MODUL VIII QUEUE

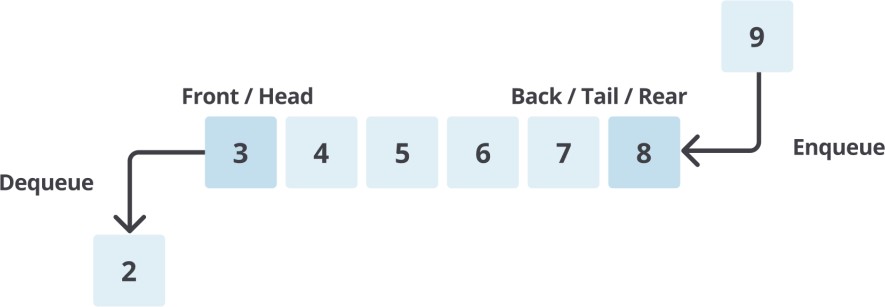
# TUJUAN PRAKTIKUM

* 1. Mahasiswa mampu menjelaskan definisi dan konsep dari queue
  2. Mahasiswa mampu menerapkan operasi tambah, menghapus pada queue
  3. Mahasiswa mampu menerapkan operasi tampil data pada queue

# DASAR TEORI

Queue adalah struktur data yang digunakan untuk menyimpan data dengan metode **FIFO** (First-In First-Out). Data yang pertama dimasukkan ke dalam queue akan menjadi data yang pertama pula untuk dikeluarkan dari queue. Queue mirip dengan konsep **antrian** pada kehidupan sehari-hari, dimana konsumen yang datang lebih dulu akan dilayani terlebih dahulu.

Implementasi queue dapat dilakukan dengan menggunakan array atau linked list. Struktur data queue terdiri dari dua pointer yaitu front dan rear. **Front/head** adalah pointer ke elemen pertama dalam queue dan **rear/tail/back** adalah pointer ke elemen terakhir dalam queue.



FIRST IN FIRST OUT (FIFO)

Perbedaan antara *stack* dan *queue* terletak pada aturan penambahan dan penghapusan elemen. Pada *stack*, operasi penambahan dan penghapusan elemen dilakukan di satu ujung yang disebut *top* (ujung atas). Elemen yang terakhir kali dimasukkan ke dalam *stack* akan berada di posisi paling atas dan akan menjadi elemen pertama yang dihapus. Sifat ini dikenal dengan istilah *LIFO* (Last In, First Out). Contoh analogi sederhana dari *stack* adalah tumpukan piring, di mana piring terakhir yang ditambahkan berada di posisi paling atas dan akan diambil atau dihapus terlebih dahulu.

Sebaliknya, pada *queue*, operasi penambahan dan penghapusan elemen dilakukan di dua ujung yang berbeda. Elemen baru ditambahkan di ujung belakang (*rear* atau *tail*), dan elemen dihapus dari ujung depan (*front* atau *head*). Proses ini mengikuti prinsip *FIFO* (First In, First Out), yang berarti elemen pertama yang dimasukkan ke dalam *queue* akan menjadi elemen pertama yang dikeluarkan. Dalam konteks *queue*, operasi penambahan elemen dikenal sebagai ***Enqueue***, dan operasi penghapusan elemen disebut ***Dequeue***.

Pada *Enqueue*, elemen ditambahkan di belakang *queue* setelah elemen terakhir yang ada, sementara pada *Dequeue*, elemen paling depan (*head*) dihapus, dan posisi *head* akan bergeser ke elemen berikutnya. Contoh penggunaan *queue* dalam kehidupan sehari-hari adalah antrean di kasir, di mana orang pertama yang datang adalah yang pertama dilayani.

# Operasi pada Queue

* enqueue() : menambahkan data ke dalam queue.
* dequeue() : mengeluarkan data dari queue.
* peek() : mengambil data dari queue tanpa menghapusnya.
* isEmpty() : mengecek apakah queue kosong atau tidak.
* isFull() : mengecek apakah queue penuh atau tidak.
* size() : menghitung jumlah elemen dalam queue.

# GUIDED

#include <iostream> using namespace std;

const int maksimalQueue = 5; // Maksimal antrian

int front = 0; // Penanda antrian

int back = 0; // Penanda

string queueTeller[5]; // Fungsi pengecekan

bool isFull() { // Pengecekan antrian penuh atau tidak

if (back == maksimalQueue) { return true; // =1

} else {

return false;

}

}

bool isEmpty() { // Antriannya kosong atau tidak

if (back == 0) { return true;

} else {

return false;

}

}

void enqueueAntrian(string data) { // Fungsi menambahkan antrian

if (isFull()) {

cout << "Antrian penuh" << endl;

} else {

if (isEmpty()) { // Kondisi ketika queue kosong

queueTeller[0] = data; front++;

back++;

} else { // Antrianya ada isi queueTeller[back] = data; back++;

}

}

}

void dequeueAntrian() { // Fungsi mengurangi antrian

if (isEmpty()) {

cout << "Antrian kosong" << endl;

} else {

for (int i = 0; i < back; i++) { queueTeller[i] = queueTeller[i + 1];

}

back--;

}

}

int countQueue() { // Fungsi menghitung banyak antrian

return back;

}

void clearQueue() { // Fungsi menghapus semua antrian

if (isEmpty()) {

cout << "Antrian kosong" << endl;

} else {

for (int i = 0; i < back; i++) { queueTeller[i] = "";

}

back = 0;

front = 0;

}

}

void viewQueue() { // Fungsi melihat antrian

cout << "Data antrian teller:" << endl; for (int i = 0; i < maksimalQueue; i++) {

if (queueTeller[i] != "") {

cout << i + 1 << ". " << queueTeller[i] <<

endl;

} else {

cout << i + 1 << ". (kosong)" << endl;

}

}

}

int main() {

enqueueAntrian("Andi"); enqueueAntrian("Maya"); viewQueue();

cout << "Jumlah antrian = " << countQueue() << endl; dequeueAntrian();

viewQueue();

cout << "Jumlah antrian = " << countQueue() << endl; clearQueue();

viewQueue();

cout << "Jumlah antrian = " << countQueue() << endl; return 0;

}

# UNGUIDED

* 1. Ubahlah penerapan konsep queue pada bagian guided dari array menjadi linked list
  2. Dari nomor 1 buatlah konsep antri dengan atribut Nama mahasiswa dan NIM Mahasiswa
  3. Modifikasi program pada soal 1 sehingga mahasiswa dapat diprioritaskan berdasarkan NIM (NIM yang lebih kecil didahulukan pada saat output).

Noted : Untuk data mahasiswa dan nim dimasukan oleh user