UFU

Nome: Antonio Carlos Neto

Curso: Ciência da Computação

Disciplina: GBC024

MAIN.C:

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#define SIZE 5 /\* Size of Queue \*/

#define SIZI 4 /\* Size of Queue - 1\*/

typedef struct PRQ

{

int ele;

int pr;

} PriorityQ;

typedef struct FilaQ\_

{

int f, r, quant;

PriorityQ PQ[SIZE];

} FilaQ;

FilaQ\* cria\_FilaQ()

{

FilaQ \*Net = (FilaQ\*) malloc(sizeof(struct FilaQ\_));

if(Net != NULL)

{

Net->f = 0;

Net->r = 0;

Net->quant = 0;

}

return Net;

}

void libera\_FilaQ(FilaQ\* Net)

{

free(Net);

}

void PQinsert(FilaQ\* Net, int elem, int pre)

{

int i, j=0; /\* Function for Insert operation \*/

if(Qfull(Net) == 1) printf("\n\n Overflow!!!!\n\n");

else

{

if(Qfull(Net) == -1) printf("\n\n ERROR IN THE LIST!!!!\n\n");

else

{

i = Net->r;

Net->r = (Net->r+1)%SIZE;

for(;;)

{

if(j >= Net->quant) break;

if(Net->PQ[(i+SIZI)%SIZE].pr < pre) break;

if(Net->quant == 0) break;

Net->PQ[i]=Net->PQ[(i+SIZI)%SIZE];

i = (i+SIZI)%SIZE;

j++;

//printf("\n VALOR I : %d",i);

}

Net->PQ[i].ele = elem;

Net->PQ[i].pr = pre;

Net->quant++;

// printf("\n\n%d , %d\n\n",Net->PQ[1].ele,Net->PQ[1].pr);

}

}

}

PriorityQ PQdelete(FilaQ\* Net)

{

PriorityQ p;

if(Qempty(Net) == -1)

{

printf("\n\n ERROR IN THE LIST!!!!\n\n");

p.ele=-1;

p.pr=-1;

return(p);

}

else if(Qempty(Net))

{

printf("\n\nUnderflow!!!!\n\n");

p.ele=-1;

p.pr=-1;

return(p);

}

else

{

p=Net->PQ[Net->f];

Net->f = (Net->f+1)%SIZE;

Net->quant--;

return(p);

}

}

int Qfull(FilaQ\* Net)

{

if(Net == NULL) return -1;

if(Net->quant == SIZE) return 1;

return 0;

}

int Qempty(FilaQ\* Net)

{

if(Net == NULL) return -1;

if(Net->quant == 0) return 1;

return 0;

}

void display(FilaQ\* Net)

{

/\* Function to display status of Queue \*/

int i,j=0, k=0;

if(Qempty(Net)) printf(" \n Empty Queue\n");

else

{

printf("Front->");

for(i = Net->f; j < Net->quant; i=(i+1)%SIZE)

{

printf("[%d,%d] ",Net->PQ[i].ele,Net->PQ[i].pr);

// printf("%d : [%d,%d] ",(Net->f + k)%SIZE,Net->PQ[i].ele,Net->PQ[i].pr);

j++;

k++;

}

printf("<-Rear\n\n");

}

}

int main()

{

int opn, x, y;

FilaQ \*Net;

Net = cria\_FilaQ();

PriorityQ p;

do

{

printf("\n### Priority Queue Operations(DSC order) ###\n\n");

printf("\nPress 1-Insert, 2-Delete,3-Display,4-Exit\n");

printf("\nYour option ? ");

scanf("%d",&opn);

switch(opn)

{

case 1:

printf("\n\nRead the element and its Priority?\n");

scanf("%d%d",&x,&y);

// printf("\n\n%d , %d\n\n",x,y);

PQinsert(Net,x,y);

break;

case 2:

p=PQdelete(Net);

if( p.ele != -1)

printf("\n\nDeleted Element is %d \n",p.ele);

break;

case 3:

printf("\n\nStatus of Queue\n\n");

display(Net);

break;

case 4:

printf("\n\n Terminating \n\n");

break;

default:

printf("\n\nInvalid Option !!! Try Again !! \n\n");

break;

}

// printf("\n\n\n\n Press a Key to Continue . . . ");

// getch();

// system("Pause");

// printf("\n\n%d , %d , %d\n\n",Net->f,Net->r,Net->quant);

// printf("\n\n%d , %d\n\n",Net->PQ[1].ele,Net->PQ[1].pr);

}

while(opn != 4);

return 0;

}