

**LAPORAN PRAKTIKUM
STRUKTUR DATA**

**MODUL 8
QUEUE**



Disusun Oleh :

NAMA : AGUNG RAMADHAN

NIM : 103112430060

Dosen:

FAHRUDIN MUKTI WIBOWO

**PROGRAM STUDI STRUKTUR DATA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2025**

A. Dasar Teori

Queue adalah salah satu struktur data linear dalam informatika yang menerapkan prinsip First In First Out (FIFO), yaitu elemen yang pertama kali masuk akan menjadi elemen pertama yang keluar. Struktur data ini memiliki dua operasi utama, yaitu *enqueue* untuk menambahkan data ke bagian belakang (rear) dan *dequeue* untuk menghapus data dari bagian depan (front). Queue banyak digunakan dalam berbagai sistem komputer, seperti penjadwalan proses pada sistem operasi, antrian printer, serta manajemen permintaan pada server, karena mampu merepresentasikan proses antrian secara teratur dan efisien.

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

Guided 1

Kode 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 // Menentukan ukuran maksimal antrean

// Mendefinisikan struktur (tipe data) untuk Queue
struct Queue
{
    int info[MAX_QUEUE]; // Array untuk menyimpan data antrean
    int head;             // Penanda untuk elemen paling depan (kepala)
    int tail;             // Penanda untuk elemen paling belakang (ekor)
    int count;            // Penghitung jumlah elemen saat ini
};

// Prosedur untuk membuat queue kosong
void createQueue(Queue &Q);

// Fungsi untuk mengecek apakah queue kosong
bool isEmpty(Queue Q);

// Fungsi untuk mengecek apakah queue penuh
bool isFull(Queue Q);

// Prosedur untuk menambahkan elemen ke queue (enqueue)
void enqueue(Queue &Q, int x);

// Fungsi untuk menghapus dan mengembalikan elemen dari queue (dequeue)
int dequeue(Queue &Q);

// Prosedur untuk menampilkan semua isi queue
void printInfo(Queue Q);

#endif
```

Kode queue.cpp

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

using namespace std; // Menggunakan namespace standar agar tidak perlu
menulis std::

// Definisi prosedur untuk membuat queue kosong
void createQueue(Queue &Q)
{
    Q.head = 0; // Set kepala ke indeks 0
    Q.tail = 0; // Set ekor ke indeks 0
    Q.count = 0; // Set jumlah elemen ke 0
}

// Definisi fungsi untuk mengecek apakah queue kosong
bool isEmpty(Queue Q)
{
    return Q.count == 0; // Kembalikan true jika jumlah elemen adalah 0
}

// Definisi fungsi untuk mengecek apakah queue penuh
bool isFull(Queue Q)
{
    return Q.count == MAX_QUEUE; // Kembalikan true jika jumlah elemen sama
    dengan maks
}

// Definisi prosedur untuk menambahkan elemen (enqueue)
void enqueue(Queue &Q, int x)
{
    if (!isFull(Q))
    {
        // Jika queue tidak penuh
        Q.info[Q.tail] = x; // Masukkan data (x) ke posisi ekor (tail)
        // Pindahkan ekor secara circular (memutar)
        Q.tail = (Q.tail + 1) % MAX_QUEUE;
        Q.count++; // Tambah jumlah elemen
    }
    else
    {
        // Jika queue penuh
        cout << "Antrean Penuh!" << endl; // Tampilkan pesan error
    }
}

// Definisi fungsi untuk menghapus elemen (dequeue)
int dequeue(Queue &Q)
{
    if (!isEmpty(Q))
    {
        // Jika queue tidak kosong
        int x = Q.info[Q.head]; // Ambil data di posisi kepala (head)
```

```

        // Pindahkan kepala secara circular (memutar)
        Q.head = (Q.head + 1) % MAX_QUEUE;
        Q.count--; // Kurangi jumlah elemen
        return x;  // Kembalikan data yang diambil
    }
    else
    {
        // Jika queue kosong
        cout << "Antrean Kosong!" << endl; // Tampilkan pesan error
        return -1;                          // Kembalikan nilai -1 sebagai
tanda error
    }
}

// Definisi prosedur untuk menampilkan isi queue
void printInfo(Queue Q)
{
    cout << "Isi Queue: [ "; // Tampilkan awalan
    if (!isEmpty(Q))
    {
        // jika tidak kosong
        int i = Q.head; // Mulai dari kepala
        int n = 0;      // Penghitung elemen yang sudah dicetak
        while (n < Q.count)
        {
            // Ulangi sebanyak jumlah elemen
            cout << Q.info[i] << " "; // Cetak info
            i = (i + 1) % MAX_QUEUE; // Geser 'i' secara circular
            n++;                     // Tambah penghitung
        }
    }
    cout << "]" << endl; // Tampilkan akhiran
}

```

Kode main.cpp

```

#include <iostream> // Menyertakan library untuk input/output
#include "queue.h"  // Menyertakan file header queue kita

using namespace std; // Menggunakan namespace standar

// Fungsi utama program
int main()
{
    Queue Q; // Deklarasikan variabel queue bernama Q

    createQueue(Q); // Panggil prosedur untuk inisialisasi queue
    printInfo(Q);   // Tampilkan isi queue (seharusnya kosong)

    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;
    // Hapus 1 elemen dan tampilkan nilainya
    cout << "Elemen keluar: " << dequeue(Q) << endl;
    printInfo(Q); // Tampilkan isi queue setelah dequeue

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

    cout << "\nDequeue 2 Elemen" << endl;
    // Hapus 1 elemen dan tampilkan nilainya
    cout << "Elemen keluar: " << dequeue(Q) << endl;
    // Hapus 1 elemen lagi dan tampilkan nilainya
    cout << "Elemen keluar: " << dequeue(Q) << endl;
    printInfo(Q); // Tampilkan isi queue

    return 0;
}

```

Screenshot:

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

agungramadhan@MacBook-Air-Agung guided % ./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 ]
agungramadhan@MacBook-Air-Agung guided %

```

Deskripsi:

Kode tersebut mengimplementasikan struktur data queue (antrean) berbasis array dengan konsep circular queue menggunakan bahasa C++. Program dibagi menjadi tiga bagian utama, yaitu header file untuk mendefinisikan struktur Queue dan prototipe fungsi, file implementasi untuk mendefinisikan operasi queue seperti enqueue, dequeue, pengecekan kosong dan penuh, serta program utama (main) untuk menguji fungsionalitas queue. Queue memiliki ukuran tetap (MAX_QUEUE) dan menggunakan penanda head, tail, serta count untuk mengelola elemen secara efisien. Program ini mendemonstrasikan proses penambahan dan penghapusan elemen sesuai prinsip FIFO (First In First Out).

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

Unguided 1

Kode queue.h ALT 1

```
#ifndef QUEUE_H
#define QUEUE_H

#define MAX_QUEUE 5
typedef int infotype;

struct Queue
{
    infotype data[MAX_QUEUE];
    int head;
    int tail;
};

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

#endif
```

Kode queue.cpp ALT 1

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

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

// Cek queue kosong
bool isEmpty(Queue Q)
{
    return (Q.head == -1 && Q.tail == -1);
}

// Cek queue penuh
bool isFull(Queue Q)
{
    return (Q.tail == MAX_QUEUE - 1);
}
```

```

// Enqueue (tambah data)
void enqueue(Queue &Q, infotype x)
{
    if (isFull(Q))
    {
        cout << "Queue penuh!" << endl;
        return;
    }

    if (isEmpty(Q))
    {
        Q.head = Q.tail = 0;
    }
    else
    {
        Q.tail++;
    }

    Q.data[Q.tail] = x;
}

// Dequeue (hapus data)
infotype dequeue(Queue &Q)
{
    if (isEmpty(Q))
    {
        cout << "Queue kosong!" << endl;
        return -1;
    }

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

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

    return x;
}

// Tampilkan isi queue
void printInfo(Queue Q)
{
    cout << Q.head << "    " << Q.tail << "    ";

    if (isEmpty(Q))

```

```

{
    cout << "empty queue" << endl;
    return;
}

for (int i = Q.head; i <= Q.tail; i++)
{
    cout << Q.data[i] << " ";
}
cout << endl;
}

```

Kode main.cpp ALT 1

```

#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;
}

```

Screenshots Output

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

● agungramadhan@Mac UNGUIDED % cd ./ALT\ 1/
● agungramadhan@Mac ALT 1 % g++ main.cpp queue.cpp -o main
● agungramadhan@Mac ALT 1 % ./main
-----
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
○ agungramadhan@Mac ALT 1 %
```

Deskripsi:

Kode ALT 1 ini merupakan implementasi struktur data queue (antrean) sederhana berbasis array statis menggunakan bahasa C++. Program terdiri dari tiga file, yaitu `queue.h` untuk mendefinisikan struktur `Queue` dan prototipe fungsi, `queue.cpp` untuk mengimplementasikan operasi dasar queue seperti inisialisasi, pengecekan kosong dan penuh, `enqueue`, `dequeue`, serta penampilan isi queue, dan `main.cpp` sebagai program utama untuk menguji fungsinya. Queue dikelola menggunakan dua indeks, yaitu head dan tail, dengan nilai -1 menandakan kondisi kosong, serta menerapkan prinsip FIFO (First In First Out) dalam proses penambahan dan penghapusan data.

Kode queue.h ALT 2

```
#ifndef QUEUE_H
#define QUEUE_H

#define MAX_QUEUE 5
typedef int infotype;

struct Queue
{
    infotype data[MAX_QUEUE];
    int head;
    int tail;
};

void createQueue(Queue &Q);
bool isEmpty(Queue Q);
```

```
bool isFull(Queue Q);
void enqueue(Queue &Q, infotype x);
infotype dequeue(Queue &Q);
void printInfo(Queue Q);

#endif
```

Kode queue.cpp ALT 2

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

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

// Cek queue kosong
bool isEmpty(Queue Q)
{
    return (Q.head == -1 && Q.tail == -1);
}

// Cek queue penuh
bool isFull(Queue Q)
{
    return (Q.tail == MAX_QUEUE - 1);
}

// Enqueue (tambah data)
void enqueue(Queue &Q, infotype x)
{
    if (isFull(Q))
    {
        cout << "Queue penuh!" << endl;
        return;
    }

    if (isEmpty(Q))
    {
        Q.head = Q.tail = 0;
    }
    else
    {
        Q.tail++;
    }

    Q.data[Q.tail] = x;
```

```

}

// Dequeue (hapus data)
infotype dequeue(Queue &Q)
{
    if (isEmpty(Q))
    {
        cout << "Queue kosong!" << endl;
        return -1;
    }

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

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

    return x;
}

// Tampilkan isi queue
void printInfo(Queue Q)
{
    cout << Q.head << "    " << Q.tail << "    ";

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

    for (int i = Q.head; i <= Q.tail; i++)
    {
        cout << Q.data[i] << " ";
    }
    cout << endl;
}

```

Kode main.cpp ALT 2

```

#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;
}

```

Screenshots Output

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

● agungramadhan@Mac ALT 2 % ./main
-----
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
○ agungramadhan@Mac ALT 2 % █

```

Deskripsi:

Kode ALT 2 ini mengimplementasikan struktur data queue (antrean) sederhana menggunakan array statis dalam bahasa C++. Program dibagi menjadi tiga file, yaitu queue.h sebagai header yang mendefinisikan struktur Queue dan prototipe fungsi, queue.cpp yang berisi implementasi operasi dasar seperti inisialisasi, pengecekan kondisi kosong dan penuh, *enqueue*, *dequeue*, serta penampilan isi queue, dan main.cpp sebagai program pengujian. Queue dikelola menggunakan dua indeks, head dan tail, dengan nilai -1 menandakan kondisi kosong, serta menerapkan prinsip FIFO (First In First Out) dalam pengelolaan data antrean.

Kode queue.h ALT 3

```
#ifndef QUEUE_H
#define QUEUE_H

#define MAX_QUEUE 5
typedef int infotype;

struct Queue
{
    infotype data[MAX_QUEUE];
    int head;
    int tail;
};

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

#endif
```

Kode queue.cpp ALT 3

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

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

// Cek queue kosong
```

```

bool isEmpty(Queue Q)
{
    return (Q.head == -1 && Q.tail == -1);
}

// Cek queue penuh
bool isFull(Queue Q)
{
    return (Q.tail == MAX_QUEUE - 1);
}

// Enqueue (tambah data)
void enqueue(Queue &Q, infotype x)
{
    if (isFull(Q))
    {
        cout << "Queue penuh!" << endl;
        return;
    }

    if (isEmpty(Q))
    {
        Q.head = Q.tail = 0;
    }
    else
    {
        Q.tail++;
    }

    Q.data[Q.tail] = x;
}

// Dequeue (hapus data)
infotype dequeue(Queue &Q)
{
    if (isEmpty(Q))
    {
        cout << "Queue kosong!" << endl;
        return -1;
    }

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

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

```

```

        return x;
    }

    // Tampilkan isi queue
    void printInfo(Queue Q)
    {
        cout << Q.head << "    " << Q.tail << "    ";

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

        for (int i = Q.head; i <= Q.tail; i++)
        {
            cout << Q.data[i] << " ";
        }
        cout << endl;
    }
}

```

Kode main.cpp ALT 3

```

#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;
}

```

Screenshots Output

```

● agungramadhan@Mac ALT 3 % ./main
-----
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
○ agungramadhan@Mac ALT 3 % █

```

Deskripsi

Kode ALT 3 ini merupakan implementasi struktur data queue (antrean) berbasis array statis menggunakan bahasa C++. Program disusun dalam tiga file, yaitu `queue.h` untuk mendefinisikan struktur Queue beserta prototipe fungsi, `queue.cpp` yang berisi implementasi operasi dasar queue seperti inisialisasi, pengecekan kondisi kosong dan penuh, proses *enqueue* dan *dequeue*, serta fungsi untuk menampilkan isi antrean, dan `main.cpp` sebagai program utama untuk menguji jalannya operasi tersebut. Pengelolaan antrean menggunakan indeks head dan tail dengan nilai -1 sebagai penanda antrean kosong, serta menerapkan prinsip FIFO (First In First Out) dalam pengolahan data.

D. Kesimpulan

Kesimpulan:^[1] Berdasarkan kode queue yang telah diberikan, dapat disimpulkan bahwa program tersebut berhasil mengimplementasikan struktur data queue (antrean) berbasis array statis dengan menerapkan prinsip FIFO (First In First Out). Penggunaan variabel head dan tail mempermudah pengelolaan posisi data masuk dan keluar, sementara pembagian program ke dalam file header, implementasi, dan program utama membuat kode lebih terstruktur dan mudah dipahami. Secara keseluruhan, kode ini menunjukkan cara kerja

dasar queue dalam pemrograman C++ serta dapat digunakan sebagai dasar pembelajaran konsep struktur data antrean.

E. Referensi

Sanggalie, D., Hidayatulloh, Q. H., Abshor, W. U., Darmadewi, P. N., & Suroni, A. (2025). *Performance Analysis of Queue Structure in CPU Scheduling Simulation: A FIFO Case Study Using Python*. Journal of Advanced Systems Intelligence and Cybersecurity.

Azura Trijayanti, I. Aulia, N. Khairunisa, F. A. Hamadi Purba, & I. Gunawan. (2025). *Implementasi Struktur Data Antrian Queue dalam Sistem Penjadwalan Proses pada Sistem Operasi*. Jurnal Publikasi Teknik Informatika.

Haikal, A. A., Indra, Z., Arfansyah, A., Al Abror, A. K., & Bastio, A. (2025). *Implementasi Algoritma First In First Out (FIFO) Berbasis Web untuk Perhitungan dan Simulasi Berbagai Kasus Antrean*. Jurnal Publikasi Ilmu Komputer dan Multimedia.

Alnatsheh, E. F. (2020). *Queue Scheduling Algorithms: A Comparative Analysis*. International Journal of Innovative Computing.

Explain the concept of a queue as a “first in, first out” (FIFO) data structure. Computer Science Wiki.