ECM-306 -Tópicos Avançados em Estrutura de Dados

Prof. Dr. Aparecido V. de Freitas – Tarefa 07

Guilherme de Campos	RA: 20.00089-8
Leonardo Campos da Costa	RA: 20.00786-8
Luis Guilherme de Souza Munhoz	RA: 20.01937-8
Enrico Giannobile	RA: 19.00610-0

lista.h

```c
#ifndef LISTA_H_DECLARED
#define LISTA_H_DECLARED

#include <stdio.h>


typedef struct
{
    int next[10];   // vetor de próximos índices
    int key[10];    // vetor de chaves (valores)
    int prev[10];   // vetor de últimos índices
    int free[10];   // vetor dos índices livres
    int L;          // índice do início da lista ligada


} Lists;

Lists createLinkedList();
Lists createLinkedListKey();
void showLists();
void showLinkedList();
void removeKey();
void emptyList();
void fillFree();
void insertKey();
void showArray();
void insertOnFree();

#endif
```

lista.c

```c
#include <stdio.h>
#include "lista.h"

Lists s1;

Lists createLinkedList(int n[10], int k[10], int p[10], int L) {

    Lists l1;
    for(int i=0 ; i<10; i++){

        l1.next[i] = n[i];
        l1.key[i] = k[i];
        l1.prev[i] = p[i];
    }
    return l1;
}

void emptyList() {

    for(int i = 0 ; i<10 ; i++) {

        removeKey(s1.key[i]);
        showLinkedList();
    }
}

Lists createLinkedListKey(int k[10]) {

    Lists l1;

    for(int i=0 ; i<10; i++) {

        l1.next[i] = i+1;
        l1.key[i] = k[i];
        l1.prev[i] = i-1;
    }
    l1.next[9] = -1;
    l1.L = 0;
    return l1;
}

void showLists()  // Imprime os vetores do struct Lists
{
    showArray(s1.next,"Next");
    showArray(s1.key,"Key");
    showArray(s1.prev,"Prev");
    showArray(s1.free,"Free");
    printf("\nL        -->%4d",s1.L);

}

void showLinkedList() { // Imprime a lista ligada

    printf("\n\n");
    printf("List     -->");

    int nextIndex = s1.L;

    while(nextIndex != -1) {

        printf("%4d ",s1.key[nextIndex]);
        nextIndex = s1.next[nextIndex];
    }
    printf("\n\n");
}

void insertKey(int key) { // Insere uma chave no final da lista ligada

    for(int i=0; i<10; i++) {

        if(s1.free[i]!= -1) {

            s1.key[s1.free[i]] = key;

            int NextIndex = s1.L;

            while(s1.next[NextIndex] != -1) {
                NextIndex = s1.next[NextIndex];
            }

            s1.next[NextIndex] = s1.free[i];
            s1.prev[s1.free[i]] = NextIndex;

            // Verifica se a lista estava totalmente vazia ao inserir a
chave
            if(s1.L==-1) {

                s1.next[s1.free[i]] = -1;
                s1.prev[s1.free[i]] = -1;
                s1.L = s1.free[i];
            }

            s1.free[i] = -1;

            return;
        }
    }
    printf("\nLista cheia, chave %d rejeitada!\n", key);
}

void showArray(int array[], char nome[]) {

    printf("\n");
    printf("%-8s -->",nome);

    for(int i=0 ; i<10; i++) {

        printf("%4d ",array[i]);
    }

    printf("\n");
}

void fillFree() { // Preenche o vetor free com -1

    for(int i = 0; i<10; i++) {

        s1.free[i] = -1;
    }

}

void insertOnFree(int index) {

    for(int i=0; i<10; i++) {

        if(s1.free[i] == -1) {

            s1.free[i] = index;

            return;
        }
    }

}

void removeKey(int target) {    // Remove uma chave da lista ligada


    int targetIndex = s1.L;

    // Percorre a lista até encontrar o alvo ou até o fim
    while(s1.key[targetIndex] != target && targetIndex != -1) {
        targetIndex = s1.next[targetIndex];
    }
    // Verifica se encontrou o alvo

    if(s1.key[targetIndex] != target) {
        printf("chave %i nao encontrada\n", target);
        return;
    }

    s1.key[targetIndex] = -1;

    // Duas variaveis para facilitar leitura
    int targetNextIndex = s1.next[targetIndex];
    int targetPrevIndex = s1.prev[targetIndex];



    // Verifica se o alvo é o primeiro elemento da lista
    if(s1.prev[targetIndex] == -1) {

        s1.L = targetNextIndex;
        s1.prev[targetNextIndex] = -1;

    }
    // Verifica se o alvo é o ultimo elemento da lista
    else if(s1.next[targetIndex] == -1) {

        s1.next[targetPrevIndex] = -1;

    }
    // Altera os indices do ultimo e proximo elemento
    else {

        s1.next[targetPrevIndex] = targetNextIndex;
        s1.prev[targetNextIndex] = targetPrevIndex;
    }

    s1.next[targetIndex] = -1;
    s1.prev[targetIndex] = -1;

    insertOnFree(targetIndex);

    return;
}
```

main.c

```c
#include <stdio.h>
#include <stdlib.h>
#include "lista.h"

Lists s1;
int ntest[10] = {1,2,3,4,5,6,7,8,9,-1};
int ktest[10] = {5,3,1,54,23,1,2,4,56,10};
int ptest[10] = {-1,0,1,2,3,4,5,6,7,8};
int Ltest = 0;

int main(void) {

    s1 = createLinkedListKey(ktest);

    fillFree();

    showLists();
    showLinkedList();

    emptyList();
    removeKey(5);

    showLists();
    showLinkedList();

    return 0;

}
```