Unidade de Jundiaí

Ciência da Computação



ATPS

**Disciplina: Estrutura de Dados**

Professor: Rodrigo Hentz

Amanda Cobeiros // RA: 8061796066

Cecília Junqueira Sartini // RA: 8483182138

Felipe dos Santos Barbosa // RA: 8062790577

Raone Thiago Cavalcante // RA: 129926389

Jundiaí

2015

**INDICE:**

**Relatório 2 - Listas Ligadas 3**

**Passo 2 3**

**Passo 3 6**

**Relatório 3 - Filas e Pilhas 9**

**Passo 2 9**

**Passo 3 11**

**Relatório 4 - Grafos 13**

**Passo 3: 13**

**Relatório 2 - Listas Ligadas**

**Passo 2:**

#include <stdio.h>

#include <stdlib.h>

typedef struct voo {

int numero\_do\_voo;

int data\_do\_voo;

int horario\_do\_voo;

char aeroporto\_de\_saida;

char aeroporto\_de\_chegada;

char rota;

int tempo\_estimado\_de\_voo;

int passageiros\_a\_bordo;

int info;

struct voo\* next;

} sLista, sVoo;

sVoo\* criarVoo(int valor)

{

sVoo\* p = (sVoo\*)malloc(sizeof(sVoo));

p->info = valor;

p->next = NULL;

return p;

}

sLista\* inserirInicio(sVoo\* voo, int valor)

{

sVoo\* novo = criarVoo(valor);

novo->next = voo;

return novo;

}

void imprimirLista(sLista\* lista)

{

sVoo\* p;

for (p = lista; p != NULL; p = p->next)

printf("info = %d\n", p->info);

}

sVoo\* buscarVoo(sLista\* lista, int valor)

{

sVoo\* p;

for (p=lista; p!=NULL; p=p->next)

if (p->info == valor) return p;

return NULL;

}

sLista\* excluirVoo(sLista\* lista, int valor)

{

sVoo\* ant = NULL; /\* ponteiro para elemento anterior \*/

sVoo\* p = lista; /\* ponteiro para percorrer a lista\*/

while (p != NULL && p->info != valor) { /\* procura elemento na lista, guardando anterior \*/

ant = p;

p = p->next;

}

/\* verifica se achou elemento \*/

if (p == NULL) return lista; /\* não achou: retorna lista original \*/

if (ant == NULL) {

/\* retira elemento do inicio \*/

lista = p->next;

}

else

{

/\* retira elemento do meio da lista \*/

ant->next = p->next;

}

free(p);

return lista;

}

sLista\* insereOrdenado(sLista\* lista, int valor)

{

sVoo\* novo = criarVoo(valor); /\* cria novo nó \*/

sVoo\* ant = NULL; /\* ponteiro para elemento anterior \*/

sVoo\* p = lista; /\* ponteiro para percorrer a lista\*/

/\* procura posição de inserção \*/

while (p != NULL && p->info < valor) {

ant = p;

p = p->next;

}

/\* insere elemento \*/

if (ant == NULL) { /\* insere elemento no início \*/

novo->next = lista;

lista = novo;

}

else { /\* insere elemento no meio da lista \*/

novo->next = ant->next;

ant->next = novo;

}

return lista;

}

int main() {

sLista\* lista; sVoo\* voo;

int vooo, desejo, numv;

do

{

printf("\n[1] Cadastrar novo voo");

printf("\n[2] Verificar numero do voo");

printf("\n[3] Excluir voo");

printf("\n[4] SAIR");

scanf("%d", & desejo);

switch (desejo)

{

case 1:

printf ("\nEntre com o numero para o novo voo: ");

scanf ("%d", &numv);

lista = inserirInicio(lista, numv);

printf("\nNumero do voo: ");

scanf("%s",& voo.numero\_do\_voo);

printf("\nData do voo: ");

scanf("%s",& voo.data\_do\_voo);

printf("\nHorario do voo: ");

scanf("%s",& voo.horario\_do\_voo);

printf("\nAeroporto de saida: ");

scanf("%s",& voo.aeroporto\_de\_saida);

printf("\nAeroporto de chegada: ");

scanf("%s",& voo.aeroporto\_de\_chegada);

printf("\nRota: ");

scanf("%s",& voo.rota);

printf("\nTempo estimado: ");

scanf("%s",& voo.tempo\_estimado\_de\_voo);

printf("\nQuantia de passageiros a bordo: ");

scanf("%s",& voo.passageiros\_a\_bordo);

case 2:

printf ("\nQual numero do voo deseja consultar? ");

scanf ("%d", &numv);

voo = buscarVoo(lista, numv);

if (voo == NULL)

printf("\nValor não encontrado.");

else printf("\nValor encontrado.");

break;

imprimirLista(lista);

case 3:

printf ("\nQual voo deseja remover? ");

scanf ("%d", &numv);

lista = excluirVoo(lista, numv);

break;

}

} while (opcao = 4);

return 0;

}

**Passo 3**:

#include <stdio.h>

#include <stdlib.h>

typedef struct passagem {

int num\_pas;

int num\_voo;

int data;

int hora;

int portao;

int info;

struct passagem\* next;

} sLista, sPass;

sPass\* criarPass(int valor)

{

sPass\* p = (sPass\*)malloc(sizeof(sPass));

p->info = valor;

p->next = NULL;

return p;

}

sLista\* inserirInicio(sPass\* passagem, int valor)

{

sPass\* novo = criarPass(valor);

novo->next = passagem;

return novo;

}

void imprimirLista(sLista\* lista)

{

sPass\* p;

for (p = lista; p != NULL; p = p->next)

printf("info = %d\n", p->info);

}

sPass\* buscarPass(sLista\* lista, int valor)

{

sPass\* p;

for (p=lista; p!=NULL; p=p->next)

if (p->info == valor) return p;

return NULL;

}

sLista\* excluirVoo(sLista\* lista, int valor)

{

sPass\* ant = NULL; /\* ponteiro para elemento anterior \*/

sPass\* p = lista; /\* ponteiro para percorrer a lista\*/

while (p != NULL && p->info != valor) { /\* procura elemento na lista, guardando anterior \*/

ant = p;

p = p->next;

}

/\* verifica se achou elemento \*/

if (p == NULL) return lista; /\* não achou: retorna lista original \*/

if (ant == NULL) {

/\* retira elemento do inicio \*/

lista = p->next;

}

else

{

/\* retira elemento do meio da lista \*/

ant->next = p->next;

}

free(p);

return lista;

}

sLista\* insereOrdenado(sLista\* lista, int valor)

{

sPass\* novo = criarPass(valor); /\* cria novo nó \*/

sPass\* ant = NULL; /\* ponteiro para elemento anterior \*/

sPass\* p = lista; /\* ponteiro para percorrer a lista\*/

/\* procura posição de inserção \*/

while (p != NULL && p->info < valor) {

ant = p;

p = p->next;

}

/\* insere elemento \*/

if (ant == NULL) { /\* insere elemento no início \*/

novo->next = lista;

lista = novo;

}

else { /\* insere elemento no meio da lista \*/

novo->next = ant->next;

ant->next = novo;

}

return lista;

}

int main() {

sLista\* lista; sPass\* passagem;

int passsagem, desejo, numv;

do

{

printf("\n[1] Cadastrar nova passagem");

printf("\n[2] Verificar numero da passagem");

printf("\n[4] SAIR");

scanf("%d", & desejo);

switch (desejo)

{

case 1:

printf ("\nEntre com o numero para a nova passagem: ");

scanf ("%d", &numv);

lista = inserirInicio(lista, numv);

printf("\nNumero do voo: ");

scanf("%s",& passagem.num\_voo);

printf("\nNumero da passagem: ");

scanf("%s",& passagem.num\_pas);

printf("\nData da passagem: ");

scanf("%s",& passagem.data);

printf("\nHorario da passagem: ");

scanf("%s",& passagem.hora);

printf("\nPortao de embarque: ");

scanf("%s", & passagem.portao);

case 2:

printf ("\nQual numero da passagem deseja consultar? ");

scanf ("%d", &numv);

voo = buscarVoo(lista, numv);

if (voo == NULL)

printf("\nValor não encontrado.");

else printf("\nValor encontrado.");

break;

imprimirLista(lista);

}

} while (opcao = 4);

return 0;

}

**Relatório 3 - Filas e Pilhas**

**Passo 2:**

#include <stdio.h>

#include <stdlib.h>

/\* run this program using the console pauser or add your own getch, system("pause") or input loop \*/

typedef struct taxiamento

{

int num\_voo;

char modelo;

char empresa;

int horario;

int info;

struct taxiametro\* next;

} sTax;

typedef struct fila

{

sTax\* first;

sTax\* last;

} sFila;

void criarFila(sFila\* fila)

{

fila->first = NULL;

fila->last = NULL;

printf("\nFila criada.");

}

sTax\* criarTax(int valor)

{

sTax\* p = (sTax\*)malloc(sizeof(sTax));

p->info = valor;

p->next = NULL;

return p;

}

int vazia(sFila\* fila)

{

return fila->first == NULL;

}

p = fila->first;

num = p->info;

fila->first = p->next;

if (fila->first == NULL) fila->last == NULL;

free(p);

return num;

}

void imprimir(sFila\* fila)

{

sTax\* p;

if(vazia(fila))

printf ("\nFila esta vazia.");

else

for (p = fila->first; p != NULL; p = p->next)

printf("\nInfo = %d", p->info);

}

int main() {

int opcao, num; sFila fila; sFila\* pfila = &fila;

printf("\n");

printf("Iniciar cadastro!");

printf("Qual o numero do voo? \n");

scanf("%d",& taxiamento.num\_voo);

printf("Qual o modelo da Aeronave? \n");

scanf("% c",& taxiamento.modelo);

printf("Qual a empresa Aerea? \n");

scanf("%d",& taxiamento.empresa);

printf("Qual o horario da saida? \n");

scanf("%d",& taxiamento.horario);

criarFila(pfila);

return 0;

}

**Passo 3:**

#include <stdio.h>

#include <stdlib.h>

/\* run this program using the console pauser or add your own getch, system("pause") or input loop \*/

typedef struct bagagem {

int codigo;

int num\_Pass;

int num\_Voo;

int data;

int hora;

int portao;

int info;

struct bagagem\* next;

} sPilha, sBag;

sBag\* criarBag(int valor)

{

sBag\* p = (sBag\*)malloc(sizeof(sBag));

p->info = valor;

p->next = NULL;

return p;

}

sPilha\* inicialize()

{

printf("\nBagagem computada\n");

return NULL;

}

int empty(sPilha\* pilha)

{

return pilha == NULL;

}

sPilha\* push(sPilha\* pilha, int valor) //inclui, empilha

{

sBag\* p = criarBag(valor);

p->next = pilha;

return p;

}

int main() {

sPilha\* pilha;

int opcao, num, bagagem;

do{

printf("\n");

printf("1 CADASTRAR\n");

printf("2 - RECUPERAR\n");

printf("0 - SAIR\n");

printf("\nEntre com a opcao: "); scanf("%d", &opcao);

switch (opcao)

{

case 1:

pilha = inicialize();

printf("\nCodigo da Bagagem: ");

scanf("%d", & bagagem.codigo);

printf("\nNumero da Passagem: ");

scanf("%d", & bagagem.num\_Pass);

printf("\nNumero do voo: ");

scanf("%d", & bagagem.num\_Voo);

printf("\nData de Embarque: ");

scanf("%d", & bagagem.data);

printf("\nHorario de embarque: ");

scanf("%d", & bagagem.hora);

printf("\nPortao de Embarque: ");

scanf("%d", & bagagem.portao);

break;

case 2:

printf(pilha); printf("\n");

break;

}

} while (opcao != 0);

return 0;

}

**Relatório 4 - Grafos**

**Passo 3:**

#include<stdlib.h>

#include<stdio.h>

#include<conio.h>

void menu();

void selecao();

void montargrafo();

void caminhagrafo();

int main()

{

menu();

return 0;

system("pause");

}

void menu()

{

printf(" | Escolha uma opcao abaixo |\n");

printf(" |--------------------------|\n");

printf(" | 1 - Cadastrar cidades |\n");

printf(" | 2 - Melhor rota |\n");

printf(" | 0 - Sair |\n");

printf(" |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\n");

selecao();

}

void selecao()

{

int desejo;

printf("\n Digite uma das opcoes: ");

scanf("%i", &desejo);

switch (desejo)

{

case 1:

montargrafo();

menu();

break;

case 2:

caminhagrafo();

menu();

break;

case 0:

system("pause");

exit(0);

break;

default:

printf("\n\nOpcao nao encontrada.\nTente Novamente\n\n");

system("pause");

system("cls");

menu();

break;

}

}

//FUNÇÃO MONTAR GRAFO (CADASTRO DE CIDADES)

void montargrafo()

{

int num,I,J;

char cidade[10][10];

char A[10][10];

char B[10][10];

do

{

system("cls");

printf(" 1 - Cadastrar cidades\n\n");

for(I=0; I<10; I++)

{

printf(" Entre com a cidade numero %d: ",I);

fflush(stdin);

scanf("%s", &cidade[I]);

}

printf("\n\n Entre com a distancia das cidades:\n\n");

for(I=0; I<10; I++)

{

for(J=0; J<10; J++)

{

printf("\n Distancia entre %s e %s <Km>: ", cidade[I],cidade[J]);

scanf("%s", &A[I][J]);

printf(" Tempo entre %s e %s <hrs>: ",cidade[I],cidade[J]);

scanf("%s", &B[I][J]);

}

}

printf(" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

printf(" | Escolha uma opcao abaixo |\n");

printf(" |--------------------------|\n");

printf(" | 1 - Cadastrar cidades |\n");

printf(" | 0 - Voltar ao menu |\n");

printf(" |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\n");

printf("\n Opcao: ");

scanf("%d",&num);

}

while(num!=0);

system("cls");

}

//FUNÇÃO CAMINHO GRAFO (MELHOR CAMINHO)

void caminhagrafo()

{

int aux,num,I,J,T,F,R,vet\_resposta[10],menor,vet\_caminho[10],vet\_peso[10],LUGAR\_INICIAL,LUGAR\_FINAL,NO\_ATUAL,N1,N2;

char cidade[10][10];

char A[10][10];

char B[10][10];

do

{

//PREENCHE OS VETORES CAMINHO E PESO

for(I=0; I<10; I++)

{

vet\_caminho[I]=-1;

vet\_peso [I]=0;

}

system("cls");

printf(" 2 - Melhor rota\n\n");

printf(" Entre a cidade inicial: ");

scanf("%d", &LUGAR\_INICIAL);

N1=LUGAR\_INICIAL;

printf(" Entre a cidade final: ");

scanf("%d", &LUGAR\_FINAL);

N2=LUGAR\_FINAL;

if(LUGAR\_INICIAL != LUGAR\_FINAL)

{

NO\_ATUAL=LUGAR\_INICIAL;

I=NO\_ATUAL;

vet\_caminho[I]= 1;

}

if(I<=10)

{

for(J=0; J<10; J++)

{

if(A[I][J] != 0)

{

if(vet\_caminho[J]== 1)

{

if(vet\_peso[J]> A[I][J])

{

if(vet\_peso[J]>-1)

{

vet\_peso[J]= A[I][J];

vet\_peso[J]= B[I][J];

NO\_ATUAL=I;

}

}

}

}

}

}

//verificando se o menor valor do vetor peso é diferente de -1 e a mesma posição no vetor caminho é igual a "F"

menor= vet\_peso[0];

for (T=0; T<10; T++)

{

if(vet\_peso[T]< menor)

{

menor = vet\_peso[T];

}

if(vet\_peso[T]!=1)

{

if(vet\_caminho[T]==F)

{

NO\_ATUAL=I;

vet\_resposta[I]=NO\_ATUAL;

}

if(NO\_ATUAL==LUGAR\_FINAL)

{

for (R=0; R<10; R++)

{

if(vet\_resposta[R]!=0)

{

aux=vet\_resposta[R];

printf("\n %d melhor rota é %dKm\n", R, vet\_resposta[R]);

}

else

I++;

}

}

}

}

printf(" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

printf(" | Escolha uma opcao abaixo |\n");

printf(" |--------------------------|\n");

printf(" | 1 - Melhor rota |\n");

printf(" | 0 - Voltar ao menu |\n");

printf(" |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\n");

printf("\n Opcao: ");

scanf("%d",&num);

}

while(num!=0);

system("cls");

}