Judson Santos Santiago

Fast **Lex**ical Analyzer

Compiladores

Introdução

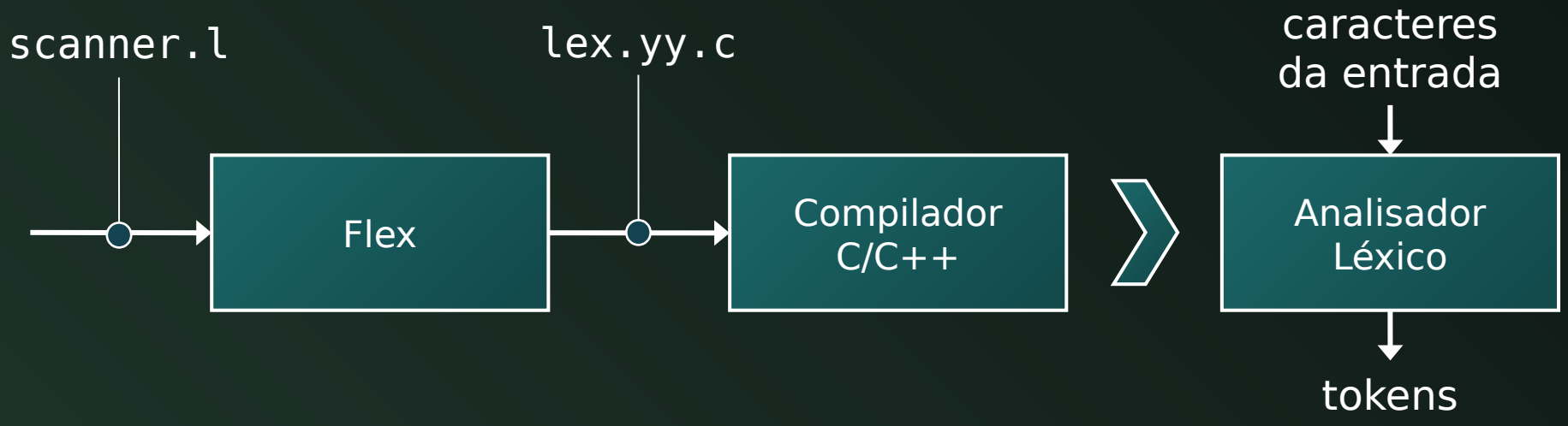
- As principais tarefas de um analisador léxico são:
  - Ler os caracteres da entrada
  - Agrupá-los em lexemas
  - Produzir tokens
- A criação de um analisador léxico pode ser automatizada
  - Especificando os tokens com expressões regulares
  - Convertendo-as em diagramas de transição
  - Gerando código para simulação dos diagramas

Introdução

- A ferramenta `Flex` é um gerador de analisadores léxicos
  - É uma implementação mais recente do Lex
  - Usa expressões regulares para descrever padrões de tokens
  - A ferramenta em si é um compilador
    - A notação de entrada é a linguagem `Flex`
    - Gera código em linguagem C/C++
- Existem ferramentas semelhantes para outras linguagens:
  - JFlex (Java), C# Flex (C#), Ragel (Objective-C e Ruby), PLY (Python), Alex (Haskell), Ocamllex (Ocaml), Gelex (Eiffel), Golex (Go), etc.

Ferramenta Flex

- Uso do `Flex`:
  - Um arquivo na linguagem `Flex` possui a extensão `.l`
  - O compilador `Flex` gera um arquivo chamado `lex.yy.c`
  - A saída do compilador C/C++ é o analisador léxico



Ferramenta Flex

- O `Flex` pode gerar código na linguagem C ou C++
  - Em C++, o arquivo gerado se chama lex.yy.cc
    - Ele é obtido com a opção `-+`, `--c++`, ou usando diretamente o flex++

```
$ flex --c++ scanner.l
$ flex++ scanner.l
```

    - O analisador é uma função `yylex()` que retorna um inteiro
      - O retorno representa o código para um dos tokens

```
yyFlexLexer lexer;
int token = lexer.yylex();
```

    - A classe `yyFlexLexer` é definida em `<FlexLexer.h>`

Ferramenta Flex

- A classe `yyFlexLexer` define também as funções:
  - `YYText()`
    retorna o “texto” do último token casado
  - `YYLeng()`
    retorna o comprimento do último token casado
  - `lineno()`
    retorna o número corrente da linha, se usado com `%option yylineno`
- O yy que aparece de forma recorrente nos nomes se refere ao gerador de analisador sintático Yacc

Estrutura do Programa Flex

- Um programa Flex possui o seguinte formato:

```
declarações
%%
regras de tradução
%%
funções auxiliares
```

  - Declarações
    - Inclusão de arquivos de cabeçalho
    - Declarações de constantes, variáveis e funções
    - Definições regulares

Estrutura do Programa Flex

- Exemplo de declarações:

```
%{
#include <iostream>
using std::cout;
// constantes para os tokens
enum {IF, THEN, ELSE, ID, NUM, RELOP};
%}

// definições regulares
delim    [ \t\n]
brancos  {delim}+
letra    [A-Za-z]
digito   [0-9]
id       {letra}({letra}|{digito})*
num      {digito}+(\.{digito}+)?(E[+-]?{digito}+)?

%%
...
```

Qualquer informação entre o par de delimitadores especiais `%{ %}`, é copiada diretamente para o arquivo `lex.yy.cc`

Estrutura do Programa Flex

- Regras de tradução
  - Regras no formato: `padrão ação`
    - São separados por espaços
      - Ação deve ser envolvida por chaves { } se ocupar mais de uma linha
    - Cada padrão é uma expressão regular
    - As expressões regulares podem usar definições regulares (declarações)
    - As ações são fragmentos de código escritos em C/C++
- Funções auxiliares
  - A definição de quaisquer funções utilizadas nas ações

Estrutura do Programa Flex

- Exemplo de regras de tradução:

```
...

%%
{brancos}                ; // nenhuma ação e nenhum retorno
if                  return IF;
then                return THEN;
else                return ELSE;
{id}                return ID;
{num}               return NUM;
"<"                 return RELOP;
"<="                return RELOP;
"="                 return RELOP;
"<>"                return RELOP;
">"                 return RELOP;
">="                return RELOP;
.                     cout << YYText() << " é um token inválido!";
%%
...
```

Estrutura do Programa Flex

- Exemplo de funções auxiliares:

```
...

%%

int insertId() {          /* função para inserir o lexema, cujo
                             primeiro caractere é apontado por YYText(),
                             e cujo tamanho é YYLeng(), na tabela de símbolos */

}

int getToken() {          /* função para buscar lexema na tabela de símbolos
                             e retornar seu token, ou zero, caso não encontrado

*/
}

int main() {              /* função principal chama o analisador léxico */
    yyFlexLexer lexer;
    lexer.yylex();
}
```

Resumo

- O Flex é um gerador de analisador léxico
  - Recebe especificações de tokens em formato de expressões regulares
  - Traduz as expressões regulares para diagramas de transição
  - Transforma os diagramas em código

- O código C/C++ gerado é um analisador léxico
  - Usado para fornecer tokens para um analisador sintático
  - Pode ser usado de forma independente em outras aplicações
    - Ferramentas textuais: conversores, tradutores, analisadores, etc.