LAPORAN PRAKTIKUM
STRUKTUR DATA

MODUL 8
QUEUE

Disusun Oleh :
NAMA : Muhammad Luthfi Arrafi Ramadhani
NIM : 103112430043

Dosen Pengampu:
Fahrudin Mukti Wibowo
PROGRAM STUDI S1 TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2025

A. Dasar Teori

Dasar teori pada modul ini membahas Queue, yaitu struktur data yang bekerja dengan prinsip FIFO (First In First Out), artinya data yang masuk lebih dulu akan keluar lebih dulu. Queue menggunakan dua penunjuk, head untuk elemen depan dan tail untuk elemen belakang, dengan dua operasi utama yaitu enqueue untuk menambah data di belakang dan dequeue untuk mengambil data dari depan. Queue bisa dibuat dengan linked list maupun array, dan setiap metode punya cara kerja berbeda, termasuk versi circular yang membuat antrian bisa berputar tanpa perlu menggeser data. Struktur ini banyak dipakai pada sistem antrean, pengelolaan tugas, dan proses yang membutuhkan urutan yang teratur.

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

Guided 1

(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
```

Deskripsi:

File queue.h ini berisi deklarasi untuk membuat dan memakai struktur antrean (queue) dengan array. Di dalamnya ada batas ukuran *MAX_QUEUE*, lalu struktur Queue yang punya tiga bagian penting, *head* sebagai posisi depan, *tail* sebagai posisi belakang, dan *count* untuk menghitung berapa banyak data di dalam antrean. File ini juga mendefinisikan fungsi-fungsi dasar seperti membuat antrean kosong, mengecek apakah antrean penuh atau kosong, menambah data (enqueue), menghapus data dari depan (dequeue), dan menampilkan isi antrean.

(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)) { // ika 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
}
```

Deskripsi:

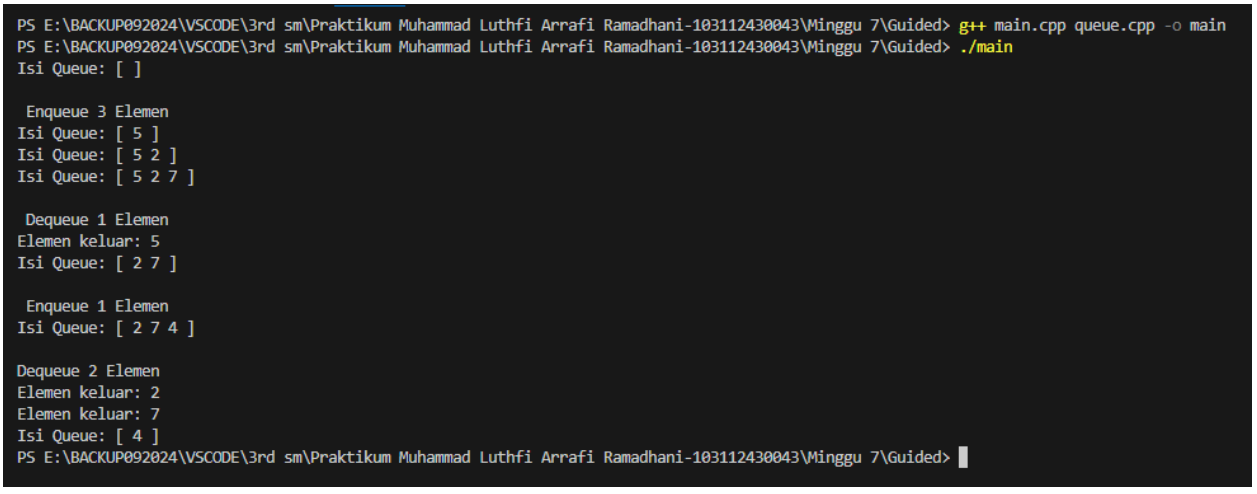
File queue.cpp ini berisi implementasi dari seluruh fungsi yang dideklarasikan di queue.h. Fungsi *createQueue()* digunakan untuk membuat antrean awal yang masih kosong dengan mengatur posisi *head, tail,* dan *count* ke nilai awal. Fungsi *isEmpty()* dan *isFull()* dipakai untuk mengecek apakah antrean sedang kosong atau sudah penuh. Penambahan data dilakukan oleh *enqueue(),* yaitu memasukkan elemen ke posisi tail lalu menggeser tail secara melingkar sambil menambah jumlah elemen. Sebaliknya, fungsi *dequeue()* mengambil data dari posisi *head*, memindahkan head secara circular, dan mengurangi jumlah elemen. Terakhir, *printInfo()* untuk menampilkan seluruh isi antrean.

(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;
}
```

Screenshots Output



Deskripsi:

File main.cpp ini berfungsi sebagai program utama untuk menjalankan semua operasi queue. Program dimulai dengan membuat antrean kosong, lalu menampilkan isinya. Setelah itu, beberapa data dimasukkan menggunakan *enqueue(),* kemudian satu per satu dikeluarkan memakai *dequeue().* Setiap perubahan selalu ditampilkan lewat *printInfo()* supaya terlihat kondisi antreannya dari awal sampai akhir.

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

Unguided 1

(queue.h)

```
// Muhammad Luthfi Arrafi Ramadhani
// 103112430043
// IF 12-06

#ifndef QUEUE_H
#define QUEUE_H

typedef int infotype;

struct Queue {
    infotype info[5];
    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
```

Deskripsi:

File queue.h ini berisi deklarasi untuk membuat struktur antrean pakai array. Di dalamnya ada tipe data Queue yang menyimpan data lewat array *info*, dan dua penanda yaitu *head* untuk posisi depan dan *tail* untuk posisi belakang. Di file ini juga terdapat fungsi-fungsi seperti *createQueue()* untuk membuat antrean kosong, *isEmptyQueue()* dan *isFullQueue()* untuk mengecek kondisi antrean, *enqueue()* utnuk menambah data, *dequeue()* untuk mengambil data dari depan, dan *printInfo()* untuk menunjukan isi antrean.

(queue.cpp)

```
// Muhammad Luthfi Arrafi Ramadhani
// 103112430043
// IF 12-06

#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 && Q.tail == -1);
}

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

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))
        return -999;

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

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

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

Deskripsi:

File queue.cpp ini berisi implementasi dari fungsi-fungsi yang telah dideklarasikan di queue.h. Di dalamnya, *createQueue()* digunakan untuk membuat antrean kosong, *isEmptyQueue()* dan *isFullQueue()* untuk mengecek apakah antrean kosong atau penuh, lalu *enqueue()* untuk memasukkan data ke bagian belakang. Fungsi *dequeue()* digunakan untuk mengambil data dari bagian depan, sekaligus mengatur ulang posisi elemen lain supaya antreannya tetap rapi. Terakhir, *printInfo()* digunakan untuk menunjukkan isi antrean dari depan sampai belakang.

(main.cpp)

```
// Muhammad Luthfi Arrafi Ramadhani
// 103112430043
// IF 12-06

#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);
    dequeue(Q);
    printInfo(Q);
    enqueue(Q, 4);
    printInfo(Q);
    dequeue(Q);
    printInfo(Q);
    dequeue(Q);
    printInfo(Q);

    return 0;
}
```

Screenshots Output

```
PS E:\BACKUP092024\VSCODE\3rd sm\Praktikum Muhammad Luthfi Arrafi Ramadhani-103112430043\Minggu 7\Unguided> g++ main.cpp queue.cpp -o main
PS E:\BACKUP092024\VSCODE\3rd sm\Praktikum Muhammad Luthfi Arrafi Ramadhani-103112430043\Minggu 7\Unguided> ./main
Hello world!
--------------------------------
 H - T   : Queue info
--------------------------------
-1 - -1  : empty queue
0 - 0    : 5
0 - 1    : 5 2
0 - 2    : 5 2 7
0 - 1    : 2 7
0 - 0    : 7
0 - 1    : 7 4
0 - 0    : 4
-1 - -1  : empty queue
PS E:\BACKUP092024\VSCODE\3rd sm\Praktikum Muhammad Luthfi Arrafi Ramadhani-103112430043\Minggu 7\Unguided>
```

Deskripsi:

File main.cpp ini digunakan untuk menjalankan program utama dari antrean. Di dalamnya, antrean dibuat kosong terlebih dahulu, lalu program memasukkan beberapa data menggunakan enqueue() dan mengeluarkan data dengan dequeue(). Setiap kali ada perubahan, printInfo() dipanggil agar isi antreannya terlihat dari awal sampai akhir.

Unguided 2

(queue.cpp)

```
// Muhammad Luthfi Arrafi Ramadhani
// 103112430043
// IF 12-06

// SOAL NOMOR 2

#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 && Q.tail == -1);
}

bool isFullQueue(Queue Q) {
   return (Q.tail == 4 && Q.head == 0);
}

void enqueue(Queue &Q, infotype x) {

   if (Q.tail == 4 && Q.head > 0) {
      int j = 0;
      for (int i = Q.head; i <= Q.tail; i++) {
         Q.info[j] = Q.info[i];
         j++;
      }
      Q.tail = j - 1;
      Q.head = 0;
   }

   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))
      return -999;

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

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

```
    } else {
        Q.head++;
    }

    return x;
}

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

```
PS E:\BACKUP092024\VSCODE\3rd sm\Praktikum Muhammad Luthfi Arrafi Ramadhani-103112430043\Minggu 7\Unguided> g++ main.cpp queue.cpp -o main
PS E:\BACKUP092024\VSCODE\3rd sm\Praktikum Muhammad Luthfi Arrafi Ramadhani-103112430043\Minggu 7\Unguided> ./main
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
2 - 2    : 7
2 - 3    : 7 4
3 - 3    : 4
-1 - -1  : empty queue
PS E:\BACKUP092024\VSCODE\3rd sm\Praktikum Muhammad Luthfi Arrafi Ramadhani-103112430043\Minggu 7\Unguided>
```

Deskripsi:

File queue.cpp ini berisi cara kerja fungsi antrean untuk mekanisme Alternatif 2, di mana *head* dan *tail* bisa bergerak. Pada bagian *enqueue*, data baru dimasukkan ke posisi *tail*, dan jika *tail* sudah mencapai batas sementara masih ada ruang kosong di depan, elemen akan digeser agar antrean bisa digunakan kembali. Fungsi *dequeue* mengambil data dari posisi head lalu memajukan head satu langkah. Selain itu, *isEmptyQueue()* dan *isFullQueue()* digunakan untuk memeriksa kondisi antrean, sedangkan *printInfo()* menampilkan isi antrean sesuai posisi head dan tail yang sedang aktif.

Unguided 3

(queue.cpp)

```
// Muhammad Luthfi Arrafi Ramadhani
// 103112430043
// IF 12-06

// SOAL NOMOR 3

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

const int MAX = 5;

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

bool isEmptyQueue(Queue Q) {
    return (Q.head == -1 && Q.tail == -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)) {
        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 : ";

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

Screenshots Output

```
PS E:\BACKUP092024\VSCODE\3rd sm\Praktikum Muhammad Luthfi Arrafi Ramadhani-103112430043\Minggu 7\Unguided> g++ main.cpp queue.cpp -o main
PS E:\BACKUP092024\VSCODE\3rd sm\Praktikum Muhammad Luthfi Arrafi Ramadhani-103112430043\Minggu 7\Unguided> ./main
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
2 - 2    : 7
2 - 3    : 7 4
3 - 3    : 4
-1 - -1  : empty queue
PS E:\BACKUP092024\VSCODE\3rd sm\Praktikum Muhammad Luthfi Arrafi Ramadhani-103112430043\Minggu 7\Unguided>
```

Deskripsi:

File queue.cpp untuk Alternatif 3 ini berisi cara kerja antrean dengan mekanisme circular, di mana posisi head dan tail bisa berputar kembali ke indeks awal saat mencapai batas array. Pada proses *enqueue*, data baru ditambahkan ke posisi tail lalu tail digeser dengan rumus modulo supaya tetap berputar. Fungsi *dequeue* mengambil elemen dari head dan memajukan head dengan cara yang sama. Pengecekan antrean kosong atau penuh dilakukan dengan melihat posisi head, tail, dan banyaknya elemen yang tersimpan. Sementara itu, *printInfo()* menampilkan isi antrean sesuai urutan dari head hingga jumlah elemen yang ada, mengikuti pola pergerakan circular pada queue ini.

D. Kesimpulan

Dari ketiga latihan tentang Queue ini bisa disimpulkan bahwa antrean bekerja dengan prinsip FIFO, yaitu data yang masuk lebih dulu akan keluar lebih dulu. Setiap alternatif punya cara kerja yang berbeda: Alternatif 1 memakai head tetap dan tail yang maju sambil menggeser elemen saat dequeue, Alternatif 2 membuat head dan tail sama-sama bergerak dan hanya menggeser elemen jika antrean penuh sebagian, sementara Alternatif 3 memakai sistem berputar (circular) sehingga head dan tail bisa kembali ke awal tanpa perlu geser elemen. Ketiga latihan ini membantu memahami berbagai cara mengatur antrean dengan array sesuai kebutuhan dan kondisi yang berbeda.

E. Referensi

Trijayanti, A., Aulia, I., Khairunisa, N., Hamadi Purba, F. A., & Gunawan, I. (2025). *Implementasi struktur data antrian queue dalam sistem penjadwalan proses pada sistem operasi*. Jurnal Publikasi Teknik Informatika, 4(2), 48–53. https://doi.org/10.55606/jupti.v4i2.4170

Gunawan, R., Yuana, H., & Kirom, S. (2023). *Implementasi metode queue pada sistem antrian online berbasis web: studi kasus Puskesmas*. JATI (Jurnal Mahasiswa Teknik Informatika), 7(3), 1538–1545. https://doi.org/10.36040/jati.v7i3.6687