# LAPORAN PRAKTIKUM
# STRUKTUR DATA

## MODUL VIII
## PENGENALAN CODE BLOCKS


**Disusun Oleh :**
NAMA : Sinta Sintiani
NIM : 103112430047

**Dosen**
FAHRUDIN MUKTI WIBOWO

**PROGRAM STUDI STRUKTUR DATA**
**FAKULTAS INFORMATIKA**
**TELKOM UNIVERSITY PURWOKERTO**

**2025**

A. Dasar Teori

Queue (antrian) adalah struktur data linear yang menerapkan prinsip FIFO (First In, First Out), di mana elemen yang pertama masuk akan menjadi elemen yang pertama diproses. Queue memiliki dua operasi utama yaitu **enqueue** untuk menambahkan elemen ke bagian belakang antrian, dan **dequeue** untuk menghapus elemen dari bagian depan antrian, serta operasi pendukung seperti pengecekan antrian kosong atau penuh. Struktur data ini dapat diimplementasikan menggunakan array maupun linked list, dan memiliki beberapa variasi seperti linear queue, circular queue, priority queue, dan deque. Queue banyak digunakan dalam berbagai aplikasi seperti pengelolaan proses pada sistem operasi, pengaturan aliran data (buffer), serta simulasi layanan publik karena mampu mengelola proses secara teratur dan terstruktur sesuai urutan kedatangan elemen.

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

Guided 1

queue.h

```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
```

queue.cpp

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

```cpp
#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;
}
```

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

    return 0;
}
```

Screenshots Output

```
PS D:\SEM 3\STRUKTUR DATA\mod 8> cd "d:\SEM 3\STRUKTUR DATA\mod 8\" ; if ($?) { g++ m
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 ]
```

Deskripsi

Program di atas merupakan implementasi struktur data Queue dengan metode circular buffer menggunakan array berukuran maksimal lima elemen. Queue memiliki operasi dasar seperti enqueue untuk menambahkan data dan dequeue untuk menghapus data dari antrian, serta fungsi pengecekan kondisi penuh dan kosong. Program diawali dengan inisialisasi queue, kemudian dilakukan proses penambahan dan penghapusan elemen secara bertahap sambil menampilkan isi antrian setelah setiap operasi, sehingga memperlihatkan cara kerja queue secara FIFO (First In, First Out) secara sederhana.

Unguided 1

queue.h

```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;
    for (int i = 0; i < MAX; i++){
        Q.info[i] = 0;
    }
}

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;
    }

    infotype x = Q.info[Q.head];
```

```
    for (int i = 0; i < Q.tail; i++){
        Q.info[i] = Q.info[i + 1];
    }

    Q.tail--;

    if (Q.tail < 0){
        Q.head = -1;
    }

    return x;
}

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

    if (isEmptyQueue(Q)){
        cout << "empty queue";
    }
    else {
        for (int i = 0; i < MAX; i++){
            cout << Q.info[i] << " ";
        }
    }
    cout << endl;
}
```

Main.cpp

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

Screenshots Output

```
PS D:\SEM 3\STRUKTUR DATA\Modul 8> cd "d:\SEM 3\STRUKTUR DATA\Modul 8\" ;
Hello world!
--------------------
H - T    | Queue Info
--------------------
-1 - -1 | empty queue
0 - 0   | 5 0 0 0 0
0 - 1   | 5 2 0 0 0
0 - 2   | 5 2 7 0 0
0 - 1   | 2 7 7 0 0
0 - 2   | 2 7 4 0 0
0 - 1   | 7 4 4 0 0
0 - 0   | 4 4 4 0 0
```

Deskripsi:

Program di atas merupakan implementasi struktur data Queue menggunakan array statis dengan batas maksimal lima elemen tanpa sistem perputaran (non-circular). Queue mendukung operasi dasar seperti enqueue untuk menambahkan data di bagian belakang, dequeue untuk menghapus data dari bagian depan, serta pengecekan kondisi penuh dan kosong. Pada operasi dequeue, elemen-elemen dalam array digeser ke kiri agar posisi head tetap berada di indeks depan. Program utama mendemonstrasikan penggunaan queue dengan menambahkan dan menghapus beberapa elemen sambil menampilkan perubahan posisi head, tail, dan isi queue setelah setiap operasi, sehingga memperlihatkan cara kerja antrian dengan prinsip FIFO (First In, First Out).

Unguided 2

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;
    for (int i = 0; i < MAX; i++){
        Q.info[i] = 0;
    }
}

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 -999;
    }

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

    Q.head++;

    if (Q.head > Q.tail){
        Q.head = -1;
        Q.tail = -1;
    }

    return x;
}

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

    if (isEmptyQueue(Q)){
        cout << "empty queue";
    }
    else {
        for (int i = 0; i < MAX; i++){
            cout << Q.info[i] << " ";
        }
    }
    cout << endl;
}
```

main.cpp

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

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

```
PS D:\SEM 3\STRUKTUR DATA\Modul 8> cd "d:\SEM 3\STRUKTUR DATA\Modul 8\"
0 - 0   | 5 0 0 0 0
0 - 1   | 5 2 0 0 0
0 - 2   | 5 2 7 0 0
1 - 2   | 5 2 7 0 0
1 - 3   | 5 2 7 4 0
2 - 3   | 5 2 7 4 0
3 - 3   | 5 2 7 4 0
```

Deskripsi

Program di atas merupakan implementasi struktur data Queue menggunakan array statis berukuran lima elemen dengan pendekatan linear (tanpa pergeseran dan tanpa circular). Queue menyediakan operasi dasar seperti enqueue untuk menambahkan data ke posisi belakang antrian, dequeue untuk menghapus elemen pada posisi depan dengan cara memajukan indeks head, serta dua fungsi pengecekan untuk mengetahui apakah queue dalam kondisi penuh atau kosong. Jika setelah operasi dequeue posisi head melewati tail, maka queue akan direset kembali ke kondisi awal (head = -1 dan tail = -1). Program utama kemudian mendemonstrasikan proses penambahan dan penghapusan elemen sambil menampilkan posisi head, tail, dan isi array sehingga menunjukkan cara kerja queue dengan prinsip FIFO (First In, First Out) secara sederhana.

Unguided 3

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;
        return;
    }

    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 x = Q.info[Q.head];

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

    return x;
}

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

main.cpp

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

```cpp
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;
}
```

Screenshots Output

```
PS D:\SEM 3\STRUKTUR DATA\Modul 8> cd "d:\SEM 3\STRUKTUR DATA\Modul 8\" ;
Hello world!
----------------------------
H - T    | Queue Info
----------------------------
-1 - -1 |       empty queue
0 - 0   |       5
0 - 1   |       5 2
0 - 2   |       5 2 7
1 - 2   |       2 7
1 - 3   |       2 7 4
2 - 3   |       7 4
3 - 3   |       4
```

Deskripsi:

Program di atas merupakan implementasi Queue menggunakan array statis dengan metode circular, sehingga posisi head dan tail dapat berputar kembali ke indeks awal saat mencapai batas ukuran maksimum. Struktur data ini mendukung operasi dasar seperti enqueue untuk menambahkan elemen dan dequeue untuk menghapus elemen berdasarkan prinsip FIFO. Pada kondisi penuh, indeks tail + 1 akan bertemu dengan head, sementara ketika kosong nilai head dan tail diset ke -1. Program utama melakukan serangkaian operasi penambahan dan penghapusan elemen sambil menampilkan posisi head, tail, serta isi queue setelah setiap perubahan, sehingga memperlihatkan cara kerja antrian circular

secara ringkas dan jelas

B. Dari praktikum yang telah dilakukan dapat disimpulkan bahwa struktur data queue bekerja dengan prinsip FIFO (First In First Out). Implementasi dengan linked list memberikan fleksibilitas tanpa batasan ukuran, sedangkan linear queue berbasis array mudah dibuat namun dapat membuang ruang kosong setelah proses dequeue. Circular queue menjadi solusi terbaik karena memanfaatkan seluruh elemen array secara efisien melalui pergerakan indeks melingkar. Praktikum ini membantu memahami perbedaan ketiga metode serta penerapan operasi dasar queue secara nyata dalam program.

C. Referensi

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

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

https://www.programiz.com/cpp-programming/queue