01158 - ALGORITMA DAN STRUKTUR DATA 2

QUEUE

05

OLEH : ANDRI HERYANDI, M.T.

MATERI

01158 - ALGORITMA DAN STRUKTUR DATA 2

- Definisi Queue
- Operasi-Operasi Pada Queue
- Queue Dengan Python
- Membuat Class Queue
  - Queue dengan Array
  - Queue dengan Linked List
- Circular Queue

DEFINISI QUEUE

- Dalam Bahasa Indonesia disebut sebagai **Antrian**
- Queue adalah sebuah daftar elemen data yang menerapkan kaidah First-In First Out (FIFO).
- Dalam Queue, elemen akan ditambahkan di akhir (belakang/rear), dan pengambilan data hanya dilakukan di elemen awal (depan/front).
- Queue dapat diimplementasikan menggunakan array atau pun linked list.
- Apa pun implementasi yang dilakukan, 2 aturan berikut harus benar-benar dipatuhi.
  - **Enqueue** (Penambahan) : Penambahan data hanya akan selalu dilakukan pada posisi belakang.
  - **Dequeue** (Pengambilan) : Pengambilan data hanya akan dilakukan pada posisi depan.

CONTOH PENERAPAN QUEUE

- Contoh penerapan queue di dunia kehidupan sehari-hari
  - Antrian di bank
  - Antrian di restoran
- Contoh penerapan queue di dunia komputer
  - Antrian email di server email.
  - Antrian proses

OPERASI-OPERASI PADA QUEUE

- Operasi-operasi utama pada queue adalah :
  - **Enqueue** : Proses menambahkan elemen baru ke dalam queue di posisi belakang.
  - **Dequeue** : Proses mengambil elemen dari queue di posisi depan.
- Operasi-operasi tambahan pada queue adalah :
  - **Initialize** : Proses menginisialisasi queue sebelum digunakan.
  - **isEmpty** : Proses melihat status queue apakah kosong atau sudah mempunyai elemen
  - **isFull** : Proses melihat status queue apakah elemen queue sudah penuh (Index Belakang = Max). Ini hanya digunakan pada queue yang diimplementasikan dengan array statis.
  - **Count** : Proses menghitung berapa elemen yang ada dalam sebuah queue.
  - **Peek** : Proses melihat elemen data yang berada pada posisi depan tanpa melakukan dequeue.
  - **Display** : Proses menampilkan isi queue tanpa melakukan dequeue.
  - **Empty** : Proses mengosongkan queue.

01158 - ALGORITMA DAN STRUKTUR DATA 2

QUEUE DENGAN PYTHON

OLEH : ANDRI HERYANDI, M.T.

IMPLEMENTASI STACK DENGAN PYTHON

- Jika anda ingin mengimplementasikan queue dengan menggunakan python, maka ada 2 cara yang bisa anda lakukan, yaitu :
  - Queue menggunakan class/struktur data bawaan dari python
  - Queue menggunakan class/struktur data buatan sendiri (*User-Defined-Class / User-Defined-Data-Structure*).

01158 - ALGORITMA DAN STRUKTUR DATA 2

QUEUE DENGAN MENGGUNAKAN CLASS/STRUKTUR DATA BUILT-IN PYTHON

OLEH : ANDRI HERYANDI, M.T.

QUEUE MENGGUNAKAN CLASS/STRUKTUR DATA BUILT-IN PYTHON

- Setidaknya ada 3 cara yang bisa anda lakukan jika ingin mengimplementasikan queue dengan menggunakan bahasa Python yaitu :
  1. Menggunakan struktur data **`list`**
  2. Menggunakan class **`deque`** dari module **`collections`**
  3. Menggunakan class **`Queue`** dari modul **`queue`**

QUEUE DENGAN MENGGUNAKAN LIST

- Jika anda ingin mengimplementasikan queue menggunakan struktur data list dalam python, maka :
  - Operasi **enqueue** bisa dilakukan dengan memanggil method **`append()`**
  - Operasi **dequeue** bisa dilakukan dengan memanggil method **`pop()`** dengan parameter 0 yang menunjukan penghapusan di data pertama.

QUEUE DENGAN MENGGUNAKAN LIST

- Contoh :

```
queue = [] # list kosong / queue kosong
print("#1 Isi Queue : ", queue)
queue.append(1)
print("#2 Isi Queue : ", queue)
queue.append(2)
print("#3 Isi Queue : ", queue)
queue.append(3)
print("#4 Isi Queue : ", queue)
data = queue.pop(0)
print("#5 Hasil Dequeue : ", data)
print("#6 Isi Queue : ", queue)
```

```
#1 Isi Queue :  []
#2 Isi Queue :  [1]
#3 Isi Queue :  [1, 2]
#4 Isi Queue :  [1, 2, 3]
#5 Hasil Dequeue :  1
#6 Isi Queue :  [2, 3]
```

QUEUE DENGAN MENGGUNAKAN COLLECTION.DEQUE

- Jika anda ingin mengimplementasikan queue menggunakan class collections.deque dalam python, maka :
  - Operasi **enqueue** bisa dilakukan dengan memanggil method **`append()`**
  - Operasi **dequeue** bisa dilakukan dengan memanggil method **`popleft()`**.

QUEUE DENGAN MENGGUNAKAN LIST

- Contoh :

```
from collections import deque

queue = deque()
print("#1 Isi Queue : ", queue)
queue.append(1) # enqueue
print("#2 Isi Queue : ", queue)
queue.append(2) # enqueue
print("#3 Isi Queue : ", queue)
queue.append(3) # enqueue
print("#4 Isi Queue : ", queue)
data = queue.popleft() # dequeue
print("#5 Hasil Dequeue : ", data)
print("#6 Isi Queue : ", queue)
```

```
#1 Isi Queue :  deque([])
#2 Isi Queue :  deque([1])
#3 Isi Queue :  deque([1, 2])
#4 Isi Queue :  deque([1, 2, 3])
#5 Hasil Dequeue :  1
#6 Isi Queue :  deque([2, 3])
```

QUEUE DENGAN MENGGUNAKAN QUEUE.QUEUE

- Jika anda ingin mengimplementasikan queue menggunakan class queue.Queue dalam python, maka :
  - Operasi **enqueue** bisa dilakukan dengan memanggil method `put()` .
  - Operasi **dequeue** bisa dilakukan dengan memanggil method `get()`.

QUEUE DENGAN MENGGUNAKAN LIST

- Contoh :

```
from queue import Queue

q1 = Queue(maxsize=3)
print("#1 : Banyak Data : ", q1.qsize())
print("#2 : Kosong ? ", q1.empty())
q1.put(11) # Enqueue
q1.put(22) # Enqueue
q1.put(33) # Enqueue
print("#3 : Penuh ? ",q1.full())
data = q1.get() # Dequeue
print("#4 : Hasil Dequeue : ", data)
print("#5 : Banyak Data   : ", q1.qsize())
```

```
#1 : Banyak Data :  0
#2 : Kosong ?  True
#3 : Penuh ?  True
#4 : Hasil Dequeue :  11
#5 : Banyak Data   :  2
```

QUEUE DENGAN MENGGUNAKAN CLASS BUATAN SENDIRI

OLEH : ANDRI HERYANDI, M.T.

01158 - ALGORITMA DAN STRUKTUR DATA 2

QUEUE MENGGUNAKAN ARRAY

OLEH : ANDRI HERYANDI, M.T.

STRUKTUR DATA

- Struktur Data Queue diimplementasikan menggunakan Array

Index Front = 0

Index Rear = 3

Index	0	1	2	3	...	Max-1
Isi/Content	Budi	Ade	Putri	Irma		

- Elemen-elemen dalam queue array :
  - Front : index yang menunjukan posisi awal dari queue
  - Rear : index yang menunjukan posisi akhir dari queue
  - Max : Maksimum elemen yang ditampung oleh queue
  - Content : array yang berisi elemen-elemen (data) dalam queue

STRUKTUR DATA (ARRAY)

- Pendeklarasian class queue menggunakan array (Bahasa Python).

```
class queueArray:
    def __init__(self, max, datatype='i'):
        self.__max = max  # batas maksimal data dalam queue
        self.__content = ar.array(datatype, [0] * max)  # inisialisasi isi queue
        self.__front = -1
        self.__rear = -1
```

- Pendeklarasian variable/objek untuk queue menggunakan array (Bahasa Python).

```
q = queueArray(5) # membuat queue array dengan maksimal 5 data tipe data integer
Q2 = queueArray(5,'d') # queue array maksimal 5 data tipe data double
```

INISIALISASI QUEUE (ARRAY)

- Inisialisasi queue adalah proses mempersiapkan sebuah queue untuk digunakan.
- Proses yang dilakukan adalah dengan men-set nilai -1 pada variable/field yang menunjuk kepada nilai **Front** dan **Rear**. Hal ini dilakukan agar penambahan data berikutnya dilakukan pada index array ke-0.
- Pada class yang sedang dibuat, proses ini ada dalam function `__init__`.

PEMERIKSAAN QUEUE KOSONG (ARRAY)

- Pemeriksaan ini digunakan untuk memeriksa apakah queue dalam keadaan kosong.
- Sebuah queue disebut kosong jika **Front** dari queue mempunyai nilai -1 (sesuai di proses inisialisasi queue).
- Pemeriksaan ini akan bisa digunakan dalam operasi-operasi lain dari queue (misalnya ketika enqueue atau pun dequeue).

PEMERIKSAAN QUEUE KOSONG (ARRAY)

- Function **isEmpty**

```
def isEmpty(self):
        return self.__front == -1
```

PEMERIKSAAN QUEUE PENUH (ARRAY)

- Pemeriksaan ini digunakan untuk memeriksa apakah queue dalam keadaan penuh.
- Sebuah queue disebut penuh jika Rear dari queue mempunyai nilai sama dengan maksimum elemen array dikurangi 1 (max – 1).
- Pemeriksaan ini akan bisa digunakan dalam proses enqueue, karena proses enqueue hanya bisa dilakukan jika queue tidak penuh.
- Pemeriksaan queue penuh hanya dilakukan jika queue diimplementasikan menggunakan array.

PEMERIKSAAN QUEUE PENUH (ARRAY)

- Function **isFull**

```
def isFull(self):
        return self.__rear == self.__max -1
```

BANYAK ELEMEN DALAM QUEUE (ARRAY)

- Proses ini digunakan untuk mengetahui berapa banyak elemen yang berada dalam queue.
- Pada queue yang diimplementasikan dengan array, banyaknya elemen queue dapat dilihat dari nilai **Rear** dari queue ditambah 1 (Rear + 1).

BANYAK ELEMEN DALAM QUEUE (ARRAY)

- Function **count**

```
def count(self):
        return self.__rear + 1
```

MENAMPILKAN ELEMEN QUEUE (ARRAY)

- Proses ini digunakan untuk menampilkan isi elemen queue
- Langkah-langkah untuk menampilkan elemen queue adalah :
  1. Lakukan pengulangan dari element pertama (**Front**) sampai element terakhir (**Rear**) untuk melakukan proses 2 misalnya dengan menggunakan variable counter i.
  2. Tampilkan data array pada posisi ke-i.
- Pada class yang sedang dibangun, proses display dilakukan dengan mengoverride function `__str__`.

MENAMPILKAN ELEMEN QUEUE (ARRAY)

- Function **display / __str__**

```
def __str__(self):
        out = "queueArray(["
        if self.isEmpty():
            out = out + " EmptyQueue "
        else:
            for i in range(self.__rear+1):
                out = out + str(self.__content[i]) + " "
        out = out + "]"
        out = out + ")"
        return out
```

MENAMBAH ELEMEN QUEUE (ENQUEUE) (ARRAY)

- Proses ini sering disebut **Enqueue**.
- Operasi **Enqueue** digunakan untuk menambahkan elemen baru di posisi Rear.
- Langkah-langkah untuk mengambah elemen queue adalah :
  1. Lakukan pemeriksaan apakah queue sudah penuh. Jika queue sudah penuh maka tampilkan pesan bahwa “Queue penuh”. Jika queue belum penuh maka lakukan Langkah 2 sampai Langkah 3.
  2. Periksa apakah queue kosong. Jika queue kosong maka seting nilai Front dan Rear dari queue menjadi 0, tetapi jika queue tidak kosong maka tambahkan nilai 1 ke Rear dari queue
  3. Simpan element baru di posisi Rear

MENAMBAH ELEMEN QUEUE (ENQUEUE) (ARRAY)

- Ilustrasi Enqueue ketika queue masih kosong

Index Front = -1 Index Rear = -1

queue

Index	0	1	2	3	...	Max - 1
Isi						

Sebelum enqueue

Enqueue(7)

Index Front = 0 Index Rear = 0

queue

Index	0	1	2	3	...	Max - 1
Isi	7					

Setelah enqueue

Penunjuk Front dan Rear berubah menjadi 0

MENAMBAH ELEMEN QUEUE (ENQUEUE) (ARRAY)

- Ilustrasi Enqueue ketika queue sudah memiliki data

Index Front = 0 Index Rear = 2

queue1

Index	0	1	2	3	...	Max - 1
Isi	7	5	8			

Sebelum enqueue

Enqueue(10)

Index Front = 0 Index Rear = 3

queue1

Index	0	1	2	3	...	Max - 1
Isi	7	5	8	10		

Setelah enqueue

Hanya penunjuk Rear yang berubah (ditambah 1)

MENAMBAH ELEMEN QUEUE (ENQUEUE) (ARRAY)

- Function **enqueue**

```
def enqueue(self, data):
        if self.isFull():
            print("Enqueue Gagal. Queue Penuh.")
        else:
            if self.isEmpty():
                self.__front = 0
                self.__rear = 0
            else:
                self.__rear = self.__rear + 1
            self.__content[self.__rear] = data
```

MENGAMBIL/MENGHAPUS ELEMEN QUEUE (DEQUEUE) (ARRAY)

- Proses ini sering disebut **Dequeue**.
- Operasi dequeue digunakan untuk mengambil elemen yang berada di posisi Front.
- Langkah-langkah untuk mengambil elemen queue adalah :
  1. Lakukan pemeriksaan apakah queue kosong. Jika queue kosong maka tampilkan pesan bahwa “Queue kosong”. Jika queue tidak kosong maka lakukan Langkah 2 dan Langkah 4.
  2. Ambil nilai yang ditunjuk oleh **Front**.
  3. Geserkan data mulai posisi setelah posisi Front ke posisi sebelumnya (data di elemen 2 diisikan ke elemen 1, data di elemen 3 diisikan ke elemen 2 dan seterusnya sampai data di element Rear).
  4. Nilai Rear dari queue dikurangi 1. Jika Rear menjadi -1 (queue jadi kosong), maka Front-pun seting dengan nilai -1 (untuk menyatakan bahwa queue kosong)

MENGAMBIL/MENGHAPUS ELEMEN QUEUE (DEQUEUE) (ARRAY)

- Ilustrasi Dequeue ketika queue memiliki elemen hanya 1

Index Front = 0 Index Rear = 0

queue

Index	0	1	2	3	...	Max - 1
Isi	7					

Sebelum deqeueue

Isi = dequeue()

Index Front = -1 Index Rear = -1

queue

Index	0	1	2	3	...	Max - 1
Isi						

Setelah dequeue

Variable isi akan berisi 7

Penunjuk Front dan Rear berubah menjadi 0

MENGAMBIL/MENGHAPUS ELEMEN QUEUE (DEQUEUE) (ARRAY)

- Ilustrasi Dequeue ketika queue memiliki elemen lebih dari 1

Index Front = 0 Index Rear = 3

queue

Index	0	1	2	3	...	Max - 1
Isi	7	5	8	9		

Sebelum deqeueue

Isi = dequeue()

Index Front = 0 Index Rear = 2

queue

Index	0	1	2	3	...	Max - 1
Isi	5	8	9			

Setelah dequeue

Variable isi akan berisi 7

Hanya penunjuk Rear yang berubah (dikurangi 1)

MENGAMBIL/MENGHAPUS ELEMEN QUEUE (DEQUEUE) (ARRAY)

- Function **dequeue**

```
def dequeue(self):
        if self.isEmpty():
            print("Dequeue gagal karena queue kosong.")
        else:
            data = self.__content[self.__front]
            for i in range(1,self.__rear+1):
                self.__content[i-1] = self.__content[i]
            self.__rear = self.__rear - 1
            if self.__rear == -1:
                self.__front = -1
            return data
```

MELIHAT NILAI PADA POSISI TERDEPAN (ARRAY)

- Proses ini sering disebut Peek.
- Operasi Peek digunakan untuk melihat element yang berada di posisi Front tanpa melakukan Dequeue (hanya lihat saja, tanpa mengambilnya).
- Catatan: Lakukan operasi ini hanya ketika queue tidak kosong.

MELIHAT NILAI PADA POSISI TERDEPAN (ARRAY)

- Function **peek**

```
def peek(self):
        if self.isEmpty():
            print("Peek gagal karena queue kosong.")
        else:
            return self.__content[self.__front]
```

MENGOSONGKAN QUEUE (ARRAY)

- Operasi ini digunakan untuk mengosongkan queue.
- Langkah untuk mengosongkan queue ada 2 cara yaitu :
  1. Seting nilai Front dan Rear menjadi -1 atau
  2. Lakukan pengulangan dequeue sampai queue kosong.
- Karena queue yang diimplementasikan menggunakan array, maka cara ke-1 yang akan digunakan.

MENGOSONGKAN QUEUE (ARRAY)

- Function **empty**

```
def empty(self):
        self.__front = -1
        self.__rear = -1
```

CONTOH IMPLEMENTASI QUEUE (ARRAY)


```
q = queueArray(max=3)
print("#1 : Isi Queue : ", q)
q.enqueue(1)
print("#2 : Isi Queue : ", q)
q.enqueue(2)
print("#3 : Isi Queue : ", q)
q.enqueue(3)
print("#4 : Isi Queue : ", q)
q.enqueue(4)
print("#5 : Isi Queue : ", q)
print("#6 : Kosong ? ", q.isEmpty())
print("#7 : Penuh ? ", q.isFull())
data = q.dequeue()
print("#8 : Hasil Dequeue : ", data)
print("#9 : Isi Queue : ", q)
print("#10 : Peek : ", q.peek())
q.empty()
print("#11 : Isi Queue : ", q)
```

```
#1 : Isi Queue :  queueArray([ EmptyQueue ])
#2 : Isi Queue :  queueArray([1 ])
#3 : Isi Queue :  queueArray([1 2 ])
#4 : Isi Queue :  queueArray([1 2 3 ])
Enqueue Gagal. Queue Penuh.
#5 : Isi Queue :  queueArray([1 2 3 ])
#6 : Kosong ?  False
#7 : Penuh ?  True
#8 : Hasil Dequeue :  1
#9 : Isi Queue :  queueArray([2 3 ])
#10 : Peek :  2
#11 : Isi Queue :  queueArray([ EmptyQueue ])
```

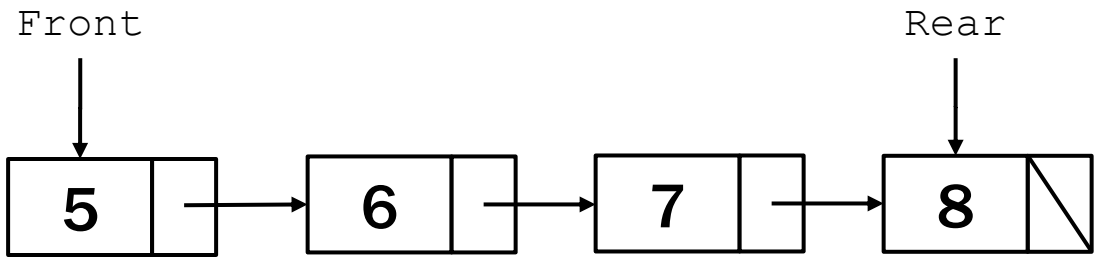
01158 - ALGORITMA DAN STRUKTUR DATA 2

QUEUE MENGGUNAKAN LINKED LIST

OLEH : ANDRI HERYANDI, M.T.

PRESENTASI QUEUE DENGAN LINKED LIST DI MEMORI

- Queue dapati diimplementasikan dengan menggunakan linked list dengan cara setiap penambahan selalu dilakukan di node belakang, dan penghapusan data hanya dilakukan di node depan.
- Sebuah queue memiliki node yang menunjuk ke node depan (**Front**) dan node yang menunjuk ke node belakang (**Rear**).



STRUKTUR DATA (LINKED LIST)

- Pendeklarasian class untuk queue menggunakan linked list (Bahasa Python).

```
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None

class queueLinkedList:
    def __init__(self):
        self.__front = None
        self.__rear = None
```

Pendeklarasian class Node

Pendeklarasian class queue

- Pendeklarasian variable/objek untuk queue menggunakan linked list (Bahasa Python).

```
q = queueLinkedList()
```

INISIALISASI QUEUE (LINKED LIST)

- Inisialisasi queue adalah proses mempersiapkan sebuah queue untuk digunakan.
- Proses yang dilakukan adalah memberikan nilai NIL/None kepada queue pada bagian Front dan Rear-nya yang menandakan bahwa queue kosong.
- Operasi ini dilakukan dalam method `__init__`

PEMERIKSAAN QUEUE KOSONG (LINKED LIST)

- Pemeriksaan ini digunakan untuk memeriksa apakah queue dalam keadaan kosong.
- Sebuah queue disebut kosong jika Front dari queue berisi nilai NIL/None.
- Pemeriksaan ini akan digunakan dalam operasi-operasi lain dalam queue.

PEMERIKSAAN QUEUE KOSONG (LINKED LIST)

- Function **isEmpty**

```
def isEmpty(self):
        return self.__front is None;
```

PEMERIKSAAN QUEUE PENUH (LINKED LIST)

Tidak ada pemeriksaan queue penuh jika queue menggunakan linked list, karena linked list tidak mempunyai batasan nilai maximal

BANYAK ELEMEN DALAM QUEUE (LINKED LIST)

- Proses ini digunakan untuk mengetahui berapa banyak elemen yang berada dalam queue.
- Pada queue yang diimplementasikan dengan linked list, banyak data bisa didapatkan dengan menelusuri linked list dari node pertama (front) sampai node terakhir (rear).

BANYAK ELEMEN DALAM QUEUE (LINKED LIST)

- Function **count**

```
def count(self):
        temp = self.__front
        banyakdata = 0
        while temp is not None:
            banyakdata = banyakdata + 1
            temp = temp.next
        return banyakdata
```

MENAMPILKAN ELEMEN QUEUE (LINKED LIST)

- Proses ini digunakan untuk menampilkan isi elemen queue
- Menampilkan isi queue dalam linked list dilakukan dengan melakukan penelusuran linked list untuk menampilkan elemen linked list satu per satu.
- Pada class ini, proses ini dilakukan dengan mengoverride function `__str__`.

MENAMPILKAN ELEMEN QUEUE (LINKED LIST)

- Procedure **display/__str__**

```
def __str__(self):
        out = "queueLinkedList(["
        if self.isEmpty():
            out = out + " EmptyQueue "
        else:
            temp = self.__front
            while temp is not None:
                out = out + str(temp.data) + " "
                temp = temp.next
        out = out + "]"
        out = out + ")"
        return out
```

MENAMBAH ELEMEN QUEUE / ENQUEUE (LINKED LIST)

- Proses ini sering disebut **Enqueue**.
- Operasi **enqueue** digunakan untuk menambahkan elemen baru di posisi **Rear**.
- Dalam queue yang menggunakan linked list, proses enqueue dilakukan dengan cara melakukan penambahan node di posisi **Rear** linked list.
- Algoritmanya adalah :
  1. Buat node baru (asumsikan bernama **newNode**). Isilah **element** dan **next** dari node baru tersebut.
  2. Periksa apakah queue masih kosong ?
     1. Jika queue masih kosong maka lakukan :
        1. isilah **Front** dari queue dengan alamat dari **newNode**.
        2. Isilah **Rear** dari queue dengan alamat dari **newNode**.
     2. Jika queue sudah ada element, maka lakukan :
        1. Isilah **Next** dari node **Rear** agar menunjuk ke **newNode**
        2. Pindahkan **Rear** dari queue ke alamat **newNode**

MENAMBAH ELEMEN QUEUE / ENQUEUE (LINKED LIST)

- Function **enqueue**

```
def enqueue(self, data):
        newNode = Node(data)
        if self.isEmpty():
            self.__front = newNode
            self.__rear = newNode
        else:
            self.__rear.next = newNode
            self.__rear = newNode
```

MENGAMBIL/MENGHAPUS ELEMEN QUEUE / DEQUEUE (LINKED LIST)

- Proses ini sering disebut Dequeue.
- Operasi dequeue digunakan untuk mengambil elemen yang berada di posisi depan (Front).
- Pada queue yang menggunakan linked list, proses dequeue sama saja dengan proses penghapusan node di awal linked list.
- Algoritmanya adalah :
  1. Periksa apakah queue kosong
     1. Jika queue kosong maka tampilkan pesan bahwa “Queue Kosong” dan operasi Dequeue gagal.
     2. Jika queue tidak kosong maka :
        1. Ambil elemen yang ada di posisi depan/front dari queue
        2. Simpan node yang ada di-front ke sebuah node lain (misalnya deletedNode)
        3. Periksa apakah queue hanya memiliki 1 element
           1. Jika queue hanya memiliki 1 elemen maka isilah Front dan Rear dari queue dengan nilai NIL.
           2. Jika queue memiliki element lebih dari 1 maka pindahkan node front ke next dari node front
        4. Hapus node yang ditunjuk oleh deletedNode

MENGAMBIL/MENGHAPUS ELEMEN QUEUE / DEQUEUE (LINKED LIST)

- Function **dequeue**

```
def dequeue(self):
        if self.isEmpty():
            print("Dequeue gagal karena queue kosong")
        else:
            elemenFront = self.__front.data
            nodeAwal = self.__front
            if self.__front == self.__rear:
                self.__front = None
                self.__rear = None
            else:
                self.__front = self.__front.next
            del nodeAwal
            return elemenFront
```

MELIHAT NILAI PADA POSISI DEPAN (LINKED LIST)

- Proses ini sering disebut Peek.
- Operasi Peek digunakan untuk melihat element yang berada di posisi depan tanpa melakukan dequeue (hanya lihat saja, tanpa mengambilnya).
- Pastikan ketika memanggil Peek, queue harus dalam kondisi tidak kosong.

MELIHAT NILAI PADA POSISI DEPAN (LINKED LIST)

- Function **peek**

```
def peek(self):
        if self.isEmpty():
            print("Peek gagal karena queue kosong")
        else:
            return self.__front.data
```

MENGOSONGKAN QUEUE (LINKED LIST)

- Operasi ini digunakan untuk mengosongkan queue.
- Langkah untuk mengosongkan queue ada 2 cara yaitu :
  1. Seting nilai Front dan Rear menjadi -1 atau
  2. Lakukan pengulangan dequeue sampai queue kosong.
- Karena queue yang diimplementasikan menggunakan linked list, maka cara ke-2 yang akan digunakan.

MENGOSONGKAN QUEUE (ARRAY)

- Function **empty**

```
def empty(self):
        while not self.isEmpty():
            self.dequeue()
```

CONTOH IMPLEMENTASI QUEUE (LINKED LIST)


```
q = queueLinkedList()
print("#1 : Isi Queue : ", q)
q.enqueue(15)
print("#2 : Isi Queue : ", q)
q.enqueue(30)
print("#3 : Isi Queue : ", q)
q.enqueue(45)
print("#4 : Isi Queue : ", q)
print("#5 : Kosong : ", q.isEmpty())
data = q.dequeue()
print("#6 : Hasil Dequeue : ", data)
print("#7 : Isi Queue : ", q)
q.empty()
print("#8 : Isi Queue : ", q)
```

```
#1 : Isi Queue :  queueLinkedList([ EmptyQueue ])
#2 : Isi Queue :  queueLinkedList([15 ])
#3 : Isi Queue :  queueLinkedList([15 30 ])
#4 : Isi Queue :  queueLinkedList([15 30 45 ])
#5 : Kosong :  False
#6 : Hasil Dequeue :  15
#7 : Isi Queue :  queueLinkedList([30 45 ])
#8 : Isi Queue :  queueLinkedList([ EmptyQueue ])
```

LATIHAN

- Buatlah sebuah sistem antrian untuk pemanggilan nasabah di Bank.
- Ketentuan :
  - Semua nasabah dianggap sama (tidak ada yang prioritas).
  - Bank memiliki 2 loket (Loket 1 dan Loket 2).
  - Menu di aplikasi :
    1. Tambah Antrian
    2. Loket 1 Memanggil
    3. Loket 2 Memanggil
    4. Keluar dari Aplikasi
  - Layar aplikasi akan menampilkan
    - No Antrian yang sedang dilayani oleh masing-masing Loket.
    - No Antrian pemanggilan selanjutnya

LATIHAN

Tampilan Aplikasi

SISTEM ANTRIAN S.A.T. BANK

Loket 1	Loket 2	No Berikutnya
X	Y	Z

```
Menu Pilihan :
1.  Tambah Antrian
2.  Loket 1 Memanggil
3.  Loket 2 Memanggil
4.  Keluar
```

Aturan :

1. Jika dipilih Menu 1, maka antrian bertambah (enqueue). Update nomor antrian berikutnya.
2. Jika dipilih Menu 2, maka ambil antrian terdepan (dequeue) oleh Loket 1. Tampilkan nomor antrian di Loket 1 dan update nomor antrian berikutnya.
3. Jika dipilih Menu 3, maka ambil antrian terdepan (dequeue) oleh Loket 2. Tampilkan nomor antrian di Loket 2 dan update nomor antrian berikutnya.
4. Jika memilih Menu 4 maka keluar dari aplikasi.

SEKIAN

Ada pertanyaan?

Silahkan tanyakan melalui Group Whatsapp, email, LMS atau comment video Youtube.

FORUM DISKUSI

LMS UNIKOM
https://lms.unikom.ac.id

Group Whatsapp Perkuliahan

Youtube Comments
https://unikom.id/YTCStrukturData