LAPORAN PRAKTIKUM

STRUKTUR DATA

MODUL VIII

QUEUE

Disusun Oleh :

Nama : Izzah Minkhotun Fannisa

NIM : 103112400198

Dosen

FAHRUDIN MUKTI WIBOWO

PROGRAM STUDI STRUKTUR DATA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

A. Dasar Teori

C++ merupakan bahasa pemrograman yang pertama kali dikembangkan oleh Bjarne Stroustrop, seorang ilmuwan komputer asal Denmark. Bahasa pemrograman C++ disempurnakan dari bahasa C yang di temukan oleh Dennis Ritchie pada awal 1970-an. Lalu resmi dirilis pada tahun 1985. C++ cukup banyak digunakan karena kemampuannya dalam memberikan kecepatan eksekusi tinggi serta dukungan terhadap manajemen memori tingkat rendah.

Queue merupakan salah satu struktur data linear yang menerapkan prinsip *FIFO (First In First Out)*, di mana elemen yang pertama dimasukkan akan menjadi elemen pertama yang dikeluarkan. Konsep ini mirip dengan antrean dalam kehidupan sehari-hari, seperti antrean di kasir atau loket pelayanan. Dalam sebuah queue terdapat dua bagian penting, yaitu head yang menunjukkan posisi elemen paling depan dan tail yang menandai elemen paling belakang. Operasi utama pada queue meliputi enqueue, yaitu proses menambahkan data ke bagian belakang antrean, serta dequeue, yaitu proses menghapus data dari bagian depan. Struktur data queue dapat diimplementasikan menggunakan array maupun linked list.

B. Guide

1. Guide 1

a. Source Code

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
```

queue.cpp

```
#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) {
    cout << "Isi Antrean: [";
```

```
    if (!isEmpty(Q)) {
        int i = Q.head;
        int n = 0;
        while (n < Q.count) {
            cout << Q.info[i];
            if (n < Q.count - 1) cout << ", ";
            i = (i + 1) % MAX_QUEUE;
            n++;
        }
        cout << "]";
    } else {
        cout << "Antrean Kosong!";
    }
    cout << endl;
}
```

main.cpp

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

using namespace std;

int main() {
    Queue Q;

    createQueue(Q);
    printInfo(Q);

    cout << "\nEnqueue 3 elemen" << endl;
    enqueue(Q, 5);
    printInfo(Q);

    enqueue(Q, 2);
    printInfo(Q);

    enqueue(Q, 7);
    printInfo(Q);

    cout << "\nDequeue 1 elemen" << endl;
    cout << "Elemen keluar: " << dequeue(Q) << endl;
    printInfo(Q);

    cout << "\nEnqueue 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;
}
```

b. Screenshot Output

```
terpreter=m1'
Isi Antrean: [Antrean Kosong!

Enqueue 3 elemen
Isi Antrean: [5]
Isi Antrean: [5, 2]
Isi Antrean: [5, 2, 7]

Dequeue 1 elemen
Elemen keluar: 5
Isi Antrean: [2, 7]

Enqueue 1 elemen
Isi Antrean: [2, 7, 4]

Dequeue 2 elemen
Elemen keluar: 2
Elemen keluar: 7
Isi Antrean: [4]
```

c. Deskripsi Program

Program ini dibuat untuk menerapkan struktur data queue (antrean) berbasis array statis dengan prinsip *FIFO (First In First Out)*. Program menyediakan operasi dasar untuk menambahkan, menghapus, dan menampilkan data dalam antrean.

Struktur Queue didefinisikan dengan array sebagai tempat penyimpanan data, penunjuk head dan tail, serta variabel count untuk mengatur jumlah elemen, disertai fungsi-fungsi pendukung untuk mengelola antrean. Implementasi program mencakup pembuatan antrean, pengecekan kondisi kosong dan penuh, proses enqueue dan dequeue, serta penampilan isi antrean.

Pada bagian utama, program menjalankan beberapa operasi enqueue dan dequeue, lalu menampilkan perubahan kondisi antrean setiap kali proses dilakukan hingga program selesai dijalankan.

C. Unguide

1. Unguide 1

a. Source Code

queue.h

```h
#ifndef QUEUE_H
#define QUEUE_H

#define MAX_QUEUE 5

typedef int infotype;

struct Queue {
    infotype info[MAX_QUEUE];
    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

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

using namespace std;

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

bool isEmptyQueue(Queue Q) {
    return Q.tail == -1;
}

bool isFullQueue(Queue Q) {
    return Q.tail == MAX_QUEUE - 1;
}

void enqueue(Queue &Q, infotype x) {
    if (!isFullQueue(Q)) {
        Q.tail++;
        Q.info[Q.tail] = x;
    }
}
```

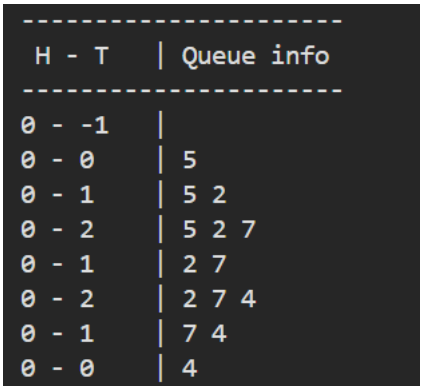
```
infotype dequeue(Queue &Q) {
    infotype x = Q.info[0];
    for (int i = 0; i < Q.tail; i++) {
        Q.info[i] = Q.info[i + 1];
    }
    Q.tail--;
    return x;
}

void printInfo(Queue Q) {
    cout << Q.head << " - " << Q.tail << "\t | ";
    for (int i = 0; i <= Q.tail; i++) {
        cout << Q.info[i] << " ";
    }
    cout << endl;
}
```

main.cpp

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

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;
}
```

b. Screenshot Output



```
----------------------
 H - T   | Queue info
----------------------
0 - -1   |
0 - 0    | 5
0 - 1    | 5 2
0 - 2    | 5 2 7
0 - 1    | 2 7
0 - 2    | 2 7 4
0 - 1    | 7 4
0 - 0    | 4
```

c. Deskripsi Program

Program ini menerapkan ADT Queue menggunakan array dengan mekanisme Alternatif 1, yaitu posisi head yang tetap dan tail yang bergerak mengikuti penambahan data. Dalam mekanisme ini, elemen antrean selalu dimulai dari indeks awal array sehingga head tidak pernah berpindah.

Struktur Queue didefinisikan menggunakan array sebagai media penyimpanan data, serta variabel head dan tail untuk menandai posisi elemen, lengkap dengan operasi dasar seperti pembuatan antrean, pengecekan kondisi kosong dan penuh, penambahan dan penghapusan data, serta penampilan isi antrean.

Setiap proses enqueue menambahkan data di bagian belakang dengan menggeser tail, sedangkan proses dequeue menghapus elemen terdepan dan menggeser seluruh elemen ke depan agar urutan tetap terjaga. Pada bagian utama, program menguji antrean dengan melakukan beberapa operasi enqueue dan dequeue, lalu menampilkan isi antrean setelah setiap proses untuk memperlihatkan perubahan data hingga program selesai dijalankan.

2. Unguide 2

a. Source Code

queue.h

```
#ifndef QUEUE_H
```

```h
#define QUEUE_H

#define MAX_QUEUE 5

typedef int infotype;

struct Queue {
    infotype info[MAX_QUEUE];
    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

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

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 == MAX_QUEUE - 1;
}

void enqueue(Queue &Q, infotype x) {
    if (!isFullQueue(Q)) {
        Q.tail++;
        Q.info[Q.tail] = x;
    }
}

infotype dequeue(Queue &Q) {
```

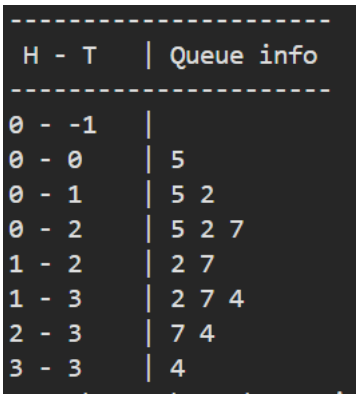
```
    return Q.info[Q.head++];
}

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

main.cpp

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

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;
}
```

b. Screenshot Output



```
---------------------
 H - T   | Queue info
---------------------
0 - -1   |
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
```

c. Deskripsi Program

Program ini menerapkan ADT Queue berbasis array dengan mekanisme Alternatif 2, di mana posisi head dan tail sama-sama bergerak mengikuti operasi yang dilakukan. Pada mekanisme ini, elemen antrean tidak perlu digeser saat data dihapus, sehingga proses dequeue menjadi lebih efisien. Struktur Queue didefinisikan menggunakan array sebagai media penyimpanan data serta variabel head dan tail sebagai penanda posisi elemen, lengkap dengan operasi dasar seperti pembuatan antrean, pengecekan kondisi kosong dan penuh, penambahan data, penghapusan data, dan penampilan isi antrean.

Operasi enqueue menambahkan data di bagian belakang dengan memajukan tail, sedangkan operasi dequeue dilakukan dengan memindahkan posisi head ke elemen berikutnya tanpa mengubah susunan data di dalam array. Pada bagian utama, program menjalankan beberapa proses enqueue dan dequeue, kemudian menampilkan isi antrean setiap kali operasi dilakukan untuk memperlihatkan perubahan posisi head, tail, dan kondisi antrean hingga program selesai dijalankan.

3. Unguide 3

a. Source Code

queue.h

```
#ifndef QUEUE_H
#define QUEUE_H

#define MAX_QUEUE 5

typedef int infotype;

struct Queue {
```

```
    infotype info[MAX_QUEUE];
    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 "queue.h"
#include <iostream>

using namespace std;

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

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

bool isFullQueue(Queue Q) {
    return (Q.tail + 1) % MAX_QUEUE == Q.head;
}

void enqueue(Queue &Q, infotype x) {
    if (!isFullQueue(Q)) {
        Q.info[Q.tail] = x;
        Q.tail = (Q.tail + 1) % MAX_QUEUE;
    }
}

infotype dequeue(Queue &Q) {
    infotype x = Q.info[Q.head];
    Q.head = (Q.head + 1) % MAX_QUEUE;
    return x;
}
```

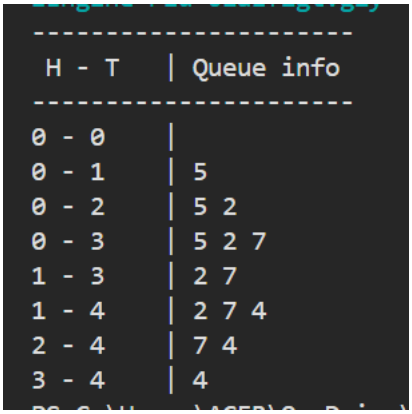
```
void printInfo(Queue Q) {
    cout << Q.head << " - " << Q.tail << "\t | ";
    int i = Q.head;
    while (i != Q.tail) {
        cout << Q.info[i] << " ";
        i = (i + 1) % MAX_QUEUE;
    }
    cout << endl;
}
```

main.cpp

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

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;
}
```

b. Screenshot Output



c. Deskripsi Program

Program ini menerapkan ADT Queue berbasis array dengan mekanisme Alternatif 3 atau circular queue, di mana posisi head dan tail bergerak secara berputar. Mekanisme ini memungkinkan pemanfaatan memori yang lebih efisien karena bagian array yang sudah dilewati dapat digunakan kembali.

Struktur Queue didefinisikan menggunakan array sebagai media penyimpanan data serta variabel head dan tail sebagai penanda posisi elemen, lengkap dengan operasi dasar seperti pembuatan antrean, pengecekan kondisi kosong dan penuh, penambahan dan penghapusan data, serta penampilan isi antrean.

Proses enqueue dan dequeue dilakukan dengan memajukan head dan tail menggunakan operasi modulo, sehingga data tidak perlu digeser saat terjadi penghapusan. Pada bagian utama, program menjalankan beberapa operasi enqueue dan dequeue, lalu menampilkan isi antrean setelah setiap proses untuk memperlihatkan perubahan posisi head, tail, dan kondisi antrean hingga program selesai dijalankan.

D. Kesimpulan

Dari penerapan serta pengujian tiga pendekatan antrean berbasis array, terlihat bahwa masing-masing metode memiliki pola kerja dan kelebihan yang berbeda. Pendekatan pertama mempertahankan posisi depan tetap, sementara penunjuk belakang berpindah mengikuti penambahan data, sehingga pengeluaran elemen harus disertai pemindahan seluruh isi, yang menyebabkan proses menjadi lebih lambat meskipun strukturnya mudah dipahami. Pendekatan kedua memungkinkan kedua penunjuk berpindah tanpa melakukan pemindahan data, sehingga proses penghapusan menjadi lebih cepat, namun kapasitas penyimpanan tidak dimanfaatkan secara maksimal karena bagian array yang telah dilewati tidak dapat digunakan kembali. Pendekatan ketiga

menerapkan sistem melingkar, di mana penunjuk depan dan belakang berputar mengikuti batas array, sehingga seluruh ruang penyimpanan dapat digunakan secara maksimal tanpa proses pemindahan data. Dengan karakteristik tersebut, pendekatan ketiga menjadi pilihan paling optimal untuk kebutuhan antrean yang menuntut kecepatan proses dan efisiensi penggunaan memori..

E. Referensi

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2022). *Introduction to algorithms (4th ed.).* MIT Press.

Goodrich, M. T., Tamassia, R., & Goldwasser, M. H. (2014). *Data structures and algorithms in Java (6th ed.).* John Wiley & Sons.

Kadir, A. (2013). Algoritma dan pemrograman menggunakan C++. Andi Publisher.

Munir, R. (2016). Algoritma dan struktur data. Informatika Bandung.

Weiss, M. A. (2012). *Data structures and algorithm analysis in C++ (4th ed.). Pearson Education.*