Nome: João Paulo de Oliveira 11611BCC046

3° Aula prática

Uberlândia

2016

1.Código fonte:

1.1 – main.c:

#include <stdio.h>

#include <stdlib.h>

#include "lista.h"

#include <sys/time.h>

int main(int argc, char \*argv[]){

struct timeval t;

TipoLista lista1, lista2, lista3;

TipoItem item;

int vetor[MAX];

TipoApontador p,r;

int i, j, k, n, posicao;

float tamanho1 = 0, tamanho2 = 0;

gettimeofday(&t,NULL);

srand((unsigned int)t.tv\_usec);

FLVazia(&lista1);

FLVazia(&lista2);

FLVazia(&lista3);

printf("Inserindo elementos na Lista 1:\n");

/\*Gera uma permutacao aleatoria de chaves entre 1 e MAX\*/

for(i = 0; i < MAX; i++) vetor[i] = i + 1;

for(i = 0; i < MAX; i++){

k = (int) (10.0 \* rand()/(RAND\_MAX + 1.0));

j = (int) (10.0 \* rand()/(RAND\_MAX + 1.0));

n = vetor[k];

vetor[k] = vetor[j];

vetor[j] = n;

}

/\*Insere cada chave na lista \*/

for (i = 0; i < MAX; i++){

item.Chave = vetor[i];

Insere(item, &lista1);

tamanho1++;

printf("Inseriu: %d \n", item.Chave);

}

printf("\nLista 1:\n");

Imprime(lista1);

printf("Inserindo elementos na Lista 2:\n");

for(i = 0; i < MAX; i++) vetor[i] = i + 1;

for(i = 0; i < MAX; i++){

k = (int) (10.0 \* rand()/(RAND\_MAX + 1.0));

j = (int) (10.0 \* rand()/(RAND\_MAX + 1.0));

n = vetor[k];

vetor[k] = vetor[j];

vetor[j] = n;

}

/\*Insere cada chave na lista \*/

for (i = 0; i < MAX; i++){

item.Chave = vetor[i] + 10;

Insere(item, &lista2);

tamanho2++;

printf("Inseriu: %d \n", item.Chave);

}

printf("\nLista 02:\n");

Imprime(lista2);

printf("Situacao da lista 1:");

Imprime(lista1);

printf("Situacao da lista 2:");

Imprime(lista2);

r = lista1.Primeiro;

printf("Retirada da Lista 01:\n");

Retira(r, &lista1, &item);

printf("\n Retirou: %d\n", item.Chave);

printf("Situacao atual da lista 1:\n");

Imprime(lista1);

do

{

printf("\nDigite a posicao:");

scanf("%d", &posicao);

}

while((posicao < 0)||(posicao > 9));

InsereItem(posicao, item, &lista1);

printf("Elemento inserido:\n");

Imprime(lista1);

conc\_Lis(&lista1, &lista2, &lista3);

printf("\nLista Concatenada:\n");

Imprime(lista3);

printf("\nElementos retirados da Lista 01:\n");

for(i = 0; i < MAX; i++)

{

k = (int) ((tamanho1) \* rand() / (RAND\_MAX + 1.0));

p = lista1.Primeiro;

Retira(p, &lista1, &item);

tamanho1--;

printf("Retirou: %d\n", item.Chave);

}

printf("Lista 1:\n");

Imprime(lista1);

printf("Lista 2:\n");

Imprime(lista2);

printf("Lista Concatenada:\n");

Imprime(lista3);

return(0);

}}

1.2 – lista.h:

#ifndef LISTA\_H\_INCLUDED

#define LISTA\_H\_INCLUDED

#define MAX 10

typedef int TipoChave;

typedef struct

{

int Chave;

} TipoItem;

typedef struct TipoCelula \*TipoApontador;

typedef struct TipoCelula

{

TipoItem Item;

TipoApontador Prox;

} TipoCelula;

typedef struct

{

TipoApontador Primeiro, Ultimo;

} TipoLista;

void FLVazia(TipoLista \*Lista);

int Vazia(TipoLista Lista);

void Insere(TipoItem x, TipoLista \*Lista);

void Retira(TipoApontador p, TipoLista \*Lista, TipoItem \*Item);

void Imprime(TipoLista Lista);

void InsereItem(int p, TipoItem x, TipoLista \*Lista);

void conc\_List(TipoLista \*Lista1, TipoLista \*Lista2, TipoLista \*Lista3);

#endif // LISTA\_H\_INCLUDED

1.3 – lista.c:

#include <stdio.h>

#include <stdio.h>

#include <stdlib.h>

#include <sys/time.h>

#include "lista.h"

void FLVazia(TipoLista \*Lista)

{

Lista -> Primeiro = (TipoApontador) malloc(sizeof(TipoCelula));

Lista -> Ultimo = Lista -> Primeiro;

Lista -> Primeiro -> Prox = NULL;

}

int Vazia(TipoLista Lista)

{

return (Lista.Primeiro == Lista.Ultimo);

}

void Insere(TipoItem x, TipoLista \*Lista)

{

Lista -> Ultimo -> Prox = (TipoApontador) malloc(sizeof(TipoCelula));

Lista -> Ultimo = Lista -> Ultimo -> Prox;

Lista -> Ultimo -> Item = x;

Lista -> Ultimo -> Prox = NULL;

}

void Retira(TipoApontador p, TipoLista \*Lista, TipoItem \*Item)

{

/\* --- Obs.: o item a ser retirado e o seguinte ao apontado por p --- \*/

TipoApontador q;

if (Vazia(\*Lista) || p == NULL || p -> Prox == NULL)

{

printf(" Erro - Lista vazia ou posicao nao existe\n");

return;

}

q = p -> Prox;

\*Item = q -> Item;

p -> Prox = q -> Prox;

if (p -> Prox == NULL) Lista -> Ultimo = p;

free(q);

}

void Imprime(TipoLista Lista)

{

int i;

TipoApontador Aux;

Aux = Lista.Primeiro -> Prox;

i = Vazia(Lista);

if(i == 1)

{

printf("(Lista Vazia)\n");

}

else

{

while (Aux != NULL)

{

printf(" / %d", Aux -> Item.Chave);

Aux = Aux -> Prox;

}

printf (" /");

printf("\n\n");

}

}

void InsereItem(int posicao, TipoItem x, TipoLista \*Lista)

{

int i = 1;

TipoCelula \*elemento, \*aux = Lista->Primeiro;

elemento = (TipoCelula\*) malloc(sizeof(TipoCelula));

elemento->Item = x;

while(aux->Prox != NULL && i <= posicao)

{

aux = aux->Prox;

i++;

}

elemento->Prox = aux->Prox;

aux->Prox = elemento;

}

void conc\_Lis(TipoLista \*Lista1, TipoLista \*Lista2, TipoLista \*Lista3)

{

TipoCelula \*aux = Lista1->Primeiro;

while(aux->Prox != NULL)

{

aux = aux->Prox;

Insere(aux->Item, Lista3);

}

aux = Lista2->Primeiro;

while(aux->Prox != NULL)

{

aux = aux->Prox;

Insere(aux->Item, Lista3);

}

}

2.Print do funcionamento: 