PROGRAM NO: 6 DATE: 14-09-2023

**AIM**

To implement queue data structure with Enqueue, Dequeue, Peek and Display.

**PROGRAM**

1. START
2. FUNC Queue(size): // Declaring queue
3. DECLARE arr by allocating required size.
4. SET rear = -1
5. SET fromt = 0
6. END FUNC
7. FUNC isEmpty():
8. IF rear = -1, return 1
9. ELSE, return 0
10. END IF, END FUNC
11. FUNC isFull():
12. IF rear = size – 1, return 1
13. ELSE, return 0
14. END IF, END FUNC
15. FUNC Enqueue(item):
16. IF isFull(), throw QueueOverflow
17. END IF
18. INCREMENT rear
19. SET arr[rear] = item
20. END FUNC
21. FUNC Dequeue(item):
22. IF isEmpty(), throw QueueUnderflow
23. END IF
24. SET popItem = arr[front]
25. INCREMENT front
26. IF front > rear
27. SET rear = -1, SET front = 0
28. END IF
29. Return popItem, END FUNC
30. FUNC Peek():
31. IF isEmpty(), throw StackEmpty
32. ELSE, return arr[front]
33. END IF, END FUNC
34. STOP

**CODE**

#include <iostream>

#include <cstdlib>

using namespace std;

class Queue {

    int \_size;

    int \_rear;

    int \_front;

    int \*arr;

    public:

    Queue(int size) {

        this->\_size = size;

        this->arr = (int\*) malloc(size \* sizeof(int));

        this->\_rear = -1;

        this->\_front = 0;

    }

    int isFull() {

        if (this->\_rear == this->\_size - 1) return 1;

        return 0;

    }

    int isEmpty() {

        if (this->\_rear == -1) return 1;

        return 0;

    }

    void Enqueue(int item) {

        if (this->isFull()) {

            cout << "Queue Overflow" << endl;

        } else {

            this->\_rear++;

            this->arr[this->\_rear] = item;

        }

    }

    void Dequeue() {

        if (this->isEmpty()) {

            cout << "Queue Underflow" << endl;

        } else {

            int poppedItem = this->arr[this->\_front];

            cout << "Popped Item: " << poppedItem << endl;

            this->\_front++;

            if (this->\_front > this->\_rear) {

                this->\_rear = -1;

                this->\_front = 0;

            }

        }

    }

    void Display() {

        cout << "[";

        for (int i=0; i<this->\_size; i++) {

            if (i < this->\_front) {

                cout << "\_";

            } else if (i <= this->\_rear) {

                cout << this->arr[i];

            } else {

                cout << "\_";

            }

            if (i != this->\_size - 1){

                cout << ", ";

            }

        }

        cout << "]" << endl;

    }

    void Peek() {

        if (this->isEmpty()) {

            cout << "Stack Underflow" << endl;

        } else {

            cout << "Front Element: " << this->arr[this->\_front] << endl;

        }

    }

};

int main() {

    int temp1, choice, size;

    cout << "Enter size of Queue: ";

    cin >> size;

    Queue q(size);

    while (1) {

        cout << "------ Queue Data Structure ------" << endl;

        cout << "1. Enqueue" << endl;

        cout << "2. Dequeue" << endl;

        cout << "3. Peek" << endl;

        cout << "4. Display" << endl;

        cout << "5. Exit" << endl;

        cout << "Enter Choice: ";

        cin >> choice;

        if (choice == 1) {

            cout << "Enter item to insert: ";

            cin >> temp1;

            q.Enqueue(temp1);

        } else if (choice == 2) {

            q.Dequeue();

        } else if (choice == 3) {

            q.Peek();

        } else if (choice == 4) {

            q.Display();

        } else if (choice == 5) {

            cout << "\n--------- Author ----------------" << endl;

            cout << "Ali Izzath Shazin" << endl;

            cout << "220071601028" << endl;

            cout << "B. Tech CSE A" << endl;

            break;

        } else {

            cout << "Invalid Option" << endl;

        }

        cout << endl;

    }

    return 0;

}

**OUTPUT**

Enter size of Queue: 3

------ Queue Data Structure ------

1. Enqueue

2. Dequeue

3. Peek

4. Display

5. Exit

Enter Choice: 1

Enter item to insert: 10

------ Queue Data Structure ------

1. Enqueue

2. Dequeue

3. Peek

4. Display

5. Exit

Enter Choice: 1

Enter item to insert: 20

------ Queue Data Structure ------

1. Enqueue

2. Dequeue

3. Peek

4. Display

5. Exit

Enter Choice: 4

[10, 20, \_]

------ Queue Data Structure ------

1. Enqueue

2. Dequeue

3. Peek

4. Display

5. Exit

Enter Choice: 2

Popped Item: 10

------ Queue Data Structure ------

1. Enqueue

2. Dequeue

3. Peek

4. Display

5. Exit

Enter Choice: 3

Front Element: 20

------ Queue Data Structure ------

1. Enqueue

2. Dequeue

3. Peek

4. Display

5. Exit

Enter Choice: 5

--------- Author ----------------

Ali Izzath Shazin

220071601028

B. Tech CSE A