Lab Assignment-9

ROLL: 2005535 | NAME: SAHIL SINGH | DATE: 21/09/21

QUES 1: [A] Write a menu driven program to perform the following operations in a QUEUE ADT

(Using an Array) by using suitable user defined functions for each case.

1. Inserting an element into the queue [Define Isfull() function to check overflow]

2. Deleting an element from the queue [Define Isempty() function to check underflow]

3. Display the elements of queue

4. Copy the content of one queue into another (Without using any additional data structure)

5. Reverse the elements of the queue (Without using any additional data structure)

SOLUTION:

#include <stdio.h>

#include <stdlib.h>

#define DEFNULL -99999

#define MAXSIZE 100

typedef *struct* Queue

{

*int* front;

*int* rear;

*int* data[MAXSIZE];

} Queue;

*void* enqueue(Queue \*, *int*);

*int* dequeue(Queue \*);

*int* peek(Queue \*);

*int* isFull(Queue \*);

*int* isEmpty(Queue \*);

*void* cpy(Queue \*, Queue \*);

*void* reverse(Queue \*);

*void* show\_queue(Queue \*); //debug

*void* util\_cpy(Queue \*, Queue \*);

*int* main()

{

    Queue q1 = {-1, -1};

    Queue q2 = {-1, -1};

*int* choice;

    do

    {

        printf("1) Insertion\n2) Display\n3) Deletion\n");

        printf("4) Copy\n5) Reverse\n6) Exit\n->: ");

        scanf("%d", &choice);

*int* val;

        printf("\n");

        switch (choice)

        {

        case 1:

            printf("Enter value to insert: ");

            scanf("%d", &val);

            enqueue(&q1, val);

            show\_queue(&q1);

            break;

        case 2:

            show\_queue(&q1);

            break;

        case 3:

            printf("Deleted element: ");

            printf("%d\n", dequeue(&q1));

            show\_queue(&q1);

            break;

        case 4:

            printf("Copied Queue:\n");

            cpy(&q1, &q2);

            show\_queue(&q2);

            break;

        case 5:

            printf("Reversed Queue:\n");

            reverse(&q1);

            show\_queue(&q1);

            break;

        default:

            printf("Exiting...\n");

        }

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

    } while (choice >= 1 && choice <= 5);

    return 0;

}

*void* enqueue(Queue \**que*, *int* *num*)

{

    if (isFull(*que*))

    {

        printf("Overflow!\n");

        return;

    }

    else if (isEmpty(*que*))

*que*->front = *que*->rear = 0;

    else

*que*->rear = (*que*->rear + 1) % MAXSIZE;

*que*->data[*que*->rear] = *num*;

}

*int* dequeue(Queue \**que*)

{

*int* retIndex = *que*->front;

    if (isEmpty(*que*))

    {

        printf("Underflow!\n");

        return DEFNULL;

    }

    else if (*que*->front == *que*->rear)

    {

*que*->front = *que*->rear = -1;

        return *que*->data[retIndex];

    }

*que*->front = (*que*->front + 1) % MAXSIZE;

    return *que*->data[retIndex];

}

*int* peek(Queue \**que*)

{

    if (isEmpty(*que*))

        return DEFNULL;

    return *que*->data[*que*->front];

}

*int* isFull(Queue \**que*)

{

    if ((*que*->rear + 1) % MAXSIZE == *que*->front)

        return 1;

    return 0;

}

*int* isEmpty(Queue \**que*)

{

    if (*que*->front == -1)

        return 1;

    return 0;

}

*void* cpy(Queue \**que1*, Queue \**que2*)

{

*que2*->front = *que2*->rear = -1;

    util\_cpy(*que1*, *que2*);

    reverse(*que1*);

}

*void* reverse(Queue \**que*)

{

    if (isEmpty(*que*))

        return;

*int* temp = dequeue(*que*);

    reverse(*que*);

    enqueue(*que*, temp);

}

*void* show\_queue(Queue \**que*)

{

    Queue tempQue = {-1, -1};

    while (!isEmpty(*que*))

    {

        enqueue(&tempQue, peek(*que*));

        printf("%d->", dequeue(*que*));

    }

    while (!isEmpty(&tempQue))

        enqueue(*que*, dequeue(&tempQue));

    printf("\b\b \n");

}

*void* util\_cpy(Queue \**que1*, Queue \**que2*)

{

    if (isEmpty(*que1*))

        return;

*int* temp = dequeue(*que1*);

    enqueue(*que2*, temp);

    util\_cpy(*que1*, *que2*);

    enqueue(*que1*, temp);

}

OUTPUT:

1) Insertion

2) Display

3) Deletion

4) Copy

5) Reverse

6) Exit

->: 1

Enter value to insert: 12

12 >

----------------------------------

1) Insertion

2) Display

3) Deletion

4) Copy

5) Reverse

6) Exit

->: 1

Enter value to insert: 13

12->13 >

----------------------------------

1) Insertion

2) Display

3) Deletion

4) Copy

5) Reverse

6) Exit

->: 1

Enter value to insert: 14

12->13->14 >

----------------------------------

1) Insertion

2) Display

3) Deletion

4) Copy

5) Reverse

6) Exit

->: 2

12->13->14 >

----------------------------------

1) Insertion

2) Display

3) Deletion

4) Copy

5) Reverse

6) Exit

->: 3

Deleted element: 12

13->14 >

----------------------------------

1) Insertion

2) Display

3) Deletion

4) Copy

5) Reverse

6) Exit

->: 4

Copied Queue:

13->14 >

----------------------------------

1) Insertion

2) Display

3) Deletion

4) Copy

5) Reverse

6) Exit

->: 5

Reversed Queue:

14->13 >

----------------------------------

1) Insertion

2) Display

3) Deletion

4) Copy

5) Reverse

6) Exit

->: 2

14->13 >

----------------------------------

1) Insertion

2) Display

3) Deletion

4) Copy

5) Reverse

6) Exit

->: 6

Exiting...

----------------------------------

QUES 2: [B] Write a menu driven program to perform the following operations in a QUEUE ADT

(Using linked list) by using suitable user defined functions for each case.

1. Inserting an element into the queue

2. Deleting an element from the queue [Define Isempty() function to check underflow]

3. Display the element of the queue

SOLUTION:

#include <stdio.h>

#include <stdlib.h>

#define DEFNULL -999999

typedef *struct* Node

{

*int* data;

*struct* Node \*link;

} Node;

typedef *struct* Queue

{

    Node \*front;

    Node \*rear;

} Queue;

*void* enqueue(Queue \*, *int*);

*int* dequeue(Queue \*);

*int* peek(Queue \*);

*int* isEmpty(Queue \*);

*void* show\_queue(Queue \*); //debug

*int* main()

{

    Queue que = {NULL, NULL};

*int* choice;

    do

    {

        printf("1) Insertion\n2) Display\n3) Deletion\n4) Exit\n->: ");

        scanf("%d", &choice);

*int* val;

        printf("\n");

        switch (choice)

        {

        case 1:

            printf("Enter value to insert: ");

            scanf("%d", &val);

            enqueue(&que, val);

            show\_queue(&que);

            break;

        case 2:

            show\_queue(&que);

            break;

        case 3:

            printf("Deleted element: ");

            printf("%d\n", dequeue(&que));

            show\_queue(&que);

            break;

        default:

            printf("Exiting...\n");

        }

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

    } while (choice >= 1 && choice <= 3);

    return 0;

}

*void* enqueue(Queue \**que*, *int* *num*)

{

    Node \*temp = (Node \*)malloc(sizeof(Node));

    temp->data = *num*;

    temp->link = NULL;

    if (isEmpty(*que*))

    {

*que*->front = *que*->rear = temp;

        return;

    }

*que*->rear->link = temp;

*que*->rear = *que*->rear->link;

}

*int* dequeue(Queue \**que*)

{

    if (isEmpty(*que*))

        return DEFNULL;

    Node \*temp = *que*->front;

*que*->front = *que*->front->link;

    if (*que*->front == NULL)

*que*->rear = NULL;

*int* n = temp->data;

    free(temp);

    return n;

}

*int* peek(Queue \**que*)

{

    if (isEmpty(*que*))

        return DEFNULL;

    return *que*->front->data;

}

*int* isEmpty(Queue \**que*)

{

    if (*que*->front == NULL)

        return 1;

    return 0;

}

*void* show\_queue(Queue \**que*)

{

    Queue temp = {NULL, NULL};

    while (!isEmpty(*que*))

    {

        printf("%d->", peek(*que*));

        enqueue(&temp, dequeue(*que*));

    }

    printf("\b\b \n");

*que*->front = temp.front;

*que*->rear = temp.rear;

}

OUTPUT:

1) Insertion

2) Display

3) Deletion

4) Exit

->: 1

Enter value to insert: 10

10 >

----------------------------------

1) Insertion

2) Display

3) Deletion

4) Exit

->: 1

Enter value to insert: 20

10->20 >

----------------------------------

1) Insertion

2) Display

3) Deletion

4) Exit

->: 1

Enter value to insert: 30

10->20->30 >

----------------------------------

1) Insertion

2) Display

3) Deletion

4) Exit

->: 2

10->20->30 >

----------------------------------

1) Insertion

2) Display

3) Deletion

4) Exit

->: 3

Deleted element: 10

20->30 >

----------------------------------

1) Insertion

2) Display

3) Deletion

4) Exit

->: 2

20->30 >

----------------------------------

1) Insertion

2) Display

3) Deletion

4) Exit

->: 4

Exiting...

----------------------------------