Enunciado Laboratório 01 de Compiladores.
Entrega 14/03/2026

Você deve:

- **Fazer o trabalho individualmente;**
- **Implementar um programa com a solução;**
- **Traduzir o código C do arquivo amostra-correcao.c para a sua linguagem sorteada, gerando sua amostra de código;**
- **Testar sua solução sobre a sua amostra de código e gerar a saída;**
- **Submeter sua solução, sua amostra de código e a saída da execução do seu código sobre a amostra, SOMENTE pelo Classroom, até 14/03/2026;**
- **O nome do(s) arquivo(s) enviado(s) deve(m) ser no seguinte formato**
  - **compil-lab1-solucao-A-nome-completo-do-aluno.extensao**
  - **compil-lab1-amostra-B-nome-completo-do-aluno.extensao**
  - **compil-lab1-resposta-C-nome-completo-do-aluno**

Escreva um programa de computador que faça análise léxica do texto de um programa de computador na linguagem sorteada para você. Nesta análise não é necessário tratar modo XML (Scala), nem validade de caractere Unicode (Go). Também não é necessário tratar quaisquer caracteres além dos comuns ASCII. Também não é necessário tratar modos especiais de strings, bastando tratar pelo menos um modo de strings. Seu programa deve identificar todos os lexemas/ tokens/unidades léxicas da linguagem (exceto aqueles excluído acima). Sugiro fortemente utilizar um gerador automático de parser léxico, tal como JavaCC, ou uma linguagem com suporte a expressões regulares, tal como Ruby.

A saída do seu programa deve conter uma linha para cada lexema/token/ unidade léxica encontrada. A primeira string até o primeiro espaço de cada linha deve ser uma palavra cujas letras estejam em MAIÚSCULO identificando a classe do token. Por exemplo, ID para identificador, NUM_DEC para número decimal. Você deve escolher os nomes para as classes de Tokens. Em seguida, deve vir um espaço seguido pela sequência de caracteres do arquivo de entrada que correspondem ao token correspondente a esta linha.

Cada aluno terá uma linguagem a ele atribuída, através do sorteio descrito no documento postado no Classroom. Abaixo vemos a tabela de linguagens com links para documentação dela.

Tabela 1

0	Ada	http://www.ada-auth.org/standards/ada22.html
1	Ansi C	http://www.cs.columbia.edu/~sedwards/papers/sgi1999c.pdf
2	C#	https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/language-specification/lexical-structure#64-tokens
3	Ceylon	http://web.mit.edu/ceylon_v1.3.3/ceylon-1.3.3/doc/en/spec/html_single/#lexical
4	D	https://dlang.org/spec/lex.html
5	Dart	https://spec.dart.dev/DartLangSpecDraft.pdf
6	Eiffel	https://www.eiffel.org/doc/eiffel/Eiffel_programming_language
7	FreePascal	https://www.freepascal.org/docs-html/ref/refch1.html#x8-70001
8	Go	https://go.dev/ref/spec
9	Haskell	https://www.haskell.org/definition/haskell2010.pdf
10	Icon	https://www2.cs.arizona.edu/icon/ftp/doc/lb1up.pdf
11	Java SE 14	https://docs.oracle.com/javase/specs/jls/se14/jls14.pdf
12	Kotlin	https://kotlinlang.org/spec/syntax-and-grammar.html#syntax-and-grammar
13	Lua	https://www.lua.org/manual/5.1/manual.html (Não é necessário tratar "long brackets" de strings)
14	Nim	https://nim-lang.org/docs/manual.html#lexical-analysis
15	OCaml	https://ocaml.org/manual/5.2/lex.html
16	Ruby	https://ruby-doc.org/docs/ruby-doc-bundle/Manual/man-1.4/syntax.html#lexical https://ruby-doc.org/docs/ruby-doc-bundle/Manual/man-1.4/index.html
17	Rust	https://doc.rust-lang.org/stable/reference/tokens.html
18	Scala	https://www.scala-lang.org/files/archive/spec/2.11/01-lexical-syntax.html
19	Smalltalk	https://wiki.squeak.org/squeak/uploads/172/standard_v1_9-i
20	Swift	https://docs.swift.org/swift- book/ReferenceManual/LexicalStructure.html