Introdução a Programação (IF669-EC)

Adriano Sarmento
Centro de Informática-UFPE

Exercícios Lista Encadeada

1ª Questão

Dado que o tipo String é definido como uma lista simplesmente encadeada descrita abaixo:

```
typedef struct string {
     char caractere;
     struct string *prox;
} String;
```

Implemente a função `str_compara` cuja assinatura é dada abaixo:

```
/* Compara 2 Strings lexicamente. Retorna 0 se
Strings str1 e str2 são iguais, -1 se str1 < str2, e
1 se str1 > str2 */
int str_compara (String* str1, String* str2);
```

Obs: Não pode usar a biblioteca string.h

2ª Questão

Considere um nó de uma lista duplamente encadeada é dada pela seguinte declaração:

```
typedef struct listaOrd {
     int valor;
     struct listaOrd *prox;
     struct listaOrd *ant;
} ListaOrd;
```

Implemente a função `insercaoOrdenada` que dado o endereço inicial de uma lista ordenada e o valor a ser inserido na lista, insira este novo elemento de forma que a lista fique ordenada de forma crescente. Esta função deve retornar o endereço inicial da lista passada como parâmetro.

3ª Questão

Considere para a resolução da questão que o TAD Fila de reais esteja implementado e que você desconheça a representação interna dele. O TAD é descrito abaixo:

```
typedef struct fila Fila;
Fila* fila_cria();
void fila_insere (Fila* f, float valor);
float fila_retira (Fila* f);
int fila_vazia (Fila* f);
void fila_libera(Fila* f);
```

Escreva uma função utilizando Fila de reais que dado um ponteiro para uma Fila, inverta a ordem de elementos da Fila.

4ª Questão

Considere para a resolução da questão que o TAD Pilha de inteiros esteja implementado e que você desconheça a representação interna dele. O TAD é descrito abaixo:

```
typedef struct pilha Pilha;
Pilha* pilha_cria();
void pilha_push (Pilha* p, int valor);
int pilha_pop (Pilha* p);
int pilha_vazia (Pilha* p);
void pilha_libera(Pilha* p);
```

Escreva uma função utilizando Pilha de inteiros que dado um ponteiro para uma Pilha, retire desta pilha o maior valor existente. Se existir maiores valores repetidos, a função deve retirar apenas um deles. No final da função, todos os elementos da pilha devem estar na ordem original, excetuando-se naturalmente o valor que foi retirado.

Solução

1ª Questão

```
/* Compara 2 Strings lexicamente. Retorna 0 se Strings str1 e str2 são
iguais, -1 se str1 < str2, e 1 se str1 > str2 */
int str_compara (String* str1, String* str2) {
    String* p;
    String* q;
    int resultado = 0;
    for (p=str1,q=str2; p != NULL && q != NULL && resultado == 0;
         p=p->prox, q=q->prox) {
        if (p->caractere > q->caractere)
           resultado = 1;
        else if (p->caractere < q->caractere)
            resultado = -1;
    }
    if (resultado == 0) {
      if ((p != NULL) && (q == NULL))
       resultado = 1;
      else if ((p == NULL) && (q != NULL))
      resultado = -1;
   }
   return resultado;
}
```

2ª Questão

```
ListaOrd* insereOrdenado (ListaOrd* inicial, int val) {
    ListaOrd* ant = NULL;
    ListaOrd* p = inicial;
    ListaOrd* novo = (ListaOrd*) malloc(sizeof(ListaOrd));
    novo->valor = val;
    while (p != NULL && p->valor < val) {
        ant = p;
        p = p->prox;
    }
    if (ant == NULL) {
        novo->prox = p;
        inicial = novo;
    ]

    } else {
        novo->prox = p;
        p->ant = novo;
        ant->prox = novo;
    }
    novo->ant = ant;

    return inicial;
```

```
}
```

3ª Questão

```
void inverteFila(Fila* fila) {

    int i, tamanho = 0;
    float* vetor;
    Fila* aux = fila_cria();

    while (!fila_vazia(fila)){
          fila_insere(aux, fila_retira(fila));
          tamanho++;
    }
    if ( tamanho != 0) {
      vetor = (float*) malloc(tamanho* sizeof(float));
      i = tamanho - 1;
      while (!fila_vazia(aux)) {
          vetor[i] = fila_retira(aux);
          i--;
      }
     i = 0;
     while (i < tamanho) {
       fila_insere(fila,vetor[i]);
       i++;
     }
   }
   fila_libera(aux);
   free(vetor);

}
```

4ª Questão

```
void retiraMaior(Pilha* pilha) {

    int maiorValor, valor;
    int tirou = 0;
    Pilha * aux = pilha_cria();
    if (!pilha_vazia(pilha)) {
          maiorValor = pilha_pop(pilha);
          pilha_push(aux,maiorValor);
    }
    while (!pilha_vazia(pilha)){
```

```
            valor = pilha_pop(pilha);
            pilha_push(aux,valor);
            if (valor > maiorValor) {
                 maiorValor = valor;
            }
     }
     while (!pilha_vazia(aux)) {
            valor = pilha_pop(aux);
            if (valor != maiorValor || tirou == 1){
                 pilha_push(pilha,valor);
            } else {
                 tirou = 1;
            }
    }
    pilha_libera(aux);

}
```