LAPORAN PRAKTIKUM
STRUKTUR DATA

MODUL VIII
QUEUE

Disusun Oleh :

NAMA : Damanik, Yohanes Geovan Ondova

NIM : 103112400022

Dosen

FAHRUDIN MUKTI WIBOWO

PROGRAM STUDI STRUKTUR DATA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

A. Dasar Teori

Pengertian Queue:

- Queue adalah struktur data yang menyerupai antrean, berprinsip dasar **FIFO (First in First Out)**, di mana proses yang pertama masuk akan diakses terlebih dahulu.
- Implementasi Queue dapat menggunakan tipe data *array* atau *linked list*.
- Pada implementasi *linked list*, proses *delete* selalu dilakukan di bagian *Head* (depan list) dan proses *insert* selalu dilakukan di bagian *Tail* (belakang list), atau sebaliknya, sesuai sifat FIFO.

Operasi dalam Queue:

- Insert (Enqueue): Operasi penyisipan yang selalu dilakukan di akhir (tail).
- Delete (Dequeue): Operasi pengambilan/penghapusan yang selalu dilakukan di awal (head).

Primitif primitif dalam Queue (Tabel)

- Contoh prototype primitif Queue dengan representasi tabel meliputi isFull, isEmpty, CreateQueue (dengan head=-1 dan tail = -1), enQueue, deQueue dan viewQueue

Guided (berisi screenshot source code & output program disertai penjelasannya)

Guided 1

queue.cpp

```
#include "queue.h"
#include <iostream>

using namespace std;

void createQueue(Queue &Q) {
    Q.head = 0;
    Q.tail = 0;
    Q.count = 0;
}

bool isEmpty(Queue Q) {
    return Q.count == 0;
}

bool isFull(Queue Q) {
    return Q.count == MAX_QUEUE;
}

void enqueue(Queue &Q, int x) {
    if (!isFull(Q)) {
        Q.info[Q.tail] = x;
        Q.tail = (Q.tail +1) %MAX_QUEUE;
        Q.count++;
    } else {
        cout << "Antrean Penuh!" << endl;
    }
}

int dequeue(Queue &Q) {
    if (!isEmpty(Q)) {
        int x = Q.info[Q.head];
        Q.head = (Q.head +1) %MAX_QUEUE;
        Q.count--;
        return x;
    } else {
        cout << "Antrean Kosong!" << endl;
        return -1;
    }
}

void printInfo(Queue Q) {
    cout << "Isi Queue: [ ";
    if (!isEmpty(Q)) {
        int i = Q.head;
```

```
        int n = 0;
        while (n < Q.count) {
            cout << Q.info[i] << " ";
            i = (i + 1) % MAX_QUEUE;
            n++;
        }
    }
    cout << "]" << endl;
}
```

queue.h

```
#ifndef QUEUE_H // Jika QUEUE_H belum didefinisikan
#define QUEUE_H // Maka definisikan QUEUE_H untuk mencegah inklusi ganda

#define MAX_QUEUE 5
struct Queue {
    int info[MAX_QUEUE];
    int head;
    int tail;
    int count;
};

void createQueue(Queue &Q);
bool isEmpty(Queue Q);
bool isFull(Queue Q);
void enqueue(Queue &Q, int x);
int dequeue(Queue &Q);
void printInfo(Queue Q);

#endif
```

main.cpp

```
#include <iostream>
#include "queue.h"
#include "queue.cpp"

using namespace std;

int main()
{
    Queue Q;

    createQueue(Q);
```

```
    printInfo(Q);

    cout << "\n Enqueue 3 Elemen" << endl;
    enqueue(Q, 5);
    printInfo(Q);
    enqueue(Q, 2);
    printInfo(Q);
    enqueue(Q, 7);
    printInfo(Q);

    cout << "\n Dequeue 1 Elemen" << endl;
    cout << "Elemen keluar: " << dequeue(Q) << endl;
    printInfo(Q);

    cout << "\n Enqueue 1 Elemen" << endl;
    enqueue(Q, 4);
    printInfo(Q);

    cout << "\n Dequeue 2 Elemen" << endl;
    cout << "Elemen keluar: " << dequeue(Q) << endl;
    cout << "Elemen keluar: " << dequeue(Q) << endl;
    printInfo(Q);

    return 0;
}
```

Output

```
PS C:\Users\Lenovo\Documents\STUKTUR DATA MODUL 8> cd "c:\Users\Lenovo\Documents\STUKTUR
{ g++ main.cpp -o main } ; if ($?) { .\main }
Isi Queue: [ ]

 Enqueue 3 Elemen
Isi Queue: [ 5 ]
Isi Queue: [ 5 2 ]
Isi Queue: [ 5 2 7 ]

 Dequeue 1 Elemen
Elemen keluar: 5
Isi Queue: [ 2 7 ]

 Enqueue 1 Elemen
Isi Queue: [ 2 7 4 ]

 Dequeue 2 Elemen
Elemen keluar: 2
Elemen keluar: 7
Isi Queue: [ 4 ]
PS C:\Users\Lenovo\Documents\STUKTUR DATA MODUL 8>
```

Deskripsi:

Program ini merupakan implementasi struktur data antrian (queue) menggunakan array sirkular dengan batas maksimal 5 elemen, yang terdiri dari tiga bagian: header file (queue.h) untuk deklarasi fungsi dan struktur, implementasi fungsi (queue.cpp), serta program utama (main.cpp) yang menguji operasi antrian. Fungsi-fungsi yang tersedia meliputi inisialisasi antrian (createQueue), pengecekan kondisi penuh atau kosong (isFull, isEmpty), penambahan elemen (enqueue), penghapusan elemen (dequeue), dan pencetakan isi antrian (printInfo). Program utama mendemonstrasikan alur kerja antrian dengan melakukan beberapa operasi enqueue dan dequeue secara berurutan, menampilkan perubahan isi antrian setiap langkah, serta menangani kasus antrian penuh atau kosong dengan pesan peringatan yang sesuai.

B. Unguided/Tugas (berisi screenshot source code & output program disertai penjelasannya)

Unguided 1

queue.cpp

```
#include <iostream>
#include "queue.h"
using namespace std;

void createQueue(Queue &Q) {
    Q.head = -1;
    Q.tail = -1;
}

bool isEmptyQueue(Queue Q) {
    return (Q.head == -1);
}

bool isFullQueue(Queue Q) {
    return (Q.tail == MAX - 1);
}

void enqueue(Queue &Q, infotype x) {
    if (isFullQueue(Q)) {
        cout << "Queue penuh!" << endl;
    } else {
        if (isEmptyQueue(Q)) {
            Q.head = 0;
            Q.tail = 0;
        } else {
            Q.tail++;
        }
        Q.info[Q.tail] = x;
    }
}

infotype dequeue(Queue &Q) {
```

```
    if (isEmptyQueue(Q)) {
        cout << "Queue kosong!" << endl;
        return -1;
    } else {
        infotype x = Q.info[Q.head];
        for (int i = Q.head; i < Q.tail; i++) {
            Q.info[i] = Q.info[i + 1];
        }
        Q.tail--;

        if (Q.tail < Q.head) {
            createQueue(Q);
        }
        return x;
    }
}

void printInfo(Queue Q) {
    cout << Q.head << " - " << Q.tail << " \t | ";

    if (isEmptyQueue(Q)) {
        cout << "(empty)" << endl;
    } else {
        for (int i = Q.head; i <= Q.tail; i++) {
            cout << Q.info[i] << " ";
        }
        cout << endl;
    }
}
```

queue.h

```
#ifndef QUEUE_H
#define QUEUE_H

const int MAX = 5;
typedef int infotype;

struct Queue {
    infotype info[MAX];
    int head;
    int tail;
};

void createQueue(Queue &Q);
bool isEmptyQueue(Queue Q);
bool isFullQueue(Queue Q);
void enqueue(Queue &Q, infotype x);
```

```
infotype dequeue(Queue &Q);
void printInfo(Queue Q);

#endif
```

Main.cpp

```
#include <iostream>
#include "queue.h"
using namespace std;

int main() {
    Queue Q;
    createQueue(Q);

    cout<<"----------------------"<<endl;
    cout<<" H - T \t | Queue info"<<endl;
    cout<<"----------------------"<<endl;

    printInfo(Q);
    enqueue(Q, 5); printInfo(Q);
    enqueue(Q, 2); printInfo(Q);
    enqueue(Q, 7); printInfo(Q);
    dequeue(Q);   printInfo(Q);
    enqueue(Q, 4); printInfo(Q);
    dequeue(Q);   printInfo(Q);
    dequeue(Q);   printInfo(Q);

    return 0;
}
```

Output

```
PS C:\Users\Lenovo\Documents\STUKTUR DATA MODUL 8> cd unguided
PS C:\Users\Lenovo\Documents\STUKTUR DATA MODUL 8\unguided> g++ main.cpp queue.cpp
PS C:\Users\Lenovo\Documents\STUKTUR DATA MODUL 8\unguided> g++ main.cpp queue.cpp -o main
PS C:\Users\Lenovo\Documents\STUKTUR DATA MODUL 8\unguided> ./main
---------------------
 H - T   | Queue info
---------------------
-1 - -1         | (empty)
0 - 0    | 5
0 - 1    | 5 2
0 - 2    | 5 2 7
0 - 1    | 2 7
0 - 2    | 2 7 4
0 - 1    | 7 4
0 - 0    | 4
PS C:\Users\Lenovo\Documents\STUKTUR DATA MODUL 8\unguided>
```

Deskripsi:

Ketiga program ini membentuk implementasi struktur data queue (antrian) menggunakan array statis dengan kapasitas maksimal 5 elemen, yang terdiri dari: queue.h sebagai header file yang mendeklarasikan konstanta, tipe data, struktur Queue, dan prototipe fungsi; queue.cpp yang berisi definisi fungsi-fungsi operasi antrian seperti createQueue, isEmptyQueue, isFullQueue, enqueue (menambahkan elemen di belakang), dequeue (menghapus elemen di depan dengan menggeser sisa elemen ke depan), dan printInfo untuk mencetak isi antrian; serta main.cpp yang menjadi program utama untuk menguji fungsi antrian dengan serangkaian operasi enqueue dan dequeue sambil menampilkan perubahan indeks head-tail dan isi antrian secara visual.

C. Kesimpulan

Implementasi struktur data queue (antrian) dapat dilakukan menggunakan array dengan dua pendekatan utama, yaitu array sirkular (seperti pada Guided) dan array linear (seperti pada Unguided). Kedua pendekatan berhasil mengimplementasikan prinsip dasar FIFO (First-In-First-Out), namun dengan karakteristik operasi yang berbeda. Array sirkular lebih efisien karena menggunakan konsep modulus untuk mengelola indeks head dan tail, sehingga memanfaatkan memori secara optimal dan menghindari penggeseran elemen saat dequeue. Sementara itu, implementasi linear lebih sederhana secara logika, tetapi kurang efisien karena mengharuskan penggeseran seluruh elemen ke depan setiap kali terjadi operasi dequeue. Praktikum ini juga menegaskan pentingnya fungsi pengecekan kondisi penuh (*isFull*) dan kosong (*isEmpty*) untuk menjaga keamanan operasi pada struktur data antrian.

D. Referensi

https://www.geeksforgeeks.org/cpp/queue-cpp-stl/

https://www.w3schools.com/cpp/cpp_queues.asp