**LAPORAN AWAL STRUKTUR DATA**

**Pertemuan ke-8**

**Double Ended Queue**

****

Disusun oleh:

Nama : ZAHARA NAULI IBRAHIM

NIM : 241011402500

Kelas : 03TPLP023

**Program Studi Teknik Informatika**

**Fakultas Ilmu Komputer**

**Universitas Pamulang**

Jl. Surya Kencana No. 1 Pamulang Telp (021)7412566, Fax. (021)7412566

Tangerang Selatan – Banten

# Landasan Teori

Dalam dunia pemrograman, queue (antrian) merupakan salah satu struktur data linear yang penting dalam pengelolaan data secara berurutan. Queue menggunakan prinsip FIFO (First In, First Out), artinya elemen pertama yang masuk akan menjadi elemen pertama yang keluar.

Untuk kebutuhan yang lebih fleksibel, digunakan Double Ended Queue (Deque). Deque adalah pengembangan dari queue linear yang memungkinkan penambahan (insert) dan penghapusan (delete) data dari kedua ujung antrian, yaitu depan (front) dan belakang (rear).

Konsep Deque mirip dengan situasi nyata, seperti lorong antrean fleksibel: orang dapat masuk atau keluar dari depan maupun belakang sesuai kebutuhan. Deque digunakan pada sistem komputer yang membutuhkan fleksibilitas pengelolaan data, antara lain:

* Sistem undo-redo pada aplikasi editor (misal text editor, spreadsheet)
* Penjadwalan tugas dengan prioritas ganda
* Algoritma sliding window pada pemrosesan data atau array
* Pengecekan palindrome atau string yang dapat dibaca dari dua arah

## 1. Pengertian Double Ended Queue (Deque)

Deque adalah struktur data antrian ganda di mana elemen dapat ditambahkan (insert) atau dihapus (delete) dari kedua ujung antrian. Deque menggabungkan prinsip FIFO dan kemampuan manipulasi dari dua sisi, sehingga lebih fleksibel dibanding queue biasa.

## 2. Ciri-ciri Deque

* Operasi dapat dilakukan di front dan rear.
* Memiliki dua pointer utama: front dan rear.
* Bisa diimplementasikan menggunakan array (ukuran tetap) atau linked list (dinamis).

## 3. Jenis-jenis Deque

1. Input-Restricted Deque – Penambahan hanya di satu ujung, penghapusan di kedua ujung.
2. Output-Restricted Deque – Penghapusan hanya di satu ujung, penambahan di kedua ujung.
3. General Deque – Bisa menambahkan dan menghapus data dari kedua ujung secara bebas.

## 4. Operasi Dasar Deque

* insertFront() → menambah elemen di depan
* insertRear() → menambah elemen di belakang
* deleteFront() → menghapus elemen dari depan
* deleteRear() → menghapus elemen dari belakang
* isEmpty() → mengecek apakah deque kosong
* isFull() → mengecek apakah deque penuh (khusus array)

## 5. Implementasi Deque dengan Array

* Menggunakan dua pointer: front dan rear.
* Biasanya dibuat circular deque agar ruang kosong di awal array tetap bisa digunakan.

## 6. Implementasi Deque dengan Linked List

* Node menyimpan data dan pointer ke node sebelumnya dan berikutnya.
* Ukuran deque bisa dinamis, menyesuaikan jumlah data.

## 7. Kelebihan Deque

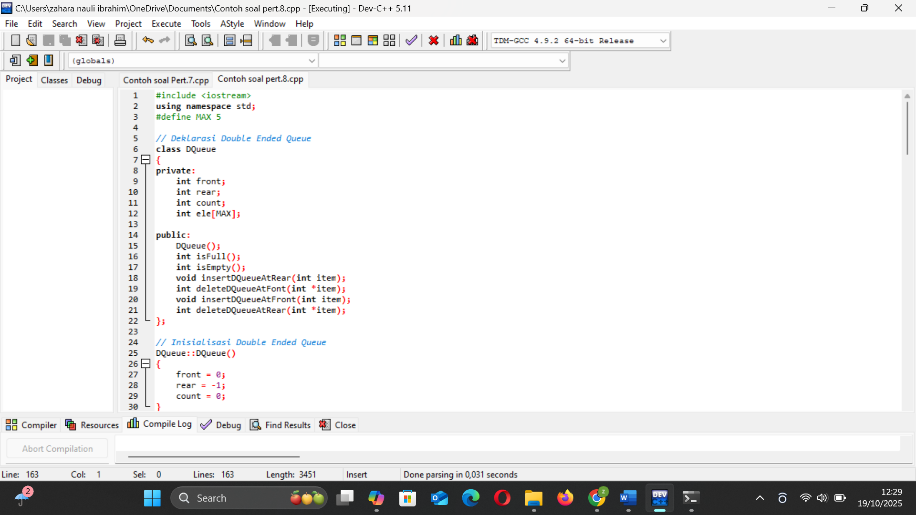
* Operasi fleksibel karena bisa menambah/hapus dari kedua ujung.
* Cocok untuk palindrome checking, sliding window, dan penjadwalan tugas.

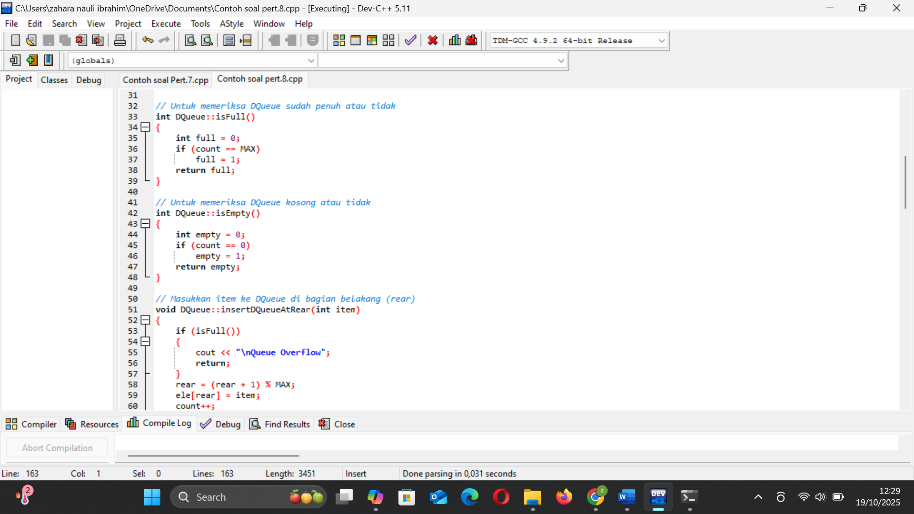
## 8. Kekurangan Deque

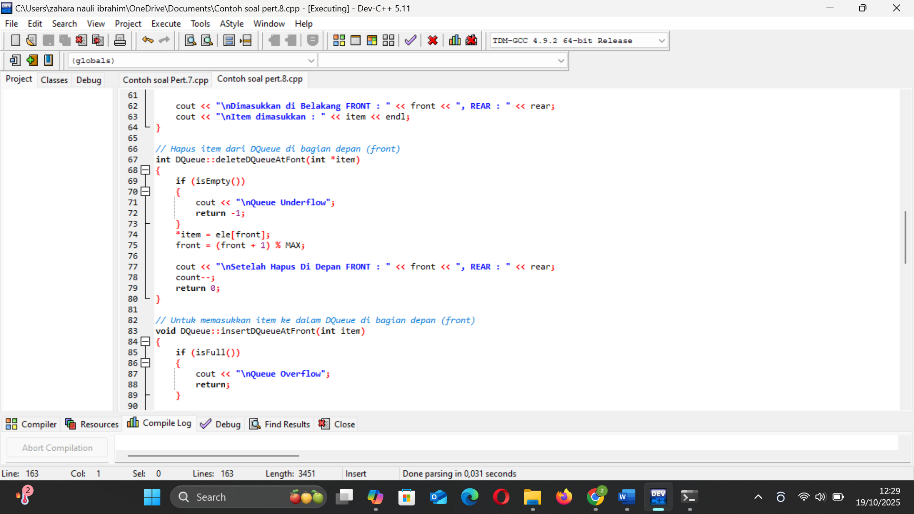
* Implementasi array memerlukan manajemen circular agar efisien.
* Implementasi linked list membutuhkan memori tambahan untuk pointer antar-node.

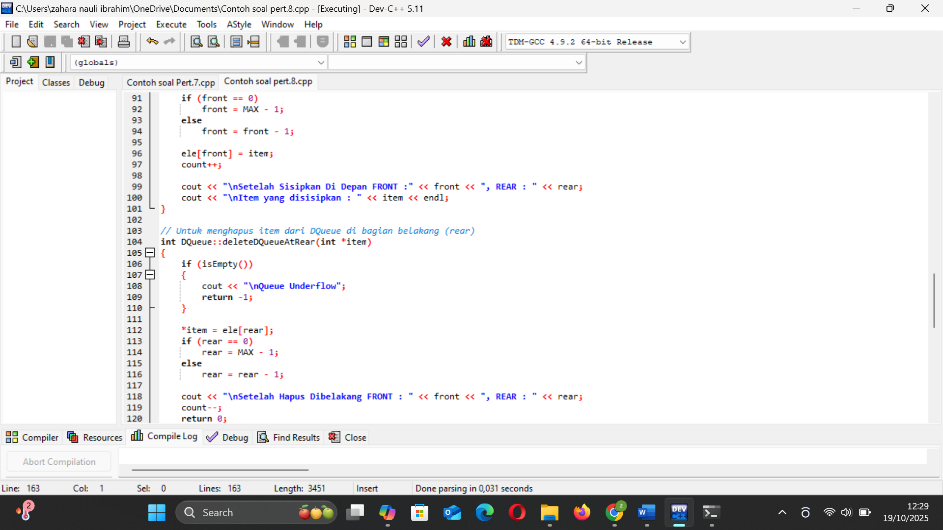
# LAPORAN PRAKTIKUM

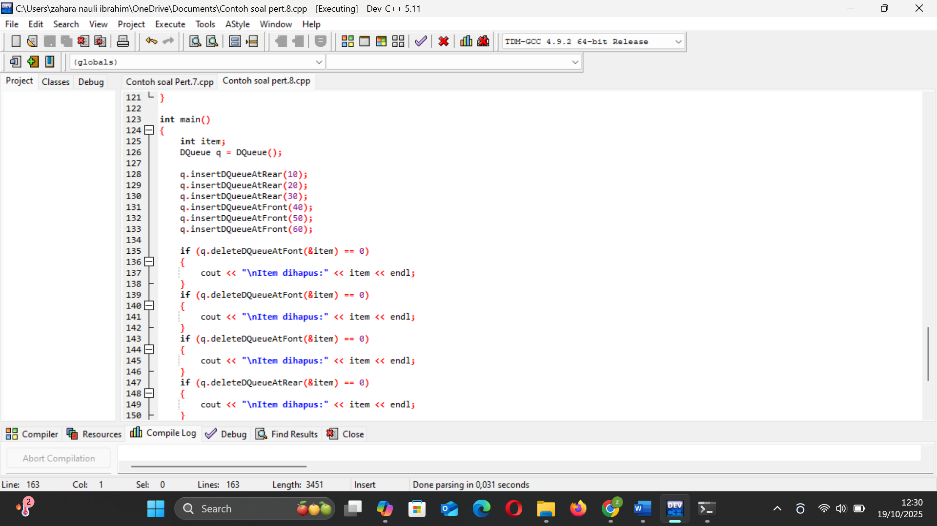
# **1) Program Implementasi DeQueue**

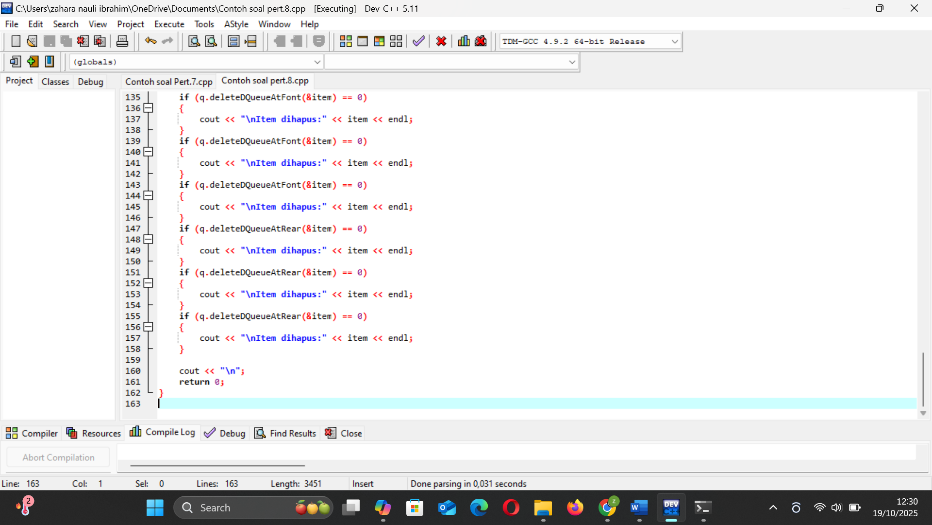
****



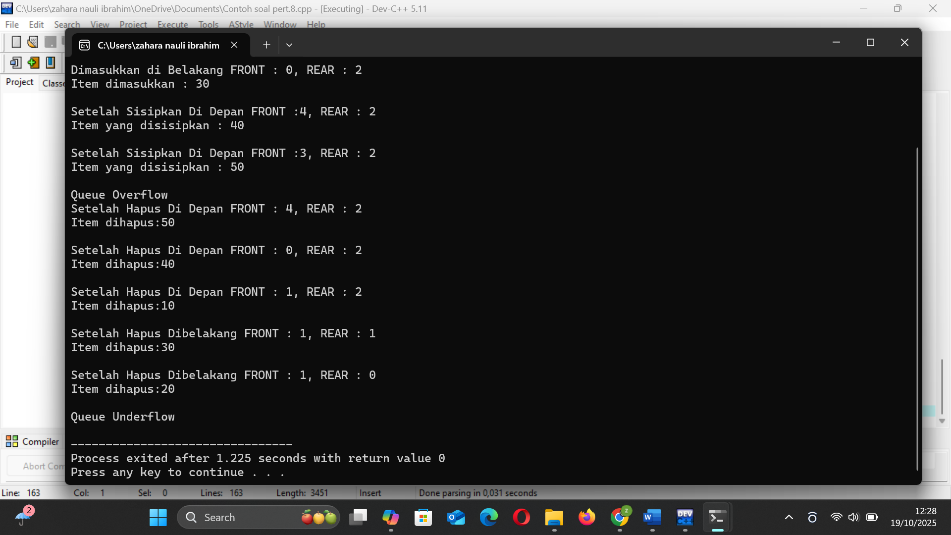








Hasil Output

****

LATIHAN 8

1) **Tulis algoritma dasar double ended queue untuk kondisi: a. Inisialisasi b. Insert sebuah record dari kanan c. Insert sebuah record dari kiri d. Delete sebuah record dari kanan e. Delete sebuah record dari kiri**

Double Ended Queue (Deque) adalah antrian yang bisa menambah (insert) atau menghapus (delete) data dari dua sisi: kiri (depan) dan kanan (belakang).

Penjelasan

Gunakan simbol:

* F → Front (depan)
* R → Rear (belakang)
* MAX → ukuran maksimum deque
* DEQUE[MAX] → array tempat data disimpan

1. Inisialisasi

* Tujuan: Mengatur kondisi awal deque agar siap digunakan.
* Algoritma: Algoritma InisialisasiDeque
* F ← -1
* R ← -1
* Selesai

Penjelasan:

* F dan R diisi -1 artinya deque kosong.
* Belum ada data, jadi posisi awal belum ditentukan.

Kesimpulan 1a:

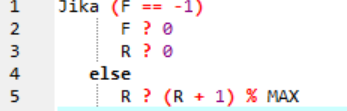
Langkah ini seperti “mengosongkan antrian” agar siap diisi nanti.

1. Insert dari Kanan (Rear)

* Algoritma: Algoritma InsertKanan(data)

1. Jika ((R + 1) % MAX == F)

* Cetak "Deque penuh, tidak bisa insert dari kanan"

1. 
2. DEQUE[R] ← data
3. Selesai

Penjelasan:

* Cek apakah deque penuh.
* Kalau masih kosong, posisi awal diisi F = R = 0.
* Kalau tidak, R maju satu posisi melingkar.
* Data baru disimpan di R.

Kesimpulan 1b:

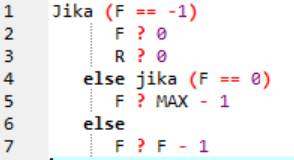
Menambah dari kanan artinya memasukkan data di “belakang” antrian.

1. Insert dari Kiri (Front)

* Algoritma: Algoritma InsertKiri(data)

1. Jika ((F == 0 dan R == MAX - 1) atau (F == R + 1))

* Cetak "Deque penuh, tidak bisa insert dari kiri"

1. 
2. DEQUE[F] ← data
3. Selesai

Penjelasan:

* Kalau penuh, tidak bisa insert.
* Kalau kosong, posisi awal diisi F = R = 0.
* Kalau F di posisi awal (0), maka mundur ke akhir (melilit).
* Data disimpan di posisi F.

Kesimpulan 1c:

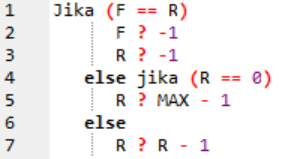
Menambah dari kiri seperti menambahkan orang di barisan paling depan.

1. Delete dari Kanan

* Algoritma: Algoritma DeleteKanan

1. Jika (F == -1)

* Cetak "Deque kosong, tidak bisa delete dari kanan"

1. Ambil data ← DEQUE[R]
2. 
3. Selesai

Penjelasan:

* Kalau kosong, tidak bisa hapus.
* Kalau hanya satu data, setelah dihapus jadi kosong.
* Kalau tidak, R mundur satu posisi melingkar.

Kesimpulan 1d:

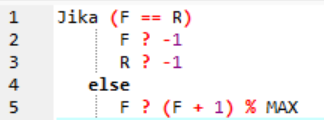
Delete kanan artinya menghapus data paling belakang.

1. Delete dari Kiri

* Algoritma: Algoritma DeleteKiri

1. Jika (F == -1)

* Cetak "Deque kosong, tidak bisa delete dari kiri"

1. Ambil data ← DEQUE[F]
2. 
3. Selesai

Penjelasan:

* Kalau kosong, tidak bisa dihapus.
* Kalau hanya satu data, setelah dihapus jadi kosong.
* Kalau lebih dari satu, F maju satu langkah ke depan.

Kesimpulan 1e:

Delete kiri artinya menghapus orang paling depan dalam antrian.

2) **Sebutkan ciri bahwa double ended queue pada: a. Kosong tidak ada isinya b. Penuh kanan, tak bisa diisi dari kanan c. Penuh kiri, tidak bisa diisi dari kiri d. Penuh total, tidak bisa diisi dari kiri maupun dari kanan e. Hanya diisi 10 pengantri.**

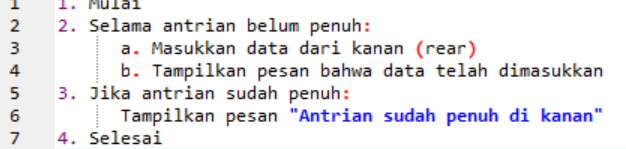
|  |  |  |
| --- | --- | --- |
| **Kondisi** | **Ciri-Ciri** | **Penjelasan Sederhana** |
| **a.** Kosong | F = -1 dan R = -1 | Tidak ada data sama sekali. |
| **b.** Penuh Kanan | (R + 1) % MAX == F | Tidak bisa lagi insert dari kanan. |
| **c.** Penuh Kiri | (F == 0 dan R == MAX - 1) atau (F == R + 1) | Tidak bisa lagi insert dari kiri. |
| **d.** Penuh Total | (R + 1) % MAX == F | Kanan dan kiri dua-duanya penuh. |
| **e.** Berisi 10 Pengantri | Jumlah elemen = 10 | Bisa dihitung dari posisi F dan R. |
|  |  |  |

Kesimpulan 2:

Kondisi deque ditentukan oleh hubungan antara Front dan Rear, apakah mereka berdekatan (penuh) atau sama-sama -1 (kosong).

3) Tulis algoritma yang lengkap untuk mengisi antrian dari kanan record per record sampai antrian penuh kanan, atau tidak bisa diisi lagi dari kanan.

Algoritma



Penjelasan prosesnya :

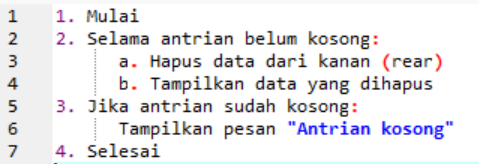
* Program akan terus menambah data dari bagian kanan (rear).
* Setiap kali menambahkan data, posisi rear akan bergeser ke kanan (atau melingkar jika circular queue).
* Proses ini terus berulang sampai antrian mencapai kapasitas maksimal (penuh).
* Saat penuh, sistem akan menghentikan proses dan memberi tahu bahwa tidak bisa menambah lagi dari kanan.

**Kesimpulan:**

Proses pengisian dari kanan akan berjalan selama masih ada ruang di antrian.  
Begitu antrian penuh, tidak ada data baru yang bisa ditambahkan lagi dari sisi kanan.

4.) Tulis algoritma lengkap untuk mendelete isi antrian dari kanan record per record sampai antrian kosong

Algoritma



**Penjelasan prosesnya:**

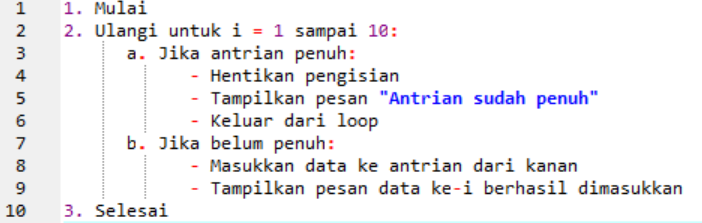
* Program akan terus menghapus data dari sisi kanan (rear) satu per satu.
* Setiap kali data dihapus, posisi rear akan bergeser ke kiri (atau ke elemen sebelumnya).
* Proses akan berhenti ketika tidak ada lagi data di dalam antrian (count == 0).
* Setelah itu akan muncul pesan bahwa antrian kosong.

**Kesimpulan:**

Proses penghapusan dari kanan dilakukan sampai semua data hilang dari antrian.  
Kalau antrian sudah kosong, tidak bisa dihapus lagi karena tidak ada data yang tersisa.

5.) Tulis algoritma yang lengkap untuk mengisi antrian dari kanan record per record sebanyak 10 record selama antrian

Algoritma



**Penjelasan prosesnya:**

* Program mencoba memasukkan 10 data satu per satu dari kanan.
* Tapi sebelum setiap pemasukan, program memeriksa dulu apakah antrian sudah penuh.
* Jika belum penuh → data dimasukkan.
* Jika sudah penuh di tengah jalan → proses langsung dihentikan, walaupun belum mencapai 10 data.

**Kesimpulan:**

Proses pengisian berhenti otomatis ketika antrian penuh, walaupun belum 10 record.  
Artinya, sistem tidak akan memaksa mengisi lebih dari kapasitas maksimal.

**LAPORAN AKHIR**

****

Disusun oleh:

Nama : ZAHARA NAULI IBRAHIM

NIM : 241011402500

Kelas : 03TPLP023

**Program Studi Teknik Informatika**

**Fakultas Ilmu Komputer**

**Universitas Pamulang**

Jl. Surya Kencana No. 1 Pamulang Telp (021)7412566, Fax. (021)7412566

Tangerang Selatan – Banten

# TUGAS AKHIR

## LATIHAN 7

## 1) Sebutkan ciri circular Queue dalam kondisi:

## Kosong

## Penuh

## Bisa diisi

## Ada isinya

## Hanya berisi 10 record

## Tempat yang kosong hanya ada 10 tempat.

## Penjelasan

Misalkan:

* F = Front (penunjuk elemen depan)
* R = Rear (penunjuk elemen belakang)
* MAX = ukuran maksimum queue

|  |  |  |
| --- | --- | --- |
| **Kondisi** | **Ciri-Ciri** | **Penjelasan Sederhana** |
| **a.** Kosong | F = -1 dan R = -1 | Artinya belum ada data sama sekali. |
| **b.** Penuh | (R + 1) % MAX == F | Artinya elemen terakhir sudah diisi, dan posisi berikutnya kembali ke depan (penuh melingkar). |
| **c.** Bisa diisi | Queue tidak penuh -> (R + 1) % MAX != F | Masih ada ruang kosong untuk data baru. |
| **d.** Ada isinya | F != -1 | Berarti minimal ada satu data yang tersimpan. |
| **e.** Hanya berisi 10 record | Jumlah data = 10 | Bisa dihitung dari posisi F dan R. |
| **f.** Tempat kosong hanya ada 10 tempat | (MAX - jumlah data) = 10 | Sisa ruang kosong di queue tinggal 10 elemen. |

**Kesimpulan 1:**

Circular Queue seperti antrian yang ujungnya menyambung ke depan. Ketika penuh, Rear akan “mengejar” Front, dan ketika kosong, keduanya sama-sama -1.

## 2) Tulis Algoritma lengkap untuk:

### Algoritma INSERT Sebuah Record

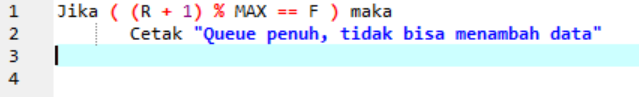
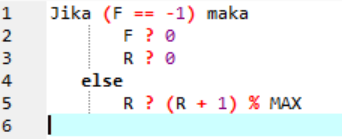
### Algoritma DELETE Sebuah Record

## Algoritma

1. Insert (Enqueue)

* Tujuan: Menambahkan data ke antrian.
* Algoritma:

Algoritma InsertCircularQueue

1. 
2. 
3. Isi Queue[R] dengan data baru
4. Selesai

**Penjelasan:**

* Langkah 1: Cek apakah penuh (tidak bisa diisi lagi).
* Langkah 2: Kalau queue kosong, set awal F = 0 dan R = 0.
* Langkah 3: Kalau tidak kosong, posisi R maju satu langkah secara melingkar.
* Langkah 4: Simpan data di posisi R.

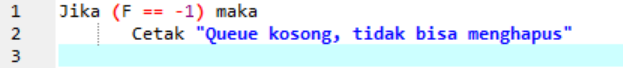
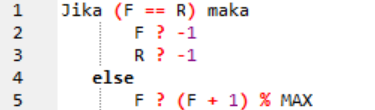
**Kesimpulan 2a:**

Insert di Circular Queue seperti memarkir mobil di antrian yang berputar — kalau penuh, kita berhenti; kalau masih ada ruang, kita isi di posisi berikutnya.

1. Delete (Dequeue

* Tujuan: Menghapus data dari antrian (dari depan).
* Algoritma:

Algoritma DeleteCircularQueue

1. 
2. Ambil data di Queue[F]
3. 
4. Selesai

**Penjelasan:**

* Langkah 1: Kalau kosong, tidak bisa dihapus.
* Langkah 2: Ambil data di depan (Front).
* Langkah 3: Kalau setelah dihapus queue jadi kosong (Front = Rear), set ulang ke -1.
* Langkah 4: Kalau tidak, geser Front ke depan secara melingkar.

**Kesimpulan 2b:**

Delete di Circular Queue seperti memanggil mobil di urutan terdepan keluar dari antrian.

## 3) Tulis Algoritma untuk mengisi antrian record per record sebanyak 10 record (Selama Belum Penuh)

## Penjelasan

**Algoritma:**

**Algoritma Isi10Record**

1. i ← 1
2. Ulangi selama (i ≤ 10) DAN (Queue belum penuh)
3. Baca data
4. Panggil InsertCircularQueue(data)
5. i ← i + 1
6. Jika Queue penuh sebelum 10 record terisi

* Cetak "Queue sudah penuh, pengisian dihentikan"

1. Selesai

**Penjelasan:**

* Langkah 1: Mulai dari record ke-1.
* Langkah 2: Selama belum penuh dan belum 10 kali, terus tambahkan data.
* Langkah 3: Kalau penuh di tengah, hentikan walau belum sampai 10.

**Kesimpulan 3:**

Algoritma ini memastikan pengisian berhenti otomatis ketika penuh — jadi tidak akan ada “kelebihan muatan”.

## 4) Tulis algoritma yang lengkap untuk mendelete isi antrian record per record sebanyak 10 record selama antrian masih ada isinya. Apabila antrian sudah kosong, walaupun belum mendelete sebanyak 10 record, maka proses delete dihentikan. (Selama Masih Ada Isi)

## Penjelasan

**Algoritma:**

**Algoritma Hapus10Record**

1. i ← 1
2. Ulangi selama (i ≤ 10) DAN (Queue belum kosong)
3. Panggil DeleteCircularQueue
4. i ← i + 1
5. Jika Queue kosong sebelum 10 record terhapus

* Cetak "Queue sudah kosong, penghapusan dihentikan"

1. Selesai

**Penjelasan:**

* Langkah 1: Mulai hitung dari record pertama yang akan dihapus.
* Langkah 2: Selama queue masih ada isinya, hapus satu per satu.
* Langkah 3: Kalau queue kosong duluan, hentikan.

**Kesimpulan 4:**

Proses delete ini seperti memanggil 10 mobil dari antrian keluar satu-satu, tapi kalau sudah habis, langsung berhenti.

## 5) Tulis untuk menghitung dan mencetak jumlah tempat(elemen) yang ada isinya bila diketahui nilai F dan R tanpa mengetahui nilai Counter.

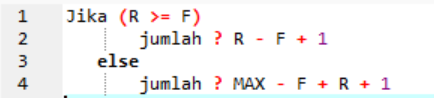
## Penjelasan

**Diketahui:** F dan R

**Ditanya:** Berapa jumlah elemen (isi Queue) sekarang?

**Algoritma:**

Algoritma HitungIsiQueue

1. 
2. 
3. Cetak "Jumlah elemen yang berisi =", jumlah
4. Selesai

**Penjelasan:**

* Jika kosong (F = -1), jumlahnya 0.
* Kalau posisi R lebih besar dari F, berarti belum melingkar.
* Kalau R lebih kecil dari F, berarti sudah melingkar, jadi hitung dari F ke akhir dan tambah dari awal ke R.
* Kesimpulan 5:
* Dengan hanya mengetahui Front dan Rear, kita masih bisa tahu berapa banyak data yang ada tanpa variabel penghitung.

**Kesimpulan Akhir :**

Circular Queue adalah bentuk antrian yang ujung belakangnya menyambung ke depan, sehingga lebih efisien dalam penggunaan ruang. Insert menambah data dari belakang, Delete menghapus dari depan. Kondisi penuh dan kosong bergantung pada posisi Front dan Rear. Proses pengisian dan penghapusan bisa diatur agar otomatis berhenti ketika sudah penuh atau kosong. Jumlah elemen bisa dihitung hanya dengan F dan R, tanpa perlu variabel tambahan.

# HASIL DAN PEMBAHASAN

## LATIHAN 7

## Soal 1 Program Inisialisasi dan Pengisian Circular Queue

Melatih proses mengisi data ke dalam Circular Queue secara berulang menggunakan pergeseran indeks yang melingkar.

### Pembahasan:

* Program mendeklarasikan array Queue[5] sebagai antrian.
* Nilai front = -1 dan rear = -1 menandakan antrian kosong.
* Setiap kali menambah data, rear akan berpindah dengan rumus (rear + 1) % 5.
* Jika front masih -1, maka diubah menjadi 0 saat data pertama masuk.
* Ketika rear bertemu front (sama nilai mod 5), antrian dianggap penuh.

### Kesimpulan:

Circular Queue memungkinkan proses antrian berulang tanpa harus menggeser elemen secara manual, cukup dengan operasi modulus pada indeks.

## Soal 2 Program Dasar Operasi Circular Queue (Insert, Delete, Reset)

Memahami cara kerja dasar operasi Circular Queue dengan kontrol kondisi penuh dan kosong.

### Pembahasan:

* Inisialisasi: mengatur front dan rear ke -1.
* Insert: menambah data ke posisi (rear + 1) % ukuran. Jika penuh, ditolak.
* Delete: menghapus data dari posisi front. Jika kosong, tidak bisa dihapus.
* Reset: mengembalikan front dan rear ke -1 untuk mengosongkan antrian.
* Program menampilkan status antrian setelah tiap operasi agar terlihat perubahannya.

### Kesimpulan:

Circular Queue dapat melakukan operasi insert dan delete secara efisien tanpa menggeser semua elemen seperti pada Linear Queue.

## Soal 3 Program Menentukan Kondisi Circular Queue

Mengetahui logika pengecekan kondisi Circular Queue (kosong, penuh, dan bisa diisi kembali).

### Pembahasan:

* Program meminta input front dan rear dari pengguna.
* Logika kondisi:
* Kosong → jika front == -1.
* Penuh → jika (rear + 1) % ukuran == front.
* Bisa diisi → jika tidak memenuhi dua kondisi di atas.
* Menjelaskan hubungan antara posisi front dan rear yang berputar secara melingkar.

### Kesimpulan:

Kondisi Circular Queue tidak hanya bergantung pada besar indeks, tetapi juga hubungan antara front dan rear yang mengikuti arah sirkular.

## Soal 4 Program Menampilkan Data Circular Queue

Melatih penampilan isi antrian melingkar sesuai urutan elemen aktif dari depan ke belakang.

### Pembahasan:

* Data ditampilkan mulai dari posisi front hingga rear dengan rumus (i + 1) % ukuran.
* Proses berjalan terus sampai seluruh isi antrian yang valid tercetak.
* Jika front == -1, maka antrian dianggap kosong dan tidak ada data yang ditampilkan.

Makna Deklarasi:

Perulangan bersifat modular — ketika mencapai indeks terakhir, ia kembali ke awal (0), menciptakan efek melingkar.

### Kesimpulan:

Menampilkan Circular Queue membutuhkan logika perulangan modular agar data yang “berputar” tetap tampil sesuai urutan antrian sebenarnya.

## Soal 5 Program Menghitung Jumlah Data dan Ruang Kosong Circular Queue

Mempelajari rumus perhitungan banyak data dan sisa ruang kosong pada antrian melingkar.

### Pembahasan:

* Kapasitas antrian dideklarasikan misalnya n = 10.
* Rumus jumlah data aktif:

(rear - front + n) % n + 1

* Rumus sisa ruang kosong:

(front - rear + n - 1) % n

* Program menghitung dan menampilkan hasil berdasarkan nilai front dan rear yang dimasukkan pengguna.

### Kesimpulan:

Perhitungan jumlah data pada Circular Queue tidak dapat menggunakan rumus biasa seperti Linear Queue; harus menggunakan operasi modulus untuk menyesuaikan indeks yang melingkar.

# 

# KESIMPULAN

1. Circular Queue merupakan pengembangan dari Linear Queue dengan logika indeks yang berputar (modulus).
2. Struktur ini efisien karena tidak perlu menggeser data saat ada elemen keluar dari depan.
3. Kondisi penuh dan kosong bergantung pada hubungan antara front dan rear, bukan nilai indeks absolut.
4. Circular Queue banyak digunakan dalam sistem antrian berulang, seperti buffer data atau sistem multitasking.