**Priority Queue to Add and Delete Elements**

#include <stdio.h>

#include <stdlib.h>

#define MAX 5

void insert\_by\_priority(int);

void delete\_by\_priority(int);

void create();

void check(int);

void display\_pqueue();

int pri\_que[MAX];

int front, rear;

void main()

{

int n, ch;

printf("\n1 - Insert an element into queue");

printf("\n2 - Delete an element from queue");

printf("\n3 - Display queue elements");

printf("\n4 - Exit");

create();

while (1)

{

printf("\nEnter your choice : ");

scanf("%d", &ch);

switch (ch)

{

case 1:

printf("\nEnter value to be inserted : ");

scanf("%d",&n);

insert\_by\_priority(n);

break;

case 2:

printf("\nEnter value to delete : ");

scanf("%d",&n);

delete\_by\_priority(n);

break;

case 3:

display\_pqueue();

break;

case 4:

exit(0);

default:

printf("\nChoice is incorrect, Enter a correct choice");

}

}

}

/\* Function to create an empty priority queue \*/

void create()

{

front = rear = -1;

}

/\* Function to insert value into priority queue \*/

void insert\_by\_priority(int data)

{

if (rear >= MAX - 1)

{

printf("\nQueue overflow no more elements can be inserted");

return;

}

if ((front == -1) && (rear == -1))

{

front++;

rear++;

pri\_que[rear] = data;

return;

}

else

check(data);

rear++;

}

/\* Function to check priority and place element \*/

void check(int data)

{

int i,j;

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

{

if (data >= pri\_que[i])

{

for (j = rear + 1; j > i; j--)

{

pri\_que[j] = pri\_que[j - 1];

}

pri\_que[i] = data;

return;

}

}

pri\_que[i] = data;

}

/\* Function to delete an element from queue \*/

void delete\_by\_priority(int data)

{

int i;

if ((front==-1) && (rear==-1))

{

printf("\nQueue is empty no elements to delete");

return;

}

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

{

if (data == pri\_que[i])

{

for (; i < rear; i++)

{

pri\_que[i] = pri\_que[i + 1];

}

pri\_que[i] = -99;

rear--;

if (rear == -1)

front = -1;

return;

}

}

printf("\n%d not found in queue to delete", data);

}

/\* Function to display queue elements \*/

void display\_pqueue()

{

if ((front == -1) && (rear == -1))

{

printf("\nQueue is empty");

return;

}

for (; front <= rear; front++)

{

printf(" %d ", pri\_que[front]);

}

front = 0;

}