

```
/**
 * Arquivo: scanner.h
 * Versão : 1.0
 * Data   : 2024-10-17 23:32
 * -----
 * Esta interface implementa um programa scanner, que "lê" uma linha de texto
 * agrupando as letras para formar palavras (tokens) que são reconhecidas como
 * uma unidade coerente. O scanner divide a string do texto em seus tokens
 * componentes. Um token é definido como:
 *
 * 1. Uma sequência de caracteres alfanuméricos consecutivos (letras ou
 *    números); OU
 *
 * 2. Uma string de apenas um caractere contendo um espaço; OU
 *
 * 3. Uma string de apenas um caractere contendo um sinal de pontuação.
 *
 * Para utilizar esta interface, inicialmente você deve inicializar o scanner:
 *
 *    iniciar_scanner(linha);
 *
 * "linha" é uma string (tipicamente obtida do usuário) que será dividida em
 * seus tokens componentes. Para obter cada token, após iniciar o scanner:
 *
 *    token = obter_proximo_token( );
 *
 * Quando o último token for lido, o predicado "final_da_linha" retorna TRUE,
 * de forma que um loop como
 *
 *    while (!final_da_linha( ))
 *    {
 *        token = obter_proximo_token( );
 *        . . . processamento do token . . .
 *    }
 *
 * serve como um idioma para o processamento de cada token da linha.
 *
 * Baseado em: The Art and Science of C, de Eric S. Roberts.
 *             Capítulo 10: Modular Development (pg. ).
 *
 * Prof.: Abrantes Araújo Silva Filho (Computação Raiz)
 *        www.computacaoraiz.com.br
 *        www.youtube.com.br/computacaoraiz
 *        github.com/computacaoraiz
 *        twitter.com/ComputacaoRaiz
 *        www.linkedin.com/company/computacaoraiz
 *        www.abrantes.pro.br
 *        github.com/abrantesasf
 */

/** Inicia Boilerplate da Interface */

#ifndef _SCANNER_H
#define _SCANNER_H

/** Includes */

#include "genlib.h"

/** Declarações de Subprogramas: */
```

```
/**
 * PROCEDIMENTO: iniciar_scanner
 * Uso: iniciar_scanner(string);
 * -----
 * Este procedimento inicializa o scanner e o configura de modo a permitir que
 * ele leia tokens a partir de uma linha (string). Após a inicialização do
 * scanner, a chamada da função "obter_proximo_token" retornará uma string que
 * contém o primeiro token da linha; a próxima chamada da função
 * "obter_proximo_token" retornará a string que contém o segundo token da linha
 * e assim por diante.
 */

void iniciar_scanner (string linha);

/**
 * FUNÇÃO: obter_proximo_token
 * Uso: str = obter_proximo_token( );
 * -----
 * Esta função retorna o próximo token da linha.
 */

string obter_proximo_token (void);

/**
 * PREDICADO: final_da_linha
 * Uso: if(final_da_linha( )) . . .
 * -----
 * Este predicado retorna TRUE se o scanner alcançou o final da linha.
 */

bool final_da_linha (void);

/** Finaliza Boilerplate da Interface */

#endif
```