

```
/**
 * Arquivo: scanner_nao_encapsulado.c
 * Versão : 1.0
 * Data   : 2024-10-18 10:52
 * -----
 * Este arquivo implementa um programa cliente para o teste da interface
 * scannerTAD.h, que "lê" um 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.
 *
 * 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
 */

#include <ctype.h>
#include "genlib.h"
#include "scannerTAD.h"
#include "simpio.h"
#include "strlib.h"

/**** Função Main: ****/

int main (void)
{
    // Frases sem acentuação, nosso scanner não lida bem com isso!
    string frase0 = "Este e um teste!";
    string frase1 = "o rato roeu a roupa do rei de roma.";
    string frase2 = "THE DESIGN OF THE UNIX OPERATING SYSTEM.";

    // Iniciar scanners para ler as frases:
    scannerTAD scanner0 = criar_scanner();
    ler_string(scanner0, frase0);

    scannerTAD scanner1 = criar_scanner();
    ler_string(scanner1, frase1);

    scannerTAD scanner2 = criar_scanner();
    configurar_tratamento_de_espacos(scanner2, 1);
    ler_string(scanner2, frase2);

    // Token obtido:
    string token;

    // Imprimir os tokens:
    printf("Frase 0: %s\n", frase0);
    for (int i = 0; i < 8; i++)
    {
        token = obter_token(scanner0);
        printf("    Token: %s\n", token);
        liberar_token(&token);
    }
}
```

```
printf("\n");

printf("Frase 1: %s\n", frase1);
for (int i = 0; i < 3; i++)
{
    token = obter_token(scanner1);
    printf("    Token: %s\n", token);
    liberar_token(&token);
}
printf("\n");

printf("Frase 2: %s\n", frase2);
for (int i = 0; i < 3; i++)
{
    token = obter_token(scanner2);
    printf("    Token: %s\n", token);
    liberar_token(&token);
}
printf("\n");

printf("Frase 1: %s\n", frase1);
for (int i = 0; i < 3; i++)
{
    token = obter_token(scanner1);
    printf("    Token: %s\n", token);
    liberar_token(&token);
}
printf("\n");

printf("Frase 2: %s\n", frase2);
for (int i = 0; i < 3; i++)
{
    token = obter_token(scanner2);
    printf("    Token: %s\n", token);
    liberar_token(&token);
}
printf("\n");

// Remover os scanners:
remover_scanner(&scanner0);
remover_scanner(&scanner1);
remover_scanner(&scanner2);
}
```