

**LAPORAN PRAKTIKUM
STRUKTUR DATA**

MODUL VIII

QUEUE



Disusun Oleh :

Muhammad Fathammubina

NIM : 103112430188

Dosen

FAHRUDIN MUKTI WIBOWO

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

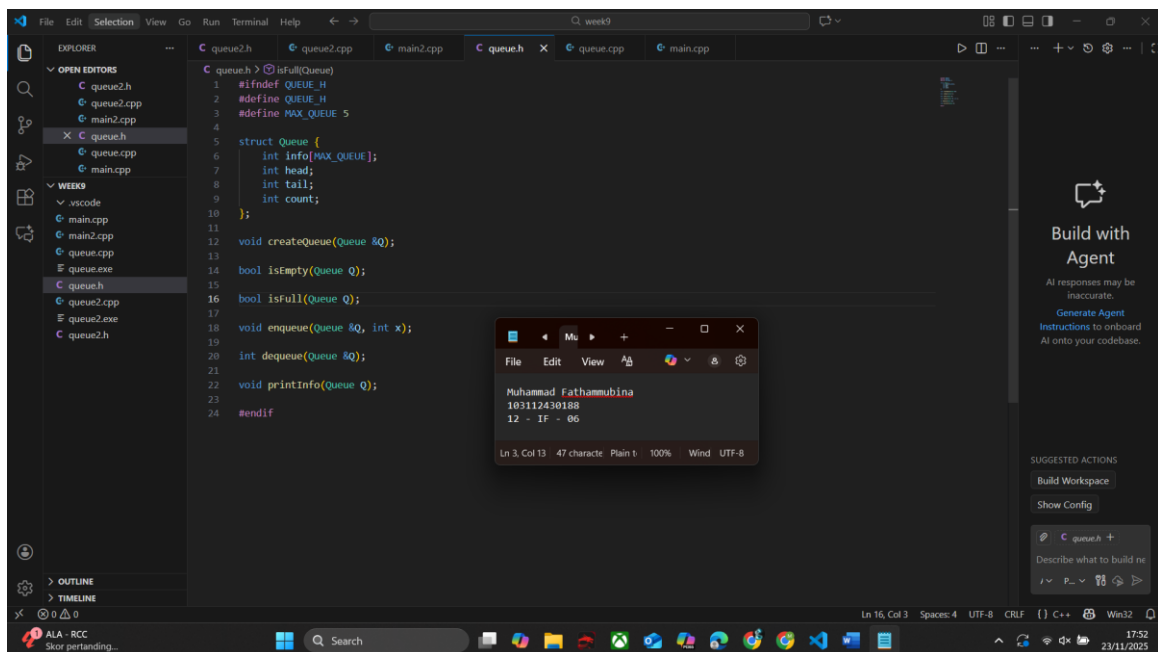
A. Dasar Teori

Queue adalah struktur data linear yang menerapkan prinsip FIFO (First In First Out), yaitu elemen yang pertama dimasukkan ke dalam antrian akan menjadi elemen yang pertama dikeluarkan. Konsep ini menyerupai antrean di dunia nyata, seperti antrean loket tiket, di mana orang yang datang lebih dahulu akan dilayani lebih dulu. Dalam representasi array, queue memiliki dua indeks penting, yaitu head sebagai penanda elemen terdepan dan tail sebagai penanda elemen terakhir. Operasi utama struktur data queue terdiri dari enqueue untuk menambahkan elemen di bagian belakang dan dequeue untuk menghapus elemen di bagian depan. Berdasarkan modul, terdapat tiga mekanisme queue menggunakan array, yaitu Alternatif 1 (head diam, tail bergerak), Alternatif 2 (head dan tail bergerak), dan Alternatif 3 atau Circular Queue (head dan tail berputar menggunakan operasi modulo). Circular queue merupakan mekanisme yang paling efisien karena tidak membutuhkan proses shifting ketika ruang array terlihat penuh secara semu. Selain itu, queue juga memiliki operasi dasar lain seperti createQueue, isEmpty, isFull, dan printInfo, yang berfungsi mengelola dan menampilkan isi antrian. Struktur data queue banyak digunakan dalam berbagai bidang seperti manajemen proses, simulasi antrean, buffer komunikasi, dan sistem antrian pada perangkat lunak.

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

Guided 1

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
```

```
#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

```

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

// Queue struct
struct Queue {
    int head;
    int tail;
    int count;
};

// Functions
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 (!isEmpty(Q)) {
        Q.head = (Q.head + 1) % MAX_QUEUE;
        Q.count++;
    } else {
        cout << "Anton Kasing!" << endl;
    }
}

int dequeue(Queue &Q) {
    if (!isEmpty(Q)) {
        int n = Q.head;
        Q.head = (Q.head + 1) % MAX_QUEUE;
        Q.count--;
        return n;
    } else {
        cout << "Anton Kasing!" << endl;
        return -1;
    }
}

void printInfo(Queue Q) {
    cout << "Info Queue: { ";
    if (!isEmpty(Q)) {
        int n = Q.head;
        int m = Q;
        while (n < Q.count) {
            cout << Q.data[n] << " ";
            n = (n + 1) % MAX_QUEUE;
        }
    }
    cout << "}" << endl;
}

```

Build with Agent
All responses may be inaccurate.
Generate Agent instructions to embed AI into your codebase.

SUGGESTED ACTIONS
Build Workspace
Show Config

ALA - RCC
Skor pertanding...

Ln 14, Col 2 | Search | UTF-8 | CRLF | 17:53
23/11/2025

```

#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;
    }

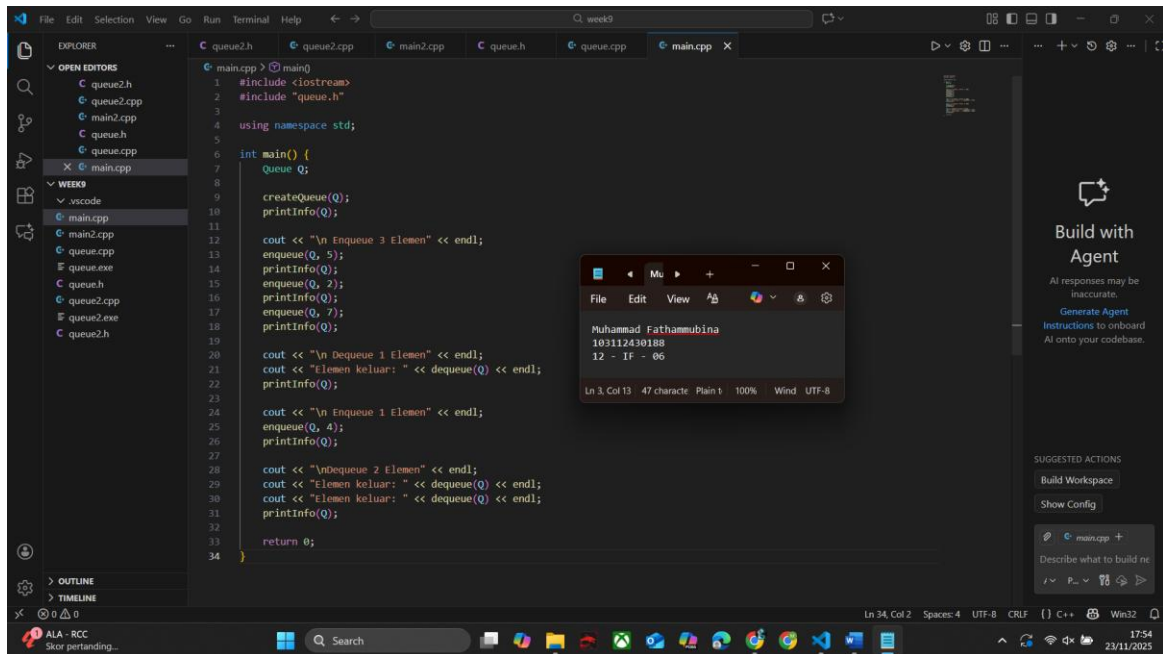
    \
    void enqueue(Queue &Q, int x) {
        if (!isFull(Q)) {
            Q.info[Q.tail] = x;
            Q.tail = (Q.tail + 1) % MAX_QUEUE;
            Q.count++;
        } else {
            cout << "Antrean Penuh!" << endl;
        }
    }

    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;
    }
}

```

main.cpp



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

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 << "\n Dequeue 2 Elemen" << endl;
```

```

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

    return 0;
}

```

Screenshots Output

```

PS C:\Users\Puan Malika\Documents\SEMESTER 3\week9> g++ main.cpp queue.cpp -o main
PS C:\Users\Puan Malika\Documents\SEMESTER 3\week9> .\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 ]
PS C:\Users\Puan Malika\Documents\SEMESTER 3\week9>

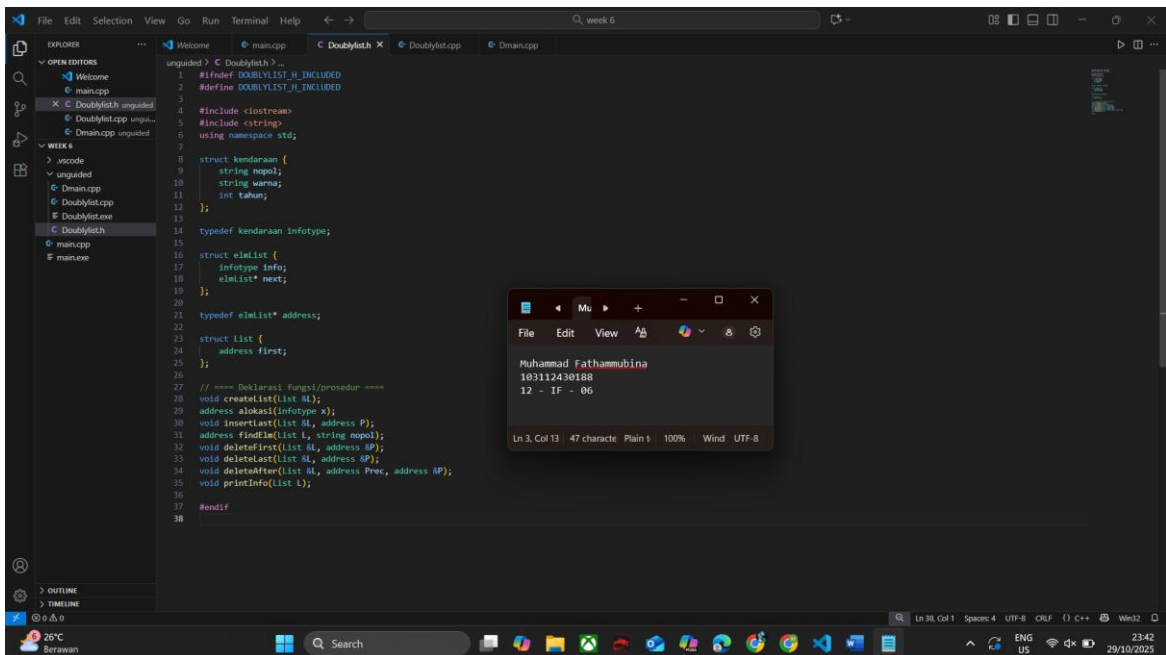
```

Deskripsi:

Program ini mengimplementasikan struktur data Circular Queue menggunakan array berukuran tetap ($MAX_QUEUE = 5$) dengan pemantauan posisi elemen melalui indeks head, tail, dan variabel count sebagai penghitung jumlah elemen saat ini. Fungsi createQueue digunakan untuk menginisialisasi queue agar berada dalam kondisi kosong, isEmpty dan isFull berfungsi untuk memeriksa apakah queue kosong atau penuh, enqueue menambahkan elemen baru ke posisi tail dan menggeser indeks secara melingkar menggunakan operasi modulo agar elemen dapat memutar kembali ke indeks awal ketika mencapai batas array, sedangkan dequeue menghapus elemen dari posisi head dengan cara yang sama. Program juga menyediakan fungsi printInfo yang menampilkan semua isi queue sesuai urutan antrean. Melalui fungsi-fungsi tersebut, program kemudian melakukan beberapa operasi enqueue dan dequeue untuk memperlihatkan bagaimana circular queue bekerja dalam mengelola data secara efisien.

Unguided 1

queue2.h



```
#ifndef QUEUE2_H
#define QUEUE2_H

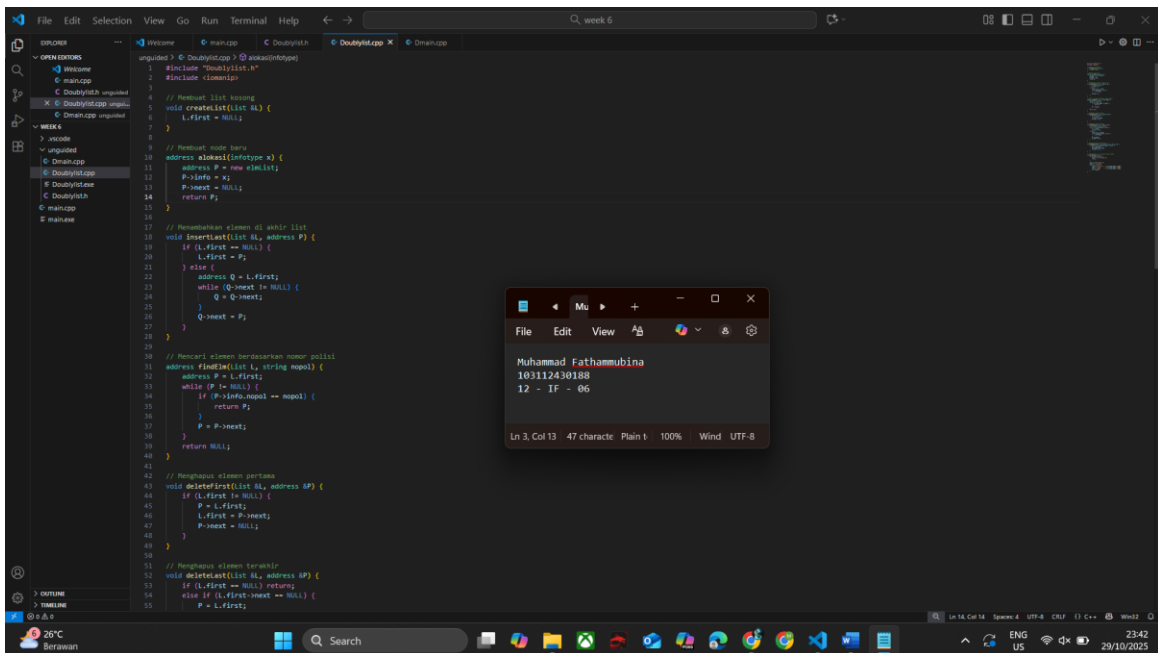
typedef int infotype;

struct Queue {
    infotype info[5];
    int head, 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
```

queue2.cpp



```
#include <iostream>
#include "queue2.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)) {
        if (Q.head > 0) {
            int j = 0;
            for (int i = Q.head; i <= Q.tail; i++) {
                Q.info[j++] = Q.info[i];
            }
            Q.tail = Q.tail - Q.head;
            Q.head = 0;
        } else {
            cout << "Queue penuh!\n";
            return;
        }
    }
}
```

```

        Q.tail++;
        Q.info[Q.tail] = x;
    }

    infotype dequeue(Queue &Q) {
        if (isEmptyQueue(Q)) {
            cout << "Queue kosong!\n";
            return -1;
        }

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

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

        return x;
    }

    void printInfo(Queue Q) {
        cout << "H=" << Q.head << " T=" << Q.tail << " | Queue info: ";

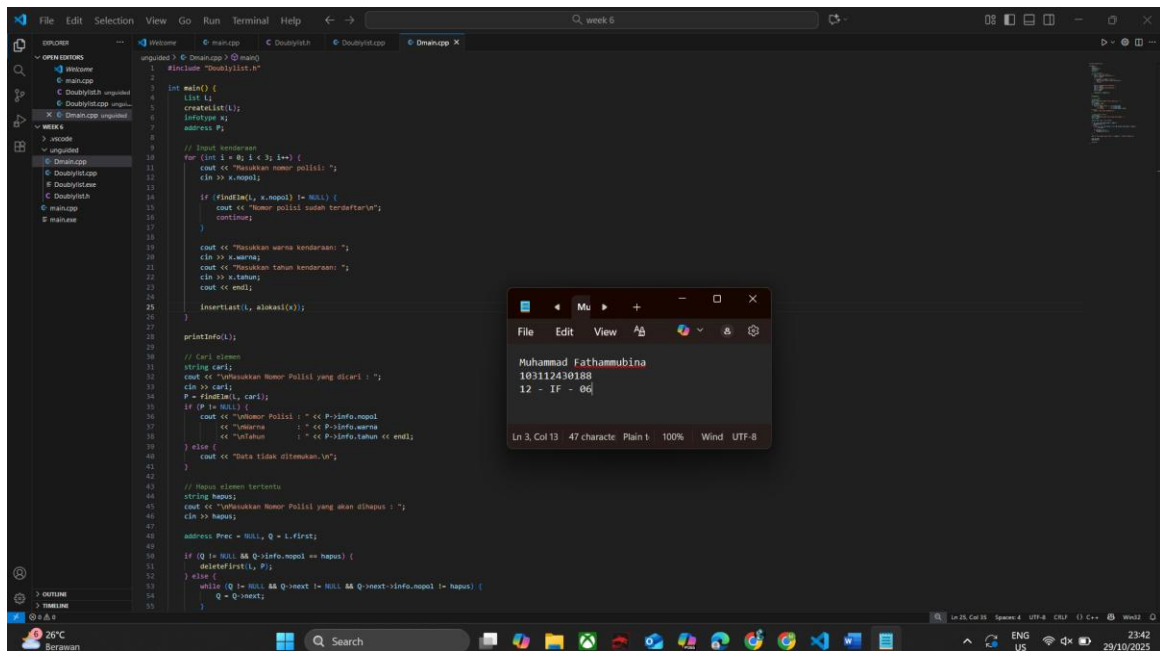
        if (isEmptyQueue(Q)) {
            cout << "empty queue\n";
            return;
        }

        for (int i = Q.head; i <= Q.tail; i++)
            cout << Q.info[i] << " ";

        cout << endl;
    }
}

```

main2.cpp



```
#include <iostream>
#include "queue2.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);

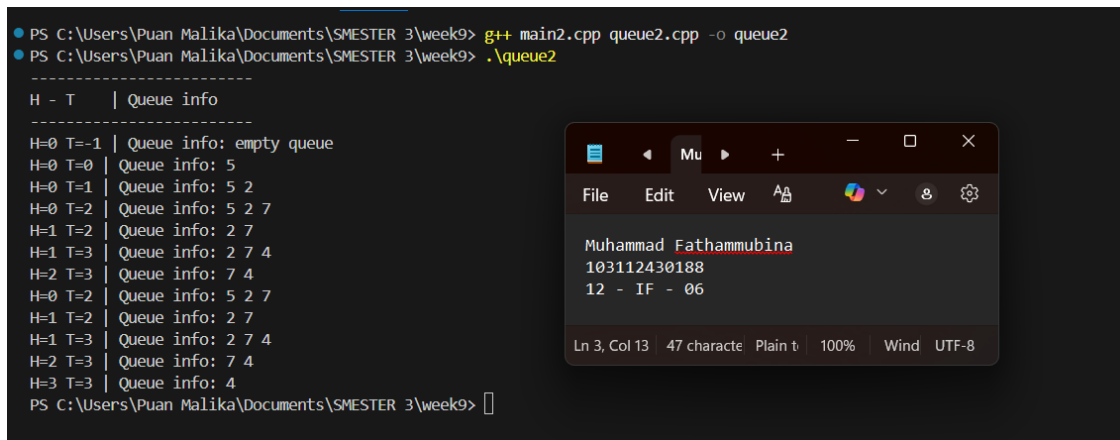
    dequeue(Q); printInfo(Q);

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

    dequeue(Q); printInfo(Q);
    dequeue(Q); printInfo(Q);

    return 0;
}
```

Screenshots Output



```
PS C:\Users\Puan_Malika\Documents\SEMESTER 3\week9> g++ main2.cpp queue2.cpp -o queue2
PS C:\Users\Puan_Malika\Documents\SEMESTER 3\week9> .\queue2

-----
H - T | Queue info
-----
H=0 T=-1 | Queue info: empty queue
H=0 T=0 | Queue info: 5
H=0 T=1 | Queue info: 5 2
H=0 T=2 | Queue info: 5 2 7
H=1 T=2 | Queue info: 2 7
H=1 T=3 | Queue info: 2 7 4
H=2 T=3 | Queue info: 7 4
H=0 T=2 | Queue info: 5 2 7
H=1 T=2 | Queue info: 2 7
H=1 T=3 | Queue info: 2 7 4
H=2 T=3 | Queue info: 7 4
H=3 T=3 | Queue info: 4
PS C:\Users\Puan_Malika\Documents\SEMESTER 3\week9>
```

Deskripsi:

Program di atas mengimplementasikan struktur data Queue menggunakan array berukuran tetap dengan mekanisme Alternatif 2, yaitu head dan tail sama-sama bergerak, dan ketika array penuh tetapi masih ada ruang di depan (karena beberapa elemen sudah di-dequeue), maka isi array akan digeser ke kiri agar ruang kosong bisa digunakan kembali. Program ini menggunakan beberapa fungsi utama, yaitu createQueue untuk menginisialisasi queue agar siap digunakan, enqueue untuk menambahkan data ke bagian belakang queue sekaligus melakukan shifting bila diperlukan, dequeue untuk menghapus elemen terdepan dan menggeser posisi head, isEmptyQueue untuk memeriksa apakah queue kosong, isFullQueue untuk memeriksa apakah array sudah penuh, dan printInfo untuk menampilkan nilai head, tail, serta seluruh isi queue. Melalui fungsi-fungsi tersebut, program melakukan serangkaian operasi enqueue dan dequeue lalu menampilkan kondisi queue setelah setiap perubahan.

C. Kesimpulan

Berdasarkan praktikum yang dilakukan, dapat disimpulkan bahwa struktur data Queue merupakan salah satu struktur data penting yang bekerja dengan prinsip FIFO. Implementasi queue menggunakan array dapat dilakukan dalam beberapa mekanisme, namun mekanisme Circular Queue memberikan efisiensi terbaik karena tidak memerlukan shifting data. Fungsi-fungsi dasar seperti enqueue, dequeue, isEmpty, isFull, dan printInfo sangat diperlukan untuk mengelola antrian dengan benar. Melalui praktikum ini, mahasiswa diharapkan memahami cara kerja queue serta mampu mengimplementasikan dan membedakan tiga bentuk representasi queue menggunakan array.

D. Referensi

Muliono, R. (2017). Abstract Data Type (ADT). Universitas Multimedia Nusantara.

Maulana, R. (2023, November 29). Struktur Data Queue: Pengertian, Fungsi, dan Jenisnya. Dicoding Blog.