**Lista Estática Sequencial**

#define MAX 1000

typedef struct{

int chave;

char nome[50];

// outros componentes

}TipoItem;

typedef struct{

TipoItem itens[MAX];

int tamanho;

}TipoLista;

void inicializaLista(TipoLista \*lista){

lista->tamanho = 0;

}

int insereNoFinal(TipoLista \*lista, TipoItem novoItem){

int pos = lista->tamanho;

if(pos < MAX){

lista->itens[pos] = novoItem;

lista->tamanho++;

}

else return -1;

return 1;

}

void imprimeLista(TipoLista \*lista){

int i=0;

for(i=0; i < lista->tamanho; i++)

printf("%d %s¥n",lista->itens[i].chave,lista->itens[i].nome);

} int main()

{

int i,n;

TipoLista lista;

TipoItem item;

inicializaLista(&lista);

scanf("%d",&n);

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

{

scanf("%d %s",&item.chave,item.nome);

insereNoFinal(&lista,item);

}

imprimeLista(&lista);

return 0;

}

**Lista dinâmica encadeada**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

typedef struct TipoItem{

int chave;

char nome[50];

struct TipoItem \*prox; //aponta para próximo elemento

// outros componentes

}TipoItem;

typedef struct{

TipoItem \*inicio; //apontador para o início da lista

int tamanho;

}TipoLista;

void inicializaLista(TipoLista \*lista){

lista->inicio = NULL;

lista->tamanho = 0;

}

void imprimeLista(TipoLista \*lista){

TipoItem \*p = lista->inicio;

printf("List size=%d¥n",lista->tamanho);

while(p!=NULL){

printf("%d %s¥n",p->chave,p->nome);

p = p->prox;

}

void insereNoFinal(TipoLista \*lista, TipoItem novoItem){

TipoItem \*novoItemPtr;

TipoItem \*p = lista->inicio;

novoItemPtr = malloc(sizeof(TipoItem));

if(novoItemPtr == NULL) exit(1); //alocação falhou

\*novoItemPtr = novoItem;

if(p==NULL){ //Lista vazia

lista->inicio = novoItemPtr;

}

else{

while(p->prox!=NULL){

p = p->prox;

}

p->prox = novoItemPtr;

}

lista->tamanho++;

}

int removePrimeiroNome(TipoLista \*lista, char \*nome){

int encontrou=-1;

TipoItem \*p1 = lista->inicio;

TipoItem \*p2; //p2 vai apontar para antecessor de p1

p2 = NULL;

while(p1!=NULL){

if(strcmp(p1->nome, nome)==0){

encontrou = 1;

if(p2 == NULL){ //p1 é início da lista

lista->inicio = p1->prox;

}

else{

p2->prox = p1->prox; //remove item da sequência

}

free(p1); //libera memória alocada ao item removido

lista->tamanho--;

break;

}

p2 = p1;

p1 = p1->prox; //p1 e p2 passam para próximo item

}

return encontrou;

}

int main()

{

int op,i,n;

TipoLista lista;

TipoItem item;

item.prox = NULL;

inicializaLista(&lista);

scanf("%d",&n);

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

{

scanf("%d",&op);

if(op==1){

scanf("%d %s",&item.chave,item.nome);

insereNoFinal(&lista,item);

}

else if(op==2){

scanf("%s",item.nome);

if(removePrimeiroNome(&lista,item.nome) < 0){

printf("%s was not found¥n",item.nome);

}

}

}

imprimeLista(&lista);

return 0;

}

**Lista duplamente encadeada**

Implementação estática (vetor)

typedef struct TipoItem{

int info;

int esq, dir;

}TipoItem;

typedef struct{

TipoItem nos[MAX];

int inicio,fim;

}TipoLista;

Implementação dinâmica

typedef struct TipoItem{

int info;

struct TipoItem \*esq,\*dir;

}TipoItem;

typedef struct{

TipoItem \*inicio,\*fim;

int tamanho;

}TipoLista;

**Lista circular dinâmica**

typedef struct TipoItem{

int chave;

struct TipoItem \*prox;

// outros componentes

}TipoItem;

typedef struct{

TipoItem \*ultimo;

int tamanho;

}TipoLista;

void inicializaLista(TipoLista \*lista){

lista->ultimo = NULL;

lista->tamanho = 0;

}

void imprimeLista(TipoLista \*lista){

TipoItem \*p =lista->ultimo;

if(p != NULL){

p=p->prox;

}

while(p!=lista->ultimo){

printf("%d¥n",p->chave);

p=p->prox;

}

if(p!=NULL)

printf("%d¥n",p->chave); }

void insereNoFinal(TipoLista \*lista, TipoItem novoItem){

TipoItem \*novoNo;

novoNo = malloc(sizeof(TipoItem));

\*novoNo = novoItem;

if(lista->ultimo == NULL){

novoNo->prox = novoNo;

lista->ultimo = novoNo;

}

else{

novoNo->prox = lista->ultimo->prox;

lista->ultimo->prox = novoNo;

lista->ultimo=novoNo;

}

lista->tamanho++;

}

void removeProx(TipoLista \*lista, TipoItem \*no){

TipoItem \*p;

if(lista->tamanho==1){

lista->ultimo=NULL;

}

else if(no->prox == lista->ultimo){

lista->ultimo=no;

}

p = no->prox;

no->prox = p->prox;

free(p);

lista->tamanho--;

}

**FILA**

#define MAX 10

int fila[MAX],primeiro,ultimo;

void inicializar(){

primeiro=0;

ultimo=0;

}

int vazio(){

return (primeiro==ultimo);

}

int cheio(){

return ((ultimo+1)%MAX == primeiro);

}

int inserir(int v){

if(cheio())

return 0;

fila[ultimo]=v;

ultimo = (ultimo+1)%MAX;

printf("%d entrando na fila¥n",v);

return 1;

}

int remover(){

if(vazio()){

return 0;

}

printf("%d saindo da fila¥n",fila[primeiro]);

primeiro=(primeiro+1)%MAX;

return 1;

}

**PILHA**

#include <stdio.h>

#include <stdlib.h>

#define MAX 100

typedef struct {

char s[MAX];

int topo;

}tipoPilha;

void inicializar(tipoPilha \*pilha){

pilha->topo = -1; // pilha vazia

}

int push(tipoPilha \*pilha, char x){

if(pilha->topo < MAX){

pilha->topo++;

pilha->s[pilha->topo] = x;

}

else return 0;

return 1;

}

char pop(tipoPilha \*pilha){

char returnVal;

if(pilha->topo < 0){

returnVal = -1; //retorna erro de pilha vazia

}

else{

returnVal = pilha->s[pilha->topo];

pilha->topo--;

}

return returnVal;

}

int main()

{

tipoPilha pilha; char c,x; int eval=1;

inicializar(&pilha);

while((scanf("%c",&c)>0)){

if(c == '¥n') break;

if(c=='('){

if(!push(&pilha,c)){

printf("Estouro de pilha¥n");

exit(1);

}

}

else if(c==')'){

x = pop(&pilha);

if(x!='(') eval=0;

} }

if(pilha.topo>=0) eval=0; //sobrou

if(eval) printf("Correto¥n");

else printf("Incorreto¥n");

return 0;

}