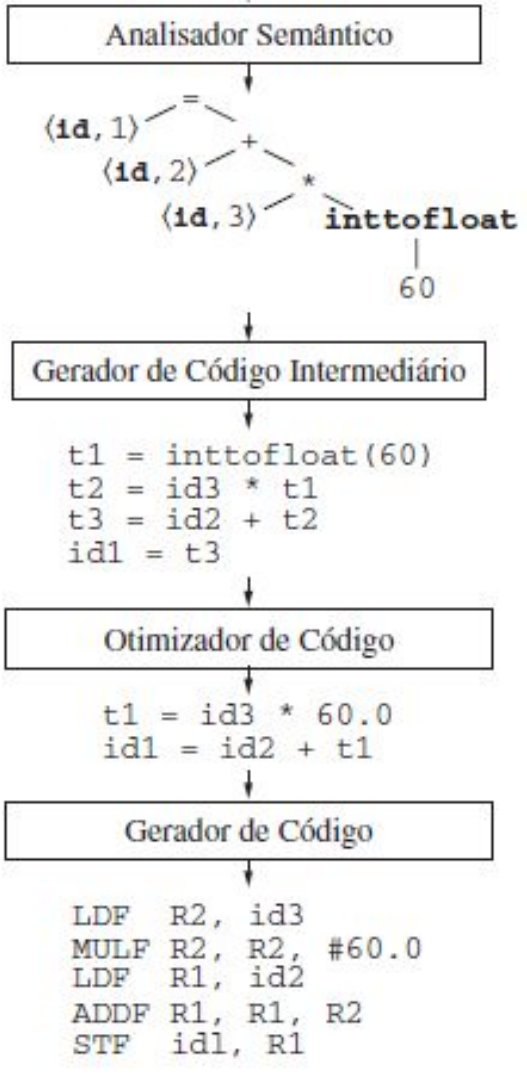
<exp> → <exp> <op> <exp>
| ID | INT

<op> → “+” | “*”

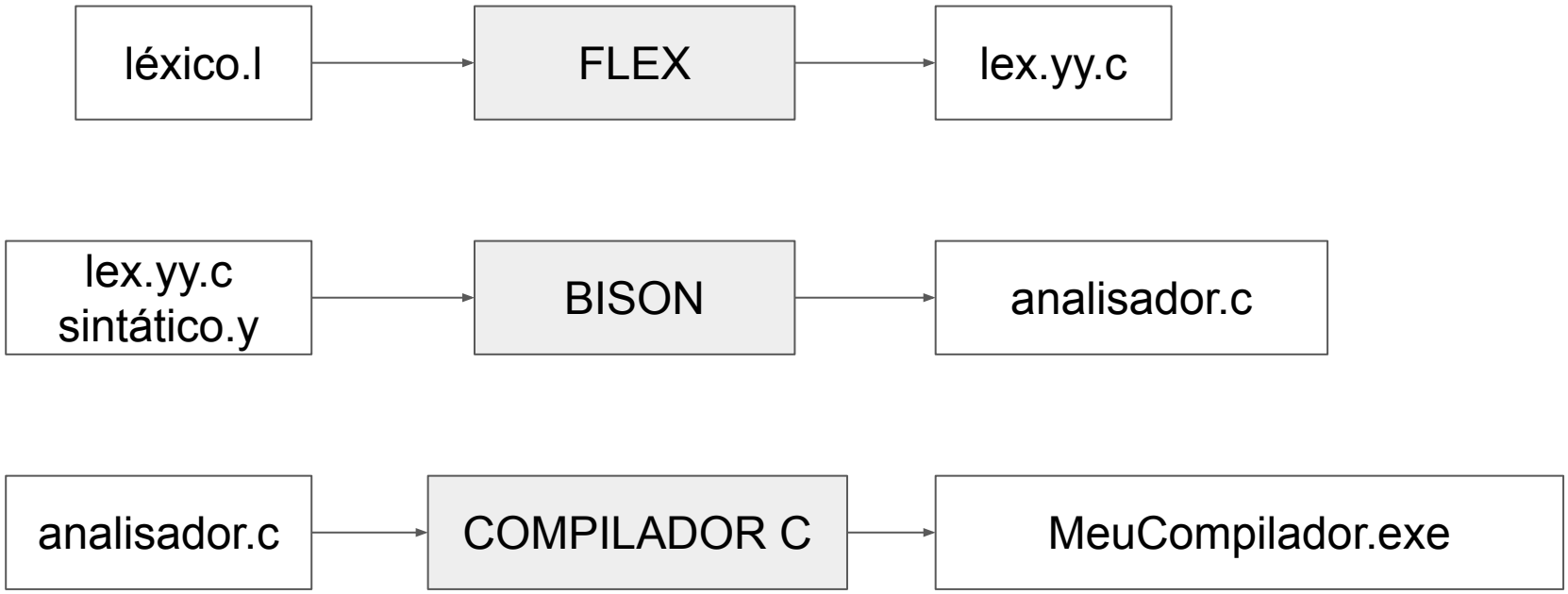
Etapas de Análise: Semântica

Exemplos:

- Erros de tipos de dados: `int x = "texto";`
- Variáveis não declaradas
- Variáveis não inicializadas
- Violações de escopo
- Argumentos inválidos
- Retornos incompatíveis
- Uso indevido de operadores
- Acesso a posições inválidas
- Violações de conversão
- Erros de ponteiros



Etapas utilizando FLEX/BISON



Evolução das linguagens de programação

- Linguagens de baixo nível

Linguagem	Ano	Descrição
Assembly	~1947-1950	Primeira linguagem simbólica usada para facilitar a programação em código de máquina.
Fortran	1957	Primeira linguagem de alto nível amplamente usada, voltada para cálculos científicos e engenharia.
Lisp	1958	Criada por John McCarthy, uma das primeiras linguagens funcionais, usada em IA.
COBOL	1959	Criada para aplicações comerciais e de negócios, focada em legibilidade.

Evolução das linguagens de programação

- Linguagens estruturadas e de propósito geral

Linguagem	Ano	Descrição
ALGOL	1960	Introduziu a estrutura de blocos e influenciou muitas linguagens futuras.
BASIC	1964	Criada por John Kemeny e Thomas Kurtz para ensino de programação.
Simula	1967	Primeira linguagem com conceitos de orientação a objetos.
C	1972	Criada por Dennis Ritchie nos laboratórios Bell, base para sistemas operacionais como Unix.
Pascal	1970	Criada por Niklaus Wirth para ensino de programação estruturada.
Prolog	1972	Baseada em lógica, usada para IA e sistemas especialistas.
SQL	1974	Criada para manipulação de bancos de dados relacionais.

Evolução das linguagens de programação

- Linguagens Orientadas a objeto de para programação comercial

Linguagem	Ano	Descrição
C++	1983	Criada por Bjarne Stroustrup como uma extensão do C, adicionando orientação a objetos.
Objective-C	1984	Criada por Brad Cox, usada nos sistemas Apple.
Perl	1987	Popular para manipulação de texto e administração de sistemas.
Haskell	1990	Linguagem puramente funcional.
Python	1991	Criada por Guido van Rossum, com foco em simplicidade e produtividade.
Visual Basic	1991	Criada pela Microsoft, facilitando a programação para interfaces gráficas.
Ruby	1995	Criada por Yukihiro Matsumoto, combinando simplicidade e expressividade.
Java	1995	Criada pela Sun Microsystems, com o lema "Escreva uma vez, rode em qualquer lugar".
JavaScript	1995	Criada por Brendan Eich na Netscape, tornou-se essencial para a web.
PHP	1995	Criada por Rasmus Lerdorf para desenvolvimento web dinâmico.

Evolução das linguagens de programação

- Linguagens para Web, Mobile e Escalabilidade

Linguagem	Ano	Descrição
C#	2000	Criada pela Microsoft como alternativa ao Java.
Scala	2003	Combina programação funcional e OO.
Go	2009	Criada pelo Google, otimizada para concorrência e eficiência.
Rust	2010	Criada pela Mozilla, focada em segurança de memória e concorrência.
Kotlin	2011	Criada pela JetBrains, tornou-se oficial para Android.
Swift	2014	Criada pela Apple para substituir Objective-C no iOS.

Evolução das linguagens de programação

- IA, Concorrência e Computação Distribuída

Linguagem	Ano	Descrição
Julia	2012	Linguagem de alto desempenho para computação científica.
TypeScript	2012	Superconjunto do JavaScript com tipagem estática.
Dart	2011	Criada pelo Google, usada no Flutter para apps móveis.
Carbon	2022	Proposta como sucessora do C++ para maior segurança e produtividade.

Conclusão

- Uma linguagem de programação de alto nível define uma abstração de programação: o programador escreve um algoritmo usando a linguagem, e o compilador deve traduzir esse programa para a linguagem objeto;
- Um compilador faz uso de várias etapas para transformar um código fonte em um programa objeto;
- O conhecimento de expressões e gramáticas regulares é essencial no estudo dos compiladores;
- Ler o capítulo 1 do livro, onde há mais detalhes sobre o estudos dos compiladores e das linguagens de programação.

Exercícios

Defina expressões regulares para reconhecer os tokens corretamente nas seguintes operações em uma linguagem de programação

A. operação de atribuição
B. operação de declaração
C. operações de desvio condicional (is…else)
D. testes lógicos
E. operações de repetição (for, while ou do-while)
F. definição de funções
G. comentários

Obs.: Ler o TCC sobre o GALS para conhecer as regras da sintaxe