Filas:

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

struct Node{

int num;

struct Node \*prox;

};

typedef struct Node node;

int tam;

int menu(void);

void opcao(node \*FILA, int op);

void inicia(node \*FILA);

int vazia(node \*FILA);

node \*aloca();

void insere(node \*FILA);

node \*retira(node \*FILA);

void exibe(node \*FILA);

void libera(node \*FILA);

int main(void)

{

node \*FILA = (node \*) malloc(sizeof(node));

if(!FILA){

printf("Sem memoria disponivel!\n");

exit(1);

}else{

inicia(FILA);

int opt;

do{

opt=menu();

opcao(FILA,opt);

}while(opt);

free(FILA);

return 0;

}

}

int menu(void)

{

setlocale(LC\_ALL, "Portuguese");

int opt;

printf("\n");

printf("Escolha a opcao\n");

printf("0. Sair\n");

printf("1. Zerar fila\n");

printf("2. Exibir fila\n");

printf("3. Adicionar Elemento na Fila\n");

printf("4. Retirar Elemento da Fila\n");

printf("5. Exibe o primeiro elemento\n");

printf("6. Fila vazia\n");

printf("7. Fila Cheia\n");

printf("Opcao: "); scanf("%d", &opt);

printf("\n");

printf("\n");

return opt;

}

void opcao(node \*FILA, int op)

{

node \*tmp;

switch(op){

case 0:

libera(FILA);

break;

case 1:

libera(FILA);

inicia(FILA);

break;

case 2:

exibe(FILA);

break;

case 3:

insere(FILA);

break;

case 4:

tmp= retira(FILA);

if(tmp != NULL){

printf("Retirado: %3d\n\n", tmp->num);

libera(tmp);

}

break;

case 5:

if(FILA!=NULL){

printf("O primeiro elemental da fila é: %d", FILA->prox->num);

printf("\n");

printf("\n");

}else{

printf("A fila está vazia.");

printf("\n");

printf("\n");

}

break;

case 6:

if(FILA->prox==NULL){

printf("A fila esta vazia.");

printf("\n");

}else{

printf("A fila não está vazia.");

printf("\n");

}

break;

case 7:

tmp = FILA;

int j=0;

while(tmp->prox!=NULL){

tmp=tmp->prox;

j++;

}

printf("A fila tem %d elementos.", j);

printf("\n");

break;

default:

printf("Comando invalido\n\n");

}

}

void inicia(node \*FILA)

{

FILA->prox = NULL;

tam=0;

}

int vazia(node \*FILA)

{

if(FILA->prox == NULL)

return 1;

else

return 0;

}

node \*aloca()

{

node \*novo=(node \*) malloc(sizeof(node));

if(!novo){

printf("Sem memoria disponivel!\n");

exit(1);

}else{

printf("Novo elemento: "); scanf("%d", &novo->num);

return novo;

}

}

void insere(node \*FILA)

{

node \*novo=aloca();

novo->prox = NULL;

if(vazia(FILA))

FILA->prox=novo;

else{

node \*tmp = FILA->prox;

while(tmp->prox != NULL)

tmp = tmp->prox;

tmp->prox = novo;

}

tam++;

}

node \*retira(node \*FILA)

{

if(FILA->prox == NULL){

printf("Fila ja esta vazia\n");

return NULL;

}else{

node \*tmp = FILA->prox;

FILA->prox = tmp->prox;

tam--;

return tmp;

}

}

void exibe(node \*FILA)

{

if(vazia(FILA)){

printf("Fila vazia!\n\n");

return ;

}

node \*tmp;

tmp = FILA->prox;

printf("Fila :");

while( tmp != NULL){

printf("%5d", tmp->num);

tmp = tmp->prox;

}

printf("\n ");

int count;

for(count=0 ; count < tam ; count++)

printf(" ^ ");

printf("\nOrdem:");

for(count=0 ; count < tam ; count++)

printf("%5d", count+1);

printf("\n\n");

}

void libera(node \*FILA)

{

if(!vazia(FILA)){

node \*proxNode,

\*atual;

atual = FILA->prox;

while(atual != NULL){

proxNode = atual->prox;

free(atual);

atual = proxNode;

}

}

}