03. Queue

ARNA FARIZA

YULIANA SETIOWATI

POLITEKNIK ELEKTRONIKA NEGERI SURABAYA

Capaian Pembelajaran

1. Mahasiswa mengerti konsep stack dan operasi pada queue.
2. Mahasiswa dapat menggunakan queue untuk memecahkan permasalahan pemrograman.

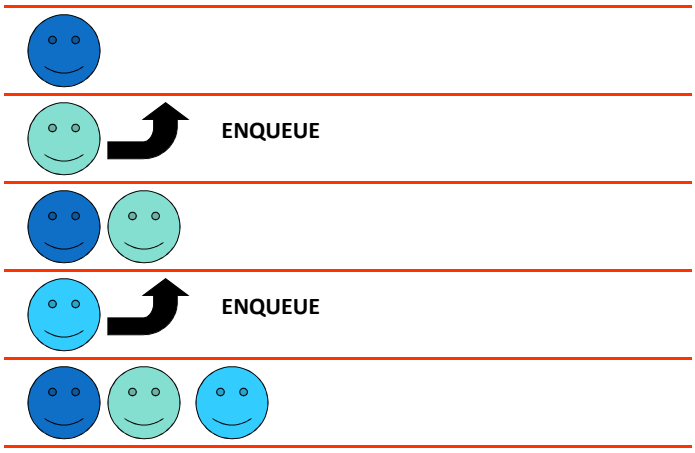
Materi

- Apakah Queue itu?
- Operasi pada Queue : ENQUEUE dan DEQUEUE

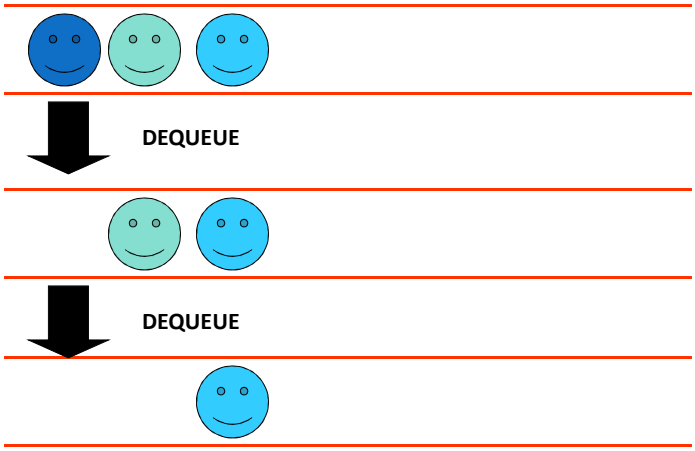
Apakah Queue itu?

- Merupakan konsep First In First Out (FIFO)
- Data yang disimpan pertama akan diambil lebih dahulu
- Implementasi queue dapat menggunakan array atau linked list
- Pada implementasi queue dengan array, kemungkinan queue bisa penuh
- Pada implementasi queue dengan linked list, queue tidak pernah penuh

Ilustrasi Queue



Ilustrasi Queue



Elemen Queue

- Elemen/item yang diletakkan ke penyimpan
- Penunjuk depan → front
- Penunjuk belakang → rear
- Jumlah elemen queue → count

Representasi Queue dengan Array

```
#define MAX 5
typedef int Itemtype
typedef struct {
   Itemtype item[MAX];
   int count;
   int front;
   int rear;
} Queue;
```

Operasi pada Queue

- Enqueue : menyimpan item ke Queue
- Dequeue : menghapus item dari Queue
- Inisialisasi : inisialisasi awal Queue
- Penuh : Cek apakah queue dalam kondisi kosong
- Kosong : Cek apakah queue dalam kondisi penuh

Operasi Inisialisasi

- Menginisialisasi count sama dengan 0
- Menginisialisasi front dan rear menunjuk ke indeks 0

```
void Inisialisasi (Queue *q)
{
  q->count=0;
    q->front = 0;
    q->rear = 0;
}
```

Operasi Penuh

- Melakukan pengecekan apakah queue Penuh (bila count bernilai MAX) atau Tidak (Jika penuh return value=1, sebaliknya return value=0)
- Digunakan bila melakukan operasi ENQUEUE

```
int Penuh (Queue *q)
{
    return (q->count==MAX);
}
```


Operasi Kosong

- Melakukan pengecekan apakah queue Kosong (bila count bernilai 0) atau Tidak (Jika kosong return value=1, sebaliknya return value=0)
- Digunakan bila melakukan operasi DEQUEUE

```
int Kosong (Queue *q)
{
    return (q->count==0);
}
```

Operasi ENQUEUE

- Bila array penuh (count=MAX), tidak dapat melakukan operasi Enqueue
- Menyimpan data pada posisi rear
- Setelah dilakukan penyimpanan, posisi rear di-increment (dengan circular queue), count di-increment

```
void Enqueue (Queue *q, Itemtype x)
{
   if(Penuh(q))
      printf("Queue Penuh, Data tidak dapat disimpan\n");
   else {
      q->item[q->rear]=x;
      q->rear =(q->rear+1) % MAX
      q->count++;
   }
}
```


Operasi DEQUEUE

- Bila array Kosong tidak dapat dilakukan operasi Dequeue
- Mengambil data pada posisi front
- Setelah mengambil data posisi front di-increment (dengan circular queue), count di-decrement

```
Itemtype Dequeue(Queue *q)
{
   Itemtype temp;
   if(Kosong(q)) {
      printf("Queue Kosong, tidak dapat mengambil data\n");
      return ' ';
   }
   else {
      temp=q->item[q->front];
      q->front = (q->front+1) % MAX;
      q->count--;
      return(temp);
   }
}
```

Rangkuman

- Queue merupakan konsep penyimpanan item secara FIFO, item yang masuk dahulu akan keluar lebih dahulu
- Elemen pada Queue terdiri dari : item yang disimpan di penyimpan, penunjuk depan (front), penunjuk belakang (rear) dan jumlah item (count)
- Operasi pada Queue : ENQUEUE dan DEQUEUE
- Operasi tambahan pada Queue : Inisialisasi, Penuh, Kosong

Latihan

Buatlah operasi shift menggunakan queue

- Input : bilangan desimal dan jumlah shift
- Output : bilangan desimal setelah shift

Contoh :

```
Masukkan bilangan desimal : 25
Masukkan jumlah shift : 3
Bilangan desimal setelah shift : 7
```