Yash Sarang.

Roll No: 47, Class : D6AD.

Data Structures. Experiment-07.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Aim : Implement Priority Queue ADT using array.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Theory:

A priority queue is a special type of queue in which each element is associated with a priority and is served according to its priority.

The general rules of processing the elements of a priority queue are

* An element with higher priority is processed before an element with a lower priority
* Two elements with the same priority are processed on a first-come-first-served (FCFS) basis

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Algorithm:

Algorithm to enter element in queue

STEP 1: START

STEP 2: IF((Front == 0)&&(Rear == N-1))

PRINT “Overflow Condition”

STEP 3: Else

IF(Front == -1)

Front = Rear =0

Queue[Rear] = Data

Priority[Rear] = Priority

STEP 4: i = Rear

STEP 5: IF(p>Priority[i])

Queue[i+1] = Queue[i]

Priority[i+1] = Priority[i]

STEP 6: ELSE

Queue[i+1] = data

Priority[i+1] = p

Rear++

STOP

STEP 7: i--

STEP 8: IF i>Front Repeat 5

STEP 9: STOP

Algorithm to delete element in queue

STEP 1: START

STEP 2:

IF(Front == -1)

PRINT “Queue Under flow condition”

STEP 3: ELSE

PRINT”Q[f],Pr[f]”

STEP 4: IF(Front==Rear)

Front = Rear = -1

STEP 5: ELSE

FRONT++

STEP 6: STOP

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Code:

#include <stdio.h>

#include <stdlib.h>

#define N 5

int queue[N],pr[N];

int r = -1,f = -1;

void enqueue(int data,int p)

{

int i;

if((f==0)&&(r==N-1))

printf("Queue is full");

else

{

if(f==-1)

{

f = r = 0;

queue[r] = data;

pr[r] = p;

}

else

{

for(i = r;i>=f;i--)

{

if(p>pr[i])

{

queue[i+1] = queue[i];

pr[i+1] = pr[i];

}

else

break;

}

queue[i+1] = data;

pr[i+1] = p;

r++;

}

}

}

void print()

{

int i;

for(i=f;i<=r;i++)

{

printf("\nElement = %d\tPriority = %d",queue[i],pr[i]);

}

}

int dequeue()

{

if(f == -1)

{

printf("Queue is Empty");

}

else

{

printf("Deleted Element = %d\t Priority = %d",queue[f],pr[f]);

if(f==r)

f = r = -1;

else

f++;

}

}

int main()

{

int opt,n,i,data,p;

do{

system("cls");

printf("\n\*\*\*\*\*Priority Queue\*\*\*\*\*");

printf("\n\n1. Insert Element");

printf("\n2. Delete Element");

printf("\n3. Display");

printf("\n4. Exit");

printf("\n\n Enter your choice: ");

scanf("%d",&opt);

switch(opt){

case 1:

printf("\nEnter your data and Priority of data: ");

scanf("%d %d",&data,&p);

enqueue(data,p);

break;

case 2:

dequeue();

break;

case 3:

print();

break;

case 4:

printf(“\nThank You”);

break;

default:

printf("\nInvalid Input");

break;

}

getch();

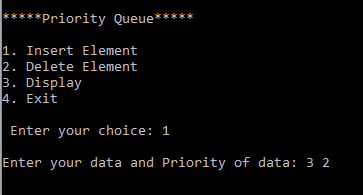
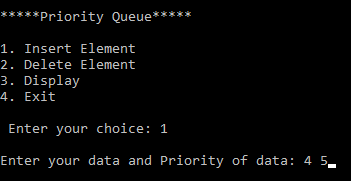
}while(opt!=4);

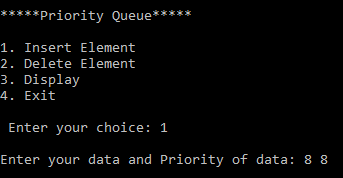
return 0;

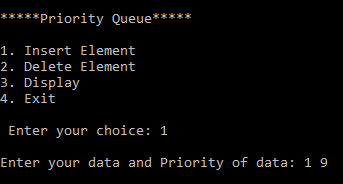
}

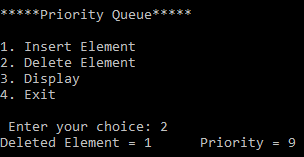
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

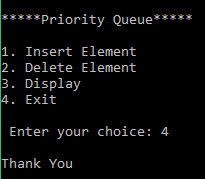
Output:

****

****

****

****

****

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_