Laporan Praktikum

Struktur Data



Disusun Oleh :

**Irgi Fatihul Ihsan (2311533010)**

Dosen Pengampu : Dr. Wahyudi, S.T, M.T

Departemen Informatika

Fakultas Teknologi Informasi

Universitas Andalas

Tahun 2024

**Queue**

1. **Tujuan Praktikum**
2. Memahami penggunaan Queue untuk struktur data pada bahasa pemograman Java
3. **Pendahuluan**

Another fundamental data structure is the ***queue***. It is a close “cousin” of the stack, but a queue is a collection of objects that are inserted and removed according to the ***first-in, first-out*** (***FIFO***) principle. That is, elements can be inserted at any time, but only the element that has been in the queue the longest can be next removed.

We usually say that elements enter a queue at the back and are removed from the front. A metaphor for this terminology is a line of people waiting to get on an amusement park ride. People waiting for such a ride enter at the back of the line and get on the ride from the front of the line. There are many other applications of queues.

1. **Metode Praktikum**
2. **Array-Based Queue**

Program berikut merupakan queue yang dibuat menggunakan basis array.

public class inputQueue {

int front, rear, size;

int capacity;

int array[];

public inputQueue (int capacity) {

this.capacity = capacity;

front = this.size = 0;

rear = capacity - 1;

array = new int[this.capacity];

}

boolean isFull (inputQueue queue) {

return (queue.size == queue.capacity);

}

boolean isEmpty(inputQueue queue) {

return (queue.size == 0);

}

void enqueue(int item) {

if(isFull(this))

return;

this.rear = (this.rear + 1) % this.capacity;

this.array[this.rear] = item;

this.size = this.size + 1;

System.***out***.println(item + " enqueue to queue");

}

int dequeue() {

if (isEmpty(this))

return Integer.***MIN\_VALUE***;

int item = this.array[this.front];

this.front = (this.front + 1) % this.capacity;

this.size = this.size - 1;

return item;

}

int front() {

if(isEmpty(this))

return Integer.***MIN\_VALUE***;

return this.array[this.front];

}

int rear() {

if (isEmpty(this))

return Integer.***MIN\_VALUE***;

return this.array[this.rear];

}

}

Method isFull mengembalikan nilai boolean apakah kapasitas dari array yang digunakan sebagai basis Queue sudah penuh. Method isEmpty mengembalikan nilai boolean apakah queue kosong. Method enqueue untuk menambahkan elemen ke queue, sedangkan dequeue untuk mengembalikan elemen dari queue. Terakhir, method front mengembalikan nilai terdepan atau nilai yang pertama di-enqueue, dan method rear mengembalikan nilai terakhir dari queue.

Berikut contoh program yang mengimplementasikan array-based queue yang telah dibuat di kelas inputQueue:

public static void main(String[] args) {

inputQueue queue = new inputQueue(1000);

queue.enqueue(10);

queue.enqueue(20);

queue.enqueue(30);

queue.enqueue(40);

System.***out***.println("Front item is " + queue.front());

System.***out***.println("Rear item is " + queue.rear());

System.***out***.println(queue.dequeue() + " dequeue from queue");

System.***out***.println("Front item is " + queue.front());

System.***out***.println("Rear item is " + queue.rear());

}

Output dari program tersebut adalah sebagai berikut:

10 enqueue to queue

20 enqueue to queue

30 enqueue to queue

40 enqueue to queue

Front item is 10

Rear item is 40

10 dequeue from queue

Front item is 20

Rear item is 40

1. **Membalikkan isi queue (reverse)**

Program berikut membalikkan isi dari sebuah queue:

public static void main(String[] args) {

Queue<Integer> q = new LinkedList<Integer>();

q.add(1);

q.add(2);

q.add(3);

System.***out***.println("sebelum reverse " + q);

Stack<Integer> s = new Stack<Integer>();

while (!q.isEmpty()) {

s.push(q.remove());

}

while (!s.empty()) {

q.add(s.pop());

}

System.***out***.println("sesudah reverse= " + q);

}

Untuk membalikkan isi dari queue diperlukan sebuah stack untuk menyimpan elemen-elemen dari queue. Karena prinsip LIFO pada stack, maka jika elemen-elemen queue dikeluarkan, dan dimasukkan ke dalam stack, lalu dikeluarkan kembali dari stack, maka elemen-elemen dari queue tersebut akan terbalik.

Berikut output dari program tersebut:

sebelum reverse [1, 2, 3]

sesudah reverse= [3, 2, 1]

1. **Kesimpulan Praktikum**

Queue adalah kumpulan objek yang dimasukkan dan dikeluarkan dari queue tersebut, berdasarkan prinsip ***first-in, first out (FIFO).*** Yang artinya, elemen bisa dimasukkan kapan saja, tapi hanya elemen yang paling lama berada dalam queue atau elemen yang pertama dimasukkan yang bisa dikeluarkan dari queue.