LAPORAN PRAKTIKUM
STRUKTUR DATA

MODUL VIII
QUEUE

Disusun Oleh :
NAMA : ZAKI MAULA DHIA
NIM : 103112400127

Dosen
FAHRUDIN MUKTI WIBOWO

PROGRAM STUDI STRUKTUR DATA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2025

A. Dasar Teori

Queue adalah struktur data linier yang menerapkan prinsip oeprasi dimana elemen data yang masuk pertama akan keluar lebih dulu. Prinsip ini dikenal dengan istilah FIFO(First In, First Out). Konsepnya hampir sama dengan stack, bedanya adalah operasi penambahan dan penghapusan pada ujung yang berbeda. Penghapusan dilakukan pada bagian depan dan penambahan dilakukan pada bagian belakang.

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

Guided 1

```
// file queue.h
#ifndef QUEUE_H
#define QUEUE_H

#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
```

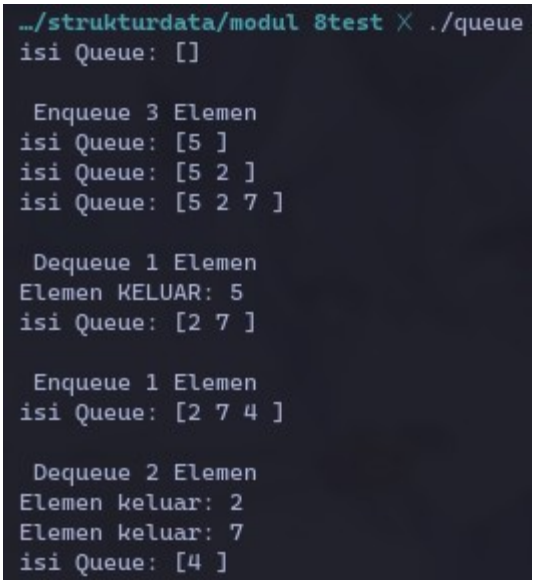
```
// file queue.cpp
#include "queue.h"
```

```
#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;
}
```

```
// file main.cpp
#include <iostream>
#include "queue.h"
#include "queue.cpp"
using namespace std;

int main()
{
    Queue Q;
    createQueue(Q);
    printInfo(Q);
    cout << "\nEnqueue 3 Elemen" << endl;
    enqueue(Q, 5);
    printInfo(Q);
    enqueue(Q, 2);
    printInfo(Q);
    enqueue(Q, 7);
    printInfo(Q);

    cout << "\nDequeue 1 Elemen" << endl;
    cout << "Elemen KELUAR: " << dequeue(Q) << endl;
    printInfo(Q);

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

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

    return 0;
}
```

Screenshots Output



Deskripsi:

Program ini menggunakan struktur data queue, karena struktur data queue menggunakan prinsip First In First Out(FIFO) maka program akan mengeluarkan isi queue dimulai dari head menggunakan function dequeue(), dan untuk memasukan elemen kedalam queue digunakan function enqueue(), seperti program di atas queue diisi dengan elemen 5,2,7 menggunakan enqueue, dan kemudian dikeluarkan dari yang pertama masuk.

C. Unguided (berisi screenshot source code & output program disertai penjelasannya)

Unguided 1

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

int main() {
```

```
    cout << "Hello World" << endl;

    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;
}
```

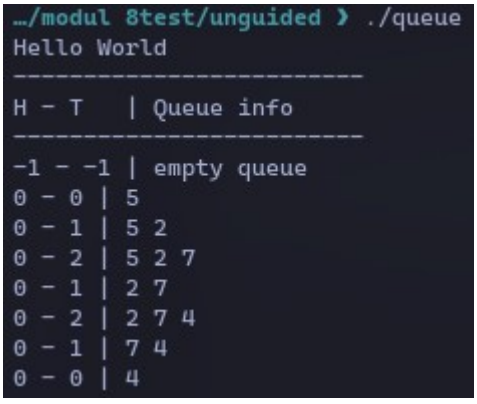
```
// queue.h
#ifndef QUEUE_H
#define QUEUE_H

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
```

```
// 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.tail == -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 -999;
    } else {
        infotype temp = 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) {
            Q.head = -1;
            Q.tail = -1;
        }
        return temp;
    }
}

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

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

Screenshots Output



Deskripsi:

Program tersebut memiliki fungsi yang sama dengan guided 1 diatas, dengan menerapkan mekanisme queue alternatif 1 head diam sedangkan tail bergerak, didapatkan hasil output seperti diatas, angka yang pertama dimasukan akan diam dan yang bergerak adalah angka yang selanjutnya dimasukan.

Unguided 2

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

int main() {

    cout << "Hello World" << endl;

    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;
}
```

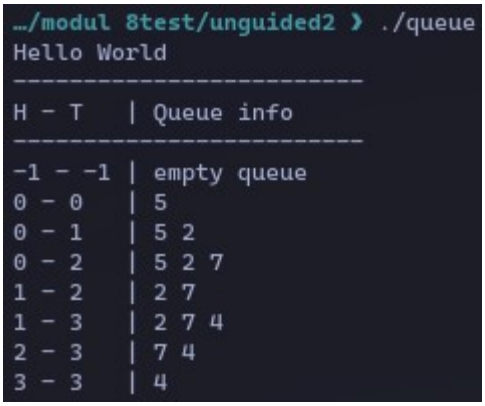
```
// queue.h
#ifndef QUEUE_H
```

```
#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
```

```
// 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 + 1) % MAX == Q.head);
}

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.tail + 1) % MAX;
        }
        Q.info[Q.tail] = x;
    }
}

infotype dequeue(Queue &Q) {
    if (isEmptyQueue(Q)) {
        cout << "Queue kosong!" << endl;
        return -999;
    }

    infotype hasil = Q.info[Q.head];

    if (Q.head == Q.tail) {
        Q.head = -1;
        Q.tail = -1;
    } else {
        Q.head = (Q.head + 1) % MAX;
    }

    return hasil;
}

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

    if (isEmptyQueue(Q)) {
        cout << "empty queue" << endl;
        return;
    }

    int i = Q.head;
    while (true) {
        cout << Q.info[i] << " ";
        if (i == Q.tail) break;
        i = (i + 1) % MAX;
    }
    cout << endl;
}
```

Screenshots Output



Deskripsi:

Program tersebut adalah implementasi dari mekanisme queue alternatif 2 dimana head dan tail sama-sama bergerak sehingga value H dan T ikut bertambah tidak seperti program unguided 1, hanya T yang bertambah.

Unguided 3

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

int main() {

    cout << "Hello World" << endl;

    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;
}
```

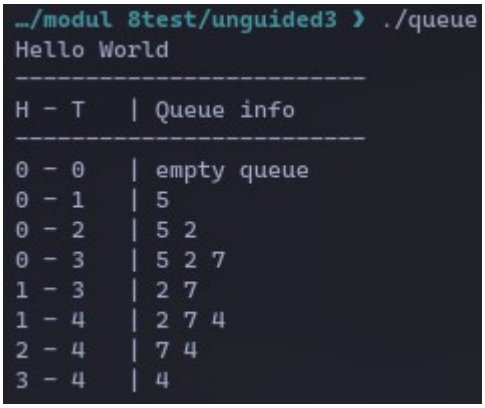
```
// queue.h
#ifndef QUEUE_H
#define QUEUE_H

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
```

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

void createQueue(Queue &Q) {
    Q.head = 0;
```

```
    Q.tail = 0;
}

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

bool isFullQueue(Queue Q) {
    return ((Q.tail + 1) % MAX == Q.head);
}

void enqueue(Queue &Q, infotype x) {
    if (isFullQueue(Q)) {
        cout << "Queue penuh!" << endl;
        return;
    }

    Q.info[Q.tail] = x;
    Q.tail = (Q.tail + 1) % MAX;
}

infotype dequeue(Queue &Q) {
    if (isEmptyQueue(Q)) {
        cout << "Queue kosong!" << endl;
        return -999;
    }

    infotype temp = Q.info[Q.head];
    Q.head = (Q.head + 1) % MAX;
    return temp;
}

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

    if (isEmptyQueue(Q)) {
        cout << "empty queue" << endl;
        return;
    }

    int i = Q.head;
    while (i != Q.tail) {
        cout << Q.info[i] << " ";
        i = (i + 1) % MAX;
```

```
    }
    cout << endl;
}
```

Screenshots Output



Deskripsi:

Program tersebut menerapkan mekanisme queue alternatif 3 yang dimana head dan tail berputar, dalam mekanisme queue alternatif 3 harus terdapat satu slot yang kosong sehingga walaupun diset MAX = 5 yang hanya bisa di isi cuman 4.

D. Kesimpulan

kesimpulan dari materi queue dalam modul 8 dapat diambil kesumpulan, mekanisme queue ada memiliki tiga alternatif yang memiliki tujuan yang sama yaitu menerapkan prinsip FIFO, hanya berbeda pada cara mengelola posisi head dan tail. Struktur data queue mempunyai keunggulan yang mendukung pola FIFO(First In First Out), sehingga sangat ideal untuk antrian nyata.

E. Referensi

- Tresnawati, S., Ekawati, N., Anggreini, N. L., & Rohmayani, D. (2024). Pelatihan Mengenalkan Struktur Data pada SMK Negeri 2 Cimahi. *PUAN INDONESIA*, *6*(1), 341-350.
- Melladia, M., Efendi, G., & Zahmi, A. (2024). *Algoritma dan Struktur Data dengan Pemograman Pascal dan Phyton*. CV. Gita Lentera.