Experiment Name / No.: 5

Camlin | Page No. 17
Date / /

IMPLEMENTATION OF QUEUES & CIRCULAR QUEUES

AIM

write a C program to implement Queues and circular Queue using arrays and perform the operations on the stack i) Push ii) POP iii) peek iv) isempty v) Isfull vi) Display

ALGORITHM

main() function

1 Start
2 Ask the user, weather to use normal queue or circular queue
3 If user selects normal queue
3.1 Ask the user weather to insert(), Delete() Display(), isempty(), isfull(), or quit
3.2 If user chooses insert
3.2.1 call the insert() function
3.3 If user chooses delete, call the delete() function
3.4 If user chooses display, call the display()
3.5 If user chooses isfull, call the isfull()
3.6 If user chooses exit, exit out of program
4 If user selects circular queue
4.1 Ask the user weather to insert(), Delete() Display(), isempty(), isfull() or quit

Teacher's Signature:

4.2 If user chooses insert, call the enqueue() function

4.3 If user chooses delete, call the dequeue() function

4.4 If user chooses display, call the display() function

4.5 If user chooses isempty,

4.5.1 x = isempty()

4.5.2 If x = 1, print queue is empty

4.5.3 If x = 0, print queue is not empty

4.6 If user chooses isfull,

4.6.1 x = isfull()

4.6.2 If x = 1, print queue is full

4.6.3 If x = 0, print queue is not full

5 repeat the steps while user chooses exit.

6 Stop

→ insert() function

1 start

2 If rear = MAX - 1, print queue overflow

3 else

3.1 If front == -1, front = 0

3.2 input the element to be inserted from user

3.3 rear = rear + 1

3.4 queue_array[rear] = add_item

4 Stop

-> delete() function

1 Start
2 if front == -1 or front > rear
2.1 print, queue underflow
3 else
3.1 display queue_array[front] as element to be deleted from queue
3.2 front = front + 1
4 Stop

-> qdisplay() function

1 Start
2 if front == -1, print queue is empty
3 else
3.1 i = front
3.2 repeat while i <= rear
3.2.1 print queue_array[i]
3.2.2 increment i by 1
4 Stop

-> qisempty() function

1 Start
2 if front > rear or (front == -1 and rear == -1)

Teacher's Signature: _____

2.1 print queue is empty
3 else print queue is not empty
4 Stop
5
-> qisfull() function
1 Start
2 if rear == MAX-1
2.1 print queue is full
3 else print queue is not full
4 Stop

-> isfull() function
1 Start
2 if (front = rear+1) or (front = 0 and rear = max-1)
2.1 return 1
3 else return 0
4 Stop

-> isempty() function
1 Start
2 if front == -1, return 1
3 else return 0
4 Stop
5

Teacher's Signature: _____

Experiment Name / No.: _____

Camlin | Page No. 21
Date / /

→ enqueue() function

1 start

2 ask from the user, the element to be inserted

3 if (isfull()), print queue is full

4 else

4.1 if front == -1, front = 0

4.2 rear = (rear + 1) % MAX

10 4.3 queue_array[rear] = element

4.4 print the element which is inserted

5 stop

→ dequeue() function

15 start

2 if (isempty), print queue is empty

3 else, element = queue_array[front]

3.1 if front == rear, front = -1, rear = -1

3.2 else front = (front + 1) % max

20 3.3 print the deleted element as element

4 stop

→ display() function

1 start

25 if (isempty())

2.1 print empty queue

Teacher's Signature: _____

3 else

3.1 i = front

3.2 repeat while i != rear

5 3.2.1 print queue_array[i]

3.2.2 i = (i+1) % MAX

3.3 print queue_array[i]

4 stop

10 CONCLUSION

The program has been executed correctly and output has been verified.

15

20

25

Teacher's Signature:

Experiment Name / No.: 6

Camlin | Page No. 23
Date / /

IMPLEMENTATION OF PRIORITY QUEUE

AIM

Write a menu driven C program to implement priority queues using arrays

ALGORITHM

→ main() function

1 Start
2 ask the user to weather insert(), show() or delete
3 if user chooses insert
  3.1 get the value to be inserted and priority
  3.2 call the enque(data, pr) function
4 if user chooses delete, call the dequeue() function
5 if user chooses show, call the print() function
6 Stop

→ enqueue() function

1. if r == N-1, print queue is full
2. else
  2.1 if f == -1, f = r = 0, Q[r] = data, pr[r] = p
  2.2 else
    2.2.1 i = r, repeat while i >= f

Teacher's Signature:

2.2.1.1 if P > PR[i], Q[i+1] = Q[i], pr[i+1] = PR[i]
2.2.1.2 else, breaks
2.2.2 Q[i+1] = data
2.2.3 pr[i+1] = P
2.2.4 r = r+1
3 STOP

-> print() function
10 start
2 i = f, repeat while i<r
2.1 print element and priority
2.2 i = i+1
3 stop
-> dequeue() function
15 start
2 if f == -1, print queue is empty
3 else
3.1 print the Q[f] and pr[f] as deleted element and priority
20 3.2 if f == r, f = r = -1
3.3 else f++
4 STOP

25 CONCLUSION
Program has been executed correctly

Teacher's Signature: _____