

## Latihan Soal Modul 8

### STRUKTUR DATA – Ganjil 2025/2026

#### Queue

#### 1 Klinik Sehat Yogi

Sebuah pusat kesehatan bernama "**Klinik Sehat Yogi**" sedang mengadakan program vaksinasi massal selama satu bulan. Program ini menarik ratusan warga dari berbagai kategori yang mendaftar setiap harinya. Klinik menerapkan **sistem antrean berbasis queue** untuk memastikan pengelolaan alur dan prioritas warga secara efektif.

##### 1.1 Aturan Pendaftaran

Setiap hari, warga yang berhasil mendaftar akan menerima kode antrean unik yang valid untuk satu hari. Setiap hari ada dua queue terpisah yang disediakan:

Tipe Prioritas	Deskripsi
Queue Prioritas	Dikhususkan untuk lansia (usia $\geq 60$ tahun) dan tenaga kesehatan. Mereka ditempatkan di urutan pertama untuk mendapatkan layanan.
Queue Normal	Untuk warga yang tidak memenuhi syarat prioritas.

1. Kapasitas Klinik Sehat Yogi adalah 5 orang per hari. Jika jumlah warga yang hadir melebihi kapasitas, warga di Queue Normal akan mendapatkan prioritas pelayanan pada hari berikutnya.
2. Jika seorang warga dengan kondisi darurat medis datang ke klinik, warga tersebut otomatis dipindahkan ke posisi paling depan di Queue Prioritas untuk segera dilayani. Warga lain dalam antrean prioritas akan tetap berada di antrean sesuai urutan awal mereka.

##### 1.2 ADT Program

Buatlah ADT queue (`queue.h`) dengan struktur berikut:

```
type address : pointer to elmqueue

type infotype <
    nama : string
    usia : integer
    pekerjaan : string
    prioritas : boolean
    nomorAntrean : integer
```

```
waktuDaftar : integer
>

type Queue <
  head : address
  tail : address
>

type elmqueue <
  info : infotype
  next : address
>
```

### 1.3 Implementasi Subprogram

Buatlah subprogram (`queue.cpp`) untuk queue nya.

```
procedure createQueue(in/out Q : Queue)
{I.S. -
 F.S. Terbentuk queue Q dengan head dan tail bernilai nil (nullptr).}
algorithm
  Q.head <- nil
  Q.tail <- nil
endprocedure

function isEmpty(Q : Queue) -> boolean
{I.S. Terdefinisi queue Q.
 F.S. Mengembalikan nilai true jika queue kosong, dan false jika sebaliknya.}
algorithm
  return Q.head == nil and Q.tail == nil
endprocedure

function allocate(nama : string, usia : integer, pekerjaan : string, nomorAntrean
  : integer, waktuDaftar : integer) -> address
{I.S. Terdefinisi nilai string usia, integer pekerjaan, string nomorAntrean, dan
 integer waktuDaftar.
 F.S. Mengembalikan elemen baru p dengan info yang sudah terisi dari nilai input,
 next sama dengan nil (nullptr).}
dictionary
  p : address
  x : infotype
algorithm
  alloc(p)
  x.nama <- nama
  x.usia <- usia
  x.pekerjaan <- pekerjaan
  x.prioritas <- usia >= 60 or pekerjaan == "tenaga_kesehatan"
  x.nomorAntrean <- nomorAntrean
  x.waktuDaftar <- waktuDaftar
  p.info <- x
  p.next <- nil

  return p
endprocedure

function front(Q : Queue) -> address
{I.S. Terdefinisi queue Q.
 F.S. Mengembalikan elemen pertama queue Q.}
algorithm
  return Q.head
```

**endprocedure**

**function** back(Q : Queue) -> address  
{I.S. Terdefinisi queue Q.  
F.S. Mengembalikan elemen terakhir queue Q.}

**algorithm**  
    **return** Q.tail  
**endprocedure**

**procedure** enqueue(in/out Q : Queue, in p : address)  
{I.S. Terdefinisi queue Q, dan elemen p.  
F.S. Elemen p ditambahkan ke dalam queue Q dengan ketentuan: (1) dimasukkan di depan antrian jika prioritas bernilai true, dan (2) dimasukkan di belakang antrian jika prioritas bernilai false.}

**dictionary**  
    temp : address  
    inputPasienPrioritas : **boolean**  
    semuaPasienPrioritas : **boolean**  
    tidakAdaPasienPrioritas : **boolean**

**algorithm**  
    **if not** isEmpty(Q) **then**  
        inputPasienBukanPrioritas <- **not** p.info.prioritas  
        semuaPasienPrioritas <- Q.tail.info.prioritas  
        tidakAdaPasienPrioritas <- **not** Q.head.info.prioritas  
    **endif**  
  
    **if** isEmpty(Q) **then**  
        Q.head <- p  
        Q.tail <- p  
    **else if** (inputPasienPrioritas **or** semuaPasienPrioritas) **then**  
        Q.tail.next <- p  
        Q.tail <- p  
    **else if** (tidakAdaPasienPrioritas) **then**  
        p.next <- Q.head  
        Q.head <- p  
    **else**  
        temp <- Q.head  
  
        **while** temp.next != nil **and** temp.next.info.prioritas **do**  
            temp <- temp.next  
        **endwhile**  
  
        p.next <- temp.next  
        temp.next <- p  
    **endif**  
**endprocedure**

**procedure** dequeue(in/out Q : Queue, out p : address)  
{I.S. Terdefinisi queue Q yang mungkin kosong.  
F.S. Elemen pertama pada queue Q terhapus dan disimpan dalam elemen p jika Q tidak kosong, dan p bernilai nil (nullptr) jika Q kosong.}

**algorithm**  
    **if not** isEmpty(Q) **then**  
        p <- Q.head  
        Q.head <- p.next  
  
        **if** Q.head == nil **then**  
            Q.tail <- nil  
        **endif**  
    **else**  
        p <- nil  
    **endif**  
**endprocedure**

```

function size(Q : Queue) -> integer
{I.S. Terdefinisi queue Q.
 F.S. Mengembalikan jumlah elemen dalam queue Q.}
dictionary
    count : integer
    p : address

algorithm
    count <- 0
    p <- Q.head

    while p != nil do
        count <- count + 1
        p <- p.next
    endwhile

    return count
endprocedure

procedure printPatient(in p : address)
{I.S. Terdefinisi elemen p.
 F.S. Menampilkan seluruh informasi warga yang ada dalam antrian Q dengan
 menampilkan nama, usia, pekerjaan, prioritas, dan nomor antrian nya.}

procedure serveQueue(in/out Q : Queue)
{I.S. Terdefinisi queue Q.
 F.S. Melayani warga dalam antrian Q maksimal 5 orang dengan mencetak data warga
 (menggunakan prosedur printPatient), serta menampilkan teks "Vaksinasi
 berhasil.", setelah itu menghapus warga dalam antrian.}

procedure reassignQueue(in/out Q : Queue)
{I.S. Terdefinisi antrian Q.
 F.S. Mengubah data warga non-prioritas menjadi prioritas.}

procedure emergencyHandle(in/out Q : Queue, in nomorAntrean : integer)
{I.S. Terdefinisi antrian Q.
 F.S. Mengubah data warga menjadi prioritas yang memiliki nomor antrian sama
 dengan `nomorAntrean`, serta menempatkan pada antrian paling depan.}

```

## 1.4 Main Program

Buatkan program utama (**main.cpp**) dengan menggunakan seluruh subprogram yang telah di implementasikan. Ikutilah alur program sesuai dengan alur di bawah (teks bold bergaris bawah adalah input).

### 1.4.1 enqueue

Berikut adalah tampilan jika menggunakan prosedur **enqueue**.

```

masukkan banyak pasien yang ingin di input: 11

pasien ke-1
nama: andi
usia: 25
pekerjaan: guru
nomor antrian: 1

pasien ke-2

```

```
nama: bela
usia: 42
pekerjaan: tenaga kesehatan
nomor antrean: 2

pasien ke-3
nama: citra
usia: 33
pekerjaan: mahasiswa
nomor antrean: 3

pasien ke-4
nama: doni
usia: 45
pekerjaan: petani
nomor antrean: 4

pasien ke-5
nama: eka
usia: 70
pekerjaan: pensiunan
nomor antrean: 2

...
```

#### 1.4.2 size

Berikut adalah tampilan jika memanggil prosedur `size`.

```
banyak pasien dalam antrean: 11
```

#### 1.4.3 serveQueue

Berikut adalah tampilan jika memanggil prosedur `serveQueue`.

```
-----
nama: bela
usia: 42
pekerjaan: tenaga_kesehatan
prioritas: 1
nomor antrean: 2
Vaksinasi berhasil.
-----
nama: eka
usia: 70
pekerjaan: pensiunan
prioritas: 1
nomor antrean: 5
Vaksinasi berhasil.
-----
nama: heri
usia: 66
pekerjaan: tenaga_kesehatan
prioritas: 1
nomor antrean: 8
Vaksinasi berhasil.
-----
nama: andi
```

```
usia: 25  
pekerjaan: guru  
prioritas: 0  
nomor antrean: 1  
Vaksinasi berhasil.
```

```
-----  
nama: citra  
usia: 33  
pekerjaan: mahasiswa  
prioritas: 0  
nomor antrean: 3  
Vaksinasi berhasil.
```

#### 1.4.4 emergencyHandle

Berikut adalah tampilan jika memanggil prosedur `emergencyHandle` dan `serveQueue`.

```
masukkan nomor antrean yang ingin di prioritaskan: 4
```

```
-----  
nama: doni  
usia: 45  
pekerjaan: petani  
prioritas: 1  
nomor antrean: 4  
Vaksinasi berhasil.
```

```
-----  
nama: faisal  
usia: 29  
pekerjaan: karyawan_swasta  
prioritas: 0  
nomor antrean: 6  
Vaksinasi berhasil.
```

```
-----  
nama: gina  
usia: 54  
pekerjaan: pedagang  
prioritas: 0  
nomor antrean: 7  
Vaksinasi berhasil.
```

```
-----  
nama: indah  
usia: 31  
pekerjaan: wirausaha  
prioritas: 0  
nomor antrean: 9  
Vaksinasi berhasil.
```

```
-----  
nama: joni  
usia: 40  
pekerjaan: supir  
prioritas: 0  
nomor antrean: 10  
Vaksinasi berhasil.
```

#### 1.4.5 reassignQueue

Berikut adalah tampilan jika memanggil prosedur `reassignQueue` dan `serveQueue`.

data prioritas pasien telah di update.

-----

nama: kiki  
usia: 58  
pekerjaan: ibu\_rumah\_tangga  
prioritas: 1  
nomor antrean: 11  
Vaksinasi berhasil.