Aula 04

#include <stdio.h>

#include <stdlib.h>

struct Node

{

int dado;

struct Node \*next;

};

void insere\_inicio(struct Node\*\* inicio\_ref, int new\_dado)

{

}

void insere\_meio(struct Node\* prev\_node, int new\_dado)

{

}

void insere\_fim(struct Node\*\* inicio\_ref, int new\_dado)

{

}

void printList(struct Node \*node)

{

}

int main()

{

struct Node\* inicio = NULL;

insere\_fim(&inicio, 6);

insere\_inicio(&inicio, 7);

insere\_inicio(&inicio, 1);

insere\_fim(&inicio, 4);

insere\_meio(inicio->next, 8);

printf("\n Lista Criada: ");

printList(inicio);

return 0;

}

Exemplo Completo

#include <stdio.h>

#include <stdlib.h>

struct Node

{

int dado;

struct Node \*next;

};

void insere\_inicio(struct Node\*\* inicio\_ref, int new\_dado)

{

struct Node\* new\_node = (struct Node\*) malloc(sizeof(struct Node));

new\_node->dado = new\_dado;

new\_node->next = (\*inicio\_ref);

(\*inicio\_ref) = new\_node;

}

void insertAfter(struct Node\* prev\_node, int new\_dado)

{

if (prev\_node == NULL)

{

printf("the given previous node cannot be NULL");

return;

}

struct Node\* new\_node =(struct Node\*) malloc(sizeof(struct Node));

new\_node->dado = new\_dado;

new\_node->next = prev\_node->next;

prev\_node->next = new\_node;

}

void insere\_fim(struct Node\*\* inicio\_ref, int new\_dado)

{

struct Node\* new\_node = (struct Node\*) malloc(sizeof(struct Node));

struct Node \*ultimo = \*inicio\_ref;

new\_node->dado = new\_dado;

new\_node->next = NULL;

if (\*inicio\_ref == NULL)

{

\*inicio\_ref = new\_node;

return;

}

while (ultimo->next != NULL)

ultimo = ultimo->next;

ultimo->next = new\_node;

return;

}

void printList(struct Node \*node)

{

while (node != NULL)

{

printf(" %d ", node->dado);

node = node->next;

}

}

int main()

{

struct Node\* inicio = NULL;

insere\_fim(&inicio, 6);

insere\_inicio(&inicio, 7);

insere\_inicio(&inicio, 1);

insere\_fim(&inicio, 4);

insertAfter(inicio->next, 8);

printf("\n Lista Criada: ");

printList(inicio);

return 0;

}

Exemplo Fila

#include <stdio.h>

#include <stdlib.h>

struct No {

int chave;

struct No\* prox;

};

struct Fila {

struct No \*inicio, \*fim;

};

struct No\* newNode(int k)

{

struct No\* temp = (struct No\*)malloc(sizeof(struct No));

temp->chave = k;

temp->prox = NULL;

return temp;

}

struct Fila\* inicia\_Fila()

{

struct Fila\* q = (struct Fila\*)malloc(sizeof(struct Fila));

q->inicio = q->fim = NULL;

return q;

}

void insere\_fila(struct Fila\* q, int k)

{

struct No\* temp = newNode(k);

if (q->fim == NULL) {

q->inicio = q->fim = temp;

return;

}

q->fim->prox = temp;

q->fim = temp;

}

void remove\_fila(struct Fila\* q)

{

// If Fila is empty, return NULL.

if (q->inicio == NULL)

return;

struct No\* temp = q->inicio;

q->inicio = q->inicio->prox;

if (q->inicio == NULL)

q->fim = NULL;

free(temp);

}

bool search(struct Node\* inicio, int x)

{

struct Node\* atual = inicio;

while (atual != NULL)

{

if (atual->chave== x)

return true;

atual= atual->prox;

}

return false;

}

int main()

{

struct Fila\* q = inicia\_Fila();

insere\_fila(q, 10);

insere\_fila(q, 20);

remove\_fila(q);

remove\_fila(q);

insere\_fila(q, 30);

insere\_fila(q, 40);

insere\_fila(q, 50);

remove\_fila(q);

printf("Fila Inicio : %d \n", q->inicio->chave);

printf("Fila Fim : %d", q->fim->chave);

return 0;

}