LAPORAN PRAKTIKUM
STRUKTUR DATA

MODUL 8

Disusun Oleh :
NAMA : IBTIDA ZADA UTOMO
NIM : 103112430037

Dosen
FAHRUDIN MUKTI WIBOWO

PROGRAM STUDI STRUKTUR DATA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO

2025

A. Dasar Teori

Queue merupakan salah satu struktur data linear yang bekerja menggunakan prinsip FIFO (First In, First Out), di mana elemen yang pertama kali masuk akan menjadi elemen pertama yang keluar. Operasi dasar dalam queue meliputi enqueue untuk menambahkan elemen ke bagian belakang antrian dan dequeue untuk menghapus elemen dari bagian depan antrian. Untuk mengetahui kondisi antrian, digunakan fungsi isEmpty sebagai penanda bahwa antrian kosong dan isFull sebagai penanda bahwa antrian sudah penuh. Implementasi queue berbasis array dapat dibuat dalam dua bentuk, yaitu linear queue dan circular queue. Linear queue sederhana namun kurang efisien karena ruang kosong di awal array tidak dapat digunakan kembali setelah dilakukan dequeue. Sebaliknya, circular queue memanfaatkan operasi modulo sehingga indeks head dan tail bergerak secara melingkar, membuat penggunaan memori lebih optimal karena semua posisi array dapat digunakan kembali. Pada program yang diuji, ketiga versi queue—versi shifting, versi head-moving, dan versi circular—menunjukkan bagaimana pendekatan yang berbeda mempengaruhi efisiensi dan mekanisme kerja queue pada array statis.

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

Guided 1

```
#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
```

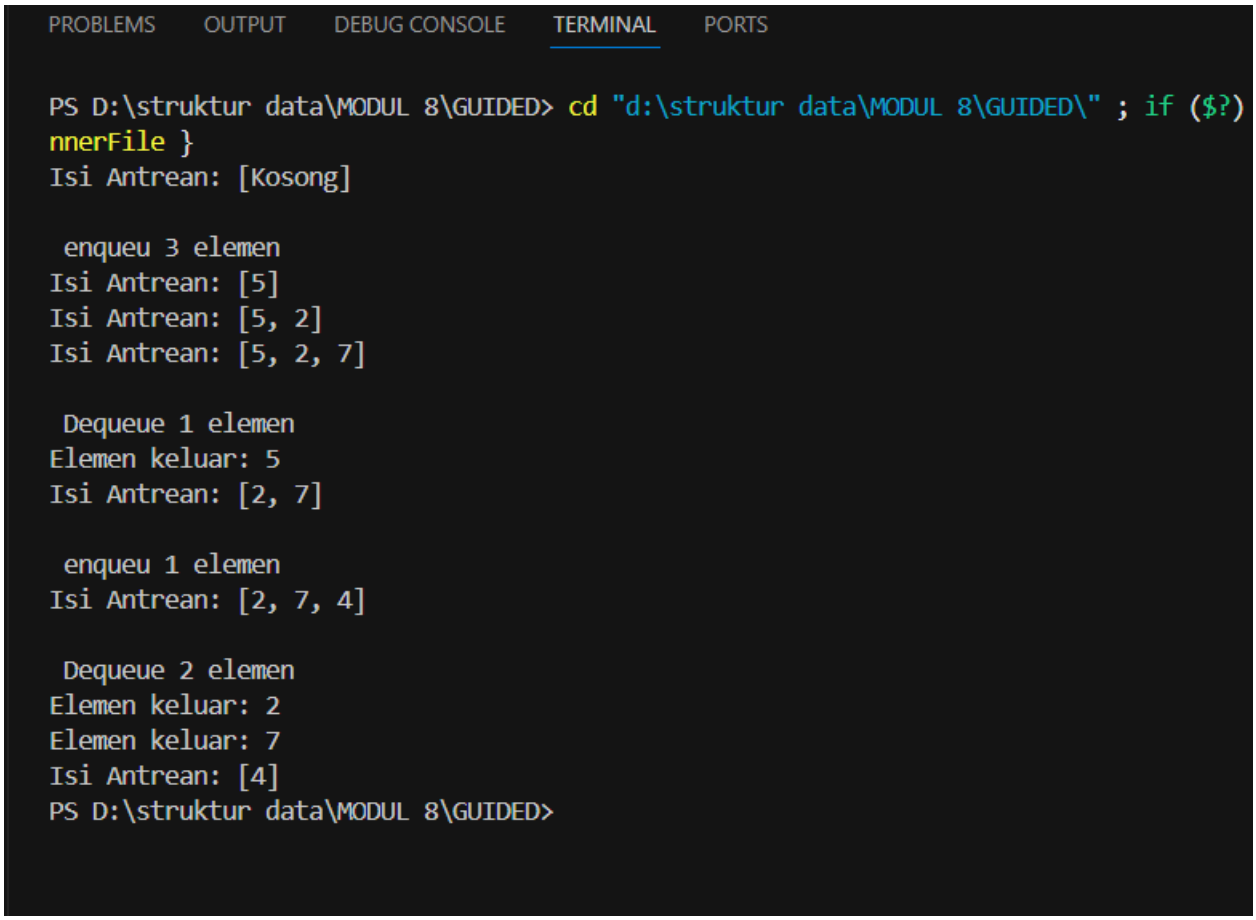
```
#include "queue.h"
#include <iostream>

using namespace std;

void createQueue(Queue &Q) {
    Q.head = 0;
    Q.tail = -1;
    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.tail = (Q.tail + 1) % MAX_QUEUE;
       Q.info[Q.tail] = x;
       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) {
    if (isEmpty(Q)) {
        cout << "Isi Antrean: [Kosong]" << endl;
        return;
    }

    cout << "Isi Antrean: [";

    int i = Q.head;
    for (int n = 0; n < Q.count; n++) {
        cout << Q.info[i];
        if (n < Q.count - 1) cout << ", ";
        i = (i + 1) % MAX_QUEUE;
    }

    cout << "]" << endl;
}
```

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

using namespace std;

int main() {
    Queue Q;

    createQueue(Q);
    printInfo(Q);

    cout <<"\n enqueu 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 enqueu 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;

}
```

Screenshots Output



Deskripsi: program diatas adalah pogram yang mengimplementasikan circular queue menggunakan array yang memiliki ukuran 5 element program ini terdiri dari 3 file yaitu queue.h, queue.cpp, dan main.cpp

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

Unguided 1

queue.h

```
#ifndef QUEUE_H
#define QUEUE_H

typedef int infotype;

typedef struct {
    infotype info[5];
    int head, tail;
} Queue;

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 = -1;
}

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

bool isFullQueue(Queue Q) {
    return (Q.tail == 4);
}

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

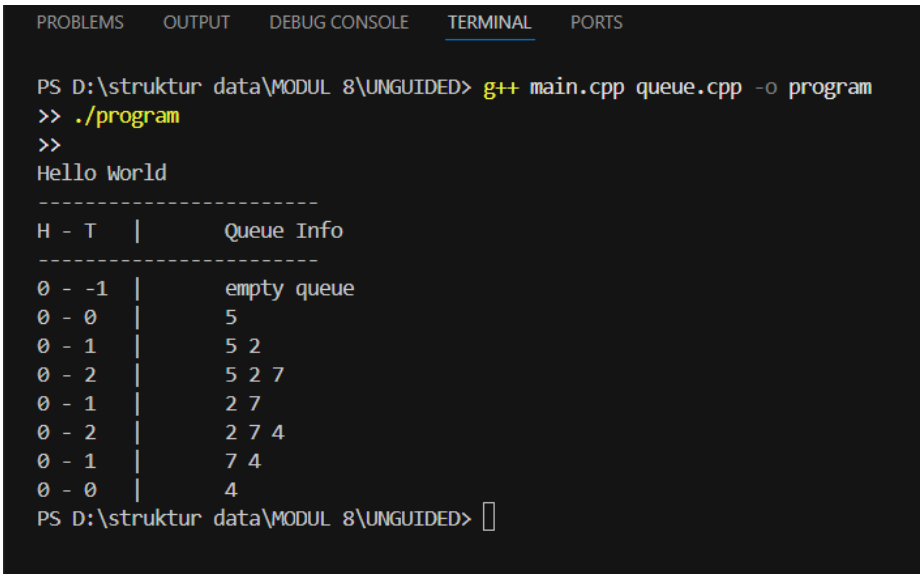
```
        }
        Q.tail--;
        return x;
    }
}

void printInfo(Queue Q) {
    cout << Q.head << " - " << Q.tail << "\t|\t";
    if (isEmptyQueue(Q)) {
        cout << "empty queue" << endl;
    } else {
        for (int i = Q.head; i <= Q.tail; i++) {
            cout << Q.info[i] << " ";
        }
        cout << endl;
    }
}
```

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|\tQueue 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;
}
```

Screenshots Output



Deskripsi: Program ini adalah implementasi struktur data Queue (antrian) berbasis array statis dengan kapasitas maksimal 5 elemen. Queue menggunakan konsep FIFO (First In, First Out), yaitu data yang masuk pertama akan keluar lebih dulu.

Unguided 2

```
#ifndef QUEUE_H
#define QUEUE_H

typedef int infotype;

typedef struct {
    infotype info[5];
    int head, tail;
} Queue;

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
```

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

void createQueue(Queue &Q) {
    Q.head = 0;
    Q.tail = -1;
}

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

bool isFullQueue(Queue Q) {
    return Q.tail == 4;
}

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

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

```
void printInfo(Queue Q) {
    cout << Q.head << " - " << Q.tail << "\t|\t";
    if (isEmptyQueue(Q)) {
        cout << "empty queue" << endl;
    } else {
        for (int i = Q.head; i <= Q.tail; i++) {
            cout << Q.info[i] << " ";
        }
        cout << endl;
    }
}
```

```
#include <iostream>
#include "queue2.h"
#include "queue2.cpp"
using namespace std;

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

    printInfo(Q);
    enqueue(Q, 10); printInfo(Q);
    enqueue(Q, 20); printInfo(Q);
    enqueue(Q, 30); printInfo(Q);
    dequeue(Q);     printInfo(Q);
    enqueue(Q, 40); printInfo(Q);

    return 0;
}
```

Screenshots Output

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

PS D:\struktur data\MODUL 8\UNGUIDED> cd "d:\struktur data\MODUL 8\UNGUIDED\" ; if ($?) {
0 - -1  |       empty queue
0 - 0   |       10
0 - 1   |       10 20
0 - 2   |       10 20 30
1 - 2   |       20 30
1 - 3   |       20 30 40
PS D:\struktur data\MODUL 8\UNGUIDED>
```

Deskripsi: Program ini merupakan implementasi struktur data Queue (antrian) menggunakan array statis berukuran 5 elemen. Queue bekerja dengan prinsip FIFO (First In, First Out), yaitu data yang masuk lebih dulu akan keluar terlebih dahulu. program juga dibagi menjadi 3 file utama yaitu queue.h, queue.cpp, dan main.cpp yang punya peran berbeda

- queue2.h untuk deklarasi struktur dan fungsi
- queue2.cpp untuk implementasi fungsi queue
- main.cpp untuk program utama

Unguided 3

```h
#ifndef QUEUE_H
#define QUEUE_H

const int MAX = 5;

struct Queue {
    int data[MAX];
    int head;
    int tail;
    int count;
};

void initQueue(Queue &Q);
bool isEmpty(Queue Q);
bool isFull(Queue Q);
void enqueue(Queue &Q, int x);
int dequeue(Queue &Q);
void printQueue(Queue Q);

#endif
```

```cpp
#include "queue3.h"
#include <iostream>
using namespace std;

void initQueue(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;
}

void enqueue(Queue &Q, int x) {
    if (isFull(Q)) {
        cout << "Queue penuh!\n";
        return;
    }
    Q.data[Q.tail] = x;
    Q.tail = (Q.tail + 1) % MAX;
    Q.count++;
}

int dequeue(Queue &Q) {
    if (isEmpty(Q)) {
        cout << "Queue kosong!\n";
        return -1;
    }
    int x = Q.data[Q.head];
    Q.head = (Q.head + 1) % MAX;
    Q.count--;
    return x;
}

void printQueue(Queue Q) {
    if (isEmpty(Q)) {
        cout << "Queue kosong!\n";
```

```
        return;
    }

    cout << "Isi Queue: ";
    int idx = Q.head;
    for (int i = 0; i < Q.count; i++) {
        cout << Q.data[idx] << " ";
        idx = (idx + 1) % MAX;
    }
    cout << endl;
}
```

```
#include <iostream>
#include "queue3.h"
#include "queue3.cpp"
using namespace std;

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

    enqueue(Q, 10);
    enqueue(Q, 20);
    enqueue(Q, 30);

    printQueue(Q);

    cout << "Dequeued: " << dequeue(Q) << endl;

    printQueue(Q);

    enqueue(Q, 40);
    enqueue(Q, 50);
    enqueue(Q, 60);

    printQueue(Q);

    return 0;
}
```

Screenshots Output

```
PS D:\struktur data\MODUL 8\UNGUIDED> cd "d:\struktur data\MODUL 8\UNGUIDED\" ; if ($?) { g++ main3.cpp -o main3 } ; if ($?) { .\main3 }
Isi Queue: 10 20 30
Dequeued: 10
Isi Queue: 20 30
Isi Queue: 20 30 40 50 60
PS D:\struktur data\MODUL 8\UNGUIDED>
```

Deskripsi:

Program ini adalah implementasi circular queue menggunakan array statis berukuran 5 elemen. Circular queue memungkinkan antrian berputar menggunakan operasi modulo sehingga lebih efisien, karena tidak perlu menggeser elemen ketika terjadi dequeue. Program ini terdiri dari tiga file utama: queue3.h, queue3.cpp, dan main3.cpp. Setiap file memiliki perannya masing-masing.

C. Kesimpulan

Berdasarkan praktikum yang telah dilakukan, dapat disimpulkan bahwa struktur data queue merupakan antrian yang bekerja dengan aturan FIFO sehingga setiap elemen diproses secara berurutan sesuai waktu masuknya. Implementasi queue menggunakan array menunjukkan bahwa linear queue memiliki keterbatasan dalam pemanfaatan ruang, khususnya saat terjadi operasi dequeue. Circular queue terbukti lebih efisien karena dapat memanfaatkan seluruh kapasitas array melalui pergerakan indeks secara melingkar menggunakan operasi modulo. Program yang dibuat berhasil mengimplementasikan seluruh operasi dasar queue—enqueue, dequeue, pengecekan kondisi, serta pencetakan isi antrian—dan menghasilkan output yang sesuai dengan teori. Dengan demikian, praktikum ini memberikan pemahaman yang lebih mendalam mengenai cara kerja queue, perbedaannya dengan struktur data lainnya, serta pentingnya memilih metode implementasi yang efisien.

D. Referensi

- https://www.w3schools.com/cpp/default.asp
- "C++ ABI Summary". 20 March 2001. Archived from the original on 10 July 2018. Retrieved 30 May 2006.
- "Bjarne Stroustrup's FAQ – Is C a subset of C++?". Archived from the original on 6 February 2016. Retrieved 5 May 2014.