# LAPORAN PRAKTIKUM STRUKTUR DATA

# PERTEMUAN 8

# QUEUE



## Nama :

Reyner Atira Prasetyo (2311104057)

S1SE-07-02

## Dosen :

Wahyu Andi Saputra, S.Pd., M.Eng.

# PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

# FAKULTAS INFORMATIKA

**TELKOM UNIVERSITY PURWOKERTO**

**2024**

# TUJUAN

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

# TOOL

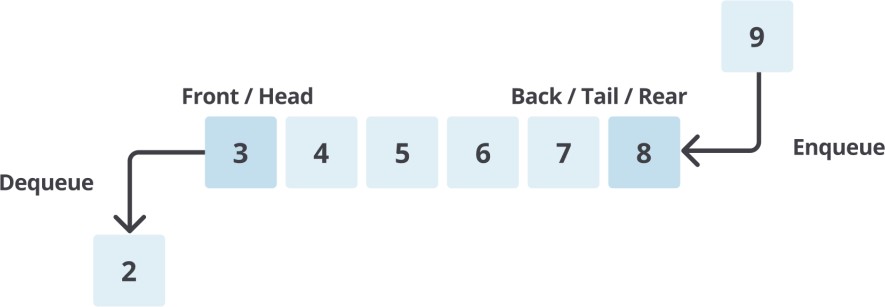
1. Visual Studio Code

2. GCC

# 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

# Guided1.cpp

# 

# Output :

# 

# Guided2.cpp

# 

# Output :

# 

# Guided3.cpp

# 

# Output :

# 

# UNGUIDED

# Unguided1.cpp

# 

# Output :

# 

# Unguided2.cpp

# 

# Output :

# 

# 

# 

# 

# Unguided3.cpp

# 

# Output :

# 

# 

# KESIMPULAN

# Pada praktikum ini, kami mempelajari implementasi struktur data Queue menggunakan bahasa C++. Queue adalah struktur data yang menerapkan prinsip FIFO (First In, First Out), di mana elemen yang pertama kali dimasukkan akan menjadi yang pertama kali keluar. Praktikum ini melibatkan pembuatan Queue menggunakan array dan linked list.

# Langkah pertama adalah mendefinisikan struktur data Queue yang terdiri dari dua operasi utama, yaitu enqueue (menambahkan elemen ke dalam antrian) dan dequeue (menghapus elemen dari antrian). Operasi enqueue memeriksa apakah antrian penuh, sedangkan operasi dequeue memeriksa apakah antrian kosong.

# Selanjutnya, implementasi dilakukan menggunakan array dengan ukuran tetap dan menggunakan linked list dinamis. Pada implementasi array, pointer untuk elemen depan dan belakang antrian dikelola untuk memudahkan proses enqueue dan dequeue. Pada implementasi linked list, setiap elemen diwakili oleh sebuah node yang memiliki pointer ke node berikutnya.

# Hasil praktikum menunjukkan bahwa implementasi Queue dengan array lebih efisien dalam hal penggunaan memori tetapi terbatas pada ukuran tetap, sementara linked list lebih fleksibel dalam mengelola ukuran antrian namun memiliki overhead lebih besar dalam manajemen memori.

# Secara keseluruhan, praktikum ini memberikan pemahaman yang lebih baik tentang cara kerja dan penerapan struktur data Queue dalam bahasa C++.