INF1007 - PROGRAMAÇÃO II

LISTA DE EXERCÍCIOS 9

1. Em uma agenda telefônica os contatos são cadastrados com os seguintes dados:

   - Nome – nome do contato (máximo 40 caracteres);
   - Telefone – cadeia de caracteres com o número do telefone do contato (máximo 15 caracteres);
   - Celular – cadeia de caracteres com o número do celular do contato (máximo 15 caracteres);
   - Email – cadeia de caracteres com o email do contato (máximo 40 caracteres);
   - DataAniversario – data de aniversario do contato (contendo dia e mês);

   Essas informações podem ser representadas em dois tipos estruturados: `Data` e `Contato`.

Data
Dia
Mês

Contato
Nome
Telefone
Celular
Email
DataAniversario

   Utilizando listas encadeadas, escreva um programa que permita o cadastro, edição, 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;

- `atualiza_contato` – modifica os dados de um contato já 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 alterar os dados do contato encontrado com base nas novas informações fornecidas pelo usuário. Se o contato buscado não for encontrado, o programa deve exibir uma mensagem informando o usuário sobre esse fato;

- `remove_duplicados` – remove todos os contatos duplicados existentes na lista. Um contato é considerado duplicado se existir outro contato na agenda com o mesmo nome;

- `libera_agenda` – libera a memória alocada dinamicamente para armazenar os contatos da agenda;

Em seguida, você deve implementar a função principal do programa que permita ao usuário realizar todas as operações da agenda: cadastro, impressão, edição, remoção, busca e remover duplicados. 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
```

O programa deve permitir que o usuário realize operações na agenda de contatos até o momento que ele desejar sair do programa escolhendo a opções 7.

2. O sistema de uma editora utiliza uma lista simplesmente encadeada para armazenar os dados dos seus livros. Está lista é automaticamente ordenada crescentemente pelos anos dos livros e cada nó da lista é representado pelo tipo estruturado `NoDaLista`:

```
typedef struct noDaLista NoDaLista;

struct noDaLista
{
    char titulo[51];
    char autor[51];
    int ano;
    int quantidade;
    NoDaLista *prox;
};
```

Com base nessa definição, implemente as seguintes funções:

- `quantidade_livros` – a função recebe por parâmetro o endereço do primeiro elemento da lista encadeada e deve retornar a quantidade total de livros em estoque (somatório dos campos `quantidade` existentes na estrutura dos nós da lista);

- `livros_ano` – a função recebe por parâmetro o endereço do primeiro elemento da lista encadeada e um determinado ano. A função deve exibir na tela os dados de todos os livros existentes na lista que foram publicados no ano especificado;

- `livros_ano_rec` – a função recebe por parâmetro o endereço do primeiro elemento da lista encadeada e um determinado ano. A função deve exibir na tela os dados de todos os livros existentes na lista que foram publicados no ano especificado. Obrigatoriamente a função deve ser implementada de forma recursiva;

- `separa_ano` – a função recebe a lista encadeada (ou seja, o endereço do primeiro nó da lista) e um ano, e MODIFICANDO a lista original, retorna uma (sub)lista encadeada somente com os livros daquele ano. Não devem ser criados novos nós. O trecho correspondente aos livros desse ano deve ser "cortado" da lista original. A lista original deve ser "emendada", e o endereço do primeiro nó do trecho (sublista) arrancado deve ser retornado (sendo necessário colocar NULL no campo prox do último nó dessa (sub)lista). A sua função deve levar em conta a ordenação da lista original. A função retorna o endereço do primeiro nó da (sub) lista resultante. Para facilitar, considere que o ano solicitado nunca é o do primeiro livro ou do último livro. Se não existir nenhum livro para o ano recebido, a função retorna NULL.

Em seguida, continue a implementação do programa baixo para testar as suas funções. O seu programa deve exibir na tela o total de livros em estoque, todos os livros de um determinado ano fornecido pelo usuário (usando a função iterativa e a recursiva) e também as listas resultantes ao realizar a separação de um determinado ano.

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

typedef struct noDaLista NoDaLista;
struct noDaLista
{
   char titulo[51];
   char autor[51];
   int ano;
   int quantidade;
   NoDaLista *prox;
};

NoDaLista* insere_livro(NoDaLista* lst, char *nome, char *autor, int ano,int qtd)
{
   NoDaLista* p = (NoDaLista*)malloc(sizeof(NoDaLista));
   strcpy(p->titulo, nome);
   strcpy(p->autor, autor);
   p->ano = ano;
   p->quantidade = qtd;
   p->prox = lst;
   return p;
}

int main (void)
{
   NoDaLista *lista = NULL;
   lista = insere_livro(lista, "Game Development: Using Unity and C#",
   "Millington", 2013, 43);
   lista = insere_livro(lista, "Game Coding Complete", "McShaffry", 2012, 32);
   lista = insere_livro(lista, "Game Development Essentials: An Introduction",
   "Novak", 2011, 42);
   lista = insere_livro(lista, "Prolog Programming for Artificial Intelligence",
   "Bratko", 2011, 31);
   lista = insere_livro(lista, "Beginning C++ Through Game Programming",
   "Dawson", 2010, 54);
   lista = insere_livro(lista, "Fundamentals of Database Systems", "Elmasri",
   2010, 46);
   lista = insere_livro(lista, "Software Engineering: Theory and Practice",
   "Pfleeger", 2009, 28);
   lista = insere_livro(lista, "Introduction to Algorithms", "Cormen", 2009, 31);
   lista = insere_livro(lista, "Artificial Intelligence: A Modern Approach",
   "Russell and Norvig", 2009, 45);
   lista = insere_livro(lista, "Artificial Intelligence for Games", "Millington",
   2009, 29);
   lista = insere_livro(lista, "Pattern Recognition and Machine Learning",
   "Bishop", 2006, 13);
   lista = insere_livro(lista, "Algorithms", "Dasgupta", 2006, 25);
   lista = insere_livro(lista, "Algorithm Design", "Kleinberg", 2006, 38);
   lista = insere_livro(lista, "Introducao a Estruturas de Dados", "Waldemar
   Celes", 2004, 31);
   lista = insere_livro(lista, "C - A Linguagem de Programacao", "Kernighan",
   2000, 21);
   lista = insere_livro(lista, "Machine Learning", "Mitchell", 1997, 42);


   return 0;
}
```