LISTA 3 | LISTAS ENCADEADAS

Alinne Oliveira e Enos Andrade

1 Dado trecho de código a seguir, crie um programa em C que realize as seguintes operações sobre a lista: Criação; Inserção; Impressão; Teste de vazia; Busca; Remover um elemento; Excluir toda a lista.

```c
struct elemento{
    int info;
    struct elemento *prox;
};
typedef struct elemento Elemento;
```

Solução

```c
#include <stdio.h>
#include <malloc.h>
#include <stdbool.h>

typedef struct elemento{
    int info;
    struct elemento* prox;
} ELEMENTO;

typedef struct {
    ELEMENTO* inicio;
} LISTA;

void criarLista(LISTA *l){
    l->inicio = NULL;
}

void inserirElemListaInicio(LISTA *l, int info){
    ELEMENTO* e = malloc(sizeof(ELEMENTO));
    e->info = info;
    e->prox = l->inicio;
    l->inicio = e;
}

void imprimirLista(LISTA *l){
    ELEMENTO* end = l->inicio;
    printf("Lista: \" ");
    while(end != NULL){
        printf("%i ", end->info);
        end = end->prox;
    }
    printf("\"\n");
}

bool estaVazia(LISTA *l){
    if(l == NULL || l->inicio == NULL) return true;
    return false;
}

ELEMENTO* buscaSequencial(LISTA *l, int info){
    ELEMENTO* pos = l->inicio;
    while(pos != NULL){
```

```
43          if(pos->info == info) return pos;
44          pos = pos->prox;
45      }
46      return NULL;
47  }
48
49  bool excluirElemLista(LISTA *l, int info){
50      ELEMENTO* ant = NULL;
51      ELEMENTO* i = l->inicio;
52      while(i != NULL && i->info != info){
53          ant = i;
54          i = i->prox;
55      }
56      if(i == NULL) return false; // O elemento não existe;
57      if(ant == NULL) l->inicio = i->prox;
58      else ant->prox = i->prox;
59      free(i);
60      return true;
61  }
62
63  void excluirLista(LISTA *l){
64      ELEMENTO* e = l->inicio;
65      while(e != NULL){
66          ELEMENTO* apagar = e; //Variável auxiliar
67          e = e->prox;
68          free(apagar);
69      }
70      l->inicio = NULL;
71  }
72
73  void verificarLista(LISTA *l){
74      if(estaVazia(l)) printf("A lista esta vazia!\n");
75      else printf("A lista nao esta vazia!\n");
76      imprimirLista(l);
77  }
78
79  int main(){
80      LISTA *l = (LISTA *) malloc(sizeof(LISTA));
81      criarLista(l);
82      printf("Lista criada!\n");
83      verificarLista(l);
84
85      printf("\nInserindo os elementos\n");
86      int regs[3];
87      regs[0] = 7;
88      regs[1] = 14;
89      regs[2] = 33;
90      inserirElemListaInicio(l, regs[0]);
91      inserirElemListaInicio(l, regs[1]);
92      inserirElemListaInicio(l, regs[2]);
93      verificarLista(l);
94
95      printf("\nRemovendo o primeiro elemento\n");
96      if(excluirElemLista(l, regs[2])) printf("Elemento %d excluido\n", regs[2]);
97      else printf("Elemento %d nao excluido\n", regs[2]);
98      verificarLista(l);
99
100     printf("\nExcluindo lista\n");
101     excluirLista(l);
```

```
102     verificarLista(l);
103 }
```

2 A informação associada a cada nó de uma lista encadeada pode ser mais complexa, sem alterar o encadeamento dos elementos. As funções apresentadas para manipular listas de inteiros podem ser adaptadas para tratar listas de outros tipos. O campo da informação pode ser representado por um ponteiro para uma estrutura, em lugar da estrutura em si independente da informação armazenada na lista, a estrutura do nó é sempre composta por: um ponteiro para a informação e um ponteiro para o próximo nó da lista.

Baseado nestas informações e no trecho de código a seguir, implemente uma função que insira elementos em uma lista de Pontos.

```
struct ponto{
    float x;
    float y;
};
typedef struct ponto Ponto;
```

```
struct elemento {
    Ponto* info;
    struct elemento *prox;
};
typedef struct elemento Elemento;
```

Solução

```
#include <stdio.h>
#include <malloc.h>

typedef struct ponto{
    float x;
    float y;
} Ponto;

typedef struct elemento{
    Ponto* info;
    struct elemento *prox;
} Elemento;

typedef struct{
    Elemento* inicio;
} Lista;

void criarLista(Lista *l){
    l->inicio = NULL;
}

void exibirLista(Lista *l){
    Elemento* end = l->inicio;
    printf("Lista: \"");
```

```
    while(end != NULL){
        printf("(%.1f, %.1f) ", end->info->x, end->info->y);
        end = end->prox;
    }
    printf("\"\n");
}

void inserirElemListaInicio(Lista *l, Ponto* p){
    Elemento* e = (Elemento*) malloc(sizeof(Elemento));
    e->info = p;
    e->prox = l->inicio;
    l->inicio = e;
}

int main(){

    Lista *l = (Lista*) malloc(sizeof(Lista));
    criarLista(l);
    Ponto* p = (Ponto *) malloc(sizeof(Ponto));
    p->x = 5.0;
    p->y = 10.0;
    Ponto* q = (Ponto *) malloc(sizeof(Ponto));
    q->x = 20.0;
    q->y = 30.0;
    inserirElemListaInicio(l, p);
    inserirElemListaInicio(l, q);
    exibirLista(l);

    return 0;
}
```

3 Em uma agenda telefônica os contatos são cadastrados com os seguintes dados: Nome, Telefone e DataAniversario; Essas informações podem ser representadas em um tipo estruturado: Contato

CONTATO
Nome
Telefone
Aniversário

Utilizando listas encadeadas, escreva um programa que permita o cadastro, remoção, busca e impressão de contatos desta agenda telefônica. Os elementos da lista encadeada para armazenar contatos são representados pela seguinte estrutura:

```
struct elemento {
    Contato info;
    struct elemento* prox;
};
typedef struct elemento Elemento;
```

O seu programa deve implementar as seguintes funções:

cria_agenda – cria uma nova lista encadeada retornando um ponteiro para NULL;

insere_contato – insere um novo contato na lista encadeada respeitando a ordem alfabética dos nomes dos contatos já existentes na agenda;

lista_contatos – exibe na tela todos os dados dos contatos existentes na agenda;

busca_contato – busca um contato na agenda com base em um determinado nome informado pelo usuário. A função retorna o endereço de memória do elemento encontrado ou NULL caso o contato não seja encontrado;

remove_contato – deleta um determinado contato existente na agenda. A função deve permitir ao usuário buscar por um contato na agenda (utilizando a função busca_contato previamente criada) e em seguida remover da lista o contato. Se o contato buscado não for encontrado, o programa deve exibir uma mensagem informando o usuário sobre esse fato;

O programa deve exibir um menu para o usuário escolher as operações desejadas. Exemplo:

1 – Inserir Contato
2 – Listar Contatos
3 – Buscar Contato
4 – Editar Contato
5 – Remover Contato
6 – Remover Contatos Duplicados
7 – Sair

Solução

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct{
    char nome[30];
    char telefone[30];
    char aniversario[30];
} Contato;

typedef struct elemento{
    Contato info;
    struct elemento *prox;
} Elemento;

typedef struct{
    int qtd;
    Elemento* inicio;
} Lista;

void cria_agenda(Lista *l){
    l->inicio = NULL;
    l->qtd = 0;
}

void insere_contato (Lista *l, Contato c){
    Elemento* ant = NULL;
    Elemento* atual = l->inicio;
    Elemento* e = (Elemento*) malloc(sizeof(Elemento));
    e->info = c;
```

```
31      for(int i = -1; i < l->qtd -1; i++){
32        if (strcmp(e->info.nome, atual->info.nome) <= 0) break;
33        ant = atual;
34        atual = atual->prox;
35      }
36      if (ant == NULL) {
37          e->prox = l->inicio;
38          l->inicio = e;
39      }else{
40          e->prox = ant->prox;
41          ant->prox = e;
42      }
43      l->qtd++;
44  }
45
46  void lista_contatos(Lista *l){
47      printf("\n------------------- LISTAR CONTATOS -------------------\n");
48      Elemento *e = l->inicio;
49      int i = 1;
50      if(l->qtd > 0){
51          while(e != NULL){
52              printf("%d - Nome: %s - Telefone: %s - Aniversario: %s\n", i,
                    ↪  e->info.nome, e->info.telefone, e->info.aniversario);
53              e = e->prox;
54              i++;
55          }
56      }else{
57          printf("Voce ainda nao inseriu contatos na agenda\n");
58      }
59  }
60
61  Elemento* busca_contato(Lista *l, char busca[30], int* n){
62      Elemento *e = l->inicio;
63      for (int i = 0; i < l->qtd; i++) {
64          if(strcmp(e->info.nome, busca) == 0){
65              *n = i+1;
66              return e;
67          }
68          e = e->prox;
69      }
70      return NULL;
71  }
72
73  void remove_contato(Lista *l, Elemento *e){
74      Elemento *ant = NULL;
75      Elemento *atual = l->inicio;
76      while (atual != NULL && atual != e) {
77          ant = atual;
78          atual = atual->prox;
79      }
80      if (ant == NULL) l->inicio = atual->prox;
81      else ant->prox = atual->prox;
82      free(atual);
83      l->qtd--;
84  }
85
86  void remove_duplicados(Lista *l){
87      printf("\n------------------- REMOVER CONTATOS DUPLICADOS
          ↪  -------------------\n");
```

```
88      if (l->qtd <= 1) {
89          printf("\nNenhum contato duplicado!\n");
90          return;
91      }
92      Elemento *e = l->inicio;
93      Elemento *prox = e->prox;
94      int qtd = 0;
95      while (e != NULL) {
96          while (prox != NULL) {
97              if (strcmp(e->info.nome, prox->info.nome) == 0 &&
                ↪  strcmp(e->info.telefone, prox->info.telefone) == 0 &&
                ↪  strcmp(e->info.aniversario, prox->info.aniversario) == 0){
98                  Elemento *aux = prox;
99                  remove_contato(l, prox);
100                 prox = aux->prox;
101                 qtd++;
102             }else prox = prox->prox;
103         }
104         e = e->prox;
105         if (e->prox == NULL) break;
106         prox = e->prox;
107     }
108     printf("\n%d contatos foram removidos\n", qtd);
109 }
110
111 //Impementacao
112 void inserirContato(Lista *l){
113     printf("\n------------------- INSERIR CONTATO -------------------\n");
114     Contato contato;
115     char nome[30], telefone[30], aniversario[30];
116     printf("Nome: "); setbuf(stdin, NULL);
117     gets(contato.nome);
118     printf("Telefone: "); setbuf(stdin, NULL);
119     gets(contato.telefone);
120     printf("Data de aniversario: "); setbuf(stdin, NULL);
121     gets(contato.aniversario);
122     insere_contato(l, contato);
123     printf("\nContato inserido com sucesso!\n");
124 }
125
126 Elemento* buscarContato(Lista *l){
127     printf("\n------------------- BUSCAR CONTATO -------------------\n");
128     char busca[30];
129     printf("Nome do contato: "); setbuf(stdin, NULL);
130     gets(busca);
131     int n;
132     Elemento* resposta = busca_contato(l, busca, &n);
133     if (resposta == NULL) {
134         printf("\nContato nao encontrado\n");
135     }else{
136         printf("\nO contato esta na posicao %d da agenda\n", n);
137         printf("%d - Nome: %s - Telefone: %s - Aniversario: %s\n", n,
            ↪  resposta->info.nome, resposta->info.telefone,
            ↪  resposta->info.aniversario);
138     }
139     return resposta;
140 }
141
142 void removerContato(Lista *l){
```

```
143     if(l->qtd == 0){
144         printf("\nVoce ainda nao inseriu contatos na agenda\n");
145         return;
146     }
147     Elemento *e = buscarContato(l);
148     if(e == NULL) return;
149     int resposta = -1;
150     printf("\n------------------- REMOVER CONTATO -------------------\n");
151     do{
152         printf("\nDeseja realmente remover o contato? [1 - SIM / 0 - NAO]: ");
153         scanf("%d",&resposta);
154         if(resposta != 1 && resposta != 0) printf("\nOPCAO INVALIDA!\n");
155     }while(resposta != 1 && resposta != 0);
156     if (resposta == 0) printf("O contato nao foi removido\n");
157     else{
158         remove_contato(l, e);
159         printf("O contato foi removido\n");
160     }
161 }
162 
163 void editarContato(Lista *l){
164     if(l->qtd == 0){
165         printf("\nVoce ainda nao inseriu contatos na agenda\n");
166         return;
167     }
168     Elemento *e = buscarContato(l);
169     if(e == NULL) return;
170     int resposta = -1;
171     printf("\n------------------- EDITAR CONTATO -------------------\n");
172     remove_contato(l, e);
173     Contato contato;
174     char nome[30], telefone[30], aniversario[30];
175     printf("Nome: "); setbuf(stdin, NULL);
176     gets(contato.nome);
177     printf("Telefone: "); setbuf(stdin, NULL);
178     gets(contato.telefone);
179     printf("Data de aniversario: "); setbuf(stdin, NULL);
180     gets(contato.aniversario);
181     insere_contato(l, contato);
182     printf("\nContato editado com sucesso!\n");
183 }
184 
185 void menu(){
186     printf("\n------------------- MENU -------------------\n");
187     printf("1 - Inserir contato\n");
188     printf("2 - Listar contatos\n");
189     printf("3 - Buscar contatos\n");
190     printf("4 - Editar contato\n");
191     printf("5 - Remover contato\n");
192     printf("6 - Remover contatos duplicados\n");
193     printf("7 - Sair do programa\n: ");
194 }
195 
196 int main(){
197     Lista* l = (Lista*) malloc(sizeof(Lista));
198     cria_agenda(l);
199     int resposta;
200     do{
201         menu();
```

```
202         scanf("%d", &resposta);
203         switch (resposta){
204             case 1: inserirContato(l); break;
205             case 2: lista_contatos(l); break;
206             case 3: buscarContato(l); break;
207             case 4: editarContato(l); break;
208             case 5: removerContato(l); break;
209             case 6: remove_duplicados(l); break;
210             case 7: printf("\nSaindo do programa..."); break;
211             default: printf("\nOPCAO INVALIDA!\n"); break;
212         }
213     }while(resposta != 7);
214     return 0;
215 }
```