**MAKALAH**

**QUEUE (ANTRIAN)**

Dibuat untuk memenuhi tugas pertama pada mata kuliah Algoritma dan Struktur Data di semester satu (I)



Dosen Pengampu : Hesmi Aria Yanti, S.Kom., M.Kom.

**Disusun Oleh:**

1. Rahmat Alfajri (2259201006)
2. Rama Apriando (2259201023)
3. Annisa Salsabila (2259201001)
4. Annisa Nur Wahidah (2259201003)
5. Rizky Satrio Wibowo (2259201005)
6. Cindiya Andharesta (2259201141)
7. Anabil Ilham Ramadhan (2259201002)

**Program Studi Sistem dan Teknologi Informasi**

**Fakultas Teknik dan Ilmu Komputer**

**Universitas Muhammadiyah Kotabumi Tahun 2022/2023**

Jl.Hasan Kepala Ratu N0.1052, Sindang Sari, Kotabmi, Lampung Utara.

34517

**KATA PENGANTAR**

Puji syukur penulis panjatkan kepada Allah SWT atas segala limpahan rahmat dan karunia-Nya penulis dapat menyelesaikan makalah yang berjudul “MATERI QUEUE(PENUTUP)”. Makalah ini terdiri dari 3 bab, yaitu bab pendahuluan, pembahasan, dan penutup, serta daftar Pustaka. Setiap bab tersebut terangkai secara komprehensif untuk membahas mengenai keseluruhan generasi computer.

Penulis sangat berharap supaya makalah ini dapat bermanfaat, khususnya bagi kami selaku mahasiswa. Kami sangat menyadari bahwa makalah ini jauh dari kata sempurna. Untuk itu, kami sangat terbuka atas kritik dan saran positif bagi Bapak/Ibu sekalian.

Terima kasih dan penghargaan kami selaku penulis sebesar-besarnya kepada yang terhormat:

1. Dosen mata kuliah Algoritma Dan Struktur Data, Ibu Hesmi Aria Yanti, S.Kom., M.Kom,

2. Seluruh pihak-pihak anggota kelompok yang mendukung kelancaran dan penyusunan makalah ini.

Semoga Allah SWT memberikan balasan atas segala amal yang telah dilakukan dan dengan pembuatan makalah ini dapat bermanfaat bagi mahasiswa Sistem dan Teknologi Informasi.

Kotabumi, 8 Oktober 2022

Penulis

**DAFTAR ISI**

**HALAMAN JUDUL** i

**KATA PENGANTAR** ii

**DAFTAR ISI**....................................................................................................................iii

**BAB I PENDAHULUAN** 1

* 1. Latar Belakang 1
  2. Rumusan Masalah 1
  3. Tujuan 2

**BAB II PEMBAHASAN** 3

**BAB I**

**PENDAHULUAN**

**1.1. Latar Belakang**

Queue/antrian adalah ordered list dengan penyisipan di satu ujung, sedang penghapusan di ujung lain. Ujung penyisipan biasa disebut rear/tail, sedang ujung penghapusan disebut front/head. Fenomena yang muncul adalah elemen yang lebih dulu disisipkan akan juga lebih dulu diambil. Queue berdisiplin FIFO (First In, First Out). Queue merupakan kasus khusus ordered list. Dengan karakteristik terbatas itu maka kita dapat melakukan optimasi representasi ADT Queue untuk memperoleh kerja paling optimal.

Karakteristik Queue memang terbatas, tetapi Queue merupakan kakas dasar penyelesaian masalah-masalah besar, seperti simulasi fenomena antrian di dunia nyata, serta fenomena antrian di pengolahan data. Beberapa fenomena dunia nyata berupa antrian diantaranya : antrian pembelian tiket di depan loket untuk bis, kereta api, bioskop; antrian mobil di depan gerbang jalan tol; antrian kendaraan di jalanan umum dll.

**1.2. Rumusan Masalah**

Dari uraian latar belakang di atas dapat ditarik rumusah masalah yaitu :

1. Apakah Queue itu ?

2. Bagaimana implementasi Queue dengan Linear Array ?

3. Bagaimana implementasi Queue dengan Circullar Array ?

**1.3. Tujuan Penulisan**

Adapun tujuan penulisan makalah ini adalah:

1. Agar pembaca lebih mengenal sapa itu Queue
2. Agar pembaca dapat mengenal queue dan

**BAB II**

**PEMBAHASAN**

**2.1. Queue (Antrian)**

Secara harfiah queue dapat diartikan sebagai antrian. Queue merupakan kumpulan data dengan penambahan data hanya melalui satu sisi, yaitu belakang tail dan penghapusan data hanya melalui sisi depan head. Berbeda dengan stack yang bersifat LIFO maka queue bersifat FIFO First In First Out, yaitu data yang pertama masuk akan keluar terlebih dahulu dan data yang terakhir masuk akan keluar terakhir. Berikut ini adalah gambaran struktur data queue. Elemen yang pertama kali masuk ke dalam queue disebut elemen depan fronthead of queue, sedangkan elemen yang terakhir kali masuk ke queue disebut elemen belakang reartail of queue. Perbedaan antara stack dan queue terdapat pada aturan penambahan dan penghapusan elemen. Pada stack, operasi penambahan dan penghapusan elemen dilakukan di satu ujung. Elemen yang terakhir kali dimasukkan akan berada paling dekat dengan ujung atau dianggap paling atas sehingga pada operasi penghapusan, elemen teratas tersebut akan dihapus paling awal, sifat demikian dikenal dengan LIFO.

Sifat yang demikian dikenal dengan FIFO. Operasi operasi standar pada queue adalah: ‐

1. Membuat queue atau inisialisasi.

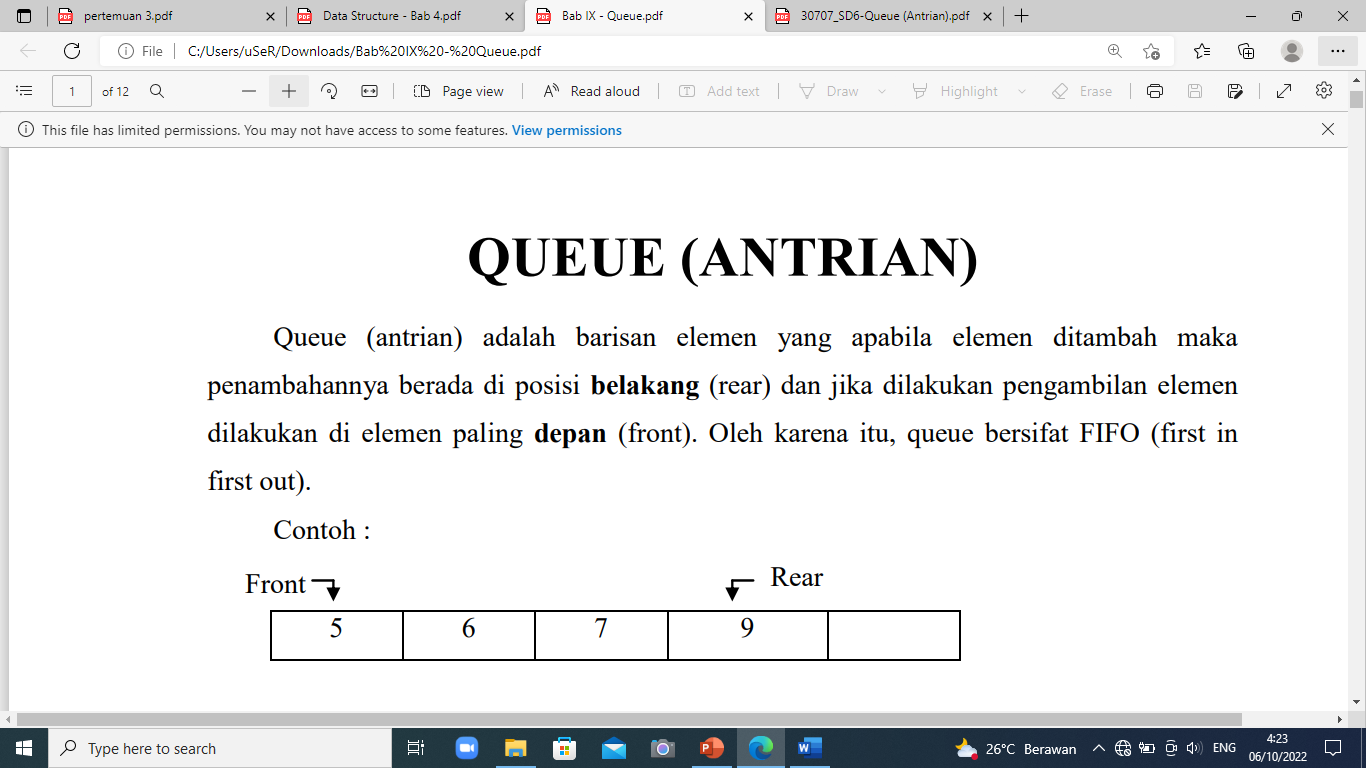
2. Mengecek apakah queue penuh.

3. Mengecek apakah queue kosong.

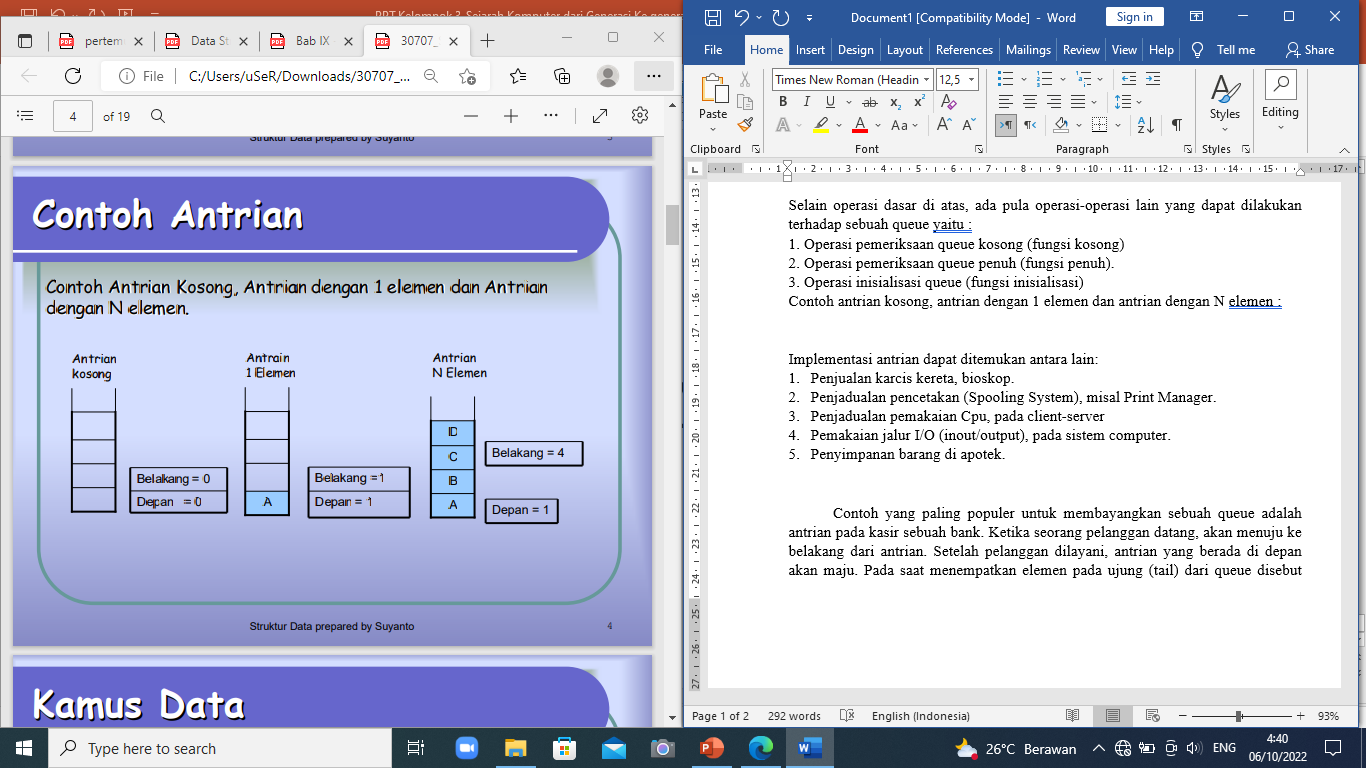
4. Memasukkan elemen ke dalam queue atau InQueue Insert Queue.

5. Menghapus elemen queue atau DeQueue Delete Queue.

Contoh :



Contoh antrian kosong, antrian dengan 1 elemen dan antrian dengan N elemen :



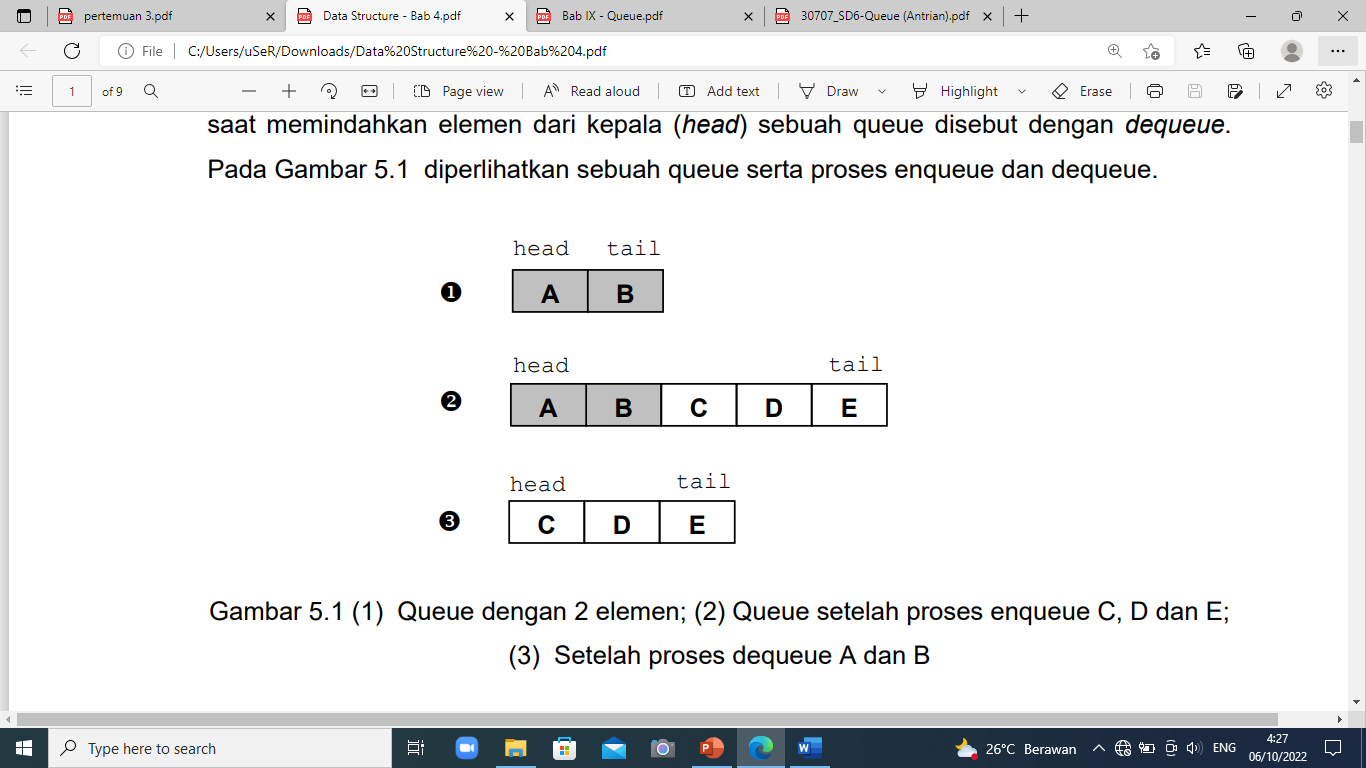
Implementasi antrian dapat ditemukan antara lain:

1. Penjualan karcis kereta, bioskop.
2. Penjadualan pencetakan (Spooling System), misal Print Manager.
3. Penjadualan pemakaian Cpu, pada client-server
4. Pemakaian jalur I/O (inout/output), pada sistem computer.

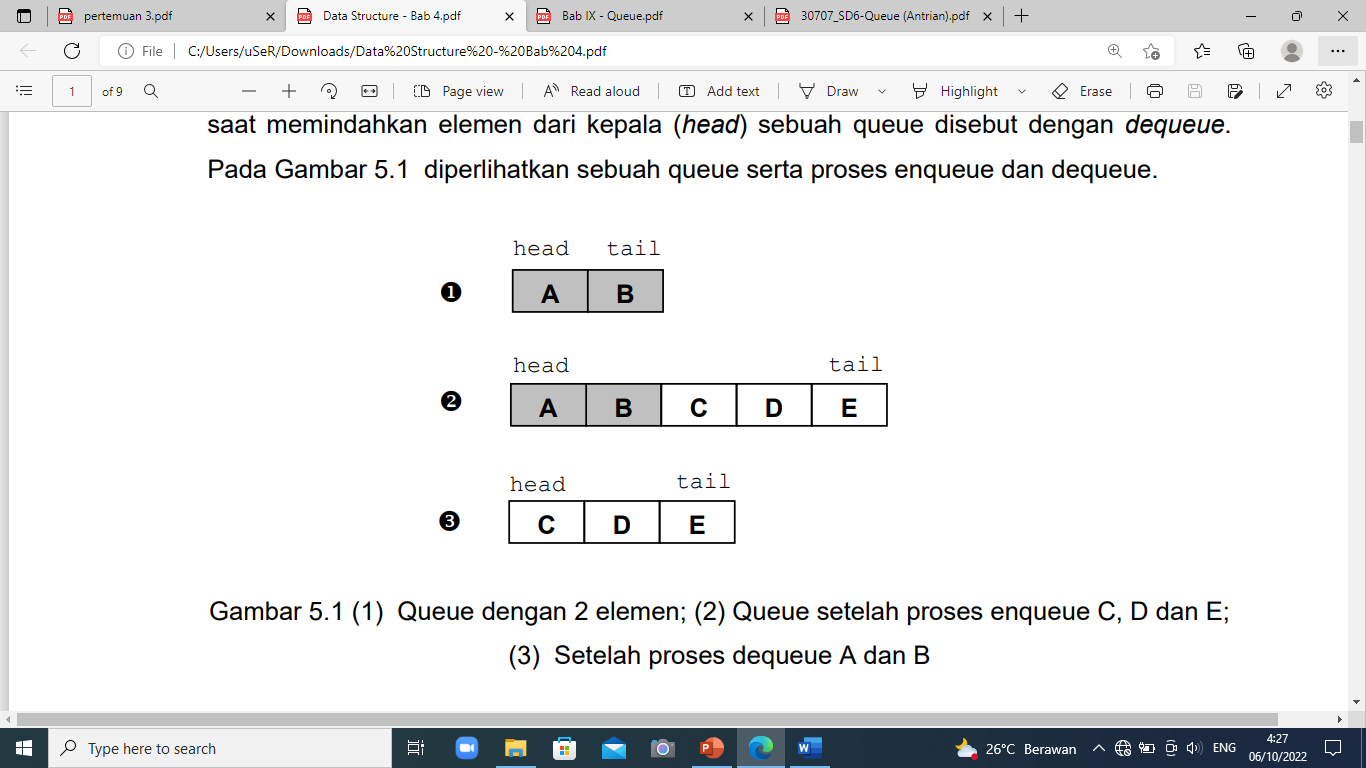
5. Penyimpanan barang di apotek.

Contoh yang paling populer untuk membayangkan sebuah queue adalah antrian pada kasir sebuah bank. Ketika seorang pelanggan datang, akan menuju ke belakang dari antrian. Setelah pelanggan dilayani, antrian yang berada di depan akan maju. Pada saat menempatkan elemen pada ujung (tail) dari queue disebut dengan enqueue, pada saat memindahkan elemen dari kepala (head) sebuah queue disebut dengan dequeue.

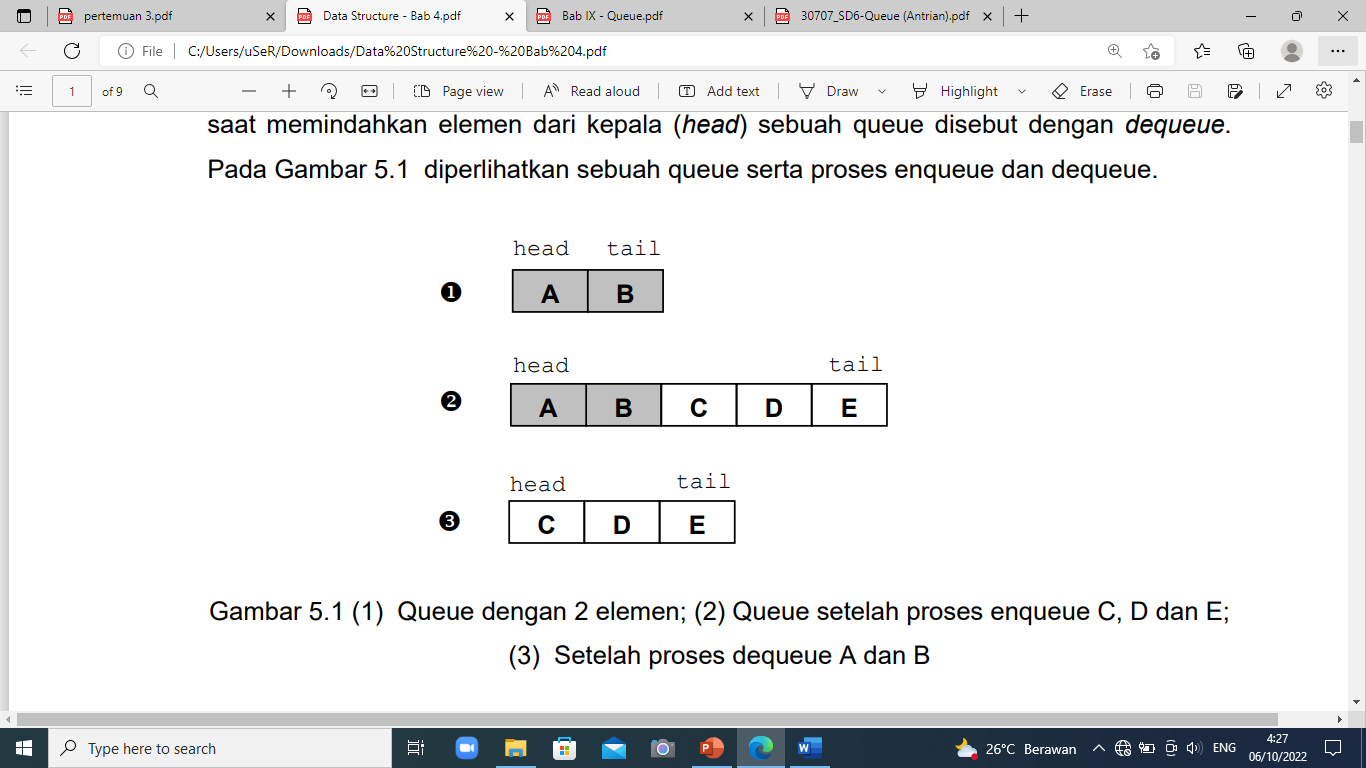
* Queue dengan 2 elemen



* Queue setelah proses enqueue C,D,E.



* Setelah proses dequeue A dan B.



**2.1.1. Operasi-operasi dasar dari sebuah queue**

1. Enqueue : proses penambahan elemen di posisi belakang

2. Dequeue : proses pengambilan elemen di posisi depan.

3. IsEmpty: Memeriksa apakah antrian kosong

4. IsFull: Memeriksa apakah antrian sudah penuh

5. Peek: Mendapatkan nilai bagian depan antrian tanpa menghapusnya

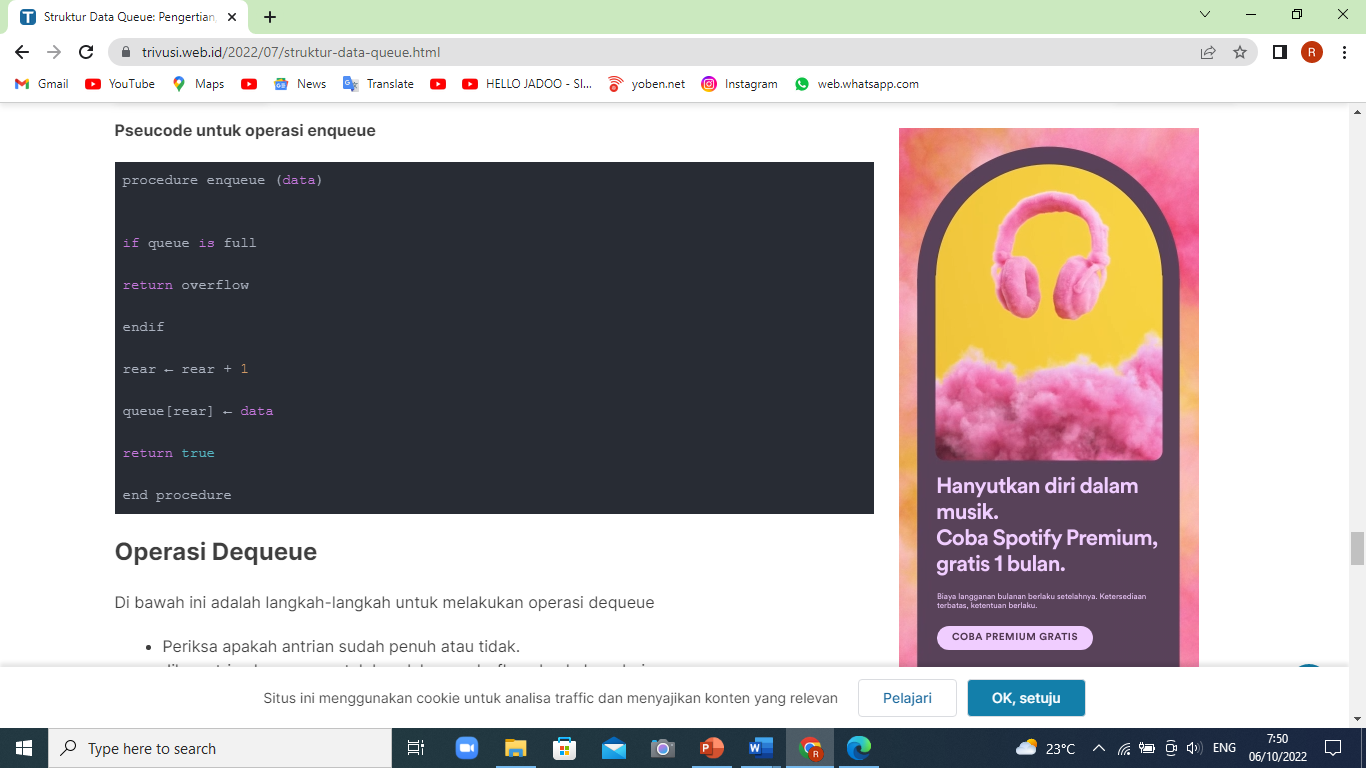
6. Initialize: Membuat antrian baru tanpa elemen data (kosong)

Namun, secara umum antrian memiliki 2 operasi utama, yaitu enqueue dan dequeue.

**2.1.2. Operasi Enqueue**

Di bawah ini adalah langkah-langkah untuk enqueue (memasukkan) data ke dalam antrian

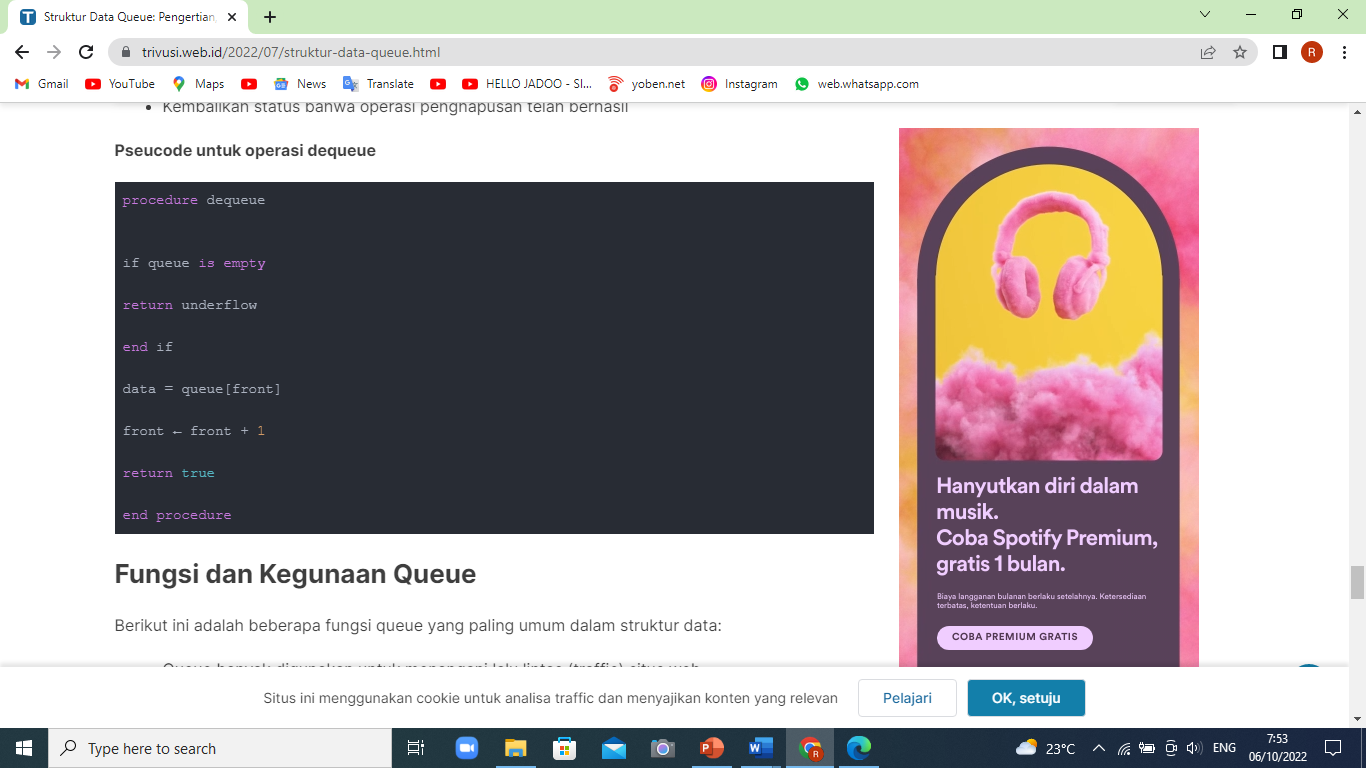
* + - Periksa apakah antrian sudah penuh atau tidak.
    - Jika antrian penuh – cetak kesalahan overflow dan keluar dari program.
    - Jika antrian tidak penuh – naikkan pointer belakang untuk menunjuk ke ruang kosong berikutnya.
    - Tambahkan elemen pada posisi yang ditunjuk oleh pointer belakang.
    - Kembalikan status bahwa penambahan telah berhasil



**2.1.3. Operasi Dequeue**

Di bawah ini adalah langkah-langkah untuk melakukan operasi dequeue

* + - Periksa apakah antrian sudah penuh atau tidak.
    - Jika antrian kosong – cetak kesalahan underflow dan keluar dari program.
    - Jika antrian tidak kosong – akses elemen data yang ditunjuk oleh pointer depan.
    - Geser pointer depan untuk menunjuk ke elemen data berikutnya yang tersedia.
    - Kembalikan status bahwa operasi penghapusan telah berhasil



**2.1.4. Karakteristik antrian**

Adapun karakteristik antrian, yakni sebagai berikut:

1. Elemen antrian yaitu item-item data yang terdapat di elemen antrian

2. Front (elemen terdepan dari antrian)

3. Tail (elemen terakhir dari antrian)

4. Count (jumlah elemen pada antrian)

5. Status antrian apakah penuh atau kosong.

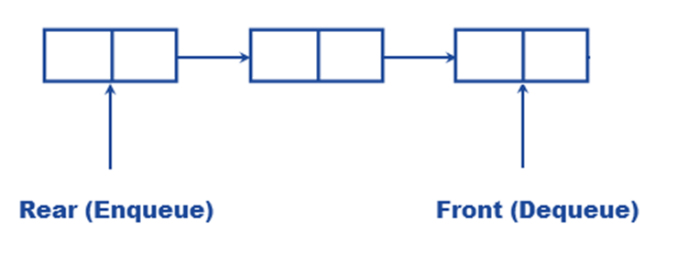
- Penuh, jika elemen pada antrian mencapai kapasitas maximum antrian. Pada kondisi ini, tidak mungkin dilakukan panambahan ke antrian.

- Kosong, jika tidak ada elemen pada antrian. Pada kondisi ini, tidak mungkin dilakukan pengambilan elemen dari antrian,

**2.1.5. Jenis-Jenis Queue**

1. Simpel Queue

Simple queue adalah struktur data queue paling dasar di mana penyisipan item dilakukan di simpul belakang (rear atau tail) dan penghapusan terjadi di simpul depan (front atau head).



2. Circular Queue

Pada circular queue, simpul terakhir terhubung ke simpul pertama. Queue jenis ini juga dikenal sebagai Ring Buffer karena semua ujungnya terhubung ke ujung yang lain. Penyisipan terjadi di akhir antrian dan penghapusan di depan antrian.

|  |
| --- |
| [Circular Queue](https://www.naukri.com/learning/articles/wp-content/uploads/sites/11/2022/06/image-86-768x707.png) |
|  |

**2.1.6. Priority Queue**

Priority Queue adalah strruktur data queue dimana simpul akan memiliki beberapa prioritas yang telah ditentukan. Simpul dengan prioritas terbesar akan menjadi yang pertama dihapus dari antrian. Sedangkan penyisipan item terjadi sesuai urutan kedatangannya.

|  |
| --- |
| [Priority Queue](https://www.naukri.com/learning/articles/wp-content/uploads/sites/11/2022/06/image-84-768x682.png) |
|  |

Aplikasi priority queue antara lain algoritma jalur terpendek Dijkstra, algoritma prim, dan teknik kompresi data seperti kode Huffman.

**2.1.7. Double-Ended Queue (Dequeue)**

Dalam double-ended queue (dequeue), operasi penyisipan dan penghapusan dapat terjadi di ujung depan dan belakang dari queue.

|  |
| --- |
| [Double-Ended Queue (Dequeue)](https://www.naukri.com/learning/articles/wp-content/uploads/sites/11/2022/06/image-85-1024x164.png) |
|  |

**2.1.8. Representasi Antrian**

Representasi antrian secara sekuen relatif lebih sulit dibanding stack. Seperti dijelaskan di atas bahwa antrian juga merupakan satu kumpulan data. Dengan demikian tipe data yang sesuai untuk menyajikan antrian adalah menggunakan array atau linked list.

**2.2. Implementasi Queue dengan Linear Array**

Disebut juga queue dengan model fisik, yaitu bagian depan queue selalu menempati posisi pertama array. Queue dengan linear array secara umum dapat dideklarasikan sebagai berikut:

Const

MaxQueue = {jumlah};

Type

TypeQueue = byte;

Var

Queue : Array[1..MaxQueue] of TypeQueuel

Head, Tail : Byte;

Operasi operasi queue dengan linear array: ‐

1. Create

Procedure create berguna untuk menciptakan queue yang baru dan kosong yaitu dengan cara memberikan nilai awal (head) dan nilai akhir (tail) dengan nol (0). Nol menunjukan bahwa queue masih kosong.

Procedure Create;

Begin

Head := 0; Tail := 0;

End;

2. Empty

Function empty berguna untuk mengecek apakah queue masih kosong atau sudah berisi data. Hal ini dilakukan dengan mengecek apakah tail bernilai nol atau tidak, jika ya maka kosong.

Function Empty : Boolean;

Begin

If Tail = 0 then

Empty := true

Else

Empty := False;

End;

3. Full

Function Full : Boolean;

Begin

If Tail = MaxQueue then

Full := true

Else

Full := False

End;4. EnQueue

Procedure EnQueue berguna untuk memasukkan 1 elemen ke dalam queue. Procedure Enqueue(Elemen : Byte);

Begin

If Empty then

Begin

Head := 1;

Tail := 1;

Queue[head] := Elemen;

End

Else

If Not Full then

Begin

Inc(Tail);

Queue[tail] := Elemen;

End;

End;

5. DeQueue

Procedure DeQueue berguna untuk mengambil 1 elemen dari Queue, operasi ini sering disebut juga SERVE. Hal ini dilakukan dengan cara memindahkan semua elemen satu langkah ke posisi di depannya, sehingga otomatis elemen yang paling depan akan tertimpa dengan elemen yang terletak dibelakangnya.

Procedure DeQueue;

Var I : Byte;

Begin

If Not Empty then

Begin

For I := Head to Tail 1 do ‐

Queue[I] := Queue[I+1];

Dec(Tail);

End;

End;

6. Clear

Procedure clear berguna untuk menghapus semua elemen dalam queue dengan jalan mengeluarkan semua elemen tersebut satu per satu sampai kosong dengan memanfaatkan procedure DeQueue.

Procedure Clear;

Begin

While Not Empty then

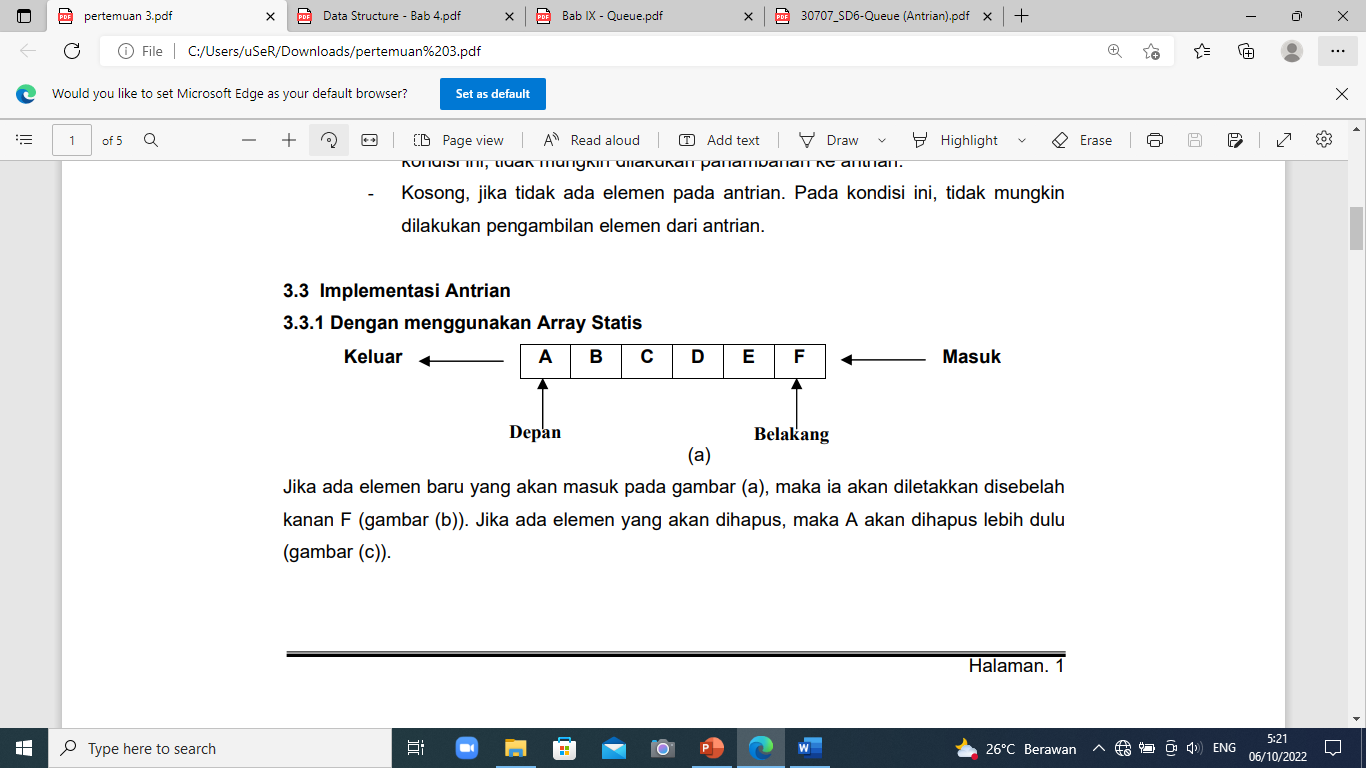
DeQueue;

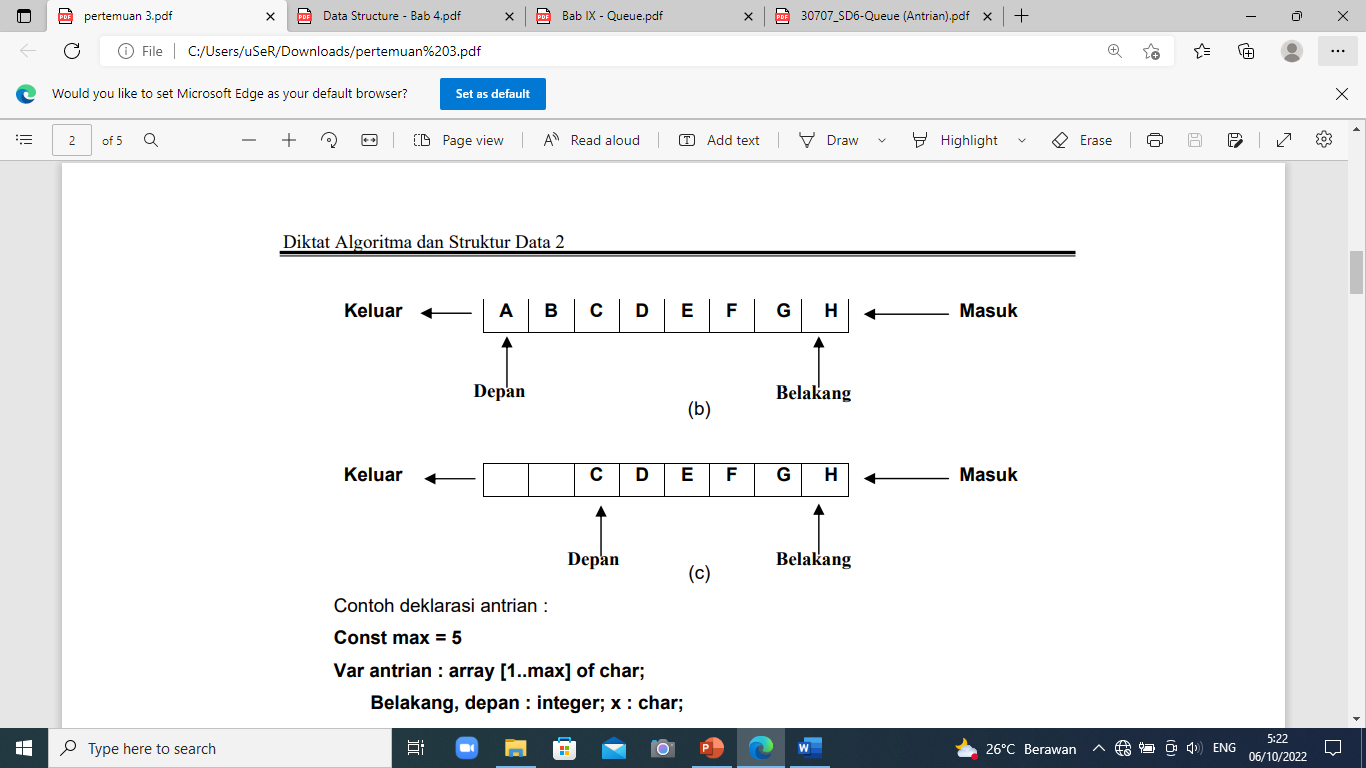
End;

* Array dibagi menjadi 2, yakni :

A. Implementasi dengan menggunakan Array Statis

Jika ada elemen baru yang akan masuk pada gambar (a), maka ia akan diletakkan disebelah kanan F (gambar (b)). Jika ada elemen yang akan dihapus, maka A akan dihapus lebih dulu (gambar (c)).

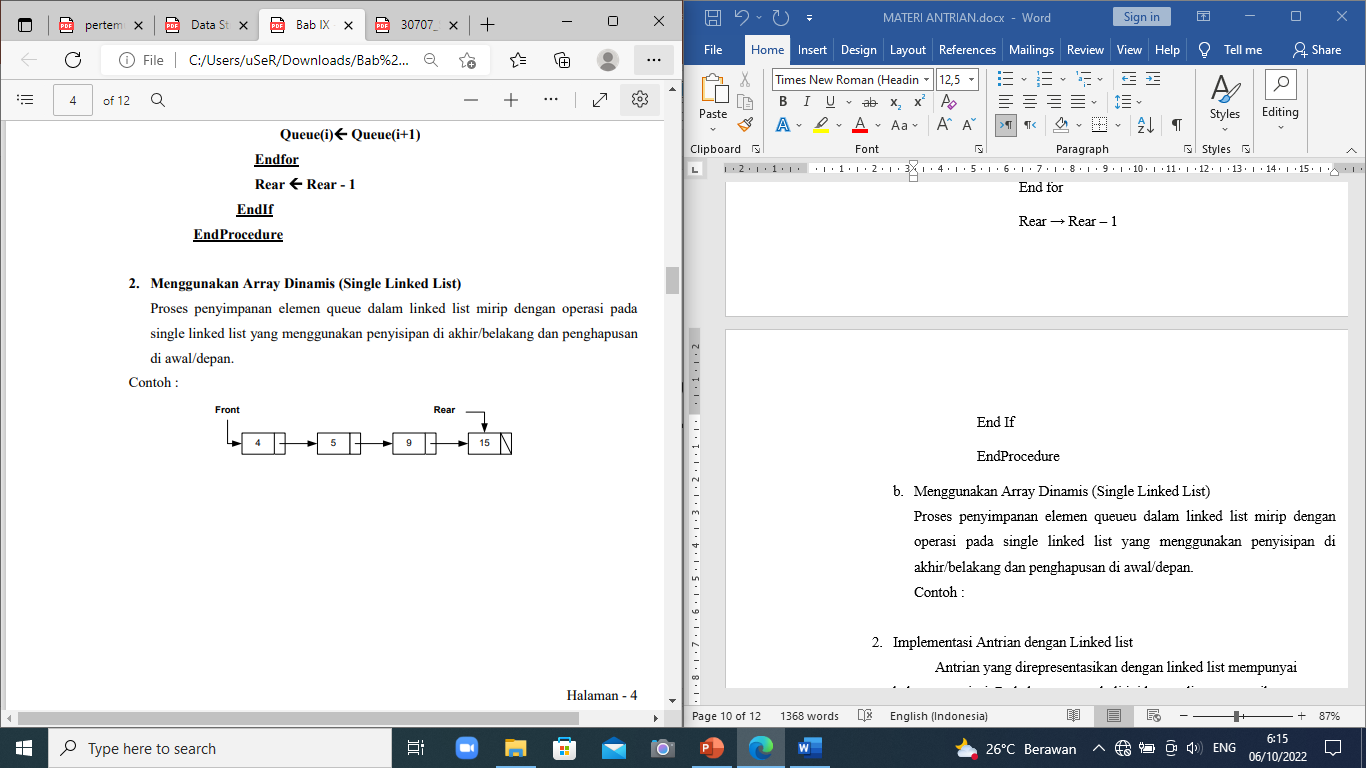




B. Menggunakan Array Dinamis (Single Linked List)

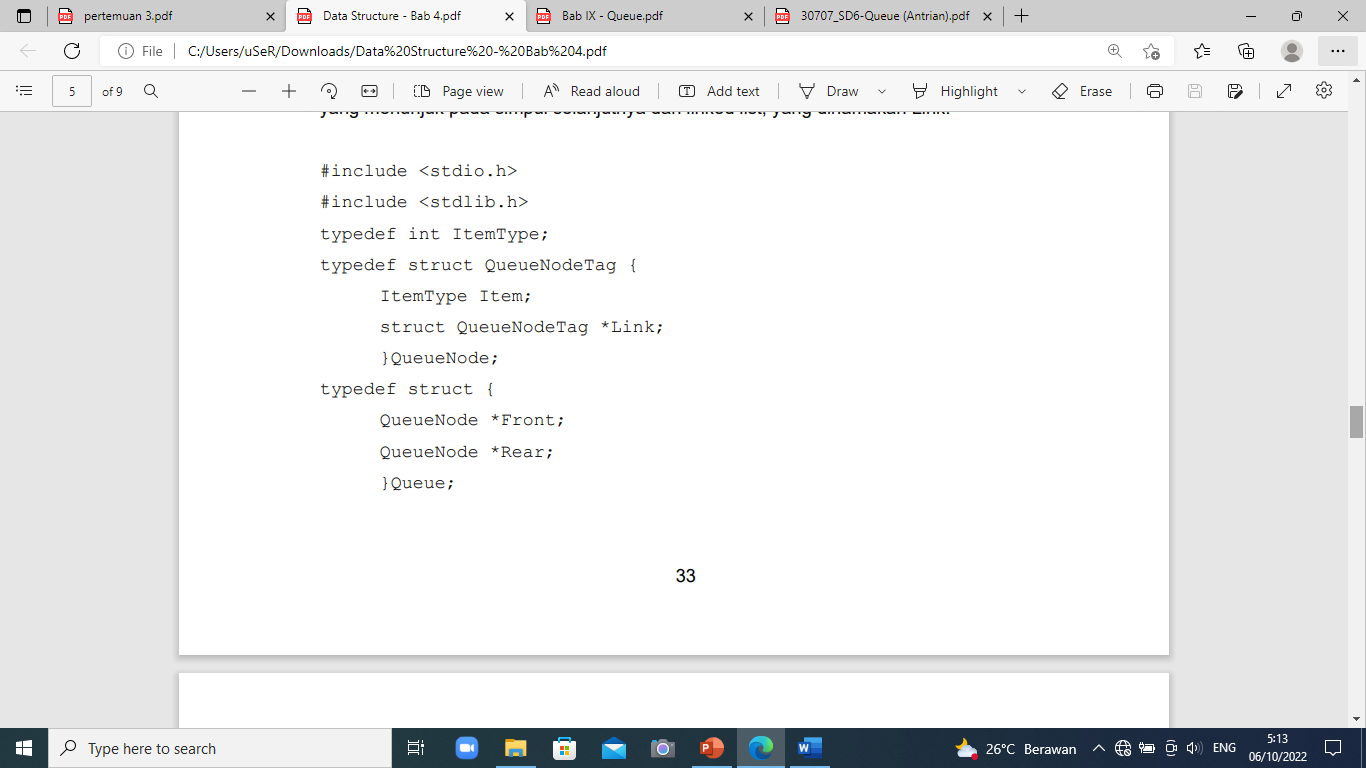
Proses penyimpanan elemen queueu dalam linked list mirip dengan operasi pada single linked list yang menggunakan penyisipan di akhir/belakang dan penghapusan di awal/depan.

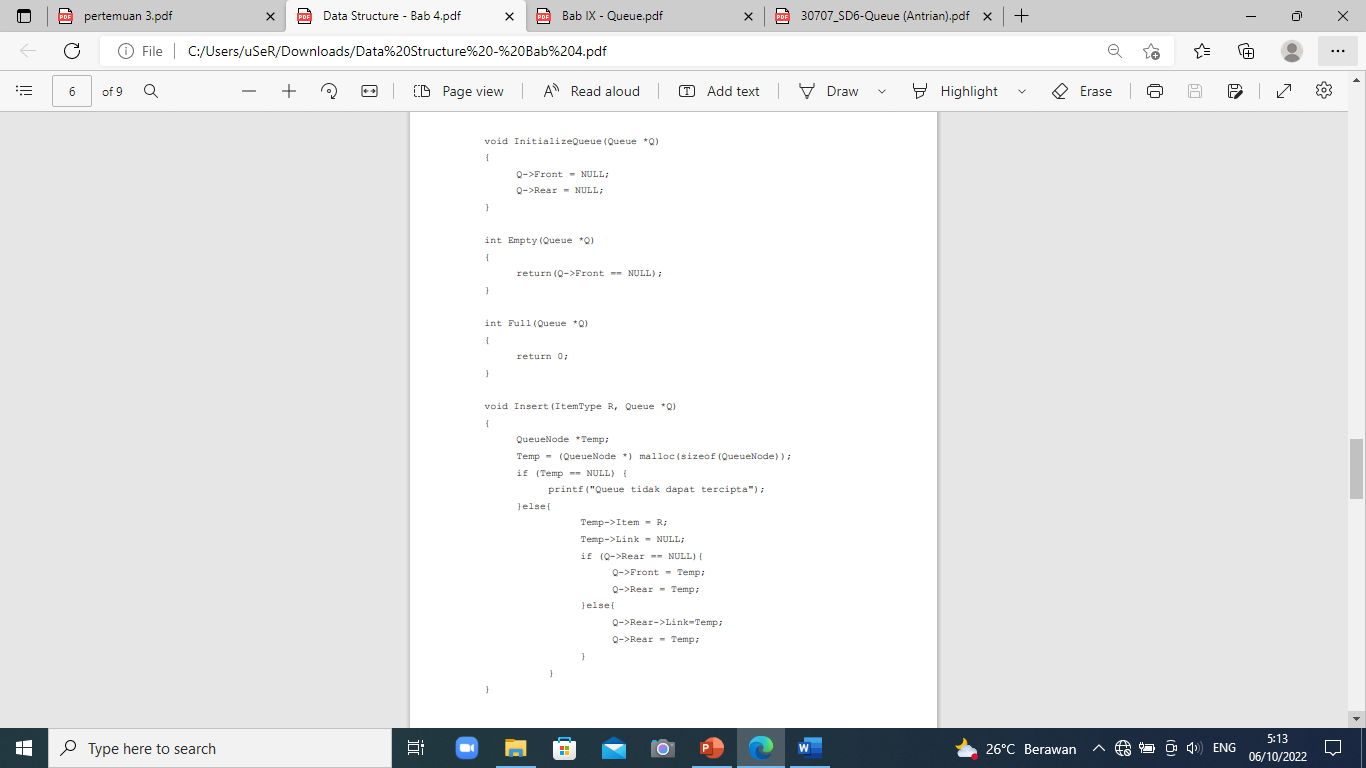
Contoh :

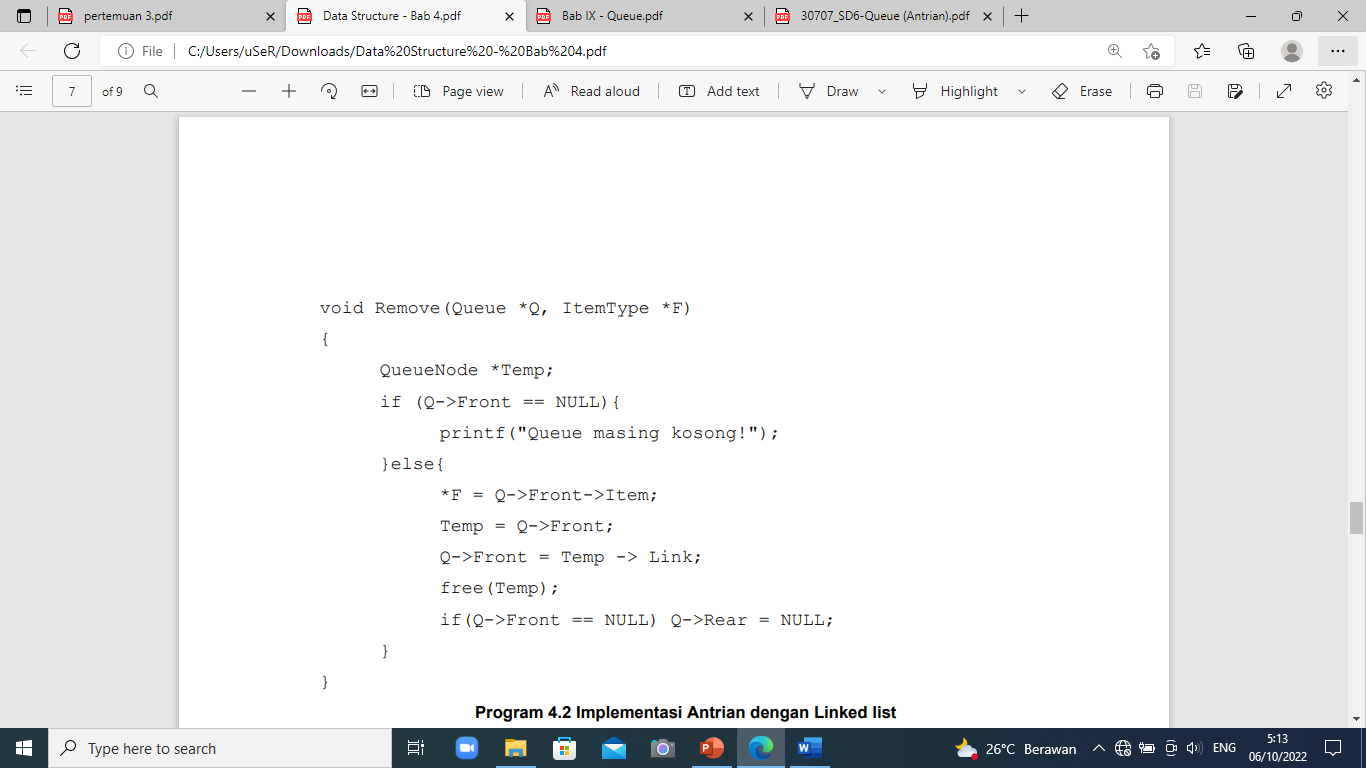


**2.3. Implementasi Antrian dengan Linked list**

Antrian yang direpresentasikan dengan linked list mempunyai beberapa variasi. Pada kesempatan kali ini hanya direpresentasikan satu macam saja. Linked list yang digunakan di sini menggunakan struktur yang berisi pointer yang menunjuk pada simpul Front dan Rear dari linked list. Masing-masing simpul berisi data dari antrian dan juga link yang menunjuk pada simpul selanjutnya dari linked list, yang dinamakan Link.







**2.4 Queue Dengan Circular Array**

Jika kita menggunakan array untuk queue seperti di atas, maka ketika ada proses pengambilan (dequeue) ada proses pergeseran data. Proses pergeseran data ini pasti memerlukan waktu apalagi jika elemen queue-nya banyak. Oleh karena itu solusi agar proses pergeseran dihilangkan adalah dengan metode circular array.

Queue dengan circular array dapat dibayangkan sebagai berikut :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Front = 1 |  |  | Rear = 4 |  |
| 5 | 6 | 7 | 9 |  |

Atau agar lebih jelas, array di atas dapat dibayangkan ke dalam bentuk seperti lingkaran di bawah ini.

**4 2**

**5**

**1**

**5**

**9**

**6**

**7**

**3**

**Front = 1**

**Rear = 4**

Aturan-aturan dalam queue yang menggunakan circular array adalah :

1. Proses penghapusan dilakukan dengan cara nilai depan (Front) ditambah 1 : Front = Front + 1.
2. Proses penambahan elemen sama dengan linear array yaitu nilai belakang(Rear) ditambah 1 : Rear = Rear + 1.
3. Jika Front = MaksQueue dan ada elemen yang akan dihapus, maka nilai Front = 1.
4. Jika Rear = MaksQueue dan Front tidak 1 maka jika ada elemen yang akan ditambahkan, nilai belakang=1
5. Jika hanya tinggal 1 elemen di queue (Front = Rear), dan akan dihapus, maka Front diisi 0 dan Rear diisi dengan 0 (queue kosong).

**Contoh :**

**A**

**B**

**C**

**B**

**E**

**C**

**D**

**B**

**E**

**C**

**D**

**E**

**F**

**E**

**F**

**G**

**D**

**D**

**E**

**H**

**F**

**G**

**H**

**F**

**G**

**H**

**H**

Untuk mempermudah, elemen dari queue bertipe karakter.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| No | Operasi | Status | Queue | | | | |
| 1 | 2 | 3 | 4 | 5 |
| 1. | Inisialisasi | Front = 0  Rear = 0 |  |  |  |  |  |
| 2. | Enqueue A, B, C | Front = 1  Rear = 3 | **A** | **B** | **C** |  |  |
| 3. | Dequeue | Front = 2  Rear = 3 |  | **B** | **C** |  |  |
| 4. | Enqueue D, E | Front = 2  Rear = 5 |  | **B** | **C** | **D** | **E** |
| 5. | Dequeue 2 kali | Front = 4  Rear = 5 |  |  |  | **D** | **E** |
| 6. | Enqueue F | Front = 4  Rear = 1 | **F** |  |  | **D** | **E** |
| 7. | Enqueue G, H | Front = 4  Rear = 3 | **F** | **G** | **H** | **D** | **E** |
| 8. | Dequeue | Front = 5  Rear = 4 | **F** | **G** | **H** |  | **E** |
| 9. | Dequeue | Front = 1  Rear = 3 | **F** | **G** | **H** |  |  |
| 10. | Dequeue 2 kali | Front = 3  Rear = 3 |  |  | **H** |  |  |
| 11. | Dequeue | Front = 0  Rear = 0 |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **4** | **5**  **3**  Operasi | **1**  1 | **2** | **4** | **5 1**  **3**  Operasi 2 | | **2** | **4** | **5 1**  **3**  Operasi 3 | | **2** | **4** | **5 1**  **3**  Operasi 4 | | **2** | **4** | **5 1**  **2**  **3**  Operasi 5 | | | **5 1**  **4 2**  **3**  Operasi 6 |
|  | **5** | **1** |  |  | **5** | **1** |  |  | **5** | **1** |  |  | **5** | **1** |  |  | **5** | **1** |  |  |
| **4** |  |  | **2** | **4** |  |  | **2** | **4** |  |  | **2** | **4** |  |  | **2** | **4** |  |  | **2** |
| **3** | | | | **3** | | | | **3** | | | | **3** | | | | **3** | | | |
| Operasi 7 | | | | Operasi 8 | | | | Operasi 9 | | | | Operasi 10 | | | | Operasi 11 | | | |

Operasi-operasi yang dapat terjadi dalam queue yang menggunakan circular array adalah :

1. **Penciptaan queue**

Penciptaan queue adalah proses pemberian nilai 0 untuk penunjuk Front dan penunjuk Rear dari queue

Front → 0

Rear → 0

1. **Fungsi Kosong**

Suatu queue yang menggunakan circular array dapat dikatakan kosong jika nilai dari posisi Front atau Rear mempunyai nilai 0.

Function Kosong(Input Front : Integer) → Boolean

{I.S. : Penunjuk Front sudah terdefinisi}

{F.S. : Menghasilkan fungsi kosong}

Kamus:

Algoritma :

Kosong → false

If (Front = 0) Then Kosong → true

EndIf EndFunction

1. **Fungsi Penuh**

Suatu queue akan disebut penuh jika terjadi 2 hal yaitu

* + Jika Front ada diposisi 1 dan Rear ada diposisi MaksQueue
  + Atau jika Front bernilai sama dengan posisi Rear +1.

Function Penuh (Input Front, Rear : Integer) → Boolean

{I.S. : Penunjuk Front dan Rear sudah terdefinisi}

{F.S. : Menghasilkan fungsi penuh} Kamus:

Algoritma :

PenuhßFalse

If ((Front = 1) and (Rear = maks)) or (Front = Rear+1) Then PenuhßTrue

EndIf EndFunction

1. **Operasi Enqueue**

Proses enqueue hanya bisa dilakukan jika queue tidak kosong. Ada beberapa hal yang harus diperhatikan ketika akan melakukan enqueue pada circular array, diantaranya adalah:

* + Kondisi ketika queue masih kosong. Jika ini terjadi, maka posisi Front dan Rear bernilai 0.
  + Ketika nilai Rear sama dengan MaksQueue, maka posisi Rear bernilai 1
  + Ketika nilai Rear masih lebih kecil dari MaksQueue, maka posisi Rear ditambah 1 : Rear = Rear + 1

Procedure Enqueue (Input elemen : tipedata, I/O Front,Rear : Integer) {I.S. : Data yang akan dimasukkan (elemen), penunjuk Front dan penunjuk Rear sudah terdefinisi}

{F.S. : Menghasilkan Queue yang sudah bertambah satu data} Kamus:

Algoritma :

If (Kosong(Front)) Then Front  1

Rear  1 Else

If (Not Penuh(Front,Rear)) Then If (Rear = MaksQueue) Then

Rear  1 Else

Rear  Rear + 1 EndIf

Queue(Rear)  elemen

EndIf

EndProcedure

1. **Antrian Berprioritas**

Dalam antrian yang telah dibahas di atas, semua elemen yang masuk dalam antrian dianggap mempunyai prioritas yang sama, sehingga elemen yang masuk lebih dahulu akan diproses lebih dahulu. Dalam praktek, elemen-elemen yang akan masuk dalam suatu antrian ada yang dikatakan mempunyai prioritas yang lebih tinggi dibanding yang lain. Antrian yang demikian ini disebut dengan antrian berprioritas (priority queue).

Dalam antrian berprioritas, setiap elemenn yang akan msuk dalam antrian sudah

ditentukan lebih dahulu prioritasnya. Dalam hal ini berlaku dua ketentuan, yaitu:

1. Elemen-elemen yang mempunyai prioritas lebih tinggi akan diproses lebih dahulu.

2. Dua elemen yang mempunyai prioritas sama akan dikerjakan sesuai dengan urutan

pada saat kedua elemen ini masuk dalam antrian.

Salah satu contoh antrian berprioritas ini adalah pada sistem berbagi waktu (time-sharing system) dimana program yang mempunyai prioritas tinggi akan dikerjakan lebih dahulu dan program-program yang berprioritas sama akan membentuk antrian biasa. Ada beberapa cara untuk mengimplementasikan antrian berprioritas. Salah satu caranya adalah dengan menggunakan linked list. Jika kita menggunakan linked list, khususnya single linked list atau double linked list, maka ada ketentuan lain yang perlu diperhatikan, yaitu:

1. Setiap node dari linked list terdiri tiga bagian, yaitu bagian informasi, angka prioritas dan bagian-bagian penyambung ke simpul lain.
2. Simpul X mendahului (terletak di sebelah kiri) simpul Y, jika prioritas X lebih tinggi dibanding prioritas Y atau jika prioritas X dan Y sama, maka simpul X datang lebih dahulu dibanding dengan Y.

Biasanya dibuat suatu perjanjian bahwa angka prioritas yang lebih kecil menunjukkan derajad prioritas yang lebih tinggi. Sebagai contoh, jika angka prioritaS pada simpul X adalah 1 dan pada simpul Y adalah 2, maka dikatakan bahwa simpul X berprioritas lebih tinggi dibanding dengan simpul Y.

**2.5. Fungsi dan Kegunaan Queue**

Berikut ini adalah beberapa fungsi queue yang paling umum dalam struktur data:

1. Queue banyak digunakan untuk menangani lalu lintas (traffic) situs web.
2. Membantu untuk mempertahankan playlist yang ada pada aplikasi media player
3. Queue digunakan dalam sistem operasi untuk menangani interupsi.
4. Membantu dalam melayani permintaan pada satu sumber daya bersama, seperti printer, penjadwalan tugas CPU, dll.
5. Digunakan dalam transfer data asinkronus misal pipeline, IO file, dan socket.

**2.6. Kelebihan Queue**

Kelebihan queue di antarnya:

1. Data dalam jumlah besar dapat dikelola secara efisien.
2. Operasi seperti penyisipan dan penghapusan dapat dilakukan dengan mudah karena mengikuti aturan masuk pertama keluar pertama.
3. Queue berguna ketika layanan tertentu digunakan oleh banyak konsumen.
4. Queue cepat untuk komunikasi antar-proses data.
5. Queue dapat digunakan dalam implementasi struktur data lainnya.

**2.7. Kekurangan Queue**

Kelemahan struktur data queue adalah sebagai berikut:

1. Operasi seperti penyisipan dan penghapusan elemen dari tengah cenderung banyak memakan waktu.
2. Dalam queue konvensional, elemen baru hanya dapat dimasukkan ketika elemen yang ada dihapus dari antrian.
3. Mencari elemen data pada struktur queue membutuhkan time complexity O(N).
4. Ukuran maksimum antrian harus ditentukan sebelumnya.