```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include "estruturas.h"

/*****************************************/
/*Samuel Anjos nº 13305                  */
/*Luis Camilo nº13987                   */
/*****************************************/


/*
*   adiciona o novo cliente
*/
struct listaClientes * adicionarCliente(struct listaClientes * head){

    struct listaClientes *current;
    current = (struct listaClientes*) malloc(sizeof(struct listaClientes));

    /* verificar se é o primeiro node, se for criar o funcionario */
    if (head == NULL)
        head = menuCriaNovoCliente();

    else{
        /* ir até ao ultimo node */
        current = head;
        while (current->next != NULL)
            current = current->next;

        /* criar novo node */

        current->next = menuCriaNovoCliente();
    }

    return head;

}// fim função

/*
*   imprime a lista de clientes
*/
struct listaClientes * imprimeClientes(struct listaClientes * head){

    struct listaClientes * current;
    system("cls");
    /* verificar se a lista tem nodes */
    if (head == NULL){
        alteraCorTexto(12);
        printf("Nao existem Clientes!\n");
        alteraCorTexto(15);
    }

    current = head;
    printf("------------------------------------------------------------------------------------------------------------ ↙
    ------------------------------------------------------------");
    printf("%6s", "NIF");
    printf("%4s", " ");
    printf("%11s", "DATA NASCI");
    printf("%20s", "NOME PROPRIO");
    printf("%19s", "APELIDO");
    printf("%17s", "TELEFONE");
    printf("%21s", "MORADA");
    printf("%27s", "LOCALIDADE");
    printf("%19s", "COD POSTAL\n");
    printf("------------------------------------------------------------------------------------------------------------ ↙
    ------------------------------------------------------------\n");
    while (current != NULL){
        /* imprime node */
        printCliente(current);
        alteraCorTexto(8);
        printf("------------------------------------------------------------------------------------------------------------ ↙
    ------------------------------------------------------------\n");
```

```
        alteraCorTexto(15);
        /* proximo node */
        current = current->next;
    }
    //printf("\n");
    printf("------------------------------------------------------------------------------------------------------ ↙
    ------------------------------------------------------------\n");
    system("pause");

    return head;

}// fim função
/*
*   imprime no ecra um node
*/
void printCliente(struct listaClientes * current){

    printf("%-11d", current->cliente.dadosClt.nif);
    printf("%d/%d/%d\t", current->cliente.dataNascimento.dia,
        current->cliente.dataNascimento.mes,
        current->cliente.dataNascimento.ano);
    printf(" %20s", current->cliente.dadosClt.nomeProprio);
    printf(" %20s", current->cliente.dadosClt.apelido);
    printf(" %11d", current->cliente.dadosClt.telefone);
    printf(" %30s", current->cliente.moradaClt.rua);
    printf(" %20s", current->cliente.moradaClt.localidade);
    printf(" %12s", current->cliente.moradaClt.codigoPostal);

    printf("\n");

}// fim função


/*
*   grava funcionarios no ficheiro binario
*/
void escreveClientes(struct listaClientes *head){
    FILE *fp;
    struct listaClientes *current;
    /* chama a função para abri o ficheiro */
    fp = abreFicheiroBinario("clientes.dat", "wb");

    current = head;
    /* escreve no ficheiro um node de cada vez da lista */
    while (current != NULL){
        fwrite(current, sizeof(struct listaClientes), 1, fp);
        current = current->next;
    }

    fclose(fp);

}// fim função

/*
*   função para liebertar a lista de memoria
*/
void libertaListaClientes(struct listaClientes *head){
    struct listaClientes *temp, *current = head;

    while (current != NULL){
        temp = current->next;
        free(current);  // liberta o node
        current = temp;
    }
}// fim função


/*
*   carregamento de clientes do ficheiro binarios
*/
struct listaClientes * carregaClientes(struct listaClientes * head){

    FILE *fp;
    struct listaClientes  *temp;
```

```
    struct listaClientes *current;

    fp = abreFicheiroBinario("clientes.dat", "rb");

    /* percorre o ficheio até ao fim */
    while (!feof(fp)){
        //current = (struct funcionario *) malloc (sizeof (struct funcionario));
        if ((current = (struct listaClientes *) malloc(sizeof(struct listaClientes))) == NULL){
            printf("Erro: reservad de memoria carrega clientes.");
            exit(1);
        }
        /* devolve 0 qunado não tem sucesso */
        if (fread(current, sizeof(struct listaClientes), 1, fp)>0){

            current->next = NULL;
            temp = head;

            /* preencher o primeiro node */
            if (head == NULL){
                head = current;
            }
            /* preencher o resto dos nodes */
            else{

                while (temp->next != NULL)
                    temp = temp->next;

                temp->next = current;
            }

        }// fim if
    } // fim while

    fclose(fp);

    return head;
}// fim função

void editarCliente(struct listaClientes *head){
    /* campo pelo qual pretendemos pesquisa a lista ligada */
    int nif;
    /* struct que pretendemos para  editar */
    struct listaClientes * clientePretendido;
    /* perguntar ao utilizador qual o nif do funcionario */
    printf("%61s", " "); printf("Introduza o nif do Cliente:");
    scanf("%d", &nif);
    fflush(stdin);

    /* cliente prentendido para editar*/
    clientePretendido = procuraNodeCliente(head, &nif);
    /* cliente existe */
    if (clientePretendido != NULL)
        /* editar cliente*/
        menuEditarCliente(clientePretendido);

    else{
        printf("%61s", " ");
        alteraCorTexto(12); printf("Cliente nao existe.\n"); alteraCorTexto(15);
    }
    system("pause");

}// fim função

/*
* função que procura os clientes por nif
*/
struct listaClientes * procuraNodeCliente(struct listaClientes * head, int *nif){

    struct listaClientes * current;

    current = head;
    /*traversa lista*/
    while (current != NULL){
        if (current->cliente.dadosClt.nif == *nif)
```

```
            return current;
        /*avança para o proximo node */
        current = current->next;
    }

    return NULL;


}// fim função

/*
* função para procurar os clientes por apelido
*/
struct listaClientes * procuraNodeAplelido(struct listaClientes * head, char * apelido ){

    struct listaClientes * current;

    current = head;
    /*traversa lista*/
    while (current != NULL){
        if ( ! _stricmp (current->cliente.dadosClt.apelido, apelido))
            return current;
        /*avança para o proximo node */
        current = current->next;
    }

    return NULL;


}// fim função



/*
* função utilizada para pergutar qual o cliente para removes
*/

struct listaClientes * apagaCliente(struct listaClientes *head){
    /* campo pelo qual pretendemos pesquisa a lista ligada */
    int nif;
    /* struct que pretendemos para  editar */
    struct listaClientes * clientePretendido;
    /* perguntar ao utilizador qual o nif do funcionario */
    printf("%61s", " "); printf("Introduza o nif do cliente:");
    scanf("%d", &nif);
    fflush(stdin);

    /* funcionario pretendido para apagar */
    clientePretendido = procuraNodeCliente(head, &nif);
    /* funcionario existe*/
    if (clientePretendido != NULL){
        /*apagar funcionario da lista */
        head = removerCliente(head, clientePretendido);

        printf("%61s", " "); alteraCorTexto(10); printf("O cliente foi apagado com sucesso!\n");
    alteraCorTexto(15);

    }/* funcionario não exite*/
    else{
        printf("%61s", " ");
        alteraCorTexto(12); printf("Esse funcionario nao exist.\n"); alteraCorTexto(15);
    }

    printf("%61s", " "); system("pause");

    return head;
}//fim função

/*
* função para remover o node no meio da lista
*/
struct listaClientes * removerCliente(struct listaClientes *head, struct listaClientes *current){
    struct listaClientes * temp = head;
```

```
    if (current == NULL)
        return NULL;

    /*se o node é o primeiro */
    if (current == head){
        //return head;
        return deleteHeadClientes(head);
    }

    /*se o node é o ultimo */
    if (current->next == NULL)
        return deleteUltimoClientes(head);

    /*se o node é no meio da lista ligada */
    while (temp != NULL){
        if (temp->next == current)
            break;
        temp = temp->next;
    }

    struct listaClientes * temp2 = temp->next;
    temp->next = temp2->next;
    temp2->next = NULL;
    free(temp2);

    return head;

}//fim função

/*
* função para remover o primeiro node da lista ligada de clientes
*/
struct listaClientes* deleteHeadClientes(struct listaClientes*head){
    struct listaClientes * primeiro = head;

    //struct funcionario * primeiro = head;
    head = head->next;

    /* confirmar que exite mais que um node */
    if (primeiro == head)
        head = NULL;

    primeiro = NULL;
    free(primeiro);

    return head;
}//fim função

/*
*   remover o ultimo node da lista ligada de clientes
*/
struct listaClientes * deleteUltimoClientes(struct listaClientes * head){
    struct listaClientes * current = head;
    struct listaClientes * ultimo = NULL;

    /* transversar a lista até ao penultimo */
    while (current->next != NULL){
        ultimo = current;
        current = current->next;
    }

    if (ultimo != NULL)
        ultimo->next = NULL;

    free(current);

    return head;

}// fim função
```