DSA Experiment No.5

Experiment No (5)

AIM – A double ended queue (deque) is a linear list in which addition and deletion may be made at either end. Obtain a data representation mapping a deque into 1D array. Write a C++ program to simulate deque with function to add and delete elements from either end of the queue.

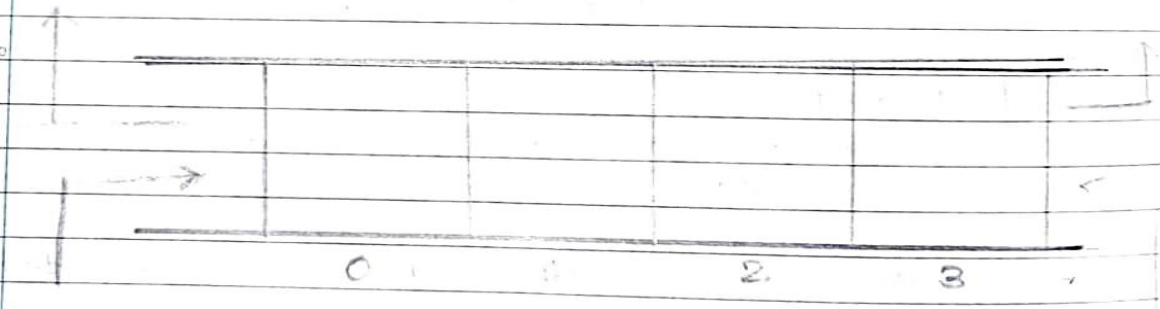
Theory :

Double Ended Queue

Deque.

The Deque stands for double ended queue. In the queue insertion takes place from one end and deletion from other end.

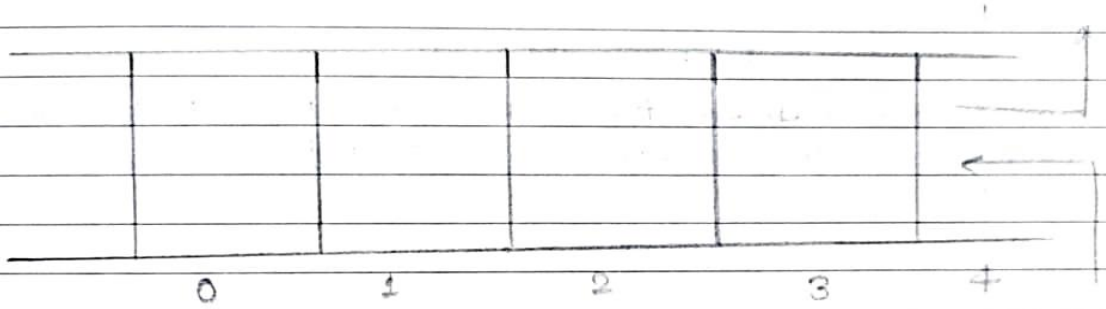
The end where insertion occurs known as the rear end. whereas the end at which the deletion occurs known as front end.



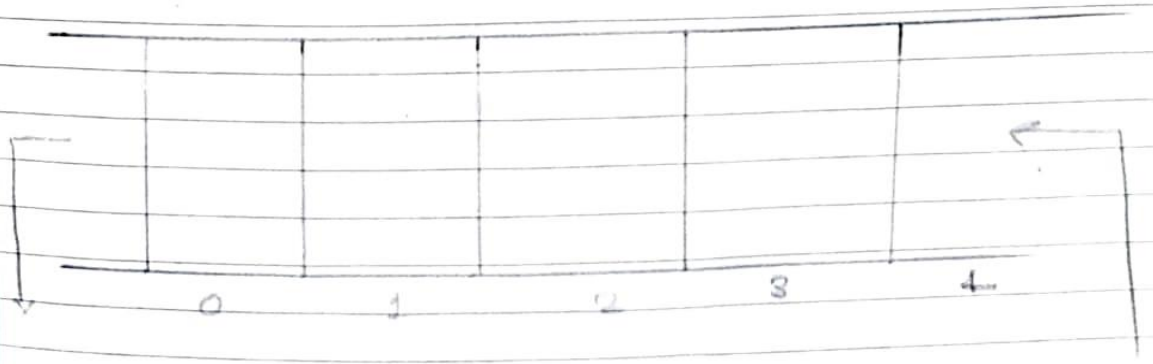
- Deque can be used both as stack and queue as it allows the insertion and deletion operation on both ends.

rudraksh.karpe.cs@ghrcem.raisoni.net

In stack LIFO rule is followed where both insertion and deletion can be performed from only one end. In deque, the insertion and deletion can be performed from one side, thus we conclude that deque can be considered as stack.



In deque the deletion can be performed from one end and insertion can be from another end. Whereas, the queue follows the FIFO rule in ~~the~~ which element is inserted at one end and deleted from another end. This makes the dequeue as a queue.

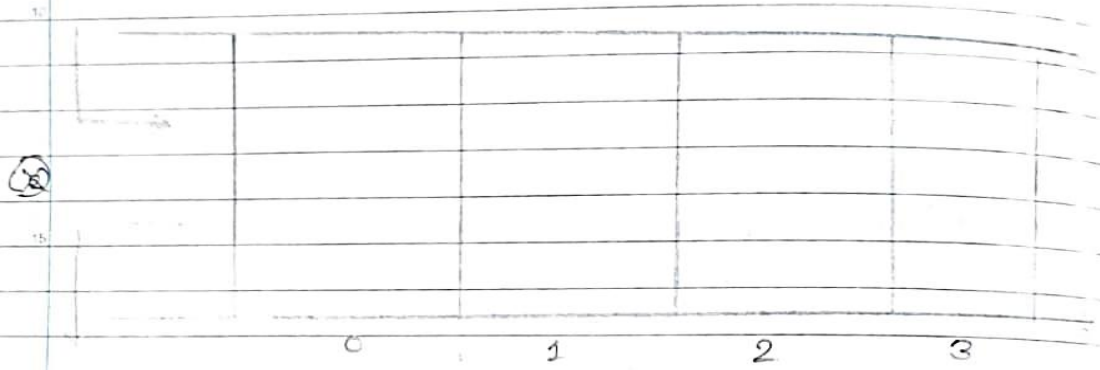


Camlin

There are two types of queues, Input restricted queue and Output restricted queue.

(1) Input restricted queue →

- The input-restricted queue means some restrictions are applied to the insertion.
- In this method, the insertion is applied to the one end while the deletion is applied from both the ends



(2) Output restricted queue →

- The restrictions are applied to the deletion operation.
- The deletion can be applied from only one end, whereas the insertion is possible from both ends.

rudraksh.karpe.cs@ghrcem.raisoni.net

Page :
Date :

Operations on Deque →

* Insert at front
* Delete from end
* Insert at rear
* Delete from rear

Peek can also be performed instead insertion and deletion. Through peek operation, we can get the front and rear element of deque.

We can perform two or more operations on dequeue →

* isFull () : True value is returned when stack is full else false.
* isEmpty() : True value is returned when stack is empty else false.

Applications of Deque :

(*) The deque can be used as a stack and queue; therefore, it can perform both redo and undo operations.

(*) It can be used as palindrome checker means that if we read the string from both ends, then the string would be the same.

Camlin

rudraksh.karpe.cs@ghrcem.raisoni.net

Program code:

```
#include <iostream>
using namespace std;
#define SIZE 5

class dequeue
{
   int a[10], front, rear, count;

public:
   dequeue();
   void add_at_beg(int);
   void add_at_end(int);
   void delete_fr_front();
   void delete_fr_rear();
   void display();
};

dequeue::dequeue()
{
   front = -1;
   rear = -1;
   count = 0;
}

void dequeue::add_at_beg(int item)
{
   int i;
   if (front == -1)
   {
      front++;
      rear++;
      a[rear] = item;
      count++;
   }
   else if (rear >= SIZE - 1)
```

```
    {
        cout << "\nInsertion is not possible,overflow!!!!";
    }
    else
    {
        for (i = count; i >= 0; i--)
        {
            a[i] = a[i - 1];
        }
        a[i] = item;
        count++;
        rear++;
    }
}
void dequeue::add_at_end(int item)
{

    if (front == -1)
    {
        front++;
        rear++;
        a[rear] = item;
        count++;
    }
    else if (rear >= SIZE - 1)
    {
        cout << "\nInsertion is not possible,overflow!!!";
        return;
    }
    else
    {
        a[++rear] = item;
    }
}

void dequeue::display()
{
```

```
   for (int i = front; i <= rear; i++)
   {
      cout << a[i] << " ";
   }
}

void dequeue::delete_fr_front()
{
   if (front == -1)
   {
      cout << "Deletion is not possible:: Dequeue is empty";
      return;
   }
   else
   {
      if (front == rear)
      {
         front = rear = -1;
         return;
      }
      cout << "The deleted element is " << a[front];
      front = front + 1;
   }
}

void dequeue::delete_fr_rear()
{
   if (front == -1)
   {
      cout << "Deletion is not possible:Dequeue is empty";
      return;
   }
   else
   {
      if (front == rear)
      {
```

```
        front = rear = -1;
      }
      cout << "The deleted element is " << a[rear];
      rear = rear - 1;
   }
}

int main()
{
   int c, item;
   dequeue d1;

   do
   {
      cout << "\n\n---DEQUEUE OPERATION---\n";
      cout << "\n1-Insert at beginning";
      cout << "\n2-Insert at end";
      cout << "\n3_Display";
      cout << "\n4_Deletion from front";
      cout << "\n5-Deletion from rear";
      cout << "\n6_Exit";
      cout << "\nEnter your choice<1-4>:";
      cin >> c;

      switch (c)
      {
      case 1:
         cout << "Enter the element to be inserted:";
         cin >> item;
         d1.add_at_beg(item);
         break;

      case 2:
         cout << "Enter the element to be inserted:";
         cin >> item;
         d1.add_at_end(item);
         break;
```

```
        case 3:
            d1.display();
            break;

        case 4:
            d1.delete_fr_front();
            break;
        case 5:
            d1.delete_fr_rear();
            break;

        case 6:
            exit(1);
            break;

        default:
            cout << "Invalid choice";
            break;
        }

    } while (c != 7);
    return 0;
}
```

Output of the program:

Insertion of element at beginning and end:

```
PS R:\GHRCEM\DSA Lab\Assignment 4> cd "r:\GHRCEM\DSA Lab\Assignment 4\" ; if ($?) { g++ tempCodeRunnerFile.cpp -o tempCodeRunnerFile } ; if ($?) { .\tempCodeRunnerFile }

Enter your choice<1-4>:1
Enter the element to be inserted:2


---DEQUEUE OPERATION---

1-Insert at beginning
2-Insert at end
3_Display
4_Deletion from front
5-Deletion from rear
6_Exit
Enter your choice<1-4>:2
Enter the element to be inserted:4


---DEQUEUE OPERATION---

1-Insert at beginning
2-Insert at end
3_Display
4_Deletion from front
5-Deletion from rear
6_Exit
Enter your choice<1-4>:3
2 4

---DEQUEUE OPERATION---
```

Deletion from front and rear:

```
---DEQUEUE OPERATION---

1-Insert at beginning
2-Insert at end
3_Display
4_Deletion from front
5-Deletion from rear
6_Exit
Enter your choice<1-4>:4
The deleted element is 2

---DEQUEUE OPERATION---

1-Insert at beginning
2-Insert at end
3_Display
4_Deletion from front
5-Deletion from rear
6_Exit
Enter your choice<1-4>:5
The deleted element is 0

---DEQUEUE OPERATION---

1-Insert at beginning
2-Insert at end
3_Display
4_Deletion from front
5-Deletion from rear
6_Exit
Enter your choice<1-4>:3


---DEQUEUE OPERATION---

1-Insert at beginning
2-Insert at end
3_Display
4_Deletion from front
5-Deletion from rear
6_Exit
Enter your choice<1-4>:6
PS R:\GHRCEM\DSA Lab\Assignment 4>
```

rudraksh.karpe.cs@ghrcem.raisoni.net