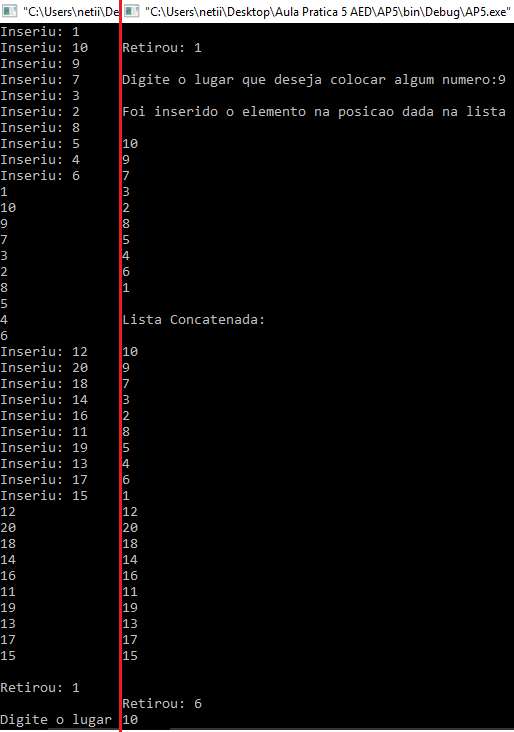
UFU

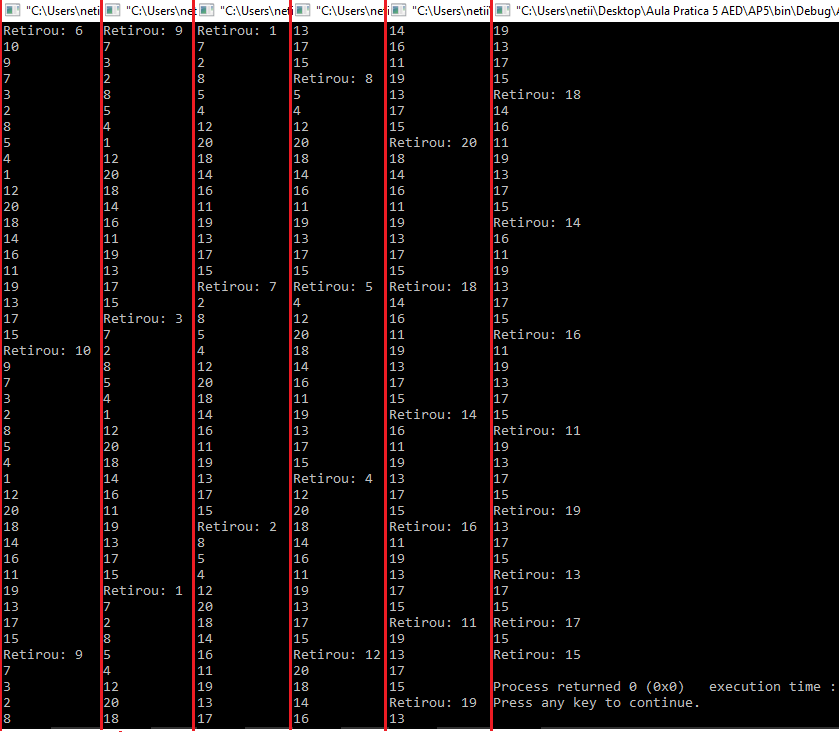
Nome: Antonio Carlos Neto

Curso: Ciência da Computação

Disciplina: GBC024

EXECUÇÃO:





Main.c:

#include <sys/time.h>

#include <stdlib.h>

#include <stdio.h>

#include "apontadores.h"

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

{

struct timeval t;

TipoLista lista, lista2, lista3;

int vetor[MAX];

TipoApontador p, r;

int i, j, k, n, item, lugar, w = 0;

float tam = 0, tam2 = 0, tam3 = 0;

gettimeofday(&t,NULL);

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

FLVazia(&lista);

FLVazia(&lista2);

FLVazia(&lista3);

TipoCelula \*aux = lista2.Primeiro;

/\*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 = vetor[i];

Insere(item, &lista);

tam++;

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

}

Imprime(lista);

/\*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 = vetor[i] + MAX;

Insere(item, &lista2);

tam2++;

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

}

Imprime(lista2);

r = lista.Primeiro;

Retira(r, &lista, &item);

printf("\nRetirou: %d\n", item);

do

{

printf("\nDigite o lugar que deseja colocar algum numero:");

scanf("%d", &lugar);

}

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

Insercao(lugar, item, &lista);

printf("\nFoi inserido o elemento na posicao dada na lista\n\n");

Imprime(lista);

Concatena(&lista, &lista2, &lista3);

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

Imprime(lista3);

printf("\n\n");

/\*Retira cada chave da lista \*/

tam3 = 20;

for(i = 0; i < (MAX\*2); i++)

{

/\*escolhe uma chave aleatoriamente \*/

k = (int) ((tam3) \* rand() / (RAND\_MAX + 1.0));

//printf("\n\n%d",k);

//system("PAUSE");

aux = lista3.Primeiro;

while(aux->Prox != NULL && w <= k)

{

aux = aux->Prox;

w++;

}

p = aux;

/\*retira chave apontada \*/

Retira(aux, &lista3, &item);

tam3--;

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

Imprime(lista3);

}

Imprime (lista3);

return(0);

}

apontadores.c:

#include <sys/time.h>

#include <stdlib.h>

#include <stdio.h>

#include "apontadores.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(int 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, int \*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)

{

TipoApontador Aux;

Aux = Lista.Primeiro -> Prox;

while (Aux != NULL)

{

printf("%d\n", Aux -> Item);

Aux = Aux -> Prox;

}

}

void Insercao(int posicao, int x, TipoLista \*Lista)

{

int i = 1;

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

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

elem->Item = x;

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

{

aux = aux->Prox;

i++;

}

elem->Prox = aux->Prox;

aux->Prox = elem;

}

void Concatena(TipoLista \*Lista, TipoLista \*Lista2, TipoLista \*Lista3)

{

TipoCelula \*aux = Lista->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);

}

}

apontadores.h:

#ifndef APONTADORES\_H\_INCLUDED

#define APONTADORES\_H\_INCLUDED

#define MAX 10

typedef struct TipoCelula \*TipoApontador;

typedef struct TipoCelula {

int Item;

TipoApontador Prox;

} TipoCelula;

typedef struct {

TipoApontador Primeiro, Ultimo;

} TipoLista;

void FLVazia(TipoLista \*Lista);

int Vazia(TipoLista Lista);

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

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

void Imprime(TipoLista Lista);

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

void Concatena(TipoLista \*Lista, TipoLista \*Lista2, TipoLista \*Lista3);

#endif // APONTADORES\_H\_INCLUDED