MINISTÉRIO DA EDUCAÇÃO
CEFET/RJ
CAMPUS NOVA FRIBURGO
CURSO DE BACHARELADO EM SISTEMAS DE INFORMAÇÃO

Algoritmos e Estruturas de Dados
Professor Rodrigo Reis Gomes
Simulado
Semestre 2024/2

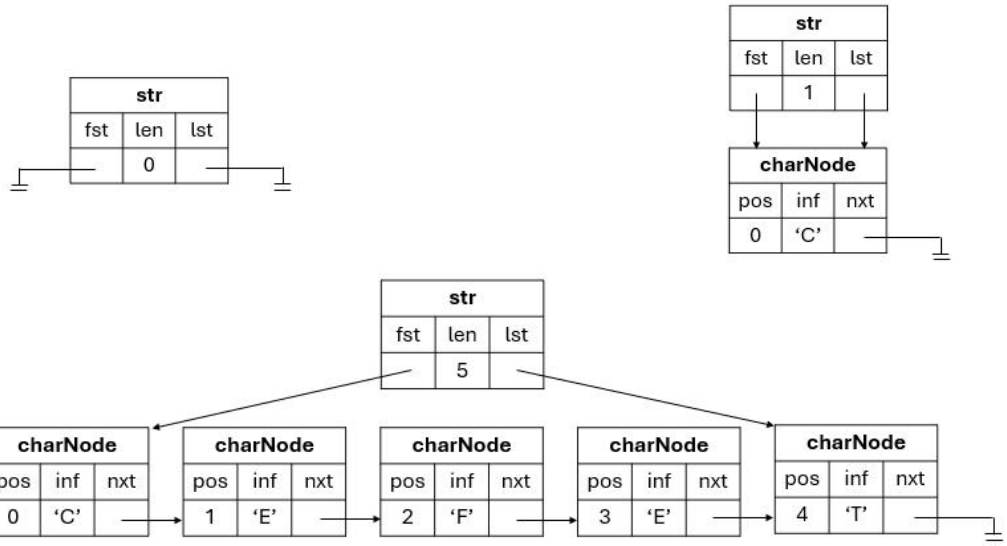
Aluno(a): _____Data: _____/_____/_____

Descrição:

A linguagem de programação C não oferece um tipo nativo capaz de representar um texto. O programador precisa, por exemplo, modelar dados textuais usando ponteiros para caracteres ou vetores de caracteres. O objetivo deste documento é oferecer instruções para você construir uma biblioteca, ou um arquivo de cabeçalho ".h", que reúna tipos e funções para implementar a manipulação de textos em C.

A ideia baseia-se em dois tipos de registro: *charNode* e *str*. O primeiro representa um nó de uma lista encadeada que armazena um caractere do texto, a posição desse caractere no texto e um ponteiro que deve fazer a ligação para o próximo nó da lista, ou ficar aterrado, quando o nó for final. O segundo modela um tipo que aponta para os nós primeiro e último da lista que armazena os caracteres do texto, ainda guardando a informação acerca do tamanho dessa lista, isto é: a quantidade de caracteres do texto.

Eis ilustrações que indicam um texto vazio, um texto unitário e um texto com mais de um caractere:



Seguem os códigos em linguagem C que descrevem esses tipos:

```c
typedef struct TCharNode
{
  int pos; //posição do nó na string
  char inf; //caractere presente no nó
  struct TCharNode * nxt; //ponteiro para o próximo nó
} charNode;

typedef struct TStr
{
  charNode * fst; //ponteiro para o primeiro nó da string
  charNode * lst; //ponteiro para o último nó da string
  int len; //comprimento da string
} str;
```

Aluno(a): _____Data: _____/_____/_____

Em prosseguimento, quatro funções precisam ser programadas. São elas:

```
/*****************************************************************************
Inicializa a variável do tipo str usada como argumento (passado por referência) para o parâmetro
desta função.
*****************************************************************************/
void  initStr(str *);

/*****************************************************************************
Lê, do teclado, uma variável do tipo str usada como argumento (passado por referência) para o
parâmetro desta função.
*****************************************************************************/
void getStr(str *);

/*****************************************************************************
Imprime, no teclado, uma variável do tipo str usada como argumento para o parâmetro desta função.
*****************************************************************************/
void printStr(str);

/*****************************************************************************
Libera a área de memória que foi utilizada para armazenar uma variável do tipo str. Essa variável é
usada como argumento (passado por referência) para o parâmetro desta função.
*****************************************************************************/
void freeStr(str *);
```

Depois dessas codificações, o tipo *str* poderá ser usado, por exemplo, como no arquivo ".c" descrito abaixo:

```
#include<stdio.h>
#include<stdlib.h>
#include"UString.h" //Arquivo que contém as especificações do tipo str.

int main()
{
  str s;
  initStr(&s);
  printf("Escreva um texto qualquer:\n");
  getStr(&s);
  printf("Seu texto:\n");
  printStr(s);
  freeStr(&s);

  printf("\n\n");
  system("PAUSE");
  return 0;
}
```

Implemente as funções do arquivo ".h". Seguem algumas considerações sobre elas:

Aluno(a): _____Data: _____/_____/_____

a) **(2,5 pontos)** Sobre a função *initStr*, os campos *fst*, *lst* e *len* precisam receber os valores iniciais que configuram um texto vazio.

b) **(2,5 ponto)** Sobre a função *freeStr*, toda a área de memória alocada para a construção de um elemento do tipo *str* precisa ser liberada e os campos *fst*, *lst* e *len* precisam ter seus valores iniciais restaurados.

c) **(2,5 pontos)** Sobre a função *getStr*,
   i. é importante resetar qualquer valor que possa existir na variável que receberá seu novo valor via teclado;
   ii. é preciso ler caracteres do teclado, um a um, até a leitura do caractere ENTER ('\n');

   Observação: a função *getchar*, da biblioteca *stdio.h*, faz a leitura de um caractere do teclado. Pesquise sobre o seu funcionamento.

   iii. é necessário incluir cada um desses caracteres, desconsiderando o ENTER, num novo nó (*charNode*) a fazer parte da lista encadeada apontada pelo tipo *str*;
   iv. e ajustar os valores dos outros campos de cada novo nó (*charNode*), mas também do próprio texto (*str*).

d) **(2,5 pontos)** Sobre a função *printStr*, é preciso percorrer todos os nós que comportam o texto para imprimir o caractere presente em cada um deles.

Muita atenção e bom simulado.