Leitor de Token

Um Token na Ciência da Computação é um segmento de texto ou símbolo que representa uma unidade lógica da string. Por exemplo:

A string: “Pergunta: Qual é o significado da variável cont2? É a quantidade de elementos do vetor.”

Tokens:

Pergunta : . Qual . é . o . significado . da . variável . cont2 ? . Representa . os . 10 . elementos . do . vetor .

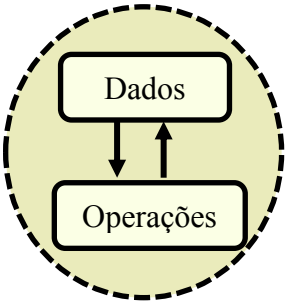
Tarefa 1

Escreva uma estrutura de dados que implemente o TAD Leitor de Tokens.

Dados:

```
typedef struct {
  char* str;
  int tam;
  int pos;

}TokenReader;
```



Operações

TokenReader *newTokenReader(char *str);

Cria um novo leitor de tokens e devolve seu endereço. Essa operação permite que vários leitores sejam criados e utilizados simultaneamente.

void freeTokenReader(TokenReader *reader);

Essa função libera a memória ocupada pelo leitor `reader` alocado na função newTokenReader.

char* nextToken(TokenReader *reader);

Esta função devolve o próximo token do Leitor `reader`. Se não houver mais tokens, a função devolve uma string de comprimento 0.

A string devolvida pela função deverá ser alocada dinamicamente (malloc) para que o usuário da função possa liberar o espaço pela função free quando não for mais necessário.

Caso nenhuma string for vinculada ao leitor a função também deverá devolver uma string de tamanho 0.

void setTokenString(TokenReader *reader, char* str);

A função carrega a string `str` no leitor `reader`.

`int hasMoreTokens(TokenReader *reader);`

A função devolve `1` se o leitor `reader` ainda tem tokens a serem consumidos e `0` se todos os tokens foram consumidos.

Exemplo da utilização do Leitor

```
1.   char *s = “Pergunta: Qual é o significado da variável cont2?”;
2.   TokenReader reader = newTokenReader(s);
3.   char* str = nextToken(reader);
4.   printf(“ %s \n”, str);                          //Imprime “Pergunta”
5.   free(str);
6.   str = nextToken(reader);
7.   printf(“ %s \n”, str);
8.   free(str);                                      //Imprime “:”
9.   while(hasMoreTokens(reader)){                   //Imprime o restante dos tokens
10.  str = nextToken(reader);
11.  printf(“ %s \n”, str);
12.  free(str);
13.  }
14.  freeTokenReader(reader);
```

Observações:

Outras funções podem ser criadas com o objetivo de auxiliar as funções definidas no TAD. Por exemplo, poderíamos definir uma função para nos auxiliar na extração de substrings da string original, ou uma função para providenciar uma cópia de string com alocação dinâmica.

```
char* subString(char* s, int beginIndex, int endIndex);
```

Modelo de cabeçalho dos programas

```
/**********************************
Algoritmos e Estrutura de Dados 1
Prof. Rafael Liberato
----------------------------------
Exercício: <Lista>.<Exercício>.<Item>
Aluno: <Nome do aluno>
Data: <data>
**********************************/
```

Exemplo:

```
/**********************************
Algoritmos e Estrutura de Dados 1
Prof. Rafael Liberato
----------------------------------
Exercício: APS
Aluno: João da Silva
Data: 10/12/12
**********************************/
```