UNISINOS		
	Curso	Ciência da Computação - Híbrido
	Atividade Acadêmica	Tradutores
	Ano/Semestre	2023/01a
	Professor	Leandro Teodoro
	Data de Entrega	06/09/2023
	Resposta ao Desafio M2 – Tradutores	

Construa um Analisador Léxico que reconheça:

- **Variáveis ou identificadores:** este analisador léxico deve ser capaz de reconhecer nomes de variáveis, funções, parâmetros de funções em um código fonte. Para esta tarefa, você **não precisa** realizar **a análise de escopo** para a definição dos identificadores:

  Exemplo:
  - Trecho de código:
  int x = 7;
  int y;

  - Tokens gerados:
  [reserved_word, int] **[id, 1]** [Equal_Op, =] [num, 7]
  [reserved_word, int] **[id, 2]**

- **Constantes numéricas (números inteiros):** este analisador léxico deve ser capaz de reconhecer um número inteiro qualquer e convertê-lo para os respectivos tokens:

  Exemplo:
  - Trecho de código:
  int x = 7 + 25 * 52;

  - Tokens gerados:
  [reserved_word, int] [id, 1] [Equal_Op, =] **[num, 7]** [Arith_Op, +] **[num, 25]**
  [Arith_Op, *] **[num, 52]**

- **Palavras reservadas:** este analisador léxico deve ser capaz de reconhecer palavras reservadas. Por exemplo, *do, while, if, else, switch, for, return, null, int, float, double, string, bool, break, case, etc* e convertê-las para os respectivos tokens:

  Exemplo:
  - Trecho de código:
  if(x == 10)

  - Tokens gerados:
  [reserved_word, if] [id, 1] [Relational_Op, ==] [num, 10]

- **Operadores relacionais:** este analisador léxico deve ser capaz de reconhecer os operadores relacionais: *<, <=, ==, !=, >=, >* e convertê-los para os respectivos tokens:

  Exemplo:

  - Trecho de código:
  while(x != 0)

  - Tokens gerados:
  [reserved_word, while] [id, 1] **[Relational_Op, !=]** [num, 0]

- **Números de ponto flutuante (números reais):** este analisador léxico deve ser capaz de reconhecer números reais quaisquer e convertê-los para os respectivos tokens:

  Exemplo:

  - Trecho de código:
  int x = 7.15 - 2.13;

  - Tokens gerados:
  [reserved_word, int] [id, 1] [Equal_Op, =] **[num, 7.15]** [Arith_Op, -] **[num, 2.13]**

- **Remoção de espaços em branco e comentários:** este analisador léxico deve ser capaz de reconhecer espaços em branco e comentários no código fonte e removê-los (ignorá-los) .

  Exemplo:

  - Trecho de código:
  //Comentário 1
  /* Comentário 2 */

- **Strings:** este analisador léxico deve ser capaz de reconhecer os strings e convertê-las para seus respectivos tokens:

  Exemplo:

  - Trecho de código:
  String sexo = “masculino”;

  - Tokens gerados:
  [reserved_word, String] [id, 1] [equal, =] **[string_literal, masculino]**

- **Operadores lógicos:** este analisador léxico deve ser capaz de reconhecer os operadores lógicos:|| && e convertê-los para os respectivos tokens:
  Exemplo:
  - Trecho de código:
  if(idade > 70 && sexo == “masculino”)

  - Tokens gerados:
  [reserved_word, if] [id, 1] [Relational_Op, >] [num, 70] **[logic_op, &&]** [id, 2] [Relational_Op, ==] [Relational_Op, string_literal]

- **Demais caracteres:** este analisador léxico deve ser capaz de reconhecer os caracteres: **= ( ) { } , ;** e convertê-los para seus respectivos tokens:
  Exemplo:

  [equal, =] [l_paren, (] [r_paren,)] [l_bracket, {] [r_bracket, }] [r_bracket, }] [comma, ,] [semicolon, ;]

O trabalho pode ser realizado em grupos de até **3 alunos**, bem como deverá ser entregue no Canvas até o dia **24/08**. A seguir, o código que o analisador léxico deve receber para gerar o conjunto de tokens descrito anteriormente:

```
int main()
{
    int v[] = {5, 10, 15, 3, 10, 76, 5, 13, 33, 45};
    int * pt;
    int i;

    pt = v; //Atribui o endereço do vetor

    AlterarVetor(v, 10);

    for(int i = 0; i < 10; i++)
    {
        printf("V[%i] = %i\r\n", i, *(pt + i));
    }

    CalculoMedia();
    VerificaNumero();

    return 0;
}
```

O trabalho pode ser elaborado utilizando o FLex (gcc/g++), JFlex (java), etc.
Um ótimo trabalho a todos!