

# Compiladores — Folha laboratorial 2

Pedro Vasconcelos, DCC/FCUP

Setembro 2022

## Um analisador lexical simples em Haskell

Considere um analisador lexical simples para os seguintes *tokens*:

**ID** identificadores (`xyz12`, `main`, `Abc`)

**NUM** constantes inteiras (`123`, `42`)

**LPAREN**, **RPAREN** parêntesis (`(` e `)`)

**COMMA** vírgula `,`

**IF** uma palavra reservada `if`

No diretório `src` encontra uma implementação deste analisador em Haskell como um programa que lê toda a entrada padrão e imprime a lista de *tokens*. Comece por compilar o programa:

```
$ ghc HsLexer.hs -o HsLexer
```

Depois experimente executá-lo com pequenos exemplos; deve obter a lista de *tokens* na saída padrão:

```
$ echo "if (xyz12) (foo(42,12,yy7))" | ./HsLexer
[IF,LPAREN,ID "xyz12",RPAREN,LPAREN,ID "foo",LPAREN,NUM 42,COMMA,NUM 12,COMMA,
ID "yy7",RPAREN,RPAREN]
```

Também pode redirectionar a entrada a partir de um ficheiro para testar exemplos com várias linhas; tente introduzir espaços ou caracteres inválidos e interprete os resultados.

## Exercício 1: Extender o analisador

Pretende-se que acrescente algumas funcionalidades extra a este analisador; pode optar por uma das duas linguagens (excepto na alínea (g) cuja questão é específica da linguagem C).

- Acrescente novos *tokens* `LBRACE` e `RBRACE` para chavetas `{` e `}` e `SEMICOLON` para `;`.
- Acrescente novos *tokens* para mais palavras reservadas e tipos: `WHILE`, `FOR`, `INT`, `FLOAT`.
- Corriga o caso em falta nos identificadores (`_` deve ser aceite como se fosse uma letra).

- (d) Acrescente um novo *token* REAL para números de vírgula-flutuante (e.g. 0.5, 123.45); para simplificar não precisa de considerar números em notação científica (e.g. 1.23e9).
- (e) Acrescente o tratamento de comentários multi-linha `/* ... */` e até ao final da linha `// ...`; note que os comentários devem ser ignorados (não são *tokens*).

## Exercício 2: Expressões regulares

- (a) Escreva expressões regulares para descrever cada um dos *tokens* deste analisador (incluindo os acrescentados no Exercício 1).
- (b) Generalize a expressão para o *token* REAL de forma a aceitar também números em *notação científica*. Exemplos: 12.34e+12, 1e-12, 123.4E+9
- (c) Escreva expressões regulares para comentários até ao final da linha (`// ...`) e comentários multi-linha (`/* ... */`). Atenção: estes últimos são mais difíceis de descrever como expressão regular do que parece!

## Exercício 3: Re-implementação usando alex ou flex

Re-implemente o analisador léxico usando um gerador automático `alex` ou `flex`. Para tal deve criar um ficheiro `.x` ou `.lex` com as descrições de *tokens* usando expressões regulares, correr o gerador para obter o código Haskell ou C e o compilador para obter o analisador.

```
$ alex Lexer.x
$ ghc Lexer.hs -o Lexer
```

Nota: neste exercício o analisador é um programa completo; em programas maiores, o analisador será um módulo que deve compilado juntamente com outros. Nesses casos iremos usar uma ferramenta como `make` ou `cabal` para automatizar os passos acima. k