LAPORAN PRAKTIKUM
STRUKTUR DATA

MODUL 8
PENGENALAN BAHASA C++

Disusun Oleh :
NAMA : M.IRFAN ADIB
NIM : 103112400257

Dosen
FAHRUDIN MUKTI WIBOWO

PROGRAM STUDI STRUKTUR DATA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2025

A. Dasar Teori

Dasar teori Queue (Antrian) adalah konsep struktur data linear yang mengikuti prinsip FIFO (First In, First Out), yaitu elemen yang pertama masuk akan menjadi elemen yang pertama keluar. Queue bekerja layaknya antrian di kehidupan sehari-hari: data yang datang lebih dulu akan diproses lebih dulu. Struktur data ini memiliki dua ujung penting, yaitu head sebagai posisi pengambilan data dan tail sebagai posisi penambahan data. Operasi utama pada queue meliputi enqueue (menambahkan elemen ke bagian belakang), dequeue (menghapus elemen dari bagian depan), serta pengecekan kondisi empty dan full untuk memastikan operasi berjalan dengan aman. Pada penerapannya, queue dapat dibuat menggunakan array statis, array melingkar (circular queue), atau menggunakan linked list. Circular queue digunakan untuk memaksimalkan penggunaan ruang array dengan cara membuat indeks tail kembali ke indeks awal setelah mencapai batas array. Queue banyak digunakan dalam sistem penjadwalan proses, pengelolaan buffer, simulasi antrian, serta berbagai algoritma yang membutuhkan urutan pemrosesan yang teratur.

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

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) {
    if (isEmpty(Q)) {
        cout << "Isi Antrean: [Kosong]" << endl;
        return;
    }

    cout << "Isi Antrean: [";

    int i = Q.head;
    for (int n = 0; n < Q.count; n++) {
        cout << Q.info[i];
        if (n < Q.count - 1) cout << ", ";
        i = (i + 1) % MAX_QUEUE;
    }

    cout << "]" << endl;
}
```

main.cpp

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

using namespace std;

int main() {
    Queue Q;

    createQueue(Q);
    printInfo(Q);

    cout <<"\n enqueu 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 enqueu 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

```
Isi Antrean: [Kosong]

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

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

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

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

Deskripsi: Program ini menunjukkan cara kerja antrean, seperti antrean di loket tiket. Antrean ini dibuat kosong di awal. Kemudian, program mulai memasukkan tiga angka (5, 2, dan 7) satu per satu, di mana setiap angka ini selalu ditempatkan di bagian belakang barisan. Setelah itu, program mengeluarkan satu angka dari bagian depan barisan, yang pasti angka 5 karena dia yang pertama masuk (ini aturan antrean: siapa yang datang pertama, dia dilayani pertama). Lalu, satu angka lagi (4) ditambahkan ke

belakang barisan. Terakhir, dua angka lagi dikeluarkan dari depan (2 lalu 7). Jadi, kode ini adalah contoh sederhana bagaimana antrean bekerja, dengan elemen selalu masuk dari belakang dan keluar dari depan.

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

#queue.h

```
#ifndef QUEUE_H
#define QUEUE_H

#include <iostream>
#include <cstdlib>

typedef int infotype;
#define MAX_SIZE 5

typedef struct {
    infotype A[MAX_SIZE];
    int head;
    int tail;
} Queue;

void CreateQueue(Queue &Q);
bool isEmptyQueue(Queue Q);
void printInfo(Queue Q);

bool isFullQueue(Queue Q);
infotype dequeue(Queue &Q);
#endif
```

#queue.cpp

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

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_SIZE == Q.head);
}

void printInfo(Queue Q) {
    std::cout << std::setw(4) << Q.head;
    std::cout << std::setw(4) << Q.tail;
    std::cout << " : ";

    if (isEmptyQueue(Q)) {
        std::cout << "empty queue" << std::endl;
    } else {
        int i = Q.head;
        do {
            std::cout << Q.A[i];
            if (i != Q.tail) {
                std::cout << " ";
            }

            i = (i + 1) % MAX_SIZE;
        } while (i != (Q.tail + 1) % MAX_SIZE);

        std::cout << std::endl;
    }
}

void enqueue(Queue &Q, infotype x) {
    if (isFullQueue(Q)) return;

    if (isEmptyQueue(Q)) {
        Q.head = 0;
        Q.tail = 0;
    } else {
        Q.tail = (Q.tail + 1) % MAX_SIZE;
    }
    Q.A[Q.tail] = x;
}

infotype dequeue(Queue &Q) {
    if (isEmptyQueue(Q)) return 0;

    infotype removed_value = Q.A[Q.head];
```

```
    if (Q.head == Q.tail) {
        CreateQueue(Q);
    } else {
        Q.head = (Q.head + 1) % MAX_SIZE;
    }
    return removed_value;
}
```

#main.cpp

```
// File: Alt3/main_alt3.cpp
#include "queue.h"
#include "queue.cpp"
#include <iomanip>

void run_test_alt3() {
    std::cout << "Hello world!" << std::endl;
    std::cout << "H    T    : Queue Info" << std::endl;
    std::cout << "----------------------------------------" <<
std::endl;

    Queue Q;
    CreateQueue(Q);

    printInfo(Q);

    for (int i = 1; i < MAX_SIZE; i++) {
        enqueue(Q, i * 10);
        printInfo(Q);
    }

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

    enqueue(Q, 50);
    printInfo(Q);
    enqueue(Q, 60);
    printInfo(Q);

    enqueue(Q, 70);
    printInfo(Q);
```

```
}

int main() {
    run_test_alt3();
    return 0;
}
```

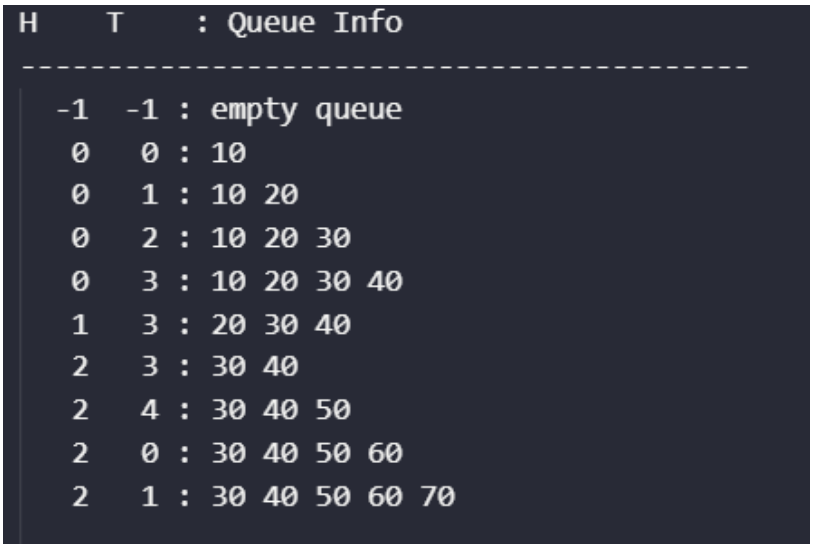
Screenshots Output 1

```
H        T        : Queue Info
----------------------------------------
-1       -1       : empty queue
0        0        : 3
0        1        : 3 4
0        2        : 3 4 5
0        1        : 4 5
0        2        : 4 5 6
0        1        : 5 6
0        0        : 6
-1       -1       : empty queue
-1       -1       : empty queue
```

Screenshots Output 2

```
H    T    : Queue Info
----------------------------------------
    -1    -1 : empty queue
     0     0 : 10
     0     1 : 10 20
     0     2 : 10 20 30
     0     3 : 10 20 30 40
     0     4 : 10 20 30 40 50
     1     4 : 20 30 40 50
     2     4 : 30 40 50
     2     4 : 30 40 50
```

Screenshots Output 3



Deskripsi:

Program ini merupakan implementasi dari Circular Queue, yaitu struktur data antrian yang elemen terakhirnya terhubung kembali ke elemen pertama sehingga dapat memanfaatkan ruang array secara efisien. Program menyediakan beberapa fungsi dasar: membuat queue baru, mengecek apakah queue kosong atau penuh, menambahkan data ke antrian (enqueue), menghapus data dari antrian (dequeue), dan menampilkan seluruh isi antrian beserta posisi head dan tail. Proses enqueue akan menempatkan data di posisi paling belakang secara melingkar, sementara dequeue mengambil data di posisi paling depan. Ketika antrian kosong atau penuh, program menangani kondisi tersebut agar operasi tetap aman. Dengan pendekatan circular, antrian dapat terus digunakan tanpa membuang ruang array meskipun ada elemen yang sudah keluar dari antrian.

D. Kesimpulan

Secara keseluruhan, materi queue memberikan pemahaman penting tentang bagaimana mengelola data yang harus diproses secara berurutan menggunakan prinsip FIFO (First In, First Out). Melalui konsep dasar seperti enqueue, dequeue, pengecekan kondisi kosong dan penuh, serta penerapan circular queue, kita belajar bagaimana struktur data ini mampu mengatur aliran data dengan efisien tanpa membuang ruang penyimpanan. Queue juga menunjukkan bagaimana antrian bekerja dalam sistem nyata, seperti penjadwalan tugas, buffer data, dan proses layanan yang membutuhkan urutan yang teratur. Dengan memahami cara kerja queue, kita dapat membangun program yang lebih terstruktur, rapi, dan sesuai dengan kebutuhan sistem yang harus menangani data secara berurutan dan berkelanjutan.

E. Referensi

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

Stroustrup, B. (2014). *Programming: Principles and Practice Using C++* (2nd ed.). Addison-Wesley Professional.