//------- Inclusoes

#include <stdio.h>

#include <stdlib.h>

#include <stdint.h>

#include <string.h>

//-------- Fim Inclusoes

//-------- Constantes

#define SUCESSO (0)

#define TRUE (1==1)

#define FALSE (1!=1)

#define max2(a,b) ((a >b)? a : b)

//-------- Fim Constantes

//-------- Dado Armazenado Pilha

#define TAM\_MAX (50 + 1) //<<<<<<<<<<<<<<<<<<<<<<

struct dados\_s{

char nome[TAM\_MAX];

char nome2[TAM\_MAX];

}; //<<<<<<<<<<<<<<<<<<<<<<

typedef struct dados\_s casal\_t;

//-------- Fim Dado Armazenado Pilha

//-------- Estrutura Pilha Lista

struct itemListaE\_s {

struct itemListaE\_s\* proxItem;

casal\_t dadoArmazenado;

};

typedef struct itemListaE\_s itemListaE\_t;

typedef itemListaE\_t itemPilha\_t;

struct pilha\_s {

uint32\_t capacidade;

uint32\_t numItens;

itemPilha\_t\* topoPilha;

};

typedef struct pilha\_s pilha\_t;

//-------- Fim Estrutura Pilha Lista

//-------- Modelagem de Funcoes de Manipulacao e tipos

typedef uint32\_t boolean\_t;

typedef void(\*imprimirDado\_f)(casal\_t dadoImprimir);

typedef void(\*atribuirDado\_f)(casal\_t\* enderecoArmazenar, casal\_t dadoArmazenar);

typedef int(\*compararDado\_f)(casal\_t dadoUm, casal\_t dadoDois);

//-------- Fim Modelagem de Funcoes de Manipulacao e tipos

//-------- Funcoes Pilha Lista

void inicializaPilha(pilha\_t\* pilha, uint32\_t capacidade) {

if (capacidade != 0) { //Se capacidade 0, nao tem porque inicializar pilha.

pilha->capacidade = capacidade;

pilha->numItens = 0;

pilha->topoPilha = NULL;

}

}

boolean\_t pilhaVazia(pilha\_t\* pilha) {

return (pilha->numItens == 0);

}

boolean\_t empilhaPilha(pilha\_t\* pilha, casal\_t dado, atribuirDado\_f metodoDeAtribuicao) {

boolean\_t conseguiuEmpilhar = FALSE;

if (pilha->capacidade != pilha->numItens) { // Verificar se capacidade maxima foi atingida para que seja inserido um novo item.

itemPilha\_t\* novo = malloc(sizeof(itemPilha\_t));

if (novo != NULL) { // Confirma se existe espaco alocado em novo para que possa inserir o dado e posiciona-lo no topo da pilha.

metodoDeAtribuicao(&novo->dadoArmazenado, dado);

novo->proxItem = pilha->topoPilha;

pilha->numItens++;

pilha->topoPilha = novo;

conseguiuEmpilhar = TRUE;

}

}

return conseguiuEmpilhar;

}

boolean\_t desempilhaPilha(pilha\_t\* pilha, casal\_t\* enderecoRetornoDado, atribuirDado\_f metodoDeAtribuicao) {

boolean\_t conseguiuDesempilhar = FALSE;

if (!pilhaVazia(pilha)) { //Confirma se tem elementos na pilha.

//Armazena endereco do topo da pilha e proximo item.

itemPilha\_t\* atualTopo = pilha->topoPilha;

itemPilha\_t\* temp = atualTopo->proxItem;

metodoDeAtribuicao(enderecoRetornoDado, atualTopo->dadoArmazenado); // Salva o dado da pilha.

free(atualTopo); // Libera topo da pilha.

pilha->topoPilha = temp; //Posiciona topo da pilha no elemento seguinte.

pilha->numItens--;

conseguiuDesempilhar = TRUE;

}

return conseguiuDesempilhar;

}

void liberaPilha(pilha\_t\* pilha) {

itemPilha\_t\* atualTopo = pilha->topoPilha;

while (atualTopo != NULL) {

itemPilha\_t\* tmp = atualTopo->proxItem;

free(atualTopo);

atualTopo = tmp;

}

}

void imprimePilha(pilha\_t\* pilha, imprimirDado\_f metodoDeImpressao) {

itemPilha\_t\* atual = pilha->topoPilha;

while (atual != NULL) {

metodoDeImpressao(atual->dadoArmazenado);

atual = atual->proxItem;

}

}

//-------- Fim Funcoes Pilha Lista

//-------- Funcoes Criadas para ensinar a armazenar dado, comparar e imprimir. <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

void atribui(casal\_t \*enderecoArmazenar, casal\_t dado){

strcpy(enderecoArmazenar->nome, dado.nome);

strcpy(enderecoArmazenar->nome2, dado.nome2);

}

void imprime(casal\_t dado){

printf("Nome: %s\nNome2: %s\n\n", dado.nome, dado.nome2);

}

////-------- Funcoes Criadas para ensinar a armazenar dado, comparar e imprimir.

//-------- Main

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

pilha\_t pilha;

casal\_t endereco;

int i;

casal\_t dados[] = {{{"pessoa1"},{"pessoa2"}},{{"pessoa3"},{"pessoa4"}},{{"pessoa5"},{"pessoa6"}},{{"pessoa7"},{"pessoa8"}},{{"pessoa9"},{"pessoa10"}},{{"pessoa11"},{"pessoa12"}},{{"pessoa13"},{"pessoa14"}},{{"pessoa15"},{"pessoa16"}},{{"pessoa17"},{"pessoa18"}},{{"pessoa19"},{"pessoa20"}},{{"pessoa21"},{"pessoa22"}},{{"pessoa23"},{"pessoa24"}},{{"pessoa25"},{"pessoa26"}},{{"pessoa27"},{"pessoa28"}},{{"pessoa29"},{"pessoa30"}},{{"pessoa31"},{"pessoa32"}},{{"pessoa33"},{"pessoa34"}},{{"pessoa35"},{"pessoa36"}},{{"pessoa37"},{"pessoa38"}},{{"pessoa39"},{"pessoa40"}}};

inicializaPilha(&pilha, 10);

for( i = 0; i < 20; i++) {

empilhaPilha(&pilha, dados[i], atribui);

}

desempilhaPilha(&pilha, &endereco, atribui);

imprimePilha(&pilha, imprime);

return SUCESSO;

}

//-------- Fim Main